

JDBC, JPA/Hibernate, Mybatis의 차이를 이해하다.

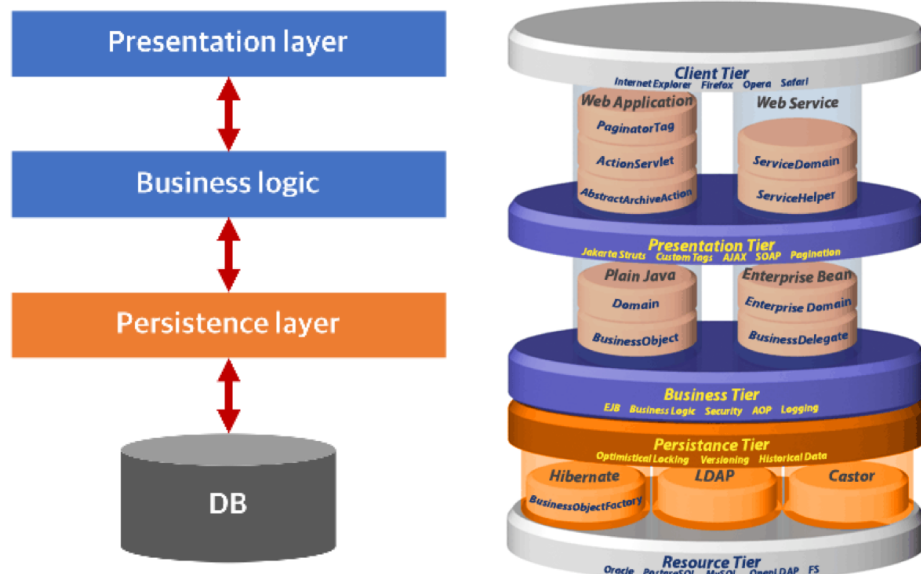
## Goal

- 영속성(Persistence)의 개념을 이해한다.
- SQL Mapper와 ORM의 차이에 대해 이해한다.
- JDBC(Data Transfer Object)란 무엇인지 이해한다.
- JPA/Hibernate란 무엇인지 이해한다.
- Mybatis란 무엇인지 이해한다.

## 영속성(Persistence)

- 데이터를 생성한 프로그램이 종료되더라도 사라지지 않는 데이터의 특성을 말한다.
- 영속성을 갖지 않는 데이터는 단지 메모리에서만 존재하기 때문에 프로그램을 종료하면 모두 잃어버리게 된다. 때문에 파일 시스템, 관계형 데이터베이스 혹은 객체 데이터베이스 등을 활용하여 데이터를 영구하게 저장하여 영속성 부여한다.
- **Persistence Layer**
  - 프로그램의 아키텍처에서, 데이터에 영속성을 부여해주는 계층을 말한다.
  - JDBC를 이용하여 직접 구현할 수 있지만 Persistence framework를 이용한 개발이 많이 이루어진다.

◦



### ◦ 계층 참고

- 프레젠테이션 계층 (Presentation layer) - UI 계층 (UI layer) 이라고도 함
- 애플리케이션 계층 (Application layer) - 서비스 계층 (Service layer) 이라고도 함

- 비즈니스 논리 계층 (Business logic layer) - 도메인 계층 (Domain layer) 이라고도 함
- 데이터 접근 계층 (Data access layer) - 영속 계층 (Persistence layer) 이라고도 함
- **Persistence Framework**
  - JDBC 프로그래밍의 복잡함이나 번거로움 없이 간단한 작업만으로 데이터베이스와 연동되는 시스템을 빠르게 개발할 수 있으며 안정적인 구동을 보장한다.
  - Persistence Framework는 SQL Mapper와 ORM으로 나눌 수 있다.
    - 아래 참고
    - Ex) JPA, Hibernate, Mybatis 등

## SQL Mapper와 ORM

Persistence Framework는 SQL Mapper와 ORM으로 나눌 수 있다.

- ORM은 데이터베이스 객체를 자바 객체로 매핑함으로써 객체 간의 관계를 바탕으로 SQL을 자동으로 생성해주지만 SQL Mapper는 SQL을 명시해줘야 한다.
- ORM은 관계형 데이터베이스의 '관계'를 Object에 반영하자는 것이 목적이라면, SQL Mapper는 단순히 필드를 매핑시키는 것이 목적이라는 점에서 지향점의 차이가 있다.

