Hibernate 5 - Interceptor example

In Hibernate, interceptors are used to inspect the changes in entity’s property values before they are written and after they are read from a database.

You can use the Hibernate interceptor to perform the various operations such as logging, auditing, profiling etc.

In Hibernate, an interceptor can be either Session-scoped or SessionFactory-scoped.

Session-scoped interceptors are used when a Session is opened. The following code snippet shows the how to add an interceptor to a Session.

Session session = HibernateUtil.getSessionFactory()

.withOptions()

.interceptor(**new** LoggingInterceptor())

.openSession();

SessionFactory-scoped or global interceptors are used when SessionFactory is configured and these interceptors will be applied to applied to all Session opened from that SessionFactory. The following code snippet shows the how to add an interceptor to a SessionFactory.

SessionFactory sessionFactory = metadata.getSessionFactoryBuilder()

.applyInterceptor(**new** LoggingInterceptor())

.build();

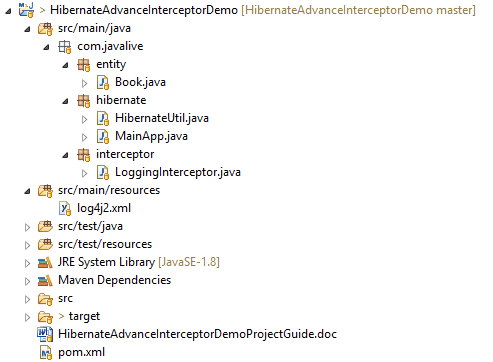
The following example demonstrates how to implement a custom interceptor to log the SQL queries and entity’s property values when an entity is saved.

Tools and technologies used for this application are -

* Hibernate ORM 5.2.12.Final
* Log4j 2.8.2
* JavaSE 1.8
* MySQL Server 5.7.12
* Eclipse Oxygen.1

Project structure

Final project structure of our application will look like as follows.



Jar dependencies

Open pom.xml file of your maven project, add the dependencies below.

<dependencies>

<dependency>

<groupId>org.hibernate</groupId>

<artifactId>hibernate-core</artifactId>

<version>5.2.12.Final</version>

</dependency>

<dependency>

<groupId>mysql</groupId>

<artifactId>mysql-connector-java</artifactId>

<version>6.0.6</version>

</dependency>

<dependency>

<groupId>org.apache.logging.log4j</groupId>

<artifactId>log4j-core</artifactId>

<version>2.8.2</version>

</dependency>

<dependency>

<groupId>org.apache.logging.log4j</groupId>

<artifactId>log4j-api</artifactId>

<version>2.8.2</version>

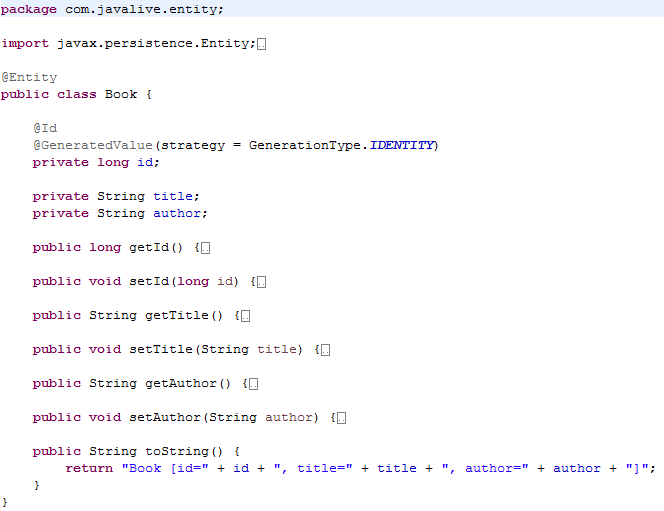
</dependency>

</dependencies>

 Entity class

Create a simple @Entity class as follows.

**Book.java**



Hibernate Interceptor

To create an interceptor, you can either implement the org.hibernate.Interceptor or extend the org.hibernate.EmptyInterceptor class.

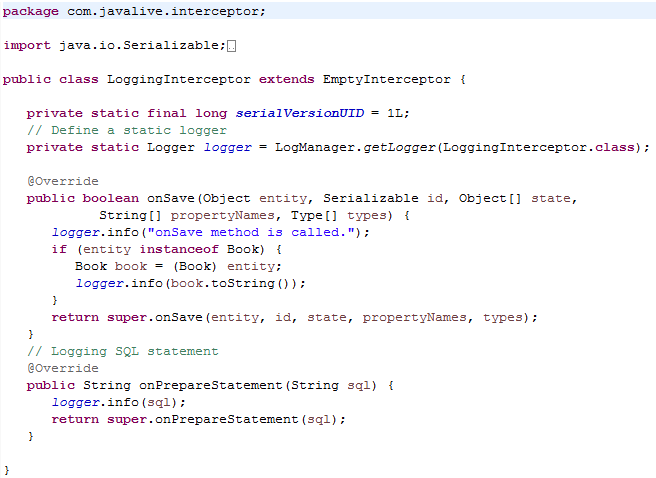
The Interceptor interface contains various methods as follows.

| **Methods** | **Description** |
| --- | --- |
| onSave() | Called before an object is saved. |
| onLoad() | Called just before an object is initialized. |
| onDelete() | Called before an object is deleted. |
| preFlush() | Called before a flush. |
| postFlush() | Called after a flush. |

For all available methods in Interceptor, you can refer-  [Hibernate JavaDoc](http://docs.jboss.org/hibernate/orm/current/javadocs/org/hibernate/Interceptor.html).

