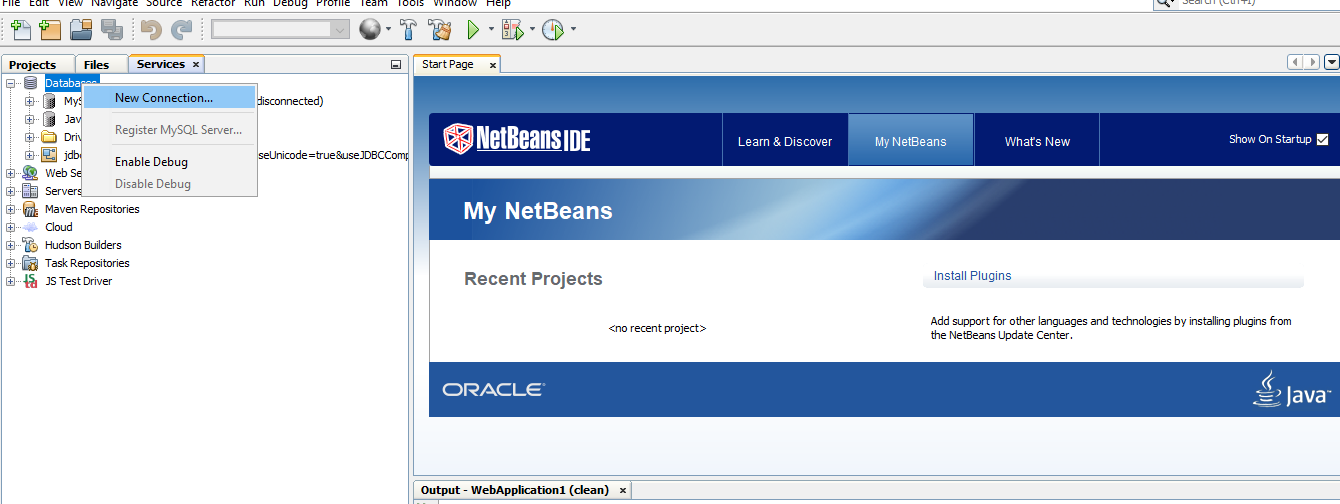
Connection to the Database:-

1 First go to the Services tab into it.

2 Then into the services tab Databases then click on the right click button.

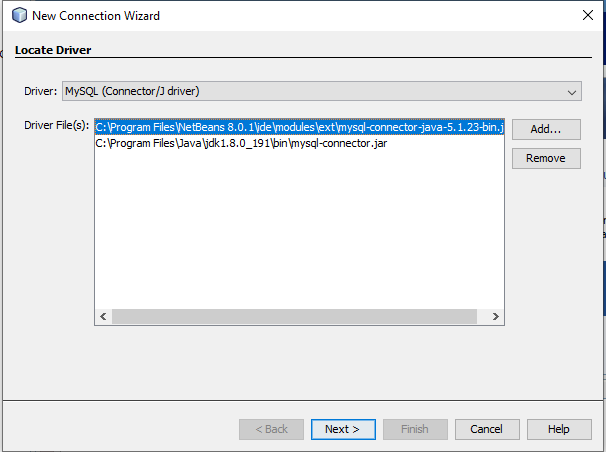
3 Then click on the new connection button



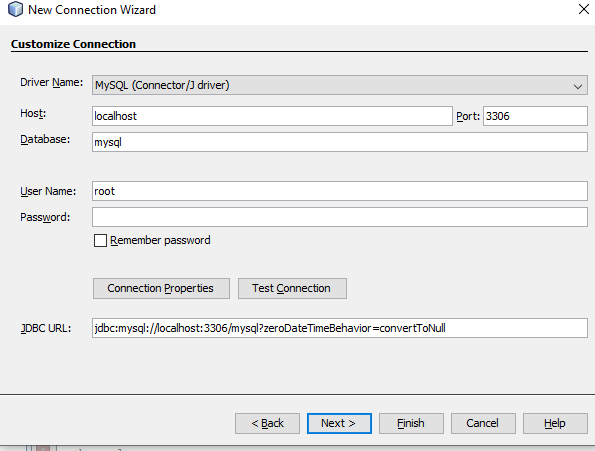
4 Then the new Connection Wizard is open.

Then in the driver Select MySQL(Connector/J driver).

