EclipseWorld 2008

October 28-30

Eclipse Persistence Services
The Full Monty

Peter Krogh, Oracle
EclipseLink Project co-Lead
Doug Clarke, Oracle
EclipseLink Project co-Lead

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

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

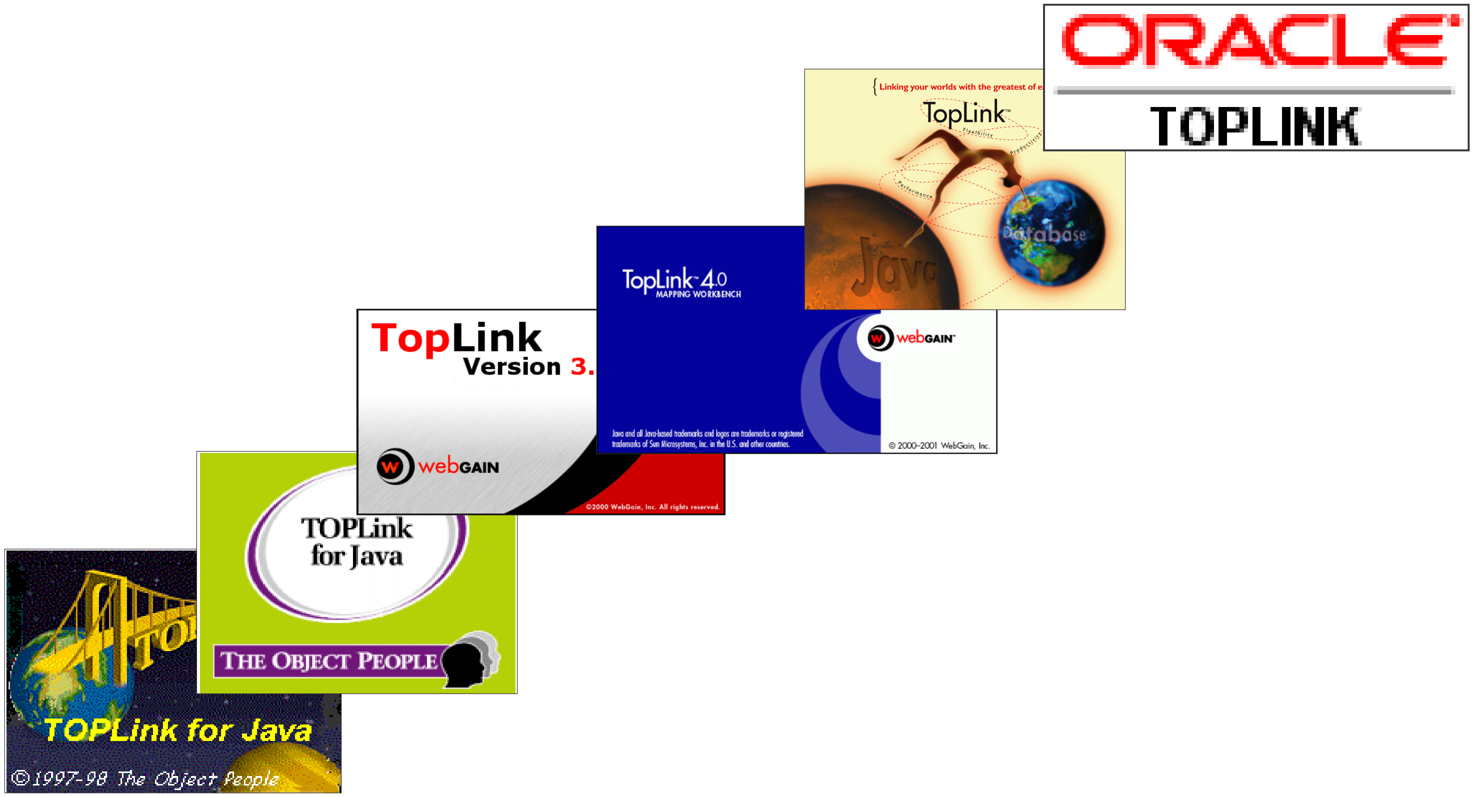
What is Eclipse?

- Eclipse is an open source community
- Eclipse is more then 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, …

