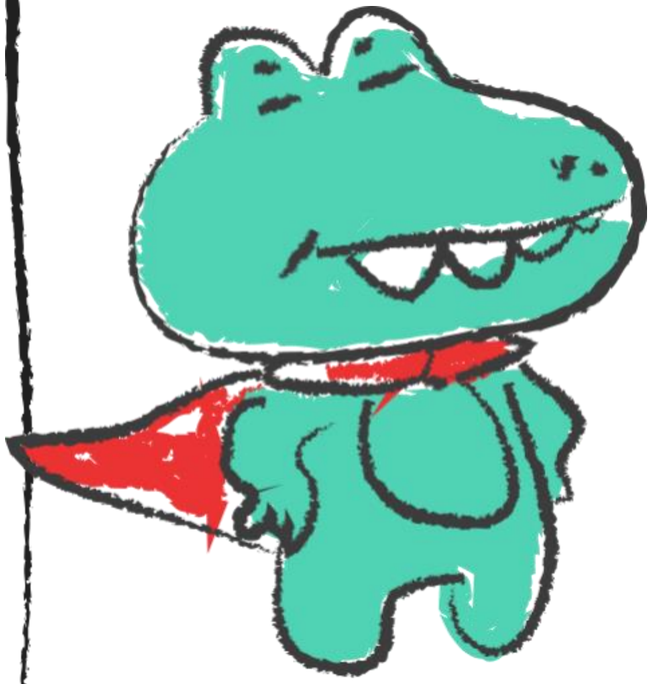
ORM과 SQL MAPPER



SI2팀 엄예지

목차라구용

첫번째 ORM

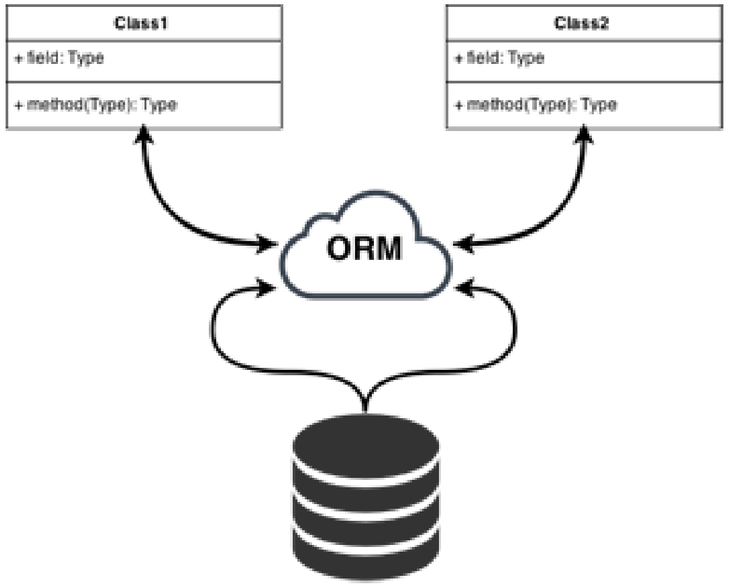
두번째 SQL MAPPER

