# **Flow of hibernate**

# 

# 1 Application loads configuration (hibernate.cfg.xml). 2. Hibernate builds a SessionFactory. 3. Application requests a Session from SessionFactory. 4. Inside a Session:

# ●Begin Transaction

# ●Perform CRUD operations

# ●Commit or Rollback transaction

# 5. Hibernate translates operations -> SQL-> executes via JDBC. 6. Results are returned as Java objects

# 

# 

# 

# 

# 

# 

# 

# **Java Project with Hibernate Connection**

## **1. Project Setup**

You can create a **Maven Project** (recommended) or a simple Java project.

### **Add Hibernate & MySQL dependencies (in pom.xml if using Maven):**

<dependencies>  
 <!-- Hibernate Core -->  
 <dependency>  
 <groupId>org.hibernate</groupId>  
 <artifactId>hibernate-core</artifactId>  
 <version>6.4.4.Final</version>  
 </dependency>  
  
 <!-- MySQL JDBC Driver -->  
 <dependency>  
 <groupId>mysql</groupId>  
 <artifactId>mysql-connector-j</artifactId>  
 <version>8.3.0</version>  
 </dependency>  
  
 <!-- JPA API (optional but useful) -->  
 <dependency>  
 <groupId>jakarta.persistence</groupId>  
 <artifactId>jakarta.persistence-api</artifactId>  
 <version>3.1.0</version>  
 </dependency>  
 </dependencies>

## 

## 

## 

## 

## **2. Hibernate Configuration File (hibernate.cfg.xml)**

Create this file in your project’s resources folder.

<?xml version="1.0" encoding="UTF-8"?>  
 <!DOCTYPE hibernate-configuration PUBLIC  
 "-//Hibernate/Hibernate Configuration DTD 5.3//EN"  
 "http://hibernate.sourceforge.net/hibernate-configuration-5.3.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">your\_password</property>  
  
 <!-- JDBC connection pool -->  
 <property name="hibernate.connection.pool\_size">10</property>  
  
 <!-- SQL dialect -->  
 <property name="hibernate.dialect">org.hibernate.dialect.MySQLDialect</property>  
  
 <!-- Show SQL in console -->  
 <property name="hibernate.show\_sql">true</property>  
 <property name="hibernate.format\_sql">true</property>  
  
 <!-- Automatically create/update schema -->  
 <property name="hibernate.hbm2ddl.auto">update</property>  
  
 <!-- Mapping class -->  
 <mapping class="com.example.model.Student"/>  
 </session-factory>  
 </hibernate-configuration>

## 

## 

## **3. Entity Class (POJO)Plain Old Java Object,**

package com.example.model;  
  
 import jakarta.persistence.\*;  
  
 @Entity  
 @Table(name = "students")  
 public class Student {  
 @Id  
 @GeneratedValue(strategy = GenerationType.IDENTITY)  
 private int id;  
  
 @Column(name = "name", nullable = false)  
 private String name;  
  
 @Column(name = "email", unique = true)  
 private String email;  
  
 public Student() {}  
  
 public Student(String name, String email) {  
 this.name = name;  
 this.email = email;  
 }  
  
 // 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 String getEmail() { return email; }  
 public void setEmail(String email) { this.email = email; }  
 }

## 

## 

## 

## **4. Hibernate Utility Class**

package com.example.util;  
  
 import org.hibernate.SessionFactory;  
 import org.hibernate.cfg.Configuration;  
  
 public class HibernateUtil {  
 private static final SessionFactory sessionFactory;  
  
 static {  
 try {  
 sessionFactory = new Configuration()  
 .configure("hibernate.cfg.xml")  
 .buildSessionFactory();  
 } catch (Throwable ex) {  
 System.err.println("SessionFactory creation failed." + ex);  
 throw new ExceptionInInitializerError(ex);  
 }  
 }  
  
 public static SessionFactory getSessionFactory() {  
 return sessionFactory;  
 }  
 }

## **5. Main Class (Insert Data Example)**

package com.example;  
  
 import com.example.model.Student;  
 import com.example.util.HibernateUtil;  
 import org.hibernate.Session;  
 import org.hibernate.Transaction;  
  
 public class MainApp {  
 public static void main(String[] args) {  
 Student student = new Student("Vikash Dubey", "vikash@example.com");  
  
 Session session = HibernateUtil.getSessionFactory().openSession();  
 Transaction tx = null;  
  
 try {  
 tx = session.beginTransaction();  
 session.save(student);  
 tx.commit();  
 System.out.println("Student saved successfully!");  
 } catch (Exception e) {  
 if (tx != null) tx.rollback();  
 e.printStackTrace();  
 } finally {  
 session.close();  
 }  
 }  
 }