jax 09
Konferenz für Java™, Enterprise Architekturen, SOA

OSGi Expert Day: OSGi Persistence

Doug Clarke
Director of Product Management, Oracle Corp.
EclipseLink Project co-Lead

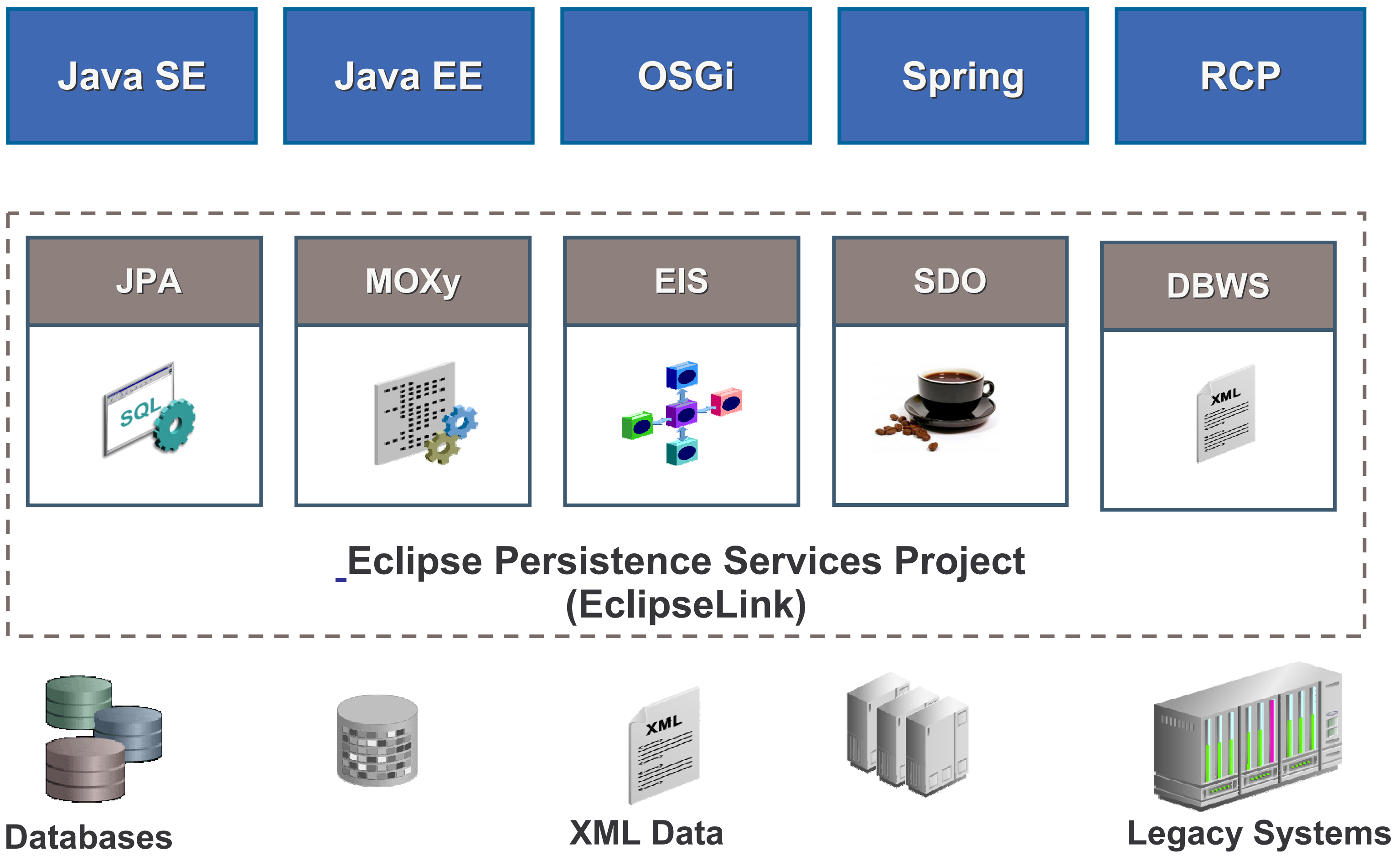
History of EclipseLink

TOPLink for Java
©1997-98 The Object People
TOPLink for Java
THE OBJECT PEOPLE
TopLink Version 3.
webGAIN
TopLink 4.0 MAPPING WORKBENCH
© 2000-2001 WebGain, Inc.
{Linking your worlds with the greatest of ease}
TopLink
Java
Database

