

JEE - Java Enterprise Edition

The JPA EntityManager



Java Persistence API Entity Manager

- Methods to **insert** and **remove** entities, as well as **merge updates** from *detached entities*
- Also provides a rich API for querying entities
- Some of the most important functions are
 - public void **persist**(Object entity);
 - public <T> T **find**(Class <T> entityClass, Object primaryKey);
 - public <T> T **merge**(T entity);
 - public void **remove**(Object entity);
 - public void **lock**(Object entity, LockModeType lockMode);
 - public void **refresh**(Object entity);
 - public Query create**Query**(String queryString);
 - public void close();
 - public boolean isOpen();



Entity Manager Persisting Entities

- Act of **inserting an entity into a database**

```
CustomerEntity newCustomer = new CustomerEntity();  
newCustomer.setFirstName("Uwe");  
newCustomer.setLastName("van Heesch");  
entityManager.persist(newCustomer);
```

- New Entity is **queued** for insertion
- Object instance becomes ***managed***
- Throws **IllegalArgumentException** if parameter is not an Entity Bean

Entity Manager Finding Entities

- Act of **locating objects in a database**
- **2 methods** of finding an entity
 - find() or getReference()

```
CustomerEntity customer = null;  
customer = entityManager.find(CustomerEntity.class,2);
```

```
CustomerEntity customer = null;  
try {  
    customer = entityManager.getReference(CustomerEntity.class,2);  
} catch(EntityNotFoundException nnf){  
    // recover error  
}
```

Entity Manager Finding Entities

- **2 methods** of finding an entity (continued)
 - Queries
 - very **analogous to** creating and executing **JDBC PreparedStatement**

```
Query query = entityManager.createNamedQuery("from CustomerEntity c where id=2");  
CustomerEntity customer = (CustomerEntity)query.getSingleResult();
```

- unlike EJB2.1 there are **no finder methods**
- all **object instances remain managed** as long as the persistence context remains active
- **changes** will be **synchronized automatically**



Entity Manager Merging Entities

- JPA allows to **merge state changes** made to **detached entities**
- E.g. an entity is changed in a remote client application after being allocated via a SessionBean
- Entity **outside the persistence context** is **detached**, thrf changes are **not synchronized**



Entity Manager Merging Entities

- Entity **outside the persistence context** is **detached**, thrf changes are **not synchronized**

```
// ...  
CustomerEntity customer = (CustomerEntity)query.getSingleResult();  
return customer;  
// ...  
// some changes outside the persistence context  
CustomerEntity mergedEntity = entityManager.merge(customer);
```

- merge() operation **returns a managed instance**
- the **parameter remains detached** and unmanaged

Entity Manager Removing Entities

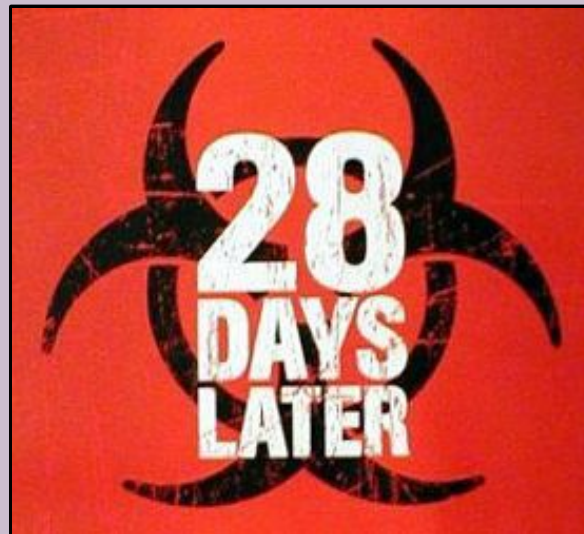
- Entities can be **removed from the database**

```
CustomerEntity cust = entityManager.find(CustomerEntity.class, 2);  
entityManager.remove(cust);  
cust.setLastName("Stallman");
```

- After remove() is invoked, the customer can still be used, but the **entity is detached** and therefore **no longer managed**
- the remove() method can only be **made undone by re-creating the entity** using the **persist()** method again

Entity Manager Locking Entities

- **EntityManager supports both read and write locks**
- Locks are discussed in detail later, when talking about **transactions**



Entity Manager Refreshing Entities

- refresh() method **refreshes the state** of an entity bean from the database
- **no changes** are made **to the database**
- all **changes in the bean** are **overwritten**



NetBeans Demonstration



JEE - Java Enterprise Edition

The JPA EntityManager - Questions?

