JPA API ile Çalışmak

harezmi bilişim çözümleri

Container Managed PersistenceContext + CMT

PersistenceContext Servlet, EJB veya Spring bean içerisine enjekte edilebilir

```
@Stateless
public class PetClinicService {

    @PersistenceContext
    private EntityManager entityManager;

    public void create(Pet pet) {
        entityManager.persist(pet);
    }
}
```

Transactional EntityManager nesnesi sadece JTA transaction tipinde enjekte edilebilir

TX begin/commit/rollback container tarafından transparan biçimde halledilir

Transactional işlem sonunda EntityManager uygulama içerisinden kapatılmaz

```
<?xml version="1.0" encoding="UTF-8"?>
<persistence version="2.1"
xmlns="http://xmlns.jcp.org/xml/ns/persistence"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/persistence
http://xmlns.jcp.org/xml/ns/persistence/persistence_2_1.xsd">
    <persistence-unit name="jpa-hibernate" transaction-type="JTA">
        <jta-data-source>jdbc/myDS</jta-data-source>
        <properties>
            <property name="hibernate.ejb.cfgfile" value="/hibernate.cfg.xml" />
        </properties>
    </persistence-unit>
</persistence>
```

Container Managed PersistenceContext + BMT

Container'a JTA'nın EJB içerisinde programatik yönetileceği belirtilmiştir

```
@Stateless
@TransactionManagement(TransactionManagementType.BEAN)
public class PetClinicService {

    @PersistenceContext
    private EntityManager entityManager;

    @Resource
    private UserTransaction userTransaction;

    public void create(Pet pet) {
        try {
            userTransaction.begin();

            entityManager.persist(pet);

            userTransaction.commit();

        } catch (Exception ex) {
            userTransaction.rollback();
            throw ex;
        }
    }
}
```

Container tarafından yönetilen UserTransaction ve EntityManager nesneleri EJB içerisine enjekte edilirler

EntityManager aktif UserTransaction'a otomatik olarak dahil olur

TX begin/commit/rollback işlemleri uygulama içerisinde gerçekleştirilir

Persistence unit konfigürasyonu olarak yine transaction-type=”JTA” kullanılması zorunludur

Application Managed PersistenceContext + Local TX

```
@Stateless
public class PetClinicService {

    @PersistenceUnit
    private EntityManagerFactory entityManagerFactory;

    public void create(Pet pet) {
        EntityManager entityManager = entityManagerFactory.createEntityManager();
        EntityTransaction transaction = entityManager.getTransaction();
        try {
            transaction.begin();
            entityManager.persist(pet);
            transaction.commit();
        } catch (Exception ex) {
            transaction.rollback();
            throw ex;
        } finally {
            entityManager.close();
        }
    }
}
```

Container tarafından yönetilen EMF nesnesi Servlet, EJB veya Spring bean içerisine enjekte edilebilir

EntityManager yaratmak, TX başlatmak commit/rollback yapmak ve entityManager'ın kapatılması uygulamanın görevidir transaction-type="RESOURCE_LOCAL" olduğu için JTA UserTransaction ile etkileşim mümkün değildir

```
<persistence-unit name="jpa-hibernate" transaction-type="RESOURCE_LOCAL">
    <non-jta-data-source>jdbc/myDS</non-jta-data-source>
    <properties>
        <property name="hibernate.ejb.cfgfile" value="/hibernate.cfg.xml" />
    </properties>
</persistence-unit>
```

Application Managed PersistenceContext + CMT

```
@Stateless
public class PetClinicService {

    @PersistenceUnit
    private EntityManagerFactory entityManagerFactory;

    public void create(Pet pet) {
        EntityManager entityManager = entityManagerFactory.createEntityManager();
        try {
            entityManager.persist(pet);
        } finally {
            entityManager.close();
        }
    }
}
```

EJB'nin default TX yöntemi CMT'dir

CMT kullanıldığı için container metot çağrısının başında aktif bir JTA UserTransaction'ın olmasını garanti eder

EntityManager'da bu aktif JTA transaction'a otomatik olarak katılır (join)

Metot sonunda EntityManager'ın kapatılması uygulamanın görevidir

```
<persistence-unit name="jpa-hibernate" transaction-type="JTA">
    <jta-data-source>jdbc/myDS</jta-data-source>
    <properties>
        <property name="hibernate.ejb.cfgfile" value="/hibernate.cfg.xml" />
    </properties>
</persistence-unit>
```

Application Managed PersistenceContext + BMT

```
@Stateless
@TransactionManagement(TransactionManagementType.BEAN)
public class PetClinicService {

    @PersistenceUnit
    private EntityManagerFactory entityManagerFactory;

    @Resource
    private UserTransaction userTransaction;

    public void create(Pet pet) {
        EntityManager entityManager = null;
        try {
            userTransaction.begin();
            entityManager = entityManagerFactory.createEntityManager();
            entityManager.persist(pet);
            userTransaction.commit();
        } catch (Exception ex) {
            userTransaction.rollback();
            throw ex;
        } finally {
            if(entityManager != null) entityManager.close();
        }
    }
}
```

EntityManager'ın UserTransaction başlatıldıktan sonra yaratılması önemlidir
transaction-type="JTA" olduğu için yaratılan EntityManager otomatik olarak JTA UserTransaction'a katılır

EntityManager'ı kapatmak yine uygulamanın görevidir

Application Managed PersistenceContext + BMT + Join

```
@Stateless
@TransactionManagement(TransactionManagementType.BEAN)
public class PetClinicService {

    @PersistenceUnit
    private EntityManagerFactory entityManagerFactory;

    @Resource
    private UserTransaction userTransaction;

    public void create(Pet pet) {
        EntityManager entityManager = entityManagerFactory.createEntityManager();
        try {
            userTransaction.begin();

            entityManager.joinTransaction();

            entityManager.persist(pet);
            userTransaction.commit();
        } catch (Exception ex) {
            userTransaction.rollback();
            throw ex;
        } finally {
            if(entityManager != null) entityManager.close();
        }
    }
}
```

EntityManager UserTransaction başlatılmadan önce yaratıldığı için joinTransaction() metodu ile aktif UserTransaction'a katılması gerekir

EntityManager'ı kapatmak yine uygulamanın görevidir

İletişim

- Harezmi Bilişim Çözümleri
- Kurumsal Java Eğitimleri
- http://www.java-egitimleri.com
- info@java-egitimleri.com