Create a LoggingInterceptor by extending the EmptyInterceptor class and override the onSave() method to log an entity when it is saved.

**LoggingInterceptor.java**

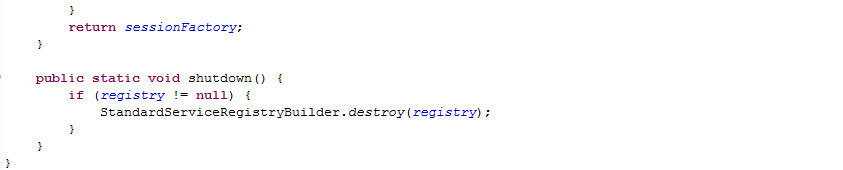
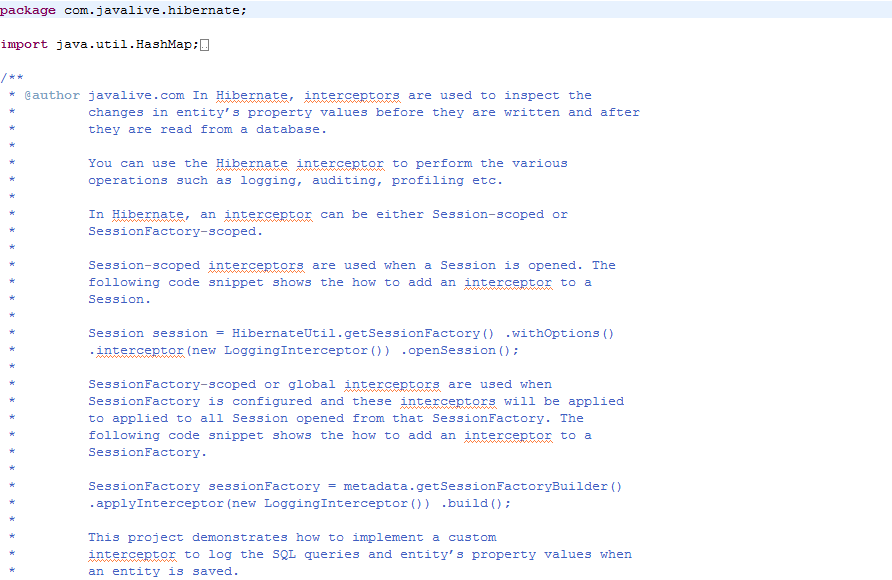


Hibernate Utility class

Create a helper class HibernateUtil to bootstrap hibernate.

Map the Book entity using the #MetadataSources.addAnnotatedClass() method.

**HibernateUtil.java**



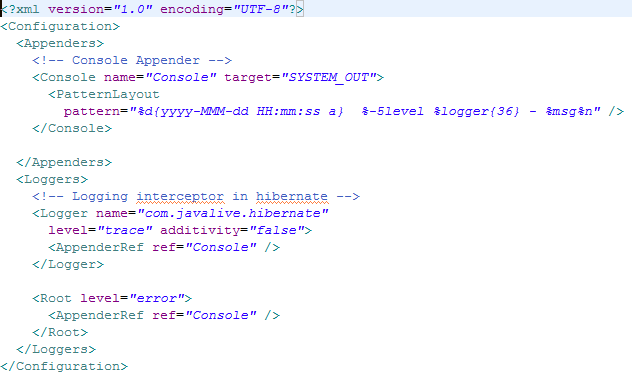
If you want to use the LoggerInterceptor as a SessionFactory-scoped interceptor, just uncomment the lines from HibernateUtil class.

Hibernate Interceptor logging

In this example, we will use the log4j 2 API to log the overridden methods of a custom interceptor.

Create a log4j2.xml file under src/main/resources folder and write the following code in it.

**log4j2.xml**

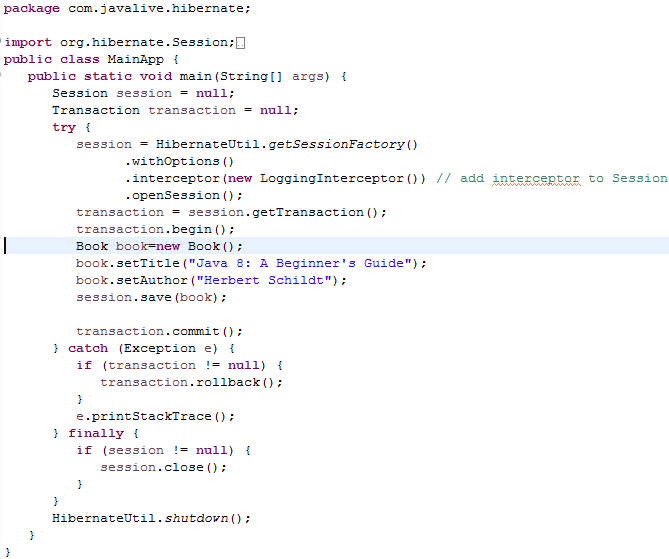


Run Application

Create the MainApp class to test the custom interceptor class.

You need to specify the LoggerInterceptor when a Session is opened as follows.

**MainApp.java**



After executing the MainApp, the output of your program will look like as follows.