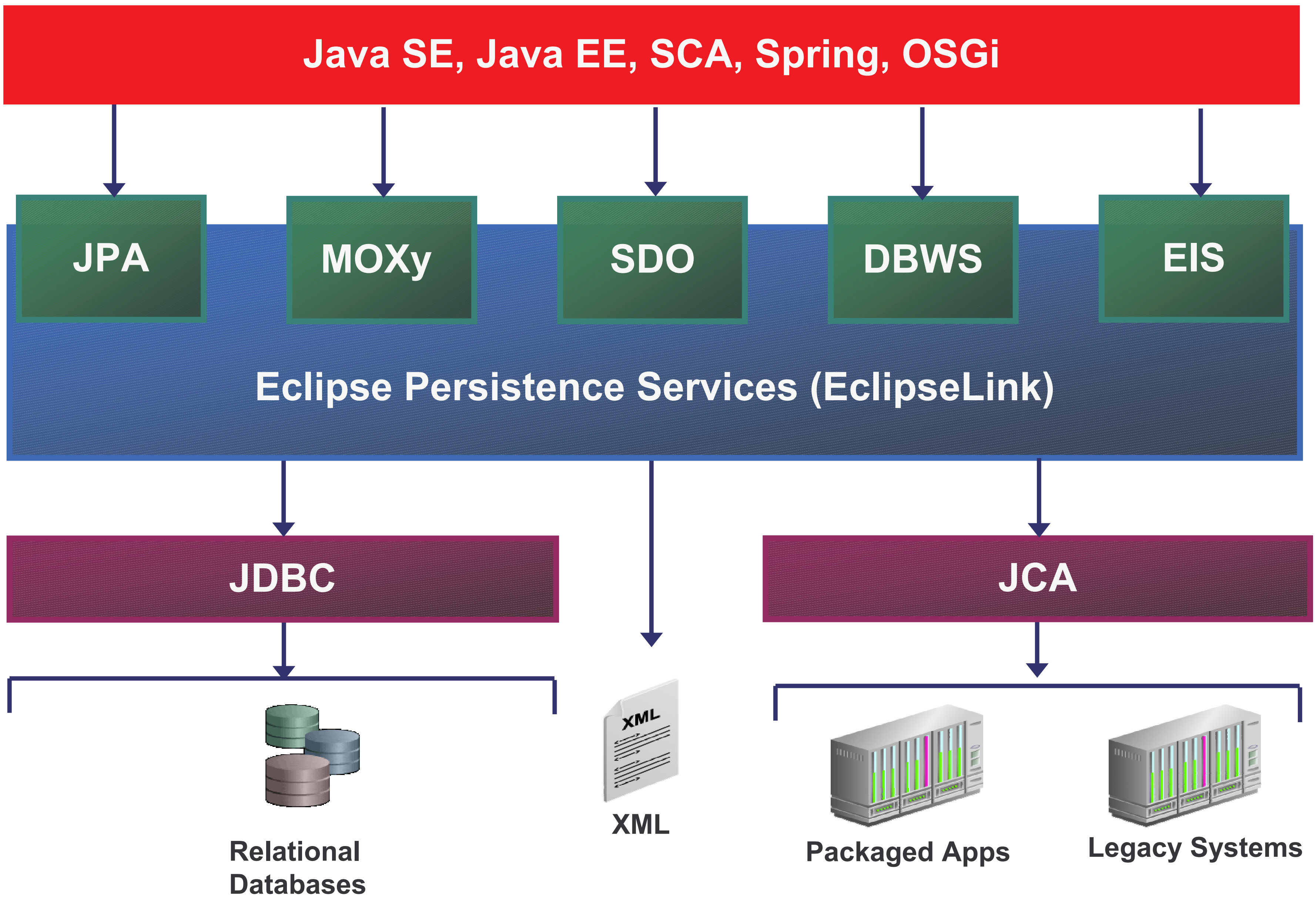
Why Eclipse?

- 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

History of EclipseLink


TOPLink for Java
©1997-98 The Object People
TOPLink for Java
THE OBJECT PEOPLE
TopLink Version 3.
webGAIN
©2000 WebGain, Inc. All rights reserved.
TopLink 4.0 MAPPING WORKBENCH
Java and all Java-based trademarks and logos are trademarks or registered trademarks of Sun Microsystems, Inc. in the U.S. and other countries.
webGAIN
© 2000-2001 WebGain, Inc.
Linking your worlds with the greatest of e
TopLink
Java
Database
ORACLE
TOPLINK

1996 → 2008



Java SE, Java EE, SCA, Spring, OSGi

JPA

MOXy

SDO

DBWS

EIS

Eclipse Persistence Services (EclipseLink)

JDBC

JCA

Relational Databases

XML

Packaged Apps

Legacy Systems

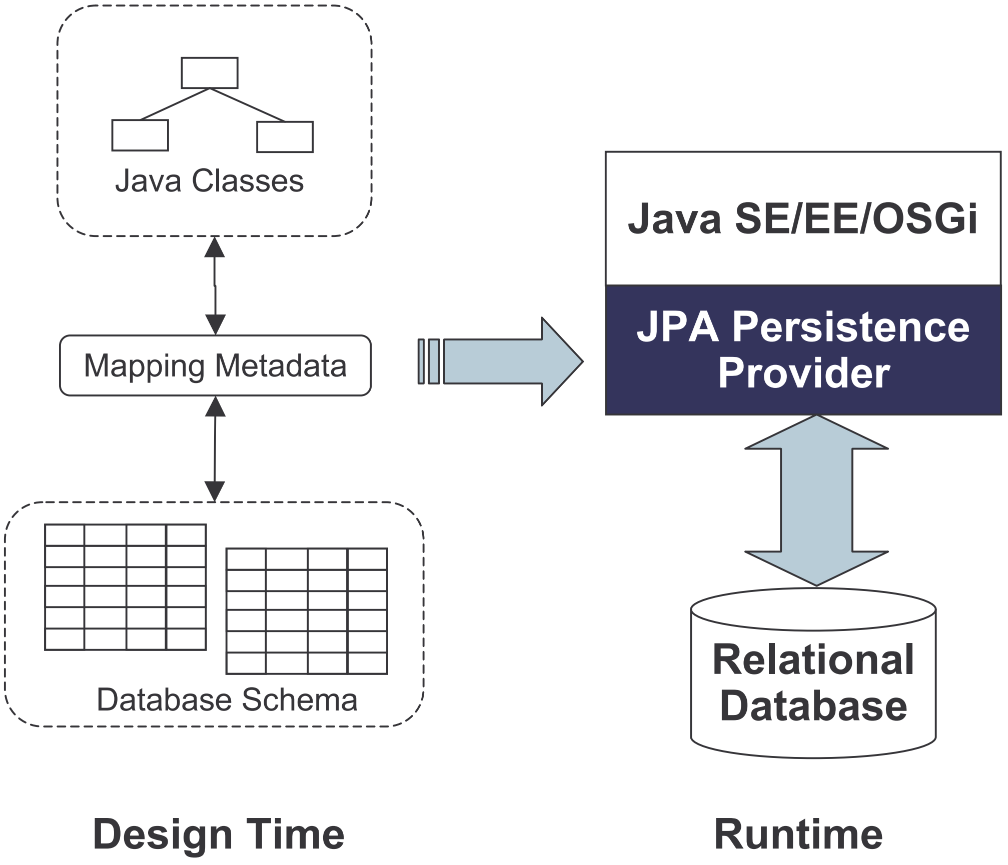
Importance

- First comprehensive open source persistence solution
  - Object-Relational and much more
- Based upon product with 13 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
- Upcoming JPA 2.0 Reference 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

Where does EclipseLink JPA fit?