세번째 COMPARE

네번째 TALKING ABOUT

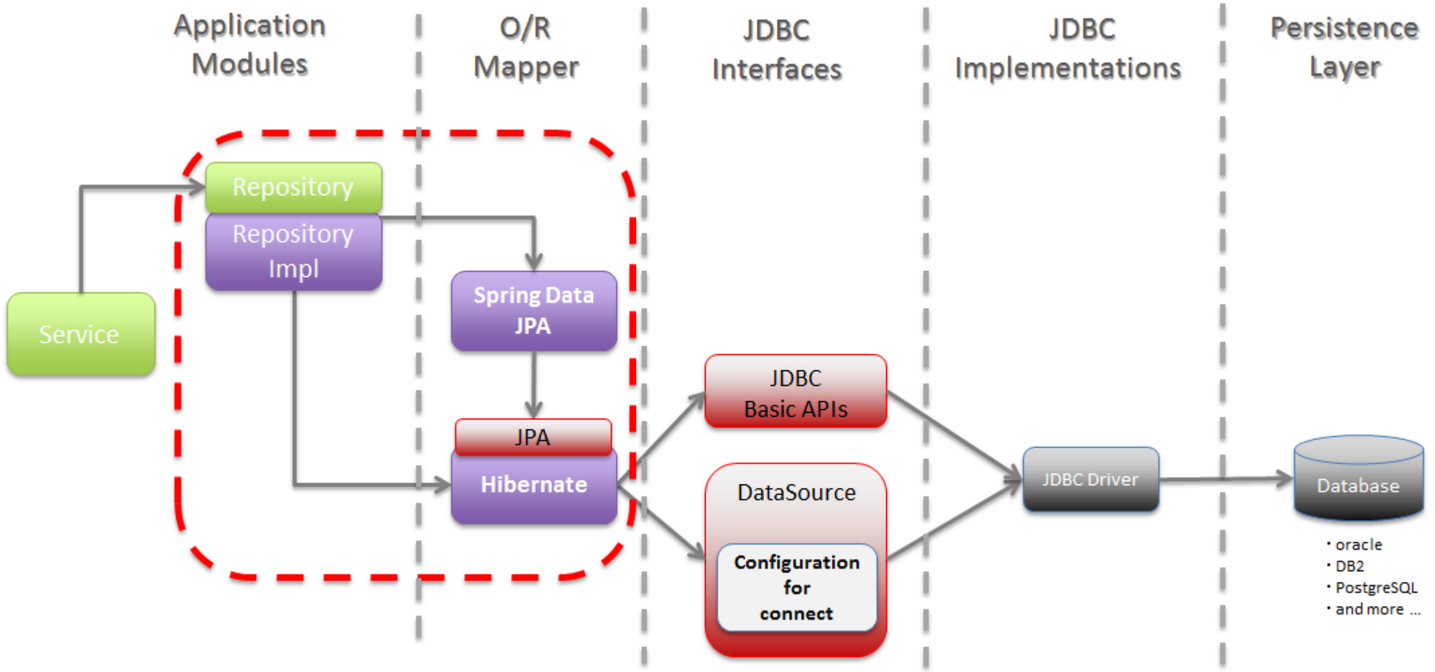
ORM

ORM (Object-Relatinal Mapping)



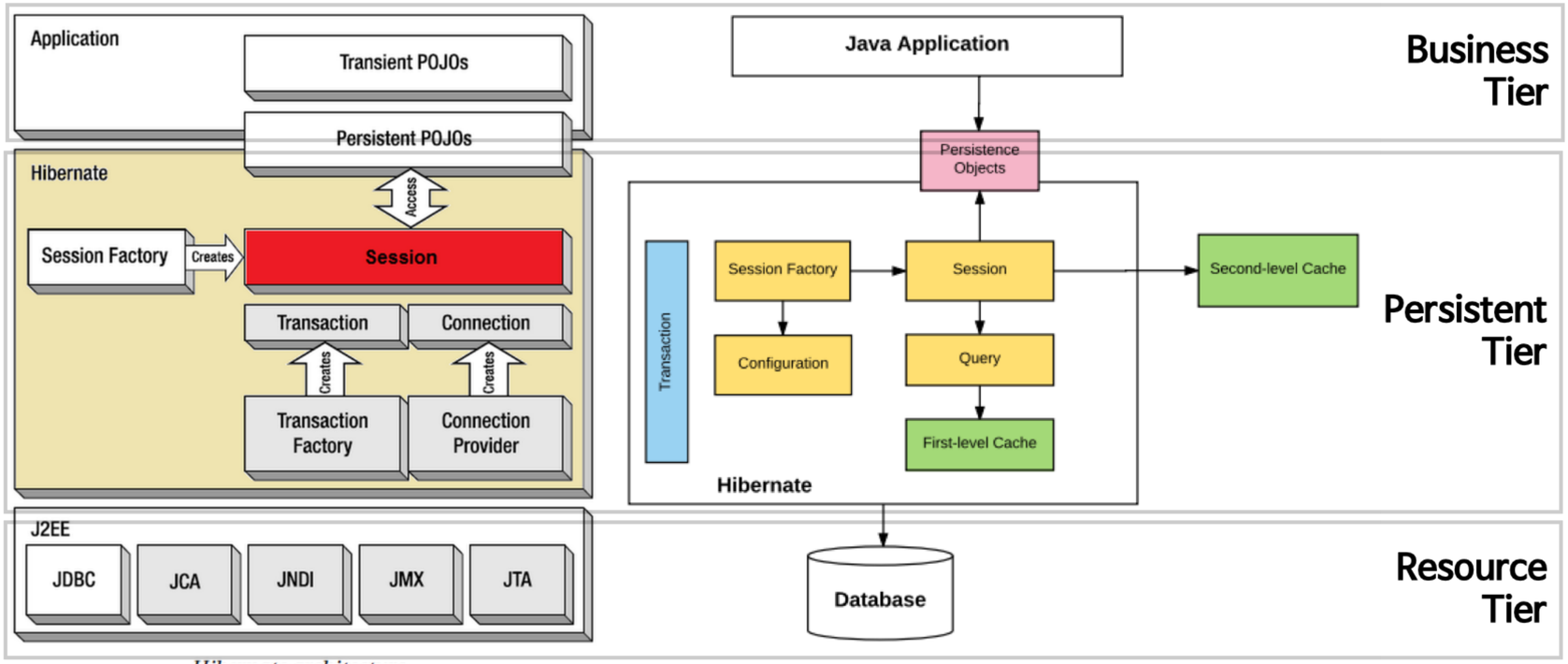
- 객체(클래스)와 관계(관계형 데이터 베이스)와의 설정
- 객체간의 관계를 바탕으로 SQL을 자동 생성하여 테이블과 객체간의 불일치를 해결하는 것
- Object <= 매핑 => DB데이터 에서의 매핑 역할
- JPA, Hibernate

