**Basic Hibernate Application :**

In this article, we are going to create a simple example of hibernate application using eclipse IDE.After finishing of this article You will learn how to create and run the Hibernate based projects in Eclipse IDE, via XML mapping file (hbm).

Tools and Technologies :

* JDK 1.6 /1.7/1.8
* Hibernate 3.6.3.Final
* Eclipse
* MySQL 5.0/5.5/5.7.

For developing first hibernate application in Eclipse, we need to follow bellow steps:

Step 1 : Create the java project :

Open Eclipse IDE,click on File-> New -> project -> Java and select java project.Now specify the project name as FirstHibernateEx then click on next -> finish .

* Hibernate Tutorial
* Hibernate Basics
  + ORM Introduction
  + Introduction To Hibernate Framework
  + Hibernate Configuration File
  + Hibernate Application In Eclipse IDE
  + Object States In Hibernate – Transient,Persistent And Detached
  + Difference Between Hibernate Save(),SaveOrUpdate() And Persist() Methods
  + Web Application With Hibernate
  + Hibernate Example With Annotation
  + Integrating JSP,Servlet And Hibernate In An MVC Application
  + CRUD Operations Using Hibernate 3
* Hibernate Inheritance Mapping
  + Inheritance Mapping In Hibernate
  + Hibernate Inheritance: Table Per Class Hierarchy
  + Hibernate Inheritance: Table Per Subclass Hierarchy
  + Hibernate Inheritance: Table Per Concrete Class Hierarchy
* Hibernate Query Language (HQL)
  + Hibernate Query Language (HQL) Example Tutorial
  + HQL Tutorial HQL Select, Update And Delete Queries
* Composite Primary Keys In Hibernate

Basic Hibernate Application :

In this article, we are going to create a simple example of hibernate application using eclipse IDE.After finishing of this article You will learn how to create and run the Hibernate based projects in Eclipse IDE, via XML mapping file (hbm).

Tools and Technologies :

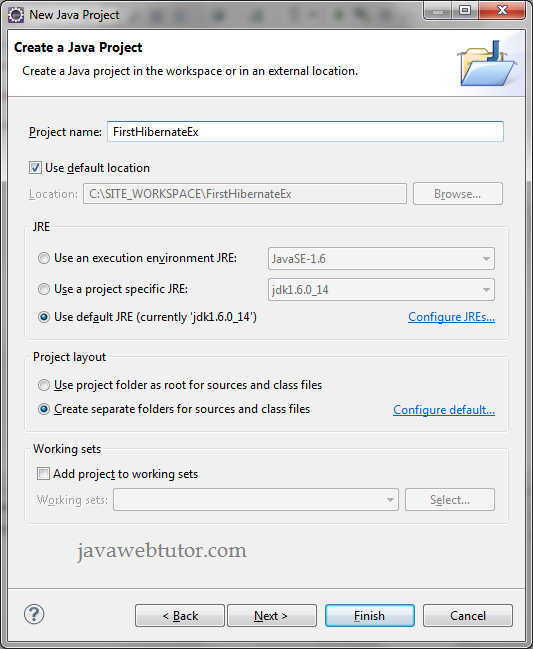
* JDK 1.6
* Hibernate 3.6.3.Final
* Eclipse
* MySQL 5.0.

For developing first hibernate application in Eclipse, we need to follow bellow steps:

Step 1 : Create the java project :

Open Eclipse IDE,click on File-> New -> project -> Java and select java project.Now specify the project name as FirstHibernateEx then click on next -> finish .

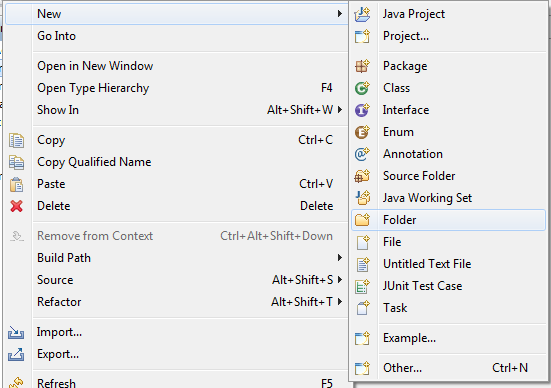
* Hibernate Tutorial
* Hibernate Basics
  + ORM Introduction
  + Introduction To Hibernate Framework
  + Hibernate Configuration File
  + Hibernate Application In Eclipse IDE
  + Object States In Hibernate – Transient,Persistent And Detached
  + Difference Between Hibernate Save(),SaveOrUpdate() And Persist() Methods
  + Web Application With Hibernate
  + Hibernate Example With Annotation
  + Integrating JSP,Servlet And Hibernate In An MVC Application
  + CRUD Operations Using Hibernate 3
* Hibernate Inheritance Mapping
  + Inheritance Mapping In Hibernate
  + Hibernate Inheritance: Table Per Class Hierarchy
  + Hibernate Inheritance: Table Per Subclass Hierarchy
  + Hibernate Inheritance: Table Per Concrete Class Hierarchy
* Hibernate Query Language (HQL)
  + Hibernate Query Language (HQL) Example Tutorial
  + HQL Tutorial HQL Select, Update And Delete Queries
* Composite Primary Keys In Hibernate



