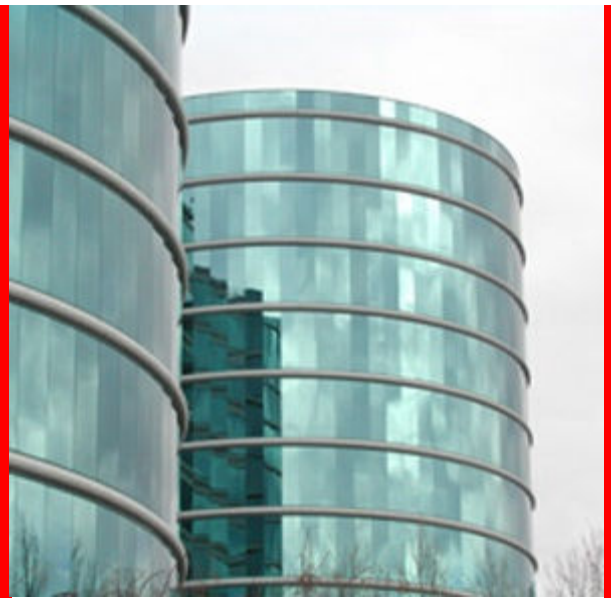


ORACLE

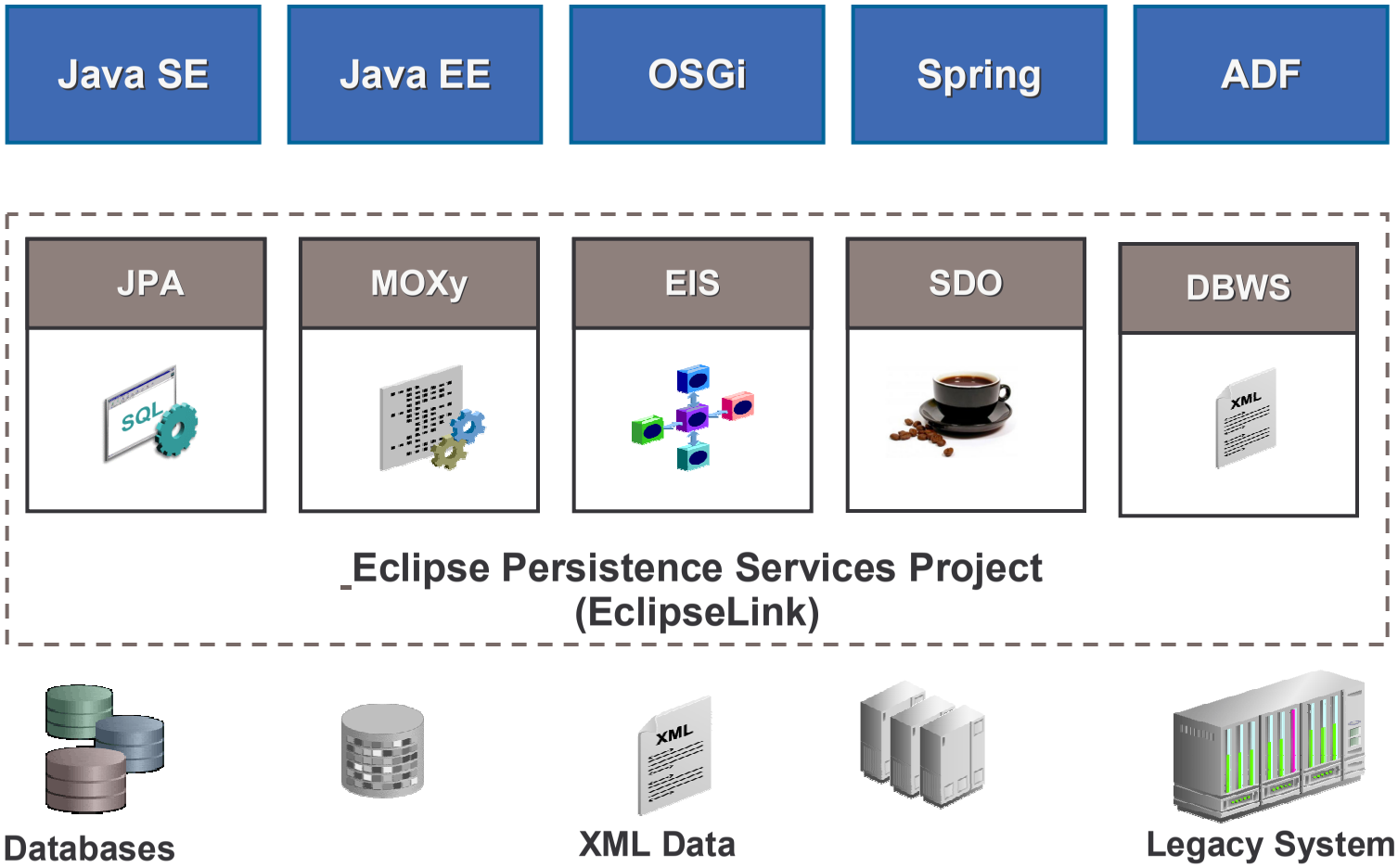
JAVAFORUM 2008 stuttgart am 3. Juli 2008

