자바 스프링 프레임워크

가로선

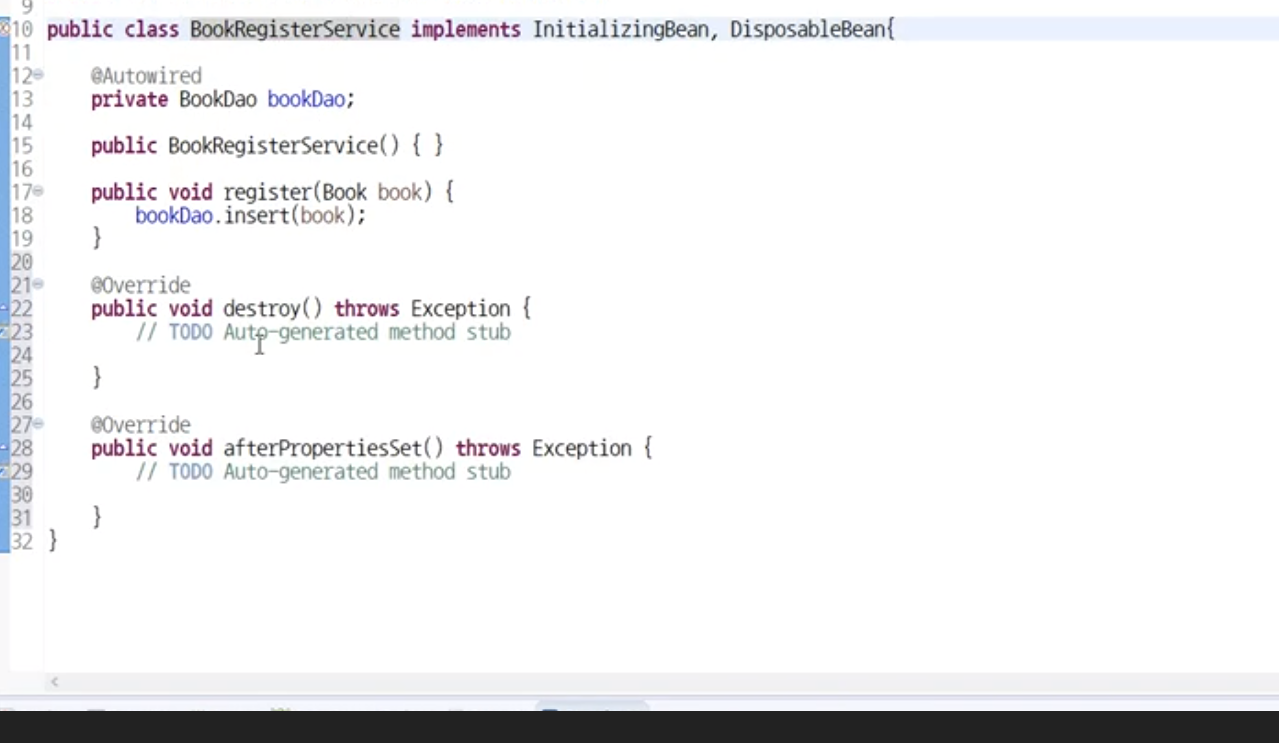
# 생명 주기

1. 컨테이너 생성 시 Bean 객체 생성
2. getBean 등으로 Bean 객체 사용
3. close()를 이용해 컨테이너 소멸과 동시에 Bean 객체 소멸

=> Bean 객체의 생명주기는 스프링 컨테이너의 생명주기와 같다.

