# EclipseLink
## The Eclipse Persistence Services Project

**Shaun Smith**
**shaun.smith@oracle.com**



## Agenda

- What is EclipseLink?
- EclipseLink Components
  - JPA
  - MOXy
  - DBWS
  - SDO
- Future Directions

## About Me

- Shaun Smith
  - Product Manager for Oracle TopLink
    - Involved with object-relational and object-XML mapping technology for about a dozen years.
  - EclipseLink Ecosystem Development Lead
  - Eclipse Dali JPA Tools Committer
  - Eclipse Teneo Project Committer



## Eclipse Persistence Services

- Eclipse runtime project
  - Nicknamed “EclipseLink”
  - Founding Eclipse Runtime Project member
- Lead by Oracle
- Contribution of full TopLink source
- Based upon product with 12 years of commercial usage

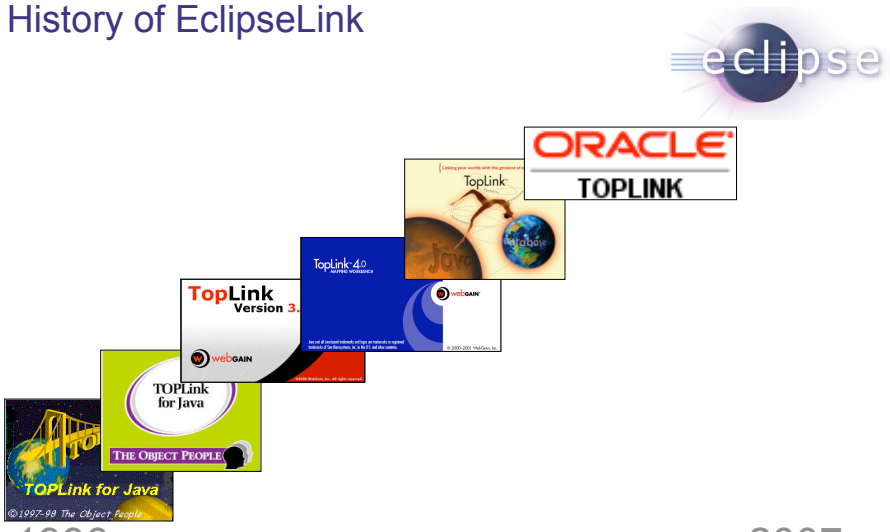
## Not Just an IDE—Runtime projects at Eclipse

| | Name | Functional area |
|---|---|---|
| Mature projects | • BIRT | • Reporting engine |
| | • Equinox | • OSGi |
| | • RAP | • AJAX platform |
| | • eRCP | • embedded RCP |
| | • ECF | • Communication |
| | • *EclipseLink* | • Object Persistence |
| Projects in incu-bation | • Swordfish | • SOA Runtime |
| | • Riena | • C/S Appl. Platform |
| Proposed | • EILF | • Enterprise Search |

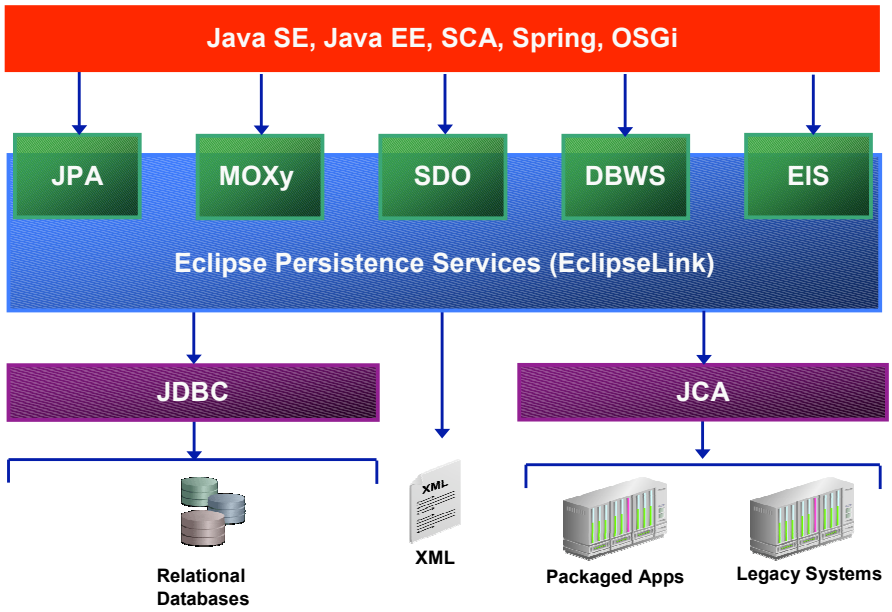
**Many more Eclipse projects provide runtimes: CDO, EMF, Higgins, Net4j, TPTP, ...**



