Digital-Nurture-4.0-JavaFSE – Hands On Documentation

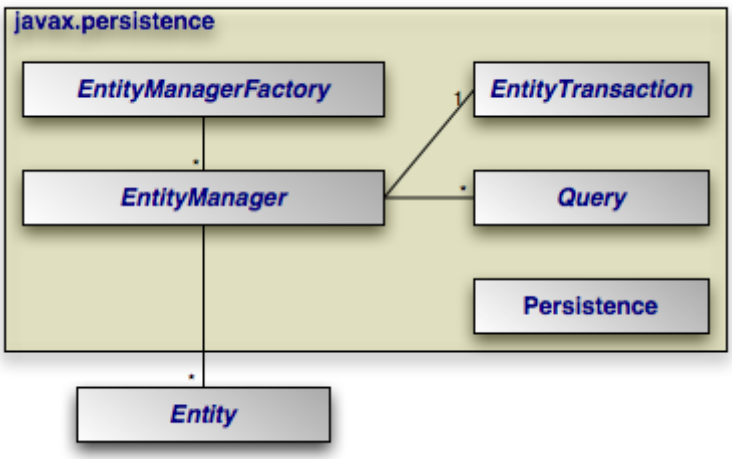
Difference Between Java Persistence API, Hibernate and Spring Data JPA

1. Java Persistence API (JPA)

JPA is a **standard specification** that defines how Java objects should be persisted to relational databases using annotations and interfaces. It provides the core concepts for object-relational mapping, such as EntityManager for transactions and lifecycle management, but it doesn't contain any working code. To actually execute persistence operations, a concrete implementation like Hibernate is required.

Uses of JPA:

- Annotating Java classes to define entities and relationships (@Entity, @OneToMany).
- Managing database interactions using the EntityManager API.
- Ensuring code portability across different JPA providers (Hibernate, EclipseLink, etc.).

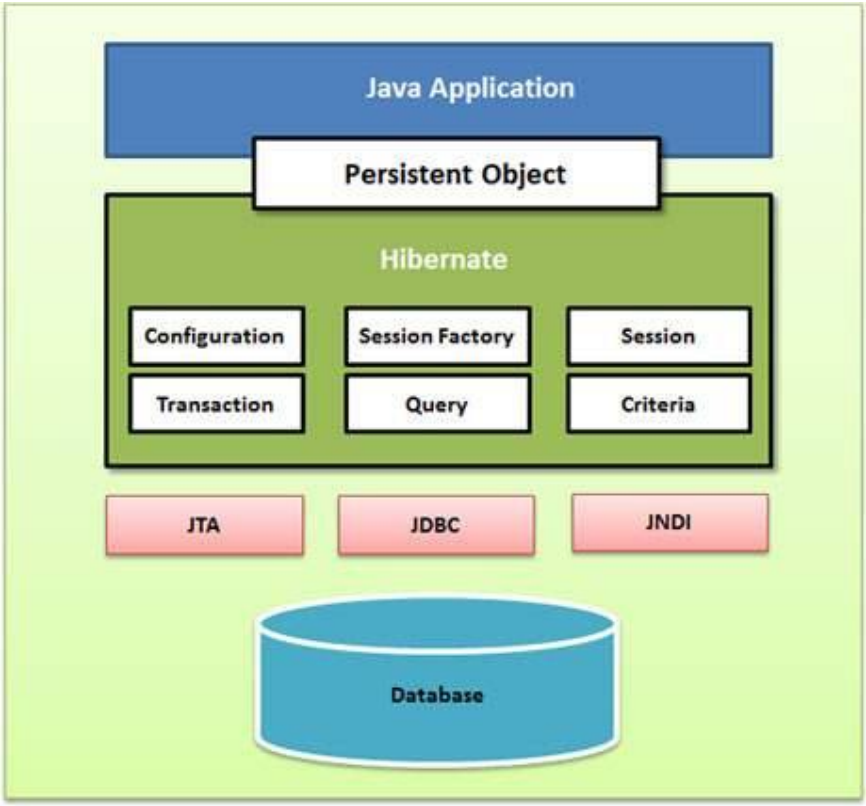


2. Hibernate

Hibernate is a **concrete ORM framework** that implements JPA and offers additional features beyond the specification. It simplifies database access by mapping Java classes to tables and automating SQL generation. Hibernate supports both JPA-based programming and its own native APIs such as Session and HQL, allowing for greater control and extended features when needed.

