Hibernate 4.0 Ve Üstü Versiyonlarda Event Listener Ekleme

Hibernate **4.0** versiyonundan sonra **event listener** ekleme yöntemi değişmiştir. Bu makelemizde Hibernate 4.0 versiyonu ve üst versiyonlarda event listener nasıl eklenir ondan bahsedeceğiz.

`org.hibernate.integrator.spi.Integrator` **interface** geliştiricilere **SessionFactory** üretilirken, yeni özelliklerin eklenmesi kolaylığı sağlar. Bu özelliklerden birisi de event listener eklemedir. Hibernate'in sağladığı bir servis `org.hibernate.integrator.spi.Integrator` interface'ini implement eden sınıfları, `java.util.ServiceLoader` mekanizmasını kullanarak bulur. Bu işlemin gerçekleşmesi için geliştiricilerin sadece **META-INF/services/org.hibernate.integrator.spi.Integrator** isimli bir dosya oluşturup, bu dosya içerisine `org.hibernate.integrator.spi.Integrator` interface'ini implement eden sınıfları **paket adları** ile beraber her satıra bir tane sınıf adı gelecek şekilde eklemeleri gerekmektedir.

Integrator Interface

```
1  public interface Integrator {
2      /**
3       * Perform integration.
4       *
5       * @param configuration The configuration used to create the session factory
6       * @param sessionFactory The session factory being created
7       * @param serviceRegistry The session factory's service registry
8       */
9      public void integrate(Configuration configuration, SessionFactoryImplementor sessionFactory,
10         SessionFactoryServiceRegistry serviceRegistry);
11
12     /**
13      * Tongue-in-cheek name for a shutdown callback.
14      *
15      * @param sessionFactory The session factory being closed.
16      * @param serviceRegistry That session factory's service registry
17      */
18     public void disintegrate(SessionFactoryImplementor sessionFactory, SessionFactoryServiceRegistry serviceRegistry);
19
20     /**
21      * Ignore this form!  Just do nothing in impl.  It uses the new metamodel api slated for completion in 5.0
22      */
23     public void integrate(MetadataImplementor metadata, SessionFactoryImplementor sessionFactory,
24                 SessionFactoryServiceRegistry serviceRegistry );
25 }
```

integrate() metodu **SessionFactory** üretilirken, bu işleme dahil etmek istediğimiz işlemleri tanımlamayı sağlar. **disintegrate()** metodu ise **SessionFactory** kapatılırken yapılan işlemlere ek olarak yeni işlemlerin eklenmesini sağlar.

Event Listener Ekleme

Event listener eklemek için **integrate()** metodunu kullanacağız. Bu metodta dikkat edersek, **serviceRegistery** parametresi bulunmaktadır. Listener eklerken, bu parametrenin **getService()** metodunu kullanacağız.

Örnek:

```
1  public class MyIntegrator implements Integrator {
2      public void integrate(Configuration configuration, SessionFactoryImplementor sessionFactory,
3           SessionFactoryServiceRegistry serviceRegistry) {
4
5          // EventListenerRegistry servisi cagriliyor. Bu servis araciligi ile listener eklenecek
6          final EventListenerRegistry eventListenerRegistry = serviceRegistry.getService( EventListenerRegistry.class );
7
8           // EventListenerRegistry listener eklemek için 3 yol saglar:
9          //  1) setListeners metodu ile var olan bir listener override edilir.
10         eventListenerRegistry.setListeners( EventType.PRE_LOAD, new MyPreLoadFirstListener());
11         //  2) event turune gore, listener zincirinin ilk halkasını olusturur.
12         eventListenerRegistry.appendListeners( EventType.PRE_LOAD,  new MyPreLoadFirstListener());
13
14         //  3) event turune gore, listener zincirinin son halkasını olusturur.
15         eventListenerRegistry.appendListeners( EventType.PRE_LOAD,  new MyPreLoadSecondListener());
16     }
17
18     @Override
19     public void integrate(MetadataImplementor metadata, SessionFactoryImplementor sessionFactory,
20          SessionFactoryServiceRegistry serviceRegistry) {
21
22     }
23
24     @Override
25     public void disintegrate(SessionFactoryImplementor sessionFactory,
26          SessionFactoryServiceRegistry serviceRegistry) {
27
28     }
29 }
```

MyIntegrator isimli `org.hibernate.integrator.spi.Integrator` interface'ini implement eden bir sınıf tanımladık. Bu sınıfın **integrate()** metodu içerisinde ise event listener'ları set ettik.

Örnekte yer alan **MyPreLoadFirstListener, MyPreLoadSecondListener** isimli sınıflar ise şunlardır:

```java
import org.hibernate.event.spi.PreLoadEvent;
import org.hibernate.event.spi.PreLoadEventListener;

public class MyPreLoadFirstListener implements PreLoadEventListener {
    @Override
    public void onPreLoad(PreLoadEvent event) {
          //entity getEntity metodu ile cagrilir. Admin bir entity sinifidir
          if(event.getEntity() instanceof Admin){
                System.out.println("1");
          }

    }
}

public class MyPreLoadSecondListener implements PreLoadEventListener {
    @Override
    public void onPreLoad(PreLoadEvent event) {
        System.out.println("2");
    }
}
```

Ekran çıktısı 1 1 2 şeklinde olacaktır.

Not: PreLoadEvent sınıfının ve diğer event sınıflarının(PostLoadEvent, PreInsertEvent vs.) **getEntity()** metodu kullanılarak hangi Entity'nin bu event'i meydana getirdiği bulunur. Genelde event listener implementasyonlarında **instanceOf** operatörü kullanılır.

Not: Eklenen servisleri kapatmak için aşağıdaki kodu eklemek gereklidir:

```java
//burada getEventListenerGroup ayni turden olan listener'lari getirir. Tur belirtmek icin EventType enum kullanilir
eventListenerRegistry.getEventListenerGroup(EventType.PRE_LOAD).clear();
```

Son olarak **META-INF** klasörü yaratılır. Bu klasörün içine **services** isimli klasör eklenir. services klasörününe ise **org.hibernate.integrator.spi.Integrator** dosyası eklenir. Bu dosyanın içerisinde **Integrator** interface'ini implement eden sınıf ismi/isimleri yazılır. Örneğimizdeki **MyIntegrator** sınıfı paket adıyla birlikte aşağıdaki gibi yazılmıştır:

```
mucayufa.hibernate.eventListeners.MyIntegrator
```