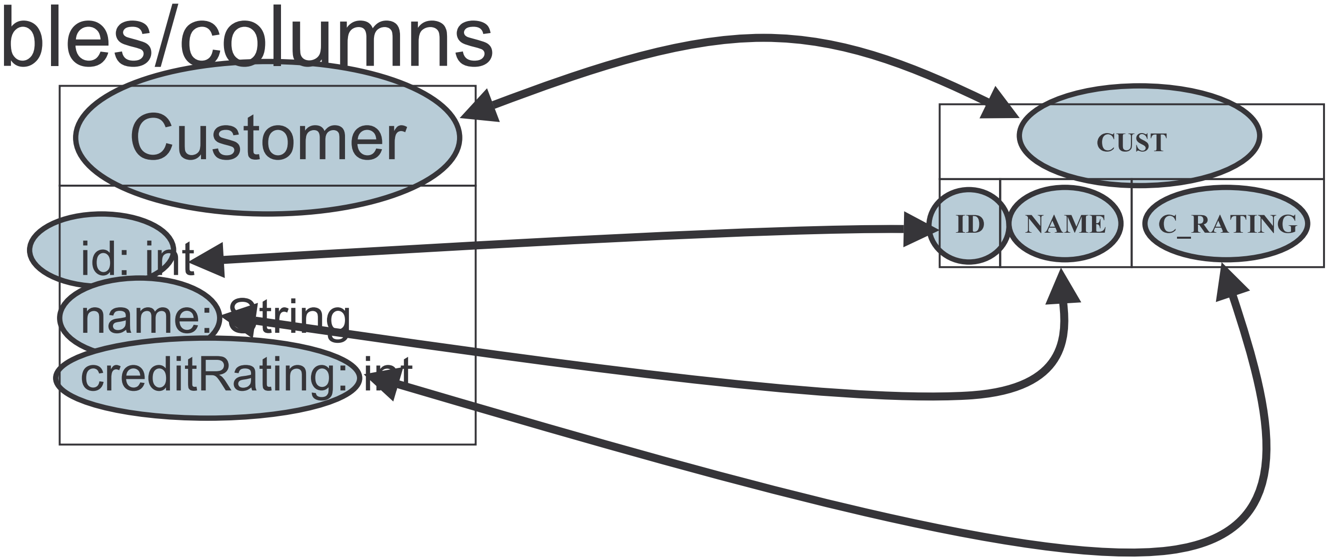


EclipseLink JPA Config

- 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

Mapping

- The activity of ‘Mapping’ is the process of connecting objects/attributes to tables/columns



Object-Relational Mappings

- Core JPA Mappings
  - Id
  - Basic
  - Relationships
    - OneToOne
    - ManyToOne
    - OneToMany
    - ManyToMany
  - And more…
- Annotations and/or XML