TopLink Essentials

1996 → 2008

EclipseLink Project



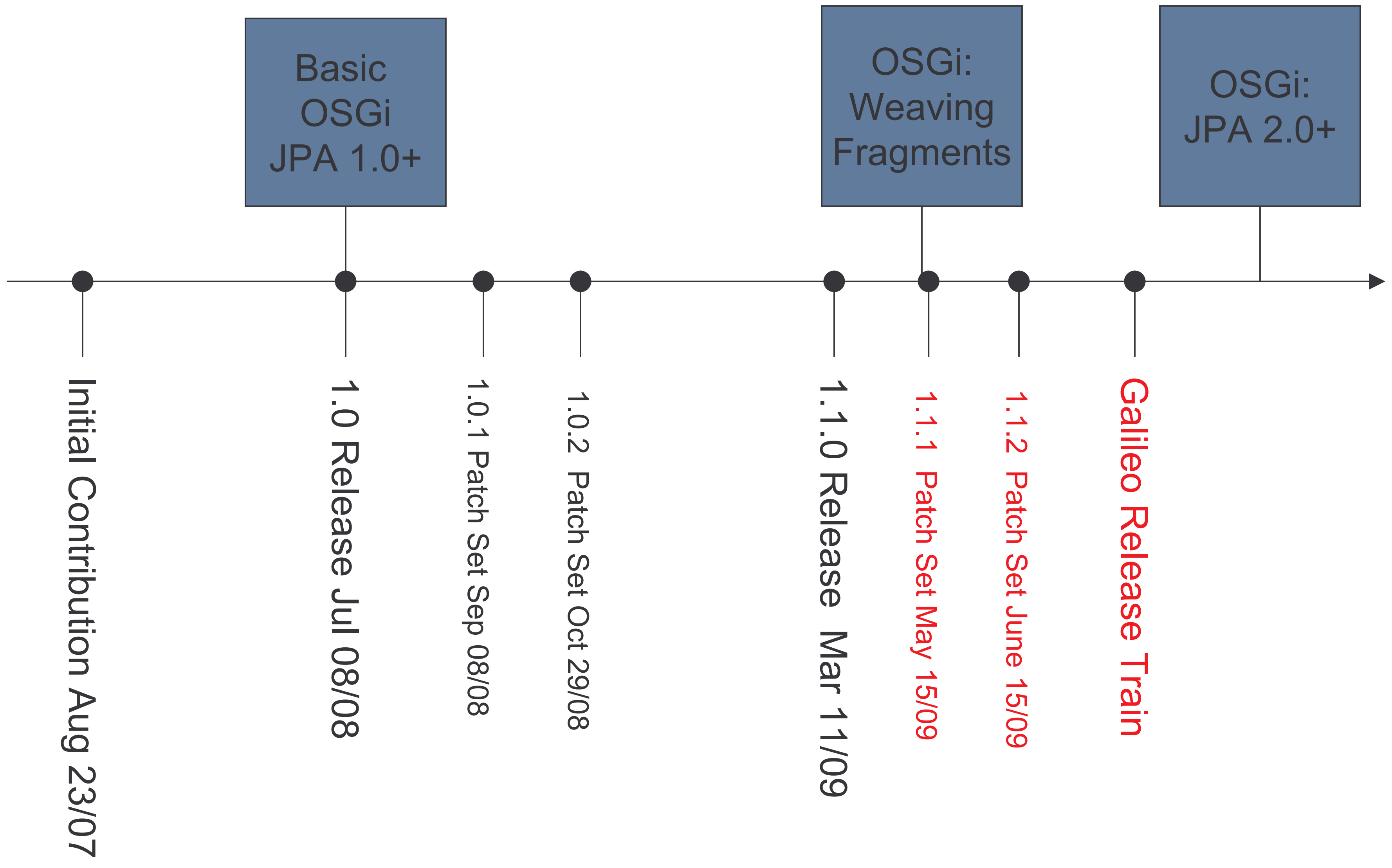
EclipseLink and OSGi – The Plan

- 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

OSGi Enabled Challenges

- EclipseLink Bundles
  - Bundle per component
  - Need to allow users flexibility per persistence service
- Metadata access
  - XML and Annotations in calling bundle(s)
- Resource lookup
  - JDBC, Parsers
  - User provided policy implementations

