

Oracle TopLink

Developer's Guide, Volume 1

10g Release 3 (10.1.3)

B13593-02

February 2006

Understanding TopLink

Copyright © 1997, 2006, Oracle. All rights reserved.

Primary Author: Peter Purich, Rick Sapir, Liza Rekadze

The Programs (which include both the software and documentation) contain proprietary information; they are provided under a license agreement containing restrictions on use and disclosure and are also protected by copyright, patent, and other intellectual and industrial property laws. Reverse engineering, disassembly, or decompilation of the Programs, except to the extent required to obtain interoperability with other independently created software or as specified by law, is prohibited.

The information contained in this document is subject to change without notice. If you find any problems in the documentation, please report them to us in writing. This document is not warranted to be error-free. Except as may be expressly permitted in your license agreement for these Programs, no part of these Programs may be reproduced or transmitted in any form or by any means, electronic or mechanical, for any purpose.

If the Programs are delivered to the United States Government or anyone licensing or using the Programs on behalf of the United States Government, the following notice is applicable:

U.S. GOVERNMENT RIGHTS Programs, software, databases, and related documentation and technical data delivered to U.S. Government customers are "commercial computer software" or "commercial technical data" pursuant to the applicable Federal Acquisition Regulation and agency-specific supplemental regulations. As such, use, duplication, disclosure, modification, and adaptation of the Programs, including documentation and technical data, shall be subject to the licensing restrictions set forth in the applicable Oracle license agreement, and, to the extent applicable, the additional rights set forth in FAR 52.227-19, Commercial Computer Software–Restricted Rights (June 1987). Oracle Corporation, 500 Oracle Parkway, Redwood City, CA 94065

The Programs are not intended for use in any nuclear, aviation, mass transit, medical, or other inherently dangerous applications. It shall be the licensee's responsibility to take all appropriate fail-safe, backup, redundancy and other measures to ensure the safe use of such applications if the Programs are used for such purposes, and we disclaim liability for any damages caused by such use of the Programs.

Oracle, JD Edwards and PeopleSoft are registered trademarks of Oracle Corporation and/or its affiliates. Other names may be trademarks of their respective owners.

The Programs may provide links to Web sites and access to content, products, and services from third parties. Oracle is not responsible for the availability of, or any content provided on, third-party Web sites. You bear all risks associated with the use of such content. If you choose to purchase any products or services from a third party, the relationship is directly between you and the third party. Oracle is not responsible for: (a) the quality of third-party products or services; or (b) fulfilling any of the terms of the agreement with the third party, including delivery of products or services and warranty obligations related to purchased products or services. Oracle is not responsible for any loss or damage of any sort that you may incur from dealing with any third party.

Contents

Preface	li

Part I Building a TopLink Application

1 Understanding TopLink

What is TopLink?	1-1
Solving the Object-Persistence Impedance Mismatch	1-2
TopLink Key Features	1-4
TopLink Application Architectures.....	1-4

2 Understanding TopLink Application Development

Developing Your Application With TopLink.....	2-1
Typical Development Stages	2-1
Oracle Development Support.....	2-3
Designing Your Application With TopLink	2-3
Understanding TopLink Usage.....	2-3
Relational Database Usage	2-4
Object-Relational Database Usage.....	2-4
Oracle XML Database (XDB) Usage	2-4
Enterprise Information System (EIS) Usage.....	2-4
XML Usage.....	2-4
Understanding Target Platforms	2-4
Selecting an Architecture With TopLink.....	2-5
Tiers.....	2-5
Three Tier	2-5
Two Tier	2-6
Service Layer.....	2-6
EJB Session Beans.....	2-7
EJB Entity Beans	2-7
Plain Old Java Objects (POJO)	2-8
Data Access	2-8
Data Type	2-8
Multiple Data Sources	2-8
Isolating Data Access.....	2-8
Historical Data Access.....	2-9

Caching	2-9
Cache Type.....	2-9
Refreshing	2-9
Cache Coordination	2-9
Locking	2-10
Optimistic Locking	2-10
Pessimistic Locking.....	2-10
Building and Using the Persistence Layer	2-11
Implementation Options	2-11
Persistent Class Requirements	2-11
Persistence Layer Components	2-12
Mapping Metadata	2-12
Session Metadata.....	2-12
Cache.....	2-12
Queries and Expressions.....	2-13
Transactions	2-13
Using the Persistence Layer	2-13
Deploying the Application.....	2-14
Understanding Deployments	2-14
TopLink in a J2EE Application.....	2-14
Optimizing and Customizing the Application.....	2-15
Troubleshooting the Application.....	2-15
Understanding Object Persistence.....	2-15
Application Object Model.....	2-16
Data Storage Schema	2-16
Primary Keys and Object Identity	2-16
Mappings.....	2-16
Foreign Keys and Object Relationships	2-17
Inheritance.....	2-17
Concurrency.....	2-17
Caching	2-18
Nonintrusive Persistence	2-18
Indirection	2-18
Understanding TopLink Metadata	2-19
Advantages of the TopLink Metadata Architecture	2-19
Creating Project Metadata	2-20
Descriptors and Mappings	2-20
Data Source Login Information	2-20
Creating Session Metadata	2-21
Deploying Metadata	2-21
Understanding the Three-Tier Architecture	2-21
Example Implementations	2-22
Advantages and Disadvantages	2-22
Variation Using Remote Sessions	2-22
Technical Challenges	2-22
Understanding the Two-Tier Architecture	2-23
Example Implementations	2-24

Advantages and Disadvantages	2-24
Technical Challenges	2-24
Understanding the EJB Session Bean Facade Architecture.....	2-24
Example Implementation.....	2-25
Advantages and Disadvantages	2-25
Understanding Session Beans	2-25
Technical Challenges	2-26
Unit of Work Merge.....	2-26
Understanding the EJB Entity Beans with CMP Architecture	2-27
Example Implementation.....	2-28
Advantages and Disadvantages	2-28
Technical Challenges	2-28
External JDBC Pools	2-29
JTA/JTS Integration.....	2-29
Cache Coordination.....	2-29
Maintaining Bidirectional Relationships.....	2-29
Managing Dependent Objects in EJB 1.1	2-30
Managing Dependent Objects in EJB 2.0.....	2-32
Managing Collections of EJBObjects in EJB 1.1	2-32
Managing Collections of EJBObjects in EJB 2.0	2-33
Understanding the EJB Entity Beans With BMP Architecture	2-33
Example Implementations.....	2-35
Advantages and Disadvantages	2-35
Technical Challenges	2-35
External JDBC Pools	2-35
JTA/JTS Integration.....	2-36
Cache Coordination.....	2-36
Understanding the Web Services Architecture	2-36
Example Implementations.....	2-36
Advantages and Disadvantages	2-36
Technical Challenges	2-37

Part II Using TopLink Development Tools

3 Understanding TopLink Development Tools

Development Environment.....	3-2
TopLink Run-Time Environment	3-2

4 Using TopLink Workbench

Understanding TopLink Workbench	4-1
Configuring the TopLink Workbench Environment	4-2
Working With TopLink Workbench	4-3
Using the Menus	4-5
Menu Bar Menus.....	4-5
Context Menus	4-5
Using the Toolbars	4-6

Standard Toolbar	4-6
Context Toolbar	4-7
Using the Navigator.....	4-9
Using the Editor	4-10
Using the Problems Window	4-11
Using the Online Help.....	4-12
Working With TopLink Workbench Preferences	4-12
General Preferences	4-13
Help Preferences	4-14
Mappings Preferences	4-15
Class Preferences	4-16
EJB Preferences	4-16
Database Preferences	4-17
Sessions Configuration Preferences	4-18
New Names Preferences	4-19
Platform Preferences.....	4-20
Working With Databases	4-21
Working With Database Tables in the Navigator Window	4-21
Logging In and Out of a Database.....	4-21
Creating New Tables	4-22
Importing Tables From a Database	4-22
Removing Tables.....	4-24
Renaming Tables.....	4-24
Refreshing Tables From the Database.....	4-24
Working With Database Tables in the Editor Window	4-25
Working With Column Properties	4-25
Setting a Primary Key for Database Tables.....	4-26
Creating Table References	4-27
Creating Field Associations.....	4-28
Generating Data From Database Tables	4-29
Generating SQL Creation Scripts.....	4-29
Generating Classes and Descriptors From Database Tables	4-30
Generating EJB Entities and Descriptors From Database Tables.....	4-31
Generating Tables on the Database.....	4-32
Working With XML Schemas	4-33
Working With XML Schemas in the Navigator.....	4-33
Working With XML Schema Structure	4-34
Importing an XML Schema.....	4-34
Configuring XML Schema Reference	4-36
Using TopLink Workbench	4-36
Using Java	4-37
Configuring XML Schema Namespace.....	4-38
Using TopLink Workbench	4-38
Using Java	4-39
Working With Classes	4-40
Creating Classes	4-40
Using TopLink Workbench	4-40

Configuring Classes	4-41
Configuring Class Information	4-41
Configuring Class Modifiers	4-42
Configuring Class Interfaces	4-43
Adding Attributes	4-44
Configuring Attribute Modifiers	4-44
Configuring Attribute Type Information	4-45
Configuring Attribute Accessing Methods	4-46
Adding Methods	4-47
Configuring Method Modifiers	4-48
Configuring Method Type Information	4-48
Configuring Method Parameters	4-49
Importing and Updating Classes	4-50
Using TopLink Workbench	4-50
Managing Nondesoriptor Classes	4-51
Renaming Packages	4-52
Using TopLink Workbench	4-52
Integrating TopLink Workbench With Apache Ant	4-53
Configuring Ant to Use TopLink Workbench Tasks	4-53
Library Dependencies	4-53
Declaring TopLink Workbench Tasks	4-54
Understanding TopLink Workbench Ant Task API	4-54
Creating TopLink Workbench Ant Tasks	4-55
mappings.validate	4-56
Parameters	4-57
Parameters Specified as Nested Elements	4-57
Examples	4-57
session.validate	4-57
Parameters	4-58
Parameters Specified as Nested Elements	4-58
Examples	4-58
mappings.export	4-58
Parameters	4-58
Parameters Specified as Nested Elements	4-59
Examples	4-59
classpath	4-59
Parameters	4-60
Parameters Specified as Nested Elements	4-60
Examples	4-60
ignoreerror	4-60
Parameters	4-60
Parameters Specified as Nested Elements	4-60
Examples	4-61
ignoreerrorset	4-61
Parameters	4-61
Parameters Specified as Nested Elements	4-61
Examples	4-61

loginspec.....	4-62
Parameters.....	4-62
Parameters Specified as Nested Elements.....	4-62
Examples	4-62

5 Using an Integrated Development Environment

Configuring TopLink for Oracle JDeveloper	5-1
Using TopLink Mappings.....	5-1
Using TopLink Sessions	5-3
Configuring TopLink Workbench With Source Control Management Software.....	5-3
Using a Source Control Management System.....	5-3
Merging Files	5-4
Merging Project Files.....	5-4
Merging Table, Descriptor, and Class Files	5-5
Sharing Project Objects.....	5-6
Managing the ejb-jar.xml File.....	5-6
Working With Locked Files	5-6

6 Using the Schema Manager

Understanding the Schema Manager	6-1
Schema Manager Java and Database Type Conversion	6-3
Sequencing	6-3
Creating a Table Creator	6-4
Using TopLink Workbench During Development.....	6-4
Using the Default Table Generator at Run Time	6-4
Using Java.....	6-4
Creating a TableCreator Class.....	6-5
Creating a TableDefinition Class	6-5
Adding Fields to a TableDefinition.....	6-5
Defining Sybase and Microsoft SQL Server Native Sequencing.....	6-6
Creating Tables With a Table Creator.....	6-6
Automatic Database Table Creation.....	6-6

Part III Deploying a TopLink Application

7 Integrating TopLink With an Application Server

Application Server Support	7-1
Application Server Integration Concepts	7-2
Software Requirements	7-2
XML Parser Platform Configuration.....	7-2
Configuring XML Parser Platform	7-3
Creating an XML Parser Platform	7-3
XML Parser Limitations	7-3
Security Permissions.....	7-4
Persistence Manager Migration	7-4
Clustering	7-4

Oracle Containers for J2EE (OC4J)	7-5
CMP Integration.....	7-5
Migrating OC4J Orion Persistence to OC4J TopLink Persistence.....	7-5
Overview.....	7-6
Using the TopLink Migration Tool from TopLink Workbench.....	7-9
Using the TopLink Migration Tool From the Command Line.....	7-10
Post-Migration Changes	7-12
Troubleshooting Your Migration.....	7-13
JTA Integration.....	7-14
BEA WebLogic Server	7-14
Classpath	7-15
CMP Integration.....	7-15
Migrating BEA WebLogic Persistence to OC4J TopLink Persistence.....	7-16
Overview	7-16
Using the TopLink Migration Tool From TopLink Workbench	7-18
Using the TopLink Migration Tool From the Command Line.....	7-18
JTA Integration.....	7-20
Security Manager	7-20
IBM WebSphere Application Server	7-21
Classpath	7-21
Configuring Classpath for IBM WebSphere Application Server 4.0.....	7-21
Configuring Classpath for IBM WebSphere Application Server 5.0 and Later	7-21
CMP Integration.....	7-22
JTA Integration.....	7-22
Clustering on IBM WebSphere Application Server	7-22
Understanding Security Permissions	7-22
Permissions Required by TopLink Features	7-23
System Properties.....	7-23
Loading project.xml or sessions.xml Files.....	7-23
Cache Coordination.....	7-23
Accessing a Data Source by Port	7-24
Logging With java.util.logging	7-24
J2EE Application Deployment	7-24
Permissions Required When doPrivileged is Disabled	7-24
Disabling doPrivileged Operation.....	7-25
Configuring Miscellaneous EJB Options	7-25
Setter Parameter Type Checking.....	7-25
Unknown Primary Key Class Support.....	7-25
Single-Object Finder Return Type Checking	7-26

8 Creating TopLink Files for Deployment

Understanding TopLink Deployment File Creation	8-1
project.xml File	8-2
XSD File Format	8-2
Non-CMP Applications and Project Metadata.....	8-2
CMP Applications and Project Metadata	8-3
Creating project.xml With TopLink Workbench.....	8-3

Creating project.xml Programatically	8-3
EJB 3.0 and the project.xml File	8-3
sessions.xml File	8-3
XSD File Format	8-4
Non-CMP Applications and Session Metadata	8-4
CMP Applications and Session Metadata	8-4
EJB 3.0 and the sessions.xml File	8-5
ejb-jar.xml File	8-5
EJB 3.0 and the ejb-jar.xml File	8-5
<J2EE-Container>-ejb-jar.xml File	8-5
OC4J and the orion-ejb-jar.xml File	8-5
BEA WebLogic Server and the weblogic-ejb-jar.xml File	8-6
EJB 3.0 and the <J2EE-Container>-jar.xml File	8-6
toplink-ejb-jar.xml File	8-6
OC4J and the toplink-ejb-jar.xml File	8-6
BEA WebLogic Server and the toplink-ejb-jar.xml File	8-7
IBM WebSphere Application Server and the toplink-ejb-jar.xml File	8-8
EJB 3.0 and the toplink-ejb-jar.xml File	8-8
Java Applications	8-8
JavaServer Pages and Servlet Applications	8-8
Session Bean Applications	8-8
CMP Applications	8-9
BMP Applications	8-9
Configuring the orion-ejb-jar.xml File for OC4J	8-9
Configuring persistence-manager Entries	8-9
Configuring pm-properties	8-10
Configuring cache-synchronization Properties	8-11
Configuring default-mapping Properties	8-12
Configuring the weblogic-ejb-jar.xml File for BEA WebLogic Server	8-13
Configuring persistence-descriptor Entries	8-13
Unsupported weblogic-ejb-jar.xml File Tags	8-14

9 Packaging a TopLink Application

Java Applications	9-1
JavaServer Pages and Servlet Applications	9-2
TopLink Domain JAR	9-2
Session Bean Applications	9-3
TopLink Domain JAR	9-3
EJB JAR	9-4
CMP Applications	9-4
EJB JAR	9-4
BMP Applications	9-5
TopLink Domain JAR	9-6
EJB JAR	9-7
Packaging With TopLink Metadata File Resource Paths	9-7

10 Deploying a TopLink Application

Java Applications	10-1
JavaServer Pages and Servlets	10-1
Session Bean Applications	10-1
CMP Applications	10-2
Deploying a CMP Application to OC4J	10-2
Deploying a CMP Application to BEA WebLogic Server	10-2
Troubleshooting ejbc	10-3
Deploying a CMP Application to IBM WebSphere Application Server 4.0	10-3
Starting the Entity Bean	10-3
BMP Applications	10-4
Hot Deployment of EJB	10-4
Hot Deployment in a CMP Application	10-4
Hot Deployment in a non-CMP Application	10-4
Using the WebSphere Deploy Tool	10-5
Using the Deploy Tool on its Own	10-5
Using the Deploy Tool With WebSphere Studio Application Developer	10-5
Troubleshooting	10-6

Part IV Optimizing and Customizing a TopLink Application

11 Optimization

Understanding Optimization	11-1
Sources of Application Performance Problems	11-2
Measuring Performance With the TopLink Profiler	11-2
Configuring the TopLink Performance Profiler	11-3
Accessing the TopLink Profiler Results	11-3
Measuring Performance With the Oracle Dynamic Monitoring System (DMS)	11-4
Configuring the Oracle DMS Profiler	11-6
OC4J Applications	11-7
Non-OC4J Applications	11-7
Accessing Oracle DMS Profiler Data Using JMX	11-7
Accessing Oracle DMS Profiler Data Using the DMS Spy Servlet.....	11-7
General Performance Optimization	11-8
Schema Optimization	11-8
Schema Case 1: Aggregation of Two Tables into One	11-9
Schema Case 2: Splitting One Table Into Many	11-9
Schema Case 3: Collapsed Hierarchy	11-11
Schema Case 4: Choosing One out of Many	11-12
Mapping and Descriptor Optimization	11-13
Session Optimization	11-13
Cache Optimization	11-13
Data Access Optimization	11-14
JDBC Driver Properties Optimization.....	11-14
Data Format Optimization.....	11-15
Batch Writing	11-15

Parameterized SQL (Binding) and Prepared Statement Caching	11-15
Query Optimization	11-16
Parameterized SQL and Prepared Statement Caching.....	11-17
Named Queries.....	11-17
Batch and Join Reading	11-17
Partial Object Queries and Fetch Groups	11-17
JDBC Fetch Size	11-17
Cursored Streams and Scrollable Cursors.....	11-18
Read Optimization Examples.....	11-19
Reading Case 1: Displaying Names in a List	11-20
Reading Case 2: Batch Reading Objects.....	11-22
Reading Case 3: Using Complex Custom SQL Queries	11-24
Reading Case 4: Using View Objects	11-24
Reading Case 5: Inheritance Views	11-26
Write Optimization Examples.....	11-26
Writing Case: Batch Writes.....	11-27
Unit of Work Optimization	11-30
Application Server and Database Optimization	11-30

12 Customization

Overview	12-1
Creating Custom Data Types	12-1
Using Public Source.....	12-2

Part V Troubleshooting a TopLink Application

13 TopLink Exception Reference

Descriptor Exceptions (1 – 200)	13-2
Concurrency Exceptions (2001 – 2006)	13-25
Conversion Exceptions (3001– 3008)	13-26
Database Exceptions (4002 – 4018).....	13-27
Optimistic Lock Exceptions (5001 – 5008)	13-29
Query Exceptions (6001 – 6121)	13-30
Validation Exceptions (7001 – 7147)	13-42
EJB QL Exceptions (8001 – 8010)	13-55
Session Loader Exceptions (9000 - 9010)	13-56
EJB Exception Factory Exceptions (10001 - 10069)	13-58
Communication Exceptions (12000 - 12003).....	13-64
XML Data Store Exceptions (13000 - 13020)	13-64
Deployment Exceptions (14001 - 14033)	13-67
Synchronization Exceptions (15001 - 15025)	13-70
SDK Data Store Exceptions (17001 - 17006)	13-71
EIS Exceptions (17007 – 17025)	13-72
JMS Processing Exceptions (18001 - 18002).....	13-74
SDK Descriptor Exceptions (19001 - 19003)	13-75
Default Mapping Exceptions (20001 - 20008)	13-75

Discovery Exceptions (22001 - 22004).....	13-77
Remote Command Manager Exceptions (22101 - 22111)	13-77
Transaction Exceptions (23001 - 23011).....	13-79
XML Conversion Exceptions (25001 - 25016)	13-80
Migration Utility Exceptions (26001 - 26017).....	13-82
EJB JAR XML Exceptions (72000 – 72022)	13-84

14 TopLink Workbench Error Reference

Miscellaneous Errors (1 – 89).....	14-1
Project Errors (100 – 126)	14-2
Descriptor Errors (200 – 350).....	14-4
Mapping Errors (400 – 483)	14-11
Table Errors (500 – 610).....	14-15
XML Schema Errors (700 – 706)	14-20
Session Errors (800 – 812)	14-21
Common Classpath Problems.....	14-22
Data Source Problems	14-23
Database Connection Problems	14-23

15 Troubleshooting Application Deployment

Generating Deployment JAR Files	15-1
Common J2SE Deployment Exceptions.....	15-1
Classpath Exceptions	15-2
Communication Exceptions.....	15-2
Descriptor Validation Exceptions	15-2
Common BEA WebLogic Server Deployment Exceptions	15-3
Common BEA WebLogic Server 6.1 Exceptions.....	15-5
Development Exceptions	15-6
Deployment and Run Time Exceptions	15-6
Common BEA WebLogic 7.0 Exceptions.....	15-8
Development Exceptions	15-8
Deployment Exceptions	15-9
Common BEA WebLogic 8.1 Exceptions.....	15-10
Development Exceptions	15-10
Deployment Exceptions	15-11
Common IBM WebSphere Application Server Exceptions	15-12
Problems at Run Time	15-14
Common TopLink for IBM WebSphere Deploy Tool Exceptions.....	15-14

Part VI TopLink Tutorial

16 Planning the Tutorial Project

Overview	16-1
Before you Begin	16-1
Understanding the Tutorial Project.....	16-2
Source Files.....	16-2

Object Model Classes.....	16-2
Application Classes	16-3
Data Model.....	16-4
Creating and Populating Database Tables	16-7

17 Building the Tutorial Project

Step 1: Creating a Project	17-1
TopLink Workbench Project.....	17-1
Adding Java Classes	17-3
Step 2: Adding a Data Source.....	17-5
Creating a Database Development Login.....	17-5
Importing Database Tables	17-6
Project-Level Sequencing	17-8
Step 3: Associating Descriptors to Tables	17-9
Associating With Multiple Tables	17-10
Descriptor-Level Sequencing.....	17-11
Step 4: Mapping Attributes	17-12
Direct-to-Field Mappings.....	17-12
Using Automap	17-13
Additional Direct-to-Field Mappings	17-14
One-to-One Mappings.....	17-14
Additional One-to-One Mappings	17-16
One-to-Many Mappings.....	17-16
Additional One-to-Many Mappings	17-18
Object Type Mappings	17-18
Aggregate Object Mappings.....	17-19
Aggregate Descriptor	17-20
Aggregate Mapping.....	17-20
Direct Collection Mappings.....	17-21
Many-to-Many Mappings.....	17-21
Transformation Mappings.....	17-23
Step 5: Using Advanced Descriptor Properties	17-25
Inheritance.....	17-25
Root Parent Class	17-25
Child Class	17-26
Indicator Values	17-27
Locking Policy	17-28
Named Queries.....	17-29
Step 6: Verifying the Project	17-32

18 Deploying the Tutorial Project

Step 1: Generating Deployment Information.....	18-1
Creating the project.xml File	18-1
Step 2: Creating Sessions Configuration	18-3
Creating the sessions.xml File	18-3
Creating a Server Session.....	18-3
Associating the Project With the Session.....	18-4

Updating the data-sources.xml File.....	18-5
Creating a Database Deployment Login.....	18-6
Configuring Logging Options.....	18-7
Step 3: Packaging for Deployment.....	18-8
Domain JAR File.....	18-8
WAR File	18-8
EAR File.....	18-8
Step 4: Deploying to an Application Server.....	18-9
Step 5: Running the Tutorial Application	18-9

Part VII Mapping and Configuration Overview

19 Understanding TopLink Mapping and Configuration Concepts

Mapping and Configuration Concepts	19-1
Projects.....	19-1
Descriptors	19-2
Mappings.....	19-2

Part VIII Projects

20 Understanding Projects

TopLink Project Types	20-1
Project Concepts	20-2
Project Architecture	20-2
Relational and Nonrelational Projects.....	20-2
Persistent and Nonpersistent Projects.....	20-2
Projects and Login.....	20-3
Non-CMP Session Role: Session Login.....	20-3
CMP Deployment Role: Deployment Login	20-3
Development Role: Development Login	20-4
Projects and Platforms.....	20-4
Projects and Sequencing.....	20-4
Configuring how to Obtain Sequence Values.....	20-5
Configuring Where to Write Sequence Values.....	20-5
XML Namespaces.....	20-5
Relational Projects.....	20-6
Building Relational Projects for a Relational Database	20-6
Building Relational Projects for an Object-Relational Database	20-6
EIS Projects	20-7
Building EIS Projects with XML Records	20-9
Building EIS Projects With Indexed or Mapped Records	20-9
XML Projects	20-9
TopLink Support for Java Architecture for XML Binding (JAXB)	20-10
Understanding JAXB-Specific Generated Files	20-10
Understanding TopLink-Specific Generated Files.....	20-11
Using TopLink JAXB Compiler Generated Files at Run Time	20-13

JAXB Validation	20-14
Understanding the Project API	20-14
Project Inheritance Hierarchy.....	20-14
Understanding Sequencing in Relational Projects	20-14
Sequencing Configuration Options	20-15
Sequencing Types.....	20-16
Table Sequencing	20-16
Unary Table Sequencing	20-17
Query Sequencing.....	20-18
Default Sequencing.....	20-18
Native Sequencing With an Oracle Database Platform.....	20-18
Native Sequencing With a Non-Oracle Database Platform.....	20-19
Sequencing and Preallocation Size	20-20
Sequencing With CMP Entity Beans	20-21
Understanding XML Namespaces	20-22
TopLink Workbench Namespace Resolution	20-22
Element and Attribute Form Options	20-22
Element Form Default Qualified and Attribute Form Default Unqualified	20-23
Element and Attribute Form Default Unqualified.....	20-24
Element and Attribute Form Default Qualified	20-25
TopLink Runtime Namespace Resolution.....	20-26

21 Creating a Project

Project Creation Overview	21-1
Using TopLink Workbench	21-2
Creating New TopLink Workbench Projects.....	21-2
Using Java.....	21-3
Creating a Project for an Existing Object and Data Model	21-5
Using TopLink Workbench	21-5
Creating a Project From an Existing Object Model	21-5
Using TopLink Workbench	21-5
Creating a Project From an Existing Data Model	21-5
Using TopLink Workbench	21-6
Creating an XML Project From an XML Schema	21-6
Using TopLink Workbench	21-6
Using the Command Line	21-8
Creating a Project by Migrating an EAR to OC4J	21-9
Creating a Project From an OC4J EJB 2.0 CMP EAR at Deployment Time	21-10
Working With Projects	21-10
Opening Existing Projects	21-10
Saving Projects.....	21-11
Saving Projects With a New Name or Location	21-12
Generating the Project Status Report	21-12
Exporting Project Information	21-13
Exporting Deployment XML Information.....	21-14
Exporting Model Java Source.....	21-14
Exporting Project Java Source	21-14

Exporting Table Creator Files.....	21-15
Working With the ejb-jar.xml File	21-15
Writing to the ejb-jar.xml File.....	21-16
Reading From the ejb-jar.xml File.....	21-16

22 Configuring a Project

Configuring Common Project Options.....	22-1
Configuring Project Save Location	22-2
Using TopLink Workbench	22-2
Configuring Project Classpath.....	22-3
Using TopLink Workbench	22-3
Configuring Mapped Field Access at the Project Level.....	22-4
Using TopLink Workbench	22-4
Configuring Persistence Type	22-5
Using TopLink Workbench	22-6
Configuring Default Descriptor Advanced Properties.....	22-7
Using TopLink Workbench	22-8
Configuring Existence Checking at the Project Level.....	22-8
Using TopLink Workbench	22-9
Configuring Project Deployment XML Options.....	22-10
Using TopLink Workbench	22-10
Configuring Model Java Source Code Options.....	22-11
Using TopLink Workbench	22-11
Configuring Deprecated Direct Mappings	22-12
Using TopLink Workbench	22-13
Configuring Cache Type and Size at the Project Level	22-13
Using TopLink Workbench	22-14
Configuring Cache Isolation at the Project Level	22-16
Using TopLink Workbench	22-16
Configuring Cache Coordination Change Propagation at the Project Level.....	22-17
Using TopLink Workbench	22-18
Configuring Cache Expiration at the Project Level	22-19
Using TopLink Workbench	22-20
Configuring Project Comments.....	22-21
Using TopLink Workbench	22-21

23 Configuring a Relational Project

Relational Project Configuration Overview	23-1
Configuring Relational Database Platform at the Project Level.....	23-2
Using TopLink Workbench	23-2
Configuring Sequencing at the Project Level.....	23-3
Using TopLink Workbench	23-3
Using Java.....	23-4
Configuring Login Information	23-5
Using TopLink Workbench	23-5
Configuring Development and Deployment Logins	23-6

Using TopLink Workbench	23-6
Logging in to the Database	23-7
Configuring Named Query Parameterized SQL and Statement Caching at the Project Level.....	23-7
Using TopLink Workbench	23-8
Configuring Table Generation Options	23-9
Using TopLink Workbench	23-9
Configuring Table Creator Java Source Options	23-10
Using TopLink Workbench	23-10
Configuring Project Java Source Code Options.....	23-11
Using TopLink Workbench	23-11

24 Configuring an EIS Project

EIS Project Configuration Overview.....	24-1
Configuring EIS Data Source Platform at the Project Level.....	24-2
Using TopLink Workbench	24-2
Configuring EIS Connection Specification Options at the Project Level.....	24-2
Using TopLink Workbench	24-3

25 Configuring an XML Project

XML Project Configuration Overview	25-1

Part IX Descriptors

26 Understanding Descriptors

Descriptor Types	26-1
Descriptor Concepts.....	26-2
Descriptor Architecture.....	26-2
Descriptors and Inheritance.....	26-3
Descriptors and EJB	26-3
Nondeferred Changes	26-3
Creating a New Entity Bean and ejbCreate / ejbPostCreate Methods	26-4
Inheritance	26-4
Fetch Groups	26-4
Amendment and After-Load Methods	26-5
Descriptors and Aggregation	26-5
Aggregate and Composite Descriptors in Relational Projects	26-5
Root and Composite Descriptors in EIS Projects.....	26-8
Composite Descriptors in XML Projects.....	26-8
Descriptor Event Manager.....	26-8
Descriptor Query Manager.....	26-8
Descriptors and Sequencing	26-9
Descriptors and Locking	26-9
Default Root Element.....	26-9
Relational Descriptors.....	26-11
Object-Relational Descriptors	26-11

EIS Descriptors	26-11
XML Descriptors.....	26-12
Understanding Descriptors and Inheritance	26-12
Specifying a Class Indicator.....	26-13
Using Class Indicator Fields.....	26-13
Using Class Extraction Methods.....	26-14
Inheritance and Primary Keys (Relational and EIS Only).....	26-15
Single and Multi-Table Inheritance (Relational Only).....	26-15
Single Table Inheritance.....	26-15
Multitable Inheritance.....	26-16
Aggregate and Composite Descriptors and Inheritance	26-17
Inheritance and EJB.....	26-17
Understanding Descriptors and Locking	26-17
Optimistic Version Locking Policies	26-17
Optimistic Version Locking Policies and Cascading	26-18
Optimistic Locking and Rollbacks.....	26-20
Optimistic Field Locking Policies	26-20
Pessimistic Locking Policy.....	26-21
Locking in a Three-Tier Application	26-21
Optimistic Locking in a Three-Tier Application	26-21
Pessimistic Locking in a Three-Tier Application	26-22
Understanding the Descriptor API	26-22
Descriptor Inheritance Hierarchy	26-23

27 Creating a Descriptor

Descriptor Creation Overview	27-1
Creating a Relational Descriptor.....	27-1
Using TopLink Workbench	27-2
Relational Class Descriptors.....	27-2
Relational Aggregate Descriptors.....	27-2
Relational Interface Descriptors.....	27-2
Using Java.....	27-2
Creating an Object-Relational Descriptor.....	27-3
Using Java.....	27-3
Creating an EIS Descriptor.....	27-4
Using TopLink Workbench	27-4
EIS Root Descriptors.....	27-5
EIS Composite Descriptors	27-5
Using Java.....	27-5
Creating an XML Descriptor	27-5
Using TopLink Workbench	27-5
Using Java.....	27-5
Validating Descriptors	27-6
Generating Java Code for Descriptors	27-6

28 Configuring a Descriptor

Configuring Common Descriptor Options	28-1
Configuring Primary Keys	28-2
Using TopLink Workbench	28-3
Using Java.....	28-4
Relational Projects.....	28-4
EIS Projects.....	28-4
Configuring Read-Only Descriptors	28-4
Using Read-Only Entity Beans.....	28-5
Using TopLink Workbench	28-5
Using Java.....	28-6
Configuring Unit of Work Conforming at the Descriptor Level	28-6
Using TopLink Workbench	28-7
Using Java.....	28-7
Configuring Descriptor Alias	28-7
Using TopLink Workbench	28-8
Using Java.....	28-9
Configuring Descriptor Comments	28-9
Using TopLink Workbench	28-9
Configuring Named Queries at the Descriptor Level	28-10
Using TopLink Workbench	28-10
Adding Named Queries.....	28-12
Configuring Named Query Type and Parameters	28-13
Configuring Named Query Selection Criteria.....	28-14
Configuring Read All Query Order	28-15
Configuring Named Query Optimization.....	28-16
Configuring Named Query Attributes	28-17
Configuring Named Query Group/Order Options	28-19
Creating an EIS Interaction for a Named Query	28-20
Configuring Named Query Options.....	28-22
Configuring Named Query Advanced Options.....	28-24
Using Java.....	28-25
Configuring Query Timeout at the Descriptor Level.....	28-26
Using TopLink Workbench	28-26
Using Java.....	28-27
Configuring Cache Refreshing.....	28-27
Using TopLink Workbench	28-28
Using Java.....	28-29
Configuring Query Keys.....	28-29
Using TopLink Workbench	28-31
Using Java.....	28-31
Configuring Interface Query Keys	28-33
Using TopLink Workbench	28-34
Using Java.....	28-34
Configuring Cache Type and Size at the Descriptor Level.....	28-35
Using TopLink Workbench	28-35
Using Java.....	28-37

Configuring Cache Isolation at the Descriptor Level	28-37
Using TopLink Workbench	28-37
Using Java.....	28-38
Configuring Cache Coordination Change Propagation at the Descriptor Level	28-38
Using TopLink Workbench	28-39
Using Java.....	28-40
Configuring Cache Expiration at the Descriptor Level.....	28-40
Using TopLink Workbench	28-41
Using Java.....	28-42
Configuring Cache Existence Checking at the Descriptor Level	28-42
Using TopLink Workbench	28-42
Using Java.....	28-43
Configuring a Descriptor With EJB Information	28-44
Using TopLink Workbench	28-44
Using Java.....	28-46
Configuring CMP Information	28-46
Configuring BMP Information.....	28-47
Configuring Reading Subclasses on Queries	28-47
Using TopLink Workbench	28-48
Using Java.....	28-48
Configuring Inheritance for a Child (Branch or Leaf) Class Descriptor	28-49
Using TopLink Workbench	28-49
Using Java.....	28-50
Configuring Inheritance for a Parent (Root) Descriptor.....	28-50
Using TopLink Workbench	28-51
Using Java.....	28-52
Configuring Inheritance Expressions for a Parent (Root) Class Descriptor	28-53
Using Java.....	28-55
Configuring Inherited Attribute Mapping in a Subclass.....	28-56
Using TopLink Workbench	28-56
Using Java.....	28-57
Configuring a Domain Object Method as an Event Handler.....	28-57
Using TopLink Workbench	28-58
Using Java.....	28-60
Configuring a Descriptor Event Listener as an Event Handler.....	28-60
Using Java.....	28-62
Configuring Locking Policy	28-62
Using TopLink Workbench	28-63
Using Java.....	28-65
Configuring an Optimistic Locking Policy	28-65
Configuring Optimistic Locking Policy Cascading	28-65
Configuring a Pessimistic Locking Policy	28-65
Configuring Returning Policy	28-65
Using TopLink Workbench	28-66
Using Java.....	28-67
Configuring Instantiation Policy	28-68
Using TopLink Workbench	28-68

Using Java.....	28-69
Configuring Copy Policy	28-69
Using TopLink Workbench	28-70
Using Java.....	28-70
Configuring Change Policy	28-70
Using Java.....	28-71
Configuring Deferred Change Detection Policy	28-71
Configuring Object Change Tracking Policy	28-71
Configuring Attribute Change Tracking Policy	28-72
Configuring a History Policy	28-73
Using Java.....	28-75
Configuring Write Responsibility	28-75
Configuring Wrapper Policy	28-75
Using Java.....	28-76
Configuring Fetch Groups.....	28-77
Using Java.....	28-78
Configuring Amendment Methods	28-78
Using TopLink Workbench	28-79

29 Configuring a Relational Descriptor

Relational Descriptor Configuration Overview.....	29-1
Configuring Associated Tables	29-2
Using TopLink Workbench	29-2
Using Java.....	29-3
Configuring Sequencing at the Descriptor Level	29-3
Using TopLink Workbench	29-4
Using Java.....	29-5
Configuring a Sequence by Name	29-5
Configuring the Same Sequence for Multiple Descriptors	29-5
Configuring the Platform Default Sequence	29-6
Configuring Custom SQL Queries for Basic Persistence Operations.....	29-6
Using TopLink Workbench	29-7
Using Java.....	29-8
Configuring Interface Alias	29-10
Using TopLink Workbench	29-11
Using Java.....	29-11
Configuring a Relational Descriptor as a Class or Aggregate Type.....	29-11
Using TopLink Workbench	29-12
Using Java.....	29-12
Configuring Multitable Information	29-13
Using TopLink Workbench	29-13
Using Java.....	29-14

30 Configuring an Object-Relational Descriptor

Object-Relational Descriptor Configuration Overview	30-1
Configuring Field Ordering	30-2
Using Java.....	30-2

31 Configuring an EIS Descriptor

EIS Descriptor Configuration Overview	31-1
Configuring Schema Context for an EIS Descriptor	31-2
Using TopLink Workbench	31-2
Choosing a Schema Context	31-3
Using Java	31-3
Configuring Default Root Element	31-3
Using TopLink Workbench	31-4
Choosing a Root Element	31-4
Using Java	31-5
Configuring Record Format	31-5
Using Java	31-6
Configuring Custom EIS Interactions for Basic Persistence Operations	31-6
Using TopLink Workbench	31-6
Using Java	31-8
Configuring an EIS Descriptor as a Root or Composite Type	31-8
Using TopLink Workbench	31-9
Using Java	31-9

32 Configuring an XML Descriptor

XML Descriptor Configuration Overview	32-1
Configuring Schema Context for an XML Descriptor	32-1
Using TopLink Workbench	32-2
Choosing a Schema Context	32-2
Using Java	32-3
Configuring for Complex Type of anyType	32-3
Using TopLink Workbench	32-4
Configuring Default Root Element	32-5
Using TopLink Workbench	32-5
Choosing a Root Element	32-5
Configuring Document Preservation	32-6
Using TopLink Workbench	32-6
Using Java	32-6

Part X Mappings

33 Understanding Mappings

Mapping Types	33-1
Mapping Concepts	33-2
Mapping Architecture	33-2
Example Mapping	33-3
Automatic Mappings	33-4
Automapping With TopLink Workbench at Development Time	33-4
Default Mapping in EJB 2.0 or 3.0 CMP Projects Using OC4J at Run Time	33-4
JAXB Project Generation at Development Time	33-5
Indirection	33-5

Value Holder Indirection.....	33-6
Transparent Indirect Container Indirection.....	33-8
Proxy Indirection	33-8
Indirection and EJB.....	33-9
Indirection, Serialization, and Detachment.....	33-9
Method Accessors and Attribute Accessors.....	33-10
Mapping Converters and Transformers	33-10
Serialized Object Converter.....	33-10
Type Conversion Converter	33-11
Object Type Converter	33-12
Simple Type Translator.....	33-12
Transformation Mappings.....	33-14
Mappings and XPath	33-15
XPath by Position.....	33-15
XPath by Path and Name.....	33-15
XPath by Name.....	33-16
Self XPath	33-16
Mappings and xsd:list and xsd:union Types	33-17
Mapping an xsd:union Type	33-17
Mapping an xsd:list Type	33-17
Mapping a List of Unions	33-18
Mapping a Union of Lists	33-19
Mapping a Union of Unions.....	33-19
Mappings and the jaxb:class Customization.....	33-20
all, choice, or sequence Structure.....	33-20
group Structure	33-21
sequence or choice Structure Containing a group	33-21
group Structure Containing a sequence or choice	33-22
group Structure Containing a group.....	33-23
Limitations of jaxb:class Customization Support.....	33-23
Mappings and JAXB Typesafe Enumerations.....	33-24
Understanding the Mapping API.....	33-25
Relational Mappings	33-25
Object-Relational Mappings.....	33-26
XML Mappings	33-27
EIS Mappings.....	33-27

34 Creating a Mapping

Mapping Creation Overview	34-1
Creating Mappings Manually During Development	34-1
Using TopLink Workbench	34-1
Creating Mappings Automatically During Development	34-2
Using TopLink Workbench	34-2
Creating Mappings Automatically During Deployment	34-2
Creating Mappings to Oracle LOB Database Objects.....	34-2
Using the Oracle JDBC OCI Driver or Server Driver.....	34-3
Using the Oracle JDBC Thin Driver	34-5

Removing Mappings	34-6
Using TopLink Workbench	34-6

35 Configuring a Mapping

Configuring Common Mapping Options.....	35-1
Configuring Read-Only Mappings.....	35-2
Using TopLink Workbench	35-3
Using Java.....	35-3
Configuring Indirection.....	35-3
Using TopLink Workbench	35-4
Using Java.....	35-5
Configuring ValueHolder Indirection	35-6
Configuring ValueHolder Indirection With Method Accessing.....	35-7
Configuring ValueHolder Indirection With EJB 3.0 on OC4J	35-8
Configuring IndirectContainer Indirection	35-8
Configuring Proxy Indirection.....	35-9
Configuring XPath	35-10
Using TopLink Workbench	35-11
Choosing the XPath	35-12
Configuring a Default Null Value at the Mapping Level.....	35-12
Using TopLink Workbench	35-13
Using Java.....	35-13
Configuring Method Accessing.....	35-14
Using TopLink Workbench	35-14
Using Java.....	35-15
Configuring Private or Independent Relationships.....	35-16
Using TopLink Workbench	35-17
Using Java.....	35-17
Configuring Mapping Comments.....	35-18
Using TopLink Workbench	35-18
Configuring a Serialized Object Converter	35-18
Using TopLink Workbench	35-19
Using Java.....	35-19
Configuring a Type Conversion Converter.....	35-20
Using TopLink Workbench	35-20
Using Java.....	35-21
Configuring an Object Type Converter.....	35-22
Using TopLink Workbench	35-22
Using Java.....	35-23
Configuring a Simple Type Translator.....	35-23
Using TopLink Workbench	35-24
Using Java.....	35-24
Configuring a JAXB Typesafe Enumeration Converter.....	35-25
Using Java.....	35-26
Configuring Container Policy.....	35-26
Using TopLink Workbench	35-27
Using Java.....	35-28

Configuring Attribute Transformer	35-29
Using TopLink Workbench	35-30
Using Java.....	35-31
Configuring Field Transformer Associations	35-31
Using TopLink Workbench	35-32
Specifying Field-to-Transformer Associations	35-32
Using Java.....	35-33
Configuring Mutable Mappings	35-33
Using TopLink Workbench	35-34
Using Java.....	35-34
Configuring Bidirectional Relationship	35-34
Using TopLink Workbench	35-35
Configuring the Use of a Single Node	35-36
Using TopLink Workbench	35-36
Using Java.....	35-37

Part XI Relational Mappings

36 Understanding Relational Mappings

Relational Mapping Types	36-1
Relational Mapping Concepts	36-2
Directionality	36-2
Converters and Transformers	36-3
Using a Direct Mapping	36-3
Using a Converter Mapping	36-3
Using a Transformation Mapping	36-3
Relational Mappings and EJB.....	36-3
Direct-to-Field Mapping	36-4
Direct to XMLType Mapping.....	36-4
One-to-One Mapping	36-5
One-to-One Mappings and EJB.....	36-6
Variable One-to-One Mapping	36-6
One-to-Many Mapping	36-7
One-to-Many Mappings and EJB.....	36-8
Many-to-Many Mapping	36-8
Many-to-Many Mappings and EJB.....	36-9
Aggregate Collection Mapping	36-10
Aggregate Collection Mappings and Inheritance	36-10
Aggregate Collection Mappings and EJB	36-11
Implementing Aggregate Collection Mappings	36-11
Direct Collection Mapping.....	36-11
Direct Map Mapping	36-12
Aggregate Object Mapping	36-12
Aggregate Object Mappings with a Single Source Object.....	36-13
Aggregate Object Mappings With Multiple Source Objects	36-14
Implementing an Aggregate Object Relationship Mapping	36-14
Transformation Mapping	36-15

37	Configuring a Relational Mapping	
	Configuring Common Relational Mapping Options.....	37-1
	Configuring a Database Field.....	37-2
	Using TopLink Workbench	37-4
	Configuring Reference Descriptor.....	37-5
	Using TopLink Workbench	37-5
	Configuring Batch Reading.....	37-6
	Using TopLink Workbench	37-7
	Using Java.....	37-7
	Configuring Query Key Order	37-8
	Using TopLink Workbench	37-8
	Configuring Table and Field References (Foreign and Target Foreign Keys)	37-8
	Using TopLink Workbench	37-9
38	Configuring a Relational Direct-to-Field Mapping	
	Relational Direct-to-Field Mapping Configuration Overview	38-1
39	Configuring a Relational Direct-to-XMLType Mapping	
	Relational Direct-to-XMLType Mapping Overview	39-1
	Configuring Read Whole Document.....	39-1
	Using TopLink Workbench	39-1
40	Configuring a Relational One-to-One Mapping	
	Relational One-to-One Mapping Configuration Overview	40-1
	Configuring Joining at the Mapping Level.....	40-1
	Using TopLink Workbench	40-2
41	Configuring a Relational Variable One-to-One Mapping	
	Relational Variable One-to-One Mapping Configuration Overview.....	41-1
	Configuring Class Indicator.....	41-1
	Using TopLink Workbench	41-2
	Configuring Unique Primary Key	41-3
	Understanding Unique Primary Key	41-3
	Using TopLink Workbench	41-3
	Using Java.....	41-4
	Configuring Query Key Association.....	41-4
	Using TopLink Workbench	41-4
42	Configuring a Relational One-to-Many Mapping	
	Relational One-to-Many Mapping Configuration Overview	42-1
43	Configuring a Relational Many-to-Many Mapping	
	Relational Many-to-Many Mapping Configuration Overview.....	43-1
	Configuring a Relation Table	43-1

Using TopLink Workbench	43-2
44 Configuring a Relational Aggregate Collection Mapping	
Relational Aggregate Collection Mapping Configuration Overview.....	44-1
45 Configuring a Relational Direct Collection Mapping	
Relational Direct Collection Mapping Configuration Overview	45-1
Configuring Target Table	45-1
Using TopLink Workbench	45-2
Configuring Direct Value Field.....	45-2
Using TopLink Workbench	45-2
46 Configuring a Relational Direct Map Mapping	
Relational Direct Map Mapping Configuration Overview	46-1
Configuring Direct Value Field.....	46-1
Using TopLink Workbench	46-1
Configuring Direct Key Field	46-2
Using TopLink Workbench	46-2
Configuring Key Converters.....	46-3
Using TopLink Workbench	46-3
Configuring Value Converters	46-4
Using TopLink Workbench	46-4
47 Configuring a Relational Aggregate Object Mapping	
Relational Aggregate Object Mapping Configuration Overview	47-1
Configuring Aggregate Fields	47-1
Using TopLink Workbench	47-2
Configuring Allowing Null Values.....	47-2
Using TopLink Workbench	47-2
48 Configuring a Relational Transformation Mapping	
Relational Transformation Mapping Configuration Overview	48-1
Part XII Object-Relational Mappings	
49 Understanding Object-Relational Mappings	
Object-Relational Mapping Types	49-1
Object-Relational Structure Mapping.....	49-2
Object-Relational Reference Mapping	49-2
Object-Relational Array Mapping	49-2
Object-Relational Object Array Mapping.....	49-2
Object-Relational Nested Table Mapping	49-3

50	Configuring an Object-Relational Mapping	
	Configuring Common Object-Relational Mapping Options	50-1
	Configuring Reference Class	50-2
	Using Java.....	50-2
	Configuring Attribute Name	50-2
	Using Java.....	50-3
	Configuring Field Name	50-3
	Using Java.....	50-3
	Configuring Structure Name	50-4
	Using Java.....	50-4
51	Configuring an Object-Relational Structure Mapping	
	Object-Relational Structure Mapping Configuration Overview.....	51-1
52	Configuring an Object-Relational Reference Mapping	
	Object-Relational Reference Mapping Configuration Overview	52-1
53	Configuring an Object-Relational Array Mapping	
	Object-Relational Array Mapping Configuration Overview	53-1
54	Configuring an Object-Relational Object Array Mapping	
	Object-Relational Object Array Mapping Configuration Overview.....	54-1
55	Configuring an Object-Relational Nested Table Mapping	
	Object-Relational Nested Table Mapping Configuration Overview	55-1
Part XIII EIS Mappings		
56	Understanding EIS Mappings	
	EIS Mapping Types	56-1
	EIS Mapping Concepts.....	56-2
	EIS Record Type	56-2
	Indexed Records.....	56-2
	Mapped Records	56-3
	XML Records	56-3
	XPath Support.....	56-3
	xsd:list and xsd:union Support	56-3
	jaxb:class Support.....	56-3
	Typesafe Enumeration Support	56-3
	Composite and Reference EIS Mappings	56-4
	Composite EIS Mappings	56-4
	Reference EIS Mappings	56-4
	EIS Mapping Architecture	56-5
	EIS Direct Mapping	56-5

EIS Composite Direct Collection Mapping.....	56-6
EIS Composite Object Mapping.....	56-7
EIS Composite Collection Mapping.....	56-7
EIS One-to-One Mapping.....	56-8
EIS One-to-One Mappings With Key on Source.....	56-9
EIS One-to-One Mappings With Key on Target.....	56-10
EIS One-to-Many Mapping.....	56-12
EIS One-to-Many Mappings With Key on Source.....	56-13
EIS One-to-Many Mappings With Key on Target.....	56-15
EIS Transformation Mapping.....	56-17
 57 Configuring an EIS Mapping	
Configuring Common EIS Mapping Options	57-1
Configuring Reference Descriptors.....	57-2
Using TopLink Workbench	57-2
Configuring Selection Interaction	57-3
Using TopLink Workbench	57-4
 58 Configuring an EIS Direct Mapping	
EIS Direct Mapping Configuration Overview	58-1
 59 Configuring an EIS Composite Direct Collection Mapping	
EIS Composite Direct Collection Mapping Configuration Overview.....	59-1
 60 Configuring an EIS Composite Object Mapping	
EIS Composite Object Mapping Configuration Overview.....	60-1
 61 Configuring an EIS Composite Collection Mapping	
EIS Composite Collection Mapping Configuration Overview	61-1
 62 Configuring an EIS One-to-One Mapping	
EIS One-to-One Mapping Configuration Overview.....	62-1
Configuring Foreign Key Pairs.....	62-1
Using TopLink Workbench	62-2
 63 Configuring an EIS One-to-Many Mapping	
EIS One-to-Many Mapping Configuration Overview	63-1
Configuring Foreign Key Pairs.....	63-1
Using TopLink Workbench	63-2
Configuring Delete All Interactions.....	63-3
Using TopLink Workbench	63-4
 64 Configuring an EIS Transformation Mapping	
EIS Transformation Mapping Configuration Overview.....	64-1

Part XIV XML Mappings

65 Understanding XML Mappings

XML Mapping Types	65-1
XML Mapping Concepts	65-2
Mapping to Simple and Complex Types.....	65-2
Mapping Order.....	65-3
XPath Support.....	65-3
xsd:list and xsd:union Support	65-3
xs:any and xs:anyType Support.....	65-4
jaxb:class Support.....	65-4
Typesafe Enumeration Support	65-4
Mapping Extensions	65-4
XML Direct Mapping	65-5
Mapping to a Text Node	65-5
Mapping to a Simple Text Node.....	65-5
Mapping to a Text Node in a Simple Sequence.....	65-6
Mapping to a Text Node in a Subelement.....	65-6
Mapping to a Text Node by Position	65-7
Mapping to an Attribute	65-8
Mapping to a Specified Schema Type.....	65-9
Mapping to a List Field With an XML Direct Mapping	65-10
Mapping to a Union Field With an XML Direct Mapping.....	65-10
Mapping to a Union of Lists With an XML Direct Mapping.....	65-12
Mapping to a Union of Unions With an XML Direct Mapping	65-12
Mapping With a Simple Type Translator	65-13
XML Composite Direct Collection Mapping	65-14
Mapping to Multiple Text Nodes	65-15
Mapping to a Simple Sequence.....	65-15
Mapping to a Sequence in a Subelement.....	65-15
Mapping to Multiple Attributes.....	65-16
Mapping to a Single Text Node With an XML Composite Direct Collection Mapping	65-17
Mapping to a Single Attribute With an XML Composite Direct Collection Mapping	65-18
Mapping to a List of Unions With an XML Composite Direct Collection Mapping.....	65-18
Mapping to a Union of Lists With an XML Composite Direct Collection Mapping.....	65-19
Specifying the Content Type of a Collection With an XML Composite Direct Collection Mapping	65-20
XML Composite Object Mapping	65-21
Mapping Into the Parent Record.....	65-21
Mapping to an Element.....	65-22
Mapping to Different Elements by Element Name.....	65-23
Mapping to Different Elements by Element Position	65-24
XML Composite Collection Mapping	65-25
XML Any Object Mapping	65-27
XML Any Collection Mapping	65-29
XML Transformation Mapping	65-31

66	Configuring an XML Mapping	
	Configuring Common XML Mapping Options	66-1
	Configuring Reference Descriptor.....	66-2
	Using TopLink Workbench	66-2
	Configuring Maps to Wildcard.....	66-3
	Using TopLink Workbench	66-3
67	Configuring an XML Direct Mapping	
	XML Direct Mapping Configuration Overview.....	67-1
68	Configuring an XML Composite Direct Collection Mapping	
	XML Composite Direct Collection Mapping Configuration Overview.....	68-1
69	Configuring an XML Composite Object Mapping	
	XML Composite Object Mapping Configuration Overview.....	69-1
70	Configuring an XML Composite Collection Mapping	
	XML Composite Collection Mapping Configuration Overview	70-1
71	Configuring an XML Any Object Mapping	
	XML Any Object Mapping Configuration Overview.....	71-1
72	Configuring an XML Any Collection Mapping	
	XML Any Collection Mapping Configuration Overview	72-1
73	Configuring an XML Transformation Mapping	
	XML Transformation Mapping Configuration Overview	73-1
Part XV Using TopLink Overview		
74	Understanding the Persistence Layer	
	Overview of the Persistence Layer.....	74-1
	Sessions	74-1
	Data Access	74-1
	Cache.....	74-2
	Queries and Expressions.....	74-2
	Transactions.....	74-3
Part XVI TopLink Sessions		
75	Understanding TopLink Sessions	
	Session Types.....	75-1
	Session Concepts	75-2

Session Architecture	75-2
Object Cache	75-3
Connection Pools	75-3
Query Mechanism.....	75-4
Java Object Builder	75-4
Session Configuration and the sessions.xml File.....	75-4
Session Customization	75-4
Acquiring a Session at Run Time With the Session Manager.....	75-5
Managing Session Events With the Session Event Manager	75-5
Session Event Manager Events	75-6
Session Event Listeners	75-7
Logging.....	75-7
Log Types	75-8
Log Output.....	75-10
Log Level.....	75-10
Logging SQL.....	75-10
Logging Chained Exceptions	75-10
Viewing TopLink Log Messages From the Application Server Control Console	75-10
Profiler	75-11
TopLink Profiler	75-11
Oracle Dynamic Monitoring System (DMS)	75-11
Integrity Checker.....	75-12
Exception Handlers.....	75-12
Registering Descriptors	75-12
Sessions and CMP	75-13
Sessions and Sequencing.....	75-13
Server and Client Sessions	75-13
Three-Tier Architecture Overview	75-14
Advantages of the TopLink Three-Tier Architecture	75-15
Shared Resources	75-15
Providing Read Access.....	75-15
Providing Write Access.....	75-16
Security and User Privileges	75-17
Concurrency.....	75-18
Connection Allocation.....	75-18
Unit of Work Sessions	75-19
Isolated Client Sessions	75-19
Isolated Client Sessions and Oracle Virtual Private Database (VPD)	75-20
VPD With Oracle Database Proxy Authentication	75-21
VPD Without Oracle Database Proxy Authentication	75-21
Isolated Client Session Life Cycle.....	75-22
Isolated Client Session Limitations.....	75-23
Historical Client Sessions.....	75-24
Historical Client Session Limitations	75-25
Session Broker and Client Sessions	75-25
Session Broker Architecture	75-26
Committing a Transaction with a Session Broker	75-27

Committing a Session with a JTA Driver: Two-Phase Commits	75-27
Committing a Session Without a JTA Driver: Two-Stage Commits.....	75-27
Session Broker Session Limitations	75-27
Many-to-Many Join Tables and Direct Collection Tables	75-27
Session Broker Alternatives.....	75-28
Database Linking	75-28
Multiple Sessions	75-28
Database Sessions	75-28
Remote Sessions	75-29
Architectural Overview.....	75-30
Application Layer	75-30
Transport Layer.....	75-30
Server Layer.....	75-31
Remote Session Concepts.....	75-31
Securing Remote Session Access	75-31
Queries.....	75-31
Refreshing	75-31
Indirection	75-32
Cursored Streams.....	75-32
Unit of Work	75-32
Sessions and the Cache	75-32
Server and Database Session Cache	75-32
Isolated Session Cache.....	75-33
Historical Session Cache	75-33
Understanding the Session API	75-33

76 Creating Sessions

Session Creation Overview	76-1
Creating a Sessions Configuration	76-1
Using TopLink Workbench	76-2
Configuring a Sessions Configuration	76-2
Using TopLink Workbench	76-2
Creating a Server Session	76-4
Using TopLink Workbench	76-4
Using Java.....	76-5
Creating Session Broker and Client Sessions	76-6
Using TopLink Workbench	76-6
Using Java.....	76-8
Creating Database Sessions	76-8
Using TopLink Workbench	76-8
Using Java.....	76-10
Creating Remote Sessions	76-10
Using Java.....	76-10
Server	76-11
Client.....	76-11

77 Configuring a Session

Configuring Common Session Options	77-1
Configuring a Primary Mapping Project.....	77-2
Using TopLink Workbench	77-3
Using Java.....	77-3
Configuring a Session Login.....	77-4
Configuring Logging	77-4
Using TopLink Workbench	77-5
Using Java.....	77-7
Using Session Logging API	77-7
Configuring a Session to use java.util.logging Package.....	77-8
Configuring Logging in a CMP Application	77-9
Configuring Multiple Mapping Projects.....	77-9
Using TopLink Workbench	77-9
Using Java.....	77-10
Configuring a Performance Profiler	77-10
Using TopLink Workbench	77-11
Using Java.....	77-12
Configuring an Exception Handler.....	77-12
Using TopLink Workbench	77-12
Using Java.....	77-12
Configuring Customizer Class	77-13
Using TopLink Workbench	77-14
Configuring the Server Platform.....	77-14
Using TopLink Workbench	77-15
Using Java.....	77-17
Configuring Session Event Listeners	77-17
Using TopLink Workbench	77-17
Using Java.....	77-18
Configuring the Integrity Checker	77-18
Using Java.....	77-19
Configuring Connection Policy	77-19
Using TopLink Workbench	77-20
Using Java.....	77-21
Configuring Named Queries at the Session Level	77-21
Using Java.....	77-22

78 Acquiring and Using Sessions at Run Time

Session Acquisition Overview	78-1
Understanding the Session Manager	78-2
Multiple Sessions.....	78-2
Acquiring the Session Manager	78-2
Acquiring a Session from the Session Manager	78-3
Loading a Session from sessions.xml Using Defaults.....	78-3
Loading a Session from sessions.xml with an Alternative Class Loader.....	78-4
Loading a Session from an Alternative Session Configuration File	78-4

Loading a Session Without Logging In.....	78-5
Reloading and Refreshing Session Configuration.....	78-5
Refreshing a Session when the Class Loader Changes.....	78-6
Acquiring a Client Session	78-6
Acquiring an Isolated Client Session	78-7
Acquiring a Historical Client Session.....	78-7
Acquiring a Client Session That Uses Exclusive Connections	78-7
Acquiring a Client Session that Uses Connection Properties	78-8
Acquiring a Client Session that Uses a Named Connection Pool.....	78-8
Acquiring a Client Session that Does Not Use Lazy Connection Allocation	78-9
Logging In to a Session	78-9
Using Session API	78-9
Logging Out of a Session	78-10
Storing Sessions in the Session Manager Instance	78-10
Destroying Sessions in the Session Manager Instance	78-10
 79 Configuring Server Sessions	
Server Session Configuration Overview	79-1
Configuring Internal Connection Pools	79-1
Configuring External Connection Pools	79-2
 80 Configuring Exclusive Isolated Client Sessions for Virtual Private Database	
Exclusive Isolated Client Session Configuration Overview.....	80-1
PostAcquireExclusiveConnection Event Handler	80-1
Using Java.....	80-2
PreReleaseExclusiveConnection Event Handler.....	80-2
Using Java.....	80-2
NoRowsModifiedSessionEvent Event Handler.....	80-3
Using Java.....	80-3
ValidationException Handler	80-3
 81 Configuring Historical Client Sessions	
Historical Client Session Configuration Overview.....	81-1
Configuring Historical Client Sessions Using an Oracle Platform	81-1
Configuring Historical Client Sessions Using a TopLink HistoryPolicy.....	81-1
 82 Configuring Session Broker and Client Sessions	
Session Broker and Client Session Configuration Overview	82-1
Removing, Renaming, or Adding Sessions	82-1
Using TopLink Workbench	82-2
 83 Configuring Database Sessions	
Database Session Configuration Overview.....	83-1
Configuring External Connection Pools	83-1

Part XVII Data Access

84 Understanding Data Access

Data Access Concepts	84-1
Externally Managed Transactional Data Sources	84-1
Data Source Login Types	84-2
DatabaseLogin	84-2
EISLogin	84-3
Data Source Platform Types	84-3
Database Platforms	84-3
EIS Platforms	84-4
Authentication	84-5
Simple JDBC Authentication	84-5
Oracle Database Proxy Authentication	84-5
Auditing	84-6
Connections	84-6
Connection Pools	84-7
Internal Connection Pools	84-7
External Connection Pools	84-8
Default (Write) and Read Connection Pools	84-8
Sequence Connection Pools	84-8
Application-Specific Connection Pools	84-9
Understanding Data Access API	84-9
Login Inheritance Hierarchy	84-10
Platform Inheritance Hierarchy	84-10

85 Configuring a Data Source Login

Configuring Common Data Source Login Options	85-1
Configuring User Name and Password	85-1
Using TopLink Workbench	85-2
Configuring Password Encryption	85-2
Using Java	85-2
Configuring External Connection Pooling	85-2
Using TopLink Workbench	85-3
Configuring Properties	85-4
Using TopLink Workbench	85-4
Using Java	85-5
Configuring a Default Null Value at the Login Level	85-5
Using Java	85-6

86 Configuring a Database Login

Database Login Configuration Overview	86-1
Configuring a Relational Database Platform at the Session Level	86-1
Using TopLink Workbench	86-1
Configuring Database Login Connection Options	86-2
Using TopLink Workbench	86-2

Configuring Sequencing at the Session Level.....	86-4
Using TopLink Workbench	86-4
Using Java.....	86-5
Using the Platform Default Sequence	86-5
Configuring Multiple Sequences	86-6
Configuring Query Sequencing	86-7
Configuring a Table Qualifier	86-8
Using TopLink Workbench	86-8
Configuring JDBC Options	86-9
Using TopLink Workbench	86-9
Using Java.....	86-11
Configuring Advanced Options	86-11
Using TopLink Workbench	86-11
Configuring Oracle Database Proxy Authentication	86-12
Using Java.....	86-13

87 Configuring an EIS Login

EIS Login Configuration Overview.....	87-1
Configuring an EIS Data Source Platform at the Session Level	87-1
Using TopLink Workbench	87-1
Configuring EIS Connection Specification Options at the Session Level.....	87-2
Using TopLink Workbench	87-2

88 Creating an Internal Connection Pool

Internal Connection Pool Creation Overview	88-1
Using TopLink Workbench	88-1

89 Configuring an Internal Connection Pool

Internal Connection Pool Configuration Overview	89-1
Configuring Connection Pool Sizes	89-1
Using TopLink Workbench	89-2
Configuring Properties.....	89-2
Using TopLink Workbench	89-2
Using Java.....	89-3
Configuring a Nontransactional Read Login.....	89-3
Using TopLink Workbench	89-3
Configuring Connection Pool Connection Options	89-4
Using TopLink Workbench	89-4
Configuring Exclusive Read Connections.....	89-6
Using TopLink Workbench	89-6

Part XVIII Cache

90 Understanding the Cache

Cache Architecture	90-1
Session Cache.....	90-2

Unit of Work Cache	90-2
Cache Concepts	90-2
Cache Type and Object Identity	90-3
Full Identity Map	90-3
Weak Identity Map	90-3
Soft and Hard Cache Weak Identity Maps.....	90-4
No Identity Map.....	90-4
Guidelines for Configuring the Cache and Identity Maps	90-4
Understanding the Internals of Soft and Hard Cache Weak Identity Map	90-5
Querying and the Cache.....	90-6
Handling Stale Data.....	90-6
Configure a Locking Policy	90-6
Configure the Cache on a Per-Class Basis	90-7
Force a Cache Refresh When Required on a Per-Query Basis.....	90-7
Configure Cache Invalidation	90-7
Configure Cache Coordination.....	90-7
Explicit Query Refreshes.....	90-7
Refresh Policy	90-8
EJB Finders and Refresh Policy	90-8
Cache Invalidation	90-8
Cache Coordination	90-9
Cache Isolation	90-9
Cache Locking and Transaction Isolation.....	90-9
Cache Optimization	90-9
Understanding Cache Coordination	90-10
When to use Cache Coordination.....	90-10
Coordinated Cache Architecture	90-11
Session	90-11
Descriptor	90-11
Unit of Work	90-12
Coordinated Cache Types.....	90-12
JMS Coordinated Cache	90-12
RMI Coordinated Cache	90-12
CORBA Coordinated Cache.....	90-13
Custom Coordinated Cache	90-13
Understanding the Cache API	90-13
Object Identity API	90-13
Cache Refresh API	90-14
Cache Invalidation API	90-14
Cache Coordination API	90-15

91 Configuring a Coordinated Cache

Configuring Common Coordinated Cache Options	91-1
Configuring the Synchronous Change Propagation Mode	91-2
Using TopLink Workbench	91-2
Configuring a Service Channel	91-3
Using TopLink Workbench	91-3

Configuring a Multicast Group Address	91-4
Using TopLink Workbench	91-5
Configuring a Multicast Port	91-5
Using TopLink Workbench	91-6
Configuring a Naming Service Type	91-7
Configuring JNDI Naming Service Information	91-7
Using TopLink Workbench	91-8
Configuring RMI Registry Naming Service Information	91-9
Using TopLink Workbench	91-10
Configuring an Announcement Delay	91-11
Using TopLink Workbench	91-12
Configuring Connection Handling	91-13
Using TopLink Workbench	91-13
Configuring Context Properties	91-14
Using TopLink Workbench	91-14
Configuring a Packet Time-to-Live	91-15
Using TopLink Workbench	91-16

92 Configuring a JMS Coordinated Cache

JMS Coordinated Cache Configuration Overview	92-1
Configuring a Topic Name	92-1
Using TopLink Workbench	92-1
Configuring a Topic Connection Factory Name	92-2
Using TopLink Workbench	92-2
Configuring a Topic Host URL	92-3
Using TopLink Workbench	92-3

93 Configuring an RMI Coordinated Cache

RMI Coordinated Cache Configuration Overview	93-1

94 Configuring a CORBA Coordinated Cache

CORBA Coordinated Cache Configuration Overview	94-1

95 Configuring a Custom Coordinated Cache

Custom Coordinated Cache Configuration Overview	95-1
Configuring Transport Class	95-1
Using TopLink Workbench	95-1

Part XIX Queries

96 Understanding TopLink Queries

Query Types	96-1
Query Concepts	96-2
Call.....	96-3
DatabaseQuery	96-3

Data-Level and Object-Level Queries	96-3
Summary Queries	96-3
Descriptor Query Manager	96-3
TopLink Expressions	96-3
Query Keys.....	96-4
Query Languages	96-4
SQL Queries.....	96-4
EJB QL Queries.....	96-5
XML Queries.....	96-5
EIS Interactions.....	96-6
Query by Example	96-6
Building Queries	96-6
Executing Queries	96-7
Handling Query Results	96-8
Collection Query Results.....	96-8
Report Query Results	96-8
Stream and Cursor Query Results	96-8
Session Queries	96-9
Read-Object Session Queries	96-9
Create, Update, and Delete Object Session Queries.....	96-10
Database Queries	96-10
Object-Level Read Query	96-11
ReadObjectQuery	96-11
ReadAllQuery.....	96-11
Partial Object Queries.....	96-11
Join Reading and Object-Level Read Queries.....	96-12
Fetch Groups and Object Level Read Queries	96-12
Data-Level Read Query	96-13
DataReadQuery	96-13
DirectReadQuery	96-13
ValueReadQuery	96-13
Object-Level Modify Query	96-13
WriteObjectQuery.....	96-14
UpdateObjectQuery	96-14
InsertObjectQuery.....	96-14
DeleteObjectQuery.....	96-14
UpdateAllQuery.....	96-14
Object-Level Modify Queries and Privately Owned Parts	96-14
Data-Level Modify Query	96-14
Report Query	96-15
Named Queries	96-15
Call Queries	96-16
SQL Calls	96-16
SQLCall.....	96-17
StoredProcedureCall.....	96-17
StoredFunctionCall	96-17
Oracle Extensions.....	96-17

EJB QL Calls	96-18
Enterprise Information System (EIS) Interactions	96-18
IndexedInteraction	96-19
MappedInteraction	96-19
XMLInteraction	96-19
XQueryInteraction	96-19
QueryStringInteraction	96-19
Redirect Queries	96-19
Historical Queries	96-20
Using an ObjectLevelReadQuery With an AsOfClause	96-21
Using an ObjectLevelReadQuery With Expression Operator asOf	96-21
Using an ObjectLevelReadQuery in a Historical Session.....	96-22
Interface and Inheritance Queries	96-22
Descriptor Query Manager Queries	96-22
Configuring Named Queries	96-22
Configuring Default Query Implementations	96-22
Configuring Additional Join Expressions	96-23
EJB Finders	96-23
Predefined Finders	96-24
Predefined CMP Finders for EJB 2.0	96-24
Predefined CMP Finders for EJB 1.1	96-25
Predefined BMP Finders	96-25
Default Finders	96-26
Call Finders	96-26
DatabaseQuery Finders.....	96-26
Named Query Finders.....	96-27
Primary Key Finders.....	96-27
Expression Finders.....	96-27
EJB QL Finders.....	96-27
SQL Finders.....	96-28
Redirect Finders.....	96-28
The ejbSelect Method	96-28
Queries and the Cache	96-29
Configuring the Cache	96-29
Using In-Memory Queries	96-30
Configuring Cache Usage for In-Memory Queries.....	96-30
Expression Options for In-Memory Queries.....	96-31
Handling Exceptions Resulting From In-Memory Queries.....	96-33
Primary Key Queries and the Cache	96-34
Disabling the Identity Map Cache Update During a Read Query	96-34
Refreshing the Cache	96-35
Object Refresh.....	96-35
Cascading Object Refresh	96-35
Refreshing the Identity Map Cache During a Read Query	96-35
Caching Query Results in the Session Cache	96-36
Caching Query Results	96-36
Internal Query Cache Restrictions.....	96-36

Caching and EJB Finders.....	96-36
Caching Options.....	96-37
Disabling Cache for Returned Finder Results	96-37
Refreshing Finder Results.....	96-38
Understanding the Query API.....	96-38

97 Understanding TopLink Expressions

Understanding the Expression Framework	97-1
Expressions Compared to SQL	97-1
Expression Components	97-2
Boolean Logic.....	97-3
Database Functions	97-3
Mathematical Functions	97-4
XMLType Functions	97-4
Platform and User-Defined Functions	97-4
Expressions for One-to-One and Aggregate Object Relationships.....	97-5
Expressions for Joining and Complex Relationships	97-5
Understanding Joins.....	97-5
Using TopLink Expression API For Joins.....	97-6
Parameterized Expressions	97-7
Expression Method getParameter.....	97-8
Expression Method getField.....	97-9
Query Keys and Expressions	97-9
Using Multiple Expressions	97-10
Subselects and Subqueries	97-10
Parallel Expressions	97-11
Data Queries and Expressions	97-11
getField	97-11
getTable	97-12
Creating an Expression	97-12
Using TopLink Workbench	97-12
Adding Arguments.....	97-14
Using Java.....	97-15
Creating and Using a User-Defined Function	97-16
Making a User-Defined Function Available to a Specific Platform.....	97-16
Making a User-Defined Function Available to All Platforms	97-17
Using a User-Defined Function	97-17

98 Using Basic Query API

Using Session Queries	98-1
Reading Objects With a Session Query	98-1
Reading an Object With a Session Query	98-2
Reading All Objects With a Session Query	98-2
Refreshing an Object with a Session Query	98-2
Creating, Updating, and Deleting Objects With a Session Query	98-3
Writing a Single Object to the Database with a Session Query	98-3

Writing All Objects to the Database With a Session Query	98-3
Adding New Objects to the Database with a Session Query	98-4
Modifying Existing Objects in the Database with a Session Query	98-4
Deleting Objects in the Database with a Session Query	98-4
Using DatabaseQuery Queries	98-4
Reading Objects Using a DatabaseQuery	98-4
Basic DatabaseQuery Read Operations	98-5
Reading Objects Using Partial Object Queries	98-5
Reading Objects Using Report Queries	98-6
Reading Objects Using Query By Example	98-6
Specifying Read Ordering	98-8
Specifying a Collection Class	98-9
Specifying the Maximum Rows Returned	98-9
Configuring Query Timeout at the Query Level	98-10
Using Batch Reading	98-10
Using Join Reading	98-11
Creating, Updating, and Deleting Objects with a DatabaseQuery	98-12
Write Query Overview	98-12
UpdateAll Queries	98-13
Noncascading Write Queries	98-13
Disabling the Identity Map Cache During a Write Query	98-14
Reading Data With a DatabaseQuery	98-14
Using a DataReadQuery	98-15
Using a DirectReadQuery	98-15
Using a ValueReadQuery	98-15
Updating Data With a DatabaseQuery	98-15
Specifying a Custom SQL String in a DatabaseQuery	98-16
Specifying a Custom EJB QL String in a DatabaseQuery	98-16
Using Parameterized SQL and Statement Caching in a DatabaseQuery	98-17
Using Named Queries	98-17
Using SQL Calls	98-18
Using an SQLCall	98-18
Specifying a SQLCall Input Parameter	98-19
Specifying a SQLCall Output Parameter	98-19
Specifying a SQLCall Input / Output Parameter	98-20
Using a StoredProcedureCall	98-20
Specifying an Input Parameter	98-21
Specifying an Output Parameter	98-21
Specifying an Input / Output Parameter	98-22
Using an Output Parameter Event	98-22
Using a StoredFunctionCall	98-22
Using EJB QL Calls	98-23
Using EIS Interactions	98-24
Handling Exceptions	98-24
Handling Collection Query Results	98-25
Handling Report Query Results	98-25

99 Using Advanced Query API

Using Redirect Queries	99-1
Creating a Redirect Query	99-1
Using Historical Queries	99-2
Using Queries With Fetch Groups	99-2
Configuring Default Fetch Group Behavior	99-3
Querying With a Static Fetch Group	99-3
Querying With a Dynamic Fetch Group	99-4
Querying on Interfaces	99-4
Querying on an Inheritance Hierarchy	99-4
Appending Additional Join Expressions	99-4
Using Java	99-5
Using Queries on Variable One-to-One Mappings	99-5
Using Oracle Database Features	99-5
Oracle Hints	99-6
Hierarchical Queries	99-6
startWith Parameter	99-7
connectBy Parameter	99-7
orderSibling Parameter	99-7
Using EJB Finders	99-7
Creating a Finder	99-8
ejb-jar.xml Finder Options	99-9
Using DatabaseQuery Finders	99-10
Using Named Query Finders	99-10
Using Primary Key Finders	99-11
Using EJB QL Finders	99-11
Using SQL Finders	99-12
Using Redirect Finders	99-12
Using the ejbSelect Method	99-15
Handling Cursor and Stream Query Results	99-16
Cursors and Java Iterators	99-16
Traversing Data With Scrollable Cursors	99-16
Java Streams	99-17
Cursored Stream Support	99-17
Optimizing Streams	99-18
Using Cursors and Streams With EJB Finders	99-19
Building the Query	99-19
Executing the Finder From the Client in EJB 1.1	99-19
Executing the Finder From the Client in EJB 2.0	99-20
Using Queries and the Cache	99-21
Caching Results in a ReadQuery	99-21
Configuring Cache Expiration at the Query Level	99-22

Part XX Transactions

100 Understanding TopLink Transactions

Unit of Work Architecture	100-1
Unit of Work Transaction Context.....	100-2
Unit of Work Transaction Demarcation.....	100-2
JTA Controlled Transactions	100-3
OTS Controlled Transactions	100-3
CMP Controlled Transactions	100-3
Unit of Work Transaction Isolation	100-4
Unit of Work Concepts	100-4
Unit of Work Benefits	100-4
Unit of Work Life Cycle	100-5
Unit of Work and Change Policy	100-6
Deferred Change Detection Policy	100-7
Object-Level Change Tracking Policy	100-7
Attribute Change Tracking Policy	100-8
Change Policy Mapping Support	100-8
Clones and the Unit of Work.....	100-9
Nested and Parallel Units of Work.....	100-9
Nested Unit of Work	100-10
Parallel Unit of Work.....	100-10
Commit and Rollback Transactions	100-10
Commit Transactions	100-10
Rollback Transactions.....	100-11
Primary Keys	100-11
Unit of Work Optimization.....	100-11
Understanding the Unit of Work API	100-11
Unit of Work as Session	100-12
Reading and Querying Objects with the Unit of Work.....	100-12
Locking and the Unit of Work	100-12
Example Model Object and Schema	100-12

101 Using Basic Unit of Work API

Acquiring a Unit of Work	101-1
Creating an Object	101-2
Modifying an Object	101-2
Associating a New Target to an Existing Source Object	101-3
Associating Without Reference to the Cache Object.....	101-3
Associating With Reference to the Cache Object.....	101-4
Associating a New Source to an Existing Target Object	101-6
Associating an Existing Source to an Existing Target Object	101-7
Deleting Objects	101-7
Using privateOwnedRelationship	101-8
Explicitly Deleting from the Database	101-9
Understanding the Order in Which Objects Are Deleted	101-9

102 Using Advanced Unit of Work API

Registering and Unregistering Objects	102-1
Creating and Registering an Object in One Step	102-2
Using registerNewObject	102-2
Registering a New Object with registerNewObject	102-2
Associating New Objects With One Another	102-3
Using registerAllObjects	102-4
Using Registration and Existence Checking	102-5
Check Database	102-5
Assume Existence	102-5
Assume Nonexistence	102-5
Working with Aggregates.....	102-6
Unregistering Working Clones	102-6
Declaring Read-Only Classes	102-6
Configuring Read-Only Classes for a Single Unit of Work	102-6
Configuring Default Read-Only Classes	102-7
Read-Only Descriptors	102-7
Writing Changes Before Commit Time	102-7
Using Conforming Queries and Descriptors	102-8
Guidelines for Using Conforming	102-8
Use Only In-Memory Queries that Support Conforming.....	102-8
Consider How Conforming Affects Database Results	102-9
If You Want an Object in the Conformed Results, Register it in the Unit of Work.....	102-9
Using Conforming Queries.....	102-10
Using Conforming Descriptors	102-11
Conforming Query Alternatives	102-11
Using Unit of Work Method writeChanges Instead of Conforming.....	102-11
Using Unit of Work Properties Instead of Conforming	102-12
Merging Changes in Working Copy Clones	102-12
Resuming a Unit of Work After Commit	102-14
Reverting a Unit of Work	102-14
Using a Nested or Parallel Unit of Work	102-14
Parallel Unit of Work.....	102-15
Nested Unit of Work.....	102-15
Using a Unit of Work With Custom SQL	102-15
Controlling the Order of Delete Operations	102-16
Using the Unit of Work setShouldPerformDeletesFirst Method	102-16
Using the Descriptor addConstraintDependencies Method	102-16
Using deleteAllObjects Without addConstraintDependencies	102-16
Using deleteAllObjects with addConstraintDependencies	102-17
Using Optimistic Read Locking With forceUpdateToVersionField	102-17
Forcing a Check of the Optimistic Read Lock.....	102-17
Forcing a Version Field Update	102-18
Disabling forceUpdateToVersionField	102-20
Implementing User and Date Auditing with the Unit of Work	102-20
Integrating the Unit of Work With an External Transaction Service	102-20
Acquiring a Unit of Work With an External Transaction Service.....	102-21

Using a Unit of Work When an External Transaction Exists	102-21
Using a Unit of Work When No External Transaction Exists.....	102-22
Using the Unit of Work to Handle External Transaction Timeouts and Exceptions	102-23
External Transaction Commit Timeouts.....	102-23
External Transaction Commit Exceptions	102-23
Integrating the Unit of Work With CMP	102-24
CMP Transaction Attribute.....	102-24
Local Transactions.....	102-25
Nondeferred Changes	102-25
Database Transaction Isolation Levels	102-25
General Factors Affecting Transaction Isolation Level.....	102-26
External Applications	102-26
TopLink Coordinated Cache	102-26
DatabaseLogin Method setTransactionIsolation	102-27
Reading Through the Write Connection	102-27
Managing Cache Access.....	102-28
CMP and External Transactions	102-29
Read Uncommitted Level	102-29
Read Committed Level.....	102-30
Repeatable Read Levels.....	102-30
Serializable Read Levels	102-30
Troubleshooting a Unit of Work	102-30
Avoiding the Use of Post-Commit Clones	102-30
Determining Whether or Not an Object Is the Cache Object.....	102-31
Dumping the Contents of a Unit of Work	102-32
Handling Exceptions	102-33
Exceptions at Commit Time	102-33
Exceptions During Conforming.....	102-33
Validating a Unit of Work.....	102-34
Validating the Unit of Work Before Commit Time.....	102-34
.....	103-1
.....	103-1
.....	103-1
.....	103-1
.....	103-1
.....	103-1
.....	103-1
.....	103-1
.....	103-1
.....	103-1
.....	103-1
.....	103-1
.....	103-1
.....	103-1
.....	103-1
.....	103-1
.....	103-2
.....	103-2

.....	103-2
.....	103-2
.....	103-2
.....	103-2
.....	103-2
.....	103-2
.....	103-2
.....	103-2
.....	103-2
.....	103-2

Index

Preface

Oracle TopLink Developer's Guide explains how to use Oracle TopLink to design, implement, deploy, and optimize an advanced persistence or object-to-XML transformation layer for a wide range of Java 2 Enterprise Edition (J2EE) and Java applications. It describes TopLink mapping metadata, sessions, data access, queries, transactions (both with and without an external transaction controller), and cache.

It describes how to use TopLink application development tools and how to integrate TopLink with a variety of J2EE containers. It also introduces the concepts with which you should be familiar to get the most out of TopLink.

Audience

Oracle TopLink Developer's Guide is intended for application developers creating applications that use TopLink to manage persistence.

This document assumes that you are familiar with the concepts of object-oriented programming, the Enterprise JavaBeans (EJB) specification, and your own particular Java development environment.

The document also assumes that you are familiar with your particular operating system (such as Windows, UNIX, or other). The general operation of any operating system is described in the user documentation for that system, and is not repeated in this manual.

Documentation Accessibility

Our goal is to make Oracle products, services, and supporting documentation accessible, with good usability, to the disabled community. To that end, our documentation includes features that make information available to users of assistive technology. This documentation is available in HTML format, and contains markup to facilitate access by the disabled community. Accessibility standards will continue to evolve over time, and Oracle is actively engaged with other market-leading technology vendors to address technical obstacles so that our documentation can be accessible to all of our customers. For more information, visit the Oracle Accessibility Program Web site at

<http://www.oracle.com/accessibility/>

Accessibility of Code Examples in Documentation

Screen readers may not always correctly read the code examples in this document. The conventions for writing code require that closing braces should appear on an

otherwise empty line; however, some screen readers may not always read a line of text that consists solely of a bracket or brace.

Accessibility of Links to External Web Sites in Documentation

This documentation may contain links to Web sites of other companies or organizations that Oracle does not own or control. Oracle neither evaluates nor makes any representations regarding the accessibility of these Web sites.

TTY Access to Oracle Support Services

Oracle provides dedicated Text Telephone (TTY) access to Oracle Support Services within the United States of America 24 hours a day, seven days a week. For TTY support, call 800.446.2398.

Related Documentation

For more information, see the following documents:

- *Oracle TopLink Release Notes*
- Oracle Application Server Release Notes
- *Oracle TopLink Getting Started Guide*
- *Oracle TopLink API Reference*

Conventions

The following text conventions are used in this document:

Convention	Meaning
boldface	Boldface type indicates graphical user interface elements associated with an action, or terms defined in text or the glossary.
<i>italic</i>	Italic type indicates book titles, emphasis, or placeholder variables for which you supply particular values.
monospace	Monospace type indicates commands within a paragraph, URLs, code in examples, text that appears on the screen, or text that you enter.

Part I

Building a TopLink Application

This part describes the architecture of TopLink and how TopLink fits into your development process. It contains the following chapters:

- [Chapter 1, "Understanding TopLink"](#)

This chapter contains general information on TopLink. It discusses the TopLink application space, development process, components, and the TopLink metamodel.

- [Chapter 2, "Understanding TopLink Application Development"](#)

This chapter contains an overview of the key stages in the TopLink development process including choosing an application architecture and platform, object and relational mapping, building the persistence layer, and deploying and maintaining a TopLink application.

Understanding TopLink

Oracle TopLink is an advanced, object-persistence and object-transformation framework that provides development tools and run-time capabilities that reduce development and maintenance efforts, and increase enterprise application functionality.

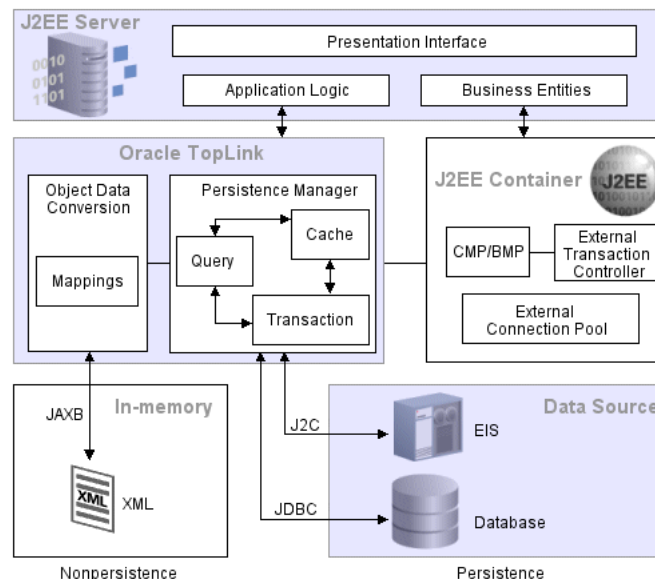
This chapter describes the following topics:

- [What is TopLink?](#)
- [Solving the Object-Persistence Impedance Mismatch](#)
- [TopLink Key Features](#)
- [TopLink Application Architectures](#)

What is TopLink?

Oracle TopLink builds high-performance applications that store persistent object-oriented data in a relational database. It successfully transforms object-oriented data into either relational data or Extensible Markup Language (XML) elements.

Figure 1–1 TopLink Runtime Architecture



Using TopLink, you can integrate persistence and object-transformation into your application, while staying focused on your primary domain problem by taking advantage of an efficient, flexible, and field-proven solution (see ["Solving the Object-Persistence Impedance Mismatch"](#) on page 1-2).

TopLink is suitable for use with a wide range of Java 2 Enterprise Edition (J2EE) and Java application architectures (see ["TopLink Application Architectures"](#) on page 1-4). Use TopLink to design, implement, deploy, and optimize an advanced, object-persistence and object-transformation layer that supports a variety of data sources and formats, including:

- Relational—for transactional persistence of Java objects to a relational database accessed using Java Database Connectivity (JDBC) drivers.
- Object-Relational—for transactional persistence of Java objects to special purpose structured data source representations optimized for storage in object-relational databases such as Oracle Database.
- Enterprise information system (EIS)—for transactional persistence of Java objects to a nonrelational data source accessed using a J2EE Connector architecture (J2C) adapter, and any supported EIS record type, including indexed, mapped, or XML.
- XML—for nontransactional, nonpersistent (in-memory) conversion between Java objects and XML Schema Document (XSD) based XML documents, using Java Architecture for XML Binding (JAXB).

TopLink includes support for container-managed persistence (CMP) containers from a variety of vendors—such as Oracle Containers for J2EE (OC4J), IBM WebSphere application server, and BEA WebLogic Server—and support for base classes that simplify bean-managed persistence (BMP) development.

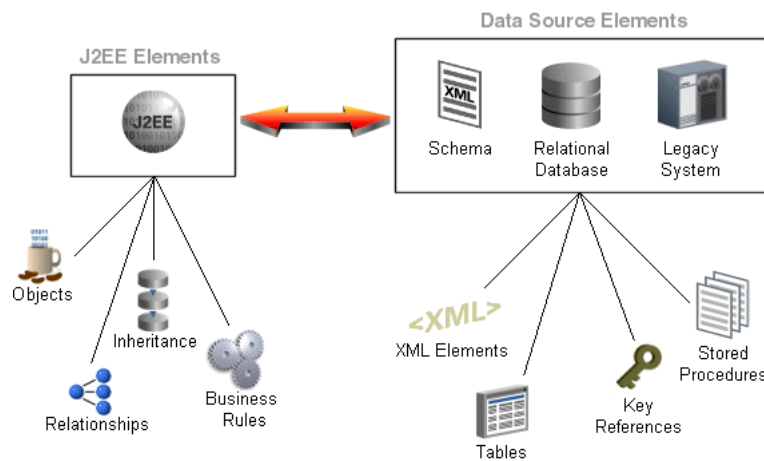
The extensive suite of development tools that TopLink provides, including Oracle TopLink Workbench, lets you quickly capture and define object-to-data source and object-to-data representation mappings in a flexible, efficient metadata format (see ["Understanding TopLink Metadata"](#) on page 2-19).

The TopLink runtime lets your application exploit this mapping metadata with a simple session facade that provides in-depth support for data access, queries, transactions (both with and without an external transaction controller), and caching.

For more information about TopLink, see ["TopLink Key Features"](#) on page 1-4.

Solving the Object-Persistence Impedance Mismatch

Java-to-data source integration is a widely underestimated problem when creating enterprise Java applications. This complex problem involves more than simply reading from and writing to a data source. The data source elements include tables, rows, columns, and primary and foreign keys. The Java and J2EE include entity classes (regular Java classes or Enterprise JavaBeans (EJB) entity beans), business rules, complex relationships, and inheritance. In a nonrelational data source, you must match your Java entities with EIS records or XML elements and schemas. These differences (as shown in [Figure 1-2](#)) are known as the object-persistence impedance mismatch.

Figure 1–2 Solving Object-Persistence Impedance Mismatch

Successful solution requires bridging these different technologies, and solving the object-persistence impedance mismatch—a challenging and resource-intensive problem. To solve this problem, you must resolve the following issues between J2EE and data source elements:

- Fundamentally different technologies
- Different skill sets
- Different staff and ownership for each of the technologies
- Different modeling and design principles

Application developers need a product that enables them to integrate Java applications with any data source, without compromising ideal application design or data integrity. In addition, Java developers need the ability to store (that is, **persist**) and retrieve business domain objects using a relational database or a nonrelational data source as a repository.

TopLink Solution

TopLink addresses the disparity between Java objects and data sources. TopLink is a persistence framework that manages relational, object-relational, EIS, and XML mappings in a seamless manner. This allows developers to rapidly build applications that combine the best aspects of object technology and the specific data source.

TopLink enables you to do the following:

- Persist Java objects to virtually *any* relational database supported by a JDBC-compliant driver.
- Persist Java objects to virtually *any* nonrelational data source supported by a J2EE Connector architecture (J2C) adapter using indexed, mapped, or XML enterprise information system (EIS) records.
- Perform in-memory conversions between Java objects and XML Schema (XSD) based XML documents using JAXB.
- Map *any* object model to *any* relational or nonrelational schema, using the Oracle TopLink Workbench graphical mapping tool.
- Use TopLink successfully, even if you are unfamiliar with SQL or JDBC, because TopLink offers a clear, object-oriented view of data sources.

TopLink Key Features

TopLink provides an extensive and thorough set of features. Java developers can use these features to rapidly build high-performance enterprise applications that are both scalable and maintainable.

Some of the primary features of TopLink are the following:

- Nonintrusive, flexible, metadata-based architecture (see "[Understanding TopLink Metadata](#)" on page 2-19)
- Comprehensive visual TopLink Workbench
- Advanced mapping support and flexibility (relational, object-relational, EIS, and XML)
- Object caching support
- Query flexibility
- Just-in-time reading
- Caching
- Object-level transaction support and integration
- Locking
- Multiple performance tuning options
- Architectural flexibility

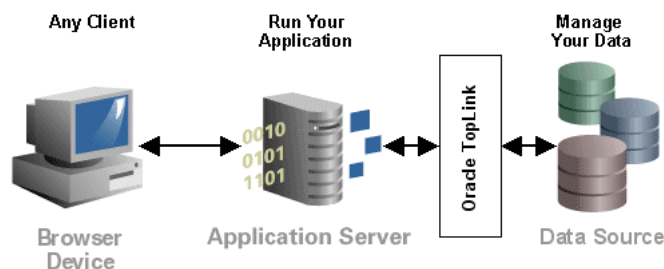
For additional information, see the TopLink page on OTN:

<http://www.oracle.com/technology/products/ias/toplink/index.html>

TopLink Application Architectures

You can use TopLink in a variety of application architectures, including three- and two-tier architectures, with or without J2EE, to access a variety of data types on both relational and nonrelational data sources.

Figure 1-3 TopLink and Your Application Architecture



For more information on strategies for incorporating TopLink into your application architecture, see "[Designing Your Application With TopLink](#)" on page 2-3.

This section introduces some common enterprise architectures used by TopLink applications:

- **Three-Tier**

The three-tier (or J2EE Web) application is one of the most common TopLink architectures. This architecture is characterized by a server-hosted environment in

which the business logic, persistent entities, and the Oracle TopLink Foundation Library all exist in a single Java Virtual Machine (JVM). See ["Understanding the Three-Tier Architecture"](#) on page 2-21 for more information.

The most common example of this architecture is a simple three-tier application in which the client browser accesses the application through servlets, JavaServer Pages (JSP) and HTML. The presentation layer communicates with TopLink through other Java classes in the same JVM, to provide the necessary persistence logic. This architecture supports multiple servers in a clustered environment, but there is no separation across JVMs from the presentation layer and the code that invokes the persistence logic against the persistent entities using TopLink.

- **EJB Session Bean Facade**

A popular variation on the three-tier application involves wrapping the business logic, including the TopLink access, in EJB session beans. This architecture provides a scalable deployment and includes integration with transaction services from the host application server. See ["Understanding the EJB Session Bean Facade Architecture"](#) on page 2-24 for more information.

Communication from the presentation layer occurs through calls to the EJB session beans. This architecture separates the application into different tiers for the deployment. The session bean architecture can persist either Java objects or EJB entity beans.

- **EJB Entity Beans with CMP**

TopLink provides CMP support for applications that require the use of EJB entity beans. This support is available on the leading application servers. TopLink CMP support provides the developer with an EJB 1.n and 2.n (and, when used with OC4J, preliminary EJB 3.0) CMP solution transparent to the application code, but still offers all the TopLink run-time benefits. See ["Understanding the EJB Entity Beans with CMP Architecture"](#) on page 2-27 for more information.

Applications can access TopLink-enabled CMP entity beans directly from the client, or from within a session bean layer. TopLink also offers the ability to use regular Java objects in relationships with EJB entity beans.

- **EJB Entity Beans with BMP**

Another option for using EJB entity beans is to use TopLink BMP in the application. This architecture enables developers to access the persistent data through the EJB application programming interface (API), but is platform independent. See ["Understanding the EJB Entity Beans With BMP Architecture"](#) on page 2-33 for complete information.

The BMP approach is portable—that is, after you create an application, you can move it from one application server platform to another.

- **Web Services**

A Web services architecture is similar to the three-tier or session-bean architecture. However, in a Web services architecture you encapsulate business logic (the service) in a Web service instead of (or in addition to) using session beans. In a Web services architecture, clients communicate with your application using XML.

As in any architecture, you can use TopLink to persist objects to relational or EIS data sources. However, in a Web services architecture you can also use TopLink to map your object model to an XML schema for use with the Web service or as the Web service XML serializer.

See "[Understanding the Web Services Architecture](#)" on page 2-36 for more information

- **Two-Tier**

A two-tier (or client/server) application is one in which the TopLink application accesses the database directly. Although less common than the other architectures discussed here, TopLink supports this architecture for smaller or embedded data processing applications. See "[Understanding the Two-Tier Architecture](#)" on page 2-23 for more information.

Understanding TopLink Application Development

This chapter describes how to build a TopLink application, including suggested development processes, architectures, and technologies.

This chapter includes information on the following topics:

- [Developing Your Application With TopLink](#)
- [Designing Your Application With TopLink](#)
- [Selecting an Architecture With TopLink](#)
- [Building and Using the Persistence Layer](#)
- [Deploying the Application](#)
- [Optimizing and Customizing the Application](#)
- [Troubleshooting the Application](#)
- [Understanding Object Persistence](#)
- [Understanding TopLink Metadata](#)

Developing Your Application With TopLink

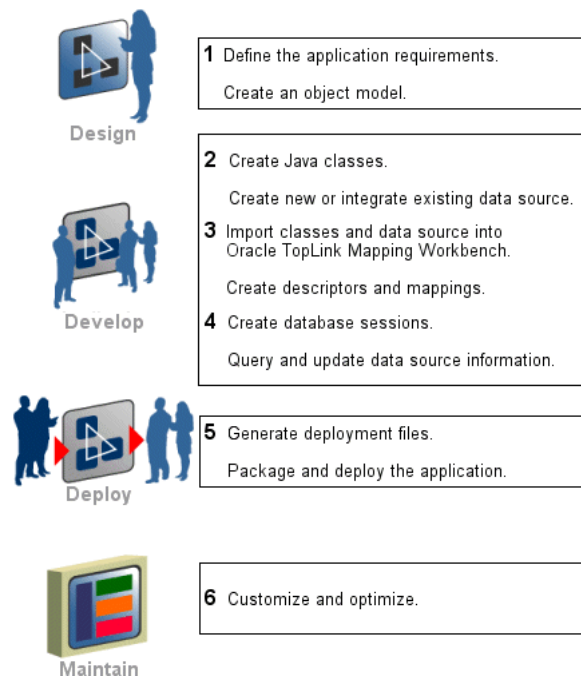
To ensure the best design for your TopLink application, Oracle recommends that you follow an iterative step-by-step development process. The flexibility of TopLink lets you use *any* development tool.

This section describes following recommended development process:

- [Typical Development Stages](#)
- [Oracle Development Support](#)

Typical Development Stages

This section describes the general development stages of a TopLink application. [Figure 2–1](#) illustrates the TopLink development process.

Figure 2-1 TopLink Development Process**Design the Application (1)**

Define your application requirements, select an architecture, and determine the target platform. See ["Designing Your Application With TopLink"](#) on page 2-3 for more information. Remember, TopLink works with *any* architecture and *any* platform.

When designing the application, you should also create an object model for the application. See ["Understanding Object Persistence"](#) on page 2-15 for more information. It is important to create the object model *before* using TopLink to map objects, because defining persistent mappings for an incorrect or rapidly changing model can be very difficult.

Develop the Application (2, 3, 4)

Create the Java classes and decide how the classes should be implemented by the data source. When working with a legacy system, decide how the classes relate to the existing data. If there is no legacy data source to integrate, decide how to store each class in the data source and create the required schema. Alternatively, you may use TopLink to create your initial tables.

Using TopLink Workbench, create descriptors and mappings for the persistent classes. Use TopLink sessions to manipulate the persistent classes, including querying and changing data. See [Part II, "Using TopLink Development Tools"](#) for more information.

Avoid building all your model's descriptors in a single iteration. Start with a small subset of your classes. Build and test their descriptors, then gradually add new descriptors and relationships. This lets you catch common problems before they proliferate through your entire design.

Write Java code to use database sessions. Sessions are used to query for database objects and write objects to the database. See [Chapter 75, "Understanding TopLink Sessions"](#) for more information.

Deploy the Application (5)

Generate, package, then deploy the necessary files to your application server. The required information will vary, depending on your environment and architecture. See [Part III, "Deploying a TopLink Application"](#) for more information.

Maintain the Application (6)

TopLink includes many options that can enhance application performance. You can customize most aspects of TopLink to suit your requirements. Use advanced TopLink features or write custom querying routines to access the database in specific ways, and to optimize performance. See [Part IV, "Optimizing and Customizing a TopLink Application"](#) for more information.

Oracle Development Support

Oracle provides additional support for TopLink developers on the Oracle Technology Network (OTN), including the following:

- Metalink support
- Discussion forums
- How-to documents
- Examples

You must register online before using OTN; registration is free of charge and can be done at:

<http://www.oracle.com/technology/membership>

If you already have a user name and password for OTN, then you can go directly to the documentation section of the OTN Web site at

<http://www.oracle.com/technology/docs>

Using your OTN user name and password, you can also access the TopLink developer's forum to post questions and get answers about using TopLink. See:

<http://forums.oracle.com/forums/forum.jsp?forum=48>

Designing Your Application With TopLink

When you design your application, you must choose how and where to use TopLink. You can use TopLink to perform a variety of persistence and data transformation functions (see ["Understanding TopLink Usage"](#) on page 2-3) on a variety of Java-supporting platforms (see ["Understanding Target Platforms"](#) on page 2-4). When you design your application architecture, keep these capabilities in mind (see ["Selecting an Architecture With TopLink"](#) on page 2-5).

Understanding TopLink Usage

This section describes the basic ways in which you can use TopLink, including the following usage types:

- [Relational Database Usage](#)
- [Object-Relational Database Usage](#)
- [Oracle XML Database \(XDB\) Usage](#)

- [Enterprise Information System \(EIS\) Usage](#)
- [XML Usage](#)

Relational Database Usage

You can use TopLink to persist Java objects to relational databases that support SQL data types accessed using JDBC.

For more information, see ["Building Relational Projects for a Relational Database"](#) on page 20-6.

Object-Relational Database Usage

You can use TopLink to persist Java objects to object-relational databases that support data types specialized for object storage (such as Oracle Database) accessed using JDBC.

For more information, see ["Building Relational Projects for an Object-Relational Database"](#) on page 20-6.

Oracle XML Database (XDB) Usage

You can use TopLink to persist XML documents to an Oracle XML database using TopLink direct-to-XMLType mappings.

For more information, see ["Relational Projects"](#) on page 20-6 and ["Direct to XMLType Mapping"](#) on page 36-4.

Enterprise Information System (EIS) Usage

You can use TopLink to persist Java objects to an EIS data source using a J2C adapter.

In this scenario, the application invokes EIS data source-defined operations by sending EIS interactions to the J2C adapter. Operations can take (and return) EIS records. Using TopLink EIS descriptors and mappings, you can easily map Java objects to the EIS record types supported by your J2C adapter and EIS data source.

This usage is common in applications that connect to legacy data sources and is also applicable to Web services.

For more information, see ["EIS Projects"](#) on page 20-7.

XML Usage

You can use TopLink for in-memory, nonpersistent Java object-to-XML transformation with XML Schema (XSD) based XML documents and JAXB.

You can use the TopLink JAXB compiler with your XSD to generate both JAXB-specific artifacts (such as content and element interfaces, implementation classes, and object factory class) and TopLink-specific artifacts (such as sessions and project XML files and TopLink Workbench project). For more information, see ["TopLink Support for Java Architecture for XML Binding \(JAXB\)"](#) on page 20-10).

This usage has many applications, including messaging and Web services.

For more information, see ["XML Projects"](#) on page 20-9.

Understanding Target Platforms

TopLink supports any enterprise architecture that uses Java, including the following:

- Java application servers and J2EE containers, such as Oracle Application Server and Oracle Containers for J2EE (OC4J)
- Java-supporting databases, such as Oracle Database
- Java-compatible browsers, such as Netscape and Internet Explorer
- Server Java platforms, such as AS/400, OS/390, and UNIX

Application packaging requirements of the specific target platform (for deployment in the host Java or J2EE environment) influences how developers use and configure TopLink. For example, developers package a J2EE application in an Enterprise Archive (EAR) file. Within the EAR file, there are several ways to package persistent entities within Web Archive (WAR) and Java Archive (JAR). How developers configure TopLink depends, in part, on how they package the application and how they use the host application server class loader.

In addition, TopLink provides custom CMP integration for a variety of application servers.

For detailed information about supported application server versions, custom integration, and configuration requirements, see ["Integrating TopLink With an Application Server"](#) on page 7-1.

Selecting an Architecture With TopLink

This section describes some of the key aspects of application architecture that apply to TopLink and discusses the various options available for each, including the following:

- [Tiers](#)
- [Service Layer](#)
- [Data Access](#)
- [Caching](#)
- [Locking](#)

Tiers

This section describes choices you need to make when deciding on how to separate client and server functionality in your application architecture.

These choices can be summarized as follows:

- [Three Tier](#)
 - [J2EE or Non-J2EE](#)
 - [Client](#)
 - * Web client
 - * XML/Web service client
 - * Java (fat) client
- [Two Tier](#)

Three Tier

Oracle recommends a three-tier application architecture. With a three-tier architecture, Oracle recommends using TopLink server sessions and client sessions (see ["Server and](#)

[Client Sessions](#)" on page 75-13) and the TopLink unit of work (see ["Understanding TopLink Transactions"](#) on page 100-1).

For more information, see ["Understanding the Three-Tier Architecture"](#) on page 2-21.

J2EE or Non-J2EE You can use TopLink in a J2EE or non-J2EE application architecture. Oracle recommends that you use a J2EE application architecture.

With a J2EE application, you should use external connection pools (see ["External Connection Pools"](#) on page 84-8). You may consider using Java Transaction API (JTA) integration (see ["JTA Controlled Transactions"](#) on page 100-3), EJB session beans, and EJB entity beans.

With a non-J2EE application, you should use internal connection pools (see ["Internal Connection Pools"](#) on page 84-7).

Client In a three-tier application architecture, you can implement any of the following types of client:

- Web client—Oracle recommends that you implement a Web client.
- XML/Web service client—With this client type, you can use TopLink XML (see ["XML Usage"](#) on page 2-4).
- Java (fat) client—With this client type, you can choose the means of communicating with the server:
 - EJB session beans—Oracle recommends this approach. You may consider using the `UnitOfWork` method `mergeClone` to handle merging deserialized objects (see ["Merging Changes in Working Copy Clones"](#) on page 102-12). The disadvantage of this approach is that your application must handle serialization. Avoid serializing deep object graphs. You should use indirection (["Configuring Indirection"](#) on page 35-3). Consider using the data-transfer-object pattern.
 - XML/Web service—Use TopLink XML (see ["XML Usage"](#) on page 2-4).
 - EJB entity bean—Use TopLink CMP integration or BMP support. The disadvantage of this approach is that remote entity beans may not perform or scale well.
 - RMI—You may consider using a TopLink remote session (see ["Remote Sessions"](#) on page 75-29). The disadvantage of this approach is that a remote session is stateful and may not scale well.

See also ["Service Layer"](#) on page 2-6.

Two Tier

With a two-tier application architecture, Oracle recommends using TopLink database sessions (see ["Database Sessions"](#) on page 75-28) and the TopLink unit of work (see ["Understanding TopLink Transactions"](#) on page 100-1). The disadvantages of this architecture are that it is not Web-enabled and does not scale well to large deployments.

For more information, see ["Understanding the Two-Tier Architecture"](#) on page 2-23.

Service Layer

This section describes choices you need to make when deciding on how to encapsulate your application's business logic (or service).

These choices can be summarized as follows:

- [EJB Session Beans](#)
 - [Stateful](#)
 - [Stateless](#)
- [EJB Entity Beans](#)
 - [Container-Managed Persistence \(CMP\)](#)
 - [Bean-Managed Persistence \(BMP\)](#)
- [Plain Old Java Objects \(POJO\)](#)

See also:

- ["Data Access"](#) on page 2-8
- ["Caching"](#) on page 2-9.

EJB Session Beans

Oracle recommends using EJB session beans.

With EJB session beans, you should use JTA integration (see ["JTA Controlled Transactions"](#) on page 100-3) and external connection pools (see ["External Connection Pools"](#) on page 84-8). You should acquire a unit of work using `Server` method `getActiveUnitOfWork` (see ["Acquiring a Unit of Work With an External Transaction Service"](#) on page 102-21). If your EJB session bean and data source are not in the same JVM, you may consider using `UnitOfWork` method `mergeClone` to handle merging deserialized objects (see ["Merging Changes in Working Copy Clones"](#) on page 102-12).

For more information, see ["Understanding the EJB Session Bean Facade Architecture"](#) on page 2-24.

Stateful If you are using stateful EJB session beans, then note that a reference to a client Session cannot be passivated. In this case, you must reacquire a client Session (["Acquiring a Session at Run Time With the Session Manager"](#) on page 75-5) on activate or per request.

Stateless If you are using stateless EJB session beans, you must acquire new client Session (["Acquiring a Session at Run Time With the Session Manager"](#) on page 75-5) for each request.

EJB Entity Beans

EJB entity bean architectures are slightly different from other TopLink architectures, because the EJB entity bean interfaces hide TopLink functionality completely from the client application developer.

You can use entity beans in almost any J2EE application. For TopLink, *how the application uses the entity beans* is not important; *how entity beans are mapped and implemented* is important to TopLink.

Container-Managed Persistence (CMP) Oracle recommends using CMP entity beans. In this case, you should use the TopLink CMP integration for your application server. You must ensure that you are using a J2EE server that TopLink supports (see ["Integrating TopLink With an Application Server"](#) on page 7-1).

For more information, see ["Understanding the EJB Entity Beans with CMP Architecture"](#) on page 2-27.

Bean-Managed Persistence (BMP) If you are using BMP entity beans, you should use the TopLink BMP integration. The disadvantages of this architecture are that the BMP architecture is restrictive and may not be provide good performance.

For more information, see ["Understanding the EJB Entity Beans With BMP Architecture"](#) on page 2-33.

Plain Old Java Objects (POJO)

If you choose to build your service layer with non-EJB Java objects with a J2EE application server, you should use external connection pools (see ["External Connection Pools"](#) on page 84-8). If you use a non-J2EE Web server, you should use internal connection pools (see ["Internal Connection Pools"](#) on page 84-7). In either case, you may consider using JTA integration (see ["JTA Controlled Transactions"](#) on page 100-3).

Data Access

This section describes choices you need to make when deciding on what type of data your application architecture must support.

These choices can be summarized as follows:

- [Data Type](#)
- [Multiple Data Sources](#)
- [Isolating Data Access](#)
- [Historical Data Access](#)

See also ["Locking"](#) on page 2-10.

Data Type

You can use TopLink to manage any of the following types of data:

- relational (see ["Relational Database Usage"](#) on page 2-4)
- object-relational (see ["Object-Relational Database Usage"](#) on page 2-4)
- Oracle XDB (see ["Oracle XML Database \(XDB\) Usage"](#) on page 2-4)
- EIS, nonrelational, legacy data (see ["Enterprise Information System \(EIS\) Usage"](#) on page 2-4)
- XML and Web service data (see ["XML Usage"](#) on page 2-4)

Multiple Data Sources

If your application architecture must access more than one data source, Oracle recommends that you use a session broker (see ["Session Broker and Client Sessions"](#) on page 75-25) and JTA integration (see ["JTA Controlled Transactions"](#) on page 100-3) for two-phase commit.

Alternatively, you may use multiple sessions.

Isolating Data Access

If your application architecture requires that some data be restricted to a private cache and isolated from the TopLink shared session cache, Oracle recommends that you use an isolated session (see ["Isolated Client Sessions"](#) on page 75-19). You can also use an

isolated session with the Oracle Virtual Private Database (VPD) feature (see ["Isolated Client Sessions and Oracle Virtual Private Database \(VPD\)"](#) on page 75-20).

Historical Data Access

If your data source maintains past or historical versions of objects, Oracle recommends that you use a TopLink historical session (see ["Historical Client Sessions"](#) on page 75-24) to access this historical data so that you can express read queries conditional on how your objects are changing over time.

Caching

This section describes choices you need to make when deciding on how to use the TopLink cache (see ["Understanding the Cache"](#) on page 90-1) in your application architecture.

These choices can be summarized as follows:

- [Cache Type](#)
- [Refreshing](#)
- [Cache Coordination](#)
 - [Protocol](#)
 - [Synchronization](#)

See also ["Locking"](#) on page 2-10.

Cache Type

Choose a cache type (see ["Cache Type and Object Identity"](#) on page 90-3) appropriate for the type of data your application processes. For example, consider a weak identity map for volatile data (see ["Guidelines for Configuring the Cache and Identity Maps"](#) on page 90-4).

Refreshing

Consider how your application architecture may be affected by stale data (see ["Handling Stale Data"](#) on page 90-6): for example, consider using query or descriptor refresh options (see ["Refreshing"](#) on page 2-9) or cache invalidation (see ["Cache Invalidation"](#) on page 90-8). Consider using an isolated session's cache (see ["Isolated Client Sessions"](#) on page 75-19) for volatile data.

Avoid using no identity map (see ["No Identity Map"](#) on page 90-4) for objects that are involved in relationships or that require object identity.

Cache Coordination

TopLink provides a distributed cache coordination feature that allows multiple, possibly distributed, instances of a session to broadcast object changes among each other so that each session's cache is kept up to date (see ["Understanding Cache Coordination"](#) on page 90-10). Before using cache coordination, ensure that it is appropriate for your application (see ["When to use Cache Coordination"](#) on page 90-10).

Protocol You can configure a coordinated cache to broadcast changes using any of the following communication protocols:

- Java Message Service (JMS)—Oracle recommends using a JMS coordinated cache (see ["JMS Coordinated Cache"](#) on page 90-12).

- Remote Method Invocation (RMI)—Oracle recommends that you use RMI cache coordination only if you require synchronous change propagation (see ["Configuring the Synchronous Change Propagation Mode"](#) on page 91-2). For more information, see ["RMI Coordinated Cache"](#) on page 90-12.
- Common Object Request Broker Architecture (CORBA)—Currently, TopLink provides support for the Sun ORB (see ["CORBA Coordinated Cache"](#) on page 90-13).

Synchronization You can configure synchronization strategy that a coordinated cache uses to determine what it broadcasts when an object changes. You can configure this at the project (see ["Configuring Cache Coordination Change Propagation at the Project Level"](#) on page 22-17) or descriptor (["Configuring Cache Coordination Change Propagation at the Descriptor Level"](#) on page 28-38) level:

- invalidate changed objects—Propagate an object invalidation that marks the object as invalid in all other sessions. This tells other sessions that they must update their cache from the data source the next time this object is read. Oracle recommends using this synchronization strategy.
- synchronize changes—Propagate a change notification that contains each changed attribute.
- synchronize changes and new objects—Propagate a change notification that contains each changed attribute. For new objects, propagate an object creation (along with all the new instance's attributes).

Locking

This section describes choices you need to make when deciding on how to use TopLink locking options in your application architecture. Oracle strongly recommends always using a locking policy in a concurrent system (see ["Configuring Locking Policy"](#) on page 28-62).

These choices can be summarized as follows:

- [Optimistic Locking](#)
- [Pessimistic Locking](#)

If you are building a three-tier application, be aware of how that architecture affects how you use locking (see ["Locking in a Three-Tier Application"](#) on page 26-21).

For more information, see ["Understanding Descriptors and Locking"](#) on page 26-17.

Optimistic Locking

Oracle recommends using TopLink optimistic locking. With optimistic locking, all users have read access to the data. When a user attempts to write a change, the application checks to ensure the data has not changed since the user read the data.

You can use version (see ["Optimistic Version Locking Policies"](#) on page 26-17) or field (see ["Optimistic Field Locking Policies"](#) on page 26-20) locking policies. Oracle recommends using version locking policies.

Pessimistic Locking

With pessimistic locking, the first user who accesses the data with the purpose of updating it locks the data until completing the update. The disadvantage of this approach is that it may lead to reduced concurrency and deadlocks.

Consider using pessimistic locking support at the query level (see ["Configuring Named Query Options"](#) on page 28-22).

If are using CMP, you may consider using bean-level pessimistic locking support (see ["Pessimistic Locking Policy"](#) on page 26-21).

Building and Using the Persistence Layer

Oracle TopLink requires that classes must meet certain minimum requirements before they can become persistent. TopLink also provides alternatives to most requirements. TopLink uses a nonintrusive approach by employing a metadata architecture that allows for minimal object model intrusions.

This section includes the following information:

- [Implementation Options](#)
- [Persistent Class Requirements](#)
- [Persistence Layer Components](#)
- [Using the Persistence Layer](#)

Implementation Options

Persistence layer components may be generated as metadata (see ["Understanding TopLink Metadata"](#) on page 2-19) from TopLink Workbench, or expressed as Java classes.

Oracle recommends using TopLink Workbench to create the necessary metadata (stored as XML). You can easily export and update the `project.xml` and `sessions.xml` files. This reduces development effort by eliminating the need to regenerate and recompile Java code each time you change the project. With TopLink Workbench, you write Java code only for your own application classes and any necessary amendment methods. For information about the XML structure of the `project.xml` and `sessions.xml` files, refer to the appropriate XML schemas (XSD) in the `<TOPLINK_HOME>\config\xsds` directory.

To use Java code, you must manually write code for each element of the TopLink project including: project, login, platform, descriptors, and mappings. This may be more efficient if your application is model-based and relies heavily on code generation. Depending on the type of project you are creating, TopLink Workbench can export Java code for projects, tables, and your model source (see ["Exporting Project Information"](#) on page 21-13).

Persistent Class Requirements

The following requirements apply to CMP EJB 1.0 and plain Java objects. For CMP EJB 2.0 entity beans, the bean requirements are defined by the EJB specification.

You can use direct access on private or protected attributes. For more information on direct and method access, see ["Configuring Method Accessing"](#) on page 35-14.

When using *nontransparent* indirection, the attributes must be of the type `ValueHolderInterface` rather than the original attribute type. The value holder does not instantiate a referenced object until it is needed.

TopLink provides *transparent* indirection for `Collection`, `List`, `Set`, and `Map` attribute types for any collection mappings. Using transparent indirection does not

require the use of the `ValueHolderInterface` or any other object model requirements.

See ["Indirection"](#) on page 33-5 for more information on indirection and transparent indirection.

Persistence Layer Components

Typically, the TopLink persistence layer contains the following components:

- [Mapping Metadata](#)
- [Session Metadata](#)
- [Cache](#)
- [Queries and Expressions](#)
- [Transactions](#)

Mapping Metadata

The TopLink application metadata model is based on the TopLink project. The project includes descriptors, mappings, and various policies that customize the run-time capabilities. You associate this mapping and configuration information with a particular data source and application by referencing the project from a session.

For more information, see the following:

- ["Creating Project Metadata"](#) on page 2-20
- ["Understanding Projects"](#) on page 20-1
- ["Understanding Descriptors"](#) on page 26-1
- ["Understanding Mappings"](#) on page 33-1

Session Metadata

A session is the primary interface between the client application and TopLink, and represents the connection to the underlying data source.

TopLink offers several different session types (see ["Understanding TopLink Sessions"](#) on page 75-1), each optimized for different design requirements and architectures. The most commonly used session is the server session, a session that clients access on the server through a client session. The server session provides a shared cache and shared connection resources. You define a session with session metadata.

For CMP projects, the TopLink run-time creates and uses a session internally, but your application does not acquire or use this session directly. Depending on the application server you use, you can specify some of the parameters for this internal session.

For more information, see the following:

- ["Creating Session Metadata"](#) on page 2-21
- ["Using the Persistence Layer"](#) on page 2-13

Cache

By default, a TopLink session provides an object-level cache that guarantees object identity and enhances performance by reducing the number of times the application needs to access the data source. TopLink provides a variety of cache options, including locking, refresh, invalidation, isolation, and coordination. Using cache coordination, you can configure TopLink to synchronize changes with other instances of the

deployed application. You configure most cache options at the session level. You can also configure cache options on a per-query basis or on a descriptor to apply to all queries on the reference class.

For more information, see ["Understanding the Cache"](#) on page 90-1.

Queries and Expressions

TopLink provides several object and data query types, and offers flexible options for query selection criteria, including the following:

- TopLink expressions
- EJB Query Language (QL)
- SQL
- Stored procedures
- Query by example

With these options, you can build any type of query. Oracle recommends using predefined queries to define application queries. Predefined queries are held in the project metadata and referenced by name. This simplifies application development and encapsulates the queries to reduce maintenance costs.

When using EJB entity beans, you can code finders completely using EJB QL (in addition to any of the other TopLink query options), enabling the application to comply with the J2EE specification.

Regardless of the architecture or persistent entity type, you are free to use any of the query options. TopLink Workbench provides the simplest way to define queries. Alternatively, you can build queries in code, using the TopLink API.

For more information, see ["Understanding TopLink Queries"](#) on page 96-1 and ["Understanding TopLink Expressions"](#) on page 97-1.

Transactions

TopLink provides the ability to write transactional code isolated from the underlying database and schema by using a **unit of work**, a specific transactional session.

The unit of work isolates changes in a transaction from other threads until it successfully commits the changes to the database. Unlike other transaction mechanisms, the unit of work automatically manages changes to the objects in the transaction, the order of the changes, and changes that might invalidate other TopLink caches. The unit of work manages these issues by calculating a minimal change set, ordering the database calls to comply with referential integrity rules and deadlock avoidance, and merging changed objects into the shared cache. In a clustered environment, the unit of work also synchronizes changes with the other servers in the coordinated cache.

If an application uses EJB entity beans, developers do not access the unit of work API directly, but they still benefit from its features: the integration between the TopLink runtime and the J2EE container automatically uses the unit of work to the application's best advantage.

For more information, see ["Understanding TopLink Transactions"](#) on page 100-1.