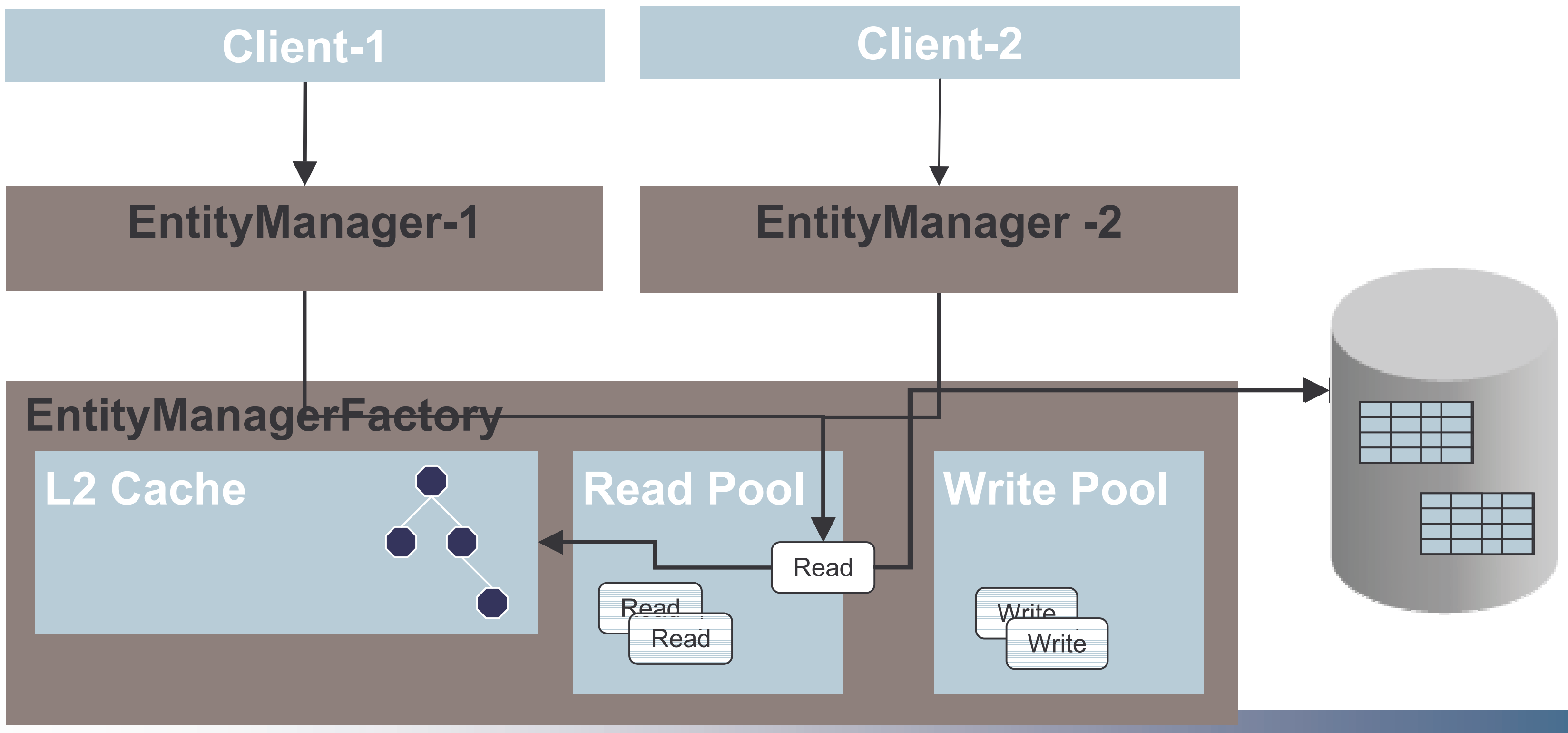
Caching

- 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

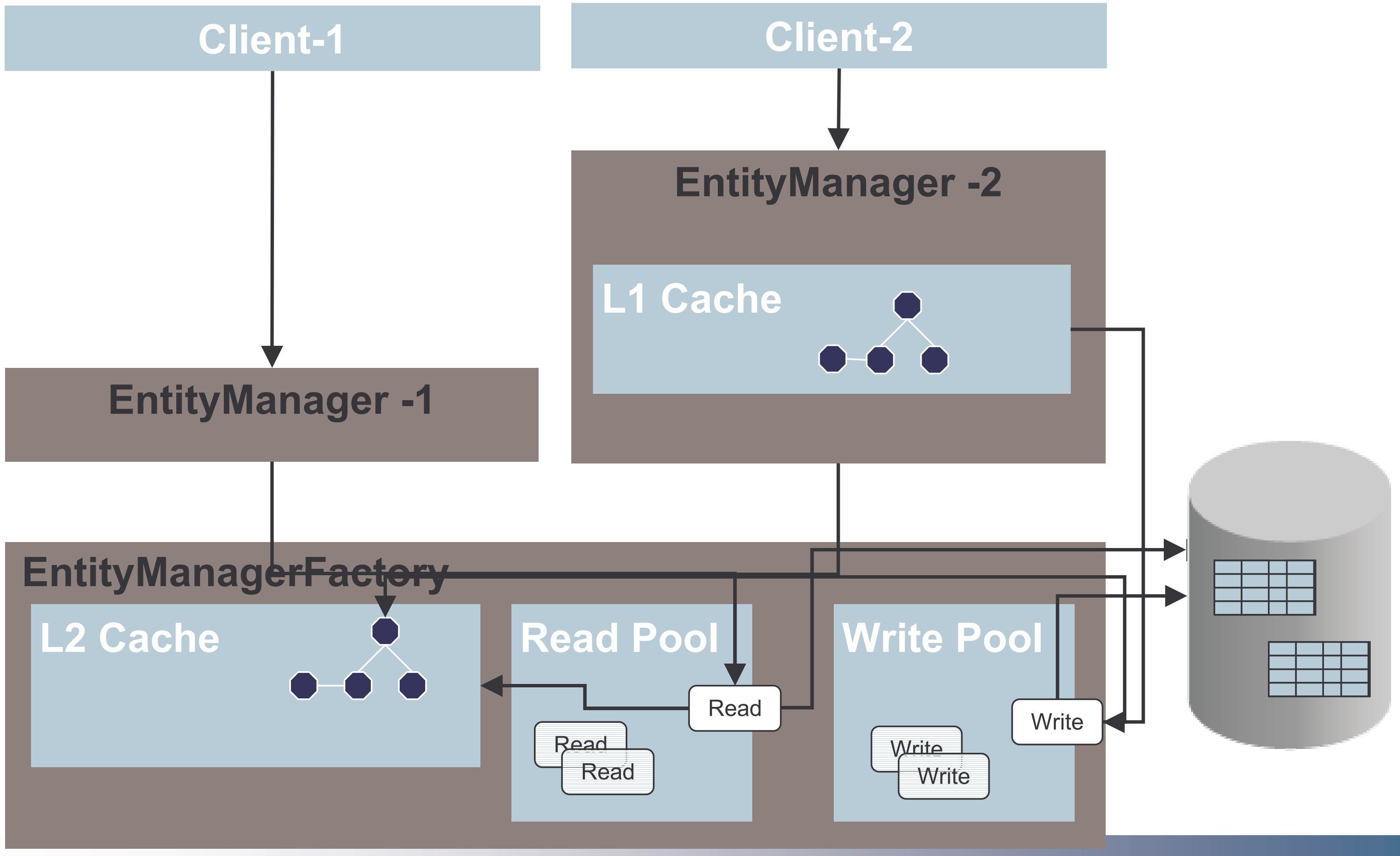
Advanced Caching

- L1 and L2 caching
  - Entity cache—not data cache
- L2 Cache Invalidation/Expiration
  - Time to live
  - Fixed Times
  - Programmable (external notification)
- L2 Cache Coordination
  - Messaging
    - JMS, RMI, IIOP, CORBA, OC4J-JGroups
  - Type specific configuration
    - Modes: Invalidate, Sync, Sync+New, None
- All configurable on a per Entity basis

