

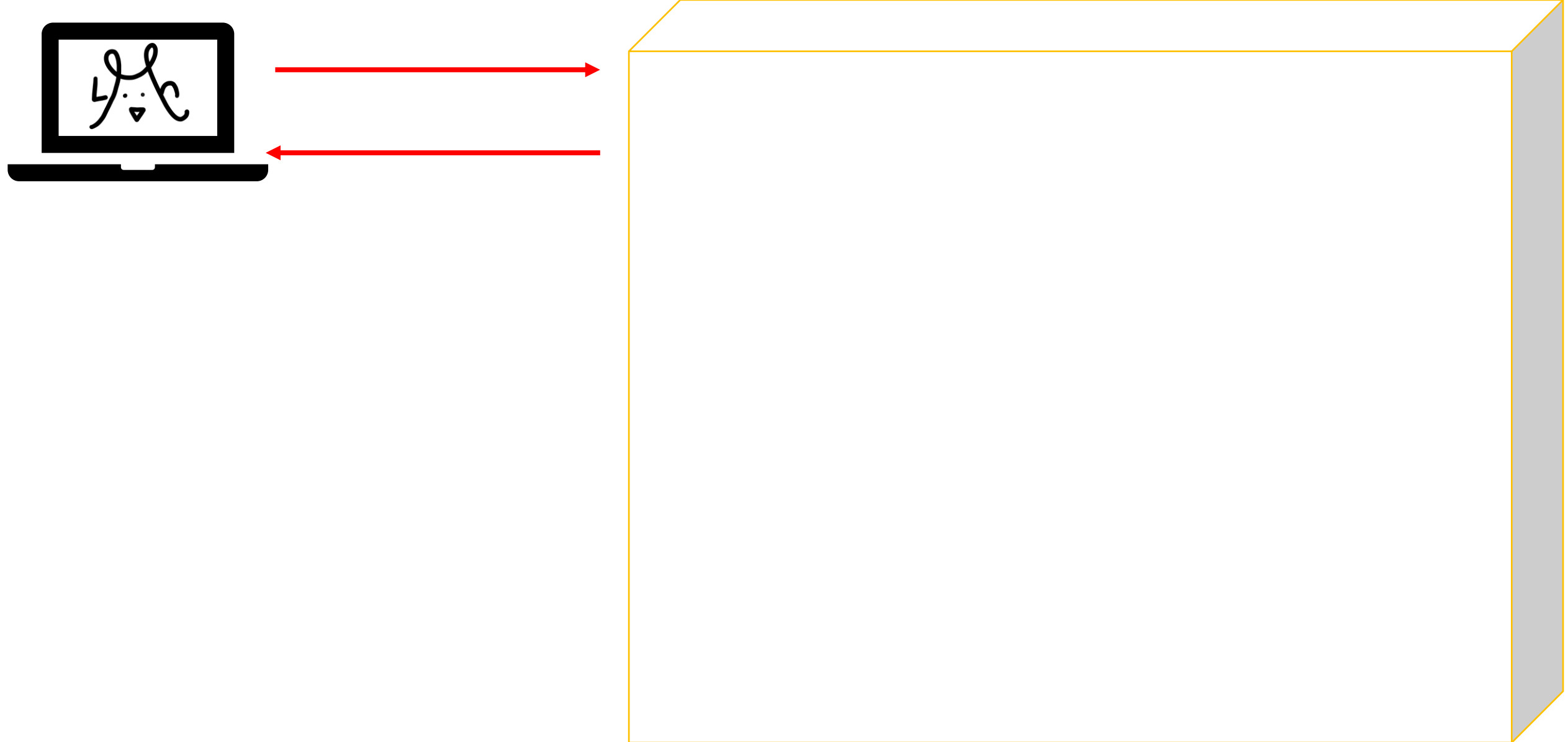
삼성 청년 SW 아카데미

Spring Framework

Spring MVC

- MVC pattern
- Spring Web MVC

MVC pattern



✓ Model

- 동작을 수행하는 코드
- 사용자 View에 어떻게 보일지에 대해서 신경x
- 데이터 질의에 대한 정보를 제공하는 기능 및 데이터에 대한 수정을 담당

✓ View

- 사용자가 화면에 무엇을 어떻게 볼 것인지를 결정.
- 사용자 화면에 보이는 부분
- 모델의 정보를 받아와 사용자에게 보여주는 역할 수행
- 자체적으로 모델의 정보를 보관 x

✓ Controller

- 모델과 뷰를 연결하는 역할을 수행.
- 사용자에게 데이터를 가져오고 수정하고 제공함.