Using the Persistence Layer

At run time, your application uses the TopLink metadata (see ["Understanding TopLink Metadata"](#) on page 2-19).

For a non-CMP project, your application loads a `session.xml` file at run time using the session manager (see ["Acquiring and Using Sessions at Run Time"](#) on page 78-1). The `session.xml` file contains a reference to the mapping metadata `project.xml` file. Using the session, your application accesses the TopLink runtime and the `project.xml` mapping metadata.

For a CMP project, the metadata required is dependent upon the J2EE application server you deploy your application to (see ["Creating TopLink Files for Deployment"](#) on page 8-1). All application servers require an `ejb-jar.xml` and a TopLink project XML file. The session configuration is dependent on the specific J2EE application server.

Deploying the Application

Application packaging (for deployment in the host Java or J2EE environment) influences TopLink use and configuration. For example, developers package a J2EE application in an EAR file. Within the EAR file, there are several ways to package persistent entities within WAR and JAR. How developers configure TopLink depends, in part, on how they package the application and how they use the class loader of the host application server.

This section discusses packaging and deployment from a TopLink perspective. However, if you deploy your application to a J2EE container, you must configure elements of your application to enable TopLink container support.

This section includes the following information:

- [Understanding Deployments](#)
- [TopLink in a J2EE Application](#)

For more information, see [Part III, "Deploying a TopLink Application"](#).

Understanding Deployments

The TopLink approach to deployment involves packaging application files into a single file, such as a JAR file, or an EAR file. This approach lets you create clean and self-contained deployments that do not require significant file management.

After creating these files, deploy the project.

TopLink in a J2EE Application

Although TopLink is an integral part of a J2EE application, in most cases the client does not interact with TopLink directly. Instead, TopLink features are invoked indirectly by way of EJB container callbacks.

As a result, the typical deployment process involves the following steps:

1. Build the project elements, including beans, classes, and data sources.
2. Define the application mappings in TopLink Workbench.
3. Build the application deployment files. Use TopLink Workbench to create the files.
4. Package and deploy the application.
5. Add code to the client application to enable it to access the TopLink application.

Optimizing and Customizing the Application

TopLink provides a diverse set of features to optimize performance including the following:

- Enhancing queries
- Tuning the cache
- Scaling to multiple server configuration

You enable or disable most features in the descriptors or session, making any resulting performance gains global.

Using TopLink EIS (see "[Enterprise Information System \(EIS\) Usage](#)" on page 2-4), you can integrate a TopLink application with legacy data sources using a J2C adapter. This is the most efficient way to customize a TopLink application to accommodate unusual or nonstandard systems.

Using TopLink XML (see "[XML Usage](#)" on page 2-4), you can integrate a TopLink application with legacy data sources using a Web service.

See [Part IV, "Optimizing and Customizing a TopLink Application"](#) for details on optimizing and customizing TopLink.

Troubleshooting the Application

See [Part V, "Troubleshooting a TopLink Application"](#) for detailed information on troubleshooting all aspects of a TopLink application including development and deployment.

Understanding Object Persistence

This section includes a brief description of relational mapping and provides important information and restrictions to guide object and relational modeling. This information is useful when building TopLink applications.

This section includes information on the following:

- [Application Object Model](#)
- [Data Storage Schema](#)
- [Primary Keys and Object Identity](#)
- [Mappings](#)
- [Foreign Keys and Object Relationships](#)
- [Inheritance](#)
- [Concurrency](#)
- [Caching](#)
- [Nonintrusive Persistence](#)
- [Indirection](#)

These sections contain additional detail on these features, and explain how to implement and use them with TopLink.

Application Object Model

Object modeling refers to the design of the Java classes that represent your application objects. With TopLink, you can use your favorite integrated development environment (IDE) or Unified Modeling Language (UML) modeling tool to define and create your application object model.

Any class that registers a descriptor with a TopLink database session is called a persistent class. TopLink does not require that persistent classes provide public accessor methods for any private or protected attributes stored in the database. Refer to ["Persistent Class Requirements"](#) on page 2-11 for more information.

Data Storage Schema

Your data storage schema refers to the design that you implement to organize the persistent data in your application. This schema refers to the data itself—not the actual data source (such as a relational database or nonrelational legacy system).

During the design phase of the TopLink application development process (see ["Typical Development Stages"](#) on page 2-1), you should decide how to implement the classes in the data source. When integrating existing data source information, you must determine how the classes relate to the existing data. If no legacy information exists to integrate, decide how you will store each class, then create the necessary schema.

You can also use TopLink Workbench (see [Chapter 4](#)) or database schema manager (see [Chapter 6](#)) to create the necessary information.

Primary Keys and Object Identity

When making objects persistent, each object requires an *identity* to uniquely identify it for storage and retrieval. Object identity is typically implemented using a unique primary key. This key is used internally by TopLink to identify each object, and to create and manage references. Violating object identity can corrupt the object model.

In a Java application, object identity is preserved if each object in memory is represented by one, and only one, object instance. Multiple retrievals of the same object return references to the same object instance—not multiple copies of the same object.

TopLink supports multiple identity maps to maintain object identity (including composite primary keys). Refer to ["Cache Type and Object Identity"](#) on page 90-3 for additional information.

Mappings

TopLink uses the metadata produced by TopLink Workbench (see ["Understanding TopLink Metadata"](#) on page 2-19) to describe how objects and beans map to the data source. This approach isolates persistence information from the object model—developers are free to design their ideal object model, and DBAs are free to design their ideal schema.

Developers use TopLink Workbench to create and manage the mapping information. At run time, TopLink uses the metadata to seamlessly and dynamically interact with the data source, as required by the application.

TopLink provides an extensive mapping hierarchy that supports the wide variety of data types and references that an object model might contain. For more information, see ["Understanding Mappings"](#) on page 33-1.

Foreign Keys and Object Relationships

A **foreign key** is a combination of columns that reference a unique key, usually the primary key, in another table. Foreign keys can be any number of fields (similar to primary key), all of which are treated as a unit. A foreign key and the primary parent key it references must have the same number and type of fields.

Foreign keys represents relationships from a column or columns in one table to a column or columns in another table. For example, if every `Employee` has an attribute `address` that contains an instance of `Address` (which has its own descriptor and table), the one-to-one mapping for the `address` attribute would specify foreign key information to find an address for a particular `Employee`.

Refer to ["Configuring Table and Field References \(Foreign and Target Foreign Keys\)"](#) on page 37-8 for more information.

Inheritance

Object-oriented systems allow classes to be defined in terms of other classes. For example: motorcycles, sedans, and vans are all *kinds of vehicles*. Each of the vehicle types is a *subclass* of the `Vehicle` class. Similarly, the `Vehicle` class is the *superclass* of each specific vehicle type. Each subclass inherits attributes and methods from its superclass (in addition to having its own attributes and methods).

Inheritance provides several application benefits, including the following:

- Using subclasses to provide specialized behaviors from the basis of common elements provided by the superclass. By using inheritance, you can reuse the code in the superclass many times.
- Implementing *abstract* superclasses that define generic behaviors. This abstract superclass may define and partially implement behavior, while allowing you to complete the details with specialized subclasses.

Refer to ["Configuring Inheritance for a Child \(Branch or Leaf\) Class Descriptor"](#) on page 28-49 and ["Configuring Inherited Attribute Mapping in a Subclass"](#) on page 28-56 for detailed information on using inheritance with TopLink.

Concurrency

To have concurrent clients logged in at the same time, the server must spawn a dedicated thread of execution for each client. J2EE application servers do this automatically. Dedicated threads enable each client to work without having to wait for the completion of other clients. TopLink ensures that these threads do not interfere with each other when they make changes to the identity map or perform database transactions.

Using the TopLink `UnitOfWork` class, your client can make transactional changes in an isolated and thread safe manner. The unit of work manages clones for the objects you modify to isolate each client's work from other concurrent clients and threads. The unit of work is essentially an object-level transaction mechanism that maintains all of the ACID (Atomicity, Consistency, Isolation, Durability) transaction principles as a database transaction. For more information on the unit of work, see ["Understanding TopLink Transactions"](#) on page 100-1.

TopLink supports configurable optimistic and pessimistic locking strategies to let you customize the type of locking that the TopLink concurrency manager uses. For more information, see ["Understanding Descriptors and Locking"](#) on page 26-17.

Caching

TopLink caching improves application performance by automatically storing data returned as objects from the database for future use. This caching provides several advantages:

- Reusing Java objects that have been previously read from the database minimizes database access
- Minimizing SQL calls to the database when objects already exist in the cache
- Minimizing network access to the database
- Setting caching policies a class-by-class and bean-by-bean basis
- Basing caching options and behavior on Java garbage collection

TopLink supports several caching policies to provide extensive flexibility. Developers can fine-tune the cache for maximum performance, based on individual application performance. Refer to [Part XVIII, "Cache"](#) for complete information.

Nonintrusive Persistence

The TopLink nonintrusive approach of achieving persistence through a metadata architecture (see ["Understanding TopLink Metadata"](#) on page 2-19) means that there are almost no object model intrusions.

To persist Java objects, TopLink does not require any of the following:

- Persistent superclass or implementation of persistent interfaces
- Store, delete, or load methods required in the object model
- Special persistence methods
- Generating source code into or wrapping the object model

When using CMP entity beans, TopLink does not require any additional intrusion to the object model, other than the CMP specification requirements.

See ["Building and Using the Persistence Layer"](#) on page 2-11 for additional information on this nonintrusive approach.

Indirection

An indirection object takes the place of an application object so the application object is not read from the database until it is needed. Using indirection allows TopLink to create *stand-ins* for related objects. This results in significant performance improvements, especially when the application requires the contents of only the retrieved object rather than all related objects.

Without indirection, each time the application retrieves a persistent object, it also retrieves *all* the objects referenced by that object. This may result in lower performance for some applications.

Note: Oracle strongly recommends that you use indirection in all situations.

TopLink provides several indirection models, such as proxy indirection, transparent indirection, and value holder indirection. TopLink also provides indirection support for EJB (see ["Indirection and EJB"](#) on page 33-9).

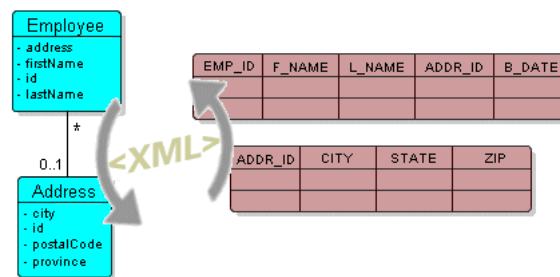
See ["Indirection"](#) on page 33-5 for more information.

Understanding TopLink Metadata

The TopLink metadata is the bridge between the development of an application and its deployed run-time environment. Capture the metadata using TopLink Workbench (see ["Creating Project Metadata"](#) on page 2-20 and ["Creating Session Metadata"](#) on page 2-21), and pass the metadata to the run-time environment using deployment `project.xml` and `sessions.xml` files. You could also manually code these files using Java and the TopLink API, but this approach is more labor-intensive.

The metadata, encapsulated in the `project.xml` file and the `sessions.xml` file, lets developers pass configuration information into the run-time environment. The run-time environment uses the information in conjunction with the persistent entities (Java objects or EJB entity beans) and the code written with the TopLink API, to complete the application.

Figure 2-2 TopLink Metadata



This section describes the following:

- [Advantages of the TopLink Metadata Architecture](#)
- [Creating Project Metadata](#)
- [Creating Session Metadata](#)
- [Deploying Metadata](#)

Advantages of the TopLink Metadata Architecture

The TopLink metadata architecture provides many important benefits, including the following:

- Stores mapping information in XML descriptors—not in the domain model objects
- By using the metadata, TopLink does not intrude in the object model or the database schema
- Allows developers to design the object model as needed, without forcing any specific design
- Allows DBAs to design the database as needed, without forcing any specific design
- Does not rely on code-generation (which can cause serious design, implementation, and maintenance issues)

- Is unintrusive: adapts to the object model and database schema, rather than requiring developers to design their object model or database schema to suit TopLink

Creating Project Metadata

A TopLink project contains the mapping metadata that the TopLink runtime uses to map objects to a data source. The project is the primary object used by the TopLink runtime.

This section describes the principal contents of project metadata, including the following:

- [Descriptors and Mappings](#)
- [Data Source Login Information](#)

For more information about creating `project.xml` metadata, see "[project.xml File](#)" on page 8-2.

Descriptors and Mappings

TopLink maps persistent entities to the database in the application, using the descriptors and mappings developers build with TopLink Workbench. TopLink Workbench supports several approaches to project development, including the following:

- Importing classes and tables for mapping
- Importing classes and generating tables and mappings
- Importing tables and generating classes and mappings
- Creating both class and table definitions

TopLink Workbench supports all these options. The most common solution is to develop the persistent entities using a development tool, such as an integrated development environment (IDE) like Oracle JDeveloper, or a modeling tool, and to develop the relational model through appropriate relational design tools. Developers then use TopLink Workbench to construct mappings that relate these two models.

Although TopLink Workbench does offer the ability to generate persistent entities or the relational model components for an application, these utilities are intended only to assist in rapid initial development strategies—not complete round-trip application development.

For more information, see "[Understanding Descriptors](#)" on page 26-1 and "[Understanding Mappings](#)" on page 33-1.

Amending Descriptors An amendment method lets you implement a TopLink feature that is not currently supported by TopLink Workbench. Simply write a Java method to amend the descriptor after it is loaded, and specify the method in TopLink Workbench for inclusion in the project metadata. See "[Configuring Amendment Methods](#)" on page 28-78 for detailed information on implementing an amendment method for a TopLink descriptor.

Data Source Login Information

For non-CMP projects, you configure a session login in the session metadata that specifies the information required to access the data source (see "[Creating Session Metadata](#)" on page 2-21).

For CMP projects, the project contains a deployment login that specifies the information required to access the data source.

For more information, see ["Projects and Login"](#) on page 20-3.

Creating Session Metadata

A TopLink session contains a reference to a particular `project.xml` file plus the information required to access the data source. The session is the primary object used by your application to access the features of the TopLink runtime.

The agent responsible for creating and accessing session metadata differs depending on whether or not you are creating a CMP project. In a non-CMP project, your application acquires and accesses a session directly (see ["Non-CMP Applications and Session Metadata"](#) on page 8-4). In a CMP project, your application indirectly accesses a session acquired internally by the TopLink runtime (see ["CMP Applications and Session Metadata"](#) on page 8-4).

Deploying Metadata

The `project.xml` and `sessions.xml` file are packaged for deployment differently according to the type of application you are deploying.

For more information, see the following:

- ["Creating TopLink Files for Deployment"](#) on page 8-1
- ["Packaging a TopLink Application"](#) on page 9-1

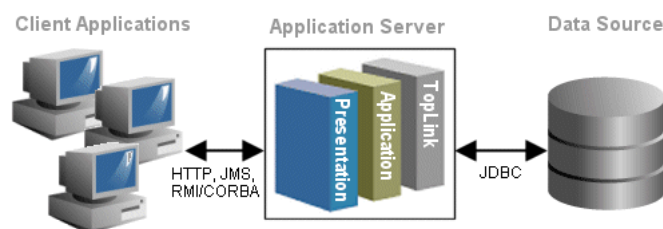
Understanding the Three-Tier Architecture

The three-tier Web application architecture generally includes the connection of a server-side Java application to the database through a JDBC connection (see [Figure 2-3](#)). In this pattern, TopLink resides within a Java server (a J2EE server or a custom server), with several possible server integration points. The application can support Web clients such as servlets, Java clients, and generic clients using XML or Common Object Request Broker Architecture (CORBA).

The three-tier application is a common architecture in which TopLink resides within a Java server (either a J2EE server or a custom server). In this architecture, the server session provides clients with shared access to JDBC connections and a shared object cache. Because it resides on a single JVM, this architecture is simple and easily scalable. The TopLink persistent entities in this architecture are generally Java objects.

This architecture often supports Web-based applications in which the client application is a Web client, a Java client, or a server component.

Figure 2-3 Three Tier Architecture



Although not all three-tier applications are Web-based, this architecture is ideally suited to distributed Web applications. In addition, although it is also common to use EJB in a Web application, this TopLink architecture does not.

Example Implementations

Examples of three-tier architecture implementation include the following:

- A Model-View-Controller Model 2 architectural design pattern that runs in a J2EE container with servlets and JSP that uses TopLink to access data without EJB.
- A Swing or Abstract Window Toolkit (AWT) client that connects to a server-side Java application through RMI, without an application server or container.

Advantages and Disadvantages

The three-tier Web application architecture offers the following advantages:

- High performance, lightweight persistent objects
- High degree of flexibility in deployment platform and configuration

The disadvantage of this architecture is it is less standard than EJB.

Variation Using Remote Sessions

TopLink includes a session type called remote session. The session offers the full session API and contains a cache of its own, but exists on the client system rather than on the TopLink server. Communications can be configured to use RMI or RMI-Internet Inter-Object Request Broker Protocol (IIOP).

Remote session operations require a corresponding client session on the server.

Although this is an excellent option for developers who wish to simplify their access from the client tier to the server tier, it is less scalable than using a client session and does not easily allow changes to server-side behavior.

For more information, see "[Remote Sessions](#)" on page 75-29.

Technical Challenges

The three-tier application with a stateless client presents several technical challenges, including the following:

- Transaction management in a Stateless Environment

A common design practice is to delimit client requests within a single unit of work (transactional session). In a stateless environment, this may affect how you design the presentation layer. For example, if a client requires multiple pages to collect information for a transaction, then the presentation layer must retain the information from page to page until the application accumulates the full set of changes or requests. At that point, the presentation layer invokes the unit of work to modify the database.

- Optimistic Locking in a Stateless Environment

In a stateless environment, take care to avoid processing out-of-date (stale) data. A common strategy for avoiding stale data is to implement optimistic locking, and store the optimistic lock values in the object.

This solution requires careful implementation if the stateless application serializes the objects, or sends the contents of the object to the client in an alternative format.

In this case, transport the optimistic lock values to the client in the HTTP contents of an edit page. You must then use the returned values in any write transaction to ensure that the data did not change while the client was performing its work.

For more information about locking, see ["Configuring Locking Policy"](#) on page 28-62.

- External JDBC Pools

By default, TopLink manages its own connection pools. You can also configure TopLink to use connection pooling offered by the host application server. This feature is useful for shared connection pools and is required for JTA/JTS integration (see ["Configuring External Connection Pooling"](#) on page 85-2).

- JTA/JTS Integration

JTA and JTS are standard Java components that enable sessions to participate in distributed transactions. You must configure TopLink to use JTA/JTS to use session beans in the architecture (see ["Integrating the Unit of Work With an External Transaction Service"](#) on page 102-20).

- Cache Coordination

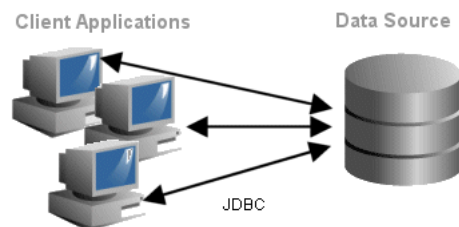
If you choose to use multiple servers to scale your application, you may require TopLink cache coordination (see ["Understanding the Cache"](#) on page 90-1).

Understanding the Two-Tier Architecture

A two-tier application generally includes a Java client that connects directly to the database through TopLink. The two-tier architecture is most common in complex user interfaces with limited deployment. The database session provides TopLink support for two-tier applications.

For more information, see [Chapter 75, "Understanding TopLink Sessions"](#).

Figure 2-4 Two-Tier Architecture



Although the two-tier architecture is the simplest TopLink application pattern, it is also the most restrictive, because each client application requires its own session. As a result, two-tier applications do not scale as easily as other architectures.

Two-tier applications are often implemented as user interfaces that directly access the database (see [Figure 2-4](#)). They can also be non-interface processing engines. In either case, the two-tier model is not as common as the three-tier model.

The following are key elements of an efficient two-tier (client-server) architecture with TopLink:

- Minimal dedicated connections from the client to the database
- An isolated object cache

Example Implementations

An example of a two-tier architecture implementation is a Java user interface (Swing/AWT) and batch data processing.

Advantages and Disadvantages

The advantage of the two-tier design is its simplicity. The TopLink database session that builds the two-tier architecture provides all the TopLink features in a single session type, thereby making the two-tier architecture simple to build and use.

The most important limitation of the two-tier architecture is that it is not scalable, because each client requires its own database session.

Technical Challenges

The current trend toward multitiered Web applications makes the two-tier architecture less common in production systems, but no less viable. Because there is no shared cache in a two-tier system, you risk encountering stale data if you run multiple instances of the application. This risk increases as the number of individual database sessions increases.

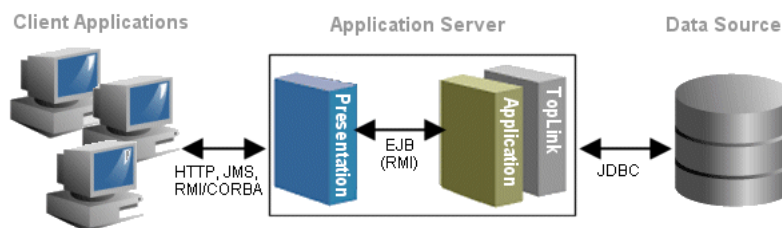
To minimize this problem, TopLink offers support for several data locking strategies. These include pessimistic locking and several variations of optimistic locking. For more information, see "[Configuring Locking Policy](#)" on page 28-62.

Understanding the EJB Session Bean Facade Architecture

This architecture is an extension of the three-tier pattern, with the addition of EJB session beans wrapping the access to the application tier. The EJB session beans provide public API access to application operations, enabling you to separate the presentation tier from the application tier. The architecture also lets you use the EJB session beans within a J2EE container.

This type of architecture generally includes JTA integration, and serialization of data to the client.

Figure 2-5 Three-Tier Architecture Using Session Beans and Java Objects



A common extension to the three-tier architecture is to combine session beans and persistent Java objects managed by TopLink. The resulting application includes session beans and Java objects on a TopLink three-tier architecture (see [Figure 2-5](#)).

The three-tier architecture creates a server session and shares it between the session beans in the application. When a session bean needs to access a TopLink session, the bean obtains a client session from the shared server session. This architecture has the following key features:

- Session beans delimit transactions.

Developers must configure TopLink to work with a JTA system and its associated connection pool.

- Accessing the persistent objects on the client side causes them to be serialized.
Ensure that when the objects re-emerge on the server-side, they properly merge into the cache to maintain identity.

Example Implementation

An example of the EJB session bean facade architecture implementation is a Model-View-Controller Model 2 architectural design pattern that runs in a J2EE container with servlets and JSP and uses the session bean, enabled by TopLink, to access data without EJB.

Advantages and Disadvantages

The EJB session bean facade architecture is a popular and effective compromise between the performance of persistent Java objects, and the benefits of EJB for standardized client development and server scalability. It offers several advantages:

- Less overhead than an EJB entity bean application
TopLink shares access to the project, descriptor, and login information across the beans in the application.
- Future compatibility with other servers
This design isolates login and EJB server-specific information from the beans, which lets you migrate the application from one application server to another without major recoding or rebuilding.
- Shared read cache
This design offers increased efficiency by providing a shared cache for reading objects.

The key disadvantage of this model is the need to transport the persistent model to the client. If the model involves complex object graphs in conjunction with indirection, this can present many challenges with inheritance, indirection, and relationships.

For more information about managing inheritance, indirection and relationships, see [Part X, "Mappings"](#).

Understanding Session Beans

Session beans model a process, operation, or service and as such, are not persistent entities. However, session beans can use persistence mechanisms to perform the services they model.

Under the session bean model, a client application invokes methods on a session bean that, in turn, performs operations on Java objects enabled by TopLink. Session beans execute all operations related to TopLink on behalf of the client.

The EJB specification describes session beans as either stateless or stateful.

Stateful beans maintain a conversational state with a client; that is, they retain information between method calls issued by a particular client. This enables the client to use multiple method calls to manipulate persistent objects.

Stateless beans do not retain data between method calls. When the client interacts with stateless session beans, it must complete any object manipulations within a single method-call.

Technical Challenges

Your application can use both stateful and stateless session beans with a TopLink client session or database session. When you use session beans with a TopLink session, the type of bean used affects how it interacts with the session.

- **Stateless Session Beans and the TopLink Session**

Stateless beans store no information between method calls from the client. As a result, re-establish the connection of the bean to the session for each client method call. Each method call through TopLink obtains a client-session, makes the appropriate calls, and releases the reference to the client-session.

- **Stateful Session Beans and the TopLink Session**

Your EJB Server configuration includes settings that affect the way it manages beans—settings designed to increase performance, limit memory footprint, or set a maximum number of beans. When you use stateful beans, the server may deactivate a stateful session bean enabled by TopLink out of the JVM memory space between calls to satisfy one of these settings. The server then reactivates the bean when required, and brings it back into memory.

This behavior is important, because a TopLink session instance does not survive passivation. To maintain the session between method calls, release the session during the passivation process and re-obtain it when you reactivate the bean.

- **External JDBC Pools**

By default, TopLink manages its own connection pools. For the session bean architecture, you must configure TopLink to use connection pooling offered by the host application server. This feature is useful for shared connection pools and is required for JTA/JTS integration (see ["Configuring External Connection Pooling"](#) on page 85-2).

- **JTA/JTS Integration**

JTA and JTS are standard Java components that enable sessions to participate in distributed transactions. You must configure TopLink to use JTA/JTS to use session beans in the architecture (see ["Integrating the Unit of Work With an External Transaction Service"](#) on page 102-20).

- **Cache Coordination**

If you choose to use multiple servers to scale your application, you may require TopLink cache coordination (see ["Understanding Cache Coordination"](#) on page 90-10).

Unit of Work Merge

You can use a unit of work to enable your client application to modify objects on the database. The unit of work merge functions employ mappings to copy the values from the serialized object into the unit of work, and to calculate changes.

For more information, see ["Merging Changes in Working Copy Clones"](#) on page 102-12.

Understanding the EJB Entity Beans with CMP Architecture

CMP is the part of the J2EE component model that provides an object persistence service that an EJB container uses to persist entity beans. CMP provides distributed, transactional, secure access to persistent data, with a guaranteed portable interface.

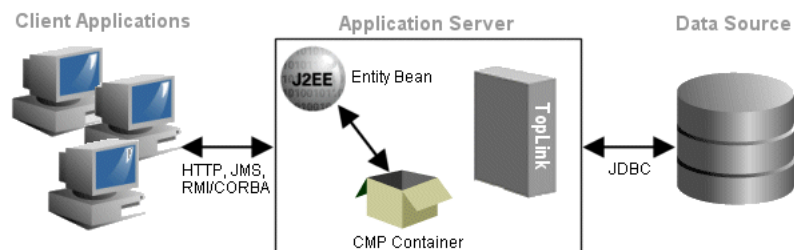
This architecture is an extension of the three-tier architecture, in which the implementation of persistence methods is handled by the container at runtime. As a bean provider, you only need to specify in a deployment descriptor (or, in EJB 3.0, an annotation), those persistent fields and relationships for which the container must handle data access and, optionally, an abstract representation of the database schema.

TopLink CMP is an extension of the TopLink persistence framework that provides custom integration to the CMP containers of various application servers (see ["Application Server Support"](#) on page 7-1). For more information about choosing an application server, see ["Understanding Target Platforms"](#) on page 2-4. TopLink integrates with the CMP container in this architecture to augment (or, in the case of OC4J, become) the container's persistence manager. In this release, when using OC4J and Java 1.5, TopLink supports a subset of the CMP features anticipated in the final EJB 3.0 specification. For more information on EJB 3.0 support, see *Oracle Containers for J2EE Enterprise JavaBeans Developer's Guide*.

Note: EJB 3.0 feature support is subject to change and dependent upon the contents of the final specification.

TopLink CMP integration is nonintrusive (see [Figure 2-6](#)). Through a combination of run-time integration and code generation, the container uses TopLink internally and the bean user interacts with CMP entity beans according to their standard API. This lets you combine the standard interfaces and power of CMP and a container, with TopLink flexibility, performance, and productivity.

Figure 2-6 Three-Tier CMP Architecture



For more information, see the following:

- ["Integrating TopLink With an Application Server"](#) on page 7-1
- ["Creating TopLink Files for Deployment"](#) on page 8-1
- ["Packaging a TopLink Application"](#) on page 9-1
- ["Deploying a TopLink Application"](#) on page 10-1
- ["Configuring Persistence Type"](#) on page 22-5

Example Implementation

An example of the EJB entity beans with CMP implementation is a Model-View-Controller Model 2 architectural design pattern that runs in a J2EE container, with servlets and JSP that access either session beans or EJB 2.0-compliant CMP entity beans enhanced by TopLink.

Advantages and Disadvantages

A three-tier architecture using EJB entity beans with CMP offers the following advantages:

- It allows for CMP beans with sophisticated TopLink features such as caching and mapping support, storing bean data across more than one table, composite primary keys, and data conversion.
- It presents a standard method to access data, which lets developers create standardized, reusable business objects.
- It is well-suited to create coarse-grained objects, which TopLink relates to dependent, lightweight, regular Java objects (TopLink can also manage CMP relationships to light-weight dependent Java objects).
- TopLink provides for lazy initialization of referenced objects and beans (see ["Indirection"](#) on page 33-5).
- TopLink provides functionality for transactional copies of beans, allowing concurrent access by several clients, rather than relying on individual serialization.
- TopLink provides advanced query capabilities, as well as dynamic querying, including the ability to define queries at the bean-level rather than the data source level and to use a rich set of querying and finder options.
- TopLink maintains bean and object identity.

The disadvantage of this architecture is that pure CMP entity bean architectures can impose a high overhead cost. This is especially true when a data model has a large number of fine-grained classes with complex relationships.

Technical Challenges

The key technical challenge in this architecture lies in integrating components into a cohesive system. For example, this architecture requires a specific TopLink integration with the application server or J2EE container.

Other issues include the following:

- [External JDBC Pools](#)
- [JTA/JTS Integration](#)
- [Cache Coordination](#)
- [Maintaining Bidirectional Relationships](#)
- [Managing Dependent Objects in EJB 1.1](#)
- [Managing Dependent Objects in EJB 2.0](#)
- [Managing Collections of EJBObjects in EJB 1.1](#)
- [Managing Collections of EJBObjects in EJB 2.0](#)

External JDBC Pools

By default, TopLink manages its own connection pools. You can also configure TopLink to use connection pooling offered by the host application server. This feature is useful for shared connection pools and is required for JTA/JTS integration (see ["Configuring External Connection Pooling"](#) on page 85-2).

JTA/JTS Integration

JTA and JTS are standard Java components that enable sessions to participate in distributed transactions. You must configure TopLink to use JTA/JTS to use session beans in the architecture (see ["Integrating the Unit of Work With an External Transaction Service"](#) on page 102-20).

Cache Coordination

If you choose to use multiple servers to scale your application, you may require TopLink cache coordination (see ["Understanding Cache Coordination"](#) on page 90-10).

Maintaining Bidirectional Relationships

When one-to-one or many-to-many relationship is bidirectional, you must maintain the back pointers as the relationships change.

When the relationship is between two EJB 2.0 entity beans, TopLink automatically maintains the relationship.

When the relationship is between two EJB 1.1 entity beans or between an entity bean and a regular Java object, you must maintain the back pointers manually.

To set the back pointer manually, do one of the following:

- Code the entity bean to maintain the back pointer when the relationship is established or modified (recommended).
- Code the client to explicitly set the back pointer.

If you code the entity bean to set back pointers, the client is freed of this responsibility. This has the advantage of encapsulating this maintenance implementation in the bean.

In a one-to-many relationship, a source bean might have several dependent target objects. For example, an `EmployeeBean` might own several dependent `PhoneNumber` instances. When you add a new dependent object (a `PhoneNumber`, in this example) to an employee, you must set the `PhoneNumber` instance's back pointer to its owner (the employee). Maintaining a one-to-many relationship in the entity bean involves getting the local object reference from the context of the `EmployeeBean`, and then updating the back pointer as [Example 2-1](#) shows.

Example 2-1 Setting the Back-Pointer in the Entity Bean

```
// obtain owner and phoneNumber
owner = empHome.findByPrimaryKey(ownerId);
phoneNumber = new PhoneNumber("cell", "613", "5551212");
// add phoneNumber to the phoneNumbers of the owner
owner.addPhoneNumber(phoneNumber);

// Maintain the relationship in the Employee's addPhoneNumber method
public void addPhoneNumber(PhoneNumber newPhoneNumber) {
    // get, then set the back pointer to the owner
    Employee owner = (Employee)this.getEntityContext().getEJBLocalObject();
    newPhoneNumber.setOwner(owner);
    // add new phone
    getPhoneNumbers().add(newPhoneNumber);
}
```


For more information, see the following:

- [Configuring Bidirectional Relationship](#)
- [Directionality](#)

Managing Dependent Objects in EJB 1.1

The EJB 1.1 specification recommends that you model entity beans so that all dependent objects are regular Java objects and not other entity beans. If you expose a dependent or privately owned object to the client application, it must be serializable (that is, it implements the `java.io.Serializable` interface) so that it can be sent over to the client and back to the server.

Because entity beans are remote objects, they are referenced remotely in a pass-by-reference fashion. When an entity bean is returned to the client, a remote reference to the bean is returned.

Unlike entity beans, regular Java objects are not remote objects. As a result, when regular Java objects are referenced remotely, they are passed by value (rather than by reference) and serialized (copied) from the remote machine on which they originally resided.

One of the effects of serializing regular Java objects between servers and clients is a loss of object identity due to the copying semantics inherent in serialization. When you serialize a dependent object from the server to the client and then back, two objects with the same primary key but different object identities exist in the server cache. These objects must be merged to avoid exceptions.

If relationships exist between entity beans and Java objects, and these objects are serialized back and forth between the client and server, consider the following:

- [Merging Dependent Objects With the Session Accessor](#)
- [Merging Dependent Objects Without the Session Accessor](#)

Merging Dependent Objects With the Session Accessor Use the `oracle.toplink.ejb.cmp.SessionAccessor` to perform the merge for you within your bean class's setter methods that take regular Java objects as their arguments.

The `SessionAccessor` provides the following static methods you can use to merge dependent objects:

- `registerOrMergeObject`—Merge the changes from the remote clone into the server version of the object for a bean's attribute. The object should not be a collection.
- `registerOrMergeAttribute`—Merge the changes from the remote clone into the server version of the object for a bean's attribute. The object may be single object or a collection.

[Example 2-2](#) shows how to use the `registerOrMergeObject` method for a noncollection attribute.

Example 2-2 Using registerOrMergeObject for a Noncollection Attribute

```
public void setAddress(Address address) {
    this.address = (Address)SessionAccessor.registerOrMergeObject(
        address,
        this.ctx
    );
}
```



```
}
```

[Example 2-3](#) shows how to use the `registerOrMergeAttribute` method for a noncollection attribute.

Example 2-3 Using `registerOrMergeAttribute` for a Non-Collection Attribute

```
public void setAddress(Address address) {
    this.address = (Address) SessionAccessor.registerOrMergeAttribute(
        address,
        "address",
        this.ctx
    );
}
```

[Example 2-4](#) shows how to use the `registerOrMergeAttribute` method for a collection.

Example 2-4 Using `registerOrMergeAttribute` for a Collection Attribute

```
public void setPhones(List phones) {
    this.phones = (List)SessionAccessor.registerOrMergeAttribute(
        phones,
        "phones",
        this.ctx
    );
    // Additional logic to set back-pointers on the phones
}
```

You may have to perform the code merge in methods that add elements to a collection. In [Example 2-5](#), because there is a risk that a phone with the same primary key can be added twice, your code must merge. If the elements in the collection can only be added once, then merging code is not required.

Example 2-5 Using `registerOrMergeObject` for a Method that Adds to a Collection

```
/* The old version of this phone number is removed from the collection. It is
   assumed that equals() returns true for phones with the same primary key value.
   If this is not true, you must iterate through the phones to see if a phone with
   the same primary key already exists in the collection */
public void addPhoneNumber(PhoneNumber phone) {
    phone.setOwner((Employee)this.ctx.getEJBObject());
    // merge new phone
    PhoneNumber serverSidePhone = (PhoneNumber)SessionAccessor.registerOrMergeObject(
        phone,
        this.ctx
    );
    getPhoneNumbers().addElement(serverSidePhone);
}
```

Merging Dependent Objects Without the Session Accessor Alternatively, you can merge the objects yourself by adding merge methods on your regular Java objects and within your set methods, as [Example 2-6](#) shows.

Example 2-6 Manually Merging Dependent Objects

```
public void setAddress(Address address) {
    if (this.address == null){
        this.address = address;
    }
    else {
```

```
        this.address.merge(address);
    }
}
```

If objects can be added to a collection more than once, you must merge them. Merging is not necessary if the objects can be added only once to a collection. Note that merging a collection requires more work. Determine if a copy of each object already exists in the collection, and if so, merge the two copies. If not, you need only add the new object to the collection

Managing Dependent Objects in EJB 2.0

Unlike EJB, TopLink dependent persistent objects can be sent back and forth between a client and the server. When objects are serialized, the risk exists the objects can cause the cache to lose the identity of the objects or attempt to cache duplicate identical objects. To avoid potential problems, use the bean set methods when adding dependent objects to relationship collections as [Example 2-7](#) shows. This enables TopLink to handle merging of objects in the cache.

Example 2-7 Managing Dependent Objects in EJB 2.0

```
addPhoneNumber(PhoneNumber phone) {
    Collection phones = this.getPhoneNumbers();
    Vector newCollection = new Vector();
    newCollection.addAll(phones);
    newCollection.add(phone);
    this.setPhones(newCollection);
}
```

Managing Collections of EJBObjects in EJB 1.1

Collections generally use the `equals` method to compare objects. However, in the case of a Java object that contains a collection of EJB 1.1 entities, the `EJBObject` method `equals` does not yield the desired results. If you manage a collection of entities under EJB 1.1, Oracle recommends the use of the `isIdentical` method to avoid problems.

In addition, the standard collection methods, such as `remove` or `contains`, frequently return unexpected results and so must be avoided.

Several options are available when dealing with collections of `EJBObjects` in EJB 1.1. One option is to create a helper class to assist with collection-type operations. [Example 2-8](#) shows the use of a helper called `EJBCollectionHelper` and [Example 2-9](#) shows a partial implementation of this helper. Alternatively, you can create a special `Collection` class that uses `isIdentical` method instead of `equals` for its comparison operations. To use `isIdentical` method, properly define the `equals` method for the primary key class.

Example 2-8 Using the EJBCollectionHelper in EJB 1.1

```
public void removeOwner(Employee previousOwner){
    EJBCollectionHelper.remove(previousOwner, getOwners());
}
```

Example 2-9 EJBCollectionHelper Implementation

```
...
public static boolean remove(javax.ejb.EJBObject ejbObject, Vector vector) {
    int index = -1;
    index = indexOf(ejbObject, vector);
    // indexOf returns -1 if the element is not found
}
```

```

        if(index == -1){
            return false;
        }
        try{
            vector.removeElementAt(index);
        } catch(ArrayIndexOutOfBoundsException badIndex){
            return false;
        }
        return true;
    }
}

public static int indexOf(javax.ejb.EJBObject ejbObject, Vector vector) {
    Enumeration elements = vector.elements();
    boolean found = false;
    int index = 0;
    javax.ejb.EJBObject current = null;
    while(elements.hasMoreElements()){
        try{
            current = (javax.ejb.EJBObject)
                elements.nextElement();
            if(ejbObject.isIdentical(current)){
                found = true;
                break;
            }
        } catch(ClassCastException wrongTypeOfElement){
            . . .
        } catch (java.rmi.RemoteException otherError){
            . . .
        }
        index++; // increment index counter
    }
    if(found){
        return index;
    } else{
        return -1;
    }
}
...

```

Managing Collections of EJBObjects in EJB 2.0

Collections generally use the `equals` method to compare objects. When using EJB 2.0 this is not a problem in the case of an entity that contains a collection of entities, because the EJB 2.0 container collection handles equality appropriately.

Understanding the EJB Entity Beans With BMP Architecture

BMP is the part of the J2EE component model that lets you, the bean provider, implement the entity bean's persistence directly in the entity bean class or in one or more helper classes that you provide.

This architecture is an extension of the three-tier architecture, in which the persistent data is bean managed within an entity bean using code that you implement. The client code accesses the data through the entity bean interface.

TopLink BMP is an extension of the TopLink persistence framework that provides base class `BMPEntityBase` as a starting point for your BMP development. This class provides an implementation for all methods required by the EJB specification (except `ejbPassivate`). Subclass `BMPEntityBase` to create a TopLink-enabled BMP entity bean.

To use the `BMPEntityBase` class, perform the following:

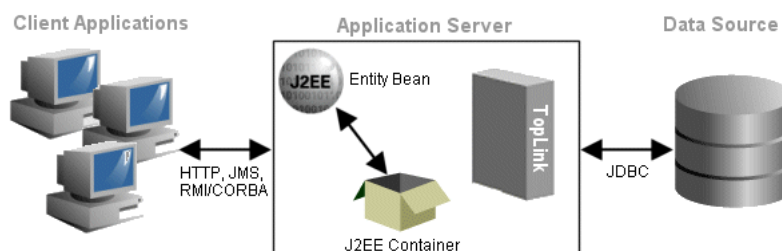
1. Create a TopLink session (see ["Understanding TopLink Sessions"](#) on page 75-1) for your application.
2. Add a `BMPWrapperPolicy` to each descriptor that represents a BMP entity bean.
The `BMPWrapperPolicy` provides TopLink with the information to create remote objects for entity beans and to extract the data out of a remote object.
3. Create the home and remote interfaces.
4. Create deployment descriptors (see ["Integrating TopLink With an Application Server"](#) on page 7-1 and ["Creating TopLink Files for Deployment"](#) on page 8-1).
5. Package your application (see ["Packaging a TopLink Application"](#) on page 9-1).
6. Deploy the beans (see ["Deploying a TopLink Application"](#) on page 10-1).

To make full use of TopLink session and unit of work features, TopLink provides a hook into its functionality through the `BMPDataStore` class. Use this class to translate EJB-required functionality into simple calls.

The `BMPDataStore` class provides implementations of `LOAD` and `STORE`, multiple finders, and `REMOVE` functionality. The `BMPDataStore` class requires a TopLink session. A single instance of `BMPDataStore` must exist for each bean type deployed within a session. When creating a `BMPDataStore`, pass in the session name of the session that the `BMPDataStore` must use to persist the beans and the class of the bean type being persisted. Store the `BMPDataStore` in a global location so that each instance of a bean type uses the correct `store` method.

TopLink BMP support (see [Figure 2-7](#)) lets you combine the standard interfaces of BMP entity beans with TopLink flexibility, performance, and productivity.

Figure 2-7 Three-Tier BMP Architecture



TopLink supports BMP with EJB 1.1 and EJB 2.0. To use BMP support with EJB 2.0, the home interface must inherit from the `oracle.toplink.ejb.EJB20Home`. To make calls to the `BMPEntityBase`, the `findAll` method must call the EJB 2.0 version of the methods. These methods are prefixed with `ejb20`. For example, in the EJB 2.0 version, the `findAll` method appears as `ejb20FindAll`.

To use local beans, use the `oracle.toplink.ejb.EJB20LocalHome` setting instead of the default `oracle.toplink.ejb.EJB20Home`. Instead of the `oracle.toplink.ejb.BMPWrapperPolicy` setting, use the `oracle.toplink.ejb.bmp.BMPLocalWrapperPolicy` setting.

To accommodate both local and remote configurations, ensure the following:

- For a bean that has a single interface, use the corresponding wrapper policy (local or remote) for the descriptor.
- Beans can only participate in relationships as either local or remote interfaces, not both.

For more information, see the following:

- ["Integrating TopLink With an Application Server"](#) on page 7-1
- ["Creating TopLink Files for Deployment"](#) on page 8-1
- ["Packaging a TopLink Application"](#) on page 9-1
- ["Deploying a TopLink Application"](#) on page 10-1
- ["Configuring Persistence Type"](#) on page 22-5
- ["Configuring the Server Platform"](#) on page 77-14

Example Implementations

An example of the EJB entity beans with BMP implementation is a Model-View-Controller Model 2 architectural design pattern that runs in a J2EE container, with servlets and JSP that access session beans and EJB 2.0-compliant BMP entity beans enhanced by TopLink.

Advantages and Disadvantages

Using BMP with a TopLink three-tier architecture offers the following advantages:

- It simplifies the BMP method calls. These can be inherited from an abstract bean class, rather than being generated.
- TopLink makes BMP easier to implement.
- It enables developers to implement database-independent code in the bean methods.
- The architecture supports features such as complex relationships, caching, object-level and dynamic queries, and the unit of work.

The main disadvantages of BMP include the following:

- You must create the persistence mechanisms in the bean code.
- It is not as transparent or efficient as CMP.
- TopLink-only Java object applications offer the same degree of independence from the application server.

Technical Challenges

The key technical challenge in this architecture lies in integrating components into a cohesive system. For example, this architecture requires a specific TopLink integration with the application server or J2EE container.

Other issues include the following:

- [External JDBC Pools](#)
- [JTA/JTS Integration](#)
- [Cache Coordination](#)

External JDBC Pools

By default, TopLink manages its own connection pools. You can also configure TopLink to use connection pooling offered by the host application server. This feature is useful for shared connection pools and is required for JTA/JTS integration (see ["Configuring External Connection Pooling"](#) on page 85-2).

JTA/JTS Integration

JTA and JTS are standard Java components that enable sessions to participate in distributed transactions. You must configure TopLink to use JTA/JTS to use session beans in the architecture (see ["Integrating the Unit of Work With an External Transaction Service"](#) on page 102-20).

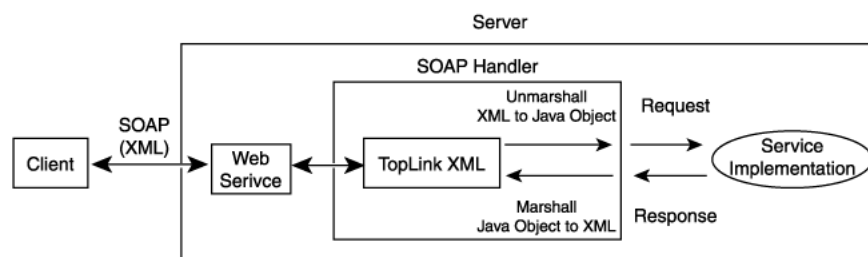
Cache Coordination

If you choose to use multiple servers to scale your application, you may require TopLink cache coordination (see ["Understanding Cache Coordination"](#) on page 90-10).

Understanding the Web Services Architecture

A Web services architecture is similar to the three-tier (see ["Understanding the Three-Tier Architecture"](#) on page 2-21) or session bean (see ["Understanding the EJB Session Bean Facade Architecture"](#) on page 2-24) architecture, however, in a Web services architecture, you encapsulate business logic (the service) in a Web service instead of (or in addition to) using session beans. In a Web services architecture, clients communicate with your application using SOAP messages (XML over HTTP).

Figure 2–8 Web Services Architecture



As in any architecture, you can use TopLink to persist objects to relational or EIS data sources. However, in a Web services architecture, you can also use TopLink to map your object model to an XML schema for use with the Web service or as the Web service XML serializer.

Example Implementations

An example of a Web services architecture implementation is the use of a Web service to expose parts of an existing application to a remote client (typically another application) by way of SOAP messages. In this application, you can use TopLink XML to unmarshall XML messages to Java objects to facilitate requests and marshall Java object responses back into XML for transmission to the client.

Advantages and Disadvantages

Using TopLink in Web services architecture has many advantages, including, but not limited to, the following:

- you can map XML messages to an existing Java object model.
- you can achieve a high level of complexity of mapping support
- compliance with the JAXB standards
- providing a scalable, high-performing solution

One debatable disadvantage is this solution's complexity over a simple RMI session bean service.

Technical Challenges

As with any technology, there are technical challenges associated with the use of TopLink in Web services architecture. These technical challenges are mostly related to special-case scenarios, such as when you need to implement a custom serializer because you have both the Java objects and the schema.

For more information, see the following:

- *Oracle TopLink as a Custom Serializer in a JAX-RPC 1.1 Web service* at <http://www.oracle.com/technology/products/ias/toplink/technical/tips/jaxRpc11/index.htm>
- *Part XIV, "XML Mappings"*

Part II

Using TopLink Development Tools

This part describes the development tools and tool support TopLink provides. It contains the following chapters.

- [Chapter 3, "Understanding TopLink Development Tools"](#)

This chapter describes the development tools and tool support TopLink provides.

- [Chapter 4, "Using TopLink Workbench"](#)

This chapter describes how to use TopLink Workbench including working with databases, generating data from database tables, and creating and editing a `sessions.xml` file.

- [Chapter 5, "Using an Integrated Development Environment"](#)

This chapter explains how to integrate TopLink with an IDE. Detailed instructions are given for the Oracle JDeveloper IDE.

- [Chapter 6, "Using the Schema Manager"](#)

This chapter explains how to use the TopLink schema manager to create databases, tables, stored procedures, and to populate database tables.

Understanding TopLink Development Tools

The TopLink runtime provides Java or J2EE applications with access to persistent entities stored in a data source. In addition to run-time capabilities, the TopLink Foundation Library includes the TopLink Application Programming Interface (API). This API enables applications to access TopLink run-time features.

TopLink includes additional development tools that simplify application development. These tools capture mapping and run-time configuration information in metadata files that TopLink passes to the application at run time.

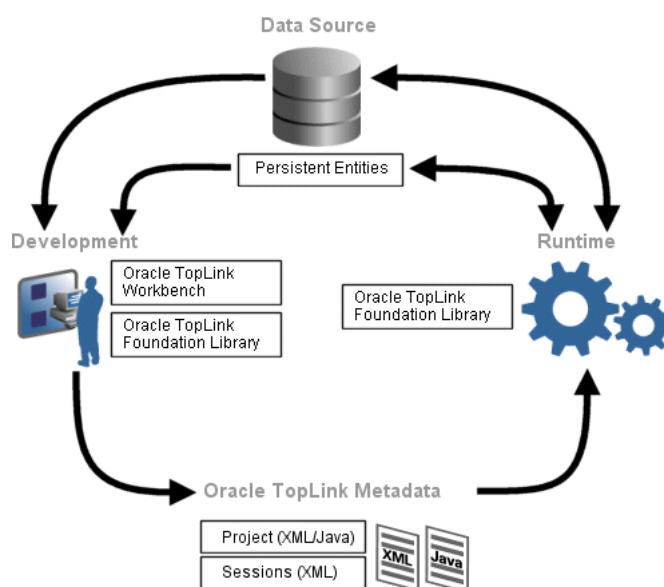
TopLink application development includes the following:

- [Development Environment](#)
- [TopLink Run-Time Environment](#)

TopLink metadata is the link between the two (see "[Understanding TopLink Metadata](#)" on page 2-19).

[Figure 3-1](#) illustrates how these elements interact with the data source.

Figure 3-1 TopLink Components in Development Lifecycle

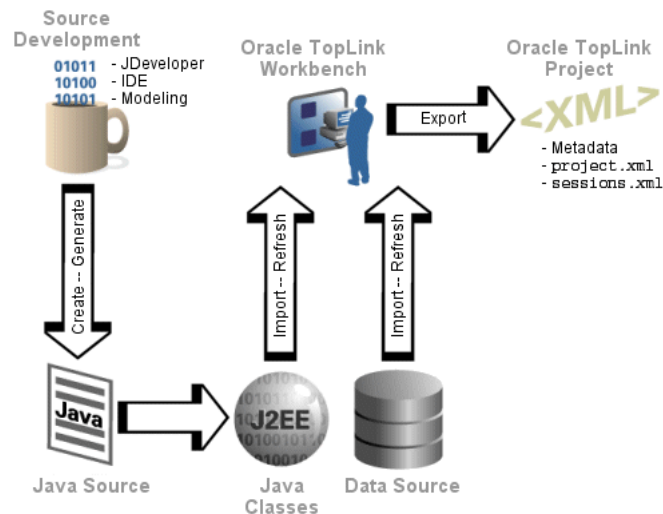


Development Environment

To create a TopLink application, use TopLink Workbench to map objects to data sources using relational and nonrelational models. Capture the resulting mappings and additional run-time configurations in the TopLink project file (`project.xml`) and build a session configuration file (`sessions.xml`). These files together represent your entire TopLink project, as shown in [Figure 3–2](#).

During development, developers can use the TopLink API to define query and transaction logic. When developers use EJB entity beans, there is generally little or no direct use of the TopLink API and there is no session or `sessions.xml` file.

Figure 3–2 TopLink Workbench in Development Environment



TopLink Workbench can import compiled entity classes (Java objects or EJB entity beans), as well as relational or nonrelational schemas through a JDBC driver (configured by the developer). Because TopLink imports the object and relational models for mapping, developers can develop the two models relatively independently from the mapping phase of a project development.

TopLink Run-Time Environment

The TopLink Foundation Library provides the TopLink run-time component. Access the run-time component either directly through the TopLink API or indirectly through a J2EE container when using EJB entity beans. The run-time environment is not a separate or external process—it is embedded within the application. Application calls invoke TopLink to provide persistence behavior. This function allows for transactional and thread-safe access to shared database connections and cached objects.

In addition to J2EE environments, TopLink fully supports non-J2EE environments as well. See ["Selecting an Architecture With TopLink"](#) on page 2-5 for more information.

Using TopLink Workbench

This chapter provides information about understanding, using, and customizing TopLink Workbench.

This chapter includes the following sections:

- [Understanding TopLink Workbench](#)
- [Configuring the TopLink Workbench Environment](#)
- [Working With TopLink Workbench](#)
- [Working With TopLink Workbench Preferences](#)
- [Working With Databases](#)
- [Working With XML Schemas](#)
- [Working With Classes](#)
- [Integrating TopLink Workbench With Apache Ant](#)

For information on using TopLink Workbench to configure sessions XML, refer to [Part XVI, "TopLink Sessions"](#).

Understanding TopLink Workbench

TopLink Workbench is a separate component from the TopLink runtime—it lets you graphically configure descriptors and map your project. TopLink Workbench can verify the descriptor options, access the data source (either a database or an XML schema), and create the database schema. Using TopLink Workbench, you can define TopLink descriptors and configurations *without using code*.

TopLink Workbench can be used during the *development* phase of the development process (see ["Developing Your Application With TopLink"](#) on page 2-1). Typically, this phase includes the following:

1. Defining an object model (a set of Java classes) to describe and solve your problem.
2. Creating a TopLink Workbench project, importing your Java classes and data sources, and using descriptors to describe how the Java classes map to your data source model.
3. Creating a TopLink session and registering your descriptors. In your application, use the session to retrieve and store objects from and to the data source.

TopLink Workbench creates a `<projectName>.mwp` file to store all TopLink project information, including object model, descriptor, and session information.

The `<projectName>.mwp` file is used only by TopLink Workbench. Typically, the only time you need to modify the `<projectName>.mwp` file is to merge changes during application development by a team of developers ("[Merging Files](#)" on page 5-4).

Using TopLink Workbench, you export this information into a `project.xml` file that your TopLink enabled application reads at run time.

For more information on using TopLink Workbench as the development environment, see [Figure 3-2](#) on page 3-2.

Configuring the TopLink Workbench Environment

TopLink Workbench reads its environment variables from the `setenv` script in the `<TOPLINK_HOME>\bin` directory.

Before you launch TopLink Workbench, you must configure its environment as follows:

1. Use a text editor to open the `<TOPLINK_HOME>\bin\setenv` script.
 - For Windows, open the `setenv.cmd` file.
 - For UNIX, open the `setenv.sh` file.
2. Ensure that the `JAVA_HOME` environment variable is set:
 - For Windows: `set JAVA_HOME=C:/j2sdk1.4.2_04`
 - For UNIX: `JAVA_HOME=/usr/local/packages/java; export JAVA_HOME`
3. Update the `DRIVER_CLASSPATH` environment variable to add the location of the following (if necessary):

Note: Do not include any Java classes for your persistent business objects in the `DRIVER_CLASSPATH` variable. Instead, add these persistent business objects in your TopLink Workbench project classpath (see "[Configuring Project Classpath](#)" on page 22-3).

- JDBC drivers - if you are using relational projects (see "[Relational Projects](#)" on page 20-6).
- J2EE Connector Architecture (J2C) adapters - if you are using EIS projects (see "[EIS Projects](#)" on page 20-7).
- `J2C.connector.jar` file - if you are using EIS projects (see "[EIS Projects](#)" on page 20-7).

The `connector.jar` file contains `javax.resource.cci` and `javax.resource.spi` interfaces that TopLink EIS uses. By default, TopLink Workbench updates its classpath to include the Java 1.5 `connector.jar` file from `<TOPLINK_HOME>/j2ee/home/lib`. If this version of the `connector.jar` file is incompatible with your environment, edit the `workbench.cmd` or `workbench.sh` file in `<TOPLINK_HOME>/bin` to change the path to this file.

At runtime, this `connector.jar` file (or its equivalent) must be on your application or application server classpath.

- Oracle Database ORACLE_HOME/rdbms/jlib/xdm.jar file - if you are using direct-to-XMLType mappings with an Oracle9i or higher database (see ["Direct to XMLType Mapping"](#) on page 36-4).
- Custom Collection class that you use to override the default Collection class that TopLink uses with a mapping container policy (see ["Configuring Container Policy"](#) on page 35-26).

[Example 4-1](#) shows how to set the DRIVER_CLASSPATH variable for Windows and [Example 4-2](#) for UNIX.

Example 4-1 Setting DRIVER_CLASSPATH on Windows

```
set DRIVER_
CLASSPATH=C:\OraHome2\jdbc\lib\ojdbc14.jar;C:\Attunity\Connect\Java\lib\attunityResourceAdapt
er.jar;C:\OraHome2\rdbms\jlib\xdm.jar
```

Note: If the path to your driver(s) contains spaces, you must enclose the path in double-quotes in the setenv.cmd file. For example:

```
set DRIVER_CLASSPATH="C:\Program Files\some directory\driver.jar"
```

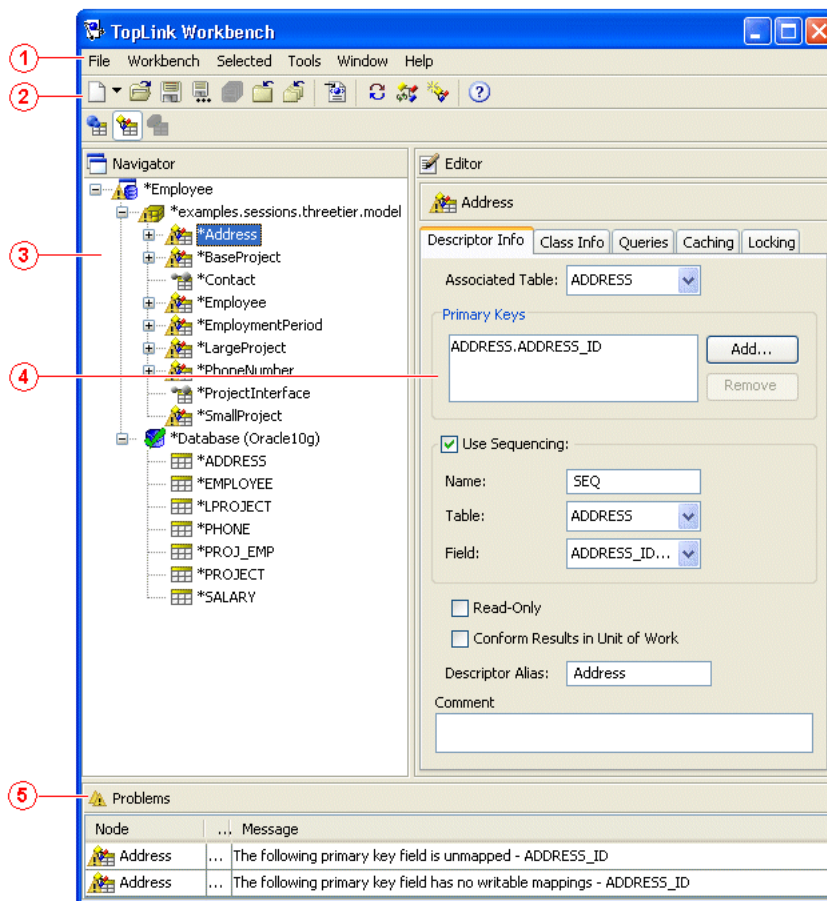
Example 4-2 Setting DRIVER_CLASSPATH on UNIX

```
DRIVER_
CLASSPATH=/OraHome2/jdbc/lib/ojdbc14.jar;/attunity/connect/java/lib/attunityResourceAdapter.j
ar;/OraHome2/rdbms/jlib/xdm.jar; export JDBC_CLASSPATH
```

4. Save and close the setenv script.

Working With TopLink Workbench

[Figure 4-1](#) shows the primary parts of TopLink Workbench window.

Figure 4–1 TopLink Workbench Window

The numbered callouts in [Figure 4–1](#) identify the following user interface components:

1. Menu bar

The menu bar contains menus for each TopLink Workbench function. Some objects also contain context-sensitive menus. See ["Using the Menus"](#) on page 4-5 for more information.

2. Toolbars

The toolbars contain shortcuts to specific functions. See ["Using the Toolbars"](#) on page 4-6 for more information.

3. Navigator window section

The Navigator window section shows the project navigation tree for all open projects (see ["Using the Navigator"](#) on page 4-9). Click the plus (+) or minus (–) sign next to an object (or double-click the object) to expand or collapse the tree. When you select an object in the **Navigator** window section, its properties appear in the **Editor** window.

4. Editor window section

The **Editor** window section contains specific property sheets and option tabs for the currently selected object. See ["Using the Editor"](#) on page 4-10 for more information.

5. Problems window section

The **Problems** window section shows messages and errors for the currently selected object in the Navigator window section (see ["Using the Problems Window"](#) on page 4-11). [Chapter 14, "TopLink Workbench Error Reference"](#) contains detailed information on each error message.

Using the Menus

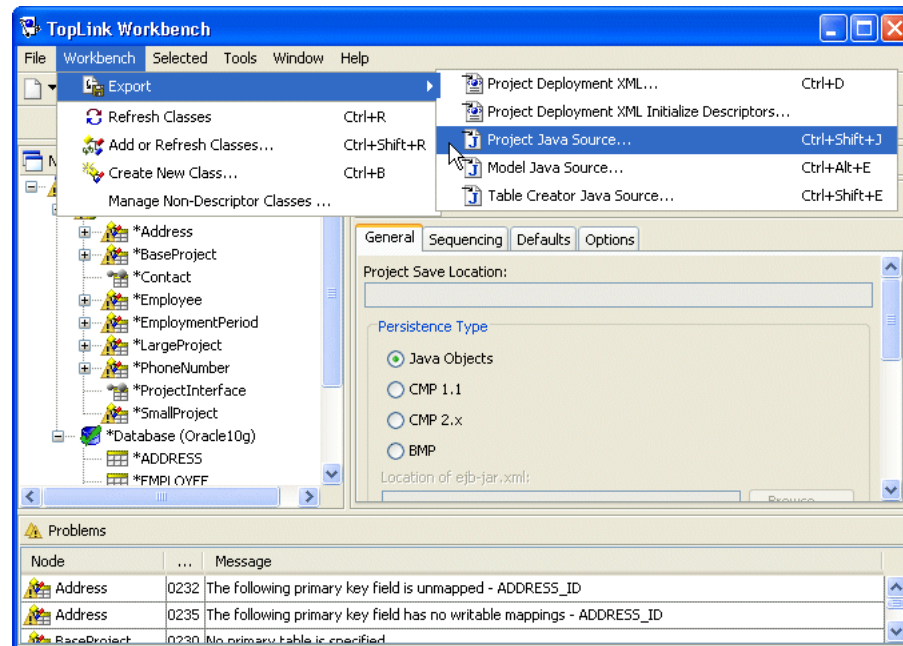
TopLink Workbench contains two types of menus:

- [Menu Bar Menus](#)
- [Context Menus](#)

Menu Bar Menu

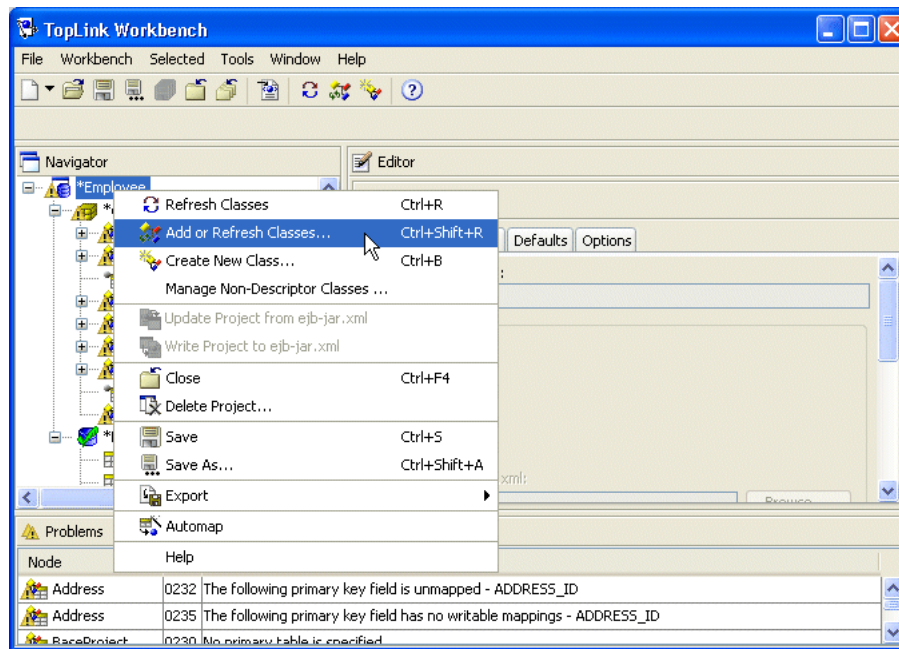
The menu bar, located at the top of the TopLink Workbench window, provides menus for each TopLink Workbench function. Some menus (such as **Selected**) are context-sensitive; the available options may vary, depending on the currently selected object.

Figure 4–2 Sample Menu Bar Menu



Context Menus

When you right-click objects in the **Navigator** window, a context menu appears with functions specific to the selected object.

Figure 4–3 Sample Context Menu

Using the Toolbars

TopLink Workbench contains the following toolbars at the top of the window:

- [Standard Toolbar](#)
- [Context Toolbar](#)

Toolbars provide tool tips: each toolbar button provides a brief description when you position the mouse pointer over it.

Standard Toolbar

The standard toolbar furnishes quick access to the standard menu options (**File**, **Edit**, **Selected**, and so on).

Table 4–1 Standard Toolbar Buttons

Button	Description	Available for ...
	New	All
	Open	All
	Save	All
	Save as	All
	Save all	All
	Close	All
	Close all	All
	Help topics	All

Table 4–1 (Cont.) Standard Toolbar Buttons

Button	Description	Available for ...
	Export deployment XML	Projects
	Refresh classes	Projects
	Add or refresh classes	Projects
	Create new class	Projects

Context Toolbar

The context toolbar provides quick access to functions for the currently selected object in the **Navigator** (see ["Using the Navigator"](#) on page 4-9). The available buttons will vary, depending on which item you have selected.

You can also right-click the item and choose the appropriate option from the context menu.

Table 4–2 Context Toolbar Buttons

Button	Description	Available for ...
	Login to database	Databases
	Logout of database	Databases
	Add new table	Databases
	Add or update tables from database	Databases
	Refresh from database	Database tables
	Remove table	Database tables
	Rename	Database tables
	Add database platform	Database platform repositories
	Rename repository	Database platform repositories
	Delete platform	Database platform repositories
	Clone platform	Database platform repositories
	Add database type	Database platform repositories
	Import schema	Schemas
	Relational aggregate descriptor	Descriptors
	Relational class descriptor	Descriptors
	Relational EJB descriptor	Descriptors
	EIS composite descriptor	Descriptors
	EIS root descriptor	Descriptors

Table 4–2 (Cont.) Context Toolbar Buttons

Button	Description	Available for ...
	EIS EJB descriptor	Descriptors
	XML descriptor	Descriptors
	Direct-to-field mapping	Attributes in relational descriptors
	Object type mapping ¹	Attributes in relational descriptors
	Type conversion mapping ¹	Attributes in relational descriptors
	Serialized mapping ¹	Attributes in relational descriptors
	Direct-to-XMLType mapping	Attributes in relational descriptors
	Direct collection mapping	Attributes in relational descriptors
	Direct map mapping	Attributes in relational descriptors
	Aggregate mapping	Attributes in relational descriptors
	One-to-one mapping	Attributes in relational descriptors
	Variable one-to-one mapping	Attributes in relational descriptors
	One-to-many mapping	Attributes in relational descriptors
	Many-to-many mapping	Attributes in relational descriptors
	Direct mapping	Attributes in EIS descriptors
	Direct collection mapping	Attributes in EIS descriptors
	Composite object mapping	Attributes in EIS descriptors
	Composite collection mapping	Attributes in EIS descriptors
	One-to-one mapping	Attributes in EIS descriptors
	One-to-many mapping	Attributes in EIS descriptors
	Direct-to-XML mapping	Attributes in XML descriptors
	Direct collection mapping	Attributes in XML descriptors
	Composite object mapping	Attributes in XML descriptors
	Composite collection mapping	Attributes in XML descriptors
	Any object mapping	Attributes in XML descriptors
	Any collection mapping	Attributes in XML descriptors
	Transformation mapping	Attributes in all descriptors
	Unmap	Attributes in all descriptors

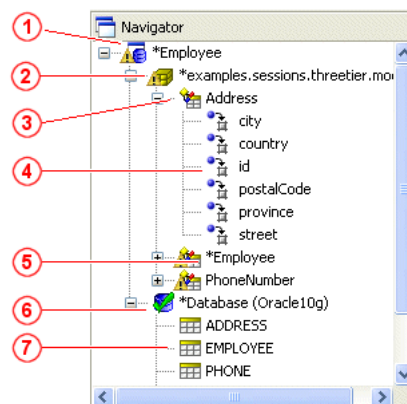
Table 4–2 (Cont.) Context Toolbar Buttons

Button	Description	Available for ...
	Session	Sessions configurations
	Session Broker	Sessions configurations
	Named connection pool	Server sessions
	Sequence connection pool	Server sessions
	Write connection pool	Server sessions
	Rename	Database sessions, session brokers
	Delete session	Database sessions, session brokers

¹ Deprecated. For more information, see ["Using a Converter Mapping"](#) on page 36-3

Using the Navigator

TopLink displays the items included in each project (descriptors, mappings, data source, and so on) in the **Navigator** on the left side of the TopLink Workbench window, as [Figure 4–4](#) shows.

Figure 4–4 Sample Navigator

The numbered callouts on [Figure 4–4](#) identify the following user interface components:

1. Project (relational project)
2. Package
3. TopLink Descriptor (relational descriptor)
4. Attribute/mapping (direct to field mapping)
5. Unsaved/changed item
6. Database
7. Database table

Click the plus (+) or minus (–) sign next to the item, or double-click the item name to expand or collapse the item.

TopLink Workbench identifies items that have been changed but not yet saved by adding an asterisk (*) in front of the item name.

When you select an item in the **Navigator**, its properties appear in the **Editor** (see ["Using the Editor"](#) on page 4-10).

To perform specific functions for an item, select the item in the **Navigator** and do one of the following:

- Right-click on the object and select the function from the context menu (see ["Context Menus"](#) on page 4-5).
- Choose a function from the **Selected** menu (see ["Menu Bar Menus"](#) on page 4-5).

For information on using the Navigator with a database in relational projects, see ["Working With Database Tables in the Navigator Window"](#) on page 4-21.

For information on using the Navigator with an XML schema in EIS projects (using XML records) and XML projects, see ["Working With XML Schemas in the Navigator"](#) on page 4-33.

Active and Inactive Descriptors

Inactive descriptors appear dimmed in the **Navigator**. Inactive descriptors are not registered with the session when the project is loaded into Java. This feature lets you define and test subsets of descriptors. To activate or deactivate a descriptor, right-click the descriptor and select **Activate/Deactivate Descriptor** from the context menu.

Figure 4-5 Sample Active and Inactive Descriptors

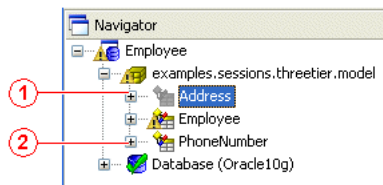


Figure 4-5 numbered callouts show the following user interface components:

1. Inactive descriptor
2. Active descriptor

Errors and Missing Information



If an element in the project (such as a descriptor or mapping) contains an error or some deficiency (sometimes called *neediness*), a warning icon appears beside the element icon in the **Navigator**, and TopLink Workbench displays a message in the Problems window (see ["Using the Problems Window"](#) on page 4-11).

[Chapter 14, "TopLink Workbench Error Reference"](#), contains more information on each TopLink Workbench error message.

Using the Editor

The **Editor**, on the right side of the TopLink Workbench window, displays the property sheet associated with the currently selected item in the **Navigator**, as [Figure 4-6](#) shows.

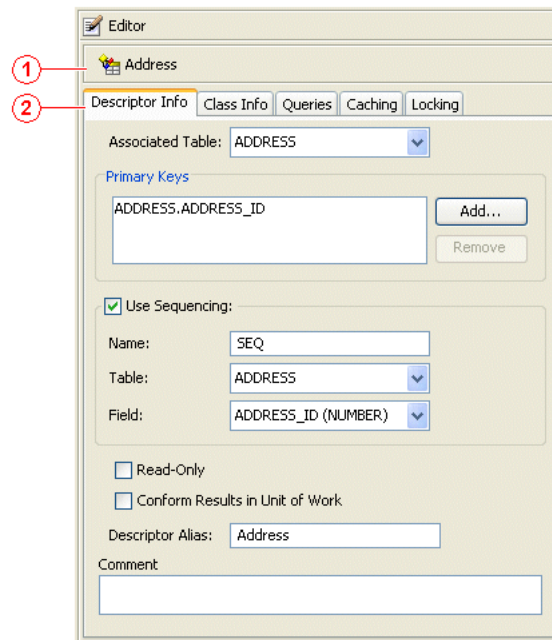
Figure 4–6 Sample Editor

Figure 4–6 numbered callouts identify the following user interface components:

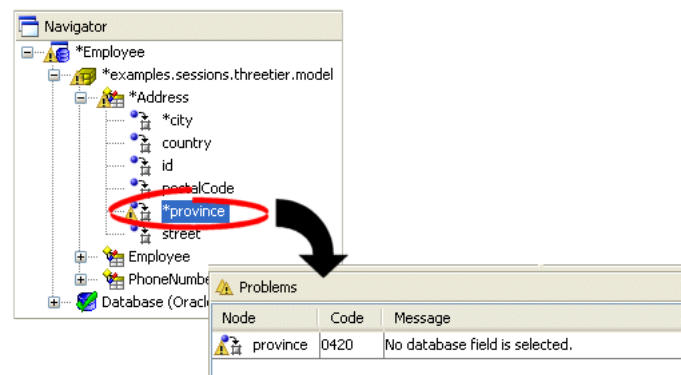
1. Selected element (from the **Navigator**)
2. Editor property tabs

Using the Problems Window



If an element in the project (such as a descriptor or mapping) contains an error or some deficiency (sometimes called *neediness*), the TopLink Workbench displays a caution icon (represented by a yellow triangle with a black exclamation point in the middle) to the left of the deficient element in the **Navigator** (see "Using the Navigator" on page 4-9) and displays a message in the Problems window as Figure 4–7 shows.

If you select the error, then TopLink Workbench displays the complete error message in the **Problems** window. Chapter 14, "TopLink Workbench Error Reference" contains detailed information on each error message.

Figure 4–7 Sample Deficient Mapping

Double-click any error message in the **Problems** window to automatically highlight the specific node in the **Navigator**. To display or hide the **Problems** window, select **Window > Show Problems** from the menu.

You can also create a status report (see ["Generating the Project Status Report"](#) on page 21-12) that includes all errors in a selected project.

[Chapter 14, "TopLink Workbench Error Reference"](#), contains more information on each TopLink Workbench error message.

Using the Online Help

TopLink Workbench contains an extensive online Help system to assist you in developing TopLink applications. You can use the online Help system in a *hosted* or *local* environment (see ["Help Preferences"](#) on page 4-14).

To receive help on any field, tab, or element in TopLink Workbench, right-click the element and select **Help** from the context menu or press F1.

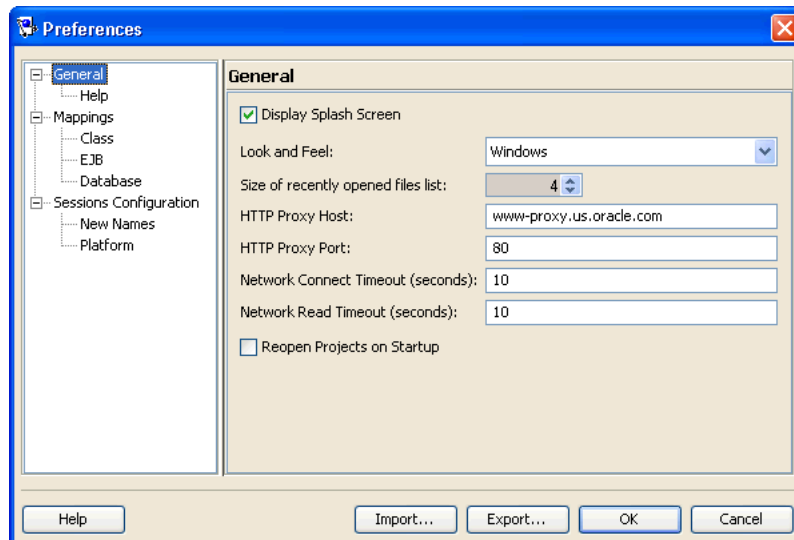


To review the complete TopLink documentation and Quick Start, click **Help**.

Working With TopLink Workbench Preferences

To customize TopLink Workbench, select **Tools > Preferences** from the menu. The **Preferences** dialog box appears.

Figure 4–8 Preferences Dialog Box



TopLink Workbench provides the following preferences:

- [General Preferences](#)
 - [Help Preferences](#)
- [Mappings Preferences](#)
 - [Class Preferences](#)
 - [EJB Preferences](#)
 - [Database Preferences](#)

- [Sessions Configuration Preferences](#)
 - [New Names Preferences](#)
 - [Platform Preferences](#)

Use this dialog box to configure TopLink Workbench preferences. After changing preferences, you must restart TopLink Workbench.

To import your preferences from an existing file, click **Import** and select the file.

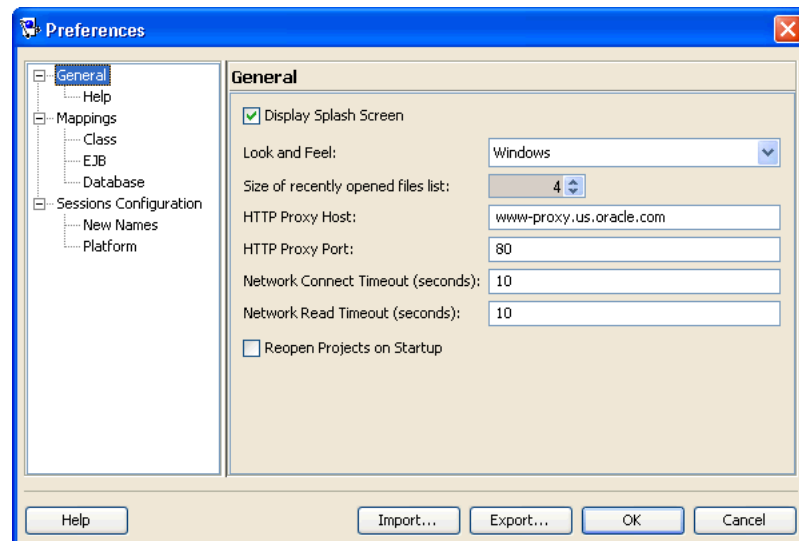
To export your preferences, click **Export** and select a directory location and filename.

General Preferences

Use the General preferences to customize the look and feel (the graphical user interface) of TopLink Workbench as well as to specify any proxy information required to access the Internet (for example, to allow TopLink to access XML schemas and on-line documentation hosted on Internet sites). Follow these steps to customize the General preferences:

1. Select **Tools > Preferences** from the menu. The **Preferences** dialog box appears.
2. Select **General** in the **Category** window.

Figure 4–9 Preferences–General Dialog Box



Use the following information to enter data in each field of the dialog box:

Field	Description
Display Splash Screen	Specify if TopLink Workbench should show the graphical splash screen when starting.
Look and Feel	Select the look and feel to use for TopLink Workbench.
Size of recently opened files list	Select the number of projects to maintain in the File > Reopen option. See " Opening Existing Projects " on page 21-10 for more information.
HTTP Proxy Host	Specify if your PC requires a proxy server to access the internet.
HTTP Proxy Port	Specify the port used by your proxy host.

Field	Description
Network Connect Timeout	Specify the timeout (in seconds) to establish a network or internet connection.
Network Read Timeout	Specify the timeout (in seconds) when accessing data from a network or internet connection.
Reopen Projects on Startup	Select to reopen the projects that were open the last time you exited the TopLink Workbench.

You must restart TopLink Workbench to apply the changes.

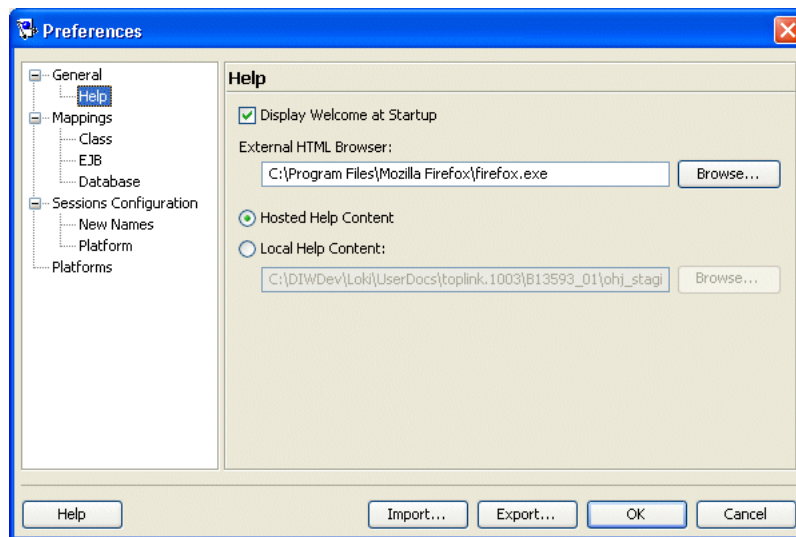
Help Preferences

Use the Help preferences to select the location of the TopLink documentation (in addition to the online Help) and other Help system preferences.

By default, TopLink installs includes only the online Help and release notes. All other documentation (such as *Oracle TopLink Getting Started Guide* and *Oracle TopLink Developer's Guide*) can be accessed from a hosted location.

1. Select **Tools > Preferences** from the menu. The **Preferences** dialog box appears.
2. Expand **General** in the **Category** window and select **Help**. The **Preferences–Help** dialog box appears.

Figure 4–10 Preferences–General–Help Dialog Box



Use the following information to enter data in each field:

Field	Description
Display Welcome at Startup	Specify if TopLink should show the Welcome screen each time you start TopLink Workbench.
External HTML Browser	Click Browse and select the location of your default Web browser. You must specify a Web browser to access the Quick Tour, Javadoc (API), and other Web-based material.

Field	Description
Hosted Help Content ¹	Access the complete TopLink documentation set located on Oracle Technology Network (OTN). Selecting this option requires your PC to have Internet access. Note: Depending on your Internet connection, using Hosted Documentation may impact performance when searching the documentation.
Local Help Content ¹	Access the complete TopLink documentation set from a local location (such as your PC or your company's intranet). Click Browse and select the help JAR file location.

¹ You must restart TopLink Workbench to apply changes to this option.

Note: When using **Hosted Help Content**, ensure that your proxy information, specified in the [General Preferences](#), is correct.

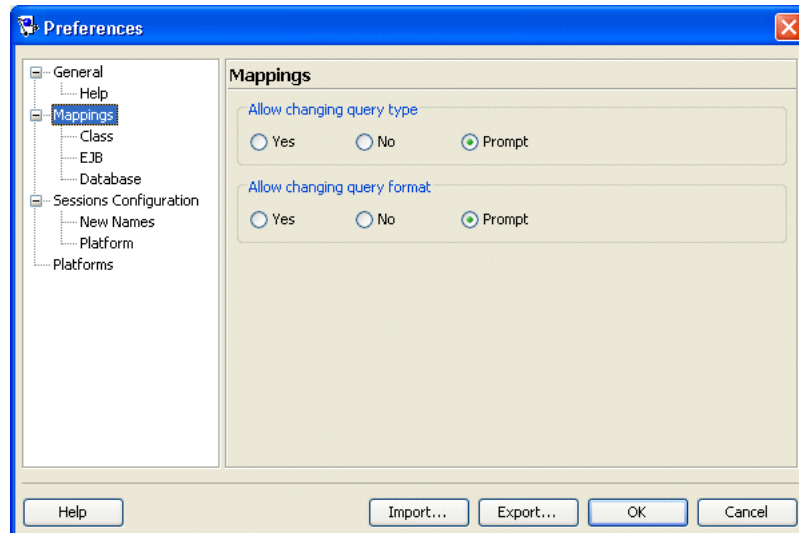
When using **Local Help Content**, download the complete documentation from OTN. See *Oracle TopLink Getting Started Guide* for more information.

Mappings Preferences

Use the Mappings preferences to specify general mapping preferences. Follow these steps to set the Mapping preferences:

1. Select **Tools > Preferences** from the menu. The **Preferences** dialog box appears.
2. Select **Mappings** in the **Category** window. The **Mappings** dialog box appears.

Figure 4-11 Preferences–Mappings Dialog Box



Use the following information to enter data in each field:

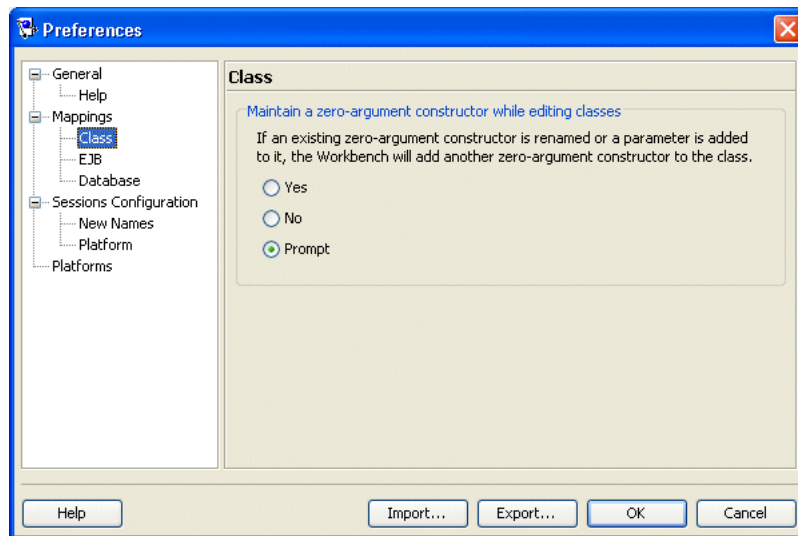
Field	Description
Allow changing query type	Configure whether or not TopLink Workbench always allows, never allows, or prompts before allowing you to change the query type associated with a descriptor.
Allow changing query format	Configure whether or not TopLink Workbench always allows, never allows, or prompts before allowing you to change the configuration of a query associated with a descriptor.

Class Preferences

Use the Class preferences to specify how TopLink Workbench maintains classes when renaming or editing a zero-argument constructor. Follow these steps to set the Class preferences:

1. Select **Tools > Preferences** from the menu. The **Preferences** dialog box appears.
2. Expand **Mappings** in the **Category** window and select **Class**.

Figure 4–12 Preferences – Mappings – Class Dialog Box

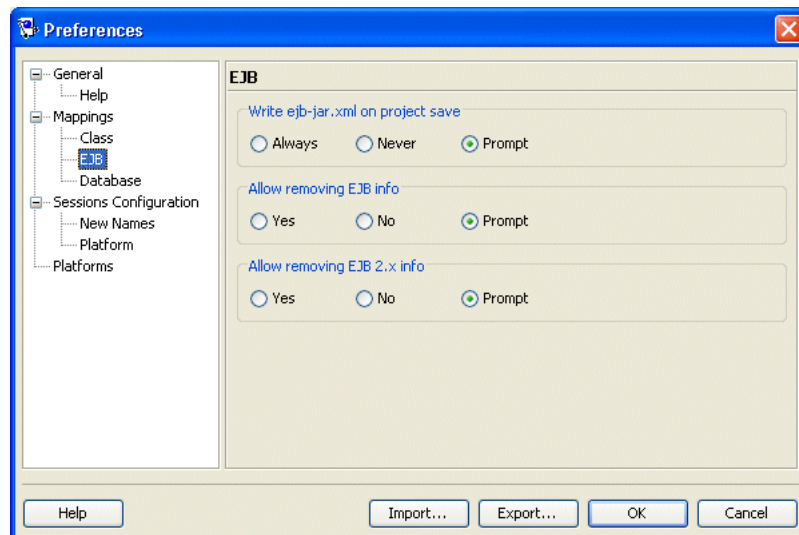


On the **Preferences–Mappings–Class** dialog box, specify how TopLink Workbench maintains classes when renaming or editing a zero-argument constructor.

EJB Preferences

Use the EJB preferences to specify how TopLink Workbench updates the `ejb-jar.xml` file when saving EJB projects. Follow these steps to set the EJB preferences:

1. Select **Tools > Preferences** from the menu. The **Preferences** dialog box appears.
2. Expand **Mappings** in the **Category** window and select **EJB**.

Figure 4-13 Preferences–Mappings–EJB Preferences Dialog Box

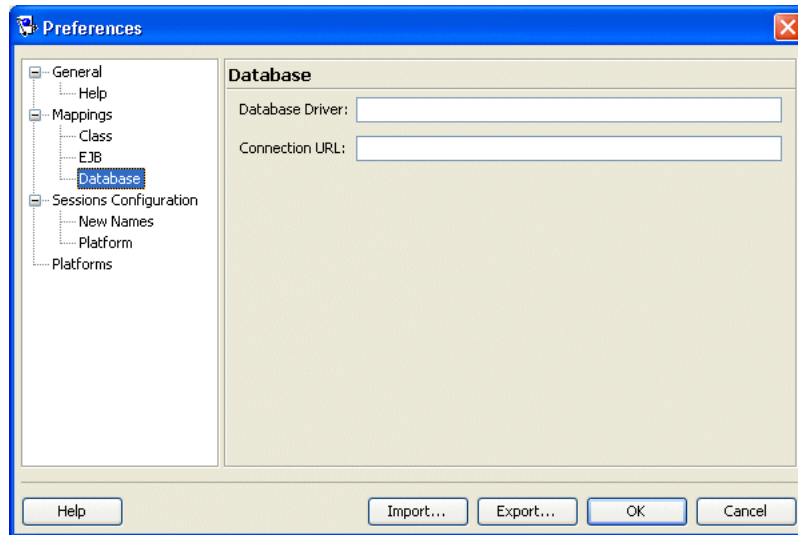
Use the following information to select how TopLink Workbench will update the `ejb-jar.xml` file:

Field	Description
Write ejb-jar.xml on project save	Configure whether or not TopLink Workbench always updates, never updates, or prompts before updating the <code>ejb-jar.xml</code> file each time you save the project.
Allow removing EJB info	Configure whether or not TopLink Workbench always allows, never allows, or prompts before allowing you to remove the EJB information associated with a descriptor. See "Configuring a Descriptor With EJB Information" on page 28-44 for more information.
Allow removing EJB 2.x info	Configure whether or not TopLink Workbench always allows, never allows, or prompts before allowing you to remove the EJB 2.x information associated with a descriptor. See "Configuring a Descriptor With EJB Information" on page 28-44 for more information.

Database Preferences

Use the Database preferences to specify custom database drivers and connection URLs for TopLink Workbench. These drivers and URLs can then be used when defining database logins. Follow these steps to set the Database preferences:

1. Select **Tools > Preferences** from the menu. The **Preferences** dialog box appears.
2. Expand **Mappings** in the **Category** window and select **Database**.

Figure 4-14 Preferences–Mappings–Database Preferences Dialog Box

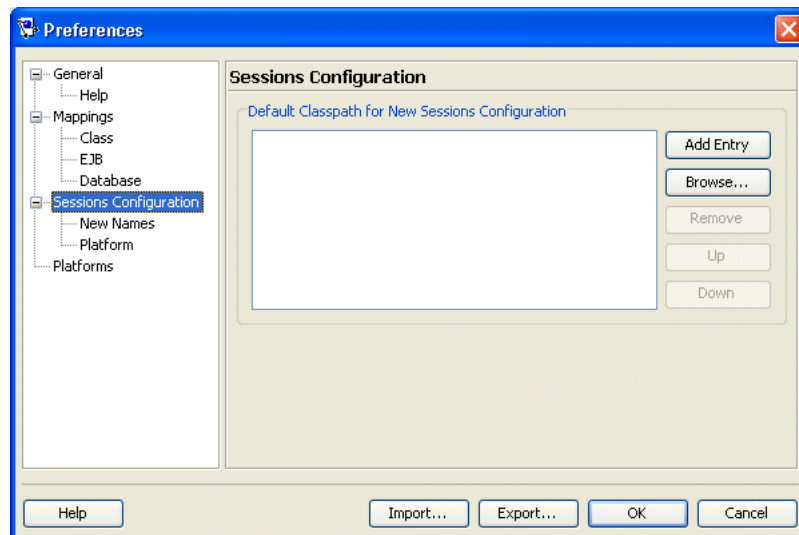
Use the following information to enter data in each field:

Field	Description
Database Driver	Enter the custom database driver class name.
Connection URL	Enter the custom database connection URL.

Sessions Configuration Preferences

Use the Sessions preferences to specify default classpaths to be added to each newly created TopLink sessions configuration for features that require an external Java class (for example, session event listeners). The entries added here will automatically appear on the Sessions Configuration property sheet (see "[Configuring a Sessions Configuration](#)" on page 76-2). Follow these steps to set the Sessions Configuration preferences:

1. Select **Tools > Preferences** from the menu. The **Preferences** dialog box appears.
2. Select **Sessions Configuration** in the **Category** window.

Figure 4–15 Preferences–Sessions Configuration Dialog Box

To add a JAR or ZIP file, click **Add Entry** or **Browse** and add the JAR or ZIP files that contain the default compiled Java classes for this sessions configuration.

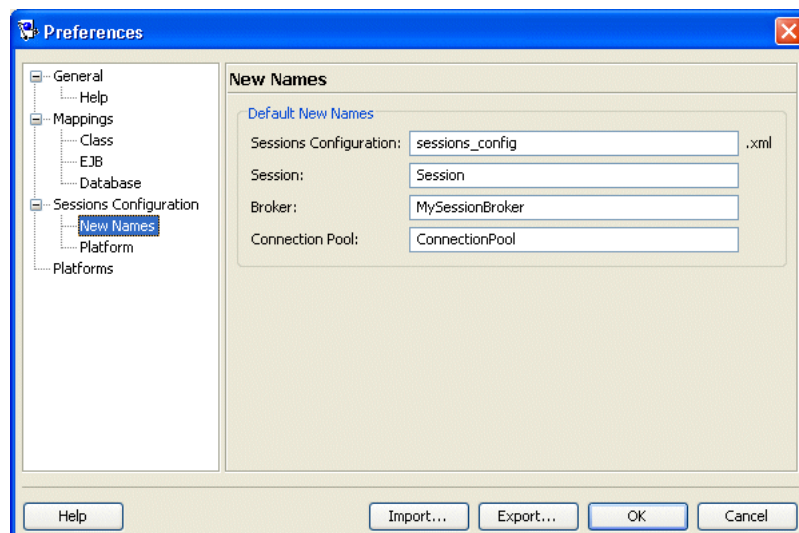
To remove a JAR or ZIP file, select the file and click **Remove**.

To change the order in which TopLink searches these JAR or ZIP files, select a file and click **Up** to move it up, or click **Down** to move it down in the list.

New Names Preferences

Use the New Names preferences to specify the default values and names of newly created sessions, session brokers, and connection pools. Follow these steps to set the New Names preferences:

1. Select **Tools > Preferences** from the menu. The **Preferences** dialog box appears.
2. Expand **Sessions Configuration** in the **Category** window and select **New Names**.

Figure 4–16 Preferences–Sessions Configuration–New Names Dialog Box

Use the following information to enter data in each field:

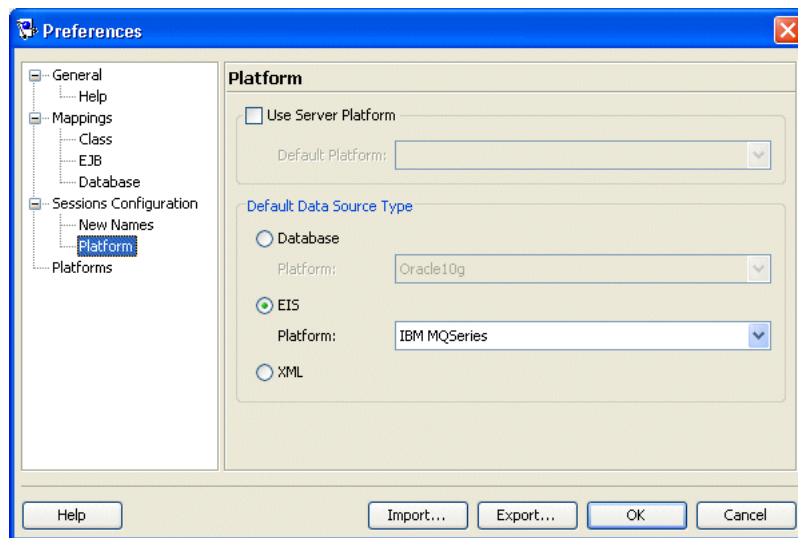
Field	Description
Sessions Configuration	Specify the default name for newly created sessions configuration files (default, <code>sessions.xml</code>). See "Creating a Sessions Configuration" on page 76-1 for more information.
Session	Specify the default name for newly created sessions (default, <code>Session</code>). See "Session Creation Overview" on page 76-1 for more information.
Broker	Specify the default name for newly created session brokers (default, <code>SessionBroker</code>). See "Creating Session Broker and Client Sessions" on page 76-6 for more information.
Connection Pool	Specify the default name for newly created connection pools (default, <code>ConnectionPool</code>). See "Creating an Internal Connection Pool" on page 88-1 for more information.

Platform Preferences

Use the Platform preferences to specify the default data source type for newly created sessions. The type selected here will automatically appear on the Create New Session dialog box. Follow these steps to set the Platform preferences:

1. Select **Tools > Preferences** from the menu. The **Preferences** dialog box appears.
2. Expand **Sessions Configuration** in the **Category** window and select **Platform**.

Figure 4-17 Preferences–Sessions Configuration–Platform Preferences Dialog Box



Use the following information to enter data in each field:

Field	Description
Use Server Platform	Specify the default application server platform for newly created sessions configuration files (default, <code>sessions.xml</code>). See "Creating a Sessions Configuration" on page 76-1 for more information.
Default Data Source Type	Select the default data source type (Database , EIS , or XML) and platform for newly created sessions. See "Configuring the Server Platform" on page 77-14 for more information.

Working With Databases

In relational projects, when you expand the database object in the **Navigator**, TopLink Workbench displays the database tables associated with the project. You can associate tables by importing them from the database, or by creating them within TopLink Workbench.

Figure 4–18 Sample Database Tables

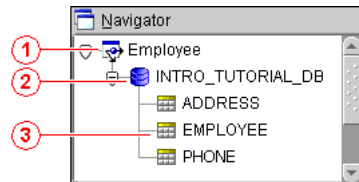


Figure 4–18 numbered callouts identify the following database icons.

1. Project
2. Database
3. Database table

Each database table property sheet contains the following tabs in the **Editor**:

- **Columns**—Add or modify the table's fields, and specify each field's properties.
- **References**—Specify references between tables.

This section includes information on the following topics:

- [Working With Database Tables in the Navigator Window](#)
- [Working With Database Tables in the Editor Window](#)
- [Generating Data From Database Tables](#)

Working With Database Tables in the Navigator Window

This section describes the following options:

- [Logging In and Out of a Database](#)
- [Creating New Tables](#)
- [Importing Tables From a Database](#)
- [Removing Tables](#)
- [Renaming Tables](#)
- [Refreshing Tables From the Database](#)

See "[Working With Database Tables in the Editor Window](#)" on page 4-25 for more information.

Logging In and Out of a Database

To log in or out of a relational database, do the following:

1. Create a database login (see "[Database Login Configuration Overview](#)" on page 86-1).

2. To log in to a relational database, right-click the database object in the **Navigator**, and choose **Log In to Database** from the context menu or choose **Selected > Log In to Database** from the menu.
3. To log out of a relational database, right-click the database object in the **Navigator** and choose **Log Out of Database** from the context menu or choose **Selected > Log Out of Database** from the menu.

Creating New Tables

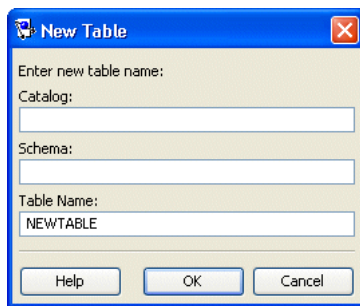
To create a new database table within TopLink Workbench, use the following procedure:



1. Select the database object in the **Navigator** window and click **Add New Table**. The **New Table** dialog box appears.

You can also right-click the database object and choose **Add New Table** from the context menu, or choose **Selected > Add New Table** from the menu.

Figure 4–19 New Table Dialog Box



2. Complete each field on the **New Table** dialog box and click **OK**.

Field	Description
Catalog	Use to identify specific database information for the table. Consult your database administrator for more information.
Schema	Use to identify specific database information for the table. Consult your database administrator for more information.
Table Name	Specify the name of this database table.

TopLink Workbench adds the database table to the project.

Although the database table has been added to the project, it has not been written to the actual database. See ["Generating Tables on the Database"](#) on page 4-32 for more information on creating the table in the database.

Continue with ["Working With Database Tables in the Editor Window"](#) on page 4-25 to use these tables in your project.

Importing Tables From a Database

TopLink Workbench can automatically read the schema for a relational database and import the table data into the project as long as your JDBC driver supports the following JDBC methods:

- `getTables`

- getTableTypes
- getImportedKeys
- getCatalogs
- getPrimaryKeys

The JDBC driver must be on the TopLink Workbench classpath (see ["Configuring the TopLink Workbench Environment"](#) on page 4-2).

To import tables from the database, use the following procedure:

1. Select the database object in the **Navigator**, and click **Add/Update Existing Tables from Database**. The **Import Tables from Database** dialog box appears.

You can also right-click on the database object in the **Navigator** and choose **Add/Update Existing Tables from Database** from the context menu or choose **Selected > Add/Update Existing Tables from Database** from the menu.

Figure 4-20 Import Tables from Database Dialog Box

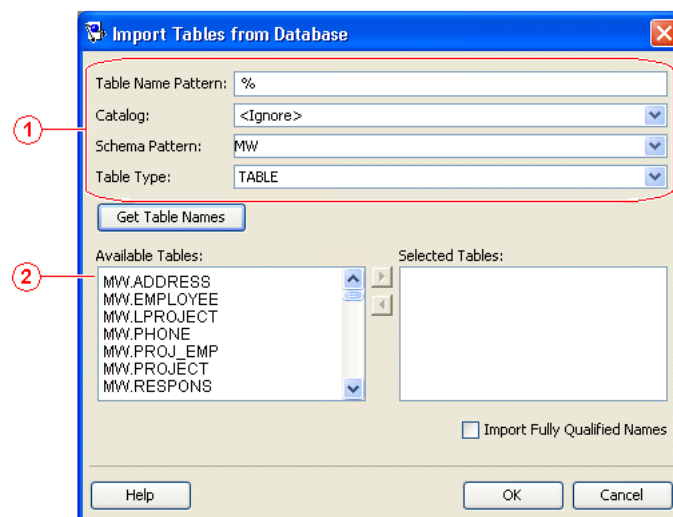


Figure 4-20 numbered callouts identify the following user interface components:

1. Filters
2. Database tables that match the filters
2. Complete each field on the **Import Tables from Database** dialog box and click **OK**.

Field	Description
Table Name Pattern	Specify the name of database table(s) to import. Use percent character (%) as a wildcard. Tables that match the Table Name Pattern can be imported.
Catalog	Specify the catalog of database table(s) to import.
Schema Pattern	Specify the schema of database table(s) to import.
Table Type	Specify the type of database table(s) to import.

Field	Description
Available Tables	Click Get Table Names to make TopLink display tables that match Table Name Pattern , Catalog , Schema Pattern , and Table Type settings.
Selected Tables	Select the tables in the Available Tables area to import, and click the right-arrow button. TopLink adds the table to the Selected Tables field. Click OK to import the tables from the database into the TopLink Workbench project.
Import Fully Qualified Names	Specify whether or not the tables' names are fully qualified against the schema and catalog.

Examine each table's properties to verify that the imported tables contain the correct information. See ["Working With Database Tables in the Editor Window"](#) on page 4-25 for more information.

Removing Tables

To remove a database table from the project, use the following procedure:



1. Select a database table in the **Navigator**, and click **Remove Table** on the toolbar. TopLink Workbench prompts for confirmation.

You can also right-click on the database object and choose **Remove** from the context menu or choose **Selected > Remove Table** from the menu.

2. Click **OK**. TopLink Workbench removes the table from the project.

Note: Although you have removed the table from the TopLink Workbench project, the table remains in the database.

Renaming Tables

To rename a database table in the TopLink Workbench project, use the following procedure:

1. Right-click the table in the **Navigator** and choose **Rename** from the context menu. The Rename dialog box appears.

You can also select the table and choose **Selected > Rename** from the menu.

2. Enter a new name and click **OK**. TopLink Workbench renames the table.

Note: Although you have renamed the table in the TopLink Workbench project, the original table name remains in the database.

Refreshing Tables From the Database

To refresh (that is, reload) the database tables in the TopLink Workbench project, use this procedure:

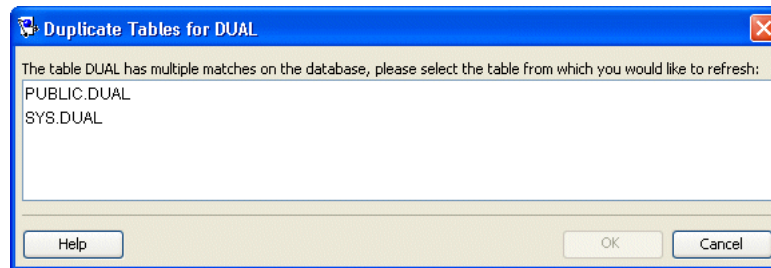


Select a database table in the **Navigator**, and click **Refresh from Database** on the toolbar.

You can also select the table and choose **Selected > Refresh from Database** from the menu, or click **Refresh**. TopLink Workbench reloads the database table.

When refreshing tables from the database, if there are multiple database tables with similar names, the **Duplicate Tables** dialog box appears.

Figure 4–21 Duplicate Table Dialog Box



Select the specific database table to update, and then click **OK**.

Working With Database Tables in the Editor Window

When you select a database table in the **Navigator**, its properties appear in the **Editor**. Each database table contains the following property tabs:

- **Columns**—Add or modify the table fields, and specify each field properties.
- **References**—Specify references between tables.

This section describes how to use these tabs to configure the following:

- [Working With Column Properties](#)
- [Setting a Primary Key for Database Tables](#)
- [Creating Table References](#)
- [Creating Field Associations](#)

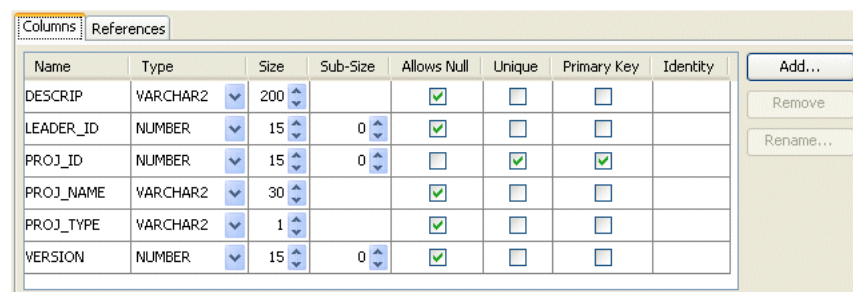
Working With Column Properties

Use the database table's **Column** tab to specify properties for the database table's fields.

To specify a table's column properties, use this procedure:

1. Select a database table in the **Navigator**. The table's property sheet displays in the **Editor**.
2. Click the **Columns** tab.

Figure 4–22 Fields Properties



- Enter data in each field on the Columns tab. Use the scroll bar to display the additional field.

Field	Description
Name	Specify the name of the field.
Type	Use the drop-down list to select the field's type. Note: The valid values will vary, depending on the database.
Size	Specify the size of the field.
Sub-Size	Specify the sub-size of the field.
Allows Null	Specify if this field can be null.
Unique	Specify whether the value must be unique within the table.
Primary Key	Specify whether or not this field is a primary key for the table (see "Setting a Primary Key for Database Tables" on page 4-26).
Identity	Use to indicate a Sybase, SQL Server or Informix identity field.

Note: Some properties may be unavailable, depending on your database type.

To add a new field, click **Add**.

To remove a field, select the field and click **Remove**.

To rename a field, select the field and click **Rename**.

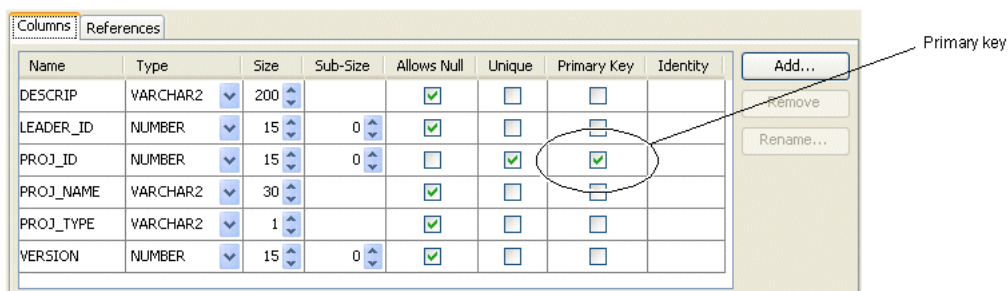
Setting a Primary Key for Database Tables

To set a primary key(s) for a database table, use this procedure:

Note: TopLink Workbench can automatically import primary key information if supported by the JDBC driver.

- Select a database table in the **Navigator**. Its property sheet appears in the **Editor**.
- Click the **Fields** tab.

Figure 4-23 *Setting Primary Key for a Database Table*



- Select the **Primary Key** field(s) for the table.

Creating Table References

References are table properties that contain the foreign key; they may or may not correspond to an actual constraint that exists on the database. TopLink Workbench uses these references when you define relationship mappings and multiple table associations.

When importing tables from the database, TopLink Workbench can automatically create references (if the driver supports this), or you can define references from the workbench. See ["Importing Tables From a Database"](#) on page 4-22.

To create a new table reference, use this procedure:

1. Select a database table in the **Navigator**. The table's properties display in the **Editor**.
2. Click the **References** tab.

Figure 4-24 *References Tab*

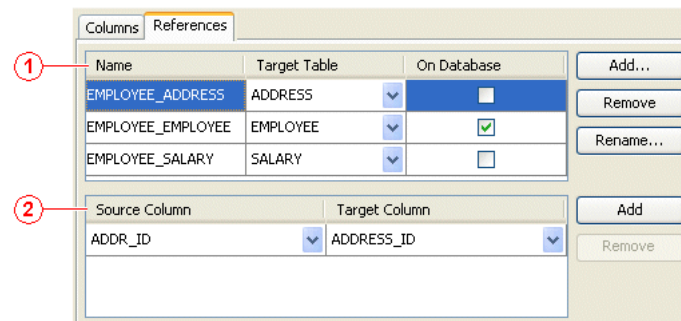
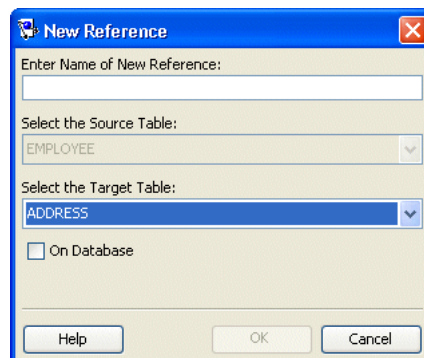


Figure 4-26 numbered callouts identify the following user interface components:

1. Table References area
2. Key Pairs area
3. In the **References** area, click **Add**. The **New Reference** dialog box appears.

Figure 4-25 *New Reference Dialog Box*



4. Complete each field on the New Reference dialog box and click **OK**.

Field	Description
Enter Name of New Reference	Specify the name of the reference table. If you leave this field blank, TopLink Workbench automatically creates a name based on the format: SOURCETABLE_TARGETTABLE.
Select the Source Table	Specify the name of the source database table (the currently selected table in the Navigator).
Select the Target Table	Use the list to specify the target table for this reference.
On Database	Specify if you want to create the reference on the database when you create the table. Not all database drivers support this option.

Continue with ["Creating Field Associations"](#) on page 4-28.

Creating Field Associations

For each table reference, you can specify one or more field associations that define how fields in the source table relate to fields in the target table. See ["Creating Table References"](#) on page 4-27.

To create new field references, use this procedure:

1. Select a database table in the Navigator. The table's properties display in the **Editor**.
2. Click the **References** tab.

Figure 4-26 References Tab

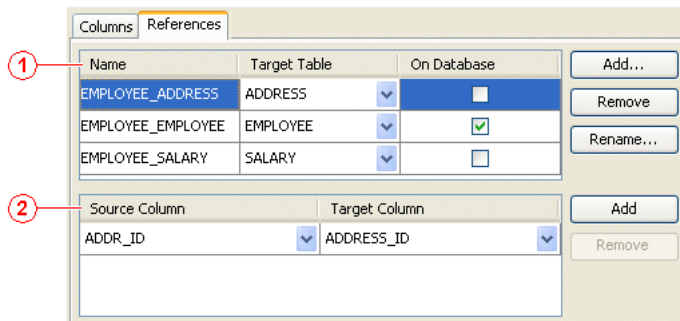


Figure 4-26 numbered callouts identify the following user interface components:

1. Table references area
2. Key pairs area
3. Select a table reference from the references area.
4. To create a new key pair, click **Add** in the key pairs area and complete each field in the key pairs area using the following information:

Field	Description
Table References Area	
Reference Name	Specify the name of this table reference
Target Table	Specify the database table that is the <i>target</i> of this reference.

Field	Description
On Table	Specify if the reference exists on the database.
Key Pairs Area	
Source Field	Select the database field from the source table.
Target Field	Select the database field from the target table.

Generating Data From Database Tables

TopLink Workbench can automatically generate a variety of information from the database tables. This section describes the following:

- [Generating SQL Creation Scripts](#)
- [Generating Classes and Descriptors From Database Tables](#)
- [Generating EJB Entities and Descriptors From Database Tables](#)
- [Generating Tables on the Database](#)

Generating SQL Creation Scripts

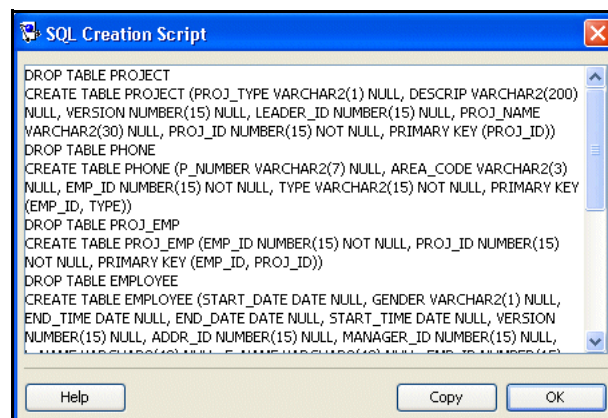
Using the TopLink Workbench, you can generate SQL scripts that you can use to create tables in a relational database.

To automatically generate SQL scripts to create the tables in a project, use this procedure:

1. Select the database table(s) in the **Navigator**.
2. Right-click the table(s) and choose **Generate Creation Script for > Selected Table** or **All Tables** from the context menu. The SQL Creation Script dialog box appears.

You can also choose **Selected > Generate Creation Script for > Selected Table** or **All Tables** from the menu.

Figure 4-27 SQL Creation Script Dialog Box



Copy the script and paste it into a file. You may need to edit the file to include additional SQL information that TopLink Workbench could not generate.

Note: If TopLink cannot determine how a particular table feature should be implemented in SQL, it generates a descriptive message in the script.

Generating Classes and Descriptors From Database Tables

TopLink Workbench can automatically generate Java class definitions, descriptor definitions, and associated mappings from the information in database tables. You can later edit the generated information if necessary.

For each table, TopLink Workbench does the following:

- Creates a class definition and a descriptor definition.
- Adds attributes to the class for each column in the table.
- Automatically generates access methods, if specified.
- Creates direct-to-field mappings for all direct (nonforeign key) fields in the table.
- Creates relationship mappings (one-to-one and one-to-many) if there is sufficient foreign key information. You may be required to determine the exact mapping type.

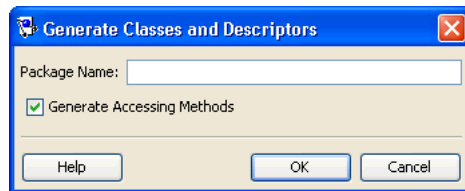
Note: Class and attribute names are generated based on the table and column names. You can edit the class properties to change their names.

To generate classes and descriptors from database tables, use the following procedure:

1. Select the database table(s) in the **Navigator**.
2. Right-click the table(s) and choose **Generate Classes and Descriptors from > Selected Table** or **All Tables** from the context menu.

You can also choose **Selected > Generate Classes and Descriptors from > Selected Table** or **All Tables** from the menu.
3. Click **Yes**. The Generate Classes and Descriptors dialog box appears.

Figure 4–28 *Generate Classes and Descriptors Dialog Box*



4. Complete each field on the Generate Classes and Descriptors dialog box and click **OK**.

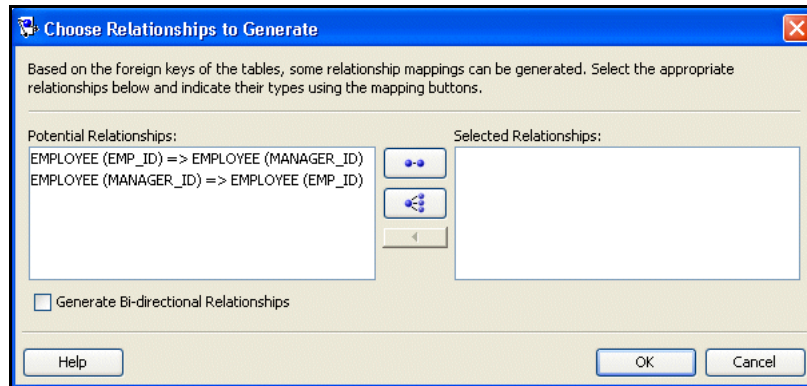
Use the following information to enter data in each field:

Field	Description
Package Name	Specify the name of package to generate. The package name must comply with Java naming standards.
Generate Accessing Methods	Specify if TopLink Workbench generates accessing methods for each class and descriptor.

If the table contains foreign key fields that may represent relationship mappings, then the Choose Relationships to Generate dialog box appears.

If the table contains foreign key fields that may represent relationship mappings, then the Choose Relationships to Generate dialog box appears.

Figure 4–29 Choose Relationships to Generate Dialog Box



5. Select an entry from **Potential Relationships** and click the **1:1 Mapping** or **1:M Mapping** button, located between the Potential Relationships and **Selected Relationships** windows. See ["Understanding Relational Mappings"](#) on page 36-1 for more information on mappings.

You can also specify whether the relationships are bidirectional. See ["Configuring Bidirectional Relationship"](#) on page 35-34 for more information.

6. Click **OK** to automatically create the relationships.

The newly created descriptors appear in the **Navigator** of TopLink Workbench.

Generating EJB Entities and Descriptors From Database Tables

Using TopLink Workbench, you can automatically generate EJB entities and descriptors for each database table, including the following:

- One EJB descriptor that implements the `<javax.ejb.EntityBean>` interface and four EJB 1.1 classes for each table
- Bean relation attributes (CMP or BMP)
- Java source for each class
- EJB-compliant method stubs

Note: This option is available only for projects with CMP or BMP persistence. See ["Configuring Persistence Type"](#) on page 22-5 for more information.

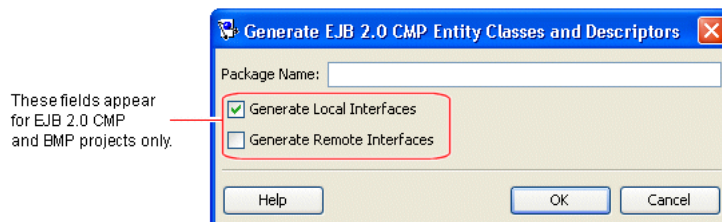
To automatically generate EJB entities and descriptors for each database table, use this procedure:

1. Select the database table(s) in the **Navigator**.
2. Right-click the table(s) and choose **Generate EJB Entities and Descriptors from > Selected Table** or **All Tables** from the context menu. TopLink Workbench prompts you to save your project.

You can also choose **Selected > Generate EJB Entities and Descriptors from > Selected Table** or **All Tables** from the menu.

3. Click **Yes** to save your project before generating EJB entities. The Generate EJB Entity Classes and Descriptors dialog box appears.

Figure 4–30 Generate EJB Entity Classes and Descriptors Dialog Box



4. Complete each field on the Generate EJB Entity Classes and Descriptors dialog box and click **OK**.

Field	Description
Package Name	Name of the package to contain the generated entities and descriptors.
Generate Local Interfaces¹	Specify if TopLink creates local interfaces for the EJB entities.
Generate Remote Interfaces¹	Specify if TopLink creates remote interfaces for the EJB entities.

¹ For CMP 2.0 and BMP projects only. See ["Configuring a Descriptor With EJB Information"](#) on page 28-44 for more information.



If the table contains foreign key fields that may represent relationship mappings, then the Choose Relationships to Generate dialog box appears. Select a potential relationship and click the **1:1 Mapping** or **1:M Mapping** button, located between the Potential Relationships and **Selected Relationships** windows.

You can also specify if the relationships are bidirectional (see ["Configuring Bidirectional Relationship"](#) on page 35-34).

Repeat for all appropriate sets of tables.

Click **OK** to generate the relationship mappings.

The system creates the remote primary key, home, and bean classes for each bean and adds this information to the project.

Generating Tables on the Database

To create a table in the database, based on the information in TopLink Workbench, use this procedure:

Note: You must log in the database before creating tables. See ["Logging in to the Database"](#) on page 23-7 for more information.

1. Select the database table(s) in the **Navigator**.

2. Right-click the table(s) and choose **Create on Database > Selected Table** or **All Tables** from the context menu.

You can also create tables by selecting **Selected > Create on Database > Selected Table** or **All Tables** from the menu.

TopLink Workbench creates the tables on the database.

Alternatively, you can generate tables at run time by exporting the information in TopLink Workbench to a TableCreator class (see ["Understanding the Schema Manager"](#) on page 6-1).

Working With XML Schemas

For XML and EIS projects, TopLink Workbench maps each TopLink descriptor to your XML schema.

