**Hibernate XML Configuration - Comprehensive Guide**

This guide provides a detailed explanation of Hibernate's XML-based configuration, covering Object-Relational Mapping (ORM) and key operations like SessionFactory, Session, Transaction, and CRUD operations.

**1. Object-Relational Mapping (ORM) in Hibernate**

Hibernate is an ORM framework that maps Java objects to database tables, eliminating the need for manual SQL queries. The mapping is defined in an XML file (.hbm.xml).

**Example: Employee Class Mapping**

<hibernate-mapping>

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

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

<generator class="increment"/> <!-- Auto-generates ID -->

</id>

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

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

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

</class>

</hibernate-mapping>

**Key Elements**

| **Element** | **Description** |
| --- | --- |
| <class> | Maps a Java class (Employee) to a database table (employee). |
| <id> | Defines the primary key (id) with a generation strategy (increment). |
| <property> | Maps a Java field (firstName) to a database column (first\_name). |

**2. Hibernate Core Components**

**1. SessionFactory**

* **Purpose**: Heavyweight, thread-safe object that reads Hibernate configuration and creates database connections.
* **Usage**: Created once per application (expensive operation).
* **Configuration File**: Uses hibernate.cfg.xml.

SessionFactory factory = new Configuration()

.configure("hibernate.cfg.xml")

.buildSessionFactory();

**2. Session**

* **Purpose**: Lightweight, short-lived object representing a single database connection.
* **Usage**: Used for CRUD operations like save(), get(), delete().
* **Thread Safety**: Not thread-safe (open and close per operation).

Session session = factory.openSession();

**3. Transaction**

* **Purpose**: Groups DB operations into a single atomic unit.
* **ACID Compliance**: Ensures atomicity, consistency, isolation, and durability.
* **Commit or Rollback**: Must commit changes or roll back on error.

Transaction tx = session.beginTransaction();

try {

// Perform DB operations

tx.commit(); // Save changes

} catch (Exception e) {

tx.rollback(); // Undo changes on failure

} finally {

session.close();

}

**3. Key Hibernate Operations**

| **Operation** | **Description** |
| --- | --- |
| session.beginTransaction() | Starts a new transaction. |
| tx.commit() | Saves changes permanently to the database. |
| tx.rollback() | Reverts changes if an error occurs. |
| session.save(object) | Inserts a new record into the database. |
| session.get(Class, id) | Retrieves an object by its primary key. |
| session.createQuery("HQL").list() | Executes an HQL query and returns results as a list. |
| session.delete(object) | Removes a record from the database. |

**Example: CRUD Operations**

Session session = factory.openSession();

Transaction tx = null;

try {

tx = session.beginTransaction();

// Insert

Employee emp = new Employee("Gopika", "K S", 50000);

session.save(emp);

// Fetch

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

// Update

fetchedEmp.setSalary(60000);

session.update(fetchedEmp);

// Delete

session.delete(fetchedEmp);

tx.commit();

} catch (Exception e) {

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

e.printStackTrace();

} finally {

session.close();}

**4. Hibernate Configuration (hibernate.cfg.xml)**

Defines database connection settings and mapping files.

<hibernate-configuration>

<session-factory>

<!-- Database Connection -->

<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>

<!-- Hibernate Settings -->

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

<property name="show\_sql">true</property> <!-- Logs SQL queries -->

<!-- Mapping Files -->

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

</session-factory>

</hibernate-configuration>

**Key Properties**

| **Property** | **Description** |
| --- | --- |
| hibernate.connection.driver\_class | JDBC driver for the database. |
| hibernate.connection.url | Database connection URL. |
| hibernate.dialect | Optimizes SQL generation for the database. |
| show\_sql | Logs generated SQL queries for debugging. |

**5. Summary**

* **ORM Mapping**: Defined in .hbm.xml files to map Java objects to database tables.
* **SessionFactory**: Singleton object for managing DB connections.
* **Session**: Short-lived object used for DB operations.
* **Transaction**: Ensures data integrity using commit() and rollback().
* **CRUD Operations**: Use save(), get(), delete(), createQuery() for interactions.