Configure Hibernate in Eclipse IDE. ?

1. Open Eclipse
2. Then Go to Help ,Then Choose option Eclipse Marketplace
3. After open eclipse marketplace ,search jboss
4. So you find jboss Tools (4.24.0) Final [Software version mention is in my case]
5. Click Install
6. A window come up ,Select only Hibernate Tools and deselect all the others .You have no need to download other tools.
7. Click Next ,and Then Finish
8. Take time to download

Configure Hibernate using XML in Eclipse IDE. ?

Nowadays XML is not used with Hibernate .We using annotation.

**Step 3.2.1:** Creating a database in MySQL

* MySQL is already installed in your practice lab. (Refer FSD: Lab Guide - Phase 2)
* Login to the MySQL command line console
* Type **CREATE DATABASE ecommerce** and press **Enter**
* Type **USE ecommerce** and press **Enter**
* Type **CREATE TABLE eproduct (ID bigint primary key auto\_increment, name varchar(100), price decimal(10,2), date\_added timestamp default now())** and press **Enter**
* We will now add some rows to the table
* Type **INSERT INTO eproduct(name, ‘HP Laptop ABC’, 12000)** and press **Enter**
* Type **INSERT INTO eproduct(name, ‘Acer Laptop ABC’, 14000)** and press **Enter**
* Type **INSERT INTO eproduct(name, ‘Lenovo Laptop ABC’, 12000)** and press **Enter**
* Type **SELECT \* from eproduct** and press **Enter** to confirm that the rows have been added
* Type **EXIT** to exit the MySQL command console

**Step 3.2.2:** Creating a dynamic web project

* Open Eclipse
* Go the **File** menu. Choose **New->Dynamic Web Project**
* Enter the project name as **HibernateConfig**. Click on **Next**
* Enter nothing in the next screen and click on **Next**
* Check the checkbox **Generate web.xml deployment descriptor** and click on **Finish**
* This will create the project files in the Project Explorer

**Step 3.2.3:** Adding the jar files for Hibernate and its dependencies

* **Hibernate.jar** file is already present in your practice lab. (Refer FSD: Lab Guide - Phase 2)
* Take **hibernate.jar** from folder mentioned in the lab guide for phase 2 and add it to your project’s **WebContent/WEB-INF/lib** folder
* **mysql-connector-java.jar** file is present in your practice lab. (Refer FSD: Lab Guide - Phase 2)
* Take **mysql-connector-java.jar file** from the folder mentioned in the lab guide for phase 2 and add it to your project’s **WebContent/WEB-INF/lib** folder
* Go to <http://www.java2s.com/Code/Jar/j/Downloadjta11jar.htm>
* Click on **jta-1\_1.jar.zip** link to download it
* Extract **jta-1\_1.jar** from it and add it to your project’s **WebContent/WEB-INF/lib** folder
* Go to <http://www.java2s.com/Code/Jar/j/Downloadjavaxxmlbindjar.htm>
* Click on **javax.xml/javax.xml.bind.jar.zip** link to download it
* Extract **javax.xml.bind.jar** from it and add it to your project’s **WebContent/WEB-INF/lib** folder
* Go to <https://jar-download.com/artifacts/com.sun.xml.bind>
* Click on the button **Download jaxb-osgi.jar** to download it
* Extract **jaxb-osgi-2.4.0-b180830.0438.jar** from it and add it to your project’s **WebContent/WEB-INF/lib** folder

**Step 3.2.4:** Configuring Hibernate with hibernate.cfg.xml

* In the Project Explorer, expand **HibernateConfig->Java Resources**
* Right click on **src** and choose **New->Other**
* Select **General->File** and click on **Next**
* In filename, enter **hibernate.cfg.xml** and click on **Finish**

Configure Hibernate using Annotations in Eclipse IDE. ?

DemoTest.java :-

**package** com.main;

**import** java.util.List;

**import** com.bean.Employee;

**import** com.service.EmployeeService;

**public** **class** DemoTest {

**public** **static** **void** main(String[] args) {

// INsert

Employee emp1 = **new** Employee();

emp1.setId(3);

emp1.setName("Ram");

emp1.setSalary(10000);

EmployeeService es = **new** EmployeeService();

String result = es.storeEmployee(emp1);

System.***out***.println(result);

/\*

EmployeeService es = new EmployeeService();

List<Employee> listOfEmp = es.getAllEmployee();

for(Employee emp:listOfEmp) {

System.out.println(emp);

}

\*/

}

}

EmployeeService.java :-

**package** com.service;

**import** java.util.Iterator;

**import** java.util.List;

**import** com.bean.Employee;

**import** com.dao.EmployeeDao;

**public** **class** EmployeeService {

EmployeeDao ed = **new** EmployeeDao();

