Philadelphia Spring Users Group
October 2005

New Persistence Features for Spring 1.3

Thomas Risberg

Introduction

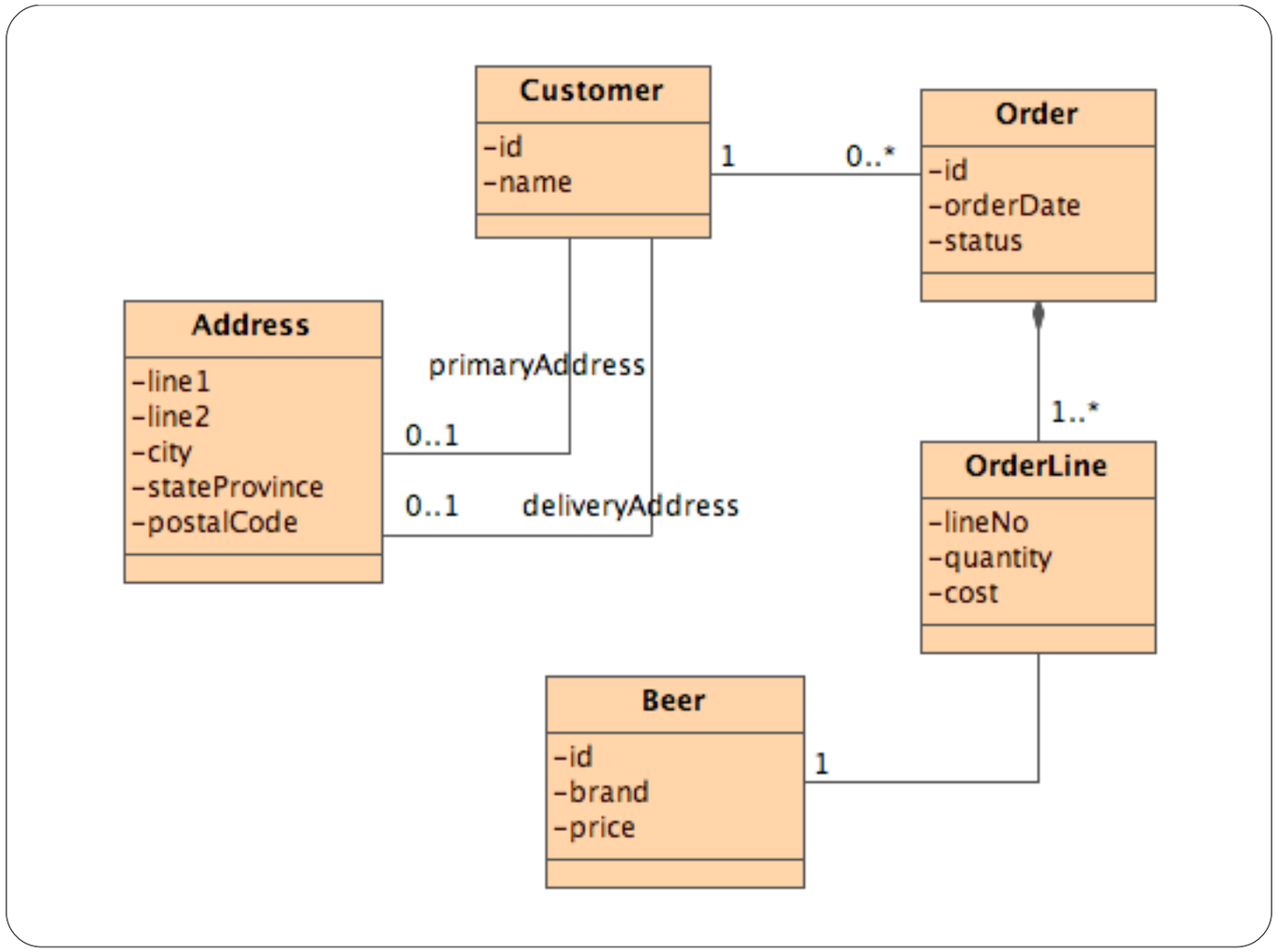
Thomas Risberg

- Independent Consultant, springdeveloper.com
- Committer on the Spring Framework and Spring Modules projects
- Supporting the JDBC and DataAccess code
- Co-author of "Professional Java Development with the Spring Framework" from Wrox