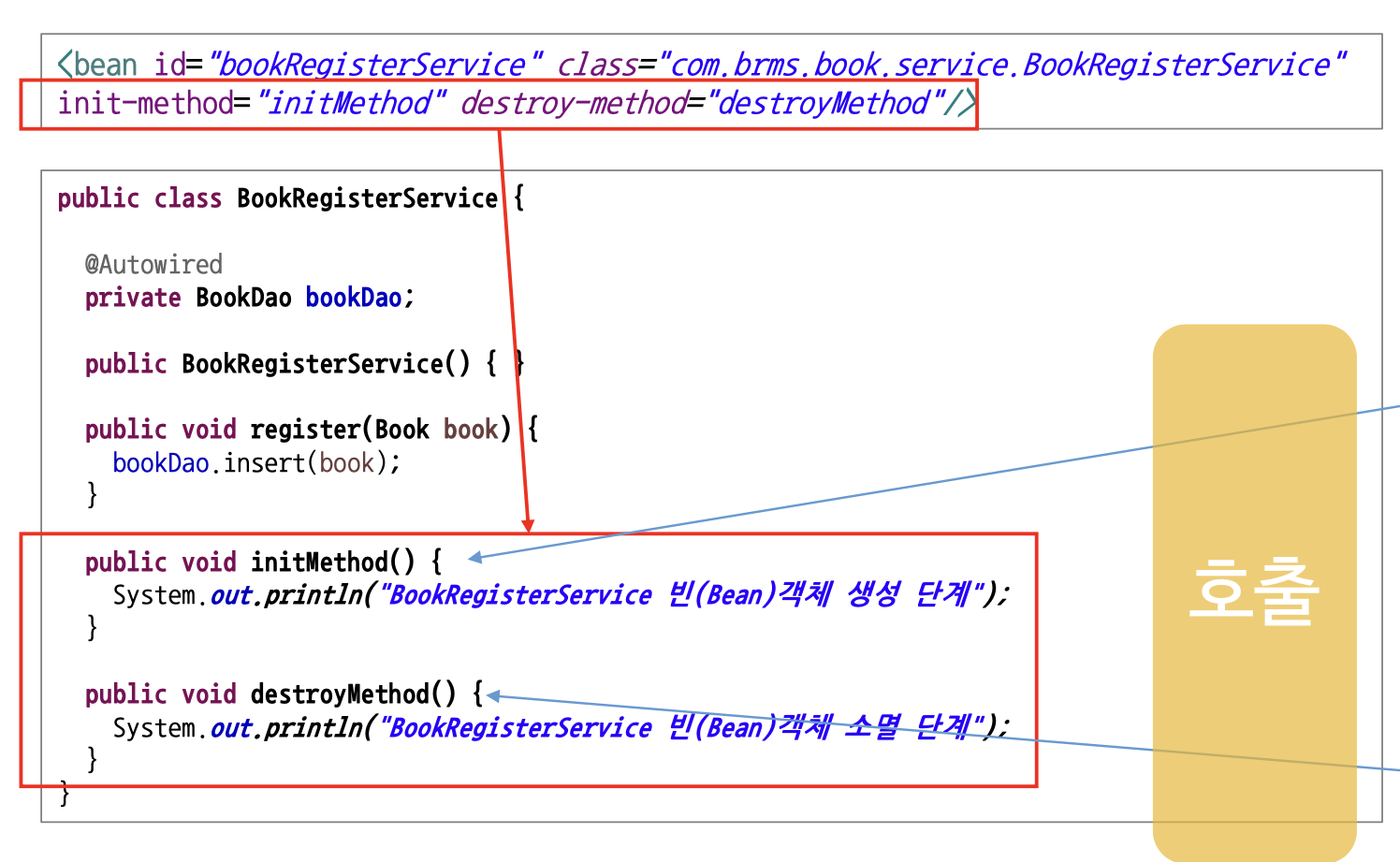
# Bean의 생성 및 소멸 시에 작동하는 함수 생성

## 인터페이스 사용



* afterPropertiesSet은 Bean이 생성될 때 실행
* destroy는 Bean이 소멸될 때 실행

## method 사용



* xml에서 init-method와 destroy-method를 지정 후, 해당 객체에 함수 생성

# 어노테이션

## @Configuration

* Bean 객체 생성을 자바에서 실행

## 





# 

# Config 파일 분리

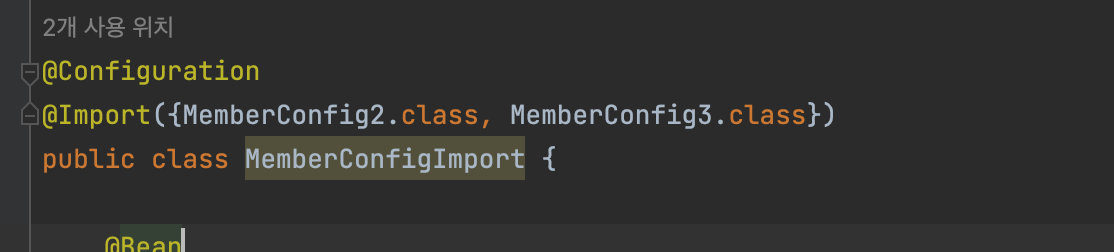
* Config 파일을 분리하더라도 같은 컨테이너에 올라가 때문에 하나의 Configuration 파일 처럼 사용 가능
* 분리된 Config 파일에서 다른 파일의 Bean 객체를 사용하기 위해서는 @Autowired 사용
* 



## Main에서 Config 파일 3개 사용

* AnnotationConfigApplicationContext ctx = new AnnotationConfigApplicationContext(MemberConfig1.class, MemberConfig2.class, MemberConfig3.class);

## @Import

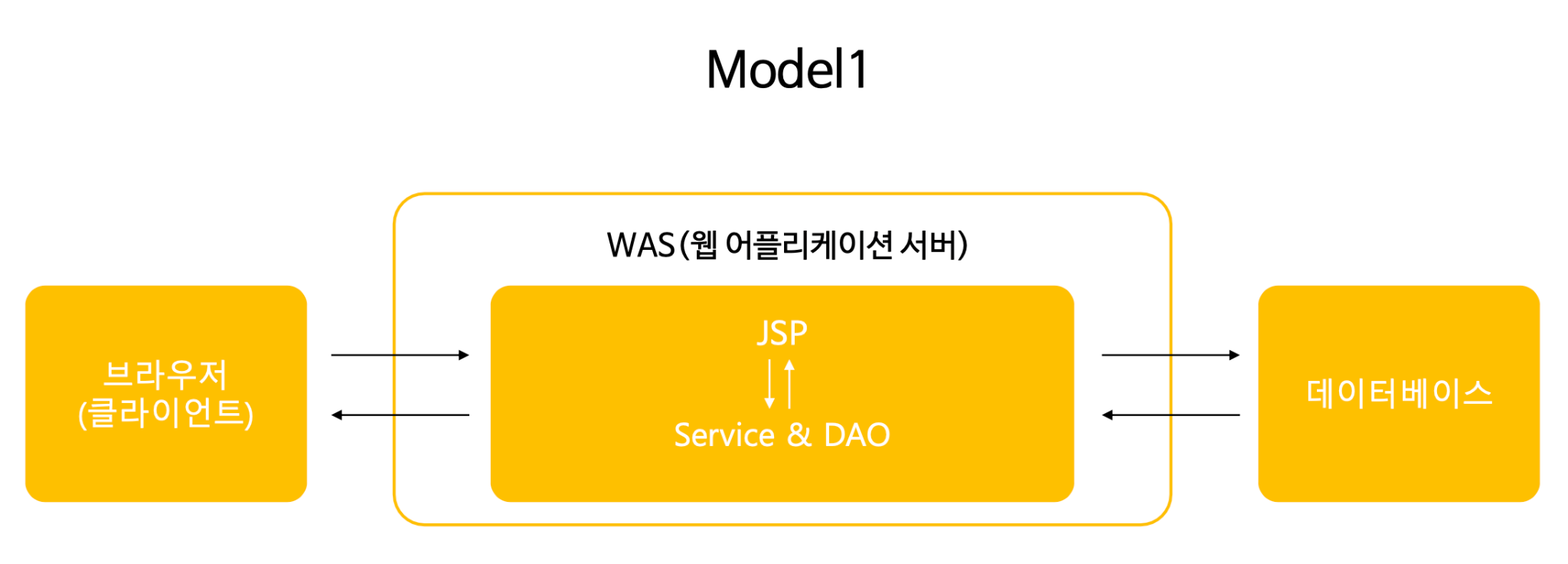


* Import 사용 시 AnnotationConfigApplicationContext에서 하나의 파일만 호출 가능
  + AnnotationConfigApplicationContext ctx = new AnnotationConfigApplicationContext(MemberConfigImport.class);

