**Step 1: Download Required Hibernate JARs**

You need the following Hibernate JAR files:

1. **Hibernate Core JARs**
2. **MySQL Connector JAR**
3. **JPA and JAXB dependencies**

**Download JARs from Hibernate official website:**

* Go to: **https://hibernate.org/orm/releases/5.6/**
* Download **Hibernate 5.6.5 Final (or latest version)**
* Extract the ZIP file and collect the required JARs

**Required JARs:**

* hibernate-core-5.6.5.Final.jar
* hibernate-commons-annotations-5.1.2.Final.jar
* antlr-2.7.7.jar
* jandex-2.0.5.Final.jar
* jboss-logging-3.4.1.Final.jar
* javax.persistence-api-2.2.jar
* mysql-connector-java-8.0.26.jar

**Step 2: Add JARs to Your Project**

* If using **Eclipse** or **NetBeans**, right-click your project → **Build Path** → **Add External JARs**
* Select all the **JAR files** and add them.

**Step 3: Create Hibernate Configuration File (hibernate.cfg.xml)**

This file configures **database connection settings**.

<?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>

<!-- Database connection settings -->

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

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

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

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

<!-- Hibernate Dialect -->

<property name="hibernate.dialect">org.hibernate.dialect.MySQL8Dialect</property>

<!-- Show SQL Queries in Console -->

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

<!-- Automatically create table -->

<property name="hibernate.hbm2ddl.auto">update</property>

<!-- Mapping file -->

<mapping resource="Emp.hbm.xml"/>

</session-factory>

</hibernate-configuration>

**Step 4: Create Entity Class (Emp.java)**

This is a **POJO class without annotations**.

import java.io.Serializable;

public class Emp implements Serializable {

private int id;

private String name;

private double salary;

// Default Constructor

public Emp() {}

// Getters and Setters

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 double getSalary() {

return salary;

}

public void setSalary(double salary) {

this.salary = salary;

}

}

**Step 5: Create Hibernate Mapping File (Emp.hbm.xml)**

This file maps **the Java class (Emp) to the database table (employees)**.

<?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="Emp" table="employees">

<!-- Primary Key Mapping -->

<id name="id" column="id">

<generator class="increment"/>

</id>

<!-- Other Columns -->

<property name="name" column="name" type="string"/>

<property name="salary" column="salary" type="double"/>

</class>

</hibernate-mapping>

**Step 6: Create Hibernate Utility Class (HibernateUtil.java)**

This class initializes **Hibernate SessionFactory**.

import org.hibernate.SessionFactory;

import org.hibernate.cfg.Configuration;

public class HibernateUtil {

private static final SessionFactory sessionFactory;

static {

try {

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

} catch (Throwable ex) {

throw new ExceptionInInitializerError(ex);

}

}

public static SessionFactory getSessionFactory() {

return sessionFactory;

}

}

**Step 7: Create Main Class (Main.java)**

This class **inserts data into the database**.

import org.hibernate.Session;

import org.hibernate.Transaction;

public class Main {

public static void main(String[] args) {

// Get Hibernate session

Session session = HibernateUtil.getSessionFactory().openSession();

Transaction tx = session.beginTransaction();

// Create an Employee Object

Emp emp = new Emp();

emp.setName("John Doe");

emp.setSalary(50000);

// Save Employee

session.save(emp);

// Commit and close

tx.commit();

session.close();

System.out.println("Employee inserted successfully!");

}

}

**Step 8: Run the Code**

* When executed, **Hibernate will create the employees table automatically**.
* It will insert one record with **name = "John Doe" and salary = 50000**.

**Output (Console)**

Hibernate: create table employees (id integer not null auto\_increment, name varchar(255), salary double, primary key (id))

Hibernate: insert into employees (name, salary) values (?, ?)

Employee inserted successfully!

**Final Notes**

✔ **No annotations are used; only XML-based configuration.**  
✔ **The hibernate.cfg.xml and Emp.hbm.xml handle table creation and mapping.**  
✔ **This is useful when working with legacy projects or older Hibernate versions**