Hibernate Second Level Cache Parent Ve Collection İşlemleri

ehcache.xml Ayarları

```xml
<?xml version="1.0" encoding="utf-8"?>
<ehcache xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:noNamespaceSchemaLocation="ehcache.xsd"
         updateCheck="true" monitoring="autodetect"
         dynamicConfig="true">
    <diskStore path="java.io.tmpdir"/>
    <defaultCache
            maxElementsInMemory="1"
            eternal="false"
            timeToIdleSeconds="120"
            timeToLiveSeconds="120"
            overflowToDisk="true"
            />

    <cache name="mucayufa.hibernate.reverseEngineering.Admin"
           maxElementsInMemory="500"
           eternal="true"
           timeToIdleSeconds="100"
           timeToLiveSeconds="100"
           overflowToDisk="false"
            />
    <cache
            name="mucayufa.hibernate.reverseEngineering.Admin.articles"
            maxElementsInMemory="450"
            eternal="false"
            timeToLiveSeconds="600"/>

    <cache name="mucayufa.hibernate.reverseEngineering.Article"
           maxElementsInMemory="500"
           eternal="false"
           timeToIdleSeconds="100"
           timeToLiveSeconds="100"
           overflowToDisk="false"
            />
</ehcache>
```

Not: Ehcache ile ilgili detaylı bilgi için **tıklayınız** (http://www.softwarevol.com/en/tutorial/Hibernate-Ehcache-Usage)
Collection'u da cache'e almak için Admin.articles tarzında kullanmak gereklidir. Burada articles Admin entity içerisindeki collection değişkenidir.

Entity Sınıfı

```java
@Entity
@Table(name = "admin"
        , catalog = "softwarevol"
)
@org.hibernate.annotations.Cache(usage =CacheConcurrencyStrategy.NONSTRICT_READ_WRITE)
public class Admin implements java.io.Serializable {
    @OneToMany(cascade = CascadeType.ALL, fetch = FetchType.LAZY, mappedBy = "admin")
    @org.hibernate.annotations.Cache(usage =CacheConcurrencyStrategy.READ_WRITE)
    public List<Article> getArticles() {
        return this.articles;
    }
}
```

Test İşlemleri

```java
private void useSecondLevelCache() {
    Session session1 = HibernateUtil.getSessionFactory().getCurrentSession();
    session1.beginTransaction();
    Admin admin=(Admin)session1.load(Admin.class,(short)1);
    System.out.println(admin.getArticles().get(0).getTitle());

    session1.getTransaction().commit();
    Session session2 = HibernateUtil.getSessionFactory().getCurrentSession();
    session2.beginTransaction();
    admin=(Admin)session2.load(Admin.class,(short)1);

    admin.setName("denem2e");
    session2.update(admin);

    System.out.println(admin.getArticles().get(0).getTitle());
```

```
16        session1.getTransaction().commit();
17
18        Session session3 = HibernateUtil.getSessionFactory().getCurrentSession();
19        session3.beginTransaction();
20
21        admin=(Admin)session3.load(Admin.class,(short)1);
22        System.out.println(admin.getArticles().get(0).getTitle());
23        session3.getTransaction().commit();
24    }
```

İlk başta(**session1**'de) admin nesnesi veritabanından çekilir. Admin nesnesi ile birlikte Article nesneleri de veritabanından çekileceği için veritabanı sorgusu gerçekleşir. **session2** de ise admin ve ona bağlı article nesnelerini, hibernate **second-level cache**'den getirir. Ayrıca bu session süresinde, admin nesnesi update olduğu için, hibernate tekrar veritabanından admin nesnesini çeker ve **second-level cache**'teki admin nesnesini günceller. Güncelleme esnasında article nesneleri tekrar veritabanından çekilmez; çünkü sadece admin nesnesi güncellenmiştir. **session3**'de hibernate, **second-level cache**'den verileri çeker.

Yukarıdaki örnekta parent entity güncelleniyor. Eğer collection güncelleme(yeni entity ekleme, silme, güncelleme) yapılmış olsa idi şu makaledeki gibi işlem yapmak gerekecekti: **http://planet.jboss.org/post/collection_caching_in_the_hibernate_second_level_cache** (http://planet.jboss.org/post/collection_caching_in_the_hibernate_second_level_cache)