## History of EclipseLink




1996 → 2007






## Significance of EclipseLink

- 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
- Shared infrastructure
  - Easily share the same domain model with multiple persistence technologies
  - Leverage metadata for multiple services

## EclipseLink JPA



## 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.

## 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;
    }
}
```

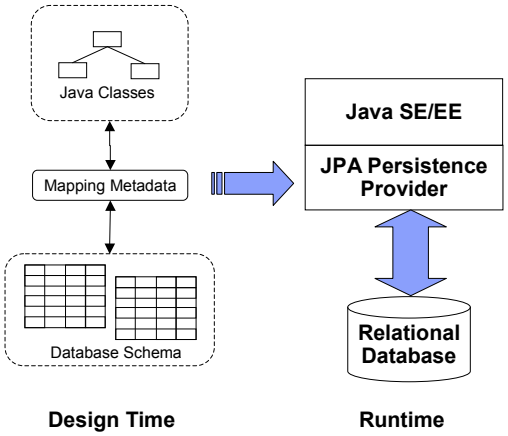
## Mappings in XML

```
<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 Design vs. Runtime

- Artifacts include:
  - Java Classes
  - Mapping Metadata
  - Database schema



## EclipseLink JPA

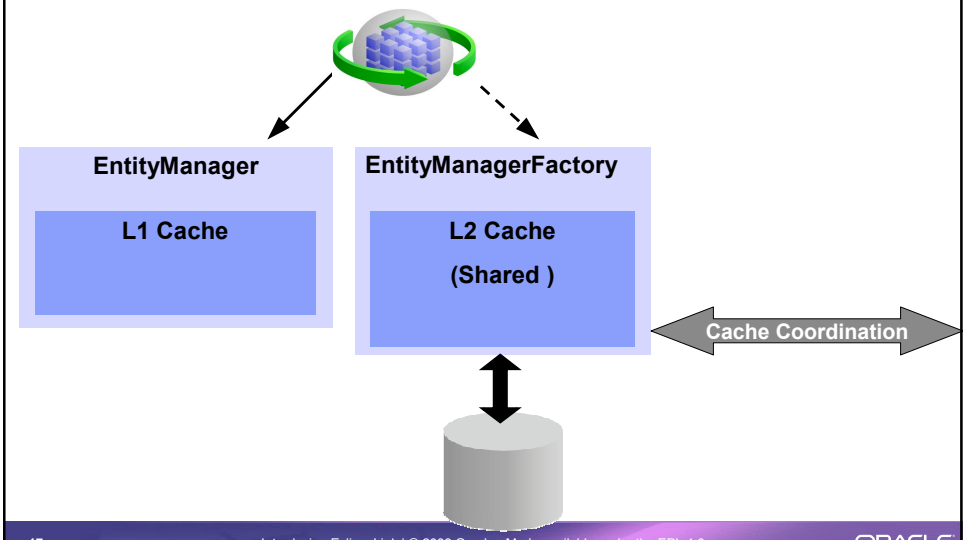
- JPA 1.0 compliant implementation
- Java EE, Java SE, Web, Spring, and OSGi
- Any JDBC/SQL compliant database
  - Advanced database extensions: stored procedures, native SQL, …
- Key infrastructure:
  - Caching, Locking, Query Framework, Mapping, …
  - JDBC connection pooling
  - Diagnostics: Logging, Profiling
  - DDL Generation
  - Customization callbacks
- Highly Extensible
- … plus many valuable advanced features



