

# Using Spring and Toplink

A brief introduction

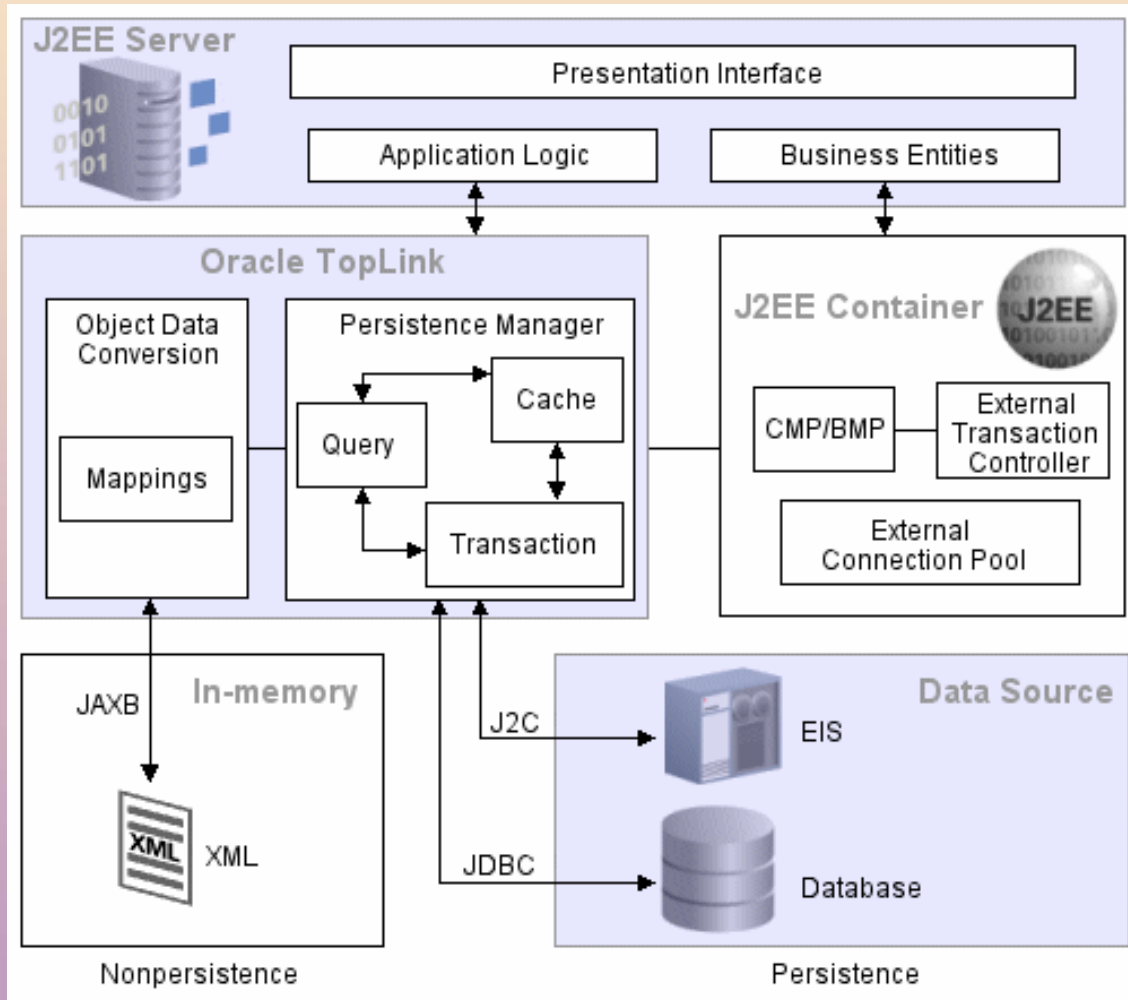


# Agenda

- Toplink
- Spring
- DAO pattern
- What's next?