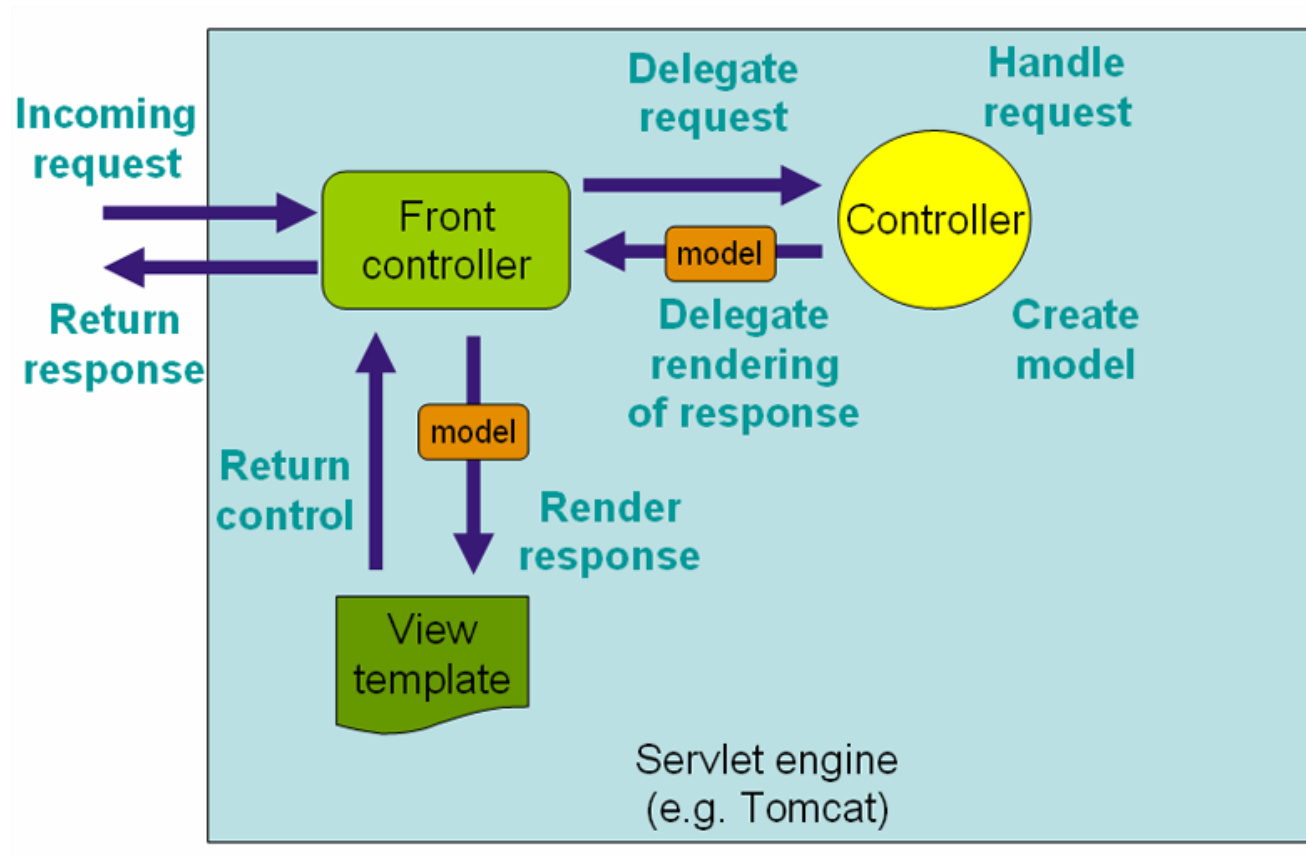
Spring Web MVC

✓ Spring Web MVC

- Servlet API를 기반으로 구축된 웹프레임워크
- 정식 명칭은 Spring Web MVC 이지만, Spring MVC로 주로 알려져 있다
- Spring Framework 이 제공하는 DI, AOP 뿐 아니라, WEB 개발을 위한 기능을 제공
- DispatcherServlet(FrontController)를 중심으로 