## 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

## EclipseLink JPA Caching








## Cache Configuration

- Cache Shared (L2)/Isolated (L1 only)
  - Entity cache—not data cache

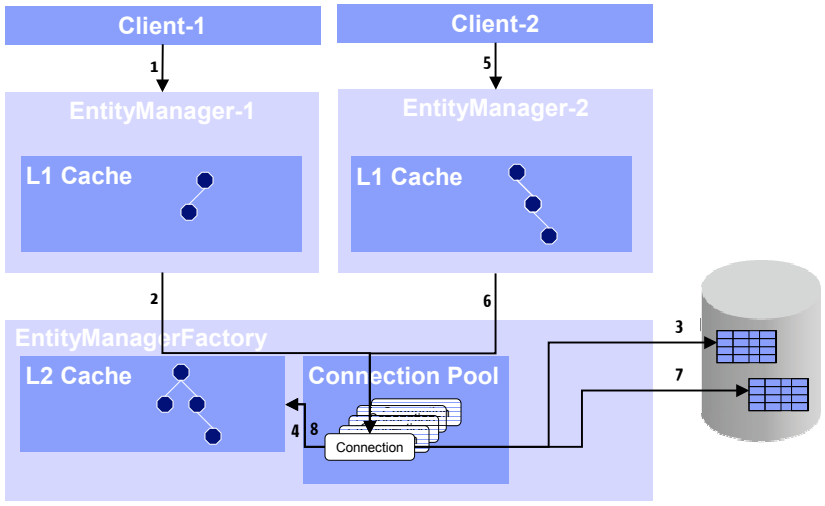
```
<property
     name="eclipselink.cache.shared.default"
     value="true"/>
```

- Cache Type & Size
  - Soft/Hard Weak
  - Weak
  - Full
  - None

```
@Cache(type = CacheType.HARD_WEAK,
       size = 500,
       shared = true,
       coordinationType =
       INVALIDATE_CHANGED_OBJECTS)
```

```
<property
     name="eclipselink.cache.type.default"
     value="Full"/>
```

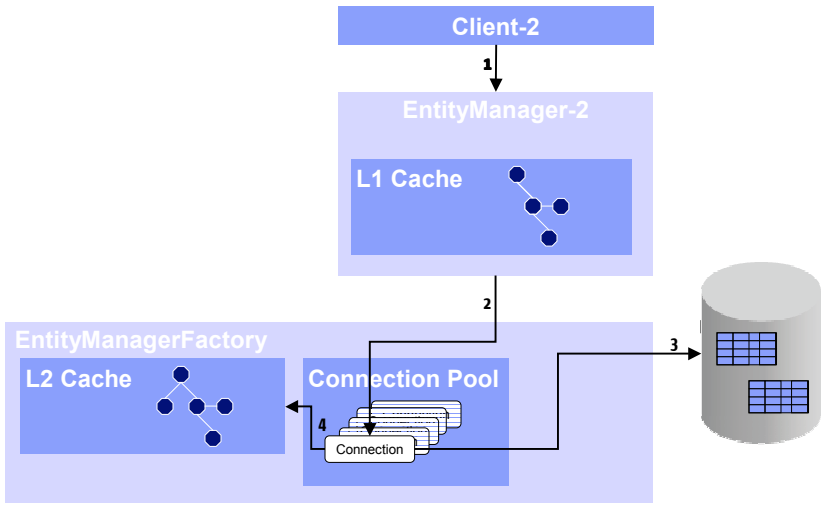
## Leveraging the Shared Cache: Reading



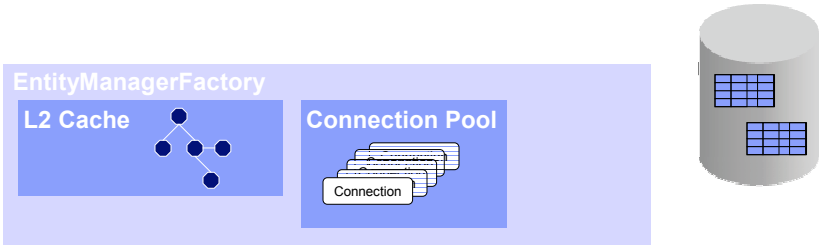





## Leveraging the Shared Cache: Writing



## Leveraging the Shared Cache: Writing






## Advanced Mappings

- @Converter
- @BasicMap and @BasicCollection
- Mapping to database structures
  - @StructConverter – Structs, ADTs
  - XML types, PL/SQL Records
- Mapping features
  - Optimistic Locking Strategies—to support legacy schemas
  - Private Owned—for orphan management
  - Join/Batch Fetch—to optimize data retrieval
  - Returning—to deal with database triggers

