Eclipse JPA
Getting Chocolate in the Peanut Butter

Doug Clarke
Principal Product Manager, Oracle TopLink
Eclipse Persistence Services Project co-Lead
douglas.clarke@oracle.com
A little about Me

- Principal Product Manager – Oracle TopLink
  - With product for 10 years
  - Product Developer
  - Consultant
  - Involved daily with development and customers
- Co-Lead Eclipse Persistence Services Project
- Frequent speaker at conferences and JUGs primarily on persistence related topics
Agenda

- Developing with JPA
  - Eclipse Dali: JPA Tooling
    - Demo
  - EclipseLink JPA
    - Demo
About Java Persistence API (JPA)

- Separate document bundled as part of EJB 3.0 specification
- Suitable for use in different modes
  - Standalone in Java SE environment
  - Hosted within a Java EE Container
- Standardization of current persistence practices
- Merging of expertise from persistence vendors and communities including: TopLink, Hibernate, JDO, EJB vendors and individuals
JPA—in a Nutshell

- A Java standard that defines:
  - how Java objects are stored in relational databases (specified using a standard set of mappings)
  - a programmer API for reading, writing, and querying persistent Java objects (“Entities”)
  - a full featured query language
  - a container contract that supports plugging any JPA runtime in to any compliant container.
JPA Entities

- Concrete classes (POJOs)
- 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 (“detached”)
Object-Relational Mappings

- Core JPA Mappings
  - Id
  - Basic
  - Relationships
    - OneToOne
    - OneToMany/ManyToMany
    - ManyToMany
  - And more...

- Annotations and/or XML
Annotations on Fields

@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;
}
}
Annotations on Properties

@Entity public class Customer {

    private String name;
    private Account account;

    @Id
    public String getName() { return name; }
    public void setName(String name) {
        this.name = name;
    }

    @OneToOne
    public Account getAccount() { return account; }
    public void setAccount(Account account) {
        this.account = account;
    }

}
Mappings in XML

```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 Configuration

- JPA runtime combines:
  - Java Classes
  - Mapping Metadata
  - Database schema
Why JPA Tools?

- How can you tell if they all match?
  - Deploy and run tests?
    - slow
    - find one problem at a time (fix, run, fix, ...)
    - definitive
  - Design time validation?
    - quick
    - finds all issues
    - validates against spec
    - runtime may not match spec
About Dali

- Support for the definition, editing, and deployment of Object-Relational (O/R) mappings for JPA Entities
- Simplify mapping definition and editing through:
  - intelligent mapping assistance
  - dynamic problem identification
  - generation and automated mapping wizards
- Extensible frameworks and tools so vendors and open source projects can provide specific support for their JPA runtimes
- A subproject of the Web Tools Platform (WTP)
Dali Goals

- Simplicity
  - Intelligent mapping assistance and automated generation

- Intuitiveness
  - Use existing and consistent modeling and tooling practices in Eclipse
  - Light-weight views offer assistance but don’t get in the way of power users

- Compliance
  - Support any and all compliant runtime implementations
  - Test using JPA Reference Implementation

- Extensibility
  - Provide the ability for vendors and open source projects to seamlessly add their own value-add features
Why ‘Dali’?

- JPA supports “The Persistence of Memory”—which is the title of a well known Salvador Dali painting.

© 2005 Salvador Dali, Gala-Salvador Dali Foundation/Artists Rights Society (ARS), New York
Dali and the Eclipse IDE

- JPA Mapping Validation
- JPA Structure and Details Views
- Java Source Editor enhancements
Mapping Validation

- Annotations and/or XML used to define JPA Entities.

- JDT validates syntax, but doesn’t understand what the annotations mean.
Mapping Validation

- Java Source Editor enhancements
- Mapping Problem Markers

```java
@Entity
public class Address {
    @Id
    private int id;
    private String street;
    private String city;
    private String province;
    private String country;
}
```

**ADDRESS**

<table>
<thead>
<tr>
<th>ID</th>
<th>STREET</th>
<th>...</th>
<th>STATE</th>
</tr>
</thead>
</table>

Default mapping won’t work!

The column province cannot be found on table Address

Address.java
Mapping Assistance

- JPA Details View

```java
@Entity
public class Address {
    @Id
    private int id;
    private String street;
    private String city;
    private String province;
    private String country;
}
```

![Persistence Properties](PersistenceProperties.png)

![Problems](Problems.png)

- The column province cannot be found on table Address

![ADDRESS](ADDRESS.png)

<table>
<thead>
<tr>
<th>ID</th>
<th>STREET</th>
<th>...</th>
<th>STATE</th>
</tr>
</thead>
</table>

© Copyright 2007, Oracle Corporation

Colorado Software Summit: October 21 – 26, 2007
Basic Mapping

No Mapping Errors!

ADDRESS

| ID | STREET | ... | STATE |

