ORACLE

JavaOne℠

Developing Java™ Persistence API Applications with the NetBeans™ IDE and EclipseLink

Doug Clarke
Oracle Corporation
EclipseLink Project co-lead
Andrei Badea
Sun Microsystems, Inc.

Agenda

- Java™ Persistence API Introduction
- EclipseLink Overview
- NetBeans™ IDE support for Java Persistence API (JPA)
- Demos
  - NetBeans IDE

JPA—in a Nutshell

- A Java technology standard that defines:
  - how Java programming language objects are stored in relational databases (specified using a standard set of mappings)
  - a programmer API for reading, writing, and querying persistent Java programming language objects (“Entities”)
  - a full featured query language
  - a container contract that supports plugging any JPA runtime in to any compliant container
- Suitable for use in different modes
  - Standalone in Java SE environment
  - Hosted within a Java EE Container
- Standardization of current persistence practices

JPA—POJO Entities

- Concrete classes
- No required interfaces
  - No required business interfaces
  - No required callback interfaces
- new() for instance creation
- Direct access or getter/setter methods
  - Can contain logic (e.g. for validation, etc.)
- “Managed” by an EntityManager
- Can leave the Container (become “detached”)

Object-Relational Mappings

- Core JPA Mappings
  - Id
  - Basic
  - Relationships
    - OneToOne
    - OneToMany/ManyToOne
    - ManyToMany
  - And more…
- Can be specified using Annotations or XML

JPA Entity Annotations

```
@Entity public class Customer {

    @Id
    private String name;
    @OneToOne
    private Account account;

    public String getName() { return name; }
    public void setName(String name) {
        this.name = name;
    }
    public Account getAccount() { return account; }
    public void setAccount(Account account) {
        this.account = account;
    }
}
```

JPA Entity—Mappings in XML

```
<entity-mappings
  xmlns="http://java.sun.com/xml/ns/persistence/orm"
…
   <entity class="Customer">
     <attributes>
       <id name="name"/>
       <one-to-one name="account"/>
     </attributes>
   </entity>
…
</entity-mappings>
```

JPA Implementations

- Persistence provider vendors include:
  - Oracle, Sun / TopLink Essentials
  - EclipseLink JPA
  - Oracle TopLink
  - BEA Kodo / Apache OpenJPA
  - RedHat™ / JBoss™ Hibernate™
  - SAP JPA
- JPA containers:
  - GlassFish™ Project/SunAS, OracleAS, SAP, BEA, JBoss, Spring Framework
- Development Environments:
  - NetBeans IDE, Eclipse, MyEclipse™, IntelliJ™,Oracle JDeveloper™

Agenda

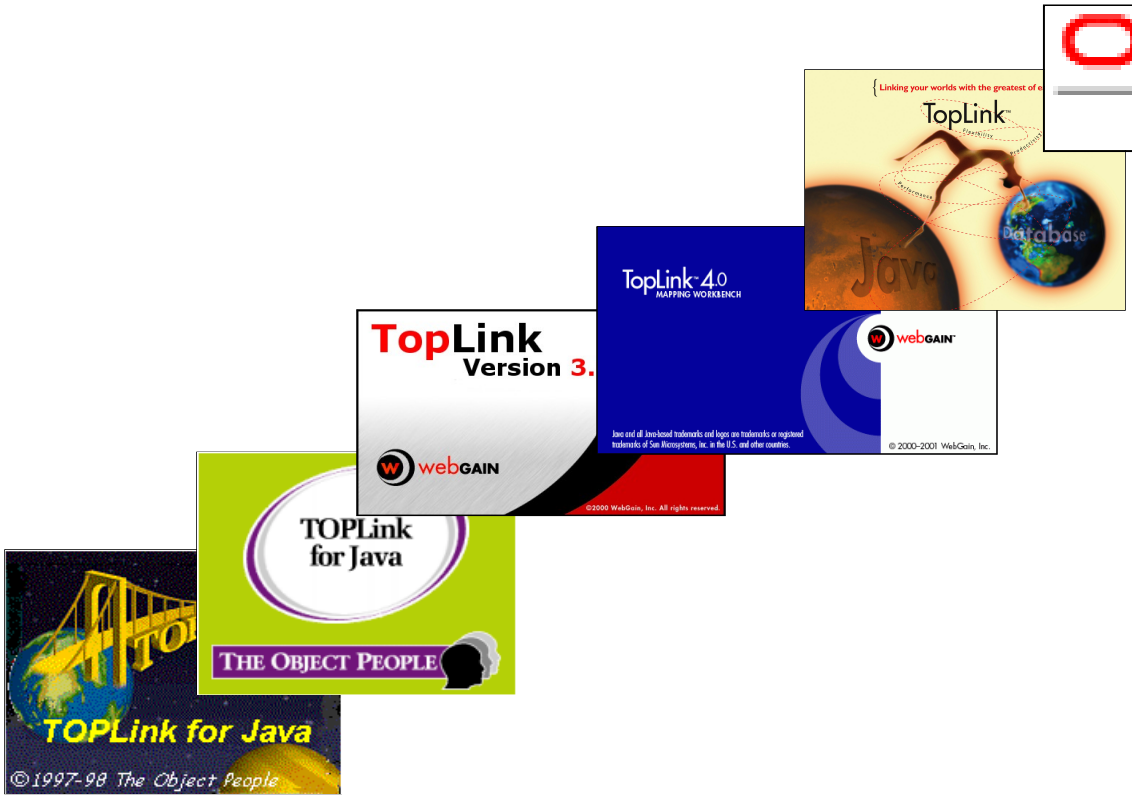
- Java™ Persistence API Introduction
- EclipseLink Overview
- NetBeans IDE support for JPA
- Demos
  - NetBeans IDE