This section includes information on the following topics:

- [Working With XML Schemas in the Navigator](#)
- [Working With XML Schema Structure](#)
- [Importing an XML Schema](#)
- [Configuring XML Schema Reference](#)
- [Configuring XML Schema Namespace](#)

Working With XML Schemas in the Navigator

After you import one or more XML schemas into your project (see [Importing an XML Schema](#) on page 4-34) and you expand the schema object in the **Navigator**, TopLink Workbench displays the schemas associated with the project.

Figure 4-31 Sample XML Schemas

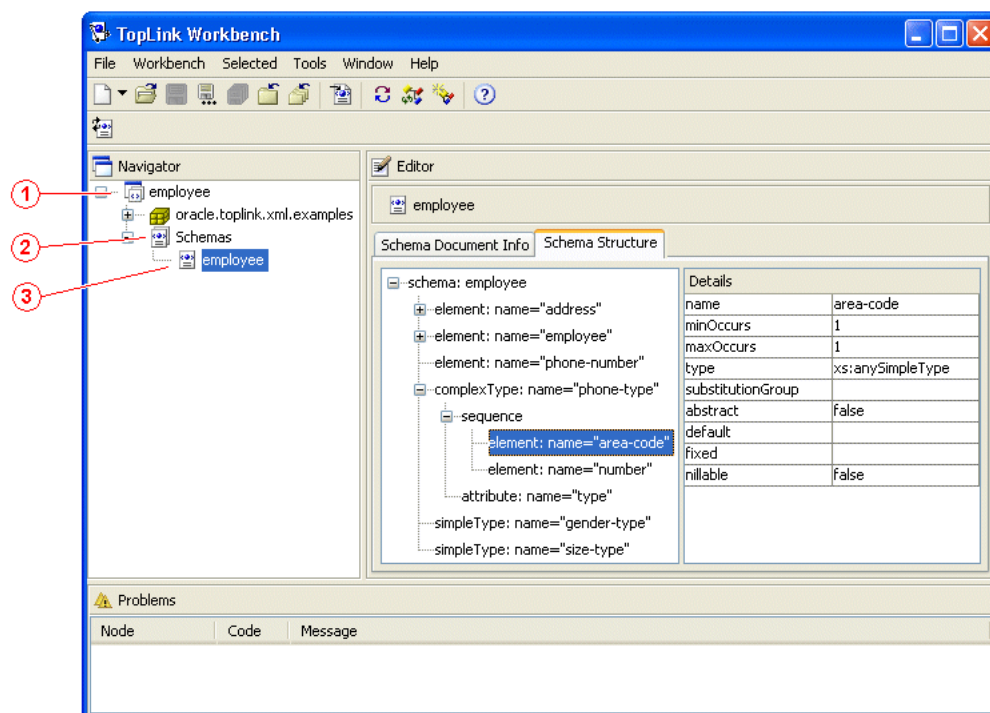


Figure 4–31 numbered callouts identify the following schema icons:

1. Project
2. Schemas object
3. Specific schema

For more information, see the following:

- ["Working With XML Schema Structure"](#) on page 4-34
- ["Configuring XML Schema Reference"](#) on page 4-36
- ["Configuring XML Schema Namespace"](#) on page 4-38

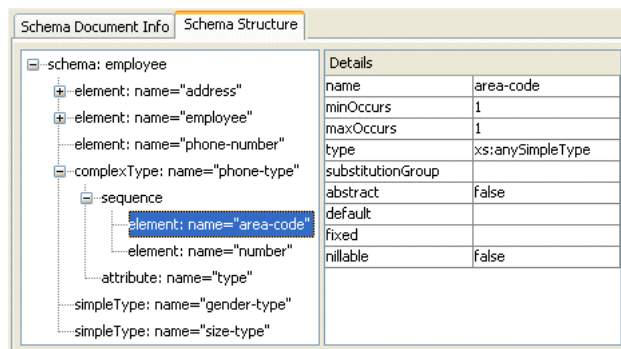
Working With XML Schema Structure

When you select a specific XML schema in the **Navigator**, you can display the structure and details of the schema using the Schema Structure tab.

To display the structure and details of a schema, use this procedure:

1. Select a schema element in the **Navigator**. Its properties appear in the **Editor**.
2. Click the **Schema Structure** tab. The Schema Structure tab appears.
3. Select an element in the schema. The element's details appear.

Figure 4–32 Schema Structure Tab



Use the following information to verify data in each field in the Schema Document Info tab:

Field	Description
Schema Structure	Displays the elements of the schema, listed in alphabetical order, in an expandable or collapsible tree structure.
Details	Displays detailed information (such as name and type) for the currently selected element in the Schema Structure area.

These fields are for display only and cannot be changed in TopLink Workbench.

Importing an XML Schema

The first step in configuring an EIS project (using XML records) or XML project is importing the XML schema(s) that your project uses.

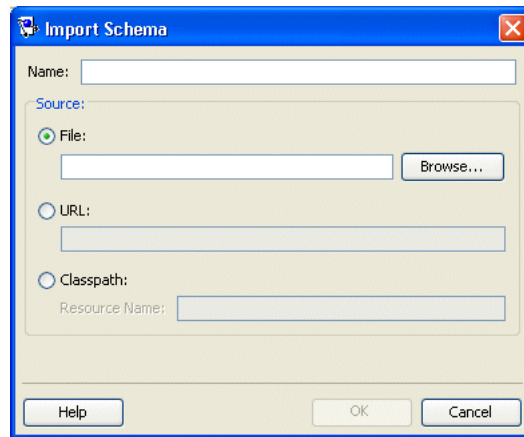
When you import a schema, you define a schema reference that gives TopLink the information it needs to locate the schema itself. Anytime after you import an XML schema, you can update the schema reference (see ["Configuring XML Schema Reference"](#) on page 4-36) if necessary.

After importing an XML schema, you can configure XML schema namespaces (see ["Configuring XML Schema Namespace"](#) on page 4-38).

To import an XML schema into an EIS project (using XML records) or an EIS project, use this procedure:

1. Right-click the schemas element in the **Navigator** and select **Import Schema** from the context menu. The Import Schema dialog box appears.

Figure 4-33 Import Schema Dialog Box



Use the following information to enter data in each field in the Import Schema dialog box:

Field	Description
Name	Specify the name of this schema. This is the display name that TopLink Workbench uses. It can be different than the name you specify when you configure Source .
Source	Select how TopLink Workbench should import the schema.
File	Specify that TopLink Workbench should import the schema from a file. Enter the fully qualified directory path and filename of the schema file.
URL	Specify that TopLink Workbench should import the schema using a URL. Enter the complete URL of the schema file. Note: When importing schemas by URL, ensure you have set your proxy information correctly. See "General Preferences" on page 4-13 for more information.
Classpath	Specify that TopLink Workbench should import the schema from the project classpath.

Field	Description
Resource Name	Enter the fully qualified name of the XML schema file including the name of the package of which it is a part. For example, if your XML schema mySchema.xsd is in C:\project\config and you add this directory to your project classpath (see "Configuring Project Classpath" on page 22-3), specify a resource name of project.config.mySchema.xsd.

To reimport *a specific schema*, right-click on the specific schema in the **Navigator** and select **Reimport Schema** from the context menu.

To reimport *all schemas in a project*, right-click on **Schemas** in the **Navigator** and select **Reimport All Schemas** from the context menu.

To change a schema's source, right-click on the specific schema in the **Navigator** window and select **Properties** from the context menu. The Schema Properties dialog appears.

Configuring XML Schema Reference

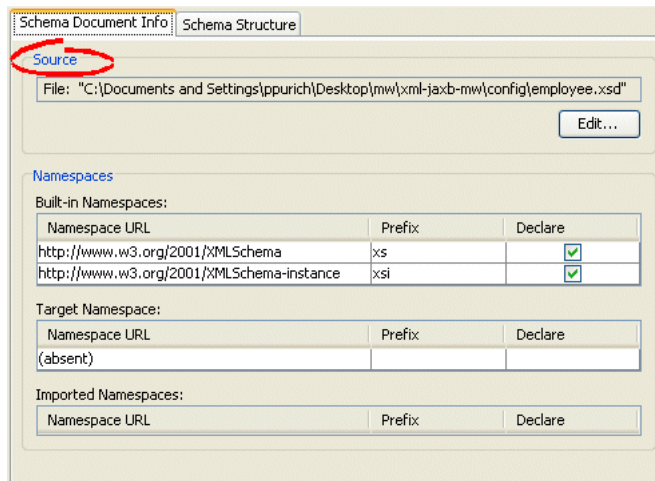
After you import an XML schema (see ["Importing an XML Schema"](#) on page 4-34), you can update its source by configuring the schema reference.

Using TopLink Workbench

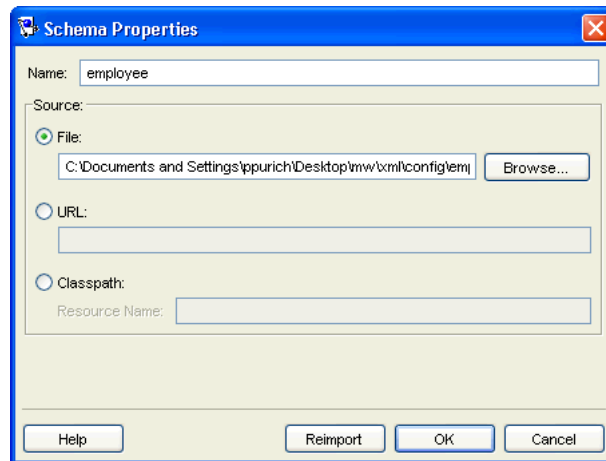
To specify the source of a schema, use this procedure:

1. Select a schema element in the **Navigator**. Its properties appear in the **Editor**.
2. Click the **Schema Document Info** tab. The Schema Document Info tab appears.

Figure 4–34 Schema Document Info Tab–Source Field



3. Click **Edit** to select a new source for the selected schema. The Schema Properties dialog box appears.

Figure 4-35 Schema Properties Dialog Box

Use the following information to complete each field in the Schema Properties dialog box:

Field	Description
Name	Specify the name of this schema. This is the display name that TopLink Workbench uses. It can be different than the name you specify when you configure Source .
Source	Select how TopLink Workbench should import the schema.
File	Specify that TopLink Workbench should import the schema from a file. Enter the fully qualified directory path and filename of the schema file.
URL	Specify that TopLink Workbench should import the schema using a URL. Enter the complete URL of the schema file. Note: When importing schemas by URL, ensure you have set your proxy information correctly. See "General Preferences" on page 4-13 for more information.
Classpath	Specify that TopLink Workbench should import the schema from the project classpath.
Resource Name	Enter the fully qualified name of the XML schema file including the name of the package of which it is a part. For example, if your XML schema mySchema.xsd is in C:\project\config and you add this directory to your project classpath (see "Configuring Project Classpath" on page 22-3), specify a resource name of project.config.mySchema.xsd.

Using Java

Use Java to configure schema reference. Create a descriptor amendment method (see ["Configuring Amendment Methods"](#) on page 28-78) that instantiates the appropriate type of XMLSchemaReference (XMLSchemaClassPathReference, XMLSchemaFileReference, or XMLSchemaURLReference) and configures the descriptor with it, as follows:

- If you are using `EISDescriptors`, the TopLink runtime does not use the schema reference; no further configuration is required.
- If you are using `XMLDescriptors`, configure the descriptor with the `XMLSchemaReference` using `XMLDescriptor` method `setSchemaReference`.

Configuring XML Schema Namespace

As defined in <http://www.w3.org/TR/REC-xml-names/>, an **XML namespace** is a collection of names, identified by a URI reference, which are used in XML documents as element types and attribute names. To promote reusability and modularity, XML document constructs should have universal names, whose scope extends beyond their containing document. XML namespaces are the mechanism which accomplishes this.

When you import an XML schema (see ["Importing an XML Schema"](#) on page 4-34) such as the one that [Example 4-3](#) shows, TopLink Workbench organizes the various namespaces that the XML schema identifies as [Table 4-3](#) shows.

Example 4-3 XML Schema with Namespace Options

```
<xsd:schema
  xmlns:<prefix>=<"URI">          <!-- TopLink Workbench Built-in Namespace -->
  targetNamespace=<"URI">          <!-- TopLink Workbench Target Namespace -->
  elementFormDefault="qualified"
  attributeFormDefault="unqualified"
  version="10.1.3">
  <xsd:import                      <!-- TopLink Workbench Imported Namespace -->
    namespace="http://xmlns.oracle.com/ias/xsds/opm"
    schemaLocation="object-persistence_1_0.xsd"
  />
  ...
</xsd:schema>
```

Table 4-3 TopLink Workbench XML Schema Categories

TopLink Workbench Category	Defined By	Purpose	When Needed
Built-in	<code>xmlns:<prefix>=<"URI"></code>	Provides access to types defined in other XML schemas for use as is.	If your project uses more than one XML schema or if you want to use <code>xsi</code> or <code>xsd</code> types.
Target	<code>targetNamespace=<"URI"></code>	The namespace you use to qualify the types you define for your application. If set, all XML documents that use these types must use this namespace qualifier.	You may need to specify a target namespace depending on how element and attribute form options are set (see "Element and Attribute Form Options" on page 20-22).
Imported	<code>xsd:import</code>	Provides access to types defined in the corresponding built-in XML schema so that you can extend the built-in types. Extended types must be qualified by the target namespace.	If your project uses more than one XML schema and you want to extend one or more built-in types.

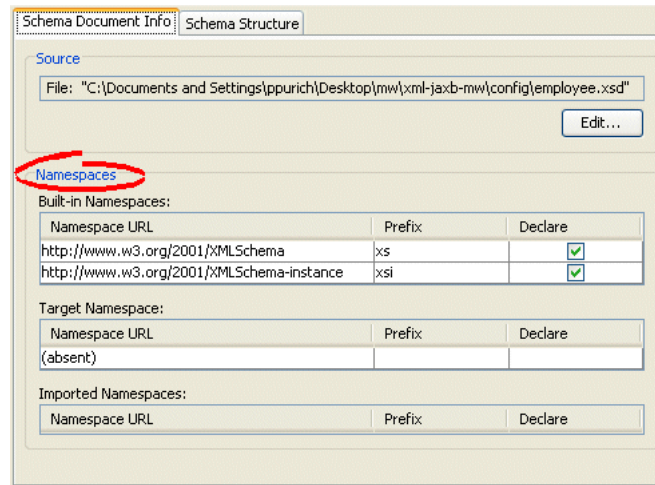
For more information, see ["Understanding XML Namespaces"](#) on page 20-22.

Using TopLink Workbench

To specify the namespaces of a schema, use this procedure:

1. Select a schema element in the **Navigator**. Its properties appear in the **Editor**.
2. Click the **Schema Document Info** tab. The Schema Document Info tab appears.

Figure 4-36 Schema Document Info Tab–Namespaces Field



Use the following information to complete each Namespaces field in the tab:

Field	Description
Built-in Namespaces	All namespaces defined by <code>xmlns:<prefix>=<URI></code> .
Target Namespaces	All namespaces defined by <code>targetNamespace=<URI></code> .
Imported Namespaces	All namespaces defined by <code>xsd:import</code> .
Prefix	Double-click in the Prefix field to specify the prefix that corresponds to the given namespace. When the TopLink runtime marshalls (writes) an object to an XML document, it uses the namespace prefixes you specify here. When the TopLink runtime unmarshalls (reads) an XML document, the document may use any prefix value as long as it corresponds to the appropriate namespace. For more information, see "TopLink Runtime Namespace Resolution" on page 20-26.
Declare	When selected, XML documents must use the corresponding URI qualifier when referring to types from this namespace. XML documents may use a different prefix value as long as that value is associated with the appropriate namespace URI. For more information, see "TopLink Runtime Namespace Resolution" on page 20-26.

Using Java

Using Java, to configure XML schema namespaces for an EIS descriptor (with XML records) or an XML descriptor, create a descriptor amendment method (see ["Configuring Amendment Methods"](#) on page 28-78) that uses `EISDescriptor` or `XMLDescriptor` method `getNamespaceResolver` to configure the descriptor's `NamespaceResolver` accordingly as [Example 4-4](#) shows.

Example 4-4 Configuring Namespaces

```

public void addToDescriptor(ClassDescriptor descriptor) {
    descriptor.getNamespaceResolver.put(
        prefix,
        namespaceURI
    );
}

```

Working With Classes

Using TopLink Workbench, you can create Java classes and packages. This section includes information on the following:

- [Creating Classes](#)
- [Configuring Classes](#)
- [Importing and Updating Classes](#)
- [Managing Nondesoriptor Classes](#)
- [Renaming Packages](#)

Creating Classes

Oracle recommends that you develop your Java classes using an IDE such as Oracle JDeveloper and import these existing classes into TopLink Workbench (see ["Importing and Updating Classes"](#) on page 4-50)

However, it is sometimes convenient to create and configure classes in TopLink Workbench: for example, when generating an object model from a database schema.

This section includes information on using TopLink Workbench to create Java classes.

For more information on using TopLink Workbench to edit classes, see ["Creating Classes"](#) on page 4-40.

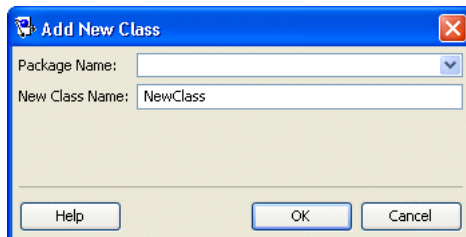
Using TopLink Workbench

To create new classes and packages from within TopLink Workbench, use this procedure:

1. Select the project in the **Navigator** and click **Create New Class**.

You can also right-click the project in the **Navigator** and choose **Create New Class** from the context menu or choose **Selected > Create New Class** from the menu.

Figure 4-37 Add New Class Dialog Box



Use the following information to enter data in each field on the Add New Class dialog box:

Field	Description
Package Name	Choose an existing package or enter a new package name. If blank, TopLink Workbench uses the default package name.
New Class Name	Enter a class name. The New Class Name must be unique within the package.

For more information on using TopLink Workbench to edit classes, see ["Configuring Classes"](#) on page 4-41.

Configuring Classes

Oracle recommends that you develop your Java classes using an IDE such as Oracle JDeveloper and import these existing classes into TopLink Workbench (see ["Importing and Updating Classes"](#) on page 4-50)

However, it is sometimes convenient to create (see ["Creating Classes"](#) on page 4-40) and configure classes in TopLink Workbench: for example, when generating an object model from a database schema.

This section describes using TopLink Workbench to edit classes, including the following:

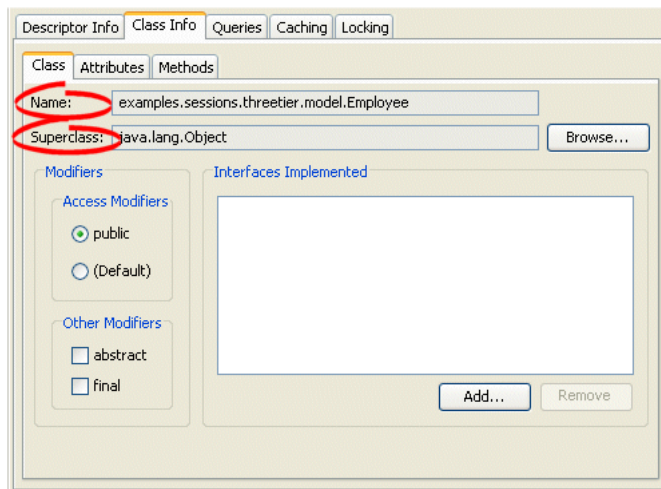
- [Configuring Class Information](#)
- [Configuring Class Modifiers](#)
- [Configuring Class Interfaces](#)
- [Adding Attributes](#)
- [Configuring Attribute Modifiers](#)
- [Configuring Attribute Type Information](#)
- [Configuring Attribute Accessing Methods](#)
- [Adding Methods](#)
- [Configuring Method Modifiers](#)
- [Configuring Method Type Information](#)
- [Configuring Method Parameters](#)

Configuring Class Information

This section includes information on [Using TopLink Workbench](#) to configure class information.

Using TopLink Workbench To configure class and superclass information, use this procedure:

1. Select a class in the **Navigator**. Its properties appear in the **Editor**.
2. Click the **Class Info** tab in the **Editor**.
3. Click the **Class** tab.

Figure 4–38 Class Tab, Class Information Fields

Use the following information to enter data in each field on the tab:

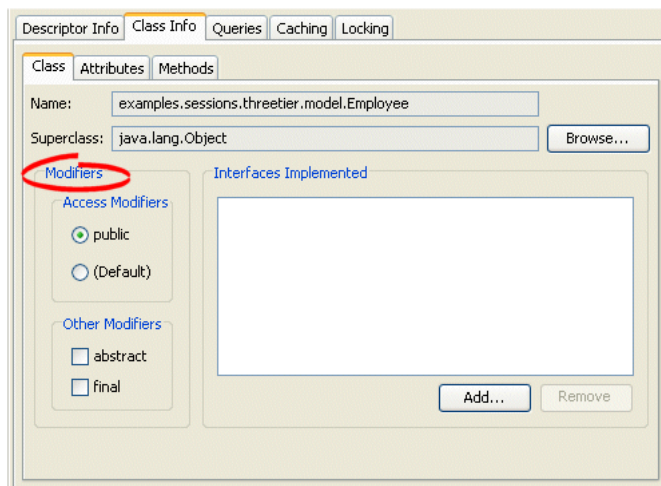
Field	Description
Name	The name of the class. This field is for display only.
Superclass	Click Browse and select a class and package that contains the class (that is, the superclass).

Configuring Class Modifiers

This section includes information on [Using TopLink Workbench](#) to configure class modifiers.

Using TopLink Workbench To configure class modifiers, use this procedure:

1. Select a class in the **Navigator**. Its properties appear in the **Editor**.
2. Click the **Class Info** tab in the **Editor**.
3. Click the **Class** tab.

Figure 4–39 Class Tab, Class Modifiers Fields

Use the following information to enter data in each field on the tab:

Field	Description
Access Modifiers	Use to specify whether the class is accessible publicly or not. Only public classes are visible to the Oracle TopLink Workbench.
Other Modifiers	Specify if the class is Final or Abstract , or both. Final classes are not included in the superclass selection lists for other classes to extend.

Configuring Class Interfaces

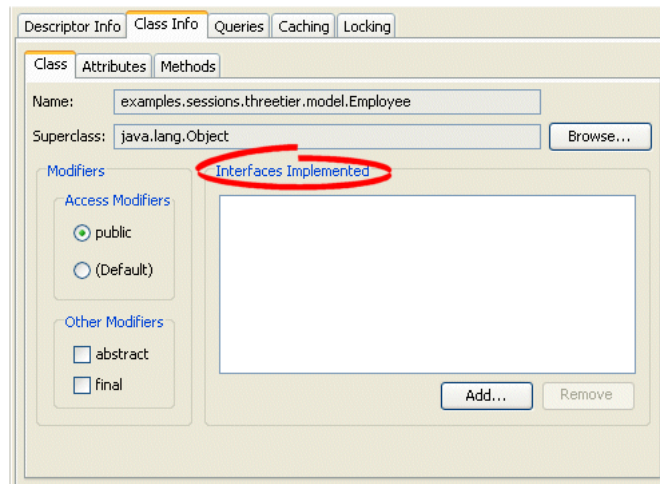
This section includes information on [Using TopLink Workbench](#) to specify the interfaces implemented by a class. You can choose any interface in the TopLink Workbench classpath (see ["Configuring Project Classpath"](#) on page 22-3).

Although you may add interfaces to a project directly (see ["Importing and Updating Classes"](#) on page 4-50), you do not need to do so in order to configure a class to implement an interface.

Using TopLink Workbench To implement interfaces, use this procedure:

1. Select a class in the **Navigator**. Its properties appear in the **Editor**.
2. Click the **Class Info** tab in the **Editor**.
3. Click the **Class** tab.

Figure 4-40 Class Tab, Interfaces Implemented Fields



Use the following information to enter data in each field on the Interfaces Implemented tab:

Field	Description
Interfaces Implemented	To add an interface, click Add and select the interface and package. To remove an interface, select the interface and click Remove

Adding Attributes

This section includes information on [Using TopLink Workbench](#) to add an attribute to a class.

Using TopLink Workbench To add a new attribute (field) to the descriptor, click **Add**.

To delete an existing attribute, select the attribute and click **Remove**.

To rename an existing attribute, select the attribute and click on **Rename**.

The **Attributes** tab contains the following tabs:

- General
- Accessors

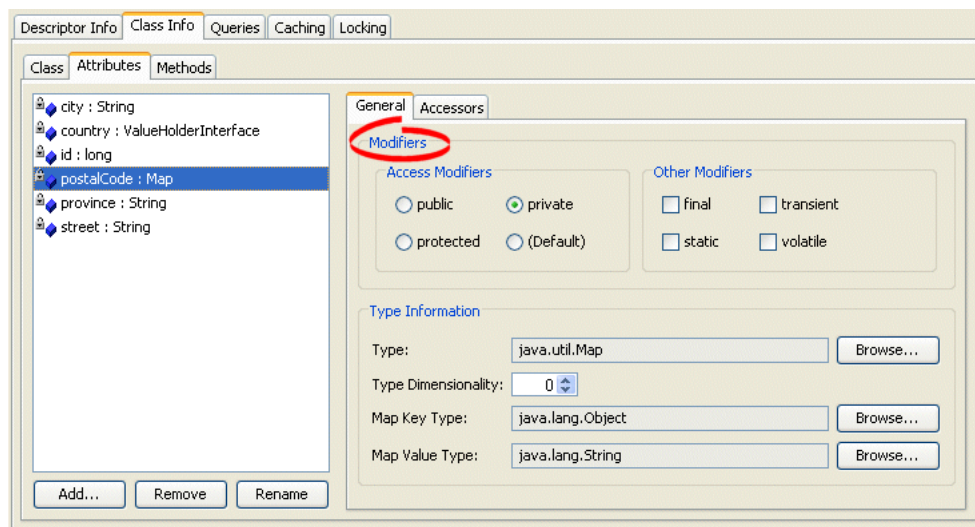
Configuring Attribute Modifiers

This section includes information on [Using TopLink Workbench](#) to configure attribute modifiers.

Using TopLink Workbench To specify access modifiers, use this procedure:

1. Select a class in the **Navigator**. Its properties appear in the **Editor**.
2. Click the **Class Info** tab in the **Editor**.
3. Click the **Attributes** tab. The Attributes tab contains two sub-tabs.
4. Click the **General** tab.

Figure 4-41 *Attributes Tab, Modifiers Fields*



Use the following information to enter data in each field on the tab:

Field	Description
Access Modifiers	Specify how the attribute is accessible:  Public  Protected—only visible within its own package and subclasses.  Private—not visible for subclasses  Default—only visible within its own package
Other Modifiers	Specify whether the attribute is Final , Static , Transient , or Volatile . Note: Selecting some modifiers may disable others.

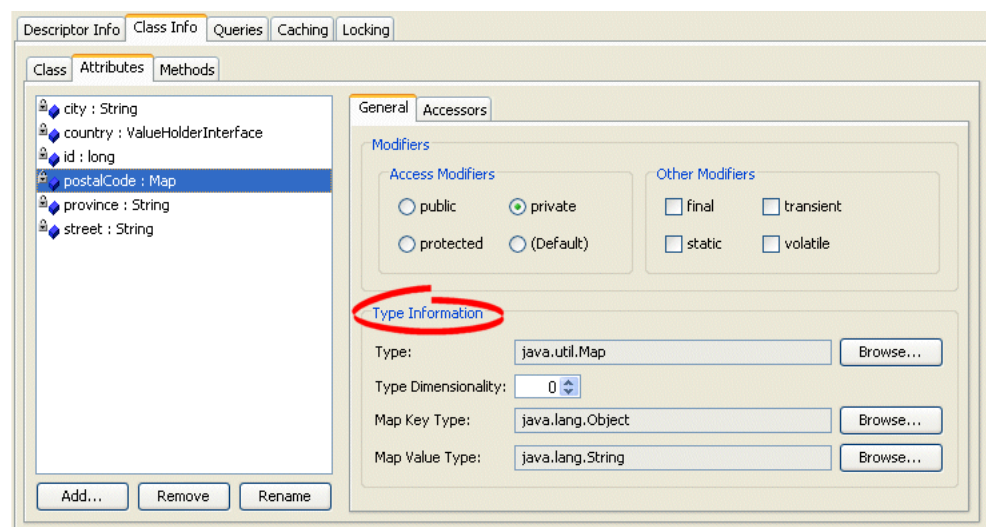
Configuring Attribute Type Information

This section includes information on [Using TopLink Workbench](#) to configure attribute type information.

Using TopLink Workbench To specify attribute type information, use this procedure:

1. Select a class in the **Navigator**. Its properties appear in the **Editor**.
2. Click the **Class Info** tab in the **Editor**.
3. Click the **Attributes** tab. The Attributes tab contains two sub-tabs.
4. Click the **General** tab.

Figure 4-42 Attributes Tab, Type Information Fields



Use the following information to enter data in each field on the tab:

Field	Description
Type	Click Browse and select a class and package for the attribute.
Type Dimensionality	Specify the length of an array. This field applies only if Type is an array.

Field	Description
Value Type	Click Browse and select a class and package for the attribute. This field applies for <code>ValueHolderInterface</code> types only.
Map Key Type	Click Browse and select a class and package for the attribute. This field applies for <code>Map</code> types only.
Map Value Type	Click Browse and select a class and package for the attribute. This field applies for <code>Map</code> types only.
Element Type	Click Browse and select a class and package for the attribute. This field applies for <code>List</code> types only.

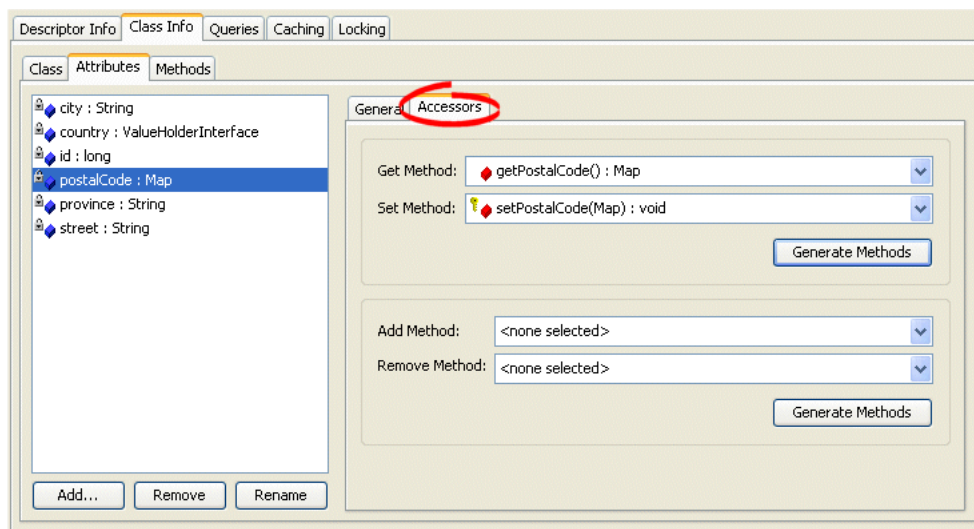
Configuring Attribute Accessing Methods

This section includes information on [Using TopLink Workbench](#) to configure attribute accessing methods. If you change an attribute and regenerate the accessing methods, TopLink *does not* remove any previously generated methods.

Using TopLink Workbench To specify attribute accessing methods, use this procedure:

1. Select a class in the **Navigator**. Its properties appear in the **Editor**.
2. Click the **Class Info** tab in the **Editor**.
3. Click the **Attributes** tab. The Attributes tab contains two sub-tabs.
4. Click the **Accessors** tab.

Figure 4–43 Attributes Tab, Accessors Fields



Use the following information to complete the fields on the Accessors tab:

Field	Description
Get Method	Choose the <code>get</code> method for the attribute. This field applies for non- <code>Collection</code> types only.
Set Method	Choose the <code>set</code> method for the attribute. This field applies for non- <code>Collection</code> types only.

Field	Description
Add Method	Choose the add method for the attribute. This field applies for List and Map types only.
Remove Method	Choose the remove method for the attribute. This field applies for List and Map types only.
Value Holder Get Method	Choose the method used to return the ValueHolderInterface type. This field applies for ValueHolderInterface types only.
Value Holder Set Method	Choose the method used to set the ValueHolderInterface type. This field applies for ValueHolderInterface types only.
Value Get Method	Choose the method used to return the actual value. This field applies for ValueHolderInterface types only.
Value Set Method	Choose the method used to set the actual value. This field applies for ValueHolderInterface types only.

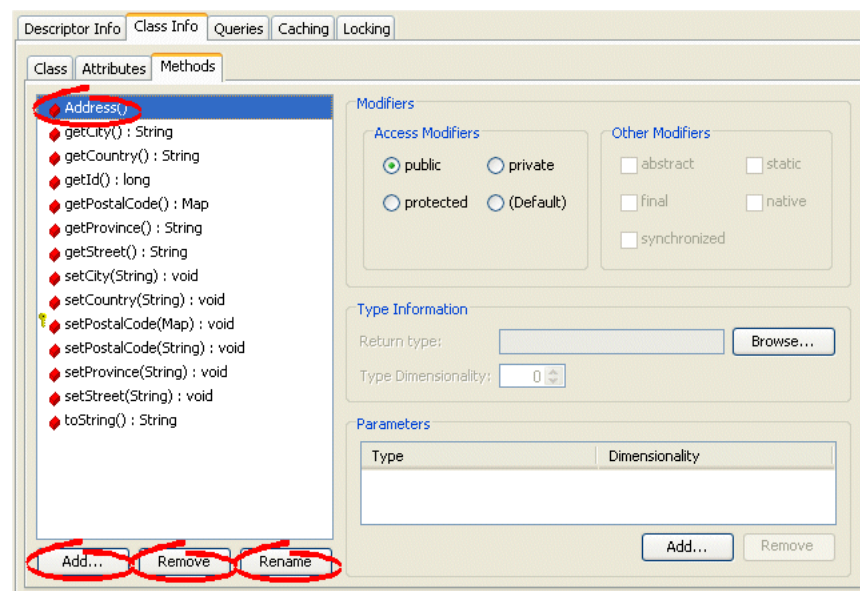
Adding Methods

This section includes information on [Using TopLink Workbench](#) to add a method to a class.

Using TopLink Workbench To add or remove methods, use this procedure:

1. Select a class in the **Navigator**. Its properties appear in the **Editor**.
2. Click the **Class Info** tab in the **Editor**.
3. Click the **Methods** tab.

Figure 4-44 Class Info–Methods Tab



To add a new method to the descriptor, click **Add**.

To delete an existing method, select the method and click **Remove**.

To rename an existing method, select the method and click **Rename**.

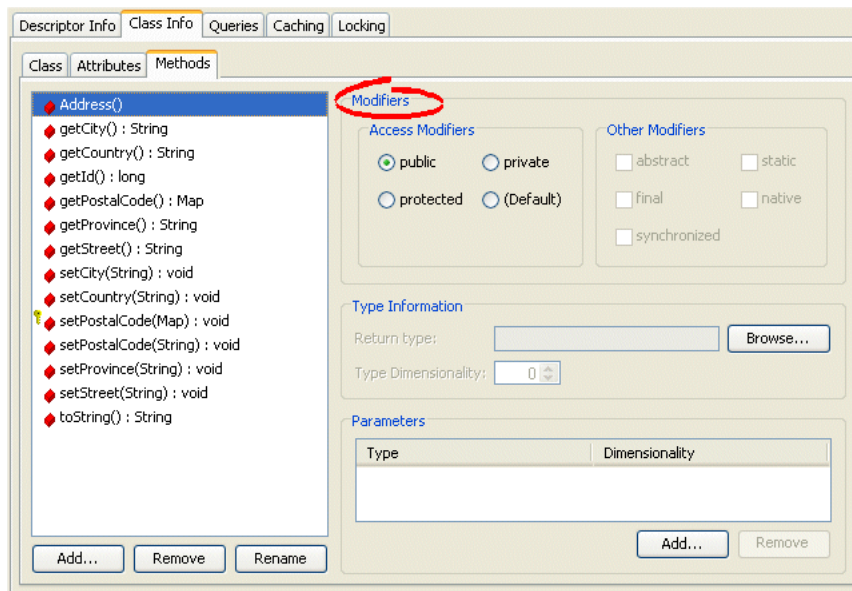
Configuring Method Modifiers

This section includes information on [Using TopLink Workbench](#) to configure method modifiers.

Using TopLink Workbench To specify access modifiers, use this procedure:

1. Select a class in the **Navigator**. Its properties appear in the **Editor**.
2. Click the **Class Info** tab in the **Editor**.
3. Click the **Methods** tab.

Figure 4–45 Methods Tab, Modifiers Fields



Use the following information to enter data in each field on the tab:

Field	Description
Access Modifiers	Specify how the method can be accessed: Public Protected—only visible within its own package and subclasses. Private—not visible for subclasses. Default—only visible within its own package.
Other Modifiers	Specify whether the method is Abstract , Final , Synchronized , Static , or Native . Note: Selecting some modifiers may disable others.

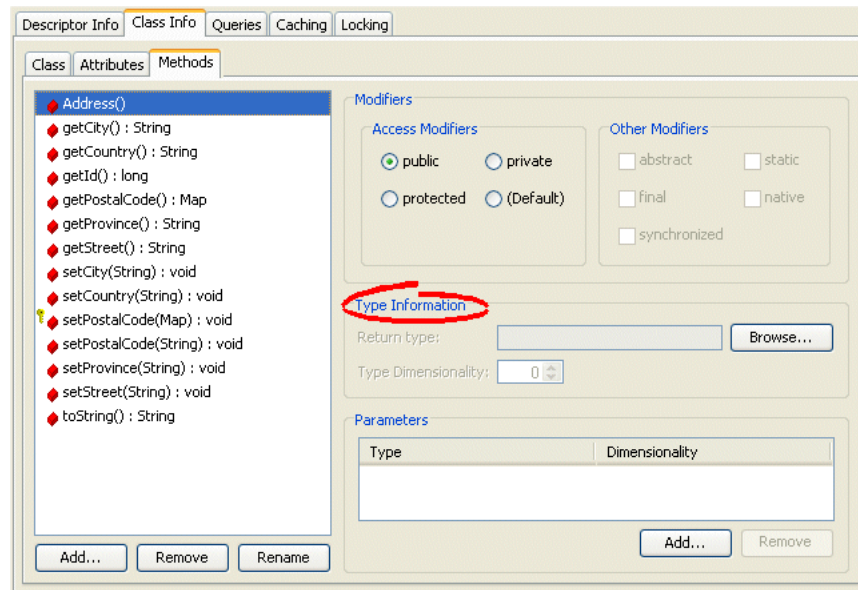
Configuring Method Type Information

This section includes information on [Using TopLink Workbench](#) to configure method type information.

Using TopLink Workbench To specify method type information, use this procedure:

1. Select a class in the **Navigator**. Its properties appear in the **Editor**.
2. Click the **Class Info** tab in the **Editor**.
3. Click the **Methods** tab.

Figure 4-46 Methods Tab, Type Information Fields



Use the following information to enter data in each field on the tab:

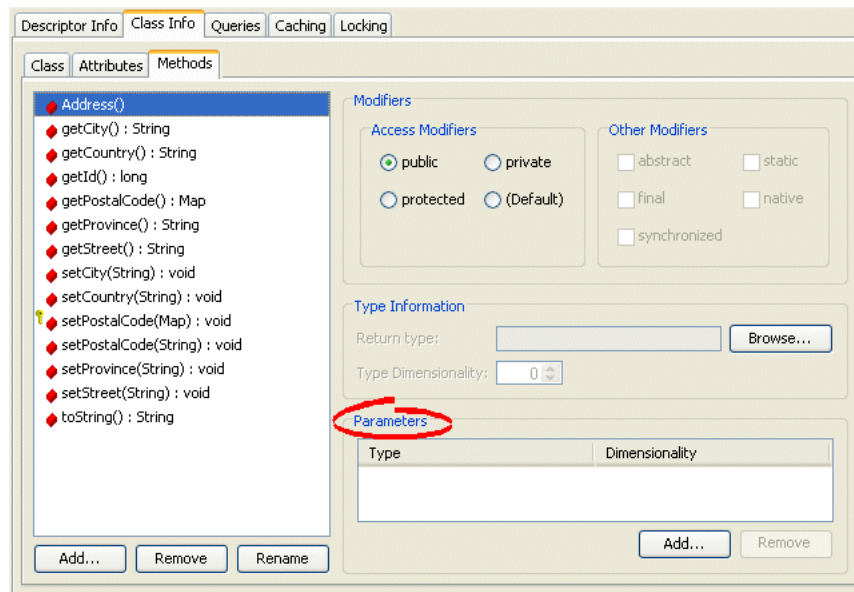
Field	Description
Return Type	Click Browse and select a class and package for the method.
Type Dimensionality	Specify the length of an array. This field applies only if Type is an array.

Configuring Method Parameters

This section includes information on [Using TopLink Workbench](#) to configure method parameters.

Using TopLink Workbench To specify additional method parameters, use this procedure:

1. Select a class in the **Navigator**. Its properties appear in the **Editor**.
2. Click the **Class Info** tab in the **Editor**.
3. Click the **Methods** tab.

Figure 4-47 Methods Tab, Method Parameters Fields

Use the following information to enter data in each field on the tab:

Field	Description
Type	Click Browse and select a class and package for the method.
Dimensionality	Specify the length of an array. This field applies only if Type is an array.

Importing and Updating Classes

This section includes information on [Using TopLink Workbench](#) to import and update Java classes.

You can import Java classes and interfaces created in any IDE.

You can import any class on the system classpath or project classpath.

If a class exists on both the system classpath and the project classpath, TopLink Workbench will update the class from the system classpath. To update or refresh from the project classpath, remove the class from the system classpath and restart TopLink Workbench.

For more information, see ["Configuring Project Classpath"](#) on page 22-3.

Using TopLink Workbench

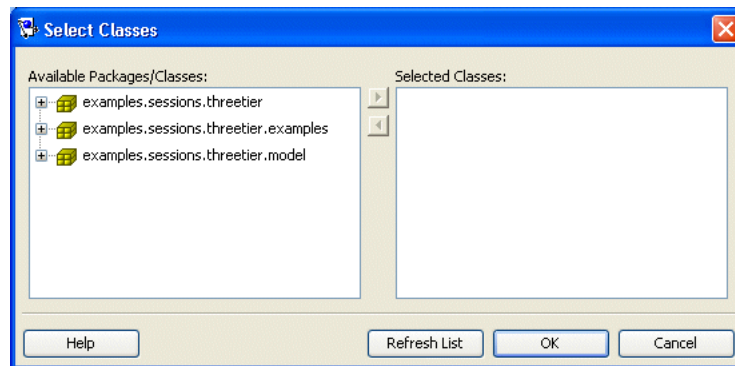
Use this procedure to update or refresh the classes in the TopLink Workbench project.

1. Define the available classes and packages for the project on the **General** tab. See ["Configuring Project Classpath"](#) on page 22-3 for information on classes and packages.



2. Click **Add or Refresh Class**. The Select Classes dialog box appears.

You can also update the classes by choosing **Selected > Add or Refresh Classes** from the menu.

Figure 4-48 Select Classes Dialog Box

Select the packages or classes (or both) to import into the project and click **OK**. TopLink Workbench adds the new classes to your project in the **Navigator**.

By default, TopLink Workbench creates the following descriptor types for each package and class (depending on your project type):

- Relational projects—Relational class descriptors (see ["Relational Class Descriptors"](#) on page 27-2)
- EIS projects—EIS composite descriptors (see ["EIS Composite Descriptors"](#) on page 27-5)
- XML projects—XML descriptors (see ["XML Descriptors"](#) on page 26-12)

See [Chapter 27, "Creating a Descriptor"](#) for more information.

Note: If the class exists on both the system classpath and the project classpath, TopLink Workbench will update the class from the system classpath. To update or refresh from the project classpath, remove the class from the system classpath and restart TopLink Workbench.

To Remove a Class from a Project:



Select the descriptor and click **Remove**, or choose **Selected > Remove** from the menu.

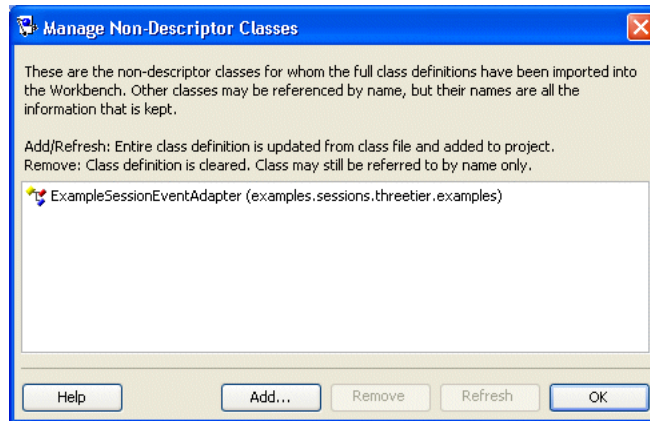
Managing Nondesoriptor Classes

Some of the mappings in your TopLink project may reference classes that do not have TopLink descriptors or are not included in the project.

To add, remove, or refresh Java classes that do not have TopLink descriptors, use this procedure:

From the menu, select **Workbench > Manage Non-Descriptor Classes**. The Manage Non-Descriptor Classes dialog box appears.

You can access the dialog box by right-clicking the TopLink project icon in the **Navigator** and selecting **Manage Non-Descriptor Classes** from the context menu.

Figure 4–49 Manage Non-Descriptor Classes Dialog Box

Select one of the following options:

- To add new classes, click **Add**. The Select Classes dialog box appears.
- To add new classes, click **Add**. The Select Classes dialog box appears (see [Figure 4–48](#) on page 4-51).

Only classes that have been added to the project's class path can be added as nondescriptor classes. See ["Configuring Project Classpath"](#) on page 22-3 for more information.

- To delete an existing class, select the class and click **Remove**.
- To refresh the classes (for example, if you edited the classes in an IDE), click **Refresh**.

Renaming Packages

When you add classes to a project, TopLink Workbench shows the classes contained in the package to which they belong (see ["Using the Navigator"](#) on page 4-9).

You can use TopLink Workbench to change the package statements in all the Java classes of a selected package (to move the all the classes contained by the selected package to a new package). This is useful if you are refactoring an existing TopLink Workbench project.

Note: The TopLink Workbench package rename feature is not intended for migrating projects from older versions of TopLink: for this, you must still use the TopLink Package Renamer. The Package Renamer updates import statements for TopLink classes: it does not change the package statements in user application classes.

For information on the TopLink Package Renamer, refer to *Oracle TopLink Release Notes* and *Oracle TopLink Getting Started Guide* for more information.

For more information on using TopLink Workbench to edit classes, see ["Configuring Classes"](#) on page 4-41.

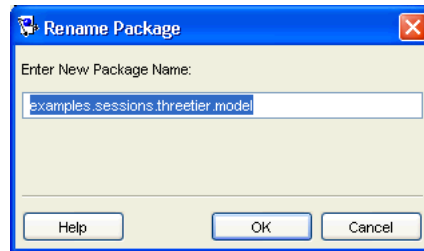
Using TopLink Workbench

To change the package of an existing class in TopLink Workbench, use this procedure:

1. Right-click the package in the **Navigator** and select **Rename**.

You can also select the package and choose **Selected > Rename** from the menu.

Figure 4–50 *Rename Package Dialog Box*



Enter the package name and click **OK**. TopLink Workbench changes the name of the package in the Navigator window.

For more information on using TopLink Workbench to edit classes, see "[Configuring Classes](#)" on page 4-41.

Integrating TopLink Workbench With Apache Ant

If you use the Apache Ant Java-based build tool, you can use the Ant task and type definitions that TopLink provides to invoke certain TopLink Workbench functions from an Ant build file. Using these tasks, you can integrate TopLink Workbench into your automated build process.

This section describes the following:

- [Configuring Ant to Use TopLink Workbench Tasks](#)
- [Understanding TopLink Workbench Ant Task API](#)
- [Creating TopLink Workbench Ant Tasks](#)

For more information about Ant, see <http://ant.apache.org/manual/>.

Configuring Ant to Use TopLink Workbench Tasks

Before you can use TopLink Workbench tasks in your Ant build files, you must consider their library dependencies (see "[Library Dependencies](#)" on page 4-53).

To declare TopLink Workbench tasks in your Ant `build.xml` file, declare them directly (see "[Declaring TopLink Workbench Tasks](#)" on page 4-54).

Library Dependencies

In addition to the Ant library dependencies (see <http://ant.apache.org/manual/install.html#librarydependencies>), [Table 4–4](#) lists the TopLink-specific JAR files that must be in your Ant classpath.

Table 4–4 *TopLink Workbench Ant Task Library Dependencies*

JAR Name	Needed For ...	Available At ...
toplinkmw.jar	TopLink Workbench Ant task and type definitions.	<TOPLINK_HOME>/jlib

Declaring TopLink Workbench Tasks

After you declare the TopLink Workbench task definitions (see [Table 4–6](#)) and data definitions (see [Table 4–4](#)) in the `toplink-ant-lib.xml` file as [Example 4–5](#) shows, you can use a TopLink Workbench task in a `build.xml` file as [Example 4–6](#) shows:

Example 4–5 Declaring TopLink Workbench Ant Task and Data Types in a `toplink-ant-lib.xml` File

```
<?xml version="1.0"?>
<antlib>
  <taskdef name="mappings.export"
    classname="oracle.toplink.workbench.ant.taskdefs.ExportDeploymentXMLTask" />

  <taskdef name="mappings.validate"
    classname="oracle.toplink.workbench.ant.taskdefs.MappingsValidateTask" />

  <taskdef name="session.validate"
    classname="oracle.toplink.workbench.ant.taskdefs.SessionValidateTask" />

  <typedef name="ignoreerror"
    classname="oracle.toplink.workbench.ant.typedefs.IgnoreError" />

  <typedef name="ignoreerrorset"
    classname="oracle.toplink.workbench.ant.typedefs.IgnoreErrorSet" />

  <typedef name="loginspec"
    classname="oracle.toplink.workbench.ant.typedefs.LoginSpec" />
</antlib>
```

Example 4–6 Specifying the `toplink-ant-lib.xml` File in the `build.xml` File

```
<project name="MyBuild" default="validate.session" basedir="." xmlns:toplink="toplinklib">
  <typedef file = "toplink-ant-lib.xml" classpathref = "mw.classpath" uri = "toplinklib" />
  ...
</project>
```

Understanding TopLink Workbench Ant Task API

[Table 4–5](#) lists the TopLink Workbench Ant task definitions that TopLink provides.

Table 4–5 TopLink Workbench Ant Task Definitions

Task Name	TopLink Class
mappings.validate	<code>oracle.toplink.workbench.ant.taskdefs.MappingsValidateTask</code>
session.validate	<code>oracle.toplink.workbench.ant.taskdefs.SessionValidateTask</code>
mappings.export	<code>oracle.toplink.workbench.ant.taskdefs.ExportDeploymentXMLTask</code>

[Table 4–6](#) lists the TopLink Workbench Ant type definitions that TopLink provides.

Table 4–6 TopLink Workbench Ant Type Definitions

Type Name	TopLink Class
ignoreerror	<code>oracle.toplink.workbench.ant.typedefs.IgnoreError</code>
ignoreerrorset	<code>oracle.toplink.workbench.ant.typedefs.IgnoreErrorSet</code>
loginspec	<code>oracle.toplink.workbench.ant.typedefs.LoginSpec</code>

Creating TopLink Workbench Ant Tasks

[Example 4-7](#) shows a typical `build.xml` file that declares and uses the TopLink Workbench Ant task and type definitions.

Example 4-7 Example Ant Build File with TopLink Workbench Ant Tasks

```
<project name="MyBuild" default="validate.session" basedir="." xmlns:toplink="toplinklib">
  <!-- ===== -->
  <!-- Properties -->
  <!-- ===== -->
  <target name="init">
    <property file="build.properties"/>

    <property name = "toplink.mwp.dir" value = "${basedir}/mw"/>
    <property name = "toplink.sessions.dir" value = "${basedir}/config"/>
    <property name = " myProject.classes" value = "${basedir}/classes "/>

    <path id = "database.classpath">
      <pathelement path = "${toplink.home}/jlib /dms.jar"/>
      <pathelement path = "${toplink.home}/jlib /OracleThinJDBC.jar"/>
    </path>
    <path id = "toplink.classpath">
      <pathelement path = "${toplink.home}/jlib /toplink.jar"/>
      <pathelement path = "${toplink.home}/jlib /ejb.jar"/>
      <pathelement path = "${toplink.home}/jlib /xmlparserv2.jar"/>
      <pathelement path = "${toplink.home}/jlib /antlr.jar"/>
    </path>
    <path id = "mw.classpath">
      <pathelement path = "${toplink.home}/jlib /tlmwcore.jar"/>
      <pathelement path = "${toplink.home}/jlib /toplinkmw.jar"/>
    </path>
    <path id = "mwplatforms.classpath">
      <pathelement path = "${toplink.home}/config"/>
    </path>

    <typedef file = "toplink-ant-lib.xml"

      classpathref = "mw.classpath"
      uri = "toplinklib" />
  </target>
  <!-- ===== -->
  <!-- Define task parameter -->
  <!-- ===== -->
  <target name="parameter.definition" depends="init">
    <toplink:ignoreerrorset id = "ignoreErrors">
      <toplink:ignoreerror code = "0233" />
    </toplink:ignoreerrorset>

    <toplink:loginspec id = "loginSpec"
      url = "jdbc:cloudscape:stagedb;create=true"
      driverclass = "COM.cloudscape.core.JDBCdriver"
      user = "scott"
      password="tiger" />
  </target>
  <!-- ===== -->
  <!-- Validate the MW Project -->
  <!-- ===== -->
  <target name="validate.project" depends="parameter.definition">

    <toplink:mappings.validate
      projectfile = "${toplink.mwp.dir}/myProject.mwp"
      reportfile = "${toplink.mwp.dir}/problem-report.html"
      reportformat = "html"
      property = "mw-valid"
      classpathref = "mwplatforms.classpath" >
```

```
<toplink:classpath refid = "mw.classpath" />
<toplink:classpath refid = "toplink.classpath" />

<toplink:ignoreerrorset refid = "ignoreErrors"/>

</toplink:mappings.validate>
</target>
<!-- ===== -->
<!-- TopLink deployment descriptor XML generation -->
<!-- ===== -->
<target name="export.deployment" depends="validate.project" if="mw-valid">

    <toplink:mappings.export
        projectfile = "${toplink.mwp.dir}/myProject.mwp"
        deploymentfile = "${toplink.sessions.dir}/sessions.xml"
        property = "export-completed"
        failonerror = "true"
        classpathref = "toplink.classpath">

        <toplink:classpath refid = "mw.classpath" />
        <toplink:classpath refid = "mwplatforms.classpath" />

        <toplink:ignoreerrorset refid = "ignoreErrors"/>
        <toplink:loginspec refid = "loginSpec" />
    </toplink:mappings.export>
</target>
<!-- ===== -->
<!-- TopLink Session Validate -->
<!-- ===== -->
<target name="validate.session" depends="export.deployment" if="export-completed">

    <toplink:session.validate
        sessionsfile = "${toplink.sessions.dir}/sessions.xml"
        sessionname = "ThreeTierEmployee"
        property = "session-valid"
        classpathref = "toplink.classpath"
        classpath = "${ myProject.classes}" >

        <toplink:classpath refid = "mw.classpath" />
        <toplink:classpath refid = " database.classpath" />

        <toplink:loginspec refid = "loginSpec" />
    </toplink:session.validate>
</target>
</project>
```

mappings.validate

The `mappings.validate` task is a testing task that you use to list of all the problems in a TopLink Workbench project (`.mwp`) file.

This task provides the ability to:

- log all the problems to a file in text or HTML format
- set an Ant property to indicate that the TopLink Workbench project is valid (has no errors)

Parameters

Table 4-7 *mappings.validate Task Parameters*

Attribute	Description	Required
projectfile	Fully qualified TopLink Workbench projects file name (.mwp).	Yes
reportfile	Fully qualified file name to which to write the output.	No
reportformat	The format of the generated output. Must be <code>html</code> or <code>text</code> .	No—default to <code>text</code> .
classpath	Project classpath.	No
classpathref	Reference to a path defined elsewhere.	No
property	The name of the property to set (true if there is no problem).	No

Parameters Specified as Nested Elements

You can specify the following parameters as nested elements of this task:

- `classpath`
- ["ignoreerror"](#) on page 4-60
- ["ignoreerrorset"](#) on page 4-61

Examples

[Example 4-8](#) shows a typical `mappings.validate` task.

Example 4-8 *A mappings.validate Task*

```
<toplink:mappings.validate
  projectfile = "${toplink.mwp.dir}/myProject.mwp"
  reportfile = "${toplink.mwp.dir}/problem-report.html"
  reportformat = "html"
  property = "mw-valid"
  classpath = "${mwplatforms.classpath}" >

  <toplink:classpath refid = "mw.classpath" />
  <toplink:classpath refid = "toplink.classpath" />

  <toplink:ignoreerrorset refid = "ignoreErrors"/>
  <toplink:ignoreerror code = "0555" />
</toplink:mappings.validate>
```

session.validate

The `session.validate` task is a testing task that you use to test your TopLink deployment XML by running TopLink.

This task provides the ability to:

- specify the test type using a nested element
- set an Ant property to indicate that the TopLink Workbench project is valid (has no errors)

Parameters

Table 4–8 *session.validate Task Parameters*

Attribute	Description	Required
sessionsfile	Fully qualified sessions.xml file.	No—default to sessions.xml and to classpath.
sessionname	Name of the session to test.	Yes
classpath	Project classpath.	No
classpathref	Reference to a path defined elsewhere.	No
property	The name of the property to set (true if valid).	No

Parameters Specified as Nested Elements

You can specify the following parameters as nested elements of this task:

- classpath
- ["loginspec"](#) on page 4-62

Examples

[Example 4–9](#) shows a typical session.validate task.

Example 4–9 *A session.validate Task*

```
<toplink:session.validate
  sessionsfile = "${toplink.sessions.dir}/sessions.xml"
  sessionname = "ThreeTierEmployee"
  property = "session-valid"
  classpathref = "toplink.classpath"
  classpath = "${ myProject.classes}" >

  <toplink:classpath refid = "mw.classpath" />
  <toplink:classpath refid = " database.classpath" />

  <toplink:loginspec refid = "loginSpec" />
</toplink:session.validate>
```

mappings.export

The mappings.export task is a generation task that you use to generate a TopLink deployment XML file for a given TopLink Workbench project (.mwp). The mappings.export task executes a mappings.validate (see ["mappings.validate"](#) on page 4-56) before executing. A BuildException is thrown if validation fails.

This task provides the ability to override the TopLink Workbench project database login information.

Parameters

Table 4–9 *mappings.export Task Parameters*

Attribute	Description	Required
projectfile	Fully qualified TopLink Workbench projects file name (.mwp).	Yes

Table 4–9 (Cont.) mappings.export Task Parameters

Attribute	Description	Required
deploymentfile	Fully qualified TopLink project deployment file name (.xml).	No—default to the name specified in the TopLink Workbench project (.mwp).
ejbjarxmlidir	The directory that contains the ejb-jar.xml file (only applicable to J2EE project).	No—default to the directory specified in the TopLink Workbench project (.mwp).
classpath	Project classpath.	No
classpathref	Reference to a path defined elsewhere.	No
failonerror	Indicates whether the build will continue even if there are export errors; defaults to true.	No
property	The name of the property to set (true if export completed successfully).	No

Parameters Specified as Nested Elements

You can specify the following parameters as nested elements of this task:

- classpath
- "loginspec" on page 4-62
- "ignoreerror" on page 4-60
- "ignoreerrorset" on page 4-61

Examples

[Example 4–9](#) shows a typical mappings.export task.

Example 4–10 A mappings.export Task

```
<toplink:mappings.export
  projectfile = "${toplink.mwp.dir}/myProject.mwp"
  deploymentfile = "${toplink.sessions.dir}/sessions.xml"
  property = "export-completed"
  failonerror = "true"
  classpathref = "toplink.classpath">

  <toplink:classpath refid = "mw.classpath" />
  <toplink:classpath refid = "mwplatforms.classpath" />

  <toplink:ignoreerrorset refid = "ignoreErrors"/>
  <toplink:ignoreerror code = "0545" />
  <toplink:loginspec
    url = "jdbc:cloudscape:stagedb;create=true"
    driverclass = "COM.cloudscape.core.JDBCdriver"
    user = "scott"
    password="tiger" />
</toplink:mappings.export>
```

classpath

Use the classpath element to define the Java classpath necessary to run a task. For more information, see <http://ant.apache.org/manual/using.html#path>.

Parameters

Table 4–10 *classpath Element Parameters*

Attribute	Description	Required
location	Specifies a single file or directory relative to the project's base directory (or an absolute filename).	No
path	Specifies one or multiple files or directories separated by a colon or semicolon.	No
refid	Reference to a path defined elsewhere.	No

Parameters Specified as Nested Elements

You can specify the following parameters as nested elements of this task:

- `pathelement`
- `fileset`
- `dirset`
- `filelist`

Examples

[Example 4–11](#) shows a typical `classpath` element.

Example 4–11 *A classpath Element*

```
<classpath>
  <pathelement path="{classpath}"/>
    <fileset dir="lib">
      <include name="**/*.jar"/>
    </fileset>
  <pathelement location="classes"/>
    <dirset dir="{build.dir}">
      <include name="apps/**/classes"/>
      <exclude name="apps/**/*Test*"/>
    </dirset>
  <filelist refid="third-party_jars"/>
</classpath>
```

ignoreerror

Use the `ignoreerror` element to instruct a TopLink Ant task to ignore a specific TopLink Foundation Library (see ["TopLink Exception Reference"](#) on page 13-1) or TopLink Workbench (see ["TopLink Workbench Error Reference"](#) on page 14-1) run-time error code.

To instruct a TopLink Ant task to ignore multiple error codes, consider using an `ignoreerrorset` element (see ["ignoreerrorset"](#) on page 4-61).

Parameters

Table 4–11 *ignoreerror Element Parameters*

Attribute	Description	Required
code	Error code of the problem to ignore.	Yes

Parameters Specified as Nested Elements

You cannot specify parameters as nested elements of this element.

Examples

[Example 4-12](#) shows a typical `ignoreerror` element. This element instructs a `mappings.export` task to ignore TopLink Workbench error code 0545.

Example 4-12 An `ignoreerror` Element

```
<toplink:mappings.export
  projectfile = "${toplink.mwp.dir}/myProject.mwp"
  deploymentfile = "${toplink.sessions.dir}/sessions.xml"
  classpathref = "toplink.classpath">

  <toplink:classpath refid = "mw.classpath" />
  <toplink:classpath refid = "mwplatforms.classpath" />

  <toplink:ignoreerror code = "0545" />

</toplink:mappings.export>
```

ignoreerrorset

Use the `ignoreerrorset` element to instruct a TopLink Ant task to ignore any of multiple TopLink Foundation Library (see ["TopLink Exception Reference"](#) on page 13-1) or TopLink Workbench (see ["TopLink Workbench Error Reference"](#) on page 14-1) run-time error codes.

Parameters

Table 4-12 *ignoreerrorset Element Parameters*

Attribute	Description	Required
id	Unique identifier for this type instance, can be used to reference this type in scripts.	No
refid	Reference to a <code>ignoreerrorset</code> defined elsewhere.	No

Parameters Specified as Nested Elements

You can specify the following parameters as nested elements of this element:

- ["ignoreerror"](#) on page 4-60

Examples

[Example 4-13](#) shows a typical `ignoreerrorset` element. This element instructs a `mappings.export` task to ignore all of TopLink Workbench error codes 0402 and 0570. Note that the `mappings.export` task also uses an explicitly `ignoreerror` element: this means that the `mappings.export` task will ignore all of error codes 0402, 0570, and 0545.

Example 4-13 An `ignoreerrorset` Element

```
<toplink:ignoreerrorset id = "ignoreErrors">
  <toplink:ignoreerror code = "0402" />
  <toplink:ignoreerror code = "0570" />
</toplink:ignoreerrorset>
...
<toplink:mappings.export
  projectfile = "${toplink.mwp.dir}/myProject.mwp"
  deploymentfile = "${toplink.sessions.dir}/sessions.xml"
  classpathref = "toplink.classpath">
```

```

<toplink:classpath refid = "mw.classpath" />
<toplink:classpath refid = "mwplatforms.classpath" />

<toplink:ignoreerrorset refid = "ignoreErrors"/>
<toplink:ignoreerror code = "0545" />

</toplink:mappings.export>

```

loginspec

Use the `loginspec` element to instruct a TopLink Ant task to override the project database login information in a TopLink Workbench project. For more information, see ["Understanding Data Access"](#) on page 84-1.

Note: You can only use this element with a relational project (see ["Relational Projects"](#) on page 20-6).

You cannot use this element with a J2EE project.

Parameters

Table 4-13 *loginspec Element Parameters*

Attribute	Description	Required
id	Unique identifier for this type instance, can be used to reference this type in scripts.	No
refid	Reference to a <code>loginspec</code> defined elsewhere.	No
driverclass	Fully qualified class of the data source driver (see "Configuring Database Login Connection Options" on page 86-2).	No—default to the class that the TopLink Workbench project specifies.
url	URL of the driver see "Configuring Database Login Connection Options" on page 86-2).	Yes
user	Login user name (see "Configuring User Name and Password" on page 85-1).	No—default to the value that the TopLink Workbench project specifies
password	Login password (see "Configuring User Name and Password" on page 85-1).	No—default to the value that the TopLink Workbench project specifies

Parameters Specified as Nested Elements

You cannot specify parameters as nested elements of this element.

Examples

[Example 4-14](#) shows a typical `loginspec` element.

Example 4-14 *A loginspec Element*

```

<toplink:mappings.export
  projectfile = "${toplink.mwp.dir}/myProject.mwp"
  deploymentfile = "${toplink.sessions.dir}/sessions.xml"
  classpathref = "toplink.classpath">

  <toplink:classpath refid = "mw.classpath" />
  <toplink:classpath refid = "mwplatforms.classpath" />

```

```
<toplink:loginspec
  url = "jdbc:cloudscape:stagedb;create=true"
  driverclass = "COM.cloudscape.core.JDBCdriver"
  user = "scott"
  password="tiger" />
</toplink:mappings.export>
```

Using an Integrated Development Environment

This chapter includes information on using TopLink with an integrated development environment (IDE). This chapter includes the following sections:

- [Configuring TopLink for Oracle JDeveloper](#)
- [Configuring TopLink Workbench With Source Control Management Software](#)

In addition to the development environment described here, TopLink can be used with *any* J2EE development environment and process.

Configuring TopLink for Oracle JDeveloper

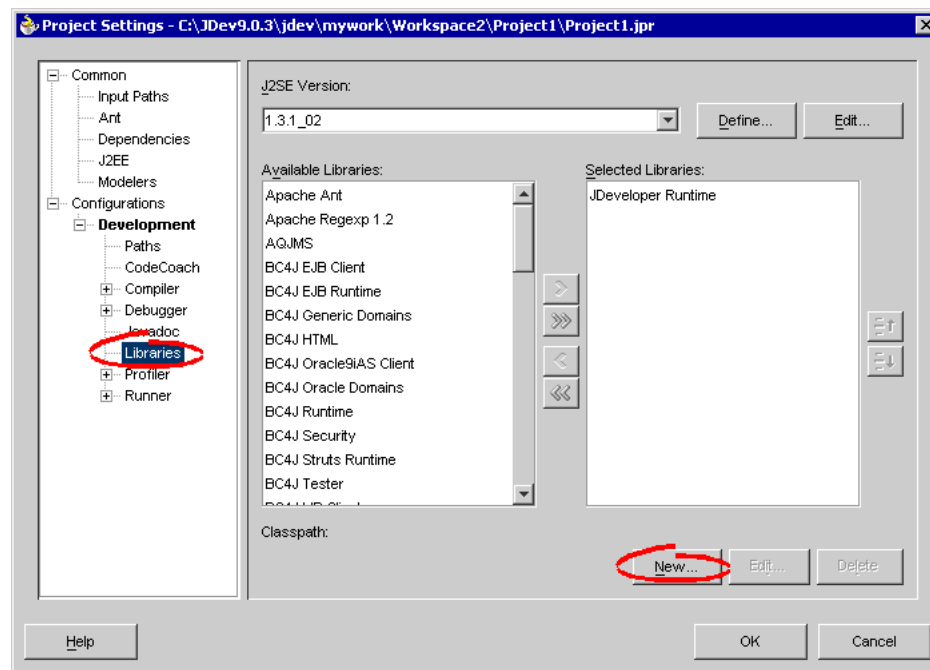
This section contains information on how to configure TopLink for use with Oracle JDeveloper. JDeveloper is a J2EE development environment with end-to-end support to develop, debug, and deploy e-business applications and Web services.

Using TopLink Mappings

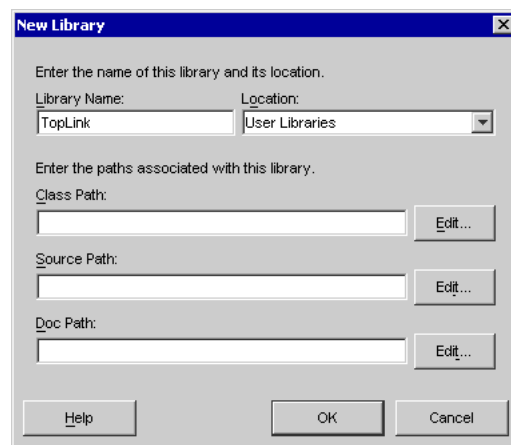
Starting with Oracle JDeveloper 10g, the standard JDeveloper installation includes an embedded TopLink editor. Refer to the JDeveloper documentation for complete information.

To use TopLink with JDeveloper 9.0.4 (and earlier), use the following procedure to add the TopLink JAR files to your JDeveloper projects:

1. Select a JDeveloper project in the **System Navigator** window.
2. Choose **Project > Project Settings**. The **Project Settings** window appears.
3. Choose **Configurations > Development > Libraries**. A list of predefined and user-defined libraries appears.

Figure 5–1 List of Available Libraries

4. Click **New** to create a new library that will contain the TopLink .jar files. The New Library dialog box appears.
5. Enter a name for the new library—for example, **TopLink**. Ensure that the default choice for Location remains as **User Libraries**.

Figure 5–2 Creating a New Library Dialog Box

6. To edit the **Class Path** and add the TopLink .jar files, click **Edit**.

Add the following to the beginning of your **Class Path**:

```
<ORACLE_HOME>\toplink\jlib\toplink.jar
<ORACLE_HOME>\toplink\jlib\antlr.jar
<ORACLE_HOME>\lib\xmlparserv2.jar
<ORACLE_HOME>\lib\xml.jar
```

7. Click **OK**. On the **Project Settings** window click **OK**.

Using an Existing User-Defined TopLink Library

After a user library is created, it can be referenced again by any other project. Revisit the **Libraries** window of the Project Settings, and add the TopLink Library to any project with which you want to use TopLink.

Using TopLink Sessions

When using TopLink 10g Release 3 (10.1.3) with Oracle JDeveloper, you should be aware that you cannot configure all `sessions.xml` options with the JDeveloper mapping editor.

To configure supported options (prior to 10g Release 3 (10.1.3)), use the JDeveloper mapping editor. For details, refer to the JDeveloper online help at:

http://www.oracle.com/technology/documentation/9i_jdev.html

To configure the new TopLink 10g Release 3 (10.1.3) options, including, Cache coordination options (see [Chapter 91, "Configuring a Coordinated Cache"](#)), Historical sessions (see [Chapter 81, "Configuring Historical Client Sessions"](#)), and Connection policy for server sessions (see ["Configuring Connection Policy"](#) on page 77-19) use one of the following methods:

- Use a `SessionEventListener.preLogin(SessionEvent)` method
- When accessing the session from the session manager, request it with the `loggedIn` option set to `false`. The returned session can then be customized and you can invoke login.

Configuring TopLink Workbench With Source Control Management Software

You can use TopLink Workbench with a source control management (SCM) system to facilitate enterprise-level team development (see ["Using a Source Control Management System"](#) on page 5-3). If you have a small development team, you can manage the changes from within XML files (see ["Sharing Project Objects"](#) on page 5-6).

When using a TopLink Workbench project in a team environment, you must synchronize your changes with other developers. See ["Merging Files"](#) on page 5-4 for more information.

Using a Source Control Management System

If you use an enterprise, file-based source control management system to manage your Java source files, you can use the same system with your TopLink Workbench project files. These project files are maintained by TopLink Workbench and written out in XML file format.

The *check in* and *check out* mechanism for the source control system defines how to manage the source (the XML source and TopLink Workbench project file) in a multiuser environment.

Checking Out and Checking In TopLink Workbench Project Files

Although your actual development process will vary depending on your SCM tool, a typical process involves the following steps:

1. Determine (based on your SCM system) which files to retrieve from the source management system.
2. Edit the project using TopLink Workbench.

3. Save the edited project. If TopLink Workbench displays the Read-Only Files dialog box, make a note of these files, they must be unlocked and possibly merged. See ["Working With Locked Files"](#) on page 5-6 for more information.
4. Merge the required project files. See ["Merging Files"](#) on page 5-4 for details.
5. Check in the modified files, then retrieve from the repository any files that have been added or modified for this TopLink Workbench project.

Merging Files

The most difficult aspect of application development is merging changes from two (or more) development team members that have simultaneously edited the same file. If one developer checks in his or her changes, a *merge* condition exists. Use a file comparison tool to determine the merged aspects of the project. The files to edit will vary, depending on the type of merge:

- [Merging Project Files](#)
- [Merging Table, Descriptor, and Class Files](#)

[Example 5–1](#) and [Example 5–2](#) demonstrate a *merge out* merging technique.

Merging Project Files

Project files contain references to the objects in the project. Generally, your project `<projectName>.mwp` contains the following elements:

- Database information: `<database>`
 - Database tables: `<tables>`
- Descriptors: `<descriptors>`
- Repository: `<repository>`
 - Classes: `<classpath-entries>`

Changes in these parts of the `.mwp` file are usually caused by adding, deleting, or renaming project elements.

To merge project files, you will generally need to merge a project file if another developer has added or removed a descriptor, table, or class, and checked in the project while you were adding or removing descriptors, tables, or classes from the same project. To merge the project's `.mwp` file, use this procedure:

1. Perform a file comparison between the `<projectName>.mwp` file in the repository and your local copy. The file comparison shows the addition or removal of a project element inside the owner (that is, `<database>`, `<descriptors>`, or `<repository>`).
2. Insert the XML script to, or delete from your local `<projectName>.mwp` file (inside the *corresponding owner element*). This brings your local code up-to-date to the current code in the code repository.
3. Retrieve any updated files, as indicated by your source control system.

Your local source now matches the repository.

Example 5–1 Merging Projects

Another developer has added and checked in a new **Employee** class descriptor to the `com.demo` package while you were working with the same TopLink Workbench project. To merge your work with the newly changed project, follow these steps:

1. Perform a file comparison on the `<projectName>.mwp` file to determine the differences between your local file and the file in the repository. Your SCM system may show the file in *merge* status.

The file comparison shows the addition of the `<package-descriptor>` tag and a `<name>` element inside that tag:

```
<package-descriptor>
  <name>com.demo.Employee.ClassDescriptor</name>
</package-descriptor>
```

2. Insert this XML into your `<projectName>.mwp` file (inside the `<descriptors>` element) to bring it up-to-date with the current files in the source repository.
3. Retrieve any new or updated files from your source control system. This includes the newly added **Employee** class descriptor.
4. Check in files that you have modified.

Merging Table, Descriptor, and Class Files

Developers who concurrently modify the same existing table, descriptor, or class file will create a merge condition for the following files:

- Table: `<tableName>.xml` (one for each table)
- Descriptor: `<descriptorName.type>.xml` (one for each descriptor)
- Class: `<className>.xml` (one for each class)

TopLink Workbench changes these files when saving a project if you have changed any of the contents within them (such as adding a mapping to a descriptor, adding an attribute to a class, or a changing a field reference in a table).

If another developer has changed an attribute in a table, descriptor, or class, while you were changing a different mapping on that same descriptor, you will need to merge your project. To merge your project, use this procedure:

1. Perform a file comparison on the specific `.xml` files in merge status (that is, table, descriptor, or class). The file comparison shows the addition or removal of an XML element.
2. Insert the XML script to, or remove from your local `.xml` file to bring it up-to-date with the current files in the source repository.

Example 5-2 Merging Files

Another developer has added and checked in the **firstName** mapping to the **Employee** class descriptor while you were changing a different mapping on that same descriptor. To merge your work with the newly changed project, use this procedure:

1. Perform a file comparison on the `com.demo.Employee.ClassDescriptor.xml` file located in `<projectRoot>/Descriptor/` directory that is in merge status.

The file comparison shows the addition of the `<mapping>` tag and the elements inside that tag:

```
<mapping>
  <uses-method-accessing>false</uses-method-accessing>
  <inherited>false</inherited>
  <read-only>false</read-only>
  <instance-variable-name>firstName</instance-variable-name>
  <default-field-names>
```

```
        <default-field-name>direct field=</default-field-name>
    </default-field-names>
    <field-handle>
        <field-handle>
            <table>EMPLOYEE</table>
            <field-name>F_NAME</field-name>
        </field-handle>
    </field-handle>
<mapping-class>MWDirectToFieldMapping </mapping-class>
</mapping>
```

2. Insert this XML block into your local `com.demo.Employee.ClassDescriptor.xml` file (inside the existing `<mapping>` element) to bring it up to date to the current files in the source repository.
3. Retrieve any new files noted as missing by your source control system. This includes any tables or descriptors that may be referenced by the new mapping.
4. Check in files that you have modified.

Sharing Project Objects

You can also share project objects by copying the table or descriptor files into the appropriate directories in the target project.

After copying the files, insert a reference to the table, descriptor, or class in the appropriate place in the `<projectName>.mwp` file. All references contained within the project file must refer to an existing object in the project.

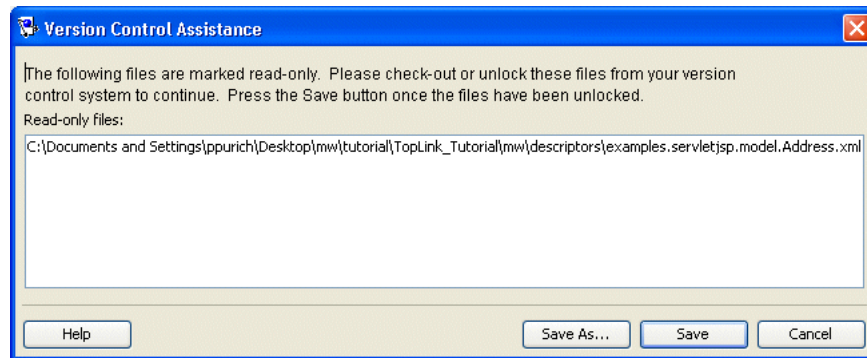
Managing the `ejb-jar.xml` File

When working in a team environment, manage the `ejb-jar.xml` file similarly to the `.xml` project files. TopLink Workbench edits and updates the `ejb-jar.xml` file, if necessary, when working with an EJB project.

If you use a version control system, perform the same check in and check out procedures. For merge conditions, use a file comparison tool to determine which elements have been added or removed. Modify the file as necessary and check in the file to exercise version control on your work.

Working With Locked Files

When working in a team environment, your source control system may lock files when you retrieve them from the repository. If TopLink Workbench attempts to save a locked file, the Version Control Assistance dialog box appears, showing the locked files.

Figure 5-3 Version Control Assistance Dialog Box

Select one of the following methods to save your project:

- Use your source control system to unlock the files, and then click **Save**.
- Click **Save As** to save the project to a new location.

See "[Saving Projects](#)" on page 21-11 for more information.

Using the Schema Manager

The `SchemaManager` and its related classes provide API that you can use from a Java application to specify database tables in a generic format, and then create and modify them in a specific relational database. This decouples your TopLink project from a particular database schema while giving you a programmatic means of creating a database schema based on your TopLink project. For example, you can use the schema manager to recreate a production database in a nonproduction environment. This lets you build models of your existing databases, and modify and test them during development.

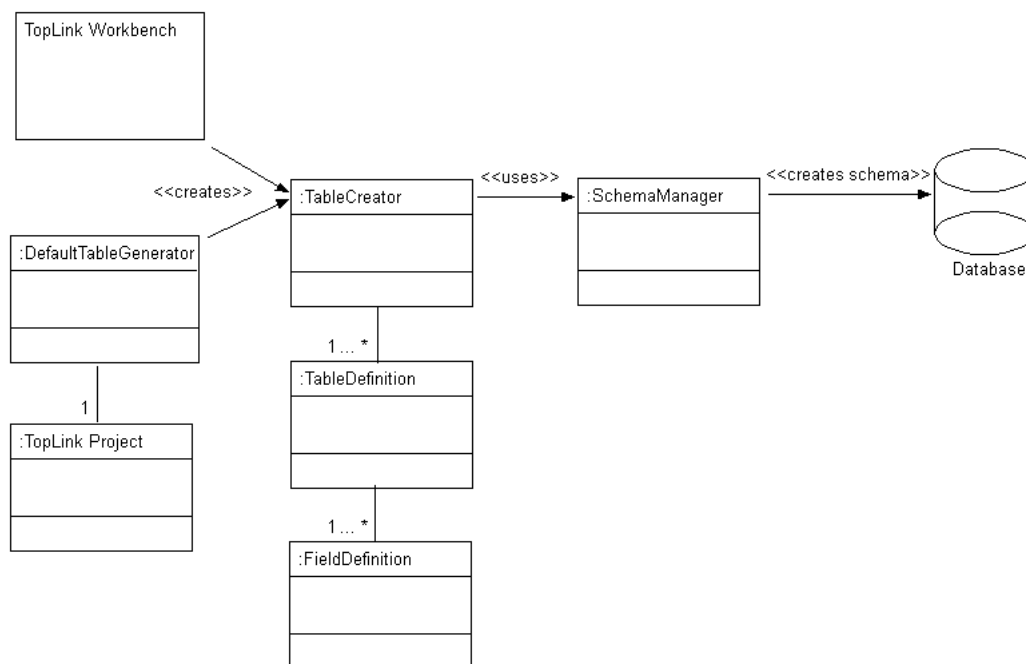
Note: You can also create database tables manually during development using TopLink Workbench (see ["Creating New Tables"](#) on page 4-22 and ["Generating Tables on the Database"](#) on page 4-32).

This chapter includes information on the following topics:

- [Understanding the Schema Manager](#)
- [Creating a Table Creator](#)
- [Creating Tables With a Table Creator](#)
- [Automatic Database Table Creation](#)

Understanding the Schema Manager

[Figure 6-1](#) summarizes the important `SchemaManager` classes and the primary means of using them.

Figure 6-1 SchemaManager Usage

Although you can use the `SchemaManager` API directly, Oracle recommends that you create a `TableCreator` class and use its API (which, in turn, uses the `SchemaManager`).

You can automatically generate a `TableCreator` using:

- TopLink Workbench during development (see ["Using TopLink Workbench During Development"](#) on page 6-4)
- `DefaultTableGenerator` at run time (see ["Using the Default Table Generator at Run Time"](#) on page 6-4)

The `TableCreator` class owns one or more `TableDefinition` classes (one for each database table) and the `TableDefinition` class owns one or more `FieldDefinition` classes (one for each field).

The `TableDefinition` class lets you specify a database table schema in a generic format. At run time, TopLink uses the session associated with your TopLink project to determine the specific database type, and uses the generic schema to create the appropriate tables and fields for that database.

After creating a `TableCreator` class, you can use its API to create and drop tables (see ["Creating Tables With a Table Creator"](#) on page 6-6). You can also configure TopLink to do this automatically (see ["Automatic Database Table Creation"](#) on page 6-6).

Because the schema manager uses Java types rather than database types, it is database-independent. However, because it does not account for database-specific optimizations, it is best-suited for development purposes rather than production. For more information on how the schema manager maps Java types to database types, see ["Schema Manager Java and Database Type Conversion"](#) on page 6-3.

Although the schema manager can handle the sequencing configuration that you specify in your TopLink project, if you are using sequencing with non-Oracle databases, there are some sequencing restrictions you should be aware of (see ["Sequencing"](#) on page 6-3).

Schema Manager Java and Database Type Conversion

[Table 6–1](#) lists the Java type to database type conversions that the schema manager supports depending on the database platform your TopLink project uses. This list is specific to the schema manager and does not apply to mappings. TopLink automatically performs conversions between any database types within mappings.

Table 6–1 Java and Database Field Type Conversion

Java Type	Oracle	DB2	Sybase	MySQL	MS Access
<code>java.lang.Boolean</code>	NUMBER	SMALLINT	BIT default 0	TINYINT(1)	SHORT
<code>java.lang.Byte</code>	NUMBER	SMALLINT	SMALLINT	TINYINT	SHORT
<code>java.lang.Byte[]</code>	LONG RAW	BLOB	IMAGE	BLOB	LONGBINARY
<code>java.lang.Character</code>	CHAR	CHAR	CHAR	CHAR	TEXT
<code>java.lang.Character[]</code>	LONG	CLOB	TEXT	TEXT	LONGTEXT
<code>java.lang.Double</code>	NUMBER	FLOAT	FLOAT(32)	DOUBLE	DOUBLE
<code>java.lang.Float</code>	NUMBER	FLOAT	FLOAT(16)	FLOAT	DOUBLE
<code>java.lang.Integer</code>	NUMBER	INTEGER	INTEGER	INTEGER	LONG
<code>java.lang.Long</code>	NUMBER	INTEGER	NUMERIC	BIGINT	DOUBLE
<code>java.lang.Short</code>	NUMBER	SMALLINT	SMALLINT	SMALLINT	SHORT
<code>java.lang.String</code>	VARCHAR2	VARCHAR	VARCHAR	VARCHAR	TEXT
<code>java.math.BigDecimal</code>	NUMBER	DECIMAL	NUMERIC	DECIMAL	DOUBLE
<code>java.math.BigInteger</code>	NUMBER	DECIMAL	NUMERIC	BIGINT	DOUBLE
<code>java.sql.Date</code>	DATE	DATE	DATETIME	DATE	DATETIME
<code>java.sql.Time</code>	DATE	TIME	DATETIME	TIME	DATETIME
<code>java.sql.Timestamp</code>	DATE	TIMESTAMP	DATETIME	DATETIME	DATETIME

For more information about database platforms that TopLink supports, see ["Database Platforms"](#) on page 84-3.

Sequencing

If you generate a `TableCreator` class using TopLink Workbench (see ["Using TopLink Workbench During Development"](#) on page 6-4) or `DefaultTableGenerator` (see ["Using the Default Table Generator at Run Time"](#) on page 6-4), then sequencing configuration is included in your `TableCreator` according to your TopLink project configuration. In this case, when you use `TableCreator` method `createTables`, it does the following:

- Creates the sequence table as defined in the session `DatabaseLogin`
- Creates or inserts sequences for each sequence name for all registered descriptors in the session
- Creates the Oracle sequence object if you use Oracle native sequencing

You can use advanced API to handle special cases like Sybase or Microsoft SQL Server native sequencing (see ["Using Java"](#) on page 6-4).

For more information about sequencing, see ["Understanding Sequencing in Relational Projects"](#) on page 20-14.

Creating a Table Creator

You can automatically generate a `TableCreator` using:

- `TopLink Workbench` during development (see ["Using TopLink Workbench During Development"](#) on page 6-4)
- `DefaultTableGenerator` at run time (see ["Using the Default Table Generator at Run Time"](#) on page 6-4)

After creating a `TableCreator` class, you can use its API to create and drop tables (see ["Creating Tables With a Table Creator"](#) on page 6-6).

Using TopLink Workbench During Development

To create a `TableCreator` class that you can use in a Java application to recreate a database schema using the `SchemaManager`, use this procedure:

1. Right-click the project in the **Navigator** and choose **Export > Table Creator Java Source** from the context menu. The Table Creator dialog box appears.

You can also select the table and choose **Selected > Export > Table Creator Java Source** from the menu.

2. Enter a name for the table creator class and click **OK**. The Save As dialog box appears.
3. Choose a location for your table creator class and click **OK**. TopLink Workbench exports the table creator Java class to the location you specify.

Using the Default Table Generator at Run Time

To create a `TableCreator` class in Java using the `DefaultTableGenerator`, use this procedure:

1. Create an instance of `DefaultTableGenerator`, passing in an instance of your TopLink project:

```
DefaultTableGenerator myDefTblGen = new DefaultTableGenerator(toplinkProject);
```

2. Create a `TableCreator` instance:

- If you want a `TableCreator` that can support any session, use:

```
TableCreator myTblCre = myDefTblGen.generateDefaultTableCreator();
```

- If you want a `TableCreator` customized for a specific TopLink session, use:

```
TableCreator myTblCre =  
myDefTblGen.generateFilteredDefaultTableCreator(toplinkSession);
```

You can also configure TopLink to use the `DefaultTableGenerator` to automatically generate and execute a `TableCreator` at run time (see ["Automatic Database Table Creation"](#) on page 6-6).

Using Java

This section describes how to create a `TableCreator` class in Java, including the following:

- [Creating a TableCreator Class](#)
- [Creating a TableDefinition Class](#)

- [Adding Fields to a TableDefinition](#)
- [Defining Sybase and Microsoft SQL Server Native Sequencing](#)

Creating a TableCreator Class

To create your own TableCreator instance, you should extend TableCreator as [Example 6-1](#) shows:

Example 6-1 Creating a TableCreator Class

```
public class MyTableCreator extends oracle.toplink.tools.schemaframework.TableCreator {

    public M7TableCreator() {
        setName("MyTableCreator");
        addTableDefinition(buildADDRESSTable());
        ...
    }

    public TableDefinition buildADDRESSTable() {
        TableDefinition table = new TableDefinition();
        ...
        return table;
    }
    ...
}
```

Creating a TableDefinition Class

The TableDefinition class includes all the information required to create a new table, including the names and properties of a table and all its fields.

The TableDefinition class has the following methods:

- setName
- addField
- addPrimaryKeyField
- addIdentityField
- addForeignKeyConstraint

All table definitions must call the setName method to set the name of the table that is described by the TableDefinition.

Adding Fields to a TableDefinition

Use the addField method to add fields to the TableDefinition. To add the primary key field to the table, use the addPrimaryKeyField method rather than the addField method.

To maintain compatibility among different databases, the type parameter requires a Java class rather than a database field type. TopLink translates the Java class to the appropriate database field type at run time. For example, the String class translates to the CHAR type for dBase databases. However, if you are connecting to Sybase, the String class translates to VARCHAR. For more information, see ["Schema Manager Java and Database Type Conversion"](#) on page 6-3.

The addField method can also be called with the fieldSize or fieldSubSize parameters for column types that require size and subsize to be specified.

Some databases require a subsize, but others do not. TopLink automatically provides the required information, as necessary.

Defining Sybase and Microsoft SQL Server Native Sequencing

Use `FieldDefinition` method `addIdentityField` to add fields representing a generated sequence number from Sybase or Microsoft SQL Server native sequencing. See ["Native Sequencing With a Non-Oracle Database Platform"](#) on page 20-19 for detailed information on using sequencing.

Creating Tables With a Table Creator

After creating a `TableCreator` class (see ["Creating a Table Creator"](#) on page 6-4), you can use its API to create and drop tables. The important `TableCreator` methods are as follows (each method takes an instance of `DatabaseSession`):

- `createTables`—this method creates tables, adds constraints, and creates sequence tables and sequences (if sequence tables already exist, this method drops them and recreates them).
- `dropTables`—his method drops all constraints and drops all tables (except sequence tables) that the `TableCreator` defines.
- `createConstraints`—this method creates constraints on all pre-existing tables that the `TableCreator` defines.
- `dropConstraints`—this method drops constraints on all pre-existing tables that the `TableCreator` defines.
- `replaceTables`—this method drops and then creates all tables that the `TableCreator` defines.

Automatic Database Table Creation

If you deploy an EJB 2.0 or 3.0 CMP project to OC4J configured to use TopLink as the persistence manager, then you can configure OC4J to automatically create (and, optionally, delete) database tables for your persistent objects.

You can configure automatic database table creation at one of three levels as [Table 6-2](#) shows. You can override the system level configuration at the application level and you can override system and application configuration at the EJB module level.

Table 6-2 Configuring Automatic Table Generation

Level	Configuration File	Setting	Values
System (global)	<OC4J_HOME>/config/application.xml	autocreate-tables	True ¹ or False
		autodelete-tables	True or False ¹
Application (EAR)	orion-application.xml	autocreate-tables	True ¹ or False
		autodelete-tables	True or False ¹
EJB Module (JAR)	orion-ejb-jar.xml	pm-properties sub-element default-mapping attribute db-table-gen ²	Create, DropAndCreate, or UseExisting ³

¹ Default.

² For more information, see ["Configuring default-mapping Properties"](#) on page 8-12.

³ See [Table 6-3](#).

If you configure automatic table generation at the EJB module level, the value you assign to the `db-table-gen` attribute corresponds to the `autocreate-tables` and `autodelete-tables` settings as [Table 6-3](#) shows.

Table 6-3 *Equivalent Settings for `db-table-gen`*

<code>db-table-gen</code> Setting	<code>autocreate-tables</code> Setting	<code>autodelete-tables</code> Setting
Create	True	False
DropAndCreate	True	True
UseExisting	False	NA

You can use this feature in conjunction with default mapping (see ["Default Mapping in EJB 2.0 or 3.0 CMP Projects Using OC4J at Run Time"](#) on page 33-4).

Part III

Deploying a TopLink Application

This part describes how to package and deploy a TopLink application to an application server. It contains the following chapters.

- [Chapter 7, "Integrating TopLink With an Application Server"](#)

This chapter contains information on software requirements for integrating TopLink with your specific application server.

- [Chapter 8, "Creating TopLink Files for Deployment"](#)

This chapter describes how to create the necessary TopLink files for deployment to your application server.

- [Chapter 9, "Packaging a TopLink Application"](#)

This chapter explains how to package the deployment files.

- [Chapter 10, "Deploying a TopLink Application"](#)

This chapter provides procedures for deploying different types of TopLink applications in a variety of environments.

Integrating TopLink With an Application Server

This chapter describes how to configure Oracle TopLink for use with J2EE containers and application servers. It includes sections on the following:

- [Application Server Support](#)
- [Application Server Integration Concepts](#)

For more information, see the following:

- ["Creating TopLink Files for Deployment"](#) on page 8-1
- ["Packaging a TopLink Application"](#) on page 9-1
- ["Deploying a TopLink Application"](#) on page 10-1

Application Server Support

TopLink can be used with any J2EE application server.

[Table 7-1](#) lists the application servers for which TopLink provides special EJB and CMP integration.

Table 7-1 TopLink Integration Support by Application Server Type

Application Server Type	TopLink Integration for Application Server Version	TopLink Integration for EJB Version
"Oracle Containers for J2EE (OC4J)" on page 7-5	■ 10.1.3	■ 1. <i>n</i> , 2. <i>n</i> , 3.0 ¹
	■ 10.1.2	■ 1. <i>n</i> , 2. <i>n</i>
	■ 9.0.4	■ 1. <i>n</i> , 2. <i>n</i>
	■ 9.0.3	■ 1. <i>n</i> , 2. <i>n</i>
"BEA WebLogic Server" on page 7-14	■ 9.0	■ 1. <i>n</i> , 2. <i>n</i>
	■ 8.1	■ 1. <i>n</i> , 2. <i>n</i>
	■ 7.0 (Service Pack 2)	■ 1. <i>n</i> , 2. <i>n</i>
	■ 6.1 (Service Pack 4)	■ 1. <i>n</i> , 2. <i>n</i>
"IBM WebSphere Application Server" on page 7-21	■ 6.0	■ 1. <i>n</i> , 2. <i>n</i>
	■ 5.1	■ 1. <i>n</i> , 2. <i>n</i>
	■ 5.0	■ 1. <i>n</i> , 2. <i>n</i>
	■ 4.0	■ 1. <i>n</i> , 2. <i>n</i>

¹ In this release, when using OC4J and Java 1.5, TopLink supports a subset of the CMP features anticipated in the final EJB 3.0 specification. EJB 3.0 feature support is subject to change and dependent upon the contents of the final specification. For more information on EJB 3.0 support, see Oracle Containers for J2EE Enterprise JavaBeans Developer's Guide.

Application Server Integration Concepts

This section describes concepts unique to TopLink application server integration, including the following:

- [Software Requirements](#)
- [XML Parser Platform Configuration](#)
- [Security Permissions](#)
- [Persistence Manager Migration](#)
- [Clustering](#)

Software Requirements

To run a TopLink application within a J2EE container, your system must meet the following software requirements:

- An application server or J2EE container (see [Table 7-1](#))
- XML parser (see ["XML Parser Platform Configuration"](#) on page 7-2)
- A JDBC driver configured to connect with your local database system (for more information, see your database administrator)
- A Java development environment, such as:
 - Oracle JDeveloper
 - IBM WebSphere Studio Application Developer (WSAD)
 - Sun Java Development Kit (JDK) 1.3.1 or later. Oracle recommends using 1.4.2 (or later).
- Any other Java environment that is compatible with the Sun JDK 1.3.1 or later
- A command-line JVM executable (such as `java.exe` or `jre.exe`)

XML Parser Platform Configuration

The TopLink run-time environment uses an XML parser to do the following:

- Read and write XML configuration files (see ["project.xml File"](#) on page 8-2 and ["sessions.xml File"](#) on page 8-3)
- Read and write TopLink Workbench project files (see ["Understanding TopLink Workbench"](#) on page 4-1)
- Perform object-to-XML transformations in EIS projects using XML records (see ["Understanding EIS Mappings"](#) on page 56-1)
- Perform object-to-XML transformations in XML projects (see ["Understanding XML Mappings"](#) on page 65-1)

Application servers use an XML parser to read deployment files such as `ejb-jar.xml` and `<J2EE container>-ejb-jar.xml` (see ["Creating TopLink Files for Deployment"](#) on page 8-1).

To avoid XML parser conflicts, you must configure your TopLink application to use the same XML parser as that used by the application server on which you deploy your application.

Internally, TopLink accesses its XML parser using an instance of `oracle.toplink.platform.xml.XMLPlatform` class.

You can configure TopLink to use any XML parser for which an `XMLPlatform` class exists (see "[Configuring XML Parser Platform](#)" on page 7-3).

You can also create your own `XMLPlatform` to provide access to an XML parser not currently supported by TopLink (see "[Creating an XML Parser Platform](#)" on page 7-3).

Configuring XML Parser Platform

TopLink provides the `XMLPlatform` instances shown in [Table 7-2](#).

Table 7-2 Supported XML Platforms

XMLPlatform...	Provides Access too...	Use With...
<code>oracle.toplink.platform.xml.xdk.XDKPlatform</code> ¹	XDKParser: this class provides access to the Oracle XML Developer's Kit (XDK) XML parser (see http://www.oracle.com/technology/tech/xml/xdkhome.html).	Oracle Containers for J2EE (OC4J)
<code>oracle.toplink.platform.xml.jaxp.JAXPPlatform</code>	JAXPParser: this class provides access to the Java SDK XML parser in the <code>javax.xml.parsers</code> package (see http://java.sun.com/j2ee/1.4/docs/tutorial/doc/JAXPIntro2.html).	BEA WebLogic Server IBM WebSphere Application Server

¹ Default:

Note: To use an XML parser not listed in [Table 7-2](#), create your own `XMLPlatform` (see "[Creating an XML Parser Platform](#)" on page 7-3).

To configure your TopLink application to use a particular instance of the `XMLPlatform` class, set system property `toplink.xml.platform` to the fully qualified name of your `XMLPlatform` class as [Example 7-1](#) shows.

Example 7-1 Configuring XML Platform

```
toplink.xml.platform=oracle.toplink.platform.xml.jaxp.JAXPPlatform
```

Creating an XML Parser Platform

Using the `oracle.toplink.platform.xml` classes included in the public source files shipped with TopLink (see "[Using Public Source](#)" on page 12-2), you can create your own instance of the `oracle.toplink.platform.xml.XMLPlatform` class to specify an XML parser not listed in [Table 7-2](#).

After creating your `XMLPlatform`, configure TopLink to use it (see "[Configuring XML Parser Platform](#)" on page 7-3).

XML Parser Limitations

Crimson (<http://xml.apache.org/crimson/>) is the XML parser supplied in the Java 2 Platform, Standard Edition (J2SE) and in some JAXP reference implementations.

If you use Crimson with the JAXP API to parse XML files whose system identifier is not a fully qualified URL, then XML parsing will fail with a *not valid URL* exception.

Other XML parsers defer validation of the system identifier URL until it is specifically referenced.

If you are experiencing this problem, consider one of the following alternatives:

- Ensure that your XML files use a fully qualified system identifier URL.
- Use another XML parser (such as the OracleAS XML Parser for Java v2).

Security Permissions

By default, when you run a TopLink-enabled application in a JVM configured with a nondefault `java.lang.SecurityManager`, the TopLink run-time environment executes certain internal functions by executing a `PrivilegedAction` with `java.security.AccessController` method `doPrivileged`. This ensures that you do not need to grant many permissions to TopLink for it to perform its most common operations. You need only grant certain permissions depending on the types of optional TopLink features you use.

For more information, see ["Understanding Security Permissions"](#) on page 7-22.

If you run a TopLink-enabled application in a JVM without a nondefault `SecurityManager`, you do not need to set any permissions.

Persistence Manager Migration

From the perspective of an application server, TopLink is a persistence manager. You can configure an application server to use TopLink as the default persistence manager.

You can only use one persistence manager for all the CMP EJB in a JAR file.

TopLink provides automated support for migrating an existing J2EE application to use TopLink as the persistence manager. For more information, see the following:

- ["Migrating OC4J Orion Persistence to OC4J TopLink Persistence"](#) on page 7-5
- ["Migrating BEA WebLogic Persistence to OC4J TopLink Persistence"](#) on page 7-16

Clustering

Most application servers include a clustering service that you can use with your TopLink application. To use TopLink with an application server cluster, use this procedure:

1. Install the `toplink.jar` file (and include it in the classpath) on each application server in the cluster to which you deploy TopLink applications.
2. Configure TopLink cache consistency options appropriate for your application.

For more information, see ["Understanding the Cache"](#) on page 90-1.

If you are deploying a CMP application, see also ["Configuring cache-synchronization Properties"](#) on page 8-11.

3. Configure clustering on each application server.

For more information, see your application server documentation.

Oracle Containers for J2EE (OC4J)

To integrate a TopLink application with OC4J, you must consider the following:

- [CMP Integration](#)
- [Migrating OC4J Orion Persistence to OC4J TopLink Persistence](#)
- [JTA Integration](#)

In addition to configuring these OC4J specific options, you must also consider the general application server integration issues in "[Application Server Integration Concepts](#)" on page 7-2.

CMP Integration

To enable TopLink CMP integration in OC4J, use the following procedure (this procedure assumes you have already installed TopLink):

1. If necessary, migrate your CMP application using the TopLink migration tool (see "[Migrating OC4J Orion Persistence to OC4J TopLink Persistence](#)" on page 7-5).
2. Evaluate your choice of `UnitOfWork` change policy (see "[Unit of Work and Change Policy](#)" on page 100-6).

If you are using EJB 3.0, you can let TopLink configure your persistent classes with the most efficient change policy (see "[Attribute Change Tracking Policy](#)" on page 100-8) at class loading time using byte code weaving.

3. Ensure that all necessary deployment descriptor files are in place (see "[Creating TopLink Files for Deployment](#)" on page 8-1 and "[Packaging a TopLink Application](#)" on page 9-1).

If you are using EJB 3.0, you can use annotations to specify most of what you formerly specified in deployment descriptors. Use deployment descriptors to override annotations or specify options not supported by annotations. For more information on what annotations are currently supported, see *Oracle Containers for J2EE Enterprise JavaBeans Developer's Guide*.

4. Optionally, consider the EJB customization options that TopLink provides (see "[Configuring Miscellaneous EJB Options](#)" on page 7-25).

Migrating OC4J Orion Persistence to OC4J TopLink Persistence

In 10g Release 3 (10.1.3), OC4J is shipped configured to use TopLink as its default persistence manager.

If you upgrade your OC4J to this release, you must migrate persistence configuration from your original `orion-ejb-jar.xml` file to the `toplink-ejb-jar.xml` file.

In this release, Oracle provides a TopLink migration tool that you can use to automate this migration for Release 2 (9.0.4) or later OC4J installations.

After using the TopLink migration tool, you may need to make some additional changes as described in "[Post-Migration Changes](#)" on page 12.

If you encounter problems during migration, see "[Troubleshooting Your Migration](#)" on page 7-13.

This section explains how to use the TopLink migration tool, including:

- [Overview](#)
- [Using the TopLink Migration Tool from TopLink Workbench](#)

- [Using the TopLink Migration Tool From the Command Line](#)
- [Post-Migration Changes](#)
- [Troubleshooting Your Migration](#)

Overview

Before using the TopLink migration tool, review this section to understand how the TopLink migration tool works and to determine what OC4J persistence manager metadata is, and is not, migrated.

Input and Output

The TopLink migration tool takes the following files as input:

- `ejb-jar.xml`
- `orion-ejb-jar.xml`

It migrates as much OC4J-specific persistence configuration as possible to a new `toplink-ejb-jar.xml` file and creates the following new files in a target directory you specify:

- `orion-ejb-jar.xml`
- `toplink-ejb-jar.xml`
- TopLink Workbench project file `TLCmpProject.mwp`

The `ejb-jar.xml` and `orion-ejb-jar.xml` files may be in an EAR, JAR, or just standalone XML files. If you migrate from standalone XML files (rather than an EAR or JAR file), ensure that the domain classes are accessible and included in your classpath.

The TopLink migration tool creates a new `orion-ejb-jar.xml` and `toplink-ejb-jar.xml` file to the target directory you specify in the same format as it reads the original files. For example, if you specify an EAR file as input, then the TopLink migration tool stages and creates a new EAR file that contains both the new `orion-ejb-jar.xml` and the new `toplink-ejb-jar.xml` file, but is otherwise identical to the original.

The TopLink Workbench project file is always created as a separate file.

Note: Oracle recommends that you make a backup copy of your `orion-ejb-jar.xml` file before using the TopLink migration tool.

Migration

As it operates, the TopLink migration tool logs all errors and diagnostic output to a log file named `oc4j_migration.log` in the output directory. If you use the TopLink migration tool from TopLink Workbench, see also the TopLink Workbench log file `oracle.toplink.workbench.log` located in your user home directory (for example, `C:\Documents and Settings\<user-name>`).

The TopLink migration tool processes descriptor, mapping, and query information from the input files as follows:

- It builds a TopLink descriptor object for each entity bean and migrates native persistence metadata like mapped tables, primary keys, and mappings for CMP and CMR fields.

- It builds a TopLink mapping object for every CMP and CMR field of an entity bean and migrates native persistence metadata like foreign key references.
- It builds a TopLink query object for each `finder` or `ejbSelect` of an entity bean and migrates persistence metadata like customized query statements.

Table 7-3 lists OC4J `<entity-deployment>` attributes and subelements from the `orion-ejb-jar.xml` file and for each, indicates whether or not the TopLink migration tool:

- Retains it in the new `orion-ejb-jar.xml` file
- Migrates it to the new `toplink-ejb-jar.xml` file

In Table 7-3, elements are identified with angle brackets. Note that in some cases an attribute is migrated when set to one value, but discarded if set to another value (for example, `exclusive-write-access`).

Table 7-3 OC4J `orion-ejb-jar.xml` Feature Migration

orion-ejb-jar.xml Feature	Retained in New orion-ejb-jar.xml	Migrated to New topLink-ejb-jar.xml
<code><entity-deployment></code>		
<code>clustering-schema</code>	✓	
<code>copy-by-value</code>	✓	
<code>data-source</code>	✓	
<code>location</code>	✓	
<code>max-instances</code>	✓	
<code>min-instances</code>	✓	
<code>max-tx-retries</code>	✓	
<code>disable-wrapper-cache</code>	✓	
<code>name</code>	✓	
<code>pool-cache-timeout</code>	✓	
<code>wrapper</code>	✓	
<code>local-wrapper</code>	✓	
<code>call-timeout</code>		✓
<code>exclusive-write-access</code>		
<code>true</code>		✓
<code>false</code>		
<code>do-select-before-insert</code>		
<code>true</code>		
<code>false</code>		
<code>isolation</code>		✓
<code>locking-mode</code>		
<code>pessimistic</code>		✓
<code>optimistic</code>		
<code>read-only</code>		✓
<code>old_pessimistic</code>		
<code>update-changed-fields-only</code>		
<code>true</code>		✓
<code>false</code>		

Table 7–3 (Cont.) OC4J orion-ejb-jar.xml Feature Migration

orion-ejb-jar.xml Feature	Retained in New orion-ejb-jar.xml	Migrated to New toplink-ejb-jar.xml
table		✓
force-update		
true		
false		✓
data-synchronization-option		
ejbCreate		
ejbPostCreate		
batch-size		
Any value greater than 1		
<ior-security-config>	✓	
<env-entry-mapping>	✓	
<resource-ref-mapping>	✓	
<resource-env-ref-mapping>	✓	
<primkey-mapping>		✓
<cmp-field-mapping>		✓
one-to-one-join		
inner		✓
outer ¹		
shared		✓
<finder-method>		✓
<persistence-type> ²		✓

¹ TopLink supports both outer and inner joins at run time. You can manually configure EJB descriptors with these options. For more information, see ["Join Reading and Object-Level Read Queries"](#) on page 96-12.

² The persistence-type attribute's table column size, if present, is discarded. For more information, see ["Recovering persistence-type Table Column Size"](#) on page 7-12.

[Table 7–4](#) lists OC4J features and their TopLink equivalents configured by the TopLink migration tool.

Table 7–4 OC4J and TopLink Feature Comparison

Feature	orion-ejb-jar.xml	toplink-ejb-jar.xml
CMP field mapping	Direct	Direct-to-field
	Serialized object	Serialized object
CMR field mapping	One-to-one	One-to-one
	One-to-many	One-to-many
	Many-to-many	Many-to-many
Partial query	Full SQL statement	SQL Call
Finder	Oracle-specific syntax	SQL Call or EJB-QL
Lazy loading (fetch group)	Lazy loading of primary key and CMP fields	Not supported Alternatively, you can manually configure the TopLink equivalent, if appropriate (see "Fetch Groups" on page 26-4).

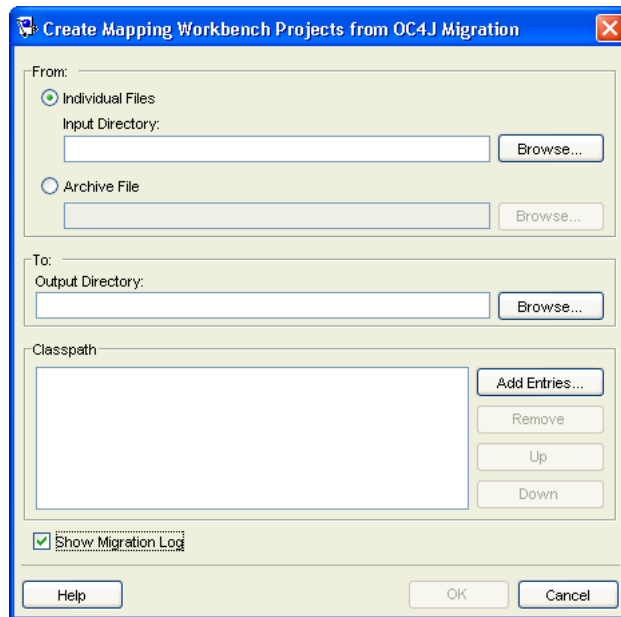
Table 7–4 (Cont.) OC4J and TopLink Feature Comparison

Feature	orion-ejb-jar.xml	toplink-ejb-jar.xml
SQL statement caching	Cache static SQL	Not supported at the bean level. TopLink supports parameterized SQL and statement caching at the session and query level (see "Understanding TopLink Queries" on page 96-1).
Locking	Optimistic: database-level Pessimistic: bean instance-level	Optimistic: object-level Pessimistic: query lock at database-level
Read-only	Attempt to change throws <code>Exception</code>	Attempt to change throws <code>Exception</code>
Validity timeout	Read-only bean validity timeout before reloaded.	Cache timeout
Isolation level	Committed Serializable	Committed Serializable Not Committed Not Repeatable
Delay update until commit	Supported	Supported (see "Configuring a Descriptor With EJB Information" on page 28-44).
Exclusive write access on bean	Default value is <code>false</code>	Assume <code>true</code>
Insert without existence check	Supported	Supported
Update changed fields only	Supported	Supported (see "Attribute Change Tracking Policy" on page 100-8).
Force update	Invoke bean life cycle <code>ejbStore</code> method even though persistent fields have not changed	Supported

Using the TopLink Migration Tool from TopLink Workbench

To use the TopLink migration tool and create a new, mapped TopLink Workbench project from an OC4J application, use this procedure:

1. From TopLink Workbench, select **File > Migrate > From OC4J 9.0.x**.

Figure 7-1 Create Project from OC4J Migration Dialog Box

2. Continue with ["Post-Migration Changes"](#) on page 7-12.

Use the following information to enter data in each field of the Create Project from OC4J dialog box:

Field	Description
From	Use these fields to specify the location of the existing OC4J files. These files may be included as part of a JAR, EAR, or individual files.
Individual Files	Select to convert from individual <code>ejb-jar.xml</code> and <code>orion-ejb-jar.xml</code> files in the Input Directory . Click Browse and select the directory location that contains the XML files to convert from.
Archive File	Select to use a specific archive file. Click Browse and select the archive file to convert from..
To	Use these fields to specify the location to which migrated files are written.
Output Directory	Click Browse and select a directory location in which to create the new XML files and TopLink Workbench project.
Classpath	If you are migrating from individual files, ensure that the domain classes are accessible and included in your classpath.
Show Migration Log	Select to have migration log output displayed in a separate window.

Using the TopLink Migration Tool From the Command Line

To use the TopLink migration tool from the command line, you must perform the following steps:

1. Ensure that the following is in your classpath:
 - `<TOPLINK_HOME>/jlib/antlr.jar`
 - `<TOPLINK_HOME>/jlib/ejb.jar`

- `<TOPLINK_HOME>/jlib/toplink.jar`
 - `<TOPLINK_HOME>/jlib/cmpmigrator.jar`
 - `<TOPLINK_HOME>/jlib/toplinkmw.jar`
 - `<TOPLINK_HOME>/jlib/tlmwcore.jar`
 - `<TOPLINK_HOME>/config`
 - `<ORACLE_HOME>/lib/xmlparserv2.jar`
2. If you intend to migrate from plain XML files (rather than an EAR or JAR file), ensure that the domain classes are accessible and included in your classpath.
 3. Make a backup copy of your original XML files.
 4. Execute the TopLink migration tool as [Example 7-2](#) illustrates using the appropriate arguments listed in [Table 7-5](#).

The usage information for the TopLink migration tool is:

```
java -Dtoplink.ejbjar.schemavalidation=<true|false>
-Dtoplink.migrationtool.generateWorkbenchProject=<true|false>
-Dhttp.proxyHost=<proxyHost> -Dhttp.proxyPort=<proxyPort>
oracle.toplink.tools.migration.TopLinkCMPMigrator -s<nativePM> -i<inputDir>
-a<ear>|<jar> -x -o<outputDir> -v
```

To identify the input files, you must specify one of `-a` or `-x`.

For troubleshooting information, see ["Troubleshooting Your Migration"](#) on page 13.

Example 7-2 Using the TopLink Migration Tool from the Command Line

```
java -Dhttp.proxyHost=www-proxy.us.oracle.com -Dhttp.proxyPort=80
oracle.toplink.tools.migration.TopLinkCMPMigrator -sOc4j-native -iC:/mywork/in
-aEmployee.ear -oC:/mywork/out -v
```

Table 7-5 TopLink Migration Tool Arguments

Argument	Description
<code>toplink.ejbjar.schemavalidation</code>	The system property used to turn on schema validation if <code>ejb-jar.xml</code> uses XML Schema (XSD) instead of DTD. The default value is <code>false</code> .
<code>toplink.migrationtool.generateWorkbenchProject</code>	The system property used enable generation of the TopLink Workbench project. The default value is <code>true</code> .
<code><proxyHost></code>	The address of your local HTTP proxy host
<code><proxyPort></code>	The port number on which your local HTTP proxy host receives HTTP requests.
<code>-s <source></code>	The name of the native persistence manager from which you are migrating. For OC4J, use the name <code>Oc4j-native</code> .
<code>-i <input-directory></code>	Fully qualified path to the input directory that contains both the <code>OC4J ejb-jar.xml</code> and <code>orion-ejb-jar.xml</code> files to migrate. Default: current working directory.
<code>-a <EAR-or-JAR></code>	Fully qualified path to the archive file (either an EAR or JAR) that contains both the <code>OC4J ejb-jar.xml</code> and <code>orion-ejb-jar.xml</code> files to migrate.

Table 7–5 (Cont.) TopLink Migration Tool Arguments

Argument	Description
-x	Tells the TopLink migration tool that the OC4J files in the input directory to migrate from are plain XML files (not in an archive file). If you use this option, ensure that the domain classes are accessible and included in your classpath.
-o <output-directory>	<targetDir> is the path to the directory into which the TopLink migration tool writes the new <code>orion-ejb-jar.xml</code> , <code>toplink-ejb-jar.xml</code> , and log files. The path may be absolute or relative to the current working directory. You must specify this argument value. Ensure that permissions are set on this directory to allow the TopLink migration tool to create files and subdirectories.
-v	Verbose mode. Tells the TopLink migration tool to log errors and diagnostic information to the console.

Post-Migration Changes

After you migrate the `orion-ejb-jar.xml` file persistence configuration to your `toplink-ejb-jar.xml` file, consider the following post-migration changes:

- [Recovering persistence-type Table Column Size](#)
- [Updating the Unknown Primary Key Class Mapping Sequence Table](#)
- [Project Customization](#)

Recovering persistence-type Table Column Size

In the `orion-ejb-jar.xml` file, you can specify this mapping, `cmp-field-mapping`, with a `persistence-type` attribute that provides both the type and column size as shown in [Example 7–3](#).

Example 7–3 A `cmp-field-mapping` with persistence-type Specifying a Column Size

```
<cmp-field-mapping ejb-reference-home="MyOtherEntity" name="myField"
persistence-name="myField" persistence-type="VARCHAR2(30)">
```

The TopLink migration tool migrates the persistence type but not the column size because a TopLink project does not normally contain this size information.

To recover the persistence-type column size, do the following:

1. Perform the migration as described in ["Using the TopLink Migration Tool From the Command Line"](#) on page 10.
2. Launch the generated TopLink Workbench project file `TLCmpProject.mwp`.
3. Log in to your database (see ["Logging In and Out of a Database"](#) on page 4-21).

TopLink Workbench retrieves all column sizes.

Updating the Unknown Primary Key Class Mapping Sequence Table

TopLink supports the use of an unknown primary key class (see ["Unknown Primary Key Class Support"](#) on page 7-25) and so the TopLink migration tool also supports this feature.

OC4J uses a native run-time key generator to generate unique keys for auto-id key fields. In contrast, TopLink uses a sequencing table.

If your OC4J persistence configuration includes the use of an unknown primary key class, then the TopLink migration tool will create a sequencing table for this purpose.

Before deploying your application after migration, you must do the following:

1. Determine the largest key value generated prior to migration.
2. Manually update the counter of the TopLink migration tool-generated sequence table to a number that must be one larger than the largest key value generated prior to migration.

Project Customization

You can customize the following components of your project:

- [EJB 3.0 Persistence Manager Customization](#)
- [EJB 2.1 Persistence Manager Customization](#)
- [Session Event Listener](#)

EJB 3.0 Persistence Manager Customization TopLink For an EJB 3.0 CMP application deployed to OC4J, you customize the TopLink persistence manager by creating a TopLink project XML file named `toplink-ejb-jar.xml` and a TopLink session XML file named `ejb3-toplink-sessions.xml` and packaging them in the META-INF directory of the EJB-JAR that contains your EJB 3.0 entities. For more information, see "Customizing the TopLink Persistence Manager in an EJB 3.0 Application" in the *Oracle Containers for J2EE Enterprise JavaBeans Developer's Guide*.

EJB 2.1 Persistence Manager Customization For an EJB 2.1 CMP application deployed to OC4J, you customize the TopLink persistence manager by configuring properties in the `orion-ejb-jar.xml` file. These properties are used to configure the TopLink session that the TopLink runtime uses internally for CMP projects. For more information, see "[Configuring persistence-manager Entries](#)" on page 8-9.

Session Event Listener After you applied the default settings to your project at deployment time, you may wish to customize the TopLink session by configuring the session event listener. The pre-login event that the session raises is particularly useful. It lets you define the custom (nondefault) specifics for the session just before the session initializes and acquires connections.

For more information, see the following:

- "[Session Customization](#)" on page 75-4
- "[Managing Session Events With the Session Event Manager](#)" on page 75-5
- "[Configuring Session Event Listeners](#)" on page 77-17

Troubleshooting Your Migration

This section describes solutions for problems you may encounter during migration, including the following:

- [Log Messages](#)
- [Unexpected Relational Multiplicity](#)

Log Messages

As it operates, the TopLink migration tool logs all errors and diagnostic output to a log file named `oc4j_migration.log` in the output directory. If you use the TopLink migration tool from TopLink Workbench, see also the TopLink Workbench log file

`oracle.toplink.workbench.log` located in your user home directory (for example, `C:\Documents and Settings\<user-name>`)

In addition to these warnings, the TopLink migration tool logs an error if it encounters a problem that prevents it from completing the migration. [Table 7–6](#) lists these problems and suggests possible solutions.

Table 7–6 TopLink Migration Tool Error Messages

Error Message	Description
There is no <code>ejb-jar.xml</code> in the input file. You must provide the <code>ejb-jar.xml</code> in order for the migration process to work.	The <code>ejb-jar.xml</code> file is missing. The TopLink migration tool stops and copies the original input files into the target directory. Verify that the <code>ejb-jar.xml</code> file is present in the specified EAR, JAR, or as a plain XML file. Empty the target directory and execute the TopLink migration tool again.
There is not an <code>orion-ejb-jar.xml</code> with native persistent metadata defined, no migration needed.	The <code>orion-ejb-jar.xml</code> file is missing. The TopLink migration tool stops and copies the original input files into the target directory. Verify that the <code>orion-ejb-jar.xml</code> file is present in the specified EAR, JAR, or as a plain XML file. Empty the target directory and execute the TopLink migration tool again.
<code>toplink-ejb-jar.xml</code> is already defined in the archive, no migration needed.	A <code>toplink-ejb-jar.xml</code> file is already present in the target directory. The TopLink migration tool stops and copies the original input files into the target directory. Remove the <code>toplink-ejb-jar.xml</code> file from the target directory. Empty the target directory and execute the TopLink migration tool again.

Unexpected Relational Multiplicity

The TopLink migration tool retrieves relationship multiplicity from the `orion-ejb-jar.xml` file and not from the `OC4J ejb-jar.xml` file.

Thus, even though the `OC4J ejb-jar.xml` file defines a relationship to be one-to-many, if the `orion-ejb-jar.xml` file defines the same relationship as many-to-many, then the TopLink migration tool will migrate the relationship as many-to-many.

JTA Integration

For applications that require JTA integration, specify the external transaction controller when you configure the server platform in your session (see ["Configuring the Server Platform"](#) on page 77-14).