L2 Cache Architecture



L1 Cache Architecture



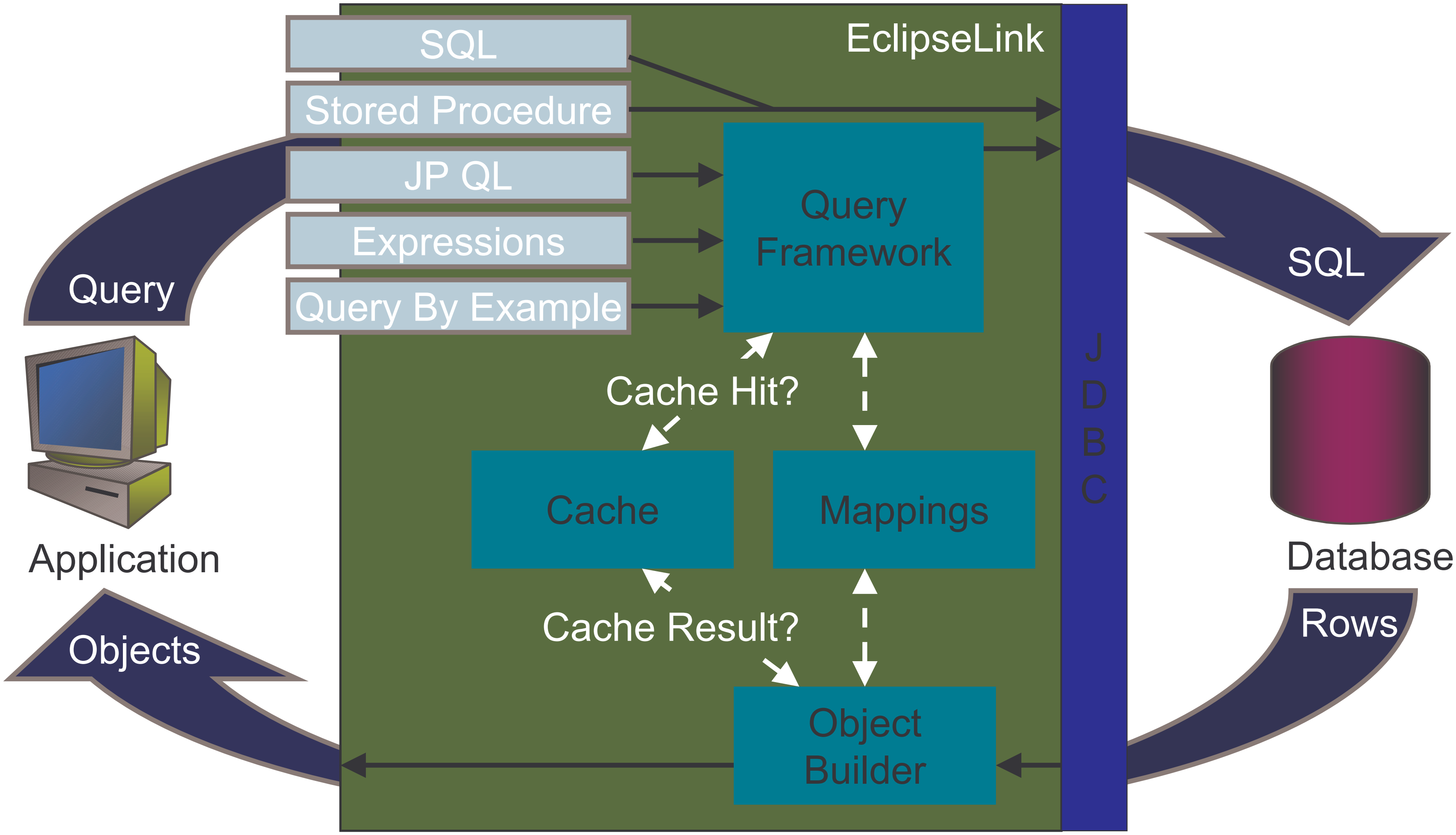
Locking

- 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
    - @OptimisticLocking
  - Pessimistic
    - Specified through query hint

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 Query Execution



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=CacheCoordinationType.SEND_OBJECT_CHANGES)
@OptimisticLocking(type=OptimisticLockingType.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