What is the EclipseLink Project?

- Eclipse runtime project
  - Nicknamed “EclipseLink”
  - Eclipse RT Project
- Comprehensive
  - EclipseLink JPA: Object-Relational
  - EclipseLink MOXy: Object-XML
  - EclipseLink SDO: Service Data Objects
  - EclipseLink DBWS: Database Web Services
  - EclipseLink EIS: Non-Relational using JCA
- Defining blueprints for OSGi persistence services
- Commercially proven solution
  - Initial contribution of Oracle TopLink

History of EclipseLink

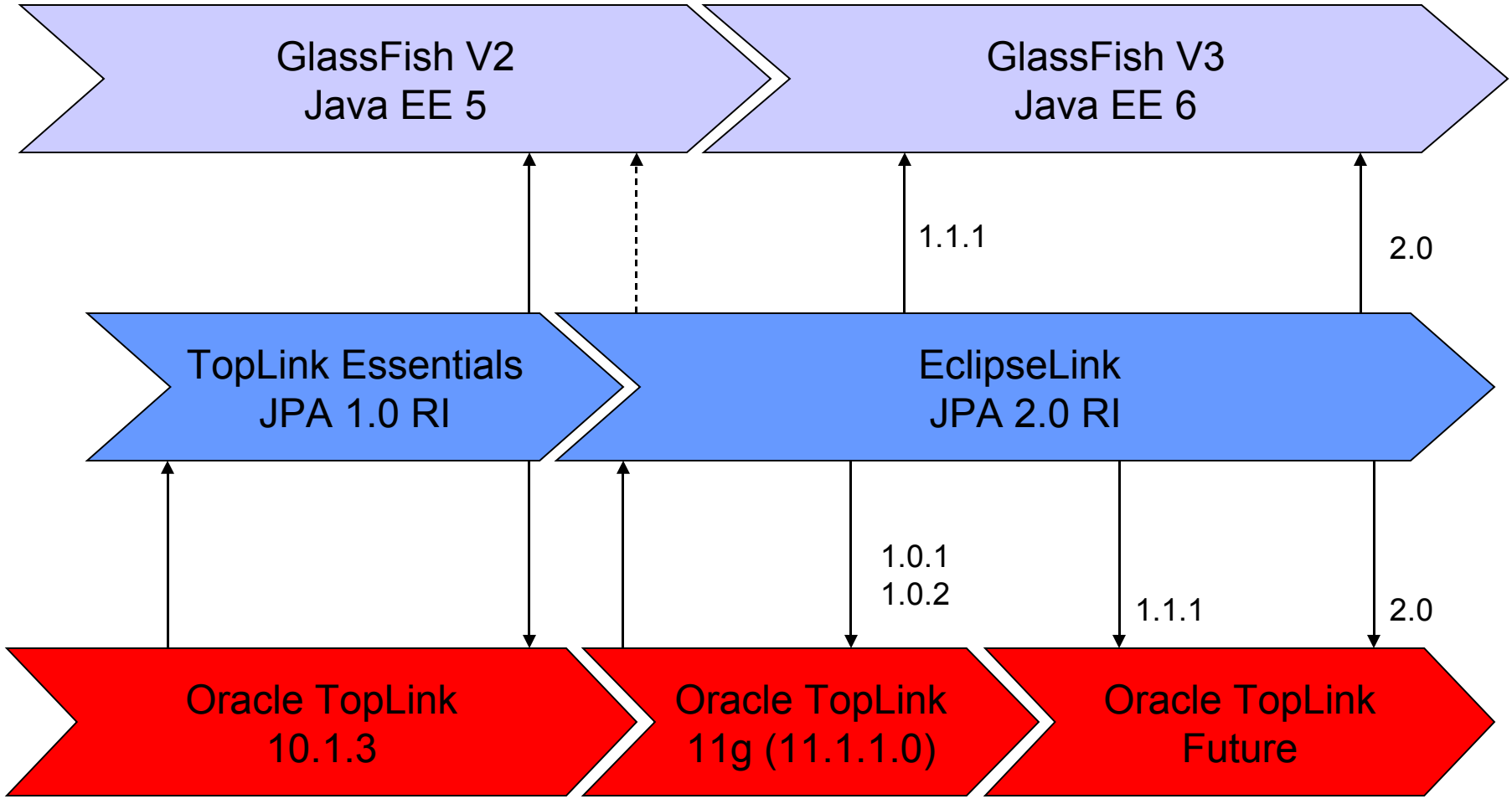
eclipse link



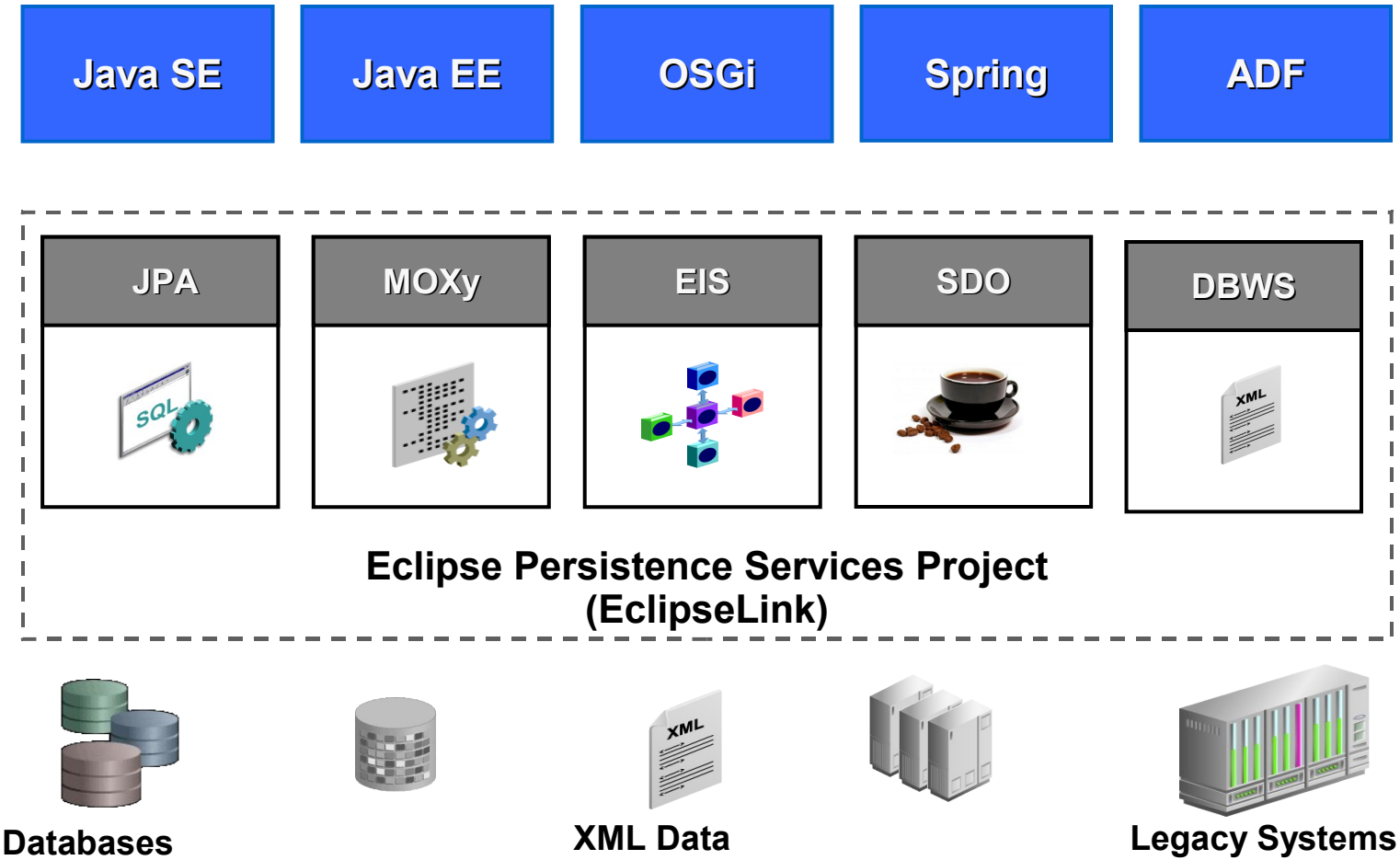
TopLink Essentials

1996 → 2009

JPA in GlassFish™



EclipseLink Project



Significance of EclipseLink

- First comprehensive open source persistence solution
  - Object-Relational and much more
- Based upon product with 12 years of commercial usage
- Shared infrastructure
  - Easily share the same domain model with multiple persistence technologies
  - Leverage metadata for multiple services
- Important part of the GlassFish and Eclipse Ecosystems

EclipseLink JPA

- JPA 1.0 compliant implementation
- Java EE platform, Java SE platform, Web, Spring, and OSGi
