Oracle® Coherence

Tutorial for Oracle Coherence

Release 3.6

E15831-01

July 2010

ORACLE®

Oracle Coherence Tutorial for Oracle Coherence, Release 3.6

E15831-01

Primary Author: Thomas Pfaeffle

Contributing Author: Noah Arliss, Mark Falco, Alex Gleyzer, Gene Gleyzer, David Guy, Charlie Helin, Adam Leftik, Tim Middleton, Brian Oliver, Patrick Peralta, Cameron Purdy

Contents

4 Working with Complex Objects

5 Loading Data Into a Cache

6 Simplifying Cache Calls and Aggregations

7 Listening for Changes and Modifying Data

8 Using JPA with Coherence

9 Interacting with the Cache and the Database

10 Working with Security

11 Caching Sessions with Coherence and WebLogic Server

A Coherence Client Application Commands

Index

List of Examples

List of Tables

List of Figures

Preface

Oracle Coherence is an in-memory data grid solution that enables organizations to predictably scale mission-critical applications by providing fast access to frequently used data. Data grid software is a middleware that reliably manages data objects in memory across many servers. By automatically and dynamically partitioning data, Oracle Coherence enables continuous data availability and transactional integrity, even in the event of a server failure. Oracle Coherence provides organizations with a robust, scale-out data abstraction layer.

Developers can easily take advantage of the features of Coherence using the standard Java collections API to access and modify data, and use the standard JavaBean event model to receive data change notifications.

Audience

This book is targeted at software developers, architects, and administrators. It provides examples of developing applications for the Oracle Coherence data grid

Documentation Accessibility

Our goal is to make Oracle products, services, and supporting documentation accessible to all users, including users that are disabled. 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.

Access to Oracle Support

Oracle customers have access to electronic support through My Oracle Support. For information, visit http://www.oracle.com/support/contact.html or visit http://www.oracle.com/accessibility/support.html if you are hearing impaired.

Related Documents

For more information, see the following documents in the Oracle Coherence library:

- *Getting Started with Oracle Coherence*
- *Developer's Guide for Oracle Coherence*
- *Client Guide for Oracle Coherence*
- *User's Guide for Oracle Coherence*Web*
- *Integration Guide for Oracle Coherence*

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.
italic	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.

1

Installing Coherence and JDeveloper

This chapter describes how to install and set up your environment for running Oracle JDeveloper Studio Edition 11g and Oracle Coherence Release 3.6. This chapter contains the following sections:

- Downloading and Installing Coherence
- Downloading and Installing JDeveloper
- Testing a Coherence Installation

This chapter assumes that you have the privileges to install software and set system environment variables as the oracle user, an understanding of how to use a terminal window, including setting environment variables, and creating and moving between directories. It also assumes that you have a working installation of Java SE (JDK) version 1.6 or higher.

This chapter also assumes that you are running one of the following Microsoft Windows operating systems: XP, Vista, 2000, 2003, or 2008.

Downloading and Installing Coherence

To download and install Oracle Coherence:

1. Download Oracle Coherence to your desktop.

   Coherence (Java edition) ships as a single zip file, typically called `coherence-<version>.zip`. You can obtain Coherence from the following URL:

   `http://www.oracle.com/technology/products/coherence/index.html`

2. Extract the contents of the zip file to a directory named `C:\oracle\product`.

   The zip file contains the coherence directory with these subdirectories:

   - `bin`, which contains command scripts
   - `doc`, which contains the product documentation
   - `lib`, which contains the required library files

Downloading and Installing JDeveloper

To download and install Oracle JDeveloper Studio Edition 11g:

1. Download Oracle JDeveloper Studio.

   You can obtain JDeveloper Studio 11g from the following URL:

```
http://www.oracle.com/technology/products/jdev/index.html
```

2. Run the JDeveloper Studio installer.

   If you are asked for a **User Role** when JDeveloper starts, select **Default Role** to enable all technologies.

   Follow the prompts in the installation screens. If the installer asks for a **Middleware Home**, enter `C:\oracle\product` if it does not exist.

Testing a Coherence Installation

In this exercise, you test whether your Coherence installation can cluster Java processes. This ensures that Coherence-based applications that ship with Coherence will run as expected. If Coherence is not capable of clustering on a single machine, then you must reconfigure your network and firewall settings.

This exercise assumes that you have installed Oracle Coherence (Java Edition) Release 3.6 (See "Downloading and Installing Coherence" on page 1-1).

Coherence uses a variety of network addresses and ports to enable communication between clustered processes. If these addresses and/or ports are unavailable due to other applications using them, or because of a firewall, then Coherence may be unreliable, may fail to cluster, or may not work at all. By default, Coherence assumes that the network addresses and ports listed in Table 1–1 are available:

Table 1–1 Network Addresses and Ports Used by Coherence

Address / Port / Type	Purpose
`224.`*version* / *version* / `Multicast`	Cluster member discovery and broadcast. The variable *version* represents the release version of Coherence. For example, the multicast address for the 3.6.0.0 release would be `224.3.6.0`. The port number also reflects the release version. For example, for the 3.6.0.0 release, the port number would be `36000`. Designing the address in this way ensures that different versions of Coherence do not inadvertently cluster with each other by default.
`localhost` / `8088+` / `Unicast`	Inter-process communication between cluster members. (`localhost` is the local IP address and not the loop back address.)

Coherence ships with two simple command-line (shell-based) applications that can be used to determine whether Coherence operates correctly.

- The "cache server," is a simple application that hosts and manages data on behalf of other applications in a cluster.
- The "cache client," is a simple application that enables a developer to access, process, and update cached data within a cluster. It also provides information about the cluster. By executing these applications on either a single host or several hosts, you can determine whether Coherence is operating correctly locally or across a network.

  When an application uses Coherence out-of-the-box, objects placed into Coherence caches are typically stored and managed in-process within the application. However, to increase the availability of the objects, Coherence may manage objects in-memory but out of the application process. This allows objects to survive possible application outages (either deliberate or accidental). To manage objects in this way, Coherence uses "cache servers". The purpose of a Coherence cache server

is to manage application state in a cluster outside the application process. It is much like a database server, but without the requirement for storage.

1. Configure and Run the Sample Cache Server Application
2. Configure and Run the Sample Cache Client Application
3. Exercise the Sample Cache Client Application
4. Troubleshooting Cache Server Clustering

Configure and Run the Sample Cache Server Application

Follow these steps to set up and run the sample cache server application.

1. Open a terminal window and verify that the `PATH` environment variable is set to include the Java JDK (for example, `\oracle\product\jdk160_14_R27.6.5-32\bin`). If the `PATH` environment variable does not include the JDK `\bin` directory, then set it as follows:

   a. Set the `JAVA_HOME` environment variable to the base of the JDK installation.

   ```
   set JAVA_HOME=\oracle\product\jdk160_14_R27.6.5-32
   ```

   b. Include `JAVA_HOME\bin` in the `PATH` environment variable.

   ```
   set PATH=%JAVA_HOME%\bin;%PATH%
   ```

2. Navigate to the directory where Coherence is installed. Edit `cache-server.cmd` and set the `COHERENCE_HOME` variable to point to the Coherence installation directory.

   ```
   cd C:\oracle\product\coherence\bin
   ```

   In `cache-server.cmd`, set the `COHERENCE_HOME` environment variable:

   ```
   COHERENCE_HOME=C:\oracle\product\coherence
   ```

   Example 1–1 illustrates `cache-server.cmd` with the edited value of `COHERENCE_HOME`.

Example 1–1 cache-server.cmd File with an Edited COHERENCE_HOME

```
@echo off
@
@rem This will start a cache server
@
setlocal

:config
@rem specify the Coherence installation directory
set coherence_home=c:\oracle\product\coherence

@rem specify the JVM heap size
set memory=512m

:start
if not exist "%coherence_home%\lib\coherence.jar" goto instructions

if "%java_home%"=="" (set java_exec=java) else (set java_exec=%java_
home%\bin\java)

:launch
```

```
set java_opts="-Xms%memory% -Xmx%memory%"

"%java_exec%" -server -showversion "%java_opts%" -cp "%coherence_
home%\lib\coherence.jar" com.tangosol.net.DefaultCacheServer %1

goto exit

:instructions

echo Usage:
echo   ^<coherence_home^>\bin\cache-server.cmd
goto exit

:exit
endlocal
@echo on
```

3. Execute the cache server application that is located in the `coherence\bin` directory.

   ```
   C:\oracle\product\coherence\bin>cache-server.cmd
   ```

   When you start the first cache server, there is a slight delay because the cache server looks for an existing cluster. When it determines that there are no clusters to join, it starts one. On start-up, the cache server produces output similar to the text in Example 1–2.

   Several important features are highlighted in the example:

   - the Java JDK version number
   - information about how configuration files are loaded. The default is to load from JAR: `Loaded operational configuration from resource "jar:file:/C:/oracle/product/coherence/lib/coherence.jar!/tangosol-coherence.xml"`
   - the Coherence release number: `Oracle Coherence 3.6.0.0...`
   - the Coherence edition: `Grid Edition: Development mode`
   - the multicast address. This address changes with each Coherence version. Note the `3.6.0` in the address for Coherence Release 3.6: `Group{Address=224.3.6.0, Port=36000, TTL=4}`
   - the Member Id indicates the number of members in your cluster. For the purposes of this exercise, the value should be 1. `ThisMember=Member(Id=1...`

Example 1–2 Output from Starting a Coherence Cache Server

```
C:\oracle\product\coherence\bin>cache-server.cmd
java version "1.6.0_14"
Java(TM) SE Runtime Environment (build 1.6.0_14-b08)
Java HotSpot(TM) Server VM (build 14.0-b16, mixed mode)

2010-05-26 14:00:10.531/0.891 Oracle Coherence 3.6.0.0 DPR3 <Info> (thread=main, member=n/a):
Loaded operational configuration from "jar:file:/C:/oracle/product/coherence/lib/coherence.
jar!/tangosol-coherence.xml"
2010-05-26 14:00:10.546/0.906 Oracle Coherence 3.6.0.0 DPR3 <Info> (thread=main, member=n/a):
Loaded operational overrides from "jar:file:/C:/oracle/product/coherence/lib/coherence.
jar!/tangosol-coherence-override-dev.xml"
```

```
2010-05-26 14:00:10.546/0.906 Oracle Coherence 3.6.0.0 DPR3 <D5> (thread=main, member=n/a):
Optional configuration override "/tangosol-coherence-override.xml" is not specified
2010-05-26 14:00:10.562/0.922 Oracle Coherence 3.6.0.0 DPR3 <D5> (thread=main, member=n/a):
Optional configuration override "/custom-mbeans.xml" is not specified

Oracle Coherence Version 3.6.0.0 DPR3 Build 16141
 Grid Edition: Development mode
Copyright (c) 2000, 2010, Oracle and/or its affiliates. All rights reserved.

2010-05-26 14:00:11.109/1.469 Oracle Coherence GE 3.6.0.0 DPR3 <Info> (thread=main, member=n/a):
Loaded cache configuration from "jar:file:/C:/oracle/
product/coherence/lib/coherence.jar!/coherence-cache-config.xml"
2010-05-26 14:00:11.718/2.078 Oracle Coherence GE 3.6.0.0 DPR3 <D4> (thread=main, member=n/a):
SystemSocketProvider bound to port 8088
2010-05-26 14:00:15.312/5.672 Oracle Coherence GE 3.6.0.0 DPR3 <Info> (thread=Cluster, member=n/a):
Created a new cluster "cluster:0xC4DB" with Member(Id=1, Timestamp=2010-05-26 14:00:11.734,
Address=130.35.99.213:8088, MachineId=49877, Location=site:us.oracle.com,machine:tpfaeffl-
lap7,process:5588,
 Role=CoherenceServer, Edition=Grid Edition, Mode=Development, CpuCount=2, SocketCount=1)
UID=0x822363D500000128D66A5256C2D51F98
2010-05-26 14:00:15.328/5.688 Oracle Coherence GE 3.6.0.0 DPR3 <Info> (thread=main, member=n/a):
Started cluster Name=cluster:0xC4DB

Group{Address=224.3.6.0, Port=36000, TTL=4}

MasterMemberSet
  (
  ThisMember=Member(Id=1, Timestamp=2010-05-26 14:00:11.734, Address=130.35.99.213:8088,
MachineId=49877, Location=site:us.oracle.com,machine:tpfaeffl-lap7,process:5588,
Role=CoherenceServer)
  OldestMember=Member(Id=1, Timestamp=2010-05-26 14:00:11.734, Address=130.35.99.213:8088,
MachineId=49877, Location=site:us.oracle.com,machine:tpfaeffl-lap7,process:5588,
Role=CoherenceServer)
  ActualMemberSet=MemberSet(Size=1, BitSetCount=2
    Member(Id=1, Timestamp=2010-05-26 14:00:11.734, Address=130.35.99.213:8088, MachineId=49877,
Location=site:us.oracle.com,machine:tpfaeffl-lap7,process:5588, Role=CoherenceServer)
    )
  RecycleMillis=1200000
  RecycleSet=MemberSet(Size=0, BitSetCount=0
    )
  )

TcpRing{Connections=[]}
IpMonitor{AddressListSize=0}

2010-05-26 14:00:15.437/5.797 Oracle Coherence GE 3.6.0.0 DPR3 <D5> (thread=Invocation:Management,
member=1): Service Management joined the cluster with senior service member 1
2010-05-26 14:00:15.781/6.141 Oracle Coherence GE 3.6.0.0 DPR3 <D5> (thread=DistributedCache,
member=1): Service DistributedCache joined the cluster with senior service member 1
2010-05-26 14:00:15.859/6.219 Oracle Coherence GE 3.6.0.0 DPR3 <D5> (thread=ReplicatedCache,
member=1): Service ReplicatedCache joined the cluster with senior service member 1
2010-05-26 14:00:15.859/6.219 Oracle Coherence GE 3.6.0.0 DPR3 <D5> (thread=OptimisticCache,
member=1): Service OptimisticCache joined the cluster with senior service member 1
2010-05-26 14:00:15.859/6.219 Oracle Coherence GE 3.6.0.0 DPR3 <D5>
(thread=Invocation:InvocationService, member=1): Service InvocationService joined the cluster with
senior service member 1
2010-05-26 14:00:15.859/6.235 Oracle Coherence GE 3.6.0.0 DPR3 <Info> (thread=main, member=1):
Services
  (
```

```
  ClusterService{Name=Cluster, State=(SERVICE_STARTED, STATE_JOINED), Id=0, Version=3.6,
OldestMemberId=1}
  InvocationService{Name=Management, State=(SERVICE_STARTED), Id=1, Version=3.1, OldestMemberId=1}
  PartitionedCache{Name=DistributedCache, State=(SERVICE_STARTED), LocalStorage=enabled,
PartitionCount=257, BackupCount=1, AssignedPartitions=257, BackupPartitions=0}
  ReplicatedCache{Name=ReplicatedCache, State=(SERVICE_STARTED), Id=3, Version=3.0,
OldestMemberId=1}
  Optimistic{Name=OptimisticCache, State=(SERVICE_STARTED), Id=4, Version=3.0, OldestMemberId=1}
  InvocationService{Name=InvocationService, State=(SERVICE_STARTED), Id=5, Version=3.1,
OldestMemberId=1}
  )

Started DefaultCacheServer...
```

Note: By default, Coherence is configured to use multicast to attempt to join a cluster and to distribute cluster events. Multicast can also be used to distribute a message efficiently to multiple nodes in the cluster. Coherence can be configured so that it does not use multicast.

The output of `cache-server.cmd` indicates whether you have one or more members in your cluster. The value of `Member Id` should be equal to 1:

```
...
In MasterMemberSet
ThisMember=Member(Id should be equal to 1)
...
```

If `Member Id` is greater than 1, then multiple clusters are being formed in your subnet. For the purposes of these exercises, there should only be one member in the cluster. Follow the steps in "Restricting Coherence to Your Own Host" on page 1-11 to restrict Coherence to your own host.

Configure and Run the Sample Cache Client Application

Follow these steps to set-up and run the sample cache client application.

1. Open another terminal window to start the cache client.

   Verify that the `PATH` environment variable is set to include `%JAVA_HOME%\bin`. If the `PATH` environment variable does not include `%JAVA_HOME%\bin`, then set the variable as described in Step 1.

2. Navigate to the `\oracle\product\coherence\bin` directory. Edit `coherence.cmd` and set the `COHERENCE_HOME` variable to point to the Coherence installation directory.

   Example 1–3 illustrates the `coherence.cmd` file, with `COHERENCE_HOME=\oracle\product\coherence`.

Example 1–3 coherence.cmd File with an Edited COHERENCE_HOME

```
@echo off
@
@rem This will start a console application
@rem demonstrating the functionality of the Coherence(tm) API
@
setlocal
```

```
:config
@rem specify the Coherence installation directory
set coherence_home=c:\oracle\product\coherence

@rem specify if the console will also act as a server
set storage_enabled=false

@rem specify the JVM heap size
set memory=64m

:start
if not exist "%coherence_home%\lib\coherence.jar" goto instructions

if "%java_home%"=="" (set java_exec=java) else (set java_exec=%java_
home%\bin\java)

:launch

if "%storage_enabled%"=="true" (echo ** Starting storage enabled console **) else
(echo ** Starting storage disabled console **)

set java_opts="-Xms%memory% -Xmx%memory% -Dtangosol.coherence.distributed.
localstorage=%storage_enabled%"

"%java_exec%" -server -showversion "%java_opts%" -cp "%coherence_
home%\lib\coherence.jar" com.tangosol.net.CacheFactory %1

goto exit

:instructions

echo Usage:
echo   ^<coherence_home^>\bin\coherence.cmd
goto exit

:exit
endlocal
@echo on
```

3. Execute the `coherence.cmd` file to start the cache client. This application shows you the basic distributed cache functionality that is built within Coherence.

   `coherence.cmd`

   Example 1–4 illustrates the output from starting the cache client. Note the following features of the output:

   - the client is the second member of the cluster (the server is the first member): `ThisMember=Member(Id=2,...`
   - at the end of the output, you should see the `Map(?)` prompt

Example 1–4 Output from Starting the Coherence Cache Client

```
** Starting storage disabled console **
java version "1.6.0_14"
Java(TM) SE Runtime Environment (build 1.6.0_14-b08)
Java HotSpot(TM) Server VM (build 14.0-b16, mixed mode)
```

```
2010-05-26 14:15:08.187/0.344 Oracle Coherence 3.6.0.0 DPR3 <Info> (thread=main, member=n/a):
Loaded operational configuration from "jar:file:/C:/oracle/product/coherence/lib/coherence.
jar!/tangosol-coherence.xml"
2010-05-26 14:15:08.203/0.360 Oracle Coherence 3.6.0.0 DPR3 <Info> (thread=main, member=n/a):
Loaded operational overrides from "jar:file:/C:/oracle/product/coherence/lib/coherence.
jar!/tangosol-coherence-override-dev.xml"
2010-05-26 14:15:08.203/0.360 Oracle Coherence 3.6.0.0 DPR3 <D5> (thread=main, member=n/a):
Optional configuration override "/tangosol-coherence-override.xml" is not specified
2010-05-26 14:15:08.203/0.360 Oracle Coherence 3.6.0.0 DPR3 <D5> (thread=main, member=n/a):
Optional configuration override "/custom-mbeans.xml" is not specified

Oracle Coherence Version 3.6.0.0 DPR3 Build 16141
 Grid Edition: Development mode
Copyright (c) 2000, 2010, Oracle and/or its affiliates. All rights reserved.

2010-05-26 14:15:08.859/1.016 Oracle Coherence GE 3.6.0.0 DPR3 <D4> (thread=main, member=n/a):
SystemSocketProvider bound to port 8090
2010-05-26 14:15:10.125/2.282 Oracle Coherence GE 3.6.0.0 DPR3 <Info> (thread=Cluster, member=n/a):
This Member(Id=2, Timestamp=2010-05-26 14:15:10.102, Address=130.35.99.213:8090, MachineId=49877,
Location=site:us.oracle.com,machine:tpfaeffl-lap7,process:636, Role=CoherenceConsole, Edition=Grid
Edition, Mode=Development, CpuCount=2, SocketCount=1) joined cluster "cluster:0xC4DB" with senior
Member(Id=1, Timestamp=2010-05-26 14:00:11.734, Address=130.35.99.213:8088, MachineId=49877,
Location=site:us.oracle.com,machine:tpfaeffl-lap7,process:5588, Role=CoherenceServer, Edition=Grid
Edition, Mode=Development, CpuCount=2, SocketCount=1)
2010-05-26 14:15:10.187/2.344 Oracle Coherence GE 3.6.0.0 DPR3 <D5> (thread=Cluster, member=n/a):
Member 1 joined Service Cluster with senior member 1

2010-05-26 14:15:10.187/2.344 Oracle Coherence GE 3.6.0.0 DPR3 <D5> (thread=Cluster, member=n/a):
Member 1 joined Service Management with senior member 1
2010-05-26 14:15:10.187/2.344 Oracle Coherence GE 3.6.0.0 DPR3 <D5> (thread=Cluster, member=n/a):
Member 1 joined Service DistributedCache with senior member 1
2010-05-26 14:15:10.187/2.344 Oracle Coherence GE 3.6.0.0 DPR3 <D5> (thread=Cluster, member=n/a):
Member 1 joined Service ReplicatedCache with senior member 1
2010-05-26 14:15:10.187/2.344 Oracle Coherence GE 3.6.0.0 DPR3 <D5> (thread=Cluster, member=n/a):
Member 1 joined Service OptimisticCache with senior member 1
2010-05-26 14:15:10.187/2.344 Oracle Coherence GE 3.6.0.0 DPR3 <D5> (thread=Cluster, member=n/a):
Member 1 joined Service InvocationService with senior member 1
2010-05-26 14:15:10.187/2.344 Oracle Coherence GE 3.6.0.0 DPR3 <Info> (thread=main, member=n/a):
Started cluster Name=cluster:0xC4DB

Group{Address=224.3.6.0, Port=36000, TTL=4}

MasterMemberSet
  (
  ThisMember=Member(Id=2, Timestamp=2010-05-26 14:15:10.102, Address=130.35.99.213:8090,
MachineId=49877, Location=site:us.oracle.com,machine:tpfaeffl-lap7,process:636,
Role=CoherenceConsole)
  OldestMember=Member(Id=1, Timestamp=2010-05-26 14:00:11.734, Address=130.35.99.213:8088,
MachineId=49877, Location=site:us.oracle.com,machine:tpfaeffl-lap7,process:5588,
Role=CoherenceServer)
  ActualMemberSet=MemberSet(Size=2, BitSetCount=2
    Member(Id=1, Timestamp=2010-05-26 14:00:11.734, Address=130.35.99.213:8088, MachineId=49877,
Location=site:us.oracle.com,machine:tpfaeffl-lap7,process:5588, Role=CoherenceServer)
    Member(Id=2, Timestamp=2010-05-26 14:15:10.102, Address=130.35.99.213:8090, MachineId=49877,
Location=site:us.oracle.com,machine:tpfaeffl-lap7,process:636, Role=CoherenceConsole)
    )
  RecycleMillis=1200000
  RecycleSet=MemberSet(Size=0, BitSetCount=0
    )
  )
```

```
TcpRing{Connections=[1]}
IpMonitor{AddressListSize=0}

2010-05-26 14:15:10.296/2.453 Oracle Coherence GE 3.6.0.0 DPR3 <D5> (thread=Invocation:Management,
member=2): Service Management joined the cluster with senior service member 1

Map (?):
```

Exercise the Sample Cache Client Application

Exercise the cache client application by entering various commands and examining the output.

1. Execute the following Coherence commands in the cache client:
   - Enter `help` to see the list of commands that are available.
   - Enter `cache myCache` to create a cache named `myCache`.

     The cache `myCache` implements the `com.tangosol.net.NamedCache` interface. A cluster can have many named caches.

     Example 1–5 illustrates that using the default configuration files (`coherence-cache-config.xml`) within the supplied `coherence.jar` file, a `NamedCache` called `myCache` is created using the distributed scheme:

Example 1–5 Output from Starting a Coherence Cache

```
Map (?): cache myCache
2010-05-26 15:00:09.296/2701.453 Oracle Coherence GE 3.6.0.0 DPR3 <Info> (thread=main, member=2):
Loaded cache configuration from "jar:file:/C:/oracle/product/coherence/lib/coherence.
jar!/coherence-cache-config.xml"
2010-05-26 15:00:09.515/2701.672 Oracle Coherence GE 3.6.0.0 DPR3 <D5> (thread=DistributedCache,
member=2): Service DistributedCache joined the cluster with senior service member 1
<distributed-scheme>
  <scheme-name>example-distributed</scheme-name>
  <service-name>DistributedCache</service-name>
  <backing-map-scheme>
    <local-scheme>
      <scheme-ref>example-binary-backing-map</scheme-ref>
    </local-scheme>
  </backing-map-scheme>
  <autostart>true</autostart>
</distributed-scheme>

Map (myCache):
```

2. Execute the following commands at the `Map(myCache)` prompt in the cache client. For definitions of these commands, see Appendix A, "Coherence Client Application Commands."
   - `get message`
   - `put message "hello"`
   - `list`
   - `size`
   - `get message`
   - `put message "second message"`

- `get message`
- `remove message`
- `size`
- `get message`
- `put message "hi"`
- `bye`

Example 1–6 illustrates the output of each of these commands.

Example 1–6 Exercising Coherence Commands

```
Map (mycache): get message
null

Map (mycache): put message "hello"
null

Map (mycache): list
message = hello

Map (mycache): size
1

Map (mycache): get message
hello

Map (mycache): put message "second message"
hello

Map (mycache): get message
second message

Map (mycache): remove message
second message

Map (mycache): size
0

Map (mycache): get message
null

Map (mycache): put message "hi"
null

Map (myCache): bye
2010-05-26 15:03:28.968/2901.125 Oracle Coherence GE 3.6.0.0 DPR3 <D5>
(thread=Invocation:Management, member=2): Service Management left the cluster
2010-05-26 15:03:28.984/2901.141 Oracle Coherence GE 3.6.0.0 DPR3 <D5>
(thread=DistributedCache, member=2): Service DistributedCache left the cluster
2010-05-26 15:03:29.093/2901.250 Oracle Coherence GE 3.6.0.0 DPR3 <D5>
(thread=Cluster, member=2): Service Cluster left the cluster

C:\oracle\product\coherence\bin>
```

3. Open another terminal window and set the `PATH` environment variable to include `%JAVA_HOME%` and `%JAVA_HOME%\bin`. (See Step 4).

4. Start a second instance of `coherence.cmd` in the new terminal window. Restart the first client. The terminal window displays a message similar to the following that describes where the first client is running. `member 3` is the second client and `Member 4` is the restarted first client.

```
Map (myCache):
2010-05-26 15:06:55.671/46.453 Oracle Coherence GE 3.6.0.0 DPR3 <D5>
(thread=Cluster, member=3): Member 4 joined Service Management with senior
member 1
```

5. Use the `cache myCache` command in the new terminal to connect to the `NamedCache` called `myCache`. Try to get and put values in different sessions. notice that each client can observe changes made by the other client.

6. Terminate one of the `coherence.cmd` shells (`bye`). Note that the other shell displays a message indicating that the member has left the cluster. For example:

```
Map (myCache):
2010-05-26 15:20:52.156/317.375 Oracle Coherence GE 3.6.0.0 DPR3 <D5>
(thread=Cluster, member=5): MemberLeft announcement from Member(Id=3,
Timestamp=2010-05-26 15:06:11.258, Address=130.35.99.213:8090, MachineId=49877,
Location=site:us.oracle.com,machine:tpfaeffl-lap7,process:3708,
Role=CoherenceConsole)
2010-05-26 15:20:52.171/317.390 Oracle Coherence GE 3.6.0.0 DPR3 <D5>
(thread=Cluster, member=5): Member(Id=3, Timestamp=2010-05-26 15:20:52.171,
Address=130.35.99.213:8090, MachineId=49877, Location=site:us.oracle.
com,machine:tpfaeffl-lap7,process:3708, Role=CoherenceConsole) left Cluster
with senior member 1
```

7. If you terminate each of the cache clients (`bye`), and then restart them, note that data from the previous session still available. This is because the data is held in the cache server.

8. If you start another Coherence cache server, and then terminate the initial one that was started (using `Ctrl+C` or by closing the command window), note that the data is still available.

9. Terminate both the `coherence.cmd` shells and `cache-server.cmd` shell. Restart `coherence.cmd`. Create a `NamedCache` called `test` using the command `cache test`. Try to put in a value as before and note the results.

10. Start a cache server by running the `cache-server.cmd` file from the `coherence\bin` directory. Try to put in a value again and note the results.

11. Terminate all the cache servers.

Troubleshooting Cache Server Clustering

If the value of Member ID in the `cache-server.cmd` output is anything other than 1, then this indicates that the cache server has clustered with one or more other cache servers or processes running Coherence. These servers or processes may be running on the network or running locally on your host. Though this is the default behavior for Coherence—to cluster with other processes running Coherence locally or on a network—it is strongly advised, while you perform this tutorial, that you restrict Coherence to your own host.

Restricting Coherence to Your Own Host

The output of `cache-server.cmd` indicates whether you have just one member in your cluster. The value of Member ID should be equal to 1:

```
...
In MasterMemberSet
ThisMember=Member(Id should be equal to 1)
...
```

If Member ID is greater than 1, it means that multiple clusters are being formed in your subnet. For the purposes of the exercises in this document, there should only be one member in the cluster.

To restrict Coherence to your own host:

1. Stop cache server by pressing `Ctrl+C`.
2. Create a directory called `backup` under `oracle\product\coherence\lib`.

   ```
   cd C:\oracle\product\coherence\lib
   mkdir backup
   ```

3. Copy `coherence.jar` from the `oracle\product\coherence\lib` directory to the `backup` directory.

   ```
   cp coherence.jar C:\oracle\product\coherence\lib\backup\coherence.jar
   ```

4. Extract `tangosol-coherence.xml` from the original `coherence.jar` in the `lib` directory.

   ```
   jar -xvf coherence.jar tangosol-coherence.xml
   ```

5. Edit the `tangosol-coherence.xml` file.

   Change the multicast listener port to a unique value, for example, `34408`. For example, if you share the port value `34407` with another user, then change your port value to `34408` and the other user can change to `34409`.

   Example 1–7 illustrates the `multicast-listener` fragment of `tangosol-coherence.xml` with its original port value of `34407`.

Example 1–7 Multicast-Listener Fragment of tangosol-coherence.xml File

```
    <multicast-listener>
      <address system-property="tangosol.coherence.clusteraddress">224.3.6.
0</address>
      <port system-property="tangosol.coherence.clusterport">36000</port>

      <!--
      Note: For production use, this value should be set to the lowest integer
      value that works. On a single server cluster, it should work at "0"; on
      a simple switched backbone, it should work at "1"; on an advanced backbone
      with intelligent switching, it may require a value of "2" or more. Setting
      the value too high can utilize unnecessary bandwidth on other LAN segments
      and can even cause the OS or network devices to disable multicast traffic.
      -->
      <time-to-live system-property="tangosol.coherence.ttl">4</time-to-live>
```

6. Use the Java `jar` command or your favorite compression utility to append the modified `tangosol-coherence.xml` file to `coherence.jar`.

   When you execute `cache-server.cmd`, you should be able to view the modified port. Also, your cluster should now be restricted to your own host. If Membership ID still displays something other than `1`, then there may be additional issues with the installation. See "Advanced Steps to Restrict Coherence to Your Own Host".

Advanced Steps to Restrict Coherence to Your Own Host

If you follow the steps in the previous section and `cache-server.cmd` still fails to return a Member Id value of 1, then there may be additional issues to resolve.

Disconnect from the network or disable networking on your host. If errors or exceptions occur when starting the cache server, your network settings might need to be modified. Try each of the following one at a time, restarting the cache server between each attempt:

- If connected to a VPN, disconnect from it. By default, most VPN networks are not configured to permit multicast and some unicast traffic. In this environment, Coherence may not work as it is configured out-of-the-box. Coherence can be configured to run across a VPN, but this requires some advanced settings.
- If you run a firewall, configure it to allow the specified addresses and ports.
- If you still experience problems, unplug or disconnect from all the networks. This includes wireless and wired networks.
- If all the preceding options fail, set up Coherence to run on a single host.

2

Using JDeveloper with Coherence

This chapter describes how to set up JDeveloper to build and run Coherence-based Java applications.

Configuring Oracle JDeveloper for Coherence

To start and configure JDeveloper for use with Coherence:

1. Open a terminal window and verify that the `PATH` environment variable is set to include the Java JDK; for example: `\oracle\product\jdk160_14_R27.6.5-32\bin`. Note that this tutorial assumes that you have a working installation of Java SE (JDK) version 1.6 or higher.

   If the `PATH` environment variable does not include the Java JDK `\bin` directory, then set the variable as follows:

   a. Set the `JAVA_HOME` environment variable to the base of the JDK installation. For example:

      ```
      set JAVA_HOME=\oracle\product\jdk160_14_R27.6.5-32
      ```

   b. Include `%JAVA_HOME%\bin` in the PATH environment variable. For example:

      ```
      set PATH=%JAVA_HOME%\bin;%PATH%
      ```

   c. Start JDeveloper. If you get a message asking you to select a role, select **Default Role**. If you get a message asking whether you want to migrate from previous versions of JDeveloper, select **No**. Close any window that displays the **Tip of the Day**.

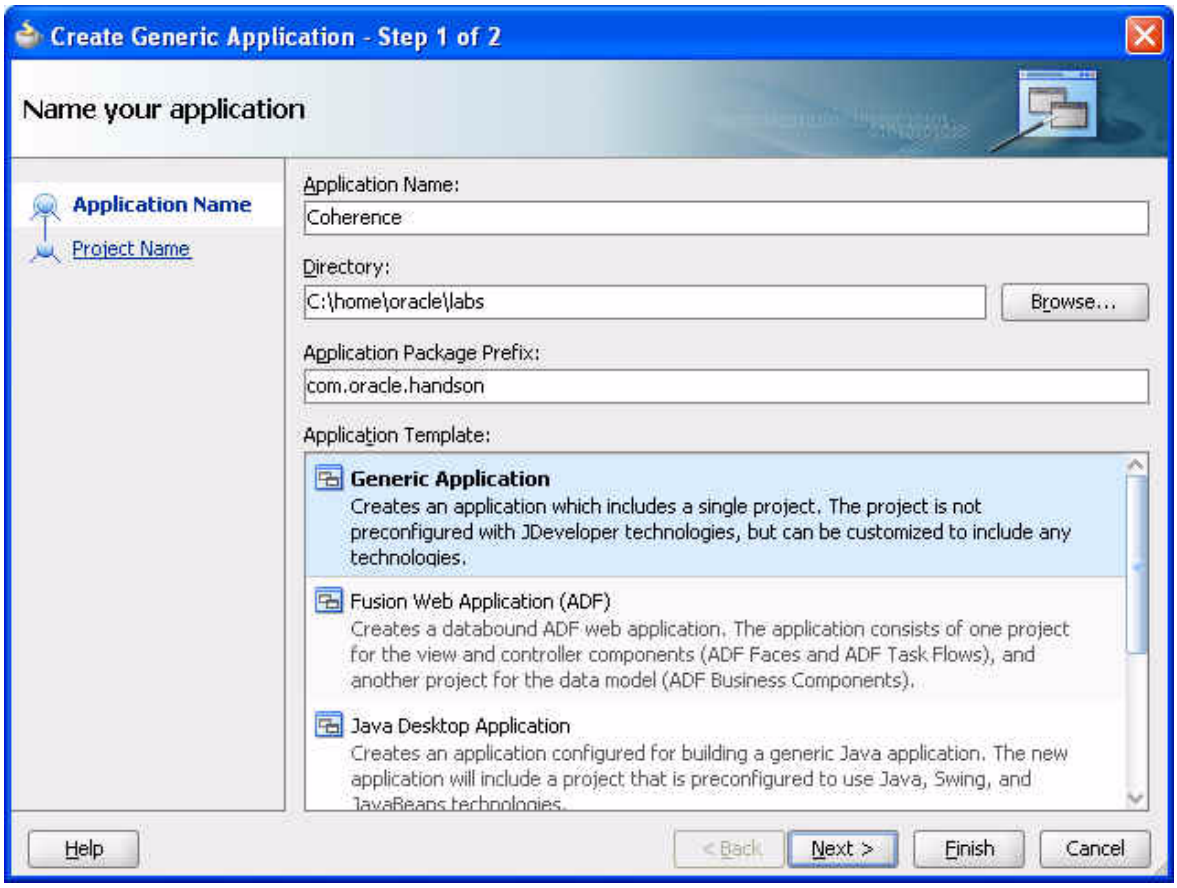
Figure 2–1 Select Role Dialog Box

2. After JDeveloper starts, select **File** then **New** to open the **New Gallery**. Select **Generic Application** in the **Items** column to create an application. An application is a way of grouping projects or source code. This new application will hold the Coherence projects.

3. In the **Create Generic Application** dialog box, change the application name to `Coherence`, the directory path to `\home\oracle\labs`, and the application package prefix to `com.oracle.handson`. Click **Finish**.

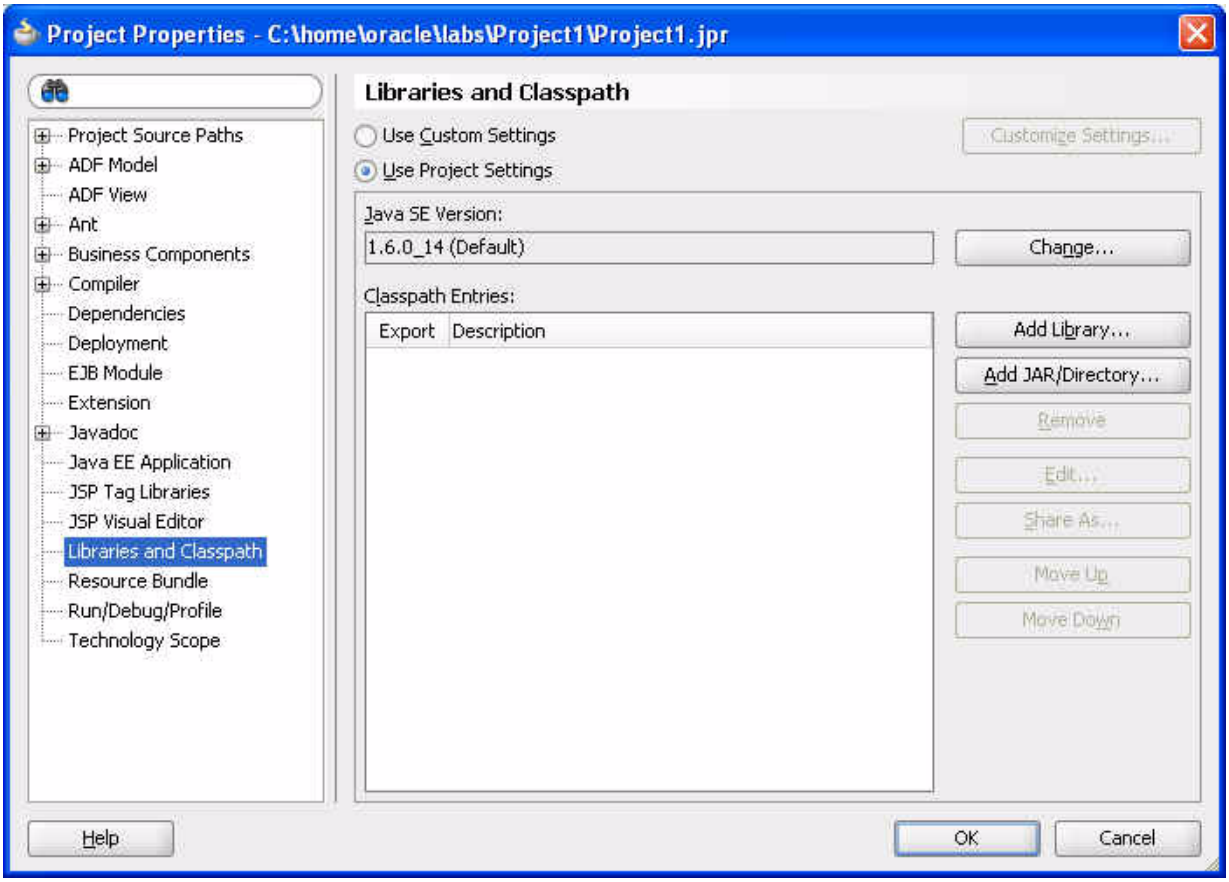
   Figure 2–2 illustrates the **Create Generic Application** dialog box.

Figure 2–2 Creating an Application in JDeveloper



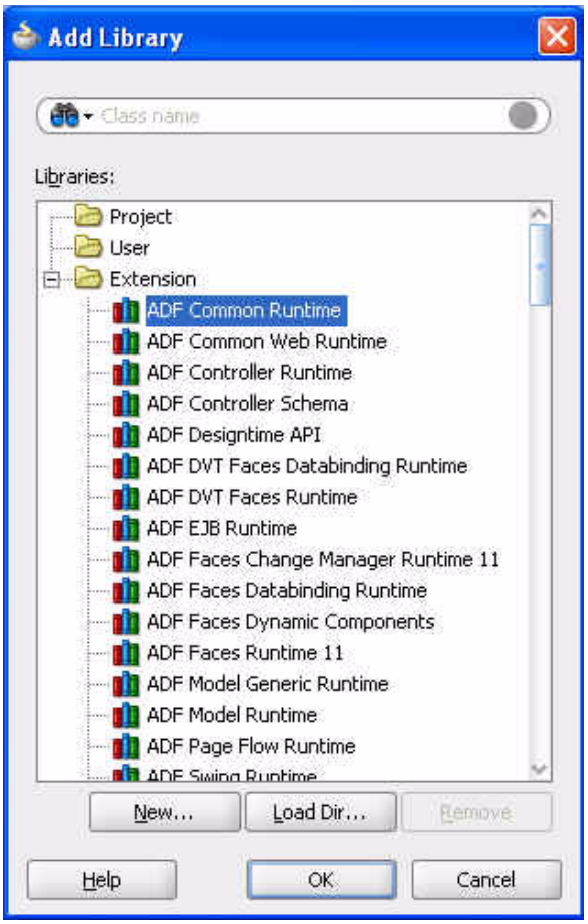
4. Add the Coherence JAR files to the default project properties. Right click the project in the **Application Navigator** and select **Project Properties**. The **Project Properties** dialog box opens.

Figure 2–3 Default Properties Dialog Box



5. Click **Libraries and Classpath**, and click the **Add Library** button. The **Add Library** dialog box opens.

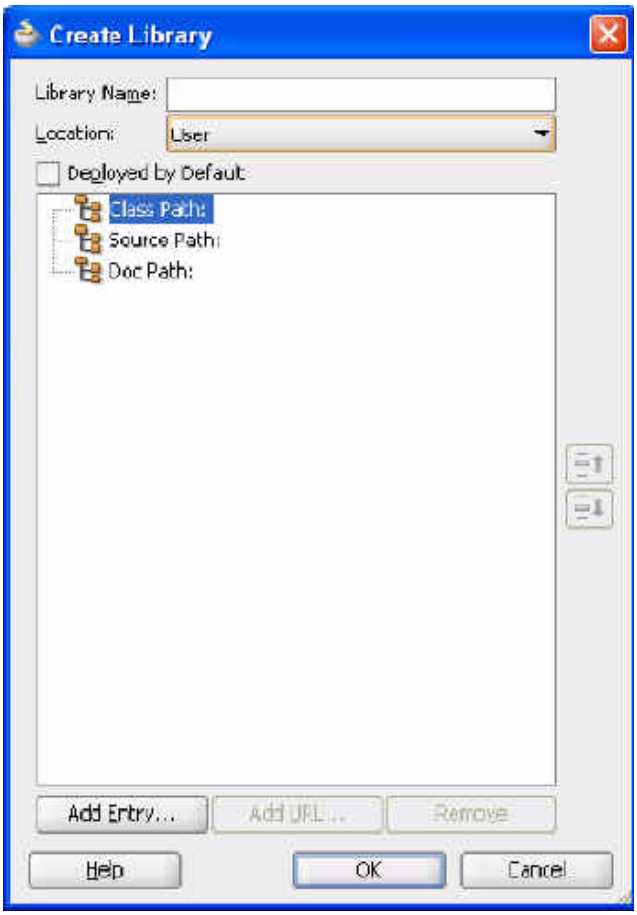
Figure 2–4 Add Library Dialog Box



6. In the **Add Library** dialog box, click **New**.

   The **Create Library** dialog box opens.

Figure 2–5 Create Library Dialog Box

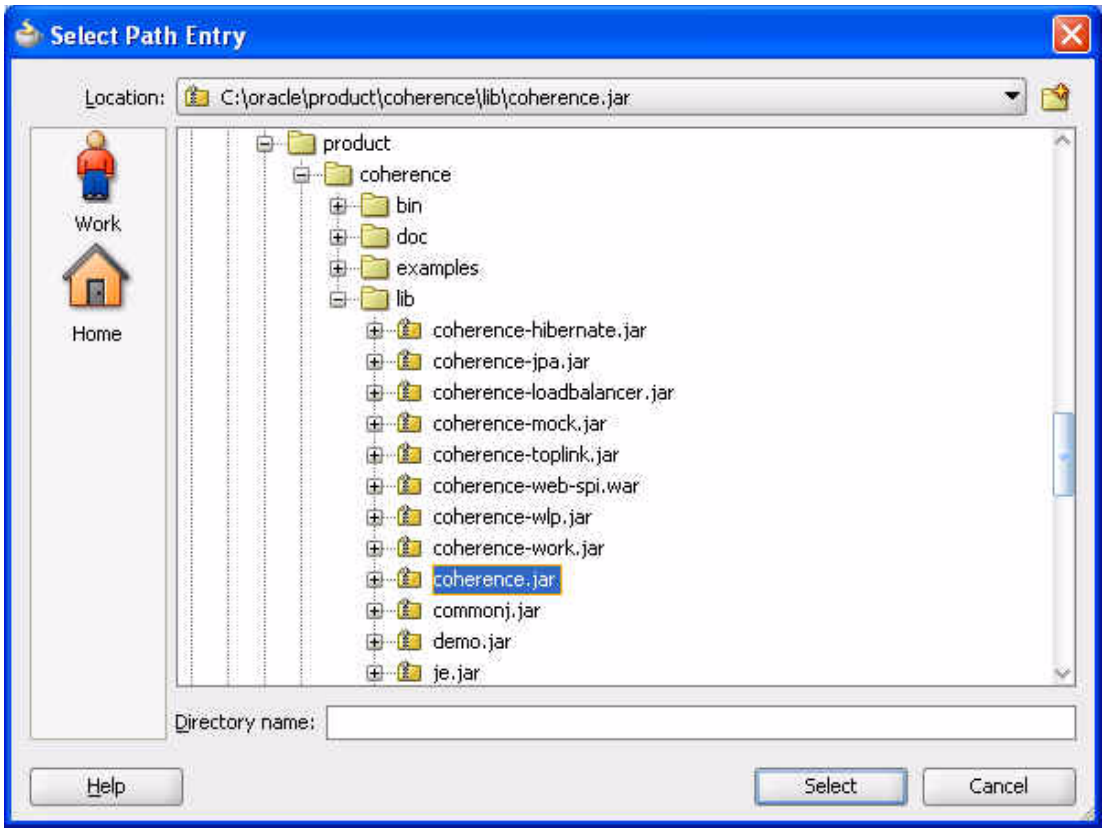


7. Click **Add Entry** to open the **Select Path Entry** dialog box.

   Find and expand the `coherence\lib` directory, which is under the `\oracle\product`.

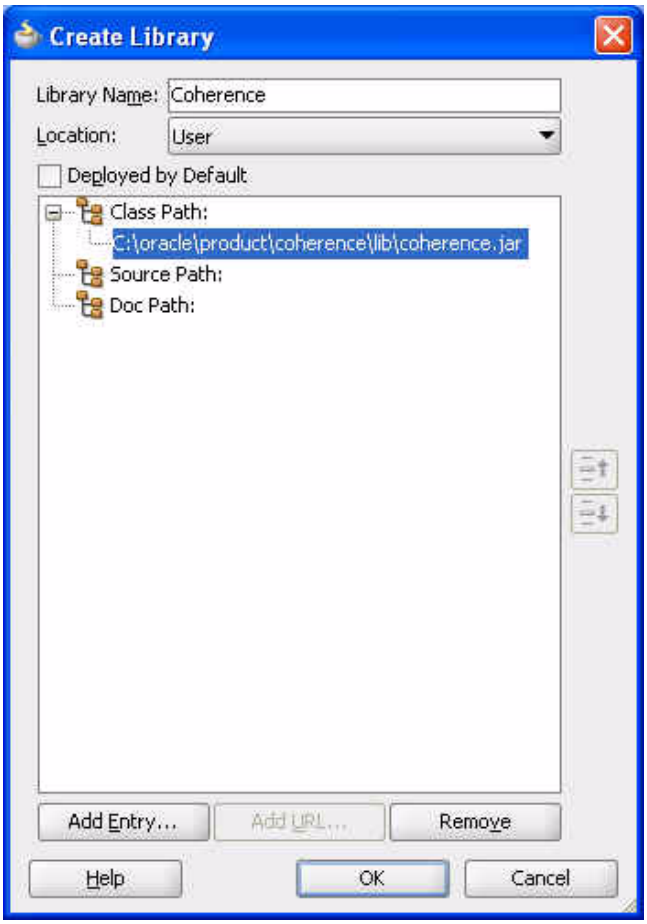
8. Highlight and select the `coherence.jar` file to add it to the classpath. Click **OK** to return to the **Create Library** dialog box.

Figure 2–6 Select Path Entry Dialog Box



9. In the **Create Library** dialog box, change the name in the **Library Name** field to **Coherence**. Use the drop-down list to change the **Location** value from **Project** to **User**. The contents of the dialog box should look like this:

Figure 2–7 Create Library Dialog Box with the Coherence Jar on the Classpath



10. Click **OK** in the **Create Library**, **Add Library**, and **Default Project Properties** dialog boxes to return to the JDeveloper IDE. Oracle JDeveloper is now set up with the correct Coherence libraries.

Learning JDeveloper Basics

This section describes common tasks that are a part of building projects in JDeveloper.

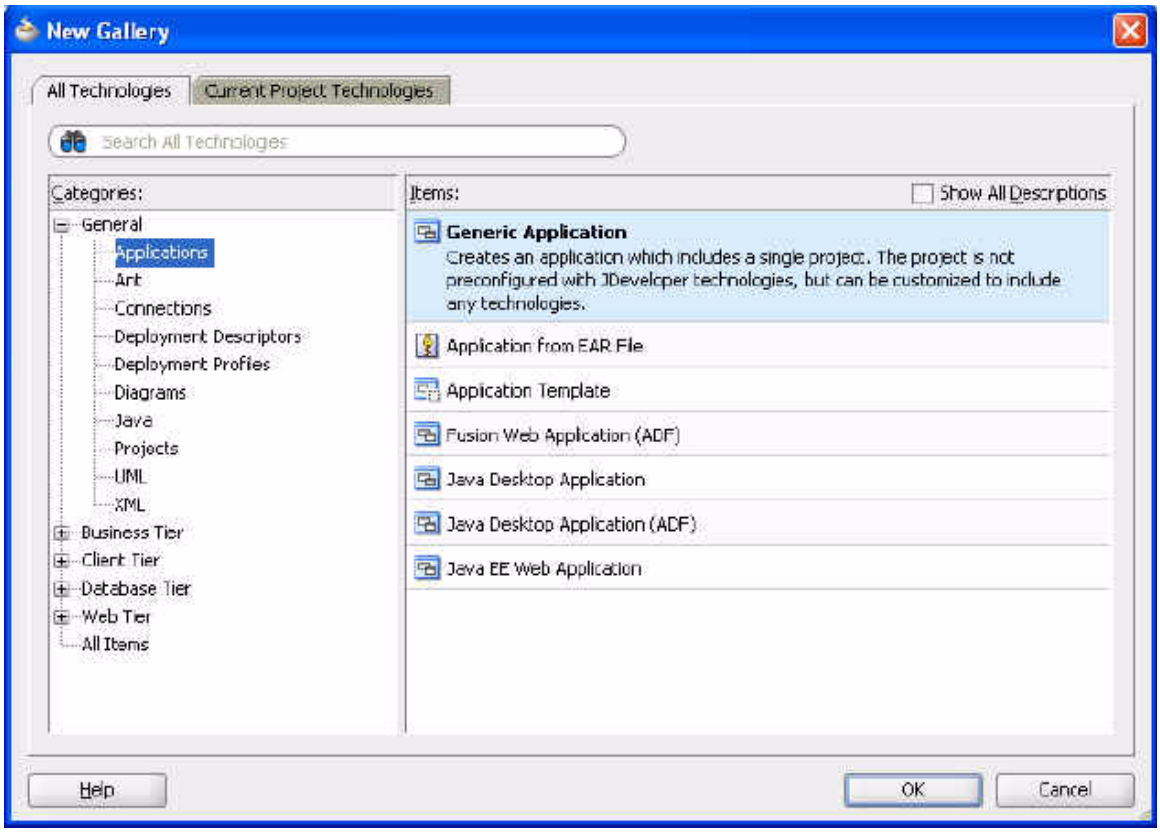
- Creating a New Application and Project
- Creating a Java Class
- Changing Project Properties, Setting Runtime Configuration

Creating a New Application and Project

To create a new application in JDeveloper:

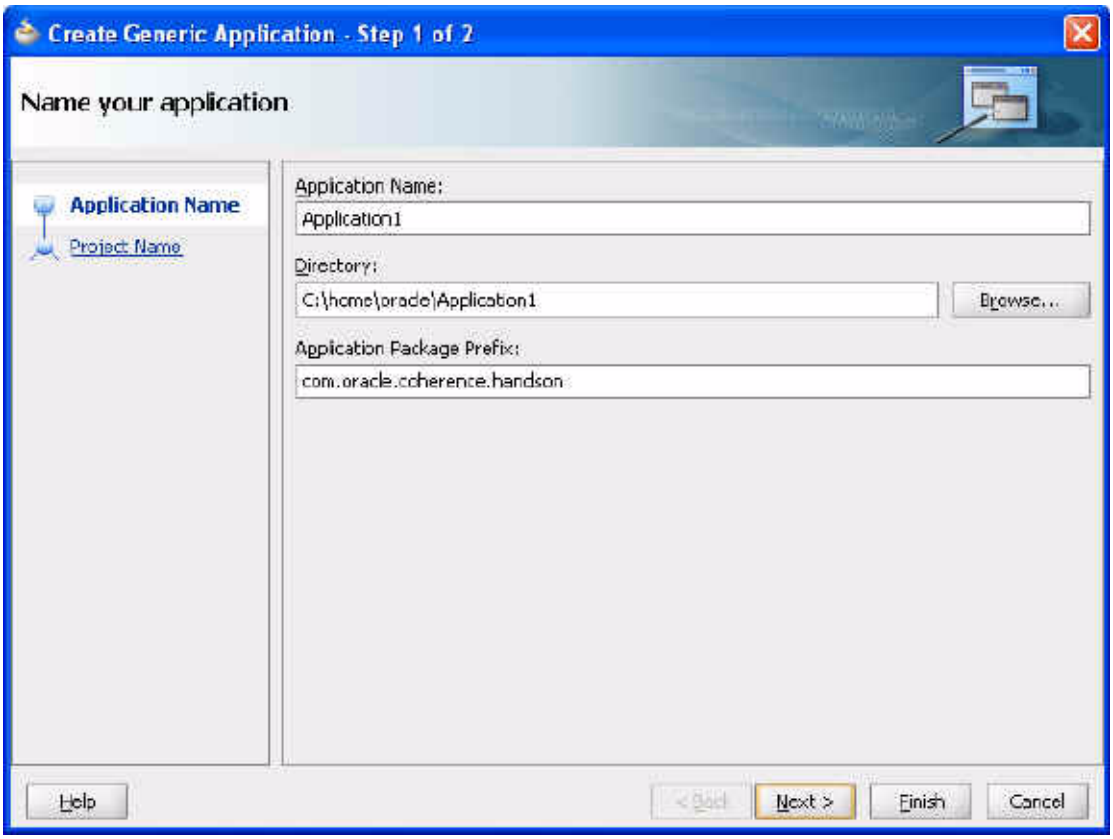
1. Choose **File** then **New...** to open the **New Gallery**.
2. Choose **General** then **Application** in the **Categories** section of the **New Gallery** dialog box and **Generic Application** in the **Items** section. Click **OK**.