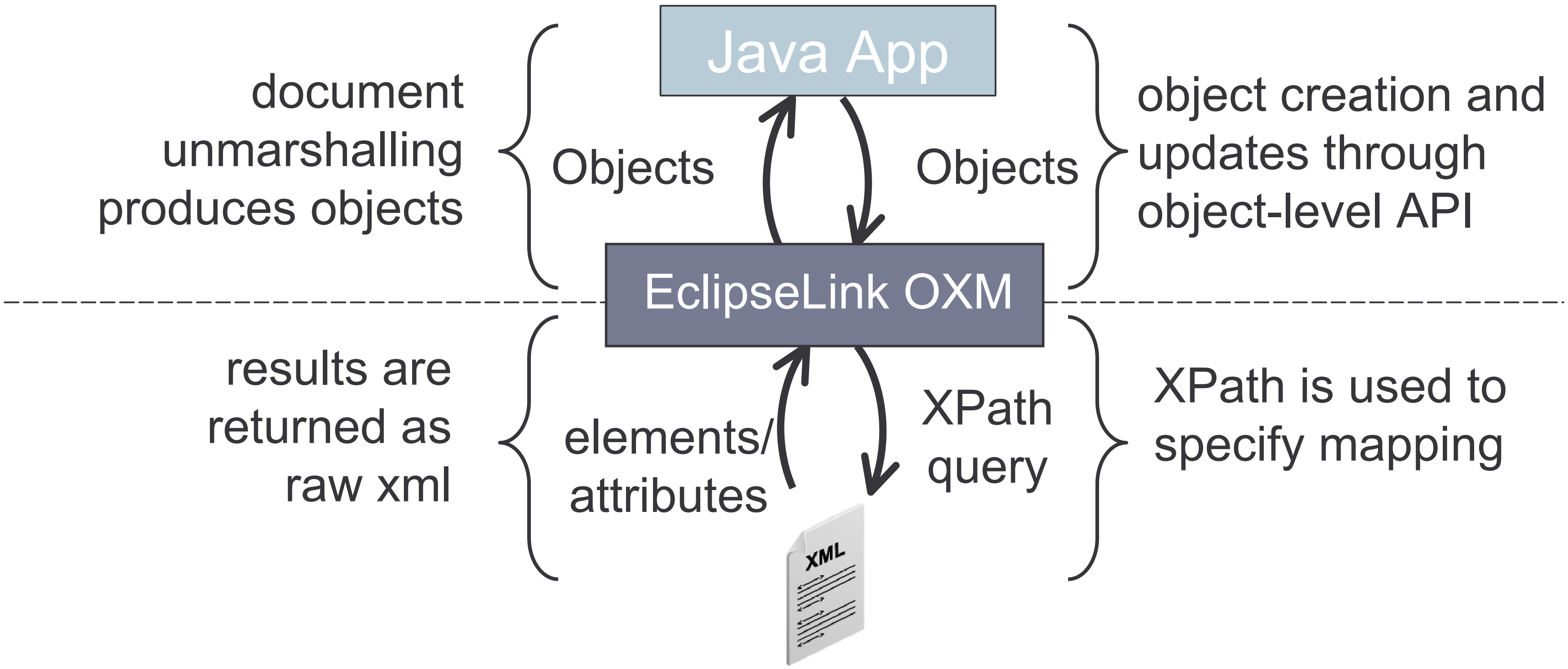
Performance and Tuning

- Highly configurable and tunable
  - Guiding principle – minimize and optimize database interactions
  - No two applications are the same—EclipseLink provides many tuning options
- Leverages underlying performance tuning features
  - Java, JDBC and the underlying database technology
  - Batch Writing
  - Parameter Binding
  - Statement Caching

EclipseLink MOXy: XML Development

- With rapid adoption of SOA and Web Services, XML has become pervasive
- XML is an ideal data exchange format, but is difficult to develop with directly
  - Requires complex, cumbersome code
  - Couples application logic to specific XML structure
  - Difficult to maintain

MOXy Binding Layer



document unmarshalling produces objects

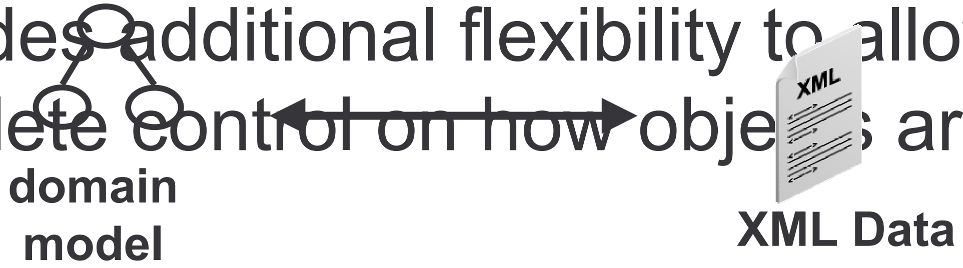
object creation and updates through object-level API

results are returned as raw xml

XPath is used to specify mapping

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: JAXB

```
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
```

