Optimistic ve Pessimistic LockMode Türleri

harezmi bilişim çözümleri

Hibernate LockMode & LockOptions Nedir?

- Entity'ler üzerinde **farklı türlerde** optimistic ve pessimistic lock alma yöntemleridir
- LockMode ve LockOptions her ikisi de **aynı** şeylerdir
- Herhangi bir lock almayıp **detached entity**'yi veya proxy'yi Session'a **tekrar attach etmek** ve **lazy ilişkilerine erişmek** için de kullanılabilir

```
session.lock(o, LockMode.READ);
```

veya

```
LockRequest lockRequest = session.buildLockRequest(LockOptions.READ);

lockRequest.lock(o);
```

LockMode Türleri

- **LockMode.NONE**
  - Sadece detached nesnelerin Session'a **reattach** yapılmasını sağlar
  - **Versiyon kontrolü yapmaz**, dolayısı ile versiyon bilgisi güncel olmayan bir entity de Session'a reattach edilebilir

```
session.lock(pet, LockMode.NONE);
```

LockMode Türleri

- **LockMode.READ** :
  - Entity'nin ve DB'deki karşılığının **versiyonlarını karşılaştırır**, eğer nesnenin versiyon bilgisi güncel değil ise hata fırlatır
  - Entity first level veya second level cache'de olsa bile **versiyon kontrolü DB'den** yapılır

```
session.lock(pet, LockMode.READ);
```

```
/* READ lock com.javaegitimleri.petclinic.model.Pet */
select ID
from T_PET
where
    ID =?
    and VERSION =?
```

LockMode Türleri

- **LockMode.UPGRADE** :
  - Eğer db SELECT...FOR_UPDATE'i destekliyorsa DB düzeyinde **pessimistic write lock** atar
  - Eğer kayıt üzerinde **başka bir lock var ise** bu lock bırakılıncaya veya timeout süresi kadar **bekler**

```
session.beginTransaction();

session.lock(pet, LockMode.UPGRADE);

session.getTransaction().commit();
```

```
/* UPGRADE lock com.javaegitimleri.petclinic.model.Pet */
   select ID
   from T_PET
where
    ID =?
    and VERSION =? for update
```

LockMode Türleri

- **LockMode.PESSIMISTIC_WRITE**:
  - LockMode.UPGRADE ile **aynı**dır
  - **UPGRADE deprecated** olduğu için bunun kullanılması önerilir
  - DB SELECT FOR UPDATE ifadesini **desteklemiyor ise LockMode.READ** gibi davranır
- **LockMode.PESSIMISTIC_READ**:
  - PESSIMISTIC_WRITE ile **aynı**dır

LockMode Türleri

- **LockMode.UPGRADE_NOWAIT**
  - LockOptions.UPGRADE ile benzerdir
  - Fakat SELECT ... FOR UPDATE NOWAIT ifadesini kullanır
  - Bu durumda eğer lock elde edilemiyorsa hemen **hata fırlatır**
  - Eğer SQL dialect NOWAIT desteklemiyorsa kendiliğinden **LockOptions.UPGRADE**'e düşer

LockMode Türleri

- **LockMode.UPGRADE_SKIPLOCKED**
  - LockOptions.UPGRADE ile benzerdir
  - Fakat SELECT ... FOR UPDATE SKIP LOCKED ifadesini kullanır
  - Eğer lock elde edilemiyorsa hemen dönülür
  - Eğer SQL dialect desteklemiyorsa kendiliğinden **LockOptions.UPGRADE**'e düşer

LockMode Türleri

- **LockMode.OPTIMISTIC**
  - TX commit aşamasında entity'nin versiyonu DB'deki karşlığı ile **kontrol edilecektir**
  - Entity stale ise **hata** fırlatılır

```
session.beginTransaction();

session.lock(pet, LockMode.OPTIMISTIC);

session.getTransaction().commit();
```

```
/* get version com.javaegitimleri.petclinic.model.Pet */
select  VERSION
from
    T_PET
where
    ID =?
```

LockMode Türleri

- **LockMode.OPTIMISTIC_FORCE_INCREMENT**
  - TX commit aşamasında entity'nin versiyonu DB'deki karşlığı ile **kontrol edilir ve bir artırılır**
  - Kontrol sırasında entity stale ise hata fırlatılır

```
session.beginTransaction();

session.lock(pet, LockMode.OPTIMISTIC_FORCE_INCREMENT);

session.getTransaction().commit();
```

```
/* forced version increment */
update  T_PET
set
    VERSION=?
where
    ID=?
    and VERSION=?
```

LockMode Türleri

- **LockMode.FORCE**
  - Entity'nin versiyonu TX içerisinde **lock anında hemen** bir artırılacaktır
  - Increment sırasında mevcut **version değerinin güncel olup olmadığı** da kontrol edilmektedir

```
session.beginTransaction();

session.lock(pet, LockMode.FORCE);

session.getTransaction().commit();
```

```
/* forced version increment */
    update  T_PET
    set VERSION=?
    where
        ID=?
        and VERSION=?
```