Spring Persistence Options

- **Spring JDBC**
- iBATIS
- JDO
- TopLink
- Hibernate
- **EJB3 Persistence / Java Persistence API**

Data Model
used in all examples



Spring JDBC
New Features

Just compare this SQL statement:

```
select id, price, brand from beers
 where price < ? and brand <> ?
```

with the following

```
select id, price, brand from beers
 where price < :maxPrice and brand <> :unwantedBrand
```

Spring JDBC New Features

Named Parameters.

```
public List getPreferredBeer() {

    Map params = new HashMap(2);
    params.put("unwantedBrand", "Heineken");
    params.put("maxPrice", new BigDecimal(25.00));

    List l = getJdbcTemplate().queryForList(
      "select id, price, brand from beers " +
      "where price < :maxPrice and brand <> :unwantedBrand",
      params);

    return l;
}
```

Spring JDBC New Features

```
Map params = new HashMap(1);
List idList = new ArrayList();
idList.add(new Long(1));
idList.add(new Long(3));
idList.add(new Long(5));
params.put("idList", idList);

List l = getJdbcTemplate().queryForList(
        "select id, price, brand from beers " +
        "where id in(:idList)",
        params);
```

Spring JDBC New Features

```
Map params = new HashMap(1);
params.put("id", new Long(2));
Map data = getJdbcTemplate().queryForMap(
        "select id, price, brand from beers " +
        "where id = :id",
        params);

data.put("price", new BigDecimal(26.00));

getJdbcTemplate().update(
        "update beers set brand = :brand, price = :price " +
        "where id = :id",
        data);
```

Spring JDBC New Features

```
Map params = new HashMap(1);
params.put("id", id);
Beer beer = (Beer)getJdbcTemplate().queryForObject(
        "select id, price, brand from beers " +
        "where id = :id",
        params,
        new RowMapper() {
            public Object mapRow(ResultSet rs, int rowCount)
                    throws SQLException {
                Beer b = new Beer();
                b.setId(rs.getLong("id"));
                b.setBrand(rs.getString("brand"));
                b.setPrice(rs.getBigDecimal("price"));
                return b;
            }
        });

beer.setPrice(new BigDecimal(28.50));

getJdbcTemplate().update(
        "update beers set brand = :brand, price = :price " +
        "where id = :id",
        new SqlBeanWrapper(beer));
```

