**WEEK:3**

Demonstrate the need and benefit of Spring Data JPA

Evolution of ORM solutions, Hibernate XML Configuration, Hibernate Annotation Configuration, Spring Data JPA, Hibernate benefits, open source, light weight, database independent query

With H2 in memory database

With MySQL

XML Configuration Example

Hibernate Configuration Example

Need of Spring Data JPA:

Spring Data JPA was introduced to address the common challenges faced when using plain JPA or Hibernate in enterprise applications:

In traditional JPA or Hibernate, developers must write the same CRUD (Create, Read, Update, Delete) methods for each entity in DAO or repository classes.

This leads to duplication and increases the risk of errors.

**Lack of Built-in Features**

Features like pagination, sorting, and dynamic query generation are not directly supported by plain JPA/Hibernate.These must be implemented manually, adding extra work.

**Benefits of Spring Data JPA:**

**•Automatic Repository Implementation**

**• Dynamic Query Methods**

**•Support for Custom Queries**

**• Built-in Pagination and Sorting**

**•Seamless Transaction Management**

**•Database Independence**

**Evolution of ORM Solutions**

**Manual JDBC (Before ORM)**

* Direct SQL queries in Java code
* Boilerplate code for connection, statements, result sets
* Error-prone and hard to maintain

**Hibernate (ORM Framework)**

**Hibernate with XML Configuration**

* Entity mapping was done using \*.hbm.xml files
* Tedious to maintain large XMLs

**Hibernate with Annotation Configuration**

* Mapping moved into Java classes using annotations
* Cleaner and easier to maintain

EXAMPLE:

@Entity

@Table(name = "EMPLOYEE")

public class Employee {

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

private Long id;

@Column(name = "EMP\_NAME")

private String name;

}