JPA (Java Persistence API)



- JAVA ORM 기술에 대한 표준 명세, JAVA에서 제공하는 API
- 자바 어플리케이션에서 관계형 데이터베이스를 사용하는 방식 정의한 인터페이스
- ORM이기 때문에 자바 클래스와 DB테이블을 매핑한다. (SQL매핑 X)

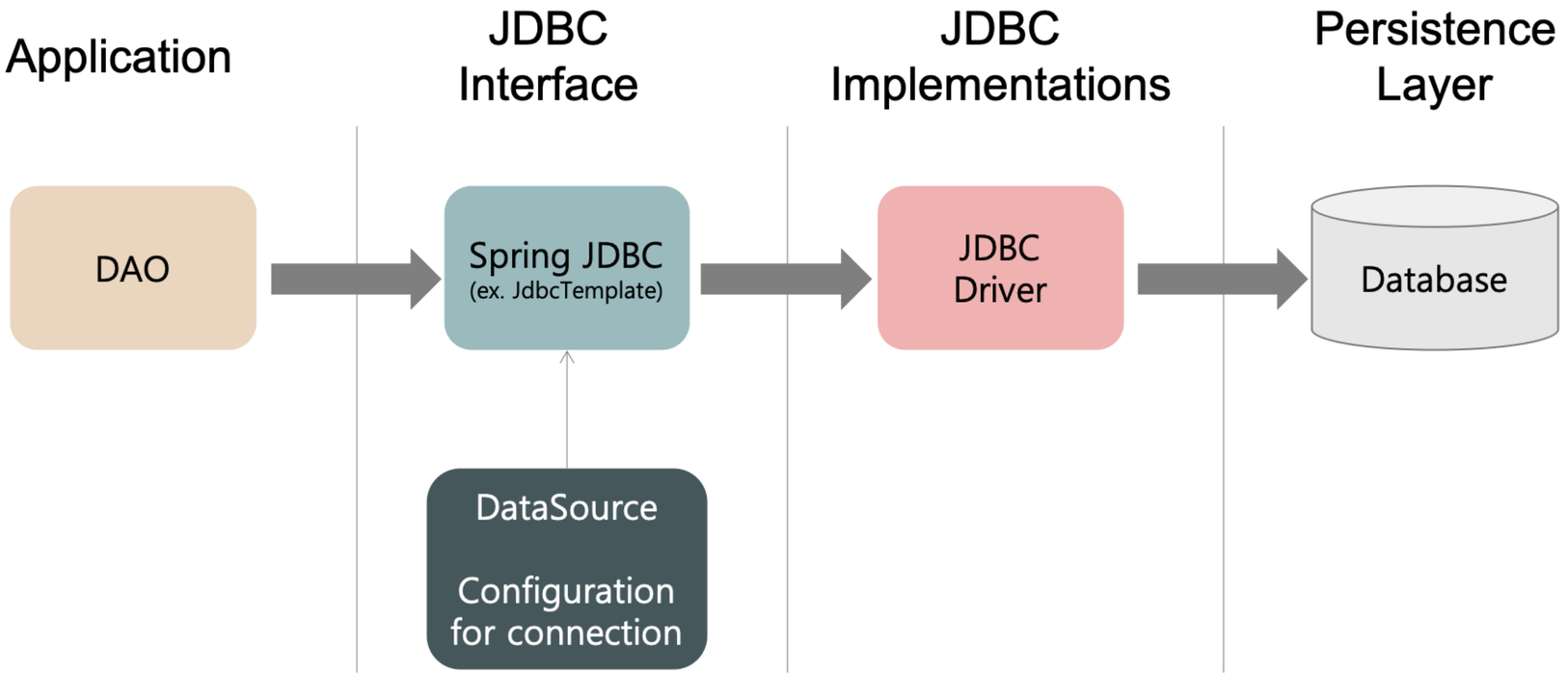
Hibernate



Hibernate architecture java-latte.blogspot.in

- JPA 구현체의 한 종류
- JDBC API 사용하지 않는것은 X
- HQL(Hibernate Query Language)이라 불리는 강력한 쿼리 언어 포함

JDBC(Java Database Connectibity)



- DB에 접근할수 있도록 자바에서 제공하는 API
- 모든 JAVA의 Data Access 기술의 근간

ORM (Object-Relatinal Mapping)

장점

- 완벽한 객체지향적인 코드
- 재사용, 유지보수, 리팩토링 용의성
- DBMS 종속성 하락

단점

- ORM이 모든걸 해결해줄 수 없다.
- 객체-관계 간의 불일치
- 프로시저가 많은 시스템에선 ORM의 객체 지향적인 장점을 활용하기 어렵다.

ORM (Object-Relatinal Mapping)

객체-관계 간의 불일치

다음과 같은 특성에서 객체-관계 간의 불일치가 생긴다.

세분성(Granularity)

경우에 따라서 데이터베이스에 있는 테이블 수보다 더 많은 클래스를 가진 모델이 생길 수 있다.

상속성(Inheritance)

RDBMS는 객체지향 프로그래밍 언어의 특징인 상속 개념이 없다.

일치(Identity)

RDBMS는 기본키(primary key)를 이용하여 동일성을 정의한다. 그러나 자바는 객체 식별(`a==b`)과 객체 동일성(`a.equals(b)`)을 모두 정의한다.

연관성(Associations)

객체지향 언어는 방향성이 있는 객체의 참조(reference)를 사용하여 연관성을 나타내지만 RDBMS는 방향성이 없는 외래키(foreign key)를 이용해서 나타낸다.