Figure 2–8 Creating a New Application in the New Gallery



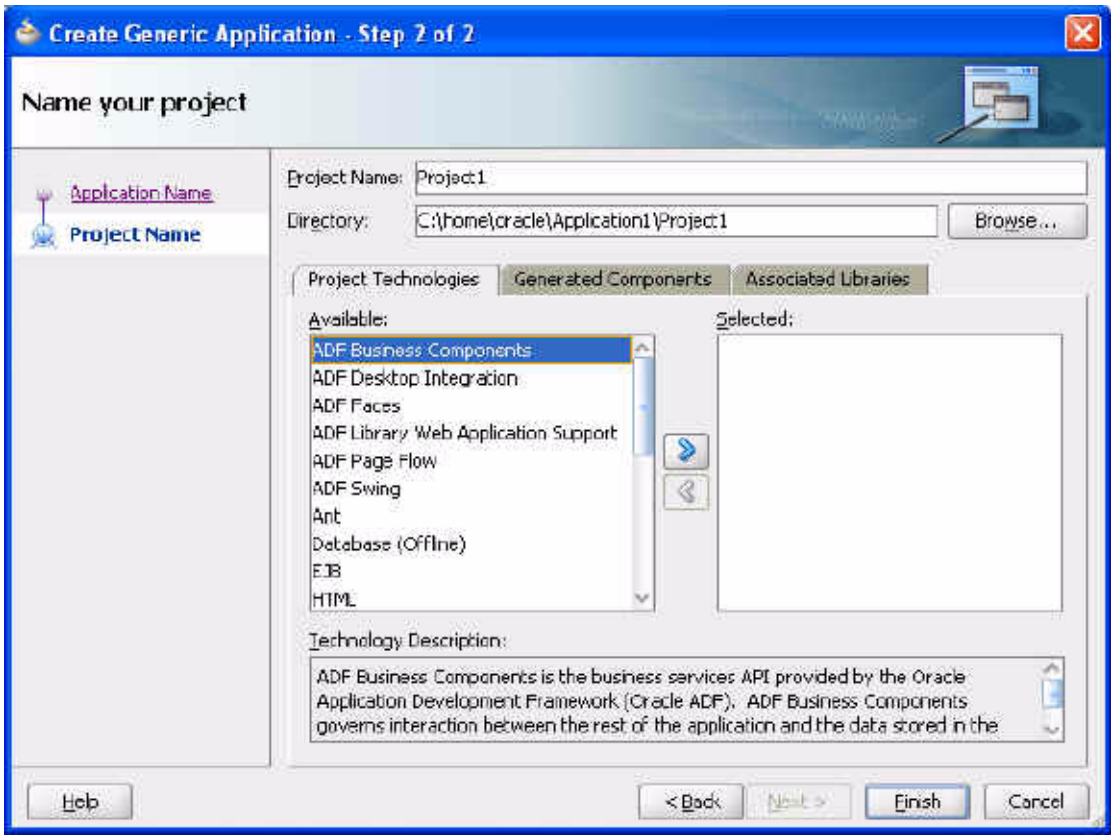
3. In the **Create Generic Application** dialog box, enter the name the project.
   - Replace **Application1** with the name of your application.
   - Ensure that directory path to the project is correct.
   - Click **Next**.

Figure 2–9 Providing an Application Name



4. JDeveloper asks you to create a new project within the application. Enter a name for the project. If necessary, select a project technology from the **Available** list. Click **Finish** to create the new project.

Figure 2–10 Providing a Project Name

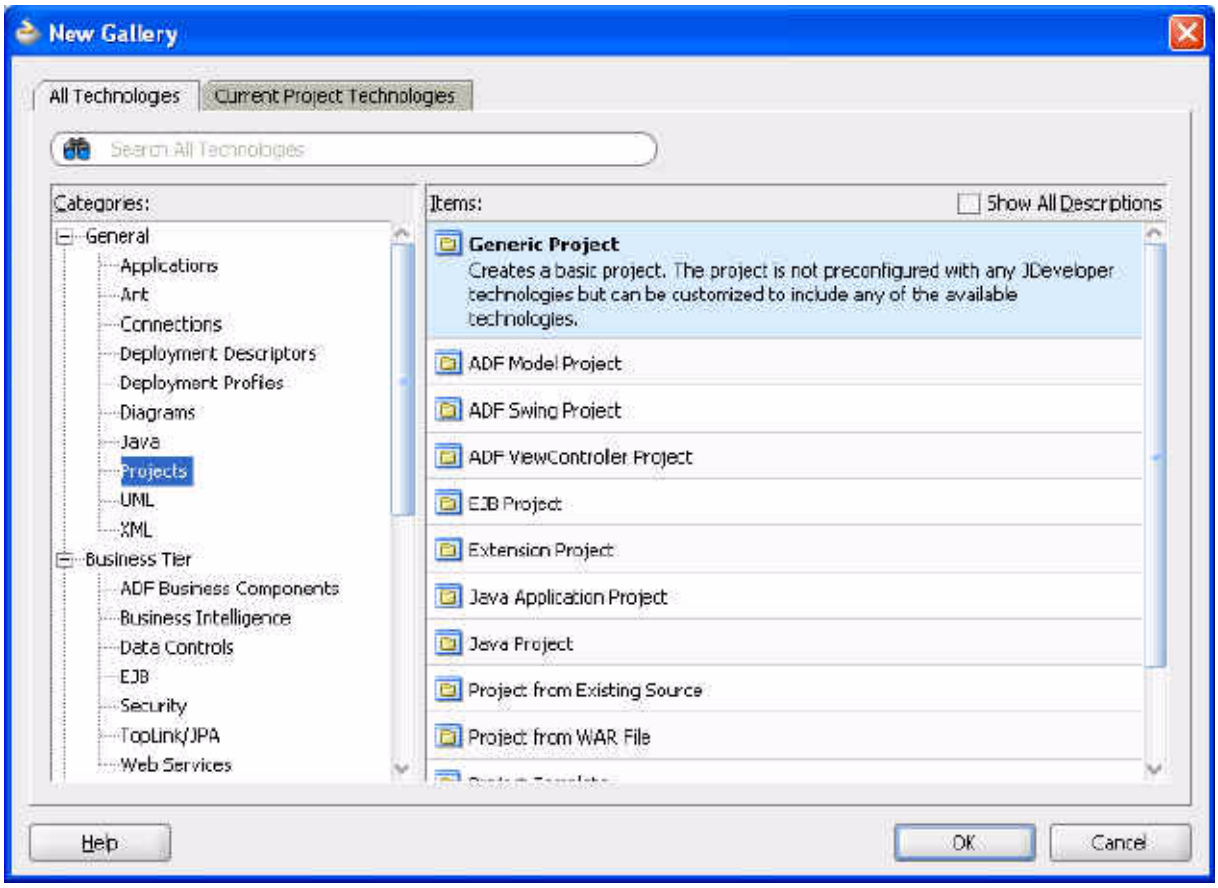


Creating a New Project in an Existing Application

To create a project in an existing application:

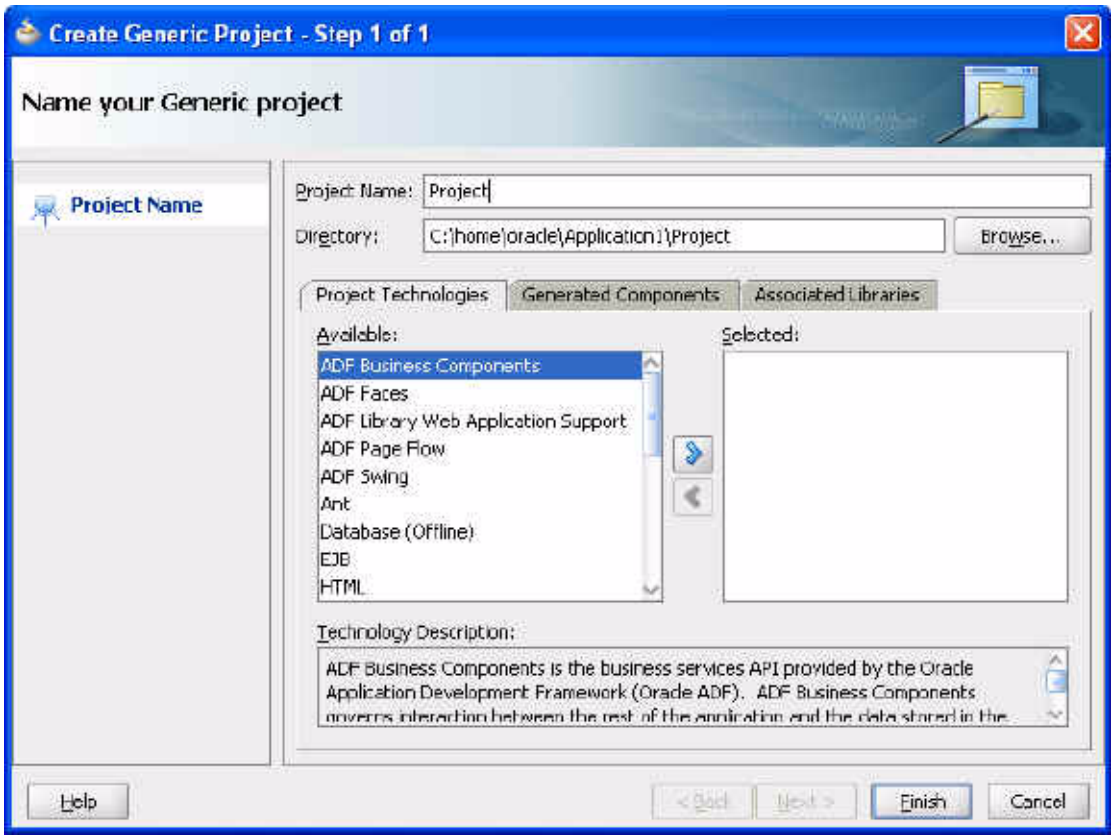
1. Right-click the Coherence application and choose **New...**. In the **New Gallery** choose **Projects** under **Categories** and **Generic Project** under **Items**. Click **OK**.

Figure 2–11 Creating a New Project in an Existing Application



2. Enter a **Project Name**, ensure that the **Default Package** is set correctly, and select a **Project Technology** if necessary. Click **Finish**.

Figure 2–12 Providing Details for the New Project

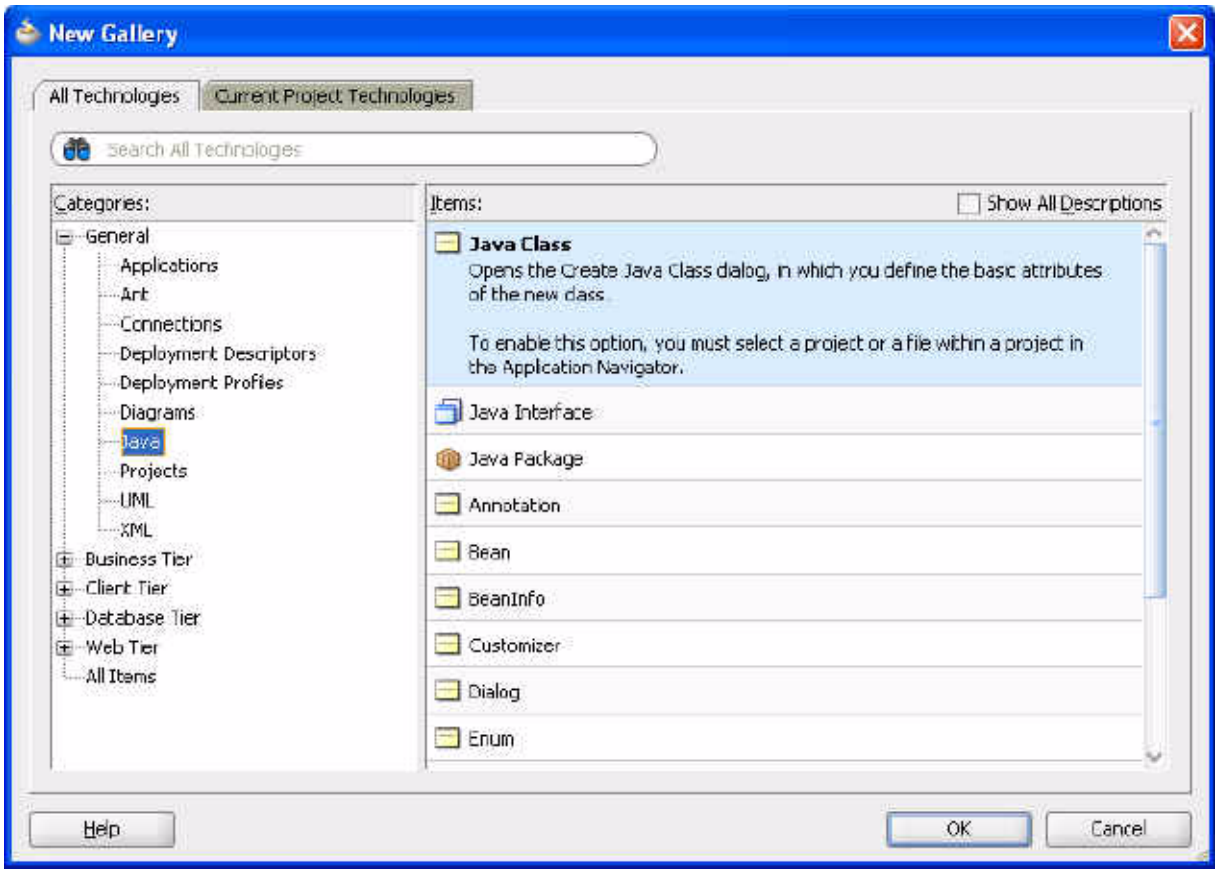


Creating a Java Class

To create a new Java class in JDeveloper:

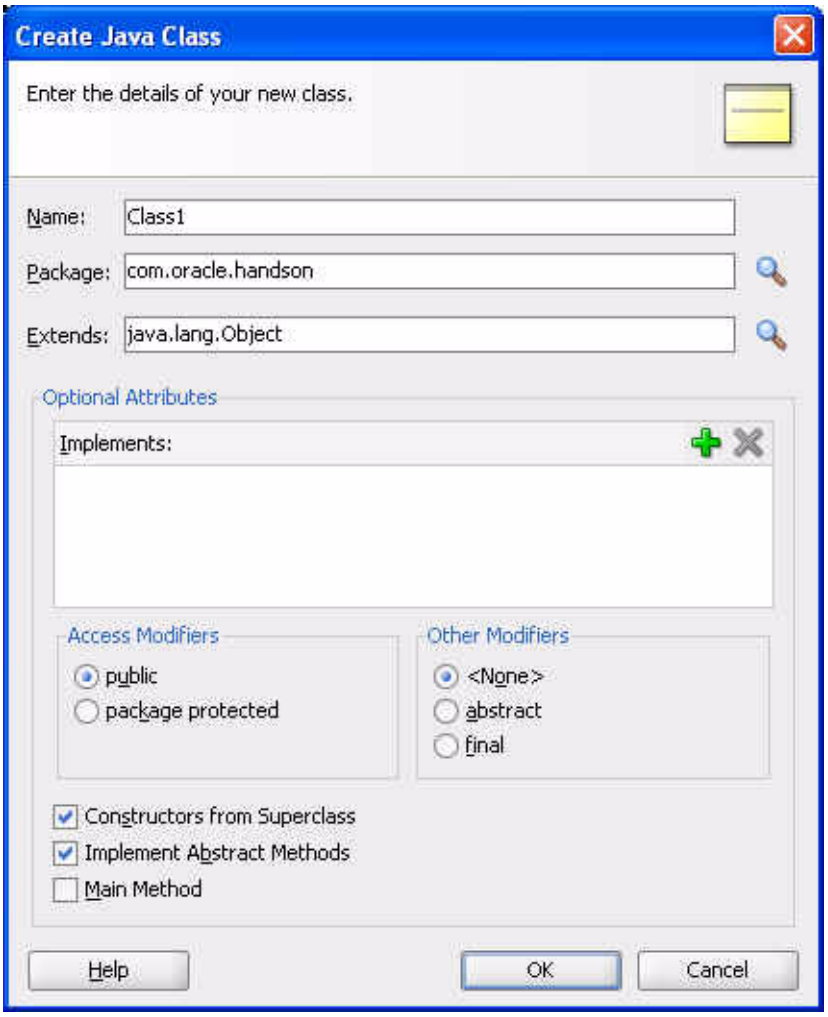
1. Right-click the project entry in the Navigator pane and select **New**.
2. Select **Java** under the **General** category and select **Java Class**. Click **OK**.

Figure 2–13 Creating a Java Class in the New Gallery



3. Replace **Class1** with the name of your class. Select the **Main Method** check box if the file must be runnable. Click **OK** to create the Java class.

Figure 2–14 Providing Details for the Java Class

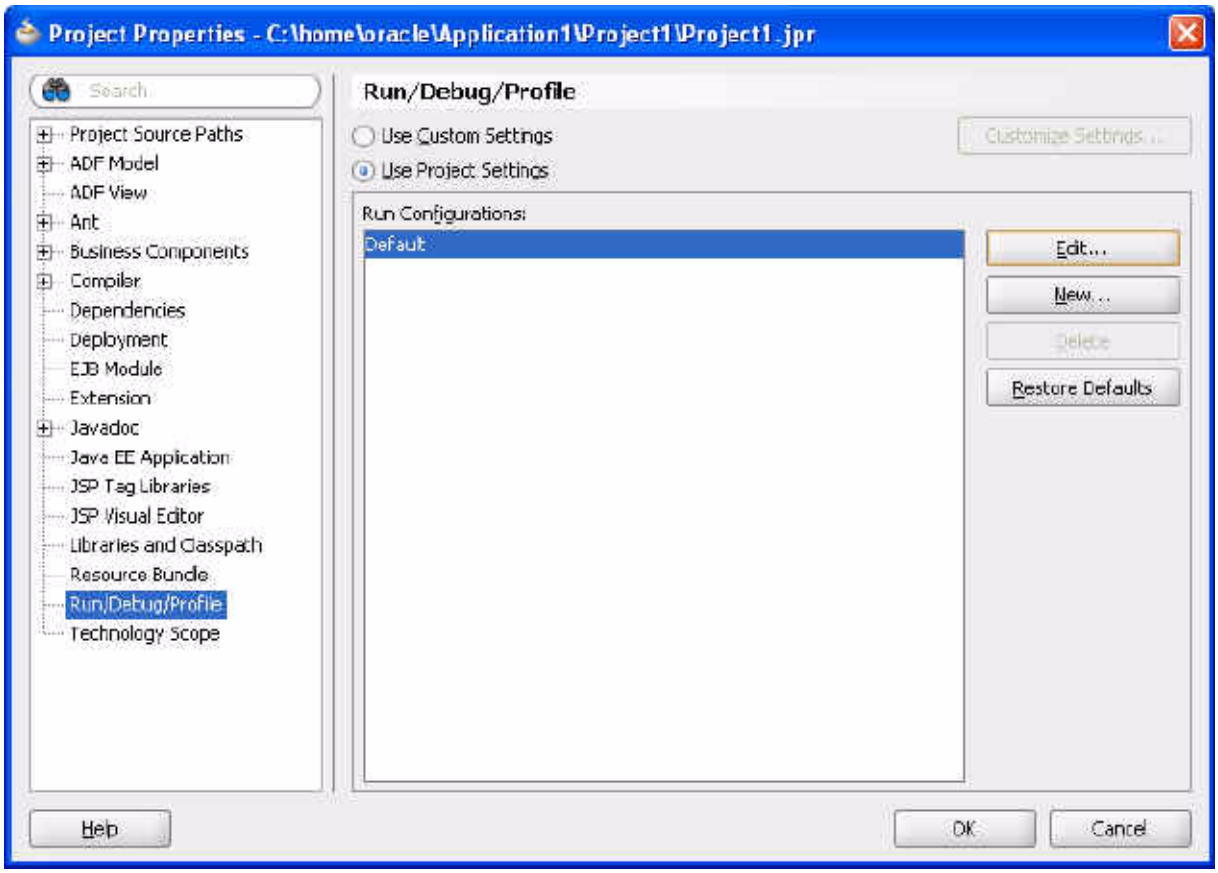


Changing Project Properties, Setting Runtime Configuration

To change the project's runtime properties in JDeveloper:

1. Right click the project and select **Project Properties.**
2. In the **Project Properties** dialog box select **Run/Debug/Profile** and click **Edit**.

Figure 2–15 Project Properties Dialog Box



3. You can set a number of values in the **Edit Run Configuration** dialog box:

   - Set **Virtual Machine** to **client**.
   - Set values for local storage and log level in the **Java Options** field.

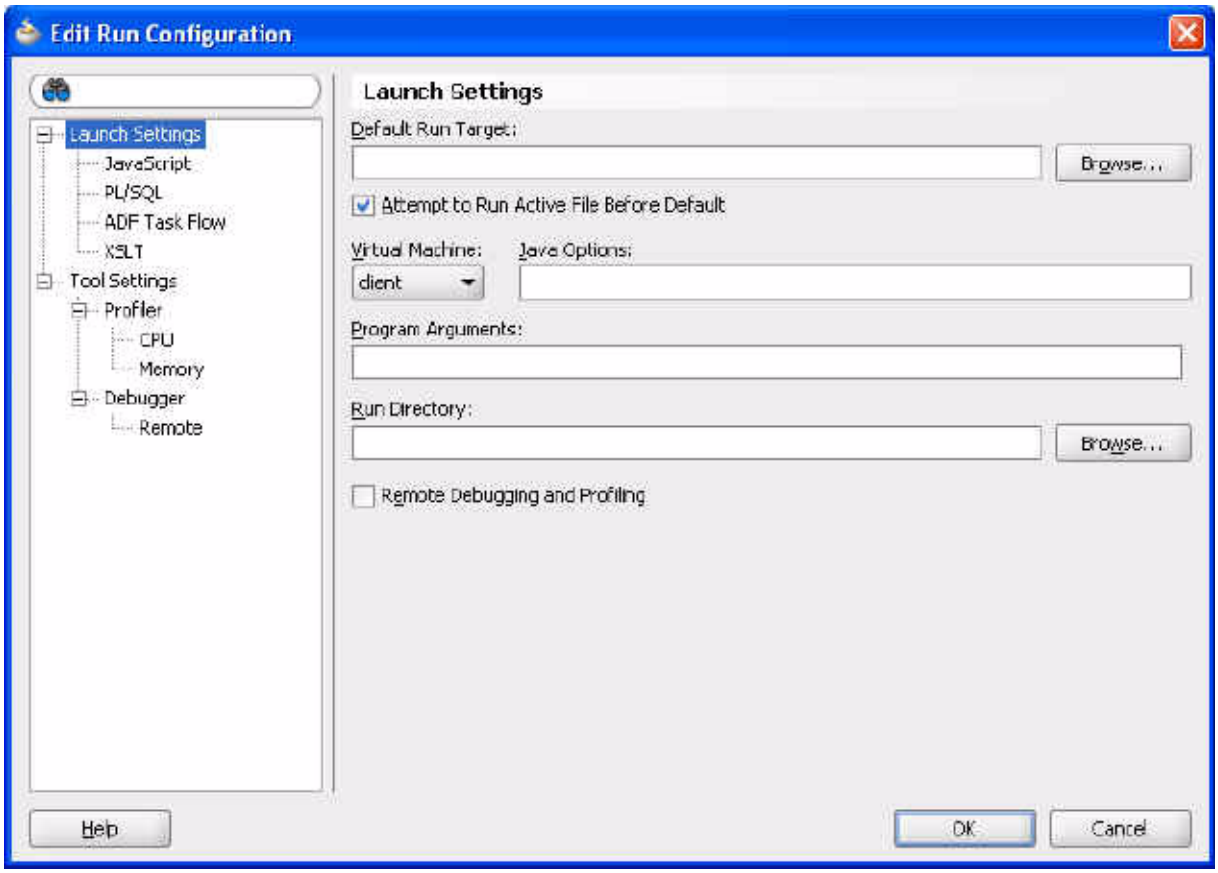
     By default, all processes start as storage-enabled. The process can store data as part of the cluster. You can modify the process so that it is not storage-enabled by using the system property `-Dtangosol.coherence.distributed.localstorage`. Setting the property to `false` specifies that the process is not storage-enabled.

     The `-Dtangosol.coherence.log.level` Java option sets the level of logging. For example:

     ```
     -Dtangosol.coherence.distributed.localstorage=false -Dtangosol.coherence.log.level=3
     ```

     Later chapters will describe when and how to set these system properties.

Figure 2–16 Setting the Runtime Configuration



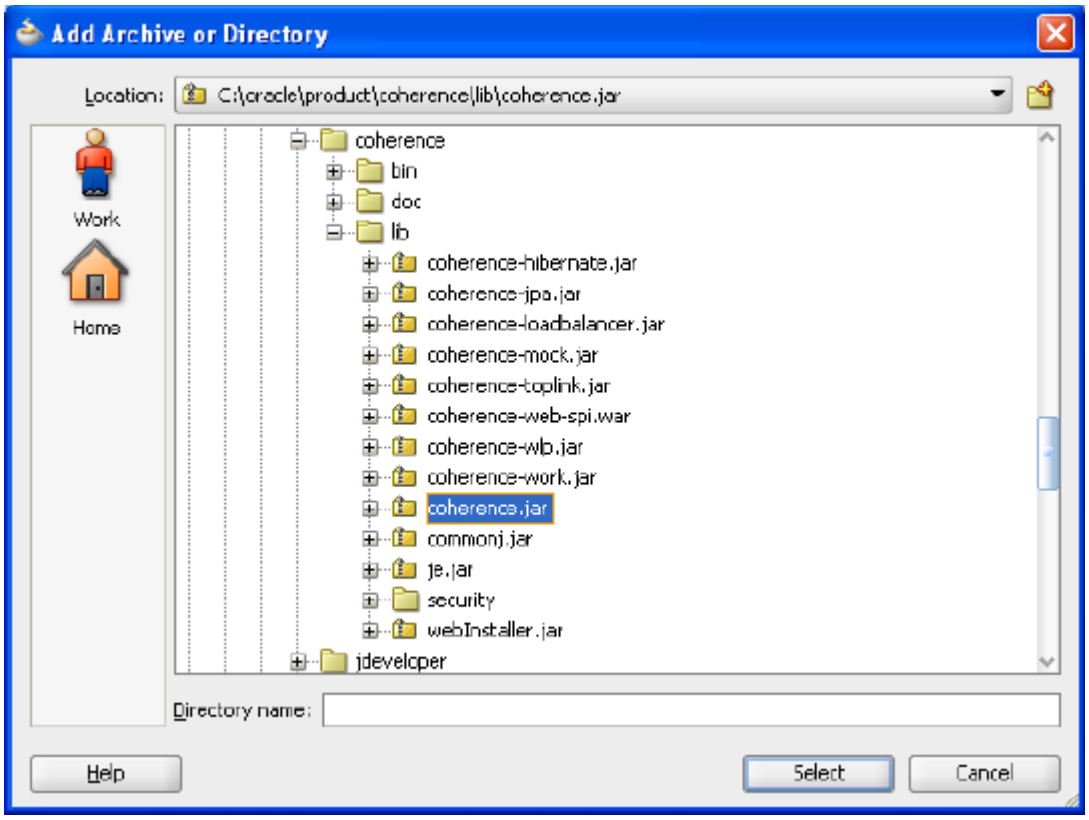
4. Click **OK** in the **Edit Run Configuration** dialog box and **OK** in the **Project Properties** dialog box to return to the JDeveloper IDE.

Adding JARS and Libraries to the Project Classpath

To add JAR files or libraries to the project classpath:

1. Right click the project and select **Project Properties.**
2. In the **Project Properties** dialog box select **Libraries and Classpaths** and click **Add JAR/Directory**.
3. Navigate to the JAR file or directory that you want to include in the classpath.

Figure 2–17 Adding JARS or Libraries to the Classpath



4. Click **Select** in the **Add Archive or Directory** dialog box. Click **OK** in the **Project Properties** dialog box.

3

Accessing the Data Grid from Java

In this exercise, you will develop a simple, Java console-based application to access, update, and remove simple types of information from a Coherence clustered cache. You also get familiar with using the Coherence Java APIs. Using JDeveloper, you will perform the following tasks:

- Create a new project
- Create a new `NamedCache`
- Put information into the cache and then retrieve it
- Retrieve information about the cache

This chapter has the following sections:

Introduction

All Coherence caches are named, have a lifetime scoped by the cluster instance in which they exist, and implement the `com.tangosol.net.NamedCache` interface. The `NamedCache` interface is an extension of `java.util.Map` and holds data and resources that are shared among the cluster members. Each `NamedCache` holds data as key/value pairs. Keys and values can be both simple and complex object types. The `NamedCache` interface provides extensions to the `Map` interface, such as locking and synchronization, storage integration, queries, event aggregations, and transactions. Table 3–1 describes some of the more commonly used methods within the `NamedCache` interface.

Table 3–1 Methods in the NamedCache Interface

Method Name	Description
`void clear()`	Removes all entries from the `NamedCache`.
`boolean containsKey(Object key)`	Returns `true` if `NamedCache` contains an entry for the key.
`boolean containsValue(Objectvalue)`	Returns `true` if there is at least one entry with this value in `NamedCache`.
`Object get(Object key)`	Gets the entry from `NamedCache` for that key.
`Object put(Object key,Objectvalue)`	Puts an object in the cache and returns the previous value (if any).

Table 3–1 (Cont.) Methods in the NamedCache Interface

Method Name	Description
`Object remove(Object key)`	Removes the mapping for this `key` from this map if present. Inherited from `ConcurrentMap`.
`Set entrySet()`	Returns a set of key/value pairs.
`Collection values()`	Gets all values back as a collection.
`CacheService getCacheService()`	Returns the `CacheService` that this `NamedCache` is a part of.

The `com.tangosol.net.CacheFactory` class is typically used to obtain an instance of a `NamedCache`. Table 3–2 describes some of the more commonly used methods in the `CacheFactory` class.

Table 3–2 Methods in the CacheFactory Class

Method Name	Description
`static Cluster ensureCluster()`	Obtains a cluster object running Coherence services.
`static void shutdown()`	Shuts down all clustered services.
`static NamedCache getCache(String cache)`	Returns an instance of a cache. Either joins an existing cache in the cluster or creates the cache if this is the first member.

For a full list of methods in the `NamedCache` interface and the `CacheFactory` class, see the Javadoc in `C:\oracle\product\coherence\doc`.

Creating Your First Coherence-Based Java Program

This section describes how to create a Java program that allows you to access, update, and remove simple types of information from a Coherence clustered cache.

1. Create a Program to Put Values in the Cache
2. Create a Program to Get Values from the Cache

Create a Program to Put Values in the Cache

Follow these steps to create a Coherence Java-based application.

1. Create a new project in JDeveloper. Name the project `InsertValue`. Ensure that directory is `C:\home\oracle\labs\InsertValue`.

   See "Creating a New Project in an Existing Application" on page 2-11 if you need detailed instructions.

2. Check the **Libraries and Classpath** value under **Project Properties** for the Coherence entry. Ensure that the full path is provided for the `coherence.jar`: `C:\oracle\product\coherence\lib\coherence.jar`
3. Create your first Coherence Java program. Name the class `MyFirstSample` and select the **Main Method** check box.

   See "Creating a Java Class" on page 2-13 if you need detailed information.

4. In the JDeveloper editor, write the code to create a `NamedCache`, put in a value, and then verify the value that you put in. Save the file after you finish coding. Example 3–1 illustrates a sample program.

Example 3–1 Creating a NamedCache; Inserting and Verifying Values

```
package com.oracle.handson;

import com.tangosol.net.CacheFactory;
import com.tangosol.net.NamedCache;

public class MyFirstSample {
    public MyFirstSample() {
    }

    public static void main(String[] args) {

       // create or get a named cache called mycache
       NamedCache myCache = CacheFactory.getCache("mycache");

       // put key, value pair into the cache.
       myCache.put("Name","Gene Smith");

       System.out.println("Value in cache is " + myCache.get("Name"));

    }
}
```

5. Stop any running cache servers.
6. Run the program in JDeveloper: right-click the `MyFirstSample.java` class in the editor and choose **Run**.
7. You should see messages similar to the following:

Figure 3–1 Output for Creating a NamedCache and Storing and Retrieving Values

```
Role=OracleHandsonMyFirstSample)
    )
  RecycleMillis=1200000
  RecycleSet=MemberSet(Size=0, BitSetCount=0
    )
  )

TcpRing{Connections=[]}
IpMonitor{AddressListSize=0}

2010-05-26 17:15:26.718/4.578 Oracle Coherence GE 3.6.0.0 DPR3 <D5>
(thread=Invocation:Management, member=1): Service Management joined the cluster with
service member 1
2010-05-26 17:15:27.000/4.860 Oracle Coherence GE 3.6.0.0 DPR3 <D5> (thread=Distribu
member=1): Service DistributedCache joined the cluster with senior service member 1
Value in cache is Gene Smith
Process exited with exit code 0.
```

Create a Program to Get Values from the Cache

Follow these steps to create a Java class which gets the value from your cache, instead of doing a put and then a get.

1. Create another Java class named `MyFirstSampleReader`. See "Creating a Java Class" on page 2-13 if you need detailed information. Example 3–2 illustrates a sample program.

Example 3–2 Getting a Value from the Cache

```
package com.oracle.handson;

import com.tangosol.net.CacheFactory;
import com.tangosol.net.NamedCache;

public class MyFirstSampleReader {

    public MyFirstSampleReader() {
    }

    public static void main(String[] args) {
        // ensure we are in a cluser
        CacheFactory.ensureCluster();

        // create or get a named cache called mycache
        NamedCache myCache = CacheFactory.getCache("mycache");

        System.out.println("Value in cache is " + myCache.get("Name"));
    }
}
```

2. Run the `MyFirstSampleReader` class. Figure 3–2 illustrates the output from the program. Note that a null value is returned. Although `MyFirstSample` successfully created and populated the `NamedCache`, the cache only existed within the `MyFirstSample` process memory. When `MyFirstSample` terminated so did the cache.

Figure 3–2 Output from the Sample Reader Program

```
2010-05-26 17:25:36.062/4.484 Oracle Coherence GE 3.6.0.0 DPR3 <D5>
(thread=Invocation:Management, member=1): Service Management joined the
service member 1
2010-05-26 17:25:36.140/4.562 Oracle Coherence GE 3.6.0.0 DPR3 <Info> (
member=1): Loaded cache configuration from
"jar:file:/C:/oracle/product/coherence/lib/coherence.jar!/coherence-cac
2010-05-26 17:25:36.312/4.734 Oracle Coherence GE 3.6.0.0 DPR3 <D5> (th
member=1): Service DistributedCache joined the cluster with senior serv
Value in cache is null
Process exited with exit code 0.
```

3. Start the `cache-server.cmd` that you used in "Testing a Coherence Installation" on page 1-2, run `MyFirstSample` to put the value in the `NamedCache`, and then run `MyFirstSampleReader` to read the value from the cache. Note the output illustrated in Figure 3–3: the "`Gene Smith`" value stored by `MyFirstSample` is returned by `MyFirstSampleReader`.

Figure 3–3 Output from the Sample Reader Program with a Running Cache Server

```
2010-05-26 17:30:49.203/2.344 Oracle Coherence GE 3.6.0.0 DPR3 <Info
member=3): Loaded cache configuration from
"jar:file:/C:/oracle/product/coherence/lib/coherence.jar!/coherence-
2010-05-26 17:30:49.390/2.531 Oracle Coherence GE 3.6.0.0 DPR3 <D5>
member=3): Service DistributedCache joined the cluster with senior s
Value in cache is Gene Smith
2010-05-26 17:30:49.437/2.578 Oracle Coherence GE 3.6.0.0 DPR3 <D4>
member=3): Asking member 1 for 128 primary partitions
2010-05-26 17:30:49.437/2.578 Oracle Coherence GE 3.6.0.0 DPR3 <D4>
member=3): ShutdownHook: stopping cluster node
2010-05-26 17:30:49.453/2.594 Oracle Coherence GE 3.6.0.0 DPR3 <D5>
member=3): Service Cluster left the cluster
2010-05-26 17:30:49.453/2.594 Oracle Coherence GE 3.6.0.0 DPR3 <D5>
member=3): Service DistributedCache left the cluster
Process exited with exit code 0.
```

This should not be the case for a process that joins the cluster only to perform an operation, such as putting in a value and then leaving, like `MyFirstSample`. By default, all processes start as storage-enabled. The process can store data as part of the cluster. You must modify the process so that it is not storage-enabled. Use the following Java command-line parameter.

```
-Dtangosol.coherence.distributed.localstorage=false
```

The Java option `-Dtangosol.coherence.distributed.localstorage` specifies whether a process is storage-enabled. Setting the parameter to `false` specifies that the process is not storage-enabled.

a. Change the project properties to disable local storage. See "Changing Project Properties, Setting Runtime Configuration" on page 2-15 for detailed information.

b. Enter the following value in the **Java Options** field:

```
-Dtangosol.coherence.distributed.localstorage=false
-Dtangosol.coherence.log.level=3
```

The Java property `tangosol.coherence.log.level=`*level* changes the level of logging produced by Coherence. The *level* can be set to a number between 0 and 9. The default is 5. A value of 0 means no logging., while a value of 9 means extremely verbose. A value of 3 is often useful enough for most application development.

4. Shut down any running cache servers and rerun your `MyFirstSample` class.

You receive a message similar to the one in Figure 3–4 indicating that storage is not enabled on the cluster, because you have set this member to be storage-disabled.

Figure 3–4 Output from JDeveloper if Storage is Disabled

```
Exception in thread "main" com.tangosol.net.RequestPolicyException: No
storage-enabled nodes exist for service DistributedCache
          at
com.tangosol.coherence.component.util.daemon.queueProcessor.service.grid.parti
tionedService.PartitionedCache$BinaryMap.onMissingStorage(PartitionedCache.CDB
:23)
          at
com.tangosol.coherence.component.util.daemon.queueProcessor.service.grid.parti
tionedService.PartitionedCache$BinaryMap.ensureRequestTarget(PartitionedCache.
CDB:39)
          at
com.tangosol.coherence.component.util.daemon.queueProcessor.service.grid.parti
tionedService.PartitionedCache$BinaryMap.put(PartitionedCache.CDB:24)
          at
com.tangosol.coherence.component.util.daemon.queueProcessor.service.grid.parti
tionedService.PartitionedCache$BinaryMap.put(PartitionedCache.CDB:1)
          at
com.tangosol.util.ConverterCollections$ConverterMap.put(ConverterCollections.j
ava:1578)
          at
com.tangosol.coherence.component.util.daemon.queueProcessor.service.grid.parti
tionedService.PartitionedCache$ViewMap.put(PartitionedCache.CDB:1)
          at
com.tangosol.coherence.component.util.SafeNamedCache.put(SafeNamedCache.CDB:1)
          at com.oracle.handson.MyFirstSample.main(MyFirstSample.java:16)
Process exited with exit code 1.
```

5. Restart the cache server and run `MyFirstSample` and `MyFirstSampleReader` again. You should now see that the data is persisted between running the two Java examples.

Creating Your First Coherence-Based Java Application

In this exercise, you develop a simple Java console-based application to access, update, and remove simple types of information from a Coherence clustered cache.

This exercise assumes you have completed "Testing a Coherence Installation" on page 1-2. Make sure you have a cache server and a cache client running.

Unlike client/server applications, in which client applications typically "connect" and "disconnect" from a server application, Coherence-based clustered applications simply ensure they are in a cluster, after which they may use the services of the cluster. Coherence-based applications typically do not "connect" to a cluster of applications; they become part of the cluster.

1. Create the Console Application
2. Run the Console Application

Create the Console Application

To create a Java console-based application to access, update, and remove simple types of information from a Coherence clustered cache:

1. Examine the methods in the `CacheFactory` class using the Coherence Java documentation (Javadoc) that is shipped in the `C:\oracle\product\coherence\doc` directory.
2. Develop a simple Java console application (Java class) called `YourFirstCoherenceApplication` that uses the `CacheFactory` class to join a cluster (using the `ensureCluster` method), and then leave the cluster (using the `shutdown` method). See "Creating a Java Class" on page 2-13 if you need detailed information on creating a Java class.
3. Examine the methods that are available in the `NamedCache` interface using the Javadoc.
4. Extend your application to use the `CacheFactory` method `getCache` to acquire a `NamedCache` for the cache called `mycache` (the same cache name used in the exercise: "Testing a Coherence Installation" on page 1-2).
5. With the `NamedCache` instance, use the "`get`" method to retrieve the value for the key "`message`" (the same key used in the exercise: "Testing a Coherence Installation" on page 1-2).
6. Output the value to the standard out using the `System.out.println(....)` method.

   Example 3–3 illustrates a sample Coherence-based Java application:

Example 3–3 Sample Coherence-Based Java Application

```
package com.oracle.handson;

import com.tangosol.net.CacheFactory;
import com.tangosol.net.NamedCache;

public class YourFirstCoherenceApplication {
public YourFirstCoherenceApplication() {
}

   public static void main(String[] args) {

      CacheFactory.ensureCluster();

      NamedCache myCache = CacheFactory.getCache("mycache");

      String message = (String)myCache.get("message");

      System.out.println(message);

      CacheFactory.shutdown();
YourFirstCoherenceApplication yourfirstcoherenceapplication = new
YourFirstCoherenceApplication();
   }

}
```

Run the Console Application

Follow these steps to run the Coherence application.

1. Ensure that you have the Coherence cache server and cache client running. In the cache client, connect to the `mycache` cache. For example:

   ```
   Map(?): cache mycache
   ```

2. Execute `YourFirstCoherenceApplication` from the JDeveloper IDE and view the result.

   Figure 3–5 illustrates the output from `YourFirstCoherenceApplication`. The output indicates that there is no data in the cache for the `message` key.

Figure 3–5 Output from Coherence-Based Java Application: No Value for the Key

```
TcpRing{Connections=[2]}
IpMonitor{AddressListSize=0}

2010-05-26 18:32:49.203/2.485 Oracle Coherence GE 3.6.0.0 DPR3
Loaded cache configuration from
"jar:file:/C:/oracle/product/coherence/lib/coherence.jar!/cohe
null
Process exited with exit code 0.
```

3. Using the running cache client, change the key "`message`." For example, enter

   ```
   Map <mycache>: put message "hello"
   ```

   Rerun `YourFirstCoherenceApplication` from the JDeveloper IDE to see the changed values. Figure 3–6 illustrates that the cache now holds the value `hello` for the key `message`.

Figure 3–6 Output from Coherence-Based Java Application: A Data Value for the Key

```
2010-05-26 18:37:06.359/2.453 Oracle Coherence GE 3.6.0.0 DPR3 <
Loaded cache configuration from
"jar:file:/C:/oracle/product/coherence/lib/coherence.jar!/cohere
hello
Process exited with exit code 0.
```

4. Rerun `YourFirstCoherenceApplication` with the following JVM parameter. Notice that the output is the same as the previous run.

   ```
   -Dtangosol.coherence.distributed.localstorage=false
   ```

5. Shut down your cache server and cache client instances. Restart the cache server and then rerun `YourFirstCoherenceApplication` (with the preceding JVM parameter set). Note the output is now `null`.

6. If you change the value of the `message` key in your application (using the `put` method), is the new value available through the cache client?

   a. For example, comment out the `get` method and add the `put` method.

      ```
      //String message = (String)myCache.get("message");
        String message = (String)myCache.put("message", "bye");
      ```

   b. Run `YourFirstCoherenceApplication`.

   c. Run the `get` command in the cache client.

      ```
      Map (mycache): get message
      ```

      You should see `bye` as the output.

4

Working with Complex Objects

In this chapter, you work with complex objects residing in the cache. Using JDeveloper, you create a new `Contact` class, and then store and retrieve `Contact` objects in the cache using POF serialization.

This chapter contains the following sections:

Introduction

Until now, you have been putting and getting `String` objects as the value in a `NamedCache`. Many of the implementations of the `get` and `put` methods in the Coherence Java API define the values and keys to be of type `Object`. For example:

```
public java.lang.Object get(java.lang.Object oKey)
public void put(java.lang.Object oKey, java.lang.Object oValue)
```

Any object can potentially be used as a value or key. This enables you to store complex objects as values in the cache.

Because Coherence might need to send the object across the wire, the object must be serializable. Object serialization is the process of saving an object's state into a sequence of bytes, and then rebuilding (deserializing) them into a live object at some future time. For example, objects that implement the `java.io.Serializable` interface are serializable.

As an alternative to using the Java `Serializable` interface, you can improve performance by using Coherence's own class for high-performance serialization, `com.tangosol.io.pof.PortableObject`. `PortableObject` is up to six times faster than the standard `java.io.Serializable` and the serialized result set is smaller.

The `PortableObject` interface provides two simple methods, `readExternal` and `writeExternal`, that permit you explicitly read and write serialized object attributes from the provided `PofReader` and `PofWriter` streams respectively. By taking control over the serialization format, Coherence provides a way to dramatically improve the performance of the process. Using POF dramatically reduces the size of the resulting binary. The size of the binary is often 5 to 10x smaller, and the conversion to-or-from the binary can be between 5 and 20 times faster, depending on the size of the object.

Creating and Caching Complex Objects

In this exercise, you will create a `Contact` object that contains names, addresses, dates of birth, and telephone numbers for employees. You will also use POF serialization to put the objects in the cache and retrieve them by implementing the `PortableObject` interface.

1. Create the Data Objects
2. Create the Complex Object
3. Create the Driver Class
4. Create the Configuration Files and Cache Server Executable
5. Run the Sample

Create the Data Objects

This section describes how to create two data objects that will later be incorporated into another data object. An `Address` object will provide employee address information and a `PhoneNumber` object will provide telephone contact information.

1. Create an `Address` object to store address information for an employee.

   a. Create a new project in JDeveloper called `Contacts`. See "Creating a New Project in an Existing Application" on page 2-11 if you need detailed information.

   b. Name the project `Contacts`. Ensure that the **Default Package** is `com.oracle.handson`, the **Java Source Path** is `C:\home\oracle\labs\Contacts\src` and the **Output Directory** is `C:\home\oracle\labs\Contacts\classes`.

      Check the **Project Properties** then **Libraries and Classpath** value for the Coherence entry. Ensure that the full path is provided for the `coherence.jar`: `C:\oracle\product\coherence\lib\coherence.jar`.

      In **Run/Debug/Profile**, edit the default configuration to disable local storage. Add the following line to the **Java Options** field:

      ```
      -Dtangosol.coherence.distributed.localstorage=false
      ```

   c. Create a new Java class called `Address`. See "Creating a Java Class" on page 2-13 if you need detailed information.

      Name the Java class `Address`. *Do not* select the **Main Method** check box.

   d. Write the class to use `PortableObject` for data serialization. In the JDeveloper code editor, change your generated `Address` class to implement `com.tangosol.io.pof.PortableObject`. Add an `import` statement for the `PortableObject` class.

   e. Import the `com.tangosol.io.pof.PofReader`, `com.tangosol.io.pof.PofWriter`, and `java.io.IOException` classes required by `PortableObject`.

   f. Add the default public constructor for `Address` that is required by `PortableObject`.

   g. Enter the following private attributes for your `Address` class. You can add others if you like.

      — `String Street1`

— String Street2

— String City

— String State

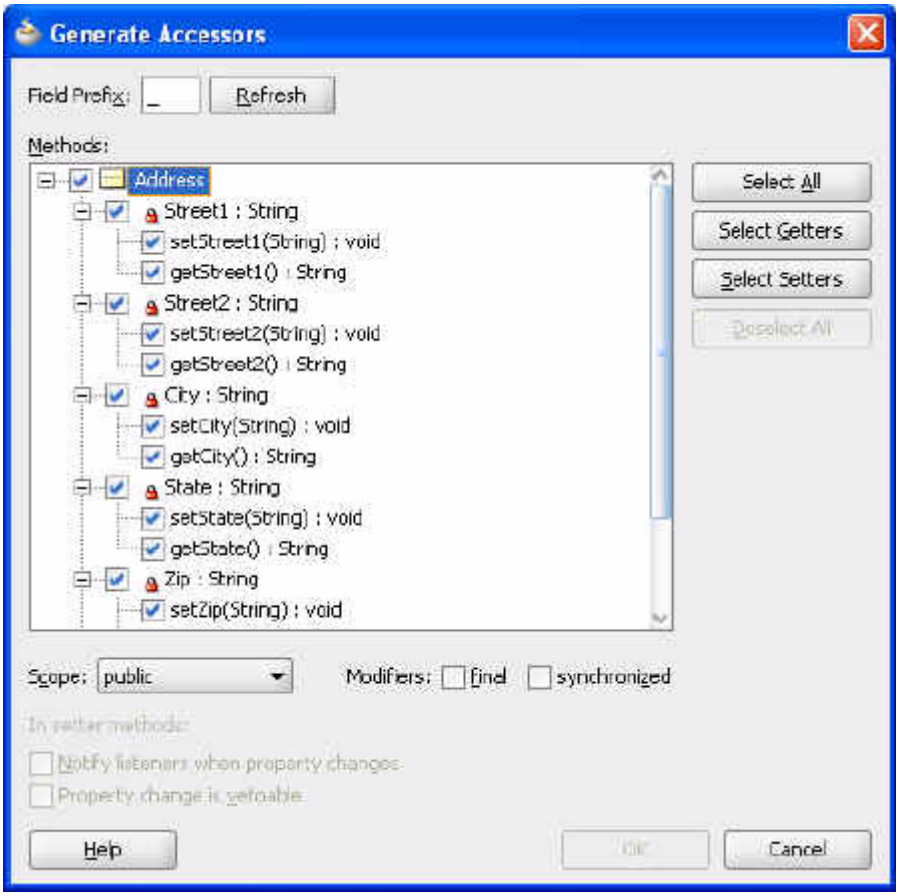
— String Country

At this point, the Address class should look similar to the following

```
 package com.oracle.handson;

import com.tangosol.io.pof.PofReader;
import com.tangosol.io.pof.PortableObject;
import com.tangosol.io.pof.PofWriter;

import java.io.IOException;

public class Address  implements PortableObject
{
           private String Street1;
           private String Street2;
           private String City;
           private String State;
           private String Zip;
           private String Country;
    /**
    * Default constructor (necessary for PortableObject implementation).
    */
    public Address()
        {
        }
}
```

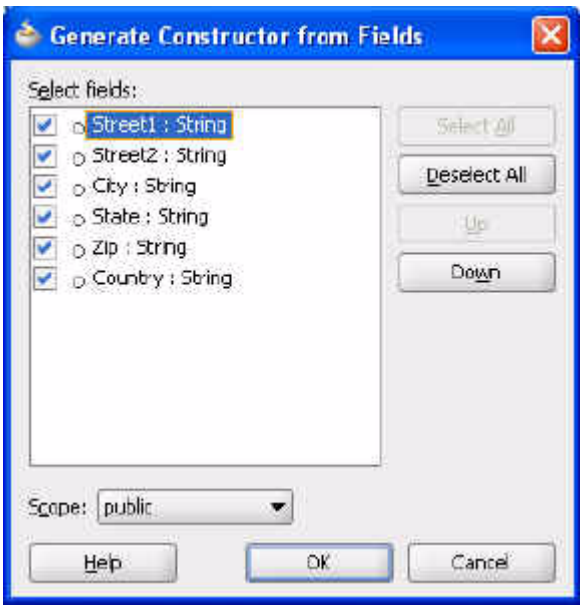
h. JDeveloper can generate the default get/set methods for your attributes. From the **Source** menu, select **Generate Accessors**. Select the check box next to the Address class. All of the attributes are now automatically selected. Click **OK** to continue.

Figure 4–1 Generate Accessors Dialog Box



i. You can also generate the default constructor and equals methods automatically. From the **Source** menu, select **Generate Constructor from Fields**, click **Select All**, and then click **OK**.

Figure 4–2 Generate Constructors Dialog Box



The generated constructor should look similar to the following:

```
    public Address(String Street1, String Street2, String City, String
State, String Zip, String Country) {
        super();
        this.Street1 = Street1;
        this.Street2 = Street2;
        this.City = City;
        this.State = State;
        this.Zip = Zip;
        this.Country = Country;
    }
```

j. Implement the `readExternal` and `writeExternal` methods as required by the `PortableObject` interface. For example, the following implementation of `readExternal` allows the values for the street, city, state, and country to be read as POF objects.

```
public void readExternal(PofReader reader)
        throws IOException
     {
     setStreet1(reader.readString(0));
     setStreet2(reader.readString(1));
     setCity(reader.readString(2));
     setState(reader.readString(3));
     setZip(reader.readString(4));
     setCountry(reader.readString(5));
}
```

k. Implement the `equals`, `hashCode`, and `toString` object methods.

Note: Cache keys and values must be serializable (for example, `java.io.Serializable`). Cache keys must also provide an implementation of the `hashCode()` and `equals()` methods, and those methods must return consistent results across cluster nodes. This implies that the implementation of `hashCode()` and `equals()` must be based solely on the object's serializable state (that is, the object's non-transient fields); most built-in Java types, such as `String`, `Integer` and `Date`, meet this requirement. Some cache implementations (specifically the partitioned cache) use the serialized form of the key objects for equality testing, which means that keys for which `equals()` returns true must serialize identically; most built-in Java types meet this requirement as well.

To support these methods, import the `com.tangosol.util.Base` and `com.tangosol.util.HashHelper` classes. The following code illustrates a sample implementation of `equals`:

```
    public boolean equals(Object oThat)
        {
        if (this == oThat)
            {
            return true;
            }
        if (oThat == null)
            {
            return false;
            }
```

```
        Address that = (Address) oThat;
        return Base.equals(getStreet1(), that.getStreet1()) &&
               Base.equals(getStreet2(), that.getStreet2()) &&
               Base.equals(getCity(),    that.getCity())    &&
               Base.equals(getState(),   that.getState())   &&
               Base.equals(getZip(),     that.getZip())     &&
               Base.equals(getCountry(), that.getCountry());
        }
```

The following code illustrates a sample implementation of `hashCode`:

```
    public int hashCode()
        {
        return HashHelper.hash(getStreet1(),
               HashHelper.hash(getStreet2(),
               HashHelper.hash(getZip(), 0)));
        }
```

The following code illustrates a sample implementation of `toString`:

```
    public String toString()
        {
        return  getStreet1() + "\n" +
                getStreet2() + "\n" +
                getCity() + ", " + getState() + " "  + getZip() + "\n" +
                getCountry();
        }
```

I. The resulting class should look similar to the following.

Example 4–1 Implementation of an Address Class

```
package com.oracle.handson;

import com.tangosol.io.pof.PofReader;
import com.tangosol.io.pof.PortableObject;
import com.tangosol.io.pof.PofWriter;

import com.tangosol.util.Base;
import com.tangosol.util.HashHelper;

import java.io.IOException;

public class Address  implements PortableObject
{
           private String Street1;
           private String Street2;
           private String City;
           private String State;
           private String Zip;
           private String Country;
    /**
    * Default constructor (necessary for PortableObject implementation).
    */
    public Address() {
        }

    public Address(String Street1, String Street2, String City, String State,
                   String Zip, String Country) {
        super();
```

```
        this.Street1 = Street1;
        this.Street2 = Street2;
        this.City    = City;
        this.State   = State;
        this.Zip     = Zip;
        this.Country = Country;
    }

    //------------ accessors--------------------------------

    public void setStreet1(String Street1) {
        this.Street1 = Street1;
    }

    public String getStreet1() {
        return Street1;
    }

    public void setStreet2(String Street2) {
        this.Street2 = Street2;
    }

    public String getStreet2() {
        return Street2;
    }

    public void setCity(String City) {
        this.City = City;
    }

    public String getCity() {
        return City;
    }

    public void setState(String State) {
        this.State = State;
    }

    public String getState() {
        return State;
    }

    public void setZip(String Zip) {
        this.Zip = Zip;
    }

    public String getZip() {
        return Zip;
    }

    public void setCountry(String Country) {
        this.Country = Country;
    }

    public String getCountry() {
        return Country;
    }
   // -------- PortableObject Interface------------------------------

    public void readExternal(PofReader reader)
```

```
            throws IOException
        {
        setStreet1(reader.readString(0));
        setStreet2(reader.readString(1));
        setCity(reader.readString(2));
        setState(reader.readString(3));
        setZip(reader.readString(4));
        setCountry(reader.readString(5));
        }

    public void writeExternal(PofWriter writer)
            throws IOException
        {
        writer.writeString(0, getStreet1());
        writer.writeString(1, getStreet2());
        writer.writeString(2, getCity());
        writer.writeString(3, getState());
        writer.writeString(4, getZip());
        writer.writeString(5, getCountry());
        }
    // ----- Object methods ---------------------------------------------------


    public boolean equals(Object oThat)
        {
        if (this == oThat)
            {
            return true;
            }
        if (oThat == null)
            {
            return false;
            }

        Address that = (Address) oThat;
        return Base.equals(getStreet1(), that.getStreet1()) &&
               Base.equals(getStreet2(), that.getStreet2()) &&
               Base.equals(getCity(),    that.getCity())    &&
               Base.equals(getState(),   that.getState())   &&
               Base.equals(getZip(),     that.getZip())     &&
               Base.equals(getCountry(), that.getCountry());
        }


    public int hashCode()
        {
        return HashHelper.hash(getStreet1(),
               HashHelper.hash(getStreet2(),
               HashHelper.hash(getZip(), 0)));
        }


    public String toString()
        {
        return  getStreet1() + "\n" +
                getStreet2() + "\n" +
                getCity() + ", " + getState() + " "  + getZip() + "\n" +
                getCountry();
        }
}
```

2. Create a `PhoneNumber` class to store telephone contact data.

   a. Create a new Java class called `PhoneNumber`. *Do not* include a `main` method.