디자인 되었으며, View Resolver, Handler Mapping, Controller 와 같은 객체와 함께 요청을 처리하도록 구성되어 있다.

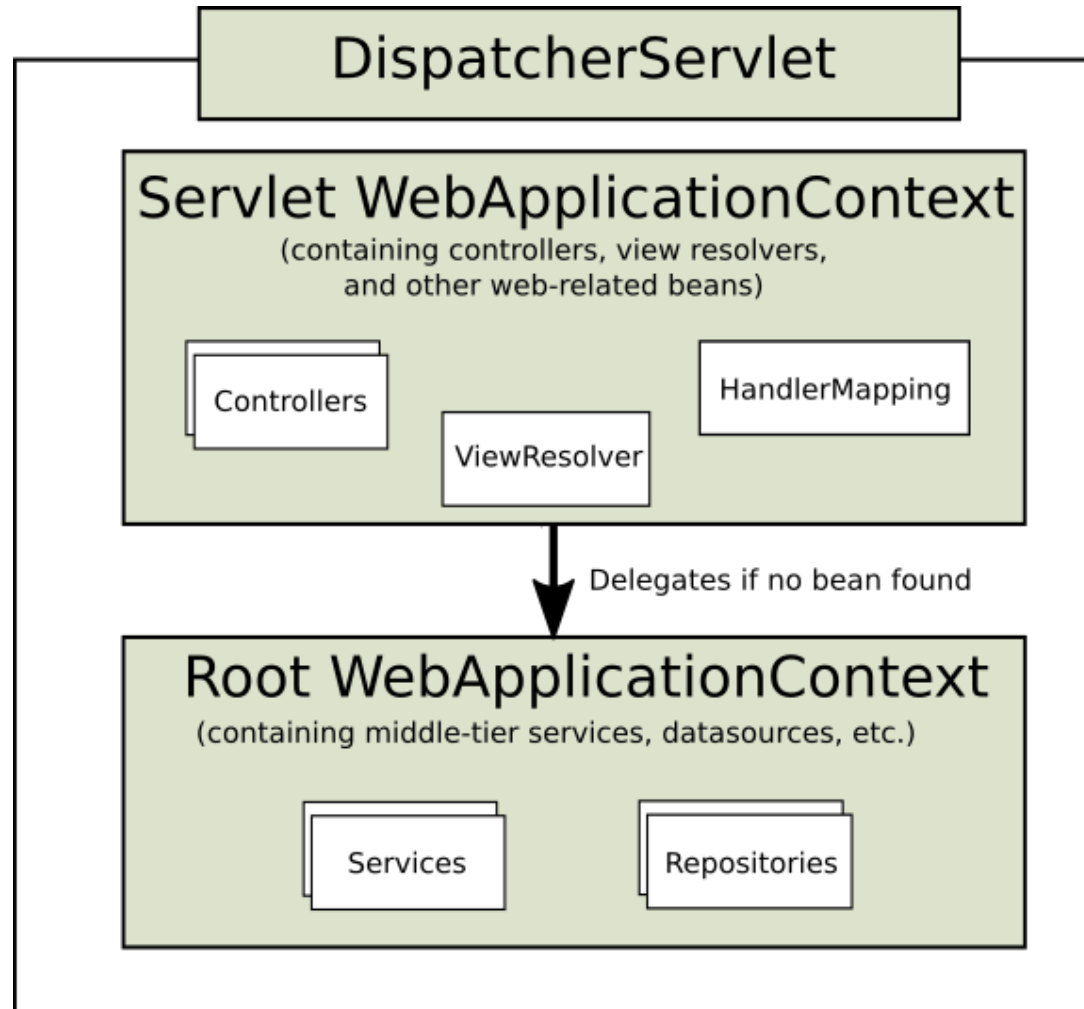
✓ Spring Web MVC

- 다른 프레임워크와 마찬가지로 front controller pattern으로 구성됨
- 중심이 되는 DispatcherServlet(front controller)은 요청처리를 위한 기능을 제공



