# Week 3 Solution

## Difference between JPA, Hibernate and Spring Data JPA

This document summarizes the key differences between Java Persistence API (JPA), Hibernate, and Spring Data JPA in the context of a library management system implemented as part of the Week 3 assignment.

## Project Structure

The project is a Maven-based Spring Boot application with the following package structure:

src/  
├── main/  
│ └── java/  
│ └── com/  
│ └── cognizant/  
│ └── orm\_learn/  
│ ├── model/  
│ │ └── Country.java  
│ ├── service/  
│ │ └── CountryService.java  
│ └── OrmLearnApplication.java  
│  
└── test/  
 └── java/  
 └── com/  
 └── cognizant/  
 └── orm\_learn/  
 └── OrmLearnApplicationTests.java

## Technology Differences

**Java Persistence API (JPA)**

- JSR 338 Specification for persisting, reading and managing data from Java objects.  
- It does not provide a concrete implementation.  
- Requires an implementation like Hibernate or EclipseLink.

**Hibernate**

- It is an ORM (Object Relational Mapping) tool that implements JPA.  
- It provides features beyond JPA like caching, lazy loading, etc.

**Spring Data JPA**

- It is a Spring module that simplifies data access using JPA.  
- Provides a layer of abstraction over JPA providers such as Hibernate.  
- Reduces boilerplate code with repository support and derived queries.

## Code Comparison

Below are code examples comparing Hibernate and Spring Data JPA:

**Hibernate Code Example:**

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:**

public interface EmployeeRepository extends JpaRepository<Employee, Integer> { }  
  
@Service  
public class EmployeeService {  
 @Autowired  
 private EmployeeRepository employeeRepository;  
  
 @Transactional  
 public void addEmployee(Employee employee) {  
 employeeRepository.save(employee);  
 }  
}

## Compilation and Execution Commands

Use the following Maven commands to compile and run the project:

**$ mvn clean install**

**$ mvn spring-boot:run**