For more information, see ["Integrating the Unit of Work With an External Transaction Service"](#) on page 102-20.

BEA WebLogic Server

To integrate a TopLink application with BEA WebLogic Server, you must consider the following:

- [Classpath](#)
- [CMP Integration](#)
- [Migrating BEA WebLogic Persistence to OC4J TopLink Persistence](#)
- [JTA Integration](#)

- [Security Manager](#)

In addition to configuring these BEA WebLogic Server-specific options, you must also consider the general application server integration issues in ["Application Server Integration Concepts"](#) on page 7-2

Classpath

To configure TopLink support for BEA WebLogic Server:

1. Add the following JAR files to the application server classpath:
`<ORACLE_HOME>\toplink\jlib\toplink.jar`
2. Ensure that your TopLink application defines an XML parser platform (see ["XML Parser Platform Configuration"](#) on page 7-2).

CMP Integration

To enable TopLink CMP integration in BEA WebLogic Server, use the following procedure. This procedure assumes you have already installed TopLink.

1. Locate the persistence directory, located above the installation drive and root directory of your BEA WebLogic Server executable, as follows:

Version	Persistence Directory (above the <code><WebLogic_INSTALL_DIR></code>)
6.1 (Service Pack 4)	\wlserver6.1\lib\persistence
7.0 (Service Pack 2)	\weblogic700\server\lib\persistence
8.1	\weblogic81\server\lib\persistence

Do one of the following:

- Use a text editor to open the `persistence.install` file in the BEA WebLogic Server persistence directory, and add a new line that references the `TopLink_CMP_Descriptor.xml` file.
 - Replace the WebLogic `persistence.install` file with the TopLink `persistence.install` file found in the `<ORACLE_HOME>\toplink\config` directory.
2. Add the following JAR files to the application server classpath:
`<ORACLE_HOME>\toplink\jlib\toplink.jar`
`<ORACLE_HOME>\lib\xmlparserv2.jar`
 3. If necessary, migrate your CMP application using the TopLink migration tool (["Migrating BEA WebLogic Persistence to OC4J TopLink Persistence"](#) on page 7-16).
 4. Ensure that all necessary deployment descriptor files are in place (see ["Creating TopLink Files for Deployment"](#) on page 8-1 and ["Packaging a TopLink Application"](#) on page 9-1).
 5. Optionally, consider the EJB customization options that TopLink provides (["Configuring Miscellaneous EJB Options"](#) on page 7-25).
 6. Start the container, and then start the TopLink application. Where supported, use a startup script to start the server. If you write your own startup script, ensure that the classpath includes the files listed in Step 2.

Migrating BEA WebLogic Persistence to OC4J TopLink Persistence

You can migrate a BEA WebLogic application that uses the default BEA WebLogic persistence manager to an Oracle Containers for J2EE (OC4J) application that uses TopLink as the persistence manager. In this release, Oracle provides a TopLink migration tool that you can use to automate this migration.

This section includes the following:

- [Overview](#)
- [Using the TopLink Migration Tool From TopLink Workbench](#)
- [Using the TopLink Migration Tool From the Command Line](#)

If you encounter problems during migration, see ["Troubleshooting Your Migration"](#) on page 7-13.

After using the TopLink migration tool, you may need to make some additional changes as described in ["Post-Migration Changes"](#) on page 12.

Overview

Before using the TopLink migration tool, review this section to understand how the TopLink migration tool works and to determine what BEA WebLogic persistence manager metadata is, and is not, migrated.

Input and Output

The TopLink migration tool takes the following files as input:

- `weblogic-ejb-jar.xml`
- `weblogic-cmp-rdbms-jar.xml`

It migrates as much BEA WebLogic-specific persistence configuration as possible to a new `toplink-ejb-jar.xml` file and creates the following new files in a target directory you specify:

- `orion-ejb-jar.xml`
- `toplink-ejb-jar.xml`
- TopLink Workbench project file `TLCmpProject.mwp`

The input BEA WebLogic files may be in an EAR, JAR, or just standalone XML files. If you migrate from standalone XML files (rather than an EAR or JAR file), ensure that the domain classes are accessible and included in your classpath.

The TopLink migration tool creates a new `orion-ejb-jar.xml` and `toplink-ejb-jar.xml` file to the target directory you specify in the same format as it reads the original files. For example, if you specify an EAR file as input, then the TopLink migration tool stages and creates a new EAR file that contains both the new `orion-ejb-jar.xml` and the new `toplink-ejb-jar.xml` file, but is otherwise identical to the original.

The TopLink Workbench project file is always created as a separate file.

Note: Oracle recommends that you make a backup copy of your `weblogic-ejb-jar.xml`, `weblogic-cmp-rdbms-jar.xml`, and `toplink-ejb-jar.xml` files before using the TopLink migration tool.

For information on configuring the `weblogic-ejb-jar.xml` file, see ["Configuring the weblogic-ejb-jar.xml File for BEA WebLogic Server"](#) on page 8-13.

Migration

As it operates, the TopLink migration tool logs all errors and diagnostic output to a log file named `wls_migration.log` in the output directory. If you use the TopLink migration tool from TopLink Workbench, see also TopLink Workbench log file `oracle.toplink.workbench.log` located in your user home directory (for example, `C:\Documents and Settings\<user-name>`).

The TopLink migration tool processes descriptor, mapping, and query information from the input files. It performs the following:

- It builds a TopLink descriptor object for each entity bean and migrates native persistence metadata like mapped tables, primary keys, and mappings for CMP and CMR fields.
- It builds a TopLink mapping object for every CMP and CMR field of an entity bean and migrates native persistence metadata like foreign key references.
- It builds a TopLink query object for each `finder` or `ejbSelect` of an entity bean and migrates persistence metadata like customized query statements.

[Table 7-7](#) lists BEA WebLogic features and their TopLink equivalents configured by the TopLink migration tool.

Table 7-7 BEA WebLogic and TopLink Feature Comparison

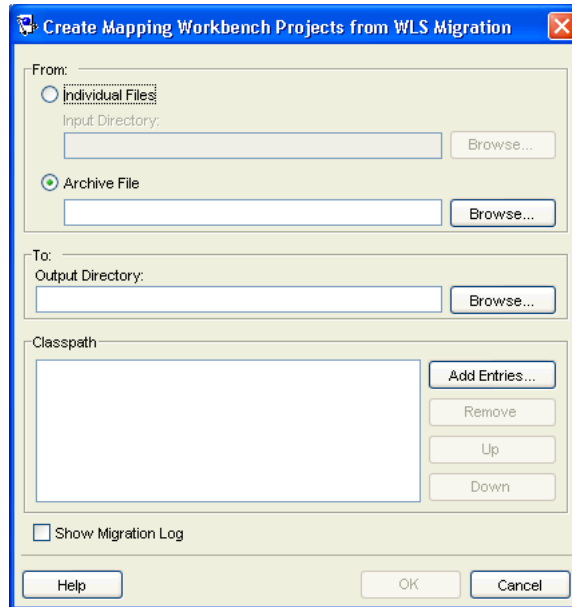
Feature	<code>weblogic-cmp-rdbms-jar.xml</code>	<code>toplink-ejb-jar.xml</code>
Direct mapping	Field mapping Table mapping	Direct-to-field
Relational mapping	WebLogic-RDBMS relation	One-to-one One-to-many Many-to-many
Multiple tables	Table mapping	Supported
Read-only	Read-only: concurrency strategy	Supported
Pessimistic locking	Enforce pessimistic concurrency on a per-bean basis	Supported
Large Objects (LOB) support	Map the current field to one of the following: 1. <code>CLOB</code> or <code>BLOB</code> in an Oracle database 2. <code>LongString</code> or <code>SybaseBinary</code> in a Sybase database	Supported for Oracle BLOB only
Optimistic locking	Optimistic: concurrency strategy Verify columns: optimistic concurrency strategy	Version Timestamp
Cascade delete	Database cascade delete	Privately owned
Sorting database dependency	Order database operations	unit of work
Lazy loading (fetch group)	Field group	Not supported. Alternatively, you can manually configure the TopLink equivalent, if appropriate (see "Fetch Groups" on page 26-4).

Using the TopLink Migration Tool From TopLink Workbench

To use the TopLink migration tool and create a new, mapped TopLink Workbench project from a BEA WebLogic application, use this procedure:

1. From TopLink Workbench, select **File > Migrate > From Weblogic Server**.

Figure 7-2 Create Projects from WebLogic Server Migration Dialog Box



2. Continue with ["Post-Migration Changes"](#) on page 7-12.

Use the following information to enter data in each field on the Create Project from WebLogic dialog box:

Field	Description
From	Use these fields to specify the location of the existing BEA WebLogic files. These files may be included as part of a JAR, EAR, or individual files.
Individual Files	Select to convert from individual <code>weblogic-ejb-jar.xml</code> and <code>weblogic-cmp-rdbms-jar.xml</code> files in the Input Directory . Click Browse and select the directory location that contains the XML files to convert from.
Archive File	Select to use a specific archive file. Click Browse and select the archive file to convert from..
To	Use this field to specify the location to which migrated files are written.
Output Directory	Click Browse and select a directory location in which to create the new XML files and TopLink Workbench project.
Classpath	If you are migrating from individual files, ensure that the domain classes are accessible and included in your classpath.
Show Migration Log	Select to have migration log output displayed in a separate window.

Using the TopLink Migration Tool From the Command Line

To use the TopLink migration tool from the command line, you must do the following:

1. Ensure that the following is in your classpath:
 - `<TOPLINK_HOME>/jlib/antlr.jar`
 - `<TOPLINK_HOME>/jlib/ejb.jar`
 - `<TOPLINK_HOME>/jlib/toplink.jar`
 - `<TOPLINK_HOME>/jlib/cmpmigrator.jar`
 - `<TOPLINK_HOME>/jlib/toplinkmw.jar`
 - `<TOPLINK_HOME>/jlib/tlmwcore.jar`
 - `<TOPLINK_HOME>/config`
 - `<ORACLE_HOME>/lib/xmlparserv2.jar`
2. If you intend to migrate from plain XML files (rather than an EAR or JAR file), ensure that the domain classes are accessible and included in your classpath.
3. Make a backup copy of your original XML files.
4. Execute the TopLink migration tool as [Example 7-4](#) illustrates using the appropriate arguments listed in [Table 7-8](#).

The usage information for the TopLink migration tool is:

```
java -Dtoplink.ejbjar.schemavalidation=<true|false>
-Dtoplink.migrationtool.generateWorkbenchProject=<true|false>
-Dhttp.proxyHost=<proxyHost> -Dhttp.proxyPort=<proxyPort>
oracle.toplink.tools.migration.TopLinkCMPMigrator -s<nativePM> -i<inputDir>
-a<ear>|<jar> -x -o<outputDir> -v
```

To identify the input files, you must specify one of `-a` or `-x`.

For troubleshooting information, see ["Troubleshooting Your Migration"](#) on page 13.

Example 7-4 Using the TopLink Migration Tool from the Command Line

```
java -Dhttp.proxyHost=www-proxy.us.oracle.com -Dhttp.proxyPort=80
oracle.toplink.tools.migration.TopLinkCMPMigrator -swebLogic -iC:/mywork/in
-aEmployee.ear -oC:/mywork/out -v
```

Table 7-8 TopLink Migration Tool Arguments

Argument	Description
<code>toplink.ejbjar.schemavalidation</code>	The system property used to turn on schema validation if <code>ejb-jar.xml</code> uses XML Schema (XSD) instead of DTD. The default value is <code>false</code> .
<code>toplink.migrationtool.generateWorkbenchProject</code>	The system property used enable generation of the TopLink Workbench project. The default value is <code>true</code> .
<code><proxyHost></code>	The address of your local HTTP proxy host
<code><proxyPort></code>	The port number on which your local HTTP proxy host receives HTTP requests.
<code>-s <source></code>	The name of the native persistence manager from which you are migrating. For BEA WebLogic, use the name <code>WebLogic</code> .
<code>-i <input-directory></code>	Fully qualified path to the input directory that contains the BEA WebLogic <code>weblogic-ejb-jar.xml</code> and <code>weblogic-cmp-rdbms-jar.xml</code> files to migrate. Default: current working directory.

Table 7–8 (Cont.) TopLink Migration Tool Arguments

Argument	Description
-a <EAR-or-JAR>	Fully qualified path to the archive file (either an EAR or JAR) that contains both the BEA WebLogic <code>weblogic-ejb-jar.xml</code> and <code>weblogic-cmp-rdbms-jar.xml</code> files to migrate.
-x	Tells the TopLink migration tool that the BEA WebLogic files in the input directory to migrate from are plain XML files (not in an archive file). If you use this option, ensure that the domain classes are accessible and included in your classpath.
-o <output-directory>	The path to the directory into which the TopLink migration tool writes the new <code>orion-ejb-jar.xml</code> , <code>toplink-ejb-jar.xml</code> , and log files. The path may be absolute or relative to the current working directory. You must specify this argument value. Ensure that permissions are set on this directory to allow the TopLink migration tool to create files and subdirectories.
-v	Verbose mode. Tells the TopLink migration tool to log errors and diagnostic information to the console.

JTA Integration

For applications that require JTA integration, specify the external transaction controller when you configure the server platform in your session (see ["Configuring the Server Platform"](#) on page 77-14).

For more information, see ["Integrating the Unit of Work With an External Transaction Service"](#) on page 102-20.

Security Manager

If you use a security manager, specify a security policy file in the `weblogic.policy` file (normally located in the BEA WebLogic install directory), as follows:

```
-Djava.security.manager
-Djava.security.policy==c:\weblogic\weblogic.policy
```

The BEA WebLogic installation procedure includes a sample security policy file. You need to edit the `weblogic.policy` file to grant permission for TopLink to use reflection.

The following example illustrates only the permissions that TopLink requires, but most `weblogic.policy` files contain more permissions than are shown in this example.

Example 7–5 A Subset of a "Grant" Section from a BEA WebLogic.policy File

```
grant {
// "enableSubstitution" required to run the WebLogic console
permission java.io.SerializablePermission "enableSubstitution";
// "modifyThreadGroup" required to run the WebLogic Server
permission java.lang.RuntimePermission "modifyThreadGroup";
//grant permission for TopLink to use reflection
    permission java.lang.reflect.ReflectPermission "suppressAccessChecks";
};
```

IBM WebSphere Application Server

To integrate a TopLink application with IBM WebSphere Application Server, you must consider the following:

- [Classpath](#)
- [CMP Integration](#)
- [JTA Integration](#)

In addition to configuring these IBM WebSphere application server-specific options, you must also consider the general application server integration issues in "[Application Server Integration Concepts](#)" on page 7-2.

Classpath

You configure the IBM WebSphere application server classpath differently depending on what version of this server you are using:

- [Configuring Classpath for IBM WebSphere Application Server 4.0](#)
- [Configuring Classpath for IBM WebSphere Application Server 5.0 and Later](#)

Configuring Classpath for IBM WebSphere Application Server 4.0

TopLink provides CMP support for IBM WebSphere application server 4.0. To configure the classpath for this version, do the following:

1. Add the following JAR files to the application server classpath directory (see [Table 7-9](#)):

```
<ORACLE_HOME>\toplink\jlib\toplink.jar
```

2. Ensure that your TopLink application defines an XML parser platform (see "[XML Parser Platform Configuration](#)" on page 7-2).

[Table 7-9](#) lists the default application classpath directories for IBM container components in IBM WebSphere application server 4.0.

Table 7-9 Classpath Directories for IBM WebSphere 4.0 Container Components

Container	Default Application Classpath
WebSphere Application Server 4.0 (for Windows)	\WebSphere\AppServer\lib\app
WebSphere Studio Application Developer 4.0 (for Windows)	\Program Files\ibm\Application Developer\plugins\com.ibm.etools.websphere.runtime\lib\app

Configuring Classpath for IBM WebSphere Application Server 5.0 and Later

TopLink provides JTA and general integration support for IBM WebSphere application server 5.0 and later. To configure the classpath for this version, do the following:

1. Create a shared library that contains the following Toplink JAR files and associate the shared library with the application:

```
<ORACLE_HOME>\toplink\jlib\toplink.jar
```

2. Ensure that your TopLink application defines an XML parser platform (see "[XML Parser Platform Configuration](#)" on page 7-2).

CMP Integration

To enable TopLink CMP integration in IBM WebSphere application server, use the following procedure. This procedure assumes you have already installed TopLink.

1. Ensure that all necessary deployment descriptor files are in place (see ["Creating TopLink Files for Deployment"](#) on page 8-1 and ["Packaging a TopLink Application"](#) on page 9-1).
2. Optionally, consider the EJB customization options that TopLink provides (["Configuring Miscellaneous EJB Options"](#) on page 7-25).

JTA Integration

For applications that require JTA integration, specify the external transaction controller when you configure the server platform in your session (see ["Configuring the Server Platform"](#) on page 77-14).

For more information, see ["Integrating the Unit of Work With an External Transaction Service"](#) on page 102-20.

Clustering on IBM WebSphere Application Server

For more information on integrating a TopLink application with an application server cluster, see ["Clustering"](#) on page 7-4.

Understanding Security Permissions

By default, when you run a TopLink-enabled application in a JVM configured with a nondefault `java.lang.SecurityManager`, the TopLink run time executes certain internal functions by executing a `PrivilegedAction` with `java.security.AccessController` method `doPrivileged`. This ensures that you do not need to grant many permissions to TopLink for it to perform its most common operations. You need only grant certain permissions depending on the types of optional TopLink features you use (see ["Permissions Required by TopLink Features"](#) on page 7-23).

While using `doPrivileged` method provides enhanced security, it will severely impact overall performance. Alternatively, you can configure TopLink to disable the use of `doPrivileged` method even when a nondefault `SecurityManager` is present (see ["Disabling doPrivileged Operation"](#) on page 7-25). In this case, you must grant TopLink all required permissions (see ["Permissions Required by TopLink Features"](#) on page 7-23 and ["Permissions Required When doPrivileged is Disabled"](#) on page 7-24).

Note: While enabling the use of `doPrivileged` method enhances TopLink application security, it does not guarantee that secure code cannot be called by application code in ways that the system did not intend. You must consider the use of `doPrivileged` method within the context of your overall application security strategy. For more information, see <http://java.sun.com/security/index.jsp>.

If you run a TopLink-enabled application in a JVM without a nondefault `SecurityManager`, you do not need to grant any permissions.

Permissions Required by TopLink Features

When you run a TopLink-enabled application in a JVM configured with a nondefault `java.lang.SecurityManager` and `doPrivileged` operation is enabled, you may need to grant additional permissions if your application requires any of the following:

- [System Properties](#)
- [Loading project.xml or sessions.xml Files](#)
- [Cache Coordination](#)
- [Accessing a Data Source by Port](#)
- [Logging With java.util.logging](#)
- [J2EE Application Deployment](#)

System Properties

By default, a TopLink-enabled application requires access to the system properties granted in the default `<JAVA_HOME>/lib/security/java.policy` file. If your application requires access to other platform-specific, environment, or custom properties, then grant further `PropertyPermission` permissions as [Example 7-6](#) shows.

Example 7-6 Permissions for System Properties

```
permission java.util.PropertyPermission "my.property", "read";
```

Loading project.xml or sessions.xml Files

Most TopLink-enabled applications read in `project.xml` and `sessions.xml` files directly. Grant permissions to the specific files or file locations as [Example 7-7](#) shows. This example assumes that both `project.xml` and `sessions.xml` files are located in the same directory (given by application-specific system property `deployment.xml.home`). Alternatively, you can specify a separate `FilePermission` for each file.

Example 7-7 Permissions for Loading Deployment XML Files

```
permission java.io.FilePermission "${deployment.xml.home}/*.xml", "read";
```

For information on `FilePermission` settings for J2EE applications, see ["J2EE Application Deployment"](#) on page 7-24.

Cache Coordination

If your application uses cache coordination (see ["Understanding Cache Coordination"](#) on page 90-10), then grant `accept`, `connect`, `listen`, and `resolve` permissions to the specific sockets used by your coordinated cache as [Example 7-8](#) shows. This example assumes that the coordinated cache multicast port (see ["Configuring a Multicast Port"](#) on page 91-5) is 1024.

Example 7-8 Permissions for Cache Coordination

```
permission java.net.SocketPermission "localhost:1024-", "accept, connect, listen, resolve";
```

Accessing a Data Source by Port

If your TopLink-enabled application accesses a data source using a socket, then grant `connect` and `resolve` permissions for that socket as [Example 7-9](#) shows. This example assumes that the host name (or IP address) of the remote host that provides the data source (such as a relational database server host) is given by application-specific system property `remote.data.source.host` and that this host accepts data source connections on port 1025.

Example 7-9 Permissions for non-J2EE Data Source Connections

```
permission java.net.SocketPermission "${remote.data.source.host}:1025-", "connect, resolve";
```

For J2EE applications, data source socket permissions are usually handled by the application server.

Logging With `java.util.logging`

If you configure your TopLink-enabled application to use `java.util.logging` package (see ["Configuring Logging"](#) on page 77-4), then grant your application `control` permissions as [Example 7-10](#) shows.

Example 7-10 Permissions for `java.util.logging`

```
permission java.util.logging.LoggingPermission "control"
```

J2EE Application Deployment

If you are deploying a TopLink-enabled J2EE application, you must grant permissions for:

- The `toplink.jar` file. For example:

```
grant codeBase "file:<TOPLINK_HOME>/jlib/toplink.jar" {  
    permission java.security.AllPermission;  
};
```

If you are using an XML platform, you must also grant the following permissions:

- The `toplink.xml.platform` system property. For Example:

```
permission java.util.PropertyPermission "toplink.xml.platform", "read"
```

Permissions Required When `doPrivileged` is Disabled

If you disable `doPrivileged` operation when you run a TopLink-enabled application in a JVM configured with a nondefault `java.lang.SecurityManager`, you must grant the following permissions:

- `java.lang.reflect.ReflectPermission "suppressAccessChecks"`
- `java.lang.RuntimePermission "accessDeclaredMembers"`
- `java.lang.RuntimePermission "getClassLoader"`

You may also have to grant additional permissions depending on the TopLink features your application uses. For more information, see ["Permissions Required by TopLink Features"](#) on page 7-23.

Disabling doPrivileged Operation

To disable `doPrivileged` operation when you run a TopLink-enabled application in a JVM configured with a nondefault `java.lang.SecurityManager`, set system property `oracle.j2ee.toplink.security.usedoprivileged` to `false`. If you are using OC4J, set system property `oracle.j2ee.security.usedoprivileged` to `false`.

To enable `doPrivileged` operation, set these system properties to `true`.

Configuring Miscellaneous EJB Options

TopLink provides system properties that you can use to customize the following EJB options:

- [Setter Parameter Type Checking](#)
- [Unknown Primary Key Class Support](#)
- [Single-Object Finder Return Type Checking](#)

Setter Parameter Type Checking

To make TopLink verify that the parameters to one-to-one and one-to-many relationship setters are of the same type as the corresponding CMR field, set system property `toplink.cts.collection.checkParameters` to a value of `true` (not case sensitive). If the setters are not the same type, then TopLink throws a `java.lang.IllegalArgumentException`.

Note: Setting this property to `true` will affect performance. Use this setting only if necessary.

If you set the property to `false` (the default value), TopLink does not make this verification. In this case, it is up to your application to make sure the parameters are of the correct type.

For more information, see the EJB 2.1 specification, section 10.3.6.

Unknown Primary Key Class Support

In special situations, you may choose not to specify the primary key class or the primary key fields for an entity bean with container-managed persistence. For example, if the entity bean does not have a natural primary key or you want the deployer to select the primary key fields at deployment time, you may choose to defer primary key type specification.

If this is the case, you must declare the type of the argument of the `findByPrimaryKey` method as `java.lang.Object` and you must also specify the primary key class (`prim-key-class`) in the deployment descriptor (`ejb-jar.xml`) as `java.lang.Object`.

TopLink now provides run-time support for such deferred primary key type specification.

For more information, see the EJB 2.1 specification, section 10.8.3.

Single-Object Finder Return Type Checking

By setting system property `toplink.cts.checkMultipleRows` to `true`, you can configure TopLink to throw a `javax.ejb.FinderException` if multiple beans are returned from a single-object finder method.

For more information, see the EJB 2.1 specification, section 10.5.6.1.

Creating TopLink Files for Deployment

This chapter includes TopLink information you need when creating deployment files for the following types of applications:

- [Java Applications](#)
- [JavaServer Pages and Servlet Applications](#)
- [Session Bean Applications](#)
- [CMP Applications](#)
- [BMP Applications](#)

For more information on packaging and deployment, see the following:

- ["Understanding TopLink Deployment File Creation" on page 8-1](#)
- ["Integrating TopLink With an Application Server" on page 7-1](#)
- ["Packaging a TopLink Application" on page 9-1](#)
- ["Deploying a TopLink Application" on page 10-1](#)

Understanding TopLink Deployment File Creation

Depending on the type of application you are deploying, you may need to create any of the following deployment files:

- [project.xml File](#)
- [sessions.xml File](#)
- [ejb-jar.xml File](#)
- [<J2EE-Container>-ejb-jar.xml File](#)
- [toplink-ejb-jar.xml File](#)

TopLink Workbench provides the ability to create deployment files from a TopLink Workbench project (see ["Exporting Project Information" on page 21-13](#)). After you build a project, you have two options to create the deployment files:

- Create XML deployment files that require no compiling.
- Create Java source files, which you compile and deploy outside of TopLink Workbench.

Oracle recommends XML deployment because XML files are easier to deploy and troubleshoot than compiled Java files. This approach gives you a very flexible configuration that enables you to make changes safely and easily. XML deployment files do not require third-party applications or compilers to deploy successfully.

If you are using EJB 3.0, you can use annotations to specify most of what you formerly specified in deployment descriptors. Use deployment descriptors to override annotations or specify options not supported by annotations. For more information on what annotations are currently supported, see Oracle Containers for J2EE Enterprise JavaBeans Developer's Guide.

project.xml File

The `project.xml` file is the core of your application. It contains the descriptors and mappings you define and also includes any named queries or finders associated with your project.

This section describes:

- [XSD File Format](#)
- [Non-CMP Applications and Project Metadata](#)
- [CMP Applications and Project Metadata](#)
- [Creating project.xml With TopLink Workbench](#)
- [Creating project.xml Programatically](#)
- [EJB 3.0 and the project.xml File](#)

XSD File Format

Starting with 10g Release 3 (10.1.3), the `project.xml` file uses an XML schema file format (XSD file) instead of the old document type definition. This defines not only the elements and attributes, but also the rules that govern how the elements and attributes are used in a valid XML file. The XSD file is formatted as standard XML and fully compliant with Oracle namespaces. Although TopLink can read both the current XSD and older DTD formats, only the current XSD format is written out.

Previously formats were defined only by the DTDs. You can now generate deployment XML files based on the following XML schemas:

- `object-persistence_1_0.xsd`: This schema defines general persistence and mapping concepts.
- `toplink-object-persistence_10_1_3.xsd`: This schema extends the general concepts to include additional TopLink specific data.

For more information, refer to the appropriate XSD in the `<TOPLINK_HOME>\config\xsds` directory. The XSD files are also available on OTN at:

- http://www.oracle.com/technology/oracleas/schema/object-persistence_1_0.xsd
- http://www.oracle.com/technology/oracleas/schema/toplink-object-persistence_10_1_3.xsd

Non-CMP Applications and Project Metadata

For a non-CMP application, you define your project metadata in a `project.xml` file.

The `project.xml` file provides a simple and flexible way to configure, modify, and troubleshoot the project metadata. Because of these attributes, the `project.xml` file is the preferred way to configure a TopLink project.

TopLink Workbench provides a graphical tool to build and edit the `project.xml` file. For information on creating projects with TopLink Workbench, see "[Creating project.xml With TopLink Workbench](#)" on page 8-3.

CMP Applications and Project Metadata

For a CMP application, how you specify project metadata is dependent upon the J2EE application server you are deploying your application (see ["toplink-ejb-jar.xml File"](#) on page 8-6).

Creating project.xml With TopLink Workbench

Because you must synchronize the `project.xml` file with the classes and data source associated with your application, Oracle recommends that you not modify this file manually. TopLink Workbench ensures proper synchronization, and is the best way to make changes to the project. Simply modify the project in TopLink Workbench and redeploy the `project.xml` file. Using this option reduces development time by eliminating the need to regenerate and recompile Java code each time the project changes.

See ["Exporting Project Information"](#) on page 21-13 for detailed information on exporting the deployment XML information.

Note: You can name this file with a name other than `project.xml`; however, for clarity, this discussion assumes that the file has not been renamed.

Creating project.xml Programmatically

Optionally, you can use the `DeploymentXMLGenerator` API to programmatically generate the `project.xml` file in either of the following ways:

- From an application, instantiate the `DeploymentXMLGenerator` and your java source. Call the following method:

```
generate (<MW_Project.mwp>, <output file.xml>)
```
- From the command line, use:

```
java -classpath
toplink.jar;toplinkmw.jar;xmlparserv2.jar;ejb.jar;
oracle.toplink.workbench.external.api.DeploymentXMLGenerator
<MW_Project.mwp> <output file.xml>
```

Before you use either method, ensure your the classpath includes the `<ORACLE_HOME>\toplink\config` directory.

EJB 3.0 and the project.xml File

If you are using EJB 3.0, you can use annotations to specify most of what you formerly specified in the `project.xml` file. To override annotations or specify options not supported by annotations, you can still provide a `project.xml` file in your EJB 3.0 application. For more information on what annotations are currently supported, see Oracle Containers for J2EE Enterprise JavaBeans Developer's Guide.

sessions.xml File

Each TopLink project belongs to a TopLink *session*. A session is the facade through which an application accesses TopLink functionality (for more information on sessions, see [Part XVI, "TopLink Sessions"](#)). Where you define a session differs depending on whether or not your application uses CMP.

This section describes:

- [XSD File Format](#)
- [Non-CMP Applications and Session Metadata](#)
- [CMP Applications and Session Metadata](#)
- [EJB 3.0 and the sessions.xml File](#)

XSD File Format

Starting with 10g Release 3 (10.1.3), the `sessions.xml` file uses an XML schema file format (XSD file) instead of the old document type definition. In addition to gaining all the benefits of using an XSD, this change ensures that the TopLink run-time environment provides better diagnostics during `sessions.xml` file loading and validation.

The XSD files are also available on OTN at:

http://www.oracle.com/technology/oracleas/schema/sessions_10_1_3.xsd

When you use the XSD formatted `sessions.xml` file, the TopLink run time separates `sessions.xml` file validation from session instantiation. Separating XML file formatting problems from Session Manager session instantiation problems simplifies troubleshooting. Exceptions thrown during validation clearly indicate that the failure is due to an invalid `sessions.xml` file as [Example 8-1](#) illustrates.

Example 8-1 Enhanced Validation Exceptions

```
Exception [TOPLINK-9010] (Oracle TopLink - 10g (10.0.3)(Build 040127Dev)):  
oracle.toplink.exceptions.SessionLoaderException  
Exception Description: A End tag does not match start tag 'session'. was thrown while parsing  
the XML file against the XML schema.  
Internal Exception: oracle.xml.parser.v2.XMLParseException: End tag does not match start tag  
'session'.
```

Non-CMP Applications and Session Metadata

For a non-CMP application, you define your sessions in a `sessions.xml` file.

The `sessions.xml` file provides a simple and flexible way to configure, modify, and troubleshoot the application sessions. Because of these attributes, the `sessions.xml` file is the preferred way to configure a TopLink session.

TopLink Workbench provides a graphical tool to build and edit the `sessions.xml` file. For information on creating sessions with TopLink Workbench, see "[Creating Sessions](#)" on page 76-1.

CMP Applications and Session Metadata

For a CMP project, how you specify session metadata is dependent upon the J2EE application server you are deploying your application:

For OC4J, the session configuration is done in the `orion-ejb-jar.xml` file. You can specify the `data-source`, some common session options, and a session customizer class (see "[OC4J and the orion-ejb-jar.xml File](#)" on page 8-5). In this case, you name the TopLink project XML file as `toplink-ejb-jar.xml` (see "[project.xml File](#)" on page 8-2)

For BEA WebLogic Server, the session configuration is done in the `toplink-ejb-jar.xml` file. You can specify the `data-source`, some common session options, and a session customizer class (see "[toplink-ejb-jar.xml File](#)" on page 8-6).

For IBM WebSphere application server, the session configuration is done in a `sessions.xml` file which must be named `toplink-ejb-jar.xml` (see ["Creating Sessions"](#) on page 76-1).

EJB 3.0 and the sessions.xml File

If you are using EJB 3.0, you cannot use annotations to specify session configuration. You must provide a `sessions.xml` file if one is applicable to your application type, even if you are using EJB 3.0.

ejb-jar.xml File

Each EJB module contains one `ejb-jar.xml` file that describes all the EJB in the module.

Most IDEs provide facilities to create the `ejb-jar.xml` file. For more information about generating this file from your IDE, see your IDE documentation.

If you build an EJB 2.0 application, Oracle recommends that you use TopLink Workbench to build the `ejb-jar.xml` file. Because TopLink Workbench can both read and write the `ejb-jar.xml` file, you can use TopLink Workbench to maintain your `ejb-jar.xml` file in the following ways:

- When you change the file manually outside of TopLink Workbench, reimport the `ejb-jar.xml` file into TopLink Workbench project to refresh the project.
- When you change the TopLink Workbench project, TopLink Workbench updates the `ejb-jar.xml` file automatically when you save the project.

For more information about managing the `ejb-jar.xml` file in TopLink Workbench, see ["Working With the ejb-jar.xml File"](#) on page 21-15 for more information.

EJB 3.0 and the ejb-jar.xml File

If you are using EJB 3.0, you can use annotations to specify most of what you formerly specified in the `ejb-jar.xml` file. To override annotations or specify options not supported by annotations, you can still provide an `ejb-jar.xml` file in your EJB 3.0 application. For more information on what annotations are currently supported, see Oracle Containers for J2EE Enterprise JavaBeans Developer's Guide.

<J2EE-Container>-ejb-jar.xml File

The contents of the `<J2EE-Container>-ejb-jar.xml` file depend on the container to which you deploy your EJB. To create this file, use the tools that accompany your container.

In most cases, the `<J2EE-Container>-ejb-jar.xml` file integrates with TopLink without revision. However, in some cases, you must make some TopLink-specific modifications.

For more information, see the following:

- [OC4J and the orion-ejb-jar.xml File](#)
- [BEA WebLogic Server and the weblogic-ejb-jar.xml File](#)
- [EJB 3.0 and the <J2EE-Container>-jar.xml File](#)

OC4J and the orion-ejb-jar.xml File

[Table 8-1](#) summarizes the scenarios in which you may choose to modify the `orion-ejb-jar.xml` file.

Table 8–1 When to Modify the orion-ejb-jar.xml File

CMP Type	Mapping Type	Action
Orion	Specified in orion-ejb-jar.xml	1. Deploy.
Orion	Default mappings	1. Edit the orion-ejb-jar.xml file to set persistence-manager attribute name to orion. 2. Deploy.
TopLink	Specified in toplink-ejb-jar.xml (default persistence manager properties)	1. Deploy.
TopLink	Specified in toplink-ejb-jar.xml (custom persistence manager properties)	1. Edit the orion-ejb-jar.xml file to set persistence-manager attribute name to toplink. 2. Edit additional persistence-manager subentries (see "Configuring the orion-ejb-jar.xml File for OC4J" on page 8-9). 3. Deploy.
TopLink	Default mappings (no toplink-ejb-jar.xml)	1. Deploy.

For more information on configuring the orion-ejb-jar.xml file, see ["Configuring the orion-ejb-jar.xml File for OC4J"](#) on page 8-9.

BEA WebLogic Server and the weblogic-ejb-jar.xml File

For more information on configuring the weblogic-ejb-jar.xml, see ["Configuring the weblogic-ejb-jar.xml File for BEA WebLogic Server"](#) on page 8-13.

EJB 3.0 and the <J2EE-Container>-jar.xml File

If you are using EJB 3.0, you can use annotations to specify most of what you formerly specified in the <J2EE-Container>-ejb-jar.xml file. To override annotations or specify options not supported by annotations, you can still provide a <J2EE-Container>-ejb-jar.xml file in your EJB 3.0 application. For more information on what annotations are currently supported, see Oracle Containers for J2EE Enterprise JavaBeans Developer's Guide.

toplink-ejb-jar.xml File

The toplink-ejb-jar.xml file is used only in CMP projects. The TopLink runtime uses properties set in the <J2EE container>-ejb-jar.xml file (see ["<J2EE-Container>-ejb-jar.xml File"](#) on page 8-5) to locate the toplink-ejb-jar.xml file and read it in.

The purpose of toplink-ejb-jar.xml file depends on the type of application server you are using:

- [OC4J and the toplink-ejb-jar.xml File](#)
- [BEA WebLogic Server and the toplink-ejb-jar.xml File](#)
- [IBM WebSphere Application Server and the toplink-ejb-jar.xml File](#)
- [EJB 3.0 and the toplink-ejb-jar.xml File](#)

OC4J and the toplink-ejb-jar.xml File

When deploying a CMP application to OC4J, the toplink-ejb-jar.xml file is the name used for the project.xml file.

To create the `toplink-ejb-jar.xml` file in this case, simply rename your `project.xml` file. For more information, see ["project.xml File"](#) on page 8-2.

BEA WebLogic Server and the `toplink-ejb-jar.xml` File

When deploying a CMP application to BEA WebLogic Server, the `toplink-ejb-jar.xml` file contains a reference to the `project.xml` file.

[Example 8-2](#) shows a typical BEA WebLogic Server `toplink-ejb-jar.xml` file:

Example 8-2 BEA WebLogic Server `toplink-ejb-jar.xml` File

```
<?xml version="1.0"?>
<!DOCTYPE topleink-ejb-jar PUBLIC "-//Oracle Corp.//DTD TopLink CMP WebLogic 10.0.3 Developer
Preview//EN" "toplink-wls-ejb-jar_10_0_3.dtd">
<topleftink-ejb-jar>
  <session>
    <name>ejb20_EmployeeDemo</name>
    <project-xml>Employee.xml</project-xml>
    <login>
      <datasource>jdbc/JSTopLinkDS</datasource>
      <non-jts-datasource>jdbc/TopLinkDS</non-jts-datasource>
    </login>
    <customization-class>
      oracle.toplink.demos.ejb.cmp.wls.employee.EmployeeCustomizer
    </customization-class>
  </session>
</topleftink-ejb-jar>
```

For BEA WebLogic Server, you can specify an optional deployment customization class (that implements `oracle.toplink.ejb.cmp.DeploymentCustomization` interface) used to allow deployment customization of TopLink mapping and run-time configuration. In [Example 8-2](#), the deployment customization class is named `EmployeeCustomizer`. This deployment customization class must be fully qualified by its package name and included in the deployment JAR.

At deployment time, the TopLink runtime creates a new instance of this class and invokes its methods `beforeLoginCustomization` (before the TopLink runtime logs in to the session) and `afterLoginCustomization` (after the TopLink runtime logs in to the session), passing in the TopLink session as a parameter.

Use your implementation of the `beforeLoginCustomization` method to configure session attributes not supported by the `pm-properties`, including the following:

- cache coordination
- parameterized SQL
- native SQL
- batch writing/batch size
- byte-array/string binding
- EIS login
- event listeners
- table qualifier
- sequencing

In [Example 8-2](#), notice the `<non-jts-datasource>` element. The usage of this element is not limited to BEA WebLogic Server `toplink-ejb-jar.xml` file. Any time you use cache coordination with Java Transaction API (JTA), you need to

configure this element through TopLink Workbench (see ["Configuring a Nontransactional Read Login"](#) on page 89-3). This enables TopLink to do the out-of-context read operations and optimization of the sequencing assignments (see ["Externally Managed Transactional Data Sources"](#) on page 84-1).

For more information about session configuration, see ["Configuring a Session"](#) on page 77-1.

IBM WebSphere Application Server and the toplink-ejb-jar.xml File

When deploying a CMP application to IBM WebSphere application server, the `toplink-ejb-jar.xml` file is the name used for the `sessions.xml` file and contains a reference to the `project.xml` file.

To create the `toplink-ejb-jar.xml` file in this case, simply rename your `sessions.xml` file. For more information, see ["sessions.xml File"](#) on page 8-3.

EJB 3.0 and the toplink-ejb-jar.xml File

If you are using EJB 3.0, you can use annotations to specify most of what you formerly specified in the `toplink-ejb-jar.xml` file. To override annotations or specify options not supported by annotations, you can still provide a `toplink-ejb-jar.xml` file in your EJB 3.0 application. For more information on what annotations are currently supported, see Oracle Containers for J2EE Enterprise JavaBeans Developer's Guide.

Java Applications

In a Java application, TopLink does not use a J2EE container for deployment. Instead, it relies on TopLink mechanisms to provide functionality and persistence. The key elements of this type of application are the lack of a J2EE container and the fact that you deploy the application by placing the application JAR file on the classpath.

Java applications require the following deployment files:

- [project.xml File](#)
- [sessions.xml File](#)

JavaServer Pages and Servlet Applications

Many designers build TopLink applications that use JavaServer Pages (JSP) and Java servlets. This type of design generally supports Web-based applications.

JSP and servlet applications require the following deployment files:

- [project.xml File](#)
- [sessions.xml File](#)

Session Bean Applications

Session beans generally model a process, operation, or service and as such, are not persistent. You can build TopLink applications that wrap interaction with TopLink in session beans. Session beans execute all TopLink-related operations on behalf of the client.

This type of design uses JTS and externally managed transactions, but does not incur the overhead associated with CMP applications. Session bean applications also scale and deploy easily.

Session bean applications require the following deployment files:

- [project.xml File](#)
- [sessions.xml File](#)

CMP Applications

Many applications use the persistence mechanisms a J2EE container offers. TopLink provides full support for this type of application.

You can only use one persistence manager for all the CMP EJB in a JAR file.

CMP applications require the following deployment files:

- [ejb-jar.xml File](#)
- [<J2EE-Container>-ejb-jar.xml File](#)
- [toplink-ejb-jar.xml File](#)

BMP Applications

If you choose to write your own persistence code with BMP, you can take advantage of the classes in `oracle.toplink.ejb.bmp` package. Whether or not you use these classes, BMP applications require the following deployment files:

- [project.xml File](#)
- [sessions.xml File](#)
- [ejb-jar.xml File](#)

Configuring the orion-ejb-jar.xml File for OC4J

To deploy a TopLink application to OC4J 10g Release 3 (10.1.3) or later, modify the `orion-ejb-jar.xml` file as follows:

- [Configuring persistence-manager Entries](#)

If you are migrating an application from a previous release of OC4J, you can use the TopLink migration tool to automatically migrate persistence information from your `orion-ejb-jar.xml` file to a new `toplink-ejb-jar.xml`. For more information, see "[Migrating OC4J Orion Persistence to OC4J TopLink Persistence](#)" on page 7-5.

Configuring persistence-manager Entries

If you are using TopLink as your OC4J persistence manager, the default persistence manager in 10g Release 3 (10.1.3), you can configure the `persistence-manager` subentry (see [Table 8-2](#)) in the `orion-ejb-jar.xml` file. For more information on the scenarios in which you would want to modify `orion-ejb-jar.xml`, see "[OC4J and the orion-ejb-jar.xml File](#)" on page 8-5.

If you are not using TopLink as your OC4J persistence manager, do not modify the `persistence-manager` subentries.

OC4J 10g Release 3 (10.1.3) and later do not support `entity-deployment` attribute `pm-name`. Use `persistence-manager` attribute name instead (see [Table 8-2](#)). When OC4J parses the `orion-ejb-jar.xml` file, if it finds a `pm-name` attribute, OC4J ignores its value and logs the following warning message:

WARNING: Use of `pm-name` is unsupported and will be removed in a future release. Specify `pm` usage using `<persistence-manager> 'name'` instead.

Table 8–2 *orion-ejb-jar.xml File persistence-manager Entries*

Entry	Description
<code>name</code>	The name of the persistence manager to use. Set this value to <code>toplink</code> . If you set the <code>name</code> property to <code>toplink</code> , you may also configure <code>pm-properties</code> (see "Configuring pm-properties" on page 8-10).
<code>class-name</code>	Do not configure this attribute. If <code>name</code> is set to <code>toplink</code> , then <code>class-name</code> is set correctly by default.
<code>descriptor</code>	This property applies only when <code>name</code> is set to <code>toplink</code> . If you export your TopLink mapping metadata to a deployment XML file, set this property to the name of the deployment XML file (default: <code>toplink-ejb-jar.xml</code>). Do not set this property if you are using a TopLink project class instead of a mapping metadata file (see <code>project-class</code> in Table 8-3).

Configuring pm-properties

When you select TopLink as the persistence manager (see `name` in [Table 8–2](#)), use the `persistence-manager` subentries for `pm-properties` (see [Table 8–3](#)) to configure the TopLink session that the TopLink run time creates and uses internally for CMP projects. The `persistence-manager` subentries take the place of a `sessions.xml` file in a CMP project.

Note: You can only configure a subset of session features using these properties and in most cases, default configuration applies. To configure all session features and to override defaults, you must use a customization class (see `customization-class` in [Table 8-3](#)).

Table 8–3 *orion-ejb-jar.xml File persistence-manager Subentries for pm-properties*

Entry	Description
<code>session-name</code>	Unique name for this TopLink-persisted EJB deployment JAR file. Must be unique among all TopLink-persisted deployed JAR files in this application server instance. When the TopLink run time internally creates a TopLink session for this TopLink-persisted deployed JAR file, the TopLink session manager stores the session instance under this <code>session-name</code> . For more information about the session manager, see "Acquiring and Using Sessions at Run Time" on page 78-1). If you do not specify a name, the TopLink runtime will generate a unique name.
<code>project-class</code>	If you export your TopLink mapping metadata to a Java class (that extends <code>oracle.toplink.sessions.Project</code>), set this property to the name of the class, fully qualified by its package name. Be sure to include the class file in the deployable JAR file. Do not set this property if you are using a mapping metadata file (see <code>descriptor</code> in Table 8-2).

Table 8–3 (Cont.) orion-ejb-jar.xml File persistence-manager Subentries for

Entry	Description
customization-class	Optional Java class (that implements <code>oracle.toplink.ejb.cmp.DeploymentCustomization</code>) used to allow deployment customization of TopLink mapping and run-time configuration. At deployment time, the TopLink run time creates a new instance of this class and invokes its methods <code>beforeLoginCustomization</code> (before the TopLink run time logs into the session) and <code>afterLoginCustomization</code> (after the TopLink runtime logs into the session), passing in the TopLink session as a parameter. Use your implementation of the <code>beforeLoginCustomization</code> method to configure session attributes not supported by the <code>pm-properties</code> including: cache coordination (see also "Configuring cache-synchronization Properties" on page 8-11), parameterized SQL, native SQL, batch writing/batch size, byte-array/string binding, EIS login, event listeners, table qualifier, and sequencing. For more information about session configuration, see "Configuring a Session" on page 77-1. The class must be fully qualified by its package name and included in the deployment JAR file.
db-platform-class	Optional TopLink database platform class (instance of <code>oracle.toplink.platform.database</code> or <code>oracle.toplink.platform.database.oracle</code>) containing TopLink support specific to a particular database. Set this value to the database platform class that corresponds to the database that your application uses. The class must be fully qualified by its package name.
remote-relationships	Optional flag to allow relationships between remote objects. This flag may be used as a way to migrate from TopLink EJB 1.1 entities (when relationships were created between remote entities). Using this flag does not comply with EJB 2.0. Valid values are: ■ <code>true</code>: All relationships will be maintained through the remote interfaces of the entities ■ <code>false</code>: Disables this feature.
cache-synchronization	See "Configuring cache-synchronization Properties" on page 8-11.
default-mapping	See "Configuring default-mapping Properties" on page 8-12.

Configuring cache-synchronization Properties

When you select TopLink as the persistence manager (see name in [Table 8–2](#)), use the `pm-properties` subentry for `cache-synchronization` (see [Table 8–4](#)) to configure TopLink cache coordination features of the session that the TopLink run time uses internally for CMP projects. For more information about TopLink cache coordination, see ["Understanding Cache Coordination"](#) on page 90-10.

When this subentry is present, you must use a customization class (see `customization-class` in [Table 8–3](#)) to complete cache coordination configuration. For more information about TopLink cache coordination configuration, see ["Configuring a Coordinated Cache"](#) on page 91-1.

Table 8–4 orion-ejb-jar.xml File pm-properties Subentries for cache-synchronization

Entry	Description
mode	An indicator of whether or not cache coordination updates should be propagated to other servers synchronously or asynchronously. Valid values are: ■ <code>asynchronous</code> (default) ■ <code>synchronous</code>

Table 8–4 (Cont.) orion-ejb-jar.xml File pm-properties Subentries for

Entry	Description
server-url	For a JMS coordinated cache: assuming that you are using the Oracle Application Server Containers for J2EE (OC4J) JNDI naming service and that all the hosts in your coordinated cache can communicate using OC4J proprietary RMI protocol ORMI, use a URL like: <pre>ormi://<JMS-host-IP>:<JMS-host-port></pre> where JMS-host-IP is the IP address of the host on which the JMS service provider is running and JMS-host-port is the port on which the JMS service provider is listening for JMS requests. For an RMI or CORBA coordinated cache: assuming that you are using the OC4J JNDI naming service and that all the hosts in your coordinated cache can communicate using OC4J proprietary RMI protocol ORMI on OC4J default port 23791, use a URL like: <pre>ormi://<session-host-IP>:23791</pre> where session-host-IP is the IP address of the host on which this session is deployed.
server-user	Optional username required to log in to the JNDI naming service.

Configuring default-mapping Properties

When you select TopLink as the persistence manager (see name in [Table 8–2](#)), use the pm-properties subentry for default-mapping (see [Table 8–5](#)) to configure the TopLink default mapping and automatic table generation feature.

For more information about TopLink default mappings, see "[Default Mapping in EJB 2.0 or 3.0 CMP Projects Using OC4J at Run Time](#)" on page 33-4.

For more information about TopLink automatic table generation, see "[Automatic Database Table Creation](#)" on page 6-6.

Table 8–5 orion-ejb-jar.xml File pm-properties Subentries for default-mapping

Entry	Description
db-table-gen	Optional element that determines what TopLink will do to prepare the database tables that are being mapped to. Valid values are: ■ Create (default): This value tells TopLink to create the mapped tables during the deployment. If the tables already exist, TopLink will log an appropriate warning messages (such as "<i>Table already existed...</i>") and keeps processing the deployment. ■ DropAndCreate: This value tells TopLink to drop tables before creating them during deployment. If a table does not initially exist, the drop operation will cause anSQLException to be thrown through the driver. However, TopLink handles the exception (logs and ignores it) and moves on to process the table creation operation. The deployment fails only if both drop and create operations fail. ■ UseExisting: This value tells TopLink to perform no table manipulation. If the tables do not exist, deployment still goes through without error. If no orion-ejb-jar.xml file is defined in your EAR file, the OC4J container generates one during deployment. In this case, to specify a value for db-table-gen, use the TopLink system property <code>toplink.defaultmapping.dbTableGenSetting</code>. For example: <code>-Dtoplink.defaultmapping.dbTableGenSetting="DropAndCreate"</code>. The orion-ejb-jar.xml property overrides the system property. If both the orion-ejb-jar.xml property and the system property are present, TopLink retrieves the setting from the orion-ejb-jar.xml file. This setting overrides autocreate-tables and autodelete-tables configuration at the application (EAR) or system level. For more information, see "Automatic Database Table Creation" on page 6-6.

Table 8–5 (Cont.) orion-ejb-jar.xml File pm-properties Subentries for default-mapping

Entry	Description
extended-table-names	An element used if the generated table names are not long enough to be unique. Values are restricted to <code>true</code> or <code>false</code> (default). When set to <code>true</code>, the TopLink run time will ensure that generated tables names are unique. In default mapping, each entity is mapped to one table. The only exception is in many-to-many mappings where there is one extra relation table involved in the source and target entities. When <code>extended-table-names</code> is set to <code>false</code> (the default), a simple table naming algorithm is used as follows: table names are defined as <code>TL_<bean_name></code>. For example, if the bean name is <code>Employee</code>, the associated table name would be <code>TL_EMPLOYEE</code>. However, if the same entity is defined in multiple JAR files in an application, or across multiple applications, table-naming collision is inevitable. To address this problem, set <code>extended-table-names</code> to <code>true</code>. When set to <code>true</code>, TopLink uses an alternative table-naming algorithm as follows: table names are defined as <code><bean_name>_<jar_name>_<app_name></code>. This algorithm uses the combination of bean, JAR, and EAR names to form a table name unique across the application. For example, given a bean named <code>Employee</code>, which is in <code>Test.jar</code>, which is in <code>Demo.ear</code> (and the application name is "Demo"), then the corresponding table name will be <code>EMPLOYEE_TEST_DEMO</code>. If there is no <code>orion-ejb-jar.xml</code> file defined in the EAR file, the OC4J container generates one during deployment. In this case, to specify a value for <code>extended-table-names</code>, use the TopLink system property <code>toplink.defaultmapping.useExtendedTableNames</code>. For example: <code>-Dtoplink.defaultmapping.useExtendedTableNames="true"</code>. The <code>orion-ejb-jar.xml</code> property overrides the system property. If both the <code>orion-ejb-jar.xml</code> property and the system property are present, TopLink retrieves the setting from the <code>orion-ejb-jar.xml</code> file.

Configuring the weblogic-ejb-jar.xml File for BEA WebLogic Server

To deploy a TopLink application to a BEA WebLogic Server, modify the `weblogic-ejb-jar.xml` file as described in:

- [Configuring persistence-descriptor Entries](#)

Avoid the `weblogic-ejb-jar.xml` tags that TopLink either does not support or does not require (see "[Unsupported weblogic-ejb-jar.xml File Tags](#)" on page 8-14).

If you are migrating a BEA WebLogic Server application to OC4J, you can use the TopLink migration tool to automatically migrate persistence information from your `weblogic-ejb-jar.xml` file to a new `toplink-ejb-jar.xml` file. For more information, see "[Migrating BEA WebLogic Persistence to OC4J TopLink Persistence](#)" on page 7-16.

Configuring persistence-descriptor Entries

Within the `weblogic-ejb-jar.xml` file, each bean must have a `persistence-descriptor` entry with subentries, as follows:

- Configure the `persistence-descriptor` entry with subentries that indicate TopLink is available and should be used:
 - If you deploy to WebLogic 6.1 (Service Pack 4), include a `persistence-type` element and a `persistence-use` element. Both elements require a `type-identifier` and a `type-version` tag. [Table 8–6](#) lists the options for the `type-identifier` tag, and [Table 8–7](#) lists the options for the `type-version` tag.

- If you deploy to WebLogic 7.0 or 8.1, include a `persistence-use` element with a `type-identifier` and a `type-version` tag. [Table 8-6](#) lists the options for the `type-identifier` tag, and [Table 8-7](#) lists the options for the `type-version` tag.
- If you use WebLogic 6.1, add the element `type-storage` to the `persistence-type` element, and set it to `META-INF\toplink-ejb-jar.xml`.
- If you use WebLogic 7.0 or 8.1, add the element `type-storage` to the `persistence-use` element, and set it to `META-INF\toplink-ejb-jar.xml`.
- Set the `enable-call-by-reference` element to `TRUE` to enable call-by-reference:

```
<weblogic-enterprise-bean>
  <ejb-name>AccountBean</ejb-name>
  ...
    <enable-call-by-reference>True</enable-call-by-reference>
  ...
</weblogic-enterprise-bean>
```

Table 8-6 WebLogic type-identifier Settings

EJB Version	XML Elements
1.1	<code><type-identifier>TopLink_CMP_1_1</type-identifier></code>
2.0	<code><type-identifier>TopLink_CMP_2_0</type-identifier></code>

Table 8-7 WebLogic type-version Settings

WebLogic EJB Version	XML Elements
6.1	<code><type-version>4.0</type-version></code>
7.0	<code><type-version>4.5</type-version></code>
8.1	<code><type-version>9.0.4</type-version></code>

Note: Although deprecated, the `type-version` setting of version 3.5 also functions correctly with WebLogic 6.1 (Service Pack 4) under EJB 1.1.

Unsupported weblogic-ejb-jar.xml File Tags

The `weblogic-ejb-jar.xml` file includes several tags that TopLink either does not support or does not require:

- `concurrency-strategy`: This tag specifies how WebLogic manages concurrent users for a given bean. Because TopLink manages concurrent access internally, it does not require this tag.

For more information about the TopLink concurrency strategy, see ["Configuring Locking Policy"](#) on page 28-62.

- `db-is-shared`: Because TopLink does not make any assumptions about the exclusivity of database access, TopLink does not require this tag. TopLink addresses multiuser access issues through various locking and refreshing policies.
- `delay-updates-until-end-of-tx`: TopLink always delays updates until the end of a transaction, and does not require this tag.

- **finders-load-bean**: TopLink always loads the bean upon execution of the finder, and does not require this tag.
- **pool**: TopLink does not use a pooling strategy for entity beans. This avoids object-identity problems that can occur due to pooling.
- **lifecycle**: This element manages beans that follow a pooling strategy. Because TopLink does not use a pooling strategy, TopLink ignores this tag.
- **is-modified-method-name**: TopLink does not require a bean developer-defined method to detect changes in the object state.
- **isolation-level**: Because isolation level settings for the cache or database transactions are specified in the TopLink project, TopLink ignores this tag.
- **cache**: Because you define TopLink cache properties in TopLink Workbench, this tag is unnecessary.

Packaging a TopLink Application

How you package the components of your application depends on the type of application and how you plan to deploy it.

This section describes TopLink-specific details applicable to the common packaging strategies used for the following types of application:

- [Java Applications](#)
- [JavaServer Pages and Servlet Applications](#)
- [Session Bean Applications](#)
- [CMP Applications](#)
- [BMP Applications](#)

If you are using EJB 3.0, you may be using annotations instead of some deployment files. Include deployment descriptors to override annotations or specify options not supported by annotations.

For information, see:

- ["Integrating TopLink With an Application Server"](#) on page 7-1
- ["Creating TopLink Files for Deployment"](#) on page 8-1
- ["Deploying a TopLink Application"](#) on page 10-1

Java Applications

For non-J2EE Java applications, it is common to package the application in a single JAR file as [Figure 9-1](#) shows.

Figure 9-1 Packaging a non-J2EE Java Application

```
domain_module.jar
  Java classes that represent the objects mapped
  project.xml
  session.xml
  META-INF
    Manifest.mf
```

This JAR contains the TopLink files and domain objects required by the application, including:

- [sessions.xml File](#)

- [project.xml File](#) (or the compiled project class file if you are not using XML files for deployment)
- The mapped classes required by the application, in a fully-resolved directory structure

When you create the JAR file, the JAR building utility automatically creates a directory structure within the JAR. Ensure that the `sessions.xml` file and the `project.xml` file (or project class file) appear at the root of the JAR file. Ensure that the class directory structure starts at the root of the JAR.

If you do not store the `project.xml` or `sessions.xml` files at the root of the JAR file, see ["Packaging With TopLink Metadata File Resource Paths"](#) on page 9-7.

JavaServer Pages and Servlet Applications

For simple J2EE applications without EJB, it is common to package the application in an Enterprise Archive (EAR) file made up of various J2EE application component archives as [Figure 9-2](#) shows.

Figure 9-2 Packaging a J2EE JSP or Servlet Application Without EJB

```
appname.ear
  META-INF
    application.xml
    orion-application.xml
  domain_module.jar
    Java classes that represent the objects mapped
    project.xml
    session.xml
    META-INF
      Manifest.mf
  web_module.war
    html pages, JSP's, etc.
    META-INF
      web.xml
      orion-web.xml
    classes
      servlet classes
    lib
  client_module.jar
    Client classes
    META-INF
      application-client.xml
      orion-application-client.xml
```

The component archives with TopLink dependencies include:

- [TopLink Domain JAR](#)

TopLink Domain JAR

The domain JAR contains the TopLink files and domain objects required by the application, including:

- [sessions.xml File](#)
- [project.xml File](#) (or the compiled Project class file if you are not using XML files for deployment)
- The mapped classes required by the application, in a fully-resolved directory structure

When you create the JAR file, the JAR building utility automatically creates a directory structure within the JAR. Ensure that the `sessions.xml` file and the `project.xml` file (or `project.class` file) appear at the root of the JAR file. Also ensure that the class directory structure starts at the root of the JAR.

If you do not store the `project.xml` or `sessions.xml` files at the root of the JAR file, see ["Packaging With TopLink Metadata File Resource Paths"](#) on page 9-7.

Session Bean Applications

For J2EE applications with non-entity EJB, it is common to package the application in an Enterprise Archive (EAR) file made up of various J2EE application component archives as [Figure 9-3](#) shows.

Figure 9-3 Packaging a J2EE Application With Non-Entity EJB

```

appname.ear
  META-INF
    application.xml
    orion-application.xml
  ejb_module_X.jar
    EJB classes - session and non-entity beans
    META-INF
      ejb-jar.xml
      orion-ejb-jar.xml - no persistence-manager subentries
  domain_module.jar
    Java classes that represent the objects mapped
    project.xml
    session.xml
    META-INF
      Manifest.mf
  web_module.war
    html pages, JSP's, etc.
    META-INF
      web.xml
      orion-web.xml
    classes
      servlet classes
    lib
  client_module.jar
    Client classes
    META-INF
      application-client.xml
      orion-application-client.xml
  
```

The component archives with TopLink dependencies include:

- [TopLink Domain JAR](#)
- [EJB JAR](#)

TopLink Domain JAR

The domain JAR contains the TopLink files and domain objects required by the application, including:

- [sessions.xml File](#)
- [project.xml File](#) (or the compiled `project.class` file if you are not using XML files for deployment)
- The mapped classes required by the application, in a fully-resolved directory structure

When you create the JAR file, the JAR building utility automatically creates a directory structure within the JAR. Ensure that the `sessions.xml` file and the `project.xml` file (or `project.class` file) appear at the root of the JAR file. Also ensure that the class directory structure starts at the root of the JAR.

If you do not store the `project.xml` or `sessions.xml` files at the root of the JAR file, see ["Packaging With TopLink Metadata File Resource Paths"](#) on page 9-7.

EJB JAR

In this type of application, the EJB JAR contains non-entity EJB. Consequently, its `orion-ejb-jar.xml` does not contain `persistence-manager` or `pm-properties` entries. These entries apply only to CMP applications.

CMP Applications

For J2EE applications that use CMP to persist entity EJB, it is common to package the application in an Enterprise Archive (EAR) file made up of various J2EE application component archives as [Figure 9-4](#) shows.

Figure 9-4 Packaging a J2EE Application With CMP Entity EJB

```
appname.ear
  META-INF
    application.xml
    orion-application.xml
  cmp_ejb_module_1.jar
    EJB classes - cmp entity beans
    META-INF
      ejb-jar.xml
      orion-ejb-jar.xml - includes persistence-manager properties
      toplink-ejb-jar.xml
  ejb_module_X.jar
    EJB classes - non-entity beans
    META-INF
      ejb-jar.xml
      orion-ejb-jar.xml - no persistence-manager subentries
  web_module.war
    html pages, JSP's, etc.
    META-INF
      web.xml
      orion-web.xml
    classes
      servlet classes
    lib
  client_module.jar
    Client classes
    META-INF
      application-client.xml
      orion-application-client.xml
```

The component archives with TopLink dependencies include:

- [EJB JAR](#)

EJB JAR

In this type of application, the EJB JAR file specifically service both non-entity and entity EJB. It includes:

- The home and remote, and all implementation code for all mapped beans in the application.

- All mapped non-EJB classes from the TopLink Workbench project
- The home and remote, and all implementation code for any session beans included in the application.
- Helper classes that contain TopLink amendment methods, and any other classes the application requires.

For example, an instance of `oracle.toplink.ejb.cmp.DeploymentCustomization` (for more information, see `customization-class` in [Table 8-3](#) in "[Configuring persistence-manager Entries](#)" on page 8-9).

Store the following XML files in the EJB JAR `\meta-inf` directory:

- [ejb-jar.xml File](#)
- [<J2EE-Container>-ejb-jar.xml File](#)
- [toplink-ejb-jar.xml File](#)

Note: If you do not use XML files for deployment, include your compiled `oracle.toplink.sessions.Project` file at the root of the EJB JAR (not in the `\meta-inf` directory).

Because the EJB JAR contains both non-entity and entity EJB, if you are using OC4J or BEA WebLogic (WLS), the `<J2EE-Container>-ejb-jar.xml` must contain `persistence-manager` and `pm-properties` entries. For more information, see "[Configuring persistence-manager Entries](#)" on page 8-9.

You must persist all of the entity EJB to the same data source. For a CMP application, TopLink does not support session broker functionality (see "[Session Broker and Client Sessions](#)" on page 75-25).

BMP Applications

For J2EE applications that use BMP to persist entity EJB, it is common to package the application in an Enterprise Archive (EAR) file made up of various J2EE application component archives as [Figure 9-5](#) shows.

Figure 9–5 Packaging a J2EE Application With BMP Entity EJB

```
appname.ear
  META-INF
    application.xml
    orion-application.xml
  bmp_ejb_module_1.jar
    EJB classes - bmp entity beans
    META-INF
      ejb-jar.xml
      orion-ejb-jar.xml - includes persistence-manager properties
      toplink-ejb-jar.xml
  ejb_module_X.jar
    EJB classes - non-entity beans
    META-INF
      ejb-jar.xml
      orion-ejb-jar.xml - no persistence-manager subentries
  domain_module.jar
    Java classes that represent the objects mapped
    project.xml
    session.xml
    META-INF
      Manifest.mf
  web_module.war
    html pages, JSP's, etc.
    META-INF
      web.xml
      orion-web.xml
    classes
      servlet classes
    lib
  client_module.jar
    Client classes
    META-INF
      application-client.xml
      orion-application-client.xml
```

The component archives with TopLink dependencies include:

- [TopLink Domain JAR](#)
- [EJB JAR](#)

TopLink Domain JAR

The domain JAR contains the TopLink files and domain objects required by the application, including:

- [sessions.xml File](#)
- [project.xml File](#) (or the compiled project.class file if you are not using XML files for deployment)
- The mapped classes required by the application, in a fully-resolved directory structure

When you create the JAR file, the JAR building utility automatically creates a directory structure within the JAR. Ensure that the `sessions.xml` file and the `project.xml` file (or `project.class` file) appear at the root of the JAR file. Also ensure that the class directory structure starts at the root of the JAR.

If you do not store the `project.xml` or `sessions.xml` files at the root of the JAR file, see ["Packaging With TopLink Metadata File Resource Paths"](#) on page 9-7.

EJB JAR

In this type of application, the EJB JAR file specifically services both non-entity and entity EJB. It includes:

- The home and remote, and all implementation code for all mapped beans in the application
- All mapped non-EJB classes from the TopLink Workbench project
- The home and remote, and all implementation code for any session beans included in the application
- Helper classes that contain TopLink amendment methods, and any other classes the application requires

Store the following XML files in the EJB JAR \meta-inf directory:

- [ejb-jar.xml File](#)
- [<J2EE-Container>-ejb-jar.xml File](#)

Because the EJB JAR does not contain entity CMP EJB, its `orion-ejb-jar.xml` must not contain `persistence-manager` or `pm-properties` entries.

For more information, see ["Configuring persistence-manager Entries"](#) on page 8-9.

Packaging With TopLink Metadata File Resource Paths

If you do not store the `project.xml` or `sessions.xml` files at the root of the JAR file, then you must provide the full resource path to the files when accessing them. Ensure that you use "/" in resources paths, not "\". Using "\" will not work in Java.

For example, in the `jar` element, reference the `project.xml` and `sessions.xml` files as:

```
<jar>/myapp/ordersys/persist/sessions.xml
<jar>/myapp/ordersys/persist/project.xml
```

In the `sessions.xml` file, reference the `project.xml` as:

```
myapp/ordersys/persist/project.xml
```

To acquire the session, use:

```
SessionManager.getManager().getSession(
    new XMLSessionConfigLoader("myapp/ordersys/persist/sessions.xml"),
    "OrdersysSession",
    getClass().getClassLoader()
);
```

For more information about acquiring sessions at run time, see ["Acquiring a Session from the Session Manager"](#) on page 78-3.

Deploying a TopLink Application

This chapter includes deployment information on the following TopLink applications:

- [Java Applications](#)
- [JavaServer Pages and Servlets](#)
- [Session Bean Applications](#)
- [CMP Applications](#)
- [BMP Applications](#)

For more information, see the following:

- ["Integrating TopLink With an Application Server"](#) on page 7-1
- ["Creating TopLink Files for Deployment"](#) on page 8-1
- ["Packaging a TopLink Application"](#) on page 9-1

Java Applications

Build the JAR file (see ["Java Applications"](#) on page 9-1) and place it on the classpath.

For more information on accessing TopLink from your client application, see ["Acquiring and Using Sessions at Run Time"](#) on page 78-1.

JavaServer Pages and Servlets

After you build the WAR and JAR files (see ["JavaServer Pages and Servlet Applications"](#) on page 9-2), build them into an EAR file for deployment. To deploy the EAR to your JSP servlet server, copy the EAR to a commonly used directory. You may also need to use server-specific deployment tools. For more information, see the server documentation.

For more information on accessing TopLink from your client application, see ["Loading a Session from sessions.xml with an Alternative Class Loader"](#) on page 78-4.

Session Bean Applications

After you build the WAR and JAR files (see ["Session Bean Applications"](#) on page 9-3), build them into an EAR file for deployment. To deploy the EAR file to your J2EE server, copy the EAR to a commonly used directory. You may also need to use server-specific deployment tools. For more information, see the server documentation.

For more information on accessing TopLink from your client application, see ["Loading a Session from sessions.xml with an Alternative Class Loader"](#) on page 78-4.

Optionally, you may also consider ["Hot Deployment of EJB"](#) on page 10-4.

CMP Applications

After you build the WAR and JAR files (see ["CMP Applications"](#) on page 9-4), build them into an EAR file for deployment. To deploy the EAR file to your J2EE server, copy the EAR to a commonly used directory. You may also need to use server-specific deployment tools. For more information, see the server documentation.

This section describes:

- [Deploying a CMP Application to OC4J](#)
- [Deploying a CMP Application to BEA WebLogic Server](#)
- [Deploying a CMP Application to IBM WebSphere Application Server 4.0](#)

For additional information on server-specific configuration, see [Chapter 7, "Integrating TopLink With an Application Server"](#).

Optionally, you may also consider ["Hot Deployment of EJB"](#) on page 10-4.

Deploying a CMP Application to OC4J

The most efficient way to deploy a CMP application to OC4J is using Oracle Enterprise Manager 10g. For more information, see *Oracle Application Server Administrator's Guide*.

When you deploy a CMP application to OC4J:

- It performs a partial EJB conformance check on the beans and their associated interfaces.
- It builds the internal OC4J classes that manage security and transactions, as well as the RMI stubs and skeletons that enable client access to the beans.
- TopLink builds concrete bean subclasses and EJB finder method implementations.

Deploying a CMP Application to BEA WebLogic Server

TopLink CMP support includes integration for BEA WebLogic Server. To enable TopLink CMP for BEA WebLogic entity beans, use the WebLogic EJB Compiler (ejbc) to compile the EJB JAR file, as follows:

- Run ejbc from the command line. Include the EJB JAR file as a command line argument. ejbc creates an EJB JAR file that contains the original classes as well as all required generated classes and files.

When you run ejbc:

- It performs a partial EJB conformance check on the beans and their associated interfaces.
- It builds the internal BEA WebLogic classes that manage security and transactions, as well as the RMI stubs and skeletons that enable client access to the beans.
- TopLink builds concrete bean subclasses and EJB finder method implementations.

For more information about running ejbc, see the BEA WebLogic Server documentation.

Troubleshooting ejbc

When you start ejbc, it processes the data in a series of stages. If errors occur while running ejbc, attempt to determine which stage causes the problem. Common problems include the following:

- Bean classes that do not conform with the EJB specification
- Classes missing from the classpath (all domain classes, required TopLink classes, and all required BEA WebLogic classes must be on the classpath)
- Java compiler (javac) problems, often caused by using an incorrect version of the JDK
- A failure when generating the RMI stubs and skeletons (a failure of RMI compiler (rmic))

Refer to [Chapter 15, "Troubleshooting Application Deployment"](#) for additional information.

Deploying a CMP Application to IBM WebSphere Application Server 4.0

TopLink CMP support includes an integration for IBM WebSphere application server 4.0. Use the following procedure to deploy your application to WebSphere application server:

1. Use the TopLink Deploy Tool for WebSphere to compile the EJB JAR file. For more information, see ["Using the WebSphere Deploy Tool"](#) on page 10-5.
2. Start the WebSphere Administration Server.
3. Start the Administration Console and deploy the compiled JAR file.

For more information about deploying the JAR file, see the IBM WebSphere application server documentation.

Note: When you deploy an application that contains an entity bean, set up a data source and associate it with the bean. For more information about how to create and associate data sources, see the IBM WebSphere application server documentation.

It is not necessary to deploy the EJB JAR file in WebSphere Studio Application Developer (WSAD), because deployment is carried out using the Deploy Tool (see ["Using the WebSphere Deploy Tool"](#) on page 10-5).

Starting the Entity Bean

You can start the bean in either the WebSphere application server or in WSAD.

To start the bean in IBM WebSphere application server, do the following:

1. Select the application that contains the entity beans.
2. Right-click and choose **Start**.

A message dialog box appears if the bean starts successfully. If an error occurs, consult [Part V, "Troubleshooting a TopLink Application"](#) for troubleshooting information.

To start the bean in WSAD, do the following:

1. Right-click the EJB project, and choose **Run on Server**.
2. To view the status of the process, open the Console tab of the Server view.

BMP Applications

After you build the WAR and JAR files, build them into an EAR file for deployment. To deploy the EAR file to your J2EE server, copy the EAR to a commonly used directory. You may also need to use server-specific deployment tools. For more information, see the server documentation.

For additional information on server-specific configuration, see [Chapter 7, "Integrating TopLink With an Application Server"](#).

Optionally, you may also consider ["Hot Deployment of EJB"](#) on page 10-4.

Hot Deployment of EJB

Many J2EE containers support *hot deployment*, a feature that enables you to deploy EJB on a running server. Hot deployment allows you to do the following:

- Deploy newly developed EJB to a running production system.
- Remove (undeploy) deployed EJB from a running server.
- Modify (redeploy) the behavior of deployed EJB by updating the bean class definition.

The client receives deployment exceptions when attempting to access undeployed or re-deployed bean instances. The client application must catch and handle the exceptions.

How you configure hot deployment of EJB depends on the type of J2EE application you are deploying:

- [Hot Deployment in a CMP Application](#)
- [Hot Deployment in a non-CMP Application](#)

For more information about hot deployment, see the J2EE container documentation.

Hot Deployment in a CMP Application

When you take advantage of hot deployment in a CMP application, consider the following:

- You must deploy all related beans (all beans that share a common TopLink project) within the same EJB JAR file. Because TopLink views deployment on a project level, deploy all the project beans (rather than just a portion of them) to maintain consistency across the project.
- When you redeploy a bean, you automatically reset its TopLink project. This flushes all object caches and rolls back any active object transactions associated with the project.

Hot Deployment in a non-CMP Application

When you take advantage of hot deployment in a non-CMP application, you must refresh the TopLink session using the `SessionManager` method `getSession` with the appropriate arguments (see ["Refreshing a Session when the Class Loader Changes"](#) on page 78-6).

If you do not use this `SessionManager` method, then your application is responsible for destroying or refreshing the session when a hot deployment (or hot redeployment) occurs.

Using the WebSphere Deploy Tool

TopLink integration for IBM WebSphere application server includes a deployment tool that helps you deploy your projects to WebSphere. The Deploy Tool for WebSphere is a graphical tool that makes project deployment to WebSphere easier to configure and execute. The Deploy Tool also includes a command-line option that lets you deploy your project while bypassing the graphical interface element of the tool.

You can use the Deploy Tool on its own (see ["Using the Deploy Tool on its Own"](#) on page 10-5) or with the WebSphere Studio Application Developer (see ["Using the Deploy Tool With WebSphere Studio Application Developer"](#) on page 10-5).

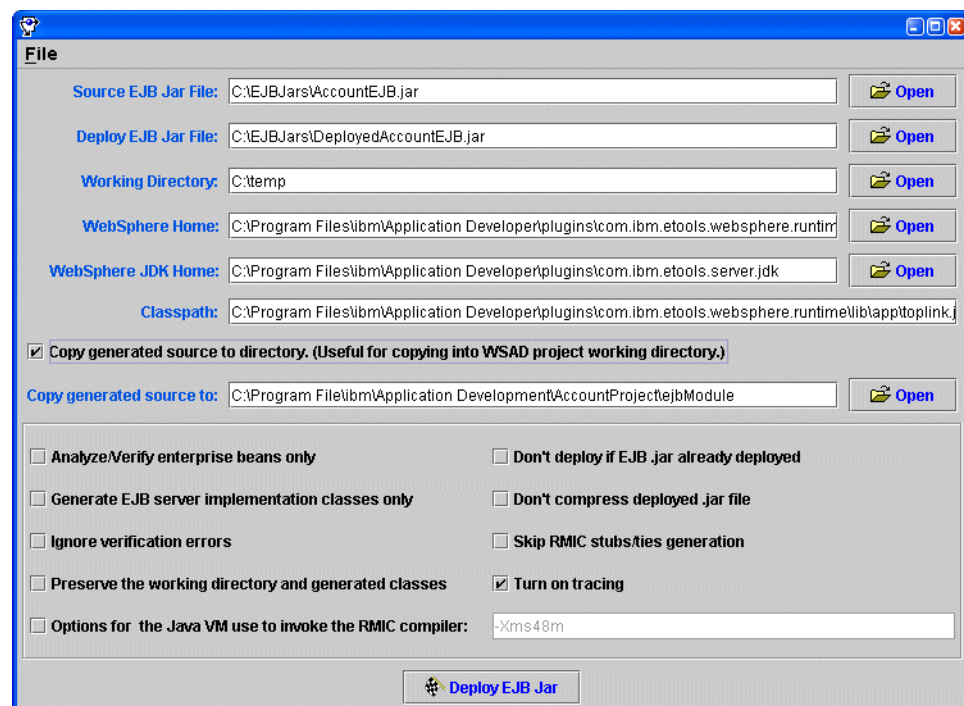
Using the Deploy Tool on its Own

To deploy a JAR file, use this procedure:

1. Start the WebSphere Deploy Tool by executing the appropriate `<TOPLINK_HOME>/bin/wasDeployTool` script: on Windows, execute `wasDeployTool.cmd`; on UNIX, execute `wasDeployTool.sh`.

The WebSphere Deploy Tool dialog box appears (see [Figure 10-1](#)).

Figure 10-1 The Deploy Tool Set Up for Use with WSAD



2. Enable the **Copy generated source to directory** option to save a copy of the generated code in the specified directory. This is a quick and efficient way to copy the files into a WSAD project working directory.
3. Enable the **Turn on tracing** option if you want to see the details of the process.
4. Click **Deploy EJB**.

Using the Deploy Tool With WebSphere Studio Application Developer

The Deploy tool is compatible with the WSAD.

To deploy from the Deploy Tool to WSAD, do the following:

1. Select the EJB Project in WSAD and choose to generate **Deploy and RMIC Code**.
2. Export the EJB Project to an EJB JAR file, making sure that the TopLink project and `toplink-ejb-jar.xml` files are included in the EJB JAR file.
3. Start the WebSphere Deploy Tool by executing the appropriate `<TOPLINK_HOME>/bin/wasDeployTool` script: on Windows, execute `wasDeployTool.cmd`; on UNIX, execute `wasDeployTool.sh`.

The WebSphere Deploy Tool dialog appears (see [Figure 10-1](#)).

4. Choose the EJB Project working directory so that TopLink overrides the WSAD deploy code with the TopLink deploy code.
5. If the source is copied to a directory other than the WSAD EJB Project directory, manually copy the source files to the WSAD EJB Project under the **ejbModule directory** of the project.
6. Enter appropriate directories in the fields of the Deploy Tool.
7. Choose **Deploy EJB JAR** to create the deployed EJB JAR file.
8. Choose **Rebuild all** from the Project menu to compile the deploy code to incorporate CMP.

Troubleshooting

The most common error you might encounter when you use the Deploy Tool is the `NoClassDefFoundError` exception. To resolve this error condition, add the required resources to the **Classpath**. The **Turn on tracing** option also helps to debug errors during deployment code generation.

When an obscure error appears during the generating stub phase, copy the Java command and run it at the command prompt. This gives a more detailed error message.

See [Chapter 7, "Integrating TopLink With an Application Server"](#) for additional information.

Part IV

Optimizing and Customizing a TopLink Application

This part describes how to optimize and customize a TopLink application. It contains the following chapters.

- [Chapter 11, "Optimization"](#)

This chapter contains information on the diverse set of features TopLink provides to optimize performance.

- [Chapter 12, "Customization"](#)

This chapter describes how to customize various aspects of TopLink, based on your application's specific needs.

TopLink provides a diverse set of features to measure and optimize application performance. You can enable or disable most features in the descriptors or session, making any resulting performance gains global.

This chapter includes the following sections:

- [Understanding Optimization](#)
- [Sources of Application Performance Problems](#)
- [Measuring Performance With the TopLink Profiler](#)
- [Measuring Performance With the Oracle Dynamic Monitoring System \(DMS\)](#)

Understanding Optimization

Performance considerations are present at every step of the development cycle. Although this implies an awareness of performance issues in your design and implementation, it does not mean that you should expect to achieve the best possible performance in your first pass.

For example, if optimization complicates the design, leave it until the final development phase. You should still plan for these optimizations from your first iteration, to make them easier to integrate later.

The most important concept associated with tuning your TopLink application is the idea of an iterative approach. The most effective way to tune your application is to do the following:

1. Measure application performance using the TopLink profiler (see "[Measuring Performance With the TopLink Profiler](#)" on page 11-2) or the Oracle Dynamic Monitoring System (DMS) profiler (see "[Measuring Performance With the Oracle Dynamic Monitoring System \(DMS\)](#)" on page 11-4)
2. Modify application components (see "[Sources of Application Performance Problems](#)")
3. Measure performance again.

To identify the changes that improve your application performance, modify only one or two components at a time. You should also tune your application in a nonproduction environment before you deploy the application.

Sources of Application Performance Problems

For various parts of a TopLink enabled application, this section describes the performance problems most commonly encountered and provides suggestions for improving performance. Areas of an application where performance problems can occur include:

- [General Performance Optimization](#)
- [Schema Optimization](#)
- [Mapping and Descriptor Optimization](#)
- [Session Optimization](#)
- [Cache Optimization](#)
- [Data Access Optimization](#)
- [Query Optimization](#)
- [Unit of Work Optimization](#)
- [Application Server and Database Optimization](#)

Measuring Performance With the TopLink Profiler

The most important challenge to performance tuning is knowing what to optimize. To improve the performance of your application, identify the areas of your application that do not operate at peak efficiency. The TopLink performance profiler helps you identify performance problems by logging performance statistics for every executed query in a given session.

Note: You should also consider using general performance profilers such as JDeveloper or JProbe to analyze performance problems. These tools can provide more detail that may be required to properly diagnose a problem.

The TopLink performance Profiler logs the following information to the TopLink log file (for general information about TopLink logging, see "[Logging](#)" on page 75-7):

- Query class
- Domain class
- Total time, total execution time of the query (in milliseconds)
- Local time, the amount of time spent on the user's workstation (in milliseconds)
- Number of objects, the total number of objects affected
- Number of objects handled per second
- Logging, the amount of time spent printing logging messages (in milliseconds)
- SQL prepare, the amount of time spent preparing the SQL script (in milliseconds)
- SQL execute, the amount of time spent executing the SQL script (in milliseconds)
- Row fetch, the amount of time spent fetching rows from the database (in milliseconds)
- Cache, the amount of time spent searching or updating the object cache (in milliseconds)

- Object build, the amount of time spent building the domain object (in milliseconds)
- Query prepare, the amount of time spent to prepare the query prior to execution (in milliseconds)
- SQL generation, the amount of time spent to generate the SQL script before it is sent to the database (in milliseconds)

This section includes information on the following topics:

- [Configuring the TopLink Performance Profiler](#)
- [Accessing the TopLink Profiler Results](#)

Configuring the TopLink Performance Profiler

To enable the TopLink performance profiler, select the **TopLink** profiler option when configuring your session (see "[Configuring a Performance Profiler](#)" on page 77-10).

The TopLink performance profiler is an instance of `oracle.toplink.tools.profiler.PerformanceProfiler` class. It provides the following public API:

- `logProfile`—enables the profiler
- `dontLogProfile`—disables the profiler
- `logProfileSummary`—organizes the profiler log into a summary of all the individual operation profiles including operation statistics like the shortest time of all the operations that were profiled, the total time of all the operations, the number of objects returned by profiled queries, and the total time that was spent in each kind of operation that was profiled.
- `logProfileSummaryByQuery`—organizes the profiler log as query summaries. This is the default profiler behavior.
- `logProfileSummaryByClass`—organizes the profiler log as class summaries. This is an alternative to the default behavior implemented by `logProfileSummaryByQuery` method.

Accessing the TopLink Profiler Results

The simplest way to view TopLink profiler results is to read the TopLink log files with a text editor. For general information about TopLink logging, such as logging file location, see "[Logging](#)" on page 75-7.

Alternatively, you can use the graphical performance profiler that the TopLink Web client provides. For more information, refer to the Web client online Help and README files.

[Example 11-1](#) shows an example of the TopLink profiler output.

Example 11-1 Performance Profiler Output

```
Begin Profile of{
ReadAllQuery(oracle.toplink.demos.employee.domain.Employee)
Profile(ReadAllQuery,# of obj=12, time=1399,sql execute=217, prepare=495, row
fetch=390, time/obj=116,obj/sec=8)
} End Profile
```

The second line of the profile contains the following information about a query:

- `ReadAllQuery(oracle.toplink.demos.employee.domain.Employee):` specific query profiled, and its arguments.
- `Profile(ReadAllQuery:` start of the profile and the type of query.
- `# of obj=12:` number of objects involved in the query.
- `time=1399:` total execution time of the query (in milliseconds).
- `sql execute=217:` total time spent preparing the SQL script.
- `prepare=495:` total time spent preparing the SQL script.
- `row fetch=390:` total time spent fetching rows from the database.
- `time/obj=116:` number of milliseconds spent on each object.
- `obj/sec=8) */:` number of objects handled per second.

Measuring Performance With the Oracle Dynamic Monitoring System (DMS)

Oracle DMS is a library that enables application and system developers to use a variety of DMS sensors to measure and export customized performance metrics for specific software components (called nouns).

TopLink includes DMS instrumentation in essential objects to provide efficient monitoring of run-time data in TopLink-enabled applications, including both J2EE and non-J2EE applications.

By enabling DMS profiling in a TopLink application (see "[Configuring the Oracle DMS Profiler](#)" on page 11-6), you can collect and easily access run-time data that can help you with application administration tasks and performance tuning.

Note: You should also consider using general performance profilers such as JDeveloper or JProbe to analyze performance problems. These tools can provide more detail that may be required to properly diagnose a problem.

[Table 11-1](#) lists the many performance and status metrics TopLink provides through DMS.

[Table 11-2](#) lists the various profiling levels you can use to adjust the level of profiling to the amount of monitoring information you require. Levels are listed in order of increasing system overhead.

You can easily access DMS data at run time using a management application that supports the Java Management Extensions (JMX) API (see "[Accessing Oracle DMS Profiler Data Using JMX](#)" on page 11-7) or using any Web browser and the DMS Spy servlet (see "[Accessing Oracle DMS Profiler Data Using the DMS Spy Servlet](#)" on page 11-7).

Table 11–1 TopLink DMS Metrics

DMS Noun Name ¹	Sensor Name	Level ²	Description
Cache	CacheHits	HEAVY	Number of times an object looked up in the cache was found.
	CacheMisses	HEAVY	Number of times an object looked up in the cache was not found.
	Caching	ALL	Time spent adding, looking up, and removing objects in the cache.
RCM ³	ChangesNotProcessed	ALL	Number of coordinated cache <code>ObjectChangeSets</code> discarded because the object was not found in the cache and was not merged.
	ChangesProcessed	ALL	Number of coordinated cache <code>ObjectChangeSets</code> for which an object was found in the cache and merged.
	MessagesReceived	HEAVY	Number of cache coordination messages received.
	MessagesSent	HEAVY	Number of cache coordination messages sent.
	RemoteChangeSets	HEAVY	Number of change sets received from remote machines and processed.
	RCMStatus	HEAVY	Cache coordination status: one of <code>not configured</code> , <code>started</code> , or <code>stopped</code> .
Connection	ConnectCalls	HEAVY	Total number of connect calls made.
	ConnectionsInUse(<i>POOL_NAME</i>)	HEAVY	Number of connections in use for the given connection pool.
	DisconnectCalls	HEAVY	Total number of disconnect calls made.
	ConnectionManagement	ALL	Time spent managing connections including connecting, reconnecting, and disconnecting from a data source.
Query	DatabaseExecute	ALL	Time spent in calls to the JDBC <code>Statement</code> . Includes time spent in calls to: <code>close</code> , <code>executeUpdate</code> , and <code>executeQuery</code> .
	DeleteQueries	HEAVY	Time spent executing delete queries, including time spent in JDBC calls.
	ObjectBuilding	ALL	Time spent building persistent objects from database rows.
	QueryPreparation	ALL	Time spent preparing a query. Does not include time spent doing SQL prepare.
	ReadQueries	HEAVY	Time spent executing read queries, including time spent in JDBC calls.
	RowFetch	ALL	Time spent fetching the JDBC result set from the database and building <code>DatabaseRecord</code> objects from the JDBC result set. Includes regular SQL calls and stored procedure calls.
	SqlGeneration	ALL	Time spent generating SQL. In the case of TopLink expressions, includes time spent converting <code>Expression</code> to SQL.
	SqlPrepare	ALL	Time spent in JDBC preparing the <code>Statement</code> . Includes the time spent in EIS creating an <code>Interaction</code> associated with a connection, and creating input and output <code>Record</code> objects.
	UpdateQueries	HEAVY	Time spent executing update queries, including time spent in JDBC calls.

Table 11–1 (Cont.) TopLink DMS Metrics

DMS Noun Name ¹	Sensor Name	Level ²	Description
	WriteQueries	HEAVY	Time spent executing write queries, including time spent in JDBC calls.
Session	ClientSession	HEAVY	Number of ClientSessions currently logged in.
	loginTime	NORMAL	Time at which the session was logged in. Once the session is logged out, the sensor no longer appears.
	SessionName	NORMAL	Name of the session.
	UnitOfWork	HEAVY	Number of UnitOfWork objects acquired from this session.
Transaction	DistributedMerge	ALL	Time spent merging remote transaction changes into the local shared cache.
	JtsAfterCompletion	ALL	Time spent on JTS afterCompletion method.
	JtsBeforeCompletion	ALL	Time spent on JTS beforeCompletion method.
	MergeTime	ALL	Time spent merging changes into the shared cache.
	OptimisticLocks	HEAVY	Number of optimistic lock exceptions thrown.
	Sequencing	ALL	Time spent maintaining the sequence number mechanism and setting the sequence number on objects.
	UnitOfWorkRegister	ALL	Time spent in registering objects with the UnitOfWork.
	UnitOfWorkCommits	ALL	Time spent in the UnitOfWork commit process.
	UnitOfWorkRollBacks	HEAVY	Number of UnitOfWork commits that were rolled back.
Miscellaneous	DescriptorEvents	ALL	Time spent by the DescriptorEventManager executing a descriptor event.
	Logging	ALL	Time spent logging TopLink activities.
	SessionEvents	ALL	Time spent by the SessionEvent manager executing a session event.

¹ DMS noun names are followed by the name of the session to which they belong. For example, `Cache(SESSION_NAME)`.

² See [Table 11–2](#) for a description of each level setting.

³ Cache Coordination

Table 11–2 DMS Metric Collection Levels

Level	Description
NONE	Disable collection of all DMS metrics.
NORMAL	Enable collection of TopLink DMS metrics. Adds very low overhead. This is the default setting.
HEAVY	Enable collection of basic TopLink DMS metrics. Adds about 1 percent overhead.
ALL	Enable all possible TopLink DMS metrics. Adds about 3 percent overhead.

Configuring the Oracle DMS Profiler

You configure DMS support in your TopLink application differently depending on the type of application it is:

- [OC4J Applications](#)

■ Non-OC4J Applications

OC4J Applications

By default, DMS metric collection is enabled for TopLink CMP applications deployed to OC4J. For BMP or non-CMP applications deployed to OC4J, you must configure DMS metric collection (see "[Configuring a Performance Profiler](#)" on page 77-10).

TopLink EJB deployed in OC4J are subject to the DMS configuration specified by the OC4J command line-property `-Doracle.dms.sensors=<level>` where `<level>` is one of the values listed in [Table 11-2](#).

Non-OC4J Applications

To enable DMS metric collection for TopLink applications deployed to an application sever other than OC4J do the following:

1. Ensure that the `dms.jar` file is in your application classpath.
By default, the `dms.jar` file is located in `<ORACLE_HOME>\lib` directory.
2. Set system property `oracle.dms.sensors=<level>` where `<level>` is one of the values listed in [Table 11-2](#).
3. To enable the DMS profiler, select the **DMS** profiler option when configuring your TopLink session (see "[Configuring a Performance Profiler](#)" on page 77-10).

Accessing Oracle DMS Profiler Data Using JMX

Using the Java Management Extensions (JMX) API, you can publish DMS profiler run-time data from a managed application (TopLink) to a JMX-compliant management application, by way of EJB-like MBean components.

When you configure your TopLink application to enable run-time services (see "[Configuring a Performance Profiler](#)" on page 77-10) and you deploy your application to OC4J, the TopLink runtime will deploy a JMX MBean so that a JMX management application can access the DMS profiler run-time data your application publishes.

For code examples that illustrates how to use DMS and JMX, see http://www.oracle.com/technology/tech/java/oc4j/1003/how_to/jmx-enabled-demo.html.

Accessing Oracle DMS Profiler Data Using the DMS Spy Servlet

Once your DMS enabled TopLink application is running, you can access the DMS data it is collecting.

The DMS Spy servlet is available in all Java processes that use DMS. It lets you monitor metrics for a single Java process from a Web browser.

To access DMS data directly using the DMS Spy servlet, do the following:

1. Ensure that the `dms.jar` file is in your application classpath.
By default, the `dms.jar` file is located in `<ORACLE_HOME>\lib`.
2. Set the following system properties for the DMS enabled Java process you want to monitor:
`oracle.dms.publisher.classes=oracle.dms.http.Httpd`
`oracle.dms.httpd.port.start=<port>`

where `<port>` is the HTTP port on which DMS accepts requests (the default value is 46080).

3. Apply the system property changes by restarting the Java process you want to monitor.
4. Using a Web browser, connect to the Java process and access the Spy servlet by entering the following URL:

`http://<host>:<port>/dms0/Spy`

where `<host>` is the host name of your Java process and `<port>` is the value specified by the `oracle.dms.httpd.port.start` system property.

The Spy servlet displays all TopLink DMS-enabled objects appropriate for the current DMS level setting. Figure 11–1 shows an example of the DMS Spy servlet display.

Figure 11–1 DMS Spy Servlet Display

Metric Tables

All Metric Tables | Text

JVM

TopLink Cache

TopLink Connections

TopLink Miscellaneous

TopLink Session

TopLink_Connections

TopLink_Connections

Name	Parent	Host	Process
Connection / (Test)	/ TopLink	xchen-pc	TopLink 46080

General Performance Optimization

Do not override TopLink default behavior unless your application absolutely requires it. Because TopLink default behavior is set for optimum results with the most common applications, the default is usually the most efficient choice for any given option. This is especially important for query or cache behavior.

Use TopLink Workbench rather than manual coding. TopLink Workbench is not only easy to use. The default configuration it exports to deployment XML (and the code it generates, if required) represents best practices optimized for most applications.

Schema Optimization

Optimization is an important consideration when you design your database schema and object model. Most performance issues occur when the object model or database schema is too complex, which can make the database slow and difficult to query. This is most likely to happen if you derive your database schema directly from a complex object model.

To optimize performance, design the object model and database schema together. However, allow each model to be designed optimally: do not require a direct one-to-one correlation between the two.

This section includes the following schema optimization examples:

- [Schema Case 1: Aggregation of Two Tables into One](#)

- [Schema Case 2: Splitting One Table Into Many](#)
- [Schema Case 3: Collapsed Hierarchy](#)
- [Schema Case 4: Choosing One out of Many](#)

Schema Case 1: Aggregation of Two Tables into One

A common schema optimization technique is to aggregate two tables into a single table. This improves read and write performance by requiring only one database operation instead of two.

[Table 11–3](#) and [Table 11–4](#) illustrate the table aggregation technique.

Table 11–3 *Original Schema (Aggregation of Two Tables Case)*

Elements	Details
Title	ACME Member Location Tracking System
Classes	Member, Address
Tables	MEMBER, ADDRESS
Relationships	Source, Instance Variable, Mapping, Target, Member, address, one-to-one, Address

The nature of this application dictates that developers always look up employees and addresses together. As a result, querying a member based on address information requires a database join, and reading a member and its address requires two read statements. Writing a member requires two write statements. This adds unnecessary complexity to the system, and results in poor performance.

A better solution is to combine the MEMBER and ADDRESS tables into a single table, and change the one-to-one relationship to an aggregate relationship. This lets you read all information with a single operation, and doubles the update and insert speed, because only a single row in one table requires modifications.

Table 11–4 *Optimized Schema (Aggregation of Two Tables Case)*

Elements	Details
Classes	Member, Address
Tables	MEMBER
Relationships	Source, Instance Variable, Mapping, Target, Member, address, aggregate, Address

Schema Case 2: Splitting One Table Into Many

To improve overall performance of the system, split large tables into two or more smaller tables. This significantly reduces the amount of data traffic required to query the database.

For example, the system illustrated in [Table 11–5](#) assigns employees to projects within an organization. The most common operation reads a set of employees and projects, assigns employees to projects, and update the employees. The employee's address or job classification is also occasionally used to determine the project on which the employee is placed.

Table 11–5 Original Schema (Splitting One Table into Many Case)

Elements	Details	Instance Variable	Mapping	Target
Title	ACME Employee Workflow System			
Classes	Employee, Address, PhoneNumber, EmailAddress, JobClassification, Project			
Tables	EMPLOYEE, PROJECT, PROJ_EMP			
Relationships	Employee	address	aggregate	Address
	Employee	phoneNumber	aggregate	EmailAddress
	Employee	emailAddress	aggregate	EmailAddress
	Employee	job	aggregate	JobClassification
	Employee	projects	many-to-many	Project

When you read a large volume of employee records from the database, you must also read their aggregate parts. Because of this, the system suffers from general read performance issues. To resolve this, break the EMPLOYEE table into the EMPLOYEE, ADDRESS, PHONE, EMAIL, and JOB tables, as illustrated in [Table 11–6](#).

Because you usually read only the employee information, splitting the table reduces the amount of data transferred from the database to the client. This improves your read performance by reducing the amount of data traffic by 25 percent.

Table 11–6 Optimized Schema (Splitting One Table into Many Case)

Elements	Details	Instance Variable	Mapping	Target
Title	ACME Employee Workflow System			
Classes	Employee, Address, PhoneNumber, EmailAddress, JobClassification, Project			
Tables	EMPLOYEE, ADDRESS, PHONE, EMAIL, JOB, PROJECT, PROJ_EMP			

Table 11–6 (Cont.) Optimized Schema (Splitting One Table into Many Case)

Elements	Details	Instance Variable	Mapping	Target
Relationships	Employee	address	one-to-one	Address
	Employee	phoneNumber	one-to-one	EmailAddress
	Employee	emailAddress	one-to-one	EmailAddress
	Employee	job	one-to-one	JobClassification
	Employee	projects	many-to-many	Project

Schema Case 3: Collapsed Hierarchy

A common mistake when you transform an object-oriented design into a relational model, is to build a large hierarchy of tables on the database. This makes querying difficult, because queries against this type of design can require a large number of joins. It is usually a good idea to collapse some of the levels in your inheritance hierarchy into a single table.

[Table 11–7](#) represents a system that assigns clients to a company’s sales representatives. The managers also track the sales representatives that report to them.

Table 11–7 Original Schema (Collapsed Hierarchy Case)

Elements	Details
Title	ACME Sales Force System
Classes	Tables
Person	PERSON
Employee	PERSON, EMPLOYEE
SalesRep	PERSON, EMPLOYEE, REP
Staff	PERSON, EMPLOYEE, STAFF
Client	PERSON, CLIENT
Contact	PERSON, CONTACT

The system suffers from complexity issues that hinder system development and performance. Nearly all queries against the database require large, resource-intensive joins. If you collapse the three-level table hierarchy into a single table, as illustrated in [Table 11–8](#), you substantially reduce system complexity. You eliminate joins from the system, and simplify queries.

Table 11–8 Optimized Schema (Collapsed Hierarchy Case)

Elements	Details
Classes	Tables
Person	none
Employee	EMPLOYEE
SalesRep	EMPLOYEE
Staff	EMPLOYEE
Client	CLIENT
Contact	CLIENT

Schema Case 4: Choosing One out of Many

In a one-to-many relationship, a single source object has a collection of other objects. In some cases, the source object frequently requires one particular object in the collection, but requires the other objects only infrequently. You can reduce the size of the returned result set in this type of case by adding an instance variable for the frequently required object. This lets you access the object without instantiating the other objects in the collection.

Table 11–9 represents a system by which an international shipping company tracks the location of packages in transit. When a package moves from one location to another, the system creates a new a location entry for the package in the database. The most common query against any given package is for its current location.

Table 11–9 Original Schema (Choosing One out of Many Case)

Elements	Details	Instance Variable	Mapping	Target
Title	ACME Shipping Package Location Tracking system			
Classes	Package, Location			
Tables	PACKAGE, LOCATION			
Relationships	Package	locations	one-to-many	Location

A package in this system can accumulate several location values in its LOCATION collection as it travels to its destination. Reading all locations from the database is resource intensive, especially when the only location of interest is the current location.

To resolve this type of problem, add a specific instance variable that represents the current location. You then add a one-to-one mapping for the instance variable, and use the instance variable to query for the current location. As illustrated in Table 11–9, because you can now query for the current location without reading all locations associated with the package, this dramatically improves the performance of the system.

Table 11–10 Optimized Schema (Choosing One out of Many Case)

Elements	Details	Instance Variable	Mapping	Target
Classes	Package, Location			
Tables	PACKAGE, LOCATION			
Relationships	Package	locations	one-to-many	Location
	Package	currentLocation	one-to-many	Location

Mapping and Descriptor Optimization

Always use indirection. It is not only critical in optimizing database access, but also allows TopLink to make several other optimizations including optimizing its cache access and unit of work processing. See ["Configuring Indirection"](#) on page 35-3.

Avoid using the existence checking option `checkCacheThenDatabase` on descriptors (see ["Configuring Cache Existence Checking at the Descriptor Level"](#) on page 28-42), unless required by the application. The default existence checking behavior offers better performance.

Avoid expensive initialization in the default constructor that TopLink uses to instantiate objects. Instead, use lazy initialization or use a TopLink instantiation policy (see ["Configuring Instantiation Policy"](#) on page 28-68) to configure the descriptor to use a different constructor.

Avoid using method access in your TopLink mappings (see ["Configuring Method Accessing"](#) on page 35-14), especially if you have expensive or potentially dangerous side-effect code in your get or set methods; use the default direct attribute access instead.

Session Optimization

Use a Server session in a server environment, not a `DatabaseSession`.

Use the TopLink client session instead of remote session. A client session is appropriate for most multiuser J2EE application server environments.

Do not pool client sessions. Pooling sessions offers no performance gains.

For more information, see ["Server and Client Sessions"](#) on page 75-13 and ["OC4J Applications"](#) on page 11-7.

Cache Optimization

Cache coordination (see ["Understanding Cache Coordination"](#) on page 90-10) is one way to allow multiple, possibly distributed, instances of a session to broadcast object changes among each other so that each session's cache can be kept up-to-date.

However, cache coordination is best suited to applications with specific characteristics (["When to use Cache Coordination"](#) on page 90-10). Before implementing cache coordination, tune the TopLink cache for each class using alternatives such as object identity type (see ["Configuring Cache Type and Size at the Descriptor Level"](#) on page 28-35), cache invalidation (see ["Cache Invalidation"](#) on page 90-8), or cache isolation (see ["Cache Isolation"](#) on page 90-9). Doing so lets you configure the optimal cache configuration for each type of class (see [Table 11-11](#)) and may eliminate the need for distributed cache coordination altogether.

Table 11-11 Identity Map and Cache Configuration by Class Type

Class Type	Identity Map Options	Cache Options
read-only	soft, hard, or full ¹	
read-mostly	soft or hard	cache invalidation or cache coordination
write-mostly	weak	cache invalidation

¹ If the number of instances is finite.

If you do use cache coordination, use JMS for cache coordination rather than RMI. JMS is more robust, easier to configure, and runs asynchronously. If you require synchronous cache coordination, use RMI.

You can configure a descriptor to control when the TopLink runtime will refresh the session cache when an instance of this object type is queried (see ["Configuring Cache Refreshing"](#) on page 28-27). Oracle does not recommend the use of **Always Refresh** or **Disable Cache Hits**.

Using **Always Refresh** may result in refreshing the cache on queries when not required or desired. As an alternative, consider configuring cache refresh on a query by query basis (see ["Refreshing the Cache"](#) on page 96-35).

Using **Disable Cache Hits** instructs TopLink to bypass the cache for object read queries based on primary key. This results in a database round trip every time an object read query based on primary key is executed on this object type, negating the performance advantage of the cache. When used in conjunction with **Always Refresh**, this option ensures that all queries go to the database. This can have a significant impact on performance. These options should only be used in specialized circumstances.

Data Access Optimization

Depending on the type of data source your application accesses, TopLink offers a variety of `LogIn` options that you can use to tune the performance of low level data reads and writes.

You can use several techniques to improve data access performance for your application. This section discusses some of the more common approaches, including:

- [JDBC Driver Properties Optimization](#)
- [Data Format Optimization](#)
- [Batch Writing](#)
- [Parameterized SQL \(Binding\) and Prepared Statement Caching](#)

JDBC Driver Properties Optimization

Consider the default behavior of the JDBC driver you choose for your application. Some JDBC driver options can affect data access performance.

Some important JDBC driver properties can be configured directly using TopLink Workbench or TopLink API (for example, see ["JDBC Fetch Size"](#) on page 11-17).

JDBC driver properties that are not supported directly by TopLink Workbench or TopLink API can still be configured as generic JDBC properties that TopLink passes to the JDBC driver.

For example, some JDBC drivers, such as Sybase JConnect, perform a database round trip to test whether or not a connection is closed: that is, calling the JDBC driver method `isClosed` results in a stored procedure call or SQL select. This database round-trip can cause a significant performance reduction. To avoid this, you can disable this behavior: for Sybase JConnect, you can set property name `CLOSED_TEST` to value `INTERNAL`.

For more information about configuring general JDBC driver properties from within your TopLink application, see ["Configuring Properties"](#) on page 85-4.

Data Format Optimization

By default, TopLink optimizes data access by accessing the data from JDBC in the format the application requires. For example, TopLink retrieves `long` data types from JDBC instead of having the driver return a `BigDecimal` that TopLink would then have to convert into a `long`.

Some older JDBC drivers do not perform data conversion correctly and conflict with this optimization. In this case, you can disable this optimization (see ["Configuring Advanced Options"](#) on page 86-11).

Batch Writing

Batch writing can improve database performance by sending groups of INSERT, UPDATE, and DELETE statements to the database in a single transaction, rather than individually.

When used without parameterized SQL, this is known as dynamic batch writing.

When used with parameterized SQL (see ["Parameterized SQL \(Binding\) and Prepared Statement Caching"](#) on page 11-15), this is known as parameterized batch writing. This allows a repeatedly executed statement, such as a group of inserts of the same type, to be executed as a single statement and a set of bind parameters. This can provide a large performance benefit as the database does not have to parse the batch.

When using batch writing, you can tune the maximum batch writing size using `setMaxBatchWritingSize` method of the `Login` interface. The meaning of this value depends on whether or not you are using parameterized SQL:

- If you are using parameterized SQL (you configure your `Login` by calling `Login` method `bindAllParameters`), the maximum batch writing size is the number of statements to batch (default: 100).
- If you are using dynamic SQL, the maximum batch writing size is the size of the SQL string buffer in characters (default: 32000).

By default, TopLink does not enable batch writing because not all databases and JDBC drivers support it. Oracle recommends that you enable batch writing for selected databases and JDBC drivers that support this option (see ["Configuring JDBC Options"](#) on page 86-9).

For a more detailed example of using batch writing to optimize write queries, see ["Batch Writing and Parameterized SQL"](#) on page 11-28.

Parameterized SQL (Binding) and Prepared Statement Caching

Using parameterized SQL, you can keep the overall length of an SQL query from exceeding the statement length limit that your JDBC driver or database server imposes.

Using parameterized SQL and prepared statement caching, you can improve performance by reducing the number of times the database SQL engine parses and prepares SQL for a frequently called query.

By default, TopLink does not enable parameterized SQL and prepared statement caching, because not all databases and JDBC drivers support it. Oracle recommends that you enable parameterized SQL and prepared statement caching for selected databases and JDBC drivers that support these options.

Not all JDBC drivers support all JDBC binding options (see ["Configuring JDBC Options"](#) on page 86-9). Selecting a combination of options may result in different

behavior from one driver to another. Before selecting JDBC options, consult your JDBC driver documentation. When choosing binding options, consider the following approach:

1. Try binding all parameters with all other binding options disabled.
2. If this fails to bind some large parameters, consider enabling one of the following options, depending on the parameter's data type and the binding options that your JDBC driver supports:
 - a. To bind large `String` parameters, try enabling string binding.
If large `String` parameters still fail to bind, consider adjusting the maximum `String` size. `TopLink` sets the maximum `String` size to 32000 characters by default.
 - b. To bind large `Byte` array parameters, try enabling byte array binding.
3. If this fails to bind some large parameters, try enabling streams for binding.
Typically, configuring string or byte array binding will invoke streams for binding. If not, explicitly configuring streams for binding may help.

For J2EE applications that use `TopLink` external connection pools, you must configure parameterized SQL in `TopLink` but you cannot configure prepared statement caching in `TopLink`. In this case, you must configure prepared statement caching in the application server connection pool. For example, in `OC4J`, if you configure your `data-source.xml` file with an emulated `data-source` (where `connection-driver` is `oracle.jdbc.OracleDriver` and `class` is `oracle.j2ee.sql.DriverManagerDataSource`), you can configure a non-zero `stmt-cache-size` that enables JDBC statement caching and defines the maximum number of statements cached.

For applications that use `TopLink` internal connection pools, you can configure parameterized SQL and prepared statement caching.

You can configure parameterized SQL and prepared statement caching at the:

- project level—applies to all named queries (see ["Configuring Named Query Parameterized SQL and Statement Caching at the Project Level"](#) on page 23-7)
- descriptor level—applies on a per-named-query basis (see ["Configuring Named Query Options"](#) on page 28-22)
- database login level—applies to all queries (see ["Configuring JDBC Options"](#) on page 86-9) and provides additional parameter binding API to alleviate the limit imposed by some drivers on SQL statement size.
- query level—applies on a per-query basis (see ["Using Parameterized SQL and Statement Caching in a DatabaseQuery"](#) on page 98-17).

Query Optimization

`TopLink` provides an extensive query API for reading, writing, and updating data. This section describes ways of optimizing query performance in various circumstances.

Before optimizing queries, consider the optimization suggestions in ["Data Access Optimization"](#) on page 11-14.

This section includes information on the following:

- [Parameterized SQL and Prepared Statement Caching](#)

- [Named Queries](#)
- [Batch and Join Reading](#)
- [Partial Object Queries and Fetch Groups](#)
- [JDBC Fetch Size](#)
- [Cursored Streams and Scrollable Cursors](#)
- [Read Optimization Examples](#)
- [Write Optimization Examples](#)

Parameterized SQL and Prepared Statement Caching

These features lets you cache and reuse a query's pre-parsed database statement when the query is re-executed.

For more information, see "[Parameterized SQL \(Binding\) and Prepared Statement Caching](#)" on page 11-15.

Named Queries

Whenever possible, use named queries in your application. Named queries help you avoid duplication, are easy to maintain and reuse, and easily add complex query behavior to the application. Using named queries also allows for the query to be prepared once, and for the SQL generation to be cached.

For more information, see "[Named Queries](#)" on page 96-15.

Batch and Join Reading

To optimize database read operations, TopLink supports both batch and join reading. When you use these techniques, you dramatically decrease the number of times you access the database during a read operation, especially when your result set contains a large number of objects.

For more information, see the following:

- ["Using Batch Reading"](#) on page 98-10
- ["Join Reading and Object-Level Read Queries"](#) on page 96-12
- ["Batch Writing"](#) on page 11-15

Partial Object Queries and Fetch Groups

Partial object queries lets you retrieve partially populated objects from the database rather than complete objects.

For CMP applications, you can use fetch groups to accomplish the same performance optimization.

For more information about partial object reading, see "[Partial Object Queries](#)" on page 96-11.

For more information about fetch groups, see "[Fetch Groups](#)" on page 26-4.

JDBC Fetch Size

The JDBC fetch size gives the JDBC driver a hint as to the number of rows that should be fetched from the database when more rows are needed.

For large queries that return a large number of objects you can configure the row fetch size used in the query to improve performance by reducing the number database hits required to satisfy the selection criteria.

Most JDBC drivers default to a fetch size of 10, so if you are reading 1000 objects, increasing the fetch size to 256 can significantly reduce the time required to fetch the query's results. The optimal fetch size is not always obvious. Usually, a fetch size of one half or one quarter of the total expected result size is optimal. Note that if you are unsure of the result set size, incorrectly setting a fetch size too large or too small can decrease performance.

Set the query fetch size with `ReadQuery` method `setFetchSize` as [Example 11-2](#) shows. Alternatively, you can use `ReadQuery` method `setMaxRows` to set the limit for the maximum number of rows that any `ResultSet` can contain.

Example 11-2 JDBC Driver Fetch Size

```
ReadAllQuery query = new ReadAllQuery();
query.setReferenceClass(Employee.class);
query.setSelectionCriteria(new ExpressionBuilder.get("id").greaterThan(100));

// Set the JDBC fetch size
query.setFetchSize(50);

// Configure the query to return results as a ScrollableCursor
query.useScrollableCursor();

// Execute the query
ScrollableCursor cursor = (ScrollableCursor) session.executeQuery(query);

// Iterate over the results
while (cursor.hasNext()) {
    System.out.println(cursor.next().toString());
}
cursor.close();
```

In this example, when you execute the query, the JDBC driver retrieves the first 50 rows from the database (or all rows if less than 50 rows satisfy the selection criteria). As you iterate over the first 50 rows, each time you call `cursor.next()`, the JDBC driver returns a row from local memory—it does not need to retrieve the row from the database. When you try to access the fifty first row (assuming there are more than 50 rows that satisfy the selection criteria), the JDBC driver again goes to the database and retrieves another 50 rows. In this way, 100 rows are returned with only two database hits.

If you specify a value of zero, then the hint is ignored: the JDBC driver returns one row at a time. The default value is zero.

Cursored Streams and Scrollable Cursors

You can configure a query to retrieve data from the database using a cursored Java stream or scrollable cursor. This lets you view a result set in manageable increments rather than as a complete collection. This is useful when you have a large result set. You can further tune performance by configuring the JDBC driver fetch size used (see ["JDBC Fetch Size"](#) on page 11-17).

For more information about scrollable cursors, see ["Handling Cursor and Stream Query Results"](#) on page 99-16.

Read Optimization Examples

TopLink provides the read optimization features listed in [Table 11–12](#).

This section includes the following read optimization examples:

- [Reading Case 1: Displaying Names in a List](#)
- [Reading Case 2: Batch Reading Objects](#)
- [Reading Case 3: Using Complex Custom SQL Queries](#)
- [Reading Case 4: Using View Objects](#)
- [Reading Case 5: Inheritance Views](#)

Table 11–12 Read Optimization Features

Feature	Function	Performance Technique
Unit of Work	Tracks object changes within the Unit of Work.	To minimize the amount of tracking required, registers only those objects that will change. For more information, see "Understanding TopLink Transactions" on page 100-1.
Indirection	Uses indirection objects to defer the loading and processing of relationships.	Provides a major performance benefit. It allows database access to be optimized and allows TopLink to internally make several optimizations in caching and unit of work.
Soft cache, weak identity map	Offers client-side caching for objects read from database, and drops objects from the cache when memory becomes low.	Reduces database calls and improves memory performance. For more information, see "Cache Type and Object Identity" on page 90-3.
Weak identity map	Offers client-side caching for objects.	Reduces database access and maintains a cache of all referenced objects. For more information, see "Cache Type and Object Identity" on page 90-3.
Batch reading and joining	Reduces database access by batching many queries into a single query that reads more data.	Dramatically reduces the number of database accesses required to perform a read query. For more information, see "Using Batch Reading" on page 98-10 and "Using Join Reading" on page 98-11.
Partial object reading and fetch groups.	Allows reading of a subset of a result set of the object's attributes.	Reduces the amount of data read from the database. For more information, see "Partial Object Queries" on page 96-11. For more information about fetch groups, see "Fetch Groups" on page 26-4

Table 11–12 (Cont.) Read Optimization Features

Feature	Function	Performance Technique
Report query	Similar to partial object reading, but returns only the data instead of the objects.	Supports complex reporting functions such as aggregation and group-by functions. Also lets you compute complex results on the database, instead of reading the objects into the application and computing the results locally. For more information, see "Report Query" on page 96-15.
JDBC fetch size and ReadQuery maximum rows	Reduces the number of database hits required to return all the rows that satisfy selection criteria.	For more information, see "JDBC Fetch Size" on page 11-17.
Cursors	Lets you view a large result set in manageable increments rather than as a complete collection	For more information, see "Cursored Streams and Scrollable Cursors" on page 11-18
Inheritance views	Allows a view to be used for queries against an inheritance superclass that can read all of its subclasses in a single query, instead of multiple queries	For more information, see "Reading Case 5: Inheritance Views" on page 11-26.

Reading Case 1: Displaying Names in a List

An application may ask the user to choose an element from a list. Because the list displays only a subset of the information contained in the objects, it is not necessary to query for all information for objects from the database.

TopLink features that optimize these types of operations include:

- [Partial Object Reading](#)
- [Report Query](#)
- [Fetch Groups](#) (CMP projects only)

These features let you query only the information required to display the list. The user can then select an object from the list.

Partial Object Reading Partial object reading is a query designed to extract only the required information from a selected record in a database, rather than all the information the record contains. Because partial object reading does not fully populate objects, you can neither cache nor edit partially read objects.

For more information about partial object queries, see ["Partial Object Queries"](#) on page 96-11.

In [Example 11–3](#), the query builds complete employee objects, even though the list displays only employee last names. With no optimization, the query reads all the employee data.

Example 11–3 No Optimization

```
/* Read all the employees from the database, ask the user to choose one and return
it. This must read in all the information for all the employees */
List list;
```

```
// Fetch data from database and add to list box
Vector employees = (Vector) session.readAllObjects(Employee.class);
list.addAll(employees);

// Display list box
....

// Get selected employee from list
Employee selectedEmployee = (Employee) list.getSelectedItemAt();

return selectedEmployee;
```

[Example 11-4](#) demonstrates the use of partial object reading. It reads only the last name and primary key for the employee data. This reduces the amount of data read from the database.

Example 11-4 Optimization Through Partial Object Reading

```
/* Read all the employees from the database, ask the user to choose one and return
it. This uses partial object reading to read just the last names of the employees.
Since TopLink automatically includes the primary key of the object, the full
object can easily be read for editing */
List list;

// Fetch data from database and add to list box
ReadAllQuery query = new ReadAllQuery(Employee.class);
query.addPartialAttribute("lastName");

// The next line avoids a query exception
query.dontMaintainCache();
Vector employees = (Vector) session.executeQuery(query);
list.addAll(employees);

// Display list box
....

// Get selected employee from list
Employee selectedEmployee = (Employee)session.readObject(list.getSelectedItemAt());
return selectedEmployee;
```

Report Query Report query lets you retrieve data from a set of objects and their related objects. Report query supports database reporting functions and features.

For more information, see ["Report Query Results"](#) on page 96-8.

[Example 11-5](#) demonstrates the use of report query to read only the last name of the employees. This reduces the amount of data read from the database compared to the code in [Example 11-3](#), and avoids instantiating employee instances.

Example 11-5 Optimization Through Report Query

```
/* Read all the employees from the database, ask the user to choose one and return
it. The report query is used to read just the last name of the employees. Then the
primary key stored in the report query result to read the real object */
List list;

// Fetch data from database and add to list box
ExpressionBuilder builder = new ExpressionBuilder();
ReportQuery query = new ReportQuery (Employee.class, builder);
```

```

query.addAttribute("lastName");
query.retrievePrimaryKeys();
Vector reportRows = (Vector) session.executeQuery(query);
list.addAll(reportRows);

// Display list box
....

// Get selected employee from list
ReportQueryResult result = (ReportQueryResult) list.getSelectedItemAt();
Employee selectedEmployee =
    (Employee)result.readObject(Employee.Class,session);

```

Although the differences between the unoptimized example ([Example 11-3](#)) and the report query optimization in [Example 11-5](#) appear to be minor, report queries offer a substantial performance improvement.

Fetch Groups Fetch groups, applicable only to CMP projects, are similar to partial object reading, but does allow caching of the objects read. For objects with many attributes or reference attributes to complex graphs (or both), you can define a fetch group that determines what attributes are returned when an object is read. Because TopLink will automatically execute additional queries when the `get` method is called for attributes not in the fetch group, ensure that the unfetched data is not required: refetching data can become a performance issue.

For more information about querying with fetch groups, see ["Using Queries With Fetch Groups"](#) on page 99-2.

[Example 11-6](#) demonstrates the use of a static fetch group.

Example 11-6 Configuring a Query with a FetchGroup Using the FetchGroupManager

```

// Create static fetch group at the descriptor level
FetchGroup group = new FetchGroup("nameOnly");
group.addAttribute("firstName");
group.addAttribute("lastName");
descriptor.getFetchGroupManager().addFetchGroup(group);

// Use static fetch group at query level
ReadAllQuery query = new ReadAllQuery(Employee.class);
query.setFetchGroupName("nameOnly");

// Only Employee attributes firstName and lastName are fetched.
// If you call the Employee get method for any other attribute, TopLink executes
// another query to retrieve all unfetched attribute values. Thereafter, calling that get
// method will return the value directly from the object

```

Reading Case 2: Batch Reading Objects

The way your application reads data from the database affects performance. For example, reading a collection of rows from the database is significantly faster than reading each row individually.

A common performance challenge is to read a collection of objects that have a one-to-one reference to another object. This typically requires one read operation to read in the source rows, and one call for each target row in the one-to-one relationship.

To reduce the number of read operations required, use join and batch reading.

[Example 11-7](#) illustrates the unoptimized code required to retrieve a collection of

objects with a one-to-one reference to another object. [Example 11–8](#) and [Example 11–9](#) illustrate the use of joins and batch reading to improve efficiency.

