Ravi Oza

Hibernate Application in Eclipse

For creating the first hibernate application in Eclipse IDE, we need to follow following steps:

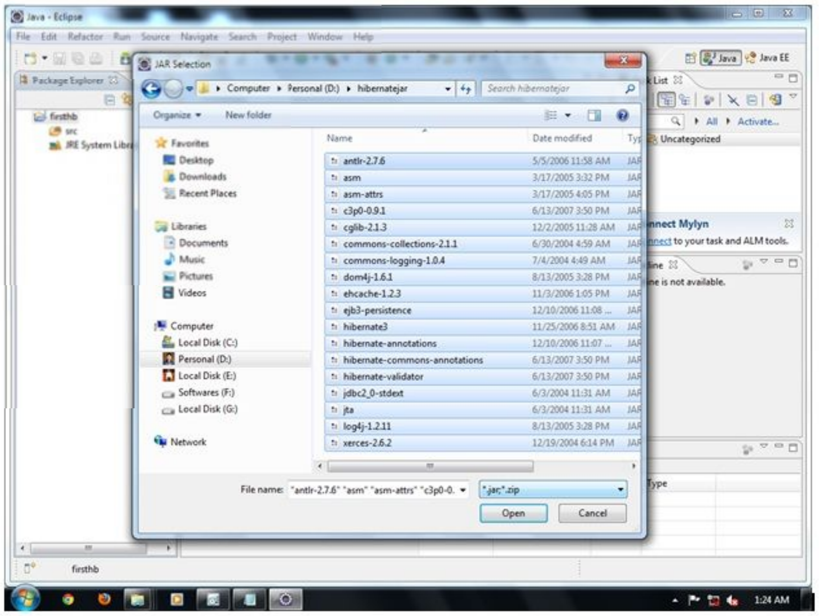
1. Create the java project
2. Add jar files for hibernate
3. Create the Persistent class
4. Create the mapping file for Persistent class
5. Create the Configuration file
6. Create the class that retrieves or stores the persistent object
7. Run the application

1) Create the java project

Create the java project by **File Menu** - **New** - **project** - **java project** . Now specify the project name e.g. firsthb then **next** - **finish** .

2) Add jar files for hibernate

To add the jar files **Right click on your project** - **Build path** - **Add external archives**. Now select all the jar files as shown in the image given below then click open.



Name	Date modified	Type
antlr-2.7.6	5/5/2006 11:58 AM	JAR
asm	3/17/2005 3:32 PM	JAR
asm-attrs	3/17/2005 4:05 PM	JAR
c3p0-0.9.1	6/13/2007 3:50 PM	JAR
cglib-2.1.3	12/2/2005 11:28 AM	JAR
commons-collections-2.1.1	6/30/2004 4:59 AM	JAR
commons-logging-1.0.4	7/4/2004 4:49 AM	JAR
dom4j-1.6.1	8/13/2005 3:28 PM	JAR
ehcache-1.2.3	11/3/2006 1:05 PM	JAR
ejb3-persistence	12/10/2006 11:08 ...	JAR
hibernate3	11/25/2006 8:51 AM	JAR
hibernate-annotations	12/10/2006 11:07 ...	JAR
hibernate-commons-annotations	6/13/2007 3:50 PM	JAR
hibernate-validator	6/13/2007 3:50 PM	JAR
jdbc2_0-stdext	6/3/2004 11:31 AM	JAR
jta	6/3/2004 11:31 AM	JAR
log4j-1.2.11	8/13/2005 3:28 PM	JAR
xerces-2.6.2	12/19/2004 6:14 PM	JAR

File name: "antlr-2.7.6" "asm" "asm-attrs" "c3p0-0.

.jar;.zip

In this example, we are connecting the application with oracle database. So you must add the ojdbc14.jar file.

3) Create the Persistent class

Here, we are creating the same persistent class which we have created in the previous topic. To create the persistent class, Right click on src - New - Class - specify the class with package name (e.g. hiber) - finish.