✓ Spring MVC

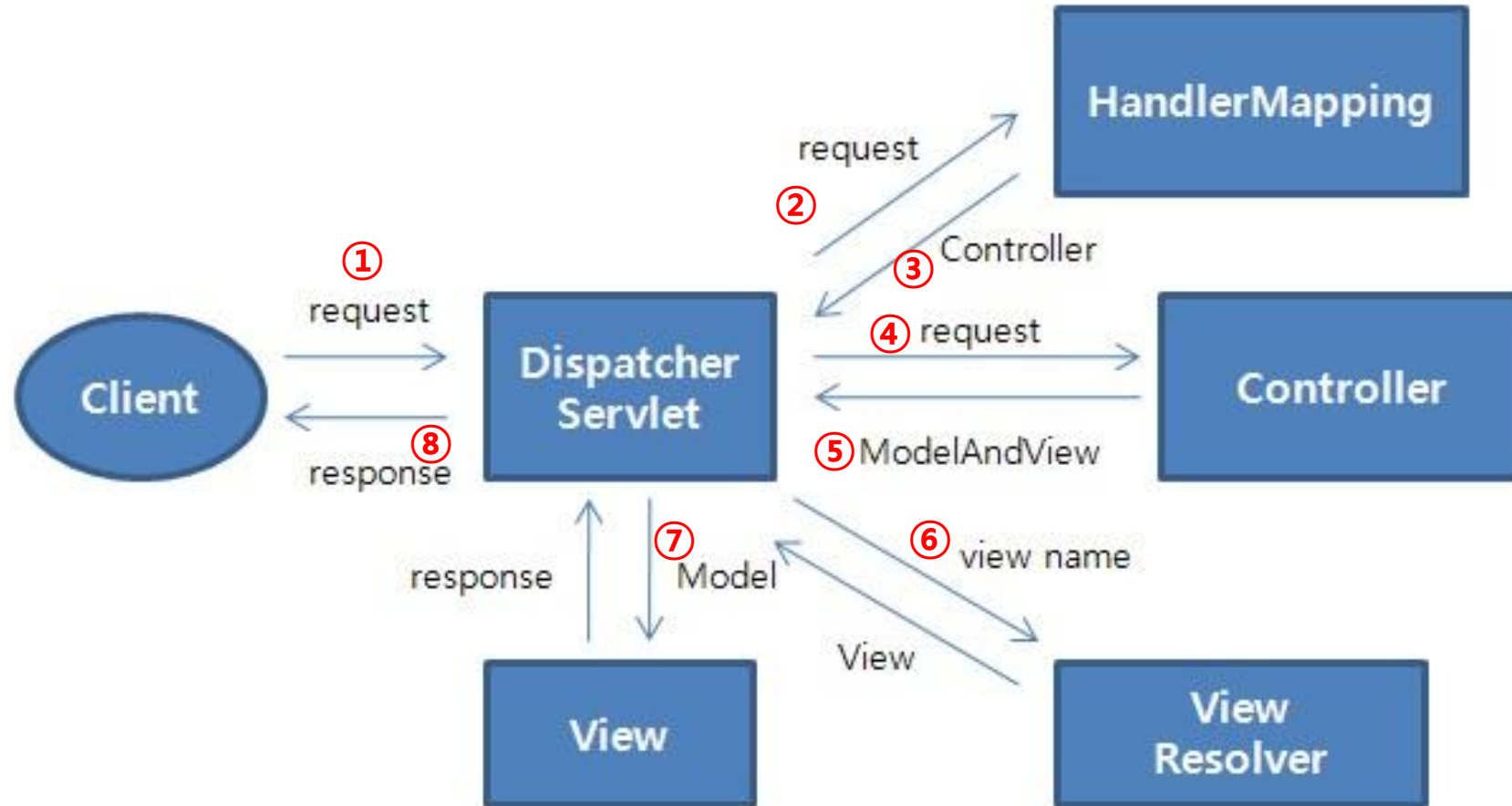
▪ 컨테이너 구성



✓ Spring MVC 구성요소

- DispatcherServlet → 클라이언트 요청처리(요청 및 처리 결과 전달)
- HandlerMapping → 요청을 어떤 Controller가 처리할 지 결정
- Controller → 요청에 따라 수행할 메서드를 선언하고, 요청처리를 위한 로직 수행(비즈니스 로직 호출)
- ModelAndView → 요청처리를 하기 위해서 필요한 혹은 그 결과를 저장하기 위한 객체
- ViewResolver → Controller에 선언된 view이름을 기반으로 결과를 반환할 View를 결정
- View → 응답화면 생성

✓ Spring MVC - 요청 처리 흐름



✓ Spring MVC - 요청 처리 흐름

1. 클라이언트 요청이 들어오면 DispatcherServlet이 받는다.
2. HandlerMapping이 어떤 Controller가 요청을 처리할지 결정한다.
3. DispatcherServlet은 Controller에 요청을 전달
4. Controller는 요청을 처리한다.
5. 결과(요청처리를 위한 data, 결과를 보여줄 view의 이름)를 ModelAndView에 담아 반환
6. ViewResolver에 의해서 실제 결과를 처리할 View를 결정하고 반환
7. 결과를 처리할 View에 ModelAndView를 전달
8. DispatcherServlet은 View가 만들어낸 결과를 응답

✓ Spring Web MVC 구성하기 -1

- DispatcherServlet 생성
- web.xml (pjt/src/main/webapp/WEB-INF/web.xml)

```
<!-- Processes application requests -->
<servlet>
  <servlet-name>appServlet</servlet-name>
  <servlet-class>org.springframework.web.servlet.DispatcherServlet</servlet-class>
  <init-param>
    <param-name>contextConfigLocation</param-name>
    <param-value>/WEB-INF/spring/appServlet/servlet-context.xml</param-value>
  </init-param>