Example 11–7 No Optimization

```
/* Read all the employees, and collect their address' cities. This takes N + 1
   queries if not optimized */

// Read all the employees from the database. This requires 1 SQL call
Vector employees = session.readAllObjects(Employee.class,
    new ExpressionBuilder().get("lastName").equal("Smith"));

//SQL: Select * from Employee where l_name = 'Smith'

// Iterate over employees and get their addresses.
// This requires N SQL calls
Enumeration enum = employees.elements();
Vector cities = new Vector();
while(enum.hasMoreElements())
    Employee employee = (Employee) enum.nextElement();
    cities.addElement(employee.getAddress().getCity());

//SQL: Select * from Address where address_id = 123, etc
}
```

Example 11–8 Optimization Through Joining

```
/* Read all the employees, and collect their address' cities. Although the code
   is almost identical because joining optimization is used it takes only 1
   query */

// Read all the employees from the database, using joining.
// This requires 1 SQL call
ReadAllQuery query = new ReadAllQuery();
query.setReferenceClass(Employee.class);
query.setSelectionCriteria(new
    ExpressionBuilder().get("lastName").equal("Smith"));
query.addJoinedAttribute("address");
Vector employees = session.executeQuery(query);

/* SQL: Select E.*, A.* from Employee E, Address A where E.l_name = 'Smith' and
   E.address_id = A.address_id Iterate over employees and get their addresses.
   The previous SQL already read all the addresses, so no SQL is required */
Enumeration enum = employees.elements();
Vector cities = new Vector();
while (enum.hasMoreElements()) {
    Employee employee = (Employee) enum.nextElement();
    cities.addElement(employee.getAddress().getCity());
}
```

Example 11–9 Optimization Through Batch Reading

```
/* Read all the employees, and collect their address' cities. Although the code
   is almost identical because batch reading optimization is used it takes only
   2 queries */

// Read all the employees from the database, using batch reading.
// This requires 1 SQL call, note that only the employees are read
ReadAllQuery query = new ReadAllQuery();
```

```

query.setReferenceClass(Employee.class);
query.setSelectionCriteria(new
    ExpressionBuilder().get("lastName").equal("Smith"));
query.addBatchReadAttribute("address");
Vector employees = (Vector)session.executeQuery(query);

// SQL: Select * from Employee where l_name = 'Smith'

// Iterate over employees and get their addresses.
// The first address accessed will cause all the addresses
// to be read in a single SQL call
Enumeration enum = employees.elements();
Vector cities = new Vector();
while (enum.hasMoreElements()) {
    Employee employee = (Employee) enum.nextElement();
    cities.addElement(employee.getAddress().getCity());
    // SQL: Select distinct A.* from Employee E, Address A
    // where E.l_name = 'Smith' and E.address_id = A.address_i
}

```

Because the two-phase approach to the query ([Example 11-8](#) and [Example 11-9](#)) accesses the database only twice, it is significantly faster than the approach illustrated in [Example 11-7](#).

Joins offer a significant performance increase under most circumstances. Batch reading offers a further performance advantage in that it allows for delayed loading through value holders, and has much better performance where the target objects are shared.

For example, if employees in [Example 11-7](#), [Example 11-8](#), and [Example 11-9](#) are at the same address, batch reading reads much less data than joining, because batch reading uses a SQL DISTINCT call to filter duplicate data.

Batch reading is available for one-to-one, one-to-many, many-to-many, direct collection, direct map and aggregate collection mappings. Joining is only available for one-to-one and one-to-many mappings. Note that one-to-many joining will return a large amount of duplicate data and so is normally less efficient than batch reading.

WARNING: Allowing an unverified SQL string to be passed into methods (for example: `readAllObjects(Class class, String sql)` method) makes your application vulnerable to SQL injection attacks.

Reading Case 3: Using Complex Custom SQL Queries

TopLink provides a high-level query mechanism. However, if your application requires a complex query, a direct SQL or stored procedure call may be the best solution.

For more information about executing SQL calls, see ["SQLCall"](#) on page 96-17.

Reading Case 4: Using View Objects

Some application operations require information from several objects rather than from just one. This can be difficult to implement, and resource-intensive. [Example 11-10](#) illustrates unoptimized code that reads information from several objects.

Example 11-10 No Optimization

```

/* Gather the information to report on an employee and return the summary of the

```


information. In this situation, a hash table is used to hold the report information. Notice that this reads a lot of objects from the database, but uses very little of the information contained in the objects. This may take 5 queries and read in a large number of objects */

```
public Hashtable reportOnEmployee(String employeeName) {
    Vector projects, associations;
    Hashtable report = new Hashtable();
    // Retrieve employee from database
    Employee employee = session.readObject(Employee.class,
        new ExpressionBuilder.get("lastName").equal(employeeName));
    // Get all the projects affiliated with the employee
    projects = session.readAllObjects(Project.class,
        "SELECT P.* FROM PROJECT P," +
        "EMPLOYEE E WHERE P.MEMBER_ID = E.EMP_ID AND E.L_NAME = " +
        employeeName);
    // Get all the associations affiliated with the employee
    associations = session.readAllObjects(Association.class, "SELECT A.* " +
        "FROM ASSOC A, EMPLOYEE E WHERE A.MEMBER_ID = E.EMP_ID AND E.L_NAME = " +
        employeeName);
    report.put("firstName", employee.getFirstName());
    report.put("lastName", employee.getLastName());
    report.put("manager", employee.getManager());
    report.put("city", employee.getAddress().getCity());
    report.put("projects", projects);
    report.put("associations", associations);
    return report;
}
```

To improve application performance in these situations, define a new read-only object to encapsulate this information, and map it to a view on the database. To set the object to be read-only, use the `addDefaultReadOnlyClass` API in the `oracle.toplink.sessions.Project` class.

Example 11–11 Optimization Through View Object

```
CREATE VIEW NAMED EMPLOYEE_VIEW AS (SELECT F_NAME = E.F_NAME, L_NAME = E.L_
NAME, EMP_ID = E.EMP_ID, MANAGER_NAME = E.NAME, CITY = A.CITY, NAME = E.NAME
FROM EMPLOYEE E, EMPLOYEE M, ADDRESS A
WHERE E.MANAGER_ID = M.EMP_ID
AND E.ADDRESS_ID = A.ADDRESS_ID)
```

Define a descriptor for the `EmployeeReport` class as follows:

- Define the descriptor as usual, but specify the `tableName` as `EMPLOYEE_VIEW`.
- Map only the attributes required for the report. In the case of the `numberOfProjects` and `associations`, use a transformation mapping to retrieve the required data.

You can now query the report from the database in the same way as any other object enabled by `TopLink`.

Example 11–12 View the Report from [Example 11–11](#)

```
/* Return the report for the employee */
public EmployeeReport reportOnEmployee(String employeeName) {
    EmployeeReport report;
    report = (EmployeeReport) session.readObject(EmployeeReport.class,
        new ExpressionBuilder.get("lastName").equal(employeeName));
    return report;
}
```

}

WARNING: Allowing an unverified SQL string to be passed into methods (for example: `readAllObjects(Class class, String sql)` and `readObject(Class class, String sql)` method) makes your application vulnerable to SQL injection attacks.

Reading Case 5: Inheritance Views

If you have an inheritance hierarchy that spans multiple tables and frequently query for the root class, consider defining an inheritance all-subclasses view. This allows a view to be used for queries against an inheritance superclass that can read all of its subclasses in a single query instead of multiple queries.

For more information about inheritance, see ["Descriptors and Inheritance"](#) on page 26-3.

For more information about querying on inheritance, see ["Querying on an Inheritance Hierarchy"](#) on page 99-4.

Write Optimization Examples

TopLink provides the write optimization features listed in [Table 11-13](#).

This section includes the following write optimization examples:

- [Writing Case: Batch Writes](#)

Table 11-13 Write Optimization Features

Feature	Effect on Performance
Unit of Work	Improves performance by updating only the changed fields and objects. Minimizes the amount of tracking required (which can be expensive) by registering only those objects that will change. For more information, see "Understanding TopLink Transactions" on page 100-1). Note: The Unit of Work supports marking classes as read-only (see "Configuring Read-Only Descriptors" on page 28-4 and "Declaring Read-Only Classes" on page 102-6). This avoids tracking of objects that do not change.
Batch writing	Lets you group all insert, update, and delete commands from a transaction into a single database call. This dramatically reduces the number of calls to the database (see "Cursors" on page 11-28 and "Batch Writing and Parameterized SQL" on page 11-28).
Parameterized SQL	Improves performance for frequently executed SQL statements (see "Parameterized SQL and Prepared Statement Caching" on page 11-17).
Sequence number preallocation	Dramatically improves insert performance. (see "Sequence Number Preallocation" on page 11-28).
Multiprocessing	Splitting a batch job across threads lets you synchronize reads from a cursored stream and use parallel Units of Work for performance improvements even on a single machine (see "Multiprocessing" on page 11-29).

Table 11–13 (Cont.) Write Optimization Features

Feature	Effect on Performance
<i>Does exist</i> alternatives	The <i>does exist</i> call on write object can be avoided in certain situations by checking the cache for <i>does exist</i> , or assuming the existence of the object (see "Configuring Existence Checking at the Project Level" on page 22-8 or "Configuring Cache Existence Checking at the Descriptor Level" on page 28-42 and "Using Registration and Existence Checking" on page 102-5).

Writing Case: Batch Writes

The most common write performance problem occurs when a batch job inserts a large volume of data into the database. For example, consider a batch job that loads a large amount of data from one database, and then migrates the data into another. The objects involved:

- Are simple individual objects with no relationships
- Use generated sequence numbers as their primary key
- Have an address that also uses a sequence number

The batch job loads 10,000 employee records from the first database and inserts them into the target database. With no optimization, the batch job reads all the records from the source database, acquires a Unit of Work from the target database, registers all objects, and commits the Unit of Work.

Example 11–13 No Optimization

```
/* Read all the employees, acquire a Unit of Work, and register them */

// Read all the employees from the database. This requires 1 SQL call,
// but will be very memory intensive as 10,000 objects will be read
Vector employees = sourceSession.readAllObjects(Employee.class);

//SQL: Select * from Employee

// Acquire a Unit of Work and register the employees
UnitOfWork uow = targetSession.acquireUnitOfWork();
uow.registerAllObjects(employees);
uow.commit();

// SQL: Begin transaction
// SQL: Update Sequence set count = count + 1 where name = 'EMP'
// SQL: Select count from Sequence
// SQL: ... repeat this 10,000 times + 10,000 times for the addresses ...
// SQL: Commit transaction
// SQL: Begin transaction
// SQL: Insert into Address (...) values (...)
// SQL: ... repeat this 10,000 times
// SQL: Insert into Employee (...) values (...)
// SQL: ... repeat this 10,000 times
// SQL: Commit transaction
```

This batch job performs poorly, because it requires 60,000 SQL executions. It also reads huge amounts of data into memory, which can raise memory performance issues. TopLink offers several optimization features to improve the performance of this batch job.

To improve this operation, do the following:

- Use TopLink batch read operations and cursor support (see ["Cursors"](#) on page 11-28).
- Use batch writing or parameterized batch writing to write to the database (see ["Batch Writing and Parameterized SQL"](#) on page 11-28).
If your database does not support batch writing, use parameterized SQL to implement the write query.
- Implement sequence number preallocation (see ["Sequence Number Preallocation"](#) on page 11-28).
- Implement multiprocessing (see ["Multiprocessing"](#) on page 11-29).

Cursors To optimize the query in [Example 11-13](#), use a cursored stream to read the Employees from the source database. You can also employ a weak identity map instead of a hard or soft cache identity map in both the source and target databases.

To address the potential for memory problems, use the `releasePrevious` method after each read to stream the cursor in groups of 100. Register each batch of 100 employees in a new Unit of Work and commit them.

Although this does not reduce the amount of executed SQL, it does address potential out-of-memory issues. When your system runs out of memory, the result is performance degradation that increases over time, and excessive disk activity caused by memory swapping on disk.

For more information, see ["Cursored Streams and Scrollable Cursors"](#) on page 11-18.

Batch Writing and Parameterized SQL Batch writing lets you combine a group of SQL statements into a single statement and send it to the database as a single database execution. This feature reduces the communication time between the application and the server, and substantially improves performance.

You can enable batch writing alone (dynamic batch writing) using `Login` method `useBatchWriting`. If you add batch writing to [Example 11-13](#), you execute each batch of 100 employees as a single SQL execution. This reduces the number of SQL executions from 20,200 to 300.

You can also enable batch writing and parameterized SQL (parameterized batch writing) and prepared statement caching. Parameterized SQL avoids the prepare component of SQL execution. This improves write performance because it avoids the prepare cost of an SQL execution. For parameterized batch writing you would get one statement per Employee, and one for Address: this reduces the number of SQL executions from 20,200 to 400. Although this is more than dynamic batch writing alone, parameterized batch writing also avoids all parsing, so it is much more efficient overall.

Although parameterized SQL avoids the prepare component of SQL execution, it does not reduce the number of executions. Because of this, parameterized SQL alone may not offer as big of a gain as batch writing. However, if your database does not support batch writing, parameterized SQL will improve performance. If you add parameterized SQL in [Example 11-13](#), you must still execute 20,200 SQL executions, but parameterized SQL reduces the number of SQL PREPAREs to 4.

For more information, see ["Batch Writing"](#) on page 11-15.

Sequence Number Preallocation SQL select calls are more resource-intensive than SQL modify calls, so you can realize large performance gains by reducing the number of select calls you issue. The code in [Example 11-13](#) uses the select calls to acquire

sequence numbers. You can substantially improve performance if you use sequence number preallocation.

In TopLink, you can configure the sequence preallocation size on the login object (the default size is 50). [Example 11-13](#) uses a preallocation size of 1 to demonstrate this point. If you stream the data in batches of 100 as suggested in ["Cursors"](#), set the sequence preallocation size to 100. Because employees and addresses in the example both use sequence numbering, you further improve performance by letting them share the same sequence. If you set the preallocation size to 200, this reduces the number of SQL execution from 60,000 to 20,200.

For more information about sequencing preallocation, see ["Sequencing and Preallocation Size"](#) on page 20-20.

Multiprocessing You can use multiple processes or multiple machines to split the batch job into several smaller jobs. In this example, splitting the batch job across threads enables you to synchronize reads from the cursored stream, and use parallel Units of Work on a single machine.

This leads to a performance increase, even if the machine has only a single processor, because it takes advantage of the wait times inherent in SQL execution. While one thread waits for a response from the server, another thread uses the waiting cycles to process its own database operation.

[Example 11-14](#) illustrates the optimized code for this example. Note that it does not illustrate multiprocessing.

Example 11-14 Fully Optimized

```
/* Read each batch of employees, acquire a Unit of Work, and register them */
targetSession.getLogin().useBatchWriting();
targetSession.getLogin().setSequencePreallocationSize(200);
targetSession.getLogin().bindAllParameters();
targetSession.getLogin().cacheAllStatements();
targetSession.getLogin().setMaxBatchWritingSize(200);

// Read all the employees from the database into a stream. This requires 1 SQL
// call, but none of the rows will be fetched.
ReadAllQuery query = new ReadAllQuery();
query.setReferenceClass(Employee.class);
query.useCursoredStream();
CursoredStream stream;
stream = (CursoredStream) sourceSession.executeQuery(query);
//SQL: Select * from Employee. Process each batch
while (! stream.atEnd()) {
    Vector employees = stream.read(100);
    // Acquire a unit of work to register the employees
    UnitOfWork uow = targetSession.acquireUnitOfWork();
    uow.registerAllObjects(employees);
    uow.commit();
}
//SQL: Begin transaction
//SQL: Update Sequence set count = count + 200 where name = 'SEQ'
//SQL: Select count from Sequence where name = 'SEQ'
//SQL: Commit transaction
//SQL: Begin transaction
//BEGIN BATCH SQL: Insert into Address (...) values (...)
//... repeat this 100 times
//Insert into Employee (...) values (...)
//... repeat this 100 times
```

```
//END BATCH SQL:  
//SQL: Commit transactionJava optimization
```

Unit of Work Optimization

For best performance when using a unit of work, consider the following tips:

- Register objects with a unit of work only if objects are eligible for change. If you register objects that will not change, the unit of work needlessly clones and processes those objects.
- Avoid the cost of existence checking when you are registering a new or existing object (see ["Using Registration and Existence Checking"](#) on page 102-5).
- Avoid the cost of change set calculation on a class you know will not change by telling the unit of work that the class is read-only (see ["Declaring Read-Only Classes"](#) on page 102-6).
- Avoid the cost of change set calculation on an object read by a `ReadAllQuery` in a unit of work that you do not intend to change by unregistering the object (see ["Unregistering Working Clones"](#) on page 102-6).
- Before using conforming queries, be sure that it is necessary. For alternatives, see ["Using Conforming Queries and Descriptors"](#) on page 102-8.

If your performance measurements show that you have a performance problem during unit of work commit, consider using object level or attribute level change tracking, depending on the type of objects involved and how they typically change. For more information, see ["Unit of Work and Change Policy"](#) on page 100-6.

Application Server and Database Optimization

Configuring your application server and database correctly can have a big impact on performance and scalability. Ensure that you correctly optimize these key components of your application in addition to your TopLink application and persistence.

For your application or J2EE server, ensure your memory, thread pool and connection pool sizes are sufficient for your server's expected load, and that your JVM has been configured optimally.

Ensure that your database has been configured correctly for optimal performance and its expected load.

This chapter includes the following sections:

- [Overview](#)
- [Creating Custom Data Types](#)
- [Using Public Source](#)

Overview

By design, TopLink can adapt to a variety of relational and nonrelational data sources.

To integrate TopLink with a data source that is not directly supported by the TopLink API, Oracle recommends that you use an EIS project (see ["EIS Projects"](#) on page 20-7) or an XML project (["XML Projects"](#) on page 20-9).

Using an EIS project, you can integrate your TopLink-enabled application with any nonrelational data source that supports a J2C adapter and any supported EIS record type, including indexed, mapped, or XML. If no J2C adapter exists for your target data source, you can concentrate your integration efforts on creating an adapter.

Simultaneously, you can build your application according to J2C specifications.

Although this still requires custom development effort, it is more efficient than trying to extend TopLink classes and provides you with a J2C adapter that you can leverage in any other project (making it a better value).

Using an XML project, you can integrate your TopLink-enabled application with Web services or other XML-message based designs.

The remainder of this chapter describes other customization options provided by the TopLink API.

Creating Custom Data Types

TopLink provides support for all the most common Java data types. [Table 12-1](#) lists the TopLink mapping extensions that you can use to support custom data types. You can also create your object converter to allow conversion between a data type and your own Java type.

Table 12-1 Mapping Extensions for Custom Data Types

Extension	Description
"Object Type Converter" on page 33-12	An extension of direct and direct collection mappings that lets you match a fixed number of data values to Java objects. Use this converter when the values in the schema differ from those in Java

Table 12–1 (Cont.) Mapping Extensions for Custom Data Types

Extension	Description
"Serialized Object Converter" on page 33-10	An extension of direct and direct collection mappings that lets you map serializable objects, such as multimedia data, to a binary format in a data source, such as a base64 element in an XML document or Binary Large Object (BLOB) field in a database
"Type Conversion Converter" on page 33-11	An extension of direct and direct collection mappings that lets you explicitly map a data source type to a Java type. For example, a <code>java.util.Date</code> in Java can be mapped to a <code>java.sql.Date</code> in the data source.
"Simple Type Translator" on page 33-12	An extension of direct and direct collection mappings that lets you automatically translate an XML element value to an appropriate Java type based on the element's <code><type></code> attribute as defined in your XML schema.

Using Public Source

The source code to most public classes is available in `<TOPLINK_HOME>\jlib\source.jar`.

This is provided for debugging purposes.

Part V

Troubleshooting a TopLink Application

This part describes how to troubleshoot a TopLink application at time of deployment and at run time. It contains the following chapters.

- [Chapter 13, "TopLink Exception Reference"](#)

This chapter contains detailed information on each TopLink error code and message.

- [Chapter 14, "TopLink Workbench Error Reference"](#)

This chapter describes common problems and their solutions when using TopLink Workbench.

- [Chapter 15, "Troubleshooting Application Deployment"](#)

This chapter explains common problems and solutions when deploying TopLink applications.

TopLink Exception Reference

This chapter lists the TopLink exception error codes, information about the likely cause of the problem and a possible corrective action. Each exception code corresponds to an exception class and includes the following information:

- The exception number in the format, EXCEPTION [TOPLINK-XXXX]
- A description of the problem, taken from the raised exception

This chapter contains information on the following exceptions:

- [Descriptor Exceptions \(1 – 200\)](#), on page 13-2
- [Concurrency Exceptions \(2001 – 2006\)](#), on page 13-25
- [Conversion Exceptions \(3001– 3008\)](#), on page 13-26
- [Database Exceptions \(4002 – 4018\)](#), on page 13-27
- [Optimistic Lock Exceptions \(5001 – 5008\)](#), on page 13-29
- [Query Exceptions \(6001 – 6121\)](#), on page 13-30
- [Validation Exceptions \(7001 – 7147\)](#), on page 13-42
- [EJB QL Exceptions \(8001 – 8010\)](#), on page 13-55
- [Session Loader Exceptions \(9000 - 9010\)](#), on page 13-56
- [EJB Exception Factory Exceptions \(10001 - 10069\)](#), on page 13-58
- [Communication Exceptions \(12000 - 12003\)](#), on page 13-64
- [XML Data Store Exceptions \(13000 - 13020\)](#), on page 13-64
- [Deployment Exceptions \(14001 - 14033\)](#), on page 13-67
- [Synchronization Exceptions \(15001 - 15025\)](#), on page 13-70
- [SDK Data Store Exceptions \(17001 - 17006\)](#), on page 13-71
- [EIS Exceptions \(17007 – 17025\)](#), on page 13-72
- [JMS Processing Exceptions \(18001 - 18002\)](#), on page 13-74
- [SDK Descriptor Exceptions \(19001 - 19003\)](#), on page 13-75
- [Default Mapping Exceptions \(20001 - 20008\)](#), on page 13-75
- [Discovery Exceptions \(22001 - 22004\)](#), on page 13-77
- [Remote Command Manager Exceptions \(22101 - 22111\)](#), on page 13-77
- [Transaction Exceptions \(23001 - 23011\)](#), on page 13-79
- [XML Conversion Exceptions \(25001 - 25016\)](#), on page 13-80

- [Migration Utility Exceptions \(26001 - 26017\)](#), on page 13-82
- [EJB JAR XML Exceptions \(72000 – 72022\)](#), on page 13-84

Descriptor Exceptions (1 – 200)

`DescriptorException` is a development exception that is raised when insufficient information is provided to the descriptor. The message that is returned includes the name of the descriptor or mapping that caused the exception. If a mapping within the descriptor caused the error, then the name and parameters of the mapping are part of the returned message, as shown in [Example 13–1](#).

Internal exception, mapping and descriptor appear only if TopLink has enough information about the source of the problem to provide this information.

Format

EXCEPTION [TOPLINK - error code]: Exception name
EXCEPTION DESCRIPTION: Message
INTERNAL EXCEPTION: Message
MAPPING: Database mapping
DESCRIPTOR: Descriptor

Example 13–1 Descriptor Exception

EXCEPTION [TOPLINK - 75]: oracle.toplink.exceptions.DescriptorException
EXCEPTION DESCRIPTION: The reference class is not specified.

1: ATTRIBUTE_AND_MAPPING_WITH_INDIRECTION_MISMATCH

Cause: `attributeName` is not declared as type `ValueHolderInterface`, but the mapping uses indirection. The mapping is set to use indirection, but the related attribute is not defined as type `ValueHolderInterface`. It is raised on foreign reference mappings.

Action: If you want to use indirection on the mapping, change the attribute to type `ValueHolderInterface`. Otherwise, change the mapping associated with the attribute so that it does not use indirection.

2: ATTRIBUTE_AND_MAPPING_WITHOUT_INDIRECTION_MISMATCH

Cause: `attributeName` is declared as type `ValueHolderInterface`, but TopLink is unable to use indirection. The attribute is defined to be of type `ValueHolderInterface`, but the mapping is not set to use indirection. It is raised on foreign reference mappings.

Action: If you do not want to use indirection on the mapping, change the attribute so it is not of type `ValueHolderInterface`. Otherwise, change the mapping associated with the attribute to use indirection.

6: ATTRIBUTE_NAME_NOT_SPECIFIED

Cause: The attribute name is missing or not specified in the mapping definition.

Action: Specify the attribute name in the mapping by calling the method `setAttributeName(String attributeName)`.

7: ATTRIBUTE_TYPE_NOT_VALID

Cause: When using Java 2, the specified `attributeName` is not defined as type `vector`, or a type that implements the `Map` or `Collection` interface. This

occurs in one-to-many mapping, many-to-many mapping, and collection mapping when mapping is set not to use indirection, and the attribute type is not declared.

Action: Declare the attribute to be of type `java.util.Vector`.

8: CLASS_INDICATOR_FIELD_NOT_FOUND

Cause: The class indicator field is defined, but the descriptor is set to use inheritance. When using inheritance, a class indicator field or class extraction method must be set. The class indicator field is used to create the right type of domain object.

Action: Set either a class indicator field or class extraction method.

9: DIRECT_FIELD_NAME_NOT_SET

Cause: The direct field name from the target table is not set in the direct collection mapping.

Action: Specify the direct field name by calling the method `setDirectFieldName(String fieldName)`.

10: FIELD_NAME_NOT_SET_IN_MAPPING

Cause: The field name is not set in the mapping. It is raised from direct to field mapping, array mapping, and structure mapping.

Action: Specify the field name by calling the method `setFieldName(String fieldName)`.

11: FOREIGN_KEYS_DEFINED_INCORRECTLY

Cause: One-to-one mapping foreign key is defined incorrectly. Multiple foreign key fields were set for one-to-one mapping by calling the method `setForeignKeyFieldName(String fieldName)`.

Action: Use the method `addForeignKeyFieldName(String sourceForeignKeyName, String targetPrimaryKeyFieldName)` to add multiple foreign key fields.

12: IDENTITY_MAP_NOT_SPECIFIED

Cause: The descriptor must use an identity map to use the **Check cache does exist** option. The descriptor has been set not to use identity map, but the existence checking is set to be performed on identity map.

Action: Either use identity map, or set the existence checking to some other option.

13: ILLEGAL_ACCESS_WHILE_GETTING_VALUE_THRU_INSTANCE_VARIABLE_ACCESSOR

Cause: `TopLink` is unable to access the `attributeName` instance variable in object `objectName`. The instance variable in the domain object is not accessible. This exception is raised when `TopLink` tries to access the instance variable using the `java.lang.reflect` Java package. The error is a purely Java exception, and `TopLink` wraps only the reflection exception.

Action: Inspect the internal exception, and refer to the Java documentation.

14: ILLEGAL_ACCESS_WHILE_CLONING

Cause: `TopLink` is unable to clone the object `domainObject` because the clone method `methodName` is not accessible. The method name specified using `useCloneCopyPolicy(String cloneMethodName)` or the `clone()` method to create the clone on the domain object, is not accessible by `TopLink` using Java

reflection. The error is a purely Java exception, and TopLink wraps only the reflection exception.

Action: Inspect the internal exception, and refer to the Java documentation.

15: ILLEGAL_ACCESS_WHILE_CONSTRUCTOR_INSTANTIATION

Cause: The domain class does not define a public default constructor, which TopLink needs to create new instances of the domain class.

Action: Define a public default constructor or use a different instantiation policy.

16: ILLEGAL_ACCESS_WHILE_EVENT_EXECUTION

Cause: The descriptor callback method `eventMethodName` with `DescriptorEvent` as an argument is not accessible. This exception is raised when TopLink tries to access the event method using Java reflection. The error is a purely Java exception, and TopLink wraps only the reflection exception.

Action: Inspect the internal exception, and refer to the Java documentation.

17: ILLEGAL_ACCESS_WHILE_GETTING_VALUE_THRU_METHOD_ACCESSOR

Cause: Trying to invoke inaccessible `methodName` on the object `objectName`. The underlying get accessor method to access an attribute in the domain object is not accessible. This exception is raised when TopLink tries to access an attribute through a method using the `java.lang.reflect` Java package. The error is a purely Java exception, and TopLink wraps only the reflection exception.

Action: Inspect the internal exception, and refer to the Java documentation.

18: ILLEGAL_ACCESS_WHILE_INSTANTIATING_METHOD_BASED_PROXY

Cause: The method used by the transformation mapping using a valueholder is invalid. This exception is raised when TopLink tries to access the method using Java reflection. The problem occurs when the method base valueholder is instantiated.

Action: Inspect the internal exception, and refer to the Java documentation.

19: ILLEGAL_ACCESS_WHILE_INVOKING_ATTRIBUTE_METHOD

Cause: On transformation mapping, the underlying attribute method that is used to retrieve values from the database row while reading the transformation mapped attribute is not accessible.

Action: Inspect the internal exception, and refer to the Java documentation.

20: ILLEGAL_ACCESS_WHILE_INVOKING_FIELD_TO_METHOD

Cause: On transformation mapping, the method `methodName` that is used to retrieve value from the object while writing the transformation mapped attribute is not accessible. The error is a purely Java exception, and TopLink wraps only the reflection exception.

Action: Inspect the internal exception, and refer to the Java documentation.

21: ILLEGAL_ACCESS_WHILE_INVOKING_ROW_EXTRACTION_METHOD

Cause: TopLink was unable to extract data row, because TopLink cannot access the row specified in the `databaseRow` argument of the method. The method to extract class from row on the domain object is not accessible. The error is a purely Java exception, and TopLink wraps only the reflection exception.

Action: Inspect the internal exception, and refer to the Java documentation.

22: ILLEGAL_ACCESS_WHILE_METHOD_INSTANTIATION

Cause: TopLink is unable to create a new instance, because the method `methodName` that creates instances on the domain class is not accessible. The error is a purely Java exception, and TopLink wraps only the reflection exception.

Action: Inspect the internal exception, and refer to the Java documentation.

23: ILLEGAL_ACCESS_WHILE_OBSOLETE_EVENT_EXECUTION

Cause: The descriptor callback method `eventMethodName` with `Session` as an argument is inaccessible. This exception is raised when TopLink tries to access the event method using Java reflection. The error is a purely Java exception, and TopLink wraps only the reflection exception.

Action: Inspect the internal exception, and refer to the Java documentation.

24: ILLEGAL_ACCESS_WHILE_SETTING_VALUE_THRU_INSTANCE_VARIABLE_ACCESSOR

Cause: The `attributeName` instance variable in the object `objectName` is not accessible through Java reflection. The error is raised by Java, and TopLink wraps only the reflection exception.

Action: Inspect the internal exception, and refer to the Java documentation.

25: ILLEGAL_ACCESS_WHILE_SETTING_VALUE_THRU_METHOD_ACCESSOR

Cause: TopLink is unable to invoke a method `setMethodName` on the object with parameter `parameter`. The attribute's set accessor method is not accessible through Java reflection. The error is raised by Java and TopLink wraps only the reflection exception.

Action: Inspect the internal exception, and refer to the Java documentation.

26: ILLEGAL_ARGUMENT_WHILE_GETTING_VALUE_THRU_INSTANCE_VARIABLE_ACCESSOR

Cause: TopLink is unable to get a value for an instance variable `attributeName` of type `typeName` from the object. The specified object is not an instance of the class or interface declaring the underlying field. An object is accessed to get the value of an instance variable that does not exist.

Action: Inspect the internal exception, and refer to the Java documentation.

27: ILLEGAL_ARGUMENT_WHILE_GETTING_VALUE_THRU_METHOD_ACCESSOR

Cause: TopLink is unable to invoke method `methodName` on the object `objectName`. The get accessor method declaration on the domain object differs from the one that is defined. The number of actual and formal parameters differ, or an unwrapping conversion has failed.

Action: Inspect the internal exception, and refer to the Java documentation.

28: ILLEGAL_ARGUMENT_WHILE_INSTANTIATING_METHOD_BASED_PROXY

Cause: The method that the method-based proxy uses in a transformation mapping is receiving invalid arguments when the valueholder is being instantiated. This exception is raised when TopLink tries to access the method using the `java.lang.reflect` Java package.

Action: Inspect the internal exception, and refer to the Java documentation.

29: ILLEGAL_ARGUMENT_WHILE_INVOKING_ATTRIBUTE_METHOD

Cause: The number of actual and formal parameters differ, or an unwrapping conversion has failed. On transformation mapping, the method used to retrieve values from the database row while reading the transformation mapped attribute is getting an invalid argument.

Action: Inspect the internal exception, and refer to the Java documentation.

30: ILLEGAL_ARGUMENT_WHILE_INVOKING_FIELD_TO_METHOD

Cause: The number of actual and formal parameters differs for method `methodName`, or an unwrapping conversion has failed. On transformation mapping, the method used to retrieve value from the object while writing the transformation mapped attribute is getting an invalid argument. The error is a purely Java exception, and TopLink wraps only the reflection exception.

Action: Inspect the internal exception, and refer to the Java documentation.

31: ILLEGAL_ARGUMENT_WHILE_OBSOLETE_EVENT_EXECUTION

Cause: The number of actual and formal parameters for the descriptor callback method `eventMethodName` differs, or an unwrapping conversion has failed. The callback event method is invoked with an invalid argument. This exception is raised when TopLink tries to invoke the event method using Java reflection. The error is a purely Java exception, and TopLink wraps only the reflection exception.

Action: Inspect the internal exception, and refer to the Java documentation.

32: ILLEGAL_ARGUMENT_WHILE_SETTING_VALUE_THRU_INSTANCE_VARIABLE_ACCESSOR

Cause: An invalid value is being assigned to the attribute instance variable. TopLink is unable to set a value for an instance variable `attributeName` of type `typeName` in the object. The specified object is not an instance of the class or interface that is declaring the underlying field, or an unwrapping conversion has failed.

TopLink assigns value by using Java reflection. Java raises the error and TopLink wraps only the reflection exception.

Action: Inspect the internal exception, and refer to the Java documentation.

33: ILLEGAL_ARGUMENT_WHILE_SETTING_VALUE_THRU_METHOD_ACCESSOR

Cause: An illegal argument is being passed to the attribute's set accessor method. TopLink is unable to invoke method `setMethodName` on the object. The number of actual and formal parameters differs, or an unwrapping conversion has failed. Java raises the error and TopLink wraps only the reflection exception.

Action: Inspect the internal exception, and refer to the Java documentation.

34: INSTANTIATION_WHILE_CONSTRUCTOR_INSTANTIATION

Cause: The class does not define a public default constructor, or the constructor raised an exception. This error occurs when you invoke the default constructor for the domain object to create a new instance of the object while building new domain objects if:

- The class represents an abstract class, an interface, an array class, a primitive type, or void.
- The instantiation fails for some other reason.

Java raises the error and TopLink wraps only the reflection exception.

Action: Inspect the internal exception, and refer to the Java documentation.

35: INVALID_DATA_MODIFICATION_EVENT

Cause: Applications should never encounter this exception. This exception usually occurs at the time of developing TopLink, although in cases where the developer writes new mapping, it is possible to get this exception. In direct collection mapping and many-to-many mapping, the target table and relational table are populated at the end of the commit process, and if a data modification event is sent to any other mapping, then this exception is raised.

Action: Contact Oracle Support Services.

36: INVALID_DATA_MODIFICATION_EVENT_CODE

Cause: An application should never encounter this exception. This exception usually occurs at the time of developing TopLink, although in cases where developers write new mappings, it is possible to get this exception. In direct collection mapping and many-to-many mapping, the target table and relational table are populated at the end of the commit process, and if a data modification event is sent to these two mappings with wrong event code, then this exception is raised.

Action: Contact Oracle Support Services.

37: INVALID_DESCRIPTOR_EVENT_CODE

Cause: An application should never encounter this exception. This exception usually occurs at the time of developing TopLink. The exception means that the descriptor event manager does not support the event code passed in the event.

Action: Contact Oracle Support Services.

38: INVALID_IDENTITY_MAP

Cause: The identity map constructor failed because an invalid identity map was specified. The identity map class given in the descriptor cannot be instantiated. The exception is a Java exception that is raised by a Java reflection when TopLink instantiates the identity map class. TopLink wraps only the Java exception.

Action: Inspect the internal exception, and refer to the Java documentation.

39: JAVA_CLASS_NOT_SPECIFIED

Cause: The descriptor does not define a Java class. The Java class is not specified in the descriptor.

Action: Specify the Java Class.

40: DESCRIPTOR_FOR_INTERFACE_IS_MISSING

Cause: A descriptor for the referenced interface is not added to the session.

Action: Add that descriptor to the session.

41: MAPPING_FOR_SEQUENCE_NUMBER_FIELD

Cause: A non-read-only mapping is not defined for the sequence number field. A mapping is required so that TopLink can put and extract values for the primary key.

Action: Define a mapping.

43: MISSING_CLASS_FOR_INDICATOR_FIELD_VALUE

Cause: TopLink is missing the class for indicator field value `classFieldValue` of type `type`. There was no class entry found in the inheritance policy for the indicator field value that was read from the database. It is likely that the method `addClassIndicator(Class class, Object typeValue)` was not called for the field value. The class and `typeValue` are stored in a hash table, and later

the class is extracted from the hash table by passing `typeValue` as a key. Because `Integer(1)` is not equivalent to `Float(1)`, this exception occurs when the type of `typeValue` is incorrectly specified.

Action: Verify the descriptor.

44: MISSING_CLASS_INDICATOR_FIELD

Cause: The class indicator field is missing from the database row that was read from the database. This is performed in the inheritance model where after reading rows from the database, child domain objects are to be constructed depending upon the type indicator values.

Action: Verify the printed row for correct spelling.

45: MISSING_MAPPING_FOR_FIELD

Cause: `TopLink` is missing a mapping for `field`; a mapping for the field is not specified.

Action: Define a mapping for the field.

46: NO_MAPPING_FOR_PRIMARY_KEY

Cause: A mapping for the primary key is not specified. There should be one non-read-only mapping defined for the primary key field.

Action: Define a mapping for the primary key.

47: MULTIPLE_TABLE_PRIMARY_KEY_NOT_SPECIFIED

Cause: The multiple table primary key mapping must be specified when a custom multiple table join is used. If multiple tables are specified in the descriptor and the join expression is customized, then the primary keys for all the tables must be specified. If the primary keys are not specified, then the exception occurs.

Action: Call the method `addMultipleTablePrimaryKeyFieldName(String fieldNameInPrimaryTable, String fieldNameInSecondaryTable)` on the descriptor to set the primary keys.

48: MULTIPLE_WRITE_MAPPINGS_FOR_FIELD

Cause: Multiple writable mappings for the field `fieldName` are defined in the descriptor. Exactly one must be defined as writable; the others must be specified as read-only. When multiple write mappings are defined for the field, `TopLink` is unable to choose the appropriate mapping for writing the value of the field in the database row. Therefore, the exception is raised during the validation process of descriptors.

The most common cause of this problem occurs when the field has direct-to-field mapping, as well as one-to-one mapping. In this case, the one-to-one mapping must either be read-only or a target foreign key reference.

Action: Make one of those mappings read-only.

49: NO_ATTRIBUTE_TRANSFORMATION_METHOD

Cause: The attribute transformation method name in the transformation mapping is not specified. This method is invoked internally by `TopLink` to retrieve value to store in the domain object.

Action: Define a method and set the method name on the mapping by calling the method `setAttributeTransformation(String methodName)`.

50: NO_FIELD_NAME_FOR_MAPPING

Cause: No field name is specified in direct-to-field mapping.

Action: Set the field by calling `setFieldName(String fieldName)`.

51: NO_FOREIGN_KEYS_ARE_SPECIFIED

Cause: Neither the selection criteria nor the foreign keys were specified on one-to-one mapping. If the selection criterion is not specified, then TopLink tries to build one from the foreign keys specified in the mapping.

Action: Specify the fields.

52: NO_REFERENCE_KEY_IS_SPECIFIED

Cause: No query key named `queryKey` is found in `descriptor`. No reference key from the target table is specified on direct collection mapping.

Action: Specify the fields by calling the method `setReferenceKeyFieldName(String fieldName)`.

53: NO_RELATION_TABLE

Cause: The relation table name is not set in this many-to-many mapping.

Action: Set the relation table name by calling the method `setRelationTableName(String tableName)`.

54: NO_SOURCE_RELATION_KEYS_SPECIFIED

Cause: There are no source relation keys specified in this many-to-many mapping.

Action: Add source relation keys to the mapping.

55: NO_SUCH_METHOD_ON_FIND_OBSOLETE_METHOD

Cause: TopLink cannot find the descriptor callback method `selector` on the domain class. It must take a `Session` or a `DescriptorEvent` as its argument. TopLink tries to invoke the method using Java reflection. It is a Java exception and TopLink is wrapping only the main exception.

Action: Inspect the internal exception, and refer to the Java documentation.

56: NO_SUCH_METHOD_ON_INITIALIZING_ATTRIBUTE_METHOD

Cause: TopLink cannot find the method `attributeMethodName` with parameters `databaseRow` or `databaseRow`, `session`. TopLink wraps the Java reflection exception that is caused when the method is being created from the method name. This method is set by calling `setAttributeMethodName(String aMethodName)`.

Action: Inspect the internal exception, and refer to the Java documentation.

57: NO_SUCH_METHOD_WHILE_CONSTRUCTOR_INSTANTIATION

Cause: The constructor is inaccessible to TopLink. TopLink wraps the Java reflection exception that is caused when it is creating a new instance of the domain.

Action: Inspect the internal exception, and refer to the Java documentation.

58: NO_SUCH_METHOD_WHILE_CONVERTING_TO_METHOD

Cause: TopLink failed to find a method with signature `methodName` or `methodName(oracle.toplink.sessions.Session)`. TopLink wraps the Java reflection exception that was raised by its attempt to create a `Method` type (`java.lang.reflect`) from the method names in the transformation mapping.

Action: Ensure that the method `methodName` is defined on the domain class that owns the attribute mapped by the transformation mapping.

59: NO_SUCH_FIELD_WHILE_INITIALIZING_ATTRIBUTES_IN_INSTANCE_VARIABLE_ACCESSOR

Cause: The instance variable `attributeName` is not defined in the domain class, or it is not accessible. TopLink wraps the Java reflection exception that is caused when it is creating a `Field` type (`java.lang.reflect.Field`) from the attribute name.

Action: Inspect the internal exception, and refer to the Java documentation.

60: NO_SUCH_METHOD_WHILE_INITIALIZING_ATTRIBUTES_IN_METHOD_ACCESSOR

Cause: The method `setMethodName` or `getMethodName` is not defined for the attribute in the domain class `javaClassName`, or it is not accessible. TopLink wraps the Java reflection exception that is caused when it is creating a `Method` type from the method name.

Action: Inspect the internal exception, and refer to the Java documentation.

61: NO_SUCH_METHOD_WHILE_INITIALIZING_CLASS_EXTRACTION_METHOD

Cause: The static class extraction method `methodName` with `databaseRow` as an argument does not exist, or is not accessible. A Java reflection exception wrapped in a TopLink exception is raised when a class extraction method is being created from the method name in the inheritance policy.

Action: Inspect the internal exception, and refer to the Java documentation.

62: NO_SUCH_METHOD_WHILE_INITIALIZING_COPY_POLICY

Cause: The clone method `methodName` with no arguments does not exist, or is not accessible. A Java reflection exception wrapped in a TopLink exception is raised when a method to create clones is being created from the method name in the copy policy.

Action: Inspect the internal exception, and refer to the Java documentation.

63: NO_SUCH_METHOD_WHILE_INITIALIZING_INSTANTIATION_POLICY

Cause: The instance creation method `methodName` with no arguments does not exist, or is not accessible. A Java reflection exception wrapped in a TopLink exception is raised when a method to create the new instance is being created from the method name in the instantiation policy.

Action: Inspect the internal exception, and refer to the Java documentation.

64: NO_TARGET_FOREIGN_KEYS_SPECIFIED

Cause: The foreign keys in the target table are not specified in one-to-many mappings. These fields are not required if a selection criterion is given in the mapping, but otherwise they must be specified.

Action: Set target foreign keys or selection criteria.

65: NO_TARGET_RELATION_KEYS_SPECIFIED

Cause: There are no target relation keys specified in many-to-many mappings.

Action: Call method `addTargetRelationKeyFieldName(String targetRelationKeyFieldName, String targetPrimaryKeyFieldName)` to set the fields.

66: NOT_DESERIALIZABLE

Cause: The object cannot be deserialized from the byte array that is read from the database. The exception is raised when the serialized object mapping is converting the byte array into an object.

Action: Inspect the internal exception, and refer to the Java documentation.

67: NOT_SERIALIZABLE

Cause: The object cannot be serialized into a byte array. The exception is raised when a serialized object mapping is converting the object into a byte array.

Action: Inspect the internal exception, and refer to the Java documentation.

68: NULL_FOR_NON_NULLAggregate

Cause: The value of the aggregate in the source object `object` is `null`. Null values are not allowed for aggregate mappings unless **allow null** is specified in the aggregate mapping.

Action: Call the mapping method `allowNull`. Provide parameters only if you are making a distinction between `foo()` and `foo(integer)`.

69: NULL_POINTER_WHILE_GETTING_VALUE_THRU_INSTANCE_VARIABLE_ACCESSOR

Cause: An object is accessed to get the value of an instance variable through Java reflection. This exception is raised only on some JVMs.

Action: Inspect the internal exception, and refer to the Java documentation.

70: NULL_POINTER_WHILE_GETTING_VALUE_THRU_METHOD_ACCESSOR

Cause: The getter method is invoked to get the value of an attribute through Java reflection. This exception is raised only on some JVM.

Action: Inspect the internal exception, and refer to the Java documentation.

71: NULL_POINTER_WHILE_SETTING_VALUE_THRU_INSTANCE_VARIABLE_ACCESSOR

Cause: A null pointer exception has been raised while setting the value of the `attributeName` instance variable in the object to `value`. An object is accessed to set the value of an instance variable through Java reflection. This exception is raised only on some JVMs.

Action: Inspect the internal exception, and refer to the Java documentation.

72: NULL_POINTER_WHILE_SETTING_VALUE_THRU_METHOD_ACCESSOR

Cause: A null pointer exception has been raised while setting the value through `setMethodName` method in the object with an argument `argument`. The set accessor method is invoked to set the value of an attribute through Java reflection. This exception is raised only on some JVMs.

Action: Inspect the internal exception, and refer to the Java documentation.

73: PARENT_DESCRIPTOR_NOT_SPECIFIED

Cause: `TopLink` is unable to find the descriptor for the parent class. The descriptor of a subclass has no parent descriptor.

Action: The method `setParentClass(Class parentClass)` must be called on the subclass descriptor.

74: PRIMARY_KEY_FIELDS_NOT_SPECIFIED

Cause: The primary key fields are not set for this descriptor.

Action: Add primary key field names using method `setPrimaryKeyName(String fieldName)`.

75: REFERENCE_CLASS_NOT_SPECIFIED

Cause: The reference class is not specified in the foreign reference mapping.

Action: Set the reference class by calling the method `setReferenceClass(Class aClass)`.

77: REFERENCE_DESCRIPTOR_IS_NOT_AGGREGATE

Cause: The referenced descriptor for `className` should be set to an aggregate descriptor. An aggregate mapping should always reference a descriptor that is aggregate.

Action: Call the method `descriptorIsAggregate` on the referenced descriptor.

78: REFERENCE_KEY_FIELD_NOT_PROPERLY_SPECIFIED

Cause: The table for the reference field must be the reference table. If the reference field name that is specified in the direct collection mapping is qualified with the table name, then the table name should match the reference table name.

Action: Qualify the field with the proper name, or change the reference table name.

79: REFERENCE_TABLE_NOT_SPECIFIED

Cause: The reference table name in the direct collection mapping is not specified.

Action: Use the method `setReferenceTableName(String tableName)` on the mapping to set the table name.

80: RELATION_KEY_FIELD_NOT_PROPERLY_SPECIFIED

Cause: The table for the relation key field must be the relation table. If the source and target relation fields names that are specified in the many-to-many mapping are qualified with the table name, then the table name should match the relation table name.

Action: Qualify the field with the proper name, or change the relation table name.

81: RETURN_TYPE_IN_GET_ATTRIBUTE_ACCESSOR

Cause: The method `attributeMethodName` that is specified in the transformation mapping should have a return type set in the attribute because this method is used to extract value from the database row.

Action: Verify the method and make appropriate changes.

82: SECURITY_ON_FIND_METHOD

Cause: The descriptor callback method `selector` with `DescriptorEvent` as an argument is not accessible. Java raises a security exception when a `Method` type is created from the method name using Java reflection. The method is a descriptor event callback on the domain object that takes `DescriptorEvent` as its parameter.

Action: Inspect the internal exception, and refer to the Java documentation.

83: SECURITY_ON_FIND_OBSOLETE_METHOD

Cause: The descriptor callback method `selector` with `session` as an argument is not accessible. Java raises a security exception when a `Method` type is created from the method name using Java reflection. The method is a descriptor event callback on the domain object, which takes class and session as its parameters.

Action: Inspect the internal exception, and refer to the Java documentation.

84: SECURITY_ON_INITIALIZING_ATTRIBUTE_METHOD

Cause: Access to the method `attributeMethodName` with parameters `databaseRow` or `databaseRow`, `Session` has been denied. Java raises a security exception when a `Method` type is created from the attribute method name using Java reflection. The attribute method that is specified in the transformation mapping is used to extract value from the database row and set by calling `setAttributeTransformation(String methodName)`.

Action: Inspect the internal exception, and refer to the Java documentation.

85: SECURITY_WHILE_CONVERTING_TO_METHOD

Cause: `TopLink` failed to find a method with signature `methodName()` or `methodName(oracle.toplink.sessions.Session)`. Java raises a security exception when a `Method` type is created from the method name using Java reflection. These are the methods that extract the field value from the domain object in the transformation mapping.

Action: Inspect the internal exception, and refer to the Java documentation.

86: SECURITY_WHILE_INITIALIZING_ATTRIBUTES_IN_INSTANCE_VARIABLE_ACCESSOR

Cause: Access to the instance variable `attributeName` in the class `javaClassName` is denied. Java raises a security exception when creating a `Field` type from the given attribute name using Java reflection.

Action: Inspect the internal exception, and refer to the Java documentation.

87: SECURITY_WHILE_INITIALIZING_ATTRIBUTES_IN_METHOD_ACCESSOR

Cause: The methods `setMethodName` and `getMethodName` in the object `javaClassName` are inaccessible. Java raises a security exception when creating a `Method` type from the given attribute accessor method name using Java reflection.

Action: Inspect the internal exception, and refer to the Java documentation.

88: SECURITY_WHILE_INITIALIZING_CLASS_EXTRACTION_METHOD

Cause: The static class extraction method `methodName` with `DatabaseRow` as an argument is not accessible. Java raises a security exception when creating a `Method` type from the given class extraction method name using Java reflection. The method is used to extract the class from the database row in the inheritance policy.

Action: Inspect the internal exception, and refer to the Java documentation.

89: SECURITY_WHILE_INITIALIZING_COPY_POLICY

Cause: The clone method `methodName` with no arguments is inaccessible. Using `ClassDescriptor` method `useCloneCopyPolicy(java.lang.String methodName)`, you can specify that the creation of clones within a unit of work is done by sending the `methodName` method to the original object. If the clone method `methodName` with no arguments is inaccessible (your application does not have sufficient privileges to call the method), Java raises a security exception when reflectively accessing the method with the given method name using the `java.lang.reflect` Java package.

Action: Inspect the internal exception, and refer to the Java documentation.

90: SECURITY_WHILE_INITIALIZING_INSTANTIATION_POLICY

Cause: The instance creation method `methodName` with no arguments is inaccessible. Using any of the `ClassDescriptor` methods `useFactoryInstantiationPolicy(java.lang.Class factoryClass, java.lang.String methodName)`, `useFactoryInstantiationPolicy(java.lang.Class factoryClass, java.lang.String methodName, java.lang.String factoryMethodName)`, `useFactoryInstantiationPolicy(java.lang.Object factory, java.lang.String methodName)`, or `useMethodInstantiationPolicy(java.lang.String staticMethodName)`, you can specify how new instances are created. If any of the methods or factory methods are inaccessible (your application does not have sufficient privileges to call the method), Java raises a security exception when reflectively accessing the method with the given method name using the `java.lang.reflect` Java package.

Action: Inspect the internal exception, and refer to the Java documentation.

91: SEQUENCE_NUMBER_PROPERTY_NOT_SPECIFIED

Cause: Either the sequence field name or the sequence number name is missing. To use sequence-generated IDs, both the sequence number name and field name properties must be set.

Action: To use sequence-generated IDs, set both the sequence number name and field name properties.

92: SIZE_MISMATCH_OF_FOREIGN_KEYS

Cause: The size of the primary keys on the target table does not match the size of the foreign keys on the source in one-to-one mapping.

Action: Verify the mapping and the reference descriptor's primary keys.

93: TABLE_NOT_PRESENT

Cause: The table `tableName` is not present in the descriptor.

Action: Verify the qualified field names that are specified in the mappings and descriptor so that any fields that are qualified with the table name reference the correct table.

94: TABLE_NOT_SPECIFIED

Cause: No table is specified in the descriptor. The descriptor must have a table name defined.

Action: Call the method `addTableName(String tableName)` or `setTableName(String tableName)` to set the tables on the descriptor.

96: TARGET_FOREIGN_KEYS_SIZE_MISMATCH

Cause: The size of the foreign keys on the target table does not match the size of the source keys on the source table in the one-to-many mapping.

Action: Verify the mapping.

97: TARGET_INVOCATION_WHILE_CLONING

Cause: `TopLink` has encountered a problem in cloning the object `domainObject` clone method. The `methodName` triggered an exception. Java raises this exception when the cloned object is invoked while the object is being cloned. The clone method is specified on the copy policy that is usually invoked to create clones in unit of work.

Action: Inspect the internal exception, and refer to the Java documentation.

98: TARGET_INVOCATION_WHILE_EVENT_EXECUTION

Cause: A descriptor callback method `eventMethodName` that includes a `DescriptorEvent` as argument is not accessible. The exception occurs when the descriptor event method is invoked using Java reflection.

Action: Inspect the internal exception, and refer to the Java documentation.

99: TARGET_INVOCATION_WHILE_GETTING_VALUE_THRU_METHOD_ACCESSOR

Cause: The method `methodName` on the object `objectName` is throwing an exception. Java is throwing an exception while getting an attribute value from the object through a method accessor.

Action: Inspect the internal exception, and refer to the Java documentation.

100: TARGET_INVOCATION_WHILE_INSTANTIATING_METHOD_BASED_PROXY

Cause: A method has raised an exception. Java raises this exception while instantiating a method based proxy and instantiating transformation mapping.

Action: Inspect the internal exception, and refer to the Java documentation.

101: TARGET_INVOCATION_WHILE_INVOKING_ATTRIBUTE_METHOD

Cause: The underlying method raises an exception. Java is throwing an exception while invoking an attribute transformation method on transformation mapping. The method is invoked to extract value from the database row to set into the domain object.

Action: Inspect the internal exception, and refer to the Java documentation.

102: TARGET_INVOCATION_WHILE_INVOKING_FIELD_TO_METHOD

Cause: The method `methodName` is throwing an exception. Java is throwing exception while invoking field transformation method on transformation mapping. The method is invoked to extract value from the domain object to set into the database row.

Action: Inspect the internal exception, and refer to the Java documentation.

103: TARGET_INVOCATION_WHILE_INVOKING_ROW_EXTRACTION_METHOD

Cause: `TopLink` encountered a problem extracting the class type from row `row` while invoking a class extraction method.

Action: Inspect the internal exception, and refer to the Java documentation.

104: TARGET_INVOCATION_WHILE_METHOD_INSTANTIATION

Cause: `TopLink` is unable to create a new instance. The creation method `methodName` caused an exception.

Action: Inspect the internal exception, and refer to the Java documentation.

105: TARGET_INVOCATION_WHILE_OBSOLETE_EVENT_EXECUTION

Cause: The underlying descriptor callback method `eventMethodName` with `session` as argument raises an exception. Java is throwing an exception while invoking a descriptor event method that takes a session as its parameter.

Action: Inspect the internal exception, and refer to the Java documentation.

106: TARGET_INVOCATION_WHILE_SETTING_VALUE_THRU_METHOD_ACCESSOR

Cause: The method `setMethodName` on the object raises an exception. Java is throwing an exception while invoking a set accessor method on the domain object to set an attribute value into the domain object.

Action: Inspect the internal exception, and refer to the Java documentation.

108: `VALUE_NOT_FOUND_IN_CLASS_INDICATOR_MAPPING`

Cause: The indicator value is not found in the class indicator mapping in the parent descriptor for the class.

Action: Verify the `addClassIndicator(Class childClass, Object typeValue)` on the inheritance policy.

109: `WRITE_LOCK_FIELD_IN_CHILD_DESCRIPTOR`

Cause: The child descriptor has a write-lock field defined. This is unnecessary, because it inherits any required locking from the parent descriptor.

Action: Check your child descriptor, and remove the field.

110: `DESCRIPTOR_IS_MISSING`

Cause: The descriptor for the reference class `className` is missing from the mapping.

Action: Verify the session to see if the descriptor for the reference class was added.

111: `MULTIPLE_TABLE_PRIMARY_KEY_MUST_BE_FULLY_QUALIFIED`

Cause: Multiple table primary key field names are not fully qualified. These field names are given on the descriptor if it has more than one table.

Action: Specify the field names with the table name.

112: `ONLY_ONE_TABLE_CAN_BE_ADDED_WITH_THIS_METHOD`

Cause: You have tried to enter more than one table through this method.

Action: Use the method `addTableName(String tableName)` to add multiple tables to the descriptor.

113: `NULL_POINTER_WHILE_CONSTRUCTOR_INSTANTIATION`

Cause: The constructor is inaccessible. Java is throwing this exception while invoking a default constructor to create new instances of the domain object.

Action: Inspect the internal exception, and refer to the Java documentation.

114: `NULL_POINTER_WHILE_METHOD_INSTANTIATION`

Cause: The new instance `methodName` creation method is inaccessible. Java is throwing an exception while calling a method to a build new instance of the domain object. This method is given by the user to override the default behavior of creating new instances through a class constructor.

Action: Inspect the internal exception, and refer to the Java documentation.

115: `NO_ATTRIBUTE_VALUE_CONVERSION_TO_FIELD_VALUE_PROVIDED`

Cause: The field conversion value for the attribute value `attributeValue` was not given in the object type mapping.

Action: Verify the attribute value, and provide a corresponding field value in the mapping.

116: `NO_FIELD_VALUE_CONVERSION_TO_ATTRIBUTE_VALUE_PROVIDED`

Cause: The attribute conversion value for the `fieldValue` was not given in the object type mapping.

Action: Verify the field value, and provide a corresponding attribute value in the mapping.

118: LOCK_MAPPING_CANNOT_BE_READONLY

Cause: The domain object `className` cannot have a read-only mapping for the write-lock fields when the version value is stored in the object.

Action: Verify the mappings on the write-lock fields.

119: LOCK_MAPPING_MUST_BE_READONLY

Cause: The domain object `className` does not have a read-only mapping for the write-lock fields when the version value is stored in the cache.

Action: Verify the mappings on write-lock fields.

120: CHILD_DOES_NOT_DEFINE_ABSTRACT_QUERY_KEY

Cause: The query key `queryKeyName` is defined in the parent descriptor but not in the child descriptor. The descriptor has not defined the abstract query key.

Action: Define any class that implements the interface descriptor by the abstract query key in the interface descriptor.

122: SET_EXISTENCE_CHECKING_NOT_UNDERSTOOD

Cause: The interface descriptor `parent` does not have at least one abstract query key defined. The string given to the method `setExistenceChecking(String token)` is not understood.

Action: Contact Oracle Support Services.

125: VALUE HOLDER INSTANTIATION MISMATCH

Cause: The `DatabaseMapping` for the attribute `getAttributeName()` uses indirection and must be initialized to a new value holder.

Action: Ensure that the mapping uses indirection and that the attribute is initialized to a new value holder.

126: NO_SUB_CLASS_MATCH

Cause: No subclass matches this class `theClass` when inheritance is in aggregate relationship mapping.

Action: Verify the subclass and the relationship mapping.

127: RETURN_AND_MAPPING_WITH_INDIIRECTION_MISMATCH

Cause: The return type of the method used to get the attribute `getAttributeName()` of `DatabaseMapping` is not declared as type `ValueHolderInterface`, but the mapping is using indirection.

Action: Verify that the method used to get the attribute named `getAttributeName()` of `DatabaseMapping` returns a value holder, or change the mapping so it does not use indirection.

128: RETURN_AND_MAPPING_WITHOUT_INDIIRECTION_MISMATCH

Cause: The return type of the method used to get the attribute `getAttributeName()` of `DatabaseMapping` is declared as type `ValueHolderInterface`, but the mapping is not using indirection.

Action: Ensure that the mapping is using indirection, or change the return type from value holder.

129: PARAMETER_AND_MAPPING_WITH_INDIRECTION_MISMATCH

Cause: The return type of the method used to set the attribute `getAttributeName()` of `DatabaseMapping` is not declared as type `ValueHolderInterface`, but the mapping is using indirection.

Action: Ensure that the set method parameter is declared as a valueholder, or change the mapping so it does not use indirection.

130: PARAMETER_AND_MAPPING_WITHOUT_INDIRECTION_MISMATCH

Cause: The return type of the method used to set the attribute `getAttributeName()` of `DatabaseMapping` is declared as type `ValueHolderInterface`, but the mapping is not using indirection.

Action: Ensure that the mapping is changed to use indirection, or that the method parameter is not declared as a value holder.

131: GET_METHOD_RETURN_TYPE_NOT_VALID

Cause: The return type of the method used to get the attribute `getAttributeName()` of `DatabaseMapping` is not declared as type `Vector` (or a type that implements the `Map` or `Collection` interface if using Java 2).

Action: Declare the return type of the method used to get the attribute `getAttributeName()` of `DatabaseMapping` as type `Vector` (or a type that implements the `map` or `collection` interface if using Java 2).

133: SET_METHOD_PARAMETER_TYPE_NOT_VALID

Cause: The parameter type of the method used to set the attribute `getAttributeName()` of `DatabaseMapping` is not declared as type `Vector` (or a type that implements the `map` or `collection` interface, if using Java 2).

Action: Declare the parameter type of the method used to set the attribute `getAttributeName()` of `DatabaseMapping` as type `Vector` (or a type that implements the `Map` or `Collection` interface, if using Java 2).

135: ILLEGAL_TABLE_NAME_IN_MULTIPLE_TABLE_FOREIGN_KEY

Cause: The table in the multiple table foreign key relationship refers to an unknown table.

Action: Verify the table name.

138: ATTRIBUTE_AND_MAPPING_WITH_TRANSPARENT_INDIRECTION_MISMATCH

Cause: The attribute `getAttributeName()` of `DatabaseMapping` is not declared as a supertype of `validTypeName`, but the mapping is using transparent indirection.

Action: Verify the attribute's type and the mapping setup.

139: RETURN_AND_MAPPING_WITH_TRANSPARENT_INDIRECTION_MISMATCH

Cause: The return type of the method used to get the attribute `getAttributeName()` of `DatabaseMapping` is not declared as a super-type of `validTypeName`, but the mapping is using transparent indirection.

Action: Verify the attribute's type and the mapping setup.

140: PARAMETER_AND_MAPPING_WITH_TRANSPARENT_INDIRECTION_MISMATCH

Cause: The parameter type of the method used to set the attribute `getAttributeName()` of `DatabaseMapping` is not declared as a supertype of `validTypeName`, but the mapping is using transparent indirection.

Action: Verify the attribute's type and the mapping setup.

141: FIELD_IS_NOT_PRESENT_IN_DATABASE

Cause: The field `fieldname` is not present in the table `tableName` in the database.

Action: Verify the field name for the attribute.

142: TABLE_IS_NOT_PRESENT_IN_DATABASE

Cause: The table whose name is provided by the `Descriptor` method `getTableName` is not present in the database.

Action: Verify the table name for the descriptor.

143: MULTIPLE_TABLE_INSERT_ORDER_MISMATCH

Cause: The multiple table insert order vector specified the `Descriptor` method `getMultipleTableInsertOrder` has fewer or more tables than are specified in the `Descriptor` method `getTables`. All the tables must be included in the insert order vector.

Action: Ensure that all table names for the descriptor are present and that there are no extra tables.

144: INVALID_USE_OF_TRANSPARENT_INDIRECTION

Cause: Transparent indirection is being used with a mapping other than a `CollectionMapping`.

Action: Verify the mapping. It must be a collection mapping.

145: MISSING_INDIRECT_CONTAINER_CONSTRUCTOR

Cause: The indirect container class does not implement the constructor.

Action: Implement the constructor for the container.

146: COULD_NOT_INSTANTIATE_INDIRECT_CONTAINER_CLASS

Cause: `TopLink` is unable to instantiate the indirect container class using the constructor.

Action: Validate the constructor for the indirect container class.

147: INVALID_CONTAINER_POLICY

Cause: You have used a container policy with an incompatible version of the JDK. This container policy must only be used with JDK 1.3.1 or later.

Action: Validate the container policy being used.

148: INVALID_CONTAINER_POLICY_WITH_TRANSPARENT_INDIRECTION

Cause: The container policy is incompatible with transparent indirection.

Action: Change the container policy to be compatible with transparent indirection, or do not use transparent indirection.

149: INVALID_USE_OF_NO_INDIRECTION

Cause: `NoIndirectionPolicy` objects should not receive this message.

Action: Contact Oracle Support Services.

150: INDIRECT_CONTAINER_INSTANTIATION_MISMATCH

Cause: The mapping for the attribute `mapping.getAttributeName()` uses transparent indirection and must be initialized to an appropriate container.

Action: Initialize the mapping to an appropriate container.

151: INVALID_MAPPING_OPERATION

Cause: An invalid mapping operation has been used.

Action: See the documentation for valid mapping operations.

152: INVALID_INDIRECTION_POLICY_OPERATION

Cause: An invalid indirection policy operation has been used.

Action: See the documentation for valid indirection policy operations.

153: REFERENCE_DESCRIPTOR_IS_NOT_AGGREGATECOLLECTION

Cause: The reference descriptor for `className` is not set to an aggregate collection descriptor.

Action: Set the reference descriptor to an aggregate collection descriptor.

154: INVALID_INDIRECTION_CONTAINER_CLASS

Cause: An invalid indirection container class has been used.

Action: Verify the container class.

155: MISSING_FOREIGN_KEY_TRANSLATION

Cause: The mapping does not include a foreign key field linked to the primary key field.

Action: Link the foreign key to the appropriate primary key.

156: STRUCTURE_NAME_NOT_SET_IN_MAPPING

Cause: The structure name is not set.

Action: Set the structure name appropriately.

157: NORMAL_DESCRIPTOR_DO_NOT_SUPPORT_NON_RELATIONAL_EXTENSIONS

Cause: Relational descriptors do not support non-relational extensions.

Action: Contact Oracle Support Services.

158: PARENT_CLASS_IS_SELF

Cause: The descriptor's parent class has been set to itself.

Action: Contact Oracle Support Services.

159: PROXY_INDIRECTION_NOT_AVAILABLE

Cause: An attempt to use proxy indirection has been made, but JDK 1.3.1 or later is not being used.

Action: Use JDK 1.3.1 or later.

160: INVALID_ATTRIBUTE_TYPE_FOR_PROXY_INDIRECTION

Cause: The attribute was not specified in the list of interfaces given to use proxy indirection.

Action: Verify the attribute.

161: INVALID_GET_RETURN_TYPE_FOR_PROXY_INDIRECTION

Cause: The return type for the indirection policy is invalid for the indirection policy.

Action: Ensure that the parameter type of the getter method is correct for the indirection policy.

162: INVALID_SET_PARAMETER_TYPE_FOR_PROXY_INDIRECTION

Cause: The parameter for the setter method is incorrect for the indirection type.

Action: Ensure that the parameter type of the setter method is correct for the indirection policy.

163: INCORRECT_COLLECTION_POLICY

Cause: The container policy is invalid for the collection type.

Action: Ensure that the container policy is correct for the collection type.

164: INVALID_AMENDMENT_METHOD

Cause: The amendment method that is provided is invalid, not public, or cannot be found.

Action: Ensure that the amendment method is public, static, returns void, and has a single argument: `Descriptor`.

165: ERROR_OCCURRED_IN_AMENDMENT_METHOD

Cause: The specified amendment method threw an exception.

Action: Examine the returned exception for further information.

166: VARIABLE_ONE_TO_ONE_MAPPING_IS_NOT_DEFINED

Cause: There is no mapping for the attribute.

Action: Validate the mapping and attribute.

167: NO_CONSTRUCTOR_INDIRECT_COLLECTION_CLASS

Cause: A valid constructor was not found for the indirection container class.

Action: Add a default constructor or a constructor with a `ValueHolderInterface` in the container class.

168: TARGET_INVOCATION_WHILE_CONSTRUCTOR_INSTANTIATION

Cause: The constructor is missing.

Action: Create the required constructor.

169: TARGET_INVOCATION_WHILE_CONSTRUCTOR_INSTANTIATION_OF_FACTORY

Cause: The constructor is missing.

Action: Create the required constructor.

170: ILLEGAL_ACCESS_WHILE_CONSTRUCTOR_INSTANTIATION_OF_FACTORY

Cause: Permissions do not allow access to the constructor.

Action: Adjust the Java security permissions to permit access to the constructor.

171: INSTANTIATION_WHILE_CONSTRUCTOR_INSTANTIATION_OF_FACTORY

Cause: An instantiation failed inside the associated constructor.

Action: Determine which objects are being instantiated, and verify that all are instantiated properly.

172: NO_SUCH_METHOD_WHILE_CONSTRUCTOR_INSTANTIATION_OF_FACTORY

Cause: A method call from inside the constructor is invalid because this method does not exist.

Action: Ensure that the factory has a default constructor for the called method.

173: NULL_POINTER_WHILE_CONSTRUCTOR_INSTANTIATION_OF_FACTORY

Cause: A method on a null object was called from inside a constructor. The factory constructor was inaccessible.

Action: Examine the internal exception and take the appropriate action.

174: ILLEGAL_ACCESS_WHILE_METHOD_INSTANTIATION_OF_FACTORY

Cause: A method was called on an object from inside a factory instantiation, and Java has determined this method to be invalid.

Action: Determine why the method is invalid, and replace the method with a valid one.

175: TARGET_INVOCATION_WHILE_METHOD_INSTANTIATION_OF_FACTORY

Cause: A problem was encountered creating factory using creation method. The creation method triggered an exception.

Action: Examine the exception and take the corresponding action.

176: NULL_POINTER_WHILE_METHOD_INSTANTIATION_OF_FACTORY

Cause: A method called to instantiate a factory threw a null pointer exception. The creation method is not accessible.

Action: Do not use that method to instantiate a factory.

177: NO_MAPPING_FOR_ATTRIBUTE_NAME

Cause: Mapping is missing for the attribute.

Action: The attribute must be mapped.

178: NO_MAPPING_FOR_ATTRIBUTE_NAME_IN_ENTITY_BEAN

Cause: Cannot find mapping for attribute in entity bean.

Action: The attribute must be mapped.

179: UNSUPPORTED_TYPE_FOR_BIDIRECTIONAL_RELATIONSHIP_MAINTENANCE

Cause: The attribute uses bidirectional relationship maintenance, but has ContainerPolicy, which does not support it.

Action: The attribute must be mapped with a different collection type.

180: REFERENCE_DESCRIPTOR_CANNOT_BEAggregate

Cause: No string is defined in the DescriptorExceptionResource.java file.

Action: The reference descriptor for a relationship mapping (except AggregateCollectionMapping) cannot be aggregate. You must change the reference descriptor to non-aggregate.

181: ATTRIBUTE_TRANSFORMER_CLASS_NOT_FOUND

Cause: The AttributeTransformer class cannot be found.

Action: Ensure that the `AttributeTransformer` class exists and is on the classpath.

182: FIELD_TRANSFORMER_CLASS_NOT_FOUND

Cause: The `FieldTransformer` class cannot be found.

Action: Ensure that the `FieldTransformer` class exists and is on the classpath.

183: ATTRIBUTE_TRANSFORMER_CLASS_INVALID

Cause: The class cannot be used as an `AttributeTransformer`.

Action: Examine the internal exception stack trace and make the appropriate correction.

184: FIELD_TRANSFORMER_CLASS_INVALID

Cause: The class cannot be used as a `FieldTransformer`.

Action: Do not use the class as a `FieldTransformer`.

185: RETURNING_POLICY_FIELD_TYPE_CONFLICT

Cause: `ReturningPolicy` contains field with two different types.

Action: The field was added to `ReturningPolicy` twice with different types. The field must be added to `ReturningPolicy` once. You must remove excessive `addFieldForInsert` and/or `addInsertField` calls.

186: RETURNING_POLICY_FIELD_INSERT_CONFLICT

Cause: `ReturningPolicy` contains field that has been added twice using `addInsertField` and `addInsertFieldReturnOnly`.

Action: A field must be added to `ReturningPolicy` only once. You must remove excessive `addField` calls.

187: RETURNING_POLICY_AND_DESCRIPTOR_FIELD_TYPE_CONFLICT

Cause: `ReturningPolicy` contains field with type `[[1]]`, but the same field in descriptor has type `[[2]]`.

Action: Specify field type in `addField` method only in the event that it cannot be obtained from the descriptor.

188: RETURNING_POLICY_UNMAPPED_FIELD_TYPE_NOT_SET

Cause: `ReturningPolicy` contains unmapped field that requires type.

Action: You must specify field type in the `addField` method.

189: RETURNING_POLICY_MAPPED_FIELD_TYPE_NOT_SET

Cause: `ReturningPolicy` contains mapped field that requires type.

Action: You must specify field type in the `addField` method.

190: RETURNING_POLICY_MAPPING_NOT_SUPPORTED

Cause: `ReturningPolicy` contains a field that is mapped with unsupported mapping.

Action: You cannot use `ReturningPolicy` with this field. Do not add it to `ReturningPolicy`.

191: RETURNING_POLICY_FIELD_NOT_SUPPORTED

Cause: `ReturningPolicy` contains a field that is not supported. Field is either sequence field, class type indicator, or used for locking.

Action: You cannot use `ReturningPolicy` with this field. Do not add it to `ReturningPolicy`.

192: CUSTOM_QUERY_AND_RETURNING_POLICY_CONFLICT

Cause: `ReturningPolicy` contains a field, but custom doesn't output it.

Action: Update the custom query so that it outputs a value for this field.

193: NO_CUSTOM_QUERY_FOR_RETURNING_POLICY

Cause: There is no custom set, but `ReturningPolicy` contains one or more fields to be returned and doesn't support generating call with return.

Action: Specify a custom `InsertObjectQuery` or `UpdateObjectQuery` through `DescriptorQueryManager` `setInsertQuery`, `setInsertCall`, `setUpdateQuery`, or `setUpdateCall` that outputs values for fields added to `ReturningPolicy`.

194: CLASS_EXTRACTION_METHOD_MUST_BE_STATIC

Cause: The class extraction method must be a static method on the descriptor's class.

Action: Make the class extraction method a static method on the descriptor's class.

195: ISOLATED_DESCRIPTOR_REFERENCED_BY_SHARED_DESCRIPTOR

Cause: The shared class must not reference the isolated class.

Action: Ensure that the shared class does not reference the isolated class.

196: UPDATE_ALL_FIELDS_NOT_SET

Cause: `UpdateAllFields` has not been set or has been set to `false`. When using `setForceUpdate(true)` of `CMPPolicy` you must also call `setUpdateAllFields(true)` of `CMPPolicy`.

Action: Ensure that `updateAllFields` is set to `true` if `forceUpdate` is `true`.

197: INVALID_MAPPING_TYPE

Cause: The mapping is not the appropriate type for this descriptor.

Action: The mapping type has to map the descriptor type, e.g. relational mapping for relational descriptor, EIS mapping for EIS descriptor, and XML mapping for XML descriptor.

198: NEED_TO_IMPLEMENT_CHANGETRACKER

Cause: In order to use `ObjectChangeTrackingPolicy` or `AttributeChangeTrackingPolicy`, the object has to implement the `ChangeTracker` interface.

Action: Ensure that the object implements `ChangeTrackerInterface` in order to use `ObjectChangeTrackingPolicy` or `AttributeChangeTrackingPolicy`.

199: NEED_TO_IMPLEMENT_FETCHGROUPTRACKER

Cause: In order to use a fetch group, the domain class has to implement the `FetchGroupTracker` interface.

Action: Ensure that the domain class implements the `FetchGroupTracker` interface in order to use the fetch group.

200: ATTEMPT_TO_REGISTER_DEAD_INDIRECTION

Cause: Attempt to register an object with dead indirection as a new object. Possibly the object was deleted or removed from the cache during a merge of a serialized clone or did not exist in the cache at the time of the merge. This is a concurrency violation.

Action: Ensure that the object exists in the cache before attempting to merge a deserialized version into the cache. Consider a locking strategy. For more information, see "[Merging Changes in Working Copy Clones](#)" on page 102-12 and "[Indirection, Serialization, and Detachment](#)" on page 33-9.

Concurrency Exceptions (2001 – 2006)

`ConcurrencyException` is a development exception that is raised when a Java concurrency violation occurs. Only when a running thread is interrupted, causing the JVM to throw an `InterruptedException`, is an internal exception information displayed with the error message, as shown in [Example 13–2](#).

Format

EXCEPTION [TOPLINK - error code]: Exception name
 EXCEPTION DESCRIPTION: Message
 INTERNAL EXCEPTION: Message

Example 13–2 Concurrency Exception

EXCEPTION [TOPLINK - 2004]: oracle.toplink.exceptions.ConcurrencyException
 EXCEPTION DESCRIPTION: Signal attempted before wait on concurrency manager.
 This usually means that an attempt was made to commit or roll back a transaction before being started, or rolled back twice.

2001: WAIT_WAS_INTERRUPTED

Cause: In a multi threaded environment, one of the waiting threads was interrupted.

Action: Such exceptions are dependent on the application.

2002: WAIT_FAILURE_SERVER

Cause: A request for a connection from the connection pool has been forced to wait, and that wait has been interrupted.

Action: Such exceptions are dependent on the application.

2003: WAIT_FAILURE_CLIENT

Cause: A request for a connection from the connection pool has been forced to wait, and that wait has been interrupted.

Action: Such exceptions are dependent on the application.

2004: SIGNAL_ATTEMPTED_BEFORE_WAIT

Cause: A signal was attempted before a wait on concurrency manager. This usually means that an attempt was made to commit or roll back a transaction before it was started, or to rollback a transaction twice.

Action: Verify transactions in the application.

2005: WAIT_FAILURE_SEQ_DATABASE_SESSION

Cause: An `InterruptedException` was raised while `DatabaseSession` sequencing waited for a separate connection to become available.

Action: Examine concurrency issues involving object creation with your `DatabaseSession`.

2006: SEQUENCING_MULTITHREAD_THRU_CONNECTION

Cause: Several threads attempted to concurrently obtain sequence objects from the same `DatabaseSession` or `ClientSession`.

Action: Avoid concurrent writing through the same `DatabaseSession` or `ClientSession`.

Conversion Exceptions (3001– 3008)

`ConversionException` is a development exception that is raised when a conversion error occurs by an incompatible type conversion. The message that is returned indicates which type cast caused the exception, as shown in [Example 13–3](#).

Format

EXCEPTION [TOPLINK - error code]: Exception name

EXCEPTION DESCRIPTION: Message

INTERNAL EXCEPTION: Message

Example 13–3 Conversion Exception

EXCEPTION [TOPLINK - 3006]: oracle.toplink.exceptions.ConversionException

EXCEPTION DESCRIPTION: object must be of even length to be converted to a
ByteArray

3001: COULD_NOT_BE_CONVERTED

Cause: The object object of class `objectClass` cannot be converted to `javaClass`. The object cannot be converted to a given type.

Action: Ensure that the object being converted is of the right type.

3002: COULD_NOT_BE_CONVERTED_EXTENDED

Cause: The object object of class `objectClass` from mapping `mappingType` cannot be converted to `javaClass`. The object cannot be converted to a given type.

Action: Ensure that the object being converted is of the right type.

3003: INCORRECT_DATE_FORMAT

Cause: The date in `dateString` is in an incorrect format. The expected format is YYYY-MM-DD.

Action: Verify the date format.

3004: INCORRECT_TIME_FORMAT

Cause: The time in `timeString` is in an incorrect format. The expected format is HH:MM:SS.

Action: Verify the time format.

3005: INCORRECT_TIMESTAMP_FORMAT

Cause: The timestamp `timestampString` is in an incorrect format. The expected format is YYYY-MM-DD HH:MM:SS.NNNNNNNN.

Action: Verify the timestamp format.

3006: COULD_NOT_CONVERT_TO_BYTE_ARRAY

Cause: The `String` object must be of even length to be converted to a `ByteArray`. This object cannot be converted to a `ByteArray`.

Action: Verify the object being converted.

3007: COULD_NOT_BE_CONVERTED_TO_CLASS

Cause: The object object of class `objectClass` cannot be converted to `javaClass`. The class `javaClass` is not on the classpath.

Action: Ensure that the class `JavaClass` is on the classpath.

3008: INCORRECT_DATE_TIME_FORMAT

Cause: Incorrect date-time format object. The expected format is `YYYY-MM-DD'T'HH:MM:SS`.

Action: Ensure that the date-time object is in the expected format of `YYYY-MM-DD'T'HH:MM:SS`.

Database Exceptions (4002 – 4018)

`DatabaseException` is a run-time exception that is raised when data read from the database, or the data that is to be written to the database, is incorrect. The exception may also act as a wrapper for `SQLException`. If this is the case, the message contains a reference to the error code and error message, as shown in [Example 13–4](#).

This exception can occur on any database operation. If an execution of a SQL script is involved in a database operation causing `DatabaseException`, the exception's message, accessible through the `getMessage` method, contains the SQL that caused this exception.

This exception includes internal exception and error code information when the exception is wrapping a `SQLException`.

Format

```
EXCEPTION [TOPLINK - error code]: Exception name
EXCEPTION DESCRIPTION: Message
INTERNAL EXCEPTION: Message
ERROR CODE: Error code
```

Example 13–4 Database Exception

```
EXCEPTION [TOPLINK - 4002]: oracle.toplink.exceptions.DatabaseException
EXCEPTION DESCRIPTION: java.sql.SQLException: [INTERSOLV][ODBC dBase driver]
Incompatible datatypes in expression: >
INTERNAL EXCEPTION: java.sql.SQLException: [INTERSOLV][ODBC dBase driver]
Incompatible datatypes in expression: >
ERROR CODE: 3924
```

4002: SQL_EXCEPTION

Cause: A SQL exception was encountered, raised by the underlying JDBC bridge. `TopLink` wraps only that exception.

Action: Inspect the internal exception that was raised.

4003: CONFIGURATION_ERROR_CLASS_NOT_FOUND

Cause: The driver class name was not found.

Action: Verify the class name given in `JDBCLogin`.

4005: DATABASE_ACCESSOR_NOT_CONNECTED

Cause: The session is not connected to the database while attempting to read or write on the database.

Action: An application may have to log in again because the connection to the database might have been lost.

4006: ERROR_READING_BLOB_DATA

Cause: An error occurred reading BLOB data from the database. There are two possibilities for this exception: either the BLOB data was not read properly from the result set or TopLink cannot process the BLOB data using `ByteArrayOutputStream`.

Action: Verify whether the underlying driver supports BLOBs properly. If it does, then report this problem to Oracle Support Services.

4007: COULD_NOT_CONVERT_OBJECT_TYPE

Cause: Cannot convert object type on internal error `.java.sql.Types = type`. The object from the result set cannot be converted to the type that was returned from the metadata information.

Action: Verify whether the underlying driver supports the conversion type properly. If it does, then report this problem to Oracle Support Services.

4008: LOGOUT_WHILE_TRANSACTION_IN_PROGRESS

Cause: An attempt has been made to log out while the transaction is still in progress. You cannot log out while a transaction is in progress.

Action: Wait until the transaction is finished.

4009: SEQUENCE_TABLE_INFORMATION_NOT_COMPLETE

Cause: The sequence information given to TopLink is not sufficiently complete to get the set of sequence numbers from the database. This usually happens on native sequencing (see [Understanding Sequencing in Relational Projects](#)) on an Oracle database.

Action: Verify the data provided, especially the sequence name provided in TopLink.

4011: ERROR_PREALLOCATING_SEQUENCE_NUMBERS

Cause: An error occurred preallocating sequence numbers on the database; the sequence table information is not complete.

Action: Ensure the sequence table was properly created on the database.

4014: CANNOT_REGISTER_SYNCHRONIZATIONLISTENER_FOR_UNITOFWORK

Cause: TopLink cannot register the synchronization listener: *underlying_exception_string*. When the TopLink session is configured with an `ExternalTransactionController`, any unit of work requested by a client must operate within the context of a JTS external global transaction. When a unit of work is created and the external global transaction is not in existence, or if the system cannot acquire a reference to it, this error is reported.

Action: Verify that a JTS transaction is in progress before acquiring the unit of work.

4015 SYNCHRONIZED_UNITOFWORK_DOES_NOT_SUPPORT_COMMITANDRESUME

Cause: A synchronized `UnitOfWork` does not support the `commitAndResume` operation. When the `TopLink` session is configured with an `ExternalTransactionController`, any unit of work requested by a client must operate within the context of a JTS external global transaction (see [4014: CANNOT_REGISTER_SYNCHRONIZATIONLISTENER_FOR_UNITOFWORK](#)). The JTS specification does not support the concept of check pointing a transaction—that is, committing the work performed and then continuing to work within the same transaction context. JTS does not support nested transactions, either. As a result, if a client code invokes `commitAndResume()` on a synchronized unit of work, this error is reported.

Action: None required.

4016: CONFIGURATION_ERROR_NEW_INSTANCE_INSTANTIATION_EXCEPTION

Cause: A configuration error occurred when `TopLink` attempted to instantiate the given driver class. `TopLink` cannot instantiate the driver.

Action: Check the driver.

4017: CONFIGURATION_ERROR_NEW_INSTANCE_ILLEGAL_ACCESS_EXCEPTION

Cause: A configuration error occurred when `TopLink` attempted to instantiate the given driver class. `TopLink` cannot instantiate the driver.

Action: Check the driver.

4018: TRANSACTION_MANAGER_NOT_SET_FOR_JTS_DRIVER

Cause: The transaction manager has not been set for the `JTSSynchronizationListener`.

Action: Set a transaction manager for the `JTSSynchronizationListener`.

Optimistic Lock Exceptions (5001 – 5008)

`OptimisticLockException` is a run-time exception that is raised when the row on the database that matches the desired object is missing or when the value on the database does not match the registered number. It is used in conjunction with the optimistic locking feature. This applies only on an update or delete operation, as shown in [Example 13–5](#).

For more information about optimistic locking, see the section on [Optimistic Locking in a Stateless Environment](#) in [Chapter 2, "Understanding TopLink Application Development"](#). These exceptions should be handled in a try-catch block.

Format

```
EXCEPTION [TOPLINK - error code]: Exception Name
EXCEPTION DESCRIPTION: Message
```

Example 13–5 Optimistic Lock Exception

```
EXCEPTION [TOPLINK - 5003]: oracle.toplink.exceptions.OptimisticLockException
EXCEPTION DESCRIPTION: The object, object.toString() cannot be deleted because it
has changed or been deleted since it was last read.
```

5001: NO_VERSION_NUMBER_WHEN_DELETING

Cause: An attempt was made to delete the object `object`, but it has no version number in the identity map. This object either was never read or has already been deleted.

Action: Use SQL logging to determine the reason for the exception. The last delete operation shows the object being deleted when the exception was raised.

5003: OBJECT_CHANGED_SINCE_LAST_READ_WHEN_DELETING

Cause: The object state has changed in the database. The object `object` cannot be deleted because it has changed or been deleted since it was last read. This usually means that the row in the table was changed by some other application.

Action: Refresh the object, which updates it with the new data from the database.

5004: NO_VERSION_NUMBER_WHEN_UPDATING

Cause: An attempt has been made to update the object `object` but it has no version number in the identity map. It may not have been read before being updated, or it has been deleted.

Action: Use SQL logging to determine the reason for the exception. The last update operation shows the object being updated when the exception was raised.

5006: OBJECT_CHANGED_SINCE_LAST_READ_WHEN_UPDATING

Cause: The object state has changed in the database. The object `object` cannot be updated because it has changed or been deleted since it was last read. This usually means that the row in the table was changed by some other application.

Action: Refresh the object, which updates it with the new data from the database.

5007: MUST_HAVE_MAPPING_WHEN_IN_OBJECT

Cause: The object `aClass` must have a nonread-only mapping corresponding to the version lock field. The mapping, which is needed when the lock value is stored in the domain object rather than in a cache, was not defined for the locking field.

Action: Define a mapping for the field.

5008: NEED_TO_MAP_JAVA_SQL_TIMESTAMP

Cause: A write lock value that is stored in a domain object is not an instance of `java.sql.Timestamp`.

Action: Change the value of the attribute to be an instance of `java.sql.Timestamp`.

Query Exceptions (6001 – 6121)

`QueryException` is a development exception that is raised when insufficient information has been provided to the query. If possible, the message indicates the query that caused the exception. A query is optional and is displayed if `TopLink` is able to determine the query that caused this exception, as shown in [Example 13–6](#).

Format

EXCEPTION [TOPLINK - error code]: Exception name
EXCEPTION DESCRIPTION: Message
QUERY:

Example 13–6 Query Exception

EXCEPTION [TOPLINK - 6026]: oracle.toplink.exceptions.QueryException

EXCEPTION DESCRIPTION: The query is not defined. When executing a query on the session, the parameter that takes the query is null.

6001: ADDITIONAL_SIZE_QUERY_NOT_SPECIFIED

Cause: Cursored SQL queries must provide an additional query to retrieve the size of the result set. Failure to include the additional query causes this exception.

Action: Specify a size query.

6002: AGGREGATE_OBJECT_CANNOT_BE_DELETED

Cause: Aggregated objects cannot be written or deleted independent of their owners. No identity is maintained on such objects.

Action: Do not try to delete aggregate objects directly.

6003: ARGUMENT_SIZE_MISMATCH_IN_QUERY_AND_QUERY_DEFINITION

Cause: The number of arguments provided to the query for execution does not match the number of arguments provided with the query definition.

Action: Check the query and the query execution.

6004: BACKUP_CLONE_IS_ORIGINAL_FROM_PARENT

Cause: The object `clone` of class `clone.getClass()` with identity hash code `(System.identityHashCode()) (System.identityHashCode(clone))` is not from this unit of work space but from the parent session. The object was never registered in this unit of work but read from the parent session and related to an object registered in the unit of work.

Action: Verify that you are correctly registering your objects. If you are still having problems, use the `validateObjectSpace` method of the `UnitOfWork` to help debug where the error occurred.

6005: BACKUP_CLONE_IS_ORIGINAL_FROM_SELF

Cause: The object `clone` of class `clone.getClass()` with identity hash code `(System.identityHashCode()) (System.identityHashCode(clone))` is the original to a registered new object. Because the unit of work clones new objects that are registered, ensure that an object is registered before it is referenced by another object. If you do not want the new object to be cloned, use the `registerNewObject(Object)` method of the `UnitOfWork`.

Action: Verify that you are correctly registering your objects. If you are still having problems, use the `validateObjectSpace` method of the `UnitOfWork` to help debug where the error occurred.

6006: BATCH_READING_NOT_SUPPORTED

Cause: This mapping does not support batch reading. The optimization of batch reading all the target rows is not supported for the mapping.

Action: The problem is a TopLink development problem, and the user should never encounter this error code unless the mapping is a new custom mapping. Contact Oracle Support Services.

6007: DESCRIPTOR_IS_MISSING

Cause: The descriptor for `reference Class` is missing. The descriptor related to the class or the object is not found in the session.

Action: Verify whether or not the related descriptor was added to the session, and whether or not the query is performed on the right object or class.

6008: DESCRIPTOR_IS_MISSING_FOR_NAMED_QUERY

Cause: The descriptor `domainClassName` for the query named `queryName` is missing. The descriptor where named query is defined is not added to the session.

Action: Verify whether or not the related descriptor was added to the session, and whether or not the query is performed on the right class.

6013: INCORRECT_SIZE_QUERY_FOR_CURSOR_STREAM

Cause: The size query given on the queries returning cursor streams is not correct. The execution of the size query did not return any size.

Action: If the cursor stream query was a custom query, then check the size of the query that was specified, or report this problem to Oracle Support Services.

6014: INVALID_QUERY

Cause: Objects cannot be written in a unit of work using modify queries. They must be registered.

Action: Objects are registered in the unit of work, and during commit, the unit of work performs the required changes to the database.

6015: INVALID_QUERY_KEY_IN_EXPRESSION

Cause: The query key `key` does not exist. Usually this happens because of a misspelled query key.

Action: Check the query key that was specified in the expression and verify that a query key was added to the descriptor.

6016: INVALID_QUERY_ON_SERVER_SESSION

Cause: Objects and the database cannot be changed through the server session: all changes must be performed through a client session's unit of work. The objects cannot be changed on the server session by modifying queries. Objects are changed in the client sessions that are acquired from this server session.

Action: Use the client session's unit of work to change the object.

6020: NO_CONCRETE_CLASS_INDICATED

Cause: No concrete class is indicated for the type in this row. The type indicator read from the database row has no entry in the type indicator hash table or if class extraction method was used, it did not return any concrete class type. The exception is raised when subclasses are being read.

Action: Check the class extraction method, if specified, or check the descriptor to verify all the type indicator values were specified.

6021: NO_CURSOR_SUPPORT

Cause: No cursor support is provided for abstract class multiple table descriptors using expressions.

Action: Consider using custom SQL or multiple queries.

6023: OBJECT_TO_INSERT_IS_EMPTY

Cause: There are no fields to be inserted into the table. The fields to insert into the table `table`, are empty.

Action: Define at least one mapping for this table.

6024: OBJECT_TO_MODIFY_NOT_SPECIFIED

Cause: An object to modify is required for a modify query.

Action: Verify that the query contains an object before executing.

6026: QUERY_NOT_DEFINED

Cause: The query is not defined. When executing a query on the session, the parameter that takes the query is null.

Action: Verify that the query is passed properly.

6027: QUERY_SENT_TO_INACTIVE_UNIT_OF_WORK

Cause: The unit of work has been released and is now inactive.

Action: The unit of work, once released, cannot be reused unless the `commitAndResume` method is called.

6028: READ_BEYOND_QUERY

Cause: An attempt has been made to read from the cursor streams beyond its limits (beyond the end of the stream).

Action: Ensure that the stream is checked for an end of stream condition before attempting to retrieve more objects.

6029: REFERENCE_CLASS_MISSING

Cause: The reference class in the query is not specified. A reference class must be provided.

Action: Check the query.

6030: REFRESH_NOT_POSSIBLE_WITHOUT_CACHE

Cause: Refresh is not possible if caching is not set. The read queries that skip the cache to read objects cannot be used to refresh the objects. Refreshing is not possible without identity.

Action: Check the query.

6031: SIZE_ONLY_SUPPORTED_ON_EXPRESSION_QUERIES

Cause: TopLink did not find a size query. Size is supported only on expression queries unless a size query is given.

Action: The cursor streams on a custom query should also define a size query.

6032: SQL_STATEMENT_NOT_SET_PROPERLY

Cause: The SQL statement has not been properly set. The user should never encounter this error code unless queries have been customized.

Action: Contact Oracle Support Services.

6034: INVALID_QUERY_ITEM

Cause: TopLink is unable to validate a query item expression.

Action: Validate the expression being used.

6041: SELECTION_OBJECT_CANNOT_BE_NULL

Cause: The selection object that was passed to a read object or refresh was null.

Action: Check `setSelectionObject` method on read query.

6042: UNNAMED_QUERY_ON_SESSION_BROKER

Cause: Data read and data modify queries are being executed without the session name. Only object-level queries can be directly executed by the session broker, unless the query is named.

Action: Specify the session name.

6043: REPORT_RESULT_WITHOUT_PKS

Cause: ReportQuery without primary keys cannot read the objects. The report query result that was returned is without primary key values. An object from the result can be created only if primary keys were also read.

Action: See the documentation about retrievePrimaryKeys method on report query.

6044: NULL_PRIMARY_KEY_IN_BUILDING_OBJECT

Cause: The primary key that was read from the row databaseRow during the execution of the query was detected to be null; primary keys must not contain null.

Action: Check the query and the table on the database.

6045: NO_DESCRIPTOR_FOR_SUBCLASS

Cause: The subclass has no descriptor defined for it.

Action: Ensure the descriptor was added to the session, or check class extraction method.

6046: CANNOT_DELETE_READ_ONLY_OBJECT

Cause: The class you are attempting to delete is a read-only class.

Action: Contact Oracle Support Services.

6047: INVALID_OPERATOR

Cause: The operator data used in the expression is not valid.

Action: Check ExpressionOperator class to see a list of all the operators that are supported.

6048: ILLEGAL_USE_OF_GETFIELD

Cause: This is an invalid use of getField method's data in the expression. This is a TopLink development exception that users should not encounter.

Action: Report this problem to Oracle Support Services.

6049: ILLEGAL_USE_OF_GETTABLE

Cause: This is an invalid use of getTable method's data in the expression. This is a TopLink development exception that users should not encounter.

Action: Report this problem to Oracle Support Services.

6050: REPORT_QUERY_RESULT_SIZE_MISMATCH

Cause: The number of attributes requested does not match the attributes returned from the database in report query. This can happen as a result of a custom query on the report query.

Action: Check the custom query to ensure it is specified, or report the problem to Oracle Support Services.

6051: CANNOT_CACHE_PARTIAL_OBJECT

Cause: Partial Objects are never put in the cache. Partial object queries are not allowed to maintain the cache or to be edited. Use the dontMaintainCache method.

Action: Call the dontMaintainCache method before executing the query.

6052: OUTER_JOIN_ONLY_VALID_FOR_ONE_TO_ONE

Cause: An outer join (getAllowingNull method) is valid only for one-to-one mappings and cannot be used for the mapping.

Action: Do not attempt to use the `getAllowingNull` method for mappings other than one-to-one.

6054: CANNOT_ADD_TO_CONTAINER

Cause: TopLink is unable to add an object to a container class using `policy`. This is TopLink development exception, and the user should never encounter this problem unless a custom container policy has been written.

Action: Contact Oracle Support Services.

6055: METHOD_INVOCATION_FAILED

Cause: The invocation of a method on the object `anObject` threw a Java reflection exception while accessing the method.

Action: Inspect the internal exception, and refer to the Java documentation.

6056: CANNOT_CREATE_CLONE

Cause: Cannot create a clone of the object `anObject` using `policy`. This is a TopLink development exception, and the user should never encounter this problem unless a custom container policy has been written.

Action: Report this problem to Oracle Support Services.

6057: METHOD_NOT_VALID

Cause: The method `methodName` is not valid to call on object `aReceiver`. This is a TopLink development exception, and the user should never encounter this problem unless a custom container policy has been written.

Action: Contact Oracle Support Services.

6058: METHOD_DOES_NOT_EXIST_IN_CONTAINER_CLASS

Cause: The method named `methodName` was not found in the class `aClass`. This exception is raised when looking for a clone method on the container class. The clone is needed to create clones of the container in unit of work.

Action: Define a clone method on the container class.

6059: COULD_NOT_INSTANTIATE_CONTAINER_CLASS

Cause: The class `aClass` cannot be used as the container for the results of a query because it cannot be instantiated. The exception is a Java exception that is raised when a new interface container policy is being created using Java reflection. TopLink wraps only the Java exception.

Action: Inspect the internal exception, and refer to the Java documentation.

6060: MAP_KEY_NOT_COMPARABLE

Cause: Cannot use the object `anObject` of type `ObjectClass` as a key into `aContainer` which is of type `ContainerClass`. The key cannot be compared with the keys currently in the map. This raises a Java reflection exception while accessing the method. TopLink wraps only the Java exception.

Action: Inspect the internal exception, and refer to the Java documentation.

6061: CANNOT_ACCESS_METHOD_ON_OBJECT

Cause: Cannot reflectively access the method `aMethod` for object: `anObject` of type `ObjectClass`. This raises a Java reflection exception while accessing the method. TopLink wraps only the Java exception.

Action: Inspect the internal exception, and refer to the Java documentation.

6062: CALLED_METHOD_THREW_EXCEPTION

Cause: The method `aMethod` was called reflectively on `objectClass` and threw an exception. The method `aMethod` raises a Java reflection exception while accessing a method. `TopLink` wraps only the Java exception.

Action: Inspect the internal exception, and refer to the Java documentation.

6063: INVALID_OPERATION

Cause: This is an invalid operation `operation` on the cursor. The operation is not supported.

Action: Check the class documentation and look for the corresponding method to use.

6064: CANNOT_REMOVE_FROM_CONTAINER

Cause: Cannot remove an `Object` of type `anObjectClass` from a `ContainerClass` using `policy`. This is a `TopLink` development exception and, the user should never encounter this problem unless a custom container policy has been written.

Action: Contact Oracle Support Services.

6065: CANNOT_ADD_ELEMENT

Cause: Cannot add an element to the collection container policy (cannot add object `anObject` of type `ObjectClass` to a `ContainerClass`).

Action: Inspect the internal exception, and refer to the Java documentation.

6066: BACKUP_CLONE_DELETED

Cause: Deleted objects cannot have references after being deleted. The object clone of class `clone.getClass` with identity hash code (`System.identityHashCode()`) (`System.identityHashCode(clone)`) has been deleted, but it still has references.

Action: Ensure that you are correctly registering your objects. If you are still having problems, use the `validateObjectSpace()` method of the `UnitOfWork` to help identify where the error occurred.

6068: CANNOT_COMPARE_TABLES_IN_EXPRESSION

Cause: Cannot compare table reference to `data` in expression.

Action: Check the expression.

6069: INVALID_TABLE_FOR_FIELD_IN_EXPRESSION

Cause: Field has invalid table in this context for field `data` in expression.

Action: Check the expression.

6070: INVALID_USE_OF_TO_MANY_QUERY_KEY_IN_EXPRESSION

Cause: This is an invalid use of a query key representing a one-to-many relationship `data` in expression.

Action: Use the `anyOf` operator instead of the `get` operator.

6071: INVALID_USE_OF_ANY_OF_IN_EXPRESSION

Cause: This is an invalid use of `anyOf` for a query key not representing a to-many relationship `data` in expression.

Action: Use the `get` operator instead of the `anyOf` operator.

6072: CANNOT_QUERY_ACROSS_VARIABLE_ONE_TO_ONE_MAPPING

Cause: Querying across a variable one-to-one mapping is not supported.

Action: Change the expression such that the query is not performed across a variable one-to-one mapping.

6073: ILL_FORMED_EXPRESSION

Cause: This is an ill-formed expression in query, attempting to print an object reference into a SQL statement for `queryKey`.

Action: Contact Oracle Support Services.

6074: CANNOT_CONFORM_EXPRESSION

Cause: This expression cannot determine if the object conforms in memory. Set the query to check the database.

Action: Change the query such that it does not attempt to conform to the results of the query.

6075: INVALID_OPERATOR_FOR_OBJECT_EXPRESSION

Cause: Object comparisons can use only the `equal` or `notEqual` operators. Other comparisons must be performed through query keys or direct attribute level comparisons.

Action: Ensure the query uses only `equal` and `notEqual` if object comparisons are being used.

6076: UNSUPPORTED_MAPPING_FOR_OBJECT_COMPARISON

Cause: Object comparisons can be used only with one-to-one mappings; other mapping comparisons must be performed through query keys or direct attribute level comparisons.

Action: Use a query key instead of attempting to compare objects across the mapping.

6077: OBJECT_COMPARISON_CANNOT_BE_PARAMETERIZED

Cause: Object comparisons cannot be used in parameter queries.

Action: Change the query so that it does not attempt to use objects when using parameterized queries.

6078: INCORRECT_CLASS_FOR_OBJECT_COMPARISON

Cause: The class of the argument for the object comparison is incorrect.

Action: Ensure the class for the query is correct.

6079: CANNOT_COMPARE_TARGET_FOREIGN_KEYS_TO_NULL

Cause: Object comparison cannot be used for target foreign key relationships.

Action: Query on source primary key.

6080: INVALID_DATABASE_CALL

Cause: This is an invalid database call. The call must be an instance of `DatabaseCall`.

Action: Ensure the call being used is a `DatabaseCall`.

6081: INVALID_DATABASE_ACCESSOR

Cause: Invalid database accessor. The accessor must be an instance of `DatabaseAccessor`.

Action: Ensure the accessor being used is a `DatabaseAccessor`.

6082: METHOD_DOES_NOT_EXIST_ON_EXPRESSION

Cause: The method `methodName` with argument type `argTypes` cannot be invoked on an expression.

Action: Ensure the method being used is a supported method.

6083: IN_CANNOT_BE_PARAMETERIZED

Cause: Queries using `IN` cannot be parameterized.

Action: Disable the query prepare or binding.

6084: REDIRECTION_CLASS_OR_METHOD_NOT_SET

Cause: The redirection query was not configured properly, the class or method name was not set.

Action: Verify the configuration for the redirection class.

6085: REDIRECTION_METHOD_NOT_DEFINED_CORRECTLY

Cause: The redirection query's method is not defined or it defines with the wrong arguments. It must be public static and have the following arguments: `DatabaseQuery`, `DatabaseRow`, or `Session` (the interface).

Action: Check the redirection query's method.

6086: REDIRECTION_METHOD_ERROR

Cause: The static invoke method provided to `MethodBaseQueryRedirector` threw an Exception when invoked.

Action: Check the static invoke method for problems.

6087: EXAMPLE_AND_REFERENCE_OBJECT_CLASS_MISMATCH

Cause: There is a class mismatch between the example object and the reference class specified for this query.

Action: Ensure that the example and reference classes are compatible.

6088: NO_ATTRIBUTES_FOR_REPORT_QUERY

Cause: A `ReportQuery` has been built with no attributes specified.

Action: Specify the attribute for the query.

6089: NO_EXPRESSION_BUILDER_CLASS_FOUND

Cause: The expression has not been initialized correctly. Only a single `ExpressionBuilder` should be used for a query. For parallel expressions, the query class must be provided to the `ExpressionBuilder` constructor, and the query's `ExpressionBuilder` must always be on the left side of the expression.

Action: Contact Oracle Support Services.

6090: CANNOT_SET_REPORT_QUERY_TO_CHECK_CACHE_ONLY

Cause: The `checkCacheOnly` method was invoked on a `ReportQuery`. You cannot invoke the `checkCacheOnly` method on a `ReportQuery` because a `ReportQuery` returns data rather than objects and the `TopLink` cache is built with objects.

Action: Do not use a `ReportQuery` in this case.

6091: TYPE_MISMATCH_BETWEEN_ATTRIBUTE_AND_CONSTANT_ON_EXPRESSION

Cause: The type of the constant used for comparison in the expression does not match the type of the attribute.

Action: Contact Oracle Support Services.

6092: MUST_INSTANTIATE_VALUEHOLDERS

Cause: Uninstantiated value holders have been detected.

Action: Instantiate the value holders for the collection on which you want to query.

6093: MUST_BE_ONE_TO_ONE_OR_ONE_TO_MANY_MAPPING

Cause: The `buildSelectionCriteria` method was invoked on a mapping that was neither one-to-one nor one-to-many. Only the one-to-one and one-to-many mapping exposes this public API to build selection criteria. Using the `buildSelectionCriteria` method with other mapping types will not return correct results.

Action: Use the `buildSelectionCriteria` method only with one-to-one and one-to-many mappings.

6094: PARAMETER_NAME_MISMATCH

Cause: An unmapped field was used in a parameterized expression.

Action: Map the field or define an alternate expression that does not rely on the unmapped field.

6095: CLONE_METHOD_REQUIRED

Cause: A delegate class of an `IndirectContainer` implementation does not implement `Cloneable`. If you implement `IndirectContainer` you must also implement `Cloneable`. For example, see `oracle.toplink.indirection.IndirectSet`. The `clone` method must clone the delegate. For example, the `IndirectSet` implementation uses reflection to invoke the `clone` method because it is not included in the common interface shared by `IndirectSet` and its base delegate class, `HashSet`.

Action: Ensure that your `IndirectContainer` implementation or its delegate class implements `Cloneable`.

6096: CLONE_METHOD_INACCESSIBLE

Cause: A delegate class of an `IndirectContainer` implementation implements `Cloneable` but the `IndirectContainer` implementation does not have access to the specified clone method. That is, a `java.lang.IllegalAccessException` is raised when the delegate's clone method is invoked.

Action: Ensure that both the delegate clone method and the delegate class are public. Ensure permission is set for Java reflection in your VM security settings. See also the `invoke` method of `java.lang.reflect.Method`.

6097: CLONE_METHOD_THROW_EXCEPTION

Cause: A delegate class of an `IndirectContainer` implementation implements `Cloneable` and the `IndirectContainer` implementation has access to the specified clone method, but the specified clone method raises a `java.lang.reflect.InvocationTargetException` when invoked.

Action: Verify the implementation of the delegate's clone method.

6098: UNEXPECTED_INVOCATION

Cause: A proxy object method raises an unexpected exception when invoked (that is, some exception other than `InvocationTargetException` and `ValidationException`.)

Action: Review the proxy object to see where it is throwing the exception described in the exception Message. Ensure this exception is no longer raised.

6099: JOINING_ACROSS_INHERITANCE_WITH_MULTIPLE_TABLES

Cause: Joining with query across inheritance class with multiple table subclasses is not supported.

Action: Joining cannot be used on relationships with inheritance classes that have subclasses that span multiple tables as this requires multiple separate queries. The multiple queries cannot be joined into a single query. Use batch reading on the relationship instead, as this will provide equivalent or better performance.

6100: MULTIPLE_ROWS_DETECTED_FROM_SINGLE_OBJECT_READ

Cause: Multiple values detected for single-object read query.

Action: This is a CMP compliance option that ensures the finder methods for a single object only return a single row. Set the system property `toplink.cts.checkMultipleRows` to `false`, or ensure that the finder query only returns a single row from the database.

6101: HISTORICAL_QUERIES_MUST_PRESERVE_GLOBAL_CACHE

Cause: Executing this query could violate the integrity of the global session cache which must contain only the latest versions of objects.

Action: To execute a query that returns objects of a historical nature, you must do one of the following:

1. Use a `HistoricalSession` (`acquireSessionAsOf`). All objects read will be cached and automatically read at that time. This applies also to triggering object relationships.
2. Set `shouldMaintainCache` to `false`. You may make any object expression as of a previous time, provided none of its fields are represented in the result set (i.e. used in the `where` clause).

6102: HISTORICAL_QUERIES_ONLY_SUPPORTED_ON_ORACLE

Cause: Historical queries only work with Oracle 9R2 or later databases, as it uses the Oracle database Flashback feature.

Action: Ensure that historical queries are only used with an Oracle 9R2 database.

6103: INVALID_QUERY_ON_HISTORICAL_SESSION

Cause: You may not execute a `WriteQuery` from inside a read-only `HistoricalSession`. To restore historical objects, try the following: read the same object as it is now with a `UnitOfWork` and commit the `UnitOfWork`.

Action: To restore historical objects, read the same object as it is now with a `UnitOfWork` and commit the `UnitOfWork`.

6104: OBJECT_DOES_NOT_EXIST_IN_CACHE

Cause: The object does not exist in the cache.

Action: Ensure that the object exists in the cache.

6105: MUST_USE_CURSOR_STREAM_POLICY

Cause: Query must be re-initialized with a cursor stream policy.

Action: Re-initialize the query with a cursor stream policy.

6106: CLASS_PK_DOES_NOT_EXIST_IN_CACHE

Cause: The object with primary key does not exist in the cache.

Action: Ensure that the object exists in the cache.

6107: UPDATE_STATEMENTS_NOT_SPECIFIED

Cause: Missing update statements on `UpdateAllQuery`.

Action: Add update statements using the `addUpdate` method.

6108: INHERITANCE_WITH_MULTIPLE_TABLES_NOT_SUPPORTED

Cause: `UpdateAllQuery` does not support inheritance with multiple tables.

Action: Do not use `UpdateAllQuery` in this situation.

6109: QUERY_FETCHGROUP_NOT_DEFINED_IN_DESCRIPTOR

Cause: The named fetch group is not defined at the descriptor level.

Action: Ensure the fetch group is defined in the descriptor.

6110: CANNOT_CONFORM_UNFETCHED_ATTRIBUTE

Cause: Read query cannot conform to the unfetched attribute of the partially fetched object in the unit of work identity map.

Action: Do not use unfetched attribute conforming, or explicitly fetch the attribute before conforming.

6111: FETCH_GROUP_ATTRIBUTE_NOT_MAPPED

Cause: The fetch group attribute is not defined or mapped.

Action: Ensure that any attribute defined in a fetch group is defined in the class and mapped.

6112: FETCH_GROUP_NOT_SUPPORT_ON_REPORT_QUERY

Cause: Fetch group cannot be set on report query.

Action: Remove the fetch group setting on `ReportQuery`, or use `ReadObjectQuery` or `ReadObjectQuery` instead.

6113: FETCH_GROUP_NOT_SUPPORT_ON_PARTIAL_ATTRIBUTE_READING

Cause: Fetch group cannot be used together with partial attribute reading.

Action: Remove the partial attribute reading setting in the query.

6114: FETCHGROUP_VALID_ONLY_IF_FETCHGROUP_MANAGER_IN_DESCRIPTOR

Cause: A fetch group manager is not defined at the descriptor while attempting to set a fetch group on a query.

Action: You must define a fetch group manager at the descriptor in order to set a fetch group on the query.

6115: ISOLATED_QUERY_EXECUTED_ON_SERVER_SESSION

Cause: Queries on isolated classes, or queries set to use exclusive connections, must not be executed on a `ServerSession` or in `CMP`, outside of a transaction.

Action: Do not execute queries on isolated classes or queries set to use exclusive connections on a `ServerSession` or in `CMP` outside of a transaction.

6116: NO_CALL_OR_INTERACTION_SPECIFIED

Cause: No call or interaction method was specified for the attempted operation.

Action: Specify a call or interaction method.

6117: CANNOT_CACHE_CURSOR_RESULTS_ON_QUERY

Cause: Cannot set a query that uses a cursored result to cache query results.

Action: Do not cache query results or do not use a cursor policy.

6118: CANNOT_CACHE_ISOLATED_DATA_ON_QUERY

Cause: A query on an isolated class must not cache query results on the query.

Action: Do not cache query results for a query on an isolated class.

6119: MAPPING_FOR_EXPRESSION_DOES_NOT_SUPPORT_JOINING

Cause: The join expression is not valid, or is for a mapping type that does not support joining.

Action: Joining is supported only for one-one and one-many mappings.

6120: SPECIFIED_PARTIAL_ATTRIBUTE_DOES_NOT_EXIST

Cause: The partial attribute is not a valid attribute of the class.

Action: Ensure that this attribute exists, and is mapped.

6121: INVALID_BUILDER_IN_QUERY

Cause: The query has not been defined correctly: the expression builder is missing.

Action: Ensure the queries builder is always on the left for sub queries and parallel queries.

Validation Exceptions (7001 – 7147)

`ValidationException` is a development exception that is raised when an incorrect state is detected or an API is used incorrectly.

Format

EXCEPTION [TOPLINK - error code]: Exception name

EXCEPTION DESCRIPTION: Message

Example 13–7 Validation Exception

```
EXCEPTION [TOPLINK - 7008]: oracle.toplink.exceptions.ValidationException
EXCEPTION DESCRIPTION: The Java type javaClass is not a valid database type. The
Java type of the field to be written to the database has no corresponding type on
the database.
```

7001: LOGIN_BEFORE_ALLOCATING_CLIENT_SESSIONS

Cause: You attempted to allocate client sessions before logging into the server.

Action: Ensure you have called `login` method on your server session or database session. This error also appears in multi threaded environments as a result of concurrency issues. Check that all your threads are synchronized.

7002: POOL_NAME_DOES_NOT_EXIST

Cause: The pool name used while acquiring client session from the server session does not exist.

Action: Verify the pool name given while acquiring client session and all the existing pools on the server session.

7003: MAX_SIZE_LESS_THAN_MIN_SIZE

Cause: The maximum number of connections in a connection pool should be more than the minimum number of connections.

Action: Check `addConnectionPool(String poolName, JDBCLogin login, int minNumberOfConnections, int maxNumberOfConnections)` method on the server session.

7004: POOLS_MUST_BE_CONFIGURED_BEFORE_LOGIN

Cause: Pools must all be added before login on the server session has been done. Once logged in, you cannot add pools.

Action: Check `addConnectionPool(String poolName, JDBCLogin login, int minNumberOfConnections, int maxNumberOfConnections)` on server session. This method should be called before logging in on the server session.

7008: JAVA_TYPE_IS_NOT_A_VALID_DATABASE_TYPE

Cause: The Java type `javaClass` is not a valid database type. The Java type of the field to be written to the database has no corresponding type on the database.

Action: Check the table or stored procedure definition.

7009: MISSING_DESCRIPTOR

Cause: The descriptor `className` is not found in the session.

Action: Ensure that the related descriptor to the class was properly registered with the session.

7010: START_INDEX_OUT_OF_RANGE

Cause: This is a TopLink development exception and users should never encounter this problem. It happens when a copy of a vector is created with a start and end index.

Action: Report this problem to Oracle Support Services.

7011: STOP_INDEX_OUT_OF_RANGE

Cause: This is a TopLink development exception and users should never encounter this problem. It happens when a copy of a vector is created with a start and end index.

Action: Report this problem to Oracle Support Services.

7012: FATAL_ERROR_OCCURRED

Cause: This is a TopLink development exception and users should never encounter this problem. It happens when test cases are executed.

Action: Report this problem to Oracle Support Services. This error commonly occurs if you attempt to call the `commit` method on an invalid (or previously committed) unit of work.

If `cannotCommitUOWAgain` method of `ValidationException` appears in the stack trace, verify that the `commit` method was called on valid `UnitOfWork` instances.

7013: NO_PROPERTIES_FILE_FOUND

Cause: The `toplink.properties` file cannot be found on the system classpath.

Action: Ensure that there is a `toplink.properties` file located on the system classpath.

7017: CHILD_DESCRIPTOR_DO_NOT_HAVE_IDENTITY_MAP

Cause: An identity map is added to the child descriptor. A child descriptor shares its parent's identity map.

Action: Check the child descriptor and remove the identity map from it.

7018: FILE_ERROR

Cause: The user should never encounter this problem. It happens when test cases are executed.

Action: Contact Oracle Support Services.

7023: INCORRECT_LOGIN_INSTANCE_PROVIDED

Cause: The login instance provided to the `login` method is incorrect. A `JDBCLogin` must be provided.

Action: Use a `JDBCLogin`.

7024: INVALID_MERGE_POLICY

Cause: This is a TopLink development exception and users should never encounter it.

Action: Contact Oracle Support Services.

7025: ONLY_FIELDS_ARE_VALID_KEYS_FOR_DATABASE_ROWS

Cause: The key on the database row is not either of type `String` or of type `DatabaseField`.

Action: Contact Oracle Support Services.

7027: SEQUENCE_SETUP_INCORRECTLY

Cause: The sequence `sequenceName` is set up incorrectly, increment does not match pre-allocation size.

Action: Contact Oracle Support Services.

7028: WRITE_OBJECT_NOT_ALLOWED_IN_UNIT_OF_WORK

Cause: A `writeObject()` method is not allowed in a `UnitOfWork`.

Action: Ensure that a `writeObject()` method is not in the `UnitOfWork`.

7030: CANNOT_SET_READ_POOL_SIZE_AFTER_LOGIN

Cause: TopLink is unable to set read pool size after the server session has already been logged in.

Action: The size should be set before login.

7031: CANNOT_ADD_DESCRIPTOR_TO_SESSION_BROKER

Cause: TopLink cannot add descriptors to a session broker.

Action: Descriptors are added to the sessions contained in the session broker.

7032: NO_SESSION_REGISTERED_FOR_CLASS

Cause: The descriptor related to the domain class `domainClass` was not found in any of the sessions registered in the session broker.

Action: Check the sessions.

7033: NO_SESSION_REGISTERED_FOR_NAME

Cause: The session with the given name `sessionName` is not registered in the session broker.

Action: Check the session broker.

7038: LOG_IO_ERROR

Cause: Error while logging message to session's log.

Action: Check the internal exception.

7039: CANNOT_REMOVE_FROM_READ_ONLY_CLASSES_IN_NESTED_UNIT_OF_WORK

Cause: TopLink is unable to remove from the set of read-only classes in a nested unit of work. A nested unit of work's set of read-only classes must be equal to or be a superset of its parent's set of read-only classes.

Action: Contact Oracle Support Services.

7040: CANNOT_MODIFY_READ_ONLY_CLASSES_SET_AFTER_USING_UNIT_OF_WORK

Cause: TopLink is unable to change the set of read-only classes in a unit of work after that unit of work has been used. Changes to the read-only set must be made when acquiring the unit of work or immediately after.

Action: Contact Oracle Support Services.

7042: PLATFORM_CLASS_NOT_FOUND

Cause: The platform class `c className` was not found and a reflection exception is raised.

Action: Check the internal exception.

7043: NO_TABLES_TO_CREATE

Cause: A project does not have any tables to create on the database.

Action: Validate the project and tables you are attempting to create.

7044: ILLEGAL_CONTAINER_CLASS

Cause: The container class specified `c className` cannot be used as the container because it does not implement the `Collection` or `Map` interfaces.

Action: Implement either the `Collection` or `Map` interfaces in the container class.

7047: CONTAINER_POLICY_DOES_NOT_USE_KEYS

Cause: Invalid `Map` class was specified for the container policy. The container specified (of `Class aPolicyContainerClass`) does not require keys. You tried to use `methodName`.

Action: Use `map` class that implements the `Map` interface.

7048: METHOD_NOT_DECLARED_IN_ITEM_CLASS

Cause: The key method on the map container policy is not defined. The instance method `<methodName>` does not exist in the reference class `<className>` and therefore cannot be used to create a key in a map. A map container policy represents how to handle an indexed collection of objects. Usually the key is the primary key of the objects stored, so the policy needs to know the name of the primary key get method, to extract it from each object using reflection. For instance a user might call `policy.setKeyMethodName("getId")`.

Action: Check the second parameter of the `useMapClass` method of `DatabaseQuery`.

7051: MISSING_MAPPING

Cause: Missing the attribute `attributeName` for descriptor `descriptor` called from `source`. This is a TopLink development exception and a user should never encounter it.

Action: Contact Oracle Support Services.

7052: ILLEGAL_USE_OF_MAP_IN_DIRECTCOLLECTION

Cause: The method `useMapClass` was called on a `DirectCollectionMapping`. It is invalid to call the `useMapClass` method on a `DirectCollectionMapping`. `TopLink` cannot instantiate Java attributes mapped using a `DirectCollectionMapping` with a map. The `useMapClass` method is supported for `OneToManyMappings` and `ManyToManyMappings`. The Java 2 `Collection` interface is supported using the `useCollectionClass` method.

Action: Use the `useCollectionClass` method. Do not call the `useMapClass` method on `DirectCollectionMapping`.

7053: CANNOT_RELEASE_NON_CLIENTSESSION

Cause: `TopLink` is unable to release a session that is not a client session. Only client sessions can be released.

Action: Modify the code to ensure the client session is not released.

7054: CANNOT_ACQUIRE_CLIENTSESSION_FROM_SESSION

Cause: `TopLink` is unable to acquire a session that is not a client session. Client sessions can be acquired only from server sessions.

Action: Modify the code to ensure an acquire session is attempted only from server sessions.

7055: OPTIMISTIC_LOCKING_NOT_SUPPORTED

Cause: Optimistic locking is not supported with stored procedure generation.

Action: Do not use `OptimisticLocking` with stored procedure generation.

7056: WRONG_OBJECT_REGISTERED

Cause: The wrong object was registered into the unit of work. It should be the object from the parent cache.

Action: Ensure that the object is from the parent cache.