- Any JDBC™ Technology/SQL compliant database
  - Advanced database extensions
    - Stored procedures, Native SQL, Data types, …
- Key infrastructure:
  - Caching, Locking, Query Framework, Mapping, …
  - Schema generation
- Highly Extensible

… plus many valuable advanced features

JDBC Technology Connection Settings

- Resource-level JDBC technology settings are vendors responsibility
- Need to specify the four basic JDBC technology properties to obtain driver connections
  - Driver class, URL, username, password
- The property keys will be different, but the values for a given JDBC technology data source will be the same for all vendors
- Used when not in a container, or when managed data sources are not available or not desired

JDBC Technology Connection Settings

```
<properties>

...

<property name="javax.persistence.jdbc.driver"
  value="oracle.jdbc.Driver"/>

<property name="javax.persistence.jdbc.url"
  value="jdbc:oracle:thin:@localhost:1521:XE"/>

<property name="javax.persistence.jdbc.user"
  value="scott"/>

<property name="javax.persistence.jdbc.password"
  value="tiger"/>
```

Logging

- Users want to control over logging, but vendors use different logging APIs
- Can usually configure to use one of the well-known logging APIs
  - java.util.logging, log4J, etc.
- Common requirement is to configure the logging level to show the generated SQL

```
<property name="eclipselink.logging.level"
          value="CONFIG"/>
<property name="eclipselink.logging.level.sql"
          value="FINE"/>
```

DDL Generation

- Standard enables it but does not currently dictate that providers support it
- Mapping metadata specifies how DDL should be generated
- Vendors may offer differing levels of support, including:
  - Generating DDL to a file only
  - Generating and executing DDL in DB
  - Dropping existing tables before creating new ones

```
<property
      name="eclipselink.ddl-generation"
      value="create-tables"/>
```

Database Platform

XML:

```
<property
      name="eclipselink.target-database"
      value="Derby"/>
```

API:

```
  properties.put(
   EclipseLinkProperties.TARGET_DATABASE,
   TargetDatabase.ORACLE);
```

Values:

- Auto (Default)
- Oracle, DB2, Derby, MySQL, SQLServer, Sybase, TimesTen, JavaDB, PostgreSQL, …
- Custom DatabasePlatform classes supported

