Hibernate EntityManager

EJB 3.0 and Java Persistence

© Red Hat 2006

Specification

© Red Hat 2006

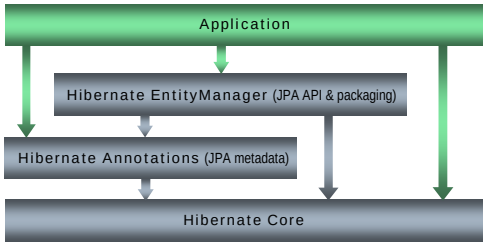
Specification Goals

- Ease of Use
  - ✓ Annotations
  - ✓ No more deployment descriptor
  - ✓ POJO based programmatic model
  - ✓ Reduce the number of “technical” artifacts
  - ✓ Facilitate Test Driven Development
- Powerful ORM solution…

EJB3 Entity

- Influenced by Hibernate and others
- Ease of use
  - ✓ POJO based
  - ✓ Metadata through annotations (no XML DDs)
  - ✓ No required interfaces nor subclassing
  - ✓ Testable in unit tests / in Java SE
- Solve/remove value object anti-pattern
- Provide full Object/Relational mapping
  - ✓ Inheritance
  - ✓ Polymorphic associations
  - ✓ Expand EJB-QL / JPA-QL

Hibernate - EJB3: the big picture



Hibernate - EJB3: the big picture

- Mix metadata
  - ✓ Annotations
  - ✓ JPA Deployment Descriptor
  - ✓ HBM XML files
- Mix APIs
  - ✓ JPA (Hibernate EntityManager)
  - ✓ Fall back on Hibernate Session
    - `session = (Session) em.getDelegate()`

ORM metadata

© Red Hat 2006

Annotation based metadata

- Configuration by exception
  - ✓ Common cases do not requires annotations
  - ✓ Sensible defaults are used
- One annotation per concept
  - ✓ Decoration principle
  - ✓ Logical annotations (@ManyToOne)
  - ✓ Physical annotations (@Table)

Minimal mapping

The POJO is actually a mapped entity

One of the properties represents the PK

2 annotations, that's it!

```
@Entity
public Document {
    @Id private Long id;
    private String title;
    private String summary
    private String content;

   //getters and setters...
}
```

```
create table Document (
    id bigint not null primary key,
    title varchar(255),
    summary varchar(255),
    content varchar(255)
)
```


Configuration by decoration

- Refine the physical model
  - ✓ @Table, @SecondaryTable, @Column
- Define the logical mapping
  - ✓ @Version, @Transient, @Embeddable
- Defining the id generation strategy
  - ✓ @GeneratedValue
  - ✓ @SequenceGenerator / @TableGenerator
- And much more…

Inheritance

- Map a hierarchy
  - ✓ Table per class hierarchy
    - @Inheritance(strategy=SINGLE_TABLE)
    - @DiscriminatorColumn
  - ✓ Table per concrete class
    - @Inheritance(strategy=TABLE_PER_CLASS)
  - ✓ Normalized model (table per subclass)
    - @Inheritance(strategy=JOINED)

Association

```
@Entity @Table(name="Shipments")
public class Shipment {
   @ManyToOne(optional=false)
   @JoinColumn(name="ADDRESS_ID")
   private Address address;
}
```

Specify the FK column

ADDRESS

ADDRESS_ID	STREET
1	Foo Str.
2	Bar Str.

SHIPMENTS

SHIP_ID	STATE	ADDRESS_ID
1	AGR	1
2	DEL	1
3	PAY	2

Bidirectional

- You don't have to
- One side has to be the owner of the association

```
@Entity
public class Address {
   @OneToMany(mappedBy="address")
   private Set<Shipment> shipments;
}
```


Associations

- Logical mapping
  - ✓ @OneToOne, @ManyToOne
  - ✓ @OneToMany, @ManyToMany
  - ✓ Fetch LAZY or EAGER
- Physical representation
  - ✓ @JoinColumn
  - ✓ @JoinTable
  - ✓ @PrimaryKeyJoinColumn

Hibernate extensions

- org.hibernate.annotations.*
  - ✓ Better fetching strategies
  - ✓ More collection support
  - ✓ Custom type extensions
  - ✓ ...
- Annotations not in classpath
  - ✓ Ignored
  - ✓ No runtime dependency on Hibernate Annotations

JBoss World 2006 LAS VEGAS

Packaging a JPA application

© Red Hat 2006

Default packaging

- Make life simple
- 1. Describe the persistence unit
  - ✓ META-INF/persistence.xml
- 2. Copy @Entity classes in a JAR
  - ✓ JAR contains your domain model
- 3. Play with your XBox 360
  - ✓ Alternative available