**public** String storeEmployee(Employee emp) {

**if**(emp.getSalary()<10000) {

**return** "Employee salary must be > 10000";

}**else** **if**(ed.storeEmployee(emp)>0) {

**return** "Record inserted successfully";

}**else** {

**return** "Record didn't insert";

}

}

/\*

public List<Employee> getAllEmployee() {

List<Employee> listOfEmp = ed.getAllEmployee();

Iterator<Employee> li = listOfEmp.iterator();

while(li.hasNext()) {

Employee emp = li.next();

emp.setSalary(emp.getSalary()+5000);

}

return listOfEmp;

}

\*/

}

EmployeeDao.java :-

**package** com.dao;

/\*

import java.sql.Connection;

import java.sql.DriverManager;

import java.sql.PreparedStatement;

import java.sql.ResultSet;

import java.util.ArrayList;

import java.util.List;

\*/

**import** org.hibernate.Session;

**import** org.hibernate.SessionFactory;

**import** org.hibernate.Transaction;

**import** org.hibernate.cfg.Configuration;

**import** com.bean.Employee;

**public** **class** EmployeeDao {

/\*

public int storeEmployee(Employee emp) {

try {

Class.forName("com.mysql.cj.jdbc.Driver");

Connection con = DriverManager.getConnection("jdbc:mysql://localhost:3306/javatraining", "root", "admin");

PreparedStatement pstmt = con.prepareStatement("insert into employee values(?,?,?)");

pstmt.setInt(1, emp.getId());

pstmt.setString(2, emp.getName());

pstmt.setFloat(3, emp.getSalary());

int res = pstmt.executeUpdate();

return res;

} catch (Exception e) {

System.out.println(e);

return 0;

}

}

public List<Employee> getAllEmployee() {

List<Employee> listOfEmp =new ArrayList<Employee>();

try {

Class.forName("com.mysql.cj.jdbc.Driver");

Connection con = DriverManager.getConnection("jdbc:mysql://localhost:3306/javatraining", "root", "admin");

PreparedStatement pstmt = con.prepareStatement("select \* from employee");

ResultSet rs = pstmt.executeQuery();

while(rs.next()) {

Employee emp = new Employee();

emp.setId(rs.getInt(1));

emp.setName(rs.getString(2));

emp.setSalary(rs.getFloat(3));

listOfEmp.add(emp);

}

} catch (Exception e) {

System.out.println(e);

}

return listOfEmp;

}

\*/

**public** **int** storeEmployee(Employee emp) {

**try** {

Configuration con = **new** Configuration();

con.configure("hibernate.cfg.xml"); //Load xml file

SessionFactory sf = con.buildSessionFactory(); //Connection setup con

Session session = sf.openSession(); //Statement or Prepared Statement

Transaction tran = session.getTransaction();

tran.begin();

session.save(emp); //insert

tran.commit();

**return** 1;

}

**catch**(Exception e) {

System.***out***.println(e);

**return** 0;

}

}

}

Employee.java :-

**package** com.bean;

**import** javax.persistence.Entity;

**import** javax.persistence.Table;

**import** javax.persistence.Id;

**import** javax.persistence.Column;

@Entity

//@Table(name = "employee") //It is optional ,if you make a table with different name .

**public** **class** Employee {

@Id //Primary key is mandatory in hibernate so use @ annotation,variable can be anything

**private** **int** id;

@Column(name="name")

**private** String name;

**private** **float** salary;

**public** **int** getId() {

**return** id;

}

**public** **void** setId(**int** id) {

**this**.id = id;

}

**public** String getName() {

**return** name;

}

**public** **void** setName(String name) {

**this**.name = name;

}

**public** **float** getSalary() {

**return** salary;

}

**public** **void** setSalary(**float** salary) {

**this**.salary = salary;

}

@Override

**public** String toString() {

**return** "Employee [id=" + id + ", name=" + name + ", salary=" + salary + "]";

}

}

Demonstrate Hibernate logging by Log4j. ?