EclipseLink Road Map & OSGi Support


Basic
OSGi
JPA 1.0+
OSGi:
Weaving
Fragments
OSGi:
JPA 2.0+
Initial Contribution Aug 23/07
1.0 Release Jul 08/08
1.0.1 Patch Set Sep 08/08
1.0.2 Patch Set Oct 29/08
1.1.0 Release Mar 11/09
1.1.1 Patch Set May 15/09
1.1.2 Patch Set June 15/09
Galileo Release Train

EclipseLink JPA & OSGi

- JPA 1.0
  - Need to access metadata from calling bundle
  - Need to access JDBC dynamically
  - Extensions needed to JPA for Provider flexibility
    - Service based provider solution
    - Direct usage supported as well
  - Weaving requires Equinox hooks
- JPA 2.0
  - Spec formalizes ‘resolver’ extensions

JPA 1.0 Provider Lookup

- javax.persistence.Persistence
  - Application Bootstrap API

```
public static EntityManagerFactory createEntityManagerFactory(
             String persistenceUnitName, Map properties)
{
      …
       findAllProviders();
…

private static void findAllProviders() throws IOException
{
        ClassLoader loader =
Thread.currentThread().getContextClassLoader();
        Enumeration<URL> resources =
            loader.getResources("META-INF/services/" +
             PersistenceProvider.class.getName());
```

JPA Bootstrapping in OSGi

- Vendor specific bootstrapping

```
import org.eclipse.persistence.jpa.;
import javax.persistence.*;

…

EntityManagerFactory emf =

        PersistenceProvider.createEntityManagerFactory(“pu-name”);
```

- Standard JPA Application Bootstrapping

```
import javax.persistence.*;

…

EntityManagerFactory emf =

                Persistence.createEntityManagerFactory(“pu-name”);
```

