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

## Java Persistence API (JPA)

• JSR 338 Specification for persisting, reading, and managing data from Java objects.  
• JPA is only a specification (interface) and does not provide an implementation.  
• It defines how ORM (Object Relational Mapping) should work, but it doesn't do anything by itself.  
• Hibernate is one of the most commonly used implementations of JPA.

## Hibernate

• Hibernate is an ORM (Object Relational Mapping) framework.  
• It implements the JPA specification and provides all the tools to interact with a relational database using Java objects.  
• It allows more control and customization than plain JPA.  
• Requires manual handling of sessions and transactions.

## Spring Data JPA

• Spring Data JPA is a Spring-based framework built on top of JPA and Hibernate.  
• It simplifies database operations and reduces boilerplate code using built-in methods.  
• It abstracts the actual JPA provider (like Hibernate) and offers repository interfaces.  
• Automatically handles transactions and sessions behind the scenes.  
• Greatly improves development speed in enterprise applications.

## Comparison Table

|  |  |  |  |
| --- | --- | --- | --- |
| Feature | JPA | Hibernate | Spring Data JPA |
| Type | Specification (JSR 338) | Framework (Implementation) | Abstraction Layer |
| Provides Implementation? | No | Yes | No (Uses JPA/Hibernate) |
| Boilerplate Code | Yes | Moderate | Minimal |
| Transaction Handling | Manual | Manual | @Transactional |
| Ease of Use | Low | Moderate | High |

## Code Comparison

### Hibernate 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

EmployeeRepository.java

public interface EmployeeRepository extends JpaRepository<Employee, Integer> {  
}

EmployeeService.java

@Service  
public class EmployeeService {  
 @Autowired  
 private EmployeeRepository employeeRepository;  
  
 @Transactional  
 public void addEmployee(Employee employee) {  
 employeeRepository.save(employee);  
 }  
}

## Summary

In real-world applications, Spring Data JPA is highly preferred due to its simplicity and ability to reduce repetitive code.  
Hibernate gives more control if needed but requires more effort to manage transactions and sessions.  
JPA on its own is just a set of rules – we always need an implementation like Hibernate or EclipseLink to make it work.