**Hands-on 2: Hibernate XML Configuration - Explanation**

**Objective:**

Understand how Hibernate performs Object-Relational Mapping (ORM) using XML configuration and how basic CRUD operations are managed using core Hibernate API.

**Object-to-Relational Mapping in XML**

In Hibernate, mapping between Java objects and database tables can be done using XML configuration files. The main file for this purpose is the .hbm.xml mapping file.

**Example: Employee.hbm.xml**

<hibernate-mapping>

<class name="com.example.Employee" table="employee">

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

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

<property name="department" column="department"/>

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

</class>

</hibernate-mapping>

**Explanation:**

* <class>: Maps a Java class to a DB table.
* <id>: Maps the primary key field.
* <property>: Maps other fields to DB columns.

This configuration tells Hibernate how to convert Java objects into database rows and vice versa.

**Hibernate Concepts:**

**1. SessionFactory**

* A heavyweight, thread-safe object created once per application.
* Built using hibernate.cfg.xml.
* Used to obtain Session objects.

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

**2. Session**

* A lightweight object used for database operations.
* Not thread-safe.
* Similar to a JDBC connection.

Session session = sessionFactory.openSession();

**3. Transaction**

* Represents a single unit of work.
* Ensures all DB changes are committed or rolled back.

Transaction tx = session.beginTransaction();

**4. beginTransaction()**

* Begins a new transaction.
* Required before executing update operations.

Transaction tx = session.beginTransaction();

**5. commit()**

* Commits the transaction.
* All changes made in the session are saved to the DB.

tx.commit();

**6. rollback()**

* Used to undo changes if an error occurs during transaction.

tx.rollback();

**7. session.save(Object)**

* Saves a new object to the database.

Employee emp = new Employee();

emp.setId(1);

emp.setName("John");

session.save(emp);

**8. session.createQuery().list()**

* Executes an HQL query and returns results as a list.

List<Employee> employees = session.createQuery("FROM Employee", Employee.class).list();

**9. session.get(Class, id)**

* Retrieves an object by primary key.
* Returns null if not found.

Employee emp = session.get(Employee.class, 1);

**10. session.delete(Object)**

* Deletes a record from the DB.

Employee emp = session.get(Employee.class, 1);

session.delete(emp);