**Hands-on 3: Hibernate Annotation Configuration – Explanation**

**1. How object-to-relational database mapping is done in the persistence class file Employee**

Hibernate uses annotations directly in the Java class to define the mapping between the class and the database table. This eliminates the need for external mapping files like .hbm.xml. Fields in the class are annotated to specify their correspondence with columns in the database.

**2. End-to-end operations in Hibernate using annotations:**

* **@Entity**
  + Marks the class as a persistent entity.
  + Required for Hibernate to recognize the class as something it should map to a database table.
* **@Table**
  + Specifies the database table name that the class maps to.
  + Optional; if not used, Hibernate assumes the table name is the same as the class name.
* **@Id**
  + Identifies the primary key field in the class.
  + Required for each entity.
* **@GeneratedValue**
  + Defines how the primary key is generated.
  + Common strategies: AUTO, IDENTITY, SEQUENCE, TABLE.
* **@Column**
  + Maps a field to a column in the table.
  + Can specify column name, length, nullable, and other constraints.

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

This file contains all database connection settings and class mappings.

* **Dialect**
  + Example: org.hibernate.dialect.MySQLDialect
  + Tells Hibernate what SQL dialect to generate (based on your DB).
* **Driver**
  + JDBC driver class used to connect to the DB.
  + Example: com.mysql.cj.jdbc.Driver
* **Connection URL**
  + JDBC URL that points to the database.
  + Example: jdbc:mysql://localhost:3306/ormlearn
* **Username**
  + DB username, e.g., root
* **Password**
  + DB password, e.g., root

The annotated class is registered in hibernate.cfg.xml using:

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

This enables Hibernate to read mapping information directly from the class.