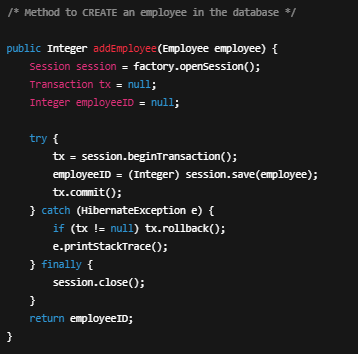
**Hands on 4  
Difference between JPA, Hibernate and Spring Data JPA**

| **Feature** | **JPA (Java Persistence API)** | **Hibernate** | **Spring Data JPA** |
| --- | --- | --- | --- |
| Type | Specification (JSR 338) | ORM framework, JPA implementation | Abstraction over JPA (usually uses Hibernate underneath) |
| Purpose | Define standard API for persisting Java objects | Provide actual implementation of JPA and more features | Simplify data access with repository pattern |
| Boilerplate Code | Still needed (e.g., EntityManager) | Less than plain JDBC, but still needs session handling | Removes most boilerplate — auto methods like findById, save, etc. |
| Requires @Entity | Yes | Yes | Yes |
| Transactions | Supported (with @Transactional) via JTA | Supported via Hibernate-specific configuration | Spring manages it with @Transactional |
| CRUD Methods | Manual implementation needed | Manual session methods like save(), get() | Provided by default through JpaRepository |
| Ease of Use | Requires deep understanding of persistence | Slightly easier than JPA | Most user-friendly; suitable for rapid development |
| Used With | EntityManager, PersistenceUnit, etc. | SessionFactory, Session, etc. | JpaRepository, CrudRepository, Spring Boot |
| Specification vs Implementation | Only defines interfaces (no implementation) | Actual implementation (and more) | Uses Hibernate or EclipseLink internally |

**Code Comparison: Hibernate vs Spring Data JPA**

**Hibernate Example**



**Spring Data JPA Example**

EmployeeRepository.java



EmployeeService.java