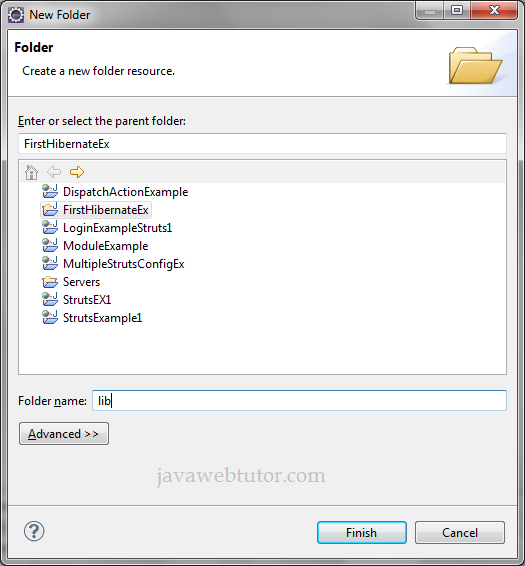
Step 2 : Add Jar files for hibernate and mysql :

First we need to create one folder in the project, so that we can place all required jar files in one place.

Right Click on your Project and select New -> Folder as shown below



In New Folder screen provide Folder name as lib(Dont confuse with lib we can assign any name to the folder) and Click Finish as shown below .



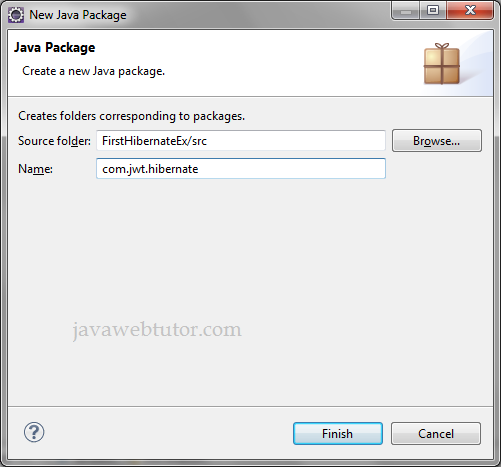
Download the latest hibernate jar files from Here. After downloading the Jar files copy following Jar files inside the lib folder which we created earlier.

* antlr-2.7.6
* commons-collections-3.1
* dom4j-1.6.1
* hibernate3
* javassist-3.4.GA
* jta-1.1
* slf4j-api-1.5.6
* slf4j-simple-1.5.6
* mysql

Now add these jar files to classpath of your project.

Step 3 : Create the persistence class :

Create a package by Right Click on your Project and navigate to New -> Package and provide the name com.jwt.hibernate to the package and click on Finish as shown bellow.



Now create a java class **Student.java** in this package by right click on the **package** -> **New** -> **Class** and add following code in this class.

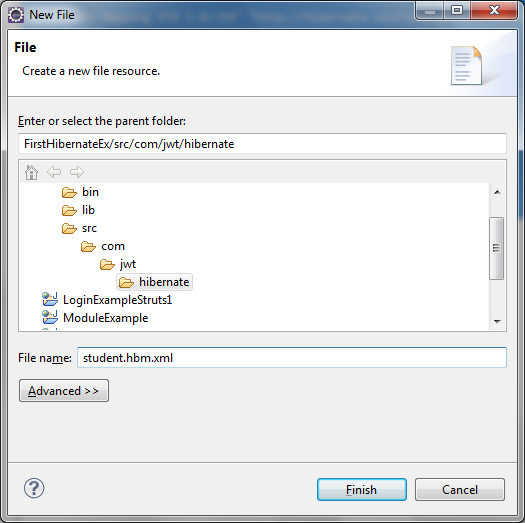
**Student.java**

?