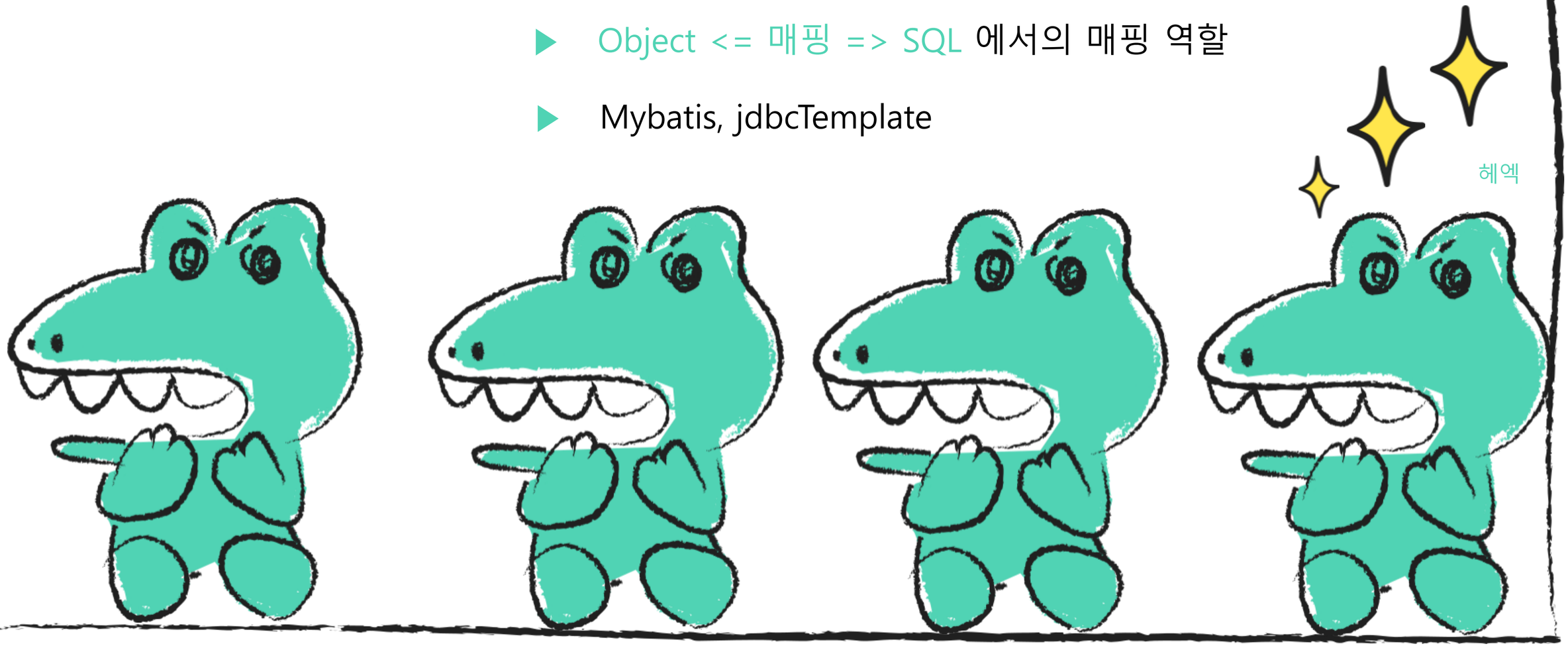
탐색(Navigation)

자바와 RDBMS에서 객체를 접근하는 방법이 근본적으로 다르다. 자바는 그래프형태로 하나의 연결에서 다른 연결로 이동하며 탐색한다. 그러나 RDBMS에서는 일반적으로 SQL문을 최소화하고 `JOIN`을 통해 여러 엔티티를 로드하고 원하는 대상 엔티티를 선택하는 방식으로 탐색한다.

