스터디 3주차

2021.08.22

스프링 입문 섹션 6 : 스프링 DB 접근 기술

스프링 입문 섹션 7 : AOP

스프링 핵심 원리 섹션 1 : 객체 지향 설계와 스프링

인프런 김영한 스프링 완전 정복 로드맵 이용

목 차

Persistence (영속성)

데이터를 생성한 프로그램이 종료되더라도 사라지지 않는 데이터 특성을 의미
영속성이 없는 데이터는 단지 메모리에서만 존재해서 프로그램 종료하면 모두 소멸

Object Persistence(영구적인 객체)

메모리 상 데이터를 파일 시스템, 관계형 DB 등을 활용하여 영구적으로 저장하여 영속성 부여

- JDBC
- Spring JDBC(JdbcTemplate)
- Persistence Framework(Hibernate, Mybatis)

Persistence Layer

프로그램의 아키텍처에서, 데이터에 영속성 부여하는 계층
Persistence FrameWork를 이용한 개발이 많이 이루어짐

Persistence FrameWork

JDBC 프로그래밍의 복잡함이나 번거로움 없이 간단한 작업만으로도 DB와 연동되는 시스템을
빠르게 개발할 수 있으며 안정적인 구동 보장 Ex) JPA, Hibernate, Mybatis 등

ORM (객체 관계 맵핑)

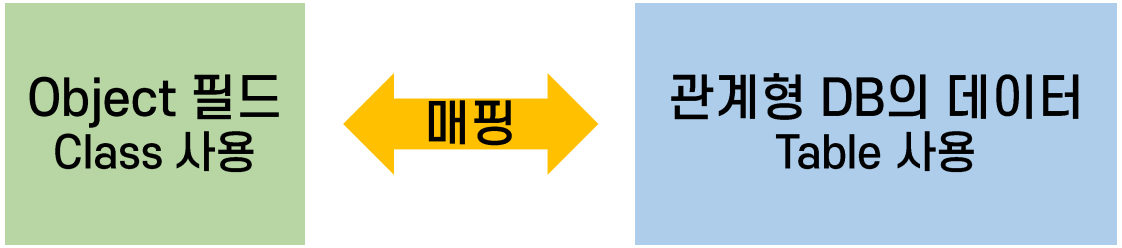
Object Relational Mapping의 약자

객체 지향 프로그래밍 언어와 DB 간의 호환되지 않는 데이터를 변환하는 프로그래밍 기법

객체 지향 언어에서 사용할 수 있는 가상 객체 DB 구축 방법

출처 - 위키백과

➔ 객체와 관계형 DB의 데이터를 자동으로 맵핑(연결)



객체는 객체대로, 관계형 DB는 관계형 DB대로 각자 설계

각자 사용하는 틀이 달라서 모델간 불일치가 존재

ORM을 통해 객체 간 관계를 바탕으로 SQL 자동 생성하여 불일치 해결

개발자가 데이터 접근보다 로직 자체에 집중할 수 있도록 해주는 Persistence Framework 기술

ORM 장단점

장점

객체 지향적인 코드로 인해 더 직관적이고 비즈니스 로직에 집중할 수 있게 도와준다.

재사용 및 유지보수의 편리성이 증가한다.

- ORM은 독립적으로 작성, 해당 객체들 재활용
- 매핑정보가 명확해서 ERD를 보는 것에 대한 의존도 ↓

DBMS에 대한 종속성이 감소한다.

- 종속적이지 않다는 것은 구현 방법 뿐만 아니라 많은 솔루션에서 자료형 타입까지 유효

단점

완벽한 ORM으로만 서비스를 구현하기가 어렵다.

- 프로젝트 복잡성이 커질수록 난이도 ↑
- 잘못 구현된 경우 속도 저하 및 심각할 경우 일관성이 무너지는 문제점 생길 수 있어 설계는 신중하게 해야한다.

프로시저가 많은 시스템에서는 ORM의 객체 지향적인 장점을 활용하기 어렵다.