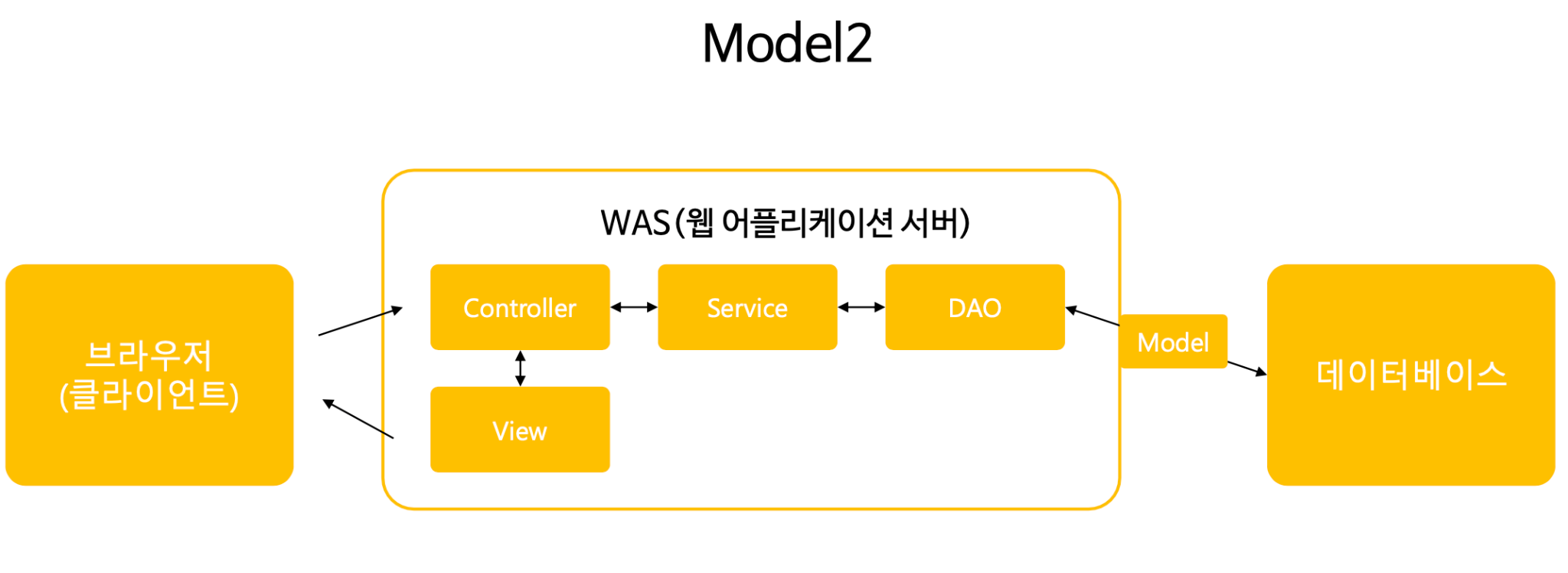
# 

# 웹 프로그래밍의 2개 모델

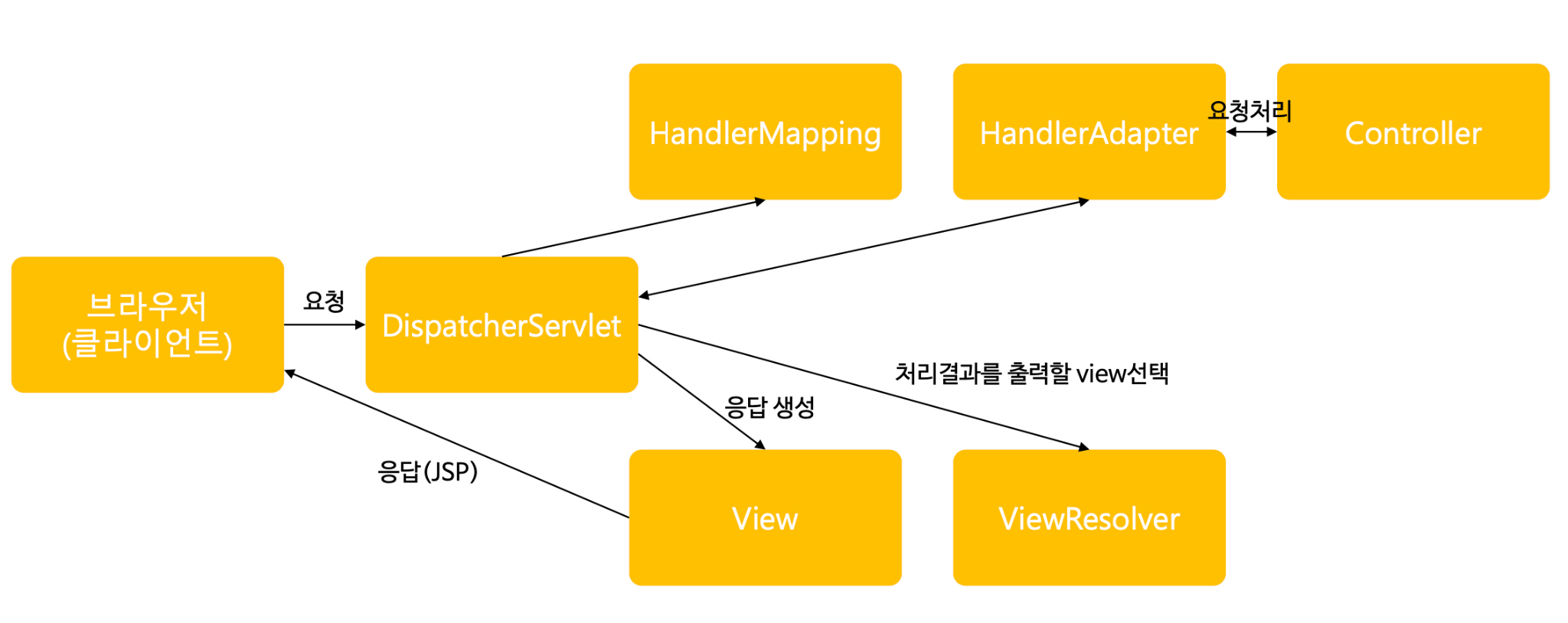
## model 1



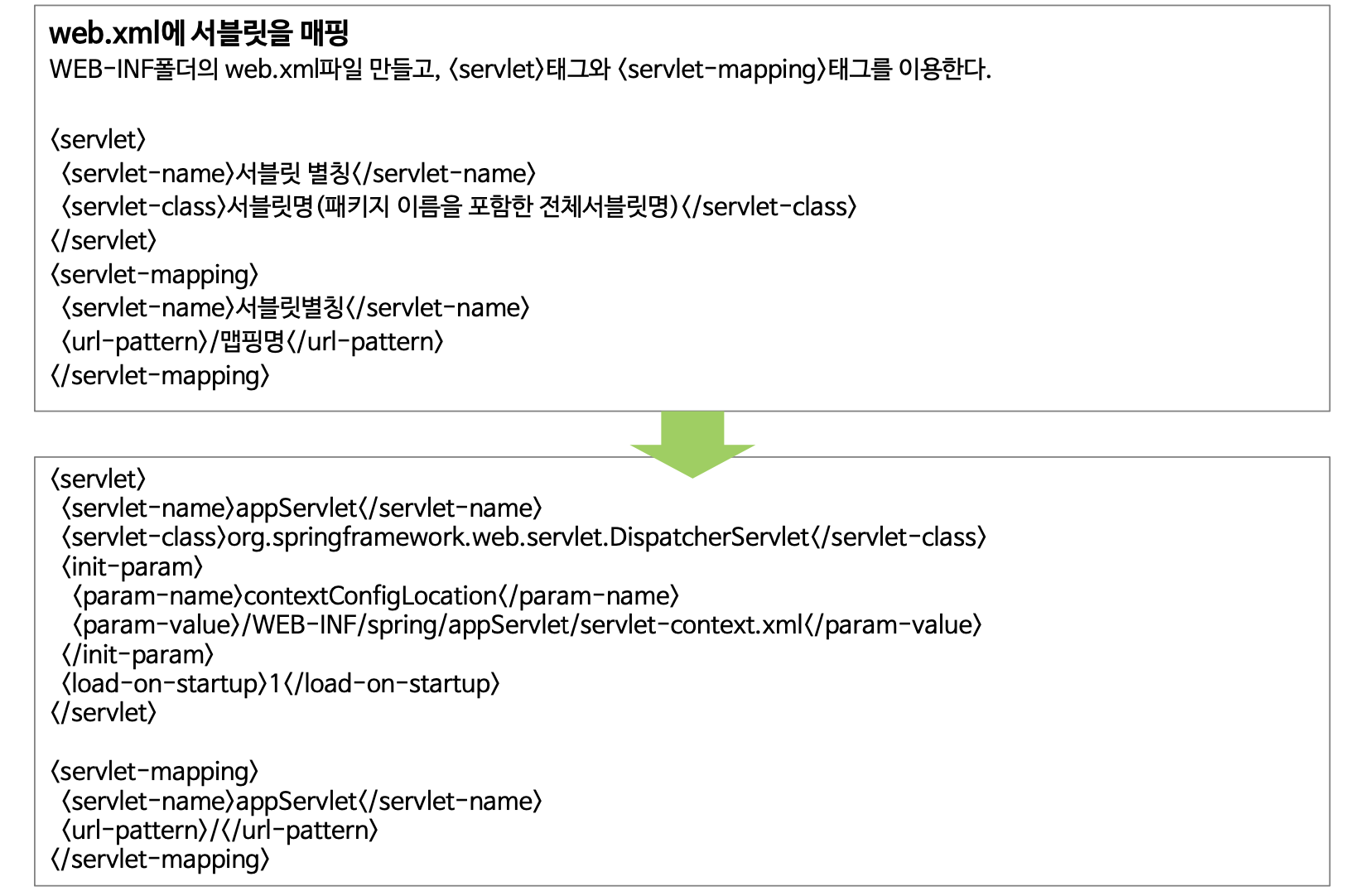
