

First Hibernate Example without IDE

Here, we are going to create the first hibernate application without IDE. For creating the first hibernate application, we need to follow following steps:

1. Create the Persistent class
2. Create the mapping file for Persistent class
3. Create the Configuration file
4. Create the class that retrieves or stores the persistent object
5. Load the jar file
6. Run the first hibernate application without IDE

1) Create the Persistent class

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.

Let's create the simple Persistent class:

Employee.java

```
package com.javatpoint.mypackage;

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;  
}  
  
}
```

2) Create the mapping file for Persistent class

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.

Let's see the mapping file for the Employee 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="com.javatpoint.mypackage.Employee" table="emp1000">
    <id name="id">
      <generator class="assigned"></generator>
    </id>

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

  </class>

</hibernate-mapping>
```

3) Create the Configuration file

The configuration file contains informations about the database and mapping file. Conventionally, its name should be 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>
```

4) Create the class that retrieves or stores the object

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

```
package com.javatpoint.mypackage;

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 session 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");

}
}
```

5) Load the jar file

For successfully running the hibernate application, you should have the hibernate4.jar file.

download the latest hibernate jar file. Some other jar files or packages are required such as

- cglib
- log4j
- commons
- SLF4J
- dom4j
- xalan
- xerces

download the required jar files for hibernate

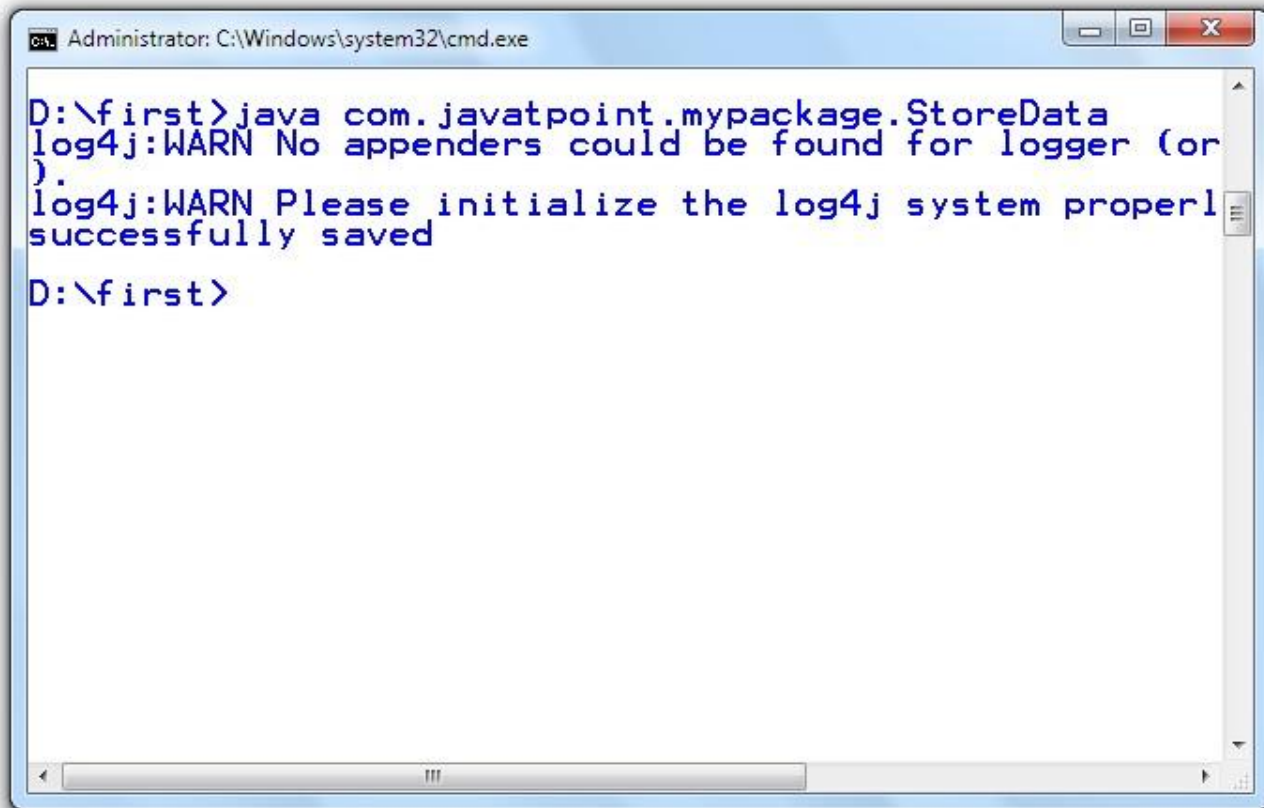
6) How to run the first hibernate application without IDE

We may run this hibernate application by IDE (e.g. Eclipse, Myeclipse, Netbeans etc.) or without IDE. We will learn about creating hibernate application in Eclipse IDE in next chapter.

To run the hibernate application without IDE:

- install the oracle10g for this example.

- load the jar files for hibernate. (One of the way to load the jar file is copy all the jar files under the JRE/lib/ext folder). It is better to put these jar files inside the public and private JRE both.
- Now Run the StoreData class by **java com.javatpoint.mypackage.StoreData**



```
Administrator: C:\Windows\system32\cmd.exe
D:\first>java com.javatpoint.mypackage.StoreData
log4j:WARN No appenders could be found for logger (or
).
log4j:WARN Please initialize the log4j system properly
successfully saved
D:\first>
```

Note: You need to connect with the internet to run this example.

download the first hibernate example

← prev

next →

Share 8