7058: INVALID_CONNECTOR

Cause: The connector selected is invalid and must be of type `DefaultConnector`.

Action: Ensure that the connector is of type `DefaultConnector`.

7059: INVALID_DATA_SOURCE_NAME

Cause: Invalid data source name: name.

Action: Verify the data source name.

7060: CANNOT_ACQUIRE_DATA_SOURCE

Cause: `TopLink` is unable to acquire the data source name or an error has occurred in setting up the data source.

Action: Verify the data source name. Check the nested SQL exception to determine the cause of the error. Typical problems include:

- The connection pool was not configured in your `config.xml` file.
- The driver is not on the classpath.
- The user or password is incorrect.
- The database server URL or driver name is not properly specified.

7061: JTS_EXCEPTION_RAISED

Cause: An exception occurred within the Java Transaction Service (JTS).

Action: Examine the JTS exception and see the JTS documentation.

7062: FIELD_LEVEL_LOCKING_NOTSUPPORTED_OUTSIDE_A_UNIT_OF_WORK

Cause: `FieldLevelLocking` is not supported outside a unit of work. In order to use field-level locking, a unit of work must be used for *all* write operations.

Action: Use a unit of work for writing.

7063: EJB_CONTAINER_EXCEPTION_RAISED

Cause: An exception occurred within the EJB container.

Action: Examine the EJB exception and see the JTS documentation.

7064: EJB_PRIMARY_KEY_REFLECTION_EXCEPTION

Cause: An exception occurred in the reflective EJB bean primary key extraction.

Action: Ensure that your primary key object is defined correctly.

7065: EJB_CANNOT_LOAD_REMOTE_CLASS

Cause: The remote class for the bean cannot be loaded or found, for the bean.

Action: Ensure that the correct class loader is set correctly.

7066: EJB_MUST_BE_IN_TRANSACTION

Cause: `TopLink` is unable to create or remove beans unless a JTS transaction is present, `bean=bean`.

Action: Ensure that the JTS transaction is present.

7068: EJB_INVALID_PROJECT_CLASS

Cause: The platform class `platformName` was not found for the `projectName` using the default class loader.

Action: Validate the project and platform.

7069: PROJECT_AMENDMENT_EXCEPTION_OCCURED

Cause: An exception occurred while looking up or invoking the project amendment method, `amendmentMethod` on the class `amendmentClass`.

Action: Validate the amendment method and class.

7070: EJB_TOPLINK_PROPERTIES_NOT_FOUND

Cause: A `toplink.properties` resource bundle must be located on the classpath in a `TopLink` directory.

Action: Validate the classpath and the location of the `TopLink` resource bundle.

7071: CANT_HAVE_UNBOUND_IN_OUTPUT_ARGUMENTS

Cause: You cannot use input or output parameters without using binding.

Action: Use binding on the `StoredProcedureCall`.

7072: EJB_INVALID_PLATFORM_CLASS

Cause: `SessionManager` failed to load the class identified by the value associated with properties `platform-class` or `external-transaction-controller-class` during initialization when it loads the `TopLink` session common properties from the `TopLink` global properties file (`sessions.xml` for non-EJB applications or `toplink-ejb-jar.xml` for EJB applications).

Action: Ensure that your TopLink global properties file is correctly configured. Pay particular attention to the `platform-class` and `external-transaction-controller-class` properties.

7073: ORACLE_OBJECT_TYPE_NOT_DEFINED

Cause: The Oracle object type with type name `typeName` is not defined.

Action: Ensure that the Oracle object type is defined.

7074: ORACLE_OBJECT_TYPE_NAME_NOT_DEFINED

Cause: The Oracle object type `typeName` is not defined.

Action: Ensure that the Oracle object type is defined.

7075: ORACLE_VARRAY_MAXIMIM_SIZE_NOT_DEFINED

Cause: The Oracle VARRAY type `typeName` maximum size is not defined.

Action: Verify the maximum size for the Oracle VARRAY.

7076: DESCRIPTOR_MUST_NOT_BE_INITIALIZED

Cause: When generating the project class, the descriptors must not be initialized.

Action: Ensure that the descriptors are not initialized before generating the project class.

7077: EJB_INVALID_FINDER_ON_HOME

Cause: The home interface `toString` method specified during creation of `BMPWrapperPolicy` does not contain a correct `findByPrimaryKey` method. A `findByPrimaryKey` method must exist that takes the `PrimaryKey` class for this bean.

Action: Ensure that a `findByPrimaryKey` method exists and is correct.

7078: EJB_NO_SUCH_SESSION_SPECIFIED_IN_PROPERTIES

Cause: The `sessionName` specified on the deployment descriptor does not match any session specified in the `toplink.properties` file.

Action: Contact Oracle Support Services.

7079: EJB_DESCRIPTOR_NOT_FOUND_IN_SESSION

Cause: The descriptor was not found in the session.

Action: Check the project being used for this session.

7080: EJB_FINDER_EXCEPTION

Cause: A `FinderException` is raised when attempting to load an object from the class with the primary key.

Action: Contact Oracle Support Services.

7081: CANNOT_REGISTER_AGGREGATE_OBJECT_IN_UNIT_OF_WORK

Cause: The aggregate object cannot be directly registered in the unit of work. It must be associated with the source (owner) object.

Action: Contact Oracle Support Services.

7082: MULTIPLE_PROJECTS_SPECIFIED_IN_PROPERTIES

Cause: The `toplink.properties` file specified multiple project files for the server. Only one project file can be specified.

Action: Specify either `projectClass`, `projectFile`, or `xmlProjectFile`.

7083: NO_PROJECT_SPECIFIED_IN_PROPERTIES

Cause: The `toplink.properties` file does not include any information on the TopLink project to use for the server. One project file must be specified.

Action: Specify either `projectClass`, `projectFile`, or `xmlProjectFile`.

7084: INVALID_FILE_TYPE

Cause: The specified file is not a valid type for reading. `ProjectReader` must be given the deployed XML project file.

Action: Contact Oracle Support Services.

7085: SUB_SESSION_NOT_DEFINED_FOR_BROKER

Cause: Unable to create an instance of the external transaction controller specified in the properties file.

Action: Contact Oracle Support Services.

7086: EJB_INVALID_SESSION_TYPE_CLASS

Cause: The session manager cannot load the class corresponding to the session's type class name.

Action: Ensure that the class name of the session's type is fully qualified in the `sessions.xml` file or `toplink.properties` file.

7087: EJB_SESSION_TYPE_CLASS_NOT_FOUND

Cause: The session manager cannot load the class corresponding to the session's type class name.

Action: Ensure that the class name of the session's type is fully qualified in the `sessions.xml` file or `toplink.properties` file.

7088: CANNOT_CREATE_EXTERNAL_TRANSACTION_CONTROLLER

Cause: The session manager cannot load the class corresponding to the external transaction controller's class name.

Action: Ensure that the class name of the external transaction controller is valid and fully qualified in the `sessions.xml` file or `toplink.properties` file.

7089: SESSION_AMENDMENT_EXCEPTION_OCCURRED

Cause: The session manager cannot load the class corresponding to the amendment class name, or it cannot load the method on the amendment class corresponding to the amendment method name.

Action: Ensure that the class name of the amendment class is fully qualified, and the amendment method exists in the amendment class in the `sessions.xml` file or `toplink.properties` file.

7091: SET_LISTENER_CLASSES_EXCEPTION

Cause: TopLink is unable to create the listener class that implements `SessionEventListener` for the internal use of `SessionXMLProject`.

Action: Contact Oracle Support Services.

7092: EXISTING_QUERY_TYPE_CONFLICT

Cause: TopLink has detected a conflict between a custom query with the same name and arguments to a session.

Action: Ensure that no query is added to the session more than once or change the query name so that the query can be distinguished from others.

7093: QUERY_ARGUMENT_TYPE_NOT_FOUND

Cause: TopLink is unable to create an instance of the query argument type.

Action: Ensure that the argument type is a fully qualified class name and the argument class is included in the classpath environment.

7094: ERROR_IN_SESSION_XML

Cause: The `sessions.xml` or `toplink.properties` files cannot be loaded.

Action: Ensure that the path to either of the files exists on the classpath environment.

7095: NO_SESSIONS_XML_FOUND

Cause: The `sessions.xml` or `toplink.properties` files cannot be loaded.

Action: Ensure that the path to either of the files exists on the classpath environment. The `sessions.xml` should be included in the root of the deployed JAR file. When using a WAR file, the `sessions.xml` file should be located in the `WEB-INF\classes` directory. When using EJB 3.0, TopLink automatically loads the `ejb3-toplink-sessions.xml` file.

7096: CANNOT_COMMIT_UOW_AGAIN

Cause: TopLink cannot invoke `commit` method on an inactive unit of work that was committed or released.

Action: Ensure you invoke `commit` method on a new unit of work or invoke `commitAndResume` method so that the unit of work can be reused. For more information about the `commitAndResume` method, see *Oracle TopLink API Reference*.

7097: OPERATION_NOT_SUPPORTED

Cause: TopLink cannot invoke a nonsupport operation on an object.

Action: Do not use the operation indicated in the stack trace.

7099: PROJECT_XML_NOT_FOUND

Cause: The file name specified for the XML-based project is incorrect.

Action: Verify the name and location of the file.

7101: NO_TOPLINK_EJB_JAR_XML_FOUND

Cause: The `toplink-ejb-jar.xml` file was not found.

Action: Ensure that the file is on your classpath.

7102: NULL_CACHE_KEY_FOUND_ON_REMOVAL

Cause: Encountered a `null` value for a cache key while attempting to remove an object from the identity map. The most likely cause of this situation is that the object has already been garbage-collected and therefore does not exist within the identity map.

Action: Ignore. The `removeFromIdentityMap` method of the Session is intended to allow garbage collection, which has already been done.

7103: NULL_UNDERLYING_VALUEHOLDER_VALUE

Cause: A `null` reference was encountered while attempting to invoke a method on an object that uses proxy indirection.

Action: Please check that this object is not `null` before invoking its methods.

7104: INVALID_SEQUENCING_LOGIN

Cause: A separate connection(s) for sequencing was requested but the sequencing login uses the external transaction controller.

Action: Either provide a sequencing login that does not use an external transaction controller or do not use separate connection(s) for sequencing.

7105: INVALID_ENCRYPTION_CLASS

Cause: Error encountered while converting encryption class.

Action: Ensure the encryption class name is correctly specified in the `sessions.xml` file and that the encryption class specified is available on the classpath. A common reason for this exception is the usage of JDK 1.3 and earlier versions. The TopLink JCE encryption mechanism requires JDK 1.4 and higher (or JDK 1.3 configured with the JCE plug-in) to function properly.

7106: ERROR_ENCRYPTING_PASSWORD

Cause: Error encountered during password string encryption.

Action: An error is raised while trying to encrypt the password string. A common reason for this exception is the usage of JDK 1.3 and earlier versions. The TopLink JCE encryption mechanism requires JDK 1.4 and higher (or JDK 1.3 configured with the JCE plug-in) to function properly.

7107: ERROR_DECRYPTING_PASSWORD

Cause: Error encountered during password string decryption.

Action: An exception was raised while trying to decrypt the password string. A common reason for this exception is the usage of JDK 1.3 and earlier versions. The TopLink JCE encryption mechanism requires JDK 1.4 and higher (or JDK 1.3 configured with the JCE plug-in) to function properly.

7108: NOT_SUPPORTED_FOR_DATASOURCE

Cause: This operation is not supported for non-relational platforms.

Action: Do not use this operation on the current platform, or use a relational database platform.

7109: PROJECT_LOGIN_IS_NULL

Cause: The login in the project used to create the session is `null`. The login used for the project must be a valid login.

Action: No login was specified for the TopLink project. A valid login must be included on any TopLink project. Login information can be added using TopLink Workbench or using Java code.

7110: HISTORICAL_SESSION_ONLY_SUPPORTED_ON_ORACLE

Cause: At present `HistoricalSession` only works with Oracle 9R2 or later databases, as it uses the Oracle database Flashback feature.

Action: Generic History Support (see `oracle.toplink.history.HistoryPolicy`) works for any database. If a `HistoryPolicy` is incorrectly set, TopLink may be defaulting to using flashback instead. An `ASOFSCN` clause is implicitly flashback only.

7111: CANNOT_ACQUIRE_HISTORICAL_SESSION

Cause: You may not acquire a `HistoricalSession` from a unit of work, another `HistoricalSession`, a `ServerSession`, or a `ServerSessionBroker`. You may acquire one from a regular session, a `ClientSession`, or a `ClientSessionBroker`.

Action: To recover objects, read the objects in both a `HistoricalSession` and `UnitOfWork`, and call `mergeCloneWithReferences(historicalObject)` method on the `UnitOfWork`.

7112: FEATURE_NOT_SUPPORTED_IN_JDK_VERSION

Cause: You have specified that TopLink use a feature, but this feature is not available in the current JDK version.

Action: You must use the version of the JDK that supports this feature.

7113: PLATFORM_DOES_NOT_SUPPORT_CALL_WITH_RETURNING

Cause: Platform does not support call with returning.

Action: Set stored procedures with output parameters in `setInsertQuery`, `setInsertCall`, `setUpdateQuery`, or `setUpdateCall` methods of the `DescriptorQueryManager`.

7114: ISOLATED_DATA_NOT_SUPPORTED_IN_CLIENTSESSIONBROKER

Cause: Isolated data is not currently supported within a `ClientSessionBroker`. Session contains descriptors representing isolated data.

Action: Ensure that isolated data is not used.

7115: CLIENT_SESSION_CANNOT_USE_EXCLUSIVE_CONNECTION

Cause: An `ExclusiveConnection` cannot be used for `ClientSession` reads without isolated data.

Action: You must update the `ConnectionPolicy` used, to remove `ExclusiveConnection` configuration or the project to set certain data to be exclusive.

7116: INVALID_METHOD_ARGUMENTS

Cause: Invalid arguments are used in the method.

Action: Refer to the public API of the calling method and use valid values for the arguments.

7117: MULTIPLE_CURSORS_NOT_SUPPORTED

Cause: There is an attempt to use more than one cursor in a `SQLCall`.

Action: TopLink currently supports only one cursor per call.

7118: WRONG_USAGE_OF_SET_CUSTOM_SQL_ARGUMENT_TYPE_METHOD

Cause: The `setCustomSQLArgumentType` method was invoked on `SQLCall`, but this method does not use custom SQL.

Action: Don't call this method on `SQLCall` that doesn't use custom SQL.

7119: CANNOT_TRANSLATE_UNPREPARED_CALL

Cause: Unprepared `SQLCall` attempted translation.

Action: `SQLCall` must be prepared before translation.

7120: CANNOT_SET_CURSOR_FOR_PARAMETER_TYPE_OTHER_THAN_OUT

Cause: Parameter in `SQLCall` cannot be used as a cursor, because it has parameter type other than OUT.

Action: Parameter used as a cursor must have parameter type OUT.

7121: PLATFORM_DOES_NOT_SUPPORT_STORED_FUNCTIONS

Cause: Platform does not support stored functions.

Action: Do not define stored functions on this platform.

7122: EXCLUSIVE_CONNECTION_NO_LONGER_AVAILABLE

Cause: The exclusive connection associated with the session is unavailable for the query on the object.

Action: Isolated objects with indirection read through an `ExclusiveIsolatedClientSession` must not have indirection triggered after the `ExclusiveIsolatedClientSession` has been released. Re-read the objects through the current `ExclusiveIsolatedClientSession`.

7123: UNIT_OF_WORK_IN_TRANSACTION_COMMIT_PENDING

Cause: A successful `writeChanges` operation has been called on this `UnitOfWork`. As the commit process has been started but not yet finalized, the only supported operations now are `commit`, `commitAndResume`, `release`, any non-object level query, or `SQLCall` execution. The operation is not allowed at this time.

Action: Execute one of the supported operations to continue.

7124: UNIT_OF_WORK_AFTER_WRITE_CHANGES_FAILED

Cause: An unsuccessful `writeChanges` operation has been called on this `UnitOfWork`. Given the possibility that partial changes have been written to the data store but not rolled back (if inside external transaction), the only supported operations now are `release`, global transaction rollback, any non-object level query or `SQLCall` execution.

Action: Determine the cause of the original failure and retry in a new `UnitOfWork`.

7125: INACTIVE_UNIT_OF_WORK

Cause: Once the `UnitOfWork` has been committed and/or released, no further operation should be performed on it.

Action: Acquire a new `UnitOfWork`, or use the `commitAndResume` method instead of `commit` method in the future.

7126: CANNOT_WRITE_CHANGES_ON_NESTED_UNIT_OF_WORK

Cause: The `writeChanges` method cannot be called on a `NestedUnitOfWork`. A nested `UnitOfWork` never writes changes directly to the data store, only the parent `UnitOfWork` does.

Action: Call the `commit` method instead, and then the `writeChanges` method on the parent `UnitOfWork`.

7127: CANNOT_WRITE_CHANGES_TWICE

Cause: You can only writes changes to the data store once.

Action: You must either roll back the transaction, or call the `commit` method on this `UnitOfWork` and start a new one.

7128: ALREADY_LOGGED_IN

Cause: Session is already logged in.

Action: Do not try to login again.

7129: INVALID_NULL_METHOD_ARGUMENTS

Cause: The method's arguments cannot have a `null` value.

Action: Ensure that the method's arguments do not have a null value.

7130: NESTED_UOW_NOT_SUPPORTED_FOR_ATTRIBUTE_TRACKING

Cause: Nested `UnitOfWork` is not supported for attribute change tracking.

Action: Do not use a nested `UnitOfWork` with attribute change tracking.

7131: WRONG_COLLECTION_CHANGE_EVENT_TYPE

Cause: The collection change event is the wrong type. The collection change event type has to be added or removed.

Action: Ensure that the collection change event type used is defined in `CollectionChangeEvent`.

7132: WRONG_CHANGE_EVENT

Cause: Wrong event class. Only `PropertyChangeEvent` and `CollectionChangeEvent` classes are supported.

Action: Ensure that the event class is either `PropertyChangeEvent` or `CollectionChangeEvent`.

7133: OLD_COMMIT_NOT_SUPPORTED_FOR_ATTRIBUTE_TRACKING

Cause: Old commit is not supported for attribute change tracking.

Action: Do not try to use attribute change tracking with an old commit.

7134: SERVER_PLATFORM_IS_READ_ONLY_AFTER_LOGIN

Cause: The server platform is read only after login.

Action: Changes to the server platform must be made before login. You must either:

1. CMP: Define a class that implements `oracle.toplink.ejb.cmp.DeploymentCustomization`, and customize its public `String beforeLoginCustomization(Session session)` class to change your server platform. Consult the documentation for defining a customization class in your `orion-ejb-jar.xml` file.
2. Non-CMP/POJO: Define a subclass of `oracle.toplink.sessions.SessionEventAdapter`, and override the public `void preLogin(SessionEvent event)` method to change your server platform. The session is contained in the event. Consult the documentation for `sessions.xml` and using `SessionEventAdapter`.

7135: CANNOT_COMMIT_AND_RESUME_UOW_WITH_UPDATE_ALL_QUERIES

Cause: You cannot commit and resume a `UnitOfWork` containing an `UpdateAllQuery`.

Action: You must either commit and continue in a new `UnitOfWork`, or do not use `UpdateAllQuery`.

7136: NESTED_UOW_NOT_SUPPORTED_FOR_UPDATE_ALL_QUERY

Cause: Nested `UnitOfWork` is not supported for an update all query.

Action: Do not use a nested `UnitOfWork` for an `UpdateAllQuery`.

7137: UNFETCHED_ATTRIBUTE_NOT_EDITABLE

Cause: The object is partially fetched (using fetch group), the unfetched attribute is not editable.

Action: Do not edit the unfetched attribute, or explicitly fetch the attribute before editing it.

7138: OBJECT_NEED_IMPL_TRACKER_FOR_FETCH_GROUP_USAGE

Cause: The object must implement `FetchGroupTracker` in order to use `fetch group`.

Action: The domain call must implement `FetchGroupTracker` in order to use `fetch group`.

7139: UPDATE_ALL_QUERIES_NOT_SUPPORTED_WITH_OTHER_WRITES

Cause: Update all queries cannot be issued within a `UnitOfWork` containing other write operations.

Action: Do not use `UpdateAllQuery` within a `UnitOfWork` containing other write operations.

7140: WRONG_SEQUENCE_TYP

Cause: Sequence type does not have method.

Action: Do not call this method on this type of sequence.

7144: PLATFORM_DOES_NOT_SUPPORT_SEQUENCE

Cause: Platform does not support sequence.

Action: Do not use this sequence type on this platform.

7145: SEQUENCE_CANNOT_BE_CONNECTED_TO_TWO_PLATFORMS

Cause: Two attempts have been made to connect to sequence, but it is already connected to one. Likely the two sessions share the `DatasourcePlatform` object.

Action: Ensure that the sequence is used by a single session only.

7146: QUERY_SEQUENCE_DOES_NOT_HAVE_SELECT_QUERY

Cause: `QuerySequence` does not have select query.

Action: Ensure that the sequence has a select query.

7147: CREATE_PLATFORM_DEFAULT_SEQUENCE_UNDEFINED

Cause: Platform cannot create platform default sequence - it doesn't override the `createPlatformDefaultSequence` method.

Action: You must either override the `createPlatformDefaultSequence` method on the platform or explicitly set default sequence by calling `setDefaultSequence` on `DatasourceLogin`.

EJB QL Exceptions (8001 – 8010)

`EJBQLException` is a run-time exception that is raised when the EJB QL string does not parse properly, or the contents cannot be resolved within the context of the `TopLink` session. The associated message typically includes a reference to the EJB QL string that caused the problem.

Format

EXCEPTION [TOPLINK - error code]: Exception name

EXCEPTION DESCRIPTION: Message

Example 13–8 EJB QL Exception

EXCEPTION [TOPLINK - 8002]: oracle.toplink.exceptions.EJBQLException

EXCEPTION DESCRIPTION: `TopLink` has encountered a problem while parsing the EJB QL string.

8001: recognitionException

Cause: The TopLink EJB QL parser does not recognize a clause in the EJB QL string.

Action: Validate the EJB QL string.

8002: generalParsingException

Cause: TopLink has encountered a problem while parsing the EJB QL string.

Action: Check the internal exception for details on the root cause of this exception.

8003: classNotFoundException

Cause: The class specified in the EJB QL string was not found.

Action: Ensure that the class is on the appropriate classpath.

8004: aliasResolutionException

Cause: TopLink was unable to resolve the alias used in the EJB QL string.

Action: Validate the identifiers used in the EJB QL string.

8005: resolutionClassNotFoundException

Cause: TopLink was unable to resolve the class for an alias. This means that the class specified cannot be found.

Action: Ensure that the class is specified properly and is on the classpath.

8006: missingDescriptorException

Cause: The class specified in the query has no TopLink descriptor.

Action: Ensure that the class has been mapped and is specified correctly in the EJB QL string.

8009: expressionNotSupported

Cause: An unsupported expression was used in the EJB QL.

Action: Change the query to use only supported expressions.

8010: generalParsingException2

Cause: TopLink has encountered a problem while parsing the EJB QL string.

Action: Check the internal exception for details on the root cause of this exception.

Session Loader Exceptions (9000 - 9010)

`SessionLoaderException` is a run-time exception that is raised if the session manager encounters a problem loading session information from a `sessions.xml` (for non-EJB applications) or `toplink-ejb-jar.xml` (for EJB applications) properties file.

Format

EXCEPTION [TOPLINK - error code]: Exception name

EXCEPTION DESCRIPTION: Message

Example 13–9 Session Loader Exception

EXCEPTION [TOPLINK - 9004]: oracle.toplink.exceptions.SessionLoaderException
 EXCEPTION DESCRIPTION: The <project-xml> file MyProject was not found on the classpath, nor on the filesystem.

9000: FINAL_EXCEPTION

Cause: The session loader caught one or more XML parsing exceptions while loading session information. The specific XML exceptions follow.

Action: Verify your session configuration XML file.

9001: UNKNOWN_TAG

Cause: An unknown tag was encountered in the specified XML node.

Action: Examine the specified XML node in your session configuration XML file. Ensure that you use only the tags defined for that node in the appropriate TopLink XSD. See the directory where you installed TopLink (e.g. <ORACLE_HOME>/toplink/config/xsds)

9002: UNABLE_TO_LOAD_PROJECT_CLASS

Cause: The specified class loader could not load a class with the name given by the project-name property.

Action: Verify the value of the project-name property and if correct, ensure that a class with that name is in your classpath.

9003: UNABLE_TO_PROCESS_TAG

Cause: The session loader caught an exception while either parsing the value of the specified tag or calling the set-method associated with the specified tag.

Action: Verify the value shown for the specified tag.

9004: COULD_NOT_FIND_PROJECT_XML

Cause: The session loader could not find the file identified by the project-xml tag on either the classpath or the file system.

Action: Verify the value of the project-xml tag and if correct, ensure that a project XML file with that name exists in your classpath or file system.

9005: FAILED_TO_LOAD_PROJECT_XML

Cause: The session loader caught an exception while trying to load the file identified by the project-xml tag either because the file could not be found or because the file could not be parsed.

Action: Verify the configuration of the project XML file and ensure that a project XML file with that name specified by the project-xml tag exists in your classpath or file system.

9006: UNABLE_TO_PARSE_XML

Cause: The session loader caught a SAX exception while trying to parse the XML at the given line and column of the specified XML file. Oracle TopLink 10g supports only UTF-8 encoding. The TopLink SAXParseException occurs if you attempt to read a non-UTF-8 formatted XML file.

Action: Verify that the XML is correctly formatted at the given line and column. Alternatively, ensure the Oracle parser is in your classpath and that it appears before any other XML parser.

9007: NON_PARSE_EXCEPTION

Cause: The session loader caught an exception unrelated to XML parsing (for example, a premature end-of-file exception) while trying to parse the specified XML file.

Action: Verify the integrity of the XML file.

9008: UN_EXPECTED_VALUE_OF_TAG

Cause: The value of an XML tag does not correspond to any known TopLink required values.

Action: Please verify the list of values for this tag.

9009: UNKNOWN_ATTRIBUTE_OF_TAG

Cause: There is an incorrect name value pair when processing transport properties for the XSD tag.

Action: Please verify that all properties have both the name and the value filled in, in the session configuration XML file.

9010: XML_SCHEMA_PARSING_ERROR

Cause: An exception was raised while parsing the XML file against the XML schema.

Action: Examine the exception and take the appropriate action.

EJB Exception Factory Exceptions (10001 - 10069)

An EJB exception factory generates run-time exceptions that are raised if a container provider specific to a given application server encounters a problem during any stage of the life cycle of an EJB bean.

Format

EXCEPTION [TOPLINK - error code]: Exception name
EXCEPTION DESCRIPTION: Message

Example 13-10 EJB Exception Factory Exception

EXCEPTION [TOPLINK - 10008]: javax.ejb.CreateException
EXCEPTION DESCRIPTION: Cannot find bean.

10001: CREATE_EXCEPTION

Cause: The `PersistenceManager` for the given application server failed to create an EJB bean (for example, a problem was encountered during the create process, such as a `NullPointerException`).

Action: Check the exception contained in the `javax.ejb.CreateException` for additional information.

10002: REMOVE_EXCEPTION

Cause: The `PersistenceManager` for the given application server failed to remove an EJB bean (for example, a problem was encountered during the remove, such as a `NullPointerException`).

Action: Check the exception contained in the `javax.ejb.RemoveException` for additional information.

10003: EJB_EXCEPTION

Cause: An internal, unexpected exception was raised.

Action: See the exception message provided.

10004: FINDER_EXCEPTION1

Cause: An unexpected exception was encountered while executing the finder method.

Action: See the exception message provided.

10007: DUPLICATE_KEY_EXCEPTION

Cause: The `PersistenceManager` for a given application server failed to create an EJB bean, because an EJB bean with the given primary key already exists.

Action: Verify the application logic to ensure the primary key is unique.

10008: OBJECT_NOT_FOUND_EXCEPTION

Cause: A scalar finder (one that returns a single object) was invoked on a home interface, and returned a `null` value.

Action: Verify the application logic to ensure the desired EJB bean exists.

10009: OBJECT_NOT_FOUND_PKEY_EXCEPTION

Cause: A finder method using the primary key indicated, returned a `null` value.

Action: Verify the application logic to ensure the desired EJB exists.

10010: CANNOT_CREATE_READ_ONLY

Cause: An attempt was made to create an entity marked as read-only using `session().getProject().setDefaultReadOnlyClasses(aVector)`. You cannot create a read-only entity.

Action: Read-only entities should be read from the database (not created by the home interface). Adjust the application to read the required entities beforehand.

10011: CANNOT_REMOVE_READ_ONLY

Cause: An attempt was made to delete an entity marked as read-only using `session().getProject().setDefaultReadOnlyClasses(aVector)`. You cannot delete a read-only entity.

Action: Determine whether or not the object should be read-only or not. If it should, do not try to remove it.

10014: ERROR_IN_NON_TX_COMMIT

Cause: The `PersistenceManager` for a given application server failed to end a local transaction (made up of a non-synchronized, non-JTA `UnitOfWork`) after a remove, create, business method, or home method invocation.

Action: See the exception message provided.

10021: ERROR_ASSIGNING_SEQUENCES

Cause: The `PersistenceManager` for a given application server, whose `shouldAssignSequenceNumbers` method returns `true`, failed to assign a sequence number to an entity.

Action: See the exception message provided.

10022: LIFECYCLE_REMOTE_EXCEPTION

Cause: A `java.rmi.RemoteException` was raised when an entity was activated, loaded, passivated, or stored.

Action: See the exception message provided.

10023: SEQUENCE_EXCEPTION

Cause: An exception was raised while handling a post-insert `DescriptorEvent` preventing the specified entity from being assigned a primary key.

Action: See the exception message provided.

10025: INTERNAL_ERROR_ACCESSING_CTX

Cause: Internal error.

Action: Contact Oracle Support Services if required.

10026: INTERNAL_ERROR_FINDING_GENSUBCLASS

Cause: Internal error.

Action: Contact Oracle Support Services.

10027: INTERNAL_ERROR_INITIALIZING_CTX

Cause: Internal error.

Action: Contact Oracle Support Services.

10028: INTERNAL_ERROR_INVALID_MAPPING

Cause: The `registerOrMergeAttribute` method of the `SessionAccessor`, called from within an EJB setter method, failed to obtain a `DatabaseMapping` for the given attribute from the `PersistenceManager`.

Action: Verify that the given attribute belongs to the EJB class and if it does, verify that a mapping exists for it.

10029: INTERNAL_ERROR_ACCESSING_PK

Cause: Failed to wrap an EJB bean for return to the application because the attempt to extract the primary key from the bean failed.

Action: See the exception message provided.

10030: INTERNAL_ERROR_ACCESSING_PKFIELD

Cause: Failed to initialize primary key fields due to `java.lang.NoSuchFieldException`.

Action: See the exception message provided.

10031: INTERNAL_ERROR_PREPARING_BEAN_INVOKE

Cause: One of the following failed with an exception other than `javax.ejb.ObjectNotFoundException`: a conforming find using the same query as a find by primary key; an Oracle Containers for J2EE `startCall` method invocation for a `BUSINESS_METHOD` operation; or a `WebLogic preInvoke` method invocation.

Action: See the exception message provided

10032: FINDER_NOT_IMPLEMENTED

Cause: Associated finder method has no implementation.

Action: Provide an implementation for the finder method.

10033: FINDER_FINDBYPK_NULLPK

Cause: A find by primary key was called with a `null` primary key value.

Action: Ensure the primary key is not `null` when the finder method is invoked

10034: REMOVE_NULLPK_EXCEPTION

Cause: A find by primary key was called with a `null` primary key value.

Action: Ensure the primary key is not null when the finder method is invoked.

10036: ERROR_DURING_CODE_GEN

Cause: The `PersistenceManager` for a given application server failed to generate a bean subclass.

Action: See the exception message provided.

10037: ERROR_EXECUTING_EJB_SELECT

Cause: An EJB select failed with an exception other than `javax.ejb.ObjectNotFoundException`.

Action: See the exception message provided.

10038: ERROR_EXECUTING_EJB_HOME

Cause: The invocation of a home interface method (excluding finder or create methods) failed.

Action: See the exception message provided.

10040: NO_ACTIVE_TRANSACTION

Cause: A create or remove EJB failed because the `PersistenceManager` does not have a transaction.

Action: Ensure your application has a transaction available. This may be a configuration problem related to your `ejb-jar.xml` file or an application logic problem in your client code.

10043: FINDER_RESULTS_ALREADY_WRAPPED

Cause: The results of a finder query could not be wrapped because they were already wrapped.

Action: If a redirect query is used, be sure to call the `setShouldUseWrapperPolicy(false)` method first.

10045: LOCAL_WRAPPER_MISSING

Cause: Error resolving the local interface.

Action: Double-check your local interface configuration.

10046: REMOTE_WRAPPER_MISSING

Cause: Error resolving the remote interface.

Action: Please double-check your remote interface configuration.

10047: CREATE_NULLPK_EXCEPTION

Cause: The `PersistenceManager` for a given application server failed to create a bean because the primary key was not defined.

Action: Make sure the primary key is defined properly, either in the application logic or through the sequence number configuration.

10049: INVALID_COLLECTION_PARAMETER_TYPE

Cause: Exception [EJB - {0}]: The argument is not of the correct type for the relationship. See section 10.3.6 of the EJB 2.1 specification.

Action: Check the parameter type to ensure that it is of the correct type for this collection.

10051: INVALID_TRANSACTION_STATE

Cause: Exception [EJB - {0}]: An attempt was made to access a `Collection` or `Iterator` outside a transaction or in one other than the one in which it was acquired. See section 10.3.8 of the EJB 2.1 specification.

Action: Check the CMR collection to determine whether it is being accessed in the same transaction.

10052: NO_SUCH_OBJECT_EXCEPTION

Cause: Exception [EJB - {0}]: Cannot find bean of type with primary key for remote method invocation.

Action: The exception will be raised if the `finder` method returns a `null` result during remote method invocation.

Action: Ensure that the bean has already existed before lookup.

10053: NO_SUCH_OBJECT_LOCAL_EXCEPTION

Cause: Exception [EJB - {0}]: Cannot find bean of type with primary key for local method invocation.

Action: This exception will be raised if the `finder` method returns a `null` result during local method invocation. Ensure that the bean already exists before lookup.

10054: INTERNAL_ERROR_ACCESSING_BEAN

Cause: Exception [EJB - {0}]: An internal error has occurred while accessing the bean.

Action: An internal exception occurred during create bean instance. Check the internal exception.

10055: ERROR_CASCADING_DELETE

Cause: Exception [EJB - {0}]: An error has occurred while cascading the delete from the bean to the relationship.

Action: An error occurred while performing cascade-delete of private owned objects. Check the underlying exception.

10056: ERROR_REL_MAINTENANCE

Cause: Exception [EJB - {0}]: An error has occurred while performing relationship maintenance for the bean with the relationship.

Action: Check the private-owned (cascade-delete) relationship in the domain object and also the underlying exception.

10057: MULTIPLE_RESULTS_FOR_SINGLE_FINDER

Cause: Exception [EJB - {0}]: Multiple results were found while executing the single object finder for the bean.

Action: Check if the query is passed properly and also the underlying TopLink query exception.

10058: CANNOT_UPDATE_READ_ONLY

Cause: Exception [EJB - {0}]: Unable to invoke method on bean because this bean is read only.

Action: Check the bean descriptor to see if the read-only flag was specified.

10059: CANNOT_MODIFY_ONEXONE_COLLECTIONSETTERS

Cause: Exception [EJB - {0}]: Unable to invoke method on bean because the target bean is read only. Either the target bean's object model (in the case of a

bidirectional relationship) or the database table would not be modified by this operation.

Action: Check the target bean descriptor for the 1:1 relationship in the domain object, to see if the read-only flag was specified.

10060: CANNOT_MODIFY_RELATIONSHIP_COLLECTION_SOURCE

Cause: Exception [EJB - {0}]: Unable to invoke method on collection in bean because the source bean is read only.

Action: Check the source bean descriptor for the 1:M and M:M relationship in the domain object, to see if the read-only flag was specified.

10061: CANNOT_MODIFY_RELATIONSHIP_COLLECTION_TARGET

Cause: Exception [EJB - {0}]: Unable to invoke method on collection in bean because the target bean is read only. Either the target bean's object model (in the case of a bidirectional relationship) or the database table will not be modified by this operation.

Action: Check the bidirectional target bean descriptor to see if the read-only flag was specified.

10062: UNABLE_TO_RETRIEVE_UNDERLYING_DATA

Cause: Exception [EJB - {0}]: Unable to retrieve underlying data for object primary keys. The transaction must be restarted and objects refreshed.

Action: The underlying object is garbage collected and removed from the cache. Restart the transaction and refresh all objects.

10063: UNKNOWN_COLLECTION_CLASS

Cause: Exception [EJB - {0}]: Unknown collection type passed to container policy.

Action: Ensure that the correct collection type is used.

10064: MODIFYING_RELATIONSHIPS_NEEDS_TX

Cause: Exception [EJB - {0}]: Any changes to fields must be performed within a transaction.

Action: Ensure that any changes are performed within a transaction.

10065: REL_MAINT_ADD_ERORR

Cause: Exception [EJB - {0}]: An error occurred while adding an object to a collection.

Action: An error occurred during relationship maintenance for the add method. Check the underlying exception.

10066: REL_MAINT_REMOVE_ERORR

Cause: Exception [EJB - {0}]: An error occurred while removing an object from a collection.

Action: An error occurred during relationship maintenance for the remove method. Check the underlying exception.

10067: REL_MAINT_ERORR

Cause: Exception [EJB - {0}]: An error occurred while performing relationship maintenance for a field.

Action: An error occurred on relationship maintenance for the cmr field. Check the underlying exception.

10068: MUST_USE_MANAGED_DS

Cause: Exception [EJB - {0}]: An error occurred with the data source. TopLink CMP requires use of a <managed-data-source>. A <native-data-source> cannot be used.

Action: Check the data-sources.xml file in OC4J. TopLink CMP requires the use of a <managed-data-source>

10069: ERROR_GETTING_FROM_JNDI

Cause: Exception [EJB - {0}]: Failed lookup of object under JNDI name.

Action: Check the JNDI name and also the underlying `javax.naming.NamingException`.

Communication Exceptions (12000 - 12003)

`CommunicationException` is a run-time exception that wraps all RMI, CORBA, or input and output exceptions that occur.

Format

EXCEPTION [TOPLINK - error code]: Exception name
EXCEPTION DESCRIPTION: Message

Example 13–11 Communication Exception

EXCEPTION [TOPLINK - 12000]: oracle.toplink.exceptions.CommunicationException
EXCEPTION DESCRIPTION: Error Sending connection service to myService.

12000: ERROR_SENDING_CONNECTION_SERVICE

Cause: Failed to add a connection to `CacheSynchronizationManager` or `RemoteCommandManager`.

Action: See the generated exception for the root cause.

12001: UNABLE_TO_CONNECT

Cause: `CacheSynchronizationManager` failed to connect to the specified service.

Action: See the generated exception for the root cause.

12002: UNABLE_TO_PROPAGATE_CHANGES

Cause: `CacheSynchronizationManager` failed to propagate changes to the specified service.

Action: See the generated exception for the root cause.

12003: ERROR_IN_INVOCATION

Cause: Error invoking a remote call.

Action: See the generated exception for the root cause.

XML Data Store Exceptions (13000 - 13020)

`XMLDataStoreException` is a run-time exception that is raised when TopLink is used to make objects persistent in the form of XML files (rather than using a relational database.)

Format

EXCEPTION [TOPLINK - error code]: Exception name
 EXCEPTION DESCRIPTION: Message

Example 13–12 XML Data Store Exception

EXCEPTION [TOPLINK - 13000]: oracle.toplink.xml.XMLDataStoreException
 EXCEPTION DESCRIPTION: File not found: C:\data\myTable\row.xml.

13000: FILE_NOT_FOUND

Cause: Failed to create a `WriteStream` for an XML file (an individual file or a file extracted from a zip archive) because the file could not be found in the file system. This can happen if the XML data accessor is trying to update an XML file and the file does not exist. This indicates an inconsistent state between the application and what is on disk.

Action: Verify that the specified file exists.

13001: UNABLE_TO_CLOSE_WRITE_STREAM

Cause: After writing a row to the XML data store, failed to close the `WriteStream` used due to a `java.io.IOException`. This can happen if the disk is full.

Action: See the generated exception for the root cause. Verify that there is sufficient disk space available for this operation.

13002: NOT_A_DIRECTORY

Cause: Creating or deleting a file source failed because the file being created or deleted was not a directory or a file exists with the same name as the directory indicated.

Action: Verify that TopLink has permissions to create the necessary directories. Verify that there is sufficient disk space available for this operation.

13003: DIRECTORY_COULD_NOT_BE_CREATED

Cause: Checking or creating a file or document source failed because the `mkdir` method of the `File` failed to create the directory named by the specified abstract path name, including any necessary but nonexistent parent.

Action: Verify that TopLink has permission to create the necessary directories. Verify that there is sufficient disk space available for this operation.

13004: DIRECTORY_NOT_FOUND

Cause: Directory does not exist and TopLink has not been set to create directories as needed (`createsDirectoriesAsNeeded` policy is `false`.)

Action: Either create the appropriate directory or configure TopLink to create directories as needed (set `createsDirectoriesAsNeeded` `true`).

13005: FILE_ALREADY_EXISTS

Cause: TopLink is attempting to create a file and the file already exists. TopLink expects to be able to create a new version of the file and will not overwrite an existing file. This can happen if the XML data accessor is trying to insert an XML file and the file already exists. This indicates an inconsistent state between the application and what is on disk.

Action: Change where TopLink is writing to, or remove the existing file.

13006: UNABLE_TO_CREATE_WRITE_STREAM

Cause: Failed to create a `WriteStream` due to a `java.io.IOException`.

Action: See the generated exception for the root cause.

13007: INVALID_FIELD_VALUE

Cause: Failed to construct an XML element to represent an object because the object was an invalid type. For a direct collection, one or more of the elements had a type that was not `null` or a `String`. For a nested row, one or more of the elements had a type that was not `DatabaseRow`.

Action: See the generated exception for the root cause. Verify the configuration of the persistent object to ensure that it can be made persistent in an XML data store.

13008: CLASS_NOT_FOUND

Cause: Failed to load the specified class due to a `java.lang.ClassNotFoundException`. This indicates a problem either with the TopLink JAR file or an improperly configured custom class loader (see the `setXMLParserJARFileNames` method of the `DatabaseLogin`).

Action: See the generated exception for the root cause. Confirm that the TopLink JAR file contains the specified class. If you are using a custom class loader, confirm that this class is included in the list of JAR files passed into the `setXMLParserJARFileNames` method of the `DatabaseLogin`.

13009: SAX_PARSER_ERROR

Cause: Failed to parse the specified XML file due to an `org.xml.sax.SAXParseException`.

Action: See the generated exception for the root cause including the line and column numbers at which the `SAXParseException` was raised.

13010: GENERAL_EXCEPTION

Cause: An operation failed due to something other than an `org.xml.sax.SAXParseException`.

Action: An exception was raised when trying either to build a parser or to build a document that caused that action to fail. See the generated exception for the root cause.

13011: IOEXCEPTION

Cause: A `ReadStream` or `WriteStream` could not be created due to a `java.io.IOException`.

Action: See the generated exception for the root cause.

13012: UNABLE_TO_CLOSE_READ_STREAM

Cause: After reading a row from the XML data store, failed to close the `ReadStream` used due to a `java.io.IOException`.

Action: See the generated exception for the root cause.

13013: HETEROGENEOUS_CHILD_ELEMENTS

Cause: Composite elements are being stored in a `DirectCollectionMapping`. `DirectCollectionMappings` in the SDK will work only with simple elements. Simple elements contain only one child of type text in XML.

Action: Ensure elements that are mapped as direct collections only contain simple elements.

Cause: Child elements of a complex element are not the same type: the type of each child element must be the same as that of the first child element.

Action: Verify that the XML document is not corrupt. If it is valid, ensure that it meets SDK requirements as illustrated in [Example 13-13](#) and [Example 13-14](#).

Cause: Child elements of a complex element do not have the same name.

Action: Verify that the XML document is not corrupt. If it is valid, ensure that it meets SDK requirements as in [Example 13-13](#) and [Example 13-14](#).

Example 13-13 XML Supported by the SDK

```
<foo>
  <bar> [string or nested elements] </bar>
  <bar> [must match the first child: either string or nested elements] </bar>
  <bar> [must match the first child: either string or nested elements] </bar>
</foo>
```

Example 13-14 XML Not Supported by the SDK

```
<foo>
  <bar> ... </bar>
  <fred> [this element will cause the exception] </fred>
  <bar> ... </bar>
</foo>
```

13017: INSTANTIATION_EXCEPTION

Cause: Failed to instantiate the specified class due to a `java.lang.InstantiationException`. This indicates a problem either with the TopLink JAR file or an improperly configured custom class loader (see the `setXMLParserJARFileNames` method of the `DatabaseLogin`).

Action: See the generated exception for the root cause. Ensure that the specified class is not an interface or an abstract class. Confirm that the TopLink JAR contains the specified class. If you are using a custom class loader, confirm that this class is included in the list of JAR files passed into the `setXMLParserJARFileNames` method of the `DatabaseLogin`.

13018: INSTANTIATION_ILLEGAL_ACCESS_EXCEPTION

Cause: Failed to instantiate the specified class due to a `java.lang.IllegalAccessException`. This indicates a problem either with the TopLink JAR file or an improperly configured custom class loader (see the `setXMLParserJARFileNames` method of the `DatabaseLogin`).

Action: See the generated exception for the root cause. Ensure that the specified class is public. Ensure permission is set for Java reflection in your VM security settings. Ensure that the specified class is not an interface or an abstract class. Confirm that the TopLink JAR file contains the specified class. If you are using a custom class loader, confirm that this class is included in the list of JAR files passed into the `setXMLParserJARFileNames` method of the `DatabaseLogin`.

13020: ELEMENT_DATA_TYPE_NAME_IS_REQUIRED

Cause: Failed to build XML for a given object because the object's data type name is null or zero length.

Action: Ensure that the element data type name is provided.

Deployment Exceptions (14001 - 14033)

`DeploymentException` is a run-time exception that is raised if problems are detected during deployment of an EJB bean. During deployment, project, sessions, and

`ejb-jar.xml` files (or their Java Class equivalents) are read, and the necessary objects are instantiated and initialized.

Format

EXCEPTION [TOPLINK - error code]: Exception name
EXCEPTION DESCRIPTION: Message

Example 13–15 Deployment Exception

EXCEPTION [TOPLINK - 14001]: oracle.toplink.ejb.DeploymentException
EXCEPTION DESCRIPTION: No TopLink project was specified for this bean.

14001: NO_PROJECT_SPECIFIED

Cause: Neither project name nor class could be read from the deployment descriptor.

Action: Verify your project configuration in your deployment descriptor. Double-check that either `project.xml` or `project-class` is specified.

14003: NO_SUCH_PROJECT_IDENTIFIER

Cause: No project exists with the identifier requested.

Action: Verify that the project name matches exactly the project name specified in your project XML file.

14004: ERROR_CREATING_CUSTOMIZATION

Cause: Could not create an instance of the `DeploymentCustomization` class.

Action: Verify the implementation of the class implementing the `DeploymentCustomization` interface. Start with the constructor, then proceed to the remainder of the implementation.

14005: ERROR_RUNNING_CUSTOMIZATION

Cause: An exception was raised when either the `afterLoginCustomization` method or the `beforeLoginCustomization` method of the `DeploymentCustomization` was called.

Action: See the generated exception for the root cause. Verify the implementation of your `DeploymentCustomization` method.

14011: ERROR_CONNECTING_TO_DATA_SOURCE

Cause: The data source could not be located in JNDI, or was not properly specified.

Action: Verify the data source attribute of the login element in your `sessions.xml` file. Ensure that the data source is present and properly configured.

14016: ERROR_CREATING_PROJECT

Cause: The project XML file name or class was specified, but a general error occurred creating the project.

Action: See the generated exception for the root cause. Verify your `project.xml` file.

14020: ERROR_IN_DEPLOYMENT_DESCRIPTOR

Cause: Error parsing the `toplink-ejb-jar.xml` file.

Action: See the generated exception for the root cause. Verify your `toplink-ejb-jar.xml` file.

14023: CANNOT_FIND_GENERATED_SUBCLASS

Cause: An internal, unexpected exception was raised.

Action: See the exception message provided.

14024: CANNOT_READ_TOPLINK_PROJECT

Cause: An internal, unexpected exception was raised while reading the project.

Action: See the exception message provided.

14026: MUST_USE_TRANSPARENT_INDIRECTION

Cause: Your project contains either a one-to-many or many-to-many relationship (between EJB 2.0 entity beans) that is not using transparent indirection.

Action: Verify your project is using transparent indirection for all one-to-many and many-to-many relationships involving EJB 2.0 entity beans.

14027: MUST_USE_VALUEHOLDER

Cause: Your project contains a one-to-one relationship (between EJB 2.0 entity beans) that is not using basic indirection.

Action: Verify your project is using basic indirection for all one-to-one relationships involving EJB 2.0 entity beans.

14028: NO_SUCH_MAPPING

Cause: The specified descriptor does not contain a corresponding mapping for the specified container managed attribute.

Action: Verify the descriptor and ensure that you configure a mapping for the specified container managed attribute.

14029: MISSING_FINDER_DEF

Cause: The specified finder is declared in the `ejb-jar.xml` file but not defined on the specified home interface.

Action: Verify the specified home interface and ensure that you define the specified finder in it.

14030: MISSING_EJB_SELECT_DEF

Cause: The specified `ejbSelect` method is declared in the `ejb-jar.xml` file but not defined on the specified abstract bean class.

Action: Verify the specified abstract bean class and ensure that you define the specified `ejbSelect` method on it.

14031: MISSING_11_CMP_FIELD

Cause: The specified EJB 1.1 CMP field is declared in the `ejb-jar.xml` file but not defined on the specified bean.

Action: Verify the specified bean class and ensure that you define the specified CMP field in it.

14032: MISSING_20_CMP_FIELD

Cause: The specified EJB 2.0 CMP field is declared in the `ejb-jar.xml` file but the corresponding abstract getter and/or setter is not defined on the specified abstract bean class.

Action: Verify the specified abstract bean class and ensure that you define the specified abstract getter and/or setter on it.

14033: MISSING_DESCRIPTOR

Cause: The descriptor for the specified class is missing.

Action: Verify your TopLink project.xml and ensure that you create and configure a descriptor for the specified class.

Synchronization Exceptions (15001 - 15025)

`SynchronizationException` is a run-time exception that is raised when a cache coordination update by TopLink to a distributed session is unsuccessful. When this occurs, the message contains a reference to the error code and error message.

Format

EXCEPTION [TOPLINK - error code]: Exception name
EXCEPTION DESCRIPTION: Message

Example 13–16

```
EXCEPTION [TOPLINK - 15001]:  
oracle.toplink.exceptions.TopLinkException.SynchronizationException  
EXCEPTION DESCRIPTION: Warning: Unable to send changes to distributed session.
```

15001: UNABLE_TO_PROPAGATE_CHANGES

Cause: Warning: Unable to send changes to distributed session.

Action: Verify that the remote server is available for synchronization. Verify the connection configuration specified for the clustering service.

15010: ERROR_LOOKING_UP_LOCAL_HOST

Cause: Could not look up local host.

Action: Verify that the local network interfaces are configured correctly.

15011: ERROR_BINDING_CONTROLLER

Cause: A `java.io.IOException` occurred when attempting to register the remote service.

Action: Verify that the naming service is available and that the chosen clustering service is configured correctly.

15012: ERROR_LOOKING_UP_CONTROLLER

Cause: Unable to find remote server's remote service.

Action: Verify that the naming service is still available and that the remote server has access permissions.

15013: ERROR_LOOKING_UP_JMS_SERVICE

Cause: Unable to find the specified JMS service.

Action: Verify server configuration of JMS service is correct and is configured for the JMS Clustering Service. Ensure that the IP address and port of the JMS service has been specified correctly in the sessions configuration file and that the service is running.

15014: ERROR_UNMARSHALLING_MSG

Cause: Could not unmarshall received session announcement.

Action: Verify that the sending server is still operating. You may need to restart the sending server to ensure synchronization.

15016: ERROR_GETTING_SYNC_SERVICE

Cause: An error occurred when attempting to initialize the synchronization service as specified in the sessions configuration file.

Action: Verify that the service has been properly configured in the sessions configuration file.

15017: ERROR_NOTIFYING_CLUSTER

Cause: An error occurred when attempting to contact other TopLink sessions.

Action: Verify that multicast support is available on your network.

15018: ERROR_JOINING_MULTICAST_GROUP

Cause: An error occurred when attempting to join the multicast group of a TopLink coordinated cache.

Action: Verify that multicast support is available on your network.

15023: ERROR_RECEIVING_ANNOUNCEMENT

Cause: A `java.io.IOException` occurred when attempting to receive a session existence announcement from a remote TopLink Session.

Action: Verify that the sending server is still operating. You may need to restart the sending server to ensure synchronization.

15025: FAIL_TO_RESET_CACHE_SYNC

Cause: The API on the development services called to reset the TopLink cache coordination, including participation in the coordinated cache, failed.

Action: Inspect the exception message and respond appropriately. The server may need to be restarted.

SDK Data Store Exceptions (17001 - 17006)

`SDKDataStoreException` is a run-time exception that is raised when SDK classes are used to customize TopLink.

Format

EXCEPTION [TOPLINK - error code]: Exception name

EXCEPTION DESCRIPTION: Message

Example 13–17 SDK Data Store Exception

EXCEPTION [TOPLINK - 17001]: `oracle.toplink.sdk.SDKDataStoreException`
EXCEPTION DESCRIPTION: The TopLink SDK does not currently support Cursor.

17001: UNSUPPORTED

Cause: A method call failed because it is not currently supported by the SDK.

Action: Avoid using the unsupported method.

17002: INCORRECT_LOGIN_INSTANCE_PROVIDED

Cause: An instance of `oracle.toplink.sdk.SDKAccessor` was passed the wrong type of `Login` (the SDK expects an instance of `DatabaseLogin`).

Action: Verify that your SDK-based application is being passed the expected type of Login.

17003: INVALID_CALL

Cause: When the `QueryManager` owned by an `SDKDescriptor` is initialized, an instance of `InvalidSDKCall` is set for each type of `Call` that is not configured. If you invoke an unconfigured `Call`, this `INVALID_CALL` error is logged rather than simply throwing a `NullPointerException` because the `INVALID_CALL` error contains more information.

Action: Avoid using the unconfigured `Call` or provide a `Call` implementation in your SDK-based application.

17004: IE_WHEN_INSTANTIATING_ACCESSOR

Cause: Failed to instantiate the specified class due to a `java.lang.InstantiationException`.

Action: Ensure that the specified class is not an interface or an abstract class.

17005: IAE_WHEN_INSTANTIATING_ACCESSOR

Cause: Failed to instantiate the specified class due to a `java.lang.IllegalAccessException`.

Action: Ensure that specified class is public. Ensure permission is set for Java reflection in your VM security settings.

17006: SDK_PLATFORM_DOES_SUPPORT_SEQUENCES

Cause: Unsupported `SDKPlatform` methods `buildSelectSequenceCall` or `buildUpdateSequenceCall` were called.

Action: Avoid using these methods or subclass `SDKPlatform` and override them with your own implementation.

EIS Exceptions (17007 – 17025)

`EISException` is a run-time exception that is raised when invoking EIS interactions. For more information on EIS interactions, see ["Enterprise Information System \(EIS\) Interactions"](#) on page 96-18.

Format

EXCEPTION [TOPLINK - error code]: Exception name
EXCEPTION DESCRIPTION: Message

Example 13-18 JMS Processing Exception

EXCEPTION [TOPLINK - 17010]: oracle.toplink.eis.EISException
EXCEPTION DESCRIPTION: Output record contains an unsupported message type.

17007: PROP_NOT_SET

Cause: The specified property must be set.

Action: Verify your interaction and ensure that the specified property is set (see ["Configuring Custom EIS Interactions for Basic Persistence Operations"](#) on page 31-6 or ["Creating an EIS Interaction for a Named Query"](#) on page 28-20).

17008: INVALID_PROP

Cause: Invalid property encountered.

Action: Verify your interaction and remove the specified property (see ["Configuring Custom EIS Interactions for Basic Persistence Operations"](#) on page 31-6 or ["Creating an EIS Interaction for a Named Query"](#) on page 28-20).

17009: PROPS_NOT_SET

Cause: The specified properties must be set.

Action: Verify your interaction and ensure that the specified properties are set (see ["Configuring Custom EIS Interactions for Basic Persistence Operations"](#) on page 31-6 or ["Creating an EIS Interaction for a Named Query"](#) on page 28-20).

17010: OUTPUT_UNSUPPORTED_MSG_TYPE

Cause: The output record contains an unsupported message type.

Action: Verify your interaction and ensure that you specify a supported message type. Ensure that the required connector JAR file is on your classpath (see ["EIS Projects"](#) on page 20-7).

17011: NO_CONN_FACTORY

Cause: No connection factory has been specified.

Action: Verify your interaction and ensure that you specify a connection factory (see ["Configuring EIS Connection Specification Options at the Project Level"](#) on page 24-2 or ["Configuring EIS Connection Specification Options at the Session Level"](#) on page 87-2).

17012: INVALID_INTERACTION_SPEC_TYPE

Cause: InteractionSpec must be a CciJMSInteractionSpec.

Action: Verify your interaction and ensure that you specify a valid interaction specification type. Ensure that the required connector JAR file is on your classpath (see ["EIS Projects"](#) on page 20-7).

17013: INVALID_RECORD_TYPE

Cause: Record must be a CciJMSRecord.

Action: Verify your interaction and ensure that you specify a valid record type. Ensure that the required connector JAR file is on your classpath (see ["EIS Projects"](#) on page 20-7).

17014: UNKNOWN_INTERACTION_SPEC_TYPE

Cause: Unknown interaction specification type.

Action: Verify your interaction and ensure that you specify a valid interaction specification type. Ensure that the required connector JAR file is on your classpath (see ["EIS Projects"](#) on page 20-7).

17015: INVALID_INPUT

Cause: Input must contain a single text element.

Action: Verify your interaction and ensure that you specify valid input (see ["Configuring Custom EIS Interactions for Basic Persistence Operations"](#) on page 31-6 or ["Creating an EIS Interaction for a Named Query"](#) on page 28-20).

17016: TIMEOUT

Cause: A time-out occurred—no message was received.

Action: Verify your interaction and the EIS you invoked it on.

17017: INPUT_UNSUPPORTED_MSG_TYPE

Cause: Input record contains an unsupported message type.

Action: Verify your interaction and ensure that you specify a valid message type. Ensure that the required connector JAR file is on your classpath (see ["EIS Projects"](#) on page 20-7).

17018: INVALID_METHOD_INVOCATION

Cause: Cannot invoke begin method on a non-transacted session.

Action: To be determined.

17019: TX_SESSION_TEST_ERROR

Cause: Problem testing for transacted session.

Action: To be determined.

17020: INVALID_AQ_INTERACTION_SPEC_TYPE

Cause: InteractionSpec must be an AQInteractionSpec.

Action: Verify your interaction and ensure that you specify a valid Oracle AQ interaction specification type. Ensure that the required connector JAR file is on your classpath (see ["EIS Projects"](#) on page 20-7).

17021: INVALID_AQ_RECORD_TYPE

Cause: Record must be an AQRecord.

Action: Verify your interaction and ensure that you specify a valid Oracle AQ record type. Ensure that the required connector JAR file is on your classpath (see ["EIS Projects"](#) on page 20-7).

17022: INVALID_AQ_INPUT

Cause: Input must contain a single raw element.

Action: Verify your Oracle AQ interaction and ensure that you specify valid input (see ["Configuring Custom EIS Interactions for Basic Persistence Operations"](#) on page 31-6 or ["Creating an EIS Interaction for a Named Query"](#) on page 28-20).

17023: INVALID_FACTORY_ATTRIBUTES

Cause: An exception occurred setting MQQueueConnectionFactory attributes.

Action: Verify your interaction and ensure that you specify an appropriate IBM MQSeries connection factory (see ["Configuring EIS Connection Specification Options at the Project Level"](#) on page 24-2 or ["Configuring EIS Connection Specification Options at the Session Level"](#) on page 87-2).

17024: COULD_NOT_DELETE_FILE

Cause: Could not delete the specified file.

Action: To be determined.

17025: GROUPING_ELEMENT_REQUIRED

Cause: This mapping requires a foreign key grouping element, as multiple foreign keys exist.

Action: Verify your EIS mappings (see ["Understanding EIS Mappings"](#) on page 56-1).

JMS Processing Exceptions (18001 - 18002)

JMSProcessingException is a run-time exception that is raised when processing JMS messages.

Format

EXCEPTION [TOPLINK - error code]: Exception name
 EXCEPTION DESCRIPTION: Message

Example 13–19 JMS Processing Exception

EXCEPTION [TOPLINK - 18001]: oracle.toplink.exceptions.JMSProcessingException
 EXCEPTION DESCRIPTION: Error while processing incoming JMS message.

18001: DEFAULT

Cause: Failed to process the incoming JMS message.

Action: See the generated exception for the root cause.

18002: NO_TOPIC_SET

Cause: JMSClusteringService failed to start because the Topic created in the JMS service for the interconnection of sessions is null.

Action: Ensure that the Topic created in the JMS service for the interconnection of sessions is set in the JMSClusteringService.

SDK Descriptor Exceptions (19001 - 19003)

SDKDescriptorException is a run-time exception that is raised when you use SDK classes to customize TopLink descriptors.

Format

EXCEPTION [TOPLINK - error code]: Exception name
 EXCEPTION DESCRIPTION: Message

Example 13–20 SDK Descriptor Exception

EXCEPTION [TOPLINK - 19001]: oracle.toplink.sdk.SDKDescriptorException
 EXCEPTION DESCRIPTION: The TopLink SDK does not currently support query result ordering.

19001: UNSUPPORTED

Cause: A method call failed because it is not currently supported by the SDK.

Action: Avoid using the unsupported method.

19002: CUSTOM_SELECTION_QUERY_REQUIRED

Cause: An SDKObjectCollectionMapping was used without a custom selection query.

Action: Set the custom selection query on the SDKObjectCollectionMapping.

19003: SIZE_MISMATCH_OF_FIELD_TRANSLATIONS

Cause: Mapping field name array and data source field name array are of different lengths.

Action: The sizes of the field translation arrays must be equal.

Default Mapping Exceptions (20001 - 20008)

DefaultMappingException is a run-time exception that is raised when an error occurs during OC4J CMP default mapping.

Format

EXCEPTION [TOPLINK - error code]: Exception name
EXCEPTION DESCRIPTION: Message

Example 13–21 Default Mapping Exception

EXCEPTION [TOPLINK - 20002]: oracle.toplink.exceptions.DefaultMappingException
EXCEPTION DESCRIPTION: The finder method with the parameters as defined in the ejb-jar.xml file, is not found in the home of bean.

20001: FINDER_PARAMETER_TYPE_NOT_FOUND

Cause: Could not find the parameter type, defined in the ejb-jar.xml file, of the finder in the entity bean.

Action: Check the finder definition in the ejb-jar.xml file and the bean home to ensure that the named finder has the specified method parameter type(s).

20002: FINDER_NOT_DEFINED_IN_HOME

Cause: The finder method with the parameters as defined in the ejb-jar.xml file, is not found in the home of bean.

Action: Check the ejb-jar.xml file and the bean home to ensure that the finder with the specified name and method parameter type(s) are defined.

20003: EJB_SELECT_NOT_DEFINED_IN_BEAN

Cause: The ejbSelect method with the parameters, as defined in the ejb-jar.xml file, is not found in the bean class.

Action: Check the ejb-jar.xml file and the bean class to ensure that the ejbSelect method is defined with the specified name and method parameter type(s).

20004: FINDER_NOT_START_WITH_FIND_OR_EJBSELECT

Cause: The finder method of bean in the ejb-jar.xml file is not well defined. It should start with either find or ejbSelect.

Action: Check the ejb-jar.xml file to ensure that the query name starts with find or ejbSelect, according to the EJB specification.

20005: GETTER_NOT_FOUND

Cause: The abstract get method is not defined in the bean.

Action: Check the ejb-jar.xml file and the bean class to ensure that the abstract get method is well defined.

20006: FIELD_NOT_FOUND

Cause: The CMP field is not defined in the bean.

Action: Check the ejb-jar.xml file and the bean class to ensure that the CMP field defined in the ejb-jar.xml file is also defined in the bean class.

20007: MAX_TRIES_EXCEEDED_FOR_LOCK_ON_CLONE

Cause: During an attempt to clone the specified object, the maximum number of attempts to lock the specified object was exceeded. TopLink failed to clone the object.

Action: Consider your locking strategy (see "[Understanding Descriptors and Locking](#)" on page 26-17).

20008: MAX_TRIES_EXCEEDED_FOR_LOCK_ON_MERGE

Cause: During an attempt to merge changes to the specified object, the maximum number of attempts to lock the specified object was exceeded. TopLink failed to merge the transaction.

Action: Consider your locking strategy (see "[Understanding Descriptors and Locking](#)" on page 26-17).

Discovery Exceptions (22001 - 22004)

DiscoveryException is a run-time exception that is raised when DiscoveryManager is operating.

Format

EXCEPTION [TOPLINK - error code]: Exception name
EXCEPTION DESCRIPTION: Message

Example 13–22 Discovery Exception

EXCEPTION [TOPLINK - 22001]: oracle.toplink.exception.DiscoveryException
EXCEPTION DESCRIPTION: Could not join multicast group.

22001: ERROR_JOINING_MULTICAST_GROUP

Cause: DiscoveryManager failed to join a multicast group due to a `java.io.IOException`: either a `MulticastSocket` could not be created or the invocation of the `joinGroup` method of the `MulticastSocket` failed.

Action: See the generated exception for the root cause.

22002: ERROR_SENDING_ANNOUNCEMENT

Cause: DiscoveryManager failed to inform other services that its service had started up.

Action: Consider increasing the announcement delay: the amount of time in milliseconds that the service should wait between the time that this remote service is available and a session announcement is sent out to other discovery managers. This may be needed to give some systems more time to post their connections into the naming service. See the `setAnnouncementDelay` method of the `DiscoveryManager`.

22003: ERROR_LOOKING_UP_LOCAL_HOST

Cause: Failed doing lookup of local host.

Action:

22004: ERROR_RECEIVING_ANNOUNCEMENT

Cause: DiscoveryManager caught a `java.io.IOException` while blocking for announcements from other `DiscoveryManagers`.

Action: See the generated exception for the root cause.

Remote Command Manager Exceptions (22101 - 22111)

RemoteCommandManagerException is a run-time exception that is raised when the remote command module is used.

Format

EXCEPTION [TOPLINK - error code]: Exception name

EXCEPTION DESCRIPTION: Message

Example 13–23 Remote Command Manager Exception

EXCEPTION [TOPLINK - 22104]:
oracle.toplink.exceptions.RemoteCommandManagerException
EXCEPTION DESCRIPTION: Could not look up host name.

22101: ERROR_OBTAINING_CONTEXT_FOR_JNDI

Cause: Failed to get a JNDI context with the specified properties due to a `javax.naming.NamingException`.

Action: See the generated exception for root cause. Verify that the properties for looking up the context are correct.

22102: ERROR_BINDING_CONNECTION

Cause: Failed to post a connection in the local naming service.

Action: See the generated exception for the root cause.

22103: ERROR_LOOKING_UP_REMOTE_CONNECTION

Cause: Failed to look up a remote connection with the specified name and URL.

Action: See the generated exception for the root cause. Verify that the remote connection and URL are correct.

22104: ERROR_GETTING_HOST_NAME

Cause: The `getLocalHost` method of the `java.net.InetAddress` failed to look up the specified host name.

Action: See the generated exception for the root cause. Verify that the host is online and reachable.

22105: ERROR_PROPAGATING_COMMAND

Cause: Failed to propagate a command to the specified connection.

Action: See the generated exception for the root cause. Verify that the remote host of the specified connection is online and reachable if the generated exception included a `CommunicationException`.

22106: ERROR_CREATING_JMS_CONNECTION

Cause: Could not create JMS connection with Topic, Topic Factory, and Context properties.

Action: Verify that the JMS Topic is configured correctly with the application server and the specified Topic, Topic Factory and Context properties info can be used by a Java client to the JMS Topic.

22107: ERROR_UNBINDING_LOCAL_CONNECTION

Cause: Could not remove local connection in local naming service under name.

Action: Restart the application server or use the application server tool to remove name from its JNDI if this tool is available.

22108: ERROR_SERIALIZE_OR_DESERIALIZE_COMMAND

Cause: Could not serialize or deserialize command.

Action: If the command is a user defined command, make sure it is serializable. If it is a TopLink command, file a bug report including the stack trace.

22109: ERROR_RECEIVING_JMS_MESSAGE

Cause: Failed to receive JMS message from JMS provider.

Action: Make sure that TopLink sessions are the only publishers to the JMS Topic and that the TopLink sessions use the same project.

22110: ERROR_DISCOVERING_IP_ADDRESS

Cause: Failed to discover local host IP address.

Action: Replace the \$HOST string of the URL with the known host name or IP address.

22111: ERROR_GETTING_SERVERPLATFORM

Cause: Failed to get server platform. The server platform must be set on `Session` or `RemoteCommandManager`.

Action: Set the correct `ServerPlatform` on the `RemoteCommandManager`. This exception is raised when the user does not use a TopLink Session and implements the `CommandProcessor`.

Transaction Exceptions (23001 - 23011)

`TransactionException` is a run-time exception that is raised when an error is encountered during a transaction. When this occurs, the message contains a reference to the error code and error message.

Format

EXCEPTION [TOPLINK - error code]: Exception name

EXCEPTION DESCRIPTION: Message

Example 13–24 Transaction Exception

EXCEPTION [TOPLINK - 23001]: oracle.toplink.exceptions.TransactionException
EXCEPTION DESCRIPTION: Error looking up external Transaction resource under JNDI name.

23001: ERROR_DOING_JNDI_LOOKUP

Cause: Error looking up external transaction resource under JNDI name.

Action: Examine the internal exception and take the appropriate action.

23002: ERROR_GETTING_TRANSACTION_STATUS

Cause: Error obtaining the status of the current externally-managed transaction.

Action: Examine the internal exception and take the appropriate action.

23003: ERROR_GETTING_TRANSACTION

Cause: Error obtaining the current externally managed transaction.

Action: Examine the internal exception and take the appropriate action.

23004: ERROR_BINDING_TO_TRANSACTION

Cause: Error obtaining the transaction manager.

Action: Examine the internal exception and take the appropriate action.

23005: ERROR_BEGINNING_TRANSACTION

Cause: Error binding to the externally managed transaction.

Action: Examine the internal exception and take the appropriate action.

23006: ERROR_COMMITTING_TRANSACTION

Cause: Error beginning a new externally managed transaction.

Action: Examine the internal exception and take the appropriate action.

23007: ERROR_ROLLING_BACK_TRANSACTION

Cause: Error committing the externally managed transaction.

Action: Examine the internal exception and take the appropriate action.

23008: ERROR_MARKING_TRANSACTION_FOR_ROLLBACK

Cause: Error rolling back the externally managed transaction.

Action: Examine the internal exception and take the appropriate action.

23009: ERROR_NO_TRANSACTION_ACTIVE

Cause: Error marking the externally managed transaction for rollback.

Action: Examine the internal exception and take the appropriate action.

23010: ERROR_INACTIVE_UOW

Cause: No externally managed transaction is currently active for this thread.

Action: Ensure that the transaction is still active.

23011: ERROR_OBTAINING_TRANSACTION_MANAGER

Cause: A `UnitOfWork` was rendered inactive before the associated externally managed transaction was complete.

Action: Examine the internal exception and take the appropriate action.

XML Conversion Exceptions (25001 - 25016)

`XMLConversionException` is a run-time exception that is raised when a conversion between `TopLink` instances and XML fails. This exception is used in cache coordination that uses XML change sets.

Format

EXCEPTION [TOPLINK - error code]: Exception name
EXCEPTION DESCRIPTION: Message

Example 13–25 XML Conversion Exception

```
EXCEPTION [TOPLINK - 25001]: oracle.toplink.exceptions.XMLConversionException  
EXCEPTION DESCRIPTION: Cannot create URL for file [\\FILE_SERVER\\command.xml].
```

25001: ERROR_CREATE_URL

Cause: Failed to create a URL for the specified file.

Action: Ensure that all specified XPath strings on mappings are valid and correct.

25002: INVALID_XPATH_INDEX_STRING

Cause: An integer index could not be parsed from this XPath string.

Action: Update any XPath strings that contain an index, and ensure that only a single integer value is contained between the square braces.

25003: MARSHAL_EXCEPTION

Cause: An error occurred marshalling the object.

Action: Check the nested exception to determine the cause of the problem. Typically, there is a problem with the object being written.

25004: UNMARSHAL_EXCEPTION

Cause: An error occurred unmarshalling the document.

Action: Check the nested exception to determine the cause of the problem. Typically, there is a problem with the XML document being read.

25005: VALIDATE_EXCEPTION

Cause: An error occurred validating the object.

Action: The XML generated from the object is invalid according to the schema. Ensure that any values set in the domain classes do not violate any schema constraints when marshalled.

25006: DEFAULT_ROOT_ELEMENT_NOT_SPECIFIED

Cause: A default root element was not specified for the XMLDescriptor mapped to the root-level object.

Action: Set a default root element on any XMLDescriptor that will be marshalled as a root-level object.

25007: DESCRIPTOR_NOT_FOUND_IN_PROJECT

Cause: A descriptor for a class was not found in the project.

Action: Make sure that any classes referenced through CompositeObject and/or CompositeCollection mappings have descriptors in the project.

25008: NO_DESCRIPTOR_WITH_MATCHING_ROOT_ELEMENT

Cause: A descriptor with a default root element was not found in the project.

Action: Set a default root element on any descriptor that maps to the root of a document. Make sure that the root element of the XML document being read in is mapped to a descriptor in the project.

25010: SCHEMA_REFERENCE_NOT_SET

Cause: A schema reference was not specified for the XMLDescriptor mapped to.

Action: Set a schema reference on the correct descriptor.

25011: NULL_ARGUMENT

Cause: A null argument was encountered.

Action: Ensure that no null parameters are passed into any marshal or unmarshal methods on XMLMarshaller.

25012: ERROR_RESOLVING_XML_SCHEMA

Cause: An error occurred resolving the XML Schema.

Action: Check any schema references and make sure that the schema can be accessed by the application at runtime.

25013: ERROR_SETTING_SCHEMAS

Cause: An error occurred while trying to set the schemas.

Action: An error was encountered while attempting to set schemas on an XML parser. Check the nested exception to determine the correct course of action.

25014: ERROR_INSTANTIATING_SCHEMA_PLATFORM

Cause: An error occurred while trying to instantiate the schema platform.

Action: An error was encountered while instantiating the schema platform. Check the nested exception to determine the correct course of action.

25015: NAMESPACE_RESOLVER_NOT_SPECIFIED

Cause: An error occurred trying to resolve the namespace URI for. A namespace resolver was not specified on the descriptor.

Action: Ensure that any descriptors that make use of namespaces have a namespace resolver specified on them.

25016: NAMESPACE_NOT_FOUND

Cause: A namespace for the prefix was not found in the namespace resolver.

Action: Ensure that any namespace prefix which is used by a descriptor or mapping has a corresponding entry in that descriptor's namespace resolver.

Migration Utility Exceptions (26001 - 26017)

`MigrationUtilityException` is a run-time exception that is raised when an error is encountered during the use of the TopLink migration utility.

Format

EXCEPTION [TOPLINK - error code]: Exception name
EXCEPTION DESCRIPTION: Message

Example 13–26 Migration Utility Exception

EXCEPTION [TOPLINK - 26002]: oracle.toplink.exceptions.MigrationUtilityException
EXCEPTION DESCRIPTION: The program security manager prevents the migration utility from creating a JAR class loader for the JAR file.

26001: WLS_MULTIPLE_JARS_WITH_INPUT_ORION_NOT_SUPPORTED

Cause: WLS->OC4J TopLink migration exception: There are multiple migratable JARs in the input EAR file, and the `orion-ejb-jar.xml` is also provided in the input. The migration utility only supports a single migratable JAR file in an EAR file if the `orion-ejb-jar.xml` file is also in the input.

Action: Remove the `orion-ejb-jar.xml` file from the input directory, or keep only one migratable JAR file in the input EAR file.

26002: FAILED_TO_CREATE_JAR_CLASSLOADER

Cause: The program security manager prevents the migration utility from creating a JAR class loader for the JAR file.

Action: Configure the program security manager to grant permission on the program to create the JAR class loader.

26003: FAILED_TO_CREATE_DIRECTORY

Cause: The program security manager prevents the migration utility from creating a directory.

Action: Configure the program security manager to grant permission on the program to create a local directory.

26004: FILE_NOT_ACCESSIBLE

Cause: The file is not accessible by the migration utility.

Action: Check to ensure that the file is on the specified path and is readable.

26005: FILE_NOT_DELETABLE

Cause: The program security manager prevents the migration utility from deleting a directory.

Action: Configure the program security manager to grant permission to the program to delete files.

26006: FAILED_TO_READ_INPUTSTREAM

Cause: A `java.io.IOException` occurred when reading data from an input stream.

Action: Check to ensure that the file is not locked and is accessible.

26007: FAILED_TO_CLOSE_STREAM

Cause: A `java.io.IOException` occurred when closing data from an input or output stream.

Action: Check to ensure that the file is not locked and is accessible.

26008: FAILED_TO_CLOSE_ZIPFILE

Cause: A `java.io.IOException` occurred when closing data from a zip file.

Action: Check to ensure that the zip file is not locked and is accessible.

26009: JAR_FILE_NOT_ACCESSIBLE

Cause: The JAR file was not found or was not accessible on the specified path.

Action: Check to ensure that the JAR file is not locked and is accessible.

26010: QUERY_NOT_WELL_DEFINED

Cause: Query method with parameters of the entity defined in the `ejb-jar.xml` file is not well defined as it does not start with `find` or `ejbSelect`.

Action: Check the `ejb-jar.xml` file to ensure that the finder name starts with either `find` or `ejbSelect` according to the EJB spec.

26011: FAIL_TO_LOAD_CLASS_FOR_QUERY

Cause: The class defined as a parameter type of the query of the entity in `ejb-jar.xml` could not be loaded.

Action: Ensure that the bean class is included on the classpath.

26012: FINDER_NOT_DEFINED_IN_ENTITY_HOME

Cause: The `finder` method with parameters defined in the `ejb-jar.xml` file is not defined at the entity's local and/or remote home interface.

Action: Check the `ejb-jar.xml` file and the bean home to ensure that the `finder` method parameter types are properly defined.

26013: EJB_SELECT_NOT_DEFINED_IN_ENTITY_BEAN_CLASS

Cause: The `ejbSelect` method with parameters defined in the `ejb-jar.xml` file is not defined at the entity's bean class.

Action: Check the `ejb-jar.xml` file and the bean class to ensure that the `ejbSelect` method parameter types are properly defined.

26014: ENTITY_IN_WLS_CMP_JAR_NOT_DEFINED_IN_WLS_EJB_JAR

Cause: The entity specified in the `weblogic-cmp-jar.xml` file is not defined in the `weblogic-ejb-jar.xml` file.

Action: Ensure that the entity is consistently defined in the deployment descriptor files.

26015: NO_ENTITY_DEFINED_IN_WLS_CMP_JAR

Cause: No entity element is defined in the `weblogic-cmp-rdbms-jar.xml` file.

Action: Ensure that all entities defined in the `ejb-jar.xml` file are mapped in the WebLogic CMP descriptor file.

26016: WLS_CMP_DESCRIPTOR_FILE_NOT_FOUND

Cause: The WebLogic CMP descriptor file specified in the `weblogic-ejb-jar.xml` file was not found in the directory.

Action: Ensure that the WebLogic CMP descriptor file, specified in the `weblogic-ejb-jar.xml` file, is included in the input directory.

26017: CMP_DESCRIPTOR_NOT_FOUND_IN_JAR

Cause: The CMP descriptor file is not found in the JAR file to be migrated.

Action: Ensure that the WebLogic CMP descriptor file, specified in the `weblogic-ejb-jar.xml` file, is included in the input archive JAR file.

EJB JAR XML Exceptions (72000 – 72022)

`EJBJarXMLException` is a run-time exception that is raised at deployment time when the `ejb-jar.xml` file is read and the required concrete EJB classes code is generated.

Format

EXCEPTION [TOPLINK - error code]: Exception name
EXCEPTION DESCRIPTION: Message

Example 13–27 EJB JAR XML Exception

EXCEPTION [TOPLINK - 72000]: oracle.toplink.exceptions.EJBJarXMLException
EXCEPTION DESCRIPTION: Error reading ejb-jar.xml file.

72000: READ_EXCEPTION

Cause: Failed to read an `ejb-jar.xml` file due to a `java.io.IOException` or `javax.xml.parsers.ParserConfigurationException`.

Action: See the generated exception for the root cause.

72001: INVALID_DOC_TYPE

Cause: Failed to parse the specified file because it did not use the expected doctype: `-//Sun Microsystems, Inc.//DTD Enterprise JavaBeans 2.0//EN`

Action: Verify that your `ejb-jar.xml` file uses the correct documentation type.

72003: EJB_2_0_ATTRIBUTE_NOT_EXIST

Cause: A mapping for a field could not be created on the descriptor.

Action: Check to ensure that there are abstract get and set methods accessible for this field.

72011: EMPTY_TEXT_ATTRIBUTE

Cause: The element cannot have an empty text attribute.

Action: Use a non-empty text attribute for the sub-element.

72012: MULTIPLE_ENTITIES_FOUND_FOR_EJB_NAME

Cause: There are multiple entities with the same EJB bean name in the XML file.

Action: Use a unique EJB bean name for each entity.

72013: INVALID_CMP_VERSION

Cause: The entity did not have a valid `cmp-version`. The `cmp-version` must be `1.x` or `2.x`.

Action: Use a `cmp-version` value of `1.x` or `2.x` only.

72014: INVALID_EJB_NAME_FOR_RELATIONSHIP_ROLE

Cause: The EJB bean name in the `ejb-relationship-role` is not found in the XML file.

Action: Use the EJB bean name of an existing entity for the `ejb-relationship-role`.

72015: INVALID_MULTIPLICITY

Cause: There was an invalid multiplicity value defined for a relationship involving the mapping in a descriptor. The multiplicity must be `One` or `Many`.

Action: Use a multiplicity value of `One` or `Many` only.

72016: INVALID_PERSISTENCE_TYPE

Cause: The entity did not have a valid persistence type. The type must be `Bean` or `Container`.

Action: Use a `persistence-type` value of `Bean` or `Container` only.

72017: INVALID_QUERY_METHOD_NAME

Cause: The query `method-name` does not start with `find` or `ejbSelect`. No information was updated for this query.

Action: Verify that the `method-name` value starts with `find` or `ejbSelect`.

72018: NOT_SINGLE_PERSISTENCE_TYPE

Cause: TopLink Workbench requires that all entities in the XML file for this project have the same `persistence-type` and/or `cmp-version`. The project will be set according to the `persistence-type` and/or `cmp-version` of the first entity in the file.

Action: Verify that all CMP entities have the same `cmp-version` value.

72022: REQUIRED_ATTRIBUTE_NOT_EXIST

Cause: There is an element that does not have a required attribute.

Action: Add the sub-element to the parent element.

TopLink Workbench Error Reference

TopLink checks each project, descriptor, and mapping to ensure that you have properly defined the required settings. Errors and warnings are displayed in the Problems window (see ["Using the Problems Window"](#) on page 4-11).

You can also create a project status report (see ["Generating the Project Status Report"](#) on page 21-12) that contains all errors in a specific project.

This chapter contains information on the following:

- Oracle TopLink Workbench Error Messages:
 - [Miscellaneous Errors \(1 – 89\)](#), on page 14-1
 - [Project Errors \(100 – 126\)](#), on page 14-2
 - [Descriptor Errors \(200 – 350\)](#), on page 14-4
 - [Mapping Errors \(400 – 483\)](#), on page 14-11
 - [Table Errors \(500 – 610\)](#), on page 14-15
 - [XML Schema Errors \(700 – 706\)](#), on page 14-20
 - [Session Errors \(800 – 812\)](#), on page 14-21
- [Common Classpath Problems](#), on page 14-22
- [Data Source Problems](#), on page 14-23

Miscellaneous Errors (1 – 89)

This section lists TopLink Workbench error codes, information about the likely **Cause** of the problem, and a possible corrective **Action**.

13: No class indicator field should be defined for the abstract class [class].

Cause: Abstract classes should not be included or contain an **Indicator Value** on a descriptor's Inheritance tab.

Action: You must either remove the **Include** option for the class on the **Inheritance** tab, or remove the **abstract** modifier option on the descriptor's **Class Info – Class** tab.

16: The accessors for an instance variable must have return type and parameters in agreement with that instance variable's type.

Cause: Either the **get** method for an instance variable does not return the correct type of instance variable, or the **set** method does not have a single argument that is the correct type of instance variable.

Action: Ensure that the get method returns the same type as its associated instance variable, and the set method takes a single argument that is the same type as the associated instance variable.

29: The database type for the mapping does not match the database type of the database field.

Cause: You selected a **Database Type** for an Object Type mapping that differs from the type in the database.

Action: Select the **Database Type** on the mapping's **General** tab that matches the actual type in the database.

34: One or more field types have not been specified.

Cause: You created a field with a missing or invalid **Type**.

Action: You must specify the **Type** for each database field on the **Field** tab.

38: EJB Class information is not compatible with project persistence type.

Cause: You entered incorrect EJB information.

Action: Ensure that the information you entered on the EJB descriptor's **EJB Info** tab is correct, based on the project's persistence type (as specified on the project's **General Properties** tab).

53: No class in inheritance tree is marked as root.

Cause: The inheritance hierarchy must contain one root descriptor.

Action: Select the **Is Root Descriptor** option on the **Inheritance** tab of one descriptor in the inheritance. See ["Descriptors and Inheritance"](#) on page 26-3.

54: No class indicator field is selected for this root class.

Cause: You selected the **Use Class Indicator Dictionary** option for the root descriptor in the inheritance hierarchy, but did not specify an indicator value for the root and its children.

Action: Use the **Indicator Type** list on the **Inheritance** tab for the root class. See ["Descriptors and Inheritance"](#) on page 26-3.

89: The root class does not include an indicator mapping for this descriptor.

Cause: The root class in the inheritance hierarchy is set to the **use class indicator dictionary** option. The dictionary does not contain an indicator value for this child class.

Action: Select an **Indicator Type** on the **Inheritance** tab of the root class that includes the child types. See ["Descriptors and Inheritance"](#) on page 26-3.

Project Errors (100 – 126)

This section lists TopLink Workbench project errors.

100: The project caches all statements by default for queries, but does not bind all parameters.

Cause: A named query that caches statements must also bind parameters.

Action: On the **Named Queries – Options** tab, change the **Cache Statements** field to **False**, or change the **Bind Parameters** field to **True**.

101: The project uses a custom sequence table, but the counter field is not specified.

Cause: On the project's **Sequencing** tab, you selected **Use Custom Sequence Table**, but did not complete the **Counter Field** field.

Action: Select the field to use as the Counter Field for this sequence table. See ["Configuring Sequencing at the Project Level"](#) on page 23-3 for details.

102: The project uses a custom sequence table, but the name field is not specified.

Cause: On the project's **Sequencing** tab, you selected **Use Custom Sequence Table**, but did not complete the **Name Field** field.

Action: Select the field to use as the **Name Field** for this sequence table. See ["Configuring Sequencing at the Project Level"](#) on page 23-3 for details.

113: The method specified for the inheritance policy's class extraction method on this descriptor is no longer a visible member of this class.

Cause: You changed the class hierarchy within the project, causing the inheritance policy to no longer be visible to the class.

Action: Ensure that the inheritance policy is visible to the class. See ["Descriptors and Inheritance"](#) on page 26-3.

117: The reference must have at least one field association.

Cause: You selected a table reference for a relationship mapping, but did not define a source and target field key pair.

Action: For variable one-to-one mappings, you must define a query key pair (in the source descriptor's tables) to use for the common query key.

118: The selected parent descriptor for this descriptor's inheritance policy does not have an associated inheritance policy.

Cause: You selected a **Parent Descriptor** for a descriptor's inheritance policy that does not have an inheritance policy.

Action: Ensure that the parent descriptor's inheritance policy is valid. See ["Descriptors and Inheritance"](#) on page 26-3.

120: The source reference does not contain any field associations.

Cause: You created a many-to-many mapping, but did not define a source and target reference for the source reference.

Action: You must define a table reference and the appropriate key pairs for each source reference. See ["Configuring Table and Field References \(Foreign and Target Foreign Keys\)"](#) on page 37-8.

121: The target reference does not contain any field associations.

Cause: You created a many-to-many mapping, but did not define a source and target reference for the target reference.

Action: You must define a table reference and the appropriate key pairs for each target reference. See ["Configuring Table and Field References \(Foreign and Target Foreign Keys\)"](#) on page 37-8.

122: There is no database associated with the query key [query key name].

Cause: You did not associate the specified query key with a database table.

Action: You must select a database **Name** and **Table** on the **Query Keys** tab.

123: This root class has no class indicator mappings for its hierarchy.

Cause: You created an inheritance policy with the **Use Class Indicator Dictionary** option but did not specify the indicator values for all subclasses.

Action: Specify the indicator values for all subclasses on the descriptor's **Inheritance** tab. See ["Configuring Inheritance for a Child \(Branch or Leaf\) Class Descriptor"](#) on page 28-49.

Note: TopLink displays a list of each subclass and indicator value if you have identified the subclasses' parent descriptor.

126: Writable mappings defined for the class indicator field *[field name]*.

Cause: The class indicator field should not contain any writable mappings.

Action: Select a **Use Class Indicator Field** on the descriptor's **Inheritance** tab that does not contain any writable mappings. See ["Configuring Inheritance for a Child \(Branch or Leaf\) Class Descriptor"](#) on page 28-49 and ["Configuring Inherited Attribute Mapping in a Subclass"](#) on page 28-56.

Descriptor Errors (200 – 350)

This section lists TopLink Workbench descriptor errors.

200: The descriptor's class is not public.

Cause: The descriptor must use a public access modifier.

Action: On the descriptor's **Class Info – Class** tab, change the **Access Modifier** option to **Public**.

201: This class is a subclass of a final class.

Cause: If you select the **Final** option on the descriptor's **Class Info – Class** tab for a class, then the class cannot contain subclasses.

Action: See ["Configuring Classes"](#) on page 4-41.

210: Two methods *[method names]* cannot have the same signature.

Cause: You created methods with identical signatures.

Action: On the **Class Info – Methods** tab, change the information for one of the methods. See ["Configuring Classes"](#) on page 4-41.

220: An aggregate shared by multiple source descriptors cannot have one-to-many or many-to-many mappings.

Cause: You attempted to create multiple one-to-many and many-to-many, or one-to-one mappings in which the target is the aggregate.

Action: Aggregate descriptors that are shared by multiple source descriptors cannot have mappings that contain a target object that references the descriptor.

221: Classes cannot reference an aggregate target with one-to-one, one-to-many, or many-to-many mappings.

Cause: You tried to select an aggregate descriptor as a reference.

Action: You cannot select an aggregate descriptor as the **Reference Descriptor** for a one-to-one, one-to-many, or many-to-many mapping.

225: The implementor *[implementor name]* no longer implements this interface.

Cause: One descriptor listed as an implementation method for this interface descriptor no longer implements this descriptor's interface.

Action: You must either remove the descriptor from the list of implementation methods or alter the descriptor's class so that it implements this descriptor's interface.

230: No primary table is specified.

Cause: The descriptor must be associated with a database table.

Action: On the descriptor's **Descriptor Info** tab, use the **Associated Table** field to select a primary table. See ["Configuring Associated Tables"](#) on page 29-2.

231: No primary key(s) specified in [table name] table.

Cause: You must specify a primary key for each database table. When importing tables from a database into TopLink Workbench, the primary key information will be retained only if the JDBC driver supports the `getPrimaryKeys` method.

Action: Ensure that a primary key is specified for each descriptor on the **Descriptor Info** tab. See ["Configuring Associated Tables"](#) on page 29-2.

232: The following primary key fields are unmapped: [field name].

Cause: Each primary key field must have a writable mapping.

Action: Ensure that the primary key(s) are mapped. See ["Configuring Associated Tables"](#) on page 29-2.

233: The number of primary keys does not match the number of primary keys on the parent.

Cause: In an inheritance hierarchy, the child class must have the same number of primary keys as the parent class.

Action: Ensure that the parent and child class have the same number of primary keys on the descriptor's **Descriptor Info** tab. See ["Configuring Associated Tables"](#) on page 29-2.

234: The primary keys do not match parent's primary keys.

Cause: In an inheritance hierarchy, the child's primary key(s) must match the root's primary key(s).

Action: Ensure that each child's **Primary Key** on the **Descriptor Info** tab matches the parent's primary key. Ensure that the parent and child class have the same primary keys on the descriptor's **Descriptor Info** tab. See ["Configuring Associated Tables"](#) on page 29-2.

235: The following fields have multiple writable mappings: [field name].

Cause: Multiple mappings cannot write to the same database field.

Action: Each database field must have a single, writable mapping.

236: No sequence field name is selected.

Cause: You selected **Use Sequencing** on a descriptor's **Descriptor Info** tab but did not specify the sequence information.

Action: Specify a **Name**, **Table**, and **Field**. See ["Configuring Sequencing at the Descriptor Level"](#) on page 29-3.

237: No sequence name is selected.

Cause: You selected **Use Sequencing** on a descriptor's **Descriptor Info** tab but did not specify the sequence information.

Action: Specify a **Name**, **Table**, and **Field**. See ["Configuring Sequencing at the Descriptor Level"](#) on page 29-3.

238: No sequence table is selected.

Cause: You selected **Use Sequencing** on a descriptor's **Descriptor Info** tab but did not specify the sequence information.

Action: Specify a **Name**, **Table**, and **Field**. See ["Configuring Sequencing at the Descriptor Level"](#) on page 29-3.

239: The selected sequence table is not one of the descriptor's associated tables.

Cause: The table used for sequencing is not associated with the descriptor.

Action: You must either associate the sequencing table with the descriptor, or select a table that is already associated with the descriptor. See ["Configuring Sequencing at the Descriptor Level"](#) on page 29-3 and ["Configuring Associated Tables"](#) on page 29-2.

240: Two queries [query names] cannot have the same signature.

Cause: Two queries for this descriptor share the same signature (query name + parameter names). This is not allowed.

Action: You must either remove one of the queries, rename one of the queries, or change the parameters so that the signatures no longer match.

241: The query [query name] has Cache Statement set to true, but does not bind parameters.

Cause: A named query that caches statements must also bind parameters.

Action: On the **Named Queries – Options** tab, either change the **Cache Statements** field to **False**, or change the **Bind Parameters** field to **True**.

242: The query [query name] does not maintain cache but does refresh the remote identity map results.

Cause: The query has **Refresh Remote Identity Map** selected but does not have **Maintain Cache** selected.

Action: You must either select **Maintain Cache** for the descriptor, or deselect **Refresh Remote Identity Map**.

243: The query [query name] does not maintain cache but does refresh the identity map results.

Cause: The query has **Refresh Identity Map** selected but does not have **Maintain Cache** selected.

Action: You must either select **Maintain Cache** for the descriptor, or deselect **Refresh Identity Map**.

245: The query [query name] refreshes identity map results but does not refresh remote identity map results.

Cause: **Refresh Identity Map Results** is selected for the query, but **Refresh Remote Identity Map Results** is not.

Action: You must either select **Refresh Remote Identity Maps** or deselect **Refresh Identity Maps**.

246: The query key [query key] does not have an associated database field

Cause: Each query key must be associated with a database field.

Action: On the **Query Keys** tab, use the **Field** option to select a database field for the query key.

247: The database field selected for query key [query key] does not exist on this descriptor's associated tables.

Cause: Each database field associated with a query key must exist on database table associated with the query key's descriptor.

Action: You must either change the database field associated with the query key, or associate the descriptor with a database table that includes the database field associated with the query key.

248: The expression [line number] on query [query name] is invalid because a parameter has not been specified.

Cause: One of the arguments in the query expression is missing or invalid.

Action: Edit the query and ensure that all query keys and parameters have been specified.

249: The expression [line number] on query [query name] is invalid because a query key has not been specified.

Cause: One of the arguments in the query expression is missing or invalid.

Action: Edit the query and ensure that all query keys and parameters have been specified.

250: The expression [line number] on query [query name] is invalid. When querying on a reference mapping, only unary operators (Is Null, Not Null) are supported.

Cause: You created an expression node that includes a reference mapping with an invalid operator.

Action: On the Expression Builder dialog box, select the node and change the **Operator** field to **IS NULL** or **NOT NULL**.

270: No schema context is specified.

Cause: Each descriptor in an XML or EIS project must be associated with an XML schema context.

Action: Select the EIS or XML descriptor in the **Navigator** and complete the **Schema Context** field on the **Descriptor Info** tab.

271: The descriptor represents a document root object, but no default root element is chosen.

Cause: Each root EIS descriptor must have a default root element.

Action: On the descriptor's **Descriptor Info** tab, complete the **Default Root Element** field.

290: No primary keys specified.

Cause: Although you have associated the descriptor with a database table, you have not identified the primary keys.

Action: Use the **Primary Keys** area of the descriptor's **Descriptor Info** tab to select the primary keys for the descriptor.

291–304: The event policy's [method type] method is no longer a visible member of this descriptor's associated class.

Cause: You changed the class hierarchy within the project, causing the method to no longer be visible to the class.

Action: Ensure that the selected method is visible to the class.

305: The write-lock field is stored in an object, but there is not a writable mapping to the field.

Cause: If the write lock field is stored in object, there must be a non-read-only mapping to it.

Action: On the mapping's **General** tab, ensure that **Read-Only** is *not* selected.

306: Database fields specified for Selected Fields type Locking Policy must be mapped: [field name]

Cause: You selected an unmapped database field for a descriptor's locking policy.

Action: On the descriptor's **Locking** tab, ensure that you have selected a mapped database field as the **Selected Field**. See ["Configuring Locking Policy"](#) on page 28-62.

307: Database fields specified for Selected Fields type Locking Policy must not be primary key fields: [field name]

Cause: The database fields you selected for the optimistic locking policy (by fields) contains the primary keys for the database table.

Action: In the **By Fields** area of the descriptor's **Locking** tab, select different fields. See ["Configuring Locking Policy"](#) on page 28-62.

308: Version locking is chosen as the Locking Policy, but the field is not specified.

Cause: If you select to use version locking with an optimistic locking policy, you must identify which database field to use for version control.

Action: Use the **Database Field** field on the descriptor's **Locking** tab to select a field to use for version control. See ["Configuring Locking Policy"](#) on page 28-62.

309: The Version Locking database field selected does not exist on this descriptor's associated tables.

Cause: The database field you selected for optimistic version locking does not exist on the descriptor's associated table.

Action: You must either select a different database field on the descriptor's **Locking** tab, or associate the descriptor with a different database table. See ["Configuring Locking Policy"](#) on page 28-62.

310: Database fields specified for Selected Fields type Locking Policy do not exist on this descriptor's associated tables: [field name]

Cause: The database fields you selected for the optimistic locking policy (by fields) do not exist on the descriptor's associated table.

Action: You must either select a different database field on the descriptor's **Locking** tab, or associate the descriptor with a different database table. See ["Configuring Locking Policy"](#) on page 28-62.

311: The method you have specified for the instantiation policy's method on this descriptor is no longer a visible member of this class.

Cause: The method selected as the instantiation method has either been removed, or its visibility has been reduced so that it is no longer publicly visible.

Action: Deselect this method as the instantiation method.

See ["Configuring Instantiation Policy"](#) on page 28-68.

312: The method you have specified for the instantiation policy's factory instantiation method on this descriptor is no longer a visible member of this class.

Cause: The method selected as the factory instantiation method has either been removed, or its visibility reduced so that it is no longer publicly visible.

Action: Deselect this method as the factory instantiation method. See ["Configuring Instantiation Policy"](#) on page 28-68.

313: The method you have specified for the instantiation policy's factory method on this descriptor is no longer a visible member of this class.

Cause: The method selected as the factory method has either been removed, or its visibility reduced so that it is no longer publicly visible.

Action: Deselect this method as the factory method. See ["Configuring Instantiation Policy"](#) on page 28-68.

314: "Use factory" is specified for the Instantiation policy, but all required information is not specified.

Cause: You selected the **Use Factory** option on the descriptor's **Instantiation Policy** tab, but did not specify the **Factory Class**, **Factory Method**, or **Instantiation Method** fields.

Action: Complete the **Factory Class**, **Factory Method**, or **Instantiation Method** fields on the descriptor's **Instantiation** tab. See ["Configuring Instantiation Policy"](#) on page 28-68.

315: "Use method" is selected for the Instantiation policy, but no method is selected.

Cause: You selected the **Use Method** option on the descriptor's **Instantiation Policy** tab, but did not specify the field.

Action: Select the **Method** on the descriptor's **Instantiation** tab. See ["Configuring Instantiation Policy"](#) on page 28-68.

316: The class does not have an accessible zero argument constructor.

Cause: No accessible zero argument constructor exists for the class associated with this descriptor.

Action: Make the zero argument constructor accessible if it exists, or create a accessible zero argument constructor if it doesn't exist.

317: No method was specified for the copying policy.

Cause: You specified that the descriptor should use a specific clone method for copying, but you did not select a method.

Action: Complete the **Use Clone Method** field on the descriptor's **Copying** tab to select a method.

318: The method specified for the copy policy on this descriptor is no longer a visible member of this class.

Cause: You changed the class hierarchy within the project, causing the copy policy to no longer be visible to the class.

Action: Ensure that the copy policy is visible to the class.

319: Primary keys do not match across associated tables and no reference(s) specified in multiple table policy information.

Cause: You attempted to associate multiple tables using a primary key.

Action: Primary key field names must match across associated tables, or references must be defined from the base table to each derived table.

320: The multiple table reference should be defined from the base table [table name] to the derived table.

Cause: This descriptor has **Inheritance** and **Multitable** advanced properties defined on it.

Action: The multiple table relationship that is defined between the base class' table and this derived class' table must be defined from base to derived.

321: The multiple table reference should not be defined on the database.

Cause: When using multitable with differently named primary keys, you must set a reference from the TOP table to the BOTTOM table. This reference must not be an actual constraint on the database.

Action: Select the table in which this is defined, and deselect the **On Database** option.

322: A class containing the desired after loading method should be specified.

Cause: You added an after-load method to a descriptor, but you did not specify a class.

Action: Complete the **After Load** tab. See "[Configuring Amendment Methods](#)" on page 28-78.

323: An after-load method must be specified.

Cause: You added an after-load method to a descriptor, but did not select an amendment method.

Action: Complete the **After Load** tab. See "[Configuring Amendment Methods](#)" on page 28-78.

324: An interface class must be specified for the interface alias.

Cause: You added an interface alias to a descriptor, but did not select an amendment method.

Action: Complete the **Interface Alias** tab.

325: The inheritance hierarchy originating in this descriptor cannot contain both aggregate and nonaggregate child descriptors.

Cause: Aggregate and class descriptors cannot be in the same inheritance hierarchy.

Action: Ensure that the inheritance hierarchy contains either aggregate *or* nonaggregate children, but not both.

326: The inheritance hierarchy originating in this descriptor cannot contain both root and composite child descriptors.

Cause: There is a mixture of root and composite descriptors among the descendents of this descriptor.

Action: Make all descendents of this descriptor the same type by either making them all root, or making them all composite. You can do this by removing the differing descriptor from the hierarchy, or changing their type to be consistent with the other descriptors in the hierarchy.

330: The returning policy insert fields do not exist on this descriptor's associated tables: [field name]

Cause: The field you selected on the descriptor's **Returning** tab does not exist on the database table associated with the descriptor.

Action: Select a different database table in the **Insert** area of the descriptor's **Returning** tab.

331: The returning policy update field [field name] does not exist on this descriptor's associated tables.

Cause: The field you selected on the descriptor's **Returning** tab does not exist on the database table associated with the descriptor.

Action: Select a different database table in the **Update** area of the descriptor's **Returning** tab.

350: Descriptors with Unknown Primary Keys must use sequencing.

Cause: **Unknown Primary Key Class** is selected for this descriptor, but the descriptor does not use sequencing.

Action: Change the descriptor so that it uses sequencing, or so that it no longer uses an unknown primary key class.

Mapping Errors (400 – 483)

This section lists TopLink Workbench mapping errors.

400: Method accessors have not been selected.

Cause: You selected **Use Method Accessing** for a mapping, but you did not select a method.

Action: You must select a **Get** and **Set** method on the mapping's **General** tab. See ["Configuring Method Accessing"](#) on page 35-14.

401, 402: The *[get/set access method]* method for this mapping's method accessing field is no longer visible to this descriptor.

Cause: You changed the class hierarchy within the project, causing the method access type (**get** or **set**) to no longer be visible to the class.

Action: Ensure that the selected method is visible to the class.

403: Mappings for EJB 2.0 CMP descriptors that use Value Holder Indirection must not use method accessing.

Cause: You cannot use method accessing on mappings for EJB 2.0 CMP descriptors that use **ValueHolder Indirection**.

Action: Because EJB attributes are code-generated, reference mappings *should not* be set to use method access. The attributes are code-generated to be of type **ValueHolder** but the abstract methods are defined to return the local interface type of the related bean.

404: Mapping references a write-lock field, but it is not read-only.

Cause: You specified a locking policy for a descriptor, but one of the attribute mappings is not read-only.

Action: Select the **Read Only** option on the mapping's **General** tab.

410: No direct field is specified.

Cause: For direct collection mappings, you must specify the direct collection information.

Action: Select a **Target Table** and **Direct Field** that the direct collection specifies.

415: No direct key field is specified.

Cause: For direct map mappings, you must specify a direct key field in the reference table that stores the primitive data value of the map key.

Action: On the direct map mapping's **General** tab, select a **Direct Key Field**. See ["Configuring Direct Key Field"](#) on page 46-2.

420: No database field is selected.

Cause: You created a direct-to-field or type conversion mapping without selecting a database field.

Action: For attributes with direct-to-field mappings, you must specify a **Database Field** on the mapping's **General** tab. For attributes with type conversion mappings, you must specify a **Database Field** on the mapping's **General** tab.

421: The selected database field does not exist on this descriptor's associated tables.

Cause: The database field mapped to an attribute is not included in the table associated with the attribute's descriptor.

Action: Ensure that the **Database Field** field on a mappings **General** tab is included in the table that you associated with the attribute's descriptor. See ["Configuring Associated Tables"](#) on page 29-2 and ["Configuring a Database Field"](#) on page 37-2.

430, 431: No null value type has been selected.

Cause: You selected to **Use Default Value When Database Field is Null** for a mapping, but did not specify the value.

Action: Specify a default **Type** or **Value**, or both on the mapping's **General** tab. See ["Configuring a Default Null Value at the Mapping Level"](#) on page 35-12.

This message may also appear after using the Package Rename tool when upgrading an older TopLink Workbench project.

440: XML type mappings are supported only on the Oracle9i Platform.

Cause: You created a Direct to XML Type mapping in relational project that uses a non-Oracle9i database.

Action: Select an Oracle9i platform as the database platform for the data source. See ["Configuring Relational Database Platform at the Project Level"](#) on page 23-2.

450: No reference descriptor is selected.

Cause: You created a mapping, but did not specify the reference descriptor

Action: You must select a **Reference Descriptor** for each relationship mapping on the mapping's **General** tab.

451: *[descriptor name]*references *[descriptor name]*, which is not active.

Cause: You tried to select an inactive descriptor as a **Reference Descriptor** on the mapping's **General** tab.

Action: You must either select a new **Reference Descriptor**, or make the descriptor active.

460: No table reference is selected.

Cause: You created a relationship mapping, but did not specify a reference table.

Action: Select (or create) a table reference for each relationship mapping on the mapping's **Table Reference** tab.

461: Table reference is invalid.

Cause: The table reference selected for this mapping is invalid.

Action: Select a different table reference for this mapping.

462: The reference *[table reference]* does not have any field associations.

Cause: You selected a table reference for a mapping, but did not add a key pair.

Action: You must specify source and target key pairs for the reference.

463: A key pair has not been completely specified for a reference.

Cause: You created a table reference without a key pair.

Action: You must specify a foreign key reference for the database table. Use the database table's **Reference** tab to add a key pair.

464: No relationship partner is specified.

Cause: You selected the **Maintains Bidirectional Relationship** option for a relationship mapping, but did not select a mapping to use as the relationship partner.

Action: Select a mapped attribute (from the reference descriptor) for this relationship. See "[Configuring Bidirectional Relationship](#)" on page 35-34.

465: The relationship partner must be a one-to-one, one-to-many, or many-to-many mapping.

Cause: You selected an invalid attribute as the Relationship Partner in a bidirectional relationship.

Action: In the **Relationship Partner** field, select a one-to-one, one-to-many, or many-to-many mapping. See "[Configuring Bidirectional Relationship](#)" on page 35-34.

466: The specified relationship partner mapping does not specify this mapping as its own relationship partner.

Cause: **Maintains Bidirectional Relationship** is selected for this mapping, but the mapping selected as the relationship partner does not have this mapping selected as its relationship partner.

Action: You must either select a different mapping for this mappings relationship partner, which has this mapping selected as it bidirectional relationship partner, or select this mapping as the bidirectional relationship partner of the mapping selected as the bidirectional relationship partner for this mapping.

467: The chosen reference descriptor is not a valid reference descriptor for this mapping.

Cause: The descriptor selected as the reference descriptor for this mapping is not a valid reference descriptor.

Action: Select a valid reference descriptor for this mapping.

470: No container class is selected.

Cause: No container class has been selected for this collection mapping.

Action: Select a Container class for this Collection mapping.

471: The container policy uses a Collection class, but the container class is not a Collection.

Cause: The selected container class for this collection mapping is not a Collection, but **Use Collection Class** is selected.

Action: Select a Container class that is a Collection for this mapping.

472: The container policy uses a Map class, but the container class is not a Map.

Cause: The selected Container class for this Collection mapping is not a Map class, but **Use Map Class** is selected.

Action: Select a Container class that is a Map class.

473: The container class must be instantiable.

Cause: The selected Container class for this Collection mapping is not instantiable.

Action: Select a Container class this is instantiable, (not an Interface, Abstract class, or Primitive class).

474: The container class does not agree with the instance variable.

Cause: The selected `Container` class for this `Collection` mapping, does not agree with the instance variable that is associated with the mapping. Either the variable is a `Map` class and the selected `Container` class is a `Collection` or vice versa.

Action: You must either select a `Container` class that agrees with the type of instance variable with which it is associated, or change the instance variable to agree with the selected `Container` class.

475: The container class is a Map, but the key method is not selected.

Cause: `Use Map Class` is selected for the `Container` policy for this `Collection` mapping, but a key method has not been selected.

Action: You must either select a key method for this `Container` policy, or change the `Container` policy to not use a map class.

476: The key method specified for this mapping is no longer visible to the owning descriptor's class.

Cause: The selected key method for the `Container` policy for this `Collection` mapping policy is not visible to the descriptor's class.

Action: You must either select a different method that is visible to the descriptor's class, or change the selected method so that it is visible.

477: The key method specified for this mapping is not valid.

Cause: The selected key method for the `Container` policy for this `Collection` mapping is invalid because it does not have the correct return type, or it does not accept more than zero parameters.

Action: You must either select a different method that is valid, or change the selected method so that it will return the correct type and accept more than zero parameters.

478: One-to-Many and Many-to-Many mappings in EJB 2.0 CMP descriptors may not use ValueHolder indirection.

Cause: A one-to-many or many-to-many mapping in an EJB 2.0 CMP descriptor is using `ValueHolder` indirection.

Action: You must either change the mapping to use no indirection or non-`ValueHolder` indirection.

480: No relation table is selected.

Cause: You created a many-to-many mapping, but did not specify a relation table. The relation table represents the relationship between the primary keys of the source table and target table.

Action: Select or create a **Relation Table** on the mapping's **General** tab.

481: The relation table is not dedicated to single, writable many-to-many mapping.

Cause: More than one many-to-many mapping in the project are using the same relation table.

Action: Each relation table should be used in one and only one many-to-many mapping.

482: No source reference is selected.

Cause: You created a many-to-many mapping, but did not select (or create) a source table reference on the mapping's **Source Reference** tab.

Action: The source table reference must contain a **Source** field (from the mapping's relation table) and a **Target** field (from one of the descriptor's associated tables).

483: No target reference is selected.

Cause: You created a many-to-many mapping, but did not select (or create) a target table reference on the mapping's **Source Reference** tab.

Action: The target table reference must contain a **Source** field (from the mapping's relation table) and a **Target** field (from one of the descriptor's associated tables).

Table Errors (500 – 610)

This section lists TopLink Workbench table errors.

500: You cannot use joining because the source and target (reference) descriptors are the same type.

Cause: You selected the **Use Joining** option on a one-to-one mapping in which the source and reference descriptors are the same.

Action: You must either deselect the **Use Joining** option or select a difference **Reference Descriptor** on the **One-to-One Mapping General** tab.

510: No query key associations have been defined.

Cause: You created a variable one-to-one mapping, but did not define a key pair.

Action: Create or select a key pair on the mapping's **Query Key Association** tab.

511: Not all query key associations have foreign key fields specified.

Cause: You created a query key association without a foreign key.

Action: You must specify a foreign key field for each query key association on the **Query Key Association** tab for variable one-to-one mapping.

512: The following specified query key names are no longer valid: *[query key]*

Cause: The query keys listed for this mapping no longer refer to the reference descriptor for this mapping. The query keys are now invalid.

Action: You must either remove the invalid query keys, or change the reference descriptor so that it corresponds with the query keys.

513: No indicator field is selected.

Cause: You created a variable one-to-one mapping, but did not specify a database field in which to store indicator values.

Action: Select the **Class Indicator Field** on the **Class Indicator Info** tab.

514: No indicator values are specified.

Cause: You created a variable one-to-one mapping, but did not specify indicator values for each object type.

Action: Select the **Indicator Type** on the **Class Indicator Info** tab.

515: *[descriptor name]* is not an implementor of the *[descriptor name]* interface, so it cannot have an indicator value.

Cause: You included a descriptor on the **Variable One-to-One Class Indicator Info** tab that is an implementor.

Action: Unselect the descriptor on the **Variable One-to-One Class Indicator Info** tab or add the descriptor to the **Implementor** tab.

516: The chosen reference descriptor is not an interface descriptor.

Cause: This variable one-to-one mapping has a reference descriptor selected which is not an interface descriptor. The reference descriptor for a variable one-to-one mapping must be an interface descriptor for the mapping to be valid.

Action: You must either choose a reference descriptor that is an interface descriptor, or change the mapping to no longer be variable.

520: No attribute transformer is specified.

Cause: No attribute transformer is specified for this transformation mapping.

Action: Select an attribute transformer for this transformation mapping.

521: The attribute transformer class is missing.

Cause: No class has been specified for the attribute transformer for this transformation mapping.

Action: Select a class for the attribute transformer.

522: The attribute transformer class [*class name*] is not a valid transformer class.

Cause: The attribute transformer class that is selected is not a valid attribute transformer class.

Action: Select a valid attribute transformer class for the transformation mapping.

523: The attribute transformer method is missing.

Cause: No method has been selected for the attribute transformer for the transformation mapping.

Action: Select a method for the attribute transformer.

524: The attribute transformer method [*method name*] is not visible to the parent descriptor's class.

Cause: The selected attribute transformer method is not visible to the descriptor class for this mapping.

Action: You must either select a different method that is visible, or change the method in the class to make it visible.

525: The attribute transformer method [*method name*] is not a valid transformer method.

Cause: The selected attribute transformer method either has the wrong return type or accepts the wrong parameters to be a valid transformer method for this transformation mapping.

Action: You must either select a method with the correct return type and parameters, or change the selected method so that it meets these criteria.

526: No field transformer associations are specified.

Cause: No field transformer association has been specified for this transformation mapping.

Action: Specify at least one field transformer association.

527: No transformer is specified for the field [*field name*].

Cause: No transformer specified for the given field.

Action: Specify a transformer for this field.

528: There is a missing field in the field transformer association.

Cause: There is no field specified for a field transformer association for this transformation mapping.

Action: Specify a field for all the field transformer associations for this transformation mapping.

529: There is a missing transformer class for the field *[field name]*.

Cause: The Transformer class is specified for this field transformer association, but the Transformer class is unspecified.

Action: Specify a Transformer class for the field transformer association for this field.

530: The transformer class *[class name]* for the field *[field name]* is not a valid transformer class.

Cause: The specified Transformer class for the field of this field transformer association is invalid.

Action: Specify a valid Transformer class for the field transformer association for this transformation mapping.

531: There is a missing transformer method for the field *[field name]*.

Cause: A transformer method is specified for this field transformer association, but the transformer method is unspecified.

Action: Specify a transformer method for the field transformer association for this field.

532: The transformer method *[method name]* for the field *[field name]* is not visible to the parent descriptor's class.

Cause: The specified transformer method for the field transformer association for this field is not visible to the descriptor or the class of this mapping.

Action: You must either choose a method that is visible to the class, or change the method so that it is visible.

533: The field transformer method *[method name]* for the field *[field name]* is not a valid transformer method.

Cause: The specified method for the field transformer association for this field either has the incorrect return type, or accepts the wrong parameters.

Action: You must either select a method that has the correct return type and parameters, or change the currently selected method so that it has the correct return type and parameters.

540: No object type is selected.

Cause: You created an object type mapping, but did not select the type.

Action: You must select the **Object Type** and **Database Type** on the **General** tab of the mapping.

542: No object-type mappings have been specified.

Cause: You created an object type mapping, but did not create an object-to-database mapping.

Action: You must specify at least one mapping (**Database Value** and **Object Value**) on the **General** tab of the mapping.

545: NCharacter, NString, and NClob database types are currently supported only on the Oracle9i platform.

Cause: You attempted to map a database type that is not supported by your database.

Action: The database type for a type conversion mapping or direct-to-field mapping can be `NCharacter`, `NString`, or `NCLOB` only if you are using an Oracle9i database.

550: Attribute is typed as a ValueHolderInterface, but the mapping does not use Value Holder Indirection.

Cause: You did not specify indirection or transparent indirection for the mapping.

Action: If the class attribute is of type `ValueHolderInterface`, you must use `ValueHolder` indirection for the mapping.

551: Mapping uses ValueHolder Indirection, but its associated attribute is not a ValueHolderInterface.

Cause: You selected indirection without a `ValueHolderInterface`.

Action: If you select the **Use Indirection (ValueHolder)** option for a one-to-many, many-to-many, or direct collection mapping, the associated class attribute must be `ValueHolderInterface`.

560: The container class for this mapping must implement oracle.toplink.indirection.IndirectContainer.

Cause: This mapping uses transparent indirection, but the `Container` class selected for its container policy is not an `IndirectContainer`.

Action: You must either select a `Container` class that is an `IndirectContainer`, or remove transparent indirection from the mapping.

570: The chosen reference descriptor is not an aggregate descriptor.

Cause: This is an aggregate mapping, but the selected reference descriptor is not an aggregate descriptor.

Action: You must either select a reference descriptor for this mapping that is an aggregate descriptor, or change this mapping to no longer be an aggregate mapping.

571: Aggregate fields are not specified.

Cause: You created an aggregate mapping without specifying specific fields.

Action: Every **Field Description** on the **Fields** tab must contain a unique **Field** for aggregate mappings.

572: Aggregate mapping fields must be unique.

Cause: You created an aggregate mapping without specifying unique fields.

Action: Every **Field Description** on the **Fields** tab must contain a unique **Field** for aggregate mappings.

573: The selected field does not exist on this descriptor's associated tables.

Cause: The field selected for one of the aggregate-path-to-fields for this aggregate mapping does not exist on any of the descriptor's associated tables.

Action: You must either select a different field for the path-to-field, or add the field to the appropriate table.

580: No XML field specified.

Cause: You mapped an attribute in an XML or EIS descriptor, but did not select an XML field.

Action: You must complete the **XML Field** field on the **General** tab of the mapping.

581: The specified XPath is not valid within the current schema.

Cause: The XPath specified for this mapping does not resolve in the schema.

Action: You must either select a different XPath, or alter the schema so that this XPath will resolve.

582: The specified XPath does not represent text data.

Cause: The XPath specified for this direct mapping does not resolve to a direct field in the schema.

Action: You must either select a different XPath, alter the schema so that this XPath will resolve to a direct field, or change the mapping type.

583: The specified XPath does not represent a single xml field.

Cause: The XPath specified for this mapping resolves to a field which is a collection, but this is not a collection mapping.

Action: You must either select a different XPath, alter the schema so that this XPath will resolve to a singular field, or change the mapping type.

590: The chosen reference descriptor is not a root eis descriptor.

Cause: The reference descriptor selected for this EIS reference mapping is not a root descriptor. Reference mappings in EIS descriptors must be root descriptors.

Action: You must either select a different reference descriptor for this mapping which is a root descriptor, or change the mapping type.

591: No relationship partner is specified.

Cause: This mapping has **Maintains Bidirectional Relationship** selected, but no relationship partner is specified.

Action: You must either deselect **Maintains Bidirectional Relationship**, or select a relationship partner.

592: The relationship partner must be an EIS One-to-One or EIS One-to-Many mapping.

Cause: The relationship partner selected for this mapping is not of the type EIS one-to-one or EIS one-to-many.

Action: You must select an EIS one-to-one or EIS one-to-many mapping as the relationship partner for this mapping, or deselect **Maintains Bidirectional Relationship**.

593: The specified relationship partner mapping does not specify this mapping as its own relationship partner.

Cause: The mapping selected as the relationship partner for this mapping does not have this mapping selected as its relationship partner. For these relationships to be bidirectional, you must select the relationship partner for both mappings.

Action: You must either go to the mapping selected as the relationship partner for this mapping and select this mapping as its relationship partner, or select a different relationship partner mapping for this mapping to maintain this mapping as its relationship partner.

594: There is a missing source XML field.

Cause: No field has been specified as the source XML field for this mapping.

Action: You must specify a source XML field.

595: There is a missing target XML field.

Cause: No field has been specified as the target XML field for this mapping.

Action: You must specify a target XML field.

600: A foreign key grouping element is required if there are multiple field pairs.

Cause: No foreign key grouping element is specified for this mapping and multiple field pairs.

Action: You must specify a foreign key grouping element.

601: The foreign key grouping element does not contain all foreign keys fields.

Cause: The specified foreign key grouping element does not contain all the foreign key fields.

Action: You must either remove the foreign key fields not contained in this foreign key grouping element, or pick a foreign key grouping element that contains all the foreign key fields.

602: A delete all interaction is specified, but the mapping is not private owned.

Cause: A delete all interaction is specified for this mapping, but the mapping is not private owned.

Action: You must either make the mapping private owned, or remove the delete all interaction.

610: At least one field pair must be specified, unless the mapping has no selection interaction and is read-only.

Cause: No field pairs are specified, and this mapping has a selection interaction specified and/or is not read-only.

Action: You must either specify a field pair for the mapping, or make the mapping read-only and remove the selection interaction.

XML Schema Errors (700 – 706)

This section lists TopLink Workbench XML schema errors.

701: A database table can only have one IDENTITY column defined.

Cause: You defined more than one identity column for this table.

