Hands on 4: Difference between JPA, Hibernate, and Spring Data JPA

# Introduction

When working with Java applications that interact with relational databases, developers often rely on Object Relational Mapping (ORM) tools and standards to simplify database operations. This exercise explores the differences between Java Persistence API (JPA), Hibernate, and Spring Data JPA, which are commonly used technologies in this space.

# Differences Explained

## Java Persistence API (JPA)

**What is it?**

* JPA is a Java specification (JSR 338) that defines a standard for mapping Java objects to relational database tables and for managing persistent data. It only provides a set of interfaces and annotations.

**Why is it used?**

* To standardize how Java applications interact with databases in an ORM style.

**Where is it used?**

* Used in any Java applications that require data persistence through ORM. JPA does not include an implementation, so it requires a provider like Hibernate.

## Hibernate

**What is it?**

* Hibernate is a popular ORM framework and a JPA implementation provider.

**Why is it used?**

* To persist Java objects to the database and retrieve them, while implementing all JPA standards plus additional features.

**Where is it used?**

* In Java projects that require advanced ORM features, caching, lazy loading, and custom query languages like HQL.

## Spring Data JPA

**What is it?**

* Spring Data JPA is a part of the larger Spring Data project. It is not a JPA provider, but rather a framework that simplifies JPA usage by abstracting boilerplate code and repository management.

**Why is it used?**

* To rapidly build data access layers by simply defining repository interfaces without writing implementation code.

**Where is it used?**

* In Spring-based applications to quickly implement CRUD operations and complex queries.

# Comparison Table

|  |  |  |
| --- | --- | --- |
| JPA | Hibernate | Spring Data JPA |
| Is only a specification (JSR 338) for mapping Java objects to database tables. | Is an **ORM framework** that **implements the JPA specification** and adds more features. | Is an **abstraction layer over JPA providers like Hibernate**, simplifying data access in Spring. |
| Does not implement JPA, only defines it. | **Implements JPA specification**, provides actual ORM engine. | **Does not implement JPA**, relies on underlying JPA providers. |
| Requires developers to write more boilerplate code for CRUD and transactions. | Still needs some manual code like sessions and transactions, but less than pure JPA. | **Eliminates most boilerplate**, uses repositories to auto-generate CRUD. |
| Does not manage transactions, leaves it to implementations. | Manages transactions via Hibernate APIs or manual coding. | Automatically manages transactions via Spring. |
| Uses standard JPQL queries. | Uses **JPQL and adds HQL (Hibernate Query Language)** for advanced queries. | Uses **JPQL, but generates queries from method names automatically.** |
| No built-in caching (defined by spec). | Supports first and second level caching. | Delegates caching to JPA provider like Hibernate. |

# Code Snippet Comparison

## Hibernate Example

Manually handles session and transaction management:

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 Example

No manual session or transaction handling, only repository usage:

public interface EmployeeRepository extends JpaRepository<Employee, Integer> {

}

@Service

public class EmployeeService {

@Autowired

private EmployeeRepository employeeRepository;

@Transactional

public void addEmployee(Employee employee) {

employeeRepository.save(employee);

}

}

# References

* <https://dzone.com/articles/what-is-the-difference-between-hibernate-and-sprin-1>
* <https://www.javaworld.com/article/3379043/what-is-jpa-introduction-to-the-java-persistence-api.html>