EclipseLink JPA

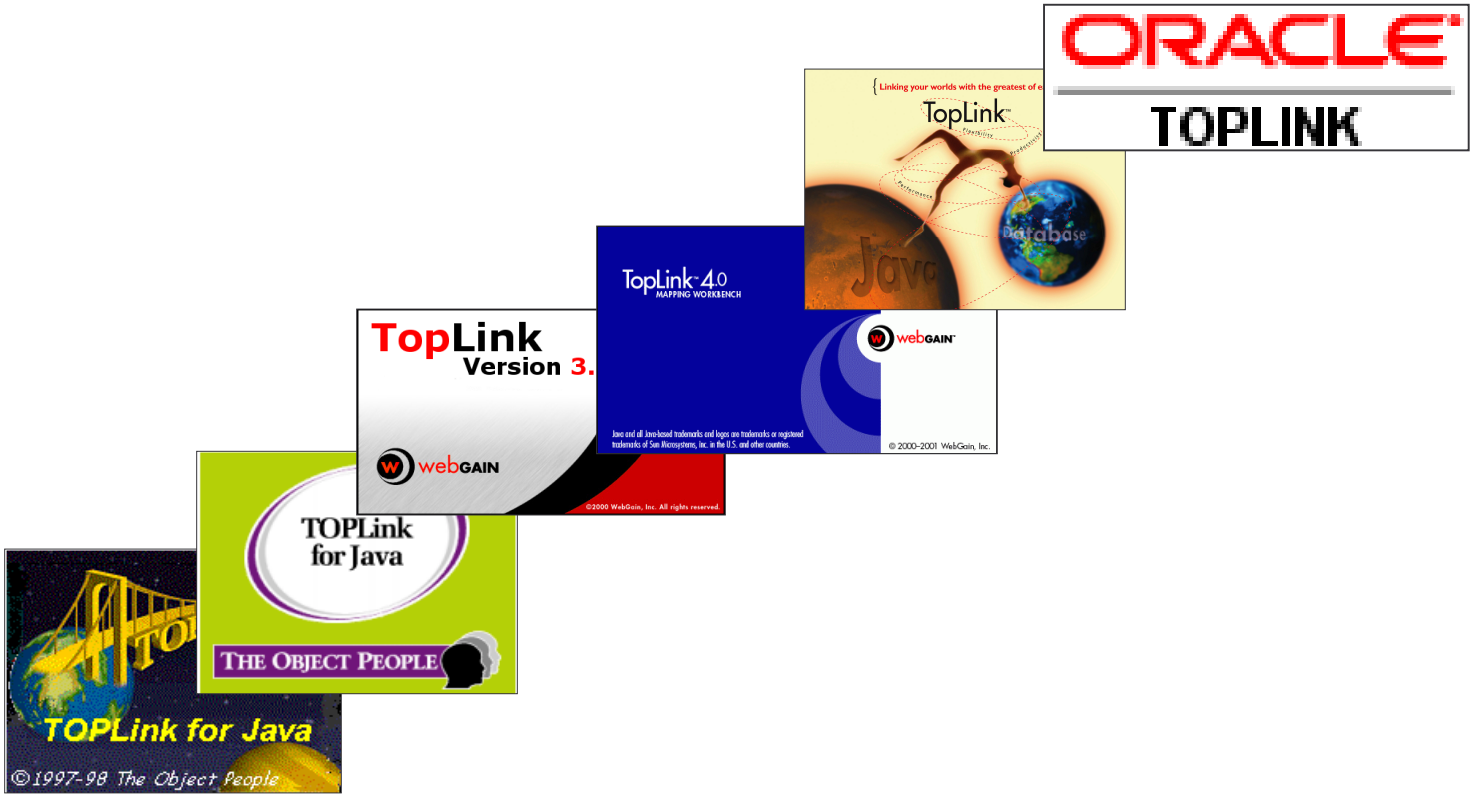
Doug Clarke, Director of Product Mangement