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40 | package com.jwt.hibernate;    public class Student {  private long id;  private String name;  private String degree;  private String roll;  private String phone;    public long getId() {  return id;  }  public void setId(long id) {  this.id = id;  }  public String getName() {  return name;  }  public void setName(String name) {  this.name = name;  }  public String getDegree() {  return degree;  }  public void setDegree(String degree) {  this.degree = degree;  }  public String getRoll() {  return roll;  }  public void setRoll(String roll) {  this.roll = roll;  }  public String getPhone() {  return phone;  }  public void setPhone(String phone) {  this.phone = phone;  }  } |

Step 4 : Create the mapping file for Persistent class :

Right click on your package then navigate to New -> File and provide the name as student.hbm.xml and click Finish.



Add bellow code into student.hbm.xml file

**student.hbm.xml**

?

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11 | <?xml version="1.0" encoding="UTF-8"?>  <!DOCTYPE hibernate-mapping PUBLIC "-//Hibernate/Hibernate Mapping DTD 3.0//EN" "<a class="vglnk" href="http://hibernate.sourceforge.net/hibernate-mapping-3.0.dtd" rel="nofollow"><span>http</span><span>://</span><span>hibernate</span><span>.</span><span>sourceforge</span><span>.</span><span>net</span><span>/</span><span>hibernate</span><span>-</span><span>mapping</span><span>-</span><span>3</span><span>.</span><span>0</span><span>.</span><span>dtd</span></a>">  <hibernate-mapping>  <class name="com.jwt.hibernate.Student" table="STUDENT">  <id column="ID" name="id" type="long" />  <property column="STUDENT\_NAME" name="name" type="string" />  <property column="DEGREE" name="degree" type="string" />  <property column="ROLL" name="roll" type="string" />  <property column="PHONE" name="phone" type="string" />  </class>  </hibernate-mapping> |

In this mapping file, Student class is linked with STUDENT table in the database, and next is the id element, means in the database table what column we need to take as primary key column, that property name we need to give here,I have been given my property name id which will mapped with ID column in the table, name is mapped with STUDENT\_NAME column ,degree is mapped with DEGREE column of the STUDENT table and so on.

**Note :-**If your database column name and persistence class variable is same then no need to specify column property in .hbm file.

Step 5 : Create the Configuration file :

The configuration file contains informations about the database and mapping file. Conventionally, its name should be hibernate.cfg.xml .Configuration file must be in classpath of your Project.Place this file in src of your project by default it will added to classpath of your project.

**hibernate.cfg.xml**

?

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18 | <?xml version="1.0" encoding="utf-8"?>  <!DOCTYPE hibernate-configuration PUBLIC  "-//Hibernate/Hibernate Configuration DTD 3.0//EN"  "<a class="vglnk" href="http://www.hibernate.org/dtd/hibernate-configuration-3.0.dtd" rel="nofollow"><span>http</span><span>://</span><span>www</span><span>.</span><span>hibernate</span><span>.</span><span>org</span><span>/</span><span>dtd</span><span>/</span><span>hibernate</span><span>-</span><span>configuration</span><span>-</span><span>3</span><span>.</span><span>0</span><span>.</span><span>dtd</span></a>">    <hibernate-configuration>  <session-factory>  <property name="hibernate.connection.driver\_class">com.mysql.jdbc.Driver</property>  <property name="hibernate.connection.url">jdbc:mysql://localhost:3306/javawebtutor</property>  <property name="hibernate.connection.username">root</property>  <property name="hibernate.connection.password">mukesh</property>  <property name="hibernate.dialect">org.hibernate.dialect.MySQLDialect</property>  <property name="show\_sql">true</property>  <property name="format\_sql">true</property>  <property name="hbm2ddl.auto">create </property>  <mapping resource="com/jwt/hibernate/student.hbm.xml" />  </session-factory>  </hibernate-configuration> |

**Note** :- In the above configuration file we mentioned **<property name="hbm2ddl.auto">create </property>**.If we are assigning this value as create,it will create table in database automatically when we run the application.So no need to write sql query for table creation.But when you are running the example second times you need to change this attribute to "update" otherwise hibernate will throw exception.