### # SQL Mapper

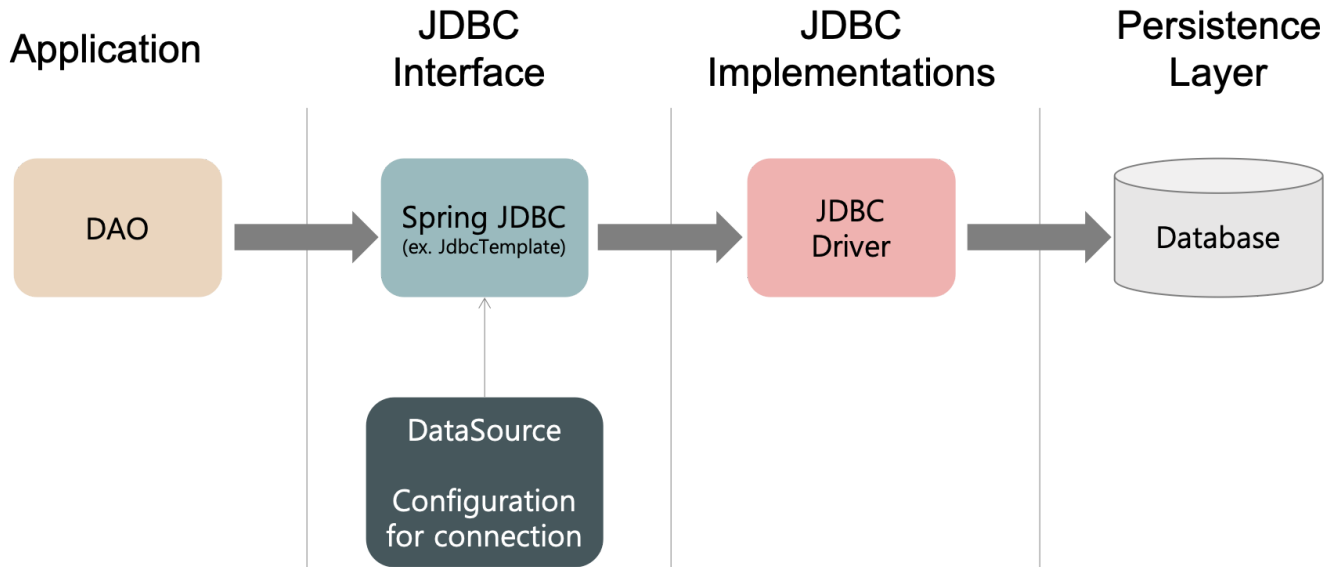
- SQL <←매핑→> Object 필드
- SQL Mapper는 SQL 문장으로 직접 데이터베이스 데이터를 다룬다.
  - 즉, SQL Mapper는 SQL을 명시해줘야 한다.
  - Ex) **Mybatis, JdbcTemplate** 등

### # ORM(Object-Relational Mapping), 객체-관계 매핑

- 데이터베이스 데이터 <←매핑→> Object 필드
  - 객체를 통해 간접적으로 데이터베이스 데이터를 다룬다.
- 객체와 관계형 데이터베이스의 데이터를 자동으로 매핑(연결)해주는 것을 말한다.
  - ORM을 이용하면 SQL Query가 아닌 직관적인 코드(메서드)로 데이터를 조작할 수 있다.
  - 객체 간의 관계를 바탕으로 SQL을 자동으로 생성한다.
- Persistent API라고도 할 수 있다.
  - Ex) **JPA, Hibernate** 등
- ORM의 장단점 참고