eclipse

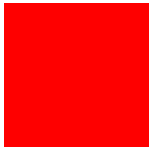
Eclipse Persistence Services Project – *"EclipseLink"*



History of EclipseLink



1996 → 2008

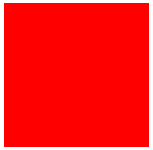


Significance

- 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

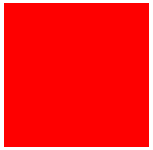
EclipseLink Road Map

- Delivery of monthly incubation milestones
  - Build and testing processes
  - Initial contribution functional
  - 1.0M10 was the last milestone
- 1.0 Release: July 9th, 2008
  - Specifications: JPA 1.0, JAXB, SDO 2.1
  - OSGi packaging and usage examples
  - Spring Framework support
- Future
  - JPA 2.0 – Reference Implementation
  - JAXB 2.1 Compliance
  - Database Web Services (DBWS)
  - SDO Data Access Service (DAS)



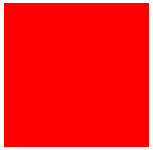
Java Persistence API (JPA)

- Separate document bundled as part of EJB 3.0 specification (JSR 220)
- 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



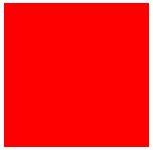
JPA—in a Nutshell

- 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
- 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 (become “detached”)



Object-Relational Mappings

- Core JPA Mappings
  - Id
  - Basic
  - Relationships
    - OneToOne
    - OneToMany/ManyToOne
    - ManyToMany
  - And more…
- Can be specified using Annotations or XML

JPA Mappings 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;
    }
}
```

JPA Mappings 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;
    }
}
```

JPA Entity—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>
```

Advanced EclipseLink JPA

- General
  - JDBC connection pooling
  - Diagnostics: Logging, Profiling
  - DDL Generation
  - Database and Server Platforms
  - Customization callbacks
- Mappings
- Caching
- Query

JDBC Connection

```
<properties>
...
<property name="eclipselink.jdbc.driver"
  value="oracle.jdbc.Driver"/>

<property name="eclipselink.jdbc.url"
  value="jdbc:oracle:thin:@localhost:1521:XE"/>

<property name="eclipselink.jdbc.user" value="scott"/>

<property name="eclipselink.jdbc.password"
  value="tiger"/>
```

Logging

```
<properties>
   ...

   <property
       name="eclipselink.logging.level"
       value="FINE"/>
   ...

</properties>
```

DDL Generation

- Mapping metadata specifies how DDL should be generated
- EclipseLink provides
  - Generating DDL to a file only
  - Generating and executing DDL in DB
  - Dropping existing tables before creating new ones

DDL Generation

```
<properties>
   ...
  <property
      name="eclipselink.ddl-generation"
       value="create-tables"/>
   ...
</properties>
```

Database Platform

XML:

```
<properties>
<property
      name="eclipselink.target-database"
      value="Derby"/>
</properties>
```

API:

