Eclipse Persistence Services
The Full Monty

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
What you will learn

- What the Eclipse Persistence Services Project is
- How this project can be used and its benefits
- Why you will want to use this project
- How you can get involved
What is Eclipse?

- Eclipse is an open source community
- Eclipse is more than just an IDE
  - Equinox (OSGi), Rich Client Platform (RCP), Higgins (Trust Framework), ...
  - Incubating
    - Maya (Deployment Framework)
    - Persistence Services Project (EclipseLink)
- Proposals
  - SOA, Rich Server Platform, ...
Why Eclipse?

- Eclipse has a strong and vibrant community with an effective governance model
- Good reputation for quality
- Interest from within the Eclipse ecosystem
- Oracle has had a positive experience with its existing participation in Eclipse projects
  - Projects lead by Oracle: Dali, BPEL, JSF
  - Other Oracle contributions: WTP and DTP
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
Oracle TopLink
Importance

- 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, Java SE, Web, Spring, and OSGi
- Any JDBC/SQL compliant database
- Extensible and pluggable
- Schema generation
- Key infrastructure:
  - Caching, Locking, Query Framework, Mapping, ...
- ... plus many valuable advanced features
EclipseLink Caching

- Entity caching
  - L2 shared across transactions/users
  - Coordination in a clustered deployment

- Application specific configuration
  - Cache isolation: per client (EM) or shared
  - Cache Type and Size: Weak, Soft-Weak, Full, None
  - Expiration/Invalidation
    - Time to live, Time of day, API
  - Coordination (cluster-messaging)
    - Messaging: JMS, RMI, CORBA, RMI-IIOP, ...
    - Mode: SYNC, SYNC+NEW, invalidate, NONE
Caching Architecture

- EntityManager
  - UnitOfWork
    - TX Cache
  - Session
    - Isolated Cache

- EntityManager Factory

- Server
  - Shared Cache

Cache Coordination
- JMS (MDB)
- RMI
- CORBA
- IIOP
Configuring the Cache

- Default: objects read are cached and trusted
- Configuration by entity type important
  - Volatility of data
  - Shared usage of data
- Configuration Parameters
  - Cache isolation, type, size, expiry, coordination
  - Refreshing
    - By query (use-case) or descriptor (always)
- Locking is the only way to avoid potential data corruption in concurrent write scenarios
Locking

- Prevent data corruption !!!
- Java Developers 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
Query Framework

- Queries can be defined using
  - Entity Model: JPQL, Expressions, Query-by-example
  - Database: SQL, Stored Procedures

- Customizable
  - Locking, Cache Usage, Refreshing
  - Optimizations: Joining, Batching, parameter binding
  - Result shaping/conversions

- Static or Dynamic
  - Stored Procedure support
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
Mapping Extensions

@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
}
Database Platform

- Native SQL (dialect) support with custom operators
- Stored Procedure & Function
- Extensible Advanced Data Types support (Struct)
- Database Security
  - Oracle DB’s VPD/OLS and Proxy Authentication
- Configurable value return from write
- Supported platforms (default = Auto)
  - MySQL, Derby, Oracle, DB2, Sybase, SQLServer, TimesTen, PostgreSQL, SQLAnywhere, HSQL, Informix, ...
Server Platform

- Simplified configuration and mediator for host container environment
- Enables
  - Direct JTA integration
  - Data Source/JDBC connection un-wrapping
  - JMX MBean deployment
  - Logging integration
- Current Server Platforms
  - SunAS/GlassFish, OracleAS/OC4J, WLS, WAS, JBoss
Performance and Tuning

- Highly configurable and tunable
  - Principle: minimize and optimize database calls
  - Enable application specific tuning
- Flexibility allows efficient business models and relational schemas to be used
- Leverages underlying performance tuning features
  - Java, JDBC and the underlying database technology
EclipseLink JPA Config

- JPA (portable)
  - Persistence.xml with EclipseLink properties
  - Mapping: Annotations and/or orm.xml
  - Query hints

- EclipseLink
  - Sessions Configuration (sessions.xml)
  - Mapping using XML or Code

- EclipseLink JPA
  - JPA + EclipseLink configurations options
  - EclipseLink annotations
EclipseLink MOXy

- Provides complete Object-XML mapping
  - Allows developers to work with XML as objects
  - Efficiently produce and consume XML
  - Document Preservation

- Supports Object-XML standard - JAXB
  - Provides additional flexibility to allow complete control on how objects are mapped
EclipseLink MOXy Benefits

- Rich set of mappings providing complete control and flexibility to map objects to any XSD
  - Direct, composite object, composite collection, inheritance, positional, path, transformation ...

