# Hands-on 2: Hibernate XML Config Implementation Walkthrough

## 1. Object to Relational Database Mapping in Hibernate XML

Hibernate uses XML configuration files to define how Java objects are mapped to database tables.   
The mapping is done using a `.hbm.xml` file which specifies class-to-table and field-to-column mappings.

Example: Employee.hbm.xml

<hibernate-mapping>  
 <class name="Employee" table="EMPLOYEE">  
 <id name="id" type="int" column="ID">  
 <generator class="native"/>  
 </id>  
 <property name="firstName" column="FIRST\_NAME" type="string"/>  
 <property name="lastName" column="LAST\_NAME" type="string"/>  
 <property name="salary" column="SALARY" type="float"/>  
 </class>  
</hibernate-mapping>

## 2. End-to-End Hibernate Operations

### 2.1 SessionFactory

It is a factory for Session objects and is created once per application. It is responsible for reading configuration and providing sessions.

### 2.2 Session

A lightweight, single-threaded object used to interact with the database. It provides methods to create, retrieve, update, and delete operations.

### 2.3 Transaction

Used to wrap a series of operations into a single unit of work. It ensures that either all changes are committed or none are.

### 2.4 beginTransaction()

Starts a new database transaction.

### 2.5 commit()

Commits the current transaction, making all changes permanent in the database.

### 2.6 rollback()

Reverts changes made during the current transaction if an error occurs.

### 2.7 session.save()

Saves a transient object to the database and returns the generated identifier.

### 2.8 session.createQuery().list()

Executes a HQL query and returns a list of results.

### 2.9 session.get()

Fetches a persistent object based on its primary key. Returns null if not found.

### 2.10 session.delete()

Removes a persistent object from the database.