```java
@Entity
public class Address {
    @Id
    private int id;
    private String street;
    private String city;
    @Column(name="STATE")
    private String province;
}
```

Persistence Properties

- Map As: Default (Basic)
- Column:
  - Name: STATE
  - Table: Default (Address)
  - Insertable: Default (True)
  - Updatable: Default (True)
JPA Structure View

- Provides a JPA specific view of Java Class or ORM XML Mapping File
- A thumbnail sketch of how an Entity is mapped
- Supports navigation between mappings
- Automatically adjusts to either property or field mapping in Java
- Represents structure in Java and XML artifacts
JPA Perspective

```
@Entity
@Table(name="PETS")
public class Pet extends NamedEntity {

    @ManyToMany
    private Owner owner;

    @Basic
    private Date birthDate;

    @Errors (1 item)
    Column "birthDate" cannot be resolved

    Map As: Basic
    Column: Default (birthDate)
    Table: Default (PETS)
    Fetch: Default (Eager)
    Optional: Default (False)
    Temporal: Default (Ordinal)
    Enumerated: Default (Ordinal)
```
Synchronizing Persistence.xml

- In Java SE environment, persistence.xml must list the Entities—Dali offers synchronization.
Synchronizing Persistence.xml

```xml
<?xml version="1.0" encoding="UTF-8"?>
<persistence xmlns="http://java.sun.com/xml/persistence"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://java.sun.com/xml/persistence http://java.sun.com/xml/persistence_1_0.xsd">
    <persistence-unit name="jdbc">
        <class>model.Customer</class>
        <class>model.CustomerInfo</class>
        <class>model.Invoice</class>
        <class>model.Phone</class>
    </persistence-unit>
</persistence>
```
Integrated F1 Help

- In any Dali view you can hit F1 to get context specific help.

One-to-one mapping

Use a One-to-One Mapping to define a relationship with one-to-many multiplicity.

1. In the Persistence Outline view, select the field to map. The Persistence Properties view (for attributes) displays the properties for the selected.
2. In the Map As field, select One-to-one.
3. Use this table to complete the remaining fields on the General tab in Persistence Properties view.

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
<th>Default</th>
</tr>
</thead>
<tbody>
<tr>
<td>Target Entity</td>
<td>The entity to which this attribute is mapped.</td>
<td>null</td>
</tr>
<tr>
<td>Cascade Type</td>
<td>See &quot;Cascade Type&quot; for details:</td>
<td>Default</td>
</tr>
<tr>
<td></td>
<td>- Default</td>
<td></td>
</tr>
<tr>
<td></td>
<td>- All</td>
<td></td>
</tr>
<tr>
<td></td>
<td>- Persist</td>
<td></td>
</tr>
<tr>
<td></td>
<td>- Merge</td>
<td></td>
</tr>
<tr>
<td></td>
<td>- Remove</td>
<td></td>
</tr>
<tr>
<td>Fetch Type</td>
<td>Defines how data is loaded from the database.</td>
<td>Eager</td>
</tr>
</tbody>
</table>

© Copyright 2007, Oracle Corporation

Doug Clarke: Eclipse JPA: Getting Chocolate in the Peanut Butter
Dali 0.5 Release Features

- Support for majority of annotations including relationships
- Integration with WTP RDB components
- JPA Details view supports mapping using database schema with table and column drop downs
- Design time validation of mappings against object and data model (including unspecified default mappings)
- Problems markers (e.g., Entity missing @Id)
- Wizards for basic Entity generation from tables
Dali 1.0 Features

- Support for XML Mapping File (orm.xml) configuration
  - Uses same views as JPA Annotation editing
- Integration with Data Tools Platform (DTP) for database meta-data
- Facet based functionality for better integration with various WTP project types
- Enhanced validation
- Extensibility for vendor specific extensions
Dali 1.0 (WTP 2.0)

- Support for XML mapping (orm.xml)
- First class component of WTP—integrated as project facet (Complete)
- Enhanced design-time validation for XML and Java Annotations mappings as well as the combination as defined by the spec. (Complete)
- Migrate from WTP RDB to Data Tools Project for database access. (Complete)
- Initial API for adopters/extenders to add runtime specific features (e.g., DDL generation)
Dali Development Scenarios

- Meet in the Middle
  - existing object and data models
- Bottom Up
  - generate mapped object model from data model
- Top Down
  - generate data model from mapped object model
The advantage of MITM is that you can focus on getting your object model and data models correct.

- **Table $\neq$ Class**
  - 1 Table could be N classes (using embedded)
