Comparison: JPA vs Hibernate vs Spring Data JPA

# 1. Java Persistence API (JPA)

JPA is like a rulebook or a standard that tells Java developers how to save data to databases. It doesn't do anything by itself—it just defines how things should be done. Think of it as the blueprint or interface. You still need someone to actually build it, and that’s where Hibernate comes in.

# 2. Hibernate

Hibernate is the actual worker that follows JPA’s rulebook. It is an ORM (Object Relational Mapping) tool that helps you interact with your database using simple Java objects. It implements all the rules JPA defines and even adds some extra features to make things more powerful. It handles creating database connections, saving data, querying, and managing transactions.

# 3. Spring Data JPA

Spring Data JPA makes things even easier. It sits on top of JPA and Hibernate and reduces the need to write repetitive boilerplate code. You just create an interface, and Spring Data JPA gives you many built-in methods like save, findById, delete etc. It also integrates well with Spring Boot and manages most of the configurations for you.

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