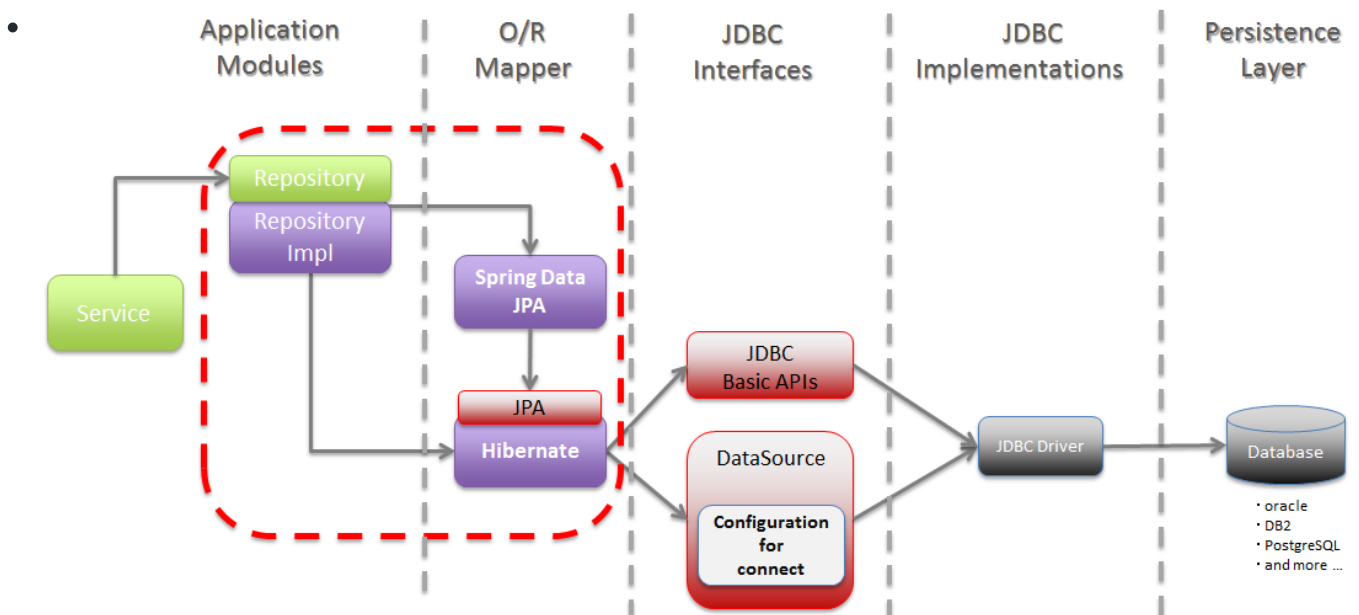
## 1. JDBC(Java Database Connectivity)

•



- JDBC는 DB에 접근할 수 있도록 Java에서 제공하는 API이다.
  - 모든 Java의 Data Access 기술의 근간
  - 즉, 모든 Persistence Framework는 내부적으로 JDBC API를 이용한다.
- JDBC는 데이터베이스에서 자료를 쿼리하거나 업데이트하는 방법을 제공한다.
- Plain JDBC vs Spring JDBC 참고

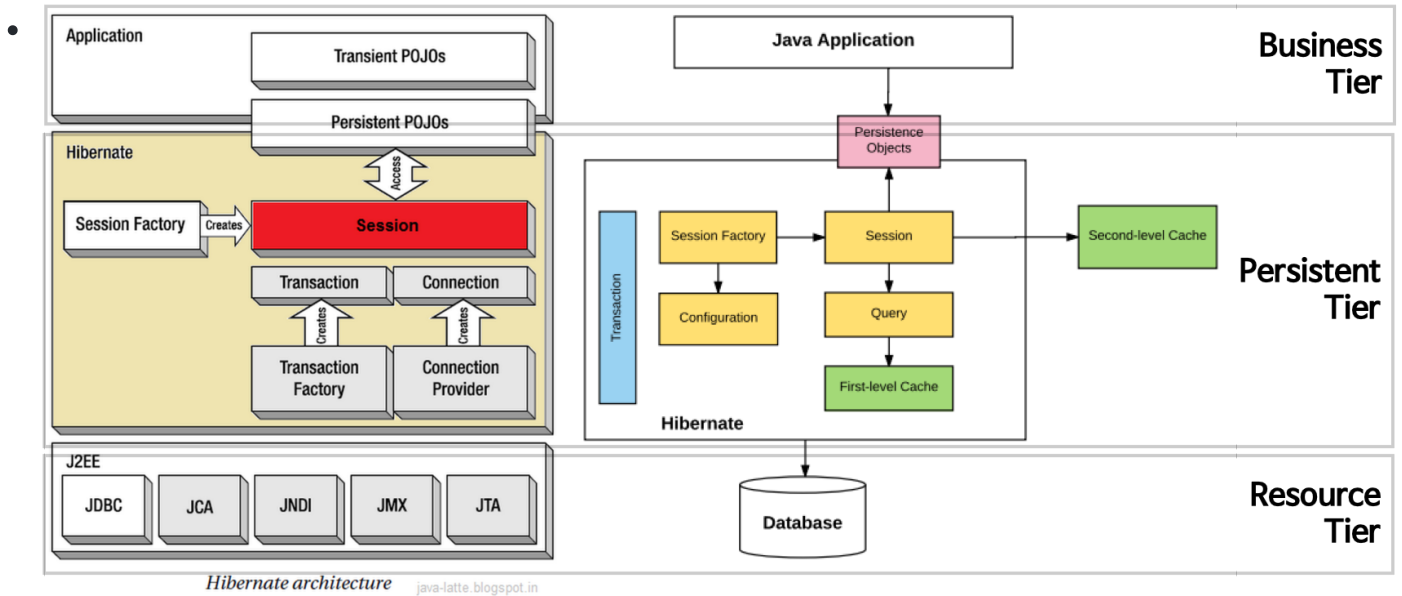
## 2. JPA(Java Persistent API).



