Week – 4

Spring Core And Maven

Spring Data JPA –

Spring Data JPA is a key part of the Spring Data umbrella, designed to simplify working with databases in Java applications using JPA (Java Persistence API). It reduces boilerplate code and allows developers to focus more on business logic than plumbing code. It's especially powerful when used alongside Spring Boot, enabling features like query creation by method name, pagination, and sorting with minimal configuration.

Quick Example • 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 - https://www.mkyong.com/springboot/spring-boot-spring-data-jpa/ ▪ With MySQL - https://www.mkyong.com/spring-boot/spring-bootspring-data-jpa-mysql-example/ ▪ XML Configuration Example - https://www.tutorialspoint.com/hibernate/hibernate_examples.htm ▪ Hibernate Configuration Example - https://www.tutorialspoint.com/hibernate/hibernate_annotations.html

---

Why Do We Need Spring Data JPA?

Traditional data access layers required manual implementation of CRUD operations, often resulting in a lot of repetitive code. Spring Data JPA addresses this problem by:

- **Auto-generating DAO layer logic** through repository interfaces.
- **Creating queries from method names** (e.g., `findByTitle()`).
- **Simplifying database access** with minimal effort.
- **Integrating seamlessly with Spring Boot** and modern databases.
- **Supporting pagination, sorting, and custom queries** out of the box.

Advantages of Spring Data JPA

- ✓ **Open Source** and actively maintained by the Spring team.
- ✓ **Lightweight** and requires minimal setup.
- ✓ **Query methods based on naming conventions** – no need to write SQL.
- ✓ **Supports multiple databases** – H2, MySQL, PostgreSQL, etc.
- ✓ **Minimal configuration** – often just a few properties in `application.properties`.

Evolution of ORM in Java

1. JDBC (Java Database Connectivity)

- Manual SQL handling and result mapping.
- Verbose, repetitive, and error-prone.

2. Hibernate with XML Mapping

- Used external XML files to map Java objects to database tables.
- Flexible but tedious and hard to maintain.

3. Hibernate with Annotations

- Introduced annotations like `@Entity`, `@Id`, `@Table`, etc.
- Simplified mapping and improved readability.

4. Spring with Hibernate

- Combined Spring's dependency injection with Hibernate ORM.
- Improved transaction management, but DAOs still needed manual implementation.

5. Spring Data JPA

- Eliminated the need for DAO implementation.
- Uses repository interfaces to abstract data access logic.
- Greatly reduces code and improves maintainability.

Example: Spring Data JPA with Book Entity

```
java
CopyEdit
@Entity
public class Book {
    @Id
    @GeneratedValue
    private Long id;
    private String title;
    private String author;
}
java
CopyEdit
@Repository
public interface BookRepository extends JpaRepository<Book, Long> {
    List<Book> findByAuthor(String author);
}
java
CopyEdit
@RestController
public class BookController {
    @Autowired
    private BookRepository bookRepository;

    @GetMapping("/books")
    public List<Book> getBooks() {
        return bookRepository.findAll();
    }
}
```

JPA vs Hibernate vs Spring Data JPA

Feature	JPA	Hibernate	Spring Data JPA
Type	Specification	Framework (JPA implementation)	Framework (built on JPA)
Role	Defines ORM behavior in Java	Implements JPA and offers extra features	Simplifies repository and CRUD operations
Configuration	Annotations (no implementation)	XML or Annotations	Mostly via `application.properties`
Features	Entity mapping, relationships	HQL, caching, lazy loading, advanced tuning	Auto-generated queries, minimal DAO code
Dependency	Requires provider like Hibernate	Can run independently or under JPA	Depends on JPA & provider (usually Hibernate)

- 🔹 **With H2 (In-Memory DB)**:
  Spring Boot + Spring Data JPA + H2
- 🔹 **With MySQL**:
  Spring Boot + Spring Data JPA + MySQL
- 🔹 **Hibernate XML Configuration Example**:
  TutorialsPoint Hibernate XML
- 🔹 **Hibernate Annotation Example**:
  TutorialsPoint Hibernate Annotations

Spring Data JPA revolutionizes how we interact with databases in Spring applications. It leverages the power of JPA and Hibernate while reducing the complexity of writing data access logic. With features like auto-generated repositories, support for various databases, and tight Spring integration, it's a go-to choice for building robust, scalable backend systems.