  <load-on-startup>1</load-on-startup>
</servlet>
```

지정하지 않을 경우 WEB-INF/<servlet-name>-servlet.xml 파일을 로딩한다.

✓ Spring Web MVC 구성하기 - 2

- root WebApplicationContext

여러 Servlet 에서 공유해야 하는 DAO, Service 등의 bean을 포함한다.

```
<!-- The definition of the Root Spring Container shared by all Servlets and Filters -->
<context-param>
  <param-name>contextConfigLocation</param-name>
  <param-value>/WEB-INF/spring/root-context.xml</param-value>
</context-param>

<!-- Creates the Spring Container shared by all Servlets and Filters -->
<listener>
  <listener-class>org.springframework.web.context.ContextLoaderListener</listener-class>
</listener>
```

지정하지 않을 경우 /WEB-INF/applicationContext.xml파일을 로딩한다.

✓ Spring Web MVC 구성하기 - 3

- servlet-context.xml (/WEB-INF/spring/appServlet/servlet-context.xml)
- MVC 구성요소 bean 등록 (handler mapping, view resolver, controller)

```
<!-- Resolves views selected for rendering by @Controllers to .jsp resources in the /WE  
<beans:bean class="org.springframework.web.servlet.view.InternalResourceViewResolver">  
  <beans:property name="prefix" value="/WEB-INF/views/" />  
  <beans:property name="suffix" value=".jsp" />  
</beans:bean>  
  
<context:component-scan base-package="com.ssafy.pjt02" />
```