Action: On the database table's **Columns** tab, leave only one identity (**Identity**) column. See ["Working With Column Properties"](#) on page 4-25.

702: A size is required for the column [column].

Cause: You did not specify any size for this column. The default size is 0.

Action: On the database table's **Columns** tab, specify the size (**Size**) for the column (field). See ["Working With Column Properties"](#) on page 4-25.

703: The reference [table reference] does not have any field pairs.

Cause: You added a reference for a table, but the reference does not include a key pair.

Action: On the database table's **References** tab, specify source and target field pairs for the table reference. See ["Creating Table References"](#) on page 4-27.

704: A key pair has not been completely specified for a reference.

Cause: A reference table is missing a complete key pair (source and target fields).

Action: You must specify a foreign key reference for the database table. On the database table's **References** tab, add a complete key pair. ["Creating Table References"](#) on page 4-27.

705: A development login has not been specified.

Cause: You created a relational TopLink Workbench project, but did not specify a development login.

Action: On the Database property sheet, select a **Development Login** from the available defined logins, or add a new login. See ["Configuring Development and Deployment Logins"](#) on page 23-6.

706: A deployment login has not been specified.

Cause: You created a relational TopLink Workbench project, but did not specify a deployment login.

Action: On the Database property sheet, select a **Deployment Login** from the available defined logins, or add a new login. See ["Configuring Development and Deployment Logins"](#) on page 23-6.

Session Errors (800 – 812)

This section lists the TopLink sessions XML errors.

801: [session name] Login - The connection URL has to be specified.

Cause: You have not specified a connection URL for the session (when using a database driver manager). Each session must have at least one login connection.

Action: On the session's **Login – Connection** tab, complete the **Driver URL** field. See ["Configuring a Session Login"](#) on page 77-4.

802: [session name] Login - The driver class has to be specified.

Cause: You have not specified a driver class for the session (when using a data source database driver).

Action: On the session's **Login – Connection** tab, complete the **Driver Class** field. See ["Configuring a Session Login"](#) on page 77-4.

803: [session or connection pool name]Login - Login - The data source name has to be specified.

Cause: You have not specified a driver class for the session login (when using a J2EE data source database driver).

Action: On the session's or connection pool's **Login – Connection** tab, complete the **Data Source** field. See ["Configuring a Session Login"](#) on page 77-4.

804: Login - Session Broker - It has to have at least one session, either a server or a database session.

Cause: You created a session broker but did not add any sessions. Each session broker must contain a session.

Action: On the session broker's **General – Sessions** tab, select a session to add to this broker. See ["Configuring Session Broker and Client Sessions"](#) on page 82-1.

805: [session name] Database Session - It has to have at least one XML file or a class specified.

Cause: Your database session does not have a primary project (an associated deployment XML file or Java class file).

Action: On the session's **Project – General** tab, complete the **Primary Project** field. See ["Configuring a Primary Mapping Project"](#) on page 77-2.

806: Login - The transport class has to be specified.

Cause: You selected a custom (user-defined) cache coordination type, but did not specify the transport class for cache coordination.

Action: On the session's **Cache Coordination** tab, complete the **Transport Class** field, or select a different cache coordination type. See ["Configuring a Coordinated Cache"](#) on page 91-1.

807: [session name] Login - The location of the log file has to be specified.

Cause: You are using standard logging and selected to have the log saved to a file, but did not select a file name and location.

Action: On the session's **Logging** tab, complete the **Log Location** field. See ["Configuring Logging"](#) on page 77-4.

811: [session or broker name] - An external transaction controller (JTA) has to be specified.

Cause: You selected a custom server platform, but did not specify the JTA for the platform.

Action: On the session or session broker's **General – Server Platform** tab, complete the **External Transaction Controller (JTA)** field. See ["Configuring the Server Platform"](#) on page 77-14.

812: [session or broker name] - A server class has to be specified.

Cause: You selected a custom server platform, but did not specify the server class for the platform.

Action: On the session or session broker's **General – Server Platform** tab, complete the **Server Class** field. See ["Configuring the Server Platform"](#) on page 77-14.

Common Classpath Problems

The following are some common TopLink Workbench error messages that may result from invalid classpath information. See ["Configuring Project Classpath"](#) on page 22-3 for more information.

The TopLink Workbench does not display the class(es) to import.

Cause: Your classes are not available for import on the **Select Classes** dialog box.

Action: Ensure that the class is in your project's classpath (on the project's **General** properties tab). Ensure that the class is in the `.zip` or `.jar` file. You cannot import compressed classes.

The TopLink Workbench generates an exception error when importing classes.

Cause: TopLink class import utility did not start correctly. One of the classes includes a static initialization method, which may cause the import utility to fail.

Action: Ensure that your project's classpath points to the root directory of your package hierarchy. For example, to import the `com.company.class` package in the `C:\classes\com\company` directory, your project classpath should be `C:\classes\`.

The TopLink Workbench fails to import the class, but does not generate an exception error.

Cause: The classpath containing your JDBC drivers should still be on your system CLASSPATH. TopLink Workbench classpath is for domain classes only.

Action: Ensure that you have properly indicated the directories that contain your domain class(es) to map on the project's **General** tab.

Data Source Problems

This section includes common problems that you may encounter when connecting TopLink to your data source. This section includes the following topics:

- [Database Connection Problems](#)

Database Connection Problems

This section describes common errors and problems you may encounter when communicating with or logging in to the database.

The class *[class]* was not found.

Cause: You attempted to log in to the database, but TopLink could not find the JDBC driver for the database.

Action: Ensure that the JDBC_CLASSPATH in the `setenv.cmd` file points to your JDBC driver JAR files. Verify that your PATH includes all files (for example, native `.dll` files) required by the driver. If the path to your JDBC driver JAR files contains spaces, then the path must be enclosed in double-quotes in the `setenv.cmd` file. For example:

```
set JDBC_CLASSPATH="C:\Program Files\some directory\driver.jar\"
```

For more information, see ["Configuring the TopLink Workbench Environment"](#) on page 4-2.

Username or password could be invalid.

Cause: TopLink was unable to log in to the database.

Action: Ensure that the **Username** and **Password** for the database are correct. Verify with your DBA that the database is set up and operating correctly.

You must define a development login.

Cause: You attempted to log in to the database from TopLink Workbench, but you did not define a development login.

Action: On the database property sheet, select a **Development Login**, or create a new **Defined Login**. See ["Configuring Login Information"](#) on page 23-5.

No database driver has been specified.

Cause: You attempted to log in to the database from TopLink Workbench, but you did not complete the login information.

Action: Complete all the required fields on the database property sheet for the selected development login. See ["Configuring Login Information"](#) on page 23-5.

Invalid URL specified.

Cause: You attempted to log in to the database from TopLink Workbench, but the URL is incorrect.

Action: Complete the **URL** field on the database property sheet for the selected development login. See ["Configuring Login Information"](#) on page 23-5.

Troubleshooting Application Deployment

This chapter discusses some of the general troubleshooting issues surrounding entity bean configuration and deployment. It lists many of the common exceptions and exception messages that you may encounter when you try to deploy persistent entity beans, using TopLink.

If you have any problems installing TopLink, using TopLink Workbench, or require more information on any run-time exceptions that are generated by TopLink, consult the appropriate documentation.

This chapter contains information on:

- [Generating Deployment JAR Files](#)
- [Common J2SE Deployment Exceptions](#)
- [Common BEA WebLogic Server Deployment Exceptions](#)
- [Common BEA WebLogic Server 6.1 Exceptions](#)
- [Common BEA WebLogic 7.0 Exceptions](#)
- [Common BEA WebLogic 8.1 Exceptions](#)
- [Common IBM WebSphere Application Server Exceptions](#)

Generating Deployment JAR Files

If you experience trouble generating the JAR files for deployment:

- Ensure all environment entries (such as classpath) are configured properly.
- Identify which step of the build is failing (copying, compiling, running EJB compiler, and so on.)

Running the EJB compiler utility involves several processes, such as compiling, code-generation, EJB compliance verification, compiling RMI stubs by running `rmic`, and so on. If an error occurs during execution of the EJB compile utility, try to determine which stage may be causing the failure.

For more information about the EJB compile utility, see the server documentation.

Common J2SE Deployment Exceptions

The following are some of the most common exceptions that may be encountered when you try to deploy a J2SE (non-J2EE) application using TopLink.

Classpath Exceptions

An exception may occur while setting up the connection pool. You must check the nested SQL exception to determine the cause of the exception. Typical reasons for this include:

- The location of JDBC driver is not specified on the classpath.
- The user name or password provided by the user is incorrect.
- The server URL or driver name is not properly specified.

Please consult the application server documentation and the JDBC driver documentation for help with this exception.

If the required TopLink JAR files have not been copied into the application extensions classpath, a classpath exception will be raised. You must ensure that the `toplink.jar` and `antlr.jar` files are copied into the `<Application Server install>\lib\app` directory.

If TopLink encounters problems finding the `deployment.project.xml` or `sessions.xml` files, a classpath exception will be raised. Refer to [Chapter 10, "Deploying a TopLink Application"](#).

JDeveloper places the `sessions.xml` file in the META-INF directory. To load the `sessions.xml` file in a non-J2EE application, you must explicitly provide the location of the `sessions.xml` file as shown in [Example 15-1](#).

Example 15-1 Location of the sessions.xml file

```
XMLSessionConfigLoader loader = new
    XMLSessionConfigLoader("META-INF/sessions.xml");
session = (DatabaseSession)SessionManager.getManager().getSession(
    loader, "MySession", Thread.currentThread().getContextClassLoader());
```

If the J2SE application is a single-user application, a `DatabaseSession` can be used instead of a `Server session`. This provides improved performance and reduces the number of database connections and login time.

Communication Exceptions

In situations where cache coordination is used, a communication exception may occur. A communication exception is a run-time exception that wraps all RMI, CORBA, or input and output exceptions that occur.

Refer to the [Communication Exceptions \(12000 - 12003\)](#) section on page 13-64 in the [TopLink Exception Reference](#) chapter for detailed information on communication exceptions that may occur.

Descriptor Validation Exceptions

A descriptor exception is a development exception that is raised when insufficient information is provided to the descriptor. The message that is returned includes the name of the descriptor or mapping that caused the exception. If a mapping within the descriptor caused the error, then the name and parameters of the mapping are part of the returned message.

The internal exception, mapping and descriptor appear only if TopLink has enough information about the source of the problem to provide this information.

Refer to the [Descriptor Exceptions \(1 – 200\)](#) section on page 13-2 in the [TopLink Exception Reference](#) chapter for detailed information on descriptor validation exceptions that may occur.

Common BEA WebLogic Server Deployment Exceptions

The following are some of the most common exceptions that are encountered when you deploy entity beans to a BEA WebLogic Server.

For more information about specific versions, see the following:

- [Common BEA WebLogic Server 6.1 Exceptions](#), on page 15-5
- [Common BEA WebLogic 7.0 Exceptions](#), on page 15-8
- [Common BEA WebLogic 8.1 Exceptions](#), on page 15-10

Assertion Error

Cause: This exception occurs if the `toplink.jar` file is not properly set on your classpath. The exception message returned is:

```
weblogic.utils.AssertionError: ***** ASSERTION FAILED
*****[Could not load class
'oracle.toplink.internal.ejb.cmp.wls.WlsCMPDeployer':
java.lang.ClassNotFoundException:
oracle.toplink.internal.ejb.cmp.wls.WlsCMPDeployerERROR: ejbc
found errors
```

Action: Ensure the `<ORACLE_HOME>/toplink/jlib/toplink.jar` file is specified on your system classpath.

Error Deploying Application

Cause: A `DeploymentException` has occurred.

Action: Refer to the specific error code. The error code appears in the square brackets in the exception message, such as `[TopLink-8001]`. These errors may refer to errors in the specification of the project location reading in the properties file or validation errors due to improper mappings.

Exception 8001

Cause: This error occurs if the TopLink project file is not specified in the `toplink-ejb-jar.xml` file. The exception message returned is:

```
<Error> <J2EE> <Error deploying application Account:Unable to
deploy EJB: AccountBean from Account.jar:LOCAL EXCEPTION
STACK:EXCEPTION [TOPLINK-8001] (TopLink (WLS CMP) - X.X.X):
oracle.toplink.ejb.DeploymentExceptionEXCEPTION DESCRIPTION:
No TopLink project was specified for this bean.
atoracle.toplink.ejb.DeploymentException.noPro
jectSpecified(DeploymentException.java:132) at
oracle.toplink.internal.ejb.cmp.ProjectDeployment.readProject
(ProjectDeployment.java:378)
```

Action: Ensure there is an entry in the `toplink-ejb-jar.xml` file for either the `project.xml` or `project-class`.

Exception 8016

Cause: This exception can occur if the location of the TopLink project file for the bean is not properly specified. The exception message returned is:

```
<Error> <J2EE> <Error deploying application Account:Unable to
deploy EJB: AccountBean from Account.jar:LOCAL EXCEPTION
STACK:EXCEPTION [TOPLINK-8016] (TopLink (WLS CMP) - X.X.X):
oracle.toplink.ejb.DeploymentExceptionEXCEPTION DESCRIPTION:
An error occurred while setting up the project:
[java.io.FileNotFoundException: Account.xml]INTERNAL
EXCEPTION: java.io.FileNotFoundException:
Account.xml:oracle.toplink.ejb.DeploymentException.errorCrea
tingProject(Unknown Source)
```

Action: Check the file name as it is specified in the `toplink-ejb-jar.xml` file, and the location of the project file on the file system.

Cannot Start Up Connection Pool

Cause: An exception has occurred while setting up the connection pool. The exception message returned is:

```
<Error> <JDBC> <Cannot startup connection pool "ejbPool"
weblogic.common.ResourceException: Cannot load driver class:
org.hsqldb.jdbcDriver>...
```

Action: Check the nested SQL exception to determine the cause of the exception. Typical reasons for this include:

- The JDBC driver is not on the classpath.
- The user name or password provided by the user is incorrect.
- The server URL or driver name is not properly specified.

Please consult the BEA WebLogic Server documentation and your JDBC driver documentation for help on the specific exception raised by BEA WebLogic.

Error Message

Cause: This problem occurs if you are using the GA version of BEA WebLogic Server 6.0. The exception message returned is:

```
weblogic.utils.AssertionError: ***** ASSERTION FAILED
*****[Could not create an instance of class 'null':
java.lang.NullPointerException at
java.lang.Class.forName0(Native Method) at
java.lang.Class.forName(Class.java:120) at
weblogic.ejb20.persistence.PersistenceType. loadClass
(PersistenceType.java:309)
```

Action: Upgrade to WebLogic Server 6.0 (Service Pack 1) or a higher release.

EJBC Found Errors

Cause: This error occurs if the `toplink.jar` file is not properly set on your classpath. The exception message returned is:

```
ERROR: ejbc found errors Error from ejbc: Error while loading
persistence resource TopLink_CMP_Descriptor.xml Make sure
that the persistence type is in your classpath.
```

Action: Ensure the `<ORACLE_HOME>/toplink/jlib/toplink.jar` file is specified on your system classpath.

EJB Deployment Exception

Cause: This exception occurs if the location of the TopLink project file for the bean is not properly specified. The exception message returned is:

```
weblogic.ejb20.EJBDeploymentException: Error Deploying CMP
EJB;; nested exception is:
weblogic.ejb20.cmp.rdbms.RDBMSException: An error occurred
setting up the project: EXCEPTION [TOPLINK-13000] (vX.X
[TopLink for WebLogic X.X] JDK1.2):
oracle.toplink.xml.XMLDataStoreException EXCEPTION
DESCRIPTION: File not found...
```

Action: Check the file name as it is specified in the `toplink-ejb-jar.xml` file, and the location of the TopLink project file on the file system.

Deploying EJB Component

Cause: A typical cause of this exception is that the `toplink-ejb-jar.xml` file is referring to a local DTD file using a file name or location that is incorrect. The exception message returned is:

```
Error deploying EJB Component:...
weblogic.ejb20.EJBDeploymentException: Exception in EJB
Deployment; nested exception is: Error while deploying
bean..., File... Not Found at
weblogic.ejb20.persistence.PersistenceType.setup
Deployer(PersistenceType.java:273)
```

Action: Ensure that all XML files refer to valid DTD files and locations.

Cannot Start Up Connection Pool ejbPool

Cause: This exception is raised to indicate that an error has occurred while setting up the connection pool. The exception message returned is:

```
Cannot startup connection pool "ejbPool"
weblogic.common.ResourceException: Could not create pool
connection. The DBMS driver exception was:...
```

Action: Check the nested SQL exception to determine the cause of the error. Typical problems include:

- The driver is not on the classpath.
- The user name or password is incorrect.
- The server URL or driver name is not properly specified.

Please consult the BEA WebLogic Server documentation and your JDBC driver documentation for help on the specific exception raised by BEA WebLogic.

Other Exceptions

Occasionally, changes made to the server's configuration file (`config.xml`) do not appear to be applied when the server is restarted. If this occurs, try removing the temporary directories created by BEA WebLogic Server. You can find them under the `wlserver6.1` directory, at the same level as the `config` directory.

Common BEA WebLogic Server 6.1 Exceptions

The following are a few of the most common exceptions you may encounter when deploying JAR files with TopLink and BEA WebLogic Server 6.1:

- [Development Exceptions](#)
- [Deployment and Run Time Exceptions](#)

Development Exceptions

Missing Persistence Type

Cause: There is no entry in the `persistence.install` file for TopLink CMP. This may occur if the TopLink installation was interrupted or a BEA WebLogic Server Service Pack was applied. The exception message returned is:

```
ERROR: Error from ejbc: Persistence type 'TopLink_CMP_2_0'
with version 'X.X' which is referenced in bean 'Account' is
not installed. The installed persistence types are:
(WebLogic_CMP_RDBMS, 6.0), (WebLogic_CMP_RDBMS, 5.1.0).ERROR:
ejbc found errors
```

Action: In the `<WebLogic InstallDir>/wlserver6.1/lib/persistence` directory, edit the `persistence.install` file to add a new line `TopLink_CMP_Descriptor.xml`, or replace your existing `persistence.install` file with the version of the file in the `<ORACLE_HOME>/toplink/config` directory.

Error Loading Persistence Resource

Cause: The `toplink.jar` file is not properly set in your classpath. The exception message returned is:

Error while loading persistence resource TopLink_CMP_Descriptor.xml Make sure that the persistence type is in your classpath.

Action: Ensure that the classpath includes the `<ORACLE_HOME>/toplink/jlib/toplink.jar` file.

Wrong BEA WebLogic Version

Cause: You are trying to compile your code using BEA WebLogic Server 6.0. The exception message returned is:

```
C:\<ORACLE_HOME>\toplink\examples\weblogic\wls61\
examples\ejb\cmp20\singlebean\Account.java:10: cannot resolve
symbol symbol : class EJBLocalObjectlocation: interface
examples.ejb.cmp20.singlebean.Accountpublic interface Account
extends EJBLocalObject {
```

Action: Compile using BEA WebLogic Server 6.1.

Deployment and Run Time Exceptions

Missing Persistence Type

Cause: There is no entry in the `persistence.install` file for TopLink CMP. This may occur if the TopLink installation was interrupted, or a BEA WebLogic Server Service Pack was applied. The exception message returned is:

Persistence type 'TopLink_CMP_2_0' with version 'X.X' which is referenced in bean 'Account' is not installed. The installed persistence types are: (WebLogic_CMP_RDBMS, 6.0), (WebLogic_CMP_RDBMS, 5.1.0).

Action: In the `<WebLogic InstallDir>/wlserver6.1/lib/persistence` directory, edit the `persistence.install` file to add a new line: `TopLink_CMP_Descriptor.xml`. You can also replace your existing `persistence.install` file with the version of the file in the `<ORACLE_HOME>/toplink/config` directory.

Error Loading Persistence Resource

Cause: The `toplink.jar` file is not properly set in your classpath. The exception message returned is:

```
<DATE and TIME> <Error> <J2EE> <Error deploying application
ejb20_cmp_order:Unable to deploy EJB: C:\<ORACLE_
HOME>\toplink\examples\weblogic\wls61\server\config\TopLink_
Domain\applications\wlnotdelete\wlap64280\ejb20_cmp_order.jar
from ejb20_cmp_order.jar:Error while loading persistence
resource TopLink_CMP_Descriptor.xml Make sure that the
persistence type is in your
classpath.atweblogic.ejb20.persistence.InstalledPersistence.i
nitialize(InstalledPersistence.java:214)atweblogic.ejb20.pers
istence.InstalledPersistence.getInstalledType(InstalledPersis
tence.java:113)
```

Action: Ensure that the classpath includes the <ORACLE_
HOME>/toplink/jlib/toplink.jar file.

Wrong Persistence Version

Cause: You may be using a persistence-version meant for BEA WebLogic Server 7.0. The exception message returned is:

```
DATE and TIME> <Error> <J2EE> <Error deploying application
ejb20_cmp_account:Unable to deploy EJB: Account from ejb20_
cmp_
account.jar:java.lang.AbstractMethodErroratweblogic.ejb20.dep
loyer.ClientDrivenBeanInfoImpl.deploy(ClientDrivenBeanInfoImp
l.java:807)atweblogic.ejb20.deployer.Deployer.deployDescripto
r(Deployer.java:1234)atweblogic.ejb20.deployer.Deployer.deplo
y(Deployer.java:947)atweblogic.j2ee.EJBComponent.deploy(EJBCo
mponent.java:30)
```

Action: Use a persistence-version of 4.0.

Cannot Start Up Data Source

Cause: An exception has occurred while setting up the data source. The exception message returned is:

```
EXCEPTION [TOPLINK-7060] (TopLink (WLS
CMP)-X.X):oracle.toplink.exceptions.ValidationExceptionEXCEPT
ION DESCRIPTION: Cannot acquire datasource
[jdbc/ejbNonJTSDatasource].INTERNAL EXCEPTION:
javax.naming.NameNotFoundException: Unable to resolve
jdbc.ejbNonJTSDatasource Resolved: '' Unresolved:'jdbc' ;
remaining name 'ejbNonJTSDatasource'
```

Action: Check the nested SQL exception to determine the cause of the exception. For more information, see ["7060: CANNOT_ACQUIRE_DATA_SOURCE"](#) on page 13-46. For more information on a specific exception raised by WebLogic, see the BEA WebLogic Server documentation and your JDBC driver documentation.

Wrong WebLogic Version

Cause: You are trying to compile your code using BEA WebLogic Server 6.0. The exception message returned is:

```
<DATE and TIME> <Error> <Management> <Error parsing XML
descriptor for application TopLink_Domain:Name=ejb20_cmp_
account,
Type=Applicationweblogic.xml.process.ProcessorFactoryExceptio
n: Could not locate processor for public id = "-//Sun
Microsystems, Inc.//DTD J2EE Application
1.3//EN"atweblogic.xml.process.ProcessorFactory.getProcessor(
```

ProcessorFactory.java:181)atweblogic.xml.process.ProcessorFactory.getProcessor(ProcessorFactory.java:164)

Action: Compile using BEA WebLogic Server 6.1.

Common BEA WebLogic 7.0 Exceptions

The following are a few of the most common exceptions you may encounter when deploying JAR files with TopLink and BEA WebLogic Server 7.0.

- [Development Exceptions](#)
- [Deployment Exceptions](#)

Development Exceptions

Missing Persistence Type

Cause: There is no entry in the `persistence.install` file for TopLink CMP. This may occur if the TopLink installation was interrupted, or a BEA WebLogic Server Service Pack was applied. The exception message returned is:

Persistence type 'TopLink_CMP_2_0' with version 'X.0' which is referenced in bean 'Account' is not installed. The installed persistence types are: (WebLogic_CMP_RDBMS, 6.0), (WebLogic_CMP_RDBMS, 5.1.0), (WebLogic_CMP_RDBMS, 7.0)ERROR: ejbc found errors

Action: In the `<WebLogic InstallDir>/weblogic700/lib/persistence` directory, edit the `persistence.install` file to add a new line: `TopLink_CMP_Descriptor.xml`. You can also replace your existing `persistence.install` file with the version of the file in the `<ORACLE_HOME>/toplink/config` directory.

Missing Persistence Type

Cause: There is no entry in the `persistence.install` file for TopLink CMP. This may occur if the TopLink installation was interrupted, or a BEA WebLogic Server Service Pack was applied. The exception message returned is:

ERROR:
atweblogic.ejb20.persistence.InstalledPersistence.initialize(InstalledPersistence.java:214)atweblogic.ejb20.persistence.InstalledPersistence.getInstalledType(InstalledPersistence.java:113)atweblogic.ejb20.deployer.MBeanDeploymentInfoImpl.getPersistenceType(MBeanDeploymentInfoImpl.java:584)

Action: In the `<WebLogic InstallDir>/weblogic700/lib/persistence` directory, edit the `persistence.install` file to add a new line: `TopLink_CMP_Descriptor.xml`. You can also replace your existing `persistence.install` file with the version of the file in the `<ORACLE_HOME>/toplink/config` directory.

Wrong WebLogic Version

Cause: You are trying to compile your JAR file using BEA WebLogic Server 6.1. The exception message returned is:

ERROR: Error processing 'META-INF/weblogic-ejb-jar.xml': The public id, "-//BEA Systems, Inc.//DTD WebLogic 7.0.0 EJB/EN", specified in the XML document is invalid. Use one of the following valid public ids:"-//BEA Systems, Inc.//DTD

WebLogic 5.1.0 EJB//EN"-//BEA Systems, Inc.//DTD WebLogic 6.0.0 EJB//EN"ERROR: ejbc found errors

Action: Compile using BEA WebLogic Server 7.0.

Deployment Exceptions

Missing Persistence Type

Cause: There is no entry in the `persistence.install` file for TopLink CMP. This may occur if the TopLink installation was interrupted or a BEA WebLogic Server Service Pack was applied. The exception message returned is:

Error from ejbc: Persistence type 'TopLink_CMP_2_0' with version 'X.0' which is referenced in bean 'Account' is not installed. The installed persistence types are: (WebLogic_CMP_RDBMS, 6.0), (WebLogic_CMP_RDBMS, 5.1.0), (WebLogic_CMP_RDBMS, 7.0). Persistence type 'TopLink_CMP_2_0' with version 'X.0' which is referenced in bean 'Account' is not installed. The installed persistence types are: (WebLogic_CMP_RDBMS, 6.0), (WebLogic_CMP_RDBMS, 5.1.0), (WebLogic_CMP_RDBMS, 7.0)

Action: In the `<WebLogic InstallDir>/weblogic7.0/lib/persistence` directory, edit the `persistence.install` file to add a new line: `TopLink_CMP_Descriptor.xml`. You can also replace your existing `persistence.install` file with the version of the file in the `<ORACLE_HOME>/toplink/config` directory.

Error Loading Persistence Resource

Cause: The `toplink.jar` file is not properly set in your classpath. The exception message returned is:

```
java.lang.NullPointerExceptionatweblogic.ejb20.deployer.EJBDeployer.deactivate(EJBDeployer.java:1513)atweblogic.ejb20.deployer.EJBDeployer.undeploy(EJBDeployer.java:301)atweblogic.ejb20.deployer.Deployer.deploy(Deployer.java:875)atweblogic.j2ee.EJBComponent.deploy(EJBComponent.java:70)
```

Action: Ensure that the classpath includes the `<ORACLE_HOME>/toplink/jlib/toplink.jar` file.

Cannot Start Up Data Source

Cause: An exception has occurred while setting up the data source. The exception message returned is:

```
EXCEPTION [TOPLINK-7060] (TopLink (WLS CMP) - X.X.X): oracle.toplink.exceptions.ValidationExceptionEXCEPTION DESCRIPTION: Cannot acquire datasource [jdbc/ejbNonJTSDatasource].INTERNAL EXCEPTION: javax.naming.NameNotFoundException: Unable to resolve jdbc/ejbNonJTSDatasource Resolved: '' Unresolved:'jdbc' ; remaining name 'ejbNonJTSDatasource'
```

Action: Check the nested SQL exception to determine the cause of the exception. For more information, see ["7060: CANNOT_ACQUIRE_DATA_SOURCE"](#) on page 13-46. For more information on a specific exception raised by WebLogic, see the BEA WebLogic Server documentation and your JDBC driver documentation.

Common BEA WebLogic 8.1 Exceptions

The following are a few of the most common exceptions you may encounter when deploying JAR files with TopLink and BEA WebLogic Server 8.1.

- [Development Exceptions](#)
- [Deployment Exceptions](#)

Development Exceptions

Missing Persistence Type

Cause: There is no entry in the `persistence.install` file for TopLink CMP. This may occur if the TopLink installation was interrupted, or a BEA WebLogic Server Service Pack was applied. The exception message returned is:

Persistence type 'TopLink_CMP_2_0' with version 'X.0' which is referenced in bean 'Account' is not installed. The installed persistence types are: (WebLogic_CMP_RDBMS, 7.0), (WebLogic_CMP_RDBMS, 6.0), (WebLogic_CMP_RDBMS, 5.1.0).

Action: In the `<WebLogic InstallDir>/weblogic81/lib/persistence` directory, edit the `persistence.install` file to add a new line: `TopLink_CMP_Descriptor.xml`. You can also replace your existing `persistence.install` file with the version of the file in the `<ORACLE_HOME>/toplink/config` directory.

Missing CMP entry in Persistence File

Cause: There is no entry in the `persistence.install` file for TopLink CMP. This may occur if the TopLink installation was interrupted, or a BEA WebLogic Server Service Pack was applied. The exception message returned is:

ERROR: ejbc couldn't invoke compiler

Action: In the `<WebLogic InstallDir>/weblogic81/lib/persistence` directory, edit the `persistence.install` file to add a new line: `TopLink_CMP_Descriptor.xml`. You can also replace your existing `persistence.install` file with the version of the file in the `<ORACLE_HOME>/toplink/config` directory.

Error Loading Persistence Resource

Cause: The `toplink.jar` file is not properly set in your classpath. The exception message returned is:

Error occurred while loading persistence resource `TopLink_CMP_Descriptor.xml`. Make sure that the persistence type is in your classpath.

ERROR: ejbc couldn't invoke compiler

Action: Ensure that the classpath includes the `<ORACLE_HOME>/toplink/jlib/toplink.jar` file.

Wrong WebLogic Version

Cause: You are trying to compile your EJB JAR file using BEA WebLogic Server 7.0. The exception message returned is:

ERROR: ejbc found errors while processing the descriptor for `std_cmp20-singlebean.jar`: ERROR: ejbc found errors while processing 'META-INF/weblogic-ejb-jar.xml': The public id, `"-//BEA Systems, Inc.//DTD WebLogic 8.1.0 EJB//EN"`, specified in the XML document is invalid. Use one of the following valid public ids: `"-//BEA Systems, Inc.//DTD WebLogic 5.1.0`

EJB//EN""-//BEA Systems, Inc.//DTD WebLogic 6.0.0
 EJB//EN""-//BEA Systems, Inc.//DTD WebLogic 7.0.0
 EJB//EN"ERRORRejbc found errors

Action: Compile using BEA WebLogic Server 8.1.

Deployment Exceptions

Missing Persistence Type

Cause: There is no entry in the `persistence.install` file for TopLink CMP. This may occur if the TopLink installation was interrupted or a BEA WebLogic Server Service Pack was applied. The exception message returned is:

```
Error Deployer BEA-149201 Failed to complete the deployment
task with ID 0 for the application _appsdirear. weblogic.management.ApplicationException:
Exception:weblogic.management.ApplicationException: prepare
failed for cmp20-singlebean.jarModule: cmp20-singlebean.jar
Error: Exception preparing module:
EJBModule(cmp20-singlebean.jar,status=NEW)Persistence type
'TopLink_CMP_2_0' with version 'X.0 which is referenced in
bean 'Account' is not installed. The installed persistence
types are: (WebLogic_CMP_RDBMS, 7.0), (WebLogic_CMP_RDBMS,
6.0), (WebLogic_CMP_RDBMS, 5.1.0)
```

Action: In the `<WebLogic InstallDir>/weblogic81/lib/persistence` directory, edit the `persistence.install` file to add a new line: `TopLink_CMP_Descriptor.xml`. You can also replace your existing `persistence.install` file with the version of the file in the `<ORACLE_HOME>/toplink/config` directory.

Error Loading Persistence Resource

Cause: The `toplink.jar` file is not properly set in your classpath. The exception message returned is:

```
Error Deployer BEA-149201 Failed to complete the deployment
task with ID 2 for the application _appsdirear.weblogic.management.ApplicationException:
Exception:weblogic.management.ApplicationException: prepare
failed for cmp20-relationships.jarModule:
cmp20-relationships.jar Error: Exception preparing module:
EJBModule(cmp20-relationships.jar,status=NEW) Unable to
deploy EJB:.\TopLink_Demos\stage\_appsdirear\cmp20-relationships_ear\cmp20-relationships.jar from
cmp20-relationships.jar: [EJB:011004]Error occurred while
loading persistence resource TopLink_CMP_Descriptor.xml. Make
sure that the persistence type is in your classpath.at
weblogic.ejb20.persistence.InstalledPersistence.initialize(In
stalledPersistence.java:212)at
weblogic.ejb20.persistence.InstalledPersistence.getInstalledT
ype(InstalledPersistence.java:114)
```

Action: Ensure that the classpath includes the `<ORACLE_HOME>/toplink/jlib/toplink.jar` file.

Common IBM WebSphere Application Server Exceptions

When the IBM WebSphere application server is started, it attempts to deploy the JAR files that are specified for deployment within the application server.

Exceptions that occur when the server is started are usually configuration problems that involve classpath issues, environment variable configuration, and database login configuration. Review the IBM WebSphere application server documentation after starting the server.

This section contains some of the exceptions that can be encountered when running the IBM WebSphere application server, along with their possible causes and recommended solutions.

Class Not Found Exception

Cause: The class is not included on the WebSphere application extensions classpath or in the EJB or WAR module.

Action: Ensure that all required classes are included in the correct location. For more information about classpath locations, see the *IBM WebSphere InfoCenter*.

Class Not Found Exception

Cause: The required TopLink JAR files have not been copied into the application extensions classpath.

Action: Ensure that the `toplink.jar` and `antlr.jar` files are copied into the `<WebSphere install>\lib\app` directory.

oracle.toplink.exceptions.DatabaseException

Cause: A TopLink exception has occurred.

Action: Refer to the specific exception code. The exception code appears in the square brackets in the exception message, such as `[TopLink-1016]`. Exceptions observed here may be exceptions in reading in the properties file, or validation errors due to improper mappings.

Refer to the [Database Exceptions \(4002 – 4018\)](#) section on page 13-27 in the [TopLink Exception Reference](#) chapter for detailed information on database exceptions that may occur; refer to the [Communication Exceptions \(12000 - 12003\)](#) section on page 13-64 in the [TopLink Exception Reference](#) chapter for detailed information on communication exceptions that may occur.

Exception [6066]

Cause: A bean was created outside of a transaction and then a second bean was created either in or out of a transaction. The exception message returned is:

```
oracle.toplink.exceptions.QueryException: The object <Object>
of class <class> with identity hashcode <hashcode> is not
from this Unit of Work object space but the parent session's.
The object was never registered in this Unit of Work, but
read from the parent session and related to an object
registered in the Unit of Work. Ensure that you are correctly
registering your objects. If you are still having problems,
you can use the UnitOfWork.validateObjectSpace() method to
help debug where the error occurred. Please see the manual
and FAQ for more information.
```

Action: Ensure that all bean creation is performed within the context of a transaction.

Cause: The bean was not removed during `ejbPassivate` method.

Action: Ensure that the `ejbPassivate` method removes the bean.

Cause: A bean-to-object relationship is not privately owned.

Action: Ensure that all bean-to-object relationships are privately owned.

Exception [7064]

Cause: An incorrect primary key object is being used with a bean. The exception message returned is:

```
oracle.toplink.exceptions.ValidationException: Exception
occurred in reflective EJB bean primary key extraction,
please ensure your primary key object is defined correctly:
key = 301, bean = <beanName>
```

Action: Ensure that you are using the correct primary key object for a bean.

Exception [7066]

Cause: An attempt was made to create or remove a bean outside of a transaction. The exception message returned is:

```
oracle.toplink.exceptions.ValidationException: Cannot create
or remove beans unless a JTS transaction is present,
bean=<bean>
```

Action: Ensure that all removal and creation of beans is performed within a transaction.

Exception [7068]

Cause: The project class that is specified in the `toplink.properties` file for the session specified on the `toplink_session_name` environment variable cannot be found. The exception message returned is:

```
oracle.toplink.exceptions.ValidationException: The project
class <projectclass> was not found for the <toplink_session_
name> using default class loader.
```

Action: Ensure that the project class provided in the exception is on the IBM WebSphere application server dependent classpath.

Exception [7069]

Cause: An amendment method was called, but cannot be found. The exception message returned is:

```
oracle.toplink.exceptions.ValidationException: An exception
occurred looking up or invoking the project amendment method,
<amendmentMethod> on the class <amendmentClass>;
```

Action: Ensure that the required amendment method exists on the class that is specified.

Exception [7070]

Cause: The `toplink.properties` file cannot be found. The exception message returned is:

```
oracle.toplink.exceptions.ValidationException: A
toplink.properties resource bundle must be located on the
classpath in a TopLink directory.
```

Action: Ensure that the location of the `toplink.properties` file is on the classpath.

Exception [7079]

Cause: The descriptor that is listed was not found in the session that is specified on the deployment descriptor. The exception message returned is:

EXCEPTION DESCRIPTION: The descriptor for [<bean class>] was not found in the session [<session name>]. Check the project being used for this session.

Action: Ensure that the project that is specified in the `toplink-ejb-jar.xml` file is the desired project. Also check that the project includes a descriptor for the missing bean class.

Exception [7101]

Cause: The `toplink-ejb-jar.xml` file was not found. The exception message returned is:

No "meta-inf/toplink-ejb-jar.xml" could be found in your classpath. The CMP session could not be read in from file.

Action: Ensure that the `toplink-ejb-jar.xml` file is located in the deployed `ejb-jar` file under the `meta-inf` directory.

Exception [9002]

Cause: The project class that is specified for the session in the `toplink-ejb-jar.xml` file cannot be found. The exception message returned is:

EXCEPTION [TOPLINK-9002] (TopLink - X.X.X):
oracle.toplink.exceptions.SessionLoaderException
EXCEPTION DESCRIPTION: Unable to load Project class [<project class>].

Action: Ensure that the project class has been included in the deployed JAR file with the entity beans.

Problems at Run Time

This section lists some of the common exceptions that can occur at run time when using the TopLink CMP for IBM WebSphere application server.

Exception [6026]

Cause: A required named query does not exist. The exception message returned is:

oracle.toplink.exceptions: Query is not defined

Action: Implement the named query. The stack trace of the exception contains the finder method that failed.

Common TopLink for IBM WebSphere Deploy Tool Exceptions

This section lists common exceptions that may occur when running the TopLink for IBM WebSphere application server Deploy Tool.

Class Not Found Exceptions

Cause: The class that is specified was not found; it is not included on the deploy tool classpath or the system classpath.

Action: Ensure that all required classes are included on the correct classpath. For more information about classpath setup, see the *IBM WebSphere Getting Started* document.

Note: The Deploy Tool calls external IBM classes to generate deployed code. Any exceptions that are thrown from these classes are written to `System.out`. Check **Tracing** to view the most detailed information.

Part VI

TopLink Tutorial

This part describes tutorial information to help you become familiar with TopLink and building applications with it. It contains the following chapters.

- [Chapter 16, "Planning the Tutorial Project"](#)

This describes the object model and data source used in the tutorial project.

- [Chapter 17, "Building the Tutorial Project"](#)

This chapter describes the steps required to build and map your TopLink Workbench tutorial project.

- [Chapter 18, "Deploying the Tutorial Project"](#)

This chapter describes the steps to create deployment and sessions information for the tutorial project.

Planning the Tutorial Project

This section contains the following information:

- [Overview](#)
- [Understanding the Tutorial Project](#)
- [Data Model](#)

Overview

In this tutorial, you will create an Employment Management System for your company to be deployed in a three-tier environment using JSP servlets. By the end of the tutorial, you will know how to store data from a Java class into a relational database, and to access existing database information from Java classes.

Specifically, you will learn how to:

- Create a new TopLink Workbench project
- Enable and add Java classes
- Create and import database tables
- Associate descriptors to tables
- Implement direct and relationship mappings (including self-relationships)
- Use indirection and value holders
- Apply several advanced features such as multiple tables and optimistic locking
- Write TopLink queries
- Create and manage database sessions
- Package and deploy a TopLink application

Before you Begin

Before building the tutorial project, do the following:

1. Download the `TopLink_Tutorial.zip` file to a host from which you can access your database and application server.

You can obtain the `TopLink_Tutorial.zip` file from:

http://www.oracle.com/technology/products/ias/toplink/doc/1013/MAIN/_tutorial/TopLink_Tutorial.zip

This file contains all required tutorial files, including Java source, JSP, SQL scripts to create and populate the database, and Ant scripts to build the EAR file and deploy it to OC4J.

2. UnZIP the `TopLink_Tutorial.zip` file to a local directory (`<TUTORIAL_HOME>`).

For more information about the contents of the `TopLink_Tutorial.zip` file, see the `<TUTORIAL_HOME>\readme.htm` file.

3. Configure tutorial environment settings file (`<TUTORIAL_HOME>\env.properties`) by following the instructions in the `<TUTORIAL_HOME>\config.htm` file.
4. Configure the TopLink Workbench environment (see ["Configuring the TopLink Workbench Environment"](#) on page 4-2).
5. Create and populate the database tables used in this tutorial (see ["Creating and Populating Database Tables"](#) on page 16-7).

Understanding the Tutorial Project

This tutorial project manages an employee database for your company. The system tracks each employee's name, address, and telephone number. The system also tracks employees' current projects, managers, and contract period.

This section includes information on the following topics:

- [Source Files](#)
- [Data Model](#)

Source Files

The tutorial uses the Java source files in the `<TUTORIAL_HOME>\src` directory, and JSP in the `<TUTORIAL_HOME>\jsp` directory. Source files are divided into two packages:

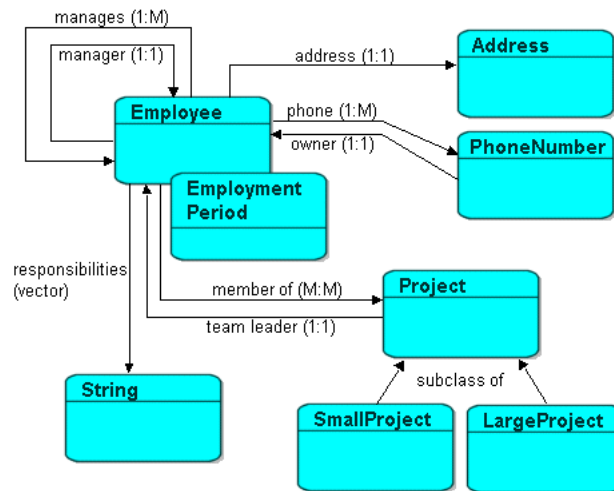
- [Object Model Classes](#)
- [Application Classes](#)

You are encouraged to experiment with these Java source files and JSP. Using the `<TUTORIAL_HOME>\build` script (see ["Step 3: Packaging for Deployment"](#) on page 18-8) you can easily compile your changes and recreate deployment artifacts.

Object Model Classes

The Java classes in the `examples.servlet.jsp.model` package represent the domain objects used in the tutorial application.

[Figure 16-1](#) illustrates the object model for this tutorial.

Figure 16-1 Tutorial Object Model

Java source files in this package include the following:

- **Employee**—Represents both full-time ACME employees and temporary contractors working on ACME projects. It includes the employees' personal information as well as references to their home addresses and telephone numbers.
- **Address**—Represents the employee's home address. The class contains country, street, city, province, and postal code information.
- **PhoneNumber**—Contains the telephone number(s) for each employee and contractor (number, area code, and type information). The class also includes a reference to the employee who owns the telephone number.
- **EmploymentPeriod**—Defines the contract term for contractors and the hire date for ACME employees. Each **Employee** class has an **EmploymentPeriod**.
- **ResponsibilityList**—Each **Employee** has a collection of text that describes the employee's job.
- **Project**—Maintains information about a particular project and the people working on it. The **Project** class contains two subclasses: **LargeProject** and **SmallProject**. Each **Employee** can be involved in more than one project.
- **TeamLeader**—Each **Project** can have a team leader (the **Employee** responsible for the project).
- **Manager**—Each **Employee** may have a manager and a collection of managed employees.

Application Classes

The Java classes in the `examples.servlet.jsp` package are instances of **HttpServlet** responsible for handling HTTP requests. They do the work of the application.

All application classes extend **JSPDemoServlet.java**. This base class is responsible for acquiring a **TopLink** session at run time with **JSPDemoServlet** method `getSession` and also provides implementations for two frequently used **TopLink** queries:

JSPDemoServlet method `readEmployee(String ID)` allows derived classes to read an `Employee` from the database for viewing only.

JSPDemoServlet method `readEmployee(String ID, UnitOfWork unitOfWork)` allows derived classes to read an `Employee` from the database within the transactional context of a `TopLink UnitOfWork`.

Examine the application classes to see how TopLink API is used to handle persistence operations within the tutorial application.

Data Model

The employee management system stores the employee data in the following database tables:

- [Table 16–1, "ADDRESS Table"](#)
- [Table 16–2, "EMPLOYEE Table"](#)
- [Table 16–3, "LPROJECT Table"](#)
- [Table 16–4, "PHONE Table"](#)
- [Table 16–5, "PROJ_EMP Relation Table Between PROJECT and EMPLOYEE"](#)
- [Table 16–6, "PROJECT Table"](#)
- [Table 16–7, "RESPONS Table"](#)
- [Table 16–8, "SALARY Table"](#)
- [Table 16–9, "SEQUENCE Table"](#)

The column types listed here are generic; the actual column types depend on the database used.

[Table 16–10](#) describes how each object model class (see ["Object Model Classes"](#) on page 16-2) relates to these database tables.

Before building this tutorial application, create and populate the following tables in your database (see ["Creating and Populating Database Tables"](#) on page 16-7).

Table 16–1 ADDRESS Table

Column Name	Column Type	Details
ADDRESS_ID	NUMERIC(15)	Primary key
CITY	VARCHAR(80)	
COUNTRY	VARCHAR(80)	
P_CODE	VARCHAR(20)	
PROVINCE	VARCHAR(80)	
STREET	VARCHAR(80)	

Table 16–2 EMPLOYEE Table

Column Name	Column Type	Details
ADDR_ID	NUMERIC(15)	FK reference to ADDRESS.ADDRESS_ID
EMP_ID	NUMERIC(15)	Primary key FK reference to SALARY.EMP_ID

Table 16-2 (Cont.) EMPLOYEE Table

Column Name	Column Type	Details
END_DATE	DATE	
END_TIME	TIME	
F_NAME	VARCHAR(40)	
GENDER	CHAR(1)	
L_NAME	VARCHAR(40)	
MANAGER_ID	NUMERIC(15)	FK reference to EMPLOYEE.EMP_ID
START_DATE	DATE	
START_TIME	TIME	
VERSION	NUMERIC(15)	

Table 16-3 LPROJECT Table

Column Name	Column Type	Details
BUDGET	NUMERIC(10,2)	
MILESTONE	DATE	
PROJ_ID	NUMERIC(15)	Primary key

Table 16-4 PHONE Table

Column Name	Column Type	Details
AREA_CODE	CHAR(3)	
P_NUMBER	CHAR(7)	
TYPE	VARCHAR(15)	Primary key
EMP_ID	NUMERIC(15)	Primary key FK reference to EMPLOYEE.EMP_ID

Table 16-5 PROJ_EMP Relation Table Between PROJECT and EMPLOYEE

Column Name	Column Type	Details
EMP_ID	NUMERIC(15)	Primary key FK reference to EMPLOYEE.EMP_ID
PROJ_ID	NUMERIC(15)	Primary key FK reference to PROJECT.PROJ_ID

Table 16-6 PROJECT Table

Column Name	Column Type	Details
DESCRIP	VARCHAR(200)	
LEADER_ID	NUMERIC(15)	FK reference to EMPLOYEE.EMP_ID
PROJ_ID	NUMERIC(15)	Primary key
PROJ_NAME	VARCHAR(30)	
PROJ_TYPE	CHAR(1)	

Table 16–6 (Cont.) PROJECT Table

Column Name	Column Type	Details
VERSION	NUMERIC(15)	

Table 16–7 RESPONS Table

Column name	Column type	Details
DESCRIP	VARCHAR(200)	
EMP_ID	NUMERIC(15)	Primary key FK reference to EMPLOYEE.EMP_ID

Table 16–8 SALARY Table

Column Name	Column Type	Details
EMP_ID	NUMERIC(15)	Primary key
SALARY	NUMERIC(22)	

Table 16–9 SEQUENCE Table

Column Name	Column Type	Details
SEQ_COUNT	NUMERIC(38)	
SEQ_NAME	VARCHAR(50)	

Table 16–10 Relationships Between Classes and Database Table

Database Table and Column	Java Class and Attribute	Database Type	Java Type
EMPLOYEE	Employee		
EMP_ID	id	NUMERIC(15)	BigDecimal
F_NAME	firstName	VARCHAR(40)	String
L_NAME	lastName	VARCHAR(40)	String
ADDR_ID	address	NUMERIC(15)	Address
<i>Not applicable</i>	phoneNumbers	NA	Vector
GENDER	gender	CHAR(1)	String
START_TIME	normalHours[0]	TIME	Time
END_TIME	normalHours[1]	TIME	Time
MANAGER_ID	manager	NUMERIC(15)	Employee
NA	managedEmployees	NA	Vector
NA	projects	NA	Vector
See Employment Period	period	NA	EmploymentPeriod
SALARY	Employee		
EMP_ID	NA	NUMERIC(15)	NA
SALARY	salary	NUMERIC(10)	int
EMPLOYEE	EmploymentPeriod		

Table 16–10 (Cont.) Relationships Between Classes and Database Table

Database Table and Column	Java Class and Attribute	Database Type	Java Type
START_DATE	startDate	DATE	Date
END_DATE	endDate	DATE	Date
RESPONS	Employee		
EMP_ID	NA	NUMERIC(15)	NA
DESCRIP	responsibilitiesList	VARCHAR(200)	String
PROJECT	LargeProject and SmallProject	NA	NA
PROJ_ID	id	NUMERIC(15)	BigDecimal
DESCRIP	description	VARCHAR(200)	String
LEADER_ID	teamLeader	NUMERIC(15)	Employee
PROJ_NAME	name	VARCHAR(30)	String
PROJ_TYPE	NA	CHAR(1)	NA
VERSION	NA	NUMERIC(15)	NA
LPROJECT	LargeProject		
PROJ_ID	NA	NUMERIC(15)	NA
BUDGET	budget	NUMERIC(10,2)	double
MILESTONE	milestoneVersion	TIMESTAMP	TimeStamp
ADDRESS	Address		
ADDRESS_ID	id	NUMERIC(15)	BigDecimal
COUNTRY	country	VARCHAR(80)	String
STREET	street	VARCHAR(80)	String
CITY	city	VARCHAR(80)	String
PROVINCE	province	VARCHAR(80)	String
P_CODE	postalCode	VARCHAR(20)	String
PHONE	PhoneNumber		
AREA_CODE	areaCode	CHAR(3)	String
P_NUMBER	number	CHAR(7)	String
EMP_ID	owner	NUMERIC(15)	Employee
TYPE	type	VARCHAR(15)	String
PROJ_EMP	*Relation Table*		
PROJ_ID	NA	NUMERIC(15)	NA
EMP_ID	NA	NUMERIC(15)	NA

Creating and Populating Database Tables

The TopLink_Tutorial.zip file contains a dbscripts.zip file that provides SQL scripts that you can use to create and populate the database tables used in the tutorial:

1. UnZIP the <TUTORIAL_HOME>\dbscripts.zip file.

This file contains two SQL scripts: `createTables.sql` and `populateTables.sql`.

2. Execute the `createTables.sql` script.

For example, using `sqlplus`:

```
C:>sqlplus scott/tiger
SQL> @createTables.sql
```

The tables required for this tutorial are created.

Note: For information on how to create database tables using TopLink Workbench, see "[Creating New Tables](#)" on page 4-22

3. Execute the `populateTables.sql` script.

For example, using `sqlplus`:

```
C:>sqlplus scott/tiger
SQL> @populateTables.sql
```

The tables are populated with the sample data required for this tutorial.

Building the Tutorial Project

In this chapter, you will create a TopLink Workbench project and map the tutorial Java classes to database tables. This section includes the following steps:

- [Step 1: Creating a Project](#)
- [Step 2: Adding a Data Source](#)
- [Step 3: Associating Descriptors to Tables](#)
- [Step 4: Mapping Attributes](#)
- [Step 5: Using Advanced Descriptor Properties](#)
- [Step 6: Verifying the Project](#)

Step 1: Creating a Project

In this step, you will create a relational TopLink project mapped to a relational database. In addition to relational projects, TopLink can create other kinds of nonrelational projects mapped to EIS data sources or XML Schema-based XML documents.

TopLink Workbench Project

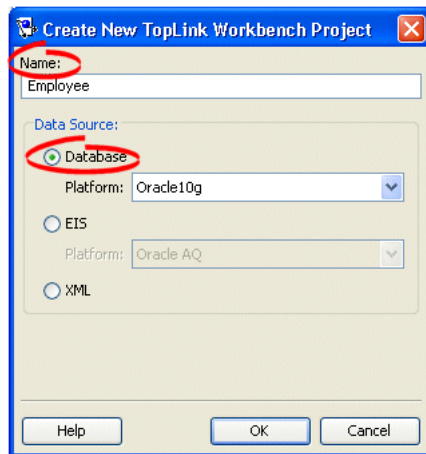
To create a new TopLink Workbench project, use this procedure:

1. Start TopLink Workbench by executing the `workbench` script in `<TOPLINK_HOME>\bin`. For windows, select the `workbench.cmd` file. For UNIX, select the `workbench.sh` file.
2. Click **New** on the toolbar and select **Project**. The Create New TopLink Workbench Project dialog box appears.



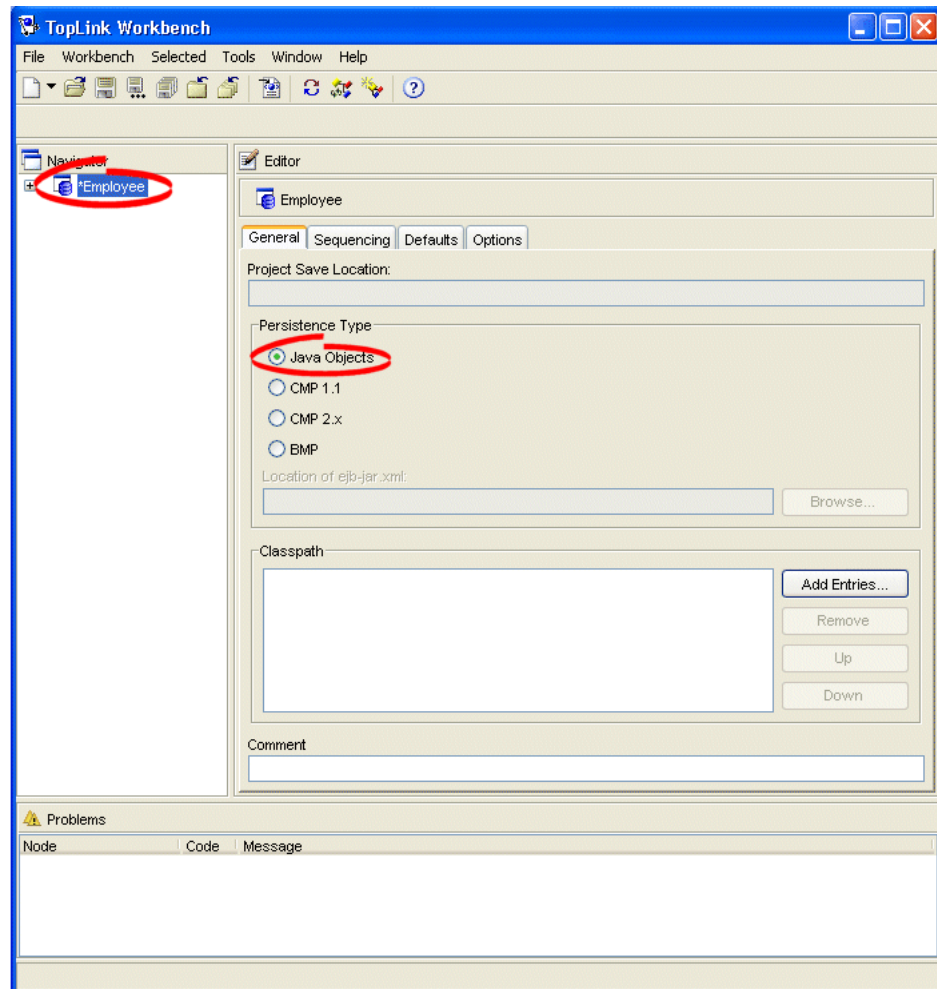
You can also create a new project by choosing **File > New > Project** from the menu.

Figure 17–1 Create New TopLink Workbench Project Dialog Box



3. In the **Name** field, type **Employee**.
4. Select **Database** as the **Data Source**, then use the **Platform** field to select your specific database platform.
5. Click **OK**. TopLink Workbench creates the new project.

Figure 17-2 TopLink Workbench



The tutorial project uses **Java Objects** as the persistence type. TopLink Workbench places an asterisk (*) beside the project name in the Navigator, indicating that you have not yet saved the project.



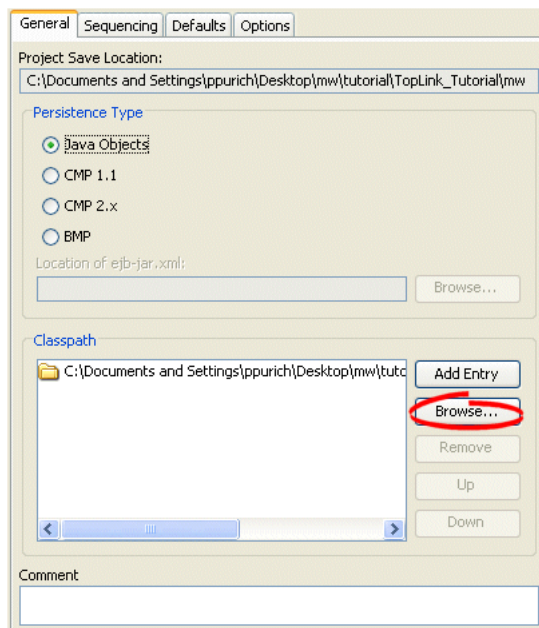
6. Click **Save** on the toolbar. TopLink Workbench prompts for a directory location.
7. Select the <TUTORIAL_HOME>\mw directory, and click **OK**.

When you create a new TopLink Workbench project, always save it in an empty directory.

Adding Java Classes

After creating the empty **Employee** project, you must add the compiled Java classes to the project's classpath.

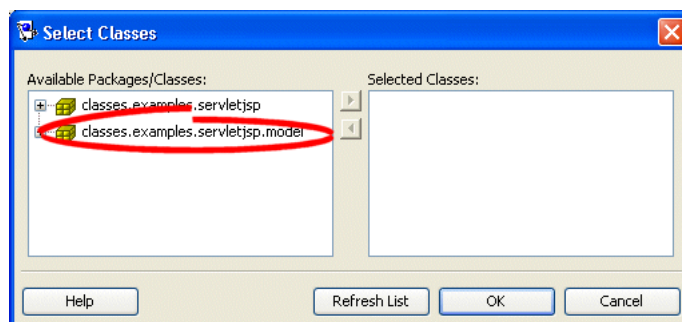
1. Select the **Employee** project in the **Navigator**. Its properties appear in the Editor.
2. Click the **General** tab. The **General** tab appears.

Figure 17-3 General Tab

3. In the **Classpath** area, click **Browse**. The Add Entries dialog box appears.
4. Select the <TUTORIAL_HOME>\servletjsp.jar that contains the compiled classes for this tutorial and click **OK**. TopLink Workbench adds the JAR file to the project's classpath.

For information on these classes, see "Source Files" on page 16-2.

5. Click **Add or Refresh Classes** to add the classes to the project. The Select Classes dialog box appears.

Figure 17-4 Select Classes Dialog Box

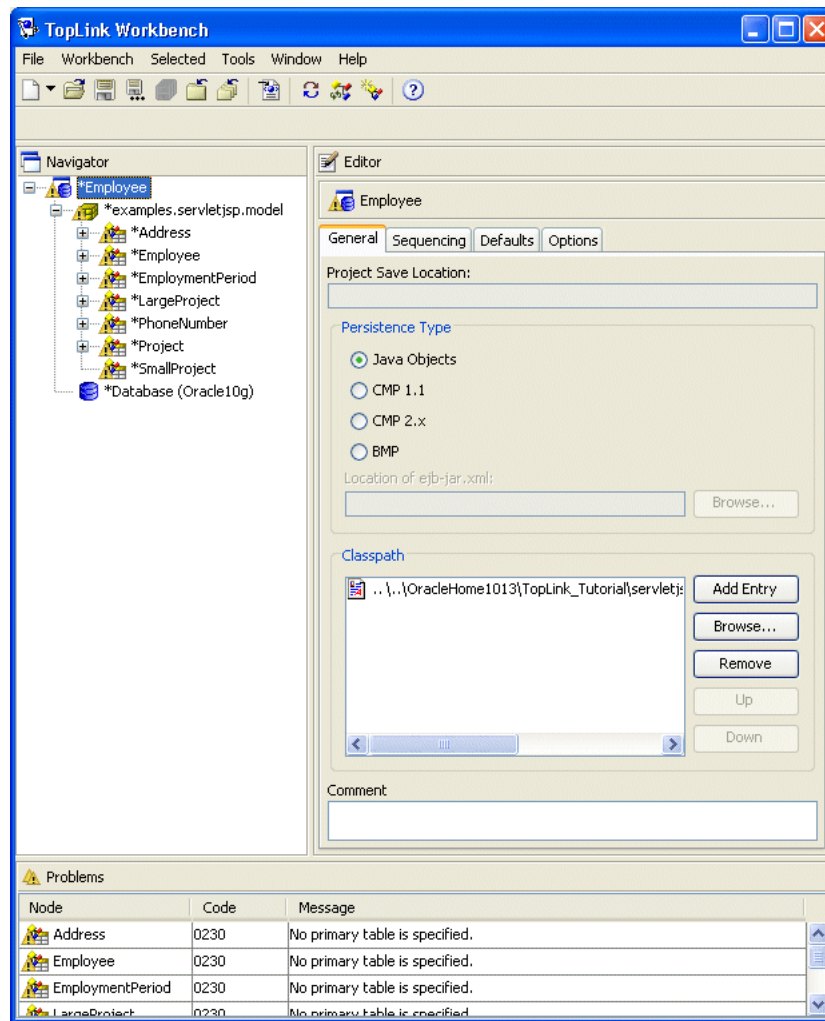
6. Select the `examples.servletjsp.model` package and click **Add Selected Classes To List**. The package moves to the **Selected Classes** area.

The other classes are used by the application at run time.

7. Click **OK**. TopLink Workbench adds the package to the project and creates TopLink descriptors for each Java class.

Descriptors are TopLink representations of Java classes. Descriptors own the mappings that relate persistent attributes to database fields.

Figure 17-5 TopLink Workbench



Notice that TopLink Workbench alerts you that there are errors and unmapped attributes in the project. You will correct these errors as you map the tutorial project.



8. Click **Save** on the toolbar to save the project.

Step 2: Adding a Data Source

Each TopLink project requires a data source. For the tutorial project, you will use a relational database. In this tutorial, you will create a development login on the project to allow TopLink Workbench to access the database during development. Later, you will create a deployment login that your application will use when you deploy it to the application server (see ["Creating a Database Deployment Login"](#) on page 18-6).

Creating a Database Development Login

To create a database login for TopLink Workbench to use during development, use this procedure:

1. Expand the **Employee** project icon in the **Navigator** and select the **Database** icon. Its properties appear in the Editor.

- Verify that the **Database Platform** is correct. Click **Change** to select a different platform, if necessary.
- In the **Defined Logins** area, click **Add**. The Add New Login dialog box appears.
 - Type the name for your login (for example, myLogin) and click **OK**. TopLink Workbench adds the newly created login to the Defined Logins area.

Figure 17–6 Database Property Sheet

The screenshot shows the 'Database Property Sheet' dialog box. It has a 'Database Platform' field set to 'Oracle10g' with a 'Change...' button. Below it is a 'Defined Logins' list box with buttons 'Add...', 'Rename...', and 'Remove'. Further down are fields for 'Driver Class', 'URL', 'User Name', and 'Password', each with a dropdown arrow. There is a 'Save Password' checkbox. At the bottom are 'Development Login' and 'Deployment Login' dropdown menus.

- Use the following information to complete the remaining login information:

Field	Description
Driver Class	Select the JDBC driver TopLink uses to communicate with your database (such as <code>oracle.jdbc.OracleDriver</code>).
URL	Specify the full location of your data source (such as <code>jdbc:oracle:thin:@myserver:port:sid</code>).
Username	Specify your username, required to log in to the database.
Password	Specify your password, required to log in to the database.
Save Password	Specify if TopLink if to save the password. When using JDK 1.4.1 (or later), the password is automatically encrypted with Java Cryptography Extension (JCE). For JDK versions prior to 1.4.1, refer to <i>Oracle TopLink Getting Started Guide</i> for information on using encrypted passwords.
Development Login	Select the login that you created (from the Defined Logins area).
Deployment Login	Ignore this field for now. You will specify a deployment login from the TopLink sessions configuration, later in this tutorial.



- Click **Login** on the tool bar to log in to the database.



Notice that the database icon in the **Navigators** changes to let you know that you are currently logged in.



- Click **Save** to save the project.

Importing Database Tables

To import the database tables (including their attributes and constraints) into the TopLink Workbench project, use this procedure:

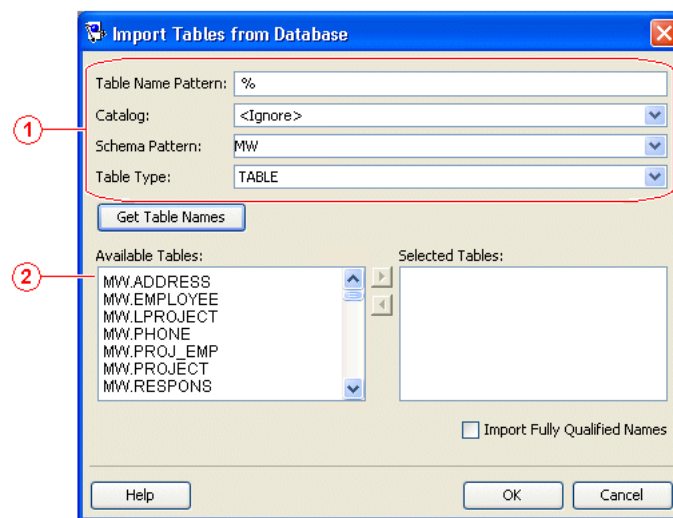
1. Select the logged-in **Database** icon in the **Navigator**. Its properties appear in the Editor.
2. Click **Add or Update Existing Tables from Database**. The Import Tables from Database dialog appears.



Figure 17-7 numbered callouts identify the following groups of user interface elements on this dialog box:

1. Filters: this area is made up of attributes Table Name Pattern, **Catalog**, **Schema Pattern**, and **Table Type**
2. Available and selected items: the tables that match the filter attributes are listed in the left hand panel and the tables that you have selected are listed in the right hand panel.

Figure 17-7 Import Tables from Database Dialog Box



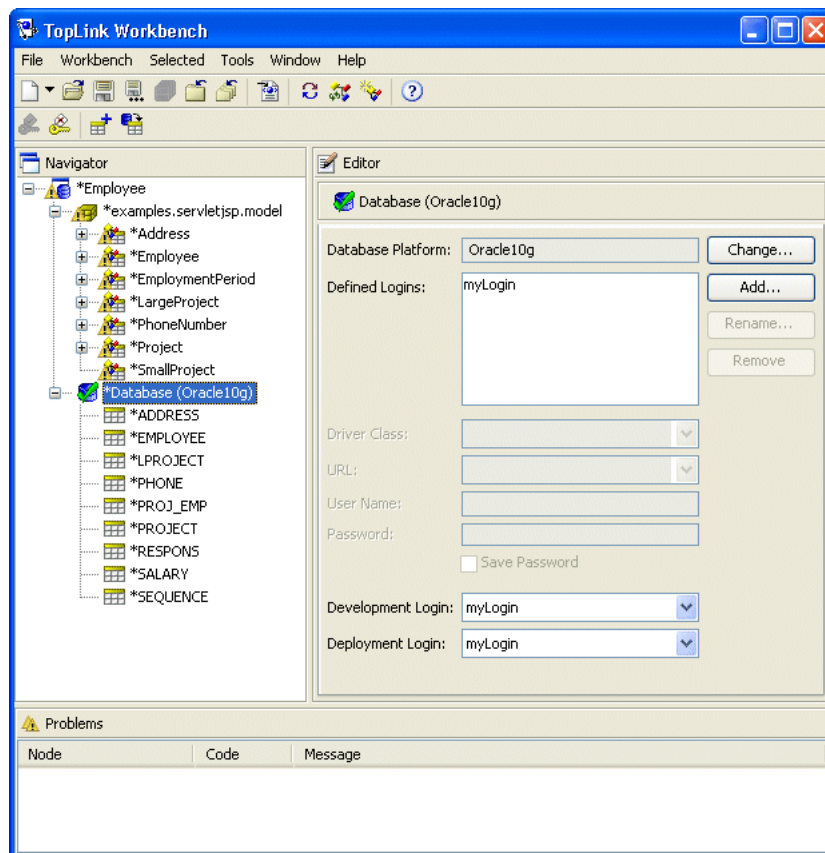
3. Use the following information to complete the filter information in the Import Tables from Database dialog box:

Field	Description
Table Name Pattern	Use the percent symbol (%) as wildcard search to display all the database tables.
Catalog	The database creation scripts do not specify a catalog. Consult your DBA for the correct information for your specific database and login.
Schema Pattern	The database creation scripts do not specify a schema. Consult your DBA for the correct information for your specific database and login.
Table Type	Select TABLE as the database table type. TopLink also supports other database table types such as views and synonyms.

4. Click **Get Table Names**. TopLink Workbench displays the database tables in the **Available Tables** area.
5. Select the following tables (see "[Data Model](#)" on page 16-4) and click **Add Selected Items**.

- ADDRESS
 - EMPLOYEE
 - LPROJECT
 - PHONE
 - PROJ_EMP
 - PROJECT
 - RESPON
 - SALARY
 - SEQUENCE
6. Click **OK**. TopLink Workbench adds the tables to the project.

Figure 17–8 TopLink Workbench



7. Click **Save** to save the project.

Project-Level Sequencing

Use this procedure to apply a sequence table to the project. The tutorial project uses custom sequencing for the **Employee** and **Address** classes. Later you will assign specific sequencing information for both classes.

1. Select the **Employee** project in the **Navigator**. Its properties appear in the Editor.
2. Click the **Sequencing** tab. The Sequencing tab appears.

Figure 17–9 Sequencing Tab

3. Use the following information to complete each field on the Sequencing tab:

Field	Description
Custom Sequence Table	Specify that the project uses a custom sequence table.
Name	Select the SEQUENCE database table.
Sequence Name Field	Select the SEQ_NAME database field.
Sequence Counter Field	Select the SEQ_COUNT database field.



4. Click **Save** to save the project.

Later, you will specify descriptor-level sequencing to track employees and their addresses.

Step 3: Associating Descriptors to Tables

To associate a descriptor with a database table, use this procedure:

1. Expand the **Employee** project icon and the `examples.servletjsp.modelpackage` in the **Navigator**.
2. Select the **Address** descriptor in the **Navigator**. Its properties appear in the Editor.
3. Select the **Descriptor Info** tab. The Descriptor Info tab appears.
4. In the **Associated Table** field, select the ADDRESS table. Notice that TopLink Workbench also correctly identifies the primary key for the table (if supported by your database driver).

If your database driver does not allow TopLink Workbench to identify the primary key, click **Add** to add the primary key, as indicated in the tables in "Data Model" on page 16-4.

Figure 17-10 Descriptor Info Tab

The screenshot shows the 'Descriptor Info' tab for a descriptor. The 'Associated Table' is set to 'ADDRESS'. Under the 'Primary Keys' section, 'ADDRESS.ADDRESS_ID' is listed. There are buttons for 'Add...' and 'Remove'. Below this, there are checkboxes for 'Use Sequencing', 'Read-Only', and 'Conform Results in Unit of Work'. There are also input fields for 'Name', 'Table', and 'Field' under 'Use Sequencing', and a 'Descriptor Alias' field set to 'Address'.

5. Use the following information to associate the remaining descriptors with database tables:

Associate This Descriptor...	With This Database Table...
Employee	EMPLOYEE
LargeProject	LPROJECT
PhoneNumber	PHONE
	Notice that the PHONE table has a compound primary key: EMP_ID and TYPE.
Project	PROJECT
SmallProject	PROJECT



6. Click **Save** to save the project.

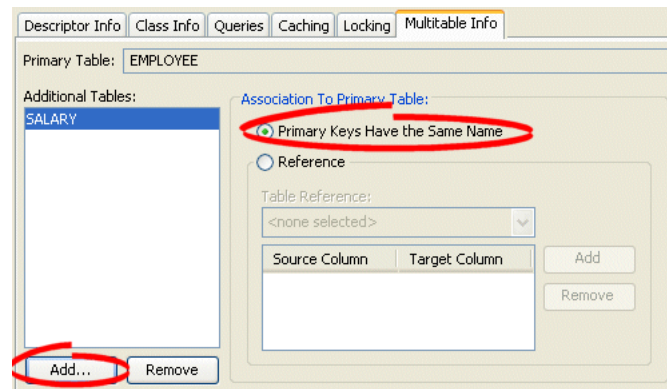
Notice that the `EmploymentPeriod` descriptor is not associated with any database tables. This class will acquire its characteristics through *aggregation*, later in this tutorial.

Associating With Multiple Tables

In this project, the `Employee` class is stored in multiple tables; although most information is in the `EMPLOYEE` table, salary information is stored in the `SALARY` table.

To spread classes across two or more tables, use this procedure:

1. Right-click the **Employee** descriptor in the **Navigator** and select **Select Advanced Properties** from the context menu. The Select Advanced Properties dialog box appears.
2. Select **Multitable Info** and click **OK**. TopLink Workbench adds the Multitable Info tab to the Editor.
3. Click the **Multitable Info** tab. The Multitable Info tab appears. The Primary Table (`EMPLOYEE`) has already been selected.

Figure 17-11 Multitable Info Tab

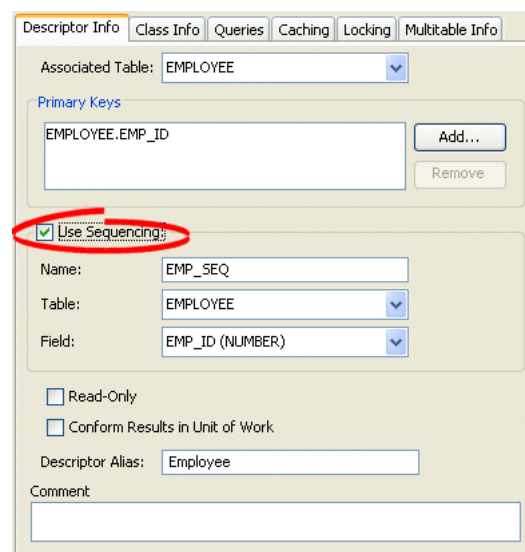
4. Click **Add** to add an additional table. The Select Associated Table dialog box appears.
5. Select the **SALARY** table, and click **OK**.
6. On the Multitable Info tab, select the **Primary Keys Have the Same Name** option.
7. Click **Save** to save the project.



Descriptor-Level Sequencing

To specify custom (nonnative) sequencing for the **Employee** class, use this procedure:

1. Select the **Employee** descriptor in the **Navigator**. Its properties appear in the Editor.
2. Click the **Descriptor Info** tab. The Descriptor Info tab appears.

Figure 17-12 Descriptor Info Tab, Sequencing Options

3. Use the following information to complete the **Sequencing** options on the Descriptor Info tab:

Field	Description
Use Sequencing	Specify that the Employee descriptor uses custom sequencing.
Name	Type EMP_SEQ as the name of this sequence.
Table	Select the EMPLOYEE database table.
Field	Select the EMP_ID database field.



- Click **Save** to save the project.

Repeat these steps to use descriptor-level sequencing for the **Address** descriptor. Use the following information to complete the **Sequencing** information for the **Address** descriptor:

Field	Description
Use Sequencing	Specify that the Address descriptor uses custom sequencing.
Name	Type ADDRESS_SEQ as the name of this sequence.
Table	Select the ADDRESS database table.
Field	Select the ADDRESS_ID database field.

Step 4: Mapping Attributes

In this step, you will create several direct and relationship mappings from the Java objects to the database tables. Review [Table 16–10](#) on page 16-6 for information on these mappings.

Direct-to-Field Mappings

In the tutorial model, each attribute of the **Address** descriptor (such as **city**) maps directly to a column in a database table. TopLink represents this relationship as a *direct-to-field* mapping.

Use this procedure to create the direct-to-field mapping of the **city** attribute of the **Address** class:

- Expand the **Address** descriptor in the **Navigator** to display its attributes.
- Select the **city** attribute and click **Direct to Field Mapping**. TopLink Workbench creates the direct-to-field mapping. Its properties appear in the Editor.



You can also create the mapping by right-clicking the **city** attribute and selecting **Map As > Direct to Field** from the context menu.

Notice that TopLink Workbench has added a warning icon to the **city** attribute, indicating that the mapping is incomplete.

Figure 17-13 General Tab of Direct-to-Field Mapping

The screenshot shows the 'General' tab of a 'Direct-to-Field Mapping' dialog. The 'Database Field' dropdown menu is highlighted with a red circle and shows 'ADDRESS.CITY (VARCHAR2)'. Below this, there are several sections: 'Method Accessing' with 'Get Method' and 'Set Method' dropdowns; 'Default Null Value' with 'Type' and 'Value' dropdowns; 'Read-Only' checkbox; and a 'Comment' text area.

3. On the General tab, use the **Database Field** field to select the ADDRESS.CITY database field.

Using Automap

Instead of mapping Java attributes manually, TopLink Workbench can attempt to automatically map attributes and database fields with similar names.

To automatically map each attribute of the Address class as a direct-to-field mapping, use this procedure:

1. Expand the **Address** descriptor in the **Navigator** to display its attributes.
2. Right-click the **Address** descriptor and select **Automap** from the context menu. TopLink Workbench automatically maps the unmapped descriptors and displays the Automap Status dialog box.
3. Click **OK**. TopLink Workbench mapped each attribute as a direct to field mapping.
4. Individually, select each attribute of the **Address** descriptor and confirm they are mapped to the following database fields:

Attribute	Database Field
country	COUNTRY
id	ADDRESS_ID
postalCode	P_CODE
province	PROVINCE
street	STREET

Note: It is not necessary to confirm the **city** attribute; the Automap function will not change attributes that are already mapped.



5. Click **Save** to save the project.

The Automap function relies on a logically named and ordered database structure. You should always review mappings created by the Automap function.

Additional Direct-to-Field Mappings

Follow the procedure from "[Direct-to-Field Mappings](#)" on page 17-12 to map the following attributes as direct-to-field mappings:

Descriptor	Attribute	Database Field
Employee	firstName	F_NAME
	id	EMP_ID (from EMPLOYEE table)
	lastName	L_NAME
	salary	SALARY (from SALARY table)
LargeProject	budget	BUDGET
	milestoneVersion	MILESTONE
PhoneNumber	areaCode	AREA_CODE
	number	P_NUMBER
	type	TYPE
Project	description	DESCRIP
	id	PROJ_ID
	name	PROJ_NAME

Leave all other attributes unmapped.

One-to-One Mappings

In the tutorial model, each employee has a single address. TopLink represents this relationship as a *one-to-one* mapping.

To create a one-to-one mapping of the **address** attribute of the **Employee** class, use this procedure:

1. Expand the **Employee** descriptor in the **Navigator** to display its attributes.
2. Select the **address** attribute and click **One to One Mapping**. TopLink Workbench creates the one-to-one mapping. Its properties appear in the Editor.



You can also create the mapping by right-clicking the **address** attribute and selecting **Map As > One to One** from the context menu.

Notice that TopLink Workbench has added a warning icon to the **address** attribute, indicating that the mapping is incomplete.

Figure 17-14 General Tab of One-to-One Mapping

The screenshot shows the 'General' tab of the TopLink configuration tool. The 'Reference Descriptor' dropdown is set to 'Address (examples.serviet,...)'. Below it, the 'Method Accessing' section has 'Get Method' and 'Set Method' dropdowns set to '<none selected>'. The 'Read-Only' checkbox is unchecked. The 'Private Owned' checkbox is checked. The 'Batch Reading' checkbox is unchecked. The 'Use Joining' checkbox is unchecked. The 'Use Indirection' checkbox is checked. The 'ValueHolder' radio button is selected, and the 'Proxy' radio button is unselected. The 'Maintains Bidirectional Relationship' checkbox is unchecked. The 'Relationship Partner' dropdown is set to '<none selected>'. There is a 'Comment' text area at the bottom.

3. Click the **General** tab. Use the following information to complete the fields on the General tab:

Field	Description
Reference Descriptor	Select Address . The address attribute (of the Employee descriptor) references the Address descriptor.
Private Owned	Select this option to automatically create, update, or delete the Address object, whenever its owner (Employee) is changed.
Use Indirection	Select ValueHolder to allow TopLink to use a value holder when retrieving and storing information from the database. This reduces database accesses and improves performance. See " Configuring Indirection " on page 35-3 for more information.

Leave the other fields empty.

4. Click the **Table Reference** tab. The Table Reference tab appears.

Figure 17-15 Table Reference Tab of One-to-One Mapping

The screenshot shows the 'Table Reference' tab of the TopLink configuration tool. The 'Table Reference' dropdown is set to 'EMPLOYEE_ADDRESS (EMPLOYEE => ADDRESS)'. Below it, there is a table with three columns: 'Source Field', 'Target Field', and 'Target Foreign Key'. The first row has 'ADDR_ID' in the 'Source Field' column, 'ADDRESS_ID' in the 'Target Field' column, and an unchecked checkbox in the 'Target Foreign Key' column. There are 'New...', 'Add', and 'Remove' buttons to the right of the table.

5. In the **Table Reference** field, select **EMPLOYEE_ADDRESS**. TopLink assigns the **Source** (ADDR_ID) and **Target** (ADDRESS_ID) fields, based on the database

foreign key constraints imported when you imported the database tables (see ["Importing Database Tables"](#) on page 17-6).

Note: For more information on creating constraints using TopLink Workbench, see ["Configuring Table and Field References \(Foreign and Target Foreign Keys\)"](#) on page 37-8.



6. Click **Save** to save the project.

Additional One-to-One Mappings

Follow the procedure from ["One-to-One Mappings"](#) on page 17-12 to map the following attributes as one-to-one mappings:

Descriptor	Attribute	Reference Descriptor	Table Reference
Employee	manager	EMPLOYEE	EMPLOYEE_EMPLOYEE
Project	teamLeader	EMPLOYEE	PROJECT_EMPLOYEE

Leave all other attributes unmapped.

One-to-Many Mappings

In the tutorial model, each employee may have multiple telephone numbers. TopLink represents this relationship as a *one-to-many* mapping.

To create a one-to-many mapping of the attributes of the `Employee` class, use this procedure:

1. Expand the **Employee** descriptor in the **Navigator** to display its attributes.
2. Select the **phoneNumbers** attribute and click **One to Many Mapping**. TopLink Workbench creates the one-to-many mapping. Its properties appear in the Editor.

You can also create the mapping by right-clicking the **phoneNumbers** attribute and selecting **Map As > One to Many** from the context menu.

Notice that TopLink Workbench has added a warning icon to the **phoneNumbers** attribute, indicating that the mapping is incomplete.

Figure 17-16 General Tab of One-to-Many Mapping

The screenshot shows the 'General' tab of the configuration tool. The 'Reference Descriptor' dropdown is set to 'PhoneNumber (examples.servlet...)'. Below it, the 'Method Accessing' section has 'Get Method' and 'Set Method' both set to '<none selected>'. The 'Read-Only' checkbox is unchecked. The 'Private Owned' checkbox is checked. The 'Batch Reading' checkbox is unchecked. The 'Use Indirection' checkbox is checked. The 'ValueHolder' radio button is selected, and the 'Transparent' radio button is unselected. The 'Maintains Bidirectional Relationship' checkbox is unchecked. The 'Relationship Partner' dropdown is set to '<none selected>'. There is an 'Advanced Container Options >>' button and a 'Comment' text area.

3. Click the **General** tab. Use the following information to complete the fields on the General tab:

Field	Description
Reference Descriptor	Select PhoneNumber . The phoneNumbers attribute (of the Employee descriptor) references the PhoneNumber descriptor.
Private Owned	Select this option to automatically create, update, or delete the PhoneNumber object, whenever its owner (Employee) is changed.
Use Indirection	Select ValueHolder to allow TopLink to use a value holder when retrieving and storing information from the database. This reduces database accesses and improves performance. See "Configuring Indirection" on page 35-3 for more information.

Leave the other fields empty.

4. Click the **Table Reference** tab. The Table Reference tab appears.

Figure 17-17 Table Reference Tab of One-to-Many Mapping

The screenshot shows the 'Table Reference' tab. The 'Table Reference' dropdown is set to 'PHONE_EMPLOYEE (PHONE => EMPLOYEE)'. Below it, there is a table with two columns: 'Source Column' and 'Target Column'. The 'Source Column' is 'EMP_ID' and the 'Target Column' is 'EMP_ID'. There are 'New...', 'Add', and 'Remove' buttons to the right of the table.

5. In the **Table Reference** field, select **PHONE_EMPLOYEE**. TopLink assigns the **Source** (**EMP_ID**) and **Target** (**EMP_ID**) fields, based on the database foreign key constraints.

- To complete this one-to-many mapping, you must create a one-to-one mapping *back from the referenced class*. Follow the procedure from ["One-to-One Mappings"](#) on page 17-14 to create a one-to-one mapping from the **PhoneNumber** descriptor's **owner** attribute to the **Employee** descriptor:

Descriptor	Attribute	Reference Descriptor	Table Reference
PhoneNumber	owner	Employee	PHONE_EMPLOYEE



- Click **Save** to save the project.

Additional One-to-Many Mappings

Follow the procedure from ["One-to-Many Mappings"](#) on page 17-12 to map the following attributes as one-to-many mappings:

Descriptor	Attribute	Reference Descriptor	Table Reference
Employee	managedEmployees	EMPLOYEE	EMPLOYEE_EMPLOYEE

This is an example of a one-to-many *self-reference*. The tutorial object model requires the **Employee** class to reference another instance of the same class. You have already created the one-to-one back-mapping from the **Employee** descriptor's **manager** attribute back to the **Employee** descriptor.

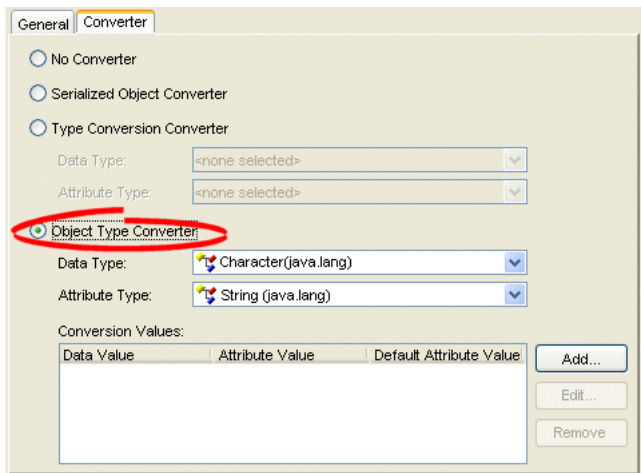
Leave all other attributes unmapped.

Object Type Mappings

In the tutorial model, each employee's gender is stored as a single letter in the database (**M** or **F**), but the object value is the full word (**Male** or **Female**). TopLink represents this relationship as an *object type* mapping.

- Expand the **Employee** descriptor in the **Navigator** to display its attributes.
- Right-click the **gender** attribute and select **Map As > Object Type** from the context menu. TopLink Workbench creates the object type mapping. Its properties appear in the Editor.
- Click the **General** tab. The General tab appears.
- On the General tab, use the **Database Field** field to select the **EMPLOYEE.GENDER** database field.
- Click the **Converter** tab. The Converter tab appears. Ensure the **Object Type Converter** is selected.

Figure 17-18 Converter Tab, Object Type Converter

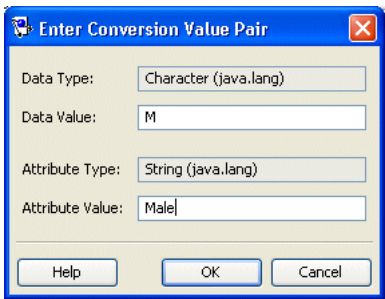


6. Use the following information to complete the fields on the Converter tab:

Field	Description
Data Type	Select Character . In the database, the GENDER column type is a single character.
Attribute Type	Select String . In the object, the gender attribute is a text string.

7. In **Conversion Values** area of the Converter tab, click **Add**. The Enter Conversion Value Pair dialog box appears.

Figure 17-19 Enter Conversion Value Pair Dialog Box



8. In the **Data Value** field, type M. In the **Attribute Value** field, type Male. Click **OK**. Repeat steps 7 and 8 to create a conversion from F to Female.



9. Click **Save** to save the project.

Aggregate Object Mappings

In TopLink, two objects—a source (parent or owning) object and a target (child or owned) object—are related by aggregation if there is a strict one-to-one relationship between them, and all the attributes of the target object can be retrieved from the same data source representation as the source object. This means that if the source object exists, then the target object must also exist, and if the source object is destroyed, then the target object is also destroyed.

In the tutorial object model, the **Employee** descriptor's **period** attribute uses an aggregate mapping to the **EmploymentPeriod** descriptor (which is associated with the **START** and **END** date fields of the **EMPLOYEE** database table).

Aggregate Descriptor

Before creating the mapping, you must create the aggregate descriptor by doing the following:



1. Select the **EmploymentPeriod** descriptor in the **Navigator**.

2. Click **Aggregate Descriptor** in the toolbar. TopLink Workbench changes the descriptor to an aggregate descriptor.

Notice that the **Descriptor Info** and **Queries** tabs have been removed in the Editor.

3. Expand the **EmploymentPeriod** descriptor and map each attribute (**StartDate** and **EndDate**) as direct-to-field mappings.

Notice that the **Database Field** field is not available on each mapping's **General** tab.

Next, you create the aggregate mappings.

Aggregate Mapping

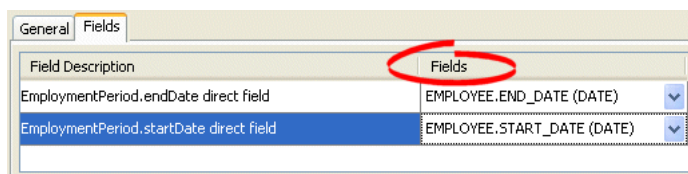
Now you will create an aggregate mapping for the **period** attribute, using the following procedure:

1. Expand the **Employee** descriptor in the **Navigator** to display its attributes.
2. Right-click the **period** attribute and select **Map As > Aggregate** from the context menu. TopLink Workbench creates the aggregate mapping. Its properties appear in the Editor.
3. On the **General** tab, use the **Reference Descriptor** field to select the **EmploymentPeriod** descriptor.

Notice that the field lists *only* the aggregate descriptor.

4. Click the **Fields** tab. The **Fields** tab appears.

Figure 17-20 Fields Tab of Aggregate Mapping



The **Field Description** column is populated automatically, based on the mapped attributes of the **EmploymentPeriod** aggregate descriptor.

5. Use the following information to select the database for each attribute:

Fields	Database Field
EmploymentPeriod.endDate direct field	END_DATE
EmploymentPeriod.startDate direct field	START_DATE



6. Click **Save** to save the project.

Direct Collection Mappings

In the tutorial model, each employee has a list of responsibilities. TopLink represents this collection of objects relationship as a *direct collection* mapping.

1. Expand the **Employee** descriptor in the **Navigator** to display its attributes.
2. Right-click the **responsibilitiesList** attribute and select **Map As > Direct Collection** from the context menu. TopLink Workbench creates the mapping. Its properties appear in the Editor.
3. Click the **General** tab. The General tab appears.

Figure 17-21 General Tab of Direct Collection Mapping

The screenshot shows the 'General' tab of the 'Direct Collection Mapping' dialog. The 'Target Table' dropdown is set to 'RESPONS'. The 'Direct Value Field' dropdown is set to 'RESPONS.DESCRIP (VARCHAR2)'. Below these, there are several checkboxes: 'Method Accessing' (unchecked), 'Read-Only' (unchecked), 'Batch Reading' (unchecked), and 'Use Indirection' (checked). Under 'Use Indirection', 'ValueHolder' is selected with a radio button, and 'Transparent' is unselected. At the bottom, there is an 'Advanced...' button and a 'Comment' text area.

4. Use the following information to complete the fields on this tab:

Field	Description
Target Table	Select the RESPONS table as the table in which to store the string value.
Direct Value Field	Select the RESPONS.DESCRIP table.
Use Indirection	Select ValueHolder indirection to allow TopLink to use a value holder when retrieving and storing information from the database. This reduces database accesses and improves performance. See " Configuring Indirection " on page 35-3 for more information.

5. Click the **Table Reference** tab. The Table Reference tab appears.
6. In the **Table Reference** field, select RESPONS_EMPLOYEE. TopLink assigns the **Source** (EMP_ID) and **Target** (EMP_ID) fields, based on the database foreign key constraints.
7. Click **Save** to save the project.



Many-to-Many Mappings

In the tutorial model, many employees may have many projects. TopLink represents this relationship between collections as a *many-to-many* mapping.

To create a many-to-many mapping, use the following procedure:

1. Expand the **Employee** descriptor in the **Navigator** to display its attributes.
2. Right-click the **projects** attribute and select **Map As > Many to Many** from the context menu. TopLink Workbench creates the mapping. Its properties appear in the Editor.
3. Click the **General** tab. The General tab appears.

Figure 17-22 General Tab of Many-to-Many Mapping

The screenshot shows the 'General' tab of a mapping editor. At the top, there are three tabs: 'General', 'Table Reference', and 'Ordering'. The 'General' tab is active. Below the tabs, there is a 'Reference Descriptor' dropdown menu with a red circle around it, showing 'Project (examples.servlet...)'. Below this, there is a 'Method Accessing' section with 'Get Method' and 'Set Method' dropdowns, both set to '<none selected>'. Below that, there are several checkboxes: 'Read-Only', 'Private Owned', 'Batch Reading', and 'Use Indirection' (which is checked and circled in red). Below these checkboxes, there are two radio buttons: 'ValueHolder' (selected) and 'Transparent'. Below the radio buttons, there is a 'Maintains Bidirectional Relationship' checkbox, which is unchecked. Below this, there is a 'Relationship Partner' dropdown menu, also set to '<none selected>'. At the bottom, there is an 'Advanced...' button and a 'Comment' text area.

4. Use the following information to complete the fields on this tab:

Field	Description
Reference Descriptor	Select the Project descriptor.
Use Indirection	Select ValueHolder indirection to allow TopLink to use a value holder when retrieving and storing information from the database. This reduces database accesses and improves performance. See " Configuring Indirection " on page 35-3 for more information.

5. Click the **Table Reference** tab. The Table Reference tab appears.

Figure 17-23 Table Reference Tab of Many-to-Many Mapping

The screenshot shows the 'Table Reference' tab of a mapping tool. It has three sections: 'General', 'Table Reference', and 'Ordering'. The 'Table Reference' section is active. It contains three main areas: 'Relation Table', 'Source Reference', and 'Target Reference'. The 'Relation Table' dropdown is set to 'PROJ_EMP'. The 'Source Reference' section has a 'Table Reference' dropdown set to 'PROJEMP_EMP (PROJ_EMP => EMPLOYEE)', with 'EMP_ID' mapped to 'EMP_ID'. The 'Target Reference' section has a 'Table Reference' dropdown set to 'PROJEMP_PROJ (PROJ_EMP => PROJECT)', with 'PROJ_ID' mapped to 'PROJ_ID'. Red circles highlight the 'Relation Table', 'Table Reference' dropdowns, and the 'Table Reference' label.

6. Use the following information to complete the fields on this tab:

Field	Description
Relation Table	Select the PROJ_EMP table.
Source Reference	
Table Reference	Select the PROJEMP_EMP table. Select the EMP_ID field (from the PROJ_EMP table) as the Source Field, and the EMP_ID field (from the EMPLOYEE table) as the Target Field.
Target Reference	
Table Reference	Select the PROJEMP_PROJ table. Select the PROJ_ID field (from the PROJ_EMP table) as the Source Field, and the PROJ_ID field (from the PROJECT table) as the Target Field.



7. Click **Save** to save the project.

Transformation Mappings

Typically, a TopLink mapping maps a single Java attribute to a single database field.

In the tutorial model, the Employee descriptor has a single attribute (normalHours) that extracts the values from two database columns (START_TIME and END_TIME).

You can use a TopLink transformation mapping for these situations when an existing TopLink mapping cannot address your application requirements.

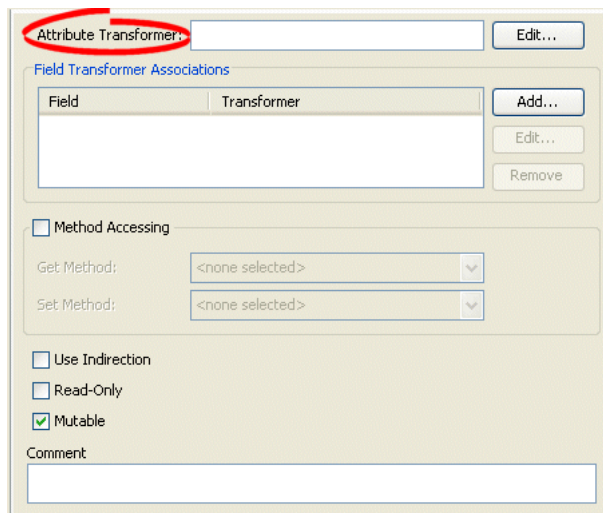
A transformation mapping is made up of an attribute transformer for field-to-attribute transformation at read (unmarshall) time (see "Configuring Attribute Transformer" on page 35-29) and one or more field transformers for attribute-to-field transformation at write (marshall) time (see "Configuring Field Transformer Associations" on page 35-31).

To create a transformation mapping, use the following procedure:

1. Expand the **Employee** descriptor in the **Navigator** to display its attributes.
2. Right-click the **normalHours** attribute and select **Map As > Transformation** from the context menu. TopLink Workbench creates the transformation mapping. Its properties appear in the Editor.

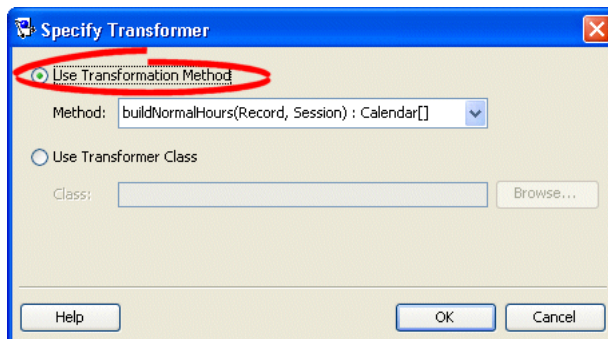
Notice that TopLink Workbench has added a warning icon to the **transformation** attribute, indicating that the mapping is incomplete.

Figure 17-24 Transformation Mapping



3. In the **Attribute Transformer** field, click **Edit**. The Specify Transformer dialog box appears.

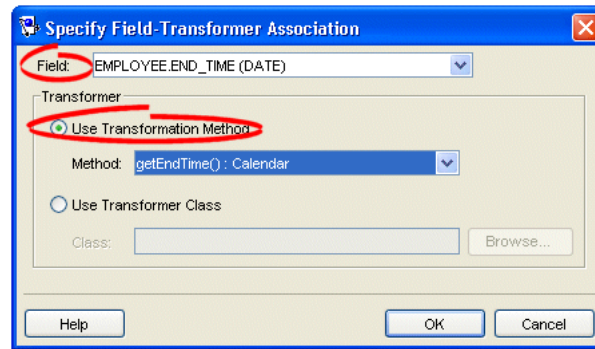
Figure 17-25 The Specify Transformer Dialog Box



4. Select **Use Transformation Method** and select the **buildNormalHours** method. Click **OK**.

You can specify the attribute transformer as a method of this domain object class (as done here) or as a separate class.

5. In the **Field Transformer Associations** area, click **Add**. The Specify Field-Transformer Association dialog box appears.

Figure 17–26 The Specify Field-Transformer Association Dialog Box

6. Use the following information to complete the fields on the Specify Field-Transformer Association dialog box and click **OK**:

Field	Description
Field	Select the END_TIME field from the EMPLOYEE table.
Transformer	
Use Transformation Method	Select the getEndTime() method.

7. Click **OK**.
8. Repeat steps 5 and 6 to add the START_TIME transformation method.
9. Click **Save** to save the project.



Step 5: Using Advanced Descriptor Properties

In the Employee tutorial project, you will explore several of the TopLink advanced descriptor properties. This step includes exercises on using:

- [Inheritance](#)
- [Locking Policy](#)
- [Named Queries](#)

Inheritance

In reviewing the tutorial object model (see [Figure 16–1](#) on page 16-3), notice that the **Project** class is extended by two subclasses: **SmallProject** and **LargeProject**. These subclasses inherit characteristics from the parent class.

To specify inheritance in TopLink Workbench, first configure inheritance attributes of the parent class and then configure the inheritance attributes of each of its derived child class.

Root Parent Class

Use this procedure to specify that **Project** is the *root parent* descriptor in the hierarchy.

1. Right-click the **Project** descriptor in the **Navigators** and select **Select Advanced Properties** from the context menu. The Select Advanced Properties dialog box appears.

2. Select **Inheritance** and click **OK**. TopLink Workbench adds the Inheritance tab.
3. Click the **Inheritance** tab. The Inheritance properties appear.

Figure 17-27 Inheritance Tab, Parent Class

Descriptor Info Class Info Queries Caching Locking **Inheritance**

☒ Read Subclasses on Query:
Read Subclasses View (Optional): <none selected>

☐ Is Child Descriptor
Parent Descriptor: <none selected>

☒ Is Root Parent Descriptor:
☐ Use Class Extraction Method:
<none selected>

☒ Use Class Indicator Field:
Field Selection:
PROJECT.PROJ_TYPE (VARCHAR2)

Indicator Selection:
☐ Use Class Name as Indicator
☒ Use Class Indicator Dictionary:
Indicator Type: String (java.lang)

Include	Class	Indicator Value
☐	Project (examples.servletjsp.model)	

Edit

4. Use the following information to complete each field on this tab:

Field	Description
Read Subclasses on Query	Select to allow TopLink to access the subclasses (LargeProject and SmallProject) in queries.
Is Root Parent Descriptor	Select to specify the Project descriptor as the root of the inheritance.
Use Class Indicator Field	Select PROJ_TYPE . This field contains the indicator value for each subclass.
Use Class Indicator Dictionary	Select Use Class Indicator Dictionary and select String as the Indicator Type . The class indicator field in the database table (PROJ_TYPE) is a string value.

Leave the other fields blank.

Child Class

Now you will identify each child class in the inheritance hierarchy (LargeProject and SmallProject subclasses) using this procedure:

1. Right-click the **LargeProject** descriptor in **Navigator** and select **Select Advanced Properties** from the context menu. The Select Advanced Properties dialog box appears.

2. Select **Inheritance** and click **OK**. TopLink Workbench adds the Inheritance tab.
3. Click the **Inheritance** tab. The Inheritance properties appear.

Figure 17-28 Inheritance Tab, Child Class

The screenshot shows the 'Inheritance' tab of a configuration window. At the top, there are tabs for 'Descriptor Info', 'Class Info', 'Queries', 'Caching', 'Locking', and 'Inheritance'. The 'Inheritance' tab is active. Below the tabs, there are several sections:

- 'Read Subclasses on Query:' with a checkbox that is unchecked.
- 'Read Subclasses View (Optional):' with a dropdown menu showing '<none selected>'.
- 'Is Child Descriptor:' with a radio button that is selected and circled in red.
- 'Parent Descriptor:' with a dropdown menu showing 'Project (examples.servletjsp.model)'.
- 'Is Root Parent Descriptor:' with a radio button that is unselected.
- 'Use Class Extraction Method:' with a dropdown menu showing '<none selected>'.
- 'Use Class Indicator Field:' with a radio button that is unselected.
- 'Field Selection:' with a dropdown menu showing '<none selected>'.
- 'Indicator Selection:' with two radio buttons: 'Use Class Name as Indicator' (unselected) and 'Use Class Indicator Dictionary:' (unselected).
- 'Indicator Type:' with a dropdown menu showing '<none selected>'.
- A table with columns 'Include', 'Class', and 'Indicator Value', and an 'Edit' button.

4. Use the following information to complete each field on this tab:

Field	Description
Is Child Descriptor	Specify that the LargeProject descriptor is a child descriptor and not the parent.
Parent Descriptor	Select the Project descriptor.

Leave the other fields blank.

5. Repeat this procedure for the **SmallProject** descriptor.

Indicator Values

To complete the inheritance, you must select which children to include in the inheritance hierarchy, using this procedure:

1. Select the **Project** descriptor in the **Navigator**. Its properties appear in the Editor.
2. Click the **Inheritance** tab. The Inheritance properties appear.

Figure 17–29 Inheritance Tab, Indicator Values

Descriptor Info Class Info Queries Caching Locking **Inheritance**

☒ Read Subclasses on Query:
Read Subclasses View (Optional): <none selected>

☐ Is Child Descriptor
Parent Descriptor: <none selected>

☒ Is Root Parent Descriptor:

☐ Use Class Extraction Method:
<none selected>

☒ Use Class Indicator Field:

Field Selection:
PROJECT.PROJ_TYPE (VARCHAR2)

Indicator Selection:

☐ Use Class Name as Indicator

☒ Use Class Indicator Dictionary:

Indicator Type: String (java.lang)

Include	Class	Indicator Value	Edit
☐	Project (examples.servletjsp.model)		
☒	LargeProject (examples.servletjsp.model)	L	
☒	SmallProject (examples.servletjsp.model)	S	

3. Select to **Include** the **LargeProject** descriptor and click **Edit**. The Enter an Indicator Value dialog box appears.
4. In the **Indicator Value** field, type **L** and click **OK**.
5. Repeat steps 3–4 to include the **SmallProject** descriptor. Use **S** as the indicator value.
6. Uncheck the **Include** column for the **Project** descriptor: because it is an abstract class, it does not require an indicator value.
7. Click **Save** to save the project.



Locking Policy

In this tutorial project, each employee's project information (the **Project** class) uses optimistic locking based on version locking. To specify this locking policy, use this procedure:

1. Select the **Project** descriptor in the **Navigator**.
2. Click the **Locking** tab. The Locking tab appears.

Figure 17-30 Locking Tab

The screenshot shows the 'Locking' tab of a configuration window. The 'Optimistic Locking' radio button is selected and highlighted with a red circle. Under the 'By Version' section, the 'Database Field' dropdown is set to 'PROJECT.VERSION (NUMBER)', 'Version Locking' is selected, and the 'Store Version in Cache' checkbox is checked. The 'By Fields' section is also visible with options for 'All Fields', 'Changed Fields', and 'Selected Fields'.

3. Use the following information to complete each field on this tab:

Field	Description
Optimistic Locking	Specify that this descriptor uses optimistic locking (instead of pessimistic locking).
By Version	Specify that this descriptor is locked by version (instead of by fields).
Database Field	Select the VERSION database field. TopLink will lock the descriptor based on the version of this database field.
Version Locking	Specify that this locked by version number (instead of time stamp).
Store Version in Cache	Specify that TopLink should store the version locking information in the cache.

4. Repeat this procedure to implement locking for the **Employee** descriptor.
5. Click **Save** to save the project.

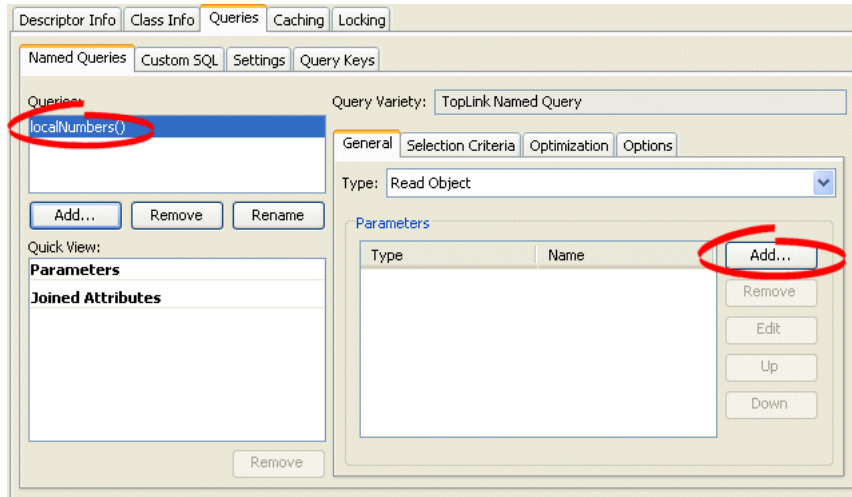
Named Queries

To create a simple query to find all employees whose telephone number includes a specific area code, use this procedure:

1. Select the **PhoneNumber** descriptor in the **Navigator**. Its properties appear in the Editor.
2. Click the **Queries** tab. The Queries tab appears, containing three subtabs.
3. Click the **Named Queries** subtab.

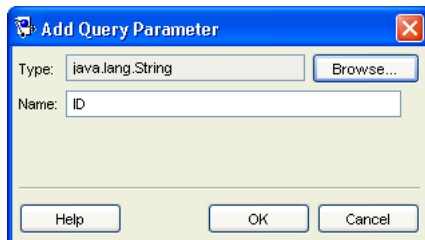
4. Click **Add** to create a new named query. In the Add Named Query dialog box, leave the **Type** as **ReadObject**, type `localNumbers` in the **Name** field, and click **OK**. TopLink Workbench adds the query.
5. Select the newly created `localNumbers` query. In the **Parameters** area of the General tab of the Named Queries tab, click **Add**. The Add Query Parameter dialog box appears.

Figure 17-31 General Tab on Named Queries Subtab of the Queries Tab

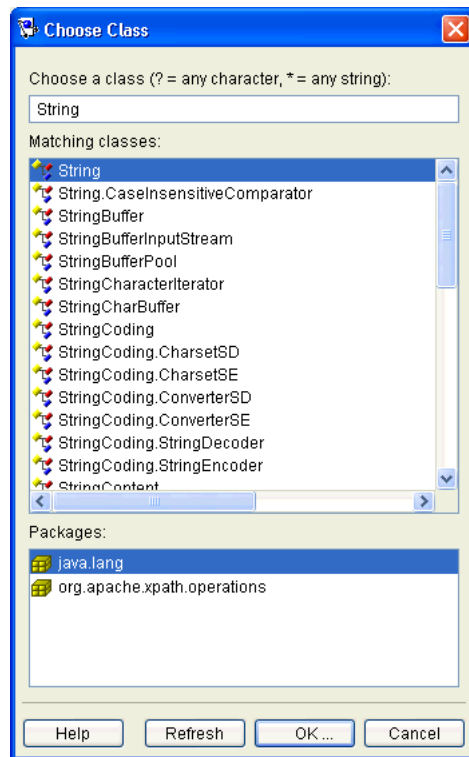


6. In the **Name** field of the Add Query Parameter dialog box, type **ID**. To configure the **Type**, click **Browse**. The Choose Class dialog box appears.

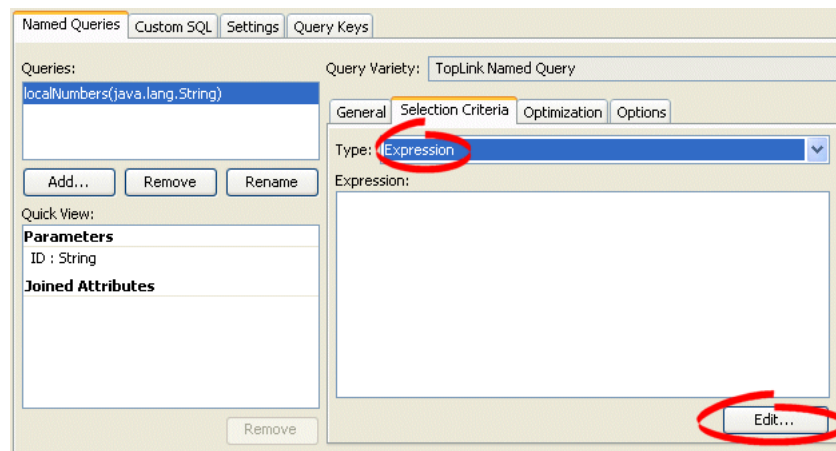
Figure 17-32 Add Query Parameter Dialog Box



7. In the **Choose a class** field of the Choose Class dialog box, type **String**. In the **Package** area, select `java.lang` and click **OK**. TopLink Workbench adds the parameter.

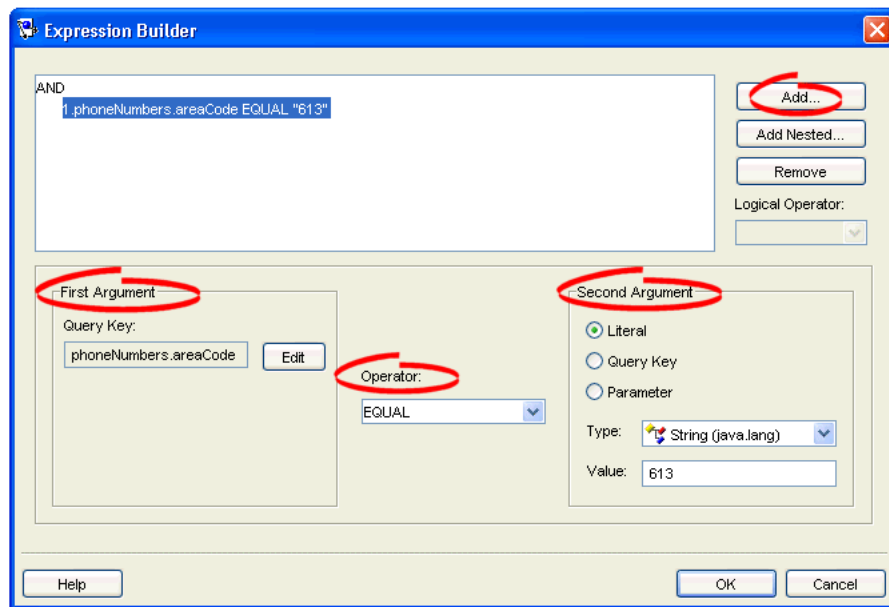
Figure 17–33 Choose Class Dialog Box

8. Click the **Selection Criteria** subtab on the Names Queries subtab of the Queries tab. The Selection Criteria subtab appears.

Figure 17–34 Selection Criteria Subtab on Named Queries Tab

9. Select **Expression** as the Type to create a TopLink expression for this query, and click **Edit**. The Expression Builder dialog box appears.
10. Click **Add** to create a new expression. TopLink Workbench adds the expression node.

Figure 17-35 Expression Builder Dialog Box



11. Use the following information to enter data in each field on this dialog box:

Field	Description
First Argument	Click Edit and select the areaCode attribute.
Operator	Select EQUAL .
Second Argument	Select Literal . Select String as the Type , type 613 as the Value .

This expression will find all employees whose telephone number has a **613** area code.

12. On the Expression Builder dialog box, click **OK**. The Selection Criteria subtab of the Named Queries subtab now includes the expression.

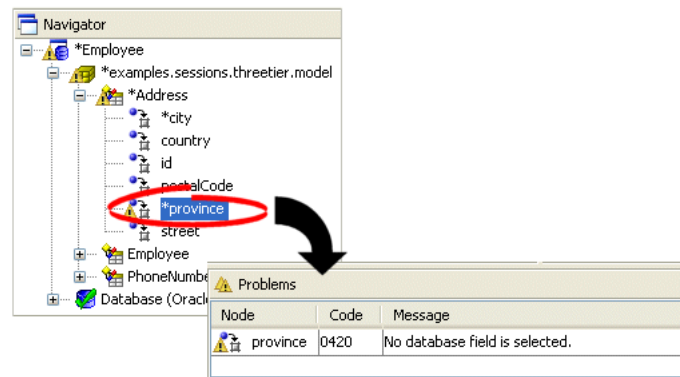


13. Click **Save** to save the project.

Step 6: Verifying the Project

Congratulations! You have built your first TopLink Workbench project.

Ensure that there are no warning icons on any element in the **Navigator**. If there are, select the element in the **Navigator** to display the complete error information in **Problems** window.

Figure 17-36 Sample Problems Window

Refer to [Chapter 14, "TopLink Workbench Error Reference"](#) for detailed information on each error.

After fixing any errors, you are ready to deploy the tutorial project. Continue with [Chapter 18, "Deploying the Tutorial Project"](#).

Deploying the Tutorial Project

This section describes how to deploy the **Employee** tutorial project. Deploying a TopLink application consists of the following steps:

- [Step 1: Generating Deployment Information](#)
- [Step 2: Creating Sessions Configuration](#)
- [Step 3: Packaging for Deployment](#)
- [Step 4: Deploying to an Application Server](#)
- [Step 5: Running the Tutorial Application](#)

Step 1: Generating Deployment Information

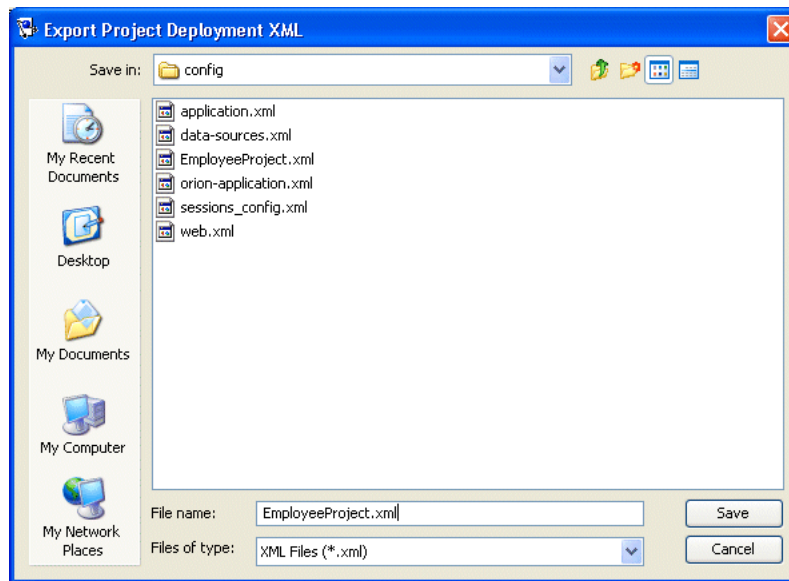
TopLink Workbench can generate multiple types of deployment information. In this tutorial project, you will generate deployment information as an XML file (named `EmployeeProject.xml` file) which will be used by the TopLink session at run time.

Creating the project.xml File



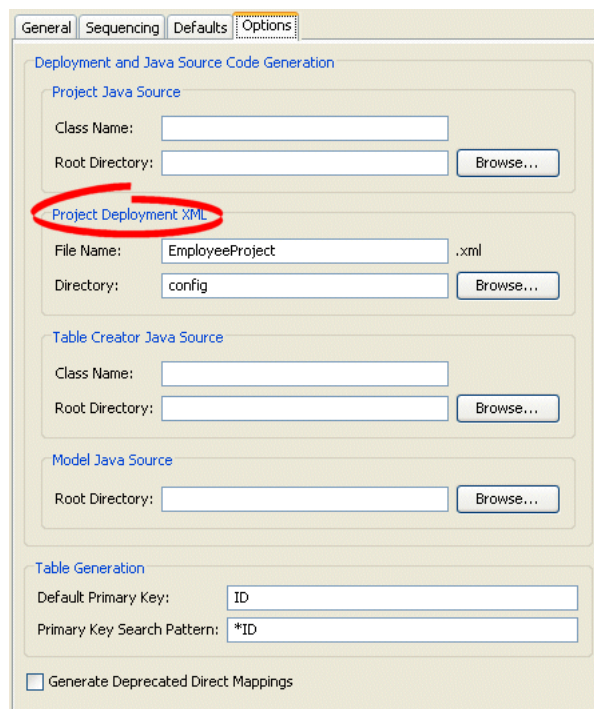
Use this procedure to export the deployment XML for the **Employee** project:

1. Select the **Employee** project in the **Navigator**, then click **Export Deployment XML**.
2. The Export Project Deployment XML dialog box appears.

Figure 18–1 Export Project Deployment XML Dialog Box

3. Configure **File name** as **EmployeeProject** and use the file chooser controls to select the `<TUTORIAL_HOME>\config` directory in which to save the `EmployeeProject.xml` file. Click **OK**.

Notice that the project's Options tab retains this **Project Deployment XML** information. When generating subsequent deployment XML for this project, you will not have to re-enter the file name and location.

Figure 18–2 Project Deployment XML Options on Options Tab

Step 2: Creating Sessions Configuration

A TopLink sessions configuration (the `sessions.xml` file) is the primary means by which the application performs all persistence operations with the database that contains its persistent objects at run time.

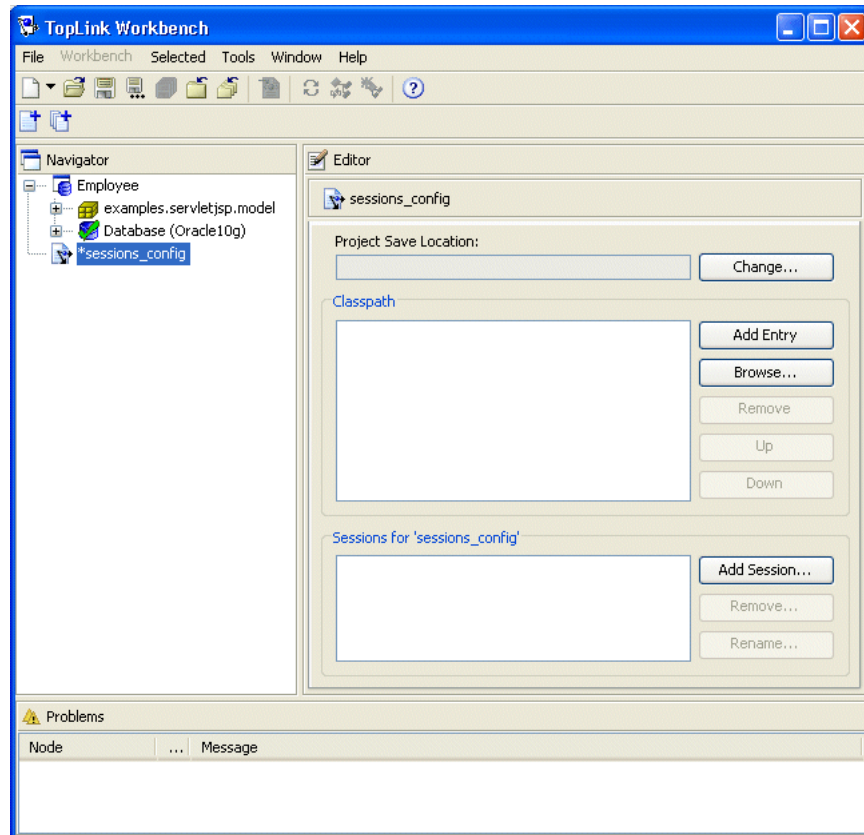
Creating the `sessions.xml` File

To create the sessions configuration for the Employee project, use this procedure:



1. Click **New** on the toolbar and select **Sessions Configuration**. TopLink Workbench creates a new sessions configuration (`sessions.xml`) file.