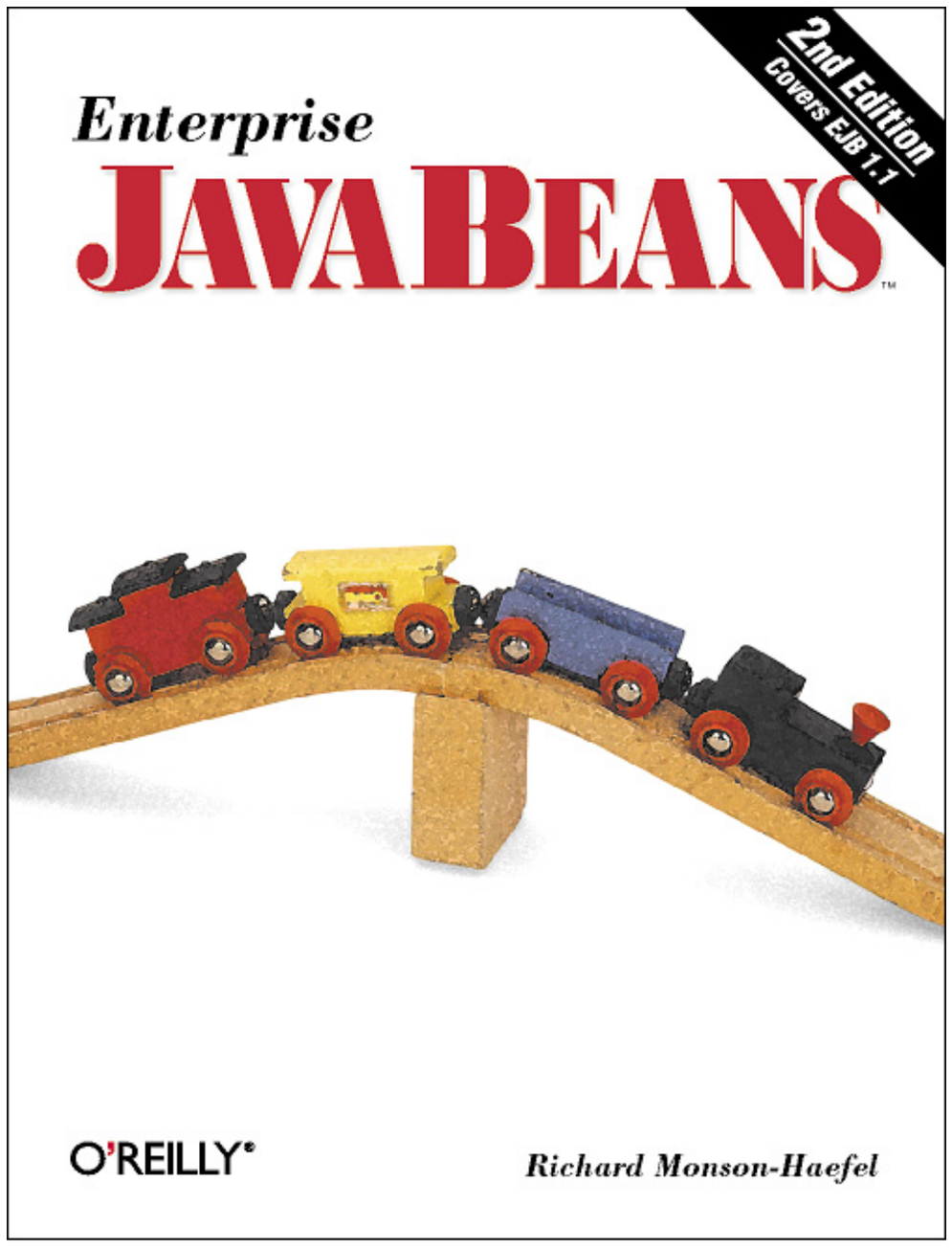
Spring ORM
New Features

- **Java Persistence API**

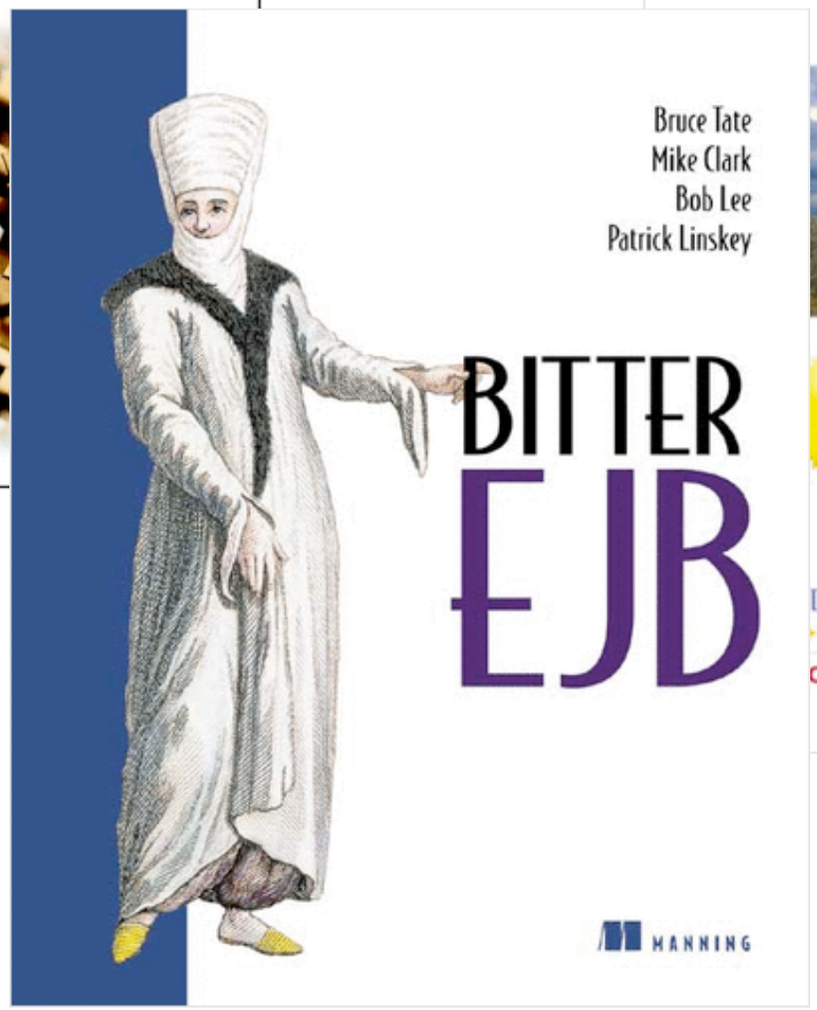