# TopLink

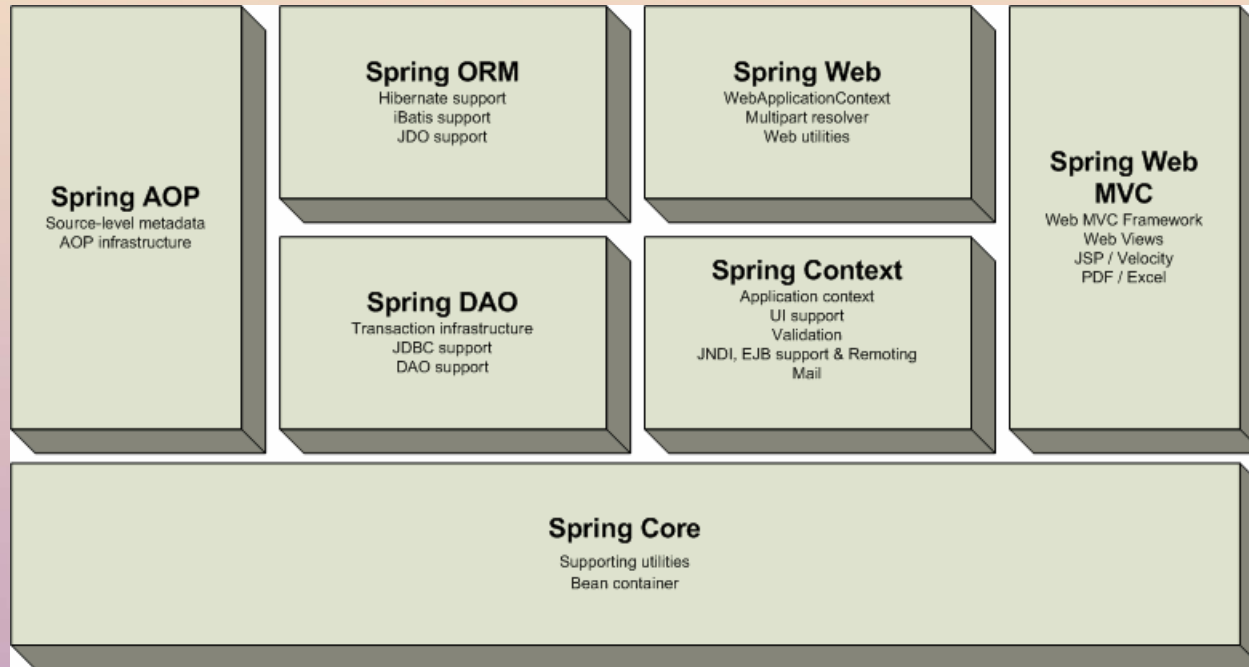


# TopLink

- O-R and O-X mapper
- More features than 'standard EJB 3.0'
- Caching, session management
- Tool support
- Proven technology -> first java version 1997
- Future direction: JPA



# Spring



# Spring

- A lightweight container
- Handles typical J2EE plumbing
- Incorporates frameworks
- Easy to test
- Current version 1.2.8
- Future: Spring 2.0