LockMode Türleri

- **LockMode.PESSIMISTIC_FORCE_INCREMENT**
  - FORCE ile aynıdır
  - TX içerisinde **entity versiyonu anında artırılacak**tır
  - LockMode.FORCE **deprecated** olmuştur
  - Onun yerine **kullanılması önerilmektedir**

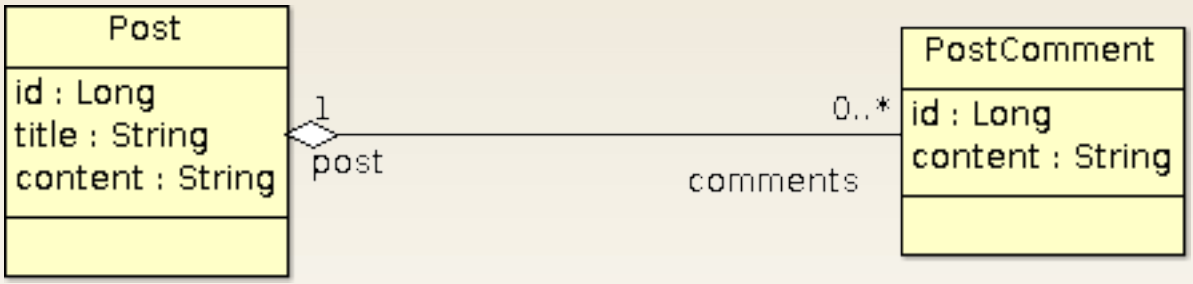
JPA LockModeType

- JPA'nın lock mode türleri **LockModeType** enum ile tanımlanmıştır
- Bire bir Hibernate LockMode türleri ile **eşleşmektedir**
- Hibernate'in **LockModeConverter** sınıfı bu eşleşmeyi yapmaktadır
- JPA'da lock yapılacak entity'nin persistence context'e **attached ve versioned olması** gerekir
- Aksi takdirde **hata** fırlatılır

JPA LockModeType

```
em.getTransaction().begin();

Pet pet = em.find(Pet.class, 1L);

...

em.lock(pet, LockModeType.READ);

...

em.getTransaction().commit();
```

Optimistic ve Pessimistic Lock Kullanım Örneği

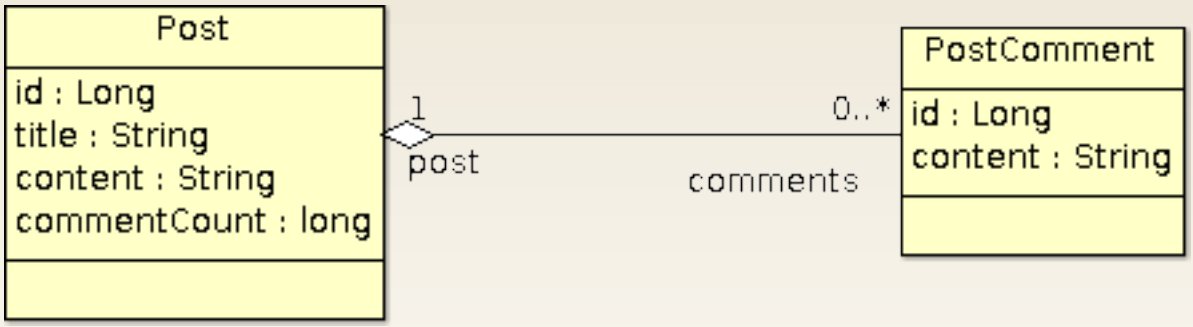



```
em.getTransaction().begin();

Post post = em.find(Post.class, 1L);
em.lock(post, LockModeType.OPTIMISTIC);
em.lock(post, LockModeType.PESSIMISTIC_READ);

PostComment pc = new PostComment();
pc.setContent("fix the f.cking typo in the title please!!!");
pc.setPost(post);

em.persist(pc);

em.getTransaction().commit();
```

TX commit anında version değerini sadece kontrol eder

TX commit'e kadar shared/write lock'ı tutar böylece başka bir TX post'u bu TX sonlanana değin değiştiremez

Optimistic ve Pessimistic Lock Kullanım Örneği




```
em.getTransaction().begin();

Post post = em.find(Post.class, 1L);
em.lock(post, LockModeType.OPTIMISTIC_FORCE_INCREMENT);


PostComment pc = new PostComment();
pc.setContent("fix the f.cking typo in the title please!!!");
pc.setPost(post);
post.setCommentCount(pc.getCommentCount()+1);

em.persist(pc);

em.getTransaction().commit();
```

TX commit anında version değerini kontrol eder ve bir artırır

Versiyon kontrolünün ve artışın hemen yapılması istenirse pessimistic_force_increment kullanılabilir

İletişim

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