      See "Creating a Java Class" on page 2-13 if you need detailed information.

   b. Use `PortableObject` for data serialization. In the JDeveloper code editor, change your generated `PhoneNumber` class to implement `com.tangosol.io.pof.PortableObject`. Add an `import` statement for the `PortableObject` class.

   c. Import the `com.tangosol.io.pof.PofReader` and `com.tangosol.io.pof.PofWriter`, and `java.io.IOException` classes required by `PortableObject`.

   d. Add the default public constructor for `PhoneNumber` that is required by `PortableObject`.

   e. Enter the following private attributes for your `PhoneNumber` class. You can add others if you like.

      —`short AccessCode`

      —`short CountryCode`

      —`short AreaCode`

      —`int LocalNumber`

   f. JDeveloper can generate the default get/set methods for your attributes. From the **Source** menu, select **Generate Accessors**. Select the check box next to the `PhoneNumber` class. All of the attributes are now automatically selected. Click **OK** to continue.

   g. You can generate the default constructor automatically. From the **Source** menu, select **Generate Constructor from Fields**, click **Select All**, and then click **OK**.

      The generated constructor should look similar to the following:

      ```
      public PhoneNumber(short AccessCode, short CountryCode, short AreaCode,
                  int LocalNumber) {
          super();
          this.AccessCode = AccessCode;
          this.CountryCode = CountryCode;
          this.AreaCode = AreaCode;
          this.LocalNumber = LocalNumber;
          }
      ```

   h. Implement the `readExternal` and `writeExternal` methods as required by the `PortableObject` interface.

   i. Implement the `equals`, `hashCode` and `toString` object methods.

   j. The resulting class should look similar to the following.

Example 4–2 Implementation of a PhoneNumber Class

```
package com.oracle.handson;

import com.tangosol.io.pof.PofReader;
import com.tangosol.io.pof.PofWriter;
import com.tangosol.io.pof.PortableObject;
```

```
import com.tangosol.util.HashHelper;

import java.io.IOException;

public class PhoneNumber implements PortableObject
{

    private short AccessCode;
    private short CountryCode;
    private short AreaCode;
    private int LocalNumber;

    /**
    * Default constructor (necessary for PortableObject implementation).
    */
    public PhoneNumber() {
    }

    public PhoneNumber(short AccessCode, short CountryCode, short AreaCode,
                int LocalNumber) {
        super();
        this.AccessCode = AccessCode;
        this.CountryCode = CountryCode;
        this.AreaCode = AreaCode;
        this.LocalNumber = LocalNumber;
    }

    //------------ accessors--------------------------------

    public void setAccessCode(short AccessCode) {
        this.AccessCode = AccessCode;
    }

    public short getAccessCode() {
        return AccessCode;
    }

    public void setCountryCode(short CountryCode) {
        this.CountryCode = CountryCode;
    }

    public short getCountryCode() {
        return CountryCode;
    }

    public void setAreaCode(short AreaCode) {
        this.AreaCode = AreaCode;
    }

    public short getAreaCode() {
        return AreaCode;
    }

    public void setLocalNumber(int LocalNumber) {
        this.LocalNumber = LocalNumber;
    }

    public int getLocalNumber() {
        return LocalNumber;
    }
```

```
// -------- PortableObject Interface-----------------------------

public void readExternal(PofReader reader)
        throws IOException
    {
    setAccessCode(reader.readShort(0));
    setCountryCode(reader.readShort(1));
    setAreaCode(reader.readShort(2));
    setLocalNumber(reader.readInt(3));
    }


public void writeExternal(PofWriter writer)
        throws IOException
    {
    writer.writeShort(0, getAccessCode());
    writer.writeShort(1, getCountryCode());
    writer.writeShort(2, getAreaCode());
    writer.writeInt(3, getLocalNumber());
    }
// ----- Object methods ---------------------------------------------

/**
* {@inheritDoc}
*/
public boolean equals(Object oThat)
    {
    if (this == oThat)
        {
        return true;
        }
    if (oThat == null)
        {
        return false;
        }

    PhoneNumber that = (PhoneNumber) oThat;
    return getAccessCode()  == that.getAccessCode()  &&
           getCountryCode() == that.getCountryCode() &&
           getAreaCode()    == that.getAreaCode()    &&
           getLocalNumber() == that.getLocalNumber();
    }

/**
* {@inheritDoc}
*/
public int hashCode()
    {
    return HashHelper.hash(getAreaCode(),
           HashHelper.hash(getLocalNumber(), 0));
    }

/**
* {@inheritDoc}
*/
public String toString()
    {
    return "+" + getAccessCode() + " " + getCountryCode() + " "
               + getAreaCode()   + " " + getLocalNumber();
    }
```

```
}
```

Create the Complex Object

The `Contact` object will provide the name, address, and telephone information of employees by incorporating the `Address` and `PhoneNumber` data objects.

1. Create a new Java class called `Contact`. *Do not* include a `main` method.

   See "Creating a Java Class" on page 2-13 if you need more information.

2. Since the class will use `PortableObject` for data serialization, change your generated `Contact` class to implement `com.tangosol.io.pof.PortableObject` in the JDeveloper code editor. Add an `import` statement for the `PortableObject` class.

3. Import the `com.tangosol.io.pof.PofReader` and `com.tangosol.io.pof.PofWriter` and `java.io.IOException` classes required by `PortableObject`.

4. Add the default public constructor for `Contact` that is required by `PortableObject`.

5. Enter the following private attributes for your `Contact` class. You can add others if you like.

   - `String FirstName`
   - `String LastName`
   - `Address HomeAddress`
   - `Address WorkAddress`
   - `Map TelephoneNumbers`
   - `java.sql.Date BirthDate`

6. JDeveloper can generate the default get/set methods for your attributes. From the Source menu, select **Generate Accessors**. Select the check box next to the `Contact` class. All of the attributes are now automatically selected. Click **OK** to continue.

7. Create an accessor, `getAge`, to calculate the age of an employee:

   ```
   public int getAge()
           {
           return (int) ((System.currentTimeMillis() - BirthDate.getTime()) /
   MILLIS_IN_YEAR);
           }
   ```

8. Add a definition for `MILLIS_IN_YEAR`.

   ```
   public static final long MILLIS_IN_YEAR = 1000L * 60L * 60L * 24L * 365L;
   ```

9. You can generate the default constructor automatically. From the **Source** menu, select **Generate Constructor** from **Fields**, click **Select All**, and then click **OK**.

   The generated constructor should look similar to the following:

   ```
   public Contact(String FirstName, String LastName, Address HomeAddress,
                  Address WorkAddress, Map TelephoneNumbers, Date BirthDate) {
       super();
       this.FirstName = FirstName;
       this.LastName = LastName;
       this.HomeAddress = HomeAddress;
       this.WorkAddress = WorkAddress;
   ```

```
        this.TelephoneNumbers = TelephoneNumbers;
        this.BirthDate = BirthDate;
    }
```

10. Implement the `readExternal` and `writeExternal` methods as required by the `PortableObject` interface.
11. Implement the `equals`, `hashCode`, and `toString` object methods.
12. The resulting class should look similar to the following.

Example 4–3 Sample Contact Class

```
package com.oracle.handson;

import com.tangosol.io.pof.PortableObject;

import com.tangosol.io.pof.PofReader;
import com.tangosol.io.pof.PofWriter;

import java.io.IOException;

import java.sql.Date;

import java.util.Iterator;
import java.util.Map;

public class Contact implements PortableObject
    {

       private String FirstName;
       private String LastName;
       private Address HomeAddress;
       private Address WorkAddress;
       private Map TelephoneNumbers;
       private java.sql.Date BirthDate;

    // ----- constructors ---------------------------------------------------

    /**
    * Default constructor (necessary for PortableObject implementation).
    */
    public Contact()
        {
        }

    public Contact(String FirstName, String LastName, Address HomeAddress,
                   Address WorkAddress, Map TelephoneNumbers, Date BirthDate) {
        super();
        this.FirstName = FirstName;
        this.LastName = LastName;
        this.HomeAddress = HomeAddress;
        this.WorkAddress = WorkAddress;
        this.TelephoneNumbers = TelephoneNumbers;
        this.BirthDate = BirthDate;
    }

    // -------- accessors ------------------------------------------

    public void setFirstName(String FirstName) {
        this.FirstName = FirstName;
```

```
}

public String getFirstName() {
    return FirstName;
}

public void setLastName(String LastName) {
    this.LastName = LastName;
}

public String getLastName() {
    return LastName;
}

public void setHomeAddress(Address HomeAddress) {
    this.HomeAddress = HomeAddress;
}

public Address getHomeAddress() {
    return HomeAddress;
}

public void setWorkAddress(Address WorkAddress) {
    this.WorkAddress = WorkAddress;
}

public Address getWorkAddress() {
    return WorkAddress;
}

public void setTelephoneNumbers(Map TelephoneNumbers) {
    this.TelephoneNumbers = TelephoneNumbers;
}

public Map getTelephoneNumbers() {
    return TelephoneNumbers;
}

public void setBirthDate(Date BirthDate) {
    this.BirthDate = BirthDate;
}

public Date getBirthDate() {
    return BirthDate;
}
/**
* Get age.
*
* @return age
*/
public int getAge()
    {
    return (int) ((System.currentTimeMillis() - BirthDate.getTime()) /
            MILLIS_IN_YEAR);
    }


// ----- PortableObject interface ---------------------------------------

/**
```

```
    * {@inheritDoc}
    */
    public void readExternal(PofReader reader)
            throws IOException
        {
        setFirstName(reader.readString(0));
        setLastName(reader.readString(1));
        setHomeAddress((Address) reader.readObject(2));
        setWorkAddress((Address) reader.readObject(3));
        setTelephoneNumbers(reader.readMap(4, null));
        setBirthDate(new Date(reader.readLong(5)));
        }

    /**
    * {@inheritDoc}
    */
    public void writeExternal(PofWriter writer)
            throws IOException
        {
        writer.writeString(0, getFirstName());
        writer.writeString(1, getLastName());
        writer.writeObject(2, getHomeAddress());
        writer.writeObject(3, getWorkAddress());
        writer.writeMap(4, getTelephoneNumbers());
        writer.writeLong(5, getBirthDate().getTime());
        }


    // ----- Object methods ---------------------------------------------------

    /**
    * {@inheritDoc}
    */
    public String toString()
        {
        StringBuffer sb = new StringBuffer(getFirstName())
                .append(" ")
                .append(getLastName())
                .append("\nAddresses")
                .append("\nHome: ").append(getHomeAddress())
                .append("\nWork: ").append(getWorkAddress())
                .append("\nTelephone Numbers");

        for (Iterator iter = TelephoneNumbers.entrySet().iterator();
             iter.hasNext(); )
            {
            Map.Entry entry = (Map.Entry) iter.next();
            sb.append("\n")
               .append(entry.getKey()).append(": ").append(entry.getValue());
            }
        return sb.append("\nBirth Date: ").append(getBirthDate()).toString();
        }
    /**
    * Approximate number of millis in a year ignoring things such as leap
    * years. Suitable for example use only.
    */
    public static final long MILLIS_IN_YEAR = 1000L * 60L * 60L * 24L * 365L;

}
```

Create the Driver Class

Create a driver class called `ContactDriver` to put `Contact` entries into the cache and retrieve them.

1. Create a new Java class called `ContactDriver` in the `Contacts` project. Ensure that it includes a `main` method.

   See "Creating a Java Class" on page 2-13 if you need detailed information.

2. In the `ContactDriver` file, create a new `NamedCache` called `contact` and put a new instance of the `Contact` object in it. Get the `Contact` object from the cache and ensure that the two objects are identical.

Example 4–4 Sample ContactDriver Class

```
package com.oracle.handson;

import com.tangosol.net.CacheFactory;
import com.tangosol.net.NamedCache;

import java.sql.Date;

import java.util.Map;
import java.util.HashMap;

public class ContactDriver {
    public ContactDriver() {
    }


    public static void main(String[] args) {
       NamedCache contact = CacheFactory.getCache("contact");

       Address homeAddress = new Address ("4157 Wash Ave", "Suite 4",
                                  "Burlingame", "CA", "94407", "USA");
       Address workAddress = new Address ("500 Oracle Pkwy", "MS989",
                                  "Redwood Shores", "CA", "94065", "USA");
       Date date = new Date(2009, 04, 01);
       PhoneNumber phonenumber = new PhoneNumber ((short)11, (short)650,
(short)506, 7000);
       Map map = new HashMap();
       map.put("home", phonenumber);

       Contact con1 = new Contact("Tom", "Dunn", homeAddress, workAddress,
                                    map, date);

       contact.put(con1.getFirstName(),con1);

       Contact con2 = (Contact)contact.get(con1.getFirstName());

       if (con2.getFirstName().equals(con1.getFirstName())) {
       System.out.println("They are the same!!");
       }
    }
}
```

Create the Configuration Files and Cache Server Executable

To use POF serialization, you must register your user-defined objects in a POF configuration file. The configuration associates the class of a user-defined object with a

numeric value. In addition, you must specify POF serialization and the name of the POF configuration file in the cache configuration file.

1. Create a POF configuration file for the `Contact`, `Address`, and `PhoneNumber` objects.

   To create a POF configuration file for your data types, use the `coherence-pof-config.xml` file as a model. You can find a copy of this file in the `coherence.jar`.

   Define `<user-type>` elements for the `Contact`, `Address`, and `PhoneNumber` objects, assign type IDs `1001`, `1002`, and `1003` to them and provide their full class names. The file should include the `coherence-pof-config.xml` file which reserves the first 1000 IDs for Coherence data types.

   Save the file as `contacts-pof-config.xml` in the `C:\home\oracle\labs` directory. Save a copy of the `coherence-pof-config.xml` file in the `labs` directory as well. Example 4–5 illustrates a sample `contacts-pof-config.xml` file.

Example 4–5 POF Configuration File

```
<?xml version="1.0"?>

<!DOCTYPE pof-config SYSTEM "pof-config.dtd">

<pof-config>
  <user-type-list>

    <!-- coherence POF user types -->
    <include>coherence-pof-config.xml</include>

    <!-- com.tangosol.examples package -->
    <user-type>
      <type-id>1001</type-id>
      <class-name>com.oracle.handson.Contact</class-name>
    </user-type>
    <user-type>
      <type-id>1002</type-id>
      <class-name>com.oracle.handson.Address</class-name>
    </user-type>
    <user-type>
        <type-id>1003</type-id>
      <class-name>com.oracle.handson.PhoneNumber</class-name>
    </user-type>
  </user-type-list>
  <allow-interfaces>true</allow-interfaces>
  <allow-subclasses>true</allow-subclasses>
</pof-config>
```

2. Create a cache configuration file. You can use the `coherence-cache-config.xml` file, which is based on the `cache-config.dtd` as a model. You can find a copy of `coherence-cache-config.xml` in the `coherence.jar`.

   Use `ExamplesPartitionedPocScheme` as the `scheme-name` and `PartitionedPofCache` as the `service-name`. The `serializer` section is responsible for mapping a POF serialized object to an appropriate serialization routine which is either a `PofSerializer` or by calling through the `PortableObject` interface. In this case, use the `com.tangosol.io.pof.`

ConfigurablePofContext class. The <init-param> section points to the name of the POF configuration file, in this case contacts-pof-config.xml.

Save the file as contacts-cache-config.xml in the C:\home\oracle\labs directory. Save a copy of the cache-config.dtd file in the labs directory as well. Example 4–6 illustrates a sample contacts-cache-config.xml file.

Example 4–6 Cache Configuration File

```
<?xml version="1.0"?>

<!DOCTYPE cache-config SYSTEM "cache-config.dtd">

<cache-config>
  <caching-scheme-mapping>
    <cache-mapping>
      <cache-name>*</cache-name>
      <scheme-name>ExamplesPartitionedPofScheme</scheme-name>
    </cache-mapping>
  </caching-scheme-mapping>

  <caching-schemes>
    <distributed-scheme>
      <scheme-name>ExamplesPartitionedPofScheme</scheme-name>
      <service-name>PartitionedPofCache</service-name>
      <serializer>
        <class-name>com.tangosol.io.pof.ConfigurablePofContext</class-name>
          <init-params>
            <init-param>
              <param-type>String</param-type>
              <param-value>contacts-pof-config.xml</param-value>
            </init-param>
          </init-params>
      </serializer>
      <backing-map-scheme>
        <local-scheme>
          <!-- each node will be limited to 250MB -->
          <high-units>250M</high-units>
          <unit-calculator>binary</unit-calculator>
        </local-scheme>
      </backing-map-scheme>
      <autostart>true</autostart>
    </distributed-scheme>
  </caching-schemes>
</cache-config>
```

3. Create an executable to start the cache server.

   Name the file contacts-cache-server.cmd and add the following information:

   - the location of the coherence.jar file to the classpath: %COHERENCE_HOME%\lib\coherence.jar
   - the location of the application class files to the classpath: C:\home\oracle\labs\Contacts\classes
   - the location of the POF configuration file to the classpath: C:\home\oracle\labs
   - the location of the server configuration with -Dtangosol.coherence.cacheconfig

- the command to start the default cache server: `com.tangosol.net.DefaultCacheServer`

Example 4–7 illustrates a sample `contacts-cache-server.cmd` executable file.

Example 4–7 Sample Cache Server Executable

```
@echo off
setlocal

if (%COHERENCE_HOME%)==() (
    set COHERENCE_HOME=c:\oracle\product\coherence
)

set CONFIG=c:\home\oracle\labs

set COH_OPTS=%COH_OPTS% -server -cp %COHERENCE_HOME%\lib\coherence.
jar;C:\home\oracle\labs\Contacts\classes;C:\home\oracle\labs;
set COH_OPTS=%COH_OPTS% -Dtangosol.coherence.cacheconfig=%CONFIG%\contacts-cache-
config.xml

java %COH_OPTS% -Xms1g -Xmx1g -Xloggc: com.tangosol.net.DefaultCacheServer %2 %3
%4 %5 %6 %7

:exit
```

Run the Sample

1. Compile the `Contacts` project if you have not done so already.

   Right-click the `Contacts` project and select **Make Contacts.jpr**.

2. Stop any cache servers that may be running.

3. Start the Contacts cache server by executing `contacts-cache-server.cmd` on the command line.

   Example 4–8 lists sample output from running `contacts-cache-server.cmd`.

Example 4–8 Starting the POF Cache Server

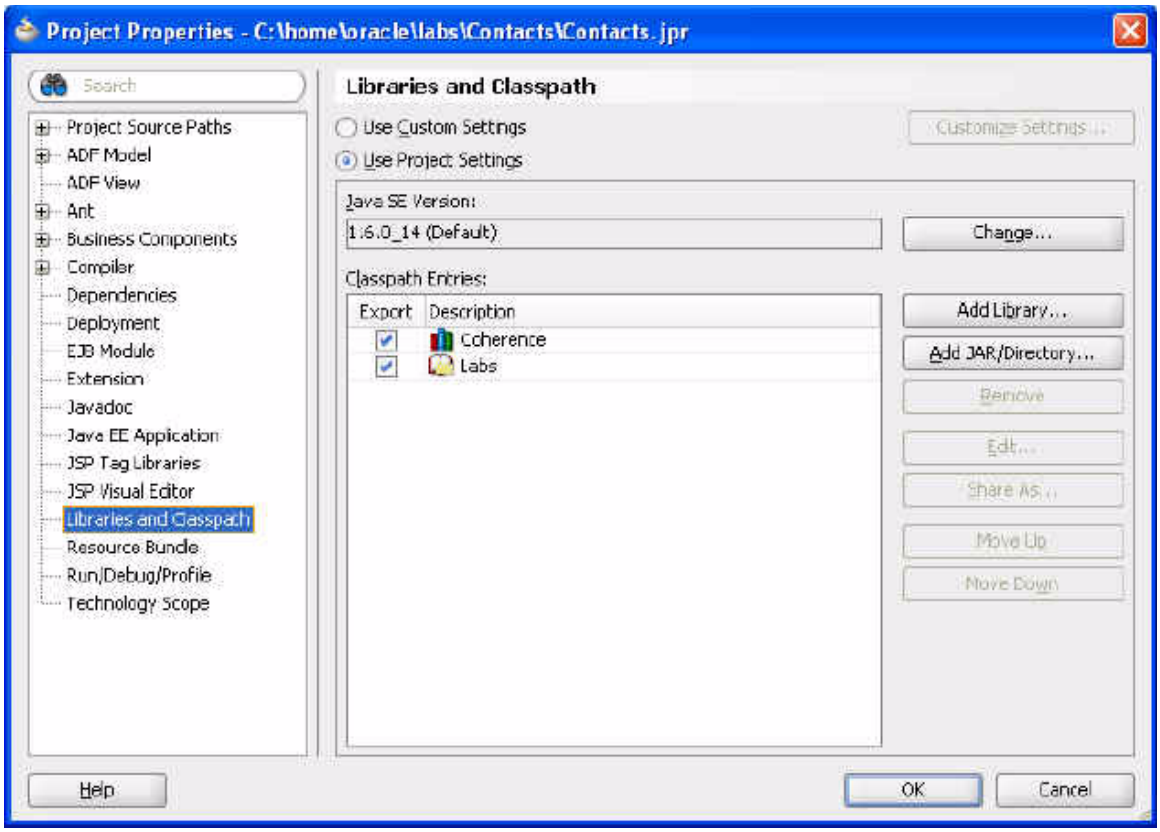
```
C:\oracle\product\coherence\bin>contacts-cache-server.cmd
2010-05-27 13:42:18.826/0.297 Oracle Coherence 3.6.0.0 DPR3 <Info> (thread=main, member=n/a):
Loaded operational configuration from "jar:file:/C:/oracle/product/coherence/lib/coherence.
jar!/tangosol-coherence.xml"
2010-05-27 13:42:18.826/0.297 Oracle Coherence 3.6.0.0 DPR3 <Info> (thread=main, member=n/a):
Loaded operational overrides from "jar:file:/C:/oracle/product/coherence/lib/coherence.
jar!/tangosol-coherence-override-dev.xml"
2010-05-27 13:42:18.826/0.297 Oracle Coherence 3.6.0.0 DPR3 <D5> (thread=main, member=n/a):
Optional configuration override "/tangosol-coherence-override.xml" is not specified
2010-05-27 13:42:18.842/0.313 Oracle Coherence 3.6.0.0 DPR3 <D5> (thread=main, member=n/a):
Optional configuration override "/custom-mbeans.xml" is not specified

Oracle Coherence Version 3.6.0.0 DPR3 Build 16141
 Grid Edition: Development mode
Copyright (c) 2000, 2010, Oracle and/or its affiliates. All rights reserved.

2010-05-27 13:42:19.092/0.563 Oracle Coherence GE 3.6.0.0 DPR3 <Info> (thread=main, member=n/a):
Loaded cache configuration from "file:/C:/home/oracle/labs/contacts-cache-config.xml"
2010-05-27 13:42:19.451/0.922 Oracle Coherence GE 3.6.0.0 DPR3 <D4> (thread=main, member=n/a):
SystemSocketProvider bound to port 8088
```

```
2010-05-27 13:42:22.904/4.375 Oracle Coherence GE 3.6.0.0 DPR3 <Info> (thread=Cluster, member=n/a):
Created a new cluster "cluster:0xC4DB" with Member(Id=1, Timestamp=2010-05-27 13:42:19.467,
Address=130.35.99.213:8088, MachineId=49877, Location=site:us.oracle.com,machine:tpfaeffl-
lap7,process:5200, Role=CoherenceServer, Edition=Grid Edition, Mode=Development, CpuCount=2,
SocketCount=1) UID=0x822363D500000128DB8051CBC2D51F98
2010-05-27 13:42:22.920/4.391 Oracle Coherence GE 3.6.0.0 DPR3 <Info> (thread=main, member=n/a):
Started cluster Name=cluster:0xC4DB

Group{Address=224.3.6.0, Port=36000, TTL=4}

MasterMemberSet
  (
  ThisMember=Member(Id=1, Timestamp=2010-05-27 13:42:19.467, Address=130.35.99.213:8088,
MachineId=49877, Location=site:us.oracle.com,machine:tpfaeffl-lap7,process:5200,
Role=CoherenceServer)
  OldestMember=Member(Id=1, Timestamp=2010-05-27 13:42:19.467, Address=130.35.99.213:8088,
MachineId=49877, Location=site:us.oracle.com,machine:tpfaeffl-lap7,process:5200,
Role=CoherenceServer)
  ActualMemberSet=MemberSet(Size=1, BitSetCount=2
    Member(Id=1, Timestamp=2010-05-27 13:42:19.467, Address=130.35.99.213:8088, MachineId=49877,
Location=site:us.oracle.com,machine:tpfaeffl-lap7,process:5200, Role=CoherenceServer)
    )
  RecycleMillis=1200000
  RecycleSet=MemberSet(Size=0, BitSetCount=0
    )
  )

TcpRing{Connections=[]}
IpMonitor{AddressListSize=0}

2010-05-27 13:42:23.029/4.500 Oracle Coherence GE 3.6.0.0 DPR3 <D5> (thread=Invocation:Management,
member=1): Service Management joined the cluster with senior service member 1
2010-05-27 13:42:23.279/4.750 Oracle Coherence GE 3.6.0.0 DPR3 <Info> (thread=Cluster, member=1):
Loaded POF configuration from "file:/C:/home/oracle/labs/contacts-pof-config.xml"
2010-05-27 13:42:23.295/4.766 Oracle Coherence GE 3.6.0.0 DPR3 <Info> (thread=Cluster, member=1):
Loaded included POF configuration from "jar:file:/C:/oracle/product/coherence/lib/coherence.
jar!/coherence-pof-config.xml"
2010-05-27 13:42:23.373/4.844 Oracle Coherence GE 3.6.0.0 DPR3 <D5>
(thread=DistributedCache:PartitionedPofCache, member=1): Service PartitionedPofCac
he joined the cluster with senior service member 1
2010-05-27 13:42:23.388/4.859 Oracle Coherence GE 3.6.0.0 DPR3 <Info> (thread=main, member=1):
Services
  (
  ClusterService{Name=Cluster, State=(SERVICE_STARTED, STATE_JOINED), Id=0, Version=3.6,
OldestMemberId=1}
  InvocationService{Name=Management, State=(SERVICE_STARTED), Id=1, Version=3.1, OldestMemberId=1}
  PartitionedCache{Name=PartitionedPofCache, State=(SERVICE_STARTED), LocalStorage=enabled,
PartitionCount=257, BackupCount=1, AssignedPartitions=257,
 BackupPartitions=0}
  )

Started DefaultCacheServer...
```

4. Set up the POF cache server to run in JDeveloper.

   Add additional `CLASSPATH` entries to the existing project properties. Navigate to **Tools** then **Project Properties** then **Libraries and Classpath**. Click **Add JAR/Directory**. In your project, add `C:\home\oracle\labs` to your `CLASSPATH`. Ensure that `coherence.jar` is in the classpath.

Figure 4–3 Adding Labs and Configuration Files to the Classpath



5. Click **Run/Debug/Profile** to edit the run-time properties. Ensure that local storage is disabled. Add the path to the `contacts-cache-config.xml` file in the **Java Options** field. The should contain the following:

   ```
   -Dtangosol.coherence.distributed.localstorage=false -Dtangosol.coherence.
   cacheconfig=c:\home\oracle\labs\contacts-cache-config.xml
   ```

   Click **OK** to save your changes to the run-time configuration and **OK** again to dismiss the **Project Properties** dialog box.

6. Run `ContactDriver.java` from the JDeveloper IDE to ensure that it works.

   In this example, the `Contact` object is converted to POF at run time. Using POF should provide significant performance improvements in terms of CPU time and the size of the binary generated.

Figure 4–4 Contacts Example Output Run from JDeveloper

```
2010-05-27 13:52:22.404/2.656 Oracle Coherence GE 3.6.0.0 DPR3 <Info>
(thread=Cluster, member=3): Loaded POF configuration from
"file:/C:/home/oracle/labs/contacts-pof-config.xml"
2010-05-27 13:52:22.435/2.687 Oracle Coherence GE 3.6.0.0 DPR3 <Info>
(thread=Cluster, member=3): Loaded included POF configuration from
"jar:file:/C:/oracle/product/coherence/lib/coherence.jar!/coherence-pof-config.xml
"
They are the same!!
Process exited with exit code 0.
```

5

Loading Data Into a Cache

In this chapter, you will learn how to populate a Coherence cache with domain objects that are read from text files.

This chapter contains the following sections:

Introduction

Until now, you have been putting and retrieving objects individually. Each `put` can result in increased network traffic, especially for partitioned and replicated caches. Additionally, each call to `put` returns the object it just replaced in the cache, which adds more unnecessary overhead. Loading the cache can be made much more efficient by using the `putAll` method instead.

This chapter assumes that you have completed "Creating and Caching Complex Objects" on page 4-2. It also assumes that you are familiar with using `java.io.BufferedReader` to read text files, `java.lang.String.split` method to parse text files, and `java.text.SimpleDateFormat` to parse dates.

Populating a Cache with Domain Objects

This exercise demonstrates how to create a console application that populates a Coherence cache with domain objects. The application can use the Coherence `com.tangosol.io.pof.PortableObject` implementation.

You will create a key that will help obtain the `Contact` object, a generator to provide data for the cache, and a loader to load the cache.

1. Create a Class with the Key for the Domain Objects
2. Edit the POF Configuration File
3. Create the Data Generator
4. Create a Console Application to Load the Cache
5. Edit the Cache Server Start-Up File
6. Run the Cache Loading Example

Create a Class with the Key for the Domain Objects

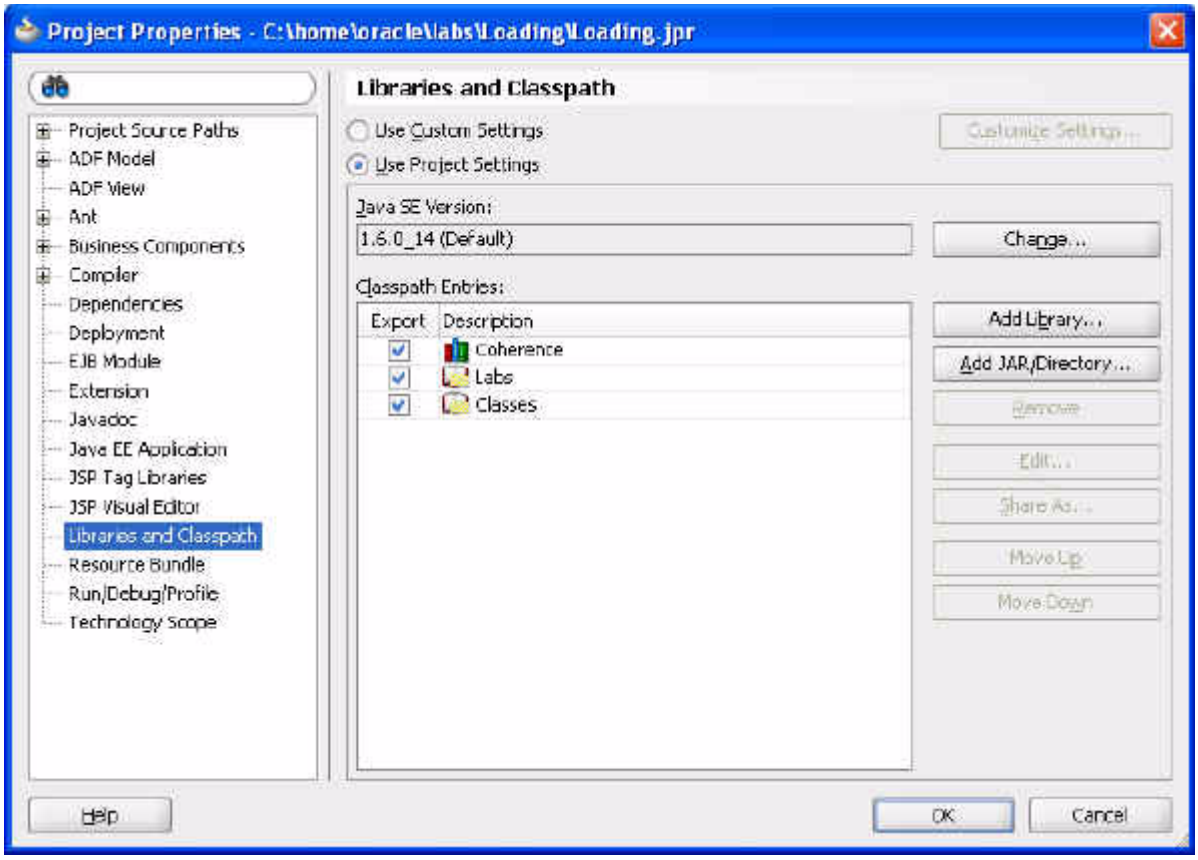
Follow these steps to create a class that contains the key for the a domain object:

1. Create a new project called `Loading`.

   See "Creating and Caching Complex Objects" on page 4-2 for information on creating a new project.

2. Add `coherence.jar` to the classpath. Also add the classes and files related to the `Address`, `PhoneNumber`, and `Contact` classes that you created in an earlier exercise (`Contacts`). These files can be found in `c:\home\oracle\labs` and `c:\home\oracle\labs\Contacts\classes`. (Hint: right-click the project and choose **Project Properties** then **Libraries and Classpath**. See "Adding JARS and Libraries to the Project Classpath" on page 2-17.)

Figure 5–1 Adding Contacts Directory to the Classpath



3. Create a contact ID class that provides a key to the employee for whom information is tracked. See "Creating a Java Class" on page 2-13 if you need detailed information on creating a Java class.

   Create the contact ID based on the employee's first name and last name. This object acts as the key to obtain the `Contact` object.

   Since this class will use POF serialization, it must implement `PortableObject` and provide implementations for the `writeExternal` and `readExternal` `PortableObject` methods and the `equals`, `hashCode`, and `toString` object methods.

Note: Cache keys and values must be serializable (for example, `java.io.Serializable`). Cache keys must also provide an implementation of the `hashCode()` and `equals()` methods, and those methods must return consistent results across cluster nodes. This implies that the implementation of `hashCode()` and `equals()` must be based solely on the object's serializable state (that is, the object's non-transient fields); most built-in Java types, such as String, Integer and Date, meet this requirement. Some cache implementations (specifically the partitioned cache) use the serialized form of the key objects for equality testing, which means that keys for which `equals()` returns true must serialize identically; most built-in Java types meet this requirement as well.

Example 5–1 illustrates a possible solution.

Example 5–1 Simple Contact ID Class

```
package com.oracle.handson;

import com.tangosol.io.pof.PofReader;
import com.tangosol.io.pof.PofWriter;
import com.tangosol.io.pof.PortableObject;

import com.tangosol.util.Base;
import com.tangosol.util.HashHelper;

import java.io.IOException;

/**
* ContactId is a key to the person for whom information is
* tracked.
*/
public class ContactId
        implements PortableObject
    {
    // ----- constructors ---------------------------------------------------

    /**
    * Default constructor (necessary for PortableObject implementation).
    */
    public ContactId() {
        }

    /**
    * Construct a contact person.
    *
    */
    public ContactId(String FirstName, String LastName)
        {
        super();
        this.FirstName = FirstName;
        this.LastName  = LastName;
        }

    // ----- accessors ------------------------------------------------------

    /**
```

```
* Return the first name.
*
*/
public String getFirstName()
    {
    return FirstName;
    }

/**
* Return the last name.
*
*/
public String getLastName()
    {
    return LastName;
    }

// ----- PortableObject interface ---------------------------------------

public void readExternal(PofReader reader)
        throws IOException
    {
    FirstName = reader.readString(0);
    LastName = reader.readString(1);
    }

public void writeExternal(PofWriter writer)
        throws IOException
    {
    writer.writeString(0, FirstName);
    writer.writeString(1, LastName);
    }

// ----- Object methods -------------------------------------------------

public boolean equals(Object oThat)
    {
    if (this == oThat)
        {
        return true;
        }
    if (oThat == null)
        {
        return false;
        }

    ContactId that = (ContactId) oThat;
    return Base.equals(getFirstName(), that.getFirstName()) &&
           Base.equals(getLastName(),  that.getLastName());
    }

public int hashCode()
    {
    return HashHelper.hash(getFirstName(),
           HashHelper.hash(getLastName(), 0));
    }

public String toString()
    {
    return getFirstName() + " " + getLastName();
```

```
        }

    // ----- data members ---------------------------------------------------

    /**
    * First name.
    */
    private String FirstName;

    /**
    * Last name.
    */
    private String LastName;
    }
```

Edit the POF Configuration File

Edit the POF configuration file. Add a `<user-type>` entry for `ContactID` to `contacts-pof-config.xml`. The file should look similar to Example 5–2.

Example 5–2 POF Configuration File with the ContactId Entry

```
<?xml version="1.0"?>

<!DOCTYPE pof-config SYSTEM "pof-config.dtd">

<pof-config>
  <user-type-list>

    <!-- coherence POF user types -->
    <include>coherence-pof-config.xml</include>

    <!-- com.tangosol.examples package -->
    <user-type>
      <type-id>1001</type-id>
      <class-name>com.oracle.handson.Contact</class-name>
    </user-type>
    <user-type>
      <type-id>1002</type-id>
      <class-name>com.oracle.handson.Address</class-name>
    </user-type>
    <user-type>
        <type-id>1003</type-id>
      <class-name>com.oracle.handson.PhoneNumber</class-name>
    </user-type>
    <user-type>
     <type-id>1004</type-id>
      <class-name>com.oracle.handson.ContactId</class-name>
    </user-type>
   </user-type-list>
  <allow-interfaces>true</allow-interfaces>
  <allow-subclasses>true</allow-subclasses>
</pof-config>
```

Create the Data Generator

Create a Java class named `DataGenerator` to generate random employee contact names and addresses. See "Creating a Java Class" on page 2-13 if you need detailed information.

Use the `Address`, `PhoneNumber`, and `Contact` classes that you created in an earlier exercise. Use `java.util.Random` to generate some random names, addresses, telephone numbers, and ages.

Example 5–3 illustrates a possible solution. This solution creates a text file, `contacts.cvs`, that contains the employee contact information. Store the file in the root of the project directory, in this case, `C:\home\oracle\labs\Loading`.

Example 5–3 Sample Data Generation Class

```
package com.oracle.handson;

import com.oracle.handson.Address;
import com.oracle.handson.Contact;
import com.oracle.handson.PhoneNumber;
import com.tangosol.util.Base;

import java.io.BufferedWriter;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.OutputStream;
import java.io.OutputStreamWriter;
import java.io.PrintWriter;

import java.sql.Date;

import java.util.Collections;
import java.util.Random;


/**
* DataGenerator is a generator of sample contacts.
*/
public class DataGenerator
    {
    // ----- static methods ---------------------------------------------------

    /**
    * Generate contacts.
    */
    public static void main(String[] asArg)
            throws IOException
        {
        String       sFile = asArg.length > 0 ? asArg[0] : FILENAME;
        int          cCon  = asArg.length > 1 ? Integer.parseInt(asArg[1]) : 1000;
        OutputStream out   = new FileOutputStream(sFile);

        generate(out, cCon);
        out.close();
        }

    /**
    * Generate the contacts and write them to a file.
    */
    public static void generate(OutputStream out, int cContacts)
            throws IOException
        {
        PrintWriter writer = new PrintWriter(new BufferedWriter(
                new OutputStreamWriter(out)));
```

```
    for (int i = 0; i < cContacts; ++i)
        {
        StringBuffer sb = new StringBuffer(256);

        //contact person
        sb.append("John,")
          .append(getRandomName())
          .append(',');

        // home and work addresses
        sb.append(Integer.toString(Base.getRandom().nextInt(999)))
          .append(" Beacon St.,,") /*street1,empty street2*/
          .append(getRandomName()) /*random city name*/
          .append(',')
          .append(getRandomState())
          .append(',')
          .append(getRandomZip())
          .append(",US,Yoyodyne Propulsion Systems,")
          .append("330 Lectroid Rd.,Grover's Mill,")
          .append(getRandomState())
          .append(',')
          .append(getRandomZip())
          .append(",US,");

        // home and work telephone numbers
        sb.append("home,")
          .append(Base.toDelimitedString(getRandomPhoneDigits(), ","))
          .append(",work,")
          .append(Base.toDelimitedString(getRandomPhoneDigits(), ","))
          .append(',');

        // random birth date in millis before or after the epoch
        sb.append(getRandomDateInMillis());

        writer.println(sb);
        }
    writer.flush();
    }

/**
* Return a random name.
*
*/
private static String getRandomName()
    {
    Random rand = Base.getRandom();
    int    cCh  = 4 + rand.nextInt(7);
    char[] ach  = new char[cCh];

    ach[0] = (char) ('A' + rand.nextInt(26));
    for (int of = 1; of < cCh; ++of)
        {
        ach[of] = (char) ('a' + rand.nextInt(26));
        }
    return new String(ach);
    }

/**
* Return a random phone muber.
* The phone number includes access, country, area code, and local
```

```
* number.
*
*/
private static int[] getRandomPhoneDigits()
    {
    Random rand = Base.getRandom();
    return new int[] {
        11,                  // access code
        rand.nextInt(99),    // country code
        rand.nextInt(999),   // area code
        rand.nextInt(9999999) // local number
        };
    }

/**
* Return a random Phone.
*
*/
private static PhoneNumber getRandomPhone()
    {
    int[] anPhone = getRandomPhoneDigits();

    return new PhoneNumber((short)anPhone[0], (short)anPhone[1],
            (short)anPhone[2], anPhone[3]);

    }
/**
* Return a random Zip code.
*
*/
private static String getRandomZip()
    {
    return Base.toDecString(Base.getRandom().nextInt(99999), 5);
    }

/**
* Return a random state.
*
*/
private static String getRandomState()
    {
    return STATE_CODES[Base.getRandom().nextInt(STATE_CODES.length)];
    }

/**
* Return a random date in millis before or after the epoch.
*
*/
private static long getRandomDateInMillis()
    {
    return (Base.getRandom().nextInt(40) - 20) * Contact.MILLIS_IN_YEAR;
    }

/**
* Generate a Contact with random information.
*
*/
public static Contact getRandomContact()
    {
    return new Contact("John",
```

```
                getRandomName(),
                new Address("1500 Boylston St.", null, getRandomName(),
                    getRandomState(), getRandomZip(), "US"),
                new Address("8 Yawkey Way", null, getRandomName(),
                    getRandomState(), getRandomZip(), "US"),
                Collections.singletonMap("work", getRandomPhone()),
                new Date(getRandomDateInMillis()));
        }

    // ----- constants ------------------------------------------------------

    /**
    * US Postal Service two letter postal codes.
    */
    private static final String[] STATE_CODES = {
            "AL", "AK", "AS", "AZ", "AR", "CA", "CO", "CT", "DE", "OF", "DC",
            "FM", "FL", "GA", "GU", "HI", "ID", "IL", "IN", "IA", "KS", "KY",
            "LA", "ME", "MH", "MD", "MA", "MI", "MN", "MS", "MO", "MT", "NE",
            "NV", "NH", "NJ", "NM", "NY", "NC", "ND", "MP", "OH", "OK", "OR",
            "PW", "PA", "PR", "RI", "SC", "SD", "TN", "TX", "UT", "VT", "VI",
            "VA", "WA", "WV", "WI", "WY"
        };
    /**
    * Default contacts file name.
    */
    public static final String FILENAME = "contacts.csv";
    }
```

Create a Console Application to Load the Cache

Create a Java class called `LoaderExample`. See "Creating a Java Class" on page 2-13 if you need detailed information.

The class will load the cache with employee data generated by the program in the previous section. Use input streams and buffered readers to load the entire employee information contained in the `contacts.csv` file into a single Coherence cache.

The application should contain the code to parse the employee information in the data file. Once you have this information, construct the individual contacts to put into the cache. To conserve processing effort and minimize network traffic, use the `putAll` method to load the cache.

Example 5–4 illustrates a possible solution.

Example 5–4 Sample Cache Loading Program

```
package com.oracle.handson;

import com.tangosol.net.CacheFactory;
import com.tangosol.net.NamedCache;

import com.oracle.handson.ContactId;
import com.oracle.handson.Address;
import com.oracle.handson.PhoneNumber;
import com.oracle.handson.Contact;

import java.io.BufferedReader;
import java.io.FileInputStream;
import java.io.IOException;
import java.io.InputStream;
import java.io.InputStreamReader;
```

```
import java.sql.Date;

import java.util.HashMap;
import java.util.Map;


/**
* LoaderExample loads contacts into the cache from a files.
*
*/
public class LoaderExample
    {
    // ----- static methods ---------------------------------------------------

    /**
    * Load contacts.
    */
    public static void main (String[] asArg)
            throws IOException
        {
        String sFile  = asArg.length > 0 ? asArg[0] : DataGenerator.FILENAME;
        String sCache = asArg.length > 1 ? asArg[1] : CACHENAME;

        System.out.println("input file: " + sFile);
        System.out.println("cache name: " + sCache);

        new LoaderExample().load(CacheFactory.getCache(sCache),
                new FileInputStream(sFile));
        CacheFactory.shutdown();
        }

    /**
    * Load cache from stream.
    *
    */
    public void load(NamedCache cache, InputStream in)
            throws IOException
        {
        BufferedReader reader   = new BufferedReader(new InputStreamReader(in));
        Map            mapBatch = new HashMap(1024);
        String         sRecord;
        int            cRecord  = 0;

        while ((sRecord = reader.readLine()) != null)
            {
            // parse record
            String[]      asPart     = sRecord.split(",");
            int           ofPart     = 0;
            String        sFirstName =  asPart[ofPart++];
            String        sLastName  =  asPart[ofPart++];
            ContactId          id    = new ContactId(sFirstName, sLastName);
            Address       addrHome   = new Address(
                                          /*streetline1*/ asPart[ofPart++],
                                          /*streetline2*/ asPart[ofPart++],
                                          /*city*/        asPart[ofPart++],
                                          /*state*/       asPart[ofPart++],
                                          /*zip*/         asPart[ofPart++],
                                          /*country*/     asPart[ofPart++]);
            Address       addrWork   = new Address(
```

```
                                    /*streetline1*/ asPart[ofPart++],
                                    /*streetline2*/ asPart[ofPart++],
                                    /*city*/        asPart[ofPart++],
                                    /*state*/       asPart[ofPart++],
                                    /*zip*/         asPart[ofPart++],
                                    /*country*/     asPart[ofPart++]);
            Map           mapTelNum  = new HashMap();

            for (int c = asPart.length - 1; ofPart < c; )
                {
                mapTelNum.put(/*type*/ asPart[ofPart++], new PhoneNumber(
                    /*access code*/  Short.parseShort(asPart[ofPart++]),
                    /*country code*/ Short.parseShort(asPart[ofPart++]),
                    /*area code*/    Short.parseShort(asPart[ofPart++]),
                    /*local num*/    Integer.parseInt(asPart[ofPart++])));
                }
            Date dtBirth = new Date(Long.parseLong(asPart[ofPart]));

            // Construct Contact and add to batch
            Contact con1 = new Contact(sFirstName, sLastName, addrHome,
                                       addrWork, mapTelNum, dtBirth);
            System.out.println(con1);
            mapBatch.put(id, con1);


            ++cRecord;
            if  (cRecord % 1024 == 0)
                {
                // load batch
                cache.putAll(mapBatch);
                mapBatch.clear();
                System.out.print('.');
                System.out.flush();
                }
            }

            if (!mapBatch.isEmpty())
               {
               // load final batch
               cache.putAll(mapBatch);
               }

        System.out.println("Added " + cRecord + " entries to cache");
        }


    // ----- constants ------------------------------------------------------

    /**
    * Default cache name.
    */
    public static final String CACHENAME = "ContactsCache";

    }
```

Edit the Cache Server Start-Up File

Edit the `contacts-cache-server.cmd` file to add the compiled classes for the `Loading` project (in this case, `C:\home\oracle\labs\Loading\classes`). Ensure

that the path to the directory where the cache configuration file is stored (in this case, `C:\home\oracle\labs`) is listed.

The `contacts-cache-server.cmd` file should look similar to Example 5–5:

Example 5–5 Sample contacts-cache-server.cmd File

```
@echo off
setlocal

if (%COHERENCE_HOME%)==() (
    set COHERENCE_HOME=c:\oracle\product\coherence
)


set COH_OPTS=%COH_OPTS% -server -cp %COHERENCE_HOME%\lib\coherence.
jar;C:\home\oracle\labs\Contacts\classes;C:\home\oracle\labs\Loading\classes;C:\ho
me\oracle\labs;
set COH_OPTS=%COH_OPTS% -Dtangosol.coherence.
cacheconfig=\home\oracle\labs\contacts-cache-config.xml

java %COH_OPTS% -Xms1g -Xmx1g -Xloggc: com.tangosol.net.DefaultCacheServer %2 %3
%4 %5 %6 %7

:exit
```

Run the Cache Loading Example

Follow these steps to run the cache loading example.

1. Add the path to the cache configuration to the **Java Options** field of the **Project Properties** then **Run/Debug/Profile**.

   ```
   -Dtangosol.coherence.cacheconfig=\home\oracle\labs\contacts-cache-config.xml
   ```

2. If you have not done so already, compile the Java files in the Loading project. Right-click the project in JDeveloper and select **Make Loading.jpr**.
3. Stop any running cache servers. Start a cache server with the `contacts-cache-server.cmd` file.
4. Run `DataGenerator` from JDeveloper, then run `LoaderExample`.

   A stream of employee contact information should appear in the JDeveloper output window. Figure 5–2 illustrates an example.

Figure 5–2 Output from the Sample Cache Loading Program

```
US
Telephone Numbers
work: +11 27 716 9267362
home: +11 27 663 6649767
Birth Date: 1957-01-03
John Hpcyzzo
Addresses
Home: 232 Beacon St.

Ghwu, OR 57133
US
Work: Yoyodyne Propulsion Systems
330 Lectroid Rd.
Grover's Mill, VT 52580
US
Telephone Numbers
work: +11 42 732 9769483
home: +11 88 479 591366
Birth Date: 1968-12-31
Added 1000 entries to cache
2010-05-28 11:14:15.223/5.468 Oracle Coherence GE 3.6.0.0 DPR3 <I
(thread=Invocation:Management, member=2): Service Management left
2010-05-28 11:14:15.286/5.531 Oracle Coherence GE 3.6.0.0 DPR3 <I
(thread=DistributedCache:PartitionedPofCache, member=2): 4> Trans
primary partitions to member 1 requesting 128
2010-05-28 11:14:15.395/5.640 Oracle Coherence GE 3.6.0.0 DPR3 <I
(thread=DistributedCache:PartitionedPofCache, member=2): Service
the cluster
2010-05-28 11:14:15.489/5.734 Oracle Coherence GE 3.6.0.0 DPR3 <I
member=2): Service Cluster left the cluster
Process exited with exit code 0.
```

Querying and Aggregating Data in the Cache

This exercise introduces the concept of querying and aggregating data in a cache. This exercise demonstrates how to:

- query the cache for specific data
- aggregate information within the cache

After putting complex objects in the named caches, you can query and aggregate information within the grid. The `com.tangosol.util.QueryMap` interface provides methods for managing the values or keys within a cache. You can use filters to restrict your results. You can also define indexes to optimize your queries.

Because Coherence serializes information when storing, you also have the overhead of deserializing when querying. When indexes are added, the *values* kept in the index itself are deserialized and therefore, offer quicker access time. The *objects* that are indexed are always kept serialized.

Some of the often-used methods in the `QueryMap` interface are:

- `Set entrySet(Filterfilter)`, which returns a set of entries that are contained in the map that satisfy the filter
- `addIndex(ValueExtractorextractor,booleanfOrdered, Comparator comparator)`, which adds an index

- `Set keySet(Filter filter)`, which is similar to `entrySet`, but returns keys, not values

It is important to note that filtering occurs at Cache Entry Owner level. In a partitioned topology, filtering can be performed in parallel because it is the primary partitions that do the filtering. The `QueryMap` interface uses the `Filter` classes. You can find more information on these classes in the API for the `com.tangosol.util.filter` package.

All Coherence `NamedCaches` implement the `com.tangosol.util.QueryMap` interface. This allows `NamedCaches` to support the searching for keys or entries in a cache that satisfy some condition. The condition can being represented as an object that implements the `com.tangosol.util.Filter` interface.

The `com.tangosol.util.filter` package contains several predefined classes that provide implementations of standard query expressions. Examples of these classes include `GreaterFilter`, `GreaterEquals`, `LikeFilter`, `NotEqualsFilter`, `InFilter`, and so on. You can use these filters to construct and compose object-based equivalents of most SQL `WHERE` clause expressions.

> **Note:** Coherence does not provide a `SQLFilter` because it is unlikely that the objects placed in a cache are modeled in a relational manner, that is, using rows and columns (as they are typically represented in a database). Additionally, it is common that objects placed in a cache are not easily modeled using relational models, for example, large blobs.

The `Filter` classes use standard Java method reflection to represent test conditions. For example, the following filter represents the condition where the value returned from the `getHomeAddress.getState` method on an object in a cache is for Massachusetts (`MA`):

```
(new EqualsFilter("getHomeAddress.getState", "MA");
```

If the object tested with this filter does not have a `get` method, then the test will fail.

A couple of examples will make things clearer:

- Return a set of people who live in a city whose name begins with the letter "`S`":

  ```
  Set sPeople = cache.entrySet(new LikeFilter("getHomeAddress.getCity","S%");
  ```

- Return a set containing people over age 42:

  ```
  final int nAge = 42;
  // Find all contacts who are older than nAge
  Set sSeniors = cache.entrySet(new GreaterFilter("getAge", nAge));
  ```

In addition to the `entrySet` and `keySet` methods defined by the `QueryMap` interface, Coherence supports the definition of indexes, using the `addIndex` method, to improve query performance. Unlike relational database systems, where indexes are defined according to well-known and strictly enforced collections of named columns (that is, a schema), Coherence does not have a schema. Though lacking a formal schema for data allows for significant flexibility and polymorphism, within applications, it means that an approach different from that of traditional database systems is required to define indexes and therefore, increase query performance.

To define the values that are to be indexed for each object placed in a cache, Coherence introduces the concept of a `ValueExtractor`. A `com.tangosol.util.`

`ValueExtractor` is a simple interface that defines an "`extract`" method. If given an object parameter, a `ValueExtractor` returns some value based on the parameter.

A simple example of a `ValueExtractor` implementation is the `com.tangosol.util.extractor.ReflectionExtractor`, which uses reflection to return the result of a method call on an object. For example:

```
new ReflectionExtractor("getCity")
```

`ValueExtractors` may be used throughout the Coherence API. Typically, however, they are used to define indexes.

An especially useful type of extractor is the `ChainedExtractor`. This is a composite `ValueExtractor` implementation based on an array of extractors. Extractors in the array are applied sequentially left-to-right, so a result of a previous extractor serves as a target object for a next one. For example:

```
new ChainedExtractor(new ReflectionExtractor("getHomeAddress"), new
ReflectionExtractor("getState"))
```

This example assumes that `HomeAddress` and `State` belong to a complex `Contact` object. `ChainedExtractor` first uses reflection to call `getHomeAddress` on each cached `Contact` object, and then uses reflection to call `getState` on the set of returned `HomeAddress`.

1. Create the Class to Query Cache Data
2. Run the Query Example
3. Edit the Query Example to Perform Aggregations
4. Run the Query and Aggregation Example

Create the Class to Query Cache Data

Create a new Java class called `QueryExample` to perform your queries. Ensure that the class has a `main` method. See "Creating a Java Class" on page 2-13 if you need detailed information.

