Difference between JPA, Hibernate, and Spring Data JPA

# 1. Conceptual Differences

|  |  |  |
| --- | --- | --- |
| Technology | Description | Example/Role |
| JPA | Java Persistence API — a standard for ORM in Java (just interfaces) | Driving License (interface) |
| Hibernate | JPA Implementation — actual library that performs ORM | Car (implementation) |
| Spring Data JPA | Spring’s abstraction over JPA to reduce boilerplate | Driverless Car (automation) |

# 2. Hands-on Examples

## 2.1 Common Entity Class

@Entity  
public class Product {  
 @Id  
 @GeneratedValue  
 private Long id;  
 private String name;  
 private double price;  
   
 // Getters and Setters  
}

## 2.2 Using Pure JPA

EntityManagerFactory emf = Persistence.createEntityManagerFactory("demo");  
EntityManager em = emf.createEntityManager();  
  
em.getTransaction().begin();  
Product product = new Product();  
product.setName("JPA Pen");  
product.setPrice(10.5);  
em.persist(product);  
em.getTransaction().commit();  
  
Product fetched = em.find(Product.class, product.getId());  
System.out.println(fetched.getName());

## 2.3 Using Hibernate

Configuration cfg = new Configuration().configure(); // reads hibernate.cfg.xml  
SessionFactory factory = cfg.buildSessionFactory();  
Session session = factory.openSession();  
  
Transaction tx = session.beginTransaction();  
Product product = new Product();  
product.setName("Hibernate Pen");  
product.setPrice(15.0);  
session.save(product);  
tx.commit();  
  
Product fetched = session.get(Product.class, product.getId());  
System.out.println(fetched.getName());

## 2.4 Using Spring Data JPA (Recommended)

* Repository Interface:

public interface ProductRepository extends JpaRepository<Product, Long> {  
 List<Product> findByName(String name);  
}

* Controller:

@RestController  
@RequestMapping("/products")  
public class ProductController {  
  
 @Autowired  
 private ProductRepository productRepo;  
  
 @PostMapping  
 public Product addProduct(@RequestBody Product product) {  
 return productRepo.save(product);  
 }  
  
 @GetMapping  
 public List<Product> getAll() {  
 return productRepo.findAll();  
 }  
  
 @GetMapping("/{name}")  
 public List<Product> findByName(@PathVariable String name) {  
 return productRepo.findByName(name);  
 }  
}

* application.properties:

spring.datasource.url=jdbc:mysql://localhost:3306/mydb  
spring.datasource.username=root  
spring.datasource.password=your\_pass  
spring.jpa.hibernate.ddl-auto=update

# 3. Summary Table

|  |  |  |  |
| --- | --- | --- | --- |
| Feature | JPA | Hibernate | Spring Data JPA |
| Type | API/Spec | Implementation | Framework abstraction |
| Setup | Manual | XML + manual | Auto with annotations |
| Boilerplate | High | Medium | Very Low |
| Ease of Use | 😐 | 😐 | 😍 |
| Spring Preferred | No | No | ✅ Yes |