# Spring data JPA hands on Objectives

# Need and Benefit of ORM

ORM means Object Relational Mapping. It helps us connect our Java objects to database tables easily. We don’t have to write long SQL queries. ORM tools like Hibernate manage this for us. It makes coding faster, cleaner, and safe from SQL injection if used properly. For example, when we save an object, ORM makes the query automatically.

**The need and benefit of Spring Data JPA**

Spring Data JPA is built on top of JPA and Hibernate. It saves us from writing a lot of boilerplate code. We can just use simple functions like .save() or .findById() without writing SQL. It also supports different databases (like MySQL, PostgreSQL, H2). We used Spring Data JPA in Hands-on 1 to 9 to do insert, update, delete operations easily.

# Core Objects in Hibernate

Hibernate uses important objects to talk to the database:  
***@SessionFactory:*** Creates sessions (like connections).  
***@Session:*** The main object used to save, update, get, or delete data.  
***@Transaction:*** Helps us do actions in a safe way (with commit/rollback).  
***@ConnectionProvider***: Gives the database connection to Hibernate.  
These were used in XML and annotation examples in the tutorials.

# Hibernate Configuration - XML vs Annotation

In old days, we wrote XML files to map Java classes to tables.  
  
XML way: define classes and table mappings in .hbm.xml and .cfg.xml.  
Annotation way: use @Entity, @Table, @Id, @Column in Java class directly.  
  
In both methods, we do:  
1. Load the configuration  
2. Get session  
3. Start transaction  
4. Do operation  
5. Commit and close  
  
In Spring Data JPA, we don’t need to do all these manually it’s automatic.

# Difference: JPA vs Hibernate vs Spring Data JPA

|  |  |  |  |
| --- | --- | --- | --- |
| Feature | JPA | Hibernate | Spring Data JPA |
| What it is | Just a specification | An ORM tool (actual code) | A wrapper over Hibernate & JPA |
| Has code? | No | Yes | Yes, but uses Hibernate underneath |
| Ease of use | Hard (need more coding) | Medium | Easy (very less code) |
| Query writing | Manual | HQL | Mostly automatic using method names |

# DML Operations Using Spring Data JPA

We did Insert, Read, Update, Delete (CRUD) using repository methods:  
  
save() - insert or update  
findById() - search by ID  
findAll() - get all data  
deleteById() - delete a row  
  
Spring handled all queries using these simple methods, with no need for SQL.  
We also configured ddl-auto = validate to make sure tables and columns exist.