A simple Persistent class should follow some rules:

- **A no-arg constructor:** It is recommended that you have a default constructor at least package visibility so that hibernate can create the instance of the Persistent class by newInstance() method.
- **Provide an identifier property (optional):** It is mapped to the primary key column of the database.
- **Declare getter and setter methods (optional):** The Hibernate recognizes the method by getter and setter method names by default.
- **Prefer non-final class:** Hibernate uses the concept of proxies, that depends on the persistent class. The application programmer will not be able to use proxies for lazy association fetching.

Employee.java

```
package hiber;

public class Employee
{
      private int id;
      private String firstName,lastName;

      public int getId()
      {     return id;  }
      public void setId(int id)
      {      this.id = id;  }
      public String getFirstName()
      {      return firstName;  }
      public void setFirstName(String firstName)
      {      this.firstName = firstName;  }
      public String getLastName()
      {      return lastName;  }
      public void setLastName(String lastName)
      {      this.lastName = lastName;  }
}
```

4) Create the mapping file for Persistent class

Here, we are creating the same mapping file as created in the previous topic. To create the mapping file, Right click on src - new - file - specify the file name (e.g. employee.hbm.xml) - ok. It must be outside the package.

The mapping file name conventionally, should be class_name.hbm.xml. There are many elements of the mapping file.

- **hibernate-mapping** is the root element in the mapping file.
- **class** It is the sub-element of the hibernate-mapping element. It specifies the Persistent class.
- **id** It is the subelement of class. It specifies the primary key attribute in the class.
- **generator** It is the subelement of id. It is used to generate the primary key. There are many generator classes such as assigned (It is used if id is specified by the user), increment, hilo, sequence, native etc. We will learn all the generator classes later.
- **property** It is the subelement of class that specifies the property name of the Persistent class.

employee.hbm.xml

```
<?xml version='1.0' encoding='UTF-8'?>
<!DOCTYPE hibernate-mapping PUBLIC
 "-//Hibernate/Hibernate Mapping DTD 3.0//EN"
 "http://hibernate.sourceforge.net/hibernate-mapping-3.0.dtd">

 <hibernate-mapping>
  <class name="hiber.Employee" table="emp1000">
    <id name="id">
     <generator class="assigned"></generator>
    </id>

    <property name="firstName"></property>
    <property name="lastName"></property>
  </class>

 </hibernate-mapping>
```

5) Create the Configuration file

The configuration file contains informations about the database and mapping file. Conventionally, its name should be hibernate.cfg.xml

The configuration file contains all the informations for the database such as connection_url, driver_class, username, password etc. The hbm2ddl.auto property is used to create the table in the database automatically. We will have in-depth learning about Dialect class in next topics. To create the configuration file, right click on src - new - file. Now specify the configuration file name e.g. hibernate.cfg.xml.

