**CTS DIGITAL NURTURE - 4.0 JAVA FSE**

**WEEK 3 -Spring Core and Maven**

**Spring Data JPA with Spring Boot, Hibernate**

**Difference between JPA, Hibernate and Spring Data JPA**

**1. Java Persistence API (JPA)**

* **JPA is a specification**, not a framework.
* Defined under **JSR 338**.
* It defines **standard interfaces and annotations** (like @Entity, @Id, @OneToMany) for **object-relational mapping (ORM)** in Java.

**2. Hibernate**

* **Hibernate is a popular implementation** of the JPA specification.
* It is an **ORM tool** that handles:
  + Mapping Java objects to database tables
  + Generating SQL
  + Caching and performance tuning
* It adds features **beyond JPA**, such as:
  + Custom query languages (HQL)
  + Lazy loading
  + Second-level caching

**3. Spring Data JPA**

* **Spring Data JPA is an abstraction** built on top of JPA (and typically Hibernate).
* It **simplifies** the process of working with the database.
* It provides:
  + Predefined repository interfaces (like JpaRepository)
  + Auto-generated query methods (e.g., findByName())
  + Reduces boilerplate code

| **Feature** | **JPA** | **Hibernate** | **Spring Data JPA** |
| --- | --- | --- | --- |
| Type | Specification | Implementation | Abstraction Framework |
| Purpose | Standard API | ORM Engine | Simplify JPA/Hibernate usage |
| Requires Code? | Yes | Yes | Less code needed (repo-based) |
| Custom Queries | No | Yes (HQL, Criteria) | Yes (method names or @Query) |
| Manages DB Access? | No | Yes | Yes |

**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/spring-boot/spring-boot-spring-data-jpa/**
    - **With MySQL - https://www.mkyong.com/spring-boot/spring-boot-spring-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.htm**

**Evolution of ORM Solutions**

**1. Hibernate with XML Configuration**

* Early Hibernate versions required **manual XML files** for mapping classes to tables.
* Tedious and error-prone (hibernate.cfg.xml, \*.hbm.xml).

**2. Hibernate with Annotations**

* Introduced annotations like @Entity, @Table, @Id.
* Removed the need for separate XML files.
* Still required boilerplate code: SessionFactory, transactions, etc.

**3. Spring + Hibernate**

* Spring integrated Hibernate and managed sessions/transactions via Spring beans.
* Reduced complexity but still required writing **DAO classes**.

**Why Spring Data JPA?**

**Key Benefits:**

1. **Minimal Code**: No need to write boilerplate CRUD methods.  
   Just extend JpaRepository or CrudRepository.
2. **Auto Query Generation**:  
   Methods like findByName(), findByDepartmentAndSalaryGreaterThan() work **without writing SQL/HQL**.
3. **Database Independent**:  
   Easily switch between H2, MySQL, PostgreSQL by changing DB config only.
4. **Integration with Spring Boot**:  
   Auto-configures DataSource, EntityManager, TransactionManager.
5. **Custom Queries**: Supports native SQL and JPQL via @Query.
6. **Pagination, Sorting, Auditing**: Built-in support with minimal setup.

**Example Use Case with H2 (In-Memory)**

* No need to install external DB.
* Easy to test.
* No setup overhead.

spring.datasource.url=jdbc:h2:mem:testdb

spring.h2.console.enabled=true

**Example Use Case with MySQL**

spring.datasource.url=jdbc:mysql://localhost:3306/employees

spring.datasource.username=root

spring.datasource.password=1234

spring.jpa.hibernate.ddl-auto=update

| **Feature** | **Hibernate (Old)** | **Spring Data JPA** |
| --- | --- | --- |
| Mapping Config | XML / Annotations | Annotations |
| DAO Layer | Manual | Auto-generated |
| Transactions | Manual | Handled by Spring |
| Query Writing | HQL / SQL | Auto / @Query methods |
| DB Switching | Manual config changes | Just change application.properties |

Code Comparison:

**Hibernet:**

public Integer addEmployee(Employee employee) {

Session session = factory.openSession();

Transaction tx = null;

Integer employeeID = null;

try {

tx = session.beginTransaction();

employeeID = (Integer) session.save(employee);

tx.commit();

} catch (HibernateException e) {

if (tx != null) tx.rollback();

e.printStackTrace();

} finally {

session.close();

}

return employeeID;

}

**Spring Data JPA Style (Auto)**

**EmployeeRepository.java**

public interface EmployeeRepository extends JpaRepository<Employee, Integer> {

}

**EmployeeService.java**

@Autowired

private EmployeeRepository employeeRepository;

@Transactional

public void addEmployee(Employee employee) {

employeeRepository.save(employee);

}