스프링 3계층 아키텍처

2022.02.06

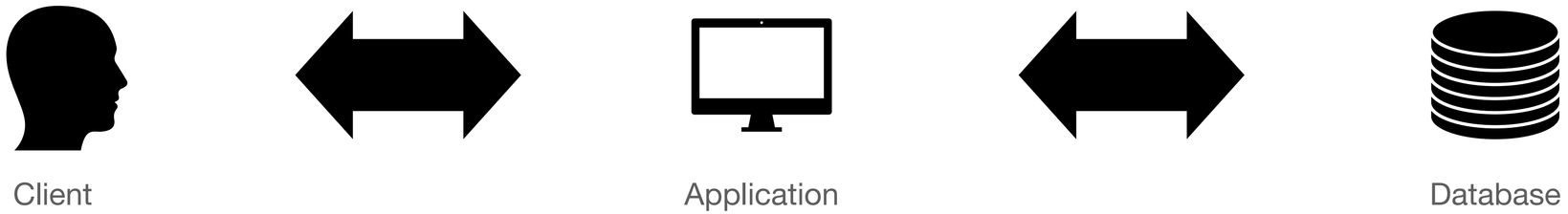
아키텍처란? (Architecture)

시스템 아키텍처란?

- 시스템의 구조, 행위 등을 정의하는 개념적 모형
- 시스템 목적을 달성하기 위해 시스템의 각 컴포넌트가 무엇이며 어떻게 상호작용하는지, 정보가 어떻게 교환되는지 나타내는 것
- 시스템을 만드는 과정에서 시스템 디자인에 대한 제약
- 설계 및 구현을 수행하는 사람에게 효율적인 기술을 전달하는 지식 전달 수단 또는 규범

웹 3계층 아키텍처란?

- 애플리케이션을 3개의 논리적 및 물리적 컴퓨팅 계층으로 구성하는 형태
- Client Tier, Application Tier, Data Tier



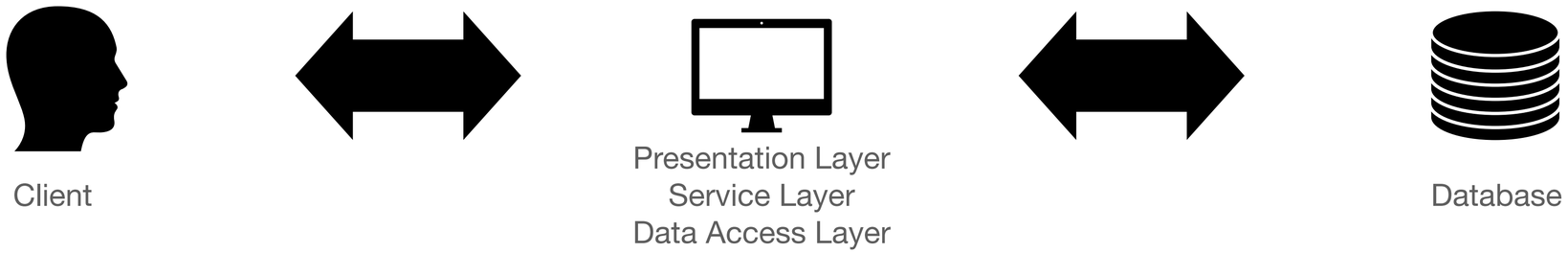
3계층 아키텍처의 장점

- 안정성 향상
- 업무 효율 증가
- 확장성 개선
- 보안성 강화

스프링 3가지 계층

스프링 3가지 계층

- 프레젠테이션 계층 : 웹 기반으로 요청과 응답을 처리
- 서비스 계층 : 비즈니스 로직을 담당
- 데이터 접근 계층 : 데이터베이스와 연동되어 데이터에 접근하는 계층



