Hibernate ScrollableResults & Iterator

harezmi bilişim çözümleri

Hibernate Sorgularında ScrollableResults

- JDBC API'deki “**scrollable resultsets**” gibidir
- RDBMS sisteminde bir “**cursor**” kullanılarak gerçekleştirilir
- Cursor sorgu sonuç listesindeki herhangi bir **kayda işaret** eder
- Uygulama içerisinden cursor **ileri/geri** hareket ettirilebilir
- Sorgu sonuçlarının **toptan hafızaya yüklenmesinin zor olduğu durumlar** için faydalıdır

Hibernate Sorgularında ScrollableResults

```
ScrollableResults resultList =
session.createQuery("from Owner").scroll();

resultList.first();
resultList.last();
resultList.get();
resultList.next();
resultList.scroll(3);
resultList.getRowNumber();
resultList.setRowNumber(5);
resultList.previous();
resultList.scroll(-3);
resultList.close();
```

scroll() metodu ScrollableResults nesnesi döner. ScrollableResults vasıtası ile sorgu sonuç listesi üzerinde iterate etmek mümkündür

Hibernate Sorgularında ScrollableResults

```
Session session = sessionFactory.openSession();
Transaction tx = session.beginTransaction();
ScrollableResults resultList =
    session.createQuery("from Owner").scroll();
int count=0;
while ( resultList.next() ) {
    Owner owner = (Owner) resultList.get(0);
    modifyOwner(owner);
    if ( ++count % 100 == 0 ) {
        session.flush();
        session.clear();
    }
}
resultList.close();
tx.commit();
session.close();
```

- ScrollMode.**SCROLL_INSENSITIVE**
  - Cursor açık olduğu müddetçe değişikliğe uğramış veri ile karşılaşılmasını engeller
- ScrollMode.**SCROLL_SENSITIVE**
  - Değişiklikleri ve yeni veriyi resultset anında yansıtır

TX sona ermeden evvel cursor kapatılmalıdır

Query.iterate()

- Query.iterate() ile bir **Iterator** elde edilerek entity'ler üzerinde iterate edilebilir
- Burada ilk SELECT sorgusu ile **sadece PK değerleri** alınır
- Daha sonra Iterator üzerinde dolaşırken nesnelerin state'leri **birincil** (persistence context) ve **ikincil** cache'lerde (second level cache) aranır
- Eğer **bulunamaz ise** her nesne için ayrı bir **SELECT** çalıştırılır

Query.iterate()

- Query.iterate() yöntemi ancak **ikincil cache aktif ise faydalı** olabilir
- Diğer durumda **N+1 SELECT** problemi doğurur
- Hibernate **Criteria** bu yöntemi desteklemez
- Iterator **son kayda** veya **Session kapatılana** değin açık tutulur
- **Hibernate.close(iterator)** ile explicit biçimde kapatılabilir

İletişim

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