```
  properties.put(
   EclipseLinkProperties.TARGET_DATABASE,
   TargetDatabase.ORACLE);
```

Target Database Platform

TargetDatabase (org.eclipse.persistence.jpa.config)

- Auto (Default)
- Oracle, Oracle8i, Oracle9i, Oracle10g, Oracle11, TimesTen
- DB2, DB2Mainframe
- Derby, JavaDB, MySQL
- Informix
- HSQL, PointBase
- PostgreSQL
- SQLAnyWhere
- SQLServer, DBase
- Sybase

Server Platform

- Enables simplified configuration of the target application server
- Used to enable integration with:
  - JTA transaction manager
  - Logging
  - JDBC connection un-wrapping

```
<property name="eclipselink.target-server"
  value="SunAS9"/>
```

Target Server Options

`TargetServer` (org.eclipse.persistence.jpa.config)

- None (Default)
- OC4J, OC4J_10_1_3, OC4J_11_1_1
- SunAS9
- WebSphere
- WebSphere_6_1
- WebLogic, WebLogic_9, WebLogic_10
- JBoss

Session Customizers

```
<property
    name="eclipselink.session.customizer"
    value="acme.MySessionCustomizer"/>
```

```
public class MySessionCustomizer
                implements SessionCustomizer {

  public void customize(Session session) {
    session.setProfiler(new PerformanceProfiler());
   }

}
```

Descriptor Customizers

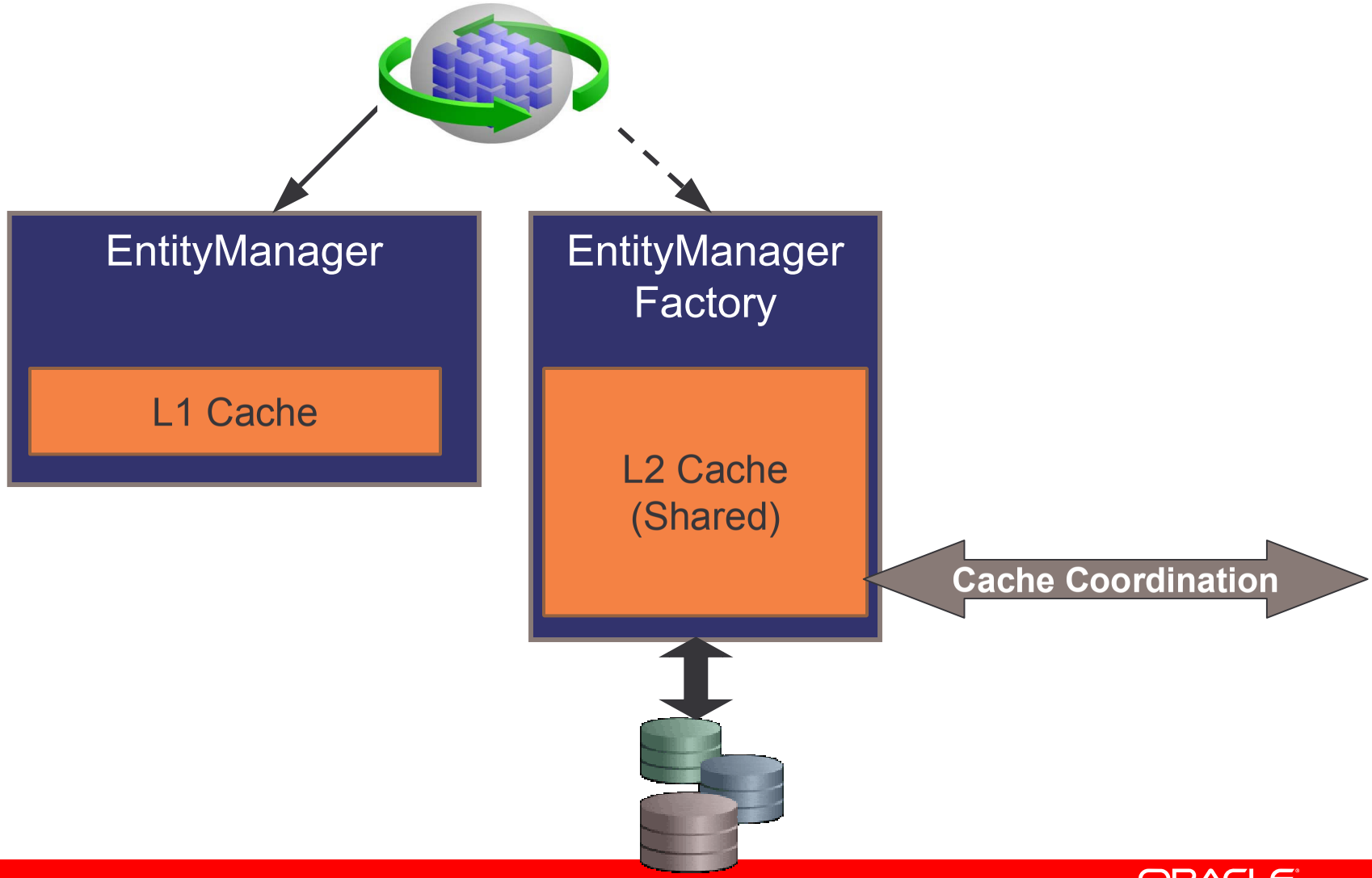
```
<property
    name="eclipselink.descriptor.customizer.Employee"
    value="acme.MyDescriptorCustomizer"/>
```

