JDBC

JDBC is an API provided by java. Using JDBC we can connect our java application to database. We know java work on concept of object and classes and database works on concept of tables and columns. So, in JDBC we have take the values from object and map it to columns of the table manually or vice-versa.

ORM (Object Relational Mapping)

Object <=> Table
Variable <=> Column

* Hibernate, iBaties etc supports ORM. And bothhave their different mechanism.

If in future we want to change from hibernate to iBeties then it will be a big problem.

Due to this Sun people decided to introduce a standard specifications. And those standard specifications should be followed by all the 3rd party API providers or ORM providers. So, whenever we have to switch b/w ORM providers, it will be easier to manage.

JPA (Java Persistence API)

Java only provides standard specifications. So, for using those we must need implementation provider like Hibernate or ibeties.

JPA is a specification that standardizes the way Java Objects are mapped to a relational database system. Being just a specification, JPA consists of a set of interfaces, like `EntityManagerFactory`, `EntityManager`, and annotations that help you map a Java entity object to a database table.

There are several JPA providers, like HIbernate, EclipseLink, or Open JPA which you can use.

Spring Data JPA

Spring Data JPA offers a solution to the `Repository` pattern. It can also generate JPA queries on your behalf through method name conventions. This module deals with enhanced support for JPA based data access layers.

Spring Data JPA aims to significantly improve the implementation of data access layers by reducing the effort to the amount that's actually needed. As a developer you write your repository interfaces, including custom finder methods, and Spring will provide the implementation automatically.

Spring Data JPA can work with Hibernate, Eclipse Link, or any other JPA provider. A very interesting benefit of using Spring or Java EE is that you can control transaction boundaries declaratively using the `@Transactional` annotation.

Spring Data

This is an umbrella project which contains many subprojects that are specific to a given database.

- Spring Data JPA
- Spring Data JDBC
- Spring Data LDAP
- Spring Data MongoDB
- Spring Data Redis
- Spring Data R2DBC
- Spring Data REST
- Spring Data for Apache Cassandra
- Spring Data for Apache Geode
- Spring Data for Apache Solr
- Spring Data for Pivotal GemFire
- Spring Data Couchbase
- Spring Data Elasticsearch
- Spring Data Envers
- Spring Data Neo4j
- Spring Data JDBC Extensions
- Spring for Apache Hadoop