## Advanced Mapping Example

```
@Entity
@Cache(type=SOFT_WEAK, coordinationType=SEND_OBJECT_CHANGES)
@Converter(name="money", converterClass=MoneyConverter.class)
@OptimisticLocking(type=CHANGED_COLUMNS)
public class Employee {
  @Id
  private int id;

  private String name;


  @OneToMany(mappedBy="owner")
  private List<PhoneNumbers> phones;

  @Convert("money")
  private Money salary
  …
}
```



## Advanced Querying

- Support for supplying customizing queries
  - Native SQL
  - @NamedStoredProcQuery
- Graph Loading Optimizations
  - Join and Batch Fetch – multi-level
- Cache Usage: In-memory
- Result caching

## Transaction Features

- Minimizes database interactions
  - Only the minimal updates are sent to the database
- Respect database integrity
  - Orders INSERT, UPDATE and DELETE statements
- JTA and RESOURCE_LOCAL support
- Attribute-level change tracking
- 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
  - Batch Writing
  - Parameter Binding
  - Statement Caching

## 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!



## JPA Development with the Eclipse IDE and the Dali Java Persistence Tools

**Salvador Dalí. *The Persistence of Memory*. 1931.**

## Demo



## EclipseLink MOXy

## Java Access of XML Data

- Direct JAXP – window on data
  - Direct use of an XML parser, uses DOM nodes and/or SAX/StAX events directly.
- Entity Beans/Business Objects
  - Accessed as objects or components (EJBs), transparent that the data is stored in XML
  - Need binding layer in middle tier to handle the object-XML mapping and conversion



## Challenge: XML Development

### Objective—obtain employee number

- JAXP

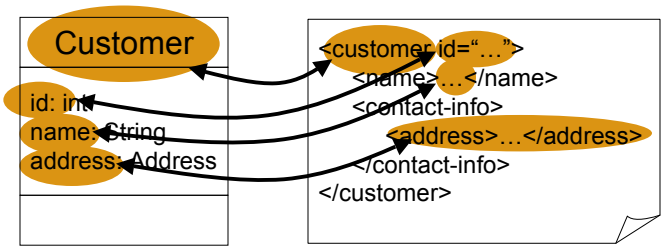
```
Node childNode = employeeElement.getFirstChild();
while(childNode != null) {
  if(childNode.getNodeName().equals("employee-number")) {
      Node employeeNumberTextNode = childNode().getFirstChild();
      employeeNumber = new
      Integer(employeeNumberTextNode.getNodeValue()).intValue();
  }
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

- Using XML binding