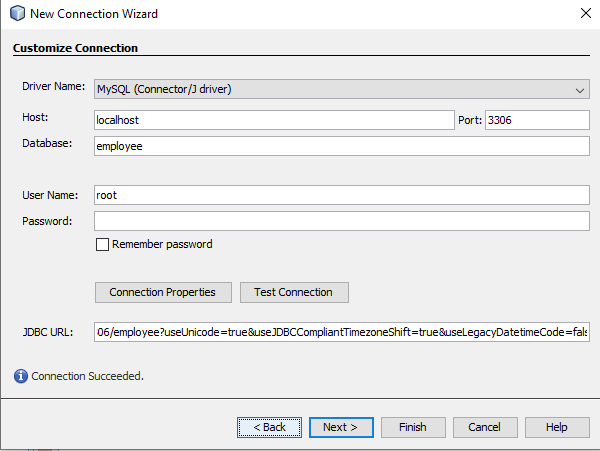
Then select the driver file and click on the next



5) Then the following screen is display into it.

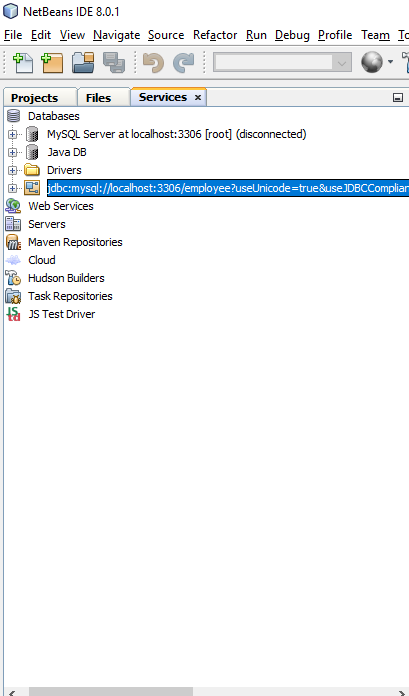


Then give the Port number and database name and user Name and password into it.



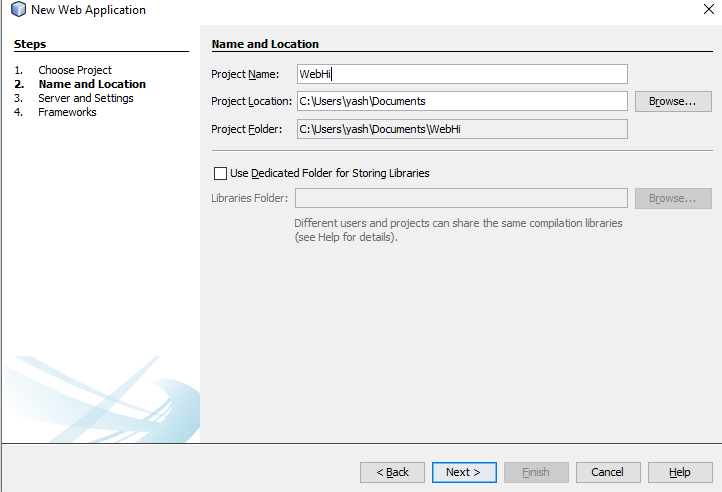
6 then test connection if it is ok then click on the next button. A new screen is display then click on the next button into it.

