**Hands-On 3:**

**Hibernate Annotation Configuration Walkthrough:**

**How Object-to-Relational Mapping is Done in the Persistence Class (Employee.java):**

In annotation-based Hibernate, the mapping between Java class and database table is embedded directly in the Java code using annotations, instead of external XML files.

**example:**

@Entity

@Table(name = "EMPLOYEE")

public class Employee {

@Id

@GeneratedValue

@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;

}

**Here:**

* The class is marked with @Entity, telling Hibernate it represents a table.
* The table name is specified with @Table.
* Each field in the class is mapped to a column using @Column.
* The primary key is indicated with @Id, and @GeneratedValue specifies how the primary key is auto-generated.

**So:** Java Class → Database Table   
 Field → Table Column   
 @Id → Primary Key   
 @GeneratedValue → Auto-increment/sequence   
 @Column → Specific column mapping

**End-to-End Hibernate Operations: Key Annotations & Config:**

**@Entity:**

* Declares that this class is a **persistent entity**, meaning Hibernate should manage it and map it to a table.Without this annotation, Hibernate ignores the class.

**Example**

@Entity

public class Employee { }

**@Table:**

* Specifies the **table name** (optional if the class name and table name are the same).Helps when the table name differs from the class name.

@Table(name = "EMPLOYEE")

If omitted, Hibernate assumes the table name is Employee (the class name).

**@Id:**

* Marks a field as the **primary key** of the table.  
  Without @Id, Hibernate won’t know how to uniquely identify records.

**Example**

@Id

private int id;

**@GeneratedValue:**

* Specifies the strategy Hibernate should use to generate the primary key value automatically.  
  Common strategies: AUTO, IDENTITY, SEQUENCE, TABLE.

**Example**

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

private int id;

In the example above, Hibernate delegates primary key generation to the database (like auto-increment).

**@Column:**

* Maps a field in Java to a column in the database.Optional if field name matches column name.

**Example**

@Column(name = "FIRST\_NAME")

private String firstName;

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

Even though mapping is done using annotations, Hibernate still needs to know:

* Which database to connect to
* Credentials
* Dialect (SQL flavor)
* Driver
* Other settings

These are defined in the **hibernate.cfg.xml** file.

**Sample hibernate.cfg.xml**

<?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.cj.jdbc.Driver</property>

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

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

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

<mapping class="com.example.Employee"/>

</session-factory>

</hibernate-configuration>

**Explanation of Each Part**

**Dialect:**

Specifies which SQL dialect Hibernate should use to generate SQL queries appropriate for the target database.  
**Examples:**

* MySQL → org.hibernate.dialect.MySQLDialect
* Oracle → org.hibernate.dialect.Oracle10gDialect
* PostgreSQL → org.hibernate.dialect.PostgreSQLDialect

**Driver:**

Specifies the JDBC driver class that Hibernate will use to connect to the database.  
**Examples:**

* MySQL → com.mysql.cj.jdbc.Driver
* Oracle → oracle.jdbc.driver.OracleDriver

**Connection URL:**

* Specifies the full JDBC URL for the database connection.

**For MySQL:**

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

**Username:**

* Specifies the database user Hibernate will use to log in.

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

**Password:**

* Specifies the password for the database user.

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

**Example End-to-End Workflow:**

**Sample Code:**

SessionFactory factory = new Configuration()

.configure("hibernate.cfg.xml")

.buildSessionFactory();

Session session = factory.openSession();

Transaction tx = null;

try {

tx = session.beginTransaction();

Employee emp = new Employee();

emp.setFirstName("John");

emp.setLastName("Doe");

emp.setSalary(5000);

session.save(emp);

tx.commit();

} catch (Exception e) {

if (tx != null) tx.rollback();

e.printStackTrace();

} finally {

session.close();

factory.close();

}