Lo4j :-

# Direct log messages to a log file

log4j.appender.file=org.apache.log4j.RollingFileAppender

log4j.appender.file.File=/home/oem/Downloads/hibernate.log

log4j.appender.file.MaxFileSize=1MB

log4j.appender.file.MaxBackupIndex=1

log4j.appender.file.layout=org.apache.log4j.PatternLayout

log4j.appender.file.layout.ConversionPattern=%d{ABSOLUTE} %5p %c{1}:%L - %m%n

# Direct log messages to stdout

log4j.appender.**stdout**=org.apache.log4j.ConsoleAppender

log4j.appender.**stdout**.Target=System.out

log4j.appender.**stdout**.layout=org.apache.log4j.PatternLayout

log4j.appender.**stdout**.layout.ConversionPattern=%d{ABSOLUTE} %5p %c{1}:%L - %m%n

# Root logger option

log4j.rootLogger=INFO, file, **stdout**

# Log everything. Good for troubleshooting

log4j.logger.org.hibernate=INFO

# Log all JDBC parameters

log4j.logger.org.hibernate.type=ALL

html code :-

<!DOCTYPE html>

<**html**>

<**head**>

<**meta** charset="UTF-8">

<**title**>Hibernate Configuration Example</**title**>

</**head**>

<**body**>

<**a** href="init">Initialize Hibernate</**a**><**br**>

</**body**>

</**html**>

Hibernate code :-

**package** com.ecommerce;

**import** org.hibernate.SessionFactory;

**import** org.hibernate.boot.Metadata;

**import** org.hibernate.boot.MetadataSources;

**import** org.hibernate.boot.registry.StandardServiceRegistry;

**import** org.hibernate.boot.registry.StandardServiceRegistryBuilder;

**public** **class** HibernateUtil {

**private** **static** **final** SessionFactory sessionFactory;

**static** {

**try** {

StandardServiceRegistry standardRegistry = **new** StandardServiceRegistryBuilder()

.configure("hibernate.cfg.xml").build();

Metadata metaData = **new** MetadataSources(standardRegistry).getMetadataBuilder().build();

sessionFactory = metaData.getSessionFactoryBuilder().build();

} **catch** (**Throwable** th) {

**throw** **new** **ExceptionInInitializerError**(th);

}

}

**public** **static** SessionFactory getSessionFactory() {

**return** sessionFactory;

}

}

Demonstrate mapping List, Set, Bag, and Map in collection using XML file. ?

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

<!DOCTYPE hibernate-configuration PUBLIC

"-//Hibernate/Hibernate Configuration DTD 3.0//EN"

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

<hibernate-configuration>

<session-factory>

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

<property name=*"hibernate.connection.password"*>admin</property>

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

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

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

<mapping class=*"com.bean.Employee"*/>

</session-factory>

</hibernate-configuration>

Demonstrate lazy collection in Hibernate. ?

ClassProduct code :-

**package** com.ecommerce;

**import** java.math.BigDecimal;

**import** java.util.Collection;

**import** java.util.Date;

**import** java.util.List;

**import** java.util.Set;

**import** java.util.Map;

**public** **class** EProduct {

**private** long ID;

**private** **String** name;

**private** **BigDecimal** price;

**private** **Date** dateAdded;

**private** **List**<Color> colors;

**private** **Set**<Finance> finance;

**private** PDescription pdescrip;

**public** EProduct() {

}

**public** long getID() {**return** **this**.ID; }

**public** **String** getName() { **return** **this**.name;}

**public** **BigDecimal** getPrice() { **return** **this**.price;}

**public** **Date** getDateAdded() { **return** **this**.dateAdded;}

**public** **List**<Color> getColors() { **return** **this**.colors;}

**public** **Set**<Finance> getFinance() { **return** **this**.finance;}

**public** PDescription getPdescrip() { **return** **this**.pdescrip;}

**public** void setID(long id) { **this**.ID = id;}

**public** void setName(**String** name) { **this**.name = name;}

**public** void setPrice(**BigDecimal** price) { **this**.price = price;}

**public** void setDateAdded(**Date** date) { **this**.dateAdded = date;}

**public** void setColors(**List**<Color> colors) { **this**.colors = colors;}

**public** void setFinance(**Set**<Finance> finance) { **this**.finance = finance;}

**public** void setPdescrip(PDescription pdescrip) { **this**.pdescrip = pdescrip;}

}

Class finance code :-

**package** com.ecommerce;

**public** **class** Finance {

**private** long FINANCEID;

**private** **String** name;

**private** **String** ftype;

**public** Finance() {

}

**public** Finance(**String** name, **String** ftype) {

**this**.FINANCEID = 0;

**this**.name = name;

**this**.ftype = ftype;

}

**public** long getFINANCEID() {**return** **this**.FINANCEID; }

**public** **String** getName() { **return** **this**.name;}

**public** **String** getFtype() { **return** **this**.ftype;}

**public** void setFINANCEID(long id) { **this**.FINANCEID = id;}

**public** void setName(**String** name) { **this**.name = name;}

**public** void setFtype(**String** ftype) { **this**.ftype= ftype;}

}

Demonstrate component mapping in Hibernate. ?

