The relationship between Spring Data JPA, JPA, and Hibernate/EclipseLink

JPA

JPA is an abbreviation that stands for Java Persistence API. It’s a specification which is part of Java EE. The Java Persistence API provides a specification for persisting, reading, and managing data from your Java object to relational tables in the database.

JPA itself doesn’t provide any implementation classes. The API jar just contains a set of interfaces which you can use to implement your persistence layer. But you can’t use JPA on its own. You need a JPA provider which implements the specification. **Hibernate** and EclipseLink are 2 popular implementations of this specification

#### Mapping Associations

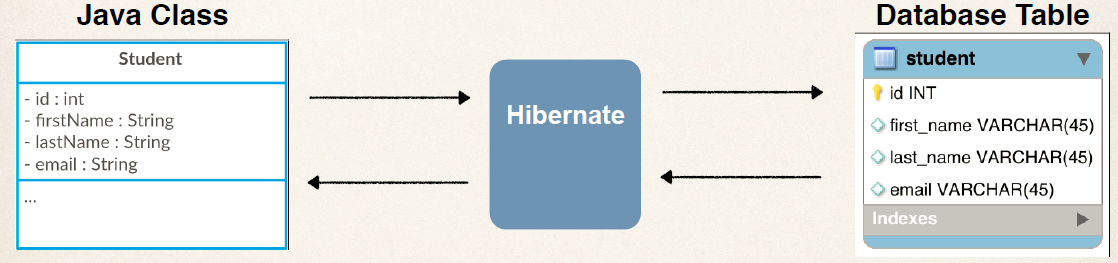
JPA doesn’t only enable you to map simple entity attributes to database columns, but it also allows you to [map associations](https://thoughts-on-java.org/ultimate-guide-association-mappings-jpa-hibernate/) between database tables to entity attributes.

|  |  |
| --- | --- |
|  | @Entity  public class Review {        ...        @ManyToOne      private Book book;        ...  } |

## EclipseLink and Hibernate

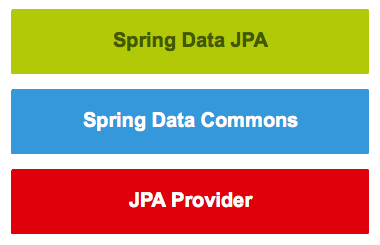
As I said before, you need a JPA provider, if you want to use the JPA specification in your project. It implements the interfaces as defined by the specification. The most popular ones are EclipseLink and Hibernate.

Hibernate is a JPA implementation, and an object-relational mapping solution for Java environments. Object-relational mapping or ORM is the programming technique to map application domain model objects to the relational database tables.