Use the `entrySet` method to get the employee contact information for:

- all employees who live in Massachusetts. Hint:

  ```
  cache.entrySet(new EqualsFilter("getHomeAddress.getState", "MA"));
  ```

- all employees who live in Massachusetts and work elsewhere. Hint:

  ```
  cache.entrySet(new AndFilter(
           new EqualsFilter("getHomeAddress.getState", "MA"),
           new NotEqualsFilter("getWorkAddress.getState", "MA")));
  ```

- all employees whose city name begins with 'S'. Hint:

  ```
  cache.entrySet(new LikeFilter("getHomeAddress.getCity", "S%"));
  ```

- all employees with last name beginning with 'S' that live in Massachusetts. Use both key and value in the query. Note that the cache entries use `ContactId` objects as the keys. You can use the `KeyExtractor` to get these values. `KeyExtractor` is a specialized value extractor that indicates that a query should be run against the key objects, rather than the values. Hint:

  ```
  cache.entrySet(new AndFilter(
       new LikeFilter(new KeyExtractor("getLastName"), "S%",
  ```

```
            (char) 0, false),
         new EqualsFilter("getHomeAddress.getState", "MA")));
```

- all employees who are older than a specified age. Hint:

```
final int nAge = 42;
    setResults = cache.entrySet(new GreaterFilter("getAge", nAge));
```

Use indexes to improve performance. Hint: Find the `addIndex` method in the Javadoc for the `QueryMap` interface.

Example 5–6 illustrates a possible solution.

Example 5–6 Sample QueryExample Class

```
package com.oracle.handson;

import com.tangosol.net.CacheFactory;
import com.tangosol.net.NamedCache;

import com.tangosol.util.extractor.ChainedExtractor;
import com.tangosol.util.extractor.KeyExtractor;
import com.tangosol.util.extractor.ReflectionExtractor;

import com.tangosol.util.filter.AlwaysFilter;
import com.tangosol.util.filter.AndFilter;
import com.tangosol.util.filter.EqualsFilter;
import com.tangosol.util.filter.GreaterFilter;
import com.tangosol.util.filter.LikeFilter;
import com.tangosol.util.filter.NotEqualsFilter;

import java.util.Iterator;
import java.util.Set;


/**
* QueryExample runs sample queries for contacts.
*
*/
public class QueryExample{

    // ----- QueryExample methods ---------------------------------------

    public static void main(String[] args) {
      NamedCache cache = CacheFactory.getCache("ContactsCache");
       query(cache);
    }
    /**
    * Perform the example queries
    *
    */
     public static void query(NamedCache cache)
        {
         // Add indexes to make queries more efficient
        ReflectionExtractor reflectAddrHome =
                new ReflectionExtractor("getHomeAddress");

        // Add an index for the age
        cache.addIndex(new ReflectionExtractor("getAge"), true, null);

        // Add index for state within home address
```

```
        cache.addIndex(new ChainedExtractor(reflectAddrHome,
                new ReflectionExtractor("getState")), true, null);

// Add index for state within work address
        cache.addIndex(new ChainedExtractor(
                new ReflectionExtractor("getWorkAddress"),
                new ReflectionExtractor("getState")), true, null);

        // Add index for city within home address
        cache.addIndex(new ChainedExtractor(reflectAddrHome,
                new ReflectionExtractor("getCity")), true, null);

        // Find all contacts who live in Massachusetts
        Set setResults = cache.entrySet(new EqualsFilter(
                "getHomeAddress.getState", "MA"));
        printResults("MA Residents", setResults);

        // Find all contacts who live in Massachusetts and work elsewhere
        setResults = cache.entrySet(new AndFilter(
                new EqualsFilter("getHomeAddress.getState", "MA"),
                new NotEqualsFilter("getWorkAddress.getState", "MA")));
        printResults("MA Residents, Work Elsewhere", setResults);

        // Find all contacts whose city name begins with 'S'
        setResults = cache.entrySet(new LikeFilter("getHomeAddress.getCity",
                "S%"));
        printResults("City Begins with S", setResults);

        final int nAge = 42;
        // Find all contacts who are older than nAge
        setResults = cache.entrySet(new GreaterFilter("getAge", nAge));
        printResults("Age > " + nAge, setResults);

        // Find all contacts with last name beginning with 'S' that live
        // in Massachusetts. Uses both key and value in the query.
        setResults = cache.entrySet(new AndFilter(
                new LikeFilter(new KeyExtractor("getLastName"), "S%",
                               (char) 0, false),
                new EqualsFilter("getHomeAddress.getState", "MA")));
        printResults("Last Name Begins with S and State Is MA", setResults);

        }

    /**
    * Print results of the query
    *
    * @param sTitle      the title that describes the results
    * @param setResults  a set of query results
    */
    private static void printResults(String sTitle, Set setResults)
        {
        System.out.println(sTitle);
        for (Iterator iter = setResults.iterator(); iter.hasNext(); )
            {
            System.out.println(iter.next());
            }
        }
    }
```

Run the Query Example

Stop all running cache servers. Restart the `Contacts` cache server with `contacts-cache-server.cmd`. Run the `DataGenerator` file, the `LoadExample` file, then the `QueryExample` file. After printing all of the contact information in the cache, it displays the results of the queries. The results should look similar to Figure 5–3.

Figure 5–3 Output of the QueryExample Class

```
2010-05-28 11:34:26.567/2.781 Oracle Coherence GE 3.6.0.0
(thread=DistributedCache:PartitionedPofCache, member=3): A
partitions
MA Residents
ConverterEntry{Key="John Lpvjygfhhl", Value="John Lpvjygfh
Addresses
Home: 694 Beacon St.

Llvdfkggi, MA 45731
US
Work: Yoyodyne Propulsion Systems
330 Lectroid Rd.
Grover's Mill, MA 97721
US
Telephone Numbers
work: +11 84 361 6356198
home: +11 82 770 1954072
Birth Date: 1980-12-28"}
ConverterEntry{Key="John Nnodax", Value="John Nnodax
Addresses
Home: 177 Beacon St.

Jrktr, MA 74160
US
Work: Yoyodyne Propulsion Systems
330 Lectroid Rd.
Grover's Mill, ME 90077
US
Telephone Numbers
work: +11 36 492 2480598
```

Edit the Query Example to Perform Aggregations

Add code in `QueryExample.java` to perform aggregations on the data in the cache. An `EntryAggregator` (`com.tangosol.util.InvocableMap.EntryAggregator`) enables you to perform operations on all or a specific set of objects and return an aggregation. `EntryAggregators` are essentially agents that execute services in parallel against the data within the cluster. Aggregations are performed in parallel and can benefit from the addition of cluster members.

There are two ways of aggregating: aggregate over a collection of keys or by specifying a filter. Example 5–7 illustrates the methods that perform these aggregations.

Example 5–7 Methods to Aggregate Over Keys or by Specifying Filters

```
Object aggregate(Collection keys, InvocableMap.entryAggregator agg)

Object aggregate(Filter filter, InvocableMap.entryAggregator agg)
```

The following example uses a filter.

1. Using aggregations, write code in the `QueryExample` class to calculate the following:
   - the number of employees that are older than a specified age. Hint: use the `GreaterFilter` and the `Count` class:

     ```
     cache.aggregate(new GreaterFilter("getAge", nAge), new Count())
     ```

   - lowest age in the set of employees. Hint: use the `AlwaysFilter` and the `LongMin` class:

     ```
     cache.aggregate(AlwaysFilter.INSTANCE, new LongMin("getAge"))
     ```

   - highest age in the set of employees. Hint: use the `AlwaysFilter` and the `LongMax` class:

     ```
     cache.aggregate(AlwaysFilter.INSTANCE, new LongMax("getAge"))
     ```

   - average age of employees. Hint: use the `AlwaysFilter` and the `DoubleAverage` class:

     ```
     cache.aggregate(AlwaysFilter.INSTANCE, new DoubleAverage("getAge")
     ```

2. Import the `Count`, `DoubleAverage`, `LongMax`, and `LongMin` aggregator classes.

   ```
   import com.tangosol.util.aggregator.Count;
   import com.tangosol.util.aggregator.DoubleAverage;
   import com.tangosol.util.aggregator.LongMax;
   import com.tangosol.util.aggregator.LongMin;
   ```

   The `QueryExample.java` file should now look similar to Example 5–8:

Example 5–8 QueryExample with Aggregation

```
package com.oracle.handson;

import com.tangosol.net.CacheFactory;
import com.tangosol.net.NamedCache;

import com.tangosol.util.aggregator.Count;
import com.tangosol.util.aggregator.DoubleAverage;
import com.tangosol.util.aggregator.LongMax;
import com.tangosol.util.aggregator.LongMin;

import com.tangosol.util.extractor.ChainedExtractor;
import com.tangosol.util.extractor.KeyExtractor;
import com.tangosol.util.extractor.ReflectionExtractor;

import com.tangosol.util.filter.AlwaysFilter;
import com.tangosol.util.filter.AndFilter;
import com.tangosol.util.filter.EqualsFilter;
import com.tangosol.util.filter.GreaterFilter;
import com.tangosol.util.filter.LikeFilter;
import com.tangosol.util.filter.NotEqualsFilter;

import java.util.Iterator;
import java.util.Set;

/**
```

```
* QueryExample runs sample queries for contacts.
*/
public class QueryExample{

    // ----- QueryExample methods --------------------------------------

    public static void main(String[] args) {
      NamedCache cache = CacheFactory.getCache("ContactsCache");
       query(cache);
    }
    /**
    * Perform the example queries
    *
    */
     public static void query(NamedCache cache)
        {
         // Add indexes to make queries more efficient
        ReflectionExtractor reflectAddrHome =
                new ReflectionExtractor("getHomeAddress");

        cache.addIndex(new ReflectionExtractor("getAge"), true, null);
        cache.addIndex(new ChainedExtractor(reflectAddrHome,
                new ReflectionExtractor("getState")), true, null);
        cache.addIndex(new ChainedExtractor(
                new ReflectionExtractor("getWorkAddress"),
                new ReflectionExtractor("getState")), true, null);
        cache.addIndex(new ChainedExtractor(reflectAddrHome,
                new ReflectionExtractor("getCity")), true, null);

        // Find all contacts who live in Massachusetts
        Set setResults = cache.entrySet(new EqualsFilter(
                "getHomeAddress.getState", "MA"));
        printResults("MA Residents", setResults);

        // Find all contacts who live in Massachusetts and work elsewhere
        setResults = cache.entrySet(new AndFilter(
                new EqualsFilter("getHomeAddress.getState", "MA"),
                new NotEqualsFilter("getWorkAddress.getState", "MA")));
        printResults("MA Residents, Work Elsewhere", setResults);

        // Find all contacts whose city name begins with 'S'
        setResults = cache.entrySet(new LikeFilter("getHomeAddress.getCity",
                "S%"));
        printResults("City Begins with S", setResults);

        final int nAge = 42;
        // Find all contacts who are older than nAge
        setResults = cache.entrySet(new GreaterFilter("getAge", nAge));
        printResults("Age > " + nAge, setResults);

        // Find all contacts with last name beginning with 'S' that live
        // in Massachusetts. Uses both key and value in the query.
        setResults = cache.entrySet(new AndFilter(
                new LikeFilter(new KeyExtractor("getLastName"), "S%",
                               (char) 0, false),
                new EqualsFilter("getHomeAddress.getState", "MA")));
        printResults("Last Name Begins with S and State Is MA", setResults);

       // Count contacts who are older than nAge
        System.out.println("count > " + nAge + ": "+ cache.aggregate(
```

```
            new GreaterFilter("getAge", nAge), new Count()));

        // Find minimum age
        System.out.println("min age: " + cache.aggregate(AlwaysFilter.INSTANCE,
            new LongMin("getAge")));

        // Calculate average age
        System.out.println("avg age: " + cache.aggregate(AlwaysFilter.INSTANCE,
            new DoubleAverage("getAge")));

        // Find maximum age
        System.out.println("max age: " + cache.aggregate(AlwaysFilter.INSTANCE,
            new LongMax("getAge")));
        }

    /**
    * Print results of the query
    *
    */
    private static void printResults(String sTitle, Set setResults)
        {
        System.out.println(sTitle);
        for (Iterator iter = setResults.iterator(); iter.hasNext(); )
            {
            System.out.println(iter.next());
            }
        }
    }
```

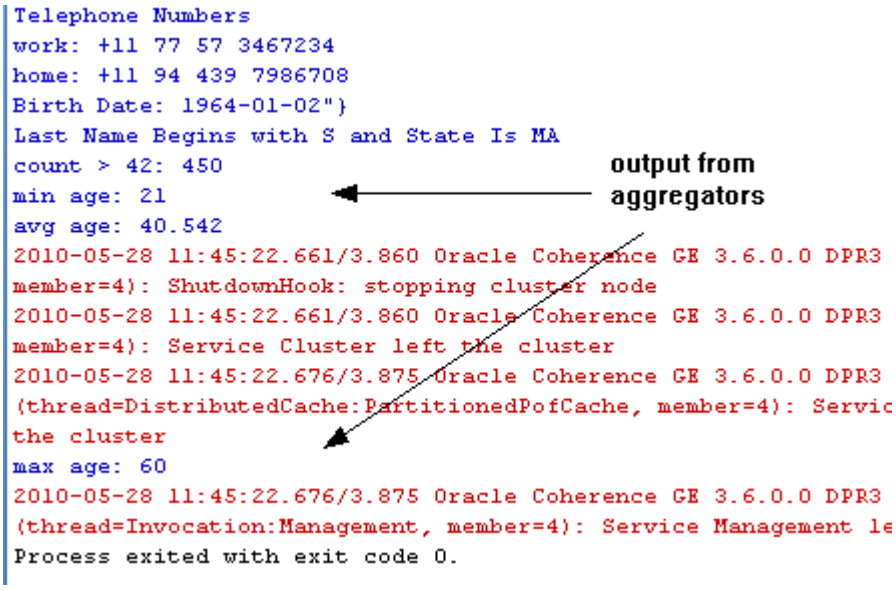
Run the Query and Aggregation Example

Follow these steps to query the cache and aggregate the results.

1. Stop the Contacts cache server and re-start it with the `contacts-cache-server.cmd` file.
2. Run the `DataGenerator`, `LoaderExample`, and `QueryExample` applications.

   You should see output similar to Figure 5–4 in JDeveloper.

Figure 5–4 Output from the Aggregators



6

Simplifying Cache Calls and Aggregations

In the previous chapter you created a file, `QueryExample.java`, that used a variety of filters, such as `AlwaysFilter`, `LikeFilter`, and `EqualsFilter`, to pull and aggregate information from the cache. The file also created indexes on the data by employing a variety of specialized `ValueExtractors`: `ChainedExtractors`, `KeyExtractors`, and `ReflectionExtractors`. The result was some very verbose Java statements. These statements can be simplified by using the `QueryHelper` API.

This chapter contains the following sections:

Introduction

To simplify filter and extractor statements, and the way in which you interact with the Coherence caches, the current release provides the `QueryHelper` API. `QueryHelper` (`com.tangosol.util.QueryHelper`) is a utility class that provides a set of `createFilter` and `createExtractor` factory methods that can build instances of `Filter` and `ValueExtractor`. The methods in the class accept a String that specifies the creation of rich Filters in a format that should be familiar to anyone that understands SQL WHERE clauses.

For example, the following statement uses `createFilter(String s)` to create a filter. It constructs a filter for employees who live in Massachusetts but work in another state.

```
..
QueryHelper.createFilter("homeAddress.state = 'MA' and workAddress.state !=
'MA'")
...
```

This statement is more simple than the equivalent filter/extractor statement using the Coherence API:

```
new AndFilter(new EqualsFilter("getHomeAddress.getState", "MA"),
      new NotEqualsFilter("getWorkAddress.getState", "MA"))
```

For more information, see the Javadoc for the `QueryHelper` API. For information on the syntax of the `WHERE` clause within the Coherence Query Language, see *Using Coherence Query Language* in the *Developer's Guide for Oracle Coherence*.

Simplifying the Query Example

This section describes how you can simplify the indexes, cache calls, and aggregations in the `QueryExample.java` file that you created in the previous chapter.

1. Import the `QueryHelper` API as a static class.

   ```
   import static com.tangosol.util.QueryHelper.*;
   ```

2. Comment out the imports for the `ChainedExtractor`, `KeyExtractor`, and `ReflectionExtractor` classes.
3. Comment out the imports for the `AlwaysFilter`, `AndFilter`, `EqualsFilter`, `GreaterFilter`, `LikeFilter`, and `NotEqualsFilter` classes.
4. In the `cache.addIndex` statements, replace instances of `ReflectionExtractor` with `createExtractor` from the QueryHelper API.

   Table 6–1 lists the `ReflectionExtractor` instances and their `createExtractor` equivalents.

Table 6–1 ReflectionExtractors and their Equivalent createExtractor Statements

Replace this ReflectionExtractor statement...	With the equivalent createExtractor statement
`cache.addIndex(new ReflectionExtractor("getAge"), true, null);`	`cache.addIndex(createExtractor("age"), true, null);`
`cache.addIndex(new ChainedExtractor(reflectAddrHome, new ReflectionExtractor("getState")), true, null);`	`cache.addIndex(createExtractor("homeAddress.state"), false, null);`
`cache.addIndex(new ChainedExtractor(new ReflectionExtractor("getWorkAddress"), new ReflectionExtractor("getState")), true, null);`	`cache.addIndex(createExtractor("workAddress.state"),false, null);`
`cache.addIndex(new ChainedExtractor(reflectAddrHome, new ReflectionExtractor("getCity")), true, null);`	`cache.addIndex(createExtractor("homeAddress.city"), true, null);`

5. Replace the calls to the `*Filter` methods in the `setResults` statements with calls to `createFilter` with the appropriate Coherence Query Language.

   Table 6–2 lists the `*Filter` instances and their `createFilter` equivalents.

Table 6–2 *Filter Statements and their Equivalent createFilter Statements in Queries

Replace this *Filter statement...	With the equivalent createFilter statement
`Set setResults = cache.entrySet(new EqualsFilter("getHomeAddress.getState", "MA"));`	`Set setResults = cache.entrySet(createFilter("homeAddress.state = 'MA'"));`
`Set setResults = cache.entrySet(new AndFilter(new EqualsFilter("getHomeAddress.getState", "MA"), new NotEqualsFilter("getWorkAddress.getState", "MA")));`	`setResults = cache.entrySet( createFilter("homeAddress.state is 'MA' and workAddress is not 'MA'"));`

*Table 6–2 (Cont.) *Filter Statements and their Equivalent createFilter Statements in Queries*

Replace this *Filter statement...	With the equivalent createFilter statement
`Set setResults = cache.entrySet(new LikeFilter("getHomeAddress.getCity", "S%"));`	`Set setResults = cache.entrySet(createFilter("homeAddress .city like 'S%'"));`

`Set setResults = cache.entrySet(new GreaterFilter("getAge", nAge));`	`// Initialize nAge and aEnv`
	`final int nAge = 42;`
	`Object[] aEnv = new Object[] {new Integer(nAge)};`
	`...`
	`Set setResults = cache.entrySet(createFilter("age > ?1",aEnv));`

`Set setResults = cache.entrySet(new AndFilter(new LikeFilter(new KeyExtractor("getLastName"), "S%", (char) 0, false), new EqualsFilter("getHomeAddress.getState", "MA")));`	`Set setResults = cache.entrySet(createFilter("lastName like 'S%' and homeAddress.state = 'MA'"));`

6. Replace the calls to the `*Filter` methods in the `aggregate` statements with calls to `createFilter` with the appropriate Coherence Query Language.

   Table 6–3 lists the `*Filter` instances and their `createFilter` equivalents.

Table 6–3 Filter Statements and their Equivalent createFilter Statements in Aggregations

Replace this *Filter statement...	With the equivalent createFilter statement
`System.out.println("count > " + nAge + ": "+ cache.aggregate(new GreaterFilter("getAge", nAge), new Count()));`	`System.out.println("count > " + nAge + ": " + cache.aggregate(createFilter("age > ?1", aEnv), new Count()));`
`System.out.println("min age: " + cache.aggregate(AlwaysFilter.INSTANCE, new LongMin("getAge")));`	`Filter always = createFilter("true");` `System.out.println("min age: " + cache.aggregate(always, new LongMin("getAge")));`
`System.out.println("avg age: " + cache.aggregate(AlwaysFilter.INSTANCE, new DoubleAverage("getAge")));`	`System.out.println("avg age: " + cache.aggregate(always, new DoubleAverage("getAge")));`
`System.out.println("max age: " + cache.aggregate(AlwaysFilter.INSTANCE, new LongMax("getAge")));`	`System.out.println("max age: " + cache.aggregate(always, new LongMax("getAge")));`

When you are finished with the code replacements, `QueryExample.java` should look similar to Example 6–1.

Example 6–1 Edited QueryExample File

```
package com.oracle.handson;

import com.tangosol.net.CacheFactory;
import com.tangosol.net.NamedCache;

import com.tangosol.util.Filter;
import static com.tangosol.util.QueryHelper.*;
```

```
import com.tangosol.util.aggregator.Count;
// import com.tangosol.util.extractor.ChainedExtractor;
// import com.tangosol.util.extractor.KeyExtractor;
// import com.tangosol.util.extractor.ReflectionExtractor;

// import com.tangosol.util.aggregator.Count;
import com.tangosol.util.aggregator.DoubleAverage;
import com.tangosol.util.aggregator.LongMax;
import com.tangosol.util.aggregator.LongMin;


// import com.tangosol.util.filter.AlwaysFilter;
// import com.tangosol.util.filter.AndFilter;
// import com.tangosol.util.filter.EqualsFilter;
// import com.tangosol.util.filter.GreaterFilter;
// import com.tangosol.util.filter.LikeFilter;
// import com.tangosol.util.filter.NotEqualsFilter;

import java.util.Iterator;
import java.util.Set;


/**
* QueryExample runs sample queries for contacts.
*
*/
public class QueryExample{

    // ----- QueryExample methods ---------------------------------------

    public static void main(String[] args) {
      NamedCache cache = CacheFactory.getCache("ContactsCache");
       query(cache);
    }
    /**
    * Perform the example queries
    *
    */
     public static void query(NamedCache cache)
        {
         // Add indexes to make queries more efficient
       // ReflectionExtractor reflectAddrHome =
       //         new ReflectionExtractor("getHomeAddress");

        // Add an index for the age
        // cache.addIndex(new ReflectionExtractor("getAge"), true, null);
        cache.addIndex(createExtractor("age"), true, null);

        // Add index for state within home address
        // cache.addIndex(new ChainedExtractor(reflectAddrHome,
        //        new ReflectionExtractor("getState")), true, null);
       cache.addIndex(createExtractor("homeAddress.state"), false, null);

        // Add index for state within work address
        // cache.addIndex(new ChainedExtractor(
        //        new ReflectionExtractor("getWorkAddress"),
        //        new ReflectionExtractor("getState")), true, null);
        cache.addIndex(createExtractor("workAddress.state"),false, null);
```

```
        // Add index for city within home address
        // cache.addIndex(new ChainedExtractor(reflectAddrHome,
        //        new ReflectionExtractor("getCity")), true, null);
        cache.addIndex(createExtractor("homeAddress.city"), true,  null);

        // Find all contacts who live in Massachusetts
        // Set setResults = cache.entrySet(new EqualsFilter(
        //         "getHomeAddress.getState", "MA"));
           Set setResults = cache.entrySet(createFilter("homeAddress.state =
'MA'"));
            printResults("MA Residents", setResults);

        // Find all contacts who live in Massachusetts and work elsewhere
        // setResults = cache.entrySet(new AndFilter(
        //        new EqualsFilter("getHomeAddress.getState", "MA"),
        //        new NotEqualsFilter("getWorkAddress.getState", "MA")));
            setResults = cache.entrySet(createFilter("homeAddress.state is 'MA'
and workAddress is not 'MA'"));
            printResults("MA Residents, Work Elsewhere", setResults);

        // Find all contacts whose city name begins with 'S'
        // setResults = cache.entrySet(new LikeFilter("getHomeAddress.getCity",
        //         "S%"));
           setResults = cache.entrySet(createFilter("homeAddress.city like
'S%'"));
            printResults("City Begins with S", setResults);

        final int nAge = 42;
        Object[] aEnv  =  new Object[] {new Integer(nAge)};
        // Find all contacts who are older than nAge
        // setResults = cache.entrySet(new GreaterFilter("getAge", nAge));
           setResults = cache.entrySet(createFilter("age > ?1",aEnv));
             printResults("Age > " + nAge, setResults);

        // Find all contacts with last name beginning with 'S' that live
        // in Massachusetts. Uses both key and value in the query.
        // setResults = cache.entrySet(new AndFilter(
        //     new LikeFilter(new KeyExtractor("getLastName"), "S%",
        //   (char) 0, false),
        //         new EqualsFilter("getHomeAddress.getState", "MA")));
           setResults = cache.entrySet(createFilter("lastName like 'S%' and
homeAddress.state = 'MA'"));
            printResults("Last Name Begins with S and State Is MA", setResults);

           // Count contacts who are older than nAge
           // System.out.println("count > " + nAge + ": "+
           //  cache.aggregate(new GreaterFilter("getAge", nAge), new Count()));
               System.out.println("count > " + nAge + ": " + cache.aggregate(
                    createFilter("age > ?1", aEnv), new Count()));

           // Find minimum age
           // System.out.println("min age: " +
cache.aggregate(AlwaysFilter.INSTANCE, new LongMin("getAge")));
             Filter always = createFilter("true");
             System.out.println("min age: " + cache.aggregate(always, new
LongMin("getAge")));

            // Calculate average age
            // System.out.println("avg age: " +
cache.aggregate(AlwaysFilter.INSTANCE, new DoubleAverage("getAge")));
```

```
              System.out.println("avg age: " + cache.aggregate(always, new
DoubleAverage("getAge")));

            // Find maximum age
            // System.out.println("max age: " +
            // cache.aggregate(AlwaysFilter.INSTANCE, new LongMax("getAge")));
              System.out.println("max age: " + cache.aggregate(always, new
LongMax("getAge")));

 System.out.println("------QueryLanguageExample completed------");



        }

    /**
    * Print results of the query
    *
    * @param sTitle      the title that describes the results
    * @param setResults  a set of query results
    */
    private static void printResults(String sTitle, Set setResults)
        {
        System.out.println(sTitle);
        for (Iterator iter = setResults.iterator(); iter.hasNext(); )
            {
            System.out.println(iter.next());
            }
        }
    }
```

Re-running the Query Example

Follow these steps to re-run the query example.

1. Stop all running cache servers.
2. Restart the Contacts cache server with `contacts-cache-server.cmd`.
3. Run the `DataGenerator`, `LoaderExample`, and `QueryExample` files.
4. After printing all of the contact information in the cache, it displays the results of the queries. The results should look similar to the following figures.

Figure 6–1 illustrates the output of the "MA Residents" filter.

Figure 6–1 Output of the "MA Residents" Filter

```
2010-05-28 12:01:37.567/2.844 Oracle Coherence GE 3.6.0.0 DPR3
(thread=DistributedCache:PartitionedPofCache, member=3): Askin
partitions
MA Residents
ConverterEntry{Key="John Abui", Value="John Abui
Addresses
Home: 772 Beacon St.

Vlxgmnxg, MA 65698
US
Work: Yoyodyne Propulsion Systems
330 Lectroid Rd.
Grover's Mill, KS 48514
US
Telephone Numbers
work: +11 79 927 6101324
home: +11 34 496 2216288
Birth Date: 1975-12-30"}
ConverterEntry{Key="John Sgbmt", Value="John Sgbmt
Addresses
Home: 665 Beacon St.
```

Figure 6–2 illustrates the output of the "MA Residents, Work Elsewhere" filter.

Figure 6–2 Output of the "MA Residents, Work Elsewhere" Filter

```
MA Residents, Work Elsewhere
ConverterEntry{Key="John Htljaoldf", Value="John Htljaoldf
Addresses
Home: 714 Beacon St.

Hblkgm, MA 89762
US
Work: Yoyodyne Propulsion Systems
330 Lectroid Rd.
Grover's Mill, ND 28424
US
Telephone Numbers
work: +11 50 339 7293147
home: +11 41 36 9741786
Birth Date: 1969-12-31"}
ConverterEntry{Key="John Tjsq", Value="John Tjsq
Addresses
Home: 901 Beacon St.

Puzoytermo, MA 18249
US
Work: Yoyodyne Propulsion Systems
330 Lectroid Rd.
Grover's Mill, LA 49379
US
Telephone Numbers
work: +11 70 313 3091289
home: +11 37 71 9404019
Birth Date: 1975-12-30"}
```

Figure 6–3 illustrates the output of the "City Begins with S" filter.

Figure 6–3 Output of the "City Begins with S" Filter

```
City Begins with S
ConverterEntry{Key="John Zirz", Value="John Zirz
Addresses
Home: 384 Beacon St.

Sryv, HI 76715
US
Work: Yoyodyne Propulsion Systems
330 Lectroid Rd.
Grover's Mill, NJ 86782
US
Telephone Numbers
work: +11 32 390 2424105
home: +11 38 983 7038618
Birth Date: 1983-12-28"}
ConverterEntry{Key="John Kuqslimeq", Value="John Kuqslimeq
Addresses
Home: 545 Beacon St.

Sojtotmdxg, AZ 43775
US
Work: Yoyodyne Propulsion Systems
330 Lectroid Rd.
Grover's Mill, MO 23117
US
Telephone Numbers
work: +11 70 376 99801
home: +11 91 214 77435
Birth Date: 1984-12-27"}
ConverterEntry{Key="John Mrkepp", Value="John Mrkepp
Addresses
Home: 649 Beacon St.

Swaiarwn, AK 63784
```

Figure 6–4 illustrates the output of the "Last Name Begins with S and State is MA" filter (empty set) and the output of the aggregators.

Figure 6–4 Output of the State and Age Aggregators

```
Last Name Begins with S and State Is MA
count > 42: 461
min age: 21
avg age: 40.724
max age: 60
------QueryLanguageExample completed------
Process exited with exit code 0.
```

7

Listening for Changes and Modifying Data

In this chapter, you will set up listeners to observe data changes within a `NamedCache`. You will also learn how `EntryProcessors` can be used to modify and process entries in the Coherence cache.

This chapter contains the following sections:

Introduction

The `com.tangosol.util.ObservableMap` interface enables you to observe and act on the changes made to cache entries. It extends `java.util.EventListener` and uses the standard bean event model within Java. All types of `NamedCaches` implement this interface. To listen for an event, you register a `MapListener` (`com.tangosol.util.MapListener`) on the cache. `MapListeners` are called on the client; that is, the listener code is executed in your client process.

There are three ways to listen for events:

- Listen for all events
- Listen for all events that satisfy a filter
- Listen for events on a particular object key

The methods listed in Example 7–1 (which can be implemented to perform the tasks in the preceding list) can be used on a `NamedCache`:

Example 7–1 Listener Methods on a NamedCache

```
void addMapListener(MapListener listener)

void addMapListener(MapListener listener, Filter filter, boolean fLite)

void addMapListener(MapListener listener, Object oKey, boolean fLite)
```

The `com.tangosol.util.MapEvent` class captures the object key, and the old and new values. You can specify a "Lite" event, in which the new and old values may not be present. Example 7–2 describes a pattern for registering these methods against a `NamedCache`. This has been done as an anonymous class. You can use the `getOldValue()` or `getNewValue()` methods in the `MapEvent` class to get the entry for which the event gets fired.

Example 7–2 Code Pattern for Registering an Event

```
namedCache.addMapListener(new MapListener() {
public void entryDeleted(MapEvent mapEvent) {
      // TODO... handle deletion event
}
public void entryInserted(MapEvent mapEvent) {
    // TODO... handle inserted event
}
public void entryUpdated(MapEvent mapEvent)
{
     // TODO... handle updated event } });
```

Creating a Cache Listener

This section describes how to create a Java class that listens on a `NamedCache` and responds to any changes it detects.

1. Create a Class to Listen for Changes in the Cache
2. Run the Cache Listener Example

Create a Class to Listen for Changes in the Cache

In the `Loading` project, create a new class that listens for a new `Contact` object entry. Name the class `ObserverExample` and ensure that it has a `main` method. See "Creating a Java Class" on page 2-13 if you need detailed information.

Within this class, add a listener to display a message whenever a new `Contact` is updated to the cache. (Hint: Use the following code to keep the Java process running until you read from the console; otherwise your program exits immediately.)

```
BufferedReader console = new BufferedReader(new InputStreamReader(System.in));
String text = console.readLine();
```

Example 7–3 illustrates a possible solution.

Example 7–3 Sample Listener Class

```
package com.oracle.handson;

import com.tangosol.net.NamedCache;

import com.tangosol.util.AbstractMapListener;
import com.tangosol.util.MapEvent;

import com.oracle.handson.Contact;

import com.tangosol.net.CacheFactory;

import java.io.IOException;


/**
 * ObserverExample observes changes to contacts.
 */
public class ObserverExample
    {

    public ObserverExample() {
```

```
}
// ----- ObserverExample methods ---------------------------------

public static void main(String[] args) {
  NamedCache cache = CacheFactory.getCache("ContactsCache");
  new ObserverExample().observe(cache);
    try {
        System.in.read();
    } catch (IOException e) {
    }
}
/**
* Observe changes to the contacts.
*
* @param cache  target cache
*/
public void observe(NamedCache cache)
    {
    cache.addMapListener(new ContactChangeListener());
    }

// ----- inner class: ContactChangeListener ------------------------

public class ContactChangeListener
        extends AbstractMapListener
    {
    // ----- MapListener interface -----------------------------------------

    public void entryInserted(MapEvent event)
        {
        System.out.println(event);
        }

    public void entryUpdated(MapEvent event)
        {
        Contact contactOld = (Contact)event.getOldValue();
        Contact contactNew = (Contact)event.getNewValue();
        StringBuffer sb  = new StringBuffer();

        if (!contactOld.getHomeAddress().equals(
                contactNew.getHomeAddress()))
            {
            sb.append("Home address ");
            }
        if (!contactOld.getWorkAddress().equals(
                contactNew.getWorkAddress()))
            {
            sb.append("Work address ");
            }
        if (!contactOld.getTelephoneNumbers().equals(
                contactNew.getTelephoneNumbers()))
            {
            sb.append("Telephone ");
            }
        if (contactOld.getAge() != contactNew.getAge())
            {
            sb.append("Birthdate ");
            }
        sb.append("was updated for ").append(event.getKey());
        System.out.println(sb);
```

```
        }

    public void entryDeleted(MapEvent event)
        {
        System.out.println(event.getKey());
        }
    }
}
```

Run the Cache Listener Example

Follow these steps to run the Cache Listener example.

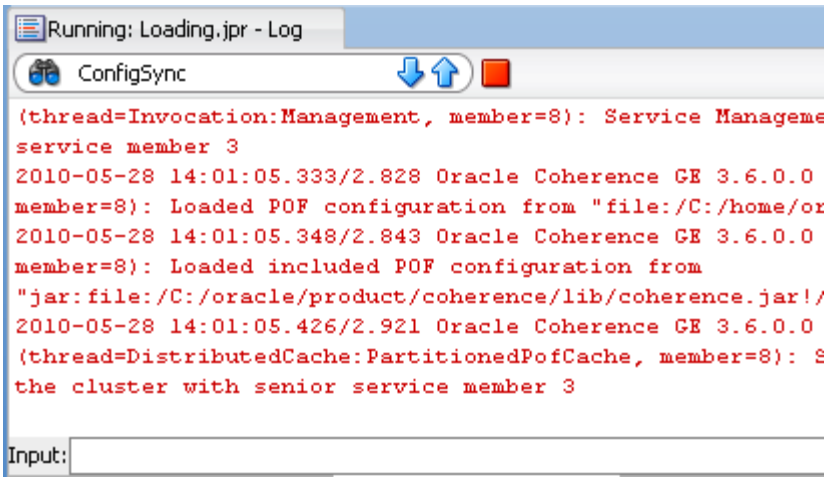
1. To enable the console input, edit the project profile:
   a. Right-click the `Loading` project and select **Project Properties**.
   b. Select **Run/Debug/Profile** at the left.
   c. Click the **Edit** button at the right and click **Tool Settings**. Ensure that the **Allow Program Input** check box in the **Edit Run Configuration** dialog box is selected.
   d. Click **OK** in the **Edit Run Configuration** dialog box and in the **Project Properties** dialog box to save your changes.
2. Turn off local storage if it is not already disabled:
   a. Right-click the `Loading` project and select **Project Properties**.
   b. Select **Run/Debug/Profile** at the left.
   c. Select the **Default** run configuration and click **Edit**. In **Java Options** field, add the command to disable local storage.

      ```
      -Dtangosol.coherence.distributed.localstorage=false
      ```

3. Check that the classes for the `Loading` project (`C:\home\oracle\labs\Loading\classes`) and the path to the configuration files (`C:\home\oracle\labs`) are present in the `contacts-cache-server.cmd` file.
4. Stop any running cache servers. Execute `contacts-cache-server.cmd` to start the cache server.
5. Load the cache by running the `LoaderExample` program from JDeveloper. If you now run the `ObserverExample`, you will see the program wait for input, as illustrated in Figure 7–1.

   In the next section, you will create a program that modifies entries in the cache and returns the changed records.

Figure 7–1 Listener Program Waiting for Events



Responding to Changes in the Cache

In this section, you will create a Java class to modify entries in the cache and return the changed records.

Until now, to perform actions on the entries in a cache, you used the put and get operations. However, there is a better way to perform operations on data that ensure consistent behavior when concurrent data access is required. `EntryProcessors` (`com.tangosol.util.InvocableMap.EntryProcessor`) are agents that perform processing against entries. The entries will be processed directly where the data is being held. The processing you perform may change the data: it may create, update, remove data, or only perform calculations. The processing can occur in parallel in a partitioned cache with multiple nodes, so it is scalable. Processing in the cache also saves the expense of I/O because data does not have to be retrieved to the client for processing.

`EntryProcessors` that work against the same key are logically queued. This allows lock-free (high performance) processing. The `com.tangosol.util.InvocableMap` interface (which the `NamedCache` implements) has the following methods for operating on data:

- `Object invoke(Object oKey, InvocableMap.EntryProcessor processor)`, which invokes the passed `EntryProcessor` against an individual object and returns the result of the invocation.
- `Map invokeAll(Collection keys, InvocableMap.EntryProcessor processor)`, which invokes the `EntryProcessor` against the collection of keys and returns the result for each invocation.
- `Map invokeAll(Filter filter, InvocableMap.EntryProcessor processor)`, which invokes the `EntryProcessor` against the entries that match the filter and returns the result for each invocation.

> **Note:** `EntryProcessor` classes must be available in the classpath for each cluster node.

To create an entry process, you can extend `com.tangosol.util.processes.AbstractProcessor` and implement the `process()` method. For example, the

following code creates an `EntryProcessor` to change the work address of employees in the `Contacts` data set:

```
public static class OfficeUpdater extends AbstractProcessor
        implements PortableObject
  ...
    public Object process(InvocableMap.Entry entry)  {
        Contact contact = (Contact) entry.getValue();
        contact.setWorkAddress(m_addrWork);
        entry.setValue(contact);
        return null;
        }
```

To invoke the `OfficeUpdater` class, you perform the following:

```
cache.invokeAll(new EqualsFilter("getHomeAddress.getState", "MA"),
             new OfficeUpdater(addrWork));
```

In this exercise, you will create a Java class with `EntryProcessors` that updates entries in the cache. The `ObserverExample` class created in the previous exercise will detect these changes and display the changed records.

1. Create a Class to Update Entries in the Cache
2. Edit the POF Configuration File
3. Run the Cache Update Example

Create a Class to Update Entries in the Cache

Follow these steps to create a file to update entries in the cache.

1. Create a class that updates entries in the cache.

   In the `Loading` project, create a class called `ProcessorExample` with a `main` method that updates the address of a `Contact` object in the cache. See "Creating a Java Class" on page 2-13 if you need detailed information.

2. Write code to find the records of `Contacts` that live in Massachusetts and update their work address to an in-state office.

   Hint: include an inner class that implements `PortableObject` (for serializing and deserializing data from the cache) and contains an `EntryProcessor` to set the work addresses. Use methods from the `Filter` class to isolate the `Contacts` whose home address is in Massachusetts.

   Example 7–4 illustrates possible code for the `ProcessorExample` class.

Example 7–4 Sample Program to Update an Object in the Cache

```
package com.oracle.handson;

import com.tangosol.net.NamedCache;

import com.tangosol.util.filter.EqualsFilter;
import com.tangosol.util.processor.AbstractProcessor;
import com.tangosol.util.InvocableMap;

import com.tangosol.io.pof.PortableObject;
import com.tangosol.io.pof.PofReader;
import com.tangosol.io.pof.PofWriter;

import com.oracle.handson.Address;
```

```
import com.oracle.handson.Contact;

import com.tangosol.net.CacheFactory;

import java.io.IOException;

/**
* ProcessorExample executes an example EntryProcessor.
*
*/
public class ProcessorExample
    {
    public ProcessorExample() {

    }
        public static void main(String[] args) {
          NamedCache cache = CacheFactory.getCache("ContactsCache");
          new ProcessorExample().execute(cache);
        }
    // ----- ProcessorExample methods ---------------------------------

    public void execute(NamedCache cache)
        {
        // People who live in Massachusetts moved to an in-state office
        Address addrWork = new Address("200 Newbury St.", "Yoyodyne, Ltd.",
                "Boston", "MA", "02116", "US");

        cache.invokeAll(new EqualsFilter("getHomeAddress.getState", "MA"),
                new OfficeUpdater(addrWork));
        }
    // ----- nested class: OfficeUpdater ------------------------------------

    /**
    * OfficeUpdater updates a contact's office address.
    */
    public static class OfficeUpdater
            extends AbstractProcessor
            implements PortableObject
        {
        // ----- constructors ---------------------------------------------

        /**
        * Default constructor (necessary for PortableObject implementation).
        */
        public OfficeUpdater() {
            }

        public OfficeUpdater(Address addrWork) {
            m_addrWork = addrWork;
            }

        // ----- InvocableMap.EntryProcessor interface ------------------

        public Object process(InvocableMap.Entry entry)
            {
            Contact contact = (Contact) entry.getValue();

            contact.setWorkAddress(m_addrWork);
            entry.setValue(contact);
            System.out.println("Work address was updated for " + contact.
```

```
getFirstName() + " " + contact.getLastName());
            return null;
            }

        // ----- PortableObject interface ------------------------------

        public void readExternal(PofReader reader)
                throws IOException
            {
            m_addrWork = (Address) reader.readObject(0);
            }

        public void writeExternal(PofWriter writer)
                throws IOException
            {
            writer.writeObject(0, m_addrWork);
            }

        // ----- data members ------------------------------------------

        private Address m_addrWork;
        }
    }
```

Edit the POF Configuration File

Edit the `contacts-pof-config.xml` file to add a user type ID for the `OfficeUpdater` entries

```
...
<user-type>
       <type-id>1006</type-id>
      <class-name>com.oracle.handson.
 ProcessorExample$OfficeUpdater</class-name>
</user-type>
...
```

Run the Cache Update Example

Follow these steps to run the cache update example:

1. In JDeveloper, compile the files in the Loading project.
2. Perform the following steps to test the `ObserverExample` and `ProcessorExample` classes.
   a. Restart your cache server.
   b. Run the `LoaderExample` class to load the cache.
   c. Run the `ObserverExample` class. Do not input any value through the **Input** area at the bottom of the messages window.
   d. Run the `ProcessorExample` to update records in the cache. You should see messages in JDeveloper, that are similar to Figure 7–2, indicting that addresses have been updated.

Figure 7–2 Output from the ObserverExample and ProcessorExample Classes

```
2010-05-28 14:12:30.801/3.265 Oracle Coherence GE 3.6.0.0 DPR3
Repeating InvokeFilterRequest due to the re-distribution of Par
132, 133, 134, 135, 136, 137, 138, 139, 140, 141, 142, 143, 144
151, 152, 153, 154, 155, 156, 157, 158, 159, 160, 161, 162, 163
170}
Work address was updated for John Xmvplt
2010-05-28 14:12:30.864/3.328 Oracle Coherence GE 3.6.0.0 DPR3
(thread=DistributedCache:PartitionedPofCache, member=16): Askin
partitions
Process exited with exit code 0.
```

8

Using JPA with Coherence

In this exercise, you learn how to use Java Persistence API (JPA) to perform object-relational mapping. This chapter contains the following sections:

- Introduction
- Mapping Relational Data to Java Objects with JPA

This exercise assumes that you have a working version of the Oracle Database 10g Express Edition (also known as the OracleXE database) installed on your system. If you do not have the database, you can download it here:

`http://www.oracle.com/technology/software/products/database/xe/index.html`

Introduction

A major enhancement in EJB technology is the addition of JPA, which simplifies the entity persistence model and adds capabilities, such as persistence for Java SE, that were not in the EJB 2.1 technology.

JPA deals with the way relational data is mapped to Java objects ("persistent entities"), the way that these objects are stored in a relational database so that they can be accessed at a later time, and the continued existence of an entity's state even after the application that uses it ends. In addition to simplifying the entity persistence model, the JPA standardizes object-relational mapping.

To determine how data is stored within a Coherence cluster, a backing map is used. By default, Coherence uses a memory-based backing map. To persist data, there are several backing map implementations.

You use the JPA implementation within this lesson. This implementation provides Object Relational Mapping (ORM) from the Java world to the database world, allowing you to use the standard Coherence get or put, and have the Coherence calls translated into database calls using JPA and Oracle TopLink (based on the open source EclipseLink project).

Mapping Relational Data to Java Objects with JPA

In this exercise, use JDeveloper to perform the following:

- Create a connection to the `HR` schema in the OracleXE database.
- Automatically generate JPA objects for the `EMPLOYEES` table.
- Modify `cache-server.cmd` to point to the sample JPA `cache-config.xml`.

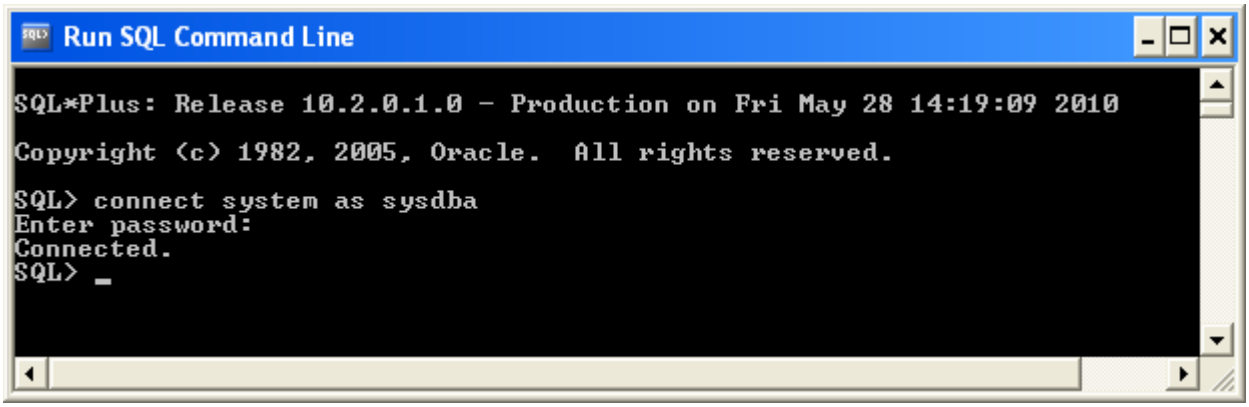
Unlock the OracleXE Database

Unlock the `HR` account in your pre-installed OracleXE database.

It is assumed that you have the OracleXE database installed on your machine and can access the HR schema. To unlock the HR account, perform the following steps:

1. Navigate to **Start** then **All Programs** then **Oracle Database 10g Express Edition** then **Run SQL Command Line**.
2. Enter `connect system as sysdba`, and then enter `welcome1` when prompted for the password. (Note: this exercise assumes that your user name is `system` and password is `welcome1`).

Figure 8–1 Connecting to the Database



3. Enter the command to unlock the account:

```
alter user hr identified by hr account unlock;
```

Figure 8–2 Unlocking the Database Account

Configure the Project for JPA

Create a new project in JDeveloper called `JPA`. See "Creating a New Project in an Existing Application" on page 2-11 if you need detailed information.

1. Set the default **Java Options** in **Run/Debug/Profile** of the **Project Properties** to the appropriate log level and to disable local storage.

   ```
   -Dtangosol.coherence.distributed.localstorage=false
   -Dtangosol.coherence.log.level=3
   ```

2. Ensure that the classpath points to the full path for the `coherence.jar` file:

   ```
   C:\oracle\product\coherence\lib\coherence.jar
   ```

Create the Database Connection and Create the JPA Persistence Unit

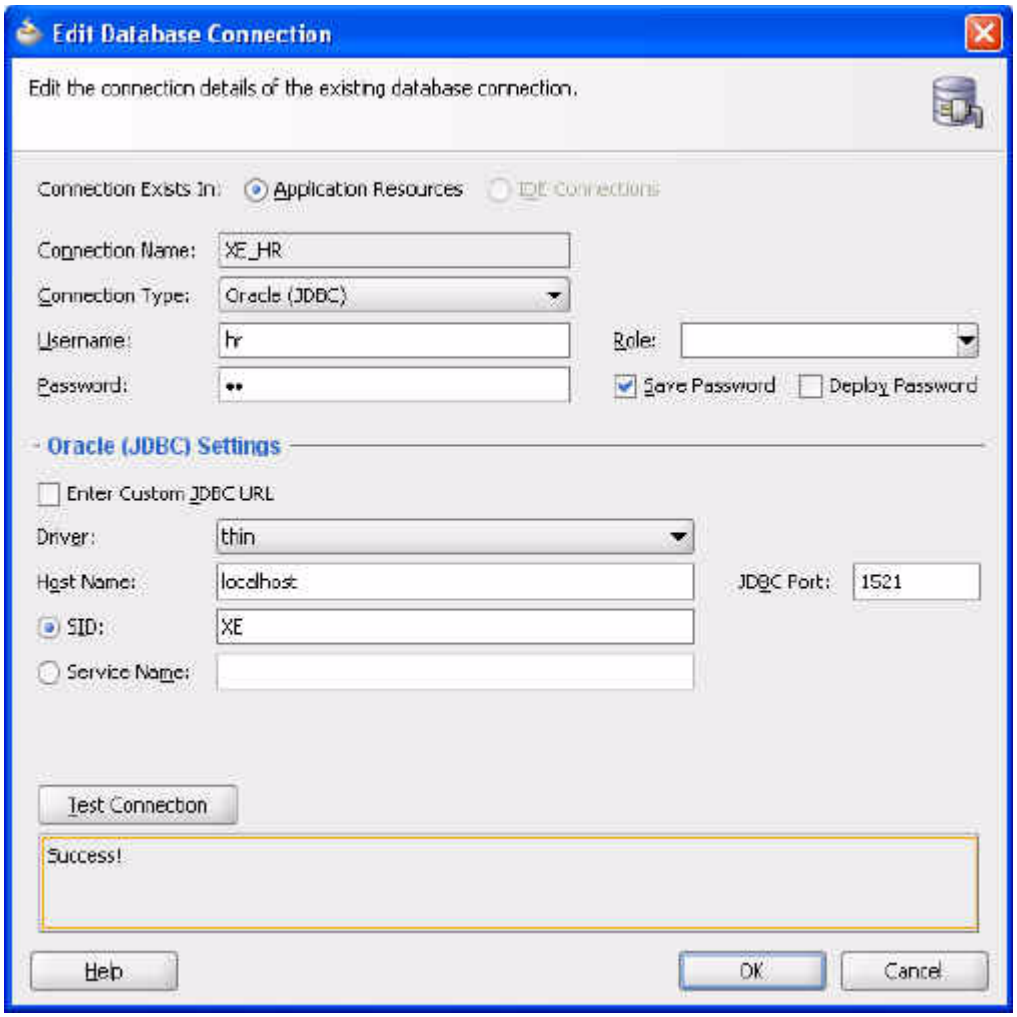
Follow these steps to connect to the HR database schema and create the JPA persistence unit.

1. In the **Application Resources** section of the navigator, right-click **Connections**, select **New Connection**, and then **Database Connection**.
2. Enter the details to connect to your HR schema and click **OK**.
   - **Connection Name**: XE_HR
   - **Connection Type**: Oracle (JDBC)
   - **Username**: hr
   - **Password**: hr
   - Click **Test Connection**.

   This should display "`Success`!"

   Click **OK**.

Figure 8–3 Defining the Database Connection

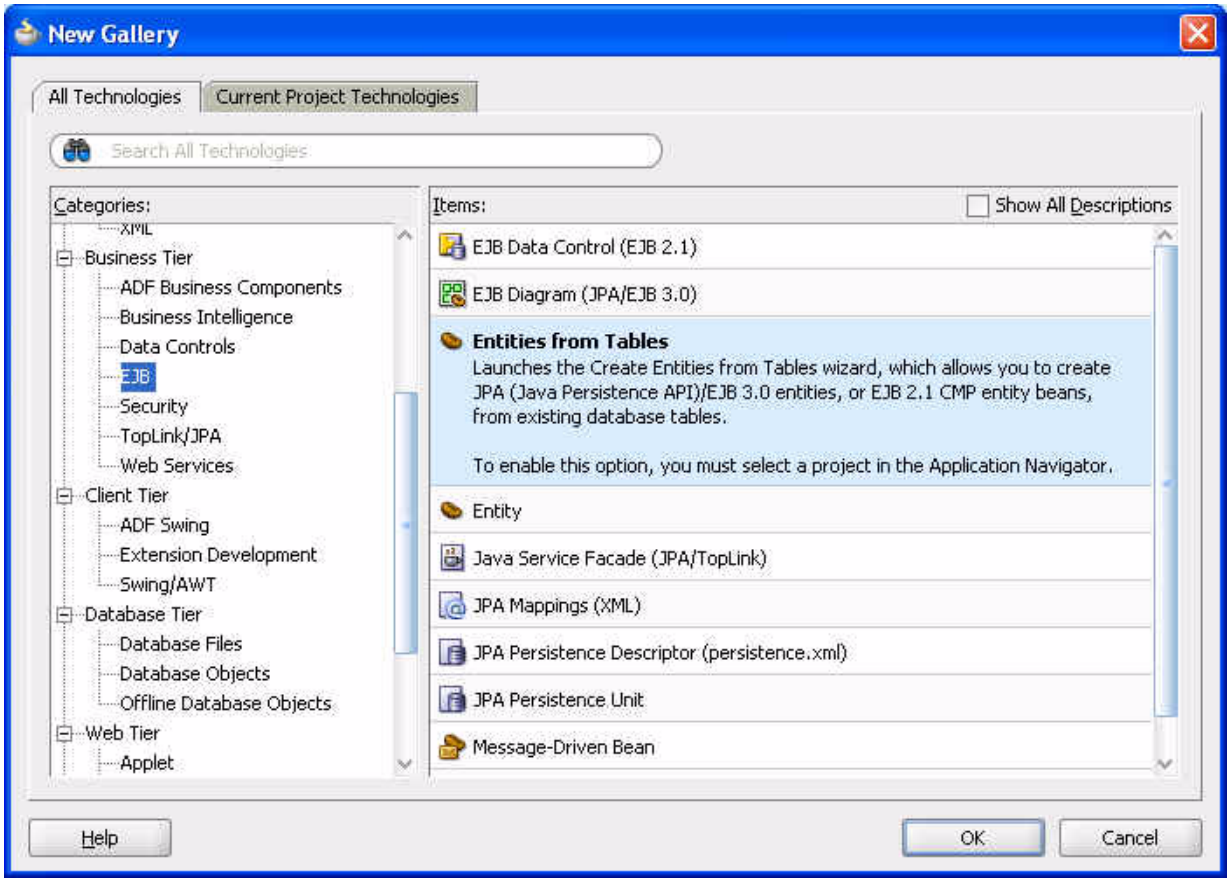


Create the JPA Persistence Unit and Entity Beans

Follow these steps to create the JPA persistence unit and the Entity Beans.