  – Requires vendor registry using services …

JPA 2.0 Provider Lookup

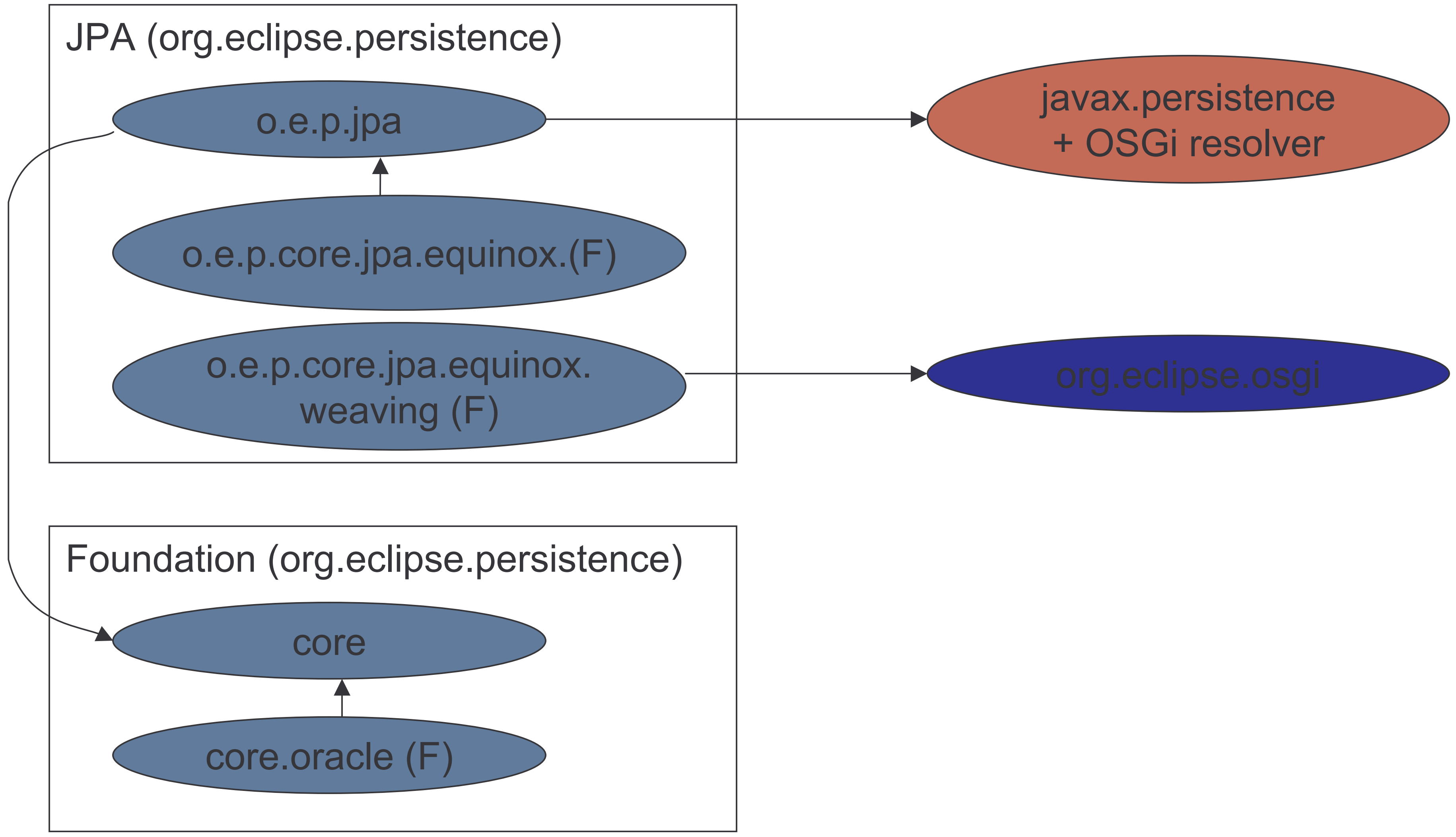
```
public static EntityManagerFactory createEntityManagerFactory(
             String persistenceUnitName, Map properties)
{
    EntityManagerFactory emf = null;
    PersistenceProviderResolver resolver =
       PersistenceProviderResolverHolder.
             getPersistenceProviderResolver();

    List<PersistenceProvider> providers =
                    resolver.getPersistenceProviders();
```