hibernate.cfg.xml

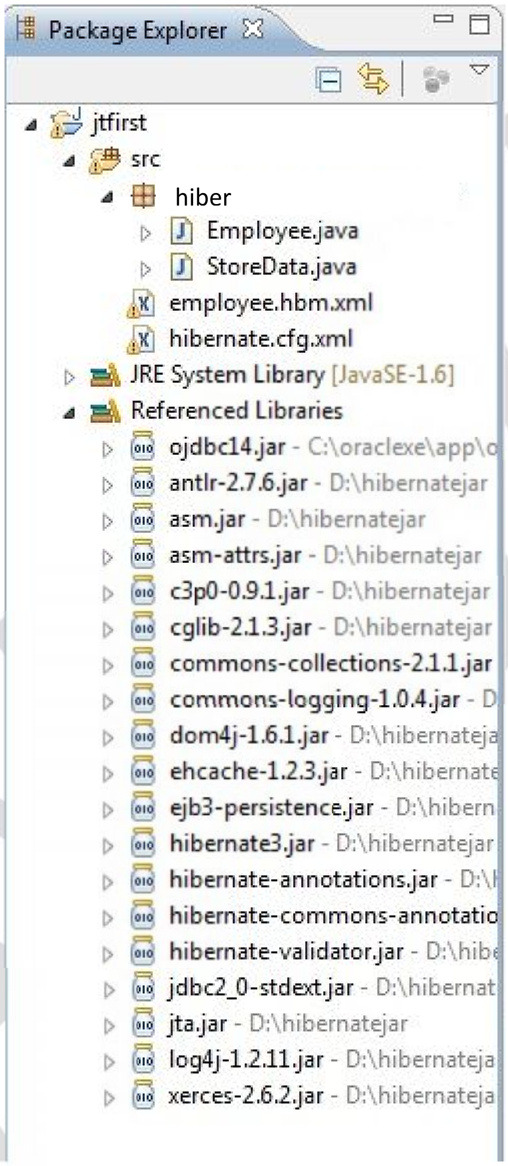
```
<?xml version='1.0' encoding='UTF-8'?>
<!DOCTYPE hibernate-configuration PUBLIC
          "-//Hibernate/Hibernate Configuration DTD 3.0//EN"
          "http://hibernate.sourceforge.net/hibernate-configuration-
3.0.dtd">

<hibernate-configuration>

 <session-factory>
      <property name="hbm2ddl.auto">update</property>
      <property name="dialect">org.hibernate.dialect.Oracle9Dialect
      </property>
      <property name="connection.url">jdbc:oracle:thin:@localhost:1521:xe
      </property>
      <property name="connection.username">system</property>
      <property name="connection.password">oracle</property>
     <property name="connection.driver_class">oracle.jdbc.driver.OracleDriver>
      </property>
     <mapping resource="employee.hbm.xml"/>
     </session-factory>

</hibernate-configuration>
```

6) Create the class that retrieves or stores the persistent object

In this class, we are simply storing the employee object to the database.

```
package hiber;

import org.hibernate.Session;
import org.hibernate.SessionFactory;
import org.hibernate.Transaction;
import org.hibernate.cfg.Configuration;

public class StoreData{
public static void main(String[] args) {

    //creating configuration object
    Configuration cfg=new Configuration();
    cfg.configure("hibernate.cfg.xml");
      //populates the data of the configuration file

    //creating seession factory object
    SessionFactory factory=cfg.buildSessionFactory();

    //creating session object
    Session session=factory.openSession();

    //creating transaction object
    Transaction t=session.beginTransaction();

    Employee e1=new Employee();
    e1.setId(115);
    e1.setFirstName("sonoo");
    e1.setLastName("jaiswal");

    session.persist(e1);//persisting the object

    t.commit();//transaction is committed
    session.close();

    System.out.println("successfully saved");
}
}
```

7) Run the application



```
jtfirst
  src
    hiber
      Employee.java
      StoreData.java
    employee.hbm.xml
    hibernate.cfg.xml
  JRE System Library [JavaSE-1.6]
  Referenced Libraries
    ojdbc14.jar - C:\oraclexe\app\o
    antlr-2.7.6.jar - D:\hibernatejar
    asm.jar - D:\hibernatejar
    asm-attrs.jar - D:\hibernatejar
    c3p0-0.9.1.jar - D:\hibernatejar
    cglib-2.1.3.jar - D:\hibernatejar
    commons-collections-2.1.1.jar
    commons-logging-1.0.4.jar - D
    dom4j-1.6.1.jar - D:\hibernateja
    ehcache-1.2.3.jar - D:\hibernate
    ejb3-persistence.jar - D:\hibern
    hibernate3.jar - D:\hibernatejar
    hibernate-annotations.jar - D:\
    hibernate-commons-annotatio
    hibernate-validator.jar - D:\hibe
    jdbc2_0-stdext.jar - D:\hibernat
    jta.jar - D:\hibernatejar
    log4j-1.2.11.jar - D:\hibernateja
    xerces-2.6.2.jar - D:\hibernateja
```

To run the hibernate application, right click on the StoreData class - Run As - Java Application.