7 At last click on the finish button.

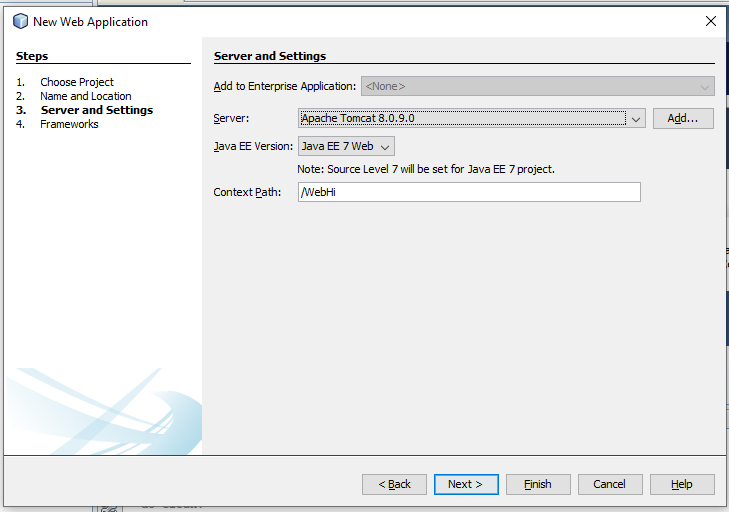


Creating the Web Application Project

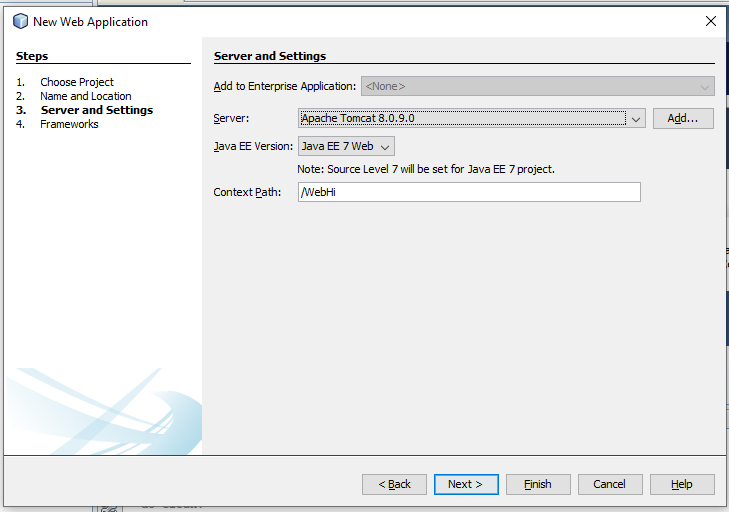
1. Choose File > New Project (Ctrl-Shift-N; ⌘-Shift-N on Mac) from the main menu. Select Web Application from the Java Web category and click Next.
2. Then give the Project name. In our case name is WebHi.



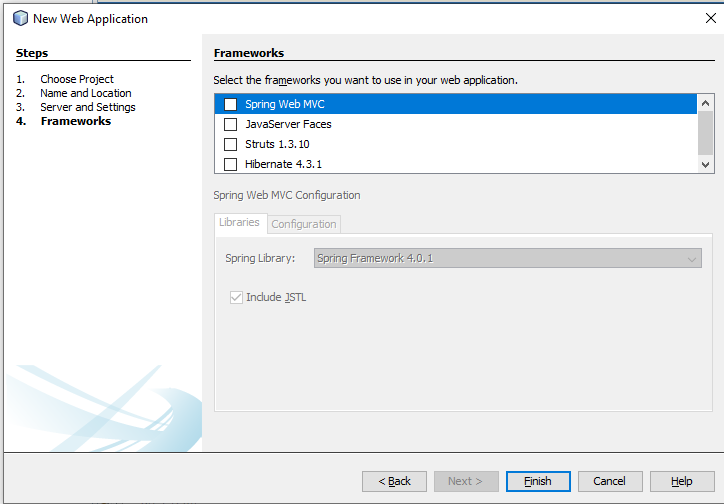
3 Click the next buttons.



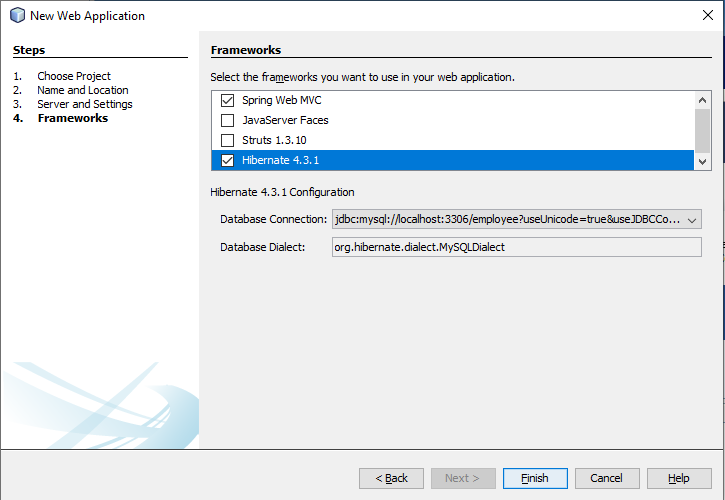
4 click the next button. Then following screen is display into it.



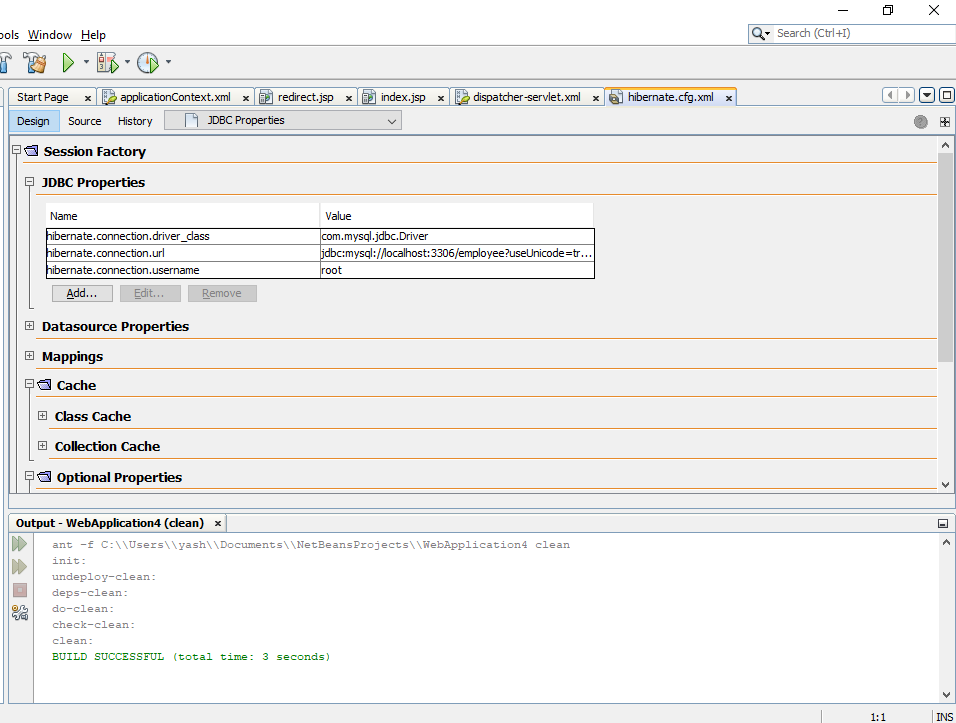
5 Then the following screen is displayed into it.