```
employee.getEmployeeNumber();
```

## Data Binding/Mapping

- The activity of ‘Mapping’ is the process of connecting objects/attributes to XML types/nodes.






## Data Binding Approaches

- Code Generation
- Declarative
  - Annotate Java Classes
  - Externalized Mapping Metadata

## EclipseLink MOXy

"**M**apping **O**bjects to **X**ML"

- Allows developers to work with XML as objects
- Efficiently produce and consume XML
- Provides support for different Object/XML mapping technologies:
  - Java Architecture for XML Binding (JAXB) 2.1
  - EclipseLink OXM / JAXB 1.0
  - Foundation of EclipseLink SDO and DBWS



## About Java Architecture for XML Binding (JAXB)

- JAXB 2 part of Java EE 5 specification
- Included in Java 6 SDK
- Suitable for use in different environments
  - Java SE environment
  - Java EE Container
  - OSGi
  - Spring

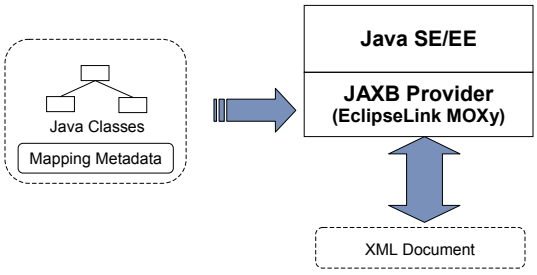
## JAXB 2—in a Nutshell

- A Java standard that defines:
  - how Java objects are converted to/from XML (specified using a standard set of mappings)
  - a programmer API for reading and writing Java objects to/from XML documents
  - a service provider interface (SPI) to allow for selection of JAXB implementation



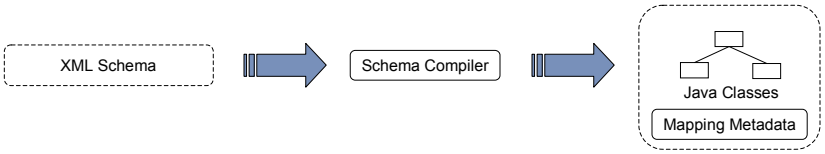
## JAXB 2 Runtime

- JAXB runtime combines:
  - Java Classes
  - Mapping Metadata



## JAXB 2 Design Time—Starting from XML Schema

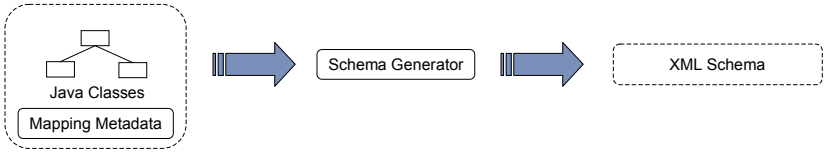
- JAXB Schema Compiler:
  - XML schema input
  - Generates Java Pojo Classes
  - Generates Mapping Annotations






## JAXB 2 Design Time—Starting From Classes

- JAXB Schema Generator:
  - (Annotated) Java Classes input
  - Generates XML Schema
  - Useful for inclusion in WSDL



## Demo



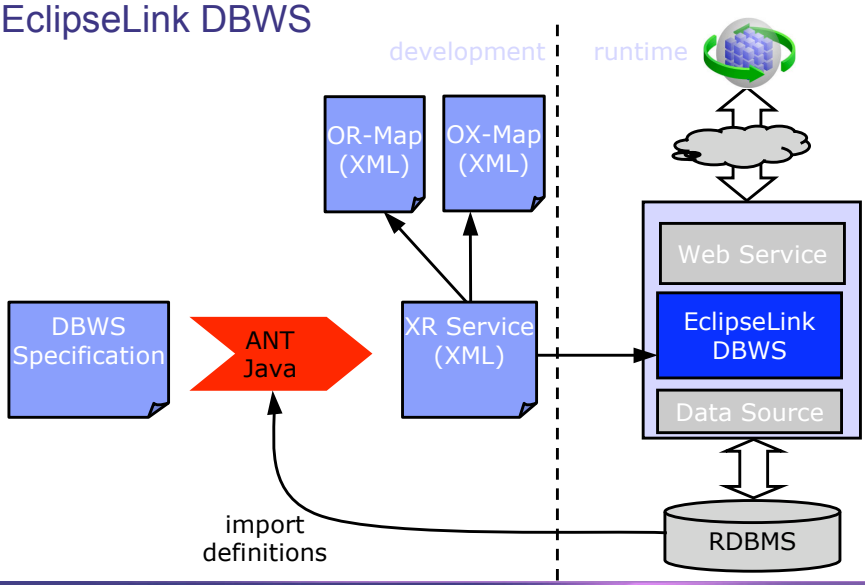
## EclipseLink DBWS

## 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



## Combining Services



## Combining Persistence 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

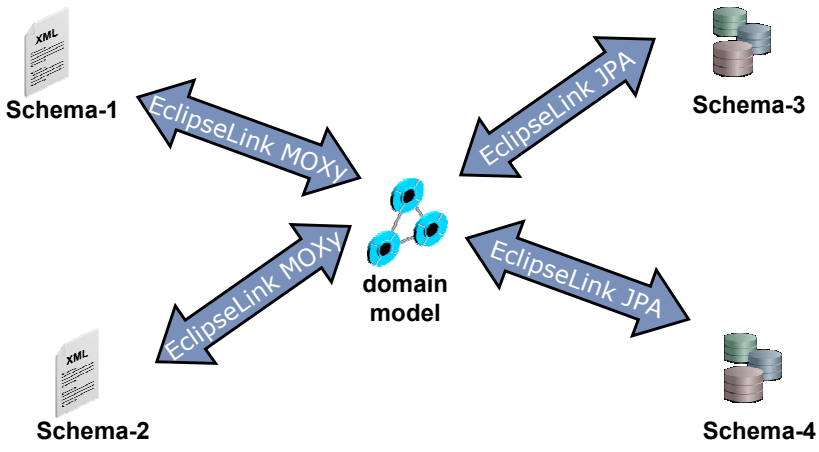
## JAXB 2.0 & JPA 1.0

### Combining JAXB 2.0 and JPA 1.0 Annotations