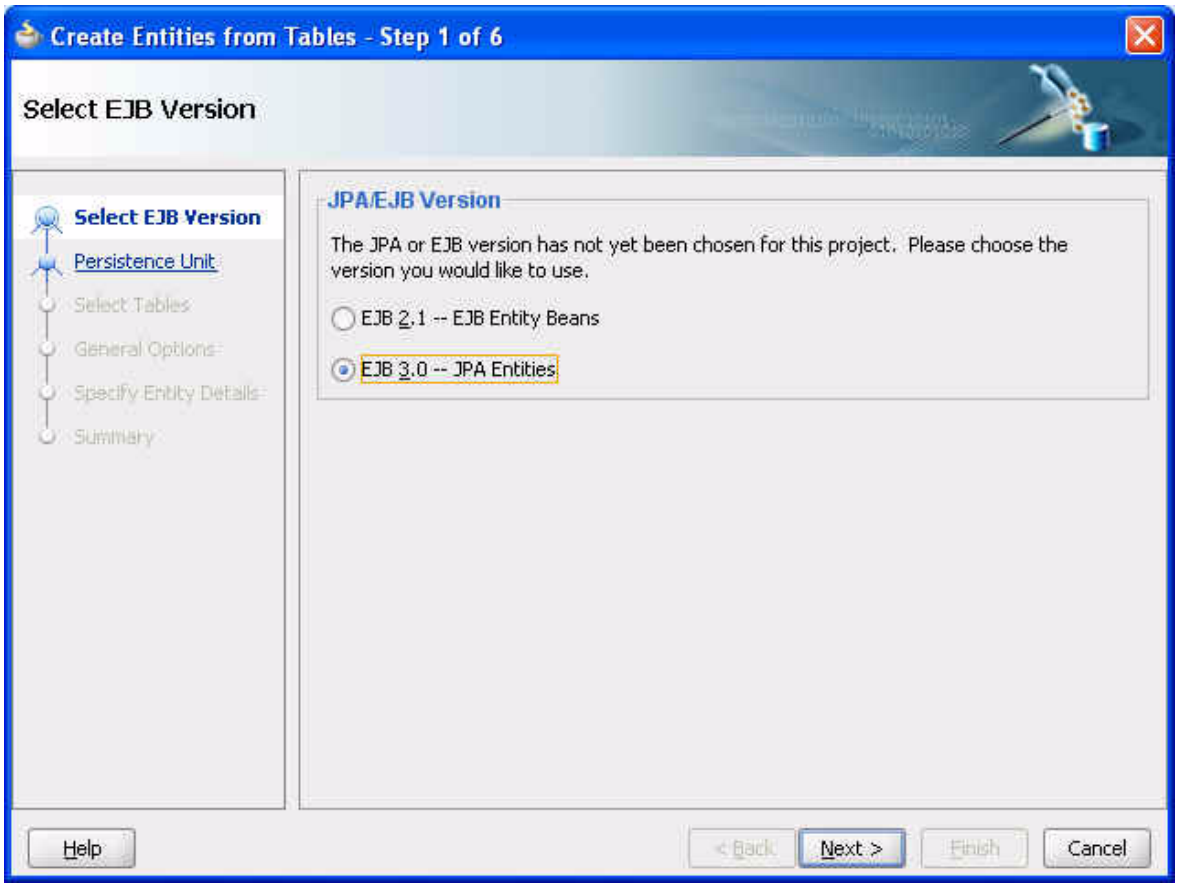
1. Right-click the `JPA` project and select **New**. Under **Business Tier**, select **EJB**, and then select **Entities from Tables**. Click **OK**.

Figure 8–4 Creating EJB Entity Beans



2. In the **Create Entities from Tables** window, select **EJB 3.0 --JPA Entities**, and then click **Next**.

Figure 8–5 Specifying the EJB Version



3. Click **New** to create a persistence unit. (Each persistence unit defines a set of classes and their mapping characteristics when persisting them.) Enter the following details and click **OK**.

   Name: JPA

   JTA Datasource Name: <leave blank>

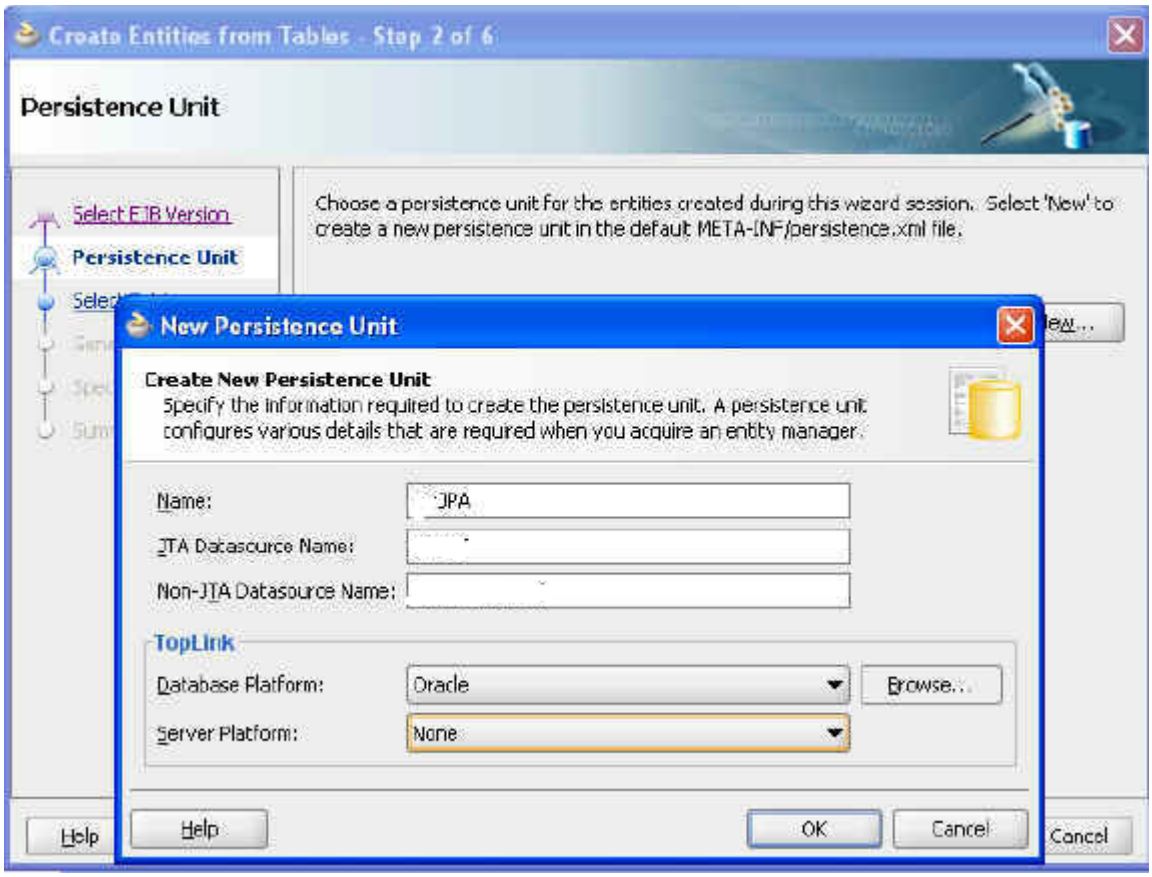
   Non-JTA Datasource Name: <leave blank>

   Database Platform: Oracle

   Server Platform: None

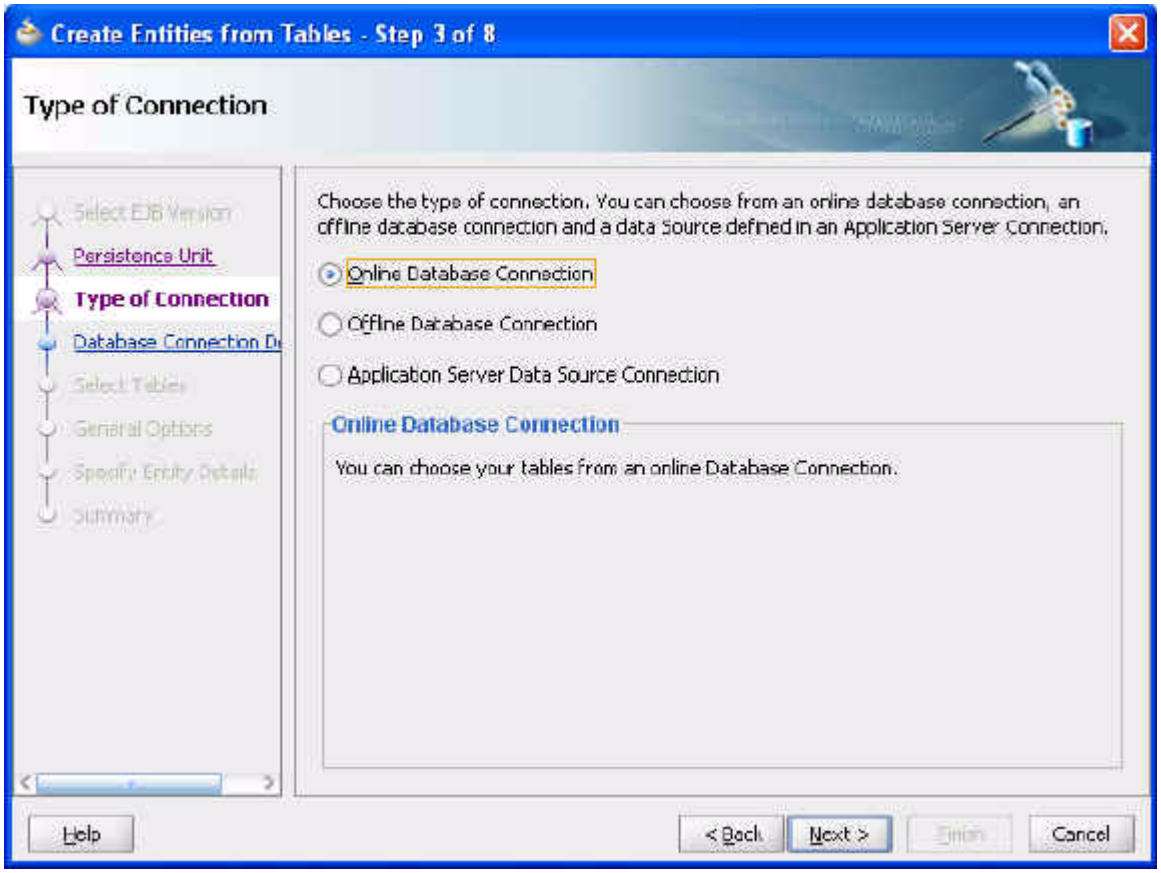
   Click **OK**. When the **Create Entities from Tables** screen returns, click **Next**.

Figure 8–6 Defining the Persistence Unit



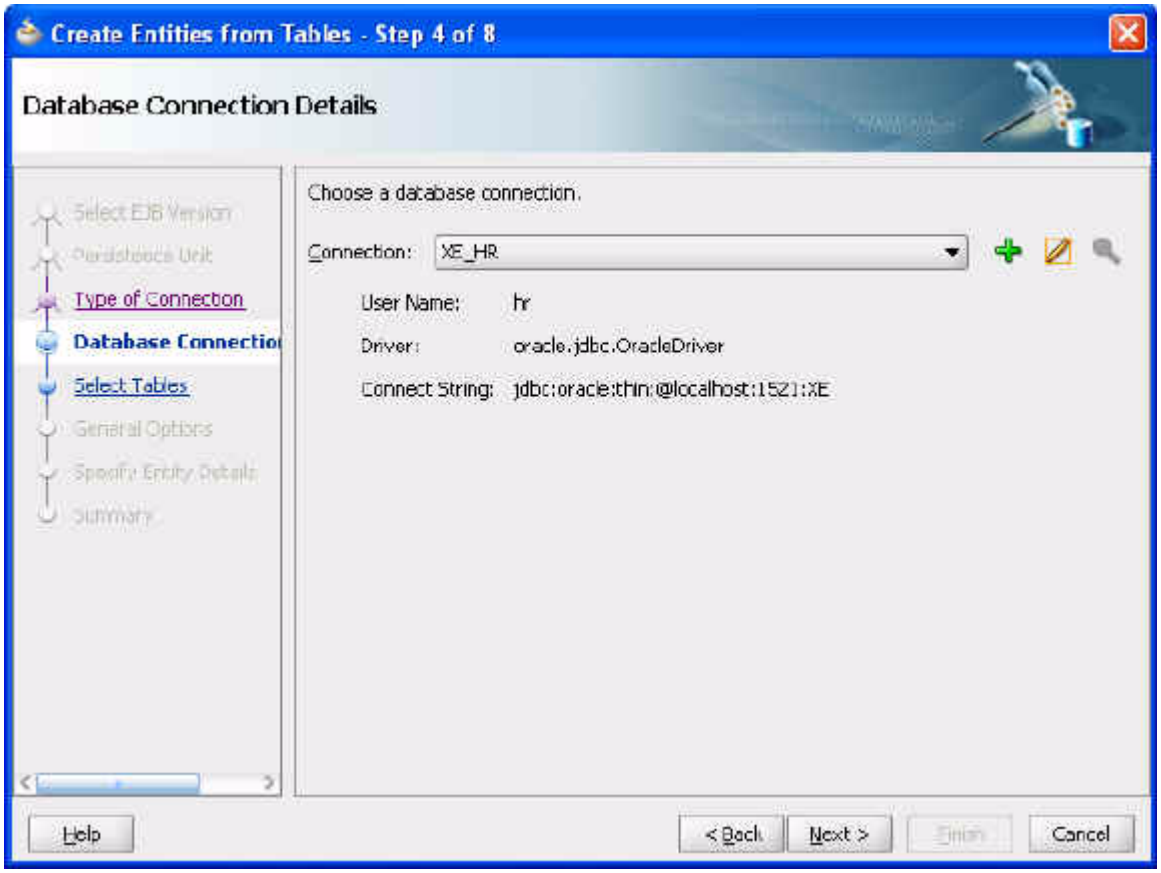
4. Select the **Online Database Connection** option and click **Next**.

Figure 8–7 Creating Entity Beans from Table Data



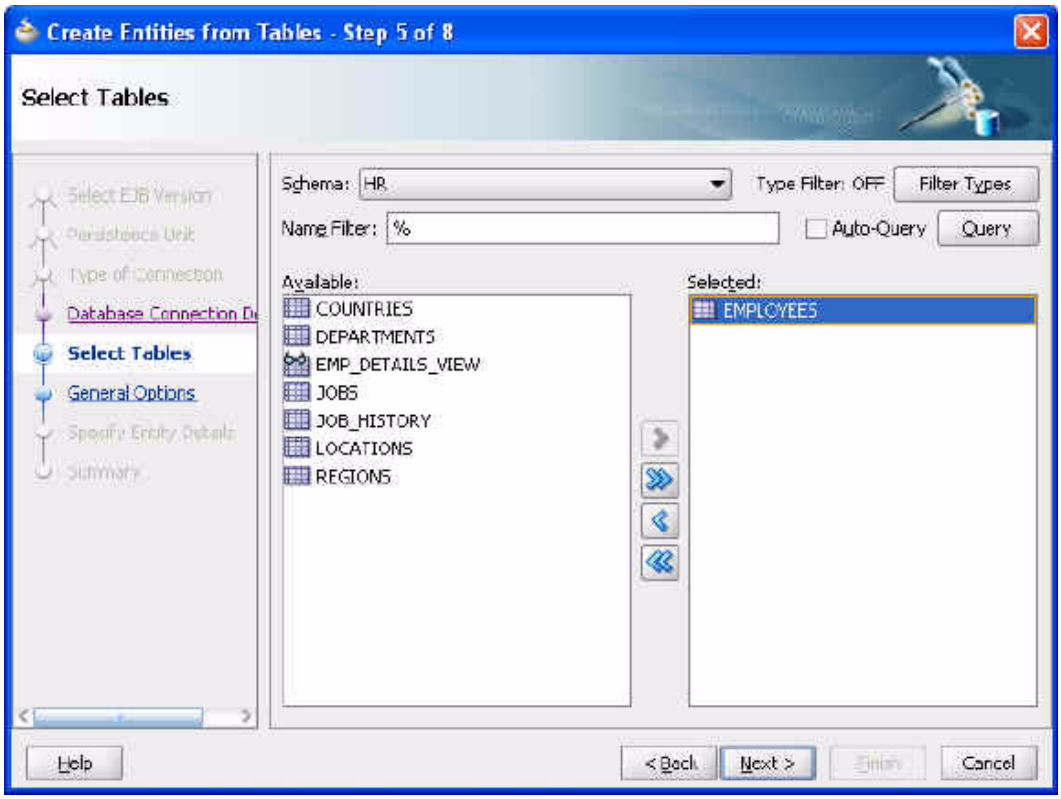
5. In the **Database Connection Details** window, click **Next**.

Figure 8–8 Choosing the Database Connection



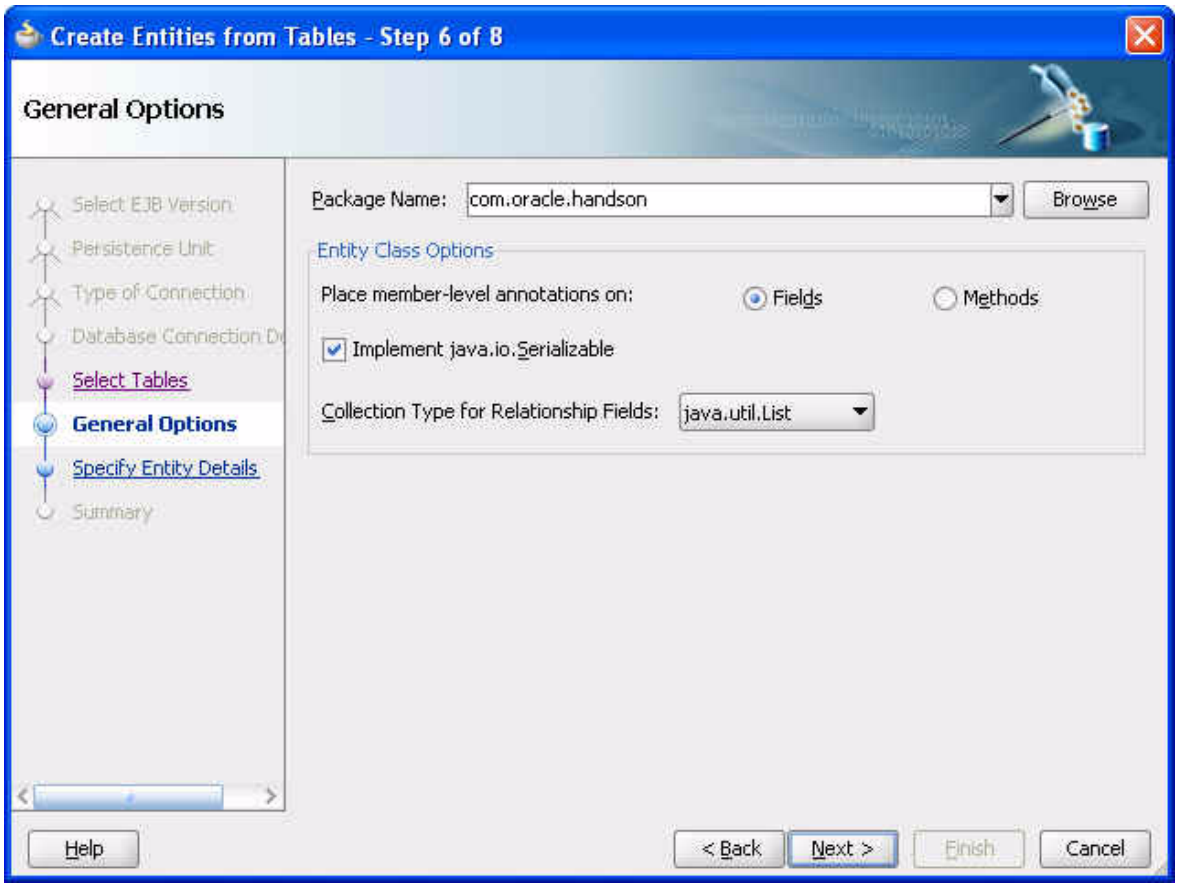
6. Click the **Query** button to display the tables illustrated in Figure 8–9. Select the `EMPLOYEES` table and shuttle it to the **Selected** column. Click **Next**.

Figure 8–9 Choosing the Table Data for the Entity Bean



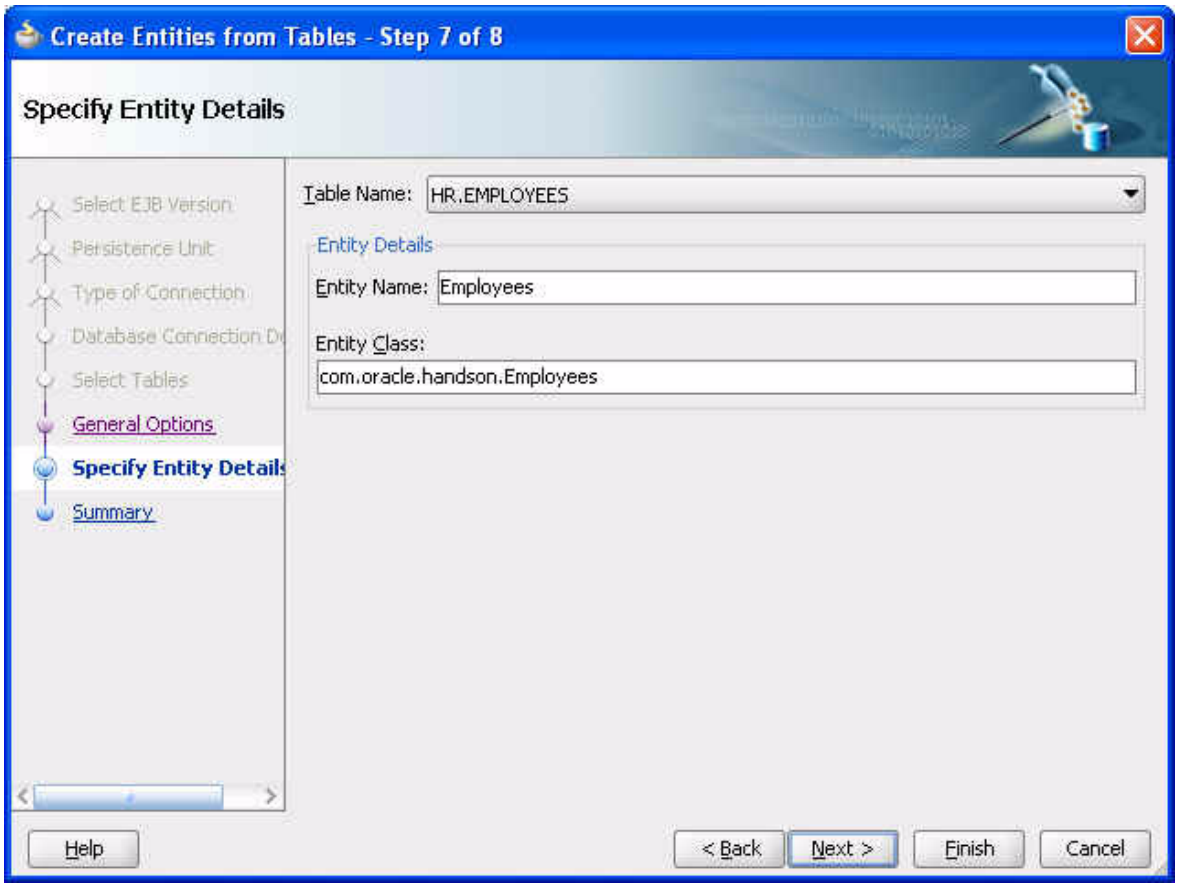
7. Accept the default **General Options** and click **Next**.

Figure 8–10 Choosing General Options for the Entity



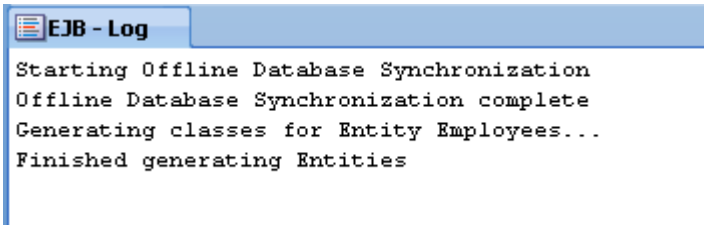
8. Accept the default **Entity Class Details** and click **Next**.

Figure 8–11 Specifying the Entity Details



9. A **Summary** page appears. Click **Finish** to create the Entity bean. You should see messages similar to Figure 8–12 in the EJB Log window of the navigator.

Figure 8–12 Generating EJB Entity Beans: the EJB Log Window



10. Replace the contents of `persistence.xml` with the code in Example 8–1 and save the file. Ensure that the connection details match your database connection details. The `persistence.xml` file appears in the `c:\home\oracle\labs\JPA\src\META-INF` folder.

Example 8–1 persistance.xml File Contents

```
<persistence xmlns:xsi="http://www.w3.org/2001/XMLSchemainstance" version="1.0"
xmlns="http://java.sun.com/xml/ns/persistence">
<persistence-unit name="JPA" transaction-type="RESOURCE_LOCAL">
    <provider>
        org.eclipse.persistence.jpa.PersistenceProvider
    </provider>
```

```
    <class>com.oracle.handson.Employees</class>
    <properties>
        <property name="javax.persistence.jdbc.driver"
value="oracle.jdbc.OracleDriver"/>
        <property name="javax.persistence.jdbc.url"
value="jdbc:oracle:thin:@localhost:1521:XE"/>
        <property name="javax.persistence.jdbc.password" value="hr"/>
        <property name="javax.persistence.jdbc.user" value="hr"/>
    </properties>
</persistence-unit>
</persistence>
```

Create the Cache Configuration File for JPA

Open a text editor and create a file named `jpa-cache-config.xml`. Use the code illustrated in Example 8–2. Save the file in the `home\oracle\labs\` directory.

Example 8–2 Cache Configuration for JPA

```
<?xml version="1.0" encoding="windows-1252" ?>
<cache-config>
  <caching-scheme-mapping>
    <cache-mapping>
      <!-- Set the name of the cache to be the entity name  -->
      <cache-name>Employees</cache-name>
      <!-- Configure this cache to use the scheme defined below  -->
      <scheme-name>jpa-distributed</scheme-name>
    </cache-mapping>
  </caching-scheme-mapping>
  <caching-schemes>
    <distributed-scheme>
      <scheme-name>jpa-distributed</scheme-name>
      <service-name>JpaDistributedCache</service-name>
      <backing-map-scheme>
        <read-write-backing-map-scheme>
          <!--
      Define the cache scheme
      -->
          <internal-cache-scheme>
            <local-scheme/>
          </internal-cache-scheme>
          <cachestore-scheme>
            <class-scheme>
              <class-name>com.tangosol.coherence.jpa.JpaCacheStore</class-name>
              <init-params>
                <!--
              This param is the entity name
              This param is the fully qualified entity class
              This param should match the value of the
              persistence unit name in persistence.xml
              -->
                  <init-param>
                    <param-type>java.lang.String</param-type>
                    <param-value>{cache-name}</param-value>
                  </init-param>
                  <init-param>
                    <param-type>java.lang.String</param-type>
                    <param-value>com.oracle.handson.{cache-name}</param-value>
                  </init-param>
                  <init-param>
```

```
                <param-type>java.lang.String</param-type>
                <param-value>JPA</param-value>
              </init-param>
            </init-params>
          </class-scheme>
        </cachestore-scheme>
      </read-write-backing-map-scheme>
    </backing-map-scheme>
    <autostart>true</autostart>
  </distributed-scheme>
 </caching-schemes>
</cache-config>
```

Create a Cache Server Start-Up File

Create a cache server file for the JPA project. You can use the `cache-server.cmd` file as a starting point.

1. Open a terminal window. Navigate to the `/oracle/product/coherence/bin` directory and copy the `cache-server.cmd` file to `jpa-cache-server.cmd`.

   ```
   cp cache-server.cmd jpa-cache-server.cmd
   ```

2. Edit `jpa-cache-server.cmd`. Declare the cache configuration file in `Java_OPTS`:

   ```
   -Dtangosol.coherence.cacheconfig=C:\home\oracle\labs\jpa-cache-config.xml
   ```

3. Add the following `CLASSPATH` to the `-cp` argument: `C:\home\oracle\labs\JPA\classes;`

   ```
   C:\home\oracle\labs\JPA\classes
   ```

4. You must also add the following JAR files to the `CLASSPATH`:

   - Coherence JPA libraries: `C:\oracle\product\coherence\lib\coherence-jpa.jar`
   - EclipseLink persistence libraries: `C:\oracle\product\modules\org.eclipse.persistence_1.0.0.0_2-0.jar`
   - Oracle JDBC JAR, `ojdbc6.jar`, for database connectivity: `C:\oracle\product\wlserver_10.3\server\lib\ojdbc6.jar`
   - Coherence TopLink libraries: `C:\oracle\product\coherence\lib\coherence-toplink.jar`
   - `javax.persistence.*` libraries: `C:\oracle\product\modules\javax.persistence_1.0.0.0_1-0-2.jar`

   The classpath should look similar to the following:

   ```
   ...
   C:\oracle\product\coherence\lib\coherence-jpa.jar;C:\oracle\product\modules\org
   .eclipse.persistence_1.0.0.0_2-0.jar;C:\oracle\product\wlserver_
   10.3\server\lib\ojdbc6.jar;C:\oracle\product\modules\javax.persistence_1.0.0.0_
   1-0-2.jar"
   ...
   ```

   Example 8–3 illustrates a modified `jpa-cache-server.cmd` file:

Example 8–3 Modified jpa-cache-server.cmd File

```
@echo off
@
@rem This will start a cache server
@
setlocal

:config
@rem specify the Coherence installation directory
set coherence_home=c:\oracle\product\coherence

@rem specify the JVM heap size
set memory=512m

:start
if not exist "%coherence_home%\lib\coherence.jar" goto instructions

if "%java_home%"=="" (set java_exec=java) else (set java_exec=%java_
home%\bin\java)

:launch

set java_opts="-Xms%memory% -Xmx%memory%
-Dtangosol.coherence.cacheconfig=\home\oracle\labs\jpa-cache-config.xml"

"%java_exec%" -server -showversion "%java_opts%" -cp "%coherence_
home%\lib\coherence.jar;C:\home\oracle\labs\JPA\classes;C:\oracle\product\coherenc
e\lib\coherence-jpa.jar;C:\oracle\product\modules\org.eclipse.persistence_1.0.0.0_
2-0.jar;C:\oracle\product\wlserver_
10.3\server\lib\ojdbc6.jar;C:\oracle\product\oracle_common\modules\oracle.toplink_
11.1.1\eclipselink-dbwsutils.jar;C:\oracle\product\modules\javax.persistence_
1.0.0.0_1-0-2.jar" com.tangosol.net.DefaultCacheServer %1

goto exit

:instructions

echo Usage:
echo   ^<coherence_home^>\bin\cache-server.cmd
goto exit

:exit
endlocal
@echo on
```

5. Save the `jpa-cache-server.cmd` file and ensure that all other cache servers are stopped. Run `jpa-cache-server.cmd`.

   ```
   C:\oracle\product\coherence\bin>jpa-cache-server.cmd
   ```

Modify JPA Project Properties

Follow these steps to modify the run time properties and edit the classpath in JDeveloper.

1. Edit the `JPA` **Project Properties** and modify the **Run/Debug/Profile** configuration. Append the following line to the existing **Java Options**.

   ```
   -Dtangosol.coherence.cacheconfig=C:\home\oracle\labs\jpa-cache-config.xml
   ```

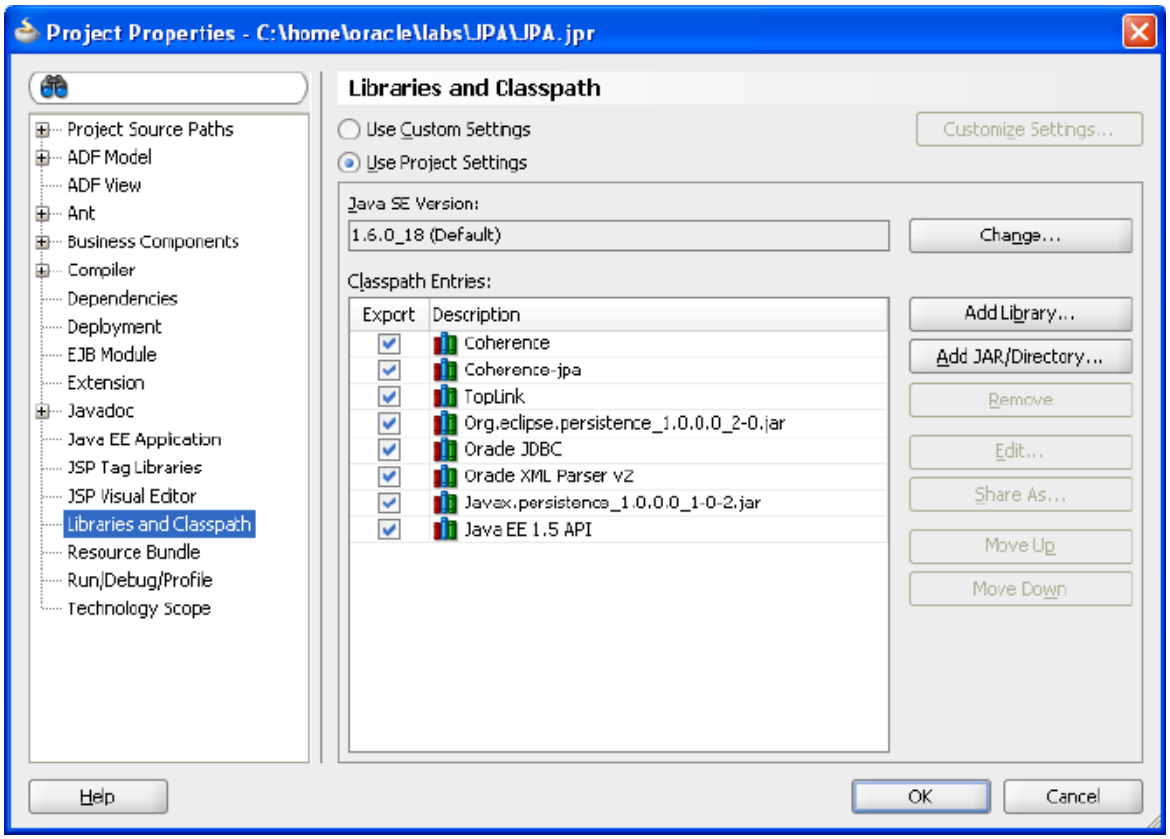
2. Add additional `CLASSPATH` entries to the existing project properties.

   Navigate to **Tools** then **Project Properties** then **Libraries and Classpath**. Use the **Add JAR/Directory** and **Add Library** buttons to add any of the following JAR files and libraries that are not already on the `CLASSPATH` (Note: the `coherence.jar` file should already be present):

   - `coherence-jpa.jar` to provide a reference to the `JpaCacheStore`.
   - `TopLink` predefined JDeveloper library. This will provide the required EclipseLink JARs and APIs
   - EcilpseLink Persistence JAR file for the Eclipse persistence API: `C:\oracle\product\modules\org.eclipse.persistence_1.0.0.0_2-0.jar`
   - `Oracle JDBC` predefined JDeveloper library for database connectivity
   - `Oracle XML Parser v2` predefined JDeveloper library for interpreting XML
   - Java Persistence JAR file for the persistence API: `C:\oracle\product\modules\javax.persistence_1.0.0.0_1-0-2.jar`
   - Java EE 1.5 API. Included by default.

   The **Libraries and Classpath** screen should look similar to Figure 8–13:

Figure 8–13 Adding JARs and Libraries to the Classpath



Create a Class to Interact with the Data Object

Create a new class in the JPA project to interact with the Employee object.

1. Create a new class with a main method called RunEmployeeExample. See "Creating a Java Class" on page 2-13 if you need detailed information.
2. Create the code to perform the following:
   a. Get an employee using EMPLOYEE_ID. EMPLOYEE_ID should be a long data type.
   b. Display the salary.
   c. Give them a 10% pay raise.
   d. Get the value again to confirm the pay raise.

   Example 8–4 illustrates a possible solution.

Example 8–4 Sample Employee Class File

```
package com.oracle.handson;

import com.tangosol.net.CacheFactory;
import com.tangosol.net.NamedCache;

public class RunEmployeeExample {
    public RunEmployeeExample() {
    }

    public static void main(String[] args) {
        long empId = 190L;  // emp 190 - Timothy Gates

        NamedCache employees = CacheFactory.getCache("Employees");

        Employees emp = (Employees)employees.get(empId);

        System.out.println("Employee " + emp.getFirstName() + " " +
                        emp.getLastName() + ", salary = $" + emp.getSalary() );

        // give them a 10% pay rise
        emp.setSalary( emp.getSalary() * 1.1);

        employees.put(empId, emp);

        Employees emp2 = (Employees)employees.get(empId);

        System.out.println("New Employee details are " + emp2.getFirstName() + " "
+ emp2.getLastName() + ", salary = $" + emp2.getSalary() );
    }
}
```

Run the JPA Example

Now that the Employees class has been annotated to persist to the database using JPA, and you have included the persistence.xml file to tell JPA where your database is, Coherence employs a CacheStore implementation that uses JPA to load and store objects to the database. When you use the get(Object key) method, the following happens:

- Coherence looks for the entry with the key.

- If the entry has not already been cached, or if it is expired from the cache, then Coherence asks the backing map, which uses JPA and EclipseLink to retrieve the data.
- If the entry is in the cache, Coherence returns the entry directly to the application without going through EclipseLink. When you use `put(Object Key,Object Value)`, Coherence uses JPA through EclipseLink to persist any changes to the database.

1. Compile the project files if you have not already done so.
2. Run `RunEmployeeExample.java`.

The output should look similar to the text illustrated Figure 8–14.

Figure 8–14 Results from the RunEmployeeExample Application

```
  ThisMember=Member(Id=2, Timestamp=2010-06-01 11:00:01.037, Address=130.35.99.213
MachineId=49877, Location=site:us.oracle.com,machine:tpfaeffl-lap7,process:2728,
Role=OracleHandsonRunEmployeeExample)
  OldestMember=Member(Id=1, Timestamp=2010-06-01 10:59:27.099, Address=130.35.99.2
MachineId=49877, Location=site:us.oracle.com,machine:tpfaeffl-lap7,process:5980, R
  ActualMemberSet=MemberSet(Size=2, BitSetCount=2
    Member(Id=1, Timestamp=2010-06-01 10:59:27.099, Address=130.35.99.213:8088, Ma
Location=site:us.oracle.com,machine:tpfaeffl-lap7,process:5980, Role=CoherenceServ
    Member(Id=2, Timestamp=2010-06-01 11:00:01.037, Address=130.35.99.213:8090, Ma
Location=site:us.oracle.com,machine:tpfaeffl-lap7,process:2728, Role=OracleHandson
    )
  RecycleMillis=1200000
  RecycleSet=MemberSet(Size=0, BitSetCount=0
    )
  )

TcpRing{Connections=[1]}
IpMonitor{AddressListSize=0}

Employee Timothy Gates, salary = $13325.46
New Employee details are Timothy Gates, salary = $14658.006
Process exited with exit code 0.
```

9

Interacting with the Cache and the Database

In this chapter, you create and configure an Oracle Coherence cache in Oracle JDeveloper using all the concepts presented in this tutorial. In this exercise you will:

- Create a Java class that creates a `NamedCache` and can put and get cache entries.
- Create a cache configuration file to define the mapping for cache names, cache types, and naming patterns.
- Create a Java class that creates a connection to the Oracle database and can retrieve and store table data.
- Create a database cache. This class will add cache entries, query the database cache, and retrieve entries.

This chapter has the following sections:

Introduction

A Coherence cache is a collection of data objects that serves as an intermediary between the database and the client applications. Database data may be loaded into a cache and made available to different applications. Thus, Coherence caches reduce load on the database and provide faster access to database data.

Coherence caches provide higher availability through database isolation and data replication. Modifications made to a cache may be synchronized with the database whenever the database is available. Even if the database or an application server node is not available, database updates are still reliable due to the lazy load and lazy write mechanism used by a Coherence cache and due to the failover and fail back provided by Oracle Coherence.

Coherence caches provide distributed processing not only across a cluster of application server nodes but also across the data objects in the cache, because data modification operations may be performed on the data objects.

Oracle Coherence also provides event-based processing. The state of data objects in a cache may be monitored and actions invoked on other processes such as the start of a business process execution language (BPEL) process.

Oracle Coherence supports different types of caches.

- Replicated caches, where data is replicated to each of the application server nodes in the cluster. This is suitable if faster read access is required but not suitable for

writes, because data has to be written to each of the nodes. The drawback of replicated caches is that they require a large memory footprint as every node has a copy of every object.

- Distributed (or "partitioned") caches, where data is distributed (load-balanced) across different nodes. Failover is implemented in a distributed cache using backups, which are also distributed across the cluster nodes.

Oracle Coherence is implemented by using services such as the cluster service, the distributed cache service, and the replicated cache service. Whichever type of cache is used, an application uses the same API to access and store data.

The cache configuration deployment descriptor is used to configure a cache. The root element of the cache configuration file is `cache-config`. Cache names and name patterns are mapped to cache types in the `caching-scheme-mapping` element using the subelement `cache-mapping`. Cache types are defined in the `caching-schemes` element. Some of the commonly used cache types are described in Table 9–1.

Table 9–1 Descriptions of Cache Types

Cache Type	Description
distributed scheme	Defines a distributed cache in which data is stored across a cluster of nodes
replicated scheme	Defines a cache in which cache entries are replicated across all the cluster nodes
read-write-backing-map scheme	Defines a map, which provides a cache of a persistent store such as a relational database
external scheme	Defines an external cache such as a disk
class scheme	Defines a custom cache implementation, which is required to implement the java.util.Map interface

Creating a Cache Application

This section describes how to create and run an application that puts and gets cache data.

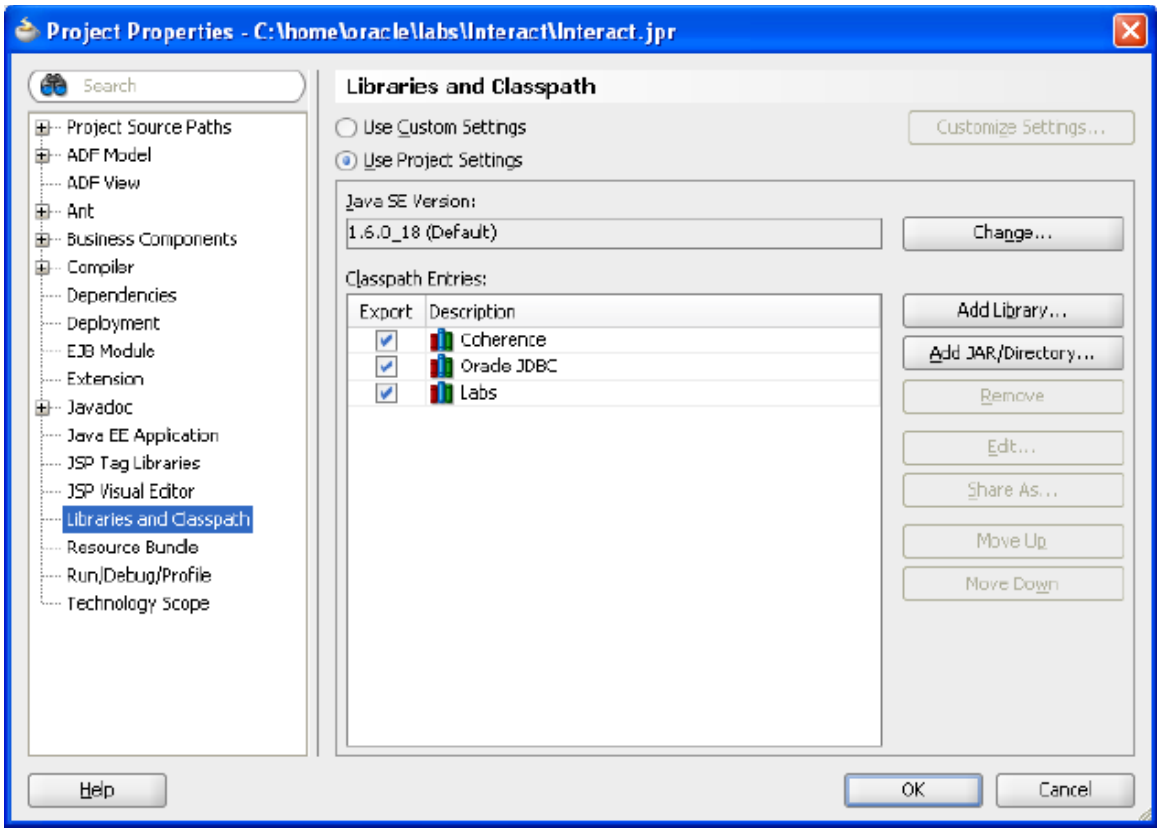
1. Create an Application which Creates a Cache
2. Create a Cache Configuration File
3. Configure the Project Properties
4. Edit the Cache Server Start-Up File
5. Run the Cache Creation Application

Create an Application which Creates a Cache

Follow these steps to create a Java class which creates a Coherence cache:

1. Create a project in Oracle JDeveloper.
   a. Create a project called `Interact` in Oracle JDeveloper. See "Creating a New Project in an Existing Application" on page 2-11 if you need detailed information.
   b. Add the Coherence JAR file `coherence.jar` and `C:\home\oracle\labs` to the project classpath. Also add the **Oracle JDBC** library, which is required for database access to the project libraries.

Figure 9–1 Adding Coherence Jar, Labs Directory, and JDBC Libraries to the Classpath



2. Create a Java class, `CoherenceCache`, that will be used to create a Coherence cache. See "Creating a Java Class" on page 2-13 if you need detailed information on creating a class.

   a. Create a cache in the `CoherenceCache` Java class. Import the `CacheFactory` class and the `NamedCache` interface.

   ```
   import com.tangosol.net.CacheFactory;
   import com.tangosol.net.NamedCache;
   ```

   b. An instance of a cache is created from the `CacheFactory` class. Create a `NamedCache` using the `getCache()` method of the `CacheFactory` class. Use the cache name `VirtualCache`, which is mapped to a distributed caching scheme.

   ```
   NamedCache cache = CacheFactory.getCache ( "VirtualCache");
   ```

   c. A `NamedCache` is a `java.util.Map` that holds resources that are shared across nodes in a cluster. Add a cache entry using the `put()` method.

   ```
   cache.put (key, "Hello Cache");
   ```

   d. A cache entry can be retrieved using the `get()` method.

   ```
   System.out.println((String)cache.get("hello"));
   ```

   Example 9–1 illustrates a possible solution. You can copy the code to the `CoherenceCache` application in Oracle JDeveloper.

Example 9–1 Implementation of a Coherence Cache

```
package com.oracle.handson;

import com.tangosol.net.CacheFactory;
import com.tangosol.net.NamedCache;

public class CoherenceCache {
    NamedCache cache;
    public CoherenceCache() {
    }
    public void putCache(){
            cache = CacheFactory.getCache ( "VirtualCache");
           String key = "hello";
           cache.put (key, "Hello Cache");

       }

       public void retrieveCache(){

           System.out.println((String)cache.get("hello"));

       }

        public static void main (String [] args) {
        CoherenceCache cache = new CoherenceCache();
        cache.putCache();
        cache.retrieveCache();
        }
}
```

Create a Cache Configuration File

Create an XML document, `cache-config.xml`, as the cache configuration deployment descriptor.

To add the XML document, right-click the project and choose **New**. In the **New Gallery** window, select **XML** under **General** categories. Select **XML Document** and click **OK**. Change the **File Name** to `cache-config.xml`. Change the directory to `C:\home\oracle\labs\`.

In the cache configuration file:

- Define mapping for cache names and naming patterns with the `cache-mapping` elements in the `caching-scheme-mapping` element.
- Map the cache name `VirtualCache` to cache type `default-distributed`.
- Define the distributed caching scheme with the `distributed-scheme` element using the `DistributedCache` service.

The cache configuration file is listed in Example 9–2. Copy the contents of this example to the `cache-config.xml` file in Oracle JDeveloper.

Example 9–2 Cache Configuration File

```
<?xml version="1.0"?>
<!DOCTYPE cache-config SYSTEM "cache-config.dtd">
<cache-config>
    <caching-scheme-mapping>

        <cache-mapping>
```

```
            <cache-name>VirtualCache</cache-name>
            <scheme-name>default-distributed</scheme-name>
        </cache-mapping>
    </caching-scheme-mapping>
    <caching-schemes>
        <!--
        Default Distributed caching scheme.
        -->
        <distributed-scheme>
            <scheme-name>default-distributed</scheme-name>
            <service-name>DistributedCache</service-name>
            <backing-map-scheme>
                <class-scheme>
                    <scheme-ref>default-backing-map</scheme-ref>
                </class-scheme>
            </backing-map-scheme>
        </distributed-scheme>
         <class-scheme>
            <scheme-name>default-backing-map</scheme-name>
            <class-name>com.tangosol.util.SafeHashMap</class-name>
         </class-scheme>
  <autostart>true</autostart>
 </caching-schemes>
</cache-config>
```

Configure the Project Properties

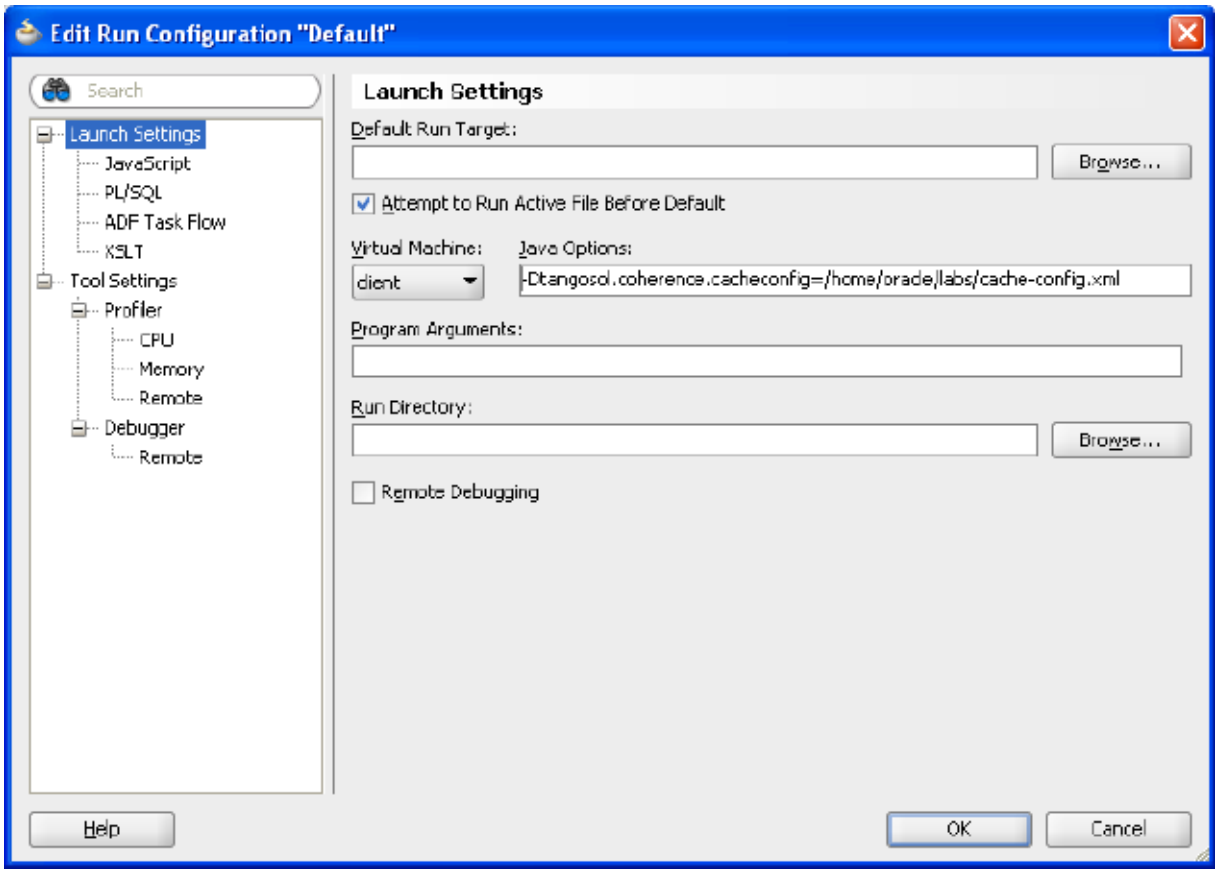
Modify the Run configuration for the application to add the cache configuration file as a run-time Java option.

1. Select the project node and select **Tools** then **Project Properties**. In the **Project Properties** window, select **Run/Debug/Profile**. The **Default** configuration is selected by default. Click **Edit** for the **Default** configuration.
2. In the **Edit Run Configuration** window, select **Launch Settings**. In the **Java Options** field, specify the cache configuration file (`cache-config.xml`) with `-Dtangosol.coherence.cacheconfig`.

   For example, for the Oracle Coherence application that you created, specify the following (your path to `cache-config.xml` may vary) in the **Java Options** field and click **OK**.

   ```
   -Dtangosol.coherence.cacheconfig=/home/oracle/labs/cache-config.xml
   ```

Figure 9–2 Setting the Runtime Options



Click **OK** in the **Run/Debug/Profile** window. The cache configuration file will be added as a run-time Java option to the Coherence Java application.

Edit the Cache Server Start-Up File

Edit the JPA cache server (`jpa-cache-server.cmd`) start-up file that you created in Chapter 8, "Using JPA with Coherence".

Replace the name of the cache configuration file, "`jpa-cache-config.xml`", with "`cache-config.xml`".

```
-Dtangosol.coherence.cacheconfig=\home\oracle\labs\cache-config.xml
```

In the classpath, replace the path to the `JPA\classes` directory with the path to the `Interact\classes` directory.

```
C:\home\oracle\labs\Interact\classes;
```

Run the Cache Creation Application

Follow these steps to run the cache creation application `CoherenceCache.java`.

1. Stop any running cache servers. Run `jpa-cache-server.cmd` to start the cache server.
2. Right-click the Oracle Coherence application `CoherenceCache.java` and click **Run**. The JDeveloper Log window displays the output:

- the operational configuration is loaded from `tangosol-coherence.xml`. This file specifies the operational and run-time settings used by Coherence for its clustering, communication, and data management services.
- the cache configuration is loaded from `cache-config.xml`.
- a new cluster is created and the `DistributedCache` service joins the cluster.
- the output of the `CoherenceCache.java` program, `Hello Cache` is displayed.

Example 9–3 Output of the Coherence Cache Application

```
2010-06-02 17:11:36.197/0.313 Oracle Coherence 3.6.0.0 DPR3 <Info> (thread=main, member=n/a):
Loaded operational configuration from "jar:file:/C:/oracle/product/coherence/lib/coherence.jar!/
tangosol-coherence.xml"
2010-06-02 17:11:36.213/0.329 Oracle Coherence 3.6.0.0 DPR3 <Info> (thread=main, member=n/a):
Loaded operational overrides from "jar:file:/C:/oracle/product/coherence/lib/coherence.jar!/
tangosol-coherence-override-dev.xml"
2010-06-02 17:11:36.213/0.329 Oracle Coherence 3.6.0.0 DPR3 <D5> (thread=main, member=n/a):
Optional configuration override "/tangosol-coherence-override.xml" is not specified
2010-06-02 17:11:36.213/0.329 Oracle Coherence 3.6.0.0 DPR3 <D5> (thread=main, member=n/a):
Optional configuration override "/custom-mbeans.xml" is not specified

Oracle Coherence Version 3.6.0.0 DPR3 Build 16141
 Grid Edition: Development mode
Copyright (c) 2000, 2010, Oracle and/or its affiliates. All rights reserved.

2010-06-02 17:11:36.431/0.547 Oracle Coherence GE 3.6.0.0 DPR3 <Info> (thread=main, member=n/a):
Loaded cache configuration from "file:/C:/home/oracle/labs/cache-config.xml"
2010-06-02 17:11:36.853/0.969 Oracle Coherence GE 3.6.0.0 DPR3 <D4> (thread=main, member=n/a):
SystemSocketProvider bound to port 8088
2010-06-02 17:11:40.353/4.469 Oracle Coherence GE 3.6.0.0 DPR3 <Info> (thread=Cluster, member=n/a):
Created a new cluster "cluster:0xC4DB" with Member(Id=1, Timestamp=2010-06-02 17:11:36.869,
Address=130.35.99.213:8088, MachineId=49877, Location=site:us.oracle.
com,machine:tpfaeffl-lap7,process:4884, Role=OracleHandsonCoherenceCache, Edition=Grid Edition,
Mode=Development, CpuCount=2, SocketCount=1) UID=0x822363D500000128FB261625C2D51F98
2010-06-02 17:11:40.353/4.469 Oracle Coherence GE 3.6.0.0 DPR3 <Info> (thread=main, member=n/a):
Started cluster Name=cluster:0xC4DB

Group{Address=224.3.6.0, Port=36000, TTL=4}

MasterMemberSet
  (
  ThisMember=Member(Id=1, Timestamp=2010-06-02 17:11:36.869, Address=130.35.99.213:8088,
MachineId=49877, Location=site:us.oracle.com,machine:tpfaeffl-lap7,process:4884,
Role=OracleHandsonCoherenceCache)
  OldestMember=Member(Id=1, Timestamp=2010-06-02 17:11:36.869, Address=130.35.99.213:8088,
MachineId=49877, Location=site:us.oracle.com,machine:tpfaeffl-lap7,process:4884,
Role=OracleHandsonCoherenceCache)
  ActualMemberSet=MemberSet(Size=1, BitSetCount=2
    Member(Id=1, Timestamp=2010-06-02 17:11:36.869, Address=130.35.99.213:8088, MachineId=49877,
Location=site:us.oracle.com,machine:tpfaeffl-lap7,process:4884, Role=OracleHandsonCoherenceCache)
    )
  RecycleMillis=1200000
  RecycleSet=MemberSet(Size=0, BitSetCount=0
    )
  )

TcpRing{Connections=[]}
IpMonitor{AddressListSize=0}
```

```
2010-06-02 17:11:40.431/4.547 Oracle Coherence GE 3.6.0.0 DPR3 <D5> (thread=Invocation:Management,
member=1): Service Management joined the cluster with senior service member 1
2010-06-02 17:11:40.697/4.813 Oracle Coherence GE 3.6.0.0 DPR3 <D5> (thread=DistributedCache,
member=1): Service DistributedCache joined the cluster with senior service member 1
Hello Cache
2010-06-02 17:11:40.775/4.891 Oracle Coherence GE 3.6.0.0 DPR3 <D4> (thread=ShutdownHook,
member=1): ShutdownHook: stopping cluster node
2010-06-02 17:11:40.775/4.891 Oracle Coherence GE 3.6.0.0 DPR3 <D5> (thread=Cluster, member=1):
Service Cluster left the cluster
Process exited with exit code 0.
```

Creating a Database Cache

In this section, create a cache backed by the Oracle database. This is also referred to as an "Oracle database cache".

