Alexandre Gama

Software Engineer at Elo7, Instructor at Caelum, Coffee drinker and Beginner Guitar Man

JPA with Entity Listeners and Callback Methods

Hello Everyone!

Today I'll talk about **Entity Listeners** and **Callback Methods** in JPA Spec!

Callback Methods and Entity Listener in JPA?

Imagine that you'll save a new blog post in you database using JPA (with Hibernate for example). Sometimes we need to change or put new data in our record before or after it is saved, right? In most cases we need to do that after certain events, like after save our data, before save our data, before delete some record or before update a data.

How can we solve our problem? Maybe most of you like to use **Triggers** for that. But we have some problems with Triggers. I worked for a company that has used 4 kinds of databases and it was not fun rewrite triggers for all databases. Another problem is you can't test your code! I like to use **TDD** in my projects, so I like to test all my business code.

To solve this problem JPA creates a mode that our code can react to certain events, like we talked above. JPA created **Callback Methods**.
Let's suppose we have the Post class bellow:

```
@Entity
public class Post {

   @Id
   private Long id;
   private String title;
   private String text;
```

```java
    private Date date; //Date? Ok, terrible! Just an example
}
```

Now we need to persist our post:

```java
public class PostRepository {

    public void save(Post post) {
       entityManager.persist(post);
    }

}
```

Great! But we need to save our post with a specific date **before** save it. We can use the @PrePersist annotation:

```java
public class Post {

    @PrePersist
    private void changeDate() {
       this.date = new Date();
    }

}
```

Nice! Now we have a **Callback Method** called **changeDate** that will be executed before persist our object! We have another types of Callback Methods and you can see bellow:

@PrePersist and @PostPersist
@PreUpdate and @PostUpdate
@PreRemove and @PostRemove
@PostLoad

If I have the same callback code for many Entities

Ok! But what if we need to call a Log method (Callback Method) in all of our classes? We will copy the entire code and paste it? It's not sounds good! For that we can create a specific class with that log code and use it in our classes. We can have our Log class like bellow using @PrePersist:

```java
public class LogListener {

    @PrePersist
    private void log(Object object) {
        System.out.println("Your log code here");
    }

}
```

Did you note that we received an object? Yes, we receive the object that has being saved at the moment! After that we need to use our LogListener class using the @EntityListeners annotation:

```java
@EntityListeners(LogListener.class)
public class Post {

    @Id
    private Long id;

    private String title;

    private String text;

    private Date date; //Date? Ok, terrible! Just an example

}
```

Awesome!

Callback Method curiosities

The @EntityListeners allow us to use multiple classes like @EntityListeners({Log.class, Audit.class, YourClass.class}). Nice!

Also, one method can be annotated with more than one **Callback annotation:**

```java
public class LogListener {

    @PrePersist
    @PreUpdate
    @PostRemove
    private void log(Object object) {
        System.out.println("Your log code here");
    }

}
```

Can I have two methods annotated with the same callback annotation?

No! You cannot have two methods being annotated by the same callback annotation. See the next code:

```java
public class LogListener {

    @PreUpdate
    @PostRemove
    private void log(Object object) {
        System.out.println("Your log code here");
    }
    @PreUpdate
    private void logAgain(Object object) {
    }
}
```

The code above is not possible because we have two @PreUpdate annotation and an exception will be throws in our face :)

Can I have two Listener Classes with methods with the same callback annotation?

Sure! You can see that now:

```java
public class LogListener {

    @PrePersist //Log class with PrePersist
    private void log(Object object) {
        System.out.println("Your log code here");
    }

}
```

```java
public class AuditListener {

    @PrePersist //Audit class with PrePersist
    private void audit(Object object) {
        System.out.println("Your audit code here");
    }
}
```

```java
@EntityListeners({LogListener.class, AuditListener.class}) //using bot
public class Post {

    @Id
    private Long id;

    private String title;

    private String text;

    private Date date;

}
```

After you persist you object you will see the follow:

```
Your log code here
Your audit code here
```

Yes, they will be printed following the sequence of the classes in the @EntityListeners.

I love XML! Can I use it?

Sure! You can indicate your callback methods in your **persistence.xml**

```xml
<entity-listeners>
    <entity-listener class="mypackage.LogListener">
        <pre-persist method-name="log"/>
    </entity-listener>
</entity-listeners>
```

That's it! See you soon!

Posted in Hibernate, JavaEE, JPA and tagged CallbackMethod, EntityListener, Hibernate, JPA on 23/03/2014 by Alexandre Gama. 3 Comments

3 comments

1. **rponte** says:
   23/03/2014 at 14:36
   Are you sure about a method callback receiving an entity as argument? If I'm not wrong, It's not possible.

   REPLY

   1. Alexandre Gama says:
      23/03/2014 at 15:34
      You're right Ponte! I wrote in Post class not in a LogListener. I'll change the class name! Thanks man!

      REPLY

2. **Johnd499** says:
   06/06/2014 at 21:40
   Great blog! I am loving it!! Will come back again. I am taking your feeds also dfkkdceafcgb

   REPLY