  - N tables could be 1 class (with secondary table)

- Use Java language features like inheritance not present in relational model

- **Dali’s validation makes MITM practical**
  - Avoids map, deploy, debug cycle
  - Provides access to database schema to provide valid choices
Bottom Up

- Generate Entities from Tables
  - Great way to bootstrap a JPA application from an existing database
  - Uses an Entity == Table approach
  - Do it once and then modify the generated Entities
    - Dali mapping validation will help identify issues resulting from modifications to Entities
Top Down

- Generate DDL from Entities
  - Some support in Dali 0.5 release
  - Removed in 1.0 in favor of using full featured support implemented by JPA runtimes.
  - Extension point in Dali to allow for plugging in runtime DDL generation.
Dali Adoption

- Oracle
  - Providing majority of resources for Dali
  - Planning to build extensions to Dali for EclipseLink
- RedHat/JBoss
  - Incorporating Dali into JBossIDE as part of Hibernate/JPA toolset
- BEA
  - Considering incorporating Dali into BEA Workshop
- SAP
  - Shipped Dali in Eclipse based SAP NetWeaver Developer Studio Java EE 5 preview
- IBM
  - Incorporating Dali into future tooling projects
Dali Summary

- Dali: the WTP project bringing developer productivity to JPA
  - Mapping validation to avoid the map, deploy, debug cycle
  - Intelligent mapping assistance to avoid problems and speed up the process of mapping
  - Integrated with WTP to support development for Java SE and EE
Demo

JPA development with Dali
Eclipse Persistence Services

- 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
Java SE, Java EE, SCA, Spring, OSGi

Eclipse Persistence Services (EclipseLink)

JPA
MOXy
SDO
DBWS
EIS

JDBC

Eclipse Persistence Services (EclipseLink)

Relational Databases

XML

Packaged Apps
Legacy Systems

Doug Clarke: Eclipse JPA: Getting Chocolate in the Peanut Butter
EclipseLink History
Key Product Features

- Metadata architecture
- Comprehensive visual mapping editors
- Advanced mapping support and flexibility
- Query flexibility
- Caching
- Concurrency protection
- Transaction support and integration
- Performance tuning options
- Application server features
EclipseLink Query Execution

- SQL
- Stored Procedure
- JPQL
- Expressions
- Query By Example

EclipseLink

Query Framework

Cache

Cache Hit?

Mappings

Cache Result?

Object Builder

Database

Application

Objects

Query

SQL

Rows
Caching

- Hold objects in-memory to avoid unnecessary database trips and object construction
- Cache manages “identity” to support bidirectional and cyclical relationships
- Flexible caching options ensure that you get maximum performance
- Numerous locking, refreshing, and synchronization options are available to minimize cache staleness
- Queries can be run in-memory only against the cache
- Cache Coordination supports clustering
Caching Architecture

- EntityManager
  - UnitOfWork
    - TX Cache
  - Session
    - Isolated Cache
- EntityManager Factory
- Server
  - Shared Cache
- Cache Coordination
  - JMS (MDB)
  - RMI
  - CORBA
  - IIOP
Advanced Caching

- Cache Invalidation/Expiration
  - Time to live
  - Fixed Times
  - Programmable (external notification)

- Shared and Isolated caching

- Cache Coordination
  - Messaging
    - JMS, RMI, IIOP, CORBA, OC4J-JGroups
  - Type specific configuration
    - Modes: Sync, Sync+New, Invalidate, None

- All configurable on a per type basis
Locking

- CRITICAL to avoid DB corruption in concurrent applications
- Java Developers want to think of locking at the object level
- Databases may need to manage locking across many applications
- EclipseLink is able to respect and participate in locks at database level
  - Optimistic: Numeric, Timestamp, All fields, Selected fields, Changed field
  - Pessimistic
Transactions

- Java apps typically support many clients sharing small number of db connections
  - Ideally would like to minimize length of transaction on database
Transaction Features

- Extensive Java transaction support through “Unit Of Work” feature
  - Minimizes database interactions
    - Calculates the minimal change set at commit time (deferred write)
    - Only the minimal updates are sent to the database
  - Respect database integrity
    - Orders INSERT, UPDATE and DELETE statements
  - JTA and RESOURCE_LOCAL support
TX Optimizations

Optional mark.dirty optimizations

- Object-level
- Attribute-level

- UnitOfWork issue SQL prior to commit