Annotation 방식으로 Controller bean을
등록하기 위해서 component-scan 설정

View Resolver bean 설정

* DefaultAnnotationHandlerMapping 을 기본으로 사용하므로 별도 등록없이 사용가능 (spring 3.0)

✓ Spring Web MVC 구성하기 - 4

- Controller class (main/java/com/ssafy/pjt02/HomeController.java)

```
① @Controller
public class HomeController {
    ② private static final Logger logger = LoggerFactory.getLogger(HomeController.class);
    ③ @RequestMapping(value = "/", method = RequestMethod.GET)
    public String home(Locale locale, Model model) {

        logger.info("Welcome home! The client locale is {}.", locale);

        ④ model.addAttribute("message", "Hello! Spring!!");
        return "home";
    }
}
```

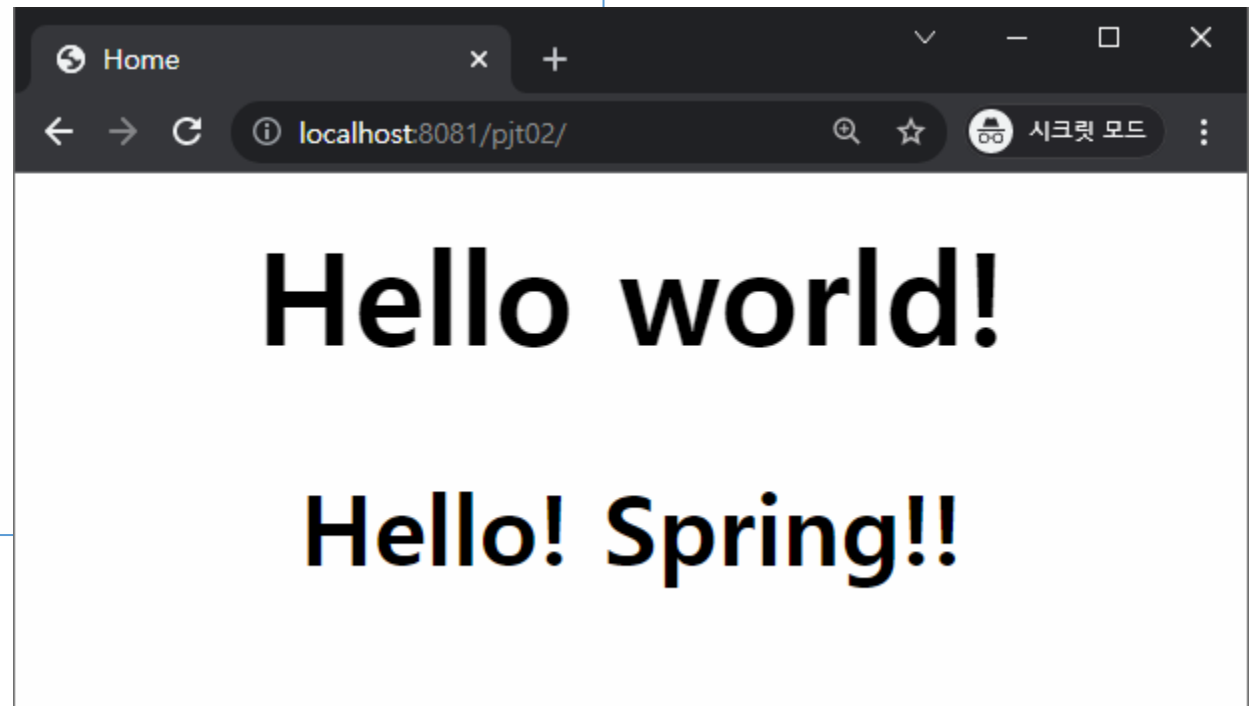
View Resolver 설정을 통해 /WEB-INF/views/home.jsp를 참조

요청 처리에 필요한 Model 객체 선언

✓ Spring Web MVC 구성하기 - 5

- View (WEB-INF/views/home.jsp)

```
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>
<%@ page session="false" %>
<html>
<head>
    <title>Home</title>
</head>
<body>
    <div align ="center">
        <h1>
            Hello world!
        </h1>
        <h2>${ message }</h2>
    </div>
</body>
</html>
```



✓ Spring Web MVC - Controller

- @RequestMapping
- URL을 클래스 또는 특정 핸들러(메서드)에 매핑
- 일반적으로 클래스에 작성하는 @RequestMapping은 요청 경로, 혹은 요청 패턴에 매칭
- 메서드 Annotation은 요청 방식(GET, POST) 등으로 범위를 좁혀 준다.

