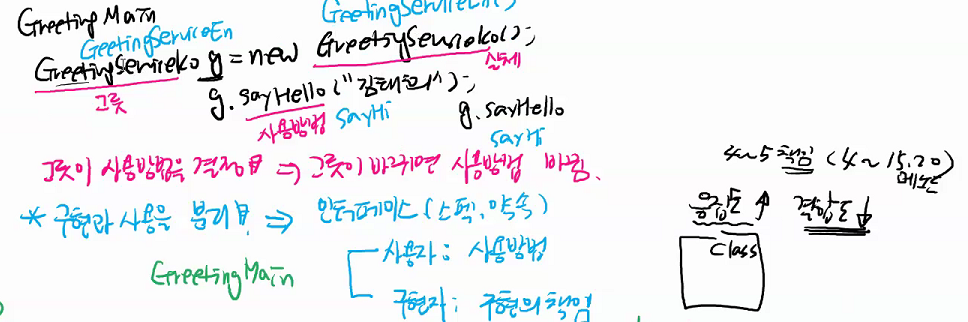
Framework

뼈대, 틀, 인프라, 반제품

개발 생산성 향상(개발 시간 감소), 코드 품질 향상

MyBatis => 데이터베이스 연동 코드 작성을 재사용 가능한 기능을 제공해서 손쉽게 작성 가능하도록 지원

외부에 제공하고 싶은 기능만 인터페이스로 드러냄



객체를 만드는 방법 1. 직접 생성 2. Factory(일괄 변경 가능, 하나의 객체)

DI(Dependency Injection)

의존성 주입 (의존 객체를 수동적으로 받는 것)

Has a : association 관계이다

사용 목적 : 의존객체의 교체(변경) 용이하게 관리하도록 하기 위해

사용하기 위한 전제조건: 상위타입을 활용한 설계 및 사용 (ex. interface)

Loose coupling의 강점을 가짐

객체는 인터페이스에 의한 의존 관계만을 알고 있으며, 이 의존 관계는 구현 클래스에 대한 차이를 모르는 채 서로 다른 구현으로 대체가 가능

DI는 크게 setter injection, constructor injection, method injection 세가지

컨테이너가 의존 구조를 object에 설정할 수 있도록 설정을 지정해줘야한다.

* 설정 방식: xml, annotation, file

@Scope(“singleton”) / xml에 scope=”singleton”

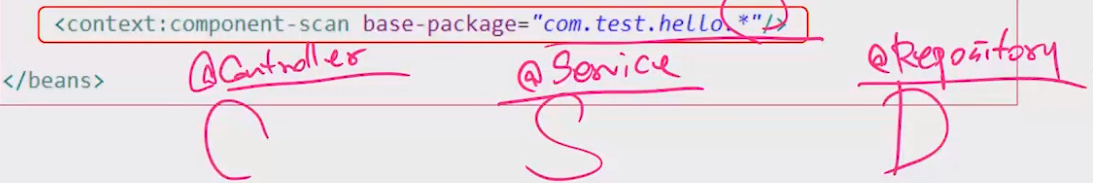
Contatiner란 객체의 생성, 사용, 소멸에 해당하는 라이프사이클 담당

라이프 사이클을 기본으로 애플리케이션 사용에 필요한 주요 기능 제공

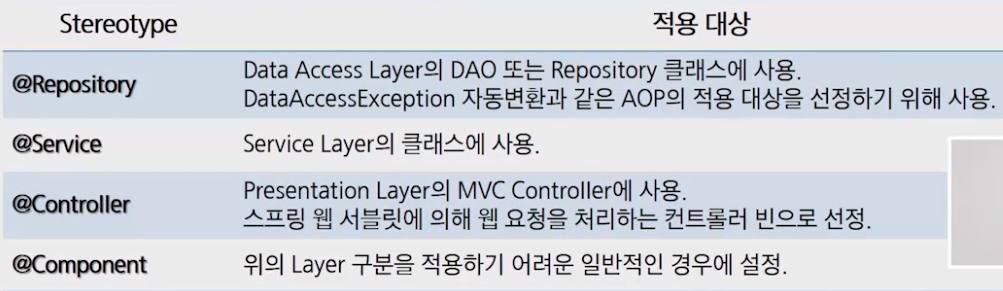
Factory 나 Singleton 패턴을 직접 구현하지 않아도 지원해줌