- Bulk Update and Delete
- Cached entities effected
Performance and Tuning

- Highly configurable and tunable
  - Guiding principle – minimize and optimize database interactions
  - No two applications are the same, EclipseLink allows for decisions on what specific behavior needs to be configurable depending on situation

- Flexibility of EclipseLink allows efficient business models and relational schemas to be used

- Leverages underlying performance tuning features
  - Java, JDBC and the underlying database technology
Performance and Tuning

- Minimal Writes, Updates
- Batch Reading, Writing
- SQL ordering
- Transformation support
- Existence checks
- Stored procedures
- Statement Caching
- Scrolling cursors
- Projection Queries
- Partial Attribute Queries
- “Just in Time” reading
- Automatic change detection
- Caching policies and sizes
- Parameterized SQL (binding)
- Pre-allocation of sequence numbers
- Cache Coordination
- Optimistic, Pessimistic locking
- Joining object retrieval optimization
- In memory querying
- Dynamic queries

AND MUCH MORE!
EclipseLink JPA Extensions

- Maintain the JPA configuration and programming model
- Expose extended functionality
  - Persistence Unit properties
  - Query hints
  - Custom annotations
  - Native API (minimize required usage)
EclipseLink JPA Extensions

- Extensions supported through annotations and XML
- Mapping
  - @BasicMap, @BasicCollection, @PrivateOwned, @JoinFetch
  - @Converter, @TypeConverter, @ObjectTypeConverter
- @Cache
  - type, size, isolated, expiry, refresh, cache usage, coordination
  - Cache usage and refresh query hints
- @NamedStoredProcedureQuery
  - IN/OUT/INOUT parameters, multiple cursor results
EclipseLink JPA Extensions

- Locking
  - Non-intrusive policies @OptimisticLocking
  - Pessimistic query hints
- JDBC Connection Pooling
- Logging: Diagnostics, SQL, Debugging
- Weaving for lazy fetch and change tracking
  - Dynamic and Static
- Customization
  - Entity Descriptor: @Customizer, @ReadOnly
  - Session Customizer
@Entity
@Cache(type=SOFT_WEAK, coordinationType=SEND_OBJECT_CHANGES)
@OptimisticLocking(type=CHANGED_COLUMNS)
@Converter(name=“money”, converterClass=MoneyConverter.class)
public class Employee {
    @Id
    private int id;

    private String name;

    @OneToMany(mappedBy=“owner”)
    @PrivateOwned
    private List<PhoneNumbers> phones;

    @Convert(“money”)
    private Money salary
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 using JDK/JRE’s
    - javaagent:
      - Optional
      - Static weaving also supported
        - Weaving of .class files before deployment
DB Platform Extensions

- Oracle native SQL and custom operators
- Isolated session cache and connections for use with Oracle DB’s VPD/OLS
- Proxy Authentication
- Support for XDB-XMLType and SQLX
- Stored Procedure & Function
- TIMESTAMP & TIMESTAMPTZ (oracle.sql)
- Configurable value return from write
- Spatial, Object-Relational, ADTs
- Historical Flashback...
Historical Support

- Generic support for any RDBMS
  - Custom support for Oracle 10g+
    - Flashback
  - Configure and use change management of objects over time
    - Versioning
    - Auditing
- Point in time querying
- Historical Session
Demo

EclipseLink JPA
EclipseLink Summary

- First comprehensive Open Source Persistence solution
  - EclipseLink JPA: Object-Relational
  - EclipseLink MOXy: Object-XML
  - EclipseLink SDO: Service Data Objects
  - EclipseLink DBWS: Database Web Services
  - EclipseLink EIS: Non-Relational using JCA

- Mature and full featured
- Get involved
EclipseLink Information

- [www.eclipse.org/eclipselink](http://www.eclipse.org/eclipselink)
- Newsgroup: [eclipse.technology.eclipselink](http://eclipse.technology.eclipselink)
- Wiki: [wiki.eclipse.org/EclipseLink](http://wiki.eclipse.org/EclipseLink)
- Blogs
  - Committer Team blog: [eclipselink.blogspot.com](http://eclipselink.blogspot.com)
  - My blog: [java-persistence.blogspot.com](http://java-persistence.blogspot.com)
Dali Information

- Dali JPA Tools—open source tools for JPA development
  
  [http://www.eclipse.org/dali](http://www.eclipse.org/dali)

- EJB 3.0 & JPA Specification
  

- TopLink Essentials—open source JPA Reference Implementation
  
  [http://otn.oracle.com/jpa](http://otn.oracle.com/jpa)
  [http://glassfish.dev.java.net/](http://glassfish.dev.java.net/)

- JPA white papers, tutorials and other resources
  
  [http://otn.oracle.com/jpa](http://otn.oracle.com/jpa)

- Java Developer’s Journal—September 2006: Hello Dali!