6 Select the Spring Web MVC and Hibernate 4.3.1 it is display look like this in to it.



7 Then click on the finish button into it. Then we can see that the hibernate.cfg.xml file is open now in design part.



Hibernate.cfg.xml file code is

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

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

<property name="hibernate.connection.url">jdbc:mysql://localhost:3306/employee?useUnicode=true&amp;useJDBCCompliantTimezoneShift=true&amp;useLegacyDatetimeCode=false&amp;serverTimezone=UTC</property>

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

</session-factory>

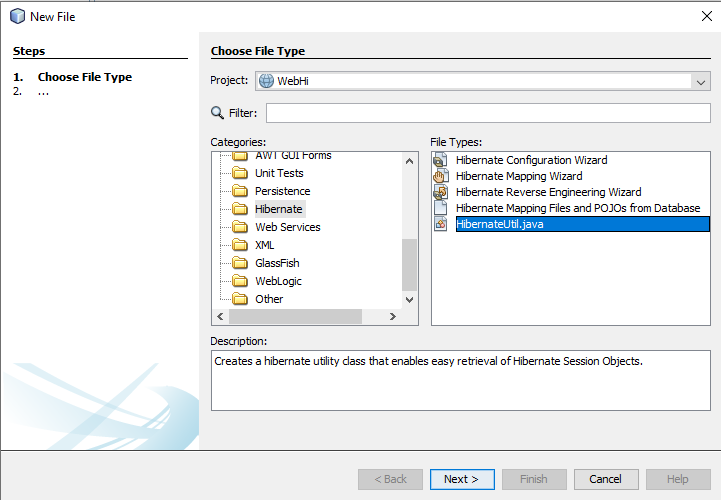
</hibernate-configuration>

Creating the HibernateUtil.java Helper File

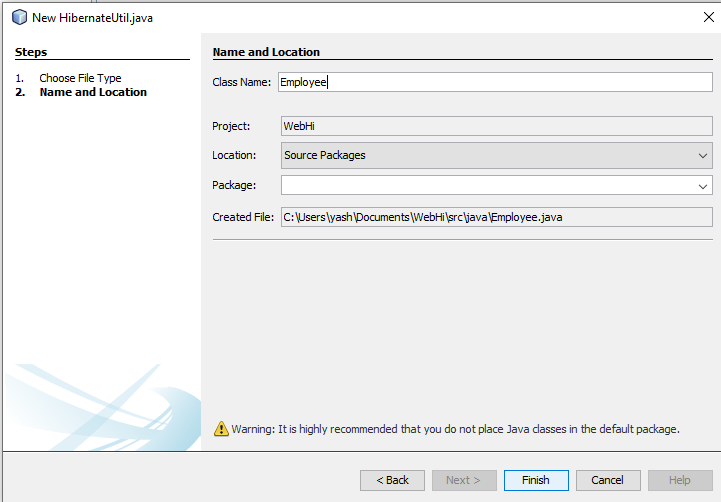
To use Hibernate you need to create a helper class that handles startup and that accesses Hibernate's SessionFactory to obtain a Session object. The class calls configure() and loads the hibernate.cfg.xml configuration file and then builds the SessionFactory to obtain the Session object.

In this section you use the New File wizard to create the helper class Employeee.java.

1. Right-click the Source Packages node and select New > Other to open the New File wizard.
2. Select Hibernate from the Categories list and HibernateUtil.java from the File Types list. Click Next.
3. Type **HibernateUtil** for the class name and **dvdrental** for the package. Click Finish.



2)



Employee.java

/\*

\* To change this license header, choose License Headers in Project Properties.

\* To change this template file, choose Tools | Templates

\* and open the template in the editor.

\*/

import org.hibernate.cfg.AnnotationConfiguration;

import org.hibernate.SessionFactory;

/\*\*

\* Hibernate Utility class with a convenient method to get Session Factory

\* object.