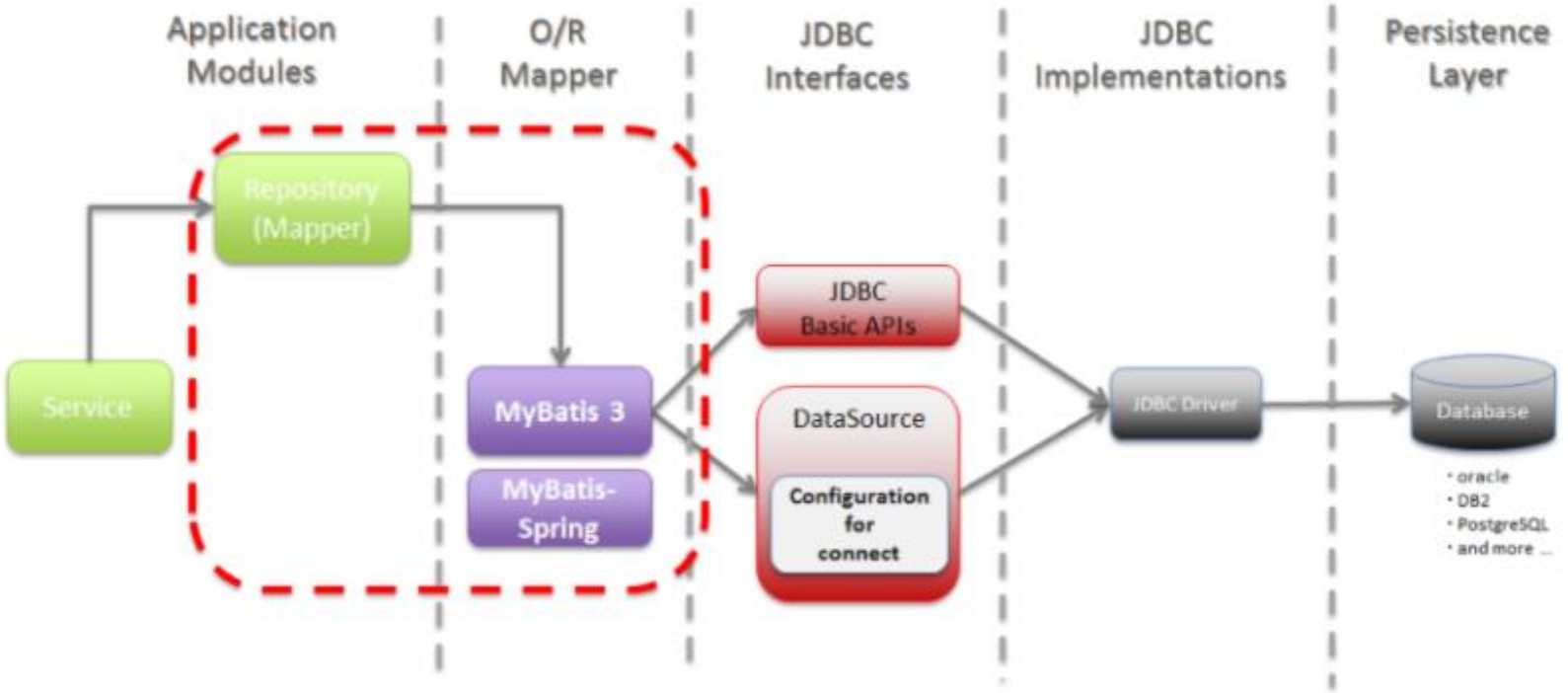
SQL MAPPER

SQL MAPPER

- SQL문으로 직접 디비를 조작
- Object <= 매핑 => SQL 에서의 매핑 역할
- Mybatis, jdbcTemplate



MYBATIS



- 개발자가 지정한 SQL, 저장 프로시저, 몇 가지 고급 매핑을 지원하는 SQL Mapper
- SQL작성을 직접 하여 객체와 매핑

Parameter Type 속성과 SQL문의 입력 파라미터 처리

```
StringBuffer    sql_str  = new StringBuffer();

sql_str.append(" INSERT INTO KUKDTALIB."+JOBGBN+"                          \n");
sql_str.append("        ( FILENM, JPCODE, LOTNUM, MNGDAT, REMARK,          \n");
sql_str.append("          REGUSR, REGDAT)                                  \n");
sql_str.append("          VALUES                                           \n");
sql_str.append("        ( ?, ?, ?, ?, ?,                                   \n");
sql_str.append("          ?, ?)                                            \n");

cstmt = new LoggableStatement(conn, sql_str.toString());
cstmt.setString(1, FILENM);
cstmt.setString(2, JPCODE);
cstmt.setString(3, LOTNUM);
cstmt.setString(4, MAKDAT);
cstmt.setString(5, REMARK);
cstmt.setString(6, USERID.toUpperCase());
cstmt.setString(7, nowdat);

cstmt.executeUpdate();
System.out.println(((LoggableStatement) cstmt).getQueryString());
```