  Also known as EJB 3 persistence

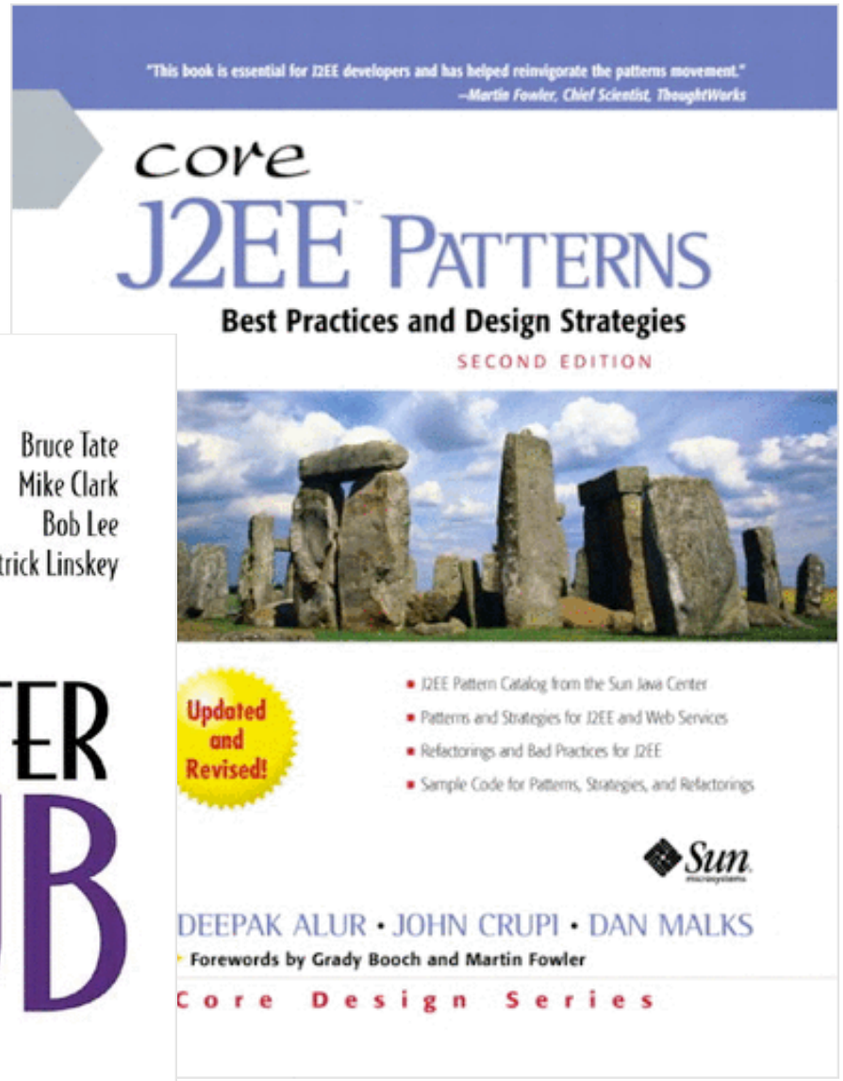
  Part of:
  JSR 220: Enterprise JavaBeans, Version 3.0

EJB EntityBeans
the beginning . . .