\*

\* @author yash

\*/

public class Employee {

private static final SessionFactory sessionFactory;

static {

try {

// Create the SessionFactory from standard (hibernate.cfg.xml)

// config file.

sessionFactory = new AnnotationConfiguration().configure().buildSessionFactory();

} catch (Throwable ex) {

// Log the exception.

System.err.println("Initial SessionFactory creation failed." + ex);

throw new ExceptionInInitializerError(ex);

}

}

public static SessionFactory getSessionFactory() {

return sessionFactory;

}

}

Generating Hibernate Mapping Files and Java Classes

In this tutorial you use a POJO (plain old Java object) to represent the data in each of the tables in the database that you will use. The Java class specifies the fields for the columns in the tables and uses simple setters and getters to retrieve and write the data. To map the POJOs to the tables you can use a Hibernate mapping file or use annotations in the class.

You can use the Hibernate Mapping Files and POJOs from a Database wizard to create multiple POJOs and mapping files based on database tables. When you use the wizard you select all the tables for which you want POJOs and mapping files and the IDE then generates the files for you based on the database tables and adds the mapping entries to hibernate.cfg.xml. When you use the wizard you can choose the files that you want the IDE to generate (only the POJOs, for example) and select code generation options (generate code that uses EJB 3 annotations, for example).

In this tutorial you use a POJO (plain old Java object) to represent the data in each of the tables in the database that you will use. The Java class specifies the fields for the columns in the tables and uses simple setters and getters to retrieve and write the data. To map the POJOs to the tables you can use a Hibernate mapping file or use annotations in the class.

You can use the Hibernate Mapping Files and POJOs from a Database wizard to create multiple POJOs and mapping files based on database tables. When you use the wizard you select all the tables for which you want POJOs and mapping files and the IDE then generates the files for you based on the database tables and adds the mapping entries to hibernate.cfg.xml. When you use the wizard you can choose the files that you want the IDE to generate (only the POJOs, for example) and select code generation options (generate code that uses EJB 3 annotations, for example).

In this tutorial you use a POJO (plain old Java object) to represent the data in each of the tables in the database that you will use. The Java class specifies the fields for the columns in the tables and uses simple setters and getters to retrieve and write the data. To map the POJOs to the tables you can use a Hibernate mapping file or use annotations in the class.

You can use the Hibernate Mapping Files and POJOs from a Database wizard to create multiple POJOs and mapping files based on database tables. When you use the wizard you select all the tables for which you want POJOs and mapping files and the IDE then generates the files for you based on the database tables and adds the mapping entries to hibernate.cfg.xml. When you use the wizard you can choose the files that you want the IDE to generate (only the POJOs, for example) and select code generation options (generate code that uses EJB 3 annotations, for example). In this tutorial you use a POJO (plain old Java object) to represent the data in each of the tables in the database that you will use. The Java class specifies the fields for the columns in the tables and uses simple setters and getters to retrieve and write the data. To map the POJOs to the tables you can use a Hibernate mapping file or use annotations in the class.

You can use the Hibernate Mapping Files and POJOs from a Database wizard to create multiple POJOs and mapping files based on database tables. When you use the wizard you select all the tables for which you want POJOs and mapping files and the IDE then generates the files for you based on the database tables and adds the mapping entries to hibernate.cfg.xml. When you use the wizard you can choose the files that you want the IDE to generate (only the POJOs, for example) and select code generation options (generate code that uses EJB 3 annotations, for example).

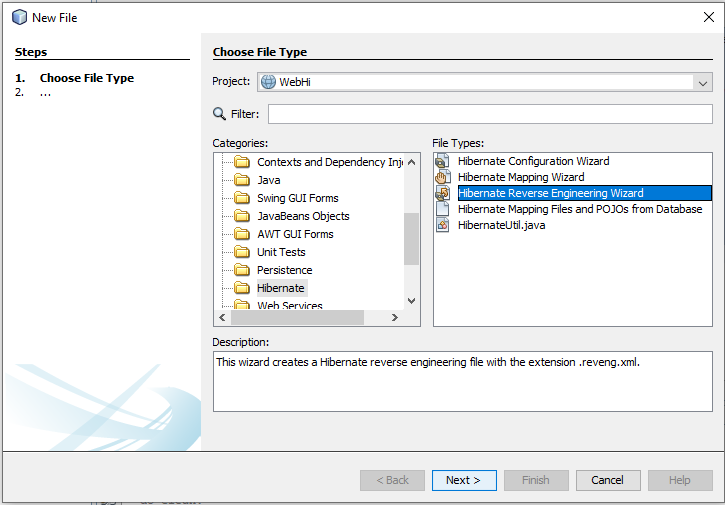
Creating the Hibernate Reverse Engineering File

If you want to use the Hibernate Mapping Files and POJOs from a Database wizard, you first need to create a hibernate.reveng.xml reverse engineering file. The Hibernate Mapping Files and POJOs from a Database wizard requires hibernate.reveng.xml and hibernate.cfg.xml.