Step 6: Create the Test class that stores the object :

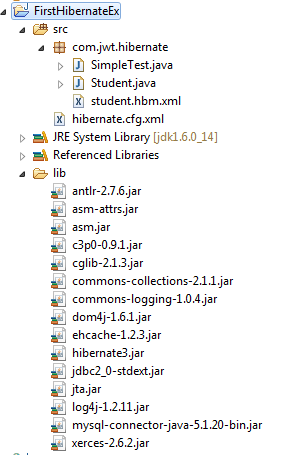
In this class, we are simply storing the student object to the database.

**SimpleTest.java**

?

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30 | package com.jwt.hibernate;    import org.hibernate.Session;  import org.hibernate.SessionFactory;  import org.hibernate.Transaction;  import org.hibernate.cfg.Configuration;    public class SimpleTest {        public static void main(String[] args) {            Configuration cfg = new Configuration();          cfg.configure("hibernate.cfg.xml");            SessionFactory factory = cfg.buildSessionFactory();          Session session = factory.openSession();          Student student = new Student();          student.setName("Mukesh");          student.setRoll("101");          student.setPhone("8888");          student.setDegree("B.E");            Transaction tx = session.beginTransaction();          session.save(student);          System.out.println("Object saved successfully.....!!");          tx.commit();          session.close();          factory.close();      }  } |

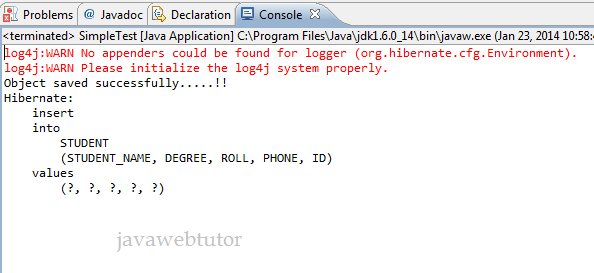
Directory structure of the project :



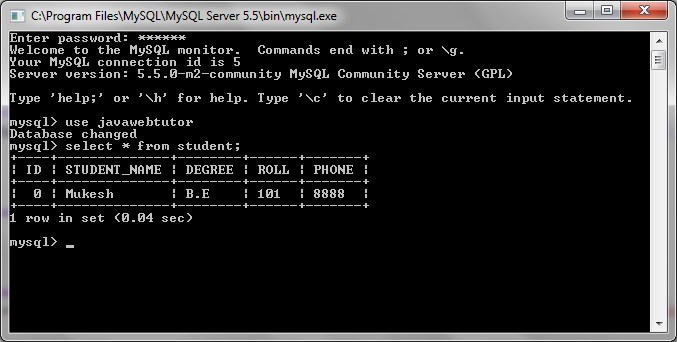
Run the Application :

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

Output in Eclipse :



In The Database :



You can download the source code of the example by clicking on the Download link below

You should add the following code in the "Employee.java" file:

package net.roseindia.model;

import java.io.Serializable;

import javax.persistence.Column;

import javax.persistence.Entity;

import javax.persistence.GeneratedValue;

import javax.persistence.Id;

import javax.persistence.Table;

/\*\*

\* @author Deepak Kumar

\* Web: http://www.roseindia.net

\*/

@Entity

@Table(name = "employee")

public class Employee implements Serializable{

@Id

@GeneratedValue

@Column(name="id")

private int id;

@Column(name="emp\_name")

private String empName;

@Column(name="emp\_address")

private String empAddress;

@Column(name="emp\_mobile\_nos")

private String empMobileNos;

public int getId() {

return id;

}

public void setId(int id) {

this.id = id;

}

public String getEmpName() {

return empName;

}

public void setEmpName(String empName) {

this.empName = empName;

}

public String getEmpAddress() {

return empAddress;

}

public void setEmpAddress(String empAddress) {

this.empAddress = empAddress;

}

public String getEmpMobileNos() {

return empMobileNos;

}

public void setEmpMobileNos(String empMobileNos) {

this.empMobileNos = empMobileNos;

}

}

Above file id annotated with the JPA annotations and maps the entity with the "employee" table of the database.