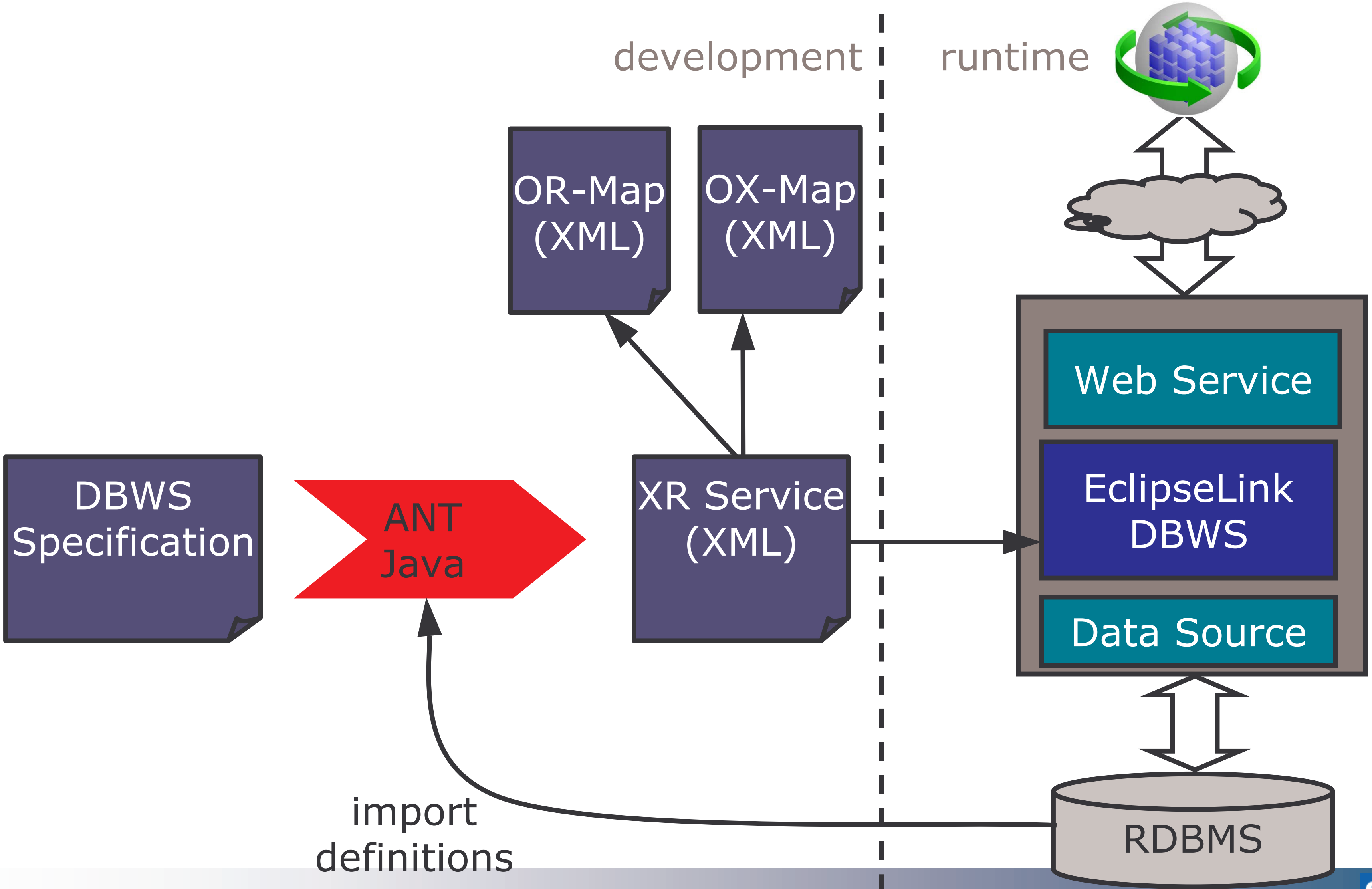
EclipseLink MOXy Benefits

- 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 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



EclipseLink SDO

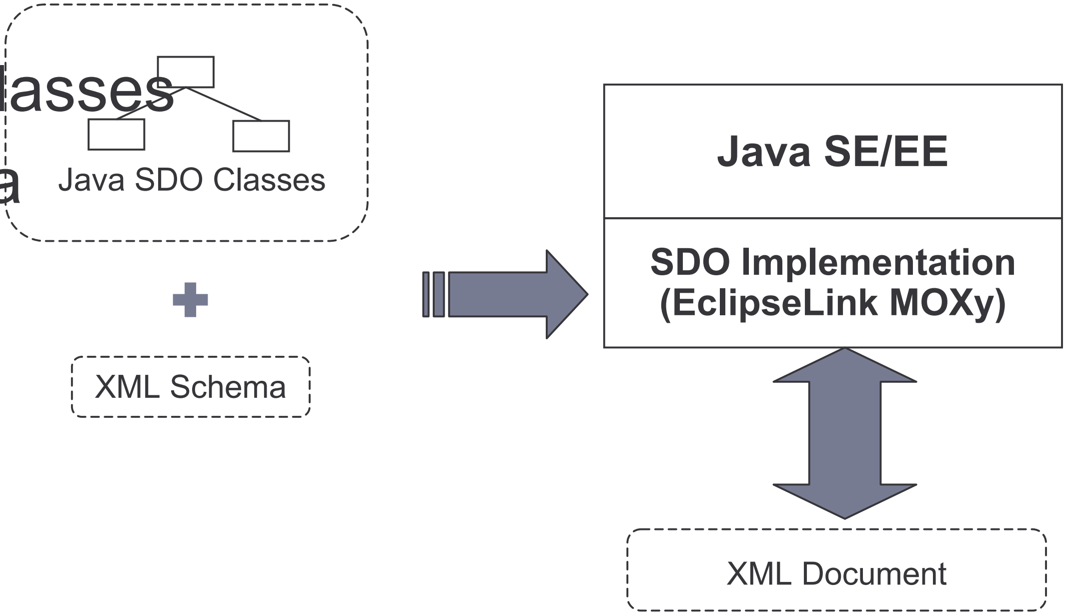
- What can you do?
  - Marshall/Unmarshall objects to/from XML
  - Define Types/Properties programmatically or derive from XSD
  - Generate JavaBean classes from XSD
  - Advanced mapping support for greater flexibility
- Why would you use it?
  - Schema/Structure unknown at compile time
  - Declarative metadata based tools/frameworks
  - XML-centric applications, need open content support
  - Dynamic content user interfaces

SDO is XSD-centric

- SDO is for applications centered around XML Schema
- “Static SDO”
  - Classes generated from XSD
  - Classes are not Pojos—they implement SDO Interfaces
- “Dynamic SDO”
  - DataObjects with types/properties derived from XSD

