Hibernate ve İkincil Ön Bellek (Caching)

harezmi bilişim çözümleri

Ön Bellek Bölgelerinin İsimlendirmesi

- Entity önbellek alanlarının ismi **sınıf isminin FQN**'idir
  - com.javaegitimleri.petclinic.model.Pet
- Collection ilişkilerinin önbellek alanlarının ismi ilişkinin ait olduğu entity sınıfın **FQN + property** ismidir
  - com.javaegitimleri.petclinic.model.Pet.visits
- Sorgular için default olarak **tek bir sorgu önbellek alanı** bulunur

Ön Bellek Bölgelerinin İsimlendirmesi

- Belirli bir SessionFactory veya persistence unit için **hibernate.cache.region_prefix** konfigürasyon property'si ile bölge adı değiştirilebilir
- Eğer uygulama **birden fazla SessionFactory** veya persistence unit kullanıyor ise bu özellik faydalıdır
- Aksi durumda cache bölge isimleri farklı persistence unit'lerde **çakışma** yaşayacaktır

EhCache İçin Ön Bellek Bölge Konfigürasyonu

```
<cache
name="com.javaegitimleri.petclinic.model.Owner"

maxElementsInMemory="500"
        eternal="true"
        timeToIdleSeconds="0"
        timeToLiveSeconds="0"
        overflowToDisk="false"
/>

<cache
name="com.javaegitimleri.petclinic.model.Pet"
    maxElementsInMemory="50000"
        eternal="false"

timeToIdleSeconds="1800"

timeToLiveSeconds="100000"
        overflowToDisk="false"
/>
```

- **overflowToDisk="false"** hafızadaki alan dolduğu vakit veriyi diskte tutup tutmamaya karar verir
- **eternal="true"** timeout üzerinden evict işlemini kontrol eder
- Eğer cache size nesne sayısından fazla olursa evict işlemi devre dışı kalır
- Son erişimden bu yana zaman aşımını **timeToIdleSeconds** belirler
- Veri cache'e eklendikten sonraki zaman aşımı süresini **timeToLiveSeconds** belirler

İkincil Ön Belleğe Programatik Erişim

- **SessionFactory.getCache()** ile **ikincil önbelleğ**e erişilebilir
- **Entity'lerin veya collection ilişkilerin** bu önbellek alanlarında mevcut olup olmadıkları sorgulanabilir
- Önbellekte tutulan entity'ler, collection ilişkileri ve sorgu sonuçları **evict** edilebilir
- Önbellek alanları **toptan temizlenebilir**

İkinci Seviye Ön Belleğe Erişimin Kontrolü

- Hibernate Session üzerinden yapılan işlemlerde veya sorgu düzeyinde **cache etkileşimi** programatik olarak yönetilebilir
- Bu işlem Session düzeyinde **session.setCacheMode()** metodu ile yapılır
- Sorgu düzeyinde ise **query.setCacheMode()** kullanılabilir
- Farklı **CacheMode** türleri mevcuttur

İkinci Seviye Ön Belleğe Erişimin Kontrolü

- **CacheMode.NORMAL**
  - default mode
  - Veri önbellekten okunur, ön belleğe yazılır.
- **CacheMode.IGNORE**
  - Cache'ten okuma yaptırmaz, veri update sırasında cache invalidate ettirilir
- **CacheMode.GET**
  - Veri cache'den okunur, ancak cache'e put yapılmaz, veri update sırasında cache invalidate ettirilir

İkinci Seviye Ön Belleğe Erişimin Kontrolü

- **CacheMode.PUT**
  - Veri cache'e put yapılır, ancak cache'den okunmaz
  - **hibernate.cache.use_minimal_puts** özelliği aktif ise put öncesi verinin cache'de olup olmadığına bakılır
- **CacheMode.REFRESH**
  - PUT moduna benzer
  - Tek farkı **hibernate.cache.use_minimal_puts** özelliğinin gözardı edilmesidir
  - Bu sayede önbellekteki veri her seferinde yenilenmiş olur

İkinci Seviye Ön Belleğe Erişimin Kontrolü

```
Session session = sessionFactory.openSession();
Transaction tx = session.beginTransaction();
session.setCacheMode(CacheMode.IGNORE);

for ( int i=0; i<100000; i++ ) {
    Item item = new Item();
    session.save(item);
    if ( i % 100 == 0 ) {
        session.flush();
        session.clear();
    }
}

tx.commit();
session.close();
```

İletişim

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