스프링입문-3 (김영한)

2021년 7월 30일 금요일 오후 9:16

컴포넌트 스캔과 자동 의존관계 설정 vs 직접 스프링 빈 등록

정형화되지 않거나 상황에 따라 구현 클래스를 변경해야 하면
직접 스프링 빈 등록한 경우가 수정하기 용이
컴포넌트 스캔의 경우 여러 코드를 수정해야 한다.

정형화된 경우에는 컴포넌트 스캔으로 처리함

스프링 컨테이너 먼저 찾아보고 static(정적 컨텐츠) 찾아봄
ex) Controller 로 매핑 해주었으면 우선적으로 처리하여 index.html 은 무시된다
아닌 경우는 static 에서 index.html 이 화면에 출력됨

html에서의 form 예시

```
<!DOCTYPE HTML>
<html xmlns:th="http://www.thymeleaf.org">
<body>
<div class="container">
    <form action="/members/new" method="post">
        <div class="form-group">
            <label for="name">이름</label>
            <input type="text" id="name" name="name" placeholder="이름을
입력하세요">
            <!-- name 이 서버로 넘어올 때의 키, placeholder 는 초기값 -->
        </div>
        <button type="submit">등록</button>
    </form>
</div> <!-- /container -->
</body>
</html>
```

```java
@PostMapping("/members/new")
public String create(MemberForm form) {
    Member member = new Member();
    member.setName(form.getName());

    memberService.join(member);

    return "redirect:/"; //홈화면으로 보내버리는 것
}
```

위의 경우 등록 버튼을 누르면 form의 action 의 url 에서 method 에서 명시한 post 를 타고 postmapping 부분 작동

Getmapping 은 보통 조회할 때
Postmapping 은 보통 데이터를 form 같은데 넣어 전달할 때 사용

스프링 DB 접근 기술
H2 (용량 작고, 가볍고, 웹 화면 제공)
순수 Jdbc (과거의 방식, 너무 빡셈, 요즘 안씀)
스프링 jdbcTemplate
JPA
스프링 데이터 JPA

h2
처음 생성 이후부터는 jdbc:h2:tcp://localhost/~/test 이렇게 접속
why? 파일로 접근하게 되면 어플리케이션과 웹 콘솔이랑 충돌날 수 있기에

DB 문
Member 테이블 만든것

```
create table member (
id bigint generated by default as identity,
name varchar(255),
primary key (id)
);
// bigint==long ,
```

// gernerated~ : 값이 설정되있지 않은 상태로 insert 하면 자동으로 id값 채워줌

개방-폐쇄 원칙(OCP : Open -Closed Principle)
:확장에는 열려있고, 수정,변경에는 닫혀있다

스프링의 DI(의존성 주입)을 사용하면 기존 코드 손대지 않고, 설정만으로 구현클래스
변경할 수 있다.

인터페이스를 기존코드를 바꾸지않아도 구현체를 바꿀 수 있는 장점 (객체지향)

@Transactional : 테스트 시작전에 Transaction을 시작하고, 테스트 완료 후에 항상 롤백한다.
=> DB에 데이터가 남지 않아 다음 테스트에 영향 주지 않음

통합 테스트 : 스프링 컨테이너,DB 까지 올리고...테스트
순수한 단위 테스트가 더 좋은 테스트일 가능성이 높음

스프링 JdbcTemplate : 순수 Jdbc와 동일한 환경설정
스프링 JdbcTemplate 과 MyBatis 같은 라이브러리는 Jdbc API에서 본 반복 코드를 대부분 제거
But SQL 은 직접 작성해야함

