Example to create the Hibernate Application in Eclipse IDE

Here, we are going to create a simple example of hibernate application using eclipse IDE. For creating the first hibernate application in Eclipse IDE, we need to follow following steps:

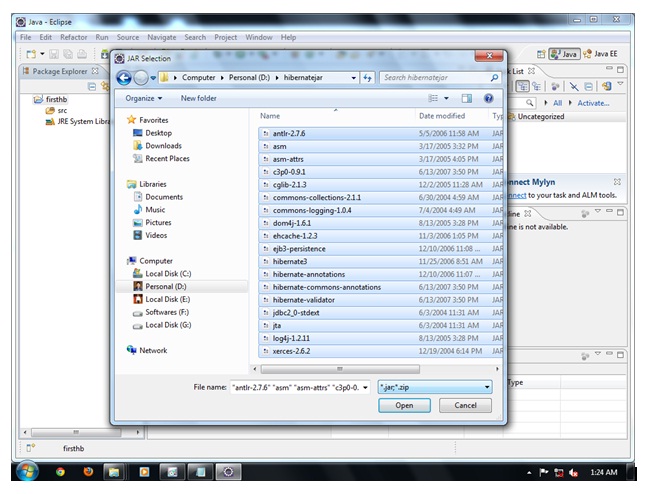
1. Create the java project
2. Add jar files for hibernate
3. Create the Persistent class
4. Create the mapping file for Persistent class
5. Create the Configuration file
6. Create the class that retrieves or stores the persistent object
7. Run the application

### 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. | |
|  |



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. com.pratik.mypackage) - finish.

#### Employee.java

1. **package** com.pratik.mypackage;
3. **public** **class** Employee {
4. **private** **int** id;
5. **private** String firstName,lastName;
7. **public** **int** getId() {
8. **return** id;
9. }
10. **public** **void** setId(**int** id) {
11. **this**.id = id;
12. }
13. **public** String getFirstName() {
14. **return** firstName;
15. }
16. **public** **void** setFirstName(String firstName) {
17. **this**.firstName = firstName;
18. }
19. **public** String getLastName() {
20. **return** lastName;
21. }
22. **public** **void** setLastName(String lastName) {
23. **this**.lastName = lastName;
24. }

27. }

### 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.

#### employee.hbm.xml

1. <?xml version='1.0' encoding='UTF-8'?>
2. <!DOCTYPE hibernate-mapping PUBLIC
3. "-//Hibernate/Hibernate Mapping DTD 3.0//EN"
4. "http://hibernate.sourceforge.net/hibernate-mapping-3.0.dtd">
6. <hibernate-mapping>
7. <**class** name="com.pratik.mypackage.Employee" table="emp">
8. <id name="id">
9. <generator **class**="increment"></generator>
10. </id>
12. <property name="firstName"></property>
13. <property name="lastName"></property>
15. </**class**>
17. </hibernate-mapping>

### 5) Create the Configuration file

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

1. <?xml version='1.0' encoding='UTF-8'?>
2. <!DOCTYPE hibernate-configuration PUBLIC
3. "-//Hibernate/Hibernate Configuration DTD 3.0//EN"
4. "http://hibernate.sourceforge.net/hibernate-configuration-3.0.dtd">
6. <hibernate-configuration>
8. <session-factory>
9. <property name="hbm2ddl.auto">update</property>
10. <property name="dialect">org.hibernate.dialect.Oracle10gDialect</property>
11. <property name="connection.url">jdbc:oracle:thin:@localhost:1521:xe</property>
12. <property name="connection.username">system</property>
13. <property name="connection.password">oracle</property>
14. <property name="connection.driver\_class">oracle.jdbc.driver.OracleDriver</property>
15. <mapping resource="employee.hbm.xml"/>
16. </session-factory>
18. </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.

1. **package** com.pratik.mypackage;
3. **import** org.hibernate.Session;
4. **import** org.hibernate.SessionFactory;
5. **import** org.hibernate.Transaction;
6. **import** org.hibernate.cfg.Configuration;
8. **public** **class** StoreData {
9. **public** **static** **void** main(String[] args) {
11. //creating configuration object
12. Configuration cfg=**new** Configuration();
13. cfg.configure("hibernate.cfg.xml");//populates the data of the configuration file
15. //creating seession factory object
16. SessionFactory factory=cfg.buildSessionFactory();
18. //creating session object
19. Session session=factory.openSession();
21. //creating transaction object
22. Transaction t=session.beginTransaction();
24. Employee e1=**new** Employee();
25. e1.setId(1001);
26. e1.setFirstName("pratik");
27. e1.setLastName("dimble");
29. session.persist(e1);//persisting the object
31. t.commit();//transaction is committed
32. session.close();
34. System.out.println("successfully saved");
36. }
37. }

### 7) Run the application

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