**Hibernate XML Configuration**

This hands-on covers the basics of how Hibernate works using XML configuration. We’ll explore how Java objects are mapped to database tables, and understand key operations like saving, fetching, and deleting records using Hibernate.

**What is Object-Relational Mapping (ORM) in Hibernate?**

Hibernate is an ORM framework that lets us **map Java classes to database tables**, so we can work with objects in our code instead of writing raw SQL queries.

In XML-based configuration, this mapping is defined in a .hbm.xml file.

**Example: Mapping a Java class to a table**

<hibernate-mapping>

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

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

<generator class="increment"/>

</id>

<property name="firstName" column="first\_name"/>

<property name="lastName" column="last\_name"/>

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

</class>

</hibernate-mapping>

Here:

* The class Employee is mapped to the employee table.
* Each field (like firstName) is mapped to a column (first\_name) in the table.
* The id tag defines the primary key.

**How Hibernate Works Internally**

Hibernate works in a few steps — it uses a session to connect to the DB, perform operations, and manage transactions. Here's how the flow works:

**1. SessionFactory**

* This is created only once in the application.
* It reads the configuration file and sets up the connection with the database.

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

**2. Session**

* Created from the SessionFactory.
* This is used to interact with the database — insert, fetch, delete, etc.

Session session = factory.openSession();

**3. Transaction**

* Used to group database operations together.
* If everything goes fine, we commit the transaction. If something fails, we roll it back.

**Common Hibernate Operations (With Explanation)**

| **Operation** | **What It Does** |
| --- | --- |
| beginTransaction() | Starts a new transaction before DB operations. |
| commit() | Confirms and saves the changes to the DB. |
| rollback() | Reverses changes if something goes wrong. |
| session.save(object) | Adds a new record (insert) to the database. |
| session.createQuery().list() | Executes a query and returns results as a list. |
| session.get(Class, id) | Retrieves a specific object from DB using its primary key. |
| session.delete(object) | Deletes the record from the DB that matches the given object. |

**Example Code:**

Session session = factory.openSession();

Transaction tx = null;

try {

tx = session.beginTransaction();

Employee emp = new Employee("John", "Doe", 50000);

session.save(emp); // INSERT operation

tx.commit();

} catch (Exception e) {

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

e.printStackTrace();

} finally {

session.close();

}

**Configuration File: hibernate.cfg.xml**

This file tells Hibernate how to connect to the database and where to find mapping files.

<hibernate-configuration>

<session-factory>

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

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

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

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

<property name="hibernate.dialect">org.hibernate.dialect.MySQL5Dialect</property>

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

<!-- Mapping file -->

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

</session-factory>

</hibernate-configuration>