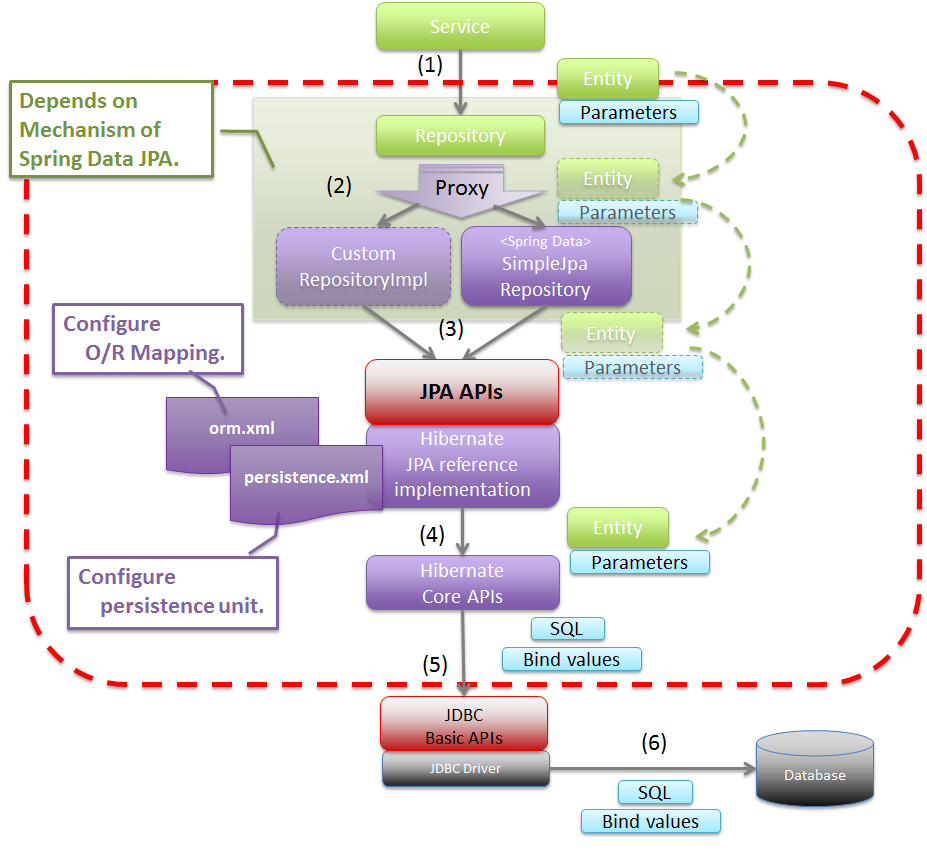
Spring Data JPA

Spring Data JPA is not a JPA provider. It is a library/framework that adds an extra layer of abstraction on the top of our JPA provider (like Hibernate). Spring Data JPA is just an abstraction used to significantly reduce the amount of boilerplate code required to implement data access layers for various persistence stores.



**Spring Data JPA** adds a layer on top of JPA. That means it uses all features defined by the JPA specification, especially the entity and [association mappings](https://thoughts-on-java.org/ultimate-guide-association-mappings-jpa-hibernate/), the entity lifecycle management, and [JPA’s query capabilities.](https://thoughts-on-java.org/jpql/) On top of that, Spring Data JPA adds its own features like a no-code implementation of the [repository pattern](https://thoughts-on-java.org/implementing-the-repository-pattern-with-jpa-and-hibernate/) and the creation of database queries from method names.

## Basic Spring Data JPA Flow



## How does JPA/Hibernate Work?

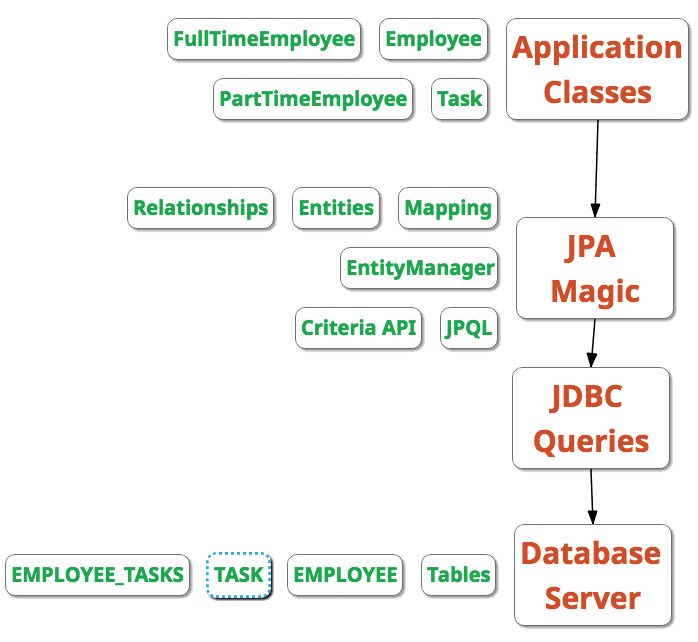
Databases are designed with Tables/Relations. Java objects are designed using OOPS. We would want to store the data from objects into tables and vice-versa.

Earlier approaches involved writing SQL Queries. JDBC, Spring JDBC and myBatis were popular approaches.

However, JPA evolved as a result of a different thought process.

*How about mapping the objects directly to tables/relationships?*

This Mapping is also called ORM - Object Relational Mapping. Before JPA, ORM was the term more commonly used to refer to these frameworks. Thats one of the reasons, Hibernate is called a ORM framework.



JPA allows to map application classes to tables in database.

* Entity Manager - Once the mappings are defined, entity manager can manage your entities. Entity Manager handles all interactions with the database
* JPQL (Java Persistence Query Language) - Provides ways to write queries to execute searches against entities. Important thing to understand is the these are different from SQL queries. JPQL queries already understand the mappings that are defined between entities. We can add additional conditions as needed.
* Criteria API defines a Java based API to execute searches against databases.

## JPA vs Hibernate

Hibernate is one of the most popular ORM frameworks.

JPA defines the specification. It is an API.

* How do you define entities?
* How do you map attributes?
* How do you map relationships between entities?
* Who manages the entities?

Hibernate is one of the popular implementations of JPA.

* Hibernate understands the mappings that we add between objects and tables. It ensures that data is stored/retrieved from the database based on the mappings.
* Hibernate also provides additional features on top of JPA. But depending on them would mean a lock in to Hibernate. You cannot move to other JPA implementations like Toplink.