Uses of Hibernate:

- Executing object-oriented queries using Hibernate Query Language (HQL).
- Handling session-level operations with fine control through Session and Criteria APIs.
- Improving performance with features like lazy loading and second-level caching.

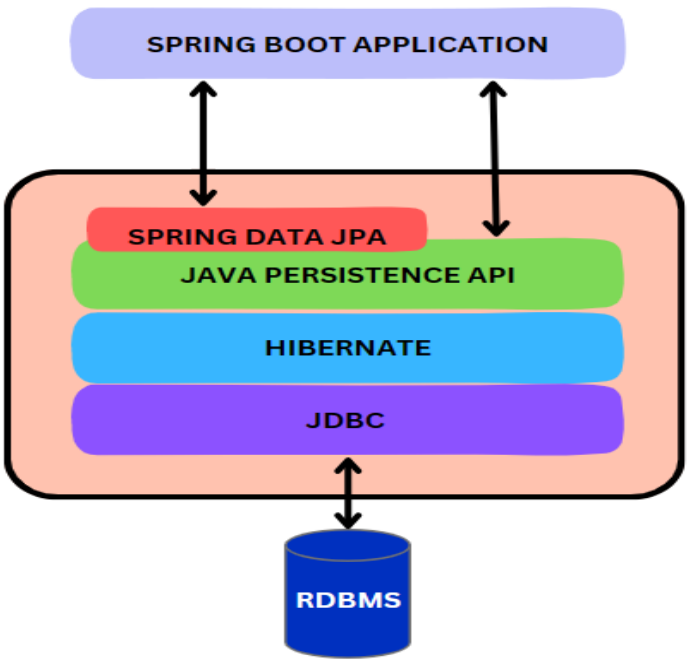


3. Spring Data JPA

Spring Data JPA is a **higher-level abstraction** that builds on top of JPA (often with Hibernate underneath) to simplify repository development in Spring applications. It removes much of the boilerplate code by generating repository implementations at runtime based on simple interface definitions. It also supports advanced features like derived queries, paging, sorting, and integration with Spring Boot.

Uses of Spring Data JPA:

- Creating repositories by extending interfaces like JpaRepository with automatic method implementation.
- Using method naming conventions to generate queries without writing JPQL.
- Seamlessly integrating data access into Spring's dependency injection and transaction management system.



How JPA Differs from Hibernate and Spring Data JPA

JPA is only a specification; it cannot perform any real operations by itself. It simply outlines how persistence should work in Java. Hibernate, on the other hand, is a full-fledged ORM framework that implements JPA and adds several powerful features like caching, HQL, and custom fetching strategies. While JPA sets the standard, Hibernate executes it and goes beyond it with additional tools and optimizations.

How Hibernate Differs from JPA and Spring Data JPA

Hibernate requires manual setup and detailed code for data access, using either JPA or its native APIs. Spring Data JPA makes this easier by allowing developers to define simple interfaces, and it handles the implementation behind the scenes. It sits on top of Hibernate (via JPA) but introduces a more declarative and productive development style, especially in Spring-based applications.

How Spring Data JPA Differs from JPA and Hibernate

JPA offers flexibility and standardization but involves more boilerplate code. Spring Data JPA improves developer experience by automating common patterns, reducing effort, and integrating closely with Spring. While both use Hibernate under the hood, Spring Data JPA is designed for speed, simplicity, and rapid application development.

REFERENCES

https://dzone.com/articles/what-is-the-difference-between-hibernate-and-sprin-1

https://www.javaworld.com/article/3379043/what-is-jpa-introduction-to-the-java-persistence-api.html

https://www.ibm.com/docs/en/was-liberty/nd?topic=liberty-java-persistence-api-jpa

https://www.geeksforgeeks.org/java/hibernate-tutorial/

https://spring.io/projects/spring-data-jpa