## Model 2



# 스프링 MVC 프레임 워크 설계 구조



# DispatcherServlet 설정



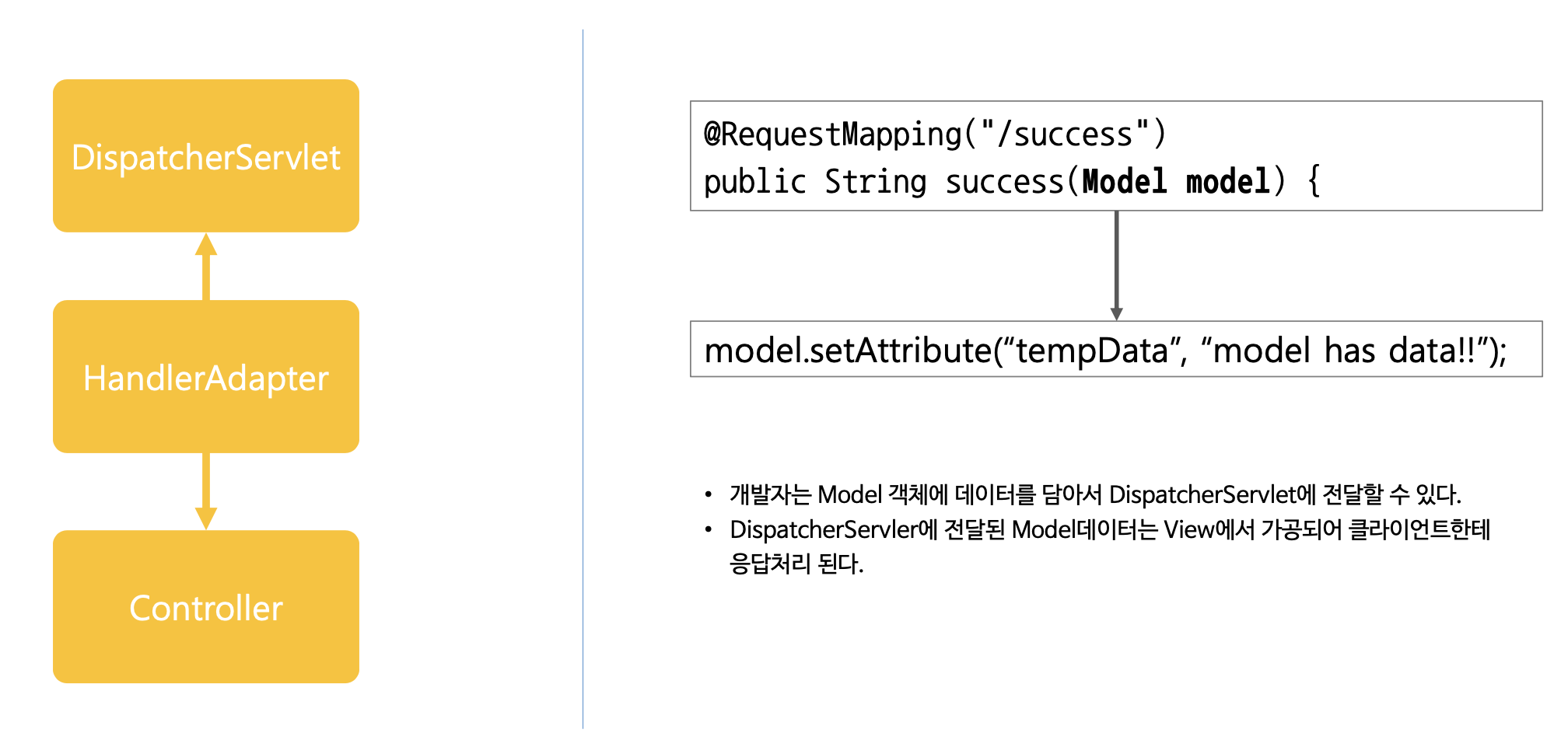
# Controller

* @Controller을 사용해 Controller 생성
* 스프링 설정파일(servlet-context.xml)에 <annotation-driven /> 추가