HelloController.java

```
@Controller
@RequestMapping("/hello")
public class HomeController {
    @RequestMapping(value = "/hello", method = RequestMethod.GET)
    public String handler1() {
        return "hello1";
    }
    @RequestMapping(value = "/hello", method = RequestMethod.POST)
    public String handler2() {
        return "hello2";
    }
}
```

✓ Spring Web MVC - Controller

▪ Controller Handler Method argument - 1

파라미터 타입	설명
HttpServletRequest HttpServletResponse HttpSession	Servlet API를 사용할 수 있다
Locale	요청 클라이언트의 Locale 정보를 포함
InputStream, Reader OutputStream, Writer	요청으로 부터 직접 데이터를 읽어오거나, 응답을 직접 생성하기 위해서 사용
Map, Model, ModelMap	View 데이터를 전달하기 위해서 사용
RedirectAttributes	리디렉션(쿼리 문자열에 추가) 시 사용할 속성 지정
Errors, BindingResult	에러와 데이터 바인딩 결과에 접근하기 위해서 사용

✓ Spring Web MVC - Controller

▪ Controller Handler Method argument - 2

파라미터 타입	설명
@PathVariable	URI 템플릿 변수에 대한 액세스
@RequestParam	multipart 파일을 포함하여 요청 파라미터에 액세스
@RequestHeader	요청 헤더에 액세스
@CookieValue	쿠키에 대한 액세스
@ModelAttribute @SessionAttribute	모든 세션 속성에 대한 액세스 요청 속성에 액세스
@ModelAttribute	모델의 속성에 액세스

✓ Spring Web MVC - Controller

▪ Controller Handler Method argument examples - 1

HomeController.java

```
@RequestMapping("/test1")
public String handler3(HttpServletRequest request, Locale locale, Model model)
{
    model.addAttribute("locale_name", locale.getDisplayName());
    model.addAttribute("url", request.getRequestURL());
    return "test1";
}
```

test1.jsp

```
<body>
    <h1>locale : ${ locale_name } </h1>
    <h1>url : ${ url } </h1>
</body>
```

Insert title here x +
localhost:8081/pjt02/hello/test1

locale : 한국어 (대한민국)

url : http://localhost:8081/pjt02/hello/test1

✓ Spring Web MVC - Controller

▪ Controller Handler Method argument examples - 2

HomeController.java

```
@RequestMapping("/test2/{id}")
public String handler4(@PathVariable int id, @RequestParam String name, Model model)
{
    model.addAttribute("path_var", id);
    model.addAttribute("name", name);
    return "test2";
}
```

test2.jsp

```
<body>
    <h1>path_var : ${ path_var } </h1>
    <h1>name : ${ name } </h1>
</body>
```

http://localhost:8081/pjt02/hello/test2/12?name=홍길동

path_var : 12

name : 홍길동

✓ Spring Web MVC - Controller

▪ Controller Handler Method argument examples - 3

```
@RequestMapping("/test3")
public String handler4_1(@RequestParam(value = "name", required = true) String name,
                        @RequestParam(value = "id", defaultValue = "1") int id,
                        Model model) {
    model.addAttribute("id", id);
    model.addAttribute("name", name);
    return "test3";
}
```

HomeController.java

required = true 설정을 해주면 요청 파라미터에 해당 속성이 없을 경우 400에러 (잘못된 요청) 발생

defaultValue 설정으로 요청 파라미터에 해당 속성이 없을 경우 기본 값 지정

http://localhost:8081/pjt02/hello/test3?name=홍길동

localhost:8081/pjt02/hello/test3?name=홍길동

id : 1

name : 홍길동

✓ Spring Web MVC - Controller

▪ Controller Handler Method argument examples - 4

```
@RequestMapping("/test3")
public String handler4(@ModelAttribute Board board) {
    return "test3";
}
```

