Week 3: Spring Data JPA with Spring Boot, Hibernate

Student Name: Jitesh Sen

Email: [jiteshsen2002@gmail.com](mailto:jiteshsen2002@gmail.com)

Superset ID: 6387362

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

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

* JPA is a *specification* (defined in JSR 338), which means it sets the rules and guidelines for persisting Java objects to relational databases.
* It provides interfaces and annotations, but **no concrete implementation**.
* Think of JPA like a blueprint or contract.
* Examples of JPA annotations: @Entity, @Table, @Id.
* **Sample JPA entity (pure JPA annotations):**

import javax.persistence.Entity;

import javax.persistence.Id;

@Entity

public class Student {

@Id

private Long id;

private String name;

// getters/setters

}

Above, @Entity and @Id come from JPA, not from any implementation. You cannot persist without a provider (like Hibernate) behind it.

1. **Hibernate**

* Hibernate is the **most popular implementation** of the JPA specification.
* In other words, Hibernate takes the JPA blueprint and actually builds the working machinery under it.
* Apart from following JPA, Hibernate has many extra features, like its own HQL (Hibernate Query Language), caching support, and advanced fetching strategies.
* Hibernate is an **ORM (Object-Relational Mapping) framework**, which helps map Java classes to database tables and handles the underlying SQL automatically.
* **Sample Hibernate style**  
  (using SessionFactory, purely Hibernate API — no Spring):

SessionFactory factory = new Configuration().configure().buildSessionFactory();

Session session = factory.openSession();

Transaction tx = session.beginTransaction();

Student s = new Student();

s.setId(1L);

s.setName("Alice");

session.save(s);

tx.commit();

session.close();

This style uses Hibernate’s native Session instead of a JPA EntityManager.

1. **Spring Data JPA**

* Spring Data JPA is a **high-level abstraction on top of JPA + Hibernate**.
* It dramatically reduces boilerplate code by providing built-in repositories (like JpaRepository) so you don’t have to write common CRUD methods.
* Spring Data JPA focuses on simplifying development by supporting query methods, pagination, sorting, custom queries with @Query, and even specifications.
* Internally, Spring Data JPA still uses JPA and a provider like Hibernate, but gives you a more developer-friendly and standardized Spring programming model.

@Repository

public interface StudentRepository extends JpaRepository<Student, Long> {

List<Student> findByNameContaining(String keyword);

}

@Autowired

private StudentRepository studentRepository;

public void addStudent() {

Student s = new Student();

s.setId(2L);

s.setName("Bob");

studentRepository.save(s);

}

This is **Spring Data JPA**: you do not manually create Session or EntityManager; Spring injects everything for you.

**In short**

* JPA: defines how to persist objects (standard specification)
* Hibernate: implements JPA (plus some advanced extras)
* Spring Data JPA: uses JPA + Hibernate and simplifies your development with auto-generated repositories and much less code