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

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

TS-5400
Agenda

➢ Java™ Persistence API Introduction

➢ EclipseLink Overview

➢ NetBeans IDE support for Java Persistence API (JPA)

➢ Demos
  • NetBeans IDE
  • Advanced EclipseLink JPA
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
  - Advanced EclipseLink JPA
Eclipse Persistence Services Project

- Eclipse runtime project
  - Nicknamed “EclipseLink”
  - Currently Incubating in Technology 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
- Commerically proven solution
  - Initial contribution of Oracle TopLink
History of EclipseLink

1996

TopLink for Java

©1997-98 The Object People

TopLink Version 3.

TopLink for Java

© 1998-2001 ObjectPeople, Inc.

TopLink Essentials

ORACLE

2008

TopLink

JavaOne®
Eclipse Persistence Services “EclipseLink”

Eclipse Persistence Services Project (EclipseLink)

Databases
XML Data
Legacy Systems
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 Eclipse Ecosystem
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

```xml
<properties>
  ...

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

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

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

  <property name="eclipselink.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

```xml
<property
  name="eclipselink.logging.level"
  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

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

XML:

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

API:

```java
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

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

- Supported Platforms (ServerPlatform)^:
  - None (default)
  - SunAS, OC4J*, WebLogic*, WebSphere*, JBoss
Concurrency Protection - Locking

- Optimistic concurrency is built into JPA, but no support for pessimistic locking is specified
- Will likely be addressed in a future JPA release
- All credible JPA implementations support pessimistic locks in some way or another
- EntityManager lock() method can be used with optimistic locking, and error handling
- EclipseLink query hint for pessimistic locking
  - query.setHint(PESSIMISTIC_LOCK, LockNoWait);
EclipseLink Caching

- EntityManager Factory
- EntityManager
- UnitOfWork
  - TX Cache
- Client Session
  - Isolated Cache
- Server Session
  - Shared Cache
- RDBMS
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

```java
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();
    }
}
```
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

```java
@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

- Delivery of monthly incubation milestones
  - Build and testing processes
  - Initial contribution functional
  - 1.0M3 was the last milestone

- 1.0 Release: June 2008
  - Specifications: JPA 1.0, Java Architecture for XML Binding (JAXB) 2.0, Service Data Objects (SDO) 2.1
  - OSGi packaging and usage examples
  - Spring Framework support

- Future Enhancements
  - JPA 2.0: Reference Implementation
  - Database Web Services (DBWS)
  - Data Access Service (DAS)
  - Simplified Data-Map Access and Dynamic Persistence
Agenda

> Java™ Persistence API Introduction

> EclipseLink Overview

> NetBeans IDE support for JPA

> Demos
  - NetBeans IDE
  - Advanced EclipseLink JPA
Java™ Persistence in NetBeans

- NetBeans IDE: open-source, free IDE not only for Java
- Current version 6.1
- 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
  - Enterprise JavaBeans 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)
Creating a Persistence Unit

- Persistence unit wizard supporting major persistence providers
  - TopLink
  - Hibernate
  - Kodo / OpenJPA
  - other (commercial) providers

- Generates a ready-to-use persistence.xml file
  - Data source
  - Provider-specific JDBC technology connection properties
  - Table creation

- Application server-aware

- Graphical persistence.xml editor
Entity Classes from Database

- Wizard to generate annotated entity classes from the tables in a database
- Application server-aware (can be pointed to a data source on your application server)
  - plain JDBC technology connections also supported
- Handles relationships by default (when a table is selected, the tables that it references through foreign keys are selected too)
- Generates named queries
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
  • Advanced EclipseLink JPA
Java Persistence Development in NetBeans IDE
Advanced EclipseLink JPA
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

➤ Technical sessions
  • TS-5509: Java™ Persistence API 2.0

➤ BOFs
  • BOF-5388: The Java™ Persistence API on the Grid

➤ Links
  • EclipseLink project: www.eclipse.org/eclipselink
  • Java Specification Request (JSR) 317 - 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/55/persistence.html
  • Doug’s BLOG: java-persistence.blogspot.com

➤ Books
  • PRO EJB 3.0 Persistence, Mike Keith and Merrick Schincariol
THANK YOU

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

TS-5400