JDBC – PreparedStateMent

Parameter Type 속성과 SQL문의 입력 파라미터 처리

```
<!-- 등록 -->
<insert id="insAllScanList">
    INSERT INTO ALSCLS ( JPCODE, FILENM, LOTNUM, MNGDAT, REMARK
                       , REGUSR, REGDAT)
                VALUES ( #{JPCODE}, #{FILENM}, #{LOTNUM}, #{MNGDAT}, #{REMARK, jdbcType=VARCHAR}
                       , #{REGUSR}, TO_CHAR(SYSDATE, 'YYYY-MM-DD HH:mm:SS') )
</insert>
```

MyBatis –
#{프로퍼티}

SQL MAPPER

장점

- JPA(ORM)에 비해 쉽다.
- SQL의 세부적인 내용 변경 시 좀 더 간편하다.
- 동적 쿼리 사용시 JPA보다 간편하게 구현 가능하다.

단점

- 데이터베이스 설정 변경시 수정할 부분이 너무 많다..
- Mapper작성부터 인터페이스 설계까지 JPA보다 많은 설계와 파일, 로직이 필요하다.
- 특정 DB에 종속적이다.

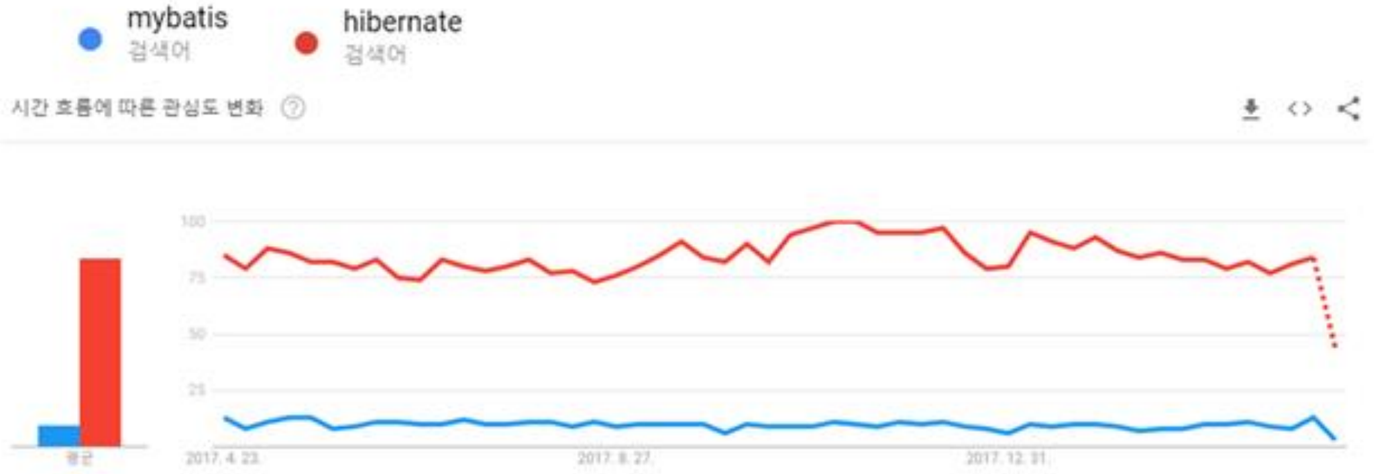
COMPARE

ORM

SQL Mapper

- DB 데이터 ← mapping → Object 필드
- 객체를 통해 간접적으로 디비 데이터를 다룸
- 객체와 디비의 데이터를 자동으로 매핑해준다
- SQL 쿼리가 아니라 메서드로 데이터를 조작
- 객체간 관계를 바탕으로 sql을 자동으로 생성
- RDB의 관계를 Object에 반영하는것이 목적
- JPA, Hibernate 등

- SQL ← mapping → Object 필드
- SQL문으로 직접 디비를 조작
- 특정 RDB에 종속적
- 필드를 매핑시키는것이 목적
- Mybatis, jdbcTemplate 등



- 우리나라의 시장은 대부분 SI 또는 금융이기 때문에 비즈니스 로직이 매우 복잡
- 안정성과 속도를 매우 중요시
- MyBatis는 쿼리를 직접 작성해야하기 때문에 Hibernate에 능숙해진다면 생산성을 상당히 높일수 있다.

TALKING ABOUT

마지막으로

발표를 들어주셔서 감사합니다