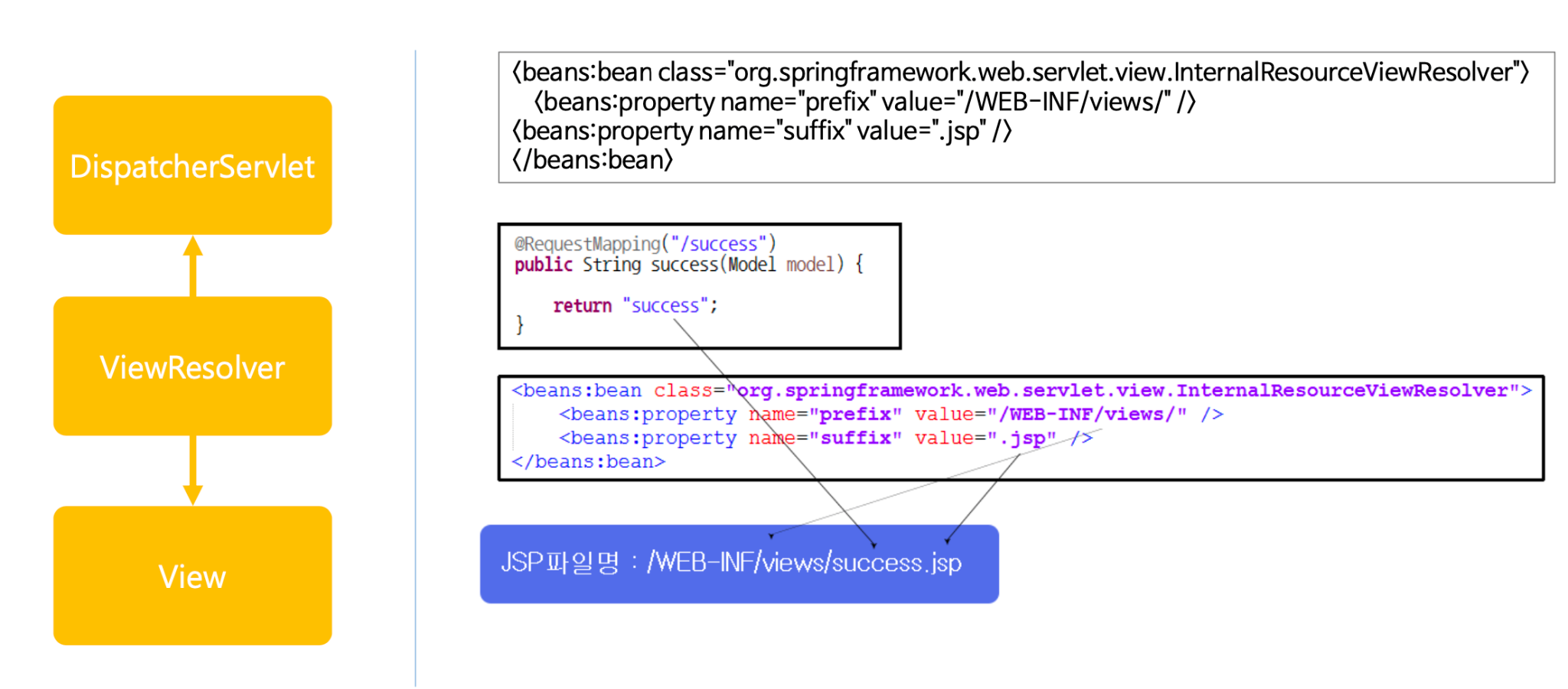
# RequestMapping

* @RequestMapping(“/path”)
* 사용자가 호출하는 요청을 매핑 시켜줌

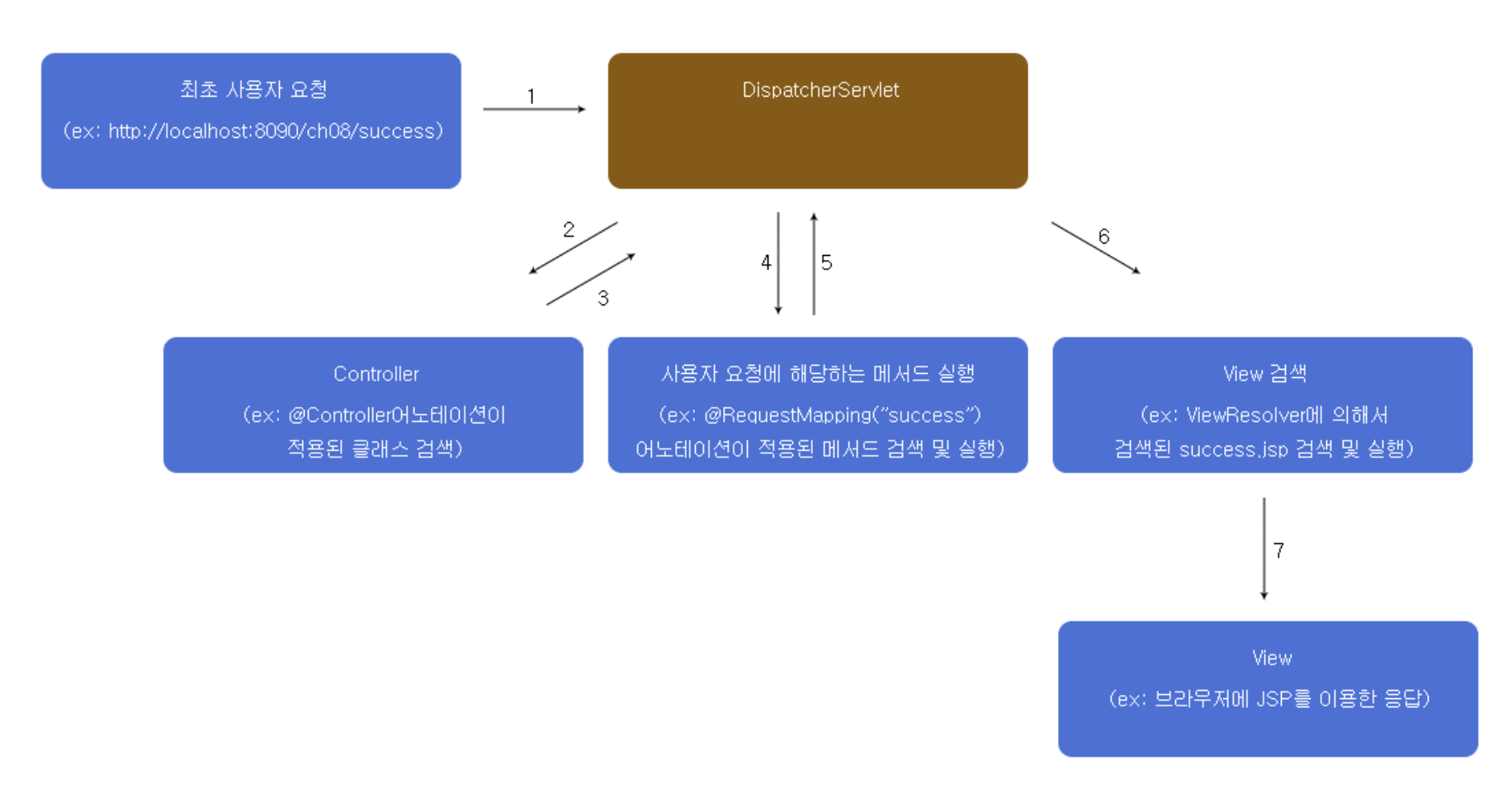
# Model 타입의 파라미터



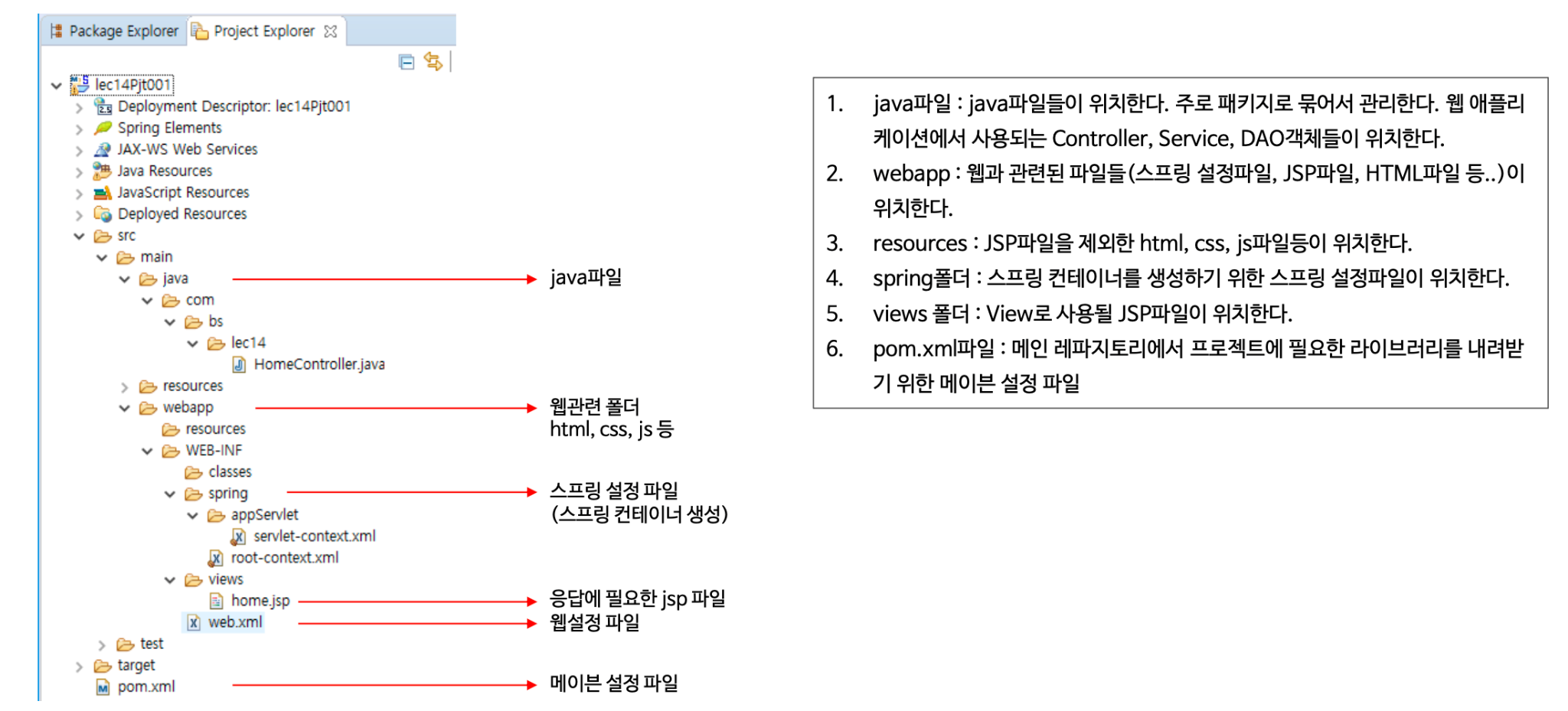
# View 객체



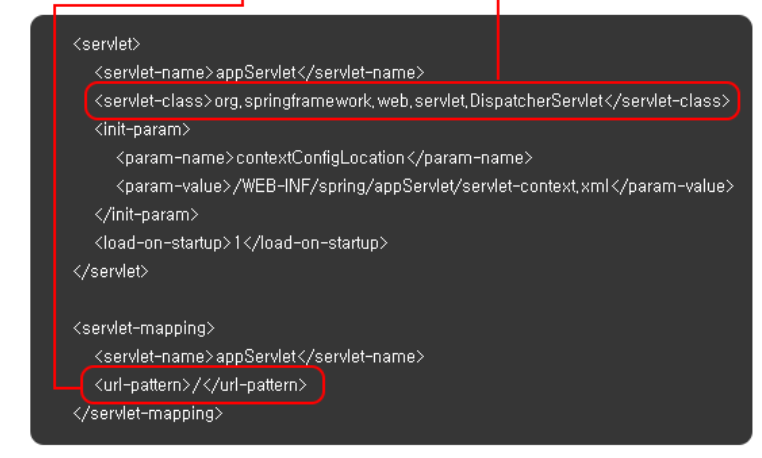
# 정리



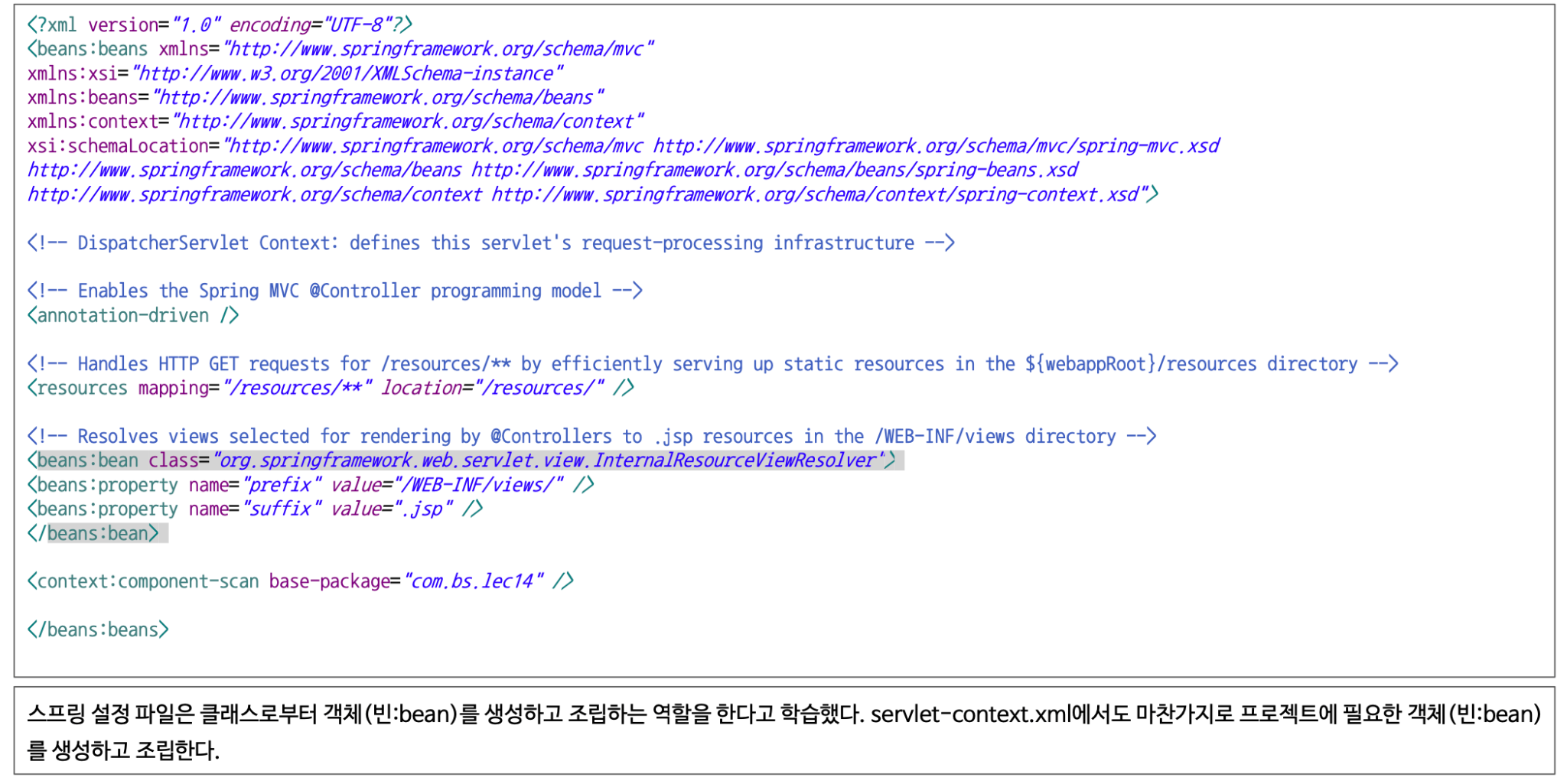
# STS 프로젝트 전체 구조



# Web.xml

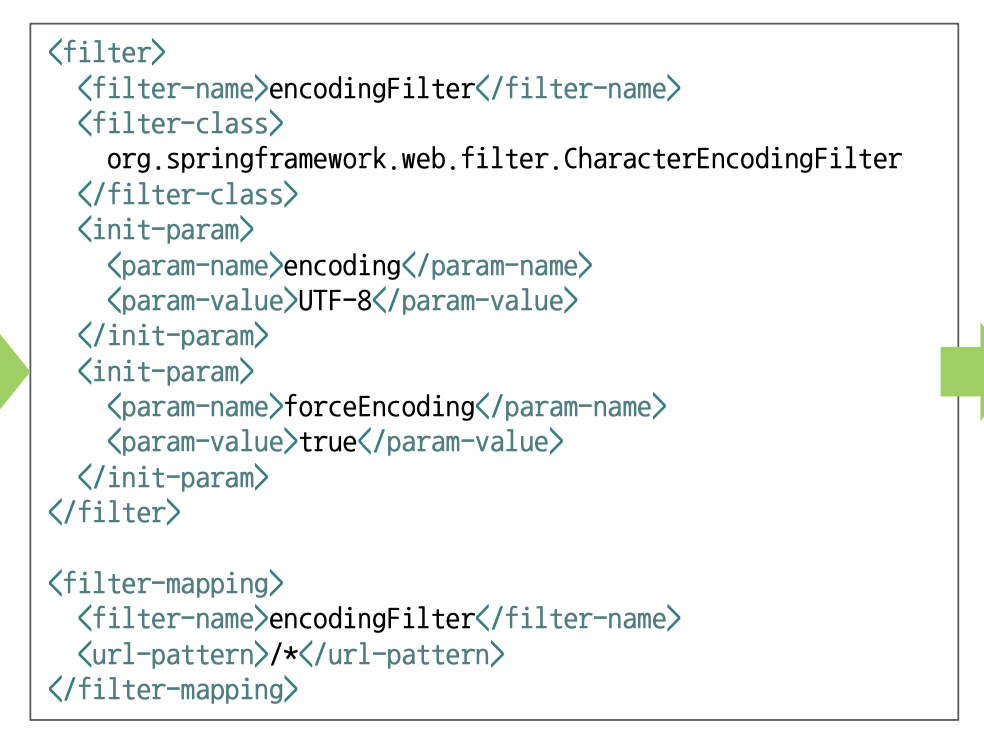


# servlet-context.xml(스프링 설정파일)



# 한글 패치

## web.xml에 filter 추가



# service 객체 구현

방법1 : new 연산자를 이용한 service 객체 생성 및 참조

### controller

MemberService service = new MemberService();

* 일반 자바와 같은 방법

방법2 : 스프링 설정파일을 이용한 서비스 객체 생성 및 의존 객체 자동 주입

### servlet-context.xml

<beans:bean id="service" class="com.bs.lec17.member.service.MemberService">

</beans:bean>

### controller

@Autowired MemberService service;

* @Autowired 사용

방법3 : 어노테이션을 이용해서 서비스 객체 생성 및 의존 객체 자동 주입

### service.java

@Service OR @Component OR @Repository

public class MemberService implements IMemberService {

### controller

@Autowired

MemberSerive service;

# Dao 객체 구현

### dao.java

@Compononet OR @Repository

public class MemberDao implements IMemberDao {

### service.java

@Autowired

MemberDao dao;

# 데이터 전송 방식

## POST

jsp: <form ~ method=”post”>

controller: @RequestMapping(value=”/memJoin”, method=RequestMethod.POST)

## GET

jsp: <form ~ method=”get”>

controller: @RequestMapping(”/memJoin”)

OR

controller: @RequestMapping(value=”/memJoin”, method=RequestMethod.GET)