JPA : 기존의 반복 코드 + SQL 도 직접 만들어서 실행해줌
SQL 과 데이터 중심의 설계에서 객체 중심의 설계로 전환할 수 있음
=> 개발 생산성 크게 높일 수 있음

jpa.hibernate 라이브러리 주로 사용
jpa.hibernate.ddl-auto =create : db 자동으로 만들어줌

JPA (Java Persistence API) : 자바의 표준 인터페이스 모음

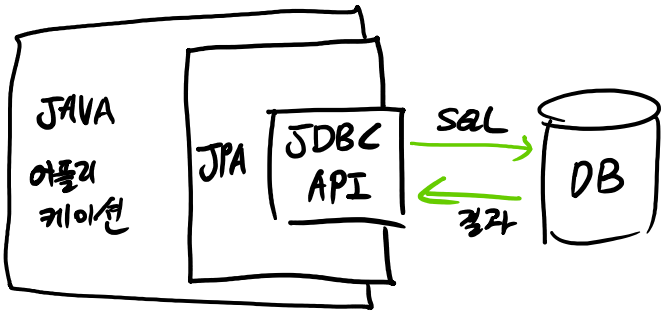
- 인터페이스라 실제 동작하는 것 아님
- 대표적인 구현체로 Hibernate, EclipseLink, DataNucleus

EJB : 과거의 자바 표준 , 과거의 ORM

- 코드 지저분
- API의 복잡성 높음 (인터페이스 많이 구현해야 함)
- 속도가 느림

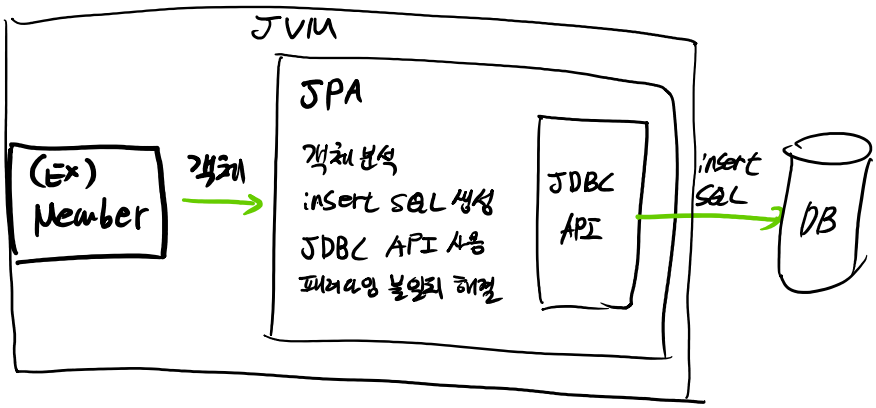
Hibernate : ORM 프레임 워크 , 오픈소스 SW

JPA의 동작 과정

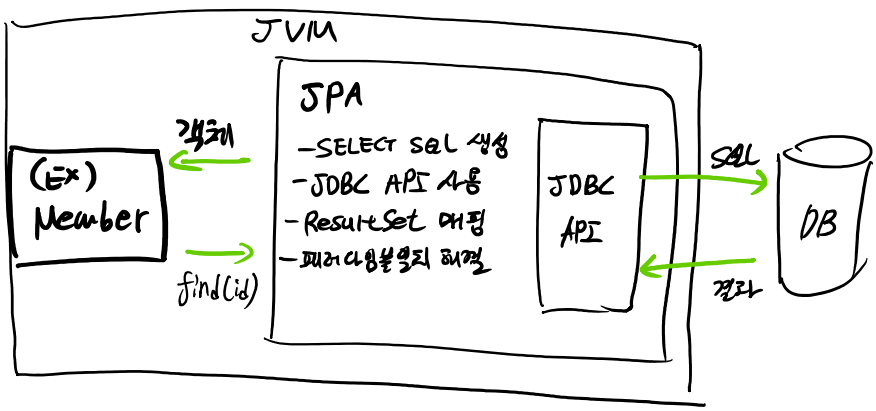


즉 JPA 내부의 JDBC API 사용해서
SQL 호출하여 DB와 통신

저장 과정



조회 과정



스프링부트가 세팅값이나 DB 연결 정보 등등 합쳐서 EntityManager 라는 것을 만들어줌
내부 리소스를 다 들고 있어서 db 통신등을 처리해줌
고로 Jpa 를 쓰려면 EntitiyManger를 주입받아야 된다

jpa 주의사항 : @Transactional 있어야 함 데이터 저장할때 Transaction 안에서 일어나야함
why??? =>????