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

<!DOCTYPE hibernate-configuration PUBLIC

"-//Hibernate/Hibernate Configuration DTD 3.0//EN"

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

<hibernate-configuration>

<session-factory>

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

<property name=*"hibernate.connection.password"*>admin</property>

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

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

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

<mapping class=*"com.bean.Employee"*/>

</session-factory>

</hibernate-configuration>

Demonstrate integration of Hibernate with spring. ?

Pom.xml file code :-

<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/maven-v4\_0\_0.xsd">

<modelVersion>4.0.0</modelVersion>

<groupId>SpringHibernateWeb</groupId>

<artifactId>SpringHibernateWeb</artifactId>

<packaging>war</packaging>

<version>0.0.1-SNAPSHOT</version>

<name>SpringHibernateWeb Maven Webapp</name>

<url>http://maven.apache.org</url>

<!-- JBoss repository for Hibernate -->

<repositories>

<repository>

<id>JBoss repository</id>

<url>http://repository.jboss.org/nexus/content/groups/public/</url>

</repository>

</repositories>

<properties>

<org.springframework.version>3.0.5.RELEASE</org.springframework.version>

</properties>

<dependencies>

<dependency>

<groupId>junit</groupId>

<artifactId>junit</artifactId>

<version>3.8.1</version>

<scope>test</scope>

</dependency>

<dependency>

<groupId>org.hibernate.javax.persistence</groupId>

<artifactId>hibernate-jpa-2.1-api</artifactId>

<version>1.0.0.Final</version>

</dependency>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-core</artifactId>

<version>${org.springframework.version}</version>

</dependency>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-expression</artifactId>

<version>${org.springframework.version}</version>

</dependency>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-beans</artifactId>

<version>${org.springframework.version}</version>

</dependency>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-context</artifactId>

<version>${org.springframework.version}</version>

</dependency>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-context-support</artifactId>

<version>${org.springframework.version}</version>

</dependency>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-jdbc</artifactId>

<version>${org.springframework.version}</version>

</dependency>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-orm</artifactId>

<version>${org.springframework.version}</version>

</dependency>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-web</artifactId>

<version>${org.springframework.version}</version>

</dependency>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-webmvc</artifactId>

<version>${org.springframework.version}</version>

</dependency>

<dependency>

<groupId>log4j</groupId>

<artifactId>log4j</artifactId>

<version>1.2.16</version>

<scope>runtime</scope>

</dependency>

<dependency>

<groupId>org.hibernate</groupId>

<artifactId>hibernate-core</artifactId>

<version>3.6.3.Final</version>

</dependency>

<dependency>

<groupId>javassist</groupId>

<artifactId>javassist</artifactId>

<version>3.12.1.GA</version>

</dependency>

<dependency>

<groupId>taglibs</groupId>

<artifactId>standard</artifactId>

<version>1.1.2</version>

<scope>runtime</scope>

</dependency>

<dependency>

<groupId>commons-dbcp</groupId>

<artifactId>commons-dbcp</artifactId>

<version>1.4</version>

</dependency>

<dependency>

<groupId>mysql</groupId>

<artifactId>mysql-connector-java</artifactId>

<version>5.1.9</version>

</dependency>

</dependencies>

<build>

<finalName>SpringHibernateWeb</finalName>

</build>

</project>

Class EProduct :-

**package** com.ecommerce.entity;

**import** java.math.BigDecimal;

**import** java.util.Date;

**import** javax.persistence.Column;

**import** javax.persistence.Entity;

**import** javax.persistence.GeneratedValue;

**import** javax.persistence.Id;

**import** javax.persistence.Table;

**@Entity**

**@Table(name= "eproduct")**

**public** **class** EProductEntity {

**@Id** **@GeneratedValue**

**@Column(name = "ID")**

**private** long ID;

**@Column(name = "name")**

**private** **String** name;

**@Column(name = "price")**

**private** **BigDecimal** price;

**@Column(name = "date\_added")**

**private** **Date** dateAdded;

**public** long getID() {**return** **this**.ID; }

**public** **String** getName() { **return** **this**.name;}

**public** **BigDecimal** getPrice() { **return** **this**.price;}

**public** **Date** getDateAdded() { **return** **this**.dateAdded;}

**public** void setID(long id) { **this**.ID = id;}

**public** void setName(**String** name) { **this**.name = name;}

**public** void setPrice(**BigDecimal** price) { **this**.price = price;}

**public** void setDateAdded(**Date** date) { **this**.dateAdded = date;}

}