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

**Java Persistence API (JPA)**

Java Persistence API is a specification (JSR 338) for accessing, persisting, and managing data between Java objects and relational databases. JPA is only an interface/standard and does not provide implementation. Hibernate is one of the most popular implementations of JPA.

**Hibernate**

Hibernate is an ORM (Object-Relational Mapping) tool. It implements JPA and provides extra features beyond the specification. It requires boilerplate code for sessions, transactions, and exception handling. Developers have to manage many low-level details manually.

**Spring Data JPA**

Spring Data JPA is a Spring module that builds on top of JPA. It does not implement JPA itself, but simplifies data access using JPA implementations like Hibernate. It provides repository interfaces that reduce boilerplate. It automatically manages transactions, queries, and standard CRUD operations. It also offers query methods via method naming conventions.

**Code Comparison**

**Using Hibernate (Manual implementation)**

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;  
}

**Using Spring Data JPA (Simplified)**

EmployeeRepository.java

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

EmployeeService.java

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