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

**Java Persistence API (JPA)**:

**Java Persistence API (JPA)** is a specification (JSR 338) provided by Java for Object-Relational Mapping (ORM). It defines how Java objects should be persisted in relational databases. However, JPA is just an interface — it does not provide any actual implementation. It requires a provider (implementation) like Hibernate or EclipseLink to work.

**Hibernate:**

**Hibernate** is one of the most widely used implementations of the JPA specification. It is an ORM framework that provides the actual code to persist, update, and retrieve Java objects from a database. Hibernate can also work independently of JPA using its own native APIs such as Session, Transaction, and HQL (Hibernate Query Language). Although it follows JPA, Hibernate adds many additional features like caching and lazy loading.

**Spring Data JPA:**

**Spring Data JPA** is a Spring framework module that provides a higher-level abstraction over JPA. It does not implement JPA itself but simplifies its usage by reducing the boilerplate code required for common database operations. Spring Data JPA works on top of an implementation like Hibernate and provides repository interfaces like JpaRepository, allowing developers to perform CRUD operations without writing implementation code. It also handles transactions automatically and supports derived query methods, pagination, and sorting out of the box.

**To summarize:**

* JPA is a specification.
* Hibernate is an implementation of the JPA specification.
* Spring Data JPA is an abstraction over JPA that works with implementations like Hibernate to make database access easier and cleaner.

**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:**

public interface EmployeeRepository extends JpaRepository<Employee, Integer> {

}

@Service

public class EmployeeService {

@Autowired

private EmployeeRepository employeeRepository;

@Transactional

public void addEmployee(Employee employee) {

employeeRepository.save(employee);

}

}