

Veri Ekleme ve Güncelleme

Burak Bayramlı

Bu belge, \LaTeX ile üretilmiştir

Nasıl

- Veri tabanı ile iletişimin temeli nesnelere olduğu için, yazarken de artık POJO nesnelerini kullanıyoruz.
- Hibernate ile bir nesneyi okuduktan sonra üzerinde set .. işlemleri gerçekleştirebiliriz (temel Java). Daha sonra bu nesneyi `entityManager.persist(obj)` ile tabana geri yazmak mümkündür.
- Ya da, bir nesneyi `new` ile hafızada yaratıp, yine set ..

işlemleri ile içine doldurduktan sonra yine persist ile onu tabana ekleyebiliriz.

Ekleme ve İlişkiler

Ekleme istediğiniz nesnenin bire bir, bire çok ya da çok çok ilişkisi olması durumunda, işaret “eden” nesnenin yeni nesneden haberdar olması gerekir. Bu durumda,

- **Bire bir:** İşaret eden nesnenin referansı yeni nesneye set edilmelidir
- **Bire çok:** Bir tarafındaki nesnenin “listesine” çok taraftaki yeni obje eklenmelidir

- **Çoka çok:** Her iki taraftaki listeye bir obje eklenmelidir. Yeni nesnedeki listeye ilişki kurulan diğer obje, diğer objedeki listeye ise yeni obje eklenmelidir.

Commit

Eğer uygulama servisi ortamında değil isek, commit çağrısını bizim yapmamız gerekecek.

```
EntityManager em = emf.createEntityManager();  
EntityTransaction tx = em.getTransaction();  
  
tx.begin();  
Xyz car = (Car) em.find(..);  
...  
tx.commit();
```

Uygulama Servisi Durumunda

```
@Stateful
public class XyzService ..
{
    @PersistenceContext EntityManager entityManager;

    @TransactionAttribute(TransactionAttributeType.REQUIRED)
    public void add(Xyz xyz)
    {
        ...
        entityManager.persist(xyz);
    }
}
```

Uygulama Servisi Durumunda

- Bu durumda biz bir EJB3 servisinin, bu durumda bir Stateful EJB'nin hangi transaction ortamında çalıştığını “beyan” ediyoruz. API ile bir çağrı yapmıyoruz.
- Bu beyandan sonra, bu nesnenin metotları ne zaman çağrılırsa, bu beyana göre transaction'ları başlatılıyor.
- `TransactionAttributeType.REQUIRED`'a göre bir metot çağrıldığında her zaman bir transaction gerekir,

var ise ona dahil olunur, yok ise hata mesajı verilir. `REQUIRES_NEW` ile muhakkak yeni bir transaction başlatılır.

- Metot sonunda transaction commit edilir. Java Exception'ı atılırsa, rollback gerçekleştirilir.
- Bu transaction, JBoss JTA sayesinde veri tabanı transaction'ını da kapsar.

Kopuk Nesneler

- Hibernate perde arkasında kendine has bir “taban bağlantısı” kavramını takip eder.
- Bir nesne, okunduğu sonra da bu bağlantının halen parçasıdır.
- Fakat Web mimarilerinde bazen güncellenmek istenen nesnelere bile, “sonradan” bir bağlantıya dahil olabilirler. Yâni üzerlerinde `persist` çağrısı yapılmak istenir.

- Bu durumda, bir nesnenin dışarıdan geldiğini Hibernate'e sinyalleme gerekiyor. Bunu `merge` çağrısı ile yapıyoruz.

Görevler

- Önce basit Car projesinde bir nesneyi okuyup, sadece `description` alanını değiştirip güncelleyin.
- Yeni bir Car nesnesi ekleyin.
- Bire çok projede yeni bir Car ekleyin, iki tarafı birbirinden haberdar edin (gerekli set ve add çağrılarını yapın)

- Ekleme işlemini çoka çok projesinde yapın.

Son