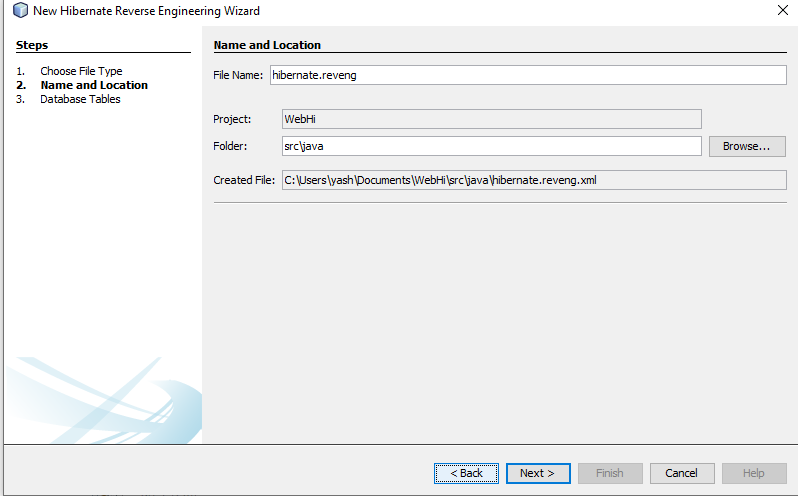
The reverse engineering file enables you to have greater control over the database mapping strategy. The Hibernate Reverse Engineering Wizard creates a reverse engineering file with a default configuration that you can edit in the XML editor.

To create the Hibernate reverse engineering file, perform the following steps.

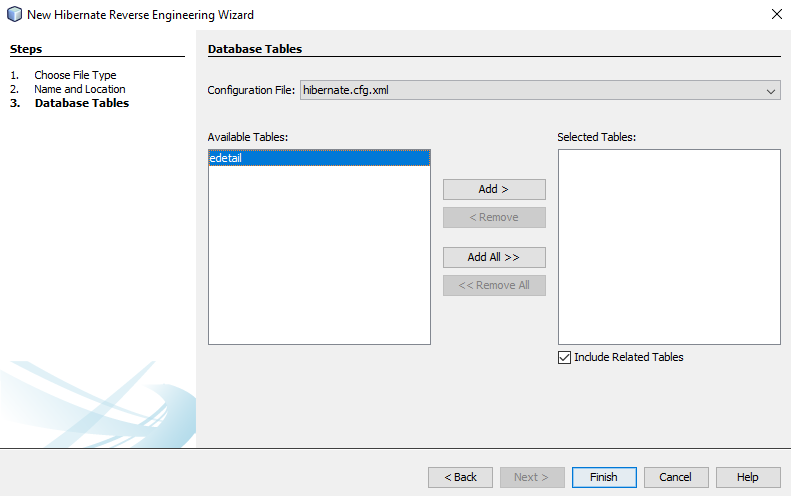
1. Right-click the Source Packages node in the Projects window and choose New > Other to open the New File wizard.
2. Select Hibernate Reverse Engineering Wizard in the Hibernate category. Click Next.



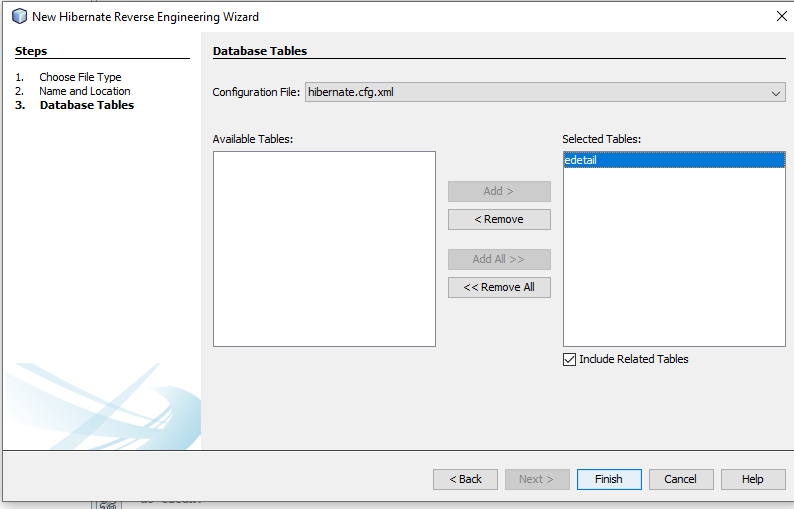
1. Specify hibernate.reveng as the File Name and src/java for the Folder. Click Next.