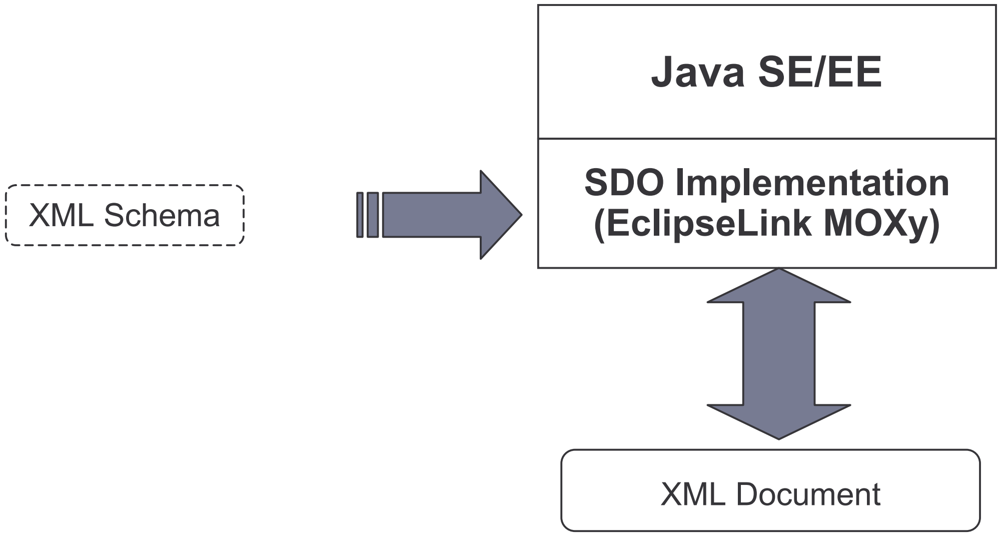
SDO Runtime—“Static SDO”

- SDO runtime combines:
  - Java SDO Classes
  - XML Schema



SDO Runtime—“Dynamic SDO”

- SDO runtime using:
  - XML Schema



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

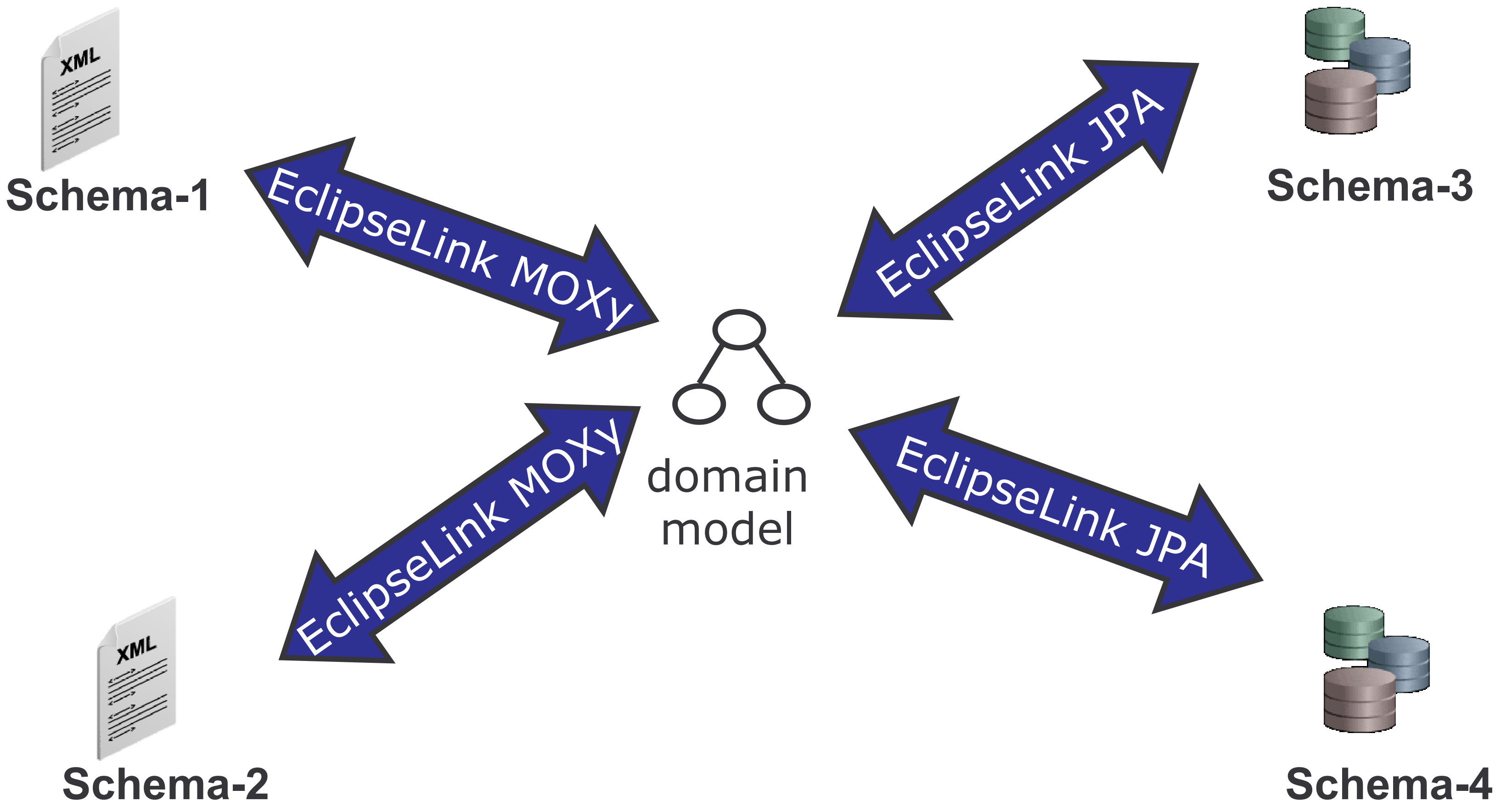
EclipseLink and OSGi

- 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



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, …

EclipseLink JPA in Spring

```
@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();
```

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 1.0, 1.0.1 and 1.0.2
- Specifications: JAXB 2.1, SDO 2.1.1
- JPA 2.0 Reference Implementation
- 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
  - 1.0.2 is now available
  - 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
  - Doug’s blog: java-persistence.blogspot.com

Q&A