- 이미 프로시저가 많은 시스템에선 다시 객체로 바꿔야한다.
- 이 과정에서 생산성 저하나 리스크가 많이 발생

JPA 제공하는 DB 기본키 생성 전략

직접 할당

기본 키를 애플리케이션에서 직접 할당

@id만 사용

```
@Id
@Column(name="username")
private String name;
```

@Id :기본 키 설정

@Column : 객체 필드를 테이블 컬럼에 매핑

자동 생성

대리 키 사용 방식

@GeneratedValue 사용

- IDENTITY : 기본키 생성을 DB에게 위임. MySQL
- SEQUENCE : DB 시퀀스를 사용하여 기본 키 할당. ORACLE
- TABLE : 키 생성 테이블 사용.
- AUTO : 선택한 DB에 따라 자동 지정. 기본 값

```
public class Member {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;
    ...
}
```

IDENTITY 활용

	IDENTITY	SEQUENCE	TABLE
DB 의존 여부	O	O	X

AOP(관점 지향 프로그래밍)

Aspect Oriented Programming 의 약자

어떤 로직을 기준으로 핵심적인 관점, 부가적인 관점으로 나누어서 보고 그 관점을 기준으로 각각 모듈화하겠다

MemberServie 클래스

```java
/* 전체 회원 조회 */
public List<Member> findMember() {
    Long start = System.currentTimeMillis();
    try {
        return memberRepository.findAll();
    }finally {
        long finish = System.currentTimeMillis();
        long timeMs = finish - start;
        System.out.println("findMembers " + timeMs + "ms");
    }
}
```

MemberServie 클래스

```java
public List<Member> findMember() {
    return memberRepository.findAll();
}
```

TimeTraceAOP 클래스

```java
public Object execute(ProceedingJoinPoint joinPoint) throws Throwable {
    long start = System.currentTimeMillis();
    try {
        return joinPoint.proceed();
    } finally {
        long finish = System.currentTimeMillis();
        long timeMs = finish - start;
        System.out.println("END: " + joinPoint.toString()+ " " + timeMs + "ms");
    }
}
```

시간 측정 로직을 공통 관심사, 전체 회원 조회 로직이 핵심 관심사

반복적으로 쓰이던 시간 측정 로직을 TimeTraceAOP 클래스로 모듈화 함.

핵심 관심사를 깔끔하게 유지할 수 있고 변경이 필요하면 TimeTraceAOP에서 로직만 수정하면 됨

@Around("execution(* hello.hellospring..*(..))")

@Around(“Pointcut”)

핵심관심사(비즈니스 메소드)의 실행 전과 후 Advice(기능, 해야할 일)메소드가 동작하는 형태

비즈니스 메소드는 proceed() 메소드가 진행하도록 한다

```
return joinPoint.proceed();
```

기능이 적용될 Join point(=끼어들 지점)를 나타내는 Pointcut을 속성값으로 가진다.

execution(접근제어자, 반환형 패키지를 포함한 클래스 경로 메소드 파라미터)

```
"execution(* hello.hellospring..*(..))"
```

첫번째 * : 접근제어자와 반환형 모두를 상관하지 않는다 는 의미

hello.hellospring.. : hello.hellospring 패키지를 포함한 모든 하위 디렉토리 를 의미

두번째 * : 어떠한 경로에 존재하는 클래스도 상관하지 않고 적용한다 는 의미

(..) : 메소드의 파라미터가 몇 개가 존재하던지 상관없이 실행하는 경우 를 의미

➔ 접근제어자, 반환형 상관하지 않고 hello.hellospring 패키지 포함한 모든 하위 디렉토리의 모든 클래스에서 메소드 파라미터 개수 상관없이 적용한다는 의미

참고한 사이트

영속성, ORM

https://gmlwjd9405.github.io/2019/02/01/orm.html

JPA 기본 키 생성 전략

https://dbjh.tistory.com/80

https://velog.io/@conatuseus/%EC%97%94%ED%8B%B0%ED%8B%B0-%EB%A7%A4%ED%95%91-2-msk0kq84v5#%EA%B8%B0%EB%B3%B8-%ED%82%A4-%EB%A7%A4%ED%95%91

AOP, @Around 어노테이션

https://engkimbs.tistory.com/746

https://velog.io/@max9106/Spring-AOP%EB%9E%80-93k5zjsm95

https://developer-joe.tistory.com/221

https://galid1.tistory.com/498