1. Create an Oracle Database Cache
2. Create a Class to Define a Custom CacheStore
3. Modify the Cache Configuration File
4. Create a Class to Create the Database Cache
5. Run the Database Cache Application

Create an Oracle Database Cache

Follow these steps to use SQL*Plus and the Oracle Database 10g Express Edition (XE) to create an Oracle database cache. This section assumes that the database is installed on your system.

1. Invoke SQL*Plus.

   Navigate to **Start** then **All Programs** then **Oracle Database 10g Express Edition** then **Run SQL Command Line**.

2. Connect as `hr` user with `hr` as the password.

   ```
   connect hr/hr;
   ```

3. Create an Oracle Database table.

   Open a text editor and copy the following SQL code. Save the file as `dbscript.sql` in the `/home/oracle/labs/` folder.

Example 9–4 SQL Script for Creating a Database Table

```
CREATE TABLE HR.CATALOG(id VARCHAR(25) PRIMARY KEY, value VARCHAR(96));
INSERT INTO HR.CATALOG VALUES('catalog1', 'Tuning Undo Tablespace');
INSERT INTO HR.CATALOG VALUES('catalog2', 'Tuning Your View Objects');
```

4. Run the SQL script.

   Example 9–5 illustrates the output from the script.

Example 9–5 Running the SQL Script for Creating a Database Table

```
SQL*Plus: Release 10.2.0.1.0 - Production on Wed Jun 2 16:30:15 2010
Copyright (c) 1982, 2005, Oracle.  All rights reserved.
SQL> connect hr/hr;
Connected.
```

```
SQL> @/home/oracle/labs/dbscript.sql

Table created

1 row created

1 row created
```

Create a Class to Define a Custom CacheStore

Follow these steps to create a Java class that connects to the database and retrieves table data.

1. Create a Java class `DBCacheStore` in Oracle JDeveloper. See "Creating a Java Class" on page 2-13 if you need detailed information.
2. Create the code to connect to the database and get table data.

   Example 9–6 illustrates a possible solution. Copy the code to the `DBCacheStore` application in Oracle JDeveloper. The `DBCacheStore` application uses Java Database Connectivity (JDBC) to access Oracle Database, but you could use another mechanism, such as Hibernate or Java Data Objects (JDO), instead.

Example 9–6 Database CacheStore Implementation

```
package com.oracle.handson;

import com.tangosol.net.cache.CacheStore;
import com.tangosol.util.Base;

import java.sql.DriverManager;
import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.util.Collection;
import java.util.Iterator;
import java.util.LinkedList;
import java.util.List;
import java.util.Map;

public class DBCacheStore

        extends Base
        implements CacheStore {

        protected Connection m_con;
        protected String m_sTableName;
        private static final String DB_DRIVER   = "oracle.jdbc.OracleDriver";


            private static final String DB_URL      =
"jdbc:oracle:thin:@localhost:1521:XE";
            private static final String DB_USERNAME = "hr";
            private static final String DB_PASSWORD = "hr";

    public DBCacheStore(String sTableName)
        {
        m_sTableName = sTableName;

        configureConnection();
```

```
    }
protected void configureConnection()
    {
    try
        {
        Class.forName("oracle.jdbc.OracleDriver");
        m_con = DriverManager.getConnection(DB_URL, DB_USERNAME, DB_PASSWORD);
        m_con.setAutoCommit(true);
        }
    catch (Exception e)
        {
        throw ensureRuntimeException(e, "Connection failed");
        }
    }

public String getTableName()
    {
    return m_sTableName;
    }

public Connection getConnection()
    {
    return m_con;
    }

public Object load(Object oKey)
    {
    Object     oValue = null;
    Connection con    = getConnection();
    String     sSQL   = "SELECT id, value FROM " + getTableName()
                + " WHERE id = ?";
    try
        {
        PreparedStatement stmt = con.prepareStatement(sSQL);

        stmt.setString(1, String.valueOf(oKey));
        ResultSet rslt = stmt.executeQuery();
        if (rslt.next())
            {
            oValue = rslt.getString(2);
            if (rslt.next())
                {
                throw new SQLException("Not a unique key: " + oKey);
                }
            }
        stmt.close();
        }
    catch (SQLException e)
        {
        throw ensureRuntimeException(e, "Load failed: key=" + oKey);
        }
    return oValue;
    }

  public void store(Object oKey, Object oValue)
    {
    Connection con     = getConnection();
    String     sTable  = getTableName();
    String     sSQL;
```

```
    if (load(oKey) != null)
        {

        sSQL = "UPDATE " + sTable + " SET value = ? where id = ?";
        }
    else
        {

        sSQL = "INSERT INTO " + sTable + " (value, id) VALUES (?,?)";
        }
    try
        {
        PreparedStatement stmt = con.prepareStatement(sSQL);
        int i = 0;
        stmt.setString(++i, String.valueOf(oValue));
        stmt.setString(++i, String.valueOf(oKey));
        stmt.executeUpdate();
        stmt.close();
        }
    catch (SQLException e)
        {
        throw ensureRuntimeException(e, "Store failed: key=" + oKey);
        }
    }
public void erase(Object oKey)
    {
    Connection con  = getConnection();
    String     sSQL = "DELETE FROM " + getTableName() + " WHERE id=?";
    try
        {
        PreparedStatement stmt = con.prepareStatement(sSQL);

        stmt.setString(1, String.valueOf(oKey));
        stmt.executeUpdate();
        stmt.close();
        }
    catch (SQLException e)
        {
        throw ensureRuntimeException(e, "Erase failed: key=" + oKey);
        }
    }
public void eraseAll(Collection colKeys)
    {
    throw new UnsupportedOperationException();
    }



public Map loadAll(Collection colKeys)
    {
    throw new UnsupportedOperationException();
    }

public void storeAll(Map mapEntries)
    {
    throw new UnsupportedOperationException();
    }

public Iterator keys()
    {
```

```
        Connection con  = getConnection();
        String     sSQL = "SELECT id FROM " + getTableName();
        List       list = new LinkedList();

        try
            {
            PreparedStatement stmt = con.prepareStatement(sSQL);
            ResultSet         rslt = stmt.executeQuery();
            while (rslt.next())
                {
                Object oKey = rslt.getString(1);
                list.add(oKey);
                }
            stmt.close();
            }
        catch (SQLException e)
            {
            throw ensureRuntimeException(e, "Iterator failed");
            }

        return list.iterator();
        }

    }
```

Modify the Cache Configuration File

Modify the cache configuration file (`cache-config.xml`) that you created earlier for the database cache.

To connect a cache to a back-end database, a cache configuration file (`cache-config.xml`) element `cachestore-scheme` is required. The `cachestore-scheme` element must be configured with a custom class that implements either the `com.tangosol.net.cache.CacheLoader` or `com.tangosol.net.cache.CacheStore` interface.

Copy the cache configuration file for the database cache in Example 9–7 and replace the existing code in the `cache-config.xml` file in Oracle JDeveloper.

Example 9–7 Database Cache Configuration File

```
<?xml version="1.0" encoding="UTF-8" ?>
<cache-config>
    <caching-scheme-mapping>

  <!--
    Caches with names that start with 'DBBacked' will be created
    as distributed-db-backed.
    -->
  <cache-mapping>
   <cache-name>DBBacked*</cache-name>
   <scheme-name>distributed-db-backed</scheme-name>
  </cache-mapping>
 </caching-scheme-mapping>
 <caching-schemes>
  <!--
    DB Backed Distributed caching scheme.
    -->
  <distributed-scheme>
   <scheme-name>distributed-db-backed</scheme-name>
```

```
   <service-name>DistributedCache</service-name>
   <backing-map-scheme>
    <read-write-backing-map-scheme>
     <internal-cache-scheme>
      <class-scheme>
       <class-name>com.tangosol.util.ObservableHashMap</class-name>
      </class-scheme>
     </internal-cache-scheme>
     <cachestore-scheme>
      <class-scheme>
       <class-name>com.oracle.handson.DBCacheStore</class-name>
       <init-params>
        <init-param>
         <param-type>java.lang.String</param-type>
         <param-value>CATALOG</param-value>
        </init-param>
       </init-params>
      </class-scheme>
     </cachestore-scheme>
     <read-only>false</read-only>
     <!--
        To make this a write-through cache just change the value below to 0 (zero)
        -->
     <write-delay-seconds>0</write-delay-seconds>
    </read-write-backing-map-scheme>
   </backing-map-scheme>
   <listener/>
   <autostart>true</autostart>
  </distributed-scheme>
 </caching-schemes>
</cache-config>
```

In the cache configuration file, you have taken care of the following:

- Define a cache name pattern `DBBacked*`, which is mapped to a distributed caching scheme `distributed-db-backed`.
- Specify the `CacheStore` scheme in the distributed scheme using the class `coherence.DBCacheStore`, which implements the `CacheStore` interface.
- An `init` parameter for the database table that is at the back end of the cache is specified for the `DBCacheStore` class. The table name is specified in the `init-param` element. The `DBCacheStore` class performs database operations such as reading and writing cache entries.
- Coherence supports read/write caching of a data source for which the `read-write-backing-map` scheme is used. The `read-write-backing-map` scheme defines a backing map, which provides a size-limited cache of a persistent store. Here, you use the Write-Through mechanism. Oracle Coherence supports the types of read/write caching described in Table 9–2:

Table 9–2 Types of Read-Write Caching Supported by Coherence

Types of Read-Write Caching	Action
Read-Through	A cache entry is read into a cache from the database when required and made available to an application.
Write-Through	Updates to cache entries are synchronized with the database without delay.
Refresh-Ahead	Cache entries are refreshed periodically.

Table 9–2 (Cont.) Types of Read-Write Caching Supported by Coherence

Types of Read-Write Caching	Action
Write-Behind	Updates to cache entries are asynchronously written to a database after a delay specified in the write-delay-seconds element in the cache configuration file.

Create a Class to Create the Database Cache

Create a Java class `DatabaseCache` for the database cache in Oracle JDeveloper. The class must contain a `main` method. See "Creating a Java Class" on page 2-13 if you need detailed information.

In the class file, add code to add a cache entry, query a database cache, and retrieve a cache entry. Add the following methods: `createCache()`, `addEntry()`, `retrieveEntry()`, `eraseEntry()`, and `queryCache()`. You can copy the code that is listed in Example 9–8.

Example 9–8 Implementation for the Database Cache Class File

```
package com.oracle.handson;

import com.tangosol.net.CacheFactory;
import com.tangosol.net.NamedCache;
import com.tangosol.net.cache.ContinuousQueryCache;
import com.tangosol.util.Filter;
import com.tangosol.util.extractor.IdentityExtractor;
import com.tangosol.util.filter.LikeFilter;

import java.util.HashSet;

import java.util.Iterator;
import java.util.Map;
import java.util.Set;

public class DatabaseCache {
    NamedCache cache;
    public DatabaseCache() {
    }

    public void createCache() {
         cache = CacheFactory.getCache("DBBackedCache");
        //cache.put(new String("catalog3"), new String("Evolving Grid
Management"));
           // System.out.println((String) cache.get( "catalog3"));

    }

    public void addEntry() {

        cache.put(new String("catalog3"), new String("Tuning Grid Management"));
        cache.put(new String("catalog4"), new String("Tuning Coherence"));
        cache.put(new String("catalog5"), new String("Tuning Database"));
        //System.out.println((String) cache.get( "catalog3"));

    }

    public void retrieveEntry() {
    System.out.println((String) cache.get( "catalog3"));
    }
```

```
    public void eraseEntry() {
    cache.remove(new String("catalog3"));
    }

    public void queryCache() {
        Filter filter = new LikeFilter(IdentityExtractor.INSTANCE, "Tuning%",
'\\', true);
        HashSet hashSet=new HashSet();
        hashSet.add(new String("catalog3"));
        hashSet.add(new String("catalog4"));
        hashSet.add(new String("catalog5"));

        Map map=cache.getAll(hashSet);
        //ContinuousQueryCache queryCache = new ContinuousQueryCache(cache,
filter);
        //Set results = queryCache.entrySet(filter);
        Set results = cache.entrySet(filter);
       /* Set results = cache.entrySet(filter);*/



       // if(results.isEmpty())
       // System.out.println("Result Set Empty");
            for (Iterator i = results.iterator(); i.hasNext();)
                {
                Map.Entry e = (Map.Entry) i.next();
                System.out.println("Catalog ID: "+e.getKey() + ", Title: "+e.
getValue());
                }
    }


    public static void main(String[] args) {
        DatabaseCache databaseCache = new DatabaseCache();
        databaseCache.createCache();
       databaseCache.addEntry();
       //databaseCache.retrieveEntry();
        //databaseCache.eraseEntry();
       databaseCache.queryCache();
    }
}
```

Note the following features of the code:

- A `NamedCache` object is created using the `getCache()` method of the `CacheFactory` class in the `createCache()` method.

  ```
  NamedCache cache = CacheFactory.getCache("DBBackedCache");
  ```

- The `DBBackedCache` matches the cache pattern `DBBacked*` and is, therefore, mapped to a distributed caching scheme `distributed-db-backed` in the `cache-config.xml` file. Add a cache entry using the `put()` method of the `NamedCache` object.

  ```
  cache.put(new String("catalog3"), new String("Tuning Grid Management"));
  ```

- Because the Write-Through mechanism is used, the new cache entry also gets synchronized with the database; a new row is added to the `CATALOG` table. Comment out all the methods except the `createCache()` and `addEntry()` methods.

- When the `put()` method is invoked, the `store()` method, which maps the new cache entry to the database table `CATALOG` using JDBC, gets invoked in the `DBCacheStore` class. The output from the Oracle Coherence application is displayed in the Log window and a new cache entry is added. The output shows that the operational configuration deployment descriptor is loaded, the cache configuration is loaded, a new cluster is created, and the `DistributedCache` service has joined the cluster.
- The new cache entry may be removed with the `remove()` method of the `NamedCache` object.

```
cache.remove(new String("catalog3"));
```

- Bulk uploading of cache entries is performed using the `putAll()` method.
- A cache entry is retrieved using the `get()` method of the `NamedCache` object. For example, retrieving the cache entry for ID `catalog1`:

```
System.out.println((String) cache.get("catalog1"));
```

- When the `get()` method is invoked, the `load()` method, which retrieves database table data using JDBC, gets invoked in the `DBCacheStore` class.
- Bulk retrieval is performed using the `getAll()` method of the `NamedCache` object.
- Oracle Coherence supports searching for cache entries based on a search criteria using filters. Coherence filters are available in the `com.tangosol.util.filter` package. In Oracle Coherence Enterprise Edition and Grid Edition, indexes may be added to the Coherence cache to improve performance. You query the database cache using a `LikeFilter` filter, which matches cache entries with a specified pattern. To query a database cache, the cache entries must be created before querying, and the cache entries must be retrieved into the cache using the `get()` or `getAll()` method before a query using a filter may be performed. Therefore, you can retrieve database data and create a collection of cache entries using the `getAll()` method.

```
HashSet hashSet=new HashSet();
hashSet.add(new String("catalog1"));
hashSet.add(new String("catalog2"));
hashSet.add(new String("catalog3"));
Map map=cache.getAll(hashSet);
```

- A `LikeFilter` filter is created to search for cache entries starting with `Tuning`.

```
Filter filter = new LikeFilter(IdentityExtractor.INSTANCE, "Tuning%", '\\',
true);
```

- The database cache is queried using the `entrySet()` method with the `LikeFilter` filter.

```
Set results = cache.entrySet(filter);
```

- Iterate over the results of the query to output the cache entries retrieved.

```
for (Iterator i = results.iterator(); i.hasNext();) {
   Map.Entry e = (Map.Entry) i.next();
   System.out.println("Catalog ID: "+e.getKey() + ", Title: "+e.getValue());
}
```

- Oracle Coherence supports continuous query using the `com.tangosol.net.cache.ContinuousQueryCache` class. A continuous query is a query that is

kept up-to-date using a continuous query cache. In a `ContinuousQueryCache`, the results of a query are updated using event listeners on events that could change the results of the query. Create a `ContinuousQueryCache` object using the `NamedCache` object and the `LikeFilter` object.

```
ContinuousQueryCache queryCache = new ContinuousQueryCache(cache, filter );
```

- A result set is created using the `entrySet()` method.

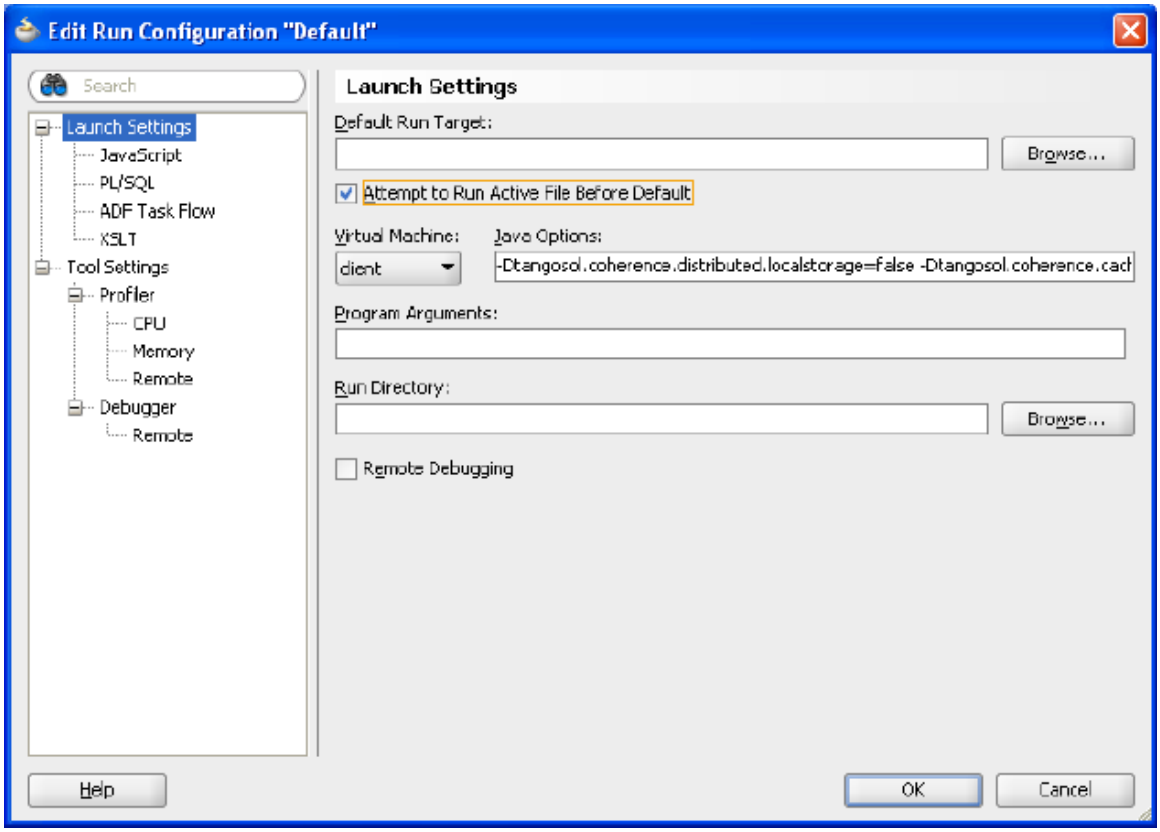
```
Set results = queryCache.entrySet(filter);
```

Run the Database Cache Application

Follow these steps to run the Oracle database cache application.

1. Modify the Run configuration for the `Interact` application to turn off local storage with `-Dtangosol.coherence.distributed.localstorage=false`.

Figure 9–3 Turning Off Local Storage



2. Stop any running cache servers. Start the cache server (`jpa-cache-server.cmd`).
3. Right-click the `DatabaseCache` application in Oracle JDeveloper and select **Run**. Figure 9–4 illustrates the expected results.

Figure 9–4 Results from Running the DatabaseCache Application

```
MasterMemberSet
  (
  ThisMember=Member(Id=3, Timestamp=2010-06-02 16:30:43.119,
Address=130.35.99.213:8090, MachineId=49877,
Location=site:us.oracle.com,machine:tpfaeffl-lap7,process:4812,
Role=OracleHandsonDatabaseCache)
  OldestMember=Member(Id=1, Timestamp=2010-06-02 16:21:06.681,
Address=130.35.99.213:8088, MachineId=49877,
Location=site:us.oracle.com,machine:tpfaeffl-lap7,process:1076, Role=CoherenceServer)
  ActualMemberSet=MemberSet(Size=2, BitSetCount=2
    Member(Id=1, Timestamp=2010-06-02 16:21:06.681, Address=130.35.99.213:8088,
MachineId=49877, Location=site:us.oracle.com,machine:tpfaeffl-lap7,process:1076,
Role=CoherenceServer)
    Member(Id=3, Timestamp=2010-06-02 16:30:43.119, Address=130.35.99.213:8090,
MachineId=49877, Location=site:us.oracle.com,machine:tpfaeffl-lap7,process:4812,
Role=OracleHandsonDatabaseCache)
    )
  RecycleMillis=1200000
  RecycleSet=MemberSet(Size=0, BitSetCount=0
    )
  )

TcpRing{Connections=[1]}
IpMonitor{AddressListSize=0}

2010-06-02 16:30:43.244/2.360 Oracle Coherence GE 3.6.0.0 DPR3 <D5>
(thread=Invocation:Management, member=3): Service Management joined the cluster with
senior service member 1
2010-06-02 16:30:43.525/2.641 Oracle Coherence GE 3.6.0.0 DPR3 <D5>
(thread=DistributedCache, member=3): Service DistributedCache joined the cluster with
senior service member 1
Catalog ID: catalog5, Title: Tuning Database
Catalog ID: catalog3, Title: Tuning Grid Management
Catalog ID: catalog4, Title: Tuning Coherence
Process exited with exit code 0.
```

If you receive any exceptions, such as the following:

```
java.lang.IllegalArgumentException: No scheme for cache: "cachename,
```

You may be able to remove them by editing the `cache-config.xml` file and replacing `DBBacked*` in the `<cache-name>` element with `*`. Save the file. Re-run the `DatabaseCache` application in Oracle JDeveloper. You should not see any exceptions now.

4. Note that because you are using a Write-Through cache, the database table also gets updated. From the SQL prompt, enter the following code:

```
select * from hr.catalog;
```

Example 9–9 illustrates the results.

Example 9–9 Output from the select Command

```
....
SQL> select * from hr.catalog;
```

```
-------------------------
VALUE
--------------------------------------------------------------------
catalog3
Tuning Grid Management
catalog4
Tuning Coherence

catalog5
Tuning Database


ID
-------------------------
VALUE
--------------------------------------------------------------------
catalog1
Tuning Undo Tablespace

catalog2
Tuning Your View Objects
```

10

Working with Security

This chapter describes how to apply security to a Coherence*Extend client. Coherence*Extend allows a wider range of consumers access to Coherence caches. These consumers include desktop applications, remote servers, and machines located across WAN connections.

This chapter has the following sections:

Introduction

Coherence*Extend consists of an extend client running outside the cluster and a proxy service running inside the cluster and hosted by one or more cache servers. The client APIs route all requests to the proxy. The proxy responds to client requests by delegating to Coherence clustered services, such as a partitioned or replicated cache service or an invocation service.

Since the extend client exists outside of the cluster, the issue of securing access to the cluster takes on greater importance. This chapter illustrates three techniques that you can use to secure access between the client and the cluster. The techniques include using identity token-based passwords, an entitled cache service, and an invocation service.

A detailed discussion of these security techniques and extend clients is beyond the scope of this tutorial. For more information on these topics, see the *Oracle Coherence Client Guide*.

Enabling Token-Based Security

You can implement token-based security to enable access between an extend client and an extend proxy in the cluster. To enable access between the extend client and an extend proxy the following files are typically required:

- client application files, which describes the functionality that wants to access the cluster
- cache configuration files, where the extend client and the extend proxy each have their own cache configuration

- operational override file, which overrides the operational and run-time settings in the default operational deployment descriptor
- server start-up files, where there will be a start-up file for the extend proxy and for the cache server in the cluster
- POF configuration deployment descriptor, which specifies custom data types when using POF to serialize objects.

To add token-based security, you must additionally provide identity transformer and asserter implementations. The transformer generates the token on the client side and the asserter validates it on the cluster side.

The following steps describe how to create and run an application for an extend client that uses token-based security to access the cluster.

1. Use a Security Helper File
2. Create an Identity Transformer
3. Create an Identity Asserter
4. Create the Password File
5. Enable the Identity Transformer and Asserter
6. Create a Cache Configuration File for the Extend Client
7. Create a Cache Configuration File for the Extend Proxy
8. Create a Start-Up File for a Cache Server with a Proxy Service
9. Create a Start-Up File for a Cache Server
10. Configure Project Classpaths and Runtime Properties
11. Run the Password Example

Use a Security Helper File

The examples in this chapter reference a security helper file which defines role-based security policies and access control to the cache. For the purposes of these examples, a file with simplified mappings is provided for you.

Cache access will be determined by a user's role. The security helper file defines several roles: `role_reader`, `role_writer`, and `role_admin`. It defines the mappings of various users to the roles, such as `BuckarooBanzai` to `ROLE_ADMIN`. It defines the mappings of roles to integer IDs, such as `ROLE_ADMIN` to `9`. The helper file also defines the cache name and the Invocation Service name used in the examples.

The key features of this file are the `login` and `checkAccess` methods. The `login` method takes a user name and constructs a simplified Distinguished Name (DN). It then associates a role with the name. `PofPrincipal` provides the `Principal` implementation.

The `checkAccess` method demonstrates where the authorization code is placed. It determines whether the user can access the cache based on a provided user role.

To create a new project and the security helper file:

1. Create a new project in JDeveloper named `Security`. Ensure that the package path is `com.oracle.handson`.

   See "Creating a New Project in an Existing Application" on page 2-11 if you need detailed information on creating a project.

2. Create a new Java file named `SecurityExampleHelper.java`.

   See "Creating a Java Class" on page 2-13 if you need detailed information on creating a Java class.

3. Copy the code illustrated in Example 10–1 into the file.

Example 10–1 A Security Helper File

```
package com.oracle.handson;

import com.tangosol.io.pof.PofPrincipal;

import com.tangosol.net.security.SecurityHelper;

import java.util.HashMap;
import java.util.HashSet;
import java.util.Iterator;
import java.util.Map;
import java.util.Set;

import java.security.Principal;

import javax.security.auth.Subject;


/**
* This class provides extremely simplified role based policies and access control.
*
*/
public class SecurityExampleHelper
    {
    // ----- static methods ---------------------------------------------------

    /**
    * Login the user.
    *
    * @param sName  the user name
    *
    * @return the authenticated user
    */
    public static Subject login(String sName)
        {
        // For simplicity, just create a Subject. Normally, this would be
        // done using JAAS.
        String sUserDN       = "CN=" + sName + ",OU=Yoyodyne";
        Set setPrincipalUser = new HashSet();

        setPrincipalUser.add(new PofPrincipal(sUserDN));
        // Map the user to a role
        setPrincipalUser.add(new PofPrincipal((String) s_mapUserToRole.
get(sName)));

        return new Subject(true, setPrincipalUser, new HashSet(), new HashSet());
        }

    /**
    * Assert that a Subject is associated with the calling thread with a
    * Principal representing the required role.
    *
    * @param sRoleRequired  the role required for the operation
```

```
*
* @throws SecurityException if a Subject is not associated with the
*         calling thread or does not have the specified role Principal
*/
public static void checkAccess(String sRoleRequired)
    {
    Subject subject = SecurityHelper.getCurrentSubject();

    if (subject == null)
        {
        throw new SecurityException("Access denied, authentication required");
        }

    Map     mapRoleToId   = s_mapRoleToId;
    Integer nRoleRequired = (Integer) mapRoleToId.get(sRoleRequired);

    for (Iterator iter = subject.getPrincipals().iterator(); iter.hasNext();)
        {
        Principal principal = (Principal) iter.next();
        String    sName     = principal.getName();

        if (sName.startsWith("role_"))
            {
            Integer nRolePrincipal = (Integer) mapRoleToId.get(sName);

            if (nRolePrincipal == null)
                {
                // invalid role
                break;
                }

            if (nRolePrincipal.intValue() >= nRoleRequired.intValue())
                {
                return;
                }
            }
        }

    throw new SecurityException("Access denied, insufficient privileges");
    }


// ----- constants  -----------------------------------------------------

public static final String ROLE_READER = "role_reader";

public static final String ROLE_WRITER = "role_writer";

public static final String ROLE_ADMIN  = "role_admin";

/**
* The cache name for security examples
*/
public static final String SECURITY_CACHE_NAME = "security";

/**
* The name of the InvocationService used by security examples.
*/
public static String INVOCATION_SERVICE_NAME = "ExtendTcpInvocationService";
```

```
// ----- static data  -------------------------------------------------

/**
* The map keyed by user name with the value being the user's role.
* Represents which user is in which role.
*/
private static Map s_mapUserToRole = new HashMap();

/**
* The map keyed by role name with the value the role id.
* Represents the numeric role identifier.
*/
private static Map s_mapRoleToId = new HashMap();


// ----- static initializer -------------------------------------------

static
    {
    // User to role mapping
    s_mapUserToRole.put("BuckarooBanzai", ROLE_ADMIN);
    s_mapUserToRole.put("JohnWhorfin", ROLE_WRITER);
    s_mapUserToRole.put("JohnBigboote", ROLE_READER);

    // Role to Id mapping
    s_mapRoleToId.put(ROLE_ADMIN, Integer.valueOf(9));
    s_mapRoleToId.put(ROLE_WRITER, Integer.valueOf(2));
    s_mapRoleToId.put(ROLE_READER, Integer.valueOf(1));
    }
}
```

Create an Identity Transformer

An identity transformer (`com.tangosol.net.security.IdentityTransformer`) is a client-side component that converts a Subject or Principal into an identity token. The token must be a type that Coherence knows how to serialize. Coherence automatically serializes the token at run time and sends it as part of the connection request to the proxy.

To create an identity transformer implementation:

1. Create a new Java class in the `Security` project named `PasswordIdentityTransformer`.

   See "Creating a Java Class" on page 2-13 if you need detailed information.

2. Import the `IdentityTransformer` interface. Ensure that `PasswordIdentityTransformer` implements `IdentityTransformer`.
3. Implement the `transformIdentity` method so that it performs the following tasks:
   - tests whether the `Subject` exists and is complete
   - gets the principal names from the `Subject` and saves them in a `String` array
   - constructs the token as a combination of the password plus the `Principal` names as Pof-able types

Example 10–2 illustrates a possible implementation of `PasswordIdentityTransformer`.

Example 10–2 Sample Identity Transformer Implementation

```
package com.oracle.handson;


import com.tangosol.net.security.IdentityTransformer;

import java.security.Principal;
import java.util.Iterator;
import java.util.Set;

import javax.security.auth.Subject;

/**
* PasswordIdentityTransformer creates a security token that contains the
* required password and then adds a list of Principal names.
*
*/
public class PasswordIdentityTransformer
        implements IdentityTransformer

    {
    // ----- IdentityTransformer interface ----------------------------------

    /**
    * Transform a Subject to a token that asserts an identity.
    *
    * @param subject the Subject representing a user.
    *
    * @return the token that asserts identity.
    *
    * @throws SecurityException if the identity transformation fails.
    */
    public Object transformIdentity(Subject subject)
            throws SecurityException
        {
        if (subject == null)
            {
            throw new SecurityException("Incomplete Subject");
            }

        Set setPrincipals = subject.getPrincipals();

        if (setPrincipals.isEmpty())
            {
            throw new SecurityException("Incomplete Subject");
            }

        String[] asPrincipalName = new String[setPrincipals.size() + 1];
        int      i               = 0;

        asPrincipalName[i++] = System.getProperty("coherence.password",
                "secret-password");

        for (Iterator iter = setPrincipals.iterator(); iter.hasNext();)
            {
            asPrincipalName[i++] = ((Principal) iter.next()).getName();
            }

        // The token consists of the password plus the principal names as an
        // array of pof-able types, in this case strings.
```

```
        return asPrincipalName;
        }
    }
```

Create an Identity Asserter

An identity asserter (`com.tangosol.net.security.IdentityAsserter`) is a cluster-side component that resides on the cache server that hosts an extend proxy service. The asserter validates that the token created by the identity transformer on the extend client contains the required credentials to access the cluster.

To create an identity asserter implementation:

1. Create a new Java class in the Security project named `PasswordIdentityAsserter`.

   See "Creating a Java Class" on page 2-13 if you need detailed information.

2. Import the `IdentityAsserter` interface. Ensure that `PasswordIdentityAsserter` implements `IdentityAsserter`.

3. Implement the `assertIdentity` method such that it:

   - validates that the token contains the correct password, and that the password is the first name in the token

Example 10–3 illustrates a a possible implementation of `PasswordIdentityAsserter`.

Example 10–3 Sample Identity Asserter Implementation

```
package com.oracle.handson;


import com.tangosol.io.pof.PofPrincipal;

import com.tangosol.net.security.IdentityAsserter;

import java.util.HashSet;
import java.util.Set;

import javax.security.auth.Subject;


/**
* PasswordIdentityAsserter asserts that the security token contains the
* required password and then constructs a Subject based on a list of
* Principal names.
*
*/
public class PasswordIdentityAsserter
        implements IdentityAsserter
    {
    // ----- IdentityAsserter interface -------------------------------------

    /**
    * Asserts an identity based on a token-based identity assertion.
    *
    * @param oToken the token that asserts identity.
    *
    * @return a Subject representing the identity.
    *
```

```
* @throws SecurityException if the identity assertion fails.
*/
public Subject assertIdentity(Object oToken)
        throws SecurityException
    {
    if (oToken instanceof Object[])
        {
        String   sPassword        = System.getProperty(
                "coherence.password", "secret-password");
        Set      setPrincipalUser = new HashSet();
        Object[] asName           = (Object[]) oToken;

        // first name must be password
        if (((String) asName[0]).equals(sPassword))
            {
            // prints the user name to server shell to ensure we are
            // communicating with it and to ensure user is validated
            System.out.println("Password validated for user: " + asName[1]);
            for (int i = 1, len = asName.length; i < len; i++)
                {
                setPrincipalUser.add(new PofPrincipal((String)asName[i]));
                }

            return new Subject(true, setPrincipalUser, new HashSet(),
                    new HashSet());
            }
        }
    throw new SecurityException("Access denied");
    }
}
```

Create the Password File

Create a Java file that will require a password to get a reference to a cache. Use the `SecurityExampleHelper.login("BuckarooBanzai")` to call the `login` method in `SecurityExampleHelper` to generate a token. At run time, the user name is associated with its Subject defined in the `SecurityExampleHelper` file. A token is generated from this Subject by `PasswordIdentityTransformer` and validated by `PasswordIdentityAsserter` as part of the connection request. If the validation succeeds, then a connection to the proxy and a reference to the cache is granted. Use `Subject.doas` to make the `Subject` available in the security context.

1. Create a new Java class with a `main` method in the `Security` project named `PasswordExample`.

   See "Creating a Java Class" on page 2-13 if you need detailed information.

2. Implement the `main` to get a reference to the cache.
3. Use `SecurityExampleHelper.login` to get a Subject for user `BuckarooBanzai`
4. Implement the `doAs` method to make the subject part of the Java security context. The subject will be available to any subsequent code. In this case `doAs` is implemented to validate whether the user can access the cache based on its defined role.

Example 10–4 illustrates a a possible implementation of `PasswordExample`.

Example 10–4 Sample Implementation to Run the Password Example

```
package com.oracle.handson;

import com.tangosol.net.CacheFactory;
import com.tangosol.net.NamedCache;

import java.io.IOException;

import java.security.PrivilegedExceptionAction;

import javax.security.auth.Subject;


/**
* This class shows how a Coherence Proxy can require a password to get a
* reference to a cache.
* <p>
* The PasswordIdentityTransformer will generate a security token that
* contains the password. The PasswordIdentityAsserter will validate the
* security token to enforce the password. The token generation and
* validation occurs automatically when a connection to the proxy is made.
*
*/
public class PasswordExample
    {
    // ----- static methods ---------------------------------------------------

    /**
    * Get a reference to the cache. Password will be required.
    */
    public static void main (String[] args){
         getCache();
           }
      public static void getCache()
        {
        System.out.println("------password example begins------");

        Subject subject = SecurityExampleHelper.login("BuckarooBanzai");

        try
            {
            NamedCache cache = (NamedCache) Subject.doAs(
                    subject, new PrivilegedExceptionAction()
                {
                public Object run()
                        throws Exception
                    {
                    NamedCache cache;

                    cache = CacheFactory.getCache(
                            SecurityExampleHelper.SECURITY_CACHE_NAME);
                    System.out.println("------password example succeeded------");
                    return cache;
                    }
                });
            }
        catch (Exception e)
            {
            // get exception if the password is invalid
            System.out.println("Unable to connect to proxy");
```

```
            e.printStackTrace();
            }
        System.out.println("------password example completed------");
        }

    }
```

Enable the Identity Transformer and Asserter

Configure an operational override file (`tangosol-coherence-override.xml`) to identify the classes that define the identity transformer (the class that transforms a `Subject` to a token on the extend client) and the identity asserter (the class that validates the token on the cluster).

1. Create an XML file in JDeveloper. Name the file `tangosol-coherence-override.xml` and store it in the `C:\home\oracle\labs\` directory.
2. Use the `identity-transformer` and `identity-asserter` elements within the `security-config` stanza to identify the full path to the `PasswordIdentityTransformer` and `PasswordIdentityAsserter` implementation classes, respectively. Set `subject-scope` to `true` to associate the identity from the current security context with the cache and remote invocation service references that are returned to the client.

Example 10–5 illustrates a possible implementation of `tangosol-coherence-override.xml`.

Example 10–5 Specifying an Identity Transformer and an Asserter

```
<?xml version='1.0'?>

<!DOCTYPE coherence SYSTEM "coherence.dtd">

<coherence>
  <security-config>
    <identity-transformer>
      <class-name>com.oracle.handson.PasswordIdentityTransformer</class-name>
    </identity-transformer>
    <identity-asserter>
      <class-name>com.oracle.handson.PasswordIdentityAsserter</class-name>
    </identity-asserter>
    <subject-scope>true</subject-scope>
  </security-config>
</coherence>
```

Create a Cache Configuration File for the Extend Client

The cache configuration file for the extend client routes cache operations to an extend proxy in the cluster. At run time, cache operations are not executed locally, but are sent to the extend proxy service.

To create a cache configuration file for an extend client:

1. Create an XML file in JDeveloper. Name the file `client-cache-config.xml` and save it in the `C:\home\oracle\labs\` directory.
2. Write the extend client cache configuration. The following list highlights some key elements:

- use the `cache-name` element to define `security` as the name of the cache. Note that there must be a cache defined in the cluster-side cache configuration that is also named `security`.
- use the `remote-cache-scheme` stanza to define the details about the remote cache
- use the `address` and `port` elements in the `tcp-initiator` stanza to identify the extend proxy service that is listening on address `localhost` at port `9099`
- use `defaults` and `serializer` with a value of `pof` to call the serializer for the custom POF configuration file (which you will create later in this chapter)

Example 10–6 illustrates a possible implementation for `client-cache-config.xml`.

Example 10–6 Sample Extend Client Cache Configuration File

```
<?xml version="1.0"?>
<!DOCTYPE cache-config SYSTEM "cache-config.dtd">

<cache-config>
  <defaults>
    <serializer>pof</serializer>
  </defaults>

  <caching-scheme-mapping>
    <cache-mapping>
      <cache-name>security</cache-name>
      <scheme-name>examples-remote</scheme-name>
    </cache-mapping>
  </caching-scheme-mapping>

  <caching-schemes>
    <remote-cache-scheme>
      <scheme-name>examples-remote</scheme-name>
      <service-name>ExtendTcpCacheService</service-name>
      <initiator-config>
        <tcp-initiator>
          <remote-addresses>
            <socket-address>
              <address system-property="tangosol.coherence.proxy.
address">localhost</address>
              <port system-property="tangosol.coherence.proxy.port">9099</port>
            </socket-address>
          </remote-addresses>
        </tcp-initiator>
      </initiator-config>
    </remote-cache-scheme>

    <remote-invocation-scheme>
      <scheme-name>remote-invocation-scheme</scheme-name>
      <service-name>ExtendTcpInvocationService</service-name>
      <initiator-config>
        <tcp-initiator>
          <remote-addresses>
            <socket-address>
              <address system-property="tangosol.coherence.proxy.
address">localhost</address>
              <port system-property="tangosol.coherence.proxy.port">9099</port>
```

```
            </socket-address>
          </remote-addresses>
          <connect-timeout>2s</connect-timeout>
        </tcp-initiator>
        <outgoing-message-handler>
          <request-timeout>5s</request-timeout>
        </outgoing-message-handler>
      </initiator-config>
    </remote-invocation-scheme>
  </caching-schemes>
</cache-config>
```

Create a Cache Configuration File for the Extend Proxy

Create a cache configuration file for the extend proxy service.

1. Create an XML file in JDeveloper. Name the file `examples-cache-config.xml` and save it in the `C:\home\oracle\labs\` directory.
2. Configure the extend proxy cache configuration file. The following list highlights some key elements:
   - use the `cache-name` element to define `security` as the name of the cache. Note that there must be a cache defined in the extend client cache configuration that is also named `security`.
   - use the `address` and `port` elements in the `acceptor-config` stanza to identify the extend proxy service that is listening on address `localhost` at port `9099`
   - use the `autostart` element with the `tangosol.coherence.extend.enabled` system property to prevent the cache server from running a proxy service.
   - use `defaults` and `serializer` with a value of `pof` to call the serializer for the custom POF configuration file (which you will create later in this chapter)

Example 10–7 illustrates a possible implementation for the `examples-cache-config.xml`

Example 10–7 Sample Cache Configuration File for the Proxy Server

```
<?xml version="1.0"?>

<!DOCTYPE cache-config SYSTEM "cache-config.dtd">

<cache-config>
  <defaults>
    <serializer>pof</serializer>
  </defaults>

  <caching-scheme-mapping>
    <cache-mapping>
      <cache-name>security</cache-name>
      <scheme-name>ExamplesPartitionedPofScheme</scheme-name>
    </cache-mapping>
  </caching-scheme-mapping>

  <caching-schemes>
    <distributed-scheme>
      <scheme-name>ExamplesPartitionedPofScheme</scheme-name>
      <service-name>PartitionedPofCache</service-name>
```

```
      <backing-map-scheme>
        <local-scheme>
          <!-- each node will be limited to 32MB -->
          <high-units>32M</high-units>
          <unit-calculator>binary</unit-calculator>
        </local-scheme>
      </backing-map-scheme>
      <autostart>true</autostart>
    </distributed-scheme>

    <!--
    Proxy Service scheme that allows remote clients to connect to the
    cluster over TCP/IP.
    -->
    <proxy-scheme>
      <scheme-name>secure-proxy</scheme-name>
      <service-name>ProxyService</service-name>

      <thread-count system-property="tangosol.coherence.extend.threads">2</thread-
count>

      <acceptor-config>
        <tcp-acceptor>
          <local-address>
            <address system-property="tangosol.coherence.extend.
address">localhost</address>
            <port system-property="tangosol.coherence.extend.port">9099</port>
            <reusable>true</reusable>
          </local-address>
        </tcp-acceptor>
      </acceptor-config>

       <autostart system-property="tangosol.coherence.extend.
enabled">false</autostart>
    </proxy-scheme>
  </caching-schemes>
</cache-config>
```

Create a Start-Up File for a Cache Server

Create a start-up file for a cache server cluster node. The start-up file must include the system properties to designate a proxy service and the cluster-side cache configuration file. You must also include the application class files and the XML configuration files on the classpath.

To create a start-up file for a cache server:

1. Create a start-up file in JDeveloper. Right-click the Security project and select **New**. In the **New Gallery**, select **General** in the **Categories** field and **File** in the Items field. In the **New File** dialog, enter `security-cache-server.cmd` as the file name and save it in the Coherence `\bin` directory (in this case, `C:\oracle\product\coherence\bin`).
2. Enter the commands to start the cache server. The key commands include:
   - the `tangosol.coherence.cacheconfig` system property to indicate the path to the cluster-side cache configuration file (in this case, `examples-cache-config.xml`)
   - the classpath defined to include the path to `coherence.jar` (`C:\oracle\product\lib\coherence.jar`), the `Security` project

classes (`C:\home\oracle\labs\Security\classes`), and the XML configuration files (`C:\home\oracle\labs`)

Note: Ensure that the XML configuration files (under `C:\home\oracle\labs`) appear before `coherence.jar` on the classpath. The classloader must encounter the custom POF configuration file (which you will create later in this chapter) before it references the one in `coherence.jar`.

Example 10–8 illustrates a possible implementation for `security-cache-server.cmd`.

Example 10–8 Start-Up File for a Cache Server

```
@echo off
@
@rem This will start a cache server
@
setlocal

:config
@rem specify the Coherence installation directory
set coherence_home=c:\oracle\product\coherence

@rem specify the JVM heap size
set memory=512m


:start
if not exist "%coherence_home%\lib\coherence.jar" goto instructions

if "%java_home%"=="" (set java_exec=java) else (set java_exec=%java_
home%\bin\java)


:launch


set COH_OPTS=%COH_OPTS% -Dtangosol.coherence.
cacheconfig=\home\oracle\labs\examples-cache-config.xml

"%java_exec%" -server -showversion %COH_OPTS% -cp C:\home\oracle\labs;"%coherence_
home%\lib\coherence.jar";C:\home\oracle\labs\Security\classes -Xms128m -Xmx128m
com.tangosol.net.DefaultCacheServer  %2 %3 %4 %5 %6 %7

goto exit

:instructions

echo Usage:
echo   ^<coherence_home^>\bin\cache-server.cmd
goto exit

:exit
endlocal
@echo on
```

Create a Start-Up File for a Cache Server with a Proxy Service

Create a file to start an extend proxy service on a cache server in the cluster. The extend client will connect to this service. The start-up file must include the system properties to designate a proxy service and the cluster-side cache configuration file. You must also include the application class files and the XML configuration files on the classpath.

For these examples, the cache server with proxy service start-up file will have the same configuration as the cache server start-up file, but it will include the system property to enable the extend proxy.

To create a start-up file for a cache server with a proxy service:

1. Create a start-up file in JDeveloper. Right-click the Security project and select **New**. In the **New Gallery**, select **General** in the **Categories** field and **File** in the Items field. In the **New File** dialog, enter `security-run-proxy.cmd` as the file name and save it in the Coherence `\bin` directory. (in this case, `C:\oracle\product\coherence\bin`).
2. Enter the commands to start the cache server. The key commands include:
   - the `tangosol.coherence.extend.enabled` system property to designate a proxy service
   - the `tangosol.coherence.cacheconfig` system property to indicate the path to the cluster-side cache configuration file (in this case, `examples-cache-config.xml`)
   - the classpath defined to include the path to `coherence.jar` (`C:\oracle\product\lib\coherence.jar`), the Security classes (`C:\home\oracle\labs\Security\classes`), and the XML configuration files (`C:\home\oracle\labs`)

   Note: Ensure that the XML configuration files (under `C:\home\oracle\labs`) appear before `coherence.jar` on the classpath. The classloader must encounter the custom POF configuration file (which you will create later in this chapter) before it references the one in `coherence.jar`.

Example 10–9 illustrates a possible implementation of the cluster-side cache server with proxy service start-up file, `security-run-proxy.cmd`.

Example 10–9 Start-Up File for a Cache Server with Proxy Service

```
@echo off
@
@REM this will start a proxy cache server
@
setlocal

:config

@rem specify the Coherence installation directory
set coherence_home=c:\oracle\product\coherence

@rem specify the JVM heap size
set memory=512m

:start
```

```
if not exist "%coherence_home%\lib\coherence.jar" goto instructions

if "%java_home%"=="" (set java_exec=java) else (set java_exec=%java_
home%\bin\java)


:launch

set COH_OPTS=%COH_OPTS% -Dtangosol.coherence.extend.enabled=true -Dtangosol.
coherence.cacheconfig=\home\oracle\labs\examples-cache-config.xml

"%JAVA_HOME%"\bin\java -server -showversion %COH_OPTS% -cp
C:\home\oracle\labs;"%coherence_home%\lib\coherence.
jar";C:\home\oracle\labs\Security\classes -Xms128m -Xmx128m com.tangosol.net.
DefaultCacheServer %2 %3 %4 %5 %6 %7

goto exit

:instructions

echo Usage:
echo   ^<coherence_home^>\bin\security-run-proxy.cmd
goto exit

:exit
endlocal
```

Configure Project Classpaths and Runtime Properties

Configure the classpath and run-time properties for the project.

1. In JDeveloper, compile the Java files in the project. Right-click the `Security` project and select **Make Security.jpr**.
2. Set the Java run-time options for the project.

   Right-click the `Security` project and select **Project Properties**. Select **Run\Debug\Profile** and edit the **Default** run configuration. In the **Java Options** field, enter the system properties to disable local storage and to identify the client-side cache configuration file `client-cache-config.xml`.

   ```
   -Dtangosol.coherence.distributed.localstorage=false -Dtangosol.coherence.
   cacheconfig=\home\oracle\labs\client-cache-config.xml
   ```

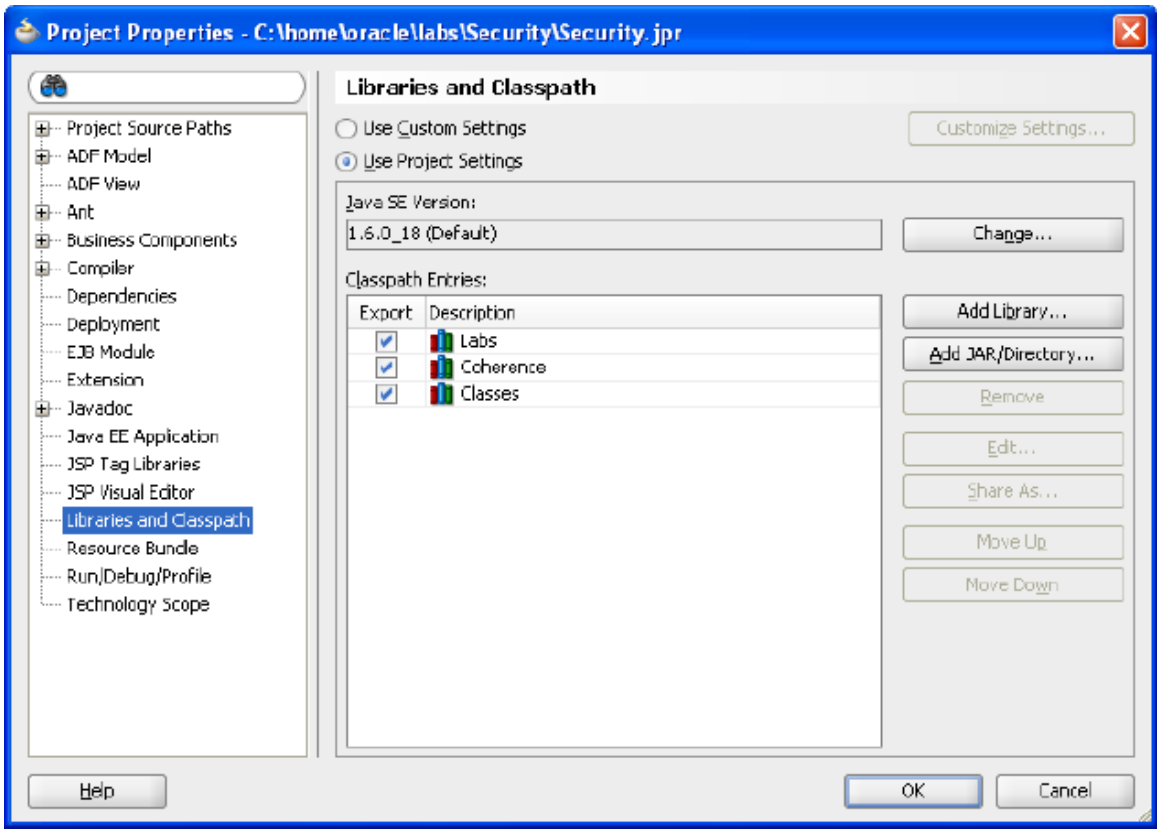
   Click **OK** to dismiss the **Edit Run** configuration window.
3. Set the libraries and classpaths for the project. In the **Project Properties** window, click **Libraries and Classpaths**. Add the `coherence.jar`, the `Security` project Java classes (`C:\home\oracle\labs\Security\classes`), and the directory containing the XML configuration files (`C:\home\oracle\labs`) to the project classpath.

   > **Note:** Ensure that the XML configuration files (under `C:\home\oracle\labs`) appear before `coherence.jar` on the classpath. The classloader must encounter the custom POF configuration file (which you will create later in this chapter) before it references the one in `coherence.jar`.

   The **Libraries and Classpath** window should look similar to Figure 10–1.

See "Changing Project Properties, Setting Runtime Configuration" on page 2-15 if you need detailed information on setting libraries and classpaths.

Figure 10–1 Libraries and Classpath for the Security Project



4. Click **OK** to dismiss the **Libraries and Classpath** dialog box, then **OK** to dismiss the **Project Properties** dialog box.
5. Save and compile the project. Right-click the project and select **Make Security.jpr**.

Run the Password Example

Run the password example to generate and validate the token, and pass it to the proxy service.

1. Compile the files in the Security project if you have not already done so.
2. Stop any running cache servers. Open a command prompt, and run the proxy server `security-run.proxy.cmd`.
3. Open another command prompt and run the cache server `security-cache-server.cmd`.
4. Run the password example: right-click `PasswordExample.java` and select **Run**.

You should see output similar to Example 10–10 in the JDeveloper Log window. The output indicates the start of the password example, the opening of the socket to the proxy server, and the completion of the example.