We have used following annotations in the model class:

**org.hibernate.annotations.Table:** for mapping it to a table

**javax.persistence.Column:** for mapping the entity field with the table column

**javax.persistence.Entity:** for specifying a class as an entity class

**javax.persistence.GeneratedValue:** primary key generation strategy.

**javax.persistence.Id:** The primary key of the entity

**javax.persistence.Table:** for mapping it to the database table

**Step 7:** Add the **hibernate.cfg.xml** file to the project. Create a new file "**hibernate.cfg.xml**" in the src folder of the project as shown below:

Add the following into the **hibernate.cfg.xml** file:

<?xml version='1.0' encoding='utf-8'?>

<!DOCTYPE hibernate-configuration PUBLIC

"-//Hibernate/Hibernate Configuration DTD//EN"

"http://hibernate.sourceforge.net/hibernate-configuration-3.0.dtd">

<hibernate-configuration>

<session-factory>

<property name="hibernate.connection.driver\_class">com.mysql.jdbc.Driver</property>

<property name="hibernate.connection.url">jdbc:mysql://localhost:3306/hibernate4</property>

<property name="hibernate.connection.username">root</property>

<property name="hibernate.connection.password">root</property>

<property name="hibernate.connection.pool\_size">10</property>

<property name="show\_sql">true</property>

<property name="dialect">org.hibernate.dialect.MySQLDialect</property>

<property name="hibernate.current\_session\_context\_class">thread</property>

<mapping class="net.roseindia.model.Employee" />

</session-factory>

</hibernate-configuration>

**Step 8:** Add MySQL Driver library

Download the latest MySQL JDBC driver from <http://dev.mysql.com/downloads/connector/j/> and then add the mysql-connector-java-5.1.27-bin.jar file into the project lib directory.  Also add this jar file into java build path.

**Step 9:** Add the Java file for initializing the Hibernate ORM

Create a new Java file named "HibernateUtil.java" as shown below:

Here is the code of the Java file which is used to initialize the ORM environment and returns the SessionFactory object to the application:

package net.roseindia;

import org.hibernate.SessionFactory;

import org.hibernate.cfg.Configuration;

import org.hibernate.service.ServiceRegistry;

import org.hibernate.service.ServiceRegistryBuilder;

/\*\*

\* @author Deepak Kumar

\* Web: http://www.roseindia.net

\*/

public class HibernateUtil {

private static final SessionFactory sessionFactory;

private static ServiceRegistry serviceRegistry;

static {

try {

Configuration configuration = new Configuration();

configuration.configure();

serviceRegistry = new ServiceRegistryBuilder().applySettings(

configuration.getProperties()).buildServiceRegistry();

sessionFactory = configuration.buildSessionFactory(serviceRegistry);

} catch (Throwable th) {

System.err.println("Enitial SessionFactory creation failed" + th);

throw new ExceptionInInitializerError(th);

}

}

public static SessionFactory getSessionFactory() {

return sessionFactory;

}

}

We will use the getSessionFactory() method to get the SessionFactory object in our application.

**Step 10:** Testing the application.

Now we are ready to test the application. Create a new file CreateData .java and then add the following code in it:

package net.roseindia;

import org.hibernate.Session;

import org.hibernate.SessionFactory;

import net.roseindia.model.\*;

/\*\*

\* @author Deepak Kumar

\* Web: http://www.roseindia.net

\*/

public class CreateData {

public static void main(String[] args) throws Exception {

SessionFactory sessFact = HibernateUtil.getSessionFactory();

Session session = sessFact.getCurrentSession();

org.hibernate.Transaction tr = session.beginTransaction();

Employee emp = new Employee();

emp.setEmpName("Deepak Kumar");

emp.setEmpMobileNos("000000");

emp.setEmpAddress("Delhi - India");

session.save(emp);

tr.commit();

System.out.println("Successfully inserted");

sessFact.close();

}

}

Here we are getting the Session object from the SessionFactory and uses the save method to save the Employee object. The save method actually instructs the hibernate to create the insert statement and then execute it against the database. Here is the output of the program:

INFO: HHH000400: Using dialect: org.hibernate.dialect.MySQLDialect

Dec 19, 2013 12:04:39 PM org.hibernate.engine.transaction.internal.TransactionFactoryInitiator initiateService

INFO: HHH000399: Using default transaction strategy (direct JDBC transactions)

Dec 19, 2013 12:04:39 PM org.hibernate.hql.internal.ast.ASTQueryTranslatorFactory

INFO: HHH000397: Using ASTQueryTranslatorFactory

Hibernate: insert into employee (emp\_address, emp\_mobile\_nos, emp\_name) values (?, ?, ?)

Successfully inserted

In this section you have created first program in Hibernate which inserts the data into database