- Development Approaches
  - Model + Annotations ➔ XSD
  - XSD ➔ Model + Annotations
  - Model + Mappings (Annotations or XML)

- Supports any JAXP compliant parser
  - SAX, DOM, StAX

Visual Mapping support using Workbench
EclipseLink MOXy: JAXB

```java
JAXBContext ctx = JAXBContext.newInstance(classes);
Marshaller marshaller = ctx.createMarshaller();

Customer customer = new Customer();
customer.setFirstName("William");
customer.setLastName("Gibson");

marshaller.marshal(customer, System.out);
```

```
jaxb.properties:

    javax.xml.bind.context.factory =
    org.eclipse.persistence.jaxb.JAXBContextFactory
```
DEMO

EclipseLink MOXy (JAXB)
EclipseLink DBWS

- Simplified and efficient access to relational data through Web Services
- Minimal configuration with development utilities to retrieve metadata and generate/package Web Service
- Developers can fully customize the database access and XML mapping of the data
- Ideal for usage within SOA/SCA
EclipseLink DBWS

DBWS Specification

ANT Java

import definitions

development

runtime

OR-Map (XML)

OX-Map (XML)

XR Service (XML)

Web Service

EclipseLink DBWS

Data Source

RDBMS
EclipseLink SDO

"What can you do?"
- Marshall/Unmarshall objects to/from XML
- Define Types/Properties programmatically or derive from XSD
- Generate JavaBean classes from XSD
- Advanced mapping support for greater flexibility

"Why would you use it?"
- Schema/Structure unknown at compile time
- Declarative metadata based tools/frameworks
- XML-centric applications, need open content support
- Dynamic content user interfaces
DEMO

EclipseLink SDO
EclipseLink EIS

- Provide persistence support for non-relational data stores using Java EE Connector Architecture (JCA)
- Mapping interaction inputs and outputs to persistent domain model
  - XML mapping leveraging EclipseLink MOXy
  - Common Client Interface (CCI) mapping
- Visual mapping Workbench support
- Out of the box support for:
  - MQSeries, OracleAQ, Sun JCA, XML Files
EclipseLink and OSGi

- Work with OSGi expert group to define OSGi persistence services blueprint
- Deliver EclipseLink as OSGi bundle(s)
- Show through examples how to leverage within an OSGi solution
- Address technical challenges as a community
Combining Services

- Metadata based approach allows the same domain model to be mapped with multiple persistence services
  - Supports usage within Web Services/SOA/SCA
  - Domain model can be shared between persistence services (JPA, MOXy, EIS)
  - Transformations are bidirectional:
    - Unmarshall XML to objects and then persist
    - Marshall persistent objects to XML
Common Domain Model

Schema-1

EclipseLink MOXy

domain model

EclipseLink JPA

EclipseLink MOXy

Schema-2

EclipseLink JPA

EclipseLink JPA

Schema-3

EclipseLink JPA

Schema-4

Doug Clarke: Eclipse Persistence Services, The Full Monty
EclipseLink and Spring

- EclipseLink JPA
  - Container
  - Template
- EclipseLink Native ORM Template
- EclipseLink MOXy
  - Direct, Spring WS, Spring Remoting, ...

- and many more possibilities...
  - Spring Batch, Spring OSGi, ...
@Repository
@Repository
@Transactional
public class EntityManagerClinic implements Clinic {

@PersistenceContext
private EntityManager em;

public Collection<Owner> findOwners(String lastName)
    throws DataAccessException
{
    Query query =
        em.createNamedQuery("Employee.findOwners");
    query.setParameter("lastName", lastName + "%%");
    return query.getResultList();
}
Using EclipseLink with the Spring Framework
EclipseLink in the Eclipse Ecosystem

- Provide an Eclipse persistence solution easily consumable by any project
  - Storage of metadata in RDBMS, XML, EIS
  - XML Messaging infrastructure
- Eclipse Projects
  - Dali JPA Tooling Project
  - Teneo to use EclipseLink for EMF model persistence
  - Maya for storage of deployment configuration
  - SOA Project for EclipseLink SDO
Where are we going?

- Delivery of initial 0.1-incubation milestone
  - Build and testing processes
  - Initial contribution functional
  - Spring Framework support
- Specifications: JAXB 2.0, SDO 2.1, JPA 2.0
- OSGi packaging and usage examples
- Database Web Services (DBWS)
- Data Access Service (DAS) - SDO with JPA
- Simplified DataMap Access and Dynamic Persistence
How can you get involved?

- **Users**
  - The 0.1-incubation milestone will be available soon
  - Try it out and provide feedback
  - File bug reports and feature requests

- **Contributors**
  - Contribute to roadmap discussions
  - Bug fixes

- **Committers**
  - Very interested in growing committer base
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
More Information

- www.eclipse.org/eclipselink
- Newsgroup: eclipse.technology.eclipselink
- Wiki: wiki.eclipse.org/index.php/EclipseLink
- Blogs
  - Committer Team blog: eclipselink.blogspot.com
  - My blog: java-persistence.blogspot.com