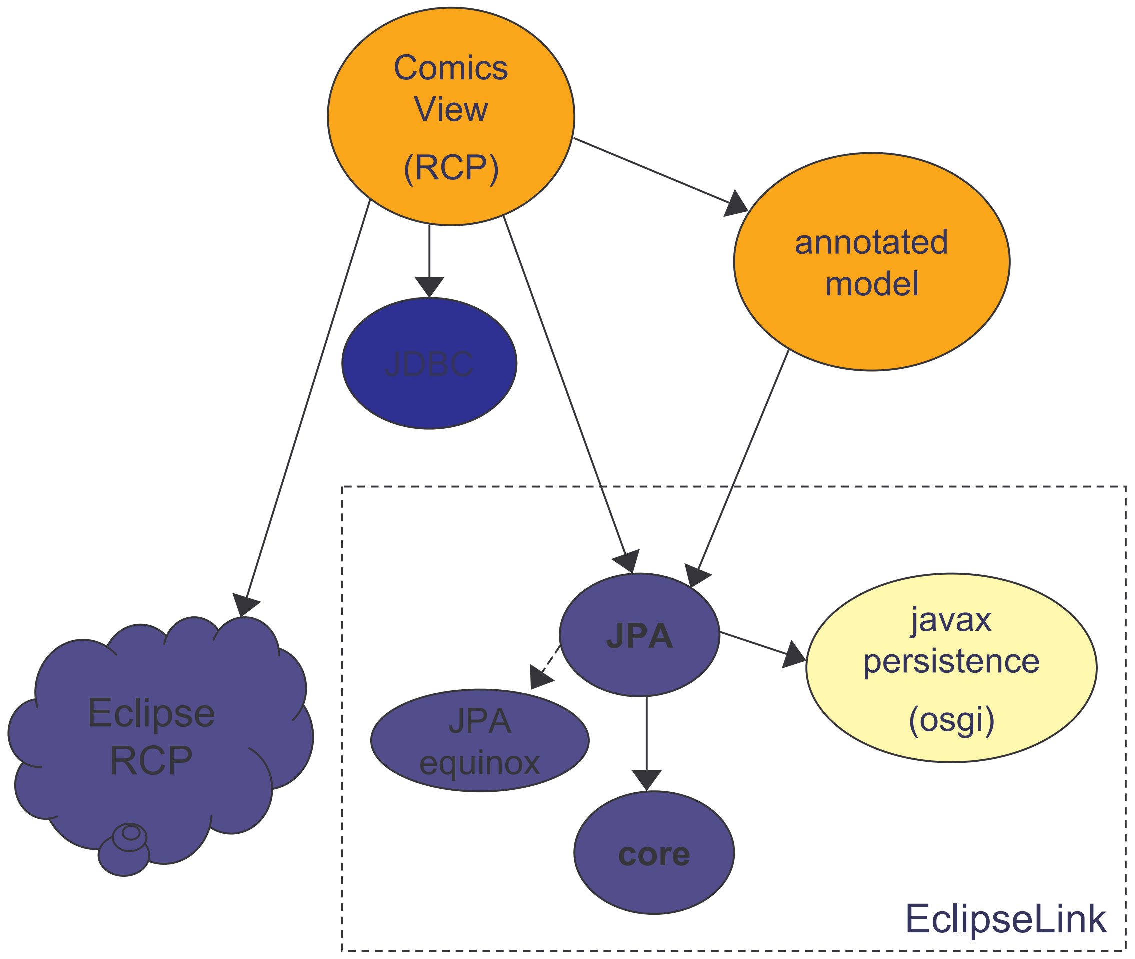
EclipseLink Weaving Support

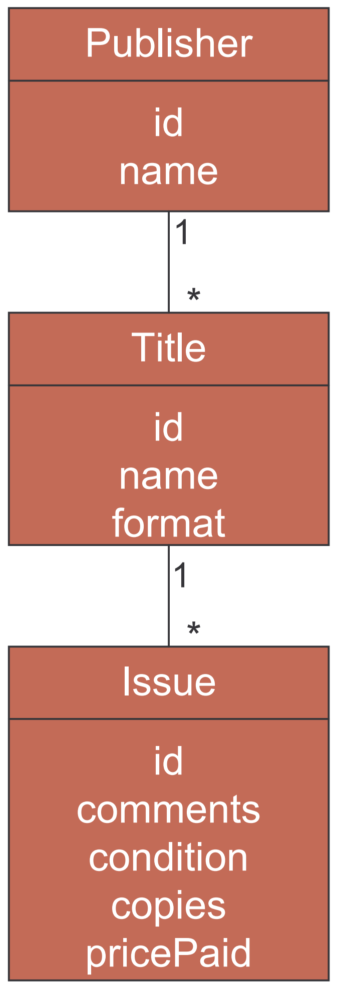
- Uses (ASM) to introduce additional functionality into the ‘POJO’ domain classes
- Used for
  - M:1 and 1:1 lazy fetching
  - Fetch Groups
  - Change Tracking
  - State Caching
- Integrated with EJB3 and Spring 2.0
- Available for Java SE platform using JDK/JRE –javaagent:
- Use is Optional (used by default when possible)
- Static weaving also supported
  - Weaving of .class files before deployment

Components, Bundles & Fragments



DEMO: Simple RCP using EclipseLink





Future Challenges & Goals

- Investigate further bundle splitting
  - As required by usage scenarios
- Address weaving enhancements
  - Optimize Equinox hook implementation
  - Address more usage configurations
    - Classes separate from XML config files
- Usability: Documentation and Examples
- Automated Testing
- Standardize
  - Provider registration
  - Weaving solution across OSGi implementation