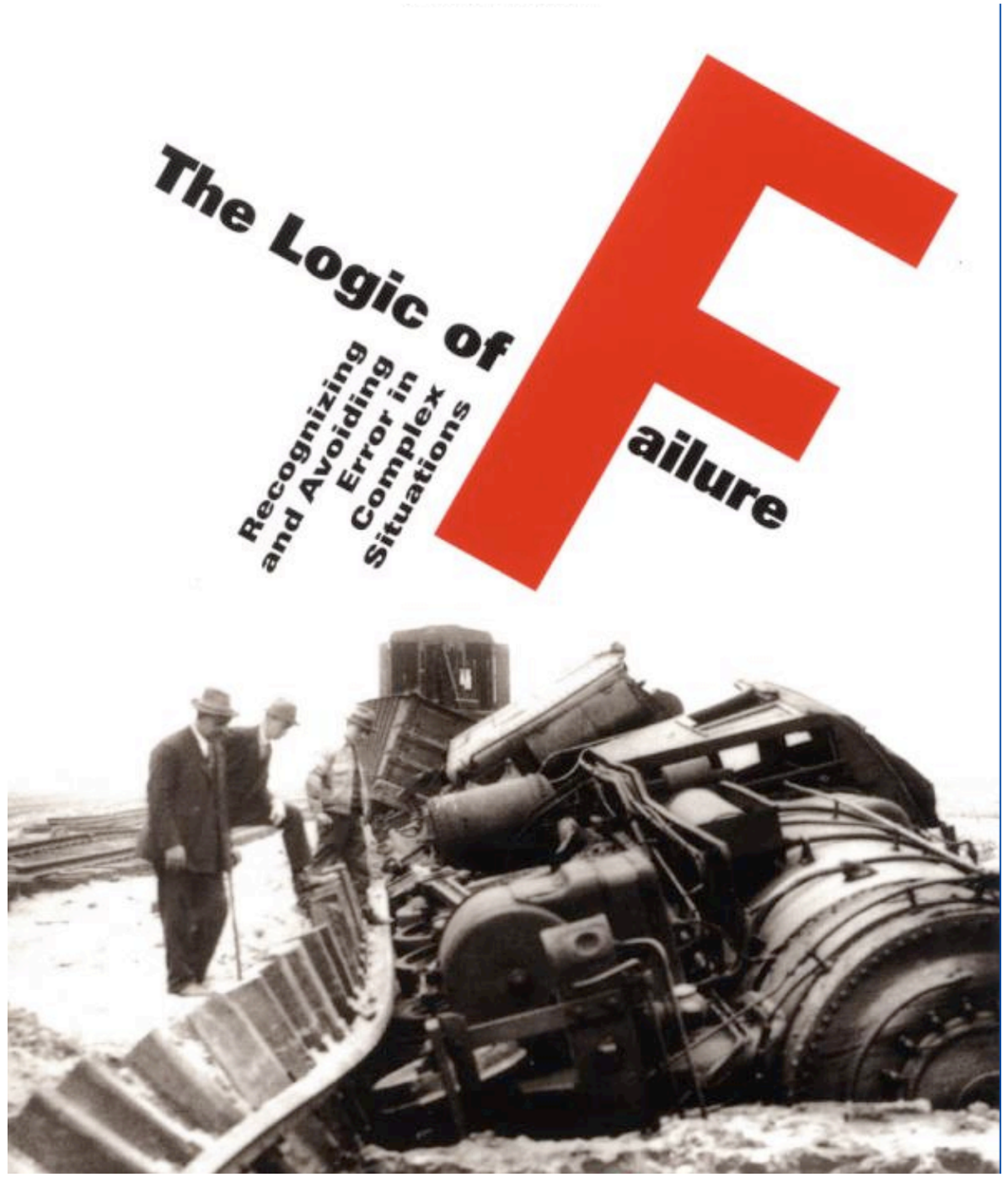


EJB EntityBeans
the problems . . .





EJB EntityBeans
the end . . .