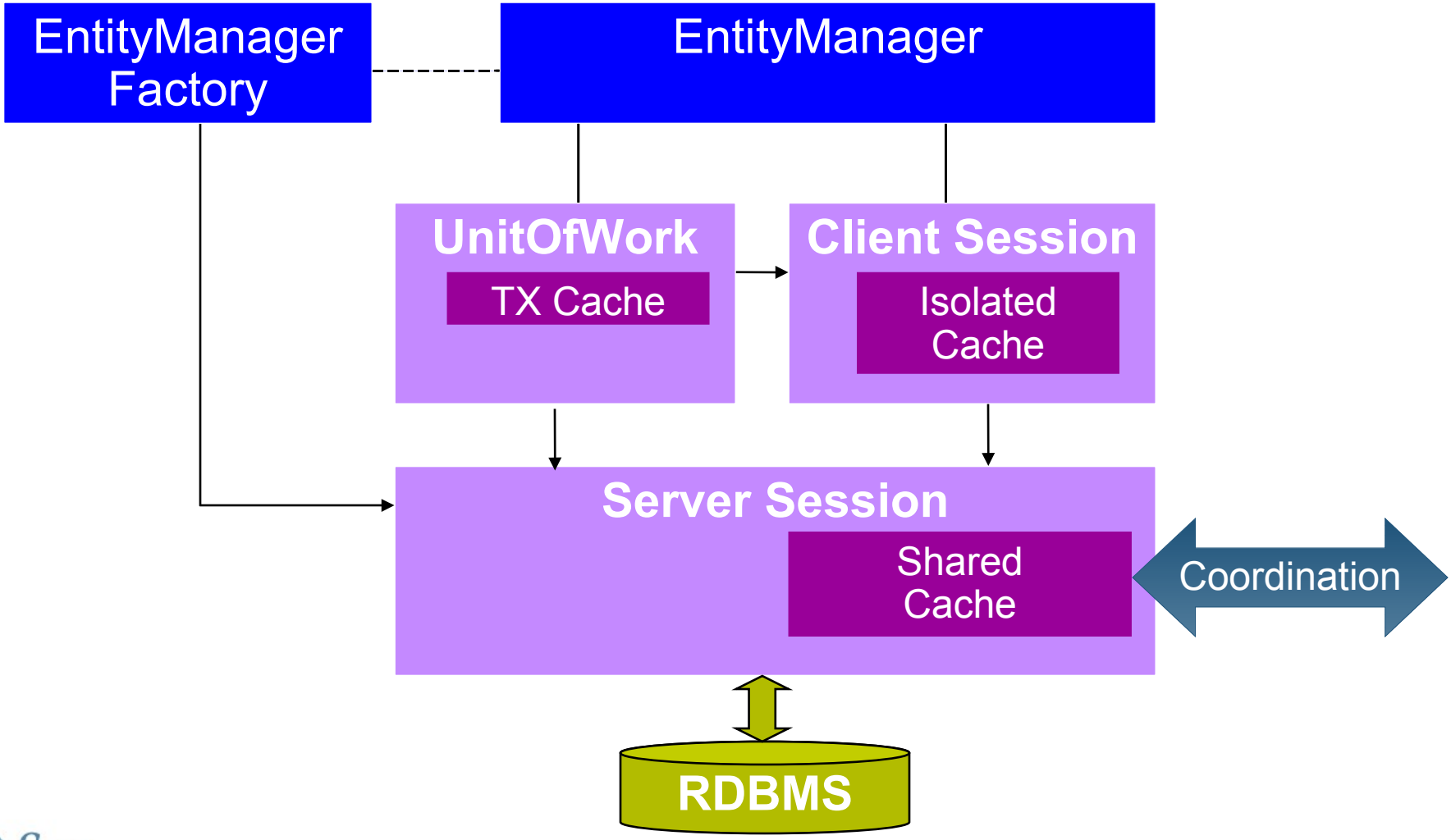
Server Platform

- Enables simplified configuration of the target application server
- Used to enable integration with:
  - Java Transaction API (JTA)
  - Logging
  - JDBC technology connection un-wrapping

```
<property name="eclipselink.target-server"
      value="SunAS9"/>
```

- Supported Platforms `(ServerPlatform)`
  - None (default)
  - SunAS, OC4J*, WebLogic*, WebSphere*, JBoss

EclipseLink Caching



Cache Configuration

- Cache Shared/Isolated

```
<property name="eclipselink.cache.shared.default"
  value="true"/>
```

- Cache Type & Size
  - SoftWeak, HardWeak
  - Weak
  - Full
  - None

```
<property name="eclipselink.cache.type.default"
  value="Full"/>
<property name="eclipselink.cache.type.MyEntity"
  value="Weak"/>
```

Minimize stale cache

- Configure the cache relative to application data's usage
  - Is the data shared between users/processes?
  - Is the data volatile?
    - Only through JPA application?
    - Through direct DB modification?
- Ensure you protect any data that can be concurrently modified with a locking strategy
  - Must handle optimistic lock failures on flush/commit
- Use query refreshing to minimize optimistic lock failures

Customization Using Properties

```
<properties>
  ...
  <property
    name="toplink.session.customizer"
    value="acme.MySessionCustomizer"/>
  <property
    name="toplink.descriptor.customizer.Employee"
    value="acme.MyDescriptorCustomizer"/>
  ...
</properties>
```