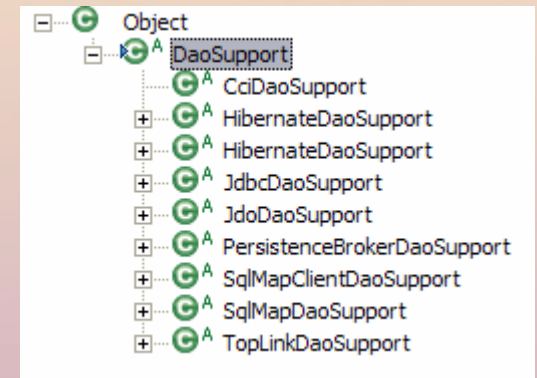
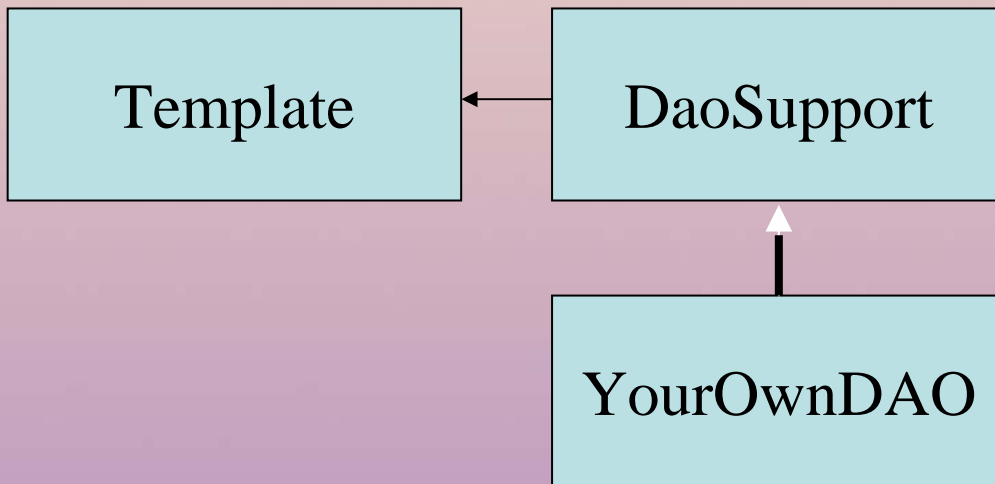
# Spring ORM and Spring DAO

- Common interface
- Connection through DataSource
- Common Exception Hierarchy
- DB vendor specific error messages
- Template pattern



# DAO pattern

- Transaction support with AOP
- Wrapped in support class



# TopLinkDAO

- Takes care of plumbing
- Familiar concepts
  - TopLink developers
  - Spring developers
  - Hibernate developers
- Some features are still missing



# TopLink 'regular'

```
public SRAdminFacadeBean() {  
    this.sessionFactory =  
        new SessionFactory("META-INF/sessions.xml", "SRDemo");  
}  
  
public List<ExpertiseArea> findExpertiseByUserId(Integer userIdParam) {  
    List<ExpertiseArea> result = null;  
  
    if (userIdParam != null){  
        Session session = getSessionFactory().acquireSession();  
        Vector params = new Vector(1);  
        params.add(userIdParam);  
        result =  
(List<ExpertiseArea>)session.executeQuery("findExpertiseByUserId",  
ExpertiseArea.class, params);  
        session.release();  
    }  
  
    return result;  
}
```

# TopLink 'regular'

```
public Product createProduct(Integer prodId, String name, String
    image,
                                String description) {
    UnitOfWork uow = getSessionFactory().acquireUnitOfWork();
    Product newInstance =
(Product)uow.newInstance(Product.class);
    newInstance.setProdId(prodId);
    newInstance.setName(name);
    newInstance.setImage(image);
    newInstance.setDescription(description);
uow.commit();

    return newInstance;
}
```

# TopLink DAO

```
public List<Team> findAllTeams(Long competitionId) {  
    return (List<Team>)getTopLinkTemplate().executeNamedQuery(Team.class,  
        "findTeamsByCompetition", new Object[] { competitionId });  
}  
  
public void saveTeam(Team team) {  
    getTopLinkTemplate().shallowMerge(team);  
}
```

- read, executeQuery, copy, refresh, register, merge, delete



# What's next

- Java Persistence API (JPA) leverages all implementations
- Tool support
  - Eclipse support for EJB 3.0 (Dali)
  - JDeveloper10.1.3
  - Spring 2.0 will support JPA
  - Spring support in JDeveloper
- Application server support



# JpaDaoSupport

- JpaDaoSupport class
- JpaTemplate class
- Optionally expose native entity manager
- Find, persist, merge, getReference, contains, remove, flush, refresh
- OpenEntityManagerInViewFilter/Interceptor



**THIS IS THE CURRENT *DEVELOPMENT* RELEASE**



# Conclusion

Go out and try it