스프링 빈 설정 : XML

annotation으로 빈을 설정할 경우 xml에 component-scan 을 설정해야함



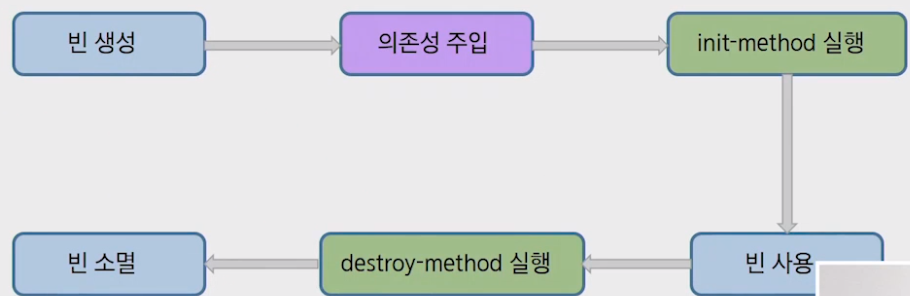
Hello 밑 패키지의 클래스를 찾아서 스프링이 관리하도록 등록



@Autowired type으로 구분

@qualifier name으로 구분

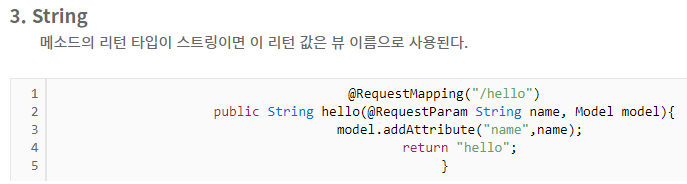
스프링 빈 생명주기

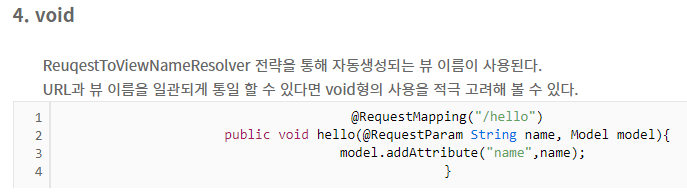


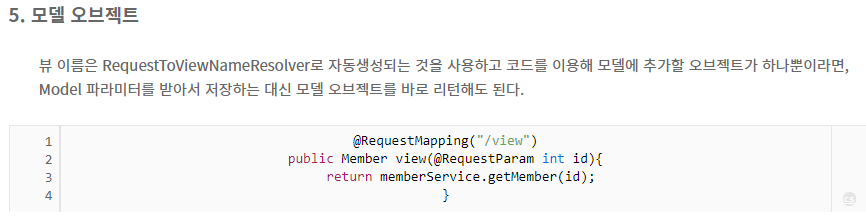
스프링 컨트롤러

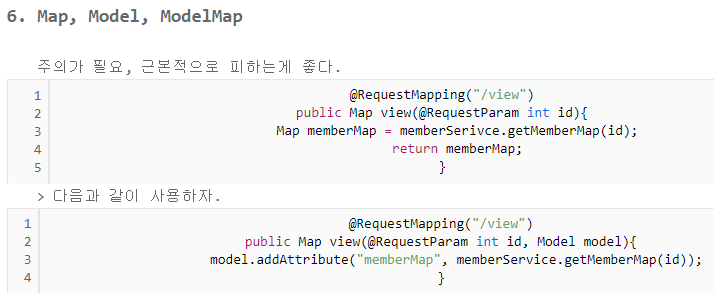
리턴타입











Parameter값