~~EJB~~ EntityManager
the next step . . .

javax.persistence

JSR220 - EJB 3.0
A Two-Part Specification

- EJB 3 specification consists of two documents:
  - Session and Message-Driven Beans
  - EJB 3 Persistence API
- Future versions of the Persistence portion of the EJB 3.0 specification will be spun off into a separate specification

From"A Technical Introduction to EJB3 Persistence" by Patrick Linskey

View entire presentation at http://www.javalobby.org/eps/ejb3/

JSR220 - EJB 3.0
Where We Started: Ease of Use

- Support simple Java classes
- Improve support for domain modelling
- Expand query capabilities
- Standardize ORM metadata
- Improve testability
- Fix DTO anti-pattern

From"A Technical Introduction to EJB3 Persistence" by Patrick Linskey

View entire presentation at http://www.javalobby.org/eps/ejb3/

JSR220 - EJB 3.0
Where We Are

- Persistence API expanded to include use outside of Java EE containers
- Evolved into “common” Java persistence API
  - Merger of expertise the entire Java ORM space
- Support for pluggable, third-party persistence providers inside an EJB3 container

From”A Technical Introduction to EJB3 Persistence” by Patrick Linskey

View entire presentation at http://www.javalobby.org/eps/ejb3/

JSR220 - EJB 3.0
EJB3 Persistence

EJB3 Persistence is made of six main parts:

- Domain Object Model
- Metadata
- Application Programming Interface
- Lifecycle Model, including detachment
- Queries
- Callbacks

From"A Technical Introduction to EJB3 Persistence" by Patrick Linskey

View entire presentation at http://www.javalobby.org/eps/ejb3/

JSR220 - EJB 3.0
Domain Model: Entities

- Entities support standard OO domain modeling techniques
  - Inheritance
  - Encapsulation
  - Polymorphic Relationships
- Can be created with the `new` operator
- There is no required superclass / interface in the EJB3 Persistence API

From"A Technical Introduction to EJB3 Persistence" by Patrick Linskey

View entire presentation at http://www.javalobby.org/eps/ejb3/

JSR220 - EJB 3.0
Entity Restrictions

- Must have one or more fields mapped to the database primary key
- Must not be `final`
- For portable optimistic locking support, must have a single version field
- Persistent data is represented via member fields (`AccessType.FIELD`) or methods (`AccessType.PROPERTY`)

JSR220 - EJB 3.0
EJB3 Metadata

- Two types of metadata:
  - **logical** metadata describing the domain object model
  - **mapping** metadata describing how the object model maps to the database schema
- Java 5 annotations and / or XML
- Most metadata settings have intelligent defaults

From"A Technical Introduction to EJB3 Persistence" by Patrick Linskey

View entire presentation at http://www.javalobby.org/eps/ejb3/

JSR220 - EJB 3.0
EJB3 Persistence API

- Interfaces defined in the `javax.persistence` package
- In EJB context, the `EntityManager` is:
  - injected into enterprise beans
  - integrated with the active transaction context
- Key interfaces:
  - `EntityManager`
  - `Query`

From"A Technical Introduction to EJB3 Persistence" by Patrick Linskey

View entire presentation at http://www.javalobby.org/eps/ejb3/

JSR220 - EJB 3.0

JPA Environments

- **Managed environment**

  JNDI Lookup of EntityManagerFactory or EntityManager

- **Standalone non-managed**

  configure peristence.xml and use Persistence.createEntityManagerFactory

JSR220 - EJB 3.0
JPA Object Life Cycle

new - no persistent identity

managed - persistent identity and currently associated with persistence context

detached - persistent identity and not currently associated with persistence context

removed - persistent identity and currently associated with persistence context and scheduled for removal

JSR220 - EJB 3.0
JPA Support in Spring

Interfaces and Classes:

```
LocalEntityManagerFactoryBean
JpaTemplate
JpaDaoSupport
JpaTransactionManager
```

JSR220 - EJB 3.0
Spring JPA Wiring

```
<!-- Service -->
<bean id="beerDistributorService"
    class="....BeerDistributorServiceJpa">
  <property name="entityManagerFactory"
      ref="entityManagerFactory"/>
</bean>
<!-- Persistence Definitions -->
<bean id="entityManagerFactory"
  class="org...orm.jpa.LocalEntityManagerFactoryBean">
  <property name="entityManagerName"
      value="BeerDistributor"/>
</bean>
<bean id="transactionManager"
  class="org...orm.jpa.JpaTransactionManager">
  <property name="entityManagerFactory"
      ref="entityManagerFactory"/>
</bean>
```