Descriptor & Session Customizers

```
public class MySessionCustomizer
                    implements SessionCustomizer {

  public void customize(Session session) {
    session.setProfiler(new PerformanceProfiler());
  }
}

public class MyDescriptorCustomizer
                    implements DescriptorCustomizer {

  public void customize(ClassDescriptor desc) {
    desc.disableCacheHits();
  }
}
```

Weaving Support

- EclipseLink makes use of Weaving (ASM) to introduce additional functionality into the JPA entity classes
  - Needed for M:1 and 1:1 lazy fetching
  - Integrated with OC4J 10.1.3.1 and Spring 2.0
  - Available for Java SE platform using JDK™ software/JRE's –javaagent:
  - Optional
  - Static weaving also supported
    - Weaving of .class files before deployment

Query Optimizations

- Graph Retrieval minimizing N+1 SQL Generation
  - JOIN FETCH using JP QL
  - EclipseLink Joining
    - Supports multi-level joining
    - Query Hint: “eclipselink.join-fetch”
  - EclipseLink Batching
    - Secondary query using initial criteria
    - Supports multi-level
    - Query Hint: “eclipselink.batch”
    - *Note: Can be faster then joining*

```
@NamedQuery( name="findAllEmployees",
        query="SELECT e FROM Employee e order by e.id"),
        hints={ @QueryHint( name=EclipseLinkQueryHints.BATCH,
                value="e.manager.phoneNumbers") }
```

EclipseLink Road Map

- 1.1.1 Release: May 12th, 2009
  - JPA 1.0
  - Java Architecture for XML Binding (JAXB) 2.0
  - Service Data Objects (SDO) 2.1.1 (JSR 235 RI)
  - OSGi packaging and usage examples
  - Spring Framework support
  - Initial Database Web Services (DBWS)
- 1.1.2 – Eclipse Galileo
- Current Development – Target EclipseLink 2.0
  - JPA 2.0: Reference Implementation
  - Advanced MOXy features and usability enhancements
  - Community enhancement requests
  - Milestones promoted into GlassFish V3

Agenda

- Java™ Persistence API Introduction
- EclipseLink Overview
- NetBeans IDE support for JPA
- Demos
  - NetBeans IDE

Java™ Persistence in NetBeans™ IDE

- NetBeans™ IDE: open-source, free IDE for Java and other technologies
- Current version 6.7 RC
- Out-of-the-box support for JPA since version 5.5 (aligned with Java EE 5 platform)
- JPA support integrated with other technologies
  - JavaServer Faces platform
  - EJB™ session beans
  - REST web services
  - Beans binding

NetBeans™ IDE JPA Features

- Wizards for common JPA artifacts
  - Persistence unit
  - Entity class
- Code generation
  - Entity classes from a database (reverse engineering)
  - Code to retrieve and persist entities (for both container- and application-managed contexts)
- Editor features
  - Hints (check the entity classes as you type in the editor)
  - JPA-aware refactoring (persistence.xml)

NetBeans™ IDE JPA Road Map

- More customization, especially in the entity class from database generation
- Integration with additional technologies (Spring Framework, NetBeans™ Platform)
- Java™ Persistence API 2.0

Agenda

- Java™ Persistence API Introduction
- EclipseLink Overview
- NetBeans IDE support for JPA
- Demos
  - NetBeans IDE

Java Persistence Development in NetBeans IDE

DEMO

Summary

- Java™ Persistence API
  - Standardized persistence
  - JPA 2.0 addressing community feedback
- EclipseLink
  - Comprehensive Persistence solution
  - Reference Implementation of JPA 2.0 in GlassFish Project V3
  - Many advanced and proven features
- NetBeans IDE
  - Out-of-the-box full-featured JPA support
  - Code generation features make it easy to get started
  - JPA support integrated with other technologies from the enterprise stack

For More Information

- Links
  - EclipseLink project
    www.eclipse.org/eclipselink
  - Java Specification Request (JSR) 317 0 JPA 2.0:
    www.jcp.org/en/jsr/detail?id=317
  - Java Persistence in the Java EE 5 Platform tutorial
    http://www.netbeans.org/kb/61/javaee/persistence.html
  - BLOG
    - java-persistence.blogspot.com
- Books
  - PRO EJB 3.0 Persistence
    - Mike Keith and Merrick Schincariol

JavaOne℠

Thank You

Doug Clarke
Director of Product Management,
Oracle Corporation
Andrei Badea,
Sun Microsystems

TS-5018