# Controller 전체 주소 지정

@Controller

@RequestMapping(“/member”)

public class MemberController{

# Jsp에서 값 받아오기

## HttpServletRequest를 사용해 값 받아오기

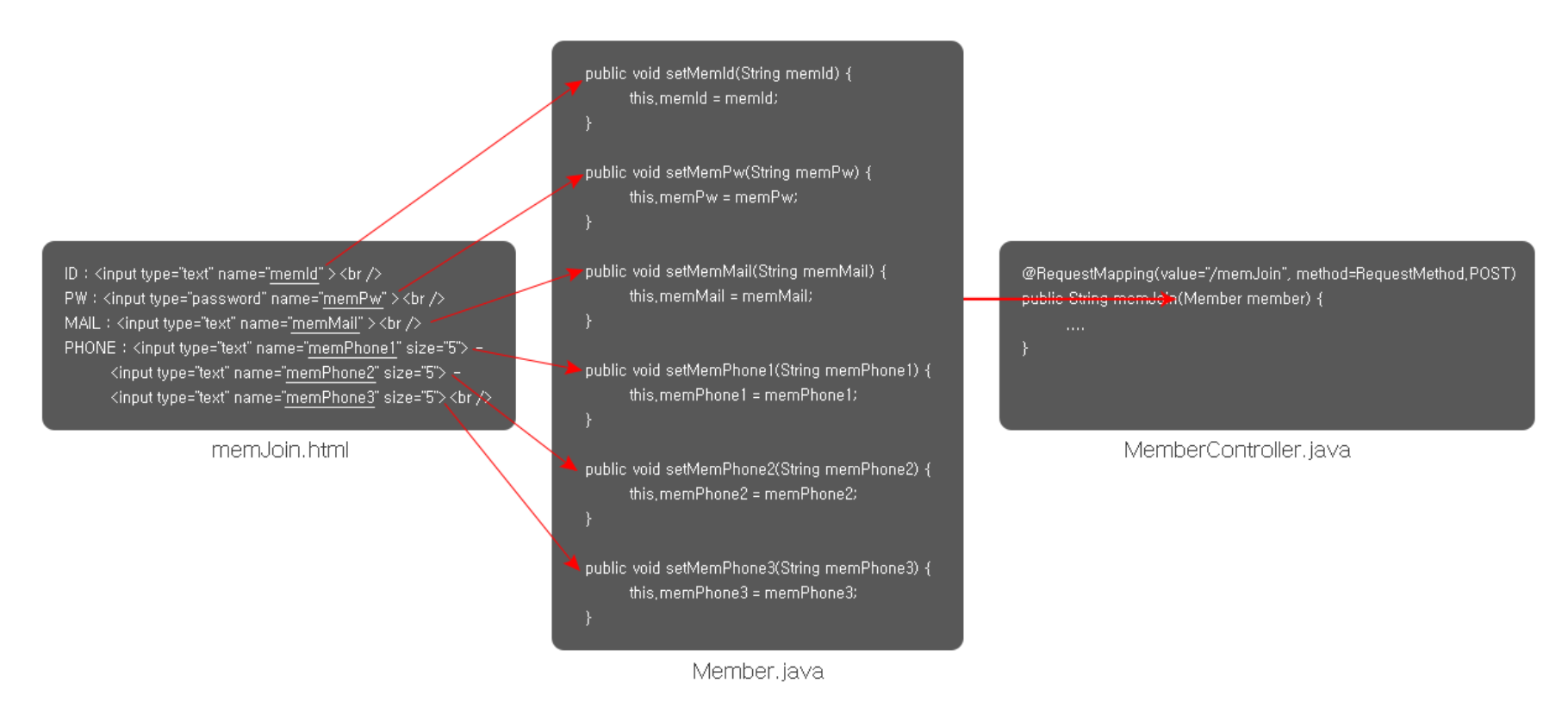
public String memLogin(Model model, HttpServletRequest request){

String memId = request.getParameter(“memId”);

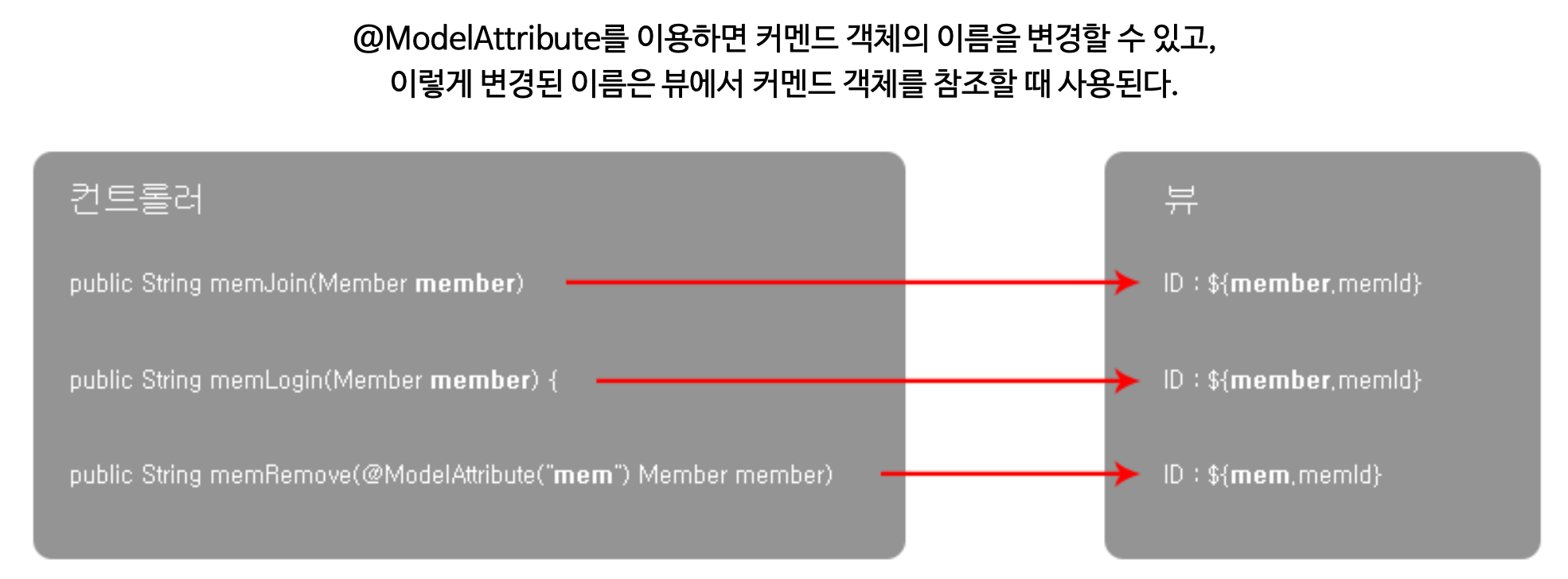
## Annotation을 사용해 값 받아오기

public String memLogin(Model model, @RequestParam(“memId”) String memId) {

## Command 객체 사용



# Command 객체 이름 변경하기



# ModelAttribute

### ex) 메소드 실행 시간 구하기

#### controller

@ModelAtrribute(“serverTime”)

public String getServerTime(Locale locale){

Date date = new Data();

DateFormat dateFormat = DateFormat.getDateTimeInstance();

return dateFormat.format(data);

}

#### jsp

<body>time : ${serverTime}</body>

@ModelAttribute의 경우 같은 파일 안의 메소드 실행 시 무조건 실행

따라서, jsp 호출 시에도 바로 적용되어 나옴

# ModelAndView

model과 View의 이름을 함께 전달