4 Select hibernate.cfg.xml from the Configuration File drop down list, if not selected



5. Select the following tables from Available Tables and click Add to add the tables to Selected Tables.



hibernate.reveng.xml :-

<?xml version="1.0" encoding="UTF-8"?>

<!DOCTYPE hibernate-reverse-engineering PUBLIC "-//Hibernate/Hibernate Reverse Engineering DTD 3.0//EN" "http://hibernate.sourceforge.net/hibernate-reverse-engineering-3.0.dtd">

<hibernate-reverse-engineering>

<schema-selection match-catalog="employee"/>

<table-filter match-name="edetail"/>

</hibernate-reverse-engineering>

Creating the Hibernate Mapping Files and POJOs

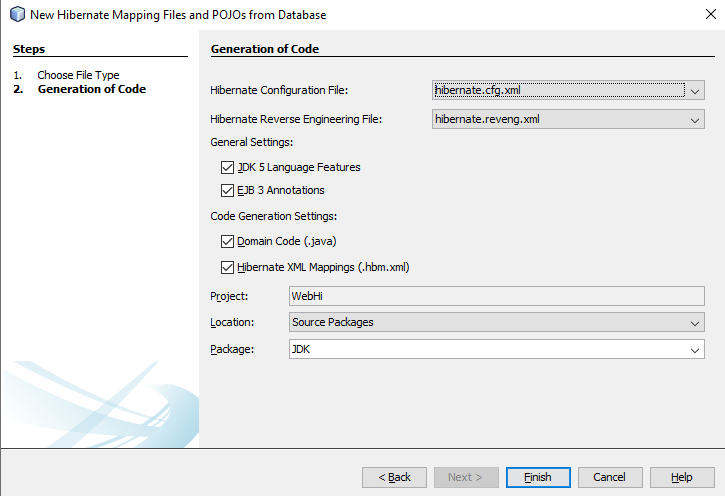
You can use the Hibernate Mapping Files and POJOs from a Database wizard to generate files for you. The wizard can generate a POJO and a corresponding mapping file for each table that you select in the wizard. The mapping files are XML files that contain data about how the columns in the tables are mapped to the fields in the POJOs. You need to have the  hibernate.reveng.xml  and  hibernate.cfg.xml  files to use the wizard.

To create the POJOS and mapping files using a wizard, perform the following steps.

1 Right-click the Source Packages node in the Projects window and choose New > Other to open the New File wizard

2 Select Hibernate Mapping Files and POJOs from a Database in the Hibernate category. Click Next.

3 Ensure that the hibernate.cfg.xml and hibernate.reveng.xml files are selected in the drop down lists.



When you click Finish the IDE generates POJOs and Hibernate mapping files with the fields mapped to the columns specified in hibernate.reveng.xml. The IDE also adds mapping entries to hibernate.cfg.xml.

In the new package following two files are created.

Edetail.java

package JDK;

// Generated 17 Mar, 2020 7:03:31 AM by Hibernate Tools 4.3.1

import javax.persistence.Column;

import javax.persistence.Entity;

import javax.persistence.Id;

import javax.persistence.Table;

/\*\*

\* Edetail generated by hbm2java

\*/

@Entity

@Table(name="edetail"

,catalog="employee"

)

public class Edetail implements java.io.Serializable {

private String eid;

private String ename;

public Edetail() {

}

public Edetail(String eid, String ename) {

this.eid = eid;

this.ename = ename;

}

@Id

@Column(name="eid", unique=true, nullable=false, length=20)

public String getEid() {

return this.eid;

}

public void setEid(String eid) {

this.eid = eid;

}

@Column(name="ename", nullable=false, length=65535)

public String getEname() {

return this.ename;

}

public void setEname(String ename) {

this.ename = ename;

}

}

Edetail.hbm.xml

<?xml version="1.0"?>

<!DOCTYPE hibernate-mapping PUBLIC "-//Hibernate/Hibernate Mapping DTD 3.0//EN"

"http://www.hibernate.org/dtd/hibernate-mapping-3.0.dtd">

<!-- Generated 17 Mar, 2020 7:03:34 AM by Hibernate Tools 4.3.1 -->

<hibernate-mapping>

<class name="JDK.Edetail" table="edetail" catalog="employee" optimistic-lock="version">

