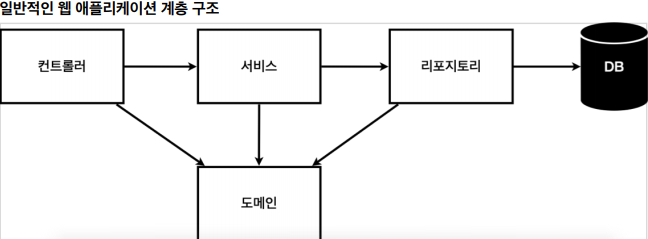
강의자료 P.17~

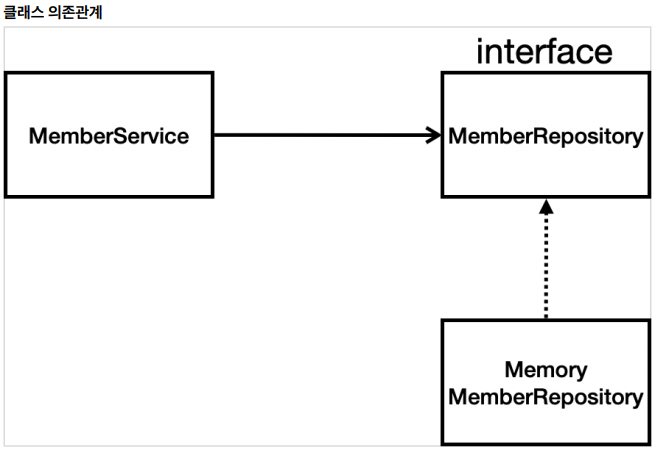


컨트롤러: 웹 MVC의 컨트롤러 역할 (API생성할 때)

서비스: 핵심 비즈니스 로직 구현 (비즈니스 도메인 객체를 활용해 구현 ex) 회원 중복가입 불가)

리포지토리: 데이터베이스에 접근, 도메인객체를 DB에 저장하고 관리

도메인: 비즈니스 도메인 객체, 예) 회원, 주문, 쿠폰 등등 주로 데이터베이스에 저장하고 관리되는 비즈니스 도메인 객체



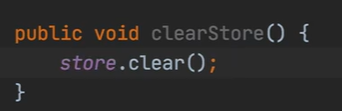
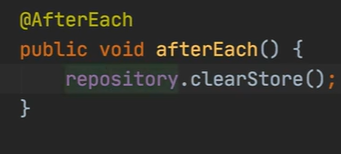
interface로 구현한 이유: 아직 데이터 저장소가 선정되지 않았기 때문에 인터페이스로 구현 클래스를 변경할 수 있도록 설계함.

**--회원 도메인과 리포지토리 만들기**

검증 방법: main 메소드를 통해 실행하거나 웹 애플리케이션의 컨트롤러를 통해 검증함

🡪준비시간이 오래 걸리고 반복 실행이 어렵다는 단점이 존재 🡪 자바에선 JUnit이라는 프레임워크로 실행해 이러한 문제점을 해결

테스트가 하나 끝나면 리포지토리를 깔끔하게 클리어 해줘야함

-->

TDD(테스트 주도 개발):강의와 반대로 테스트 틀을 먼저 만들고 코드를 짜서 검사하는 방식

@AfterEach: 한번에 여러 테스트를 실행하면 DB에 직전 결과가 남아 다음 테스트에 영향을 줄 수 있기 때문에 @AfterEach를 사용해 각 테스트가 종료될 때 마다 메모리 DB에 저장된 데이터를 삭제해줌.

@BeforeEach: 각 테스트 실행 전에 호출되어 테스트가 서로 영향이 없도록 항상 새로운 객체를 생성하고 의존관계 도 새로 맺어준다

**--스프링 빈과 의존관계**

--컴포넌트 스캔과 자동 의존관계 설정

스프링 컨테이너에 등록하고 계속 사용 가능

생성자 호출: @Autowired –스프링 컨테이너에서 멤버 서비스 자동 호출 (생정자 1개면 생략 가능)

@Component를 포함하는 것들 (모두 스프링 빈으로 자동 등록됨)

@Service: 스프링에 올라올 때 스프링 서비스에 등록해줌

@Repository: 똑같이 등록

@Controller

참고: 스프링은 스프링 컨테이너에 스프링 빈 등록할 때 싱글톤으로 등록함(유일하게 하나만 등록해서 공유). 따라서 같은 스프링 빈 = 같은 인스턴스 / 설정으로 싱글톤 아니게 가능하지만 대부분 싱글톤 사용

--자바 코드로 직접 스프링 빈 등록 (위와 달리 에노테이션을 제거하고 진행)

DI에는필드주입, setter 주입, 생성자 주입 이렇게 3가지방법이있다. 의존관계가 실행중에 동적으로변하는 경우는 거의 없으므로 생성자 주입을 권장한다.

@Autowired를 통한 DI는 스프링이 관리하는 객체에서만 동작, 스프링 빈으로 등록하지 않고 내가 직접 생성한 객체에서는 동작하지 않는다.