JSR220 - EJB 3.0
Spring JPA Usage

```
public class JpaCustomerDao extends JpaDaoSupport
        implements CustomerDao {

    public List getCustomerList() {
        return getEntityManager().createQuery(
            "select object(o) from Customer o order by o.id")
                .getResultList();
    }

    public Customer findCustomer(Long id) {
        return getEntityManager().find(Customer.class, id);
    }

    public Customer saveCustomer(Customer c) {
        if (getEntityManager().contains(c)) {
            getEntityManager().persist(c);
            return c;
        }
        else {
            return getEntityManager().merge(c);
        }
    }
}
```

JSR220 - EJB 3.0
persistence.xml - Hibernate

```
<entity-manager>
  <name>BeerDistributor</name>
  <provider>org.hibernate.ejb.HibernatePersistence</provider>
  <properties>
    <property name="hibernate.dialect"
      value="org.hibernate.dialect.PostgreSQLDialect"/>

      <!-- DataSource - either local or JNDI lookup -->
      <property name="hibernate.connection.driver_class" value="..."/>
      <property name="hibernate.connection.url" value="..."/>
      <property name="hibernate.connection.username" value="..."/>
      <property name="hibernate.connection.password" value="..."/>

      <property name="hibernate.cache.use_query_cache" value="false"/>
      <property name="hibernate.max_fetch_depth" value="3"/>
      <property name="hibernate.ejb.autodetection" value="class"/>

  </properties>
</entity-manager>
```

JSR220 - EJB 3.0
persistence.xml - Kodo

```
<?xml version="1.0"?>
<entity-manager>
  <name>BeerDistributor</name>
  <provider>kodo.persistence.PersistenceProviderImpl</provider>

  <properties>
    <!-- DataSource - either local or JNDI lookup -->
    <property name="kodo.ConnectionDriverName" value="..."/>
    <property name="kodo.ConnectionURL" value="..."/>
    <property name="kodo.ConnectionUserName" value="..."/>
    <property name="kodo.ConnectionPassword" value="..."/>

    <property name="kodo.Log"
      value="DefaultLevel=INFO, Runtime=DEBUG, Tool=INFO"/>
  </properties>

</entity-manager>
```

JSR220 - EJB 3.0
Spring Transactions

```
<bean class="org.springframework.aop.framework.autoproxy.DefaultAdvisorAutoProxyCreator"/>

<bean
    class="org.springframework.transaction.interceptor.TransactionAttributeSourceAdvisor">
  <property name="transactionInterceptor" ref="txInterceptor"/>
</bean>

<bean id="txInterceptor"
    class="org.springframework.transaction.interceptor.TransactionInterceptor">
  <property name="transactionManager" ref="transactionManager"/>
  <property name="transactionAttributeSource">
    <bean
      class="org.springframework.transaction.annotation.AnnotationTransactionAttributeSource"/>
  </property>
</bean>

<!-- Exception Translator -->
<bean class="org.springframework.aop.framework.autoproxy.BeanNameAutoProxyCreator">
  <property name="beanNames" value="beerDistributorService"/>
  <property name="interceptorNames" value="exceptionTranslator"/>
</bean>
<bean id="exceptionTranslator"
    class="org.springframework.orm.jpa.PersistenceExceptionTranslatingThrowsAdvice"/>
```

JSR220 - EJB 3.0
Spring Transactions

```
@Transactional(readOnly=true)
public interface BeerDistributorService {

    public Beer findBeer(Long id);

    @Transactional(readOnly=false)
    public Beer saveBeer(Beer b);

    public List getCustomerList();

    public Customer findCustomer(Long id);

    public Customer findCustomerAndOrders(Long id);

    @Transactional(readOnly=false)
    public Customer saveCustomer(Customer c);

}
```

Resources

Spring:
http://www.springframework.org/

“A Technical Introduction to EJB3 Persistence”:
http://www.javalobby.org/eps/ejb3/

Specification:
http://www.jcp.org/en/jsr/detail?id=220

Persistence providers:
http://www.hibernate.org/
http://www.solarmetric.com/
http://www.oracle.com/technology/index.html