EntityManager

EntityManager

- Implementerar
  - Java Persistence management API
  - Java Persistence Query Language
  - Java Persistence object lifecycle rules
  - Java Persistence configuration and packaging

- EntityManager är APIt som interagerar med persistensen.
- Det är alltså via den som alla access mot databasen sker.
  - find
  - create
  - ...

- du definierar upp hur den persistenta miljön ska se ut genom en fil, persistence.xml som ska ligga i foldern META-INF.

persistence.xml

```
<persistence xmlns="http://java.sun.com/xml/ns/persistence"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://java.sun.com/xml/ns/persistence
    http://java.sun.com/xml/ns/persistence/persistence_1_0.xsd"
  version="1.0">
  <persistence-unit name="manager1" transaction-type="JTA">
    <provider>org.hibernate.ejb.HibernatePersistence</provider>
```

persistence.xml

```
<persistence-unit name="manager1" transaction-type="RESOURCE_LOCAL">
    <properties>
      <property name="hibernate.dialect" value="org.hibernate.dialect.MySQLDialect"/>
      <property name="hibernate.connection.driver_class" value="com.mysql.driver.Driver"/>
      <property name="hibernate.connection.username" value="reine"/>
      <property name="hibernate.connection.password" value="tjo"/>
      <property name="hibernate.connection.url" value="jdbc:mysql://localhost/reine"/>
      <property name="hibernate.max_fetch_depth" value="3"/>
    </properties>
  </persistence-unit>
</persistence>
```

Att använda EntityManager

```
EntityManagerFactory emf =
  Persistence.createEntityManagerFactory("manager1");

    EntityManager em = emf.createEntityManager();

    NewEntity test = new NewEntity();

    em.persist(test);

    em.close();

    emf.close();
```

Läsa om ifrån DB

em.persist(cat);

em.flush(); // force the SQL insert and triggers to run

em.refresh(cat); //re-read the state (after the trigger executes)

```
cat = em.find(Student.class, catId);


long catId = 1234;
em.find( Student.class, new Long(catId) );
```

```
Adress street = (Adress) em.createQuery(
    "select student.adress from Student as student
   where student = ?1")
    .setParameter(1, pelle)
    .getSingleResult();
```

```
List names = new ArrayList();

names.add("Pelle");

names.add("Kalle");

Query q = em.createQuery("select student from Student student where
    student.name in (:namesList)");

q.setParameter("namesList", names);

List cats = q.list();
```

```
em = emf.createEntityManager();

Transaction tx = em.getTransaction().begin();

em.setFlushMode(FlushModeType.COMMIT);

Cat izi = em.find(Cat.class, id);

izi.setName(iznizi);

em.createQuery("from Student").getResultList();

...

em.getTransaction().commit(); // flush occurs
```

```
EntityManager em = emf.createEntityManager();
EntityTransaction tx = null;
try {
    tx = em.getTransaction();
    tx.begin();
    ...
    tx.commit();
} catch (RuntimeException e) {
    if ( tx != null && tx.isActive() ) tx.rollback();
    throw e; // or display error message
}
finally {
    em.close();
}
```

```
@Resource public UserTransaction utx;
@Resource public EntityManagerFactory factory;
public void doSomething() {
    EntityManager em = factory.createEntityManager();
    try {
    ...
    utx.commit();
} catch (RuntimeException e) {
    if (utx != null) utx.rollback();
}
finally {
    em.close();
}
```