Example 10–10 Password Example Output in the JDeveloper Log Window

```
...
```

```
------password example begins------
2010-06-09 11:50:49.220/0.344 Oracle Coherence 3.6.0.0 DPR3 <Info> (thread=main, member=n/a):
Loaded operational configuration from "jar:file:/C:/oracle/product/coherence_3.5/lib/coherence.
jar!/tangosol-coherence.xml"
2010-06-09 11:50:49.220/0.344 Oracle Coherence 3.6.0.0 DPR3 <Info> (thread=main, member=n/a):
Loaded operational overrides from "jar:file:/C:/oracle/product/coherence_3.5/lib/coherence.
jar!/tangosol-coherence-override-dev.xml"
2010-06-09 11:50:49.220/0.344 Oracle Coherence 3.6.0.0 DPR3 <Info> (thread=main, member=n/a):
Loaded operational overrides from "file:/C:/home/oracle/labs/tangosol-coherence-override.xml"
2010-06-09 11:50:49.235/0.359 Oracle Coherence 3.6.0.0 DPR3 <D5> (thread=main, member=n/a):
Optional configuration override "/custom-mbeans.xml" is not specified

Oracle Coherence Version 3.6.0.0 DPR3 Build 16141
 Grid Edition: Development mode
Copyright (c) 2000, 2010, Oracle and/or its affiliates. All rights reserved.

2010-06-09 11:50:49.470/0.594 Oracle Coherence GE 3.6.0.0 DPR3 <Info> (thread=main, member=n/a):
Loaded cache configuration from "file:/C:/home/oracle/labs/client-cache-config.xml"
2010-06-09 11:50:49.673/0.797 Oracle Coherence GE 3.6.0.0 DPR3 <Info>
(thread=ExtendTcpCacheService:TcpInitiator, member=n/a): Loaded POF configuration from
"jar:file:/C:/oracle/product/coherence_3.5/lib/coherence.jar!/pof-config.xml"
2010-06-09 11:50:49.688/0.812 Oracle Coherence GE 3.6.0.0 DPR3 <Info>
(thread=ExtendTcpCacheService:TcpInitiator, member=n/a): Loaded included POF configuration from
"jar:file:/C:/oracle/product/coherence_3.5/lib/coherence.jar!/coherence-pof-config.xml"
2010-06-09 11:50:49.782/0.906 Oracle Coherence GE 3.6.0.0 DPR3 <D5>
(thread=ExtendTcpCacheService:TcpInitiator, member=n/a): Started:
TcpInitiator{Name=ExtendTcpCacheService:TcpInitiator, State=(SERVICE_STARTED), ThreadCount=0,
Codec=Codec(Format=POF), Serializer=com.tangosol.io.pof.ConfigurablePofContext, PingInterval=0,
PingTimeout=0, RequestTimeout=0, ConnectTimeout=0, SocketProvider=SystemSocketProvider,
RemoteAddresses=[hostname/130.35.99.213:9099], KeepAliveEnabled=true, TcpDelayEnabled=false,
ReceiveBufferSize=0, SendBufferSize=0, LingerTimeout=-1}
2010-06-09 11:50:49.798/0.922 Oracle Coherence GE 3.6.0.0 DPR3 <D5> (thread=main, member=n/a):
Opening Socket connection to 130.35.99.213:9099
2010-06-09 11:50:49.798/0.922 Oracle Coherence GE 3.6.0.0 DPR3 <Info> (thread=main, member=n/a):
Connected to 130.35.99.213:9099
------password example succeeded------
------password example completed------
Process exited with exit code 0.
```

You should see a response in the proxy server shell similar to Example 10–11. It lists the CN and OU values from the distinguished name and whether the password was validated.

Example 10–11 Response from the Cache Server Running the Proxy Service Shell

```
...
2010-06-09 11:50:39.829/2.781 Oracle Coherence GE 3.6.0.0 DPR3 <Info> (thread=main, member=2):
Services
  (
  ClusterService{Name=Cluster, State=(SERVICE_STARTED, STATE_JOINED), Id=0, Version=3.6,
OldestMemberId=1}
  InvocationService{Name=Management, State=(SERVICE_STARTED), Id=1, Version=3.1, OldestMemberId=6}
  PartitionedCache{Name=PartitionedPofCache, State=(SERVICE_STARTED), LocalStorage=enabled,
PartitionCount=257, BackupCount=1, AssignedPartitions=128,
 BackupPartitions=129}
  ProxyService{Name=ProxyService, State=(SERVICE_STARTED), Id=7, Version=3.2, OldestMemberId=6}
  )

Started DefaultCacheServer...
```

```
Password validated for user: CN=BuckarooBanzai,OU=Yoyodyne
Password validated for user: CN=BuckarooBanzai,OU=Yoyodyne
Password validated for user: CN=BuckarooBanzai,OU=Yoyodyne
```

Including Role-Based Access Control to the Cluster

This section describes how to create an example that employs role-based policies to access the cluster. The code will log in to get a `Subject` with a user-ID assigned to a particular role. It will get a cache reference running in the context of the `Subject`, and then attempt various cache operations. Depending on the role granted to the user, the cache operation is allowed or denied. Note that the role mapping and role-based authorization in the example is simplified and not intended for real security use.

For example, a user with a "writer" role is allowed to put and get. A user with a "reader" role can get, but not put. A user with a "writer" role cannot destroy a cache. However, a user with an "admin" role can destroy a cache.

Note that when the cache reference is created in the context of a `Subject`, that identity is permanently associated with that reference. Any use of that cache reference is on behalf of that identity.

The example will use the `PasswordIdentityTransformer` and `PasswordIdentityAsserter` classes that you created in the previous section. The `PasswordIdentityTransformer` will generate a security token that contains the password, the user ID, and the roles. The `PasswordIdentityAsserter` (running in the proxy) will validate the security token to enforce the password, and construct a `Subject` with the proper user ID and roles. The production and assertion of the security token happens automatically.

Follow these steps to create the example:

1. Define which User Roles are Entitled to Access Cache Methods
2. Apply the Entitlements to the Cache Service
3. Create the Access Control Example Program
4. Edit the Cluster-Side Cache Configuration File
5. Run the Access Control Example

Define which User Roles are Entitled to Access Cache Methods

Create a Java file that allows access to cache methods on the basis of a user's role. To do this, you can apply access permissions to a wrapped `NamedCache` using the `Subject` passed from the client by using Coherence*Extend. The implementation will allow only clients with a specified role to access the wrapped `NamedCache`.

The class that you create in this section should extend the `com.tangosol.net.cache.WrapperNamedCache` class. This class is a convenience function that enables you to secure the methods on the `NamedCache` interface.

To determine which user role can access a cache method, include a call to `SecurityExampleHelper.checkAccess` in each cache method's implementation. As the argument to `checkAccess`, provide the user role that is permitted to access the method. Close the implementation with a call to `super`.

For example, the following code indicates that users with the `admin` role can destroy the cache.

```
public void destroy()
    {
```

```
        SecurityExampleHelper.checkAccess(SecurityExampleHelper.ROLE_ADMIN);
        super.destroy();
        }
```

In this example, users with the `reader` role can call the `aggregate` method.

```
    public Object aggregate(Filter filter, EntryAggregator agent)
        {
        SecurityExampleHelper.checkAccess(SecurityExampleHelper.ROLE_READER);
        return super.aggregate(filter, agent);
        }
```

To create a file that determines which user role can call cache methods:

1. Create a new Java class named `EntitledNamedCache` in the Security project.

   See "Creating a Java Class" on page 2-13 if you need detailed information.

2. Ensure that the class imports and extends `WrapperNamedCache`.

3. Import the `Filter`, `MapListener`, `ValueExtractor`, `Collection`, `Comparator`, `Map`, and `Set` classes. The methods in `WrapperNamedCache` (and by extension, `EntitledNamedCache`) use arguments with these types.

4. Implement the methods in `EntitledNamedCache` such that only a user with a specific role can call the method.

Example 10–12 illustrates a possible implementation of `EntitledNamedCache`.

Example 10–12 Entitled Named Cache

```
package com.oracle.handson;


import com.tangosol.net.NamedCache;

import com.tangosol.net.cache.WrapperNamedCache;


import com.tangosol.util.Filter;
import com.tangosol.util.MapListener;
import com.tangosol.util.ValueExtractor;

import java.util.Collection;
import java.util.Comparator;
import java.util.Map;
import java.util.Set;


/**
* Example WrapperNamedCache that demonstrates how entitlements can be applied
* to a wrapped NamedCache using the Subject passed from the client through
* Coherence*Extend. This implementation only allows clients with a specified
* role to access the wrapped NamedCache.
*
*/
public class EntitledNamedCache
        extends WrapperNamedCache
    {
    /**
    * Create a new EntitledNamedCache.
    *
    * @param cache  the wrapped NamedCache
```

```
*/
public EntitledNamedCache(NamedCache cache)
    {
    super(cache, cache.getCacheName());
    }


// ----- NamedCache interface -------------------------------------------

/**
* {@inheritDoc}
*/
public void release()
    {
    SecurityExampleHelper.checkAccess(SecurityExampleHelper.ROLE_READER);
    super.release();
    }

/**
* {@inheritDoc}
*/
public void destroy()
    {
    SecurityExampleHelper.checkAccess(SecurityExampleHelper.ROLE_ADMIN);
    super.destroy();
    }

/**
* {@inheritDoc}
*/
public Object put(Object oKey, Object oValue, long cMillis)
    {
    SecurityExampleHelper.checkAccess(SecurityExampleHelper.ROLE_WRITER);
    return super.put(oKey, oValue, cMillis);
    }

/**
* {@inheritDoc}
*/
public void addMapListener(MapListener listener)
    {
    SecurityExampleHelper.checkAccess(SecurityExampleHelper.ROLE_READER);
    super.addMapListener(listener);
    }

/**
* {@inheritDoc}
*/
public void removeMapListener(MapListener listener)
    {
    SecurityExampleHelper.checkAccess(SecurityExampleHelper.ROLE_WRITER);
    super.removeMapListener(listener);
    }

/**
* {@inheritDoc}
*/
public void addMapListener(MapListener listener, Object oKey, boolean fLite)
    {
    SecurityExampleHelper.checkAccess(SecurityExampleHelper.ROLE_READER);
```

```
        super.addMapListener(listener, oKey, fLite);
        }

    /**
    * {@inheritDoc}
    */
    public void removeMapListener(MapListener listener, Object oKey)
        {
        SecurityExampleHelper.checkAccess(SecurityExampleHelper.ROLE_WRITER);
        super.removeMapListener(listener, oKey);
        }

    /**
    * {@inheritDoc}
    */
    public void addMapListener(MapListener listener, Filter filter, boolean fLite)
        {
        SecurityExampleHelper.checkAccess(SecurityExampleHelper.ROLE_READER);
        super.addMapListener(listener, filter, fLite);
        }

    /**
    * {@inheritDoc}
    */
    public void removeMapListener(MapListener listener, Filter filter)
        {
        SecurityExampleHelper.checkAccess(SecurityExampleHelper.ROLE_WRITER);
        super.removeMapListener(listener, filter);
        }

    /**
    * {@inheritDoc}
    */
    public int size()
        {
        SecurityExampleHelper.checkAccess(SecurityExampleHelper.ROLE_READER);
        return super.size();
        }

    /**
    * {@inheritDoc}
    */
    public void clear()
        {
        SecurityExampleHelper.checkAccess(SecurityExampleHelper.ROLE_WRITER);
        super.clear();
        }

    /**
    * {@inheritDoc}
    */
    public boolean isEmpty()
        {
        SecurityExampleHelper.checkAccess(SecurityExampleHelper.ROLE_READER);
        return super.isEmpty();
        }

    /**
    * {@inheritDoc}
    */
```

```
public boolean containsKey(Object oKey)
    {
    SecurityExampleHelper.checkAccess(SecurityExampleHelper.ROLE_READER);
    return super.containsKey(oKey);
    }

/**
* {@inheritDoc}
*/
public boolean containsValue(Object oValue)
    {
    SecurityExampleHelper.checkAccess(SecurityExampleHelper.ROLE_READER);
    return super.containsValue(oValue);
    }

/**
* {@inheritDoc}
*/
public Collection values()
    {
    SecurityExampleHelper.checkAccess(SecurityExampleHelper.ROLE_READER);
    return super.values();
    }

/**
* {@inheritDoc}
*/
public void putAll(Map map)
    {
    SecurityExampleHelper.checkAccess(SecurityExampleHelper.ROLE_WRITER);
    super.putAll(map);
    }

/**
* {@inheritDoc}
*/
public Set entrySet()
    {
    SecurityExampleHelper.checkAccess(SecurityExampleHelper.ROLE_READER);
    return super.entrySet();
    }

/**
* {@inheritDoc}
*/
public Set keySet()
    {
    SecurityExampleHelper.checkAccess(SecurityExampleHelper.ROLE_READER);
    return super.keySet();
    }

/**
* {@inheritDoc}
*/
public Object get(Object oKey)
    {
    SecurityExampleHelper.checkAccess(SecurityExampleHelper.ROLE_READER);
    return super.get(oKey);
    }
```

```
/**
* {@inheritDoc}
*/
public Object remove(Object oKey)
    {
    SecurityExampleHelper.checkAccess(SecurityExampleHelper.ROLE_WRITER);
    return super.remove(oKey);
    }

/**
* {@inheritDoc}
*/
public Object put(Object oKey, Object oValue)
    {
    SecurityExampleHelper.checkAccess(SecurityExampleHelper.ROLE_WRITER);
    return super.put(oKey, oValue);
    }

/**
* {@inheritDoc}
*/
public Map getAll(Collection colKeys)
    {
    SecurityExampleHelper.checkAccess(SecurityExampleHelper.ROLE_READER);
    return super.getAll(colKeys);
    }

/**
* {@inheritDoc}
*/
public boolean lock(Object oKey, long cWait)
    {
    SecurityExampleHelper.checkAccess(SecurityExampleHelper.ROLE_WRITER);
    return super.lock(oKey, cWait);
    }

/**
* {@inheritDoc}
*/
public boolean lock(Object oKey)
    {
    SecurityExampleHelper.checkAccess(SecurityExampleHelper.ROLE_WRITER);
    return super.lock(oKey);
    }

/**
* {@inheritDoc}
*/
public boolean unlock(Object oKey)
    {
    SecurityExampleHelper.checkAccess(SecurityExampleHelper.ROLE_WRITER);
    return super.unlock(oKey);
    }

/**
* {@inheritDoc}
*/
public Set keySet(Filter filter)
    {
    SecurityExampleHelper.checkAccess(SecurityExampleHelper.ROLE_READER);
```

```
        return super.keySet(filter);
        }

    /**
    * {@inheritDoc}
    */
    public Set entrySet(Filter filter)
        {
        SecurityExampleHelper.checkAccess(SecurityExampleHelper.ROLE_READER);
        return super.entrySet(filter);
        }

    /**
    * {@inheritDoc}
    */
    public Set entrySet(Filter filter, Comparator comparator)
        {
        SecurityExampleHelper.checkAccess(SecurityExampleHelper.ROLE_READER);
        return super.entrySet(filter, comparator);
        }

    /**
    * {@inheritDoc}
    */
    public void addIndex(ValueExtractor extractor, boolean fOrdered, Comparator
comparator)
        {
        SecurityExampleHelper.checkAccess(SecurityExampleHelper.ROLE_WRITER);
        super.addIndex(extractor, fOrdered, comparator);
        }

    /**
    * {@inheritDoc}
    */
    public void removeIndex(ValueExtractor extractor)
        {
        SecurityExampleHelper.checkAccess(SecurityExampleHelper.ROLE_WRITER);
        super.removeIndex(extractor);
        }

    /**
    * {@inheritDoc}
    */
    public Object invoke(Object oKey, EntryProcessor agent)
        {
        SecurityExampleHelper.checkAccess(SecurityExampleHelper.ROLE_WRITER);
        return super.invoke(oKey, agent);
        }

    /**
    * {@inheritDoc}
    */
    public Map invokeAll(Collection collKeys, EntryProcessor agent)
        {
        SecurityExampleHelper.checkAccess(SecurityExampleHelper.ROLE_WRITER);
        return super.invokeAll(collKeys, agent);
        }

    /**
    * {@inheritDoc}
```

```
*/
public Map invokeAll(Filter filter, EntryProcessor agent)
    {
    SecurityExampleHelper.checkAccess(SecurityExampleHelper.ROLE_WRITER);
    return super.invokeAll(filter, agent);
    }

/**
* {@inheritDoc}
*/
public Object aggregate(Collection collKeys, EntryAggregator agent)
    {
    SecurityExampleHelper.checkAccess(SecurityExampleHelper.ROLE_READER);
    return super.aggregate(collKeys, agent);
    }

/**
* {@inheritDoc}
*/
public Object aggregate(Filter filter, EntryAggregator agent)
    {
    SecurityExampleHelper.checkAccess(SecurityExampleHelper.ROLE_READER);
    return super.aggregate(filter, agent);
    }


// ----- helper methods ---------------------------------------------------

/**
* Return the wrapped NamedCache.
*
* @return  the wrapped CacheService
*/
public NamedCache getNamedCache()
    {
    return (NamedCache) getMap();
    }
}
```

Apply the Entitlements to the Cache Service

Create a file that demonstrates how access entitlements can be applied to a wrapped `CacheService` using the `Subject` passed from the client through Coherence*Extend. The implementation should delegate access control for cache operations to the `EntitledNamedCache` you created in the previous section.

The class that you create should extend `com.tangosol.net.WrapperCacheService`. This is a convenience function that allows you to secure the methods on the `CacheService`. It also provides a mechanism to delegate between the cache service on the proxy and the client request.

Implement the methods `ensureCache`, `releaseCache`, and `destroyCache` such that only users with specific roles can use them. This will include a call to `SecurityExampleHelper.checkAccess` with a specific user role as its argument. For example, the following code ensures that only users with role `admin` can destroy the cache.

```
public void destroyCache(NamedCache map)
    {
    if (map instanceof EntitledNamedCache)
```

```
            {
            EntitledNamedCache cache =  (EntitledNamedCache) map;
            SecurityExampleHelper.checkAccess(SecurityExampleHelper.ROLE_ADMIN);
            map = cache.getNamedCache();
            }
        super.destroyCache(map);
        }
```

To create a file that applies entitlements to access the cache service:

1. Create a new Java class named `EntitledCacheService` in the `Security` project.
2. Ensure that the class imports and extends `WrapperCacheService`.
3. Implement the `ensureCache`, `releaseCache`, and `destroyCache` methods such that only users with specific roles can use them.

Example 10–13 illustrates a possible implementation of `EntitledCacheService`.

Example 10–13 Entitled Cache Service

```
package com.oracle.handson;


import com.tangosol.net.CacheService;
import com.tangosol.net.NamedCache;
import com.tangosol.net.WrapperCacheService;


/**
* Example WrapperCacheService that demonstrates how entitlements can be
* applied to a wrapped CacheService using the Subject passed from the
* client through Coherence*Extend. This implementation delegates access control
* for cache operations to the EntitledNamedCache.
*
*/
public class EntitledCacheService
        extends WrapperCacheService
    {
    /**
    * Create a new EntitledCacheService.
    *
    * @param service     the wrapped CacheService
    */
    public EntitledCacheService(CacheService service)
        {
        super(service);
        }


    // ----- CacheService interface -----------------------------------------

    /**
    * {@inheritDoc}
    */
    public NamedCache ensureCache(String sName, ClassLoader loader)
        {
        SecurityExampleHelper.checkAccess(SecurityExampleHelper.ROLE_READER);
        return new EntitledNamedCache(super.ensureCache(sName, loader));
        }
```

```
/**
* {@inheritDoc}
*/
public void releaseCache(NamedCache map)
    {
    if (map instanceof EntitledNamedCache)
        {
        EntitledNamedCache cache =  (EntitledNamedCache) map;
        SecurityExampleHelper.checkAccess(SecurityExampleHelper.ROLE_READER);
        map = cache.getNamedCache();
        }
    super.releaseCache(map);
    }

/**
* {@inheritDoc}
*/
public void destroyCache(NamedCache map)
    {
    if (map instanceof EntitledNamedCache)
        {
        EntitledNamedCache cache =  (EntitledNamedCache) map;
        SecurityExampleHelper.checkAccess(SecurityExampleHelper.ROLE_ADMIN);
        map = cache.getNamedCache();
        }
    super.destroyCache(map);
    }
}
```

Create the Access Control Example Program

Create a file to run the access control example. The role policies are defined in `SecurityExampleHelper`. The `EntitledCacheService` and `EntitledNamedCache` files enforce the policies.

The program should specify various users as arguments to `SecurityHelperFile.login`, and then attempt to perform cache read, write, and destroy operations. Based on the entitlement policies defined in `EntitledCacheService` and `EntitledNamedCache` the operations should succeed or fail.

1. Create a new Java class with a `main` method in the Security project named `AccessControlExample`.

   See "Creating a Java Class" on page 2-13 if you need detailed information.

2. Implement the `main` to access the cache.

3. Specify users defined in the `SecurityExampleHelper` file as arguments to its `login` method.

4. For each user, try to execute read (`get`), write (`put`), and `destroy` operations on the cache and provide "Success" or "Failure" messages in response.

Example 10–14 illustrates a possible implementation of `AccessControlExample.java`.

Example 10–14 Sample Program to Run the Access Control Example

```
package com.oracle.handson;


import com.tangosol.net.CacheFactory;
```

```
import com.tangosol.net.InvocationService;
import com.tangosol.net.NamedCache;
import java.security.PrivilegedExceptionAction;

import java.util.Map;

import javax.security.auth.Subject;


/**
* This class demonstrates simplified role based access control.
* <p>
* The role policies are defined in SecurityExampleHelper. Enforcmenent
* is done by EntitledCacheService and EntitledNamedCache.
*
*/
public class AccessControlExample
    {
    // ----- static methods ---------------------------------------------------

     public static void main (String[] args){
           accessCache();
             }
    /**
    * Demonstrate role based access to the cache.
    */
    public static void accessCache()
        {
        System.out.println("------cache access control example begins------");

        Subject subject = SecurityExampleHelper.login("JohnWhorfin");

        // Someone with writer role can write and read
        try
            {
            NamedCache cache = (NamedCache) Subject.doAs(
                    subject, new PrivilegedExceptionAction()
                {
                public Object run()
                        throws Exception
                    {
                    return CacheFactory.getCache(SecurityExampleHelper.SECURITY_
CACHE_NAME);
                    }
                });
            cache.put("myKey", "myValue");
            cache.get("myKey");
            System.out.println("    Success: read and write allowed");
            }
        catch (Exception e)
            {
            // get exception if not allowed to perform the operation
            e.printStackTrace();
            }

        // Someone with reader role can read but not write
        subject = SecurityExampleHelper.login("JohnBigboote");
        try
            {
            NamedCache cache = (NamedCache) Subject.doAs(
```

```
                    subject, new PrivilegedExceptionAction()
                {
                public Object run()
                        throws Exception
                    {
                    return CacheFactory.getCache(SecurityExampleHelper.SECURITY_
CACHE_NAME);
                    }
                });
            cache.get("myKey");
            System.out.println("    Success: read allowed");
            cache.put("anotherKey", "anotherValue");
            }
        catch (Exception e)
            {
            // get exception if not allowed to perform the operation
            System.out.println("    Success: Correctly cannot write");
            }

        // Someone with writer role cannot call destroy
        subject = SecurityExampleHelper.login("JohnWhorfin");
        try
            {
            NamedCache cache = (NamedCache) Subject.doAs(
                    subject, new PrivilegedExceptionAction()
                {
                public Object run()
                        throws Exception
                    {
                    return CacheFactory.getCache(SecurityExampleHelper.SECURITY_
CACHE_NAME);
                    }
                });
            cache.destroy();
            }
        catch (Exception e)
            {
            // get exception if not allowed to perform the operation
            System.out.println("    Success: Correctly cannot " +
                    "destroy the cache");
            }

        // Someone with admin role can call destroy
        subject = SecurityExampleHelper.login("BuckarooBanzai");
        try
            {
            NamedCache cache = (NamedCache) Subject.doAs(
                    subject, new PrivilegedExceptionAction()
                {
                public Object run()
                        throws Exception
                    {
                    return CacheFactory.getCache(SecurityExampleHelper.SECURITY_
CACHE_NAME);
                    }
                });
            cache.destroy();
            System.out.println("    Success: Correctly allowed to " +
                    "destroy the cache");
            }
```

```
        catch (Exception e)
            {
            // get exception if not allowed to perform the operation
            e.printStackTrace();
            }
        System.out.println("------cache access control example completed------");
        }
    }
```

Edit the Cluster-Side Cache Configuration File

Edit the cluster-side cache configuration file `examples-cache-config.xml`. Specify the full path of the class name of the cache service in the `cache-service-proxy` stanza under `proxy-config`. The `cache-service-proxy` stanza contains the configuration information for a cache service proxy managed by a proxy service.

In this case, the cache service class name is `com.oracle.handson.EntitledCacheService` proxy and the `param-type` should be `com.tangosol.net.CacheService`.

Example 10–15 illustrates the XML code to add to the configuration.

Example 10–15 Cache Service Proxy Configuration for a Cluster-Side Cache Configuration

```
...
      <proxy-config>
        <cache-service-proxy>
          <class-name>com.oracle.handson.EntitledCacheService</class-name>
          <init-params>
            <init-param>
              <param-type>com.tangosol.net.CacheService</param-type>
              <param-value>{service}</param-value>
            </init-param>
         </init-params>
        </cache-service-proxy>
      </proxy-config>
...
```

Run the Access Control Example

Run the access control example to demonstrate how access to the cache can be granted or denied, based on a user's role.

1. Compile the files in the Security project if you have not done so already.
2. Stop any running cache servers. Start the cache server running the proxy service with `security-run-proxy.cmd`.
3. Start the cache server in the cluster with `security-cache-server.cmd`.
4. Right-click the `AccessControlExample.java` file and select **Run**.

You should see output similar to Example 10–16 in the JDeveloper Log window. The messages correspond to the various users specified in `AccessControlExample` trying to execute read, write, and destroy operations on the cache.

- the `Success: read and write allowed` message corresponds to user with role `writer` attempting to read from and write to the cache
- the `Success: read allowed` message corresponds to the user with role `reader` attempting to read from the cache

- the `Success: Correctly cannot write` message corresponds to the user with role `reader` attempting to write to the cache
- the `Success: Correctly cannot destroy the cache` message corresponds to the user with role `writer` attempting to destroy the cache
- the `Success: Correctly allowed to destroy the cache` message corresponds to the user with role `admin` attempting to destroy the cache

Example 10–16 Access Control Example Output in the JDeveloper Log Window

```
...
------cache access control example begins------
2010-06-10 17:07:51.500/0.344 Oracle Coherence 3.6.0.0 DPR3 <Info> (thread=main, member=n/a):
Loaded operational configuration from "jar:file:/C:/oracle/product0529/coherence_3.5/lib/coherence.
jar!/tangosol-coherence.xml"
2010-06-10 17:07:51.500/0.344 Oracle Coherence 3.6.0.0 DPR3 <Info> (thread=main, member=n/a):
Loaded operational overrides from "jar:file:/C:/oracle/product0529/coherence_3.5/lib/coherence.
jar!/tangosol-coherence-override-dev.xml"
2010-06-10 17:07:51.500/0.344 Oracle Coherence 3.6.0.0 DPR3 <Info> (thread=main, member=n/a):
Loaded operational overrides from "file:/C:/home/oracle/labs/tangosol-coherence-override.xml"
2010-06-10 17:07:51.500/0.344 Oracle Coherence 3.6.0.0 DPR3 <D5> (thread=main, member=n/a):
Optional configuration override "/custom-mbeans.xml" is not specified

Oracle Coherence Version 3.6.0.0 DPR3 Build 16141
 Grid Edition: Development mode
Copyright (c) 2000, 2010, Oracle and/or its affiliates. All rights reserved.

2010-06-10 17:07:51.703/0.547 Oracle Coherence GE 3.6.0.0 DPR3 <Info> (thread=main, member=n/a):
Loaded cache configuration from "file:/C:/home/oracle/labs/client-cache-config.xml"
2010-06-10 17:07:51.906/0.750 Oracle Coherence GE 3.6.0.0 DPR3 <Info>
(thread=ExtendTcpCacheService:TcpInitiator, member=n/a): Loaded POF configuration from
"jar:file:/C:/oracle/product0529/coherence_3.5/lib/coherence.jar!/pof-config.xml"
2010-06-10 17:07:51.906/0.750 Oracle Coherence GE 3.6.0.0 DPR3 <Info>
(thread=ExtendTcpCacheService:TcpInitiator, member=n/a): Loaded included POF configuration from
"jar:file:/C:/oracle/product0529/coherence_3.5/lib/coherence.jar!/coherence-pof-config.xml"
2010-06-10 17:07:52.015/0.859 Oracle Coherence GE 3.6.0.0 DPR3 <D5>
(thread=ExtendTcpCacheService:TcpInitiator, member=n/a): Started:
TcpInitiator{Name=ExtendTcpCacheService:TcpInitiator, State=(SERVICE_STARTED), ThreadCount=0,
Codec=Codec(Format=POF), Serializer=com.tangosol.io.pof.ConfigurablePofContext, PingInterval=0,
PingTimeout=0, RequestTimeout=0, ConnectTimeout=0, SocketProvider=SystemSocketProvider,
RemoteAddresses=[tpfaeffl-lap7/130.35.99.213:9099], KeepAliveEnabled=true, TcpDelayEnabled=false,
ReceiveBufferSize=0, SendBufferSize=0, LingerTimeout=-1}
2010-06-10 17:07:52.031/0.875 Oracle Coherence GE 3.6.0.0 DPR3 <D5> (thread=main, member=n/a):
Opening Socket connection to 130.35.99.213:9099
2010-06-10 17:07:52.031/0.875 Oracle Coherence GE 3.6.0.0 DPR3 <Info> (thread=main, member=n/a):
Connected to 130.35.99.213:9099
    Success: read and write allowed
2010-06-10 17:07:52.125/0.969 Oracle Coherence GE 3.6.0.0 DPR3 <D5>
(thread=ExtendTcpCacheService:TcpInitiator, member=n/a): Started:
TcpInitiator{Name=ExtendTcpCacheService:TcpInitiator, State=(SERVICE_STARTED), ThreadCount=0,
Codec=Codec(Format=POF), Serializer=com.tangosol.io.pof.ConfigurablePofContext, PingInterval=0,
PingTimeout=0, RequestTimeout=0, ConnectTimeout=0, SocketProvider=SystemSocketProvider,
RemoteAddresses=[tpfaeffl-lap7/130.35.99.213:9099], KeepAliveEnabled=true, TcpDelayEnabled=false,
ReceiveBufferSize=0, SendBufferSize=0, LingerTimeout=-1}
2010-06-10 17:07:52.125/0.969 Oracle Coherence GE 3.6.0.0 DPR3 <D5> (thread=main, member=n/a):
Opening Socket connection to 130.35.99.213:9099
2010-06-10 17:07:52.125/0.969 Oracle Coherence GE 3.6.0.0 DPR3 <Info> (thread=main, member=n/a):
Connected to 130.35.99.213:9099
    Success: read allowed
    Success: Correctly cannot write
```

```
    Success: Correctly cannot destroy the cache
2010-06-10 17:07:52.187/1.031 Oracle Coherence GE 3.6.0.0 DPR3 <D5>
(thread=ExtendTcpCacheService:TcpInitiator, member=n/a): Started:
TcpInitiator{Name=ExtendTcpCacheService:TcpInitiator, State=(SERVICE_STARTED), ThreadCount=0,
Codec=Codec(Format=POF), Serializer=com.tangosol.io.pof.ConfigurablePofContext, PingInterval=0,
PingTimeout=0, RequestTimeout=0, ConnectTimeout=0, SocketProvider=SystemSocketProvider,
RemoteAddresses=[tpfaeffl-lap7/130.35.99.213:9099], KeepAliveEnabled=true, TcpDelayEnabled=false,
ReceiveBufferSize=0, SendBufferSize=0, LingerTimeout=-1}
2010-06-10 17:07:52.187/1.031 Oracle Coherence GE 3.6.0.0 DPR3 <D5> (thread=main, member=n/a):
Opening Socket connection to 130.35.99.213:9099
2010-06-10 17:07:52.187/1.031 Oracle Coherence GE 3.6.0.0 DPR3 <Info> (thread=main, member=n/a):
Connected to 130.35.99.213:9099
    Success: Correctly allowed to destroy the cache
------cache access control example completed------
Process exited with exit code 0.
```

Example 10–17 lists the output in the shell where the cache server is running the proxy service. Notice that the security exceptions in the output, correspond to the `Success: Correctly cannot write` and `Success: Correctly cannot destroy the cache` messages in the JDeveloper Log window.

Example 10–17 Output for the Cache Server Running the Proxy Service

```
2010-06-10 17:07:25.468/18.156 Oracle Coherence GE 3.6.0.0 DPR3 <D5> (thread=Cluster, member=1):
Member 2 joined Service ProxyService with senior member 1
Password validated for user: CN=BuckarooBanzai,OU=Yoyodyne
Password validated for user: CN=BuckarooBanzai,OU=Yoyodyne
Password validated for user: CN=BuckarooBanzai,OU=Yoyodyne
Password validated for user: role_writer
Password validated for user: role_writer
Password validated for user: role_writer
Password validated for user: role_reader
Password validated for user: role_reader
Password validated for user: role_reader
2010-06-10 17:07:52.140/44.828 Oracle Coherence GE 3.6.0.0 DPR3 <D5>
(thread=Proxy:ProxyService:TcpAcceptorWorker:0, member=1): An exception occurred while processing a
PutRequest for Service=Proxy:ProxyService:TcpAcceptor: java.lang.SecurityException: Access denied,
insufficient privileges
        at com.oracle.handson.SecurityExampleHelper.checkAccess(SecurityExampleHelper.java:91)
        at com.oracle.handson.EntitledNamedCache.put(EntitledNamedCache.java:66)
        at com.tangosol.coherence.component.net.extend.proxy.NamedCacheProxy.
put$Router(NamedCacheProxy.CDB:1)
        at com.tangosol.coherence.component.net.extend.proxy.NamedCacheProxy.put(NamedCacheProxy.
CDB:2)
        at com.tangosol.coherence.component.net.extend.messageFactory.NamedCacheFactory$PutRequest.
onRun(NamedCacheFactory.CDB:6)
        at com.tangosol.coherence.component.net.extend.message.Request.run(Request.CDB:4)
        at com.tangosol.coherence.component.net.extend.proxy.NamedCacheProxy.
onMessage(NamedCacheProxy.CDB:11)
        at com.tangosol.coherence.component.net.extend.Channel$MessageAction.run(Channel.CDB:13)
        at java.security.AccessController.doPrivileged(Native Method)
        at javax.security.auth.Subject.doAs(Subject.java:337)
        at com.tangosol.coherence.component.net.extend.Channel.execute(Channel.CDB:29)
        at com.tangosol.coherence.component.net.extend.Channel.receive(Channel.CDB:26)
        at com.tangosol.coherence.component.util.daemon.queueProcessor.service.
Peer$DaemonPool$WrapperTask.run(Peer.CDB:9)
        at com.tangosol.coherence.component.util.DaemonPool$WrapperTask.run(DaemonPool.CDB:32)
        at com.tangosol.coherence.component.util.DaemonPool$Daemon.onNotify(DaemonPool.CDB:63)
        at com.tangosol.coherence.component.util.Daemon.run(Daemon.CDB:42)
        at java.lang.Thread.run(Thread.java:619)
```

```
2010-06-10 17:07:52.140/44.828 Oracle Coherence GE 3.6.0.0 DPR3 <D5>
(thread=Proxy:ProxyService:TcpAcceptorWorker:1, member=1): An exception occurred while processing a
DestroyCacheRequest for Service=Proxy:ProxyService:TcpAcceptor: java.lang.SecurityException: Access
denied, insufficient privileges

        at com.oracle.handson.SecurityExampleHelper.checkAccess(SecurityExampleHelper.java:91)
        at com.oracle.handson.EntitledCacheService.destroyCache(EntitledCacheService.java:64)
        at com.tangosol.coherence.component.net.extend.messageFactory.
CacheServiceFactory$DestroyCacheRequest.onRun(CacheServiceFactory.CDB:6)
        at com.tangosol.coherence.component.net.extend.message.Request.run(Request.CDB:4)
        at com.tangosol.coherence.component.net.extend.proxy.serviceProxy.CacheServiceProxy.
onMessage(CacheServiceProxy.CDB:9)
        at com.tangosol.coherence.component.net.extend.Channel$MessageAction.run(Channel.CDB:13)
        at java.security.AccessController.doPrivileged(Native Method)
        at javax.security.auth.Subject.doAs(Subject.java:337)
        at com.tangosol.coherence.component.net.extend.Channel.execute(Channel.CDB:29)
        at com.tangosol.coherence.component.net.extend.Channel.receive(Channel.CDB:26)
        at com.tangosol.coherence.component.util.daemon.queueProcessor.service.
Peer$DaemonPool$WrapperTask.run(Peer.CDB:9)
        at com.tangosol.coherence.component.util.DaemonPool$WrapperTask.run(DaemonPool.CDB:32)
        at com.tangosol.coherence.component.util.DaemonPool$Daemon.onNotify(DaemonPool.CDB:63)
        at com.tangosol.coherence.component.util.Daemon.run(Daemon.CDB:42)
        at java.lang.Thread.run(Thread.java:619)

Password validated for user: CN=BuckarooBanzai,OU=Yoyodyne
Password validated for user: CN=BuckarooBanzai,OU=Yoyodyne
Password validated for user: CN=BuckarooBanzai,OU=Yoyodyne
```

Including Role-Based Access Control to an Invocable Object

An invocation service cluster service allows extend clients to execute invocable objects on the cluster. This example demonstrates how you can use role-based policies to determine which users can run the invocable objects.

For example, a user with a "writer" role is allowed to run an invocable object. A user with a "reader" role may not.

In this example you will create a simple invocable object that can be called from a client program. Since it will be serializable, you must also list it in a POF configuration file. You will also create a invocation service program that will test whether a user can execute methods on the service based on the user's role.

As in the previous example, this example will use `PasswordIdentityTransformer` to generate a security token that contains the password, the user ID, and the roles. The `PasswordIdentityAsserter` (running in the proxy) will be used to validate the security token to enforce the password, and construct a `Subject` with the proper user ID and roles. The production and assertion of the security token happens automatically.

Follow these steps to create the example:

1. Create an Invocable Object
2. Create an Entitled Invocation Service
3. Create the Access Invocation Service Example Program
4. Edit the Cluster-Side Cache Configuration File
5. Create a POF Configuration File

6. Run the Access Invocation Service Example

Create an Invocable Object

Create an implementation of a simple invocable object that will be used by an entitled Invocation Service. For example, the invocable object can be written to increment and return an integer.

To create an invocable object:

1. Create a new Java class named `ExampleInvocable` in the `Security` project.

   See "Creating a Java Class" on page 2-13 if you need detailed information.

2. Import the `Invocable` and `InvocationService` interfaces. Since this class will be working with serializable objects, also import `PortableObject`, `PofReader` and `PofWriter`.

3. Ensure that the `ExampleInvocable` class implements `Invocable` and `PortableObject`.

4. Implement `ExampleInvocable` to increment an integer and return the result.

5. Implement the `PofReader.readExternal` and `PofWriter.writeExternal` methods.

Example 10–18 illustrates a possible implementation of `ExampleInvocable.java`.

Example 10–18 A Sample Invocable Object

```
package com.oracle.handson;


import com.tangosol.io.pof.PortableObject;
import com.tangosol.io.pof.PofReader;
import com.tangosol.io.pof.PofWriter;

import com.tangosol.net.Invocable;
import com.tangosol.net.InvocationService;

import java.io.IOException;


/**
 * Invocable implementation that increments and returns a given integer.
 */
public class ExampleInvocable
        implements Invocable, PortableObject
    {
    // ----- constructors ---------------------------------------------

    /**
     * Default constructor.
     */
    public ExampleInvocable()
        {
        }


    // ----- Invocable interface --------------------------------------

    /**
```

```
     * {@inheritDoc}
     */
    public void init(InvocationService service)
        {
        m_service = service;
        }

    /**
     * {@inheritDoc}
     */
    public void run()
        {
        if (m_service != null)
            {
            m_nValue++;
            }
        }

    /**
     * {@inheritDoc}
     */
    public Object getResult()
        {
        return new Integer(m_nValue);
        }


    // ----- PortableObject interface ---------------------------------

    /**
     * {@inheritDoc}
     */
    public void readExternal(PofReader in)
            throws IOException
        {
        m_nValue = in.readInt(0);
        }

    /**
     * {@inheritDoc}
     */
    public void writeExternal(PofWriter out)
            throws IOException
        {
        out.writeInt(0, m_nValue);
        }


    // ----- data members ---------------------------------------------

    /**
     * The integer value to increment.
     */
    private int m_nValue;

    /**
     * The InvocationService that is executing this Invocable.
     */
    private transient InvocationService m_service;
    }
```

Create an Entitled Invocation Service

This example shows how a remote invocation service can be wrapped to provide access control. Access entitlements can be applied to a wrapped `InvocationService` using the `Subject` passed from the client by using Coherence*Extend. This implementation only allows clients with a specified role to access the wrapped invocation service.

The class that you create should extend the `com.tangosol.net.WrapperInvocationService` class. This is a convenience function that allow you to secure the methods on `InvocationService`. It also provides a mechanism to delegate between the invocation service on the proxy and the client request.

To create an entitled invocation service:

1. Create a new Java class named `EntitledInvocationService` in the `Security` project.
2. Import the `Invocable`, `InvocationObserver`, `InvocationService`, `WrapperInvocationService`, `Map`, and `Set` interfaces. Ensure that `EntitledInvocationService` class extends `WrapperInvocationService`.
3. Implement the `query` and `execute` methods. In the implementations, include a call to `SecurityExampleHelper.checkAccess` to determine whether a specified user role, in this case, `ROLE_WRITER`, can access these operations.

Example 10–19 illustrates a possible implementation of `EntitledInvocationService.java`.

Example 10–19 A Sample Entitled Invocation Service

```
package com.oracle.handson;


import com.tangosol.net.Invocable;
import com.tangosol.net.InvocationObserver;
import com.tangosol.net.InvocationService;
import com.tangosol.net.WrapperInvocationService;

import java.util.Map;
import java.util.Set;


/**
* Example WrapperInvocationService that demonstrates how entitlements can be
* applied to a wrapped InvocationService using the Subject passed from the
* client through Coherence*Extend. This implementation only allows clients with a
* specified role to access the wrapped InvocationService.
*
*/
public class EntitledInvocationService
        extends WrapperInvocationService
    {
    /**
    * Create a new EntitledInvocationService.
    *
    * @param service  the wrapped InvocationService
    */
    public EntitledInvocationService(InvocationService service)
        {
        super(service);
        }
```

```
    // ----- InvocationService interface ------------------------------------

    /**
    * {@inheritDoc}
    */
    public void execute(Invocable task, Set setMembers, InvocationObserver
observer)
        {
        SecurityExampleHelper.checkAccess(SecurityExampleHelper.ROLE_WRITER);
        super.execute(task, setMembers, observer);
        }

    /**
    * {@inheritDoc}
    */
    public Map query(Invocable task, Set setMembers)
        {
        SecurityExampleHelper.checkAccess(SecurityExampleHelper.ROLE_WRITER);
        return super.query(task, setMembers);
        }
    }
```

Create the Access Invocation Service Example Program

Create a program to run the access invocation service example. The objective of the program should be to test whether various users defined in `SecurityExampleHelper` are able to access and run an invocable object. The enforcement of the role-based policies should be provided by `EntitledInvocationService`.

To create a program to run the Access Invocation Service example:

1. Create a Java class with a `main` method in the `Security` project named `AccessInvocationServiceExample.java`.

   See "Creating a Java Class" on page 2-13 if you need detailed information.

2. Among other classes, import `ExampleInvocable`, `CacheFactory`, and `InvocationService`.

3. Implement the `main` method to invoke the `accessInvocationService` method.

4. Implement `accessInvocationService` so that various users defined in `SecurityExampleHelper` attempt to log in to the service and run the object defined in `ExampleInvocable`. Use the `SecurityExampleHelper.login` method to test whether various users can access the Invocable Service.

Example 10–20 illustrates a possible implementation of `AccessInvocationServiceExample.java`.

Example 10–20 Sample Program to Run the Access Invocation Service Example

```
package com.oracle.handson;

import com.oracle.handson.ExampleInvocable;

import com.tangosol.net.CacheFactory;
import com.tangosol.net.InvocationService;
```

```
import java.security.PrivilegedExceptionAction;

import javax.security.auth.Subject;


/**
* This class demonstrates simplified role based access control for the
* invocation service.
* <p>
* The role policies are defined in SecurityExampleHelper. Enforcmenent
* is done by EntitledInvocationService.
*
*/
public class AccessInvocationServiceExample
    {
    /**
    * Invoke the example
    *
    * @param asArg  command line arguments (ignored in this example)
    */
    public static void main(String[] asArg)
        {
        accessInvocationService();
        }

    /**
    * Access the invocation service
    */
    public static void accessInvocationService()
        {
        System.out.println("------InvocationService access control example " +
                "begins------");

        // Someone with writer role can run invocables
        Subject subject = SecurityExampleHelper.login("JohnWhorfin");

        try
            {
            InvocationService service = (InvocationService) Subject.doAs(
                    subject, new PrivilegedExceptionAction()
                {
                public Object run()
                    {
                    return CacheFactory.getService(
                            SecurityExampleHelper.INVOCATION_SERVICE_NAME);
                    }
                });
             service.query(new ExampleInvocable(), null);
             System.out.println("    Success: Correctly allowed to " +
                    "use the invocation service");
            }
        catch (Exception e)
            {
            // get exception if not allowed to perform the operation
            e.printStackTrace();
            }

        // Someone with reader role cannot cannot run invocables
        subject = SecurityExampleHelper.login("JohnBigboote");
        try
```

```
            {
            InvocationService service = (InvocationService) Subject.doAs(
                    subject, new PrivilegedExceptionAction()
                {
                public Object run()
                    {
                    return CacheFactory.getService(
                            SecurityExampleHelper.INVOCATION_SERVICE_NAME);
                    }
                });
            service.query(new ExampleInvocable(), null);
            }
        catch (Exception ee)
            {
            System.out.println("    Success: Correctly unable to " +
                    "use the invocation service");
            }
        System.out.println("------InvocationService access control example " +
                "completed------");
        }
    }
```

Edit the Cluster-Side Cache Configuration File

Edit the examples-cache-config.xml file to add the full path to the invocation service to the invocation-service-proxy stanza under the proxy-config. The invocation-service-proxy stanza contains the configuration information for an invocation service proxy managed by a proxy service.

In this case, the invocation service class name is com.oracle.handson.EntitledInvocationService and its param-type should be com.tangosol.net.InvocationService.

Example 10–21 Invocation Service Proxy Configuration for a Cluster-Side Cache

```
...
   <proxy-config>
    ...
     <invocation-service-proxy>
       <class-name>com.oracle.handson.EntitledInvocationService</class-name>
       <init-params>
         <init-param>
           <param-type>com.tangosol.net.InvocationService</param-type>
           <param-value>{service}</param-value>
         </init-param>
       </init-params>
     </invocation-service-proxy>
   </proxy-config>
...
```

Create a POF Configuration File

Create a POF configuration file named security-pof-config.xml in the C:\home\oracle\labs directory where ExampleInvocable is declared as a user type.

Example 10–22 POF Configuration File with ExampleInvocable User Type

```
<?xml version="1.0"?>
```

```
<!DOCTYPE pof-config SYSTEM "pof-config.dtd">

<pof-config>
  <user-type-list>

    <!-- coherence POF user types -->
    <include>coherence-pof-config.xml</include>

    <!-- com.tangosol.examples package -->
    <user-type>
      <type-id>1007</type-id>
      <class-name>com.oracle.handson.ExampleInvocable</class-name>
    </user-type>

  </user-type-list>

  <allow-interfaces>true</allow-interfaces>
  <allow-subclasses>true</allow-subclasses>
</pof-config>
```

Run the Access Invocation Service Example

Run the access invocation service example to demonstrate how access to the invocable object can be granted or denied, based on a user's role.

1. Stop any running cache servers. Start the cache server running the proxy service with `security-run-proxy.cmd`.
2. Start the cache server in the cluster with `security-cache-server.cmd`.
3. Compile the `Security` project in JDeveloper if you have not done so already.
4. Right-click the `AccessInvocationServiceExample.java` file and select **Run**.

You should see output similar to Example 10–23 in the JDeveloper Log window. The messages correspond to the users specified in `AccessInvocationServiceExample` trying to run the invocable object `ExampleInvocable`.

- the message `Success: Correctly allowed to use the invocation service` corresponds to the user with role `writer` attempting to run `ExampleInvocable`.
- the message `Success: Correctly unable to use the invocation service` corresponds to the user with role `reader` attempting to run `ExampleInvocable`.

Example 10–23 JDeveloper Log Window

```
C:\oracle\product0529\jdk160_18\bin\javaw.exe -client -classpath
C:\home\oracle\labs\.
adf;C:\home\oracle\labs\Security\classes;C:\home\oracle\labs;C:\oracle\product0529
\coherence_3.5\lib\coherence.jar -Dweblogic.webservice.client.proxyusername=tom.
pfaeffle -Dweblogic.webservice.client.proxypassword=p3rtha -Djavax.net.ssl.
trustStore=C:\oracle\product\wlserver_10.3\server\lib\DemoTrust.jks -Dhttp.
proxyHost=www-proxy.us.oracle.com -Dhttp.proxyPort=80 -Dhttp.nonProxyHosts=*.
local|*oraclecorp.com|*oracle.com|localhost|localhost.localdomain|127.0.0.
1|::1|tpfaeffl-lap7.us.oracle.com|tpfaeffl-lap7 -Dhttps.proxyHost=www-proxy.us.
oracle.com -Dhttps.proxyPort=80 -Dhttps.nonProxyHosts=*.local|*oraclecorp.
com|*oracle.com|localhost|localhost.localdomain|127.0.0.1|::1|tpfaeffl-lap7.us.
oracle.com|tpfaeffl-lap7 -Dtangosol.coherence.distributed.localstorage=false -
Dtangosol.coherence.cacheconfig=\home\oracle\labs\client-cache-config.xml com.
oracle.handson.AccessInvocationServiceExample
```

```
------InvocationService access control example begins------
2010-06-17 12:17:04.362/0.391 Oracle Coherence 3.6.0.0 DPR3 <Info> (thread=main,
member=n/a): Loaded operational configuration from
"jar:file:/C:/oracle/product0529/coherence_3.5/lib/coherence.jar!/tangosol-
coherence.xml"
2010-06-17 12:17:04.362/0.391 Oracle Coherence 3.6.0.0 DPR3 <Info> (thread=main,
member=n/a): Loaded operational overrides from
"jar:file:/C:/oracle/product0529/coherence_3.5/lib/coherence.jar!/tangosol-
coherence-override-dev.xml"
2010-06-17 12:17:04.362/0.391 Oracle Coherence 3.6.0.0 DPR3 <Info> (thread=main,
member=n/a): Loaded operational overrides from
"file:/C:/home/oracle/labs/tangosol-coherence-override.xml"
2010-06-17 12:17:04.362/0.391 Oracle Coherence 3.6.0.0 DPR3 <D5> (thread=main,
member=n/a): Optional configuration override "/custom-mbeans.xml" is not specified

Oracle Coherence Version 3.6.0.0 DPR3 Build 16141
 Grid Edition: Development mode
Copyright (c) 2000, 2010, Oracle and/or its affiliates. All rights reserved.

2010-06-17 12:17:04.580/0.609 Oracle Coherence GE 3.6.0.0 DPR3 <Info>
(thread=main, member=n/a): Loaded cache configuration from
"file:/C:/home/oracle/labs/client-cache-config.xml"
2010-06-17 12:17:04.799/0.828 Oracle Coherence GE 3.6.0.0 DPR3 <Info>
(thread=ExtendTcpInvocationService:TcpInitiator, member=n/a): Loaded POF
configuration from "file:/C:/home/oracle/labs/security-pof-config.xml"
2010-06-17 12:17:04.799/0.828 Oracle Coherence GE 3.6.0.0 DPR3 <Info>
(thread=ExtendTcpInvocationService:TcpInitiator, member=n/a): Loaded included POF
configuration from "jar:file:/C:/oracle/product0529/coherence_3.5/lib/coherence.
jar!/coherence-pof-config.xml"
2010-06-17 12:17:04.877/0.906 Oracle Coherence GE 3.6.0.0 DPR3 <D5>
(thread=ExtendTcpInvocationService:TcpInitiator, member=n/a): Started:
TcpInitiator{Name=ExtendTcpInvocationService:TcpInitiator, State=(SERVICE_
STARTED), ThreadCount=0, Codec=Codec(Format=POF), Serializer=com.tangosol.io.pof.
ConfigurablePofContext, PingInterval=0, PingTimeout=5000, RequestTimeout=5000,
ConnectTimeout=2000, SocketProvider=SystemSocketProvider,
RemoteAddresses=[tpfaeffl-lap7/130.35.99.213:9099], KeepAliveEnabled=true,
TcpDelayEnabled=false, ReceiveBufferSize=0, SendBufferSize=0, LingerTimeout=-1}
2010-06-17 12:17:04.877/0.906 Oracle Coherence GE 3.6.0.0 DPR3 <D5> (thread=main,
member=n/a): Opening Socket connection to 130.35.99.213:9099
2010-06-17 12:17:04.893/0.922 Oracle Coherence GE 3.6.0.0 DPR3 <Info>
(thread=main, member=n/a): Connected to 130.35.99.213:9099
    Success: Correctly allowed to use the invocation service
2010-06-17 12:17:04.955/0.984 Oracle Coherence GE 3.6.0.0 DPR3 <D5>
(thread=ExtendTcpInvocationService:TcpInitiator, member=n/a): Started:
TcpInitiator{Name=ExtendTcpInvocationService:TcpInitiator, State=(SERVICE_
STARTED), ThreadCount=0, Codec=Codec(Format=POF), Serializer=com.tangosol.io.pof.
ConfigurablePofContext, PingInterval=0, PingTimeout=5000, RequestTimeout=5000,
ConnectTimeout=2000, SocketProvider=SystemSocketProvider,
RemoteAddresses=[tpfaeffl-lap7/130.35.99.213:9099], KeepAliveEnabled=true,
TcpDelayEnabled=false, ReceiveBufferSize=0, SendBufferSize=0, LingerTimeout=-1}
2010-06-17 12:17:04.955/0.984 Oracle Coherence GE 3.6.0.0 DPR3 <D5> (thread=main,
member=n/a): Opening Socket connection to 130.35.99.213:9099
2010-06-17 12:17:04.955/0.984 Oracle Coherence GE 3.6.0.0 DPR3 <Info>
(thread=main, member=n/a): Connected to 130.35.99.213:9099
    Success: Correctly unable to use the invocation service
------InvocationService access control example completed------
Process exited with exit code 0.
```

Example 10–24 lists the output in the shell where the cache server is running the proxy service. Notice that the security exception in the output corresponds to the `Success:`

`Correctly unable to use the invocation service` message in the JDeveloper Log window, where the user with role `reader` attempts to run `ExampleInvocable`.

Example 10–24 Proxy Service Window

```
...
Started DefaultCacheServer...

Password validated for user: role_writer
Password validated for user: role_writer
Password validated for user: role_reader
Password validated for user: role_reader
2010-06-17 12:17:04.971/41.250 Oracle Coherence GE 3.6.0.0 DPR3 <D5>
(thread=Proxy:ProxyService:TcpAcceptorWorker:1, member=3): An exception occurred
while processing a InvocationRequest for Service=Proxy:ProxyService:TcpAcceptor:
java.lang.SecurityException: Access denied, insufficient privileges
        at com.oracle.handson.SecurityExampleHelper.
checkAccess(SecurityExampleHelper.java:91)
        at com.oracle.handson.EntitledInvocationService.
query(EntitledInvocationService.java:51)
        at com.tangosol.coherence.component.net.extend.messageFactory.
InvocationServiceFactory$InvocationRequest.onRun(InvocationServiceFactory.CDB:12
)
        at com.tangosol.coherence.component.net.extend.message.Request.
run(Request.CDB:4)
        at com.tangosol.coherence.component.net.extend.proxy.serviceProxy.
InvocationServiceProxy.onMessage(InvocationServiceProxy.CDB:9)
        at com.tangosol.coherence.component.net.extend.Channel$MessageAction.
run(Channel.CDB:13)
        at java.security.AccessController.doPrivileged(Native Method)
        at javax.security.auth.Subject.doAs(Subject.java:337)
        at com.tangosol.coherence.component.net.extend.Channel.execute(Channel.
CDB:29)
        at com.tangosol.coherence.component.net.extend.Channel.receive(Channel.
CDB:26)
        at com.tangosol.coherence.component.util.daemon.queueProcessor.service.
Peer$DaemonPool$WrapperTask.run(Peer.CDB:9)
        at com.tangosol.coherence.component.util.DaemonPool$WrapperTask.
run(DaemonPool.CDB:32)
        at com.tangosol.coherence.component.util.DaemonPool$Daemon.
onNotify(DaemonPool.CDB:63)
        at com.tangosol.coherence.component.util.Daemon.run(Daemon.CDB:42)
        at java.lang.Thread.run(Thread.java:619)
```

11

Caching Sessions with Coherence and WebLogic Server

This chapter describes how to cache session information for Web application instances that are deployed across WebLogic Server instances.