```
public class MyDescriptorCustomizer
                implements DescriptorCustomizer {

  public void customize(ClassDescriptor desc) {
    desc.disableCacheHits();
   }

}
```

Concurrency Protection - Locking

- Optimistic concurrency included in JPA 1.0, but no support for pessimistic locking is specified
- EclipseLink has advanced optimistic locking strategies
  - Numeric, Timestamp, All fields, Selected fields, Changed field
- EntityManager lock() method can be used with optimistic locking, and error handling
- EclipseLink supports pessimistic locking through query hints
  - `query.setHint(PESSIMISTIC_LOCK, LockNoWait);`

EclipseLink JPA Caching



Cache Configuration

- Shared

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

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

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

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

Minimize stale cache

1. Configure the cache relative to application data's usage
   - Is the data shared between users/processes?
   - Is the data volatile?
     - Only through JPA application?
     - Through direct DB modification?
2. Ensure you protect any data that can be concurrently modified with a locking strategy
   - Must handle optimistic lock failures on flush/commit
3. Use query refreshing to minimize optimistic lock failures

Additional Mappings

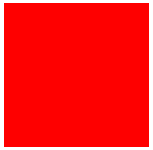
- @Converter
- @BasicMap and @BasicCollection
- Mapping to database structures
  - @StructConverter – Structs, ADTs
  - XML types, PL/SQL Records
- Mapping features
  - Private Owned
  - Join/Batch Fetch
  - Returning

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
  …
}
```

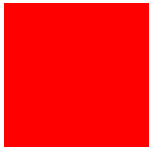
Advanced Querying

- Support for supplying custom SQL or stored procedure for any query
  - @NamedStoredProcedureQuery
- Graph Loading Optimizations
  - Join and Batch Fetch – multi-level
- Cache Usage: In-memory
- Result caching
- Fetch Groups



Weaving

- EclipseLink makes use of Weaving (ASM) to introduce additional functionality into the entity classes
  - Enables:
    - M:1 and 1:1 lazy fetching
    - Optimized Change Tracking
    - Fetch Groups
  - Integrated with EJB 3.0 and Spring
  - Available for Java SE using JDK/JRE's –javaagent:
  - Optional
  - Static weaving also supported
    - Weaving of .class files before deployment



JPA Configuration Options

- Annotations
  - JPA
  - EclipseLink
- ORM.XML
  - JPA
  - EclipseLink
  - JPA + EclipseLink
- Defaults – Configuration by Exception

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

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 Java Persistence Tooling Project
  - Teneo using EclipseLink for EMF model persistence
  - MayInstall for storage of deployment configuration
  - Eclipse RT
    - Swordfish Project (SOA) usage of EclipseLink SDO
    - …

EclipseLink Status

- RT Project
  - Initial contribution of Oracle TopLink complete
  - Full documentation available on Wiki
  - Producing Monthly Milestone builds
  - **Graduated from Incubation June 27th**
- **1.0 release planned for July 2008**
  - JPA 1.0, SDO 2.1, JAXB 2.1
  - 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 …

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
  - Shaun's blog: onpersistence.blogspot.com