**Difference between JPA, Hibernate and Spring Data JPA**

* **Java Persistence API (JPA)**

JPA is a specification (interface) provided by Java (JSR 338) that defines how Java objects (entities) are mapped to relational database tables.  
It provides a standard way to perform operations like storing, updating, deleting, and retrieving data from a database.

* **Hibernate**

Hibernate is an Object Relational Mapping (ORM) tool and a popular implementation of JPA.  
It provides the actual code that connects Java objects to database tables, along with extra features like caching, lazy loading, and fetching strategies.

* **Spring Data JPA**

Spring Data JPA is a framework from Spring that builds on top of JPA and simplifies database operations by reducing boilerplate code.  
It allows developers to perform CRUD operations using simple method names without writing complex SQL or Hibernate logic.

**Comparison:**

|  |  |  |  |
| --- | --- | --- | --- |
| Feature | JPA (Java Persistence API) | Hibernate | Spring Data JPA |
| Type | Specification (Standard API) | JPA Implementation + ORM framework | Abstraction over JPA and ORM |
| Provides Implementation? | No | Yes | No (depends on JPA provider like Hibernate) |
| Boilerplate Code | Requires EntityManager, Queries, etc. | Requires Sessions and Transactions | Eliminates boilerplate using Repository pattern |
| Ease of Use | Manual configuration | Easier than JPA | Very easy with auto-generated methods |
| Querying Style | JPQL or Criteria API | HQL, JPQL | Method names or @Query annotations |
| Transaction Management | Manual or JTA | Manual (or with Spring) | Handled by Spring automatically |
| Integration | With any Java application | With or without Spring | Works only inside Spring Framework |

**Code Comparison:**

* **Using Hibernate**

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**

public interface EmployeeRepository extends JpaRepository<Employee, Integer> {

}

**EmployeeService.java**

@Autowired

private EmployeeRepository employeeRepository;

@Transactional

public void addEmployee(Employee employee) {

employeeRepository.save(employee);

}