```
@XmlRootElement
@Entity
public class Customer {

    @XmlAttribute(name="id")
    @Id
    public int getId() {…}
    public void setId(int id) {…}

    @XmlElement(name="billing-address")
    @OneToOne
    @JoinColumn(name="ADDR_ID")
    public Address getBillingAddress() {…}
    public void setBillingAddress(Address address) {…}

}
```



## Leveraging Common Domain Model



## Demo



## Service Data Objects (SDO)

## What is SDO?

"Service Data Objects (SDO) is a data programming architecture and an API."

"The main purpose of SDO is to simplify data programming, so that developers can focus on business logic instead of the underlying technology."

—SDO 2.1 Specification



## 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 Design Time—Starting from XML Schema

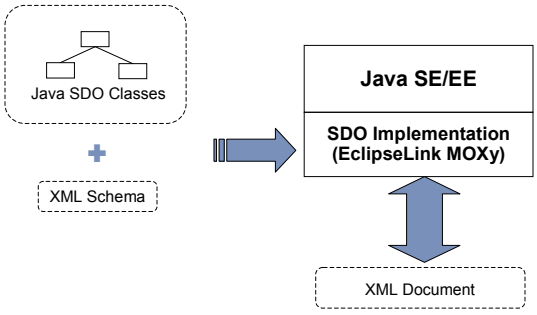
- SDO Schema Compiler:
  - XML schema input
  - Generates SDO Classes





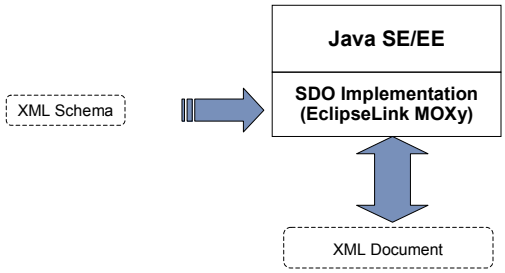

## SDO Runtime—"Static SDO"

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



## SDO Runtime—"Dynamic SDO"

- SDO runtime using:
  - XML Schema






ORACLE

eclipse

## Demo

## EclipseLink Status

- Contribution of Oracle TopLink complete
- Full documentation available on Wiki
- Milestones included in GlassFish V3 Milestones
- 1.0 Release available
  - JPA 1.0, SDO 2.1, JAXB
  - Simplified XML and annotation config of advanced features
  - Packaged for Java SE/EE and OSGi bundles
- Beyond 1.0
  - JPA 2.0 (Reference Implementation)
  - Database Web Services
  - Data Access Service (DAS) 1.0
  - and much more …



## How can you get involved?

- Users
  - The 1.0 Release
  - Try it out and provide feedback
  - File bug reports and feature requests
- Contributors (Tmax Soft, Individuals)
  - Contribute to roadmap discussions
  - Bug fixes
- Committers
  - Now have committers from Sun

## More Information

- www.eclipse.org/eclipselink
- Newsgroup: eclipse.rt.eclipselink
- Wiki: wiki.eclipse.org/EclipseLink
- Mailing Lists:
  - eclipselink-dev@eclipse.org
  - eclipselink-users@eclipse.org
- Blogs
  - Committer Team blog: eclipselink.blogspot.com
  - Shaun's blog: onpersistence.blogspot.com



## 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!