프레젠테이션 계층

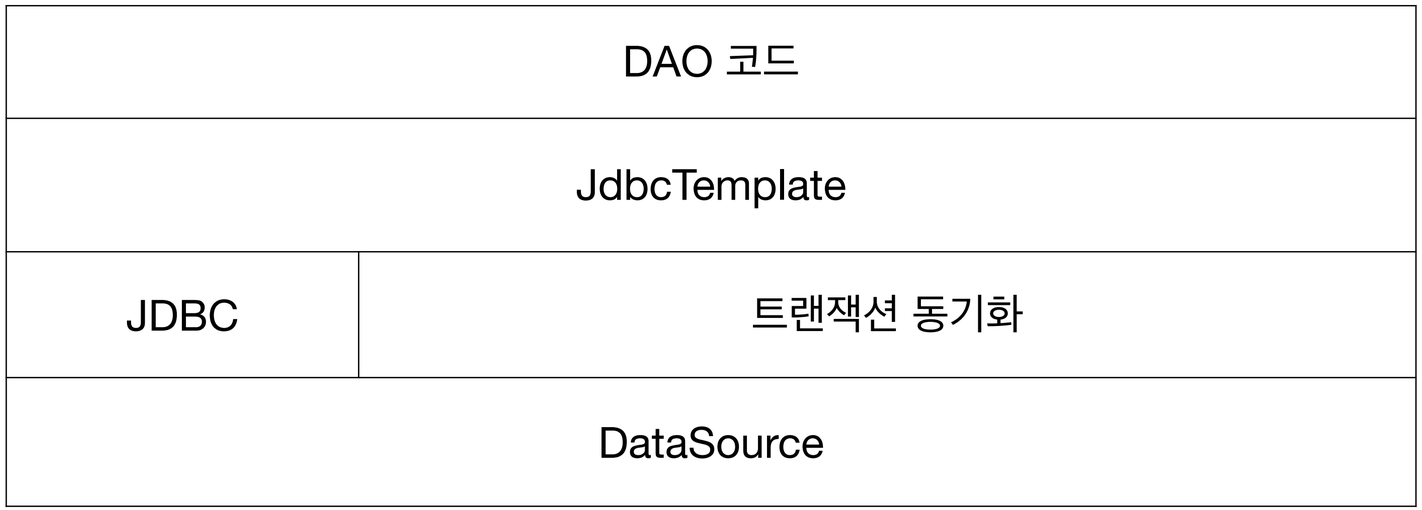
- 사용자와 상호 작용하는 담당
- HTTP 요청을 수신하고 응답하는 담당

서비스 계층

- 서비스의 핵심 로직을 담당
- 프레젠테이션 계층으로 부터 받은 데이터의 유효성 검사를 담당
- 어떠한 Data Access를 선택할지 결정
- 데이터 접근 계층을 호출하고 이를 활용함

데이터 접근 계층

- 데이터를 저장하거나 조회하기 위한 계층
- JdbcTemplate 추상화를 위한 계층



계층화를 하는 이유

- 독자적 개발 가능
- 테스트 가능
- 개발과 변경 작업 효율이 좋아짐
- 구현 방법이나 세부 로직에 서로 영향을 주지 않고 변경될 수 있음

계층화의 원칙

- 각 계층은 응집도가 높으면서, 다른 계층과는 낮은 결합도를 갖는다.
- 상위 계층은 하위 계층을 알 수 있지만, 하위 계층은 상위 계층을 인식하지 못하게 해야 한다.

프레젠테이션 계층

```java
@Controller
@RequiredArgsConstructor
public class MemberController {

    private final MemberService memberService;

    @PostMapping("/members/new")
    public String create(@Valid MemberForm form, BindingResult result) {

        if (result.hasErrors()) {
            return "members/createMemberForm";
        }

        memberService.createMember(form.toServiceDTO());

        return "redirect:/";
    }
```

서비스 계층

```java
@Service
@Transactional(readOnly = true)
@RequiredArgsConstructor
public class MemberService {

    private final MemberRepository memberRepository;

    @Transactional
    public Long join(Member member) {
        memberRepository.save(member);
        return member.getId();
    }

    @Transactional
    public Long createMember(MemberServiceDto memberServiceDto) {
        Address address = new Address(memberServiceDto.getCity(), memberServiceDto.getStreet(), memberServiceDto.getZipcode());

        Member member = new Member();
        member.setName(memberServiceDto.getName());
        member.setAddress(address);

        validateDuplicateMember(member);

        return this.join(member);
    }

    private void validateDuplicateMember(Member member) {
        List<Member> findMembers = memberRepository.findByName(member.getName());
        if (!findMembers.isEmpty()) {
            throw new IllegalStateException("이미 존재하는 회원입니다.");
        }
    }
```

데이터 접근 계층

```java
@Repository
@RequiredArgsConstructor
public class MemberRepository {

    private final EntityManager em;

    public void save(Member member) { em.persist(member); }

    public Member findOne(Long id) { return em.find(Member.class, id); }

    public List<Member> findAll() {
        return em.createQuery("select m from Member m", Member.class)
                .getResultList();
    }

    public List<Member> findByName(String name) {
        return em.createQuery("select m from Member m where m.name =:name", Member.class)
                .setParameter("name", name)
                .getResultList();
    }

}
```

감사합니다.