<id name="eid" type="string">

<column name="eid" length="20" />

<generator class="assigned" />

</id>

<property name="ename" type="string">

<column name="ename" length="65535" not-null="true" />

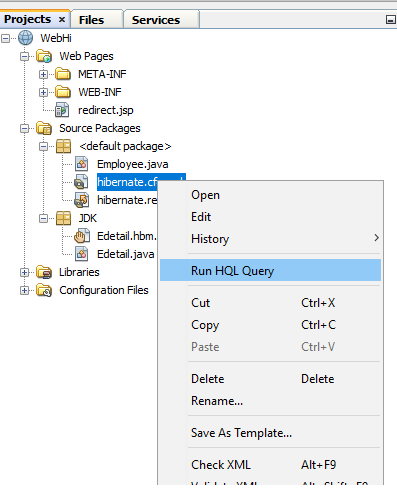
</property>

</class>

</hibernate-mapping>

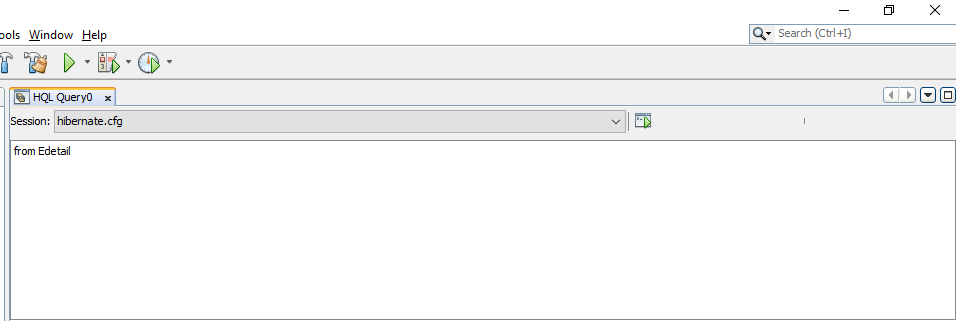
Hql Query Run:-

1. Right-click the project node in the Projects window and choose Clean and Build.
2. Right-click hibernate.cfg.xml in the Projects window and choose Run HQL Query to open the HQL query editor.

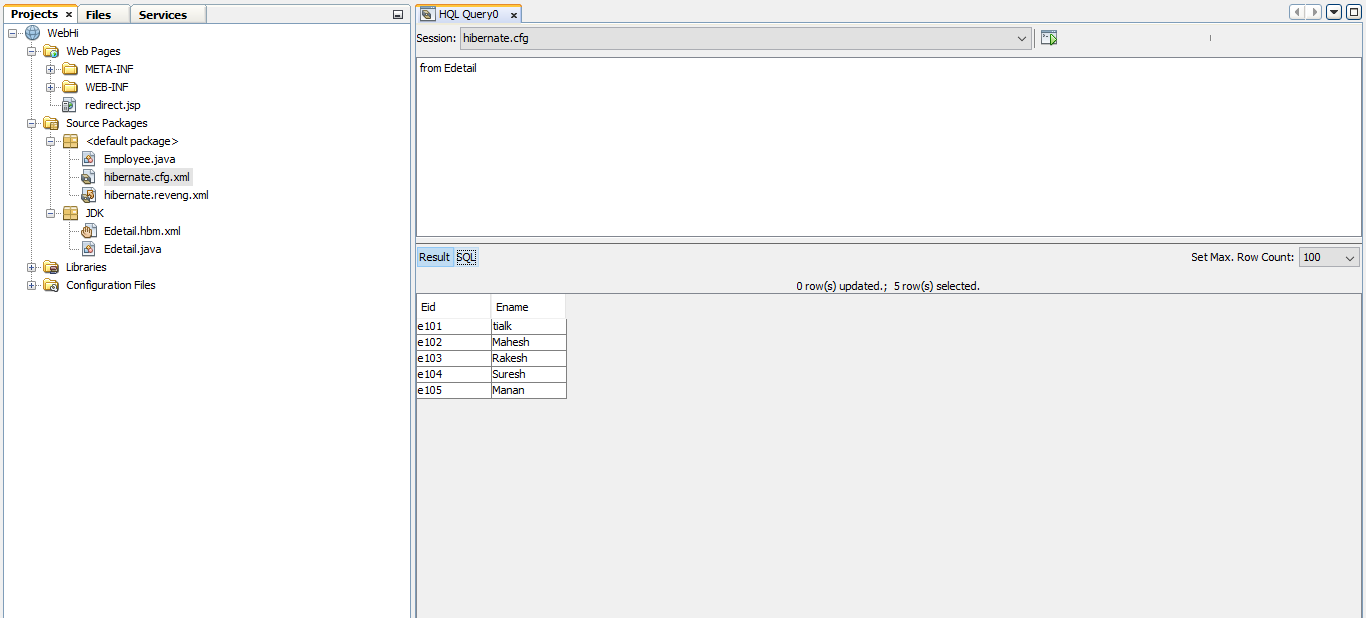


3. Select hibernate.cfg from the drop down list in the toolbar.

4.Test the connection by typing the following in the editor and clicking the Run HQL Query button ( Run HQL Query button ) in the toolbar.



When you click Run HQL Query you can see the results of the query in the bottom window of the HQL query editor.



Select specific records.