This chapter has the following sections:

Introduction

WebLogic Server includes features that allow deployed applications to easily use Coherence data caches, and seamlessly incorporate Coherence*Web for session management and TopLink Grid as an object-to-relational persistence framework. Collectively, these features are referred to as *ActiveCache*.

ActiveCache is employed by applications running on WebLogic Server and provides replicated and distributed caching services that make an application's data available to all servers in a Coherence data cluster. New features in this release provide direct access by applications to data caches, either through resource injection or component-based JNDI lookup, and let you display, monitor, create, and configure Coherence clusters using the WebLogic Server Administration Console and WLST.

Using ActiveCache with WebLogic Server instances enables you to create a data tier dedicated to caching application data and storing replicated session state. This is separate from the application tier, where the WebLogic Server instances are dedicated to running the application.

Using Coherence*Web with ActiveCache enables you to provide Coherence-based HTTP session state persistence to applications running on WebLogic Server. Coherence*Web enables HTTP session sharing and management across different Web applications, domains, and heterogeneous application servers. Session data can be stored in data caches outside of the application server, thus freeing application server heap space and enabling server restarts without losing session data.

Coherence and Coherence*Web are included in the default installation of WebLogic Server 11gR1 (10.3.3). If you do not already have WebLogic Server 11gR1 (10.3.3) on your system, you can get it at the following URL:

```
http://www.oracle.
com/technology/software/products/middleware/index.html
```

For more information on the integration of Oracle WebLogic Server, Coherence, and Coherence*Web, see *Using ActiveCache* and the *User's Guide for Oracle Coherence*Web*.

Caching Session Information for Web Application Instances

The following example demonstrates how to use ActiveCache to cache session information for Web application instances that are deployed across WebLogic Server instances. To do this, you will create a Web application and deploy it to two server instances. The application is a simple counter that stores the current count as a session attribute. Coherence*Web automatically serializes and replicates the attribute across both server instances. A browser is used to access each application instance to demonstrate that the same session attribute is used among the instances.

1. Ensure that You are Using Coherence 3.6 with WebLogic Server 10.3.3
2. Start a Cache Server
3. Configure and Start the WebLogic Server
4. Create a Machine
5. Create the WebLogic Servers
6. Create a Coherence Cluster
7. Deploy the Shared Library Files
8. Create the Counter Web Application
9. Deploy the Application
10. Start the Node Manager and the WebLogic Servers
11. Verify the Example

Ensure that You are Using Coherence 3.6 with WebLogic Server 10.3.3

By default, the installation of WebLogic Server 11gR1 (10.3.3) also installs Coherence 3.5 in the `coherence_3.5` directory. To complete this example, ensure that your server start-up files and classpaths continue to point to the Coherence 3.6 files you have already installed. Also, ensure that when you are deploying files later in this example, that you are deploying the 3.6 versions.

Start a Cache Server

Start a Coherence cache server. Example 11–1 illustrates a sample script to start the cache server. In this example, `tangosol.coherence.clusterport=7777` is the default multicast listen port of a Coherence cluster and `tangosol.coherence.clusteraddress=231.1.1.1` is the default multicast listen address.

Example 11–1 Script to Start the Cache Server

```
setlocal

    set COHERENCE_HOME=c:\oracle\product\coherence

set COH_OPTS=%COH_OPTS% -server -cp %COHERENCE_HOME%\lib\coherence.jar;%COHERENCE_
HOME%\lib\coherence-web-spi.war;
set COH_OPTS=%COH_OPTS% -Dtangosol.coherence.management.remote=true -Dtangosol.
coherence.cacheconfig=/WEB-INF/classes/session-cache-config.xml -Dtangosol.
coherence.distributed.localstorage=true -Dtangosol.coherence.clusterport=7777
-Dtangosol.coherence.clusteraddress=231.1.1.1 -Dtangosol.coherence.session.
localstorage=true

java %COH_OPTS% -Xms512m -Xmx512m -XX:MaxPermSize=256m com.tangosol.net.
DefaultCacheServer
```

```
:exit
```

Configure and Start the WebLogic Server

This example requires a Coherence Cluster.

1. Run the Oracle WebLogic Configuration Wizard (**Start** then **All Programs** then **Oracle WebLogic** then **WebLogic Server 11gR1** then **Tools** then **Configuration Wizard**) to create a WebLogic Server domain called `test_domain`.

   Before exiting the wizard, select the **Start Admin Server** check box, and click **Done**. The Configuration Wizard automatically starts the Administration Server.

2. Start the WebLogic Server Administration Console.

   From the browser, log in to the Oracle WebLogic Server Administration Console using the following URL:`http://`*`hostname`*`:7001/console`. The Console starts, and the domain home page displays.

Create a Machine

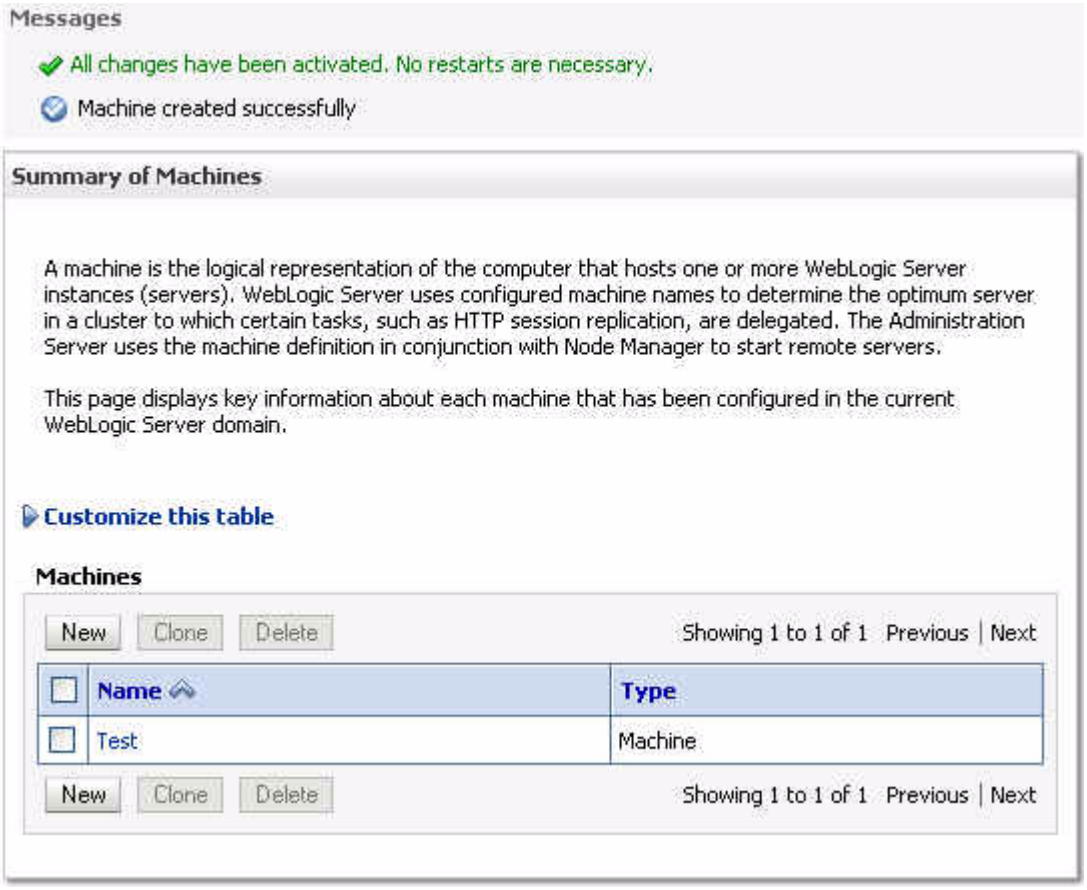
Create a Machine on which to host WebLogic Server instances.

From the **Domain Structure** window, select **Environment** then **Machines**. Click **New**. The **Create a New Machine** page displays. Enter a name for the Machine (in this case, **Test**) and click **OK**.

Figure 11–1 Creating a New Machine

The **Summary of Machines** page should look similar to Figure 11–2.

Figure 11–2 Summary of Machines



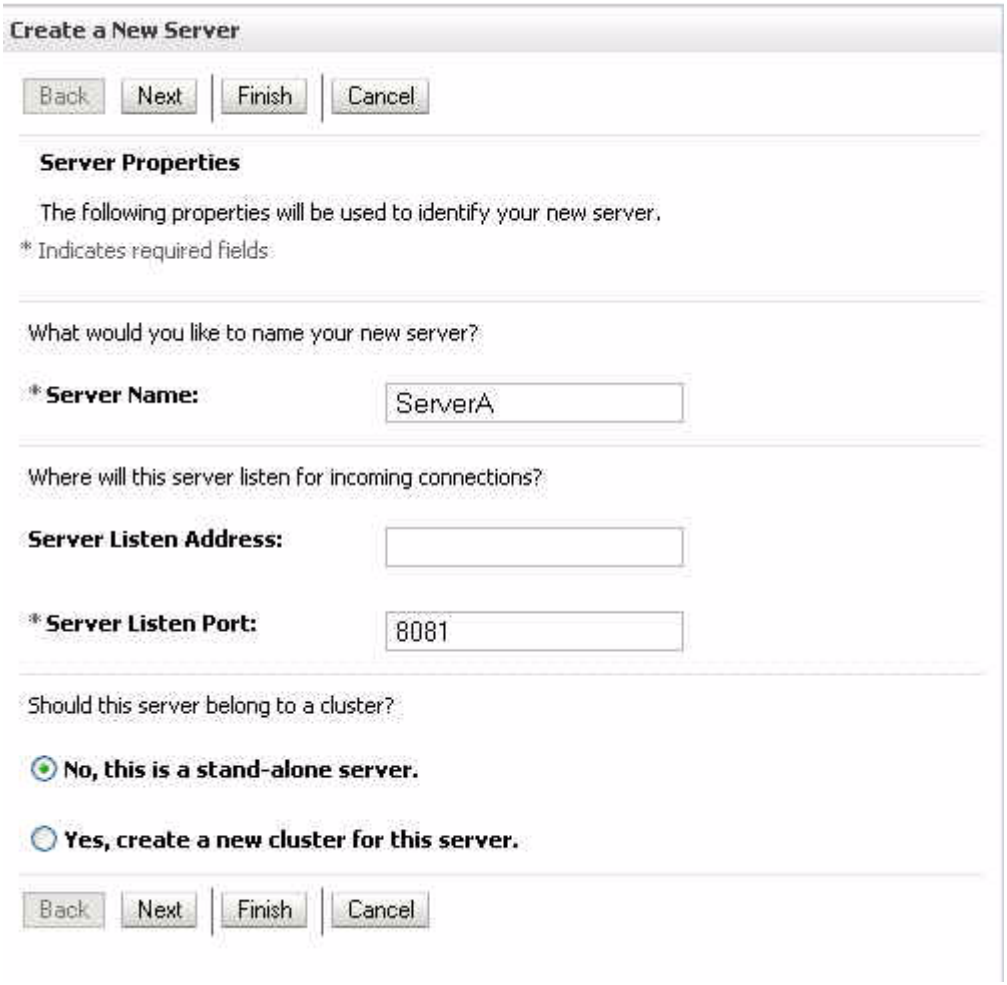
Create the WebLogic Servers

Create two server instances associated with the Machine. The application will be deployed to these servers in a later step.

1. Click the name of the Machine in the **Summary of Machines** page to open the **Settings for** <*machine*> page. Click the **Servers** tab then **Add** to create a server.
2. Select **Create a new server and associate it with this machine** in the **Add a Server to Machine** page, and click **Next**.
3. Provide details about the server in the **Add a Server to Machine** page.

   Enter `ServerA` as the **Server Name** and `8081` as the **Server Listen Port**. Enter the appropriate value for the **Server Listen Address**. Click **Finish**.

Figure 11–3 Adding a Server to a Machine



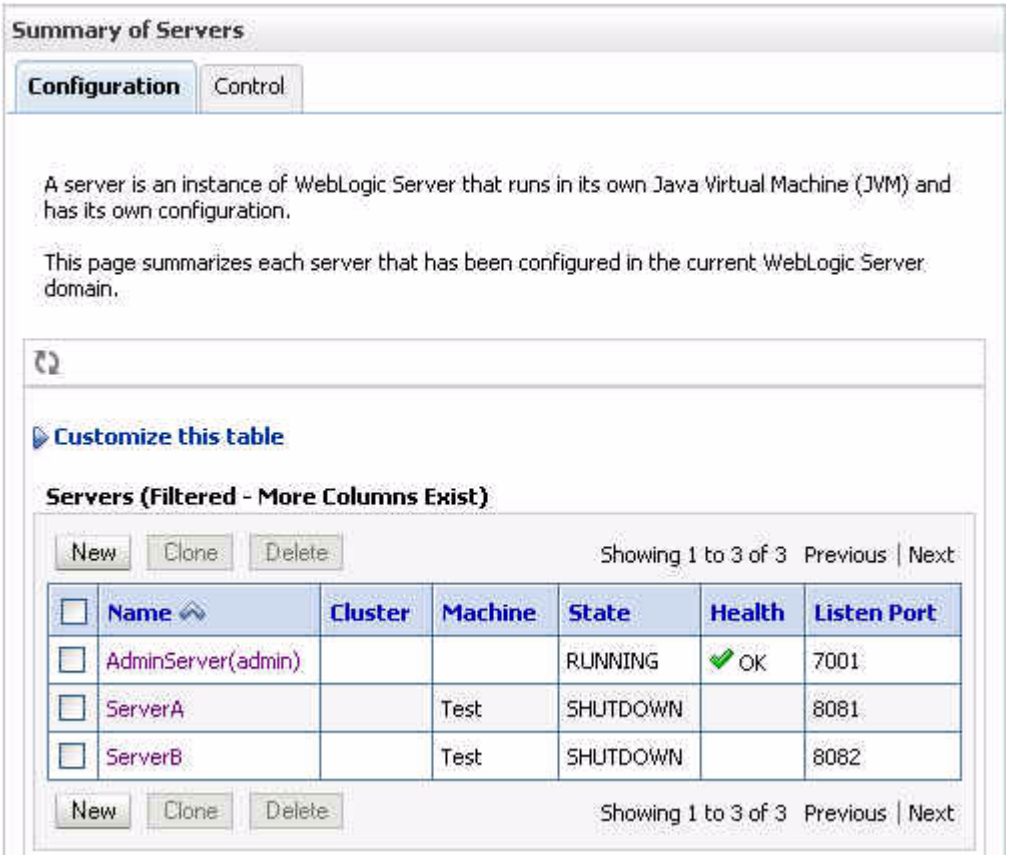
4. When you are returned to the **Settings for** ***machine*** page, repeat the previous three steps to create a second server.

   Enter `ServerB` as the **Server Name** and `8082` as the **Server Listen Port**. Enter the appropriate value for the **Server Listen Address**. Click **Finish**.

5. Expand **Environment** in the **Domain Structure** menu and click **Servers**.

   The **Summary of Servers** page displays and should be similar to Figure 11–4:

Figure 11–4 Summary of Servers Page

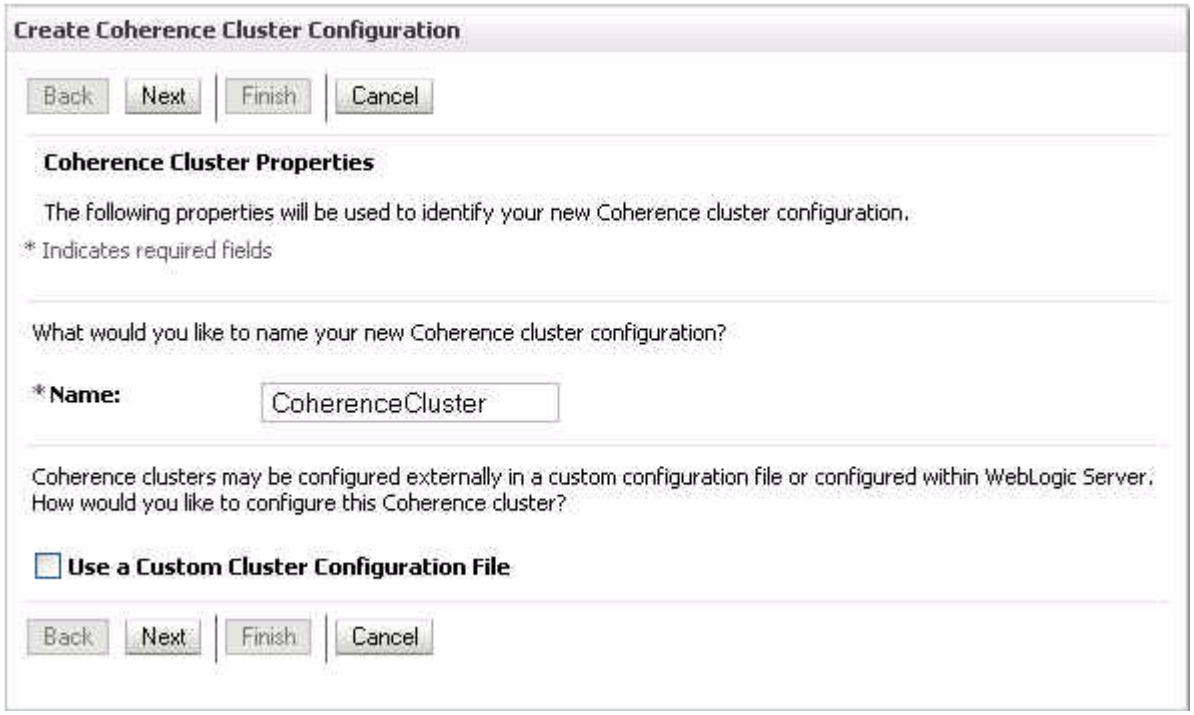


Create a Coherence Cluster

Create a Coherence Cluster.

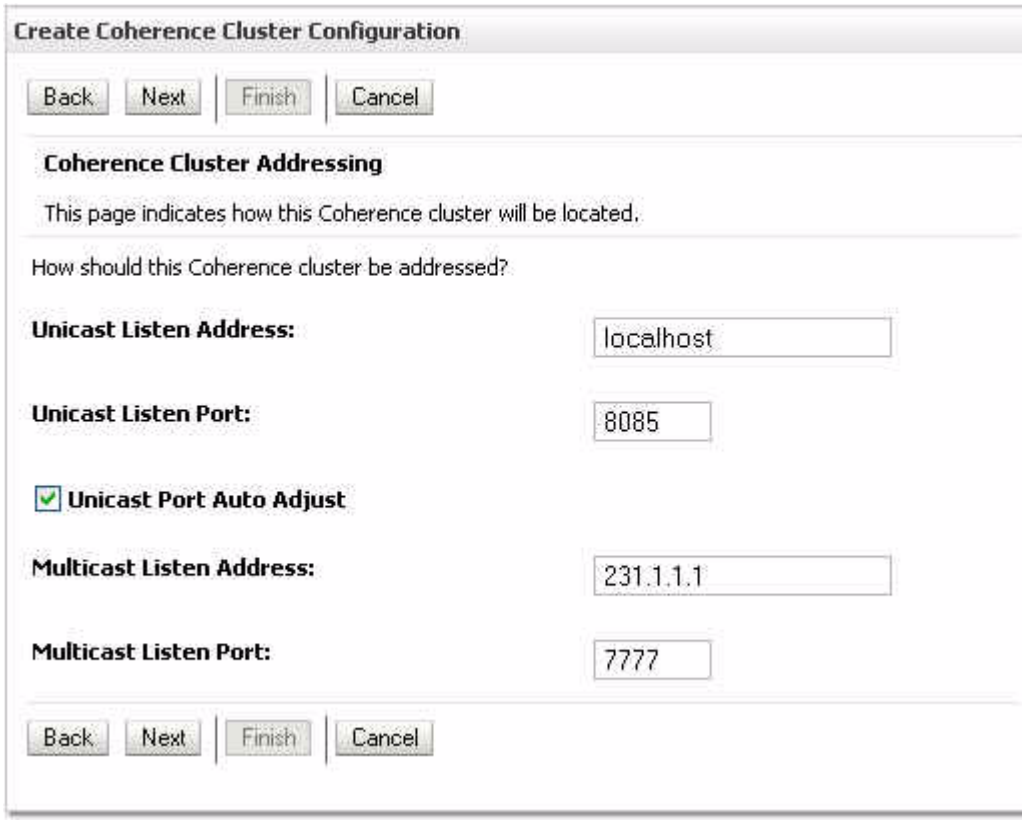
1. Click **Services** in the domain **Structure Window**. Then click **Coherence Clusters**. In the **Summary of Coherence Clusters** page, click **New**. In the **Create Coherence Cluster Configuration** page, enter `CoherenceCluster` in the **Name** field, then click **Next**.

Figure 11–5 Creating a Coherence Cluster



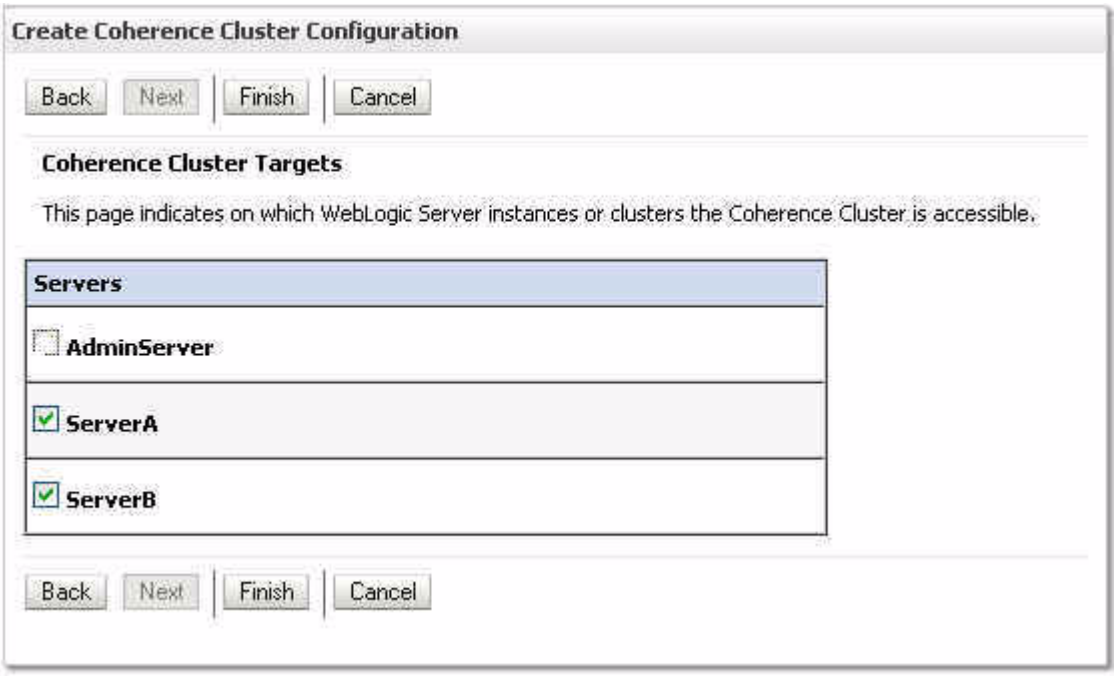
2. Enter a value such as `8085`, in the **Unicast Listen Port** field. Do not change any of the other values and click **Next**.

Figure 11–6 Specifying a Unicast Listen Port for a Coherence Cluster



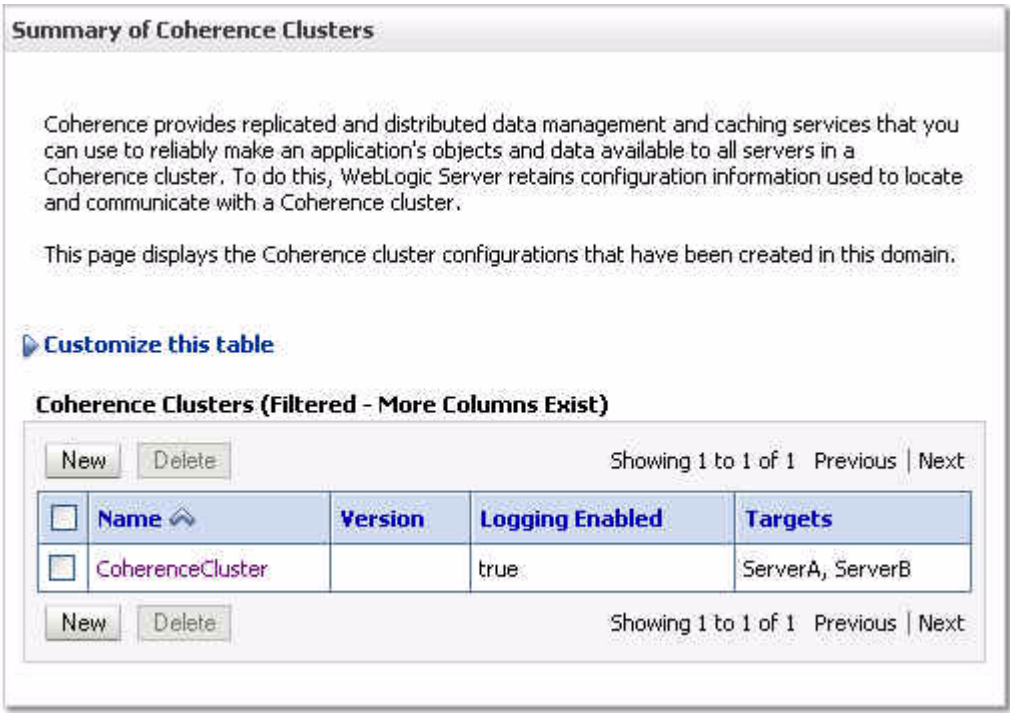
3. In the **Coherence Cluster Targets** page, select `ServerA` and `ServerB` as the targets. Click **Finish**.

Figure 11–7 Choosing Coherence Cluster Targets



The **Summary of Coherence Clusters** page should look similar to Figure 11–8.

Figure 11–8 Summary of Coherence Clusters



Deploy the Shared Library Files

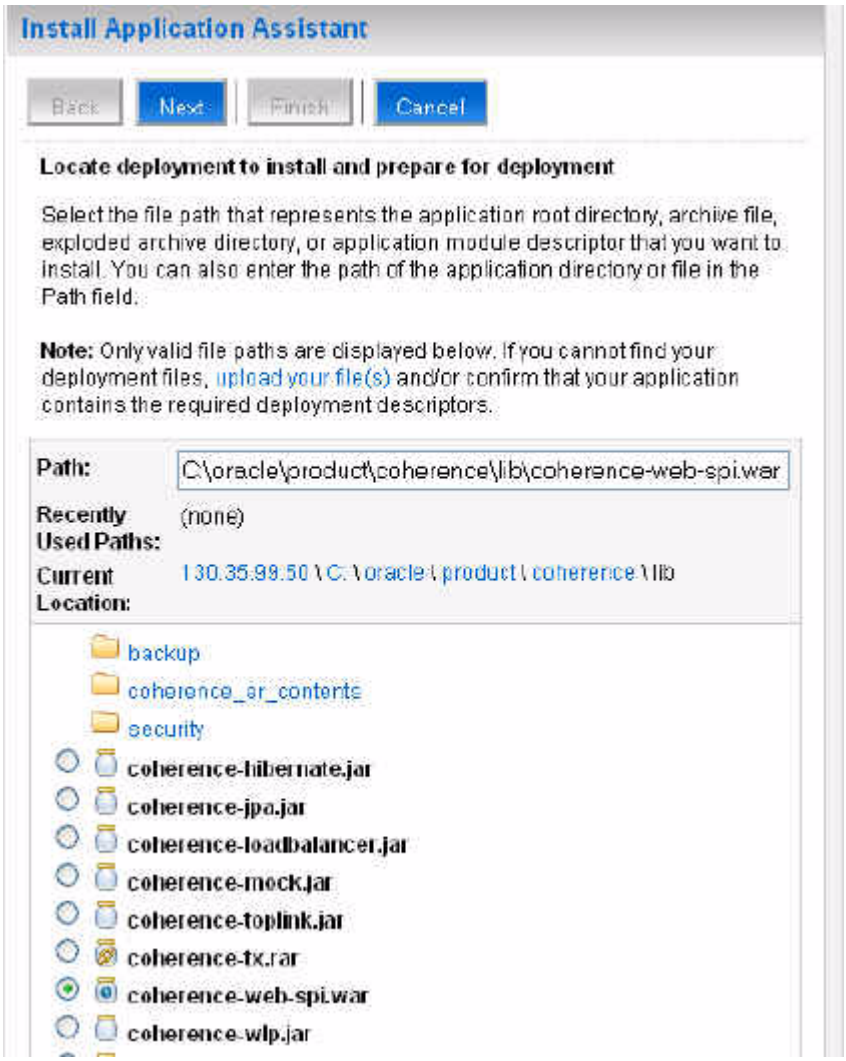
In addition to the `coherence.jar` file, Coherence provides a deployable shared library, `coherence-web-spi.war`, that contains a native plug-in to WebLogic Server's HTTP Session Management interface. Coherence also provides the `active-cache-1.0.jar` file which contains the classes that allow WebLogic Server to interact with Coherence.

You do not have to deploy `coherence.jar` for this example. It will be bundled with the application in a later step.

To deploy the `coherence-web-spi.war` and `active-cache-1.0.jar` files:

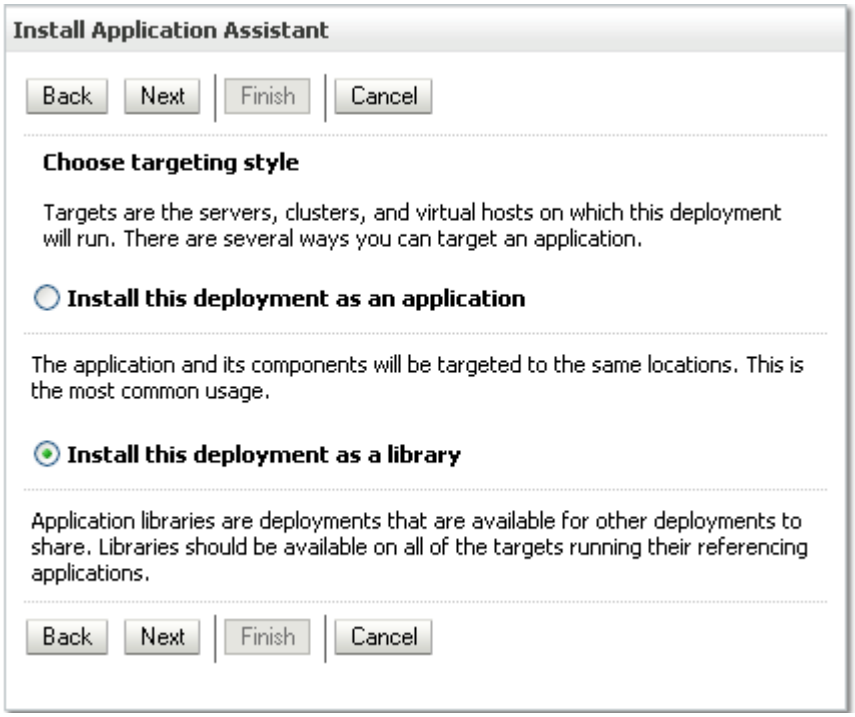
1. From the **Domain Structure** menu, click **Deployments**. The **Summary of Deployments** page displays.
2. Click **Install**. The **Install Application Assistant** screen displays.
3. Use the **Install Application Assistant** to deploy `coherence-web-spi.war` as a library to `ServerA` and `ServerB`.
    a. Locate and select the `coherence-web-spi.war` file. It resides in the `coherence\lib` directory of the Coherence installation. Click **Next**.

Figure 11–9 Selecting the coherence-web-spi.jar File for Deployment



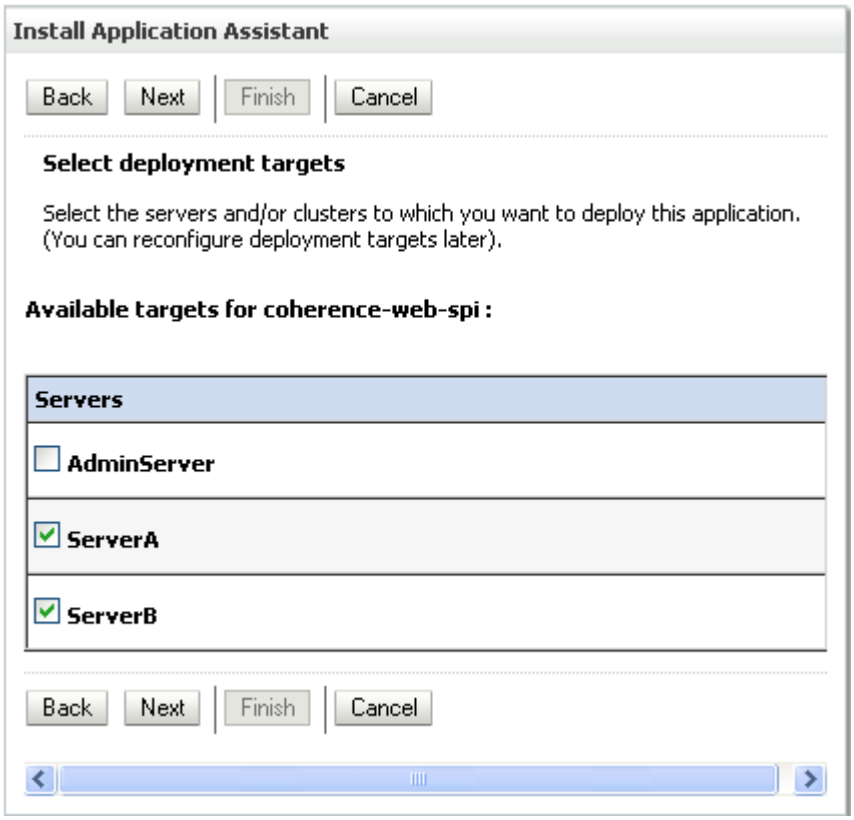
b. In the **Choose targeting style** page, ensure that **Install this deployment as a library** is selected. Click **Next**.

Figure 11–10 Installing the Deployment as a Library



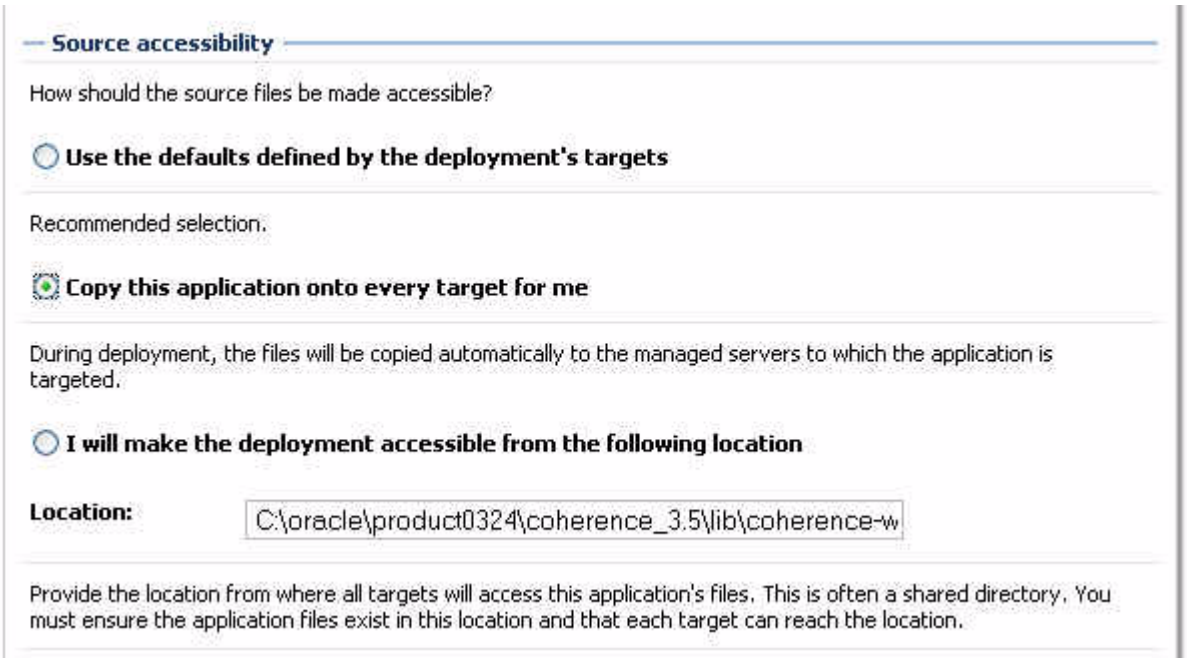
c. Select `ServerA` and `ServerB` as the deployment targets (do not deploy `coherence-web-spi.war` to the `AdminServer`). Click **Next**.

Figure 11–11 Selecting Deployment Targets



d. In the Optional Settings page, select the **Copy this application onto every target for me** option in the **Source accessibility** section.

Figure 11–12 Copying Files to Targets



e. You can click **Finish** to skip the rest of the steps in the **Install Application Assistant**. The **Summary of Deployments** page displays after the application is deployed.

4. Repeat Steps 1-3 to deploy `active-cache-1.0.jar` to `ServerA` and `ServerB` (do not deploy `active-cache-1.0.jar` to the `AdminServer`).

   The `active-cache-1.0.jar` is included in the WebLogic Server installation. Assuming that you installed the WebLogic Server at `C:\oracle\product`, you will find the file in the `C:\oracle\product\wls1033\wlserver_10.3\common\deployable-libraries` directory.

Create the Counter Web Application

The Counter Web application is a simple counter implemented as a JSP. The counter is stored as an HTTP session attribute and increments each time the page is accessed.

To create the Counter Web application:

1. Create a standard Web application directory as follows:

```
/
/WEB-INF
```

2. Copy the following code to a text file and save it as `web.xml` in the `/WEB-INF` directory.

```
<?xml version = '1.0' encoding = 'windows-1252'?>
<web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:schemaLocation="http://java.sun.com/xml/ns/j2ee http://java.sun.
com/xml/ns/j2ee/web-app_2_4.xsd"
 xmlns="http://java.sun.com/xml/ns/j2ee" version="2.5">
   <description>Empty web.xml file for Web Application</description>
```

```
</web-app>
```

3. Create a `weblogic.xml` file in the `/WEB-INF` directory.
   - Add a library references for the `coherence-web-spi.war` file.
   - Reference the Coherence Cluster in a `coherence-cluster-ref` stanza.

   Example 11–2 illustrates a sample `weblogic.xml` file.

Example 11–2 Sample weblogic.xml File

```
<weblogic-web-app xmlns="http://xmlns.oracle.com/weblogic/weblogic-web-app"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://xmlns.oracle.com/weblogic/weblogic-web-app http://www.
oracle.com/technology/weblogic/weblogic-web-app/1.1/weblogic-web-app.xsd">
      <library-ref>
           <library-name>coherence-web-spi</library-name>
      </library-ref>
     <coherence-cluster-ref>
        <coherence-cluster-name>CoherenceCluster</coherence-cluster-name>
     </coherence-cluster-ref>
</weblogic-web-app>
```

4. Bundle the `coherence.jar` file with the application: copy `coherence.jar` from the `coherence\lib` directory of the Coherence 3.6 installation to the `WEB-INF/lib` directory.
5. Copy the following code for the counter JSP to a text file and save the file as `counter.jsp` in the root of the Web application directory.

```
<html>
    <body>

<h3>
      Counter :
      <%
         Integer counter = new Integer(1);
         HttpSession httpsession = request.getSession(true);
         if (httpsession.isNew()) {
                httpsession.setAttribute("count", counter);
                out.println(counter);
         } else {
                int count = ((Integer) httpsession.getAttribute("count")).
intValue();
                httpsession.setAttribute("count", new Integer(++count));
                out.println(count);
         }
      %>
      </h3>

    </body>
</html>
```

6. Create a `manifest.mf` file in the `META-INF` directory. Add references to the `active-cache` JAR file. Example 11–3 illustrates a sample `manifest.mf` file.

Example 11–3 Sample manifest.mf File

```
Extension-List: active-cache
active-cache-Extension-Name: active-cache
active-cache-Specification-Version: 1.0
```

```
active-cache-Implementation-Version: 1.0
```

7. The Web application directory should appear as follows:

```
/
/counter.jsp
/META-INF/manifest.mf
/WEB-INF/web.xml
/WEB-INF/weblogic.xml
/WEB-INF/lib/coherence.jar
```

8. ZIP or JAR the Web application directory and save the file as `counter.war`.

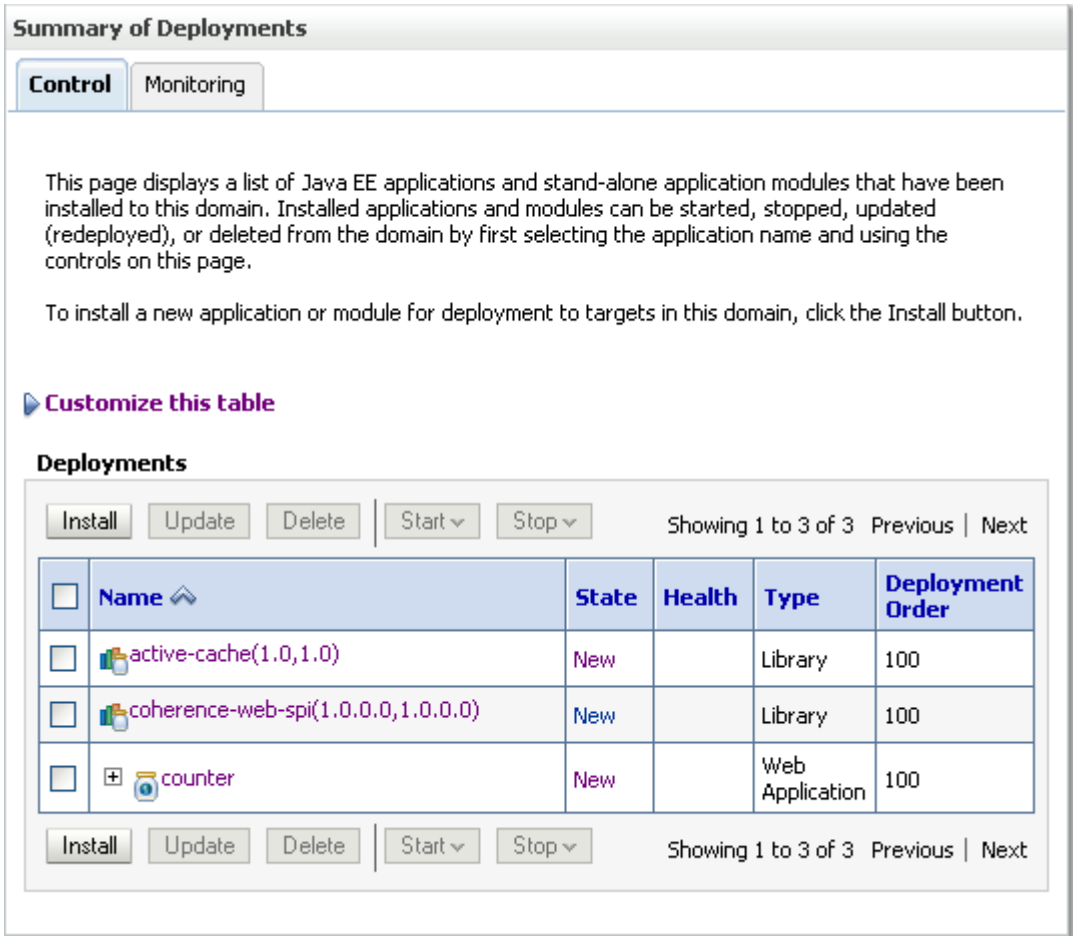
Deploy the Application

To deploy the `counter.war` application:

1. Open the **Summary of Deployments** page by clicking **Deployments** in the **Domain Structure** menu in the Oracle WebLogic Server Administration Console.
2. Click **Install**. The **Install Application Assistant** screen displays.
3. Use the **Install Application Assistant** to deploy `counter.war` to `ServerA` and `ServerB`. In the **Optional Settings** page, select the **Copy this application onto every target for me** option in the **Source accessibility** section.

   The **Summary of Deployments** page displays after the application is deployed. Figure 11–13 illustrates the page with the deployed `active-cache.jar`, `coherence-web-spi.war`, and `counter.war` files.

Figure 11–13 Summary of Deployments Page with Deployed Files



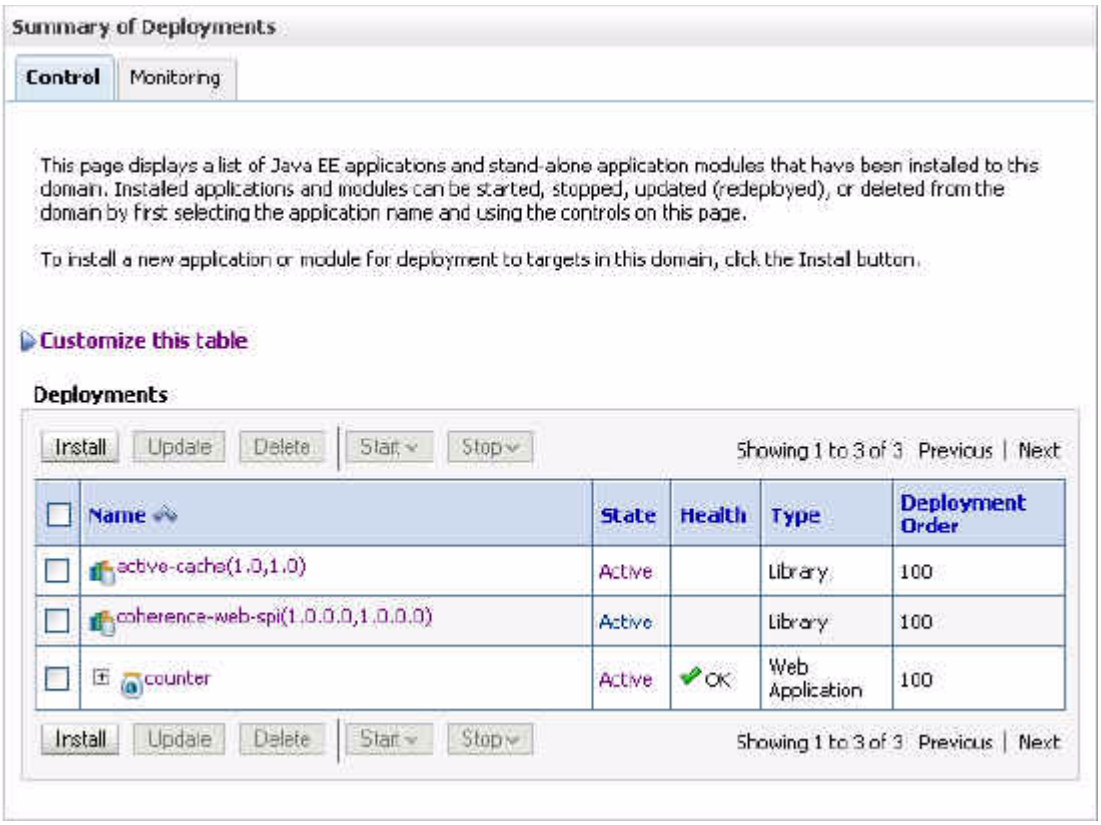
Start the Node Manager and the WebLogic Servers

Start the Node Manager then start the WebLogic Server instances from the WebLogic Server Administration Console. The Node Manager is a Java utility that runs as a separate process from Oracle WebLogic Server, and enables you to perform common operations for a Managed Server, regardless of its location with respect to its Administration Server.

1. Start the Node Manager from **Start** then **All Programs** then **Oracle WebLogic** then **WebLogic Server 11gR1** then **Tools** then **Node Manager**.
2. Click **Environment** then **Servers** in the domain **Structure Window**. From the **Summary of Servers** page in the WebLogic Server Administration Console, click the **Control** tab and start both server instances.

   Figure 11–14 illustrates the deployments table after the servers have been started.

Figure 11–14 Deployments Window Showing the Deployed Application and Libraries



Verify the Example

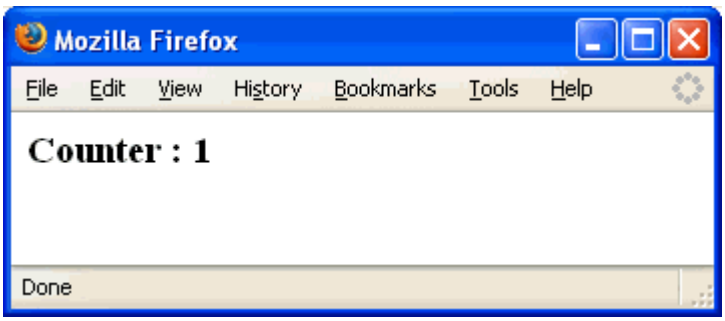
To verify the example:

1. Open a browser and access the `ServerA` counter instance using the following URL:

   `http://host:8081/counter/counter.jsp`

   The counter page displays and the counter is set to 1 as follows:

Figure 11–15 Counter Page with Counter Set to 1

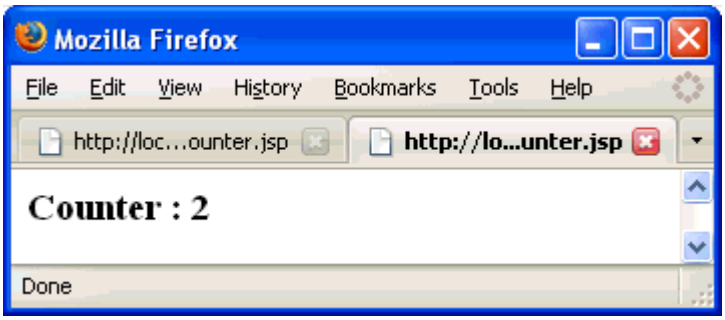


2. In a new browser (or new browser tab), access the `ServerB` counter instance using the following URL:

   `http://host:8082/counter/counter.jsp`

The counter page displays and the counter increments to 2 based on the session data.

Figure 11–16 Counter Page with Counter Set to 2



3. If you refresh the page, the counter increments to 3. Return to the original browser (or browser tab), refresh the instance and the counter displays 4.

A

Coherence Client Application Commands

This appendix describes the commands that can be issued to the Coherence command line application `coherence.cmd` (or `coherence.sh` on UNIX).

batch

You can script Coherence from the command line by creating a separate file with command-line commands. For example, you can create a file called `batchfile.txt` which contains commands to change the current cache to `testcache`, clear it, and then exit:

```
cache testcache
clear
bye
```

You can invoke this script on the UNIX platform by running the Coherence command line application:

```
bin/coherence.sh "@batch batchfile.txt"
```

On the Windows platform, substitute `bin/coherence.sh` with `bin/coherence.cmd`.

bulkput <# of iterations> <block size> <start key>

Populates the current `NamedCache` with the specified number of objects. This is best described with an example:

```
bulkput 1000 10000 1
```

This will put 1000 objects of size 10000 bytes (10 MB total) starting with key 1 into the cache. The type of the key will be `java.lang.Integer` and the type of the value is `byte[]`. (The one exception is for the size zero, the value is of type `Coherence$DebugCacheItem`, which is used to test custom serialization and `ClassLoader` issues.)

bye

Gracefully shuts down the cluster member and closes the Coherence command-line application. To test "ungraceful" shutdown, use `Ctrl+C` (Windows or UNIX) or `kill -9 <pid>` on UNIX.

clear

Clears the current `NamedCache` by calling `clear`.

destroy

Calls `destroy` on the current `NamedCache`. This will remove the `NamedCache` completely from the cluster, causing other cluster members to receive an `IllegalStateException` if they are holding on to a reference to the same cache and try to do any operation on that cache.

get <key>

Returns the value associated with this `<key>` from the cache by calling `get(key)`.

hash

Returns a hash value of the `NamedCache`. This is similar to a checksum and is mainly used for verifying the equivalency of the replicated caches.

help

Prints the list of commands available to you.

inc <key> [<count>]

Increments the specified key the specified number of times, using locking to ensure that no "missing updates" occur. For example:

```
inc counter 10000
```

This will increment the `counter` key 10000 times. If you run this command simultaneously on 10 machines, the value of the `counter` key will be 100,000 (assuming it started at 0 or did not exist previously).

kill

Shuts down the current `com.tangosol.net.CacheService` in an orderly (graceful) manner. This does not affect other service members.

list [<map name>]

Lists the contents of the specified `NamedCache`, or the current one if none is specified.

Note:

- If there are more than 50 items in the `NamedCache` only the first 50 items will be displayed through the `list` command.
- Java implements the `toString` method for `byte[]` values as `[` (that is, *array*) + `B` (that is, *byte*) + `hashcode`. For example:

  ```
  [B@701a1e
  ```

---

listen [('start' | 'stop') [('cluster' | 'local')]]

Registers (`start`) or unregisters (`stop`) a debug implementation of `com.tangosol.util.MapListener` with the current `NamedCache` (`cluster`) or with the "backing-store" `com.tangosol.util.ObservableMap` (`local`). (The backing store is the `java.util.Map` object obtained by the cache service from the `LocalCacheFactory` interface.)

For partitioned caches, the `local` listener can only be used on storage-enabled nodes. For example, with the partitioned cache service, this command will show all cache changes that this member manages:

```
listen start local
```

This command will show all cache changes, even if they are managed by a different member, and even if this member has `local-storage-enabled` set to `false`.

```
listen start cluster
```

lock <key> [<timeout>l]

Locks the specified key. If the lock cannot be obtained immediately, it returns `false`, unless a time-out has been specified. The value is the number of milliseconds, expressed as a `long` integer (that is, the value should be suffixed with an "`l`"). For example, this command attempts to lock the `counter` key for up to 10 seconds:

```
lock counter 10000l
```

maps

Lists all the `NamedCache`(s) for the current service, `com.tangosol.net.CacheService`, regardless of whether they were created by this member.

memory

Lists the "total" used, total "free", and maximum memory values for this JVM. For example:

```
Map (testcache): memory
total=64356K (64356352)
free =55658K (55658976)
max =64356K (64356352)
```

put <key> <value>

Puts the specified key/value pair into the current `NamedCache` using the `put(key, value)` method.

release

Releases the reference to the current `NamedCache` on this cluster member by calling the `release` method. This does not affect other cluster nodes. After calling release, the name of the inactive cache is still shown, because the command line tool still holds the reference to the cache. This is expected behavior. If subsequent commands are issued on this cluster node against the current `NamedCache`, an `IllegalStateException` will occur.

```
Map (testcache): release
```

```
Map (testcache):
```

remove <key>

Removes the specified key from the current `NamedCache` using the `remove(key)` method.

scan <start key> <stop key>

This method is used to check for missing keys in a sequence. For example, the following command checks the `java.lang.Integer` keys from 1 to 1000 inclusive, and prints out any keys in that range that are not present in the current `NamedCache`:

```
scan 1 1000
```

services

Lists all the services (`com.tangosol.net.Service`) known to this cluster. Some of these services may not be currently running on this cluster node.

size

Shows the count of objects in the current `NamedCache`.

unlock <key>

Unlocks the specified key in the current `NamedCache` using the `unlock(key)` method.

waitkey <start key> <stop key>

This method is used in conjunction with multiple Coherence command line processes and the `bulkput` command to time cache replication. Before one machine starts its `bulkput` command, a second machine is told to wait for the range of keys that will be placed into the cache.

For example, if you run these commands on the first JVM:

```
clear
waitkey 1 1000
```

Then run this command on the second JVM:

```
bulkput 1000 100 1
```

The first JVM will display the total replication time for 1000 entries with values of 100 bytes.

who | cluster

Lists information about each `Member` (`com.tangosol.net.Member`) in the cluster.

whoami | service

Lists the current `Service` (`com.tangosol.net.Service`) information.

worker <command>

Performs the specified command on a new temporary thread.

#<repeat count> <command>

Repeats the following `<command>` the specified number of times. Multiple commands can be delimited with a semicolon. For example, in a loop of ten iterations, this command increments the `Counter` key, then lists the current `NamedCache` values:

```
#10 inc Counter; list
```

@<command>

Silent mode. Suppresses output from the command. This can be used in conjunction with '`#`' (repeat mode). For example, in a loop of ten iterations, this command increments the `Counter` key, then lists the current `NamedCache` values. The output from the `inc` command is suppressed:

```
#10 @inc Counter; list
```

&<functionName> [paramValue]*

Uses reflection to call the specified function on the current `NamedCache` instance. This is intended primarily for internal testing use.

Index

A

B

C

S

T

U

V

W

X