Feature / Aspect	JPA (Java Persistence API)	Hibernate	Spring Data JPA
Type	Specification (JSR 338)	ORM framework and JPA implementation	Abstraction layer over JPA implementations like Hibernate
Developer	Oracle / Jakarta EE	Red Hat	Pivotal / VMware
Implements JPA	No	Yes	No, it uses any JPA implementation underneath
Has concrete implementation	No - interface only	Yes - full ORM tool	No - provides ready-made repository interfaces
Boilerplate code	High - manual EntityManager usage	Moderate - requires SessionFactory, transactions	Minimal - Just extend JpaRepository
Ease of Use	Moderate - needs configuration	Moderate - needs setup	Easy - integrates with Spring Boot
EntityManager / Session	Uses EntityManager	Uses Session (extends EntityManager)	Uses EntityManager behind the scenes
Repository/DAO Layer	Manual - write DAO logic	Manual - use DAO pattern	Auto-implemented through JpaRepository
Transactions	Manual or Spring-managed	Manual or Spring-managed	Automatically managed using @Transactional
Configuration Required	Yes - via persistence.xml or Java config	Yes - hibernate.cfg.xml or Spring config	Minimal - Spring Boot auto-configures
Query Languages Supported	JPQL	HQL, Native SQL	JPQL, Native SQL, Derived Queries
Dynamic Query Support	No built-in support	Yes - HQL	Yes - Method name-based query derivation
Caching Support	Implementation dependent	Yes - First & second-level cache	Inherited from JPA provider
Lazy/Eager Loading	Supported	Supported	Supported
Auditing Support	Manual	Manual	Built-in auditing features
DTO Projections Support	Manual mapping	Manual or custom HQL	Interface-based and class-based projections
Migration Ease	Portable	Less portable	Medium - portable if not using extensions
Typical Use Case	Standard Java EE projects	Projects needing robust ORM	Modern Spring Boot apps
Code Complexity for CRUD	High - manual transaction	Medium - session, transaction	Low - auto-generated CRUD
Documentation & Community	Standard JSR docs	Extensive documentation	Excellent Spring docs