Figure 18-3 TopLink Workbench



2. Click **Save** on the toolbar. TopLink Workbench prompts for a directory location.
3. Select the `<TUTORIAL_HOME>\config` directory in which to save the `sessions_config.xml` file, and click **OK**.

Creating a Server Session

Server sessions provide session management to a single data source (including shared object cache and connection pools) for multiple clients in a three-tier architecture using database or EIS platforms. This is the most flexible, scalable, and commonly used session.

The Employee project uses a single server session to manage this data. TopLink supports multiple server sessions and also session brokers which allow a single session to connect to multiple databases.

To create a new server session to include in the sessions configuration file, use this procedure:

1. Right-click the **sessions configuration** object in the **Navigator** and select **New > Session** from the context menu. The Create New Session dialog box appears.



You can also create a new session by clicking **New Session**, or clicking **Add Session** in the Editor.

Figure 18–4 Create New Session Dialog Box

2. Use the following information to complete each field in the Create New Session dialog box, and then click **OK**:

Field	Description
Name	Enter servletJSP as the name for this session.
Use Server Platform	Select Use Server Platform and select the platform of your application server (such as OC4J 10g).
Select Data Source	Click Database and select the Platform of your database. This should be identical to the platform of the database you created when building the Employee project.
Select Session	Select Server Session . Server sessions provide session management to a single data source (including shared object cache and connection pools) for multiple clients in a three-tier architecture. This is the most flexible, scalable, and commonly used session.

TopLink Workbench adds the session to the sessions configuration.



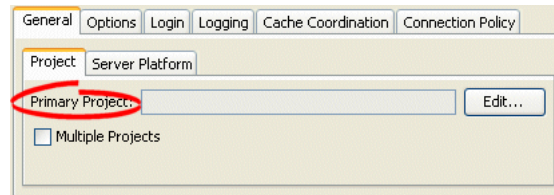
3. Click **Save** to save the project.

Associating the Project With the Session

Now that the base session has been created, you can add the deployment XML information from the Employee tutorial project to the session.

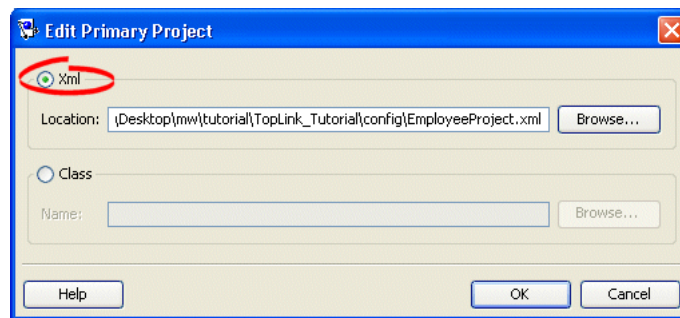
1. Expand the sessions configuration object in the **Navigator** and select the servletJSP session object. Its properties appear in the Editor.
2. Click the **Project** subtab on the **General** tab. The Project tab appears.

Figure 18-5 General Tab-Project Subtab



3. To add the deployment XML you generated from the Employee project, click **Edit**. The Edit Primary Project dialog box appears.

Figure 18-6 Edit Primary Project Dialog Box



4. Select **XML**, then click **Browse**. The file chooser dialog box appears.
5. Select the XML file you generated in "[Step 1: Generating Deployment Information](#)" on page 18-1 and click **Open**.
6. Click **OK** on the Edit Project dialog box.
7. Click **Save** to save the project.



Updating the data-sources.xml File

When building the Employee project, you defined a *development login* to access the database tables while developing the mapping project (see "[Creating a Database Development Login](#)" on page 17-5).

Because this is a servlet/JSP tutorial that is deployed to a J2EE application server, you must now configure a deployment login, using this procedure:

1. Using a text editor, open the `<TUTORIAL_HOME>\config\data-sources.xml` file (see [Example 18-1](#)).

Note: This sample `data-sources.xml` file was compatible with OC4J 10.1.3 at time of publication. You may wish to compare this sample file with the `data-sources.xml` file in your OC4J `<OC4J_HOME>/j2ee/home/config` directory to verify the XML version and DTD.

Example 18–1 Tutorial data-sources.xml File for OC4J

```

<?xml version="1.0" standalone="yes"?>
<!--<!DOCTYPE data-sources PUBLIC "Orion data-sources"
"http://xmlns.oracle.com/ias/dtds/data-sources.dtd">-->

<data-sources
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:noNamespaceSchemaLocation="http://xmlns.oracle.com/oracleas/schema/data-sources-10_
1.xsd"
  schema-major-version="10"
  schema-minor-version="1">

  <connection-pool name="TopLink Examples Connection Pool">
    <connection-factory factory-class="oracle.jdbc.pool.OracleDataSource"
      user="USERNAME"
      password="PASSWORD"
      url="URL">
    </connection-factory>
  </connection-pool>

  <managed-data-source connection-pool-name="TopLink Examples Connection Pool"
    jndi-name="jdbc/TopLinkDS"
    name="TopLinkDS"
    tx-level="local"/>

</data-sources>

```

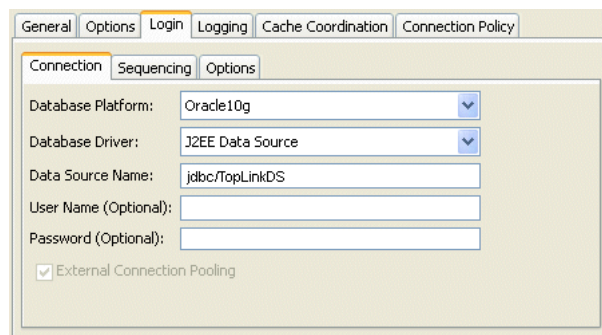
2. Configure the `user`, `password`, and `url` attributes to match the development login you created in ["Creating a Database Development Login"](#) on page 17-5.
3. Save and close the `data-sources.xml` file.

Creating a Database Deployment Login

Now you will configure a *session login* for deployment that will access the database at run time by way of the J2EE data source you defined (see ["Updating the data-sources.xml File"](#) on page 18-5), using this procedure:

1. Using TopLink Workbench, select the session object in the **Navigator**. Its properties appear in the Editor.
2. Click the **Login** tab. The Login tab appears. It contains three subtabs.
3. Click the **Connection** tab. The Connection tab appears.

Figure 18–7 Login Tab–Connection Subtab



4. Use the following information to complete each field on this tab:

Field	Description
Database Platform	Verify that the platform is correct.
Database Driver	Select J2EE Datasource .
Data Source Name	Enter the name (jdbc/TopLinkDS) of the J2EE data source configured in the tutorial data-sources.xml file (see "Updating the data-sources.xml File" on page 18-5).



5. Click **Save** to save the project.

Configuring Logging Options

The TopLink logging framework records TopLink behavior to a log file or session console. All log messages include a date stamp plus connection and server information, when applicable. In this tutorial project, you will log the login/logout per server session. After acquiring the session, detailed session information will be logged.

1. Select the session object in the **Navigator**. Its properties appear in the Editor.
2. Click the **Logging** tab. The Logging tab appears.

Figure 18–8 Logging Tab

The screenshot shows the 'Logging' tab in a configuration window. The tabs at the top are General, Options, Login, Logging (selected), Cache Coordination, and Connection Policy. The main area has radio buttons for 'No Logging', 'Server', 'Java', and 'Standard' (selected). Below 'Standard' is a 'Logging Level' dropdown set to 'fine'. There are radio buttons for 'Console' (selected) and 'File'. A 'Log Location' text box contains 'standard output' and a 'Browse...' button. An 'Options' section contains checkboxes for 'Log Exception Stack Trace' (unchecked), 'Print Connection' (checked), 'Print Date' (checked), 'Print Session' (checked), and 'Print Thread' (unchecked).

3. Use the following information to complete each field on this tab:

Field	Description
Standard	Select the TopLink standard logging mechanism.
Logging Level	Select Fine . This logging level will show you the SQL that TopLink creates. The different logging levels lets you customize the type and amount of information recorded in the log.
Console	Select to display the logging information to the standard console.



4. Click **Save** to save the project.

Step 3: Packaging for Deployment

To package the tutorial application for deployment, you must create the following files:

- [Domain JAR File](#)
- [WAR File](#)
- [EAR File](#)

To create these files, simply execute the `build` script in the `<TUTORIAL_HOME>` directory. For Windows, select the `build.cmd` file. For UNIX, select the `build.sh` file.

The `build` script compiles the Java source files and creates a domain JAR file (`servletjsp.jar`) and an EAR file (`servletjsp.ear`) in the current working directory. The EAR file contains the required WAR file.

You are encouraged to experiment with the tutorial Java source files and JSP (see ["Source Files"](#) on page 16-2). Using the `build` script, you can compile your changes and recreate the `servletjsp.jar` and `servletjsp.ear` file.

Domain JAR File

The domain JAR file contains the TopLink files and domain objects required by the application. The `<TUTORIAL_HOME>\servletjsp.jar` file contains the following:

- `EmployeeProject.xml` (the deployment XML)
- `sessions.xml` (the sessions configuration)
- `*.class` (the mapped classes required by the application, in a fully-resolved directory structure)

The `sessions.xml` file and the `employeeProject.xml` file must appear at the root of the JAR file. The class directory structure must start at the root of the JAR file.

WAR File

The WAR file contains the web application files used in the tutorial project, including the following:

- `*.jsp` files that provide the dynamic content for the application
- Static HTML content for the client application
- The same contents as the domain JAR file (see ["Domain JAR File"](#) on page 18-8)

In this tutorial, the WAR file is a component of the EAR file (see ["EAR File"](#) on page 18-8).

EAR File

The EAR file contains the WAR file and the application server-specific files required for deployment. The `<TUTORIAL_HOME>\servletjsp.ear` file contains the following:

- WAR file (see ["WAR File"](#) on page 18-8)
- `data-sources.xml` file
- `application.xml` file

- <ApplicationServer>-application.xml file (for example, for OC4J, orion-application.xml)

Step 4: Deploying to an Application Server

To deploy the tutorial application to an application server, you must deploy the EAR file as follows:

Note: This procedure assumes you are using OC4J. If you are using another application server, refer to your application server documentation for more specific information.

1. Start the OC4J application server by executing the following command:

```
java -jar <OC4J_HOME>/j2ee/home/oc4j.jar
```
2. Execute the deploy script in the <TUTORIAL_HOME> directory.
 For Windows, select the deploy.cmd file. For UNIX, select the deploy.sh file.

Step 5: Running the Tutorial Application

After deploying the tutorial application to a running application server instance, open a Web browser to your local host location.

For example, for OC4J, use:

<http://localhost:8888/servletjsp/mainPage.jsp>

Figure 18–9 shows the initial JSP for the tutorial application.

Figure 18–9 Tutorial JSP Demo Page

TopLink JSP Demo

Overview

This demo shows JSPs and servlets using TopLink's facilities to create, view and edit basic information of an employee system.

Options:

Query Employees by Name: (leave blank for findAll)

[Create A New Employee](#)

Glossary

This glossary contains terms and abbreviations that you should be familiar with when using Oracle TopLink.

attribute

A variable of a class or object. In TopLink, an *attribute* describes all instance variables of a class. Every attribute contains a single mapping. Attributes store primitive data such as integers, and simple Java types such as `String` or `Date`.

authentication

The means by which a data source validates a user's identity and determines whether or not the user has sufficient privileges to perform a given action.

bean class

The implementation of the bean. The bean is accessed from the client using the home and remote interfaces.

bean-managed persistence (BMP)

A scheme for persisting entity beans that requires the developer to manually code the methods that perform the persistence.

Compare to [container-managed persistence \(CMP\)](#).

branch class

Has a persistent superclass and also has subclasses. By default, queries performed on the branch class return instances of the branch class and any of its subclasses. However, the branch class can be configured so that queries on it return only instances of itself without instances of its subclasses.

Compare to [leaf class](#).

class

A category of objects. Classes allow data and method to be grouped together.

class indicator field

A field in the table of the root class that indicates which subclass should be instantiated

client session broker

A collection of client sessions, one from each server session associated with the session broker.

connection pool

A collection of reusable connections to a single data source.

container-managed persistence (CMP)

A scheme for persisting entity beans that uses information supplied by the developer or deployer to perform the persistence

Compare to [bean-managed persistence \(BMP\)](#).

custom SQL

Refers to any non-TopLink-generated SQL used through TopLink. This includes hard-coded SQL and stored procedure calls.

data definition language (DDL)

The data definition part of the structured query language (SQL). TopLink Workbench can generate DDL creation scripts that can be used to create tables on the desired database.

database session

A database session provides a client application with a single data store connection, for simple, standalone applications in which a single connection services all data store requests for one user.

default mapping

A relational persistence framework term that refers to making the framework automatically generate the object descriptor metadata (including such things as mappings, login data, database platform, locking, and foreign keys). Default mapping is available for TopLink projects using EJB 2.0 CMP applications with OC4J.

dependent class path (IBM WebSphere)

Location where nonbean classes are specified. TopLink requires that the bean classes be included here since they are referenced by the project.

deployment descriptor

A set of XML files that provide the additional required information to install an EJB within its server. Typically, this includes security, transaction, relationship, and persistence information.

Compare with TopLink [descriptors](#).

descriptors

An TopLink object that describes how an object's attributes and relationships are to be represented in relational database table(s). An "TopLink descriptor" is not the same as a [deployment descriptor](#), although it plays a similar role.

direct access

By default, TopLink accesses public attributes directly when writing the attributes of the object to the database or reading the attributes of the object from the database.

Compare to [method access](#).

direct mapping

There are two basic ways of storing object attributes directly in a table:

-
- The information can be stored directly if the attribute type is comparable to a database type.
 - If there is no database primitive type that is logically comparable to the attribute's type, it must be transformed on its way to and from the database

TopLink provides five classes of direct mappings.

Compare to [relationship mapping](#).

Enterprise Java Beans (EJB)

EJB are server-side domain objects that fit into a standard component-based architecture for building enterprise applications with Java. They are objects that become distributed, transactional, and secure components. TopLink Workbench uses three types of EJB: [session beans](#), [entity beans](#), and [message-driven beans](#).

expressions

The TopLink equivalent of an SQL conditional clause. TopLink expressions are specified using the `Expression` and `ExpressionBuilder` classes.

entity beans

EJB that represent a persistent data object. TopLink uses two schemes for persisting entity beans: [bean-managed persistence \(BMP\)](#) and [container-managed persistence \(CMP\)](#).

fetch group

A performance enhancement that defines a subset of object attributes to be loaded initially and ensures that all other attributes are loaded on demand.

hub

A common connection point for devices in a network.

identity map

Used to cache objects for performance and to maintain object identity.

See also [object identity](#).

independent relationship

A relationship in which the source and target are public objects that exist independently; the destruction of one object does not necessarily imply the destruction of the other.

Compare to [private relationship](#).

indirection

An indirection object is one that acts as a stand-in for another object. In TopLink, indirection is implemented through value holders, which delay database access through acting as substitute for any object relationships.

inheritance

Describes how a child class inherits the characteristics of its parent class. TopLink supports multiple approaches to database implementations that preserve the inheritance relationship.

in-memory query

A query that is run against the shared session cache.

instantiate

Create an instance of a Java class.

J2C

The J2EE Connector architecture (J2C) adapter is a way to persist Java objects to a nonrelational data source, such as XML.

J2SE

The Java 2 Platform, Standard Edition (J2SE) is the core Java technology platform. It provides software compilers, tools, runtimes, and APIs for writing, deploying, and running applets and applications in Java.

J2EE

The Java 2 Platform, Enterprise Edition (J2EE) is an environment for developing and deploying enterprise applications. J2EE includes a set of services, APIs, and protocols for developing multitiered web-based applications.

J2EE Containers

A J2EE container is a run-time environment for Enterprise Java Beans (EJB) that includes such basic functions as security, life cycle management, transaction management, and deployment services. J2EE containers are usually provided by a J2EE server, such as Oracle Containers for J2EE.

Java Messaging Service (JMS)

The JMS API is a protocol for communication that provides asynchronous communication between components in a distributed computing environment.

Java Naming and Directory Interface (JNDI)

The JDBC specification recommends using a JNDI naming service to acquire a connection to a database. TopLink supports acquiring a database connection in this fashion. To take advantage of this feature, construct and configure an instance of `oracle.toplink.jndi.JNDIConnector` and pass it to the project login object using the `setConnector` method.

Java Transaction API (JTA)

The Java Transaction API (JTA) specifies the interfaces between a transaction manager, a resource manager, an application server, and transactional applications involved in a distributed transaction system.

leaf class

Has a persistent superclass in the hierarchy but does not have subclasses; queries performed on the leaf class can return only instances of the leaf class.

Compare to [branch class](#).

locking policy

A mechanism that ensures one user does not overwrite another users's work. TopLink descriptors support optimistic and pessimistic locking policies.

mappings

Describe how individual Java objects and attributes relate to a data source.

message-driven beans

An EJB that processes asynchronous Java Messaging Service (JMS) messages. For TopLink clients, a message-driven bean is simply a JMS consumer with no conversational state and no home or remote interfaces.

method access

The application registers accessor methods for the attribute.

Compare to [direct access](#).

named query

A TopLink query that is created and stored, by name, in a session for later retrieval and execution

object identity

Ensures that each object is represented by one and only one instance in the application; that is, multiple retrievals of the same object return references to the same object instance, not multiple copies of the same object. Violating object identity can corrupt the object model.

See also [identity map](#).

optimistic locking

Also known as write locking; allows unlimited read access to objects. A client can write an object to the database only if the object has not changed since it was last read.

Compare to [pessimistic locking](#).

packet

A piece of a message transmitted over a packet-switching network. One of the key features of a packet is that it contains the destination address in addition to the data.

packet time-to-live

A number of hops that session data packets can take before expiring. The default is 2.

See also [packet](#).

persist

In object technology, the storage of an Java object by a data source.

pessimistic locking

Objects are locked before they are edited, which ensures that only one client is editing the object at any given time.

Compare to [optimistic locking](#).

primary key

A field (or combination of fields) that uniquely identifies a record in the data source.

private relationship

A relationship in which the target object is considered to be a private component of the source object; the target object cannot exist without the source and is accessible only through the source object; furthermore, if the source object is destroyed, the target object is destroyed as well.

Compare to [independent relationship](#).

query manager

An object, owned by a descriptor, that controls the way the descriptor accesses the database. The query manager generates its own default SQL to access the database in a transparent manner.

query optimization

TopLink supports two forms of query optimization: joining and batch reading. Their purpose is to optimize database access through reducing the number of database calls required to read a group of objects.

relationship

In TopLink, a reference between two TopLink-enabled objects.

relationship mapping

Persistent objects use relationship mappings to store references to instances of other persistent classes. The appropriate mapping class is chosen primarily by the cardinality of the relationship. TopLink provides five classes of relationship mappings.

Compare to [direct mapping](#).

Remote Method Invocation (RMI)

A set of protocols that enable Java objects to communicate remotely with other Java objects.

remote session

A remote session is a client-side session that communicates over RMI with a corresponding client session and server session on the server side. Remote sessions handle object identity and marshalling and unmarshalling between client side and server side.

service channel

A name of the TopLink coordinated cache channel to which sessions subscribe in order to participate in the same coordinated cache.

session beans

EJB that represent a business operation, task, or process. TopLink can use session beans to make the regular Java objects they access persistent, or to wrap other legacy applications.

stale data

An artifact of caching, in which an object in the cache is not the most recent version committed to the data source.

TopLink session broker

A mechanism that enables client applications to transparently access multiple databases through a single TopLink session.

unit of work

A transactional TopLink session that allows for a transaction to occur at the object level not only the database level. Changes to objects are not visible globally until the unit of work is committed.

value holder

A wrapping object used by TopLink to delay database access.

A

access

- data access, 84-1
- direct, 35-32
- modifiers, 4-42
- optimizing data access, 11-14
- remote sessions, 75-31

access method

- direct, 35-14
- generating, 4-30
- mappings, 35-15
- method, 35-14
- specifying, 35-32

access modifiers, classes, 4-42

acquiring

- client sessions, 78-6
- sessions, at runtime, 75-5
- unit of work, 101-1

activating descriptors, 4-10

Add Named Query dialog, 28-12, 28-18, 28-20

Add New Class dialog, 4-40, 4-53

Add New Table button, 4-22

Add or Refresh Class button, 4-50

addConstraingDependencies(), 102-16

address

- multicast group, 91-4
- multicast port, 91-5

Add/Update Existing Tables from Database

- button, 4-23

advanced properties for descriptors, 22-7

After Load tab, 28-79

aggregate collection relational mappings

- and EJB, 36-11
- configuring, 44-1
- understanding, 36-10

aggregate descriptors

- about, 26-5
- aggregate object mapping, 26-5
- EIS projects, 26-8
- EJB 3.0, 26-6
- inheritance, 26-17
- relational projects, 26-5
- XML projects, 26-8

aggregate object relational mapping

- aggregate descriptors, and, 26-5

aggregate object relational mappings

- configuring, 47-1
- understanding, 36-12

aggregation, isolated client sessions, 75-24

AllFieldsLockingPolicy, 26-20

allows none, 28-18, 97-14

allows null, 28-18, 97-14

amending descriptors, 2-20, 26-5, 28-78

- see also* after load

announcement delay, 91-11

Ant, integrating with Oracle TopLink

- Workbench, 4-53

any collection XML mappings

- configuring, 72-1
- understanding, 65-29

any object XML mappings

- configuring, 71-1
- understanding, 65-27

application development

- deploying, 10-1
- mapping, 2-20
- querying, 2-13, 74-2
- troubleshooting, 15-1

application layer, remote sessions, 75-30

application servers

- EJB support, customizing, 7-25
- integrating with Oracle TopLink, 7-1
- logging, 75-9
- optimization, 11-30
- setter parameter type checking, 7-25
- single-object finder return type checking, 7-25, 7-26
- software requirements, 7-2
- target platforms, 2-5
- unknown primary key class support, 7-25

architectures

- application, 1-4
- BMP, 1-5, 2-33
- cache, 90-1
- choosing, 2-3
- CMP, 1-5, 2-27
- EIS, 2-4, 56-5
- EJB entity beans, 1-5, 2-27, 2-33
- EJB session bean facade, 1-5, 2-24
- locking, 2-10
- optimistic locking, 2-10

- Oracle TopLink, 1-1
- pessimistic locking, 2-10
- selecting, 2-5
- session brokers, 75-26
- sessions, 75-2
- three-tier, 1-4, 2-21
- two-tier, 1-6, 2-23
- unit of work, 100-1
- web services, 1-5, 2-36
- arguments, binding in query, 29-6
- array
 - dimensionality, 4-45, 4-49
 - object-relational mappings, 49-2
- AsOfClause, 99-2
- asynchronous change propagation, 91-2
- AttributeChangeTrackingPolicy
 - about, 100-8
 - OC4J CMP integration, 100-8
 - OC4J EJB 1.x CMP integration, 100-8
 - OC4J EJB 2.x CMP integration, 100-8
 - OC4J EJB 3.0 CMP integration, 100-8
 - other application servers, 100-8
- attributes
 - adding to descriptors, 4-44
 - array dimensionality, 4-45, 4-49
 - changes, tracking, 28-72
 - final, 4-45, 4-48
 - in TopLink Workbench, Navigator window, 4-9
 - lazy loading. *see* fetch groups
 - static, 4-45, 4-48
 - transforming, 35-29, 35-31, 36-15
 - transient, 4-45, 4-48
 - unmapping, 34-6
 - volatile, 4-45, 4-48
- Attributes tab, 4-44, 4-48
- Attunity Connect platform, 84-3
- auditing
 - authentication, 84-6
 - unit of work, 102-20
- authentication
 - about, 84-5
 - auditing, 84-6
 - proxy authentication, 84-5
 - simple JDBC authentication, 84-5
 - three-tier architecture, 84-5
 - two-tier architecture, 84-5
- Automap, 34-2
- automapping descriptors
 - about, 34-2
 - see also* mappings
- automatic table generation
 - about, 33-4
 - configuring, 8-12

B

- Base64 encoded strings, 35-25
- batch options
 - mappings, 37-7
 - writing, 11-15

- batch reading
 - in query objects, 98-10
 - read optimization, and, 11-19
- batch writing
 - about, 11-15, 11-26, 86-11
 - dynamic, 11-15
 - dynamic, `setMaxBatchWritingSize()`, 11-15
 - MySQL4 platform, 86-9
 - non-parameterized, 11-15
 - parameterized, 11-15
 - `setMaxBatchWritingSize()`, 11-15
- BEA WebLogic
 - deploying to, 10-2
 - deployment exceptions, 15-3
 - modifying persistence descriptor, 8-13
 - setting classpath, 7-15
 - setting shared library, 7-15
 - transport layer, 75-31
 - using a security manager, 7-20
- @Bean fetch=lazy, 33-9
- beans
 - session beans, 2-25
 - stateful beans, 2-25
 - stateless beans, 2-25
- bidirectional relationships
 - about, 36-2
 - generating, 4-31
 - in one-to-one mappings, 35-34
 - target keys, 36-5
 - with indirection, 35-35
- `bindAllParameters()` method, 23-7
- bindings
 - arguments, 29-6
 - input paramters, 98-19
 - JAXB, 20-10
 - LOB mappings, 34-3
 - see* parameter binding
- BLOB
 - mapping to, 34-2
- BMP
 - and EJB 1.1, 2-34
 - and EJB 2.0, 2-34
 - and TopLink, 2-33
 - deployment files, 8-9
 - descriptors, 26-3, 28-44
 - packaging for deployment, 9-5
- boolean logic in expressions, 97-3
- branch classes, 26-12, 26-13
- buttons. *see* toolbars
- bye array binding, LOB mappings, 34-3
- Byte array Base64, 35-25

C

- cache
 - about, 2-12, 2-18, 90-1
 - architecture, 90-1
 - configuring, 90-6, 99-21
 - coordination, 90-9, 91-1, 92-1, 93-1
 - descriptor level, 28-35

- disabling during read query, 96-34
- distributed, 90-9
- expiration, 22-19, 28-40, 99-22
- expression limitations, 96-31
- identity maps, using, 75-29
- in-memory queries, 96-30, 96-31
- internal query object cache, 96-36, 99-21, 99-22
- invalidation, 22-19, 28-40, 90-7, 90-8
- isolated client sessions, 75-24
- isolation, 90-6, 90-9
- object cache, 96-36
- object cascading refresh, 96-35
- object refresh, 96-35
- optimizing, 11-13
- project level, 22-13
- queries, 90-6, 96-29
- query cache, 96-36
- `readObject` method, and, 98-2
- refreshing, 28-27, 90-8, 96-35
- restrictions, 96-36
- service channel, 91-3
- sessions, 75-3, 75-32, 90-2
- stale data, 90-6
- storing query results, 96-36, 99-21
- type and size, 22-13, 28-35
- unit of work cache, 90-2
- cache coordination
 - about, 90-9
 - application server clustering, and, 7-4
 - avoiding stale data, 90-7
 - CMP projects, 8-11
 - EJB Entity Beans with BMP architecture, and, 2-36
 - explicit query refreshes, 90-7
 - JMS, 92-1
 - `orion-ejb-jar.xml`, 8-11
 - packet time-to-live, 91-15
 - permissions, 7-23
 - RMI, 93-1
- cache invalidation, avoiding stale data, 90-7
- cache synchronization. *see* cache coordination
- `cacheAllStatements()`, 23-7
- `cacheQueryResults()`, 28-25
- cache-synchronization property, 8-11
- Caching tab, 28-35, 28-38, 28-39, 28-42
- call
 - call queries, 96-16
 - EIS, 98-24
 - EJBQLCall, 98-23
 - SQLCall, 98-18
 - StoredFunctionCall, 98-22
 - StoredProcedureCall, 98-20
- Call object, queries, 96-3
- cascading
 - object refresh, 96-35
 - optimistic version locking, 26-18
 - write queries, compared to non-cascading, 96-14, 98-13
- catalog, database, 4-22
- `catchExcpetions()`, 77-19
- change policy
 - about, 28-70
 - attribute change tracking, configuring, 28-72
 - deferred change detection, configuring, 28-71
 - empty transaction, 100-6
 - object change tracking, configuring, 28-71
 - unit of work, 100-6
- change tracking
 - attribute, configuring, 28-72
 - deferred, configuring, 28-71
 - object, configuring, 28-71
- changed items, displaying in TopLink Workbench Navigator window, 4-9
- `ChangedFieldsLockingPolicy`, 26-20
- changing package names, 4-52
- `checkDatabase()`, 77-18
- checking in/out projects, 5-3
- `checkInstantiationPolicy()`, 77-19
- Choose a Schema Context dialog box, 31-3, 32-2
- Choose Query Key dialog box, 97-14
- Choose Relationships to Generate dialog box, 4-30, 4-31
- Choose Root Element dialog box, 31-4, 32-5
- class extraction method
 - about, 26-14
 - inheritance, 26-15
- class indicator
 - about, 26-13
 - class extraction method, 26-14
 - class indicator field, 26-13, 41-1
- class loader
 - host application, 2-5
 - loading session, 78-4
- class modifiers, 4-42
- Class tab, 4-41, 4-42, 4-43
- classes
 - access modifiers, 4-42
 - adding and refreshing, 4-50
 - branch, 26-13
 - creating, 4-40, 4-52
 - `CursoredStream`, optimizing, 99-18
 - Database Exception, 98-24
 - `DatabaseMapping`, 33-25
 - default null values, 35-36
 - `DeleteObjectQuery`, 98-13
 - `ExpressionBuilder`, 97-15
 - generating from database, 4-30
 - `InsertObjectQuery`, 98-13
 - `InsertObjectQuery`, 100-12
 - interfaces, 4-43
 - leaf, 26-13
 - merging files, 5-5
 - methods, adding, 4-47
 - naming, 4-41
 - non-descriptor classes, 4-51
 - object model, 2-16
 - Performance Profiler, 11-3
 - persistent requirements, 2-11
 - preferences, 4-16
 - refreshing, 4-50

- removing, 4-51
- root, 26-12
- troubleshooting, 14-22
- tutorial, 16-2
- unit of work, 100-12
- UpdateObjectQuery, 98-13, 98-14
- ValueHolderInterface, 2-11, 33-7, 36-9
- VariableOneToOneMapping, 36-6
- see also* specific class name
- classpath
 - adding, 22-3
 - BEA WebLogic, 7-15
 - configuring, 4-2, 7-15, 7-21
 - connector.jar, 4-2
 - custom Collection class, 4-3
 - DRIVER_CLASSPATH, Oracle TopLink Workbench, 4-2
 - IBM WebSphere, 7-21
 - J2C adapter, 4-2
 - JDBC driver, 4-2, 4-23
 - JDBC_CLASSPATH, 4-2
 - Oracle TopLink Workbench DRIVER_CLASSPATH, 4-2
 - relative, 22-3
 - setting for BEA WebLogic, 7-15
 - troubleshooting, 14-22
 - xdb.jar file, 4-3
- class-table relationships, 16-6
- client sessions
 - about, 75-1, 75-13, 75-25
 - acquiring, 78-6
 - configuration, 82-1
 - example, 75-15
 - shared resources, 75-15
- client-controlled transactions, 102-24
- client-server architecture. *See* two-tier
- CLOB
 - mapping to, 34-2
- clones
 - copying methods, 28-69
 - merging changes, 102-12
 - post-commit, avoiding, 102-30
 - unit of work, 100-2, 100-9
- Cloudscape platform, 84-3
- clustering, integrating TopLink with, 7-4
- CMP
 - and EJB 1.1, 2-27
 - and EJB 2.x, 2-27
 - and EJB 3.0, 2-27
 - and TopLink, 2-27
 - CMPPolicy, 26-3, 28-44
 - deploying, 8-9, 10-2
 - descriptors, inheritance, 26-4
 - external transactions, 100-3
 - isolated client sessions, 75-24
 - local transaction, 102-25
 - non-deferred write, 102-25
 - OC4J persistence, 7-5
 - packaging for deployment, 9-4
 - setter parameter type checking, 7-25
 - single-object finder return type checking, 7-26
 - transaction attribute, 102-24
 - unit of work, 100-3, 102-24
 - unknown primary key class support, 7-25
- code generation, optimizing, 11-8
- collapsing items in Navigator window, 4-9
- collection class
 - specifying, 98-9
 - specifying in query objects, 98-9
- collections
 - persistent requirements for mappings, 2-11
 - query results, 96-8
- comments
 - descriptors, 28-8, 28-9
 - mappings, 35-18
 - projects, 22-21
- commit
 - and Java Transaction API, 100-10
 - failure, resuming after, 102-14
 - resuming unit of work after, 102-14
- Communication Exceptions, 13-64
- composite collection EIS mappings
 - configuring, 61-1
 - example, 56-8
 - understanding, 56-7
- composite collection XML mappings
 - configuration, 70-1
 - configuring, 70-1
 - understanding, 65-25
- composite descriptors
 - about, 26-5
 - composite object mapping, 26-8
 - EIS projects, 26-8
 - XML projects, 26-8
- composite direct collection EIS mappings
 - configuring, 59-1
 - understanding, 56-6
- composite direct collection XML mappings
 - configuring, 68-1
 - understanding, 65-14
- composite EIS descriptors, 27-5
- composite object EIS mappings
 - composite descriptors, and, 26-8
 - configuring, 60-1
 - understanding, 56-7
- composite object mappings
 - composite descriptors, and, 26-8
- composite object XML mappings
 - configuration, 69-1
 - configuring, 69-1
 - understanding, 65-21
- composite primary key, 36-9
- concrete class. *see* container policy
- concurrency
 - about, 2-17
 - exceptions, 13-25
 - server session, 75-18
- Concurrency Exceptions, 13-25
- configurations
 - about, 76-2

- creating, 76-2
- development environment, 5-1
- new, 76-2
- Oracle JDeveloper, 5-1
- session, 76-2, 76-4, 76-6, 76-9
- conforming
 - about, 102-8
 - alternatives, UnitOfWork method
 - writeChanges, 102-11
 - alternatives, UnitOfWork properties, 102-12
 - descriptors, and, 102-11
 - in-memory queries, 102-9
 - queries, alternatives to, 102-11
- ConnectBy, 99-7
- connection policy
 - configuring, 77-19
 - exclusive connections, 77-19
 - lazy connection acquisition, 77-20
- connection pool
 - about, 84-7
 - connection count, 89-1
 - ConnectionPolicy, 75-20
 - external, 84-8
 - internal, 84-7, 84-8, 84-9
 - lazy connection allocation, 75-18, 78-9
 - named, 84-9
 - parameter binding, 11-16
 - prepared statement caching, 11-16
 - read, 84-8
 - sequence, 20-5, 84-8
 - server session, 75-18
 - sessions, and, 75-3
 - size, 89-1
 - write, 84-8
- Connection Specifications tab, 24-3
- Connection tab, 24-3
- connections
 - about, 84-6
 - connection pool, 84-7
 - exclusive write connection, 77-19
 - lazy acquisition, 77-20
 - reading through the write connection, 102-27
- connector.jar, 4-2, 20-7
- container configuration file, 8-5
- container policy
 - about, 35-26
 - custom Collection class, 4-3
 - sorting, in memory, 35-26
- container-controlled transactions, 102-24
- context
 - JAXB path, 20-14
 - menus, 4-5
 - schema, 31-2, 32-2
- Context.SECURITY_CREDENTIALS, 91-9
- Context.SECURITY_PRINCIPAL, 91-9
- Conversion Exception, 13-26
- Converter tab
 - object type mappings, 35-22
- converters
 - custom, 36-3

- object type, 35-22
- coordinated announcement delay, 91-11
- coordinated cache, 8-11
 - configuring, 91-1
 - naming service, 91-7
 - service channel, 91-3
- copy policy
 - about, 28-69
 - method, 28-69
 - setting, 28-70
- copying project objects, 5-6
- Copying tab, 28-70
- CORBA
 - Oracle TopLink transport layer support, 75-31
 - Transaction Service *see* OTS
- Create New Project button, 21-2
- Create New Project dialog box, 21-2
- Create New Session dialog, 76-4, 76-6, 76-8
- Create Project from JAXB dialog, 21-6
- Create Project from OC4J dialog, 7-9
- creating
 - configurations, 76-2
 - expressions, 97-12
 - sessions, 76-4, 76-6, 76-8, 88-1
- Crimson XML parser, 7-3
- cursored streams
 - example, 99-18
 - optimizing, 99-18
 - remote sessions, 75-32
 - usage example, 75-32
- cursors
 - as query results, 96-8
 - traversing scrollable, 99-16
- Custom Calls tab, 31-6
- custom SQL
 - Custom SQL tab, 29-7
 - unit of work, 102-15
- customization
 - about, 12-1
 - data types, 12-1
 - EIS, 12-1
 - mapping extensions, 12-1
 - overview, 2-15
 - XML, 12-1

D

- data access
 - about, 84-1
 - authentication, 84-5
 - connection pool, 84-7
 - connections, 84-6
 - optimizing, 11-14, 11-15
 - platforms, 84-3
- data level queries
 - example, 97-12
 - in expressions, 97-11
- data source platform
 - about, 84-3
 - Attunity Connect database, 84-3

- Cloudscape database, 84-3
- databases, 84-3
- DB2 database, 84-3
- EIS, 84-4
- HSQL database, 84-3
- Informix database, 84-3
- J2C adapter, 84-4
- JDBC drivers, 84-3
- Microsoft Access database, 84-3
- MySQL4 database, 84-3
- Oracle database, 84-3
- Oracle8 database, 84-3
- Oracle9 database, 84-3
- PointBase database, 84-3
- SQLAnywhere database, 84-3
- SQLServer database, 84-3
- SybasePlatform database, 84-3
- data sources
 - configuring, 85-1
 - nontransactional, 84-1
 - transactional, 84-1
 - troubleshooting, 14-23
- Database Exceptions, 13-27, 98-24
- database fields, configuring, 37-2
- database functions, in expressions, 97-3
- database login
 - parameter binding, 86-9
 - prepared statement caching, 86-9
- Database Preferences, 4-17
- database queries
 - about, 96-10, 98-4
 - fetch groups, 96-12
 - join reading, 96-12
 - object level modify query, 96-13, 96-14
 - object level read query, 96-11, 96-13
 - partial object query, 96-11
 - read all query, 96-11
 - read object query, 96-11
 - report query, 96-15
- database schema
 - tables, 4-22
 - tutorial, 16-4
- database sessions
 - about, 75-2, 75-28
 - cache, 75-32
 - configuration, 83-1
 - creating, 76-8
- database tables
 - about, 4-21
 - adding to database, 4-22
 - creating, 4-22
 - descriptors and classes, generating, 4-30
 - EJB entity generation, 4-31
 - fields, 4-25
 - generating, 4-29, 4-30, 4-32
 - importing, 4-22
 - Java source generation, 21-15
 - JDBC driver classpath, 4-23
 - properties, 4-25
 - references, 4-27, 4-28
 - removing, 4-24
 - renaming, 4-24
 - schema, 4-22
 - SQL generation, 4-29
 - TopLink Workbench, Navigator window, 4-9
- DatabaseException class, 98-24
- DatabaseLogin, 84-2
- DatabaseMapping class, 33-25
- DatabaseQuery, 96-3
- DatabaseRow, 36-15
- databases
 - catalog, 4-22
 - common problems, 14-23
 - connect to, 4-21
 - creating reference tables on, 4-27
 - custom drivers, 4-18
 - disconnect from, 4-21
 - drivers, 4-18
 - exceptions, 98-24
 - fields, configuring, 37-2
 - for project, 21-3
 - Java type conversion, 6-3
 - linking, 75-28
 - log out of, 4-22
 - logging into, 4-22, 23-7
 - logins, 86-1
 - mapping. *See* mappings
 - platform, 21-3, 23-2, 24-2, 86-1, 87-1
 - preferences, 4-17
 - schema, 4-22
 - schema manager, 6-1
 - tables, 4-21
 - TopLink Workbench, Navigator window, 4-9
 - troubleshooting, 14-23
 - type conversion, schema manager, 6-3
 - using with Oracle TopLink Workbench, 4-21
- DatabaseSession class
 - logging SQL and messages, 75-10
- DB2
 - platform, 84-3
 - schema manager type conversion, 6-3
- DBase platform, 84-3
- default mapping
 - about, 33-4
 - automatic table generation, and, 33-4
 - configuring, 8-12
 - default table generator, 6-6
- Default Mapping Exception, 13-75
- default table generator
 - default mapping, 6-6
 - table creator, creating, 6-4
- defaults
 - login level null values, 85-5
 - mapping level null values, 35-12
 - null values, 35-12, 35-36, 85-5
 - optimization, 11-8
 - root, 32-5
 - see also* preferences
- DefaultSequence, 86-6
- deferred change detection

- configuring, 28-71
- DeferredChangeDetectionPolicy, 100-7
- Delete All Interaction tab, 63-3
- deleteObject(), 29-8
- DeleteObjectQuery, 98-13
- deletes
 - controlling order, 102-16
 - delete operation, 98-4
 - queries, EIS mappings, 63-3
- demarcation of unit of work transactions, 100-2
- dependent objects
 - non-deferred write, 26-4
- deploy tool
 - about, 10-5
 - troubleshooting, 10-6
 - using with WebSphere Studio Application Developer, 10-5
- deploying
 - about, 2-14, 10-1
 - application server requirements, 7-1
 - BEA WebLogic, 10-2
 - CMP applications, 10-2, 10-4
 - database login, 23-7
 - entity beans overview, 2-14
 - generating XML for, 8-3
 - hot deployment, 10-4
 - IBM WebSphere, 10-3
 - Java applications, 10-1
 - JSP and Servlet applications, 10-1
 - modifying BEA WebLogic persistence
 - descriptor, 8-13
 - non-CMP applications, 10-4
 - packaging, 9-1
 - Session Bean applications, 10-1
 - troubleshooting, 15-1
- deployment descriptors, 26-3
- Deployment Exceptions, 13-67
- deployment exceptions
 - BEA WebLogic deployment, 15-3
 - Deployment Exceptions (14001-14033), 13-67
 - IBM WebSphere deployment, 15-12
- deployment files
 - BMP applications, 8-9
 - CMP applications, 8-9
 - creating, 8-1
 - descriptors, 26-3
 - EJB 3.0, 8-2, 9-1
 - JARs, troubleshooting, 15-1
 - Java applications, 8-8
 - JSP and Servlet applications, 8-8
 - Session Bean applications, 8-8
 - XML, generating, 8-3
- deployment XML, exporting, 21-14
- DeploymentXMLGenerator, 8-3
- Descriptor Event Listener, 28-60
- Descriptor Event Manager
 - about, 26-8
 - Descriptor Event Listener, 28-60
 - domain object methods, 28-57
 - event types, 28-57

- handlers, 26-8, 28-57, 28-60
- handlers, Descriptor Event Listener, 28-60
- understanding, 26-8
- descriptor events
 - about, 26-8
 - Descriptor Event Listener, 28-60
 - domain object methods, 28-57
 - handlers, 26-8, 28-57, 28-60
 - types of, 28-57
 - understanding, 26-8
- Descriptor Exceptions, 13-2, 13-29
- Descriptor Info tab, 28-3, 28-5, 28-7, 29-2, 29-4, 31-2, 31-4, 32-2, 32-4, 32-5, 32-6
- DescriptorEventListener, 28-60
- descriptors
 - about, 19-2, 26-1
 - advanced properties, default, 22-7
 - aggregate, 26-5, 27-2
 - aggregate, EJB 3.0 and, 26-6
 - aggregate, relational projects and, 26-5
 - amending, 2-20, 26-5, 28-78
 - API, 26-22
 - architecture, 26-2
 - attributes, adding, 4-44
 - automapping, 34-2
 - automatically mapping, 34-2
 - BMP, 26-3, 28-44
 - cache refreshing, 28-27
 - change policy, 28-70
 - child inheritance, 28-49
 - class, 27-2
 - CMP, 26-3, 28-44
 - CMPPolicy, 26-3, 28-44
 - comments, 28-8, 28-9
 - composite, 26-5, 26-8, 27-5
 - composite EIS, 27-5
 - configuring, 28-1
 - conforming, 102-8
 - creating, 27-1, 27-2, 27-5
 - custom EIS interactions for basic persistence, 31-6
 - custom SQL queries for basic persistence, 29-6
 - deactivating, 4-10
 - default mappings, 33-4
 - default root, 31-4
 - deployment information, 26-3
 - Descriptor Event Listener, 28-60
 - domain object methods, 28-57
 - EIS, 26-11, 27-5, 31-1
 - EIS projects, 26-8
 - EJB, 26-3, 28-44
 - EJB information, 26-3, 28-44
 - errors, 4-10, 4-11, 14-4
 - event handlers, 26-8, 28-57, 28-60
 - events, 26-8, 28-57
 - existence checking, 11-13, 28-42
 - fetch groups, 28-77
 - files, merging, 5-5
 - generating from database, 4-30
 - hierarchy, inheritance, 26-23
 - history policy, 28-73

- identity maps, 22-14, 28-35, 28-37, 28-39
- in Java, 26-22
- inactive, 4-10
- inheritance, 26-3, 26-12, 28-49, 28-50
- instantiation, 11-13
- interface, 27-2, 28-31, 28-33
- mapping, 28-3, 29-2, 33-4, 34-1, 34-2
- merging, 5-5
- named queries, 28-10
- nondesoriptor classes, 4-51
- object-relational, 26-11, 30-1
- optimizing, 11-13
- parent inheritance, 28-50
- projects, 19-2, 26-1
- query key interfaces, 28-31
- query timeout, 28-24, 28-26
- read only, 28-4
- registering with sessions, 75-12, 77-2, 77-3, 77-10
- relational, 26-11, 27-2, 29-1
- removing, 4-51
- returning policy, 28-65
- root EIS, 27-5
- root element, 32-5
- schema context, 31-2, 32-2, 32-3
- sequencing, 26-9, 29-3
- TopLink Workbench, Navigator window, 4-9
- types of, 26-1
- validating, 27-6
- XML, 26-12, 32-1
- XML projects, 26-8
- detachement indirection, 33-9
- developing applications with Oracle TopLink, 2-1
- development environments
 - about, 3-2
 - configuring, 5-1
 - database logins, 23-7
- development process
 - about, 2-1
 - additional support, 2-3
 - stages of, 2-1
 - with Oracle TopLink, 2-1
- development tools
 - about, 3-1
 - profiler, 11-2
 - schema manager, 6-1
- dimensionality, array, 4-45, 4-49
- direct access
 - about, 22-4, 35-14
 - specifying, 35-32
- direct collection relational mappings
 - configuring, 45-1
 - example, 36-11
 - understanding, 36-11
- direct collections
 - session broker limitations, 75-27
- direct EIS mappings
 - configuring, 58-1
 - understanding, 56-5
- direct field
 - in direct collection mappings, 45-2
- direct key fields, 46-2
- direct map relational mappings
 - configuring, 46-1
 - direct keys, 46-2
 - direct value, 46-1
 - understanding, 36-12
- direct mappings
 - generating deprecated, 22-12
 - with EJB, 36-4
- direct value fields, 46-1
- direct XML mappings
 - configuring, 67-1
 - understanding, 65-5
- directionality in mappings, 36-2
- direct-to-field mappings
 - ObjectTypemapping deprecated, 36-3
 - SerializedObjectMapping deprecated, 36-3
 - type conversions, 37-3
 - TypeConversionMapping deprecated, 36-3
- direct-to-field relational mappings
 - configuring, 38-1
 - options, 38-1
 - timestamp support, 37-3
 - understanding, 36-4
- direct-to-XMLType relational mappings
 - configuring, 39-1
 - understanding, 36-4
- Discovery Exception, 13-77
- DMS profiler
 - about, 11-4, 75-11
 - accessing with JMX, 11-7
 - and JMX
 - nouns, 11-4, 75-11
 - selecting, 77-11, 77-14, 77-17, 77-20
 - sensors, 11-4, 75-11
- document information in XML schemas, 4-34, 4-36, 4-38
- documentation
 - hosted, 4-14
 - See also* Help
- does exist write object, 11-27
- dontOptimizeDataConversion(), 11-15
- doPrivileged(), 7-4
- DRIVER_CLASSPATH
 - Oracle TopLink Workbench environment, 4-2
- drivers, custom database, 4-18
- dynamic batch writing
 - about, 11-15
 - setMaxBatchWritingSize(), 11-15
- dynamic fetch groups, querying with, 99-4

E

- Editor window, about, 4-4, 4-10
- EIS
 - about, 20-8
 - architecture, 2-4
 - call, 98-24
 - custom interactions for basic persistence, per descriptor, 31-6

- indexed records, configuring, 31-5
- interactions, 31-6, 96-6, 98-24
- mapped records, configuring, 31-5
- mappings, 20-8, 56-2
- projects, 21-3
- queries, 96-6
- record format, configuring, 31-5
- XML records, configuring, 31-5
- EIS descriptors
 - composite, 27-5
 - configuring, 31-1
 - default root, 31-4
 - locking policy, 28-62
 - root descriptor, 27-5
 - schema context, 31-2
 - setDataTypeName, 31-5
 - understanding, 26-11
- EIS mappings
 - about, 56-1, 56-2
 - architecture, 56-5
 - composite collection, 56-7, 61-1
 - composite direct collection, 56-6, 59-1
 - composite object, 56-7, 60-1
 - configuring, 57-1
 - direct, 56-5, 58-1
 - jaxb:class support, 56-3
 - list support, 56-3
 - one-to-many, 56-12, 63-1
 - one-to-many, key on source, 56-13
 - one-to-many, key on target, 56-15
 - one-to-one, 56-8, 62-1
 - one-to-one, key on source, 56-9
 - one-to-one, key on target, 56-10
 - transformation, 56-17, 64-1
 - types of, 56-1
 - union support, 56-3
 - xsd:list, 56-3
 - xsd:union, 56-3
- EIS projects
 - configuring, 24-1
 - connector.jar, 20-7
 - indexed records, 20-9
 - mapped records, 20-9
 - sequencing, 20-5
 - understanding, 20-7
 - XML records, 20-9
- EIS queries, 96-6
- EIS record types, supported, 56-2
- EISLogin, 84-2
- EJB
 - descriptors, 21-10
 - isolated client sessions, 75-24
 - setter parameter type checking, 7-25
 - single-object finder return type checking, 7-25
 - unknown primary key class support, 7-25
- EJB 1.1
 - and BMP, 2-34
 - and CMP, 2-27
- EJB 1.x
 - indirection, 33-9
 - serialization, 33-9
- EJB 2.x
 - and BMP, 2-34
 - and CMP, 2-27
 - default mapping, 33-4
 - indirection, 33-9
 - serialization, 33-9
- EJB 3.0
 - <J2EE-Container>-jar.xml file, 8-6
 - and CMP, 2-27
 - attribute change tracking policy, OC4J CMP integration, 7-5, 100-8
 - Bean annotation, fetch=lazy, 33-9
 - default mapping, 33-4
 - deployment files, 8-2, 9-1
 - deployment files, OC4J CMP integration, 7-5
 - detachment, 33-9
 - Embedded annotation, 26-6
 - packaging for deployment, 9-1
 - packaging, OC4J CMP integration, 7-5
 - projects.xml file, 8-3, 8-5
 - serialization, 33-9
 - sessions.xml file, 8-5
 - toplink-ejb.xml file, 8-8
 - value holder indirection, 33-9
- EJB descriptors, opening projects with, 21-10
- EJB entities
 - BMP indirection, 33-9
 - CMP hot deployment, 10-4
 - deployment overview, 2-14
 - EJB 1.x indirection, 33-9
 - EJB 2.x indirection, 33-9
 - EJB 3.0 indirection, 33-9
 - generating, 4-31
 - hot deployment, 10-4
 - indirection, 33-9
 - inheritance, 26-4, 26-17
 - inserting after ejbCreate, 26-4
 - inserting after ejbPostCreate, 26-4
 - mapping, 22-6
 - non-CMP hot deployment, 10-4
 - non-deferred write, understanding, 26-3, 102-25
 - sequencing, 20-21
- EJB entity beans
 - and EJB 1.1, 2-27, 2-34
 - and EJB 2.x, 2-27, 2-34
 - and EJB 3.0, 2-27
 - with BMP architecture, 1-5, 2-33
 - with CMP architecture, 1-5, 2-27
- EJB Exception Factory Exceptions, 13-58
- EJB finders
 - about, 96-23
 - Call finders, 96-26
 - creating, 99-8
 - DatabaseQuery finders, about, 96-26
 - default finders, about, 96-26
 - default finders, creating, 99-8
 - EJB QL finders, about, 96-27
 - ejb-jar.xml options, 99-9
 - ejbSelect method, 96-28

- ejbSelect, creating, 99-15
 - ejbSelect, using, 99-15
 - expression finders, about, 96-27
 - named query finders, about, 96-27
 - predefined, about, 96-24
 - primary key finders, about, 96-27
 - redirect finders, about, 96-28
 - redirect finders, using, 99-12
 - single-object finder return type checking, 7-25
 - SQL finders, about, 96-28
- EJB Info tab, 28-44
- EJB JAR XML Exception, 13-84
- EJB Preferences, 4-16
- EJB QL
 - exceptions, 13-55
 - queries, 29-6, 96-5
 - query language, 96-5
- EJB session bean facade architecture
 - about, 1-5, 2-24
 - understanding, 2-24
- EJB Session Beans, 75-31
- ejbc
 - about, 10-2
 - troubleshooting, 10-3
- ejbCreate, 26-4
- ejb-jar.xml file
 - about, 8-5, 21-15
 - configuring, 8-5
 - corresponding to Oracle TopLink Workbench functions, 21-15
 - EJB finder options, 99-9
 - location, 22-7
 - managing, 5-6
 - preferences, 4-16
 - synchronization under EJB 2.0, 8-5
 - updating from, 21-16
 - writing, 21-16
- ejbPostCreate, 26-4
- @Embeddable, 26-6
- @Embedded, 26-6
- empty unit of work transactions, 100-6
- encrypting login passwords, 21-13
- enhanced validation exceptions, 8-4
- Enterprise Information Systems. *see* EIS
- entity beans
 - deployment, 2-14
 - descriptor information, 26-3
 - direct mappings, 36-4
 - indirection, BMP, 33-9
 - indirection, EJB 1.x, 33-9
 - indirection, EJB 2.x, 33-9
 - indirection, EJB 3.0, 33-9
 - sequencing with, 20-21
- environment
 - configuring, 4-2
 - JAVA_HOME, 4-2
 - JDBC_CLASSPATH, 4-2
 - proxy, 4-13, 4-14, 21-16
- error codes
 - 1-176, 13-2
 - 1-99, 14-1
 - 100-199, 14-2
 - 200-399, 14-4
 - 400-599, 14-11
 - 500-699, 14-15
 - 700-799, 14-20
 - 800-899, 14-21
 - 10001-10047, 13-58
 - 12000-12004, 13-64
 - 13000-13020, 13-64
 - 14001-14027, 13-67
 - 15001-15024, 13-70
 - 17001-17006, 13-71
 - 18001-18002, 13-74
 - 19001-19003, 13-75
 - 22001-22004, 13-77
 - 22101-22105, 13-77
 - 3001-3007, 13-26
 - 4002-4018, 13-27
 - 5001-5008, 13-29
 - 6001-6098, 13-30
 - 7001-7104, 13-42
 - 72000-72023, 13-84
 - 8001-8010, 13-55
 - 9000-9009, 13-56
- errors
 - about, 13-1
 - codes and descriptions, 13-1, 14-1
 - descriptors, 4-10, 4-11
 - migration, 7-13
 - Oracle TopLink Workbench, 14-1
- Event Manager, 75-5
- events
 - about, 26-8
 - client session, 75-6
 - database access, 75-6
 - Descriptor Event Listener, 28-60
 - domain object methods, 28-57
 - handlers, 26-8, 28-57, 28-60
 - listeners, sessions, 75-7
 - session, 75-5
 - session manager, 75-6
 - sever session, 75-6
 - types of, 28-57
 - unit of work, 75-6
- examples
 - composite collection EIS mapping, 56-8
 - context menu, 4-5
 - cursor streams, 99-17
 - direct collection mappings, 36-11
 - direct-to-field mappings, 36-4
 - exception handler, 77-13
 - indirection, 33-6
 - inheritance, 26-12
 - Oracle TopLink Workbench, 4-3
 - performance optimization, 11-22, 11-24
 - proxy indirection in code, 35-9
 - READALL finders, 99-10
 - report query, 98-6
 - scrollable cursors, 99-17

- serialized mapping, 33-10
- stored procedure call, 98-21, 98-22
- transformation mapping, 36-15
- transformation XML mapping, 56-18, 65-31
- Unit of Work, 100-6, 101-7
- write, write all, 98-3
- exception handler
 - about, 75-12
 - example, 77-13
 - selecting, 77-12
- exceptions
 - chained, 75-10
 - communication exceptions, 13-64
 - conversion exceptions, 13-26
 - database exceptions, 13-27, 98-24
 - Default mapping exception, 13-75
 - deployment exceptions, 13-67
 - descriptor exceptions, 13-2
 - discovery exceptions, 13-77
 - EJB exceptions factory, 13-58
 - EJB JAR XML exceptions, 13-84
 - EJB QL exceptions, 13-55
 - enhanced validation, 8-4
 - java.security.AccessControlException, 15-12, 15-13
 - JMS processing exceptions, 13-74
 - Migration utility exception, 13-82
 - optimistic locking, 13-29
 - query exceptions, 13-30
 - remote command manager exceptions, 13-77
 - SDK data store exceptions, 13-71
 - SDK descriptor exceptions, 13-75
 - selecting exception handler, 77-12
 - session loader exceptions, 13-56
 - synchronization exceptions, 13-70
 - Transaction exception, 13-79
 - validation exceptions, 13-42
 - XML conversion exception, 13-80
 - XML data store exceptions, 13-64
- exclusive connections
 - about, 102-28
 - internal read connection pool, 89-6
 - isolated sessions, 75-20, 77-19
 - named queries, 28-24
- existence checking, 102-5
 - descriptors, 28-42
 - projects, 22-8
- expanding items in Navigator window, 4-9
- expiration of objects in the cache, 22-19, 28-40
- explicit query refreshes, cache coordination, 90-7
- exporting
 - deployment XML, 21-14
 - Java model, 21-14
 - Java source, 21-14
 - preferences, 4-12
 - projects, 21-13
- Expression Builder, 97-12, 97-15
- Expression Builder dialog box, 97-12
- Expression class, 97-1
- ExpressionMath class, 97-1

- expressions
 - about, 96-3, 97-1
 - allows none, 28-18, 97-14
 - allows null, 28-18, 97-14
 - building, 97-12
 - comparing with SQL, 97-1
 - components, 97-2
 - creating, 97-12
 - data level queries, 97-11
 - database functions, 97-3
 - in relationships, 97-5
 - in-memory queries, limitations, 96-31
 - mathematical functions, 97-4
 - multiple, 97-10
 - one-to-one mappings, 97-5
 - outer joins, 98-11
 - parallel expressions, 97-11
 - parameterized, 97-7
 - platform functions, 97-16
 - query keys, 97-9
 - subqueries and subselects, 97-10
 - user-defined functions, 97-4, 97-16
 - using Boolean logic, 97-3
 - XML Type functions, 97-4
 - see also* queries
- external
 - applications, 102-26
 - connection pools, 84-8
 - controller, transaction, 100-1
 - JDBC pools, 2-35
 - transactions, 100-1
- external transaction controller
 - configuration, sessions, 77-14
 - session, 100-2

F

- factory name, JMS connection, 92-2
- failure, resuming unit of work after commit, 102-14
- features of Oracle TopLink, 1-4
- fetch groups
 - about, 26-4, 28-77
 - configuring, 28-77, 99-3
 - default, 28-77, 99-3
 - disabling, 99-3
 - dynamic, 99-4
 - object level read query, 96-12
 - read optimization, and, 28-77
 - size, 11-17
 - static, 99-3
- field references, 37-8
- Field uses XML Schema "type" attribute
 - option, 33-12
- fields in database tables, 4-25
- Fields tab, 47-2
- field-to-object attribute transformation, 35-29, 35-31
- files
 - JAXB-specific, 20-10
 - TopLink-specific, 20-11
 - see also* specific file name

- final attributes, 4-45, 4-48
- findAll, using, 96-26
- finders
 - caching options, 96-37
 - disabling cache, 96-37
 - managing large result sets, 99-19
 - refreshing results, 96-38
 - see also* EJB finders
- flashback queries
 - about, 96-18
 - historical client sessions, 81-1
- forceUpdateToVersionField(), 102-17
- foreign keys
 - about, 2-17, 37-8
 - configuring in EIS mappings, 62-1
 - EIS mappings, 63-1
 - multiple tables, 29-14
 - one-to-many mappings, 36-7
 - one-to-one mappings, 35-34, 36-5
 - parameterized expressions, 97-7
 - references, 14-12, 14-21
 - target, 36-6, 37-9
 - troubleshooting, 14-12, 14-21
- full identity map, 90-3

G

- garbage collection, managing, 101-8
- General Preferences dialog, 4-13
- Generate Classes and Descriptors dialog, 4-30
- Generate EJB Entity Classes and Descriptors dialog, 4-31
- generating
 - access method, 4-30
 - deployment JARs, troubleshooting, 15-1
 - deprecated direct mappings, 22-12
 - see also* exporting
- getCatalogs(), 4-22
- getField(), 97-11
- getImportedKeys(), 4-22
- getParameter, 97-8
- getPrimaryKeys(), 4-22
- getTable(), 97-12
- getTables(), 4-22
- getTableTypes(), 4-22
- getValue(), 33-7
- getValue() method, 33-7

H

- hard cache weak identity map
 - about, 90-4
 - when to use, 90-5
- help
 - about, 4-12
 - displaying, 4-12
- Help button., 4-12
- Help Preferences, 4-14, 4-16
- hierarchical queries
 - about, 96-18

- described, 99-6
- hints, Oracle Hints in queries, 96-18
- historical client sessions
 - about, 75-1, 75-24
 - cache, 75-33
 - limitations of, 75-25
- historical queries, 99-2
 - about, 96-20
 - see also* AsOfClause
- history policy, configuring, 28-73
- holders, value, 33-6
- host URL, JMS topic, 92-3
- hosted
 - documentation, 4-14
 - XSD files, 8-2, 8-4
- hot deployment
 - about, 10-4
 - CMP applications, 10-4
 - non-CMP applications, 10-4
- HSQL platform, 84-3

I

- IBM Informix Database native sequencing, 20-19
- IBM WebSphere
 - deploy tool, 10-5
 - deploying to, 10-3
 - deployment exceptions, 15-12
 - setting classpath, 7-21
- identity
 - about, 2-16, 90-3
 - cache, and, 90-3
 - using cache to preserve, 90-3
 - see also* identity map
- identity map cache
 - disabling during a write query, 98-14
 - refresh in read query, 96-35
- identity maps
 - about, 28-35, 28-37, 28-39, 75-29
 - cascading refresh during read query, 96-35
 - descriptors, 22-14, 28-35
 - example, 96-35
 - full, 90-3
 - guidelines for choosing type, 90-4
 - hard cache weak identity map, 90-4, 90-5
 - isolated client sessions, 75-24
 - no identity map, 90-4
 - refreshing during read query, 96-35
 - soft cache weak identity map, 90-4, 90-5
 - soft cache weak identity map and read optimization, 11-19
 - specifying, 28-35, 28-37, 28-39
 - weak, 90-3
 - weak identity map and read optimization, 11-19
- Identity tab. *see* Caching tab
- impedance mismatch, solving, 1-2
- Implementors tab, 28-33
- Import Tables from Database dialog, 4-23
- importing
 - classes, 4-16

- preferences, 4-12
- inactive descriptors
 - about, 4-10
 - mapping to, 37-6, 57-3
- independent relationships, 35-16
- indexed records, 56-2
- indirection
 - about, 2-18, 33-5
 - bidirectional relationships, 35-35
 - choosing the correct type, 35-4
 - configuring, 35-3
 - EJB, 33-9
 - EJB 1.x CMP, 33-9
 - EJB 2.x CMP, 33-9
 - EJB 3.0 CMP, 33-9
 - example, 33-6
 - many-to-many mappings, 36-9
 - nontransparent, 2-11
 - one-to-many mappings, 35-34
 - proxy indirection, 33-8
 - remote sessions, 75-32
 - serialization, 33-9
 - transparent, 2-11, 33-8
 - value holder, 33-6
 - ValueHolderInterface, 2-11
 - see also* proxy indirection, transparent indirection
- Informix platform, 84-3
- inheritance
 - about, 2-17, 26-3, 26-12
 - aggregate classes, 26-17
 - aggregate collection mappings, 36-10
 - branch classes, 26-12, 26-13
 - child descriptors, 28-49
 - class extraction, 26-14, 26-15
 - class indicator, 26-13, 26-14
 - descriptors, 26-3, 26-12, 28-49, 28-50
 - finding subclasses, 26-13
 - instantiating subclasses, 26-13
 - isolated client sessions, 75-24
 - leaf classes, 26-12, 26-13, 99-4
 - primary keys, 26-15
 - queries, 96-22
 - querying on hierarchy, 99-4
 - relational parent, 28-50
 - root class, 26-12, 28-49, 28-51
 - root class subclasses, finding in inheritance, 26-13
 - supporting with multiple tables, 26-16
 - supporting with one table, 26-15
 - transformed to relational model, 11-11
 - using with EJB, 26-4, 26-17
- inheritance hierarchies
 - descriptors, 26-23
 - querying on, 99-4
- Inheritance tab, 28-49, 28-51
- inherited subclasses, mapping, 28-56
- in-memory query
 - about, 90-6
 - check cache using exact primary key, 96-30
 - check cache using primary key, 96-30
 - check database if not in cache, 96-30
 - conform results in unit of work, 96-31
 - conforming, 102-9
 - expression limitations, 96-31
 - supported, 96-30
 - using, 96-30
- inner join, 97-5
- insert operation, 98-3, 98-4
- insertObject(), 29-7
- instantiation policy
 - about, 28-68
 - setting, 28-68
- Instantiation tab, 28-68
- integrity checker, 27-6
 - about, 75-12
 - configuring, 77-18
- interactions
 - about, 98-24
 - creating, 31-6
- interface alias
 - about, 29-10
 - creating, 29-11
- Interface Alias tab, 29-11
- interfaces
 - classes, implementing, 4-43
 - customizing, 4-13
 - descriptors, 27-2, 28-31, 28-33
 - queries, 96-22
 - query keys, 28-31
 - querying on, 99-4
- internal connection pool
 - about, 84-7
 - named, 84-9
 - read, 84-8
 - sequence, 84-8
 - write, 84-8
- internal query object cache
 - about, 96-36
 - configuring, 99-21, 99-22
 - expiration, 99-22
 - restrictions, 96-36
- internal transactions, 100-1
- invalidation of objects in the cache, 22-19, 28-40, 90-8
- IP address for multicast group, 91-4
- isolated client sessions
 - about, 75-1, 75-19, 80-1
 - configuration, 80-1
 - life cycle, 75-22
 - limitations of, 75-23
 - session event handlers, 75-21
 - with Oracle Virtual Private Database (VPD), 75-20
- isolated session
 - cache, 75-33
 - ConnectionPolicy, 75-20
 - exclusive connections, 75-20
 - supported databases, 75-20
- isolation
 - cache, 90-6
 - transaction levels, 100-2
 - unit of work transactions, 100-4

Iterator interface, 99-16

J

J2C adapters

- about, 84-4
- configuring for Oracle TopLink Workbench, 4-2
- EISLogin, 84-2
- selecting, 84-4
- with EIS, 21-3

<J2EE-Container>-ejb-jar.xml file, 8-5

<J2EE-Container>-jar.xml file

EJB 3.0, 8-6

J2EE

- parameter binding, 11-16
- prepared statement caching, 11-16
- web applications, 1-4

Java

- database tables, 21-15
- exporting to, 21-14
- integration with any datasource, 1-2
- iterators, 99-16
- object model, 2-11

Java applications

- deploying, 10-1
- deployment files, 8-8
- packaging for deployment, 9-1

Java Cryptography Extension, 21-13

Java Management Extensions. *see* JMX

Java Naming and Directory Interface. *See* JNDI

Java Object Builder, 75-4

Java streams

- described, 99-17
- optimizing, 99-18
- support for, 99-17

Java Transaction API

- and unit of work commit, 100-10
- and unit of work rollback, 100-11
- see also* JTA

Java Transaction Service *see* JTA

JAVA_HOME, 4-2

java.security.AccessControlException, 15-12, 15-13

java.util.Collection interface, 35-26

java.util.Map interface, 35-26

java.util.Set interface, 35-26

javax.ejb.EntityBean interface, 4-31

JAXB

- creating projects from, 21-6
- files, 20-10
- generating project from the command line, 21-8
- jaxb:class, and EIS mappings, 56-3
- jaxb:class, and XML mappings, 65-4
- proxy configuration, 21-8
- tljxb.cmd, 21-8
- tljxb.sh, 21-8
- typesafe enumeration converter, 35-25
- understanding, 20-10
- validation, 20-14
- XML projects, 20-10

JAXB typesafe enumeration converter

- configuring, 35-25
- understanding, 33-24

JAXBContext, 20-14

JCE. *see* Java Cryptography Extension

JConnect, 11-14

JDBC

- adaptor for EIS, 56-5
- database gateway for EIS, 56-5
- driver classpath, 4-23
- JConnect, 11-14
- Sybase JConnect, 11-14

JDBC drivers

- about, 84-3
- configuring for Oracle TopLink Workbench, 4-2
- fetch size, 11-14, 11-17
- general properties, 11-14
- mapping LOBs, 34-3
- selecting, 84-3

JDBC pools

- external with EJB Entity Beans with BMP architecture, 2-35

JDBC_CLASSPATH, 4-2

JDeveloper. *See* Oracle JDeveloper

JMS

- connection factory name, 92-2
- coordinated cache, 92-1
- Processing Exceptions, 13-72, 13-74
- topic host URL, 92-3
- topic name, 92-1

JMX

- about
- and DMS profiler
- DMS profiler, 11-7

JNDI naming service, 91-7

joining

- about, 96-12
- expressions, and, 97-5
- mappings, and, 40-1
- one-to-many, about, 98-11
- one-to-one mappings, 40-1, 98-11
- optimizing reads, 11-19
- queries, and, 98-11
- QueryManager expressions, 99-4
- read queries, 11-19, 98-11

JSP and Servlet applications

- deploying, 10-1
- deployment files, 8-8
- packaging for deployment, 9-2

JTA

- about, 100-3
- and unit of work, 100-3
- isolated client sessions, 75-24
- unit of work, 100-1

JTA/JTS

- using with EJB Entity Beans with BMP architecture, 2-36

just-in-time reading. *see* indirection

K

Key Converter tab, 46-3

key pairs

database table reference, 4-28

troubleshooting, 14-12, 14-21

keys

about, 2-16

foreign, 2-17, 35-34, 36-5

foreign, target, 36-6

inheritance, 26-15

multiple tables, 29-13

primary, 26-15, 29-13, 35-2, 36-6, 36-9, 41-3

read-only settings, 35-2

reference key field, 36-11, 36-12

variable class relationships, 36-6, 41-3

L

large result sets, managing in finders, 99-19

lazy attribute loading and read optimization, 28-77

lazy connection

acquisition, 77-20

allocation, 75-18, 78-9

lazy loading. *see* indirection

lazy reading. *see* indirection

leaf classes, 26-12, 26-13, 99-4

life cycle of unit of work, 100-5

LOB

mapping to, 34-2

local

documentation, 4-14

transactions, 102-25

locked files, 5-6

locking policy

AllFieldsLockingPolicy, 26-20

ChangedFieldsLockingPolicy, 26-20

configuring, 28-62

field locking, 26-20

optimistic, 26-9, 26-18, 26-20

optimistic version locking, 26-17

OptimisticLockException, 26-18, 26-21

pessimistic locking policy, 26-9, 26-21

SelectedFieldsLockingPolicy, 26-21

stale data, and, 90-6

three-tier architectures, optimistic locking
and, 26-21

three-tier architectures, pessimistic locking
and, 26-22

TimestampLockingPolicy, 26-18

understanding, 26-9, 26-17

version locking, 26-18

VersionLockingPolicy, 26-18

Locking tab, 28-63

log into database, 4-22

Log Out of Database, 4-22

log out of database, 4-22

logging

application server, 75-9

chained exceptions, 75-10

java.util.logging, 75-9

log level, 75-10

Oracle Enterprise Manager 10g, 75-10

output, 75-10

permissions, 7-24

sessions, 75-7, 77-4, 77-5

TopLink native logging, 75-8

types, 75-8

Logging tab, 77-5

login

CMP deployment, 20-3

database, 23-5, 23-7, 86-1

deployment, 20-3

development, 20-4

platforms, and, 20-4

projects, and, 20-3, 74-1, 84-2

role in project, 20-3, 84-2

session, 20-3, 77-4

session role, non-CMP, 20-3

logMessages method, 75-7

look and feel, specifying, 4-13

M

Manage Non-Descriptor Classes dialog, 4-51

management, source control, 5-3

manager, session events, 75-5

many-to-many mappings

relation table, 36-9

relation tables, 43-1

session broker limitations, 75-27

many-to-many relational mappings

configuring, 43-1

EJB, 36-9

understanding, 36-8

mapped records, 56-3

mapping extensions

custom data types, 12-1

JAXB typesafe enumeration converter, 33-24,
35-25

object type converter, 33-12, 35-22

serialized object converter, 33-10, 35-18

simple type translator, 33-12, 35-23

transformation mappings, 33-14

type conversion converter, 33-11, 35-20

mappings

about, 2-15, 2-16, 19-1, 19-2, 33-1, 34-1

access types, 35-15

aggregate collection mappings and EJB, 36-11

anyType mapping, 66-3

as part of the application development
process, 2-20

automatic, 34-2

batch options, 37-7

class hierarchy, 33-25

comments, 35-18

configuring, 34-1, 35-1

database field, 37-4

default mapping, 33-4

deprecated, generating, 22-12

direct access, 11-13, 35-14

- directionality, 36-2
- EIS mappings, 20-8, 33-27, 56-1, 57-1
- EJB 2.0 entities, 22-6
- errors, 14-11
- example, 33-3
- extensions, about, 33-10
- hierarchy, 33-25
- inactive descriptors, 37-6, 57-3
- indirection, 11-13, 33-5, 35-3
- isolated client sessions, 75-23
- manually configuring, 34-1
- many-to-many, 36-9
- many-to-many, with EJB, 36-9
- method access, 35-14, 35-32
- null values, 35-12, 35-36
- object-relational, 49-1, 50-1
- one-to-many object, with EJB, 36-8
- one-to-one with EJB, 36-6
- optimizing, 11-13
- OX mappings, 33-27
- projects, and, 19-2
- read only, 35-2, 35-3
- relation tables, 43-1, 43-2
- relational, 36-1, 37-1
- removing, 34-6
- to tables, 28-3, 29-2
- TopLink Workbench, Navigator window, 4-9
- types of, 33-1
- XML mappings, 65-1, 66-1
- mathematical functions, in TopLink
 - expressions, 97-4
- menu bar, 4-5
- menus
 - about, 4-4, 4-5
 - context menus, 4-5
 - menu bar, 4-5
- merging
 - changes in clones, 102-12
 - Oracle TopLink Workbench project files, 5-4
 - project files, 5-4
- messages, error, 13-1, 14-1
- metadata
 - about, 2-12, 2-19
 - advantages, 2-19
 - creating, 2-20, 2-21
 - mapping and configuration, 19-1
 - project metadata, 2-20
 - session metadata, 2-21
- Metalink, 2-3
- method access
 - about, 22-4, 35-14
 - setting, 35-32
- methods
 - adding, 4-47
 - getValue(), 33-7
 - setValue(), 33-7
 - wrapper policy, 28-76
 - see also* specific method name
- Microsoft Access
 - platform, 84-3

- schema manager type conversion, 6-3
- Microsoft SQL Database native sequencing, 20-19
- migrating
 - error messages, 7-13
 - OC4J persistence to TopLink, 7-5
 - Oracle TopLink Workbench projects, 21-10
 - troubleshooting, 7-13
- Migration Utility Exception, 13-82
- model source, exporting, 21-14
- modifiers, class, 4-42
- multicast group address, coordinated cache, 91-4
- multicast port, coordinated cache, 91-5
- multiple sessions, 75-28, 78-2
- multiple tables
 - about, 29-13
 - specifying for descriptors, 29-13
- multiplicity in relationships, resolving, 7-14
- multi-processing, 11-29
- Multitable Info tab, 28-48, 29-13
- mutable mappings, 35-33
- .mwp file, 4-1, 21-2
- MySQL4
 - batch writing, 86-9
 - platform, 84-3
 - primary key restrictions, 20-15
 - schema manager type conversion, 6-3

N

- named connection pools, 84-9
- named queries
 - about, 28-10, 77-21, 96-15
 - configuring, 28-10, 77-21
 - descriptor level, 28-10
 - exclusive connections, 28-24
 - options, advanced, 28-24
 - parameter binding, 28-22
 - prepared statement caching, 28-22
 - redirect query, 96-16
 - session level, 77-21
 - using, 98-17
 - when not to use, 96-16
 - when to use, 96-16
- namespaces
 - about, 20-5, 20-22
 - configuring, 4-38
- naming service
 - coordinated cache, 91-7
 - JNDI, 91-7
 - RMI, 91-9
- native sequencing
 - IBM Informix Database, 20-19
 - Microsoft SQL Database, 20-19
 - Microsoft SQL Server, 6-6
 - non-Oracle database, 20-19
 - Oracle Database, 20-18, 20-20
 - Oracle Database SEQUENCE object, 20-19
 - Sybase Database, 6-6, 20-19
- Navigator window
 - about, 4-4, 4-9

- attribute and mapping, 4-9
- database, 4-9
- database tables in, 4-21
- descriptor, 4-9
- example, 4-9
- package, 4-9
- project, 4-9
- unsaved or changed item, 4-9
- NCHAR, 33-12
- NCLOB, 33-12
- neediness warnings. *See* troubleshooting
- nested table object-relational mappings
 - configuring, 55-1
 - understanding, 49-3
- nested unit of work, 100-9, 102-15
- new projects, 21-2
- New Reference dialog box, 4-27
- New Session button, 76-4, 76-6, 76-8
- New Sessions Configuration, 76-2
- New Table dialog box, 4-22
- newInstance method, 102-2
- no identity map, 90-4
- non-cascading write queries
 - compared to cascading, 96-14, 98-13
 - creating using dontCascadeParts ()
 - method, 96-14, 98-13
- non-deferred write
 - configuring, 102-11
 - dependent objects, 26-4
 - understanding, 26-3, 102-7, 102-25
- nonintrusive persistence, 2-18
- nonpersistent projects, 20-2
- nonrelational projects, 20-2
- nontransactional data sources, 84-1
- nontransparent indirection, 2-11
- nouns
 - DMS profiler, 11-4, 75-11
- null values
 - default, 35-12, 35-36, 85-5
 - in expressions, 97-13
 - login level, 85-5
 - mapping level, 35-12
- NVARCHAR2, 33-12

O

- object array object-relational mappings
 - configuring, 54-1
 - understanding, 49-2
- object cache, 96-36
- object cache, sessions, 75-3
- object change tracking
 - configuring, 28-71
- object identity, 75-29
 - about, 2-16, 90-3
 - cache, and, 90-3
 - using cache to preserve, 90-3
 - see also* identity map
- object indirection
 - read optimization, as, 11-19

- object level modify query
 - about, 96-13, 96-14
- object level read query
 - about, 96-11, 96-13
 - fetch groups, 96-12
 - join reading, 96-12
 - partial object query, 96-11
 - read all query, 96-11
 - read object query, 96-11
- object model
 - about, 2-16
 - generating with t ljaxb.cmd, 21-9
 - optimization, 11-8
 - Oracle TopLink requirements, 2-11
 - tutorial, 16-2
- object type converter
 - about, 12-1, 33-12
 - configuring, 35-22
- object type mappings
 - configuring, 35-22
- ObjectLevelChangeTrackingPolicy, 100-7
- object-relational descriptors
 - configuring, 30-1
 - locking policy, 28-62
 - understanding, 26-11
- object-relational mappings
 - about, 49-1
 - array, understanding, 49-2
 - configuring, 50-1
 - nested table, 49-3, 55-1
 - object array, 49-2, 54-1
 - overview, 2-15
 - reference, 49-2, 52-1, 53-1
 - structure, 49-2, 51-1
- object-relational projects
 - about, 20-6
 - sequencing, 20-5
- objects
 - cascading refresh in cache, 96-35
 - creating and registering, 102-2
 - query, 98-6
 - refreshing in cache, 96-35
 - registering and unregistering, 102-1
- ObjectTypingMapping
 - see* ObjectTypeConverter
- OC4J. *See* Oracle Containers for J2EE
- one-to-many EIS mappings
 - configuring, 63-1
 - key on source, 56-13
 - key on target, 56-15
 - understanding, 56-12
- one-to-many relational mappings
 - configuring, 42-1
 - understanding, 36-7
- one-to-one EIS mappings
 - configuring, 62-1
 - key on source, 56-9
 - key on target, 56-10
 - understanding, 56-8
- one-to-one relational mappings

- configuring, 40-1
 - expressions, 97-5
 - joining, 40-1
 - understanding, 36-5
 - with EJB, 36-6
- online help, 4-14
- Open Project button, 21-10
- opening projects, 21-10
- operators
 - boolean logic, 97-3
- optimistic locking
 - about, 26-9
 - application architecture, 2-10
 - cascading locking policy, 26-18, 28-65
 - database exception, 98-24
 - exceptions, 13-29
 - field locking policy, about, 26-20
 - rollbacks, 26-20
 - version locking policy, 26-17, 26-18, 28-65
 - with `forceUpdateToVersionField()`
 - method, 102-17
- optimistic locking policy
 - field locking, about, 26-20
 - version locking, 26-17, 26-18, 28-65
- `OptimisticLockException`, 26-18, 26-21
- optimization
 - about, 11-1
 - application bottlenecks, 11-2
 - application server, 11-30
 - batch reading, 11-17
 - batch writing, 11-15
 - CMP partial object queries, 11-17
 - code generation, 11-8
 - data access, 11-14, 11-15
 - data format, 11-15
 - database, 11-30
 - descriptors, 11-13
 - DMS profiler, 11-4, 75-11
 - existence checking, 11-13
 - fetch groups, 11-17
 - fetch size, JDBC, 11-17
 - general, 11-8
 - inheritance, 26-16
 - instantiation, 11-13
 - JDBC driver, 11-14, 11-17
 - join reading, 11-17
 - mappings, 11-13
 - named queries, 11-17
 - object model, 11-8
 - overview, 2-15
 - parameter binding, 11-15
 - partial object queries, 11-17
 - prepared statement caching, 11-15
 - profiler, 11-4, 75-11
 - optimization
 - TopLink Profiler**, 11-2
 - queries, 11-16
 - reading, 11-19
 - `ReadQuery` method `setMaxRows`, 11-18
 - schema, 11-8
 - `setMaxRows`, 11-18
 - understanding, 11-1
 - unit of work, 11-30
 - writing, 11-26
- Oracle
 - development support, 2-3
 - remote session support, 75-31
- Oracle Containers for J2EE
 - creating projects from, 7-9
 - migrating to TopLink, 7-5
- Oracle Database
 - date and timestamp mappings, 37-3
 - native sequencing, 20-20
 - Oracle Database
 - SEQUENCE** object, 20-19
 - platform, 84-3
 - schema manager type conversion, 6-3
- Oracle extensions
 - hierarchical queries, 99-6
 - Oracle Hints, 99-6
- Oracle Hints, using with TopLink queries, 99-6
- Oracle JDeveloper
 - configuring with Oracle TopLink, 5-1
 - TopLink sessions, 5-3
- Oracle TopLink
 - about, 1-1, 4-1
 - application architectures, 1-4
 - architectures, 1-1
 - deploy tool for IBM WebSphere, 10-5
 - development, 2-1, 3-1
 - features, 1-4
 - integrating with application server, 7-1, 7-2
 - mapping types, 33-1
 - optimization, 11-1
 - packaging your application, 9-1
 - public source, 12-2
 - runtime components, 3-2
 - understanding, 1-1
- Oracle TopLink Sessions Editor. *see* sessions
- Oracle TopLink Workbench
 - about, 4-1
 - Ant integration, 4-53
 - classpath, 4-2
 - creating projects, 21-1
 - development process, 4-1
 - `DRIVER_CLASSPATH`, 4-2
 - environment, 4-2
 - error messages, 14-1, 14-2
 - `JDBC_CLASSPATH`, 4-2
 - parts of, 4-3
 - preferences, 4-12
 - project, 4-1, 21-2
 - proxy, 4-13, 4-14, 21-16
 - sample, 4-3
 - table creator, creating, 6-4
 - upgrading projects, 21-10
- Oracle Virtual Private Database (VPD)
 - isolated client sessions, 75-20
 - proxy authentication, 75-21, 84-6, 86-13
- `oracle.sql.TimeStamp`, 37-3

- order
 - query keys, 37-8
 - relational mappings, 37-8
- OrderSibling, 99-7
- orion-ejb-jar.xml file
 - about, 8-9
 - entity-deployment attribute pm-name, 8-9
 - modifying for Oracle TopLink, 8-9
 - persistence-manager attribute
 - class-name, 8-10
 - persistence-manager attribute descriptor, 8-10
 - persistence-manager attribute name, 8-9
 - persistence-manager subentry
 - forpm-properties
 - cache-synchronizations, 8-11
 - customization-class, 8-11
 - db-platform-class, 8-11
 - default-mapping, 8-11
 - project-class, 8-10
 - remote-relationships, 8-11
 - session-name, 8-10
- OTN (Oracle Technology Network), 1-4, 2-3
- OTS (Object Transaction Service)
 - about, 100-3
 - unit of work, 100-3
- outer joins
 - in expressions, 98-11
 - inheritance, 28-47
- output parameter event in stored procedures, 98-22
- OX mappings
 - about, 33-27
 - extensions, simple type translator, 33-13, 33-14
 - read conversions, 33-13
 - write conversions, 33-14

P

- package names
 - generating, 4-30
 - renaming, 4-52
 - TopLink Workbench, Navigator window, 4-9
 - see also* classes
- packaging for deployment
 - about, 9-1
 - BMP applications, 9-5
 - CMP applications, 9-4
 - EJB 3.0, 9-1
 - Java applications, 9-1
 - JSP and Servlet applications, 9-2
 - Session Bean applications, 9-3
- packet time-to-live cache coordination, 91-15
- parallel
 - expressions, 97-11
 - unit of work, 100-9
- parameter binding
 - about, 11-15
 - byte arrays, 11-15
 - configuring, 23-7, 28-22, 86-9, 98-17
 - database login level, 86-9
 - descriptor level, 28-22
 - external connection pools, 11-16
 - internal connection pools, 11-16
 - J2EE, 11-16
 - named queries, 28-22
 - optimizing, 11-15
 - project level, 23-7
 - queries, 98-17
 - streams, 11-15
 - strings, 11-15
 - trouble shooting, 11-15
- parameterized batch writing
 - about, 11-15
 - setMaxBatchWritingSize(), 11-15
- parameterized expressions
 - about, 97-7
 - example, 97-8
- parameterized SQL
 - enabling on queries, 98-17
 - Oracle TopLink optimization features, 11-26
 - See also* parameter binding
- parser conflicts, XML, 7-3
- partial object reading optimization, 11-19
- passwords, encryption, 21-13
- performance optimization
 - about, 11-1
 - application bottleneck, 11-2
 - examples, 11-20
 - JConnect method isClosed, 11-14
 - using Performance Profiler, 11-2
- Performance Profiler
 - about, 11-2
 - class, 11-3
- persistence
 - about, 2-18
 - BEA WebLogic deployment, 8-13
 - by reachability, 101-6
 - components of, 2-12
 - descriptor, 8-13
 - implementation options, 2-11
 - manager, 7-4
 - OC4J, 7-5
 - projects, 20-2
 - types, 22-5
 - using a persistence layer, 2-13
- persistence manager
 - default, 7-4
 - migration, 7-4
 - restrictions, 7-4
- persistent classes
 - requirements, 2-11
 - types, 22-7
- pessimistic locking
 - about, 26-9, 26-21
 - application architecture, 2-10
 - policy, 26-21
- phantom reads, preventing, 102-30
- platforms
 - Attunity Connect database, 84-3
 - Cloudscape database, 84-3

- data source, 84-3
- database, 21-3, 23-2, 24-2, 84-3, 86-1, 87-1
- DB2 database, 84-3
- EIS, 84-4
- functions in expressions, 97-16
- HSQL database, 84-3
- Informix database, 84-3
- J2C adapter, 84-4
- JDBC drivers, 84-3
- Microsoft Access database, 84-3
- MySQL4 database, 84-3
- Oracle database, 84-3
- Oracle8 database, 84-3
- Oracle9 database, 84-3
- parser, XML, 7-2
- PointBase database, 84-3
- projects, and, 20-4
- server, 77-15
- session configuration, 77-15
- SQLAnywhere database, 84-3
- SQLServer database, 84-3
- SybasePlatform database, 84-3
- XML parser, 7-2
- see also* target platforms
- PointBase platform, 84-3
- pop-up menus. *see* context menus
- ports
 - multicast group, 91-5
 - permissions, 7-23, 7-24
- post-commit clones, avoiding, 102-30
- Potential EJB Descriptors dialog box, 21-10
- pre-allocating sequence numbers, 20-20, 23-4, 86-5, 87-2
- preferences
 - class import, 4-16
 - database, 4-17
 - EJB, 4-16
 - general, 4-13
 - help, 4-14
 - importing and exporting, 4-13
 - Oracle TopLink Workbench, 4-12
 - sessions, 4-18, 4-19
- Preferences - Class dialog, 4-16
- Preferences - EJB dialog, 4-16
- Preferences - General dialog, 4-13
- Preferences - Help dialog, 4-14
- Preferences - Mappings dialog, 4-15
- Preferences dialog, 4-13
- prepared statement caching
 - about, 11-15
 - configuring, 23-7, 28-22, 86-9, 98-17
 - database login level, 86-9
 - descriptor level, 28-22
 - external connection pools, 11-16
 - internal connection pools, 11-16
 - J2EE, 11-16
 - named queries, 28-22
 - optimizing, 11-15
 - project level, 23-7
 - queries, 98-17
 - query level, 98-17
- preserving XML documents, 32-6
- primary key
 - about, 2-16
 - composite, 36-9
 - inheritance, 26-15
 - multiple tables, 29-13
 - primkey in ejb-jar.xml file, 21-15
 - read-only settings, 35-2
 - restrictions, 20-15
 - setting, 4-26, 28-3, 29-2
 - unit of work, 100-11
 - unknown, 28-45
 - variable class relationships, 36-6, 41-3
- private relationships, 35-16
- Problems window
 - about, 4-4, 4-11
 - sample, 4-11
 - see also* error messages
- profiler
 - about, 77-10
 - development tool, 11-2
 - DMS, 11-4, 75-11
 - Oracle TopLink, 11-2, 11-3, 75-11
 - selecting, 77-11, 77-14, 77-17, 77-20
- Project - Multiple Projects tab, 77-10
- Project Status Report dialog box, 21-13
- projects
 - about, 19-1, 20-1, 20-2, 21-2
 - architecture, 20-2
 - cache type and size, 22-13, 28-35
 - comments, 22-21
 - configuring, 22-1
 - copying objects, 5-6
 - creating, 7-9, 21-1, 21-2, 21-3, 21-6
 - deployment login, and, 20-3
 - deployment overview, 2-14
 - descriptors, 19-2
 - development login, and, 20-4
 - direct access to mapped fields, 22-4
 - EIS, 20-7, 20-9, 24-1
 - errors, 14-2
 - existence checking, 22-8
 - exporting, 21-13, 21-14
 - for sessions, 77-9
 - indexed records, 20-9
 - Java, 21-3
 - JAXB, 21-6
 - locked, 5-6
 - login, 20-3, 74-1, 84-2
 - login, and, 20-3
 - mapped field access, default, 22-4
 - mapped records, 20-9
 - mapping projects, creating, 21-2
 - mappings, 19-2, 21-2
 - merging, 5-4
 - metadata, 2-20
 - method access to mapped fields, 22-4
 - model, exporting, 21-14
 - .mwp file, 4-1

- nondesoriptor classes, 4-51
- non-persistent, 20-2
- nonrelational, 20-2
- object-relational, types supported, 20-6
- OC4J, creating from, 7-9
- open, 21-10
- Oracle TopLink Workbench, 21-2
- packages, renaming, 4-52
- persistence type, 20-2, 22-7
- platforms, and, 20-4
- prior TopLink versions, 21-10
- recently opened, 21-10
- relational, 20-2, 20-6, 23-1
- renaming, 21-12
- reopening, 21-10
- saving, 21-11
- sequencing, 20-4, 20-5, 23-3, 86-4
- session login, and, 20-3
- sharing, 5-6
- status report, 21-12
- team development, 5-3
- TopLink Workbench, Navigator window, 4-9
- types of, 20-1, 21-2, 26-1
- updating from `ejb-jar.xml`, 21-16
- upgrading from 2.x or 3.x, 21-10
- writing `ejb-jar.xml` file, 21-16
- XML, 20-9, 25-1
- XML records, 20-9
- `projects.xml` file
 - about, 8-2
 - EJB 3.0, 8-3, 8-5
 - schema, 8-2
 - XSD file, 8-2
- propagation mode, cache, 91-2
- proxies. *see* wrapper policy
- proxy authentication
 - about, 84-5
 - applications, 84-5
 - Oracle Virtual Private Database, 75-21, 84-6, 86-13
 - session events, 75-6
 - use cases, 84-5
- proxy indirection
 - about, 33-8
 - example, 35-9
 - restrictions, 33-8
- proxy settings, preferences, 4-13
- public source code, 12-2

Q

- qualified names, database tables, 4-23, 4-24
- queries
 - about, 96-1
 - application development process, 2-13, 74-2
 - building, 96-6
 - cache, 96-29
 - `Call` queries, 96-3, 96-16
 - cascading, 96-14, 98-13
 - concepts, 96-2

- conforming, 102-8
- database queries, 96-10, 98-4
- `DatabaseQuery`, 96-3
- descriptor query manager, 96-3
- EIS interactions, 96-6
- EJB finders, 96-23
- EJB QL query language, 96-5
- `ejb-jar.xml` file, 28-10, 29-6, 31-6
- executing, 96-7
- expressions, 96-3
- fetch groups, 96-12
- flashback queries, 81-1, 96-18
- hierarchical queries, Oracle extensions, 96-18
- hints, Oracle extensions, 96-18
- historical, 96-20, 99-2
- interface and inheritance queries, 96-22
- joining, 96-12
- languages, about, 96-4
- named queries, 96-15
- object level modify query, 96-13, 96-14
- object level read query, 96-11, 96-13
- on inheritance hierarchies, 99-4
- on interfaces, 99-4
- optimizing, 11-16
- Oracle database features, 99-5
- Oracle extensions, 96-17, 96-18
- parameter binding, 98-17
- partial object query, 96-11
- performance, 11-16
- prepared statement caching, 98-17
- query keys, 96-4
- read all query, 96-11
- read object query, 96-11
- redirect queries, 96-19
- remote sessions, 75-31
- report, 11-21, 96-8
- report query, 96-15
- results, 96-8
- returning policy, 96-17
- session queries, 96-9, 98-1
- SQL query language, 96-4
- stored functions, Oracle extensions, 96-18
- subqueries, 97-10
- summary queries, 96-3
- timeout, 28-24, 28-26, 98-10
- types, 96-1
- `UpdateAll`, 96-14
- XML query language, 96-5
- Queries tab, 28-26, 28-28
- query by example, 98-6
- query cache, 96-36
- Query Exception, 13-30
- query keys
 - about, 28-29, 28-33, 96-4
 - adding, 28-31, 28-34
 - and expressions, 97-9
 - creating, 28-31, 28-34
 - direct mappings, 28-30
 - generating, 28-30
 - in expressions, 97-9

- interface descriptors, 28-31, 28-33
- Java implementation, 28-31
- modifying, 28-30
- order, 37-8
- relationship mappings, 28-30
- specifying, 28-31, 28-34
- unmapped attributes, 28-30, 28-31
- Query Keys tab, 28-31
- query object query. *See* DatabaseQuery
- query objects
 - batch reading, 98-10
 - cache expiration, 99-22
 - caching results, 96-36, 99-21
 - examples, 98-5
 - ordering for ReadAll queries, 98-8
 - report query, 11-21
 - specifying collection class, 98-9
- query results
 - about, 96-8
 - caching, 96-36
 - collections, 96-8
 - cursors, 96-8
 - reports, 96-8
 - streams, 96-8
- query timeout example, 98-10
- QueryManager
 - about, 26-8
 - joining expressions, 99-4
- QuerySequence, 86-8

R

- read access
 - providing in sessions, 75-15
- read all operation, 98-2
- read conversions
 - simple type translator, 33-13
- read only
 - descriptors, 28-4
 - files, 5-6
 - mappings, 35-2, 35-3
- read operation, 98-2
- read optimization
 - about, 11-19
 - batch reading, 11-19
 - fetch groups, 28-77
 - joining, 11-19
 - lazy attribute loading, 28-77
 - object indirection, 11-19
 - partial object reading, 11-19
 - report query, 11-20
 - soft cache weak identity map, 11-19
 - unit of work, 11-19
 - weak identity map, 11-19
- read queries
 - cascading refresh of identity maps, 96-35
 - identity map cache refresh, 96-35
 - refreshing identity maps, 96-35
- ReadAll finders
 - executing, 99-10
- ReadAll queries
 - ordering in query objects, 98-8
- readAllObjects()
 - about, 29-8
 - example, 98-2
- reading
 - ejb-jar.xml, 21-16
 - just-in-time reading, 33-8
 - whole XML documents, 39-1
- reading through the write connection, 102-27
- read-locking, 26-9
- readObject()
 - example, 98-2
- recently opened projects, 21-10
- redirect queries
 - about, 96-19, 99-1
 - creating, 99-1
 - finders, 99-13
 - using, 99-12
- reference key field, 36-11, 36-12
- reference object-relational mappings
 - configuring, 52-1, 53-1
 - understanding, 49-2
- ReferenceMapping class, 49-2
- references
 - database tables, 4-27, 4-28
 - foreign keys, 14-12, 14-21
- Refresh from Database button, 4-24
- refresh policy
 - cache, 90-8
 - EJB finders, 90-8
- refreshing
 - cache, 28-27
 - classes, 4-50
 - refreshObject(), 98-2
 - remote sessions, 75-31
 - sessions, 78-5
- registering objects, 102-1
- registerNewObject method, 102-2
- reimporting schemas, 4-36
- relation tables
 - about, 36-9
 - many-to-many mappings, 43-1
 - mappings, 43-2
- relational descriptors
 - associated table, 29-2
 - configuring, 29-1
 - locking policy, 28-62
 - understanding, 26-11
- relational mappings
 - about, 36-1
 - aggregate collection, 36-10, 44-1
 - aggregate object, 36-12, 47-1
 - configuring, 37-1
 - converters, 36-3
 - direct collection, 36-11, 45-1
 - direct map, 36-12, 46-1
 - direct-to-field, 36-4, 38-1
 - direct-to-XMLType, 36-4, 39-1
 - many-to-many, 36-8, 43-1

- one-to-many, 36-7, 42-1
- one-to-one, 36-5, 40-1
- options, 37-1
- order, 37-8
- transformation, 36-3, 36-15, 48-1
- variable one-to-one, 36-6, 41-1
- relational projects
 - about, 20-2
 - configuring, 23-1
 - object-relational databases, 20-6
 - relational databases, 20-6
 - understanding, 20-6
- relationships
 - about, 2-17
 - bidirectional, 4-31, 35-34, 36-5
 - expressions, 97-5
 - in `ejb-jar.xml` file, 21-16
 - unexpected multiplicity, 7-14
- relative locations
 - about, 22-2
 - classpath, 22-3
- Remote Command Manager Exception, 13-77
- remote connection using RMI, example, 76-10
- remote sessions
 - about, 75-2, 75-29
 - application layer, 75-30
 - creating, 76-10
 - limitations of, 75-31
 - securing access, 75-31
 - server layer, 75-31
 - transport layer, 75-30
 - unit of work, 75-32
- Remove Class button, 4-51
- Remove Table button, 4-24
- removing sessions from brokers, 82-1
- Rename dialog, 4-24
- renaming
 - packages, 4-52
 - projects, 21-10, 21-12
- reopening projects, 21-10
- report query
 - about, 96-15
 - query objects, 11-21
 - read optimization, and, 11-20
 - using, 11-21, 96-8
- reports
 - project status, 21-12
 - query results, 96-8
 - see also* status reports
- resuming unit of work
 - after commit, 102-14
 - after commit failure, 102-14
- returning policy
 - configuring, 28-65
 - SQLCall, 96-17
- Returning tab, 28-66
- RMI
 - coordinated cache, 93-1
 - naming service, 91-9
 - remote session support, 75-31

- rollback
 - and Java Transaction API, 100-11
 - overview, 100-11
 - with optimistic locking, 26-20
- root class
 - about, 26-12
 - inheritance, 26-12
- root EIS descriptor, 27-5
- root element, descriptor, 32-5
- runtime
 - acquiring sessions, 75-5
 - components, 3-2
 - services, configuring sessions, 77-14
 - troubleshooting, 15-14

S

- Save All button, 21-11
- Save button, 21-11
- saving projects, 22-2
- schema context
 - descriptors without, 32-3
 - EIS descriptors, 31-2
 - XML descriptors, 32-2
- Schema Document Info tab, 4-36, 4-38
- schema manager
 - about, 6-1
 - automatic table creation, 6-6
 - creating a table creator, 6-4
 - DB2, 6-3
 - default table generator, 6-4
 - Java table creator, 6-4
 - MS Access, 6-3
 - MySQL, 6-3
 - Oracle, 6-3
 - Oracle TopLink Workbench table creator, 6-4
 - sequencing, 6-3, 6-6
 - Sybase, 6-3
 - table creator, 6-2, 6-4, 6-6
 - table definition, 6-2
 - type conversion, 6-3
 - usage, 6-1
- Schema Structure tab, 4-34
- schemas
 - about, 2-16
 - context for EIS descriptors, 31-2
 - context for XML descriptor, 32-2
 - data storage, 2-16
 - database, 4-22
 - default root for EIS descriptors, 31-4
 - details, 4-34
 - document information, 4-36, 4-38
 - errors, 14-20
 - importing, 4-35
 - optimizing, 11-8
 - properties, 4-36
 - reimporting, 4-36
 - schema manager, 6-1
 - structure, 4-34
 - XML schemas, 4-34

- SCM. *see* source control management
- scripts
 - SQL, generating, 4-29
 - see also* SQL
- scrollable cursor
 - traversing, 99-16
 - using for ReadAllQuery, 99-16
- SDK
 - Data Store Exception, 13-71
 - Descriptor Exception, 13-75
- security
 - cache coordination, permissions, 7-23
 - data source access, permissions, 7-24
 - disabling doPrivileged(), 7-25
 - doPrivileged(), 7-4
 - EJB, permissions, 7-24
 - enabling doPrivileged(), 7-25
 - J2EE application, permissions, 7-24
 - java.util.logging, permissions, 7-24
 - loading project.xml, permissions, 7-23
 - loading sessions.xml, permissions, 7-23
 - logging, permissions, 7-24
 - password encryption, 21-13
 - permissions by feature, 7-23
 - permissions when doPrivileged()
 - disabled, 7-24
 - port permissions, 7-23, 7-24
 - SecurityManager integration, 7-4
 - system properties, permissions, 7-23
 - understanding permissions, 7-22
 - with BEA WebLogic, 7-20
- Select Classes dialog box, 4-50
- SelectedFieldsLockingPolicy, 26-21
- sensors, DMS profiler, 11-4, 75-11
- SEQ_COUNT column in sequence table, 20-20
- sequence connection pools, 84-8
- sequence numbers, write optimization
 - features, 11-26
- SEQUENCE objects in Oracle native
 - sequencing, 20-19
- sequencing
 - about, 20-14
 - BEA WebLogic, 20-21
 - BEA WebLogic single column sequence
 - table, 20-17, 23-3, 86-4
 - configuring, 20-15
 - connection pools, 84-8
 - default, 86-5, 86-6
 - DefaultSequence class, 86-6
 - descriptors, 26-9, 29-3
 - entity beans, 20-21
 - IBM WebSphere, 20-21
 - isolated client sessions, 75-24
 - Java configuration, 20-15
 - Microsoft SQL Server, 6-6
 - native, 20-18, 20-19, 20-20
 - non-Oracle database native, 20-19
 - Oracle Database native, 20-18, 20-20
 - Oracle TopLink Workbench configuration, 20-15
 - overriding default, 86-6
 - platform default, 86-5
 - preallocation, 11-28, 20-20
 - QuerySequence class, 86-8
 - schema manager, 6-6
 - SEQ_COUNT, 20-20
 - sequence type, configuring, 23-3, 86-4
 - sessions, and, 75-13
 - stored procedures, 20-18, 86-7
 - Sybase Database, 6-6
 - table, 20-16
 - table, default column and table names, 20-17
 - unary table, 20-17
 - with stored procedures, 86-7
- serialization
 - descriptor exceptions, 102-13
 - EJB 1.x, 33-9
 - EJB 2.x, 33-9
 - EJB 3.0, 33-9
 - indirection, 33-9
 - merging into session cache with unit of
 - work, 102-12
- serialized object converter
 - about, 12-2, 33-10
 - configuring, 35-18
- serialized object mappings, 35-19
- SerializedObjectMapping. *see*
 - SerializedObjectConverter
- server layer, 75-31
- server platform
 - external transaction controller, 77-14
 - runtime services, 77-14
 - session configuration, 77-14
 - session event listener, 77-17
- Server Platform tab, 77-15
- server sessions
 - about, 75-1, 75-13, 75-15
 - cache, 75-32
 - connection options, 75-18
- service channel, coordinated cache, 91-3
- session accessor
 - about, 2-30
 - managing dependent objects with, 2-30
- session beans
 - about, 2-25
 - deploying, 8-8, 9-3, 10-1
 - model, 2-23, 2-25
 - remote session support for, 75-31
- session brokers
 - about, 75-2, 75-25
 - adding sessions to, 82-1
 - alternatives, 75-28
 - architecture, 75-26
 - configuration, 82-1
 - limitations of, 75-27
 - renaming, 82-1
 - two-phase commits, 75-27
 - two-stage commits, 75-27
- session configuration file
 - about, 76-1
 - loading alternative, 78-4

- preferences, 4-18, 4-19
 - see also* sessions.xml file
- session customizer, 75-4
- Session Event Manager, 75-5
- session events
 - example, 75-5, 75-6
 - isolated sessions, and, 75-5
 - manager, 75-5
 - proxy authentication, and, 75-6
- Session Loader Exceptions, 13-56
- Session Manager
 - about, 78-1, 78-2
 - acquiring, 78-2, 78-3
 - defaults, 78-3
 - destroying sessions, 78-10
 - J2EE defaults, 78-3
 - sessions, acquiring, 75-5
 - storing sessions, 78-10
- sessions
 - about, 2-12, 74-1, 75-1, 75-2, 76-1
 - acquiring at runtime, 75-5, 78-1
 - adding to session brokers, 82-1
 - additional mapping projects, 77-9
 - API, 75-33
 - application server logging, 75-9
 - architecture, 75-2
 - cache, 75-32, 90-2
 - client, 75-13
 - configuring, 77-1, 77-17
 - connection policy, 77-19
 - creating, 2-21, 76-1, 76-4, 76-6, 76-8, 88-1
 - customization, 75-4
 - database sessions, 75-28, 76-8
 - destroying in Session Manager, 78-10
 - errors, 14-21
 - event listeners, 75-7, 77-17
 - events, 75-6
 - external transaction controller, 77-14, 100-2
 - historical client sessions, 75-24
 - in sessions.xml file, 75-4
 - isolated client sessions, 75-19, 80-1
 - loading with alternative class loader, 78-4
 - logging, 75-7, 75-9, 75-10, 77-4, 77-5
 - logging into, 78-9
 - logging out of, 78-10
 - logins, 77-4
 - management of, 78-1
 - metadata, about, 2-21
 - multiple sessions, 75-28, 78-2
 - named queries, 77-21
 - object cache, 75-3
 - optimizing, 11-13
 - preferences, 4-18, 4-19
 - queries, 96-7, 96-9, 98-1
 - refreshing, 78-5
 - registering descriptors, 75-12, 77-2, 77-3, 77-10
 - remote sessions, 75-29, 76-10
 - sequencing, about, 75-13
 - server, 75-13
 - server platform, 77-14, 77-15
 - SQL and messages, 75-10
 - storing in Session Manager, 78-10
 - three-tier architecture, 75-14
 - transformation mappings, 36-15
 - types, 75-1
 - unit of work, 75-1, 75-19
- sessions.xml file
 - about, 8-3
 - acquiring, 75-5
 - CMP applications, 8-4, 75-4
 - configuring with JDeveloper, 5-3
 - creating, 76-1
 - default location, 75-5
 - EJB 3.0, 8-5
 - loading alternative configuration file, 78-4
 - non-CMP applications, 8-4
 - schema, 8-4
 - sessions, 75-4
 - XSD file, 8-4
- setAdditionalJoinExpression(), 99-5
- setenv.cmd file, 4-2
- setenv.sh file, 4-2
- setMaxBatchWritingSize(), 11-15
- setMultipleTableJoinExpression(), 99-5
- setShouldPerformDeletesFirst(), 102-16
- Settings tab, 28-26, 28-28
- setValue() method, 33-7
- shared library
 - setting for BEA WebLogic, 7-15
- shared library for BEA WebLogic, 7-15
- simple type translators
 - about, 12-2, 33-12
 - configuring, 35-23
 - in Java, 35-25
 - read conversions, 33-13
 - write conversions, 33-14
- soft cache weak identity map
 - about, 90-4
 - when to use, 90-5
- sorting, in memory, 35-26
- source code, public, 12-2
- source control management
 - projects, 5-3
 - with Oracle TopLink Workbench, 5-3
 - see also* team development
- source table, reference, 4-28
- source.jar, 12-2
- splash screen, 4-13
- SQL
 - call, 98-18
 - comparing with expressions, 97-1
 - custom queries for basic persistence, per descriptor, 29-6
 - EJBQLCall, 98-23
 - generating from database tables, 4-29
 - parameter binding, 11-15
 - parameterized, 11-26, 98-17
 - prepared statement caching, 11-15
 - queries, 96-4
 - scripts with binding arguments, 29-6

- SQLCall, 98-18
 - StoredFunctionCall, 98-22
 - StoredProcedureCall, 98-20
 - unit of work, 102-15
- SQL Creation Script dialog box, 4-29
- SQL DISTINCT, 11-24
- SQL Exception, 13-27
- SQLAnywhere platform, 84-3
- SQLCall
 - about, 96-17
 - binding input parameters, 98-19
 - input parameters, 98-19
 - input-output parameters, specifying, 98-20
 - output parameters, 98-19
 - Returning Policy, 96-17
 - using, 98-18
- SQLServer platform, 84-3
- stages of development with Oracle TopLink, 2-1
- stale data
 - cache, 90-6, 90-7
 - coordination, cache, 90-7
 - invalidating the cache, 90-7
 - locking policy, and, 90-6
 - per-class cache configuration, 90-7
 - per-query cache refresh, 90-7
- StartWith, 99-7
- stateful
 - beans, 2-25
 - comparing with stateless, 2-25
- stateless
 - comparing with stateful, 2-25
- static attributes, 4-45, 4-48
- static fetch groups, querying with, 99-3
- status report, generating, 21-12
- stored functions
 - about, 96-18
 - using, 98-22
- stored procedures
 - output parameter event, 98-22
 - queries, 98-20
 - sequencing, and, 20-18, 86-7
- streams
 - as query results, 96-8
 - cursor, 75-32, 99-18
- string bindings, CLOB mappings, 34-4
- structure object-relational mappings
 - configuring, 51-1
 - understanding, 49-2
- StructureMapping class, 49-2
- subqueries
 - multiple expressions, 97-10
 - subselects in expressions, 97-10
- subselects, in expressions, 97-10
- summary queries, 96-3
- superclass, 4-41
- Sybase
 - database schema manager type conversion, 6-3
 - native sequencing, 20-19
 - platform, 84-3
- Synchronization Exceptions, 13-70

- synchronous change propagation, 91-2
- system properties
 - oracle.j2ee.security.usedoprivileged, 7-25
 - oracle.j2ee.toplink.security.usedoprivileged, 7-25
 - toplink.cts.collection.checkParameters, 7-25
 - toplink.xml.platform, 7-3

T

- table creator
 - about, 6-2
 - creating, 6-4
 - using, 6-6
- Table Creator dialog, 6-4
- table generation. *see* automatic table generation
- table sequence
 - about, 20-16
 - default column and table names, 20-17
- table-class relationships, 16-6
- tables
 - adding database, 4-22
 - associating with relational descriptors, 29-2
 - database, 4-21
 - defining schema, 6-2
 - errors, 14-15
 - generating, automatic, 33-4
 - import filter, 4-23
 - mapping to descriptors, 28-3, 29-2
 - merging files, 5-5
 - multiple, 29-13
 - primary key, 4-26
 - references, 37-8
 - relation tables for mappings, 43-2
 - TableDefinition class, 6-2
 - see also* database tables
- target foreign keys
 - about, 36-6, 37-9
 - configuring, 37-9
- target platforms
 - about, 2-5
 - choosing, 2-3
- target tables
 - in direct collection mappings, 45-1
 - reference, 4-28
- team development, 5-3
- technical support, 2-3
- three-tier architecture
 - about, 1-4, 2-21
 - authentication, 84-5
 - migrating to scalable architecture, 75-29
 - overview, 2-21
 - sessions, 75-14
- timestamp support
 - about, 37-3
 - direct to field mappings, 37-3
 - Oracle Database, 37-3
 - TIMESTAMP timezone, 37-4
- TimestampLockingPolicy, 26-18
- timezone, with TIMESTAMP, 37-4

- tljxb.cmd file, 21-9
- toolbars, 4-4, 4-6
- topic name, 92-1
- TopLink. *see* Oracle TopLink
- TopLink expressions. *see* expressions
- TopLink profiler
 - about, 11-2
 - selecting, 77-11, 77-14, 77-17, 77-20
- TopLink Workbench. *see* Oracle TopLink Workbench
- toplink-ejb-jar.xml file, 8-6, 19-2
- toplink-ejb.xml file, 8-8
- Transaction Exception, 13-79
- transactional data sources, 84-1
- transactions
 - client-controlled, 102-24
 - CMP, 100-3, 102-24, 102-25
 - container-controlled, 102-24
 - demarcation, 100-2
 - external transaction controller, 77-14, 100-2
 - external, integrating, 100-2, 102-20
 - isolated client sessions, 75-24
 - isolation, 100-2, 100-4
 - JTA, 100-3
 - JTS, 100-3
 - local, 102-25
 - local, CMP, 102-25
 - OTS, 100-3
 - overview, 2-13, 74-3, 100-1
 - see also* unit of work
- transformation EIS mappings
 - configuring, 64-1
 - understanding, 56-17
- transformation mappings
 - about, 33-14
 - attribute transformation, 35-29, 35-31
 - mutable, 35-33
- transformation relational mappings
 - configuring, 48-1
 - understanding, 36-15
- transformation XML mappings
 - configuring, 73-1
 - understanding, 65-31
- transient attributes, 4-45, 4-48
- transparent indirection
 - about, 33-8
 - persistent class requirements, 2-11
- transport layer, 75-30
- troubleshooting
 - BEA WebLogic deployment, 15-3
 - data sources, 14-23
 - deploy tool, 10-6
 - deployment, 15-1
 - ejbc, 10-3
 - IBM WebSphere deployment, 15-12
 - migration from OC4J persistence, 7-13
 - Oracle TopLink Workbench, 14-1
 - unit of work, 102-30, 102-34
- tutorial
 - about, 16-1
 - database schema, 16-4
 - planning, 16-1
 - two-phase commits, 75-27
 - two-stage commits, 75-27
 - two-tier architecture
 - about, 1-6, 2-23
 - authentication, 84-5
 - understanding, 2-23
 - type conversion
 - automatic, 37-2
 - NCHAR, 33-12
 - NCLOB, 33-12
 - NVARCHAR2, 33-12
 - oracle.sql.TimeStamp, 37-3
 - schema manager, 6-3
 - String to TIMESTAMP, 33-12
 - TIMESTAMP to String, 33-12
 - type conversion converter
 - about, 12-2, 33-11
 - configuring, 35-20
 - provided by direct-to-field mappings, 37-3
 - TypeConversionMapping
 - see* TypeConversionConverter
 - types of mappings, 33-1
 - typesafe enumeration, in EIS mappings, 56-3

U

- unary table sequence
 - about, 20-17
 - BEA WebLogic single column sequence
 - table, 20-17, 23-3, 86-4
- undeployment, 10-4
- unexpected relationship multiplicity, 7-14
- unidirectional relationships, 36-2
- unit of work
 - about, 100-1, 100-2, 100-4
 - acquiring, 101-1
 - API, 100-11
 - architecture, 100-1
 - auditing, 102-20
 - benefits of, 100-4
 - cache, 90-2
 - change policy, 100-6
 - clones, 100-9
 - CMP integration, 100-3
 - commit and Java Transaction API, 100-10
 - commit, writing changes before, 102-7, 102-11, 102-23
 - conform results of in-memory query, 96-31
 - creating objects, 101-2
 - deleting objects, 101-7
 - example, 100-6
 - external transaction controller, 77-14, 100-2
 - external transactions, 100-2, 102-20
 - integrating with CMP, 102-24
 - isolation, 100-4
 - JTA integration, 100-3
 - JTS integration, 100-3
 - life cycle, 100-5
 - modifying objects, 101-2

- mutable mappings, 35-33
- nested, 100-9, 102-15
- `newInstance` method, 102-2
- optimization, 11-30, 100-11
- OTS integration, 100-3
- parallel, 100-9, 102-15
- pre-commit validation, 102-34
- primary keys, 100-11
- proxy indirection, 33-8
- queries, 100-12
- read optimization, 11-19
- read-only classes, 102-6, 102-7
- `registerNewObject` method, 102-2
- remote sessions, 75-32
- resuming, 102-14
- reverting, 102-14
- rollback, 100-11
- sessions, 75-1, 75-19
- transaction demarcation, 100-2
- transactions, 100-4
 - unit of work
 - demarcation, 100-2
- troubleshooting, 102-30
- validating objects, 102-34
- with custom SQL, 102-15
- write optimization, 11-26
- writing changes before commit, about, 102-7
- writing changes before commit, and external
 - transaction exceptions, 102-23
- writing changes before commit, and external
 - transaction timeouts, 102-23
- writing changes before commit, as alternative to
 - conforming, 102-11
- unmapping, 34-6
- unregistering objects, 102-1
- unsaved items, displaying in TopLink Workbench
 - Navigator window, 4-9
- update
 - operation, 98-3, 98-4
 - projects from prior versions, 21-10
- `UpdateAll` query, 96-14
- `updateObject()`, 29-8
- Use XML Schema "type" attribute,
 - configuring, 35-23
- `useBatchWriting()`, 86-11
- user-defined functions, in expressions, 97-16

V

- validating
 - descriptors, 27-6
 - JAXB, 20-14
 - projects, 21-12
 - unit of work, 102-34
- Validation Exceptions, 13-42
- Value Converter tab, 46-4
- value holders
 - about, 33-6
 - `ValueHolder` class, 33-6
- `ValueHolderInterface` class, 2-11, 33-7, 36-9

- variable one-to-one relational mappings
 - class indicator, 41-1
 - configuring, 41-1
 - primary key, unique, 41-3
 - understanding, 36-6
- `VariableOneToOneMapping` class, 36-6
- Varray in Oracle database. *see* array mappings
- version control, 5-6
- Version Control Assistance dialog box, 5-6
- `VersionLockingPolicy`, 26-18
- volatile attributes, 4-45, 4-48
- VPD. *see* Oracle Virtual Private Database

W

- warning icon, 4-10, 4-11
- weak identity map, 90-3
- web browser, specifying, 4-14
- web services architecture, 1-5, 2-36
- WebLogic. *see* BEA WebLogic
- `weblogic-ejb-jar.xml` file
 - described, 8-5
 - modifying for Oracle TopLink, 8-13
 - unsupported tags, 8-14
- WebSphere. *see* IBM WebSphere
- welcome screen, 4-14
- wildcard, 66-3
- wrapper policy
 - about, 28-75
 - implementing in Java, 28-76
- write
 - conversions, simple type translator, 33-14
 - write all operation, 98-3
- write query
 - disabling identity map cache, 98-14
 - non-cascading, 98-13
 - objects, 98-12
 - overview, 98-12
- write-locking, 26-9
- writing
 - batch, 11-26, 86-11
 - `ejb-jar.xml` file, 21-16
 - optimization, 11-26
 - sessions write access, 75-16

X

- `xdb.jar` file, 4-3
- XML
 - descriptor, schema context, 32-2
 - generating deployment, 8-3
 - mappings, concepts, 65-2
 - preserving data, 32-6
 - projects, 21-3
 - query language, 96-5
 - reading whole documents, 39-1
 - records, 56-3
 - schemas, 4-34
- XML Conversion Exception, 13-80
- XML Data Store Exceptions, 13-64

- XML descriptors
 - configuring, 32-1
 - schema context, 32-2
 - understanding, 26-12
- XML mappings
 - about, 65-1
 - any collection, 65-29, 72-1
 - any object, 65-27, 71-1
 - any type support, 65-4
 - composite collection, 65-25, 70-1
 - composite direct collection, 65-14, 68-1
 - composite object, 65-21, 69-1
 - concepts, 65-2
 - configuring, 66-1
 - default conversion pairs, customizing, 65-11
 - direct, 65-5, 67-1
 - extensions, 65-4
 - jaxb:class support, 65-4
 - list support, 65-3
 - reference descriptor, configuring, 66-2
 - transformation, 65-31, 73-1
 - types of, 65-1
 - union support, 65-3
 - xsd:list, 65-3
 - xsd:union, 65-3
- XML parser platform
 - about, 7-2
 - configuring, 7-3
 - creating, 7-3
 - Crimson, 7-3
 - default, 7-3
 - limitations, 7-3
 - parser conflicts, 7-3
 - toplink.xml.platform system property, 7-3
 - used by, application server, 7-2
 - used by, Oracle TopLink, 7-2
- XML projects
 - configuring, 25-1
 - JAXB support, 20-10
 - sequencing, 20-5
 - understanding, 20-9
- XML queries, 96-5
- XML schema
 - jaxb:class, and EIS mappings, 56-3
 - jaxb:class, and XML mappings, 65-4
 - jaxb:class, understanding, 33-20
 - type, 35-23
 - xs:any, understanding, 65-4
 - xs:anyType, understanding, 65-4
 - xsd:list, understanding, 33-17
 - xsd:union, understanding, 33-17
 - see also* schemas
- XML Type functions, 97-4
- XPath
 - by name, 33-16
 - by position, 33-15
 - mapping Java attributes, 35-10
 - support in OX mappings, 33-15
 - support in XML mappings, 56-3, 65-3
- XSD file

- projects.xml file, 8-2
- sessions.xml file, 8-4

Z

- zero-argument constructors
 - editing, 4-16

