***Hands on 3 : Hibernate Annotation Config implementation walk through***

**1. Object-Relational Mapping (ORM) using Annotations in Employee Class**

In annotation-based Hibernate configuration, the mapping between the Java class and the database table is done **directly inside the class file** using **JPA annotations**. There is **no need for separate .hbm.xml mapping files**.

**Sample Employee.java**

import javax.persistence.\*;

@Entity

@Table(name = "EMPLOYEE")

public class Employee {

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

@Column(name = "id")

private int id;

@Column(name = "first\_name")

private String firstName;

@Column(name = "last\_name")

private String lastName;

@Column(name = "salary")

private int salary;

// Constructors, Getters, Setters

}

**How Mapping is Done:**

| **Annotation** | **Purpose** |
| --- | --- |
| @Entity | Marks this class as a persistent entity (maps to a table) |
| @Table(name="EMPLOYEE") | Specifies the table name in the DB |
| @Id | Denotes the primary key field |
| @GeneratedValue | Tells Hibernate to auto-generate the primary key |
| @Column(name="...") | Maps Java field to a specific column name in the table |

This mapping happens **at runtime** and Hibernate uses reflection to persist these fields.

**2. Explanation of Hibernate Annotations**

**🔹 @Entity**

* Declares the class as an entity that should be persisted to a database table.
* Required on every persistent class.

**🔹 @Table(name = "EMPLOYEE")**

* Maps the class to a table named EMPLOYEE.
* Optional: If not specified, Hibernate uses the class name as the table name.

**🔹 @Id**

* Marks the primary key field in the entity.

**🔹 @GeneratedValue(strategy = GenerationType.IDENTITY)**

* Specifies how the primary key is auto-generated.
* Strategies:
  + IDENTITY: Uses auto-increment column.
  + SEQUENCE: Uses DB sequence.
  + AUTO: Hibernate picks based on DB dialect.

**🔹 @Column(name = "column\_name")**

* Maps the field to a DB column.
* Optional: If not used, Hibernate uses the field name as the column name.

**3. Hibernate Configuration: hibernate.cfg.xml**

This XML file configures:

* DB connection
* Hibernate dialect
* Mapped classes

**Sample hibernate.cfg.xml:**

<!DOCTYPE hibernate-configuration PUBLIC

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

"http://www.hibernate.org/dtd/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/test</property>

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

<property name="hibernate.connection.password">your\_password</property>

<!-- JDBC Dialect -->

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

<!-- Echo SQL statements -->

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

<!-- Register Annotated Class -->

<mapping class="Employee"/>

</session-factory>

</hibernate-configuration>

**🔷 Explanation of Key Properties:**

| **Property** | **Purpose** |
| --- | --- |
| hibernate.connection.driver\_class | Specifies the JDBC driver |
| hibernate.connection.url | The DB connection string (host, port, DB name) |
| hibernate.connection.username / password | Credentials to connect to DB |
| hibernate.dialect | Tells Hibernate how to generate SQL for specific DB (e.g., MySQLDialect, OracleDialect) |
| hibernate.show\_sql | Prints SQL queries to console for debugging |
| <mapping class="Employee"/> | Registers the annotated class (fully qualified if in a package) |

**Example Workflow in Code**

Configuration cfg = new Configuration().configure();

SessionFactory factory = cfg.buildSessionFactory();

Session session = factory.openSession();

Transaction tx = session.beginTransaction();

Employee e = new Employee("John", "Doe", 3000);

session.save(e); // persist object

tx.commit(); // save changes

session.close(); // close session

**🟩 Summary Table**

| **Element** | **Role** |
| --- | --- |
| @Entity | Declares a class as an entity |
| @Table | Maps entity to a DB table |
| @Id | Declares the primary key |
| @GeneratedValue | Auto-generates primary key |
| @Column | Maps field to DB column |
| hibernate.cfg.xml | Configures DB, dialect, and mappings |
| Dialect | Translates HQL to DB-specific SQL |
| Driver | JDBC driver for the database |
| URL | DB connection path |
| Username/Password | DB login credentials |