Persistence.xml

```
<persistence version="1.0">

  <persistence-unit
      name="sample"
      transaction-type="JTA">
    <jta-data-source>java:/DefaultDS</jta-data-source>
  </persistence-unit>

</persistence>
```

- Points to a datasource
- Entity are discovered by archive scanning

More about persistence.xml

- Other attributes
  - ✓ <provider>: JPA implementation
  - ✓ <mapping-file>: EJB3 DD
  - ✓ <jar>: other jars to consider
  - ✓ <class>: Entity
  - ✓ <exclude-unlisted-classes>: no scanning
  - ✓ <properties>: Hibernate properties
    - Dialect
    - Hbm2ddl.auto: schema generation
    - Database properties in JavaSE

Deployment and bootstrapping

In Java SE and in Java EE

© Red Hat 2006

Java SE

- Bootstrap process standardized
- Find the matching persistence.xml

```
EntityManagerFactory emf =
  Persistence.createEntityManagerFactory( “sample“ );

//keep the ref somewhere
…
EntityManager em = emf.createEntityManager();
em.getTransaction().begin();
em.persist(client);
Order order = em.get(Order.class, orderId);
em.getTransaction.commit();
em.close();
…
emf.close();
```


Java EE

- Put your persistence jar in your EAR
- Lifecycle handled for you
  - ✓ entityManagerFactory
  - ✓ entityManager
    - No connection leaking
    - No technical code

Typical Session Bean

```
@Stateless @TransactionAttribute(REQUIRED)
public EditDocumentBean implements EditDocument {
   @PersistenceContext(name=“sample”)
   private EntityManager em;

   public Document get(Long id) {
      return em.find(Document.class, id);
   }

   public Document save(Document doc) {
      return em.merge(doc);
   }
}
```


JBoss Embeddable

- I want Java EE ease of use in Java SE
- JBoss Embeddable runs in
  - ✓ Unit tests
  - ✓ Main apps
  - ✓ Weblogic
  - ✓ Websphere
  - ✓ Tomcat
- JBoss Embeddable is
  - ✓ EJB3 container
  - ✓ JTA
  - ✓ …

JPA/EJB3 design patterns

© Red Hat 2006

I've read that...

- Forget what you know about J**2**EE DP
- Design pattern useful
  - ✓ In a given context
  - ✓ To solve a given problem

Persistence context lifetime

- Persistence context should reflect
  - ✓ Use case lifetime
  - ✓ For a given user
- Maximize optimization
  - ✓ Keep track of the object changes
  - ✓ Delay the operation until needed
  - ✓ An object is loaded once

Single request/response usecases

- Implicit contract of a Stateless SB
- PC bound to the transaction lifetime
- Declare the transaction boundary
  - ✓ Stateless SB method
- PC shared across Session Beans
  - ✓ Involved in the same transaction
  - ✓ Injected through @EJB
  - ✓ PC injected (@PersistenceContext)

Several request/response cycles

- Think wizards, think conversation
- Implicit contract of a Stateful SB
- PC bound to the SFSB lifetime
- PC shared across Session Beans
  - ✓ Injected through @EJB
  - ✓ PC injected
    - @PersistenceContext(type=EXTENDED)
- Yes, a Stateful session bean
  - ✓ does scale
  - ✓ has a better contract than HttpSession replication

Data Access Object

- Does it still applies in EJB3
  - ✓ Not for database abstraction
  - ✓ Centralize code related to data access
- How do I inject my persist. Context
  - ✓ Make your DAO a local SLSB
  - ✓ SLSB and SFSB are POJOs
  - ✓ Requires 2 annotations
    - @Stateless
    - @PersistenceContext

Data Transfer Object

- Patterns used because EJB 2 Entities
  - ✓ Not serializable
  - ✓ Bound to the container
- Anti-pattern in EJB 3.0
  - ✓ Useless extra LOC
  - ✓ Parallel class hierarchy smell
  - ✓ Shotgun change smell
- You can still use it if you need
  - ✓ Clear SoC when your client side is not controlled

Hibernate EntityManager runs

- In Java SE 5
- In any J2EE container (Java SE 5 required)
  - ✓ No Persistence Context lifecycle managt though
- In JBoss EJB3
  - ✓ Embeddable
  - ✓ JBoss AS 4.0.4
  - ✓ JBoss AS 5
- In Sun Glassfish
- In ObjectWeb JOnAS and EasyBean
- Any JavaEE compliant app server

Q&A