- 자바 ORM 기술에 대한 API 표준 명세로, Java에서 제공하는 API이다.
  - 자바 플랫폼 SE와 자바 플랫폼 EE를 사용하는 응용프로그램에서 관계형 데이터베이스의 관리를 표현하는 자바 API이다.
  - 즉, JPA는 ORM을 사용하기 위한 표준 인터페이스를 모아둔 것이다.
  - 기존에 EJB에서 제공되던 엔터티 빈(Entity Bean)을 대체하는 기술이다.
- JPA 구성 요소 (세 가지)
  - 1) `javax.persistence` 패키지로 정의된 API 그 자체
  - 2) JPQL(Java Persistence Query Language).

- 3) 객체/관계 메타데이터
- 사용자가 원하는 JPA 구현체를 선택해서 사용할 수 있다.
  - JPA의 대표적인 구현체로는 Hibernate, EclipseLink, DataNucleus, OpenJPA, TopLink Essentials 등이 있다.
  - 이 구현체들을 ORM Framework라고 부른다.

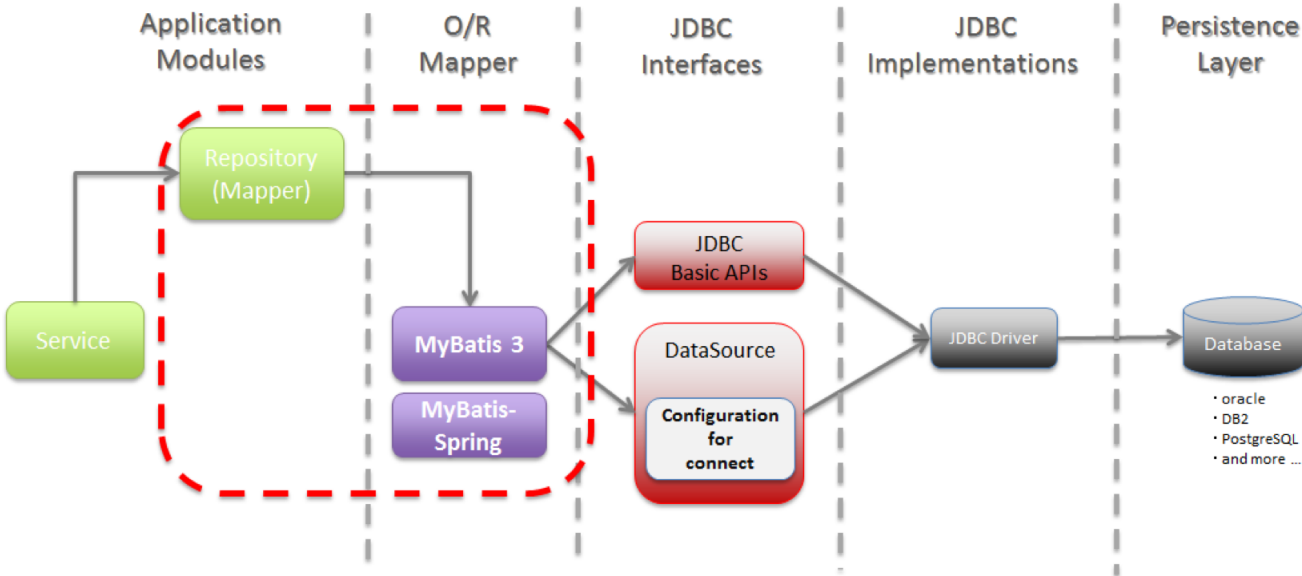
## Hibernate



- Hibernate는 JPA의 구현체 중 하나이다.
- Hibernate가 SQL을 직접 사용하지 않는다고 해서 JDBC API를 사용하지 않는다는 것은 아니다.
  - Hibernate가 지원하는 메서드 내부에서는 JDBC API가 동작하고 있으며, 단지 개발자가 직접 SQL을 직접 작성하지 않을 뿐이다.
- HQL(Hibernate Query Language)이라 불리는 매우 강력한 쿼리 언어를 포함하고 있다.
  - HQL은 SQL과 매우 비슷하며 추가적인 컨벤션을 정의할 수도 있다.
  - HQL은 완전히 객체 지향적이며 이로써 상속, 다형성, 관계등의 객체지향의 강점을 누릴 수 있다.
  - HQL쿼리는 자바 클래스와 프로퍼티의 이름을 제외하고는 대소문자를 구분한다.
  - HQL은 쿼리 결과로 객체를 반환하며 프로그래머에 의해 생성되고 직접적으로 접근할 수 있다.
  - HQL은 SQL에서는 지원하지 않는 페이지네이션이나 동적 프로파일링과 같은 향상된 기능을 제공한다.
  - HQL은 여러 테이블을 작업할 때 명시적인 join을 요구하지 않는다.
- 장점
  - 객체지향적으로 데이터를 관리할 수 있기 때문에 비즈니스 로직에 집중 할 수 있으며, 객체 지향 개발이 가능하다.
  - 테이블 생성, 변경, 관리가 쉽다. (JPA를 잘 이해하고 있는 경우)
  - 로직을 쿼리에 집중하기 보다는 객체자체에 집중 할 수 있다.
  - 빠른 개발이 가능하다.

- 단점
  - 어렵다. (많은 내용이 감싸져 있기 때문에 알아야 할 것이 많다.)
  - 잘 이해하고 사용하지 않으면 데이터 손실이 있을 수 있다. (persistence context)
  - 성능상 문제가 있을 수 있다. (이 문제 또한 잘 이해해야 해결이 가능하다.)

### 3. Mybatis

- 

The diagram illustrates the Mybatis architecture across five layers: Application Modules, O/R Mapper, JDBC Interfaces, JDBC Implementations, and Persistence Layer. In the Application Modules layer, a Service interacts with a Repository (Mapper). The Repository (Mapper) is part of the O/R Mapper