Introducing EclipseLink: The Eclipse Persistence Services Project

Shaun Smith
EclipseLink Ecosystem Development Lead
Principal Product Manager, Oracle TopLink
shaun.smith@oracle.com

© 2007 Oracle; made available under the EPL v1.0
A little about Me

- Eclipse Persistence Services Project Ecosystem Development Lead
- Eclipse Dali JPA Tools Project Co-Lead
- Eclipse Teneo Project Committer
- Product Manager for Oracle TopLink
  - Involved with object-relational and object-XML mapping technology for over 10 years.
What you will learn

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

1996

1996 - 2007
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

- Providing enterprise data access and manipulation services to the Eclipse Ecosystem
EclipseLink in the Eclipse Ecosystem

- Provide an Eclipse persistence solution easily consumable by any project
  - Storage of (meta)data in RDBMS, XML, EIS
  - XML Messaging infrastructure

- Eclipse Projects
  - Dali JPA Tooling Project
  - EclipseLink for EMF model persistence in Teneo
  - Storage of deployment configuration for Maya
  - Possible usage:
    - EclipseLink SDO for Swordfish (SOA) Project
    - Open Financial Market Platform
    - Riena
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 from Entities
- 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
  - 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
- EntityManager Factory
- UnitOfWork
  - TX Cache
- Session
  - Isolated Cache
- Server
  - Isolated Cache
  - Shared Cache
- Cache Coordination
  - JMS (MDB)
  - RMI
  - CORBA
  - IIOP
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
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 MOXy
“Mapping Objects to XML”

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

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

Java App

EclipseLink MOXy

document unmarshalling produces objects

results are returned as raw xml

XPath is used to specify mapping

object creation and updates through object-level API
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)

- Visual Mapping support using Workbench
- Partial Document Mapping
- Document Preservation
- Supports any JAXP compliant parser
  - SAX, DOM, StAX
Challenge: XML Development

Objective—obtain employee number

- **JAXP**
  
  ```java
  Node childNode = employeeElement.getFirstChild();
  while(childNode != null) {
    if(childNode.getNodeName().equals("employee-number")) {
      Node employeeNumberTextNode = childNode().getFirstChild();
      employeeNumber = new Integer(employeeNumberTextNode.getNodeValue()).intValue();
    }
    childNode.getNextSibling();
  }
  ```

- **Using XML binding**
  
  ```java
  employee.getEmployeeNumber();
  ```
MOXy API has Standard API

JAXB 2.0 Standardized runtime API

```java
// Instantiate the JAXB context. The context path
// indicates which classes are involved in the XML binding
JAXBContext context =
    JAXBContext.newInstance(CONTEXT_PATH);

// Unmarshal the objects from XML
File file = new File("input.xml");
Unmarshaller unmarshaller = context.createUnmarshaller();
Customer customer = (Customer)
    unmarshaller.unmarshal(file);

// Marshal the objects to XML
Marshaller marshaller = context.createMarshaller();
mapper.marshal(customer, System.out);
```
Mapping in MOXy

- Powerful mapping approach:
  - MOXy uses XPath expressions to identify XML content that is mapped:
    - XPath by Name, by Path and Name, by Position, Self XPath
- Extensive Mapping Types
  - Direct
  - Composite Object
  - Composite Collection
  - Direct Collection
  - Relationships
  - Transformation
  - Complex Type Inheritance
Direct Mapping: Attribute

- Mapping a Java field to an XML *attribute* is done with a DirectMapping and XPath (name).

```
: Customer

id = 1234

<customer id="1234"/>

XPath = @id
```
Direct Mapping: Elements

- Mapping a Java field to an XML *element* is done with a DirectMapping and XPath (path and name)

```
: Customer

firstName = "Jane"
lastName = "Doe"

<Customer>
  <first-name>Jane</first-name>
  <last-name>Doe</last-name>
</Customer>
```

```
XPath = first-name/text()
```

```
XPath = last-name/text()
```
Composite Object: Elements by Position Example

- An object may have multiple composite object mappings to the same reference class. Each composite object mapping must have a unique XPath, e.g.:
  - billingAddress is address[1]
  - shippingAddress is address[2]
- MOXy maintains the order of elements when marshalling to XML.
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 SDO

- Powered by EclipseLink MOXy

- What can you do?
  - Marshall/Unmarshall SDO objects to/from XML
  - Define Types/Properties programmatically or derive from XSD
  - Generate SDO classes from XSD

- 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
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 powered by EclipseLink MOXy
  - Common Client Interface (CCI) mapping
- Visual mapping Workbench support
Combining EclipseLink Component 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 to relational db
    - Marshall persistent objects to XML
Common Domain Model

Schema-1
EclipseLink MOXy

Schema-2
EclipseLink MOXy

Schema-3
EclipseLink JPA

Schema-4
EclipseLink JPA

domain model
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
Where are we going?

- Delivery of initial 1.0 incubation Milestone
  - Build and testing processes implemented
  - Initial contribution functional
  - Spring Framework support
- Specifications: JPA 1.0 (EJB 3.0), JAXB 2.1, SDO 2.1
- Data Access Service (DAS) - SDO with JPA
- Simplified DataMap Access and Dynamic Persistence
- Database Web Services (DBWS)
- OSGi packaging and usage examples
How can you get involved?

- **Users**
  - The 1.0 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/EclipseLink
- Mailing Lists:
  - eclipselink-dev@eclipse.org
  - eclipselink-users@eclipse.org
- Blogs
  - Committer Team blog: eclipselink.blogspot.com