layer, which also includes MyBatis 3 and MyBatis-Spring. These components interact with the JDBC Interfaces layer, which contains JDBC Basic APIs, DataSource, and Configuration for connect. The JDBC Implementations layer includes the JDBC Driver. Finally, the Persistence Layer contains the Database, which supports various types like Oracle, DB2, PostgreSQL, and more. A red dashed box encloses the Service, Repository (Mapper), MyBatis 3, and MyBatis-Spring components.
- 개발자가 지정한 SQL, 저장 프로시저 그리고 몇 가지 고급 매핑을 지원하는 SQL Mapper이다.
- JDBC로 처리하는 상당 부분의 코드와 파라미터 설정 및 결과 매핑을 대신해준다.
  - 기존에 JDBC를 사용할 때는 DB와 관련된 여러 복잡한 설정(Connection)들을 다루어야 했지만 SQL Mapper는 자바 객체를 실제 SQL문에 연결함으로써, 빠른 개발과 편리한 테스트 환경을 제공한다.
- 데이터베이스 record에 원시 타입과 Map 인터페이스 그리고 자바 POJO를 설정해서 매핑하기 위해 xml과 Annotation을 사용할 수 있다.
- Mybatis는 원래 Apache Foundation의 iBatis였으나, 생산성, 개발 프로세스, 커뮤니티 등의 이유로 Google Code로 이전되면서 이름이 바뀌었다.
  - iBatis와 바뀐 차이점은 아래와 같다.
    - JDK 1.5, Annotation
    - Dynamic SQL, XML Element
- 장점
  - SQL에 대한 모든 컨트롤을 하고자 할때 매우 적합하다.
  - SQL쿼리들이 매우 잘 최적화되어 있을 때에 유용하다.
- 단점
  - 애플리케이션과 데이터베이스 간의 설계에 대한 모든 조작을 하고자 할 때는 적합하지 않다.
  - 애플리케이션과 데이터베이스 간에 서로 잘 구조화되도록 많은 설정이 바뀌어야 하기 때문이다.

# References

- <https://stackoverflow.com/questions/16439249/when-to-use-servlet-or-controller>
- <https://okky.kr/article/286812>
- <https://www.slideshare.net/benjaminbkim9/what-is-persistenceinjava>
- <http://blog.woniper.net/255>
- <https://humbroll.wordpress.com/>
- <https://mingrammer.com/translation-10-common-software-architectural-patterns-in-a-nutshell/>
- <http://smartweb.sourceforge.net/persistence.html>
- <http://tinkerbellbass.tistory.com/24>
- <https://terasolunaorg.github.io/guideline/5.1.0.RELEASE/en/ArchitectureInDetail/DataAccessJpa.html>
- [유사 주제 참고](#)