HomeController.java

@ModelAttribute annotation을 인자에 적용하여 요청 파라미터 이름과 모델 속성의 이름이 일치하면 모델 속성을 지정

```
<body>
    <h1>board : ${board} </h1>
</body>
```

test3.jsp

클래스이름 으로 접근

```
public class Board {
    private int id;
    private String title;
    private String content;
}
```

Board.java

Insert title http://localhost:8081/pjt02/hello/test3?title=안녕하세요&content=반갑습니다

← → ↻ ⓘ localhost:8081/pjt02/hello/test3?title=안녕하세요&content=반갑습니다

board : Board [id=0, title=안녕하세요, content=반갑습니다]

✓ Spring Web MVC - Controller

▪ Controller Handler Method argument examples - 5

HomeController.java

```
@ModelAttribute("board")
public Board getBoardModel(Board board) {
    return board;
}

@RequestMapping("/test4")
public String handler5(Model model) {
    model.addAttribute("message", "반갑습니다");
    return "test4";
}
```

test4.jsp

```
<body>
    <h1>board : ${ board } </h1>
    <h1>message : ${ message } </h1>
</body>
```

@RequestMapping을 사용하지 않고
@ModelAttribute 를 적용

Model을 받지 않았지만 board 모델
data를 사용할 수 있다.

← → ↻ ⓘ localhost:8081/pjt02/hello/test4?title=안녕하세요&content=반갑습니다

board : Board [id=0, title=안녕하세요, content=반갑습니다]

message : 반갑습니다

✓ Spring Web MVC - Controller

▪ Controller Handler return value

파라미터 타입	설명
@ResponseBody	HttpMessageConverter 구현을 통해 변환되어 응답한다.
HttpHeaders	헤더가 있고 body가 없는 response를 반환
String	뷰 이름 반환 (ViewResolver 선언과 함께 사용)
View	렌더링 하는데 사용할 View 인스턴스
Map, Model	명시적으로 모델을 작성하지 않은 경우 사용
ModelAndView	사용할 view와 속성을 포함하는 객체
void	method에 ServletResponse , HttpServletResponse 인자가 있는 경우, 모든 요청이 처리된 것으로 간주, 그렇지 않으면 요청 URI를 view name으로 처리

✓ Spring Web MVC - Controller

- Controller Handler return value - view 이름지정
- ModelAndView, String

```
@RequestMapping("/test6")
public ModelAndView handler6() {
    //ModelAndView mav = new ModelAndView("test6");
    ModelAndView mav = new ModelAndView();
    mav.setViewName("test6");
    return mav;
}
```

ModelAndView와 String을 이용해 지정한 view name은 ViewResolver에 의해서 실제 물리 경로와 매핑된다.
/WEB-INF/views/test6.jsp
/WEB-INF/views/test7.jsp

```
@RequestMapping("/test7")
public String handler7() {
    return "test7";
}
```

```
<beans:bean class="org.springframework.web.servlet.view.Interr
    <beans:property name="prefix" value="/WEB-INF/views/" />
    <beans:property name="suffix" value=".jsp" />
</beans:bean>
```

✓ Spring Web MVC - Controller

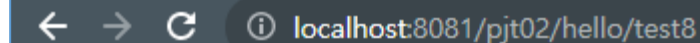
- Controller Handler return value - view 이름 암시적 지정
- 암시적으로 view가 결정되는 경우 (RequestMappingTranslator 에 의해 결정됨)
 - Map, Model 일 경우
 - return type void 이면서 HttpServletResponse, HttpServletResponse를 인자로 받지 않음

```
@RequestMapping("/test8")  
public void handler8(Model model) {  
    model.addAttribute("message", "test8");  
}
```



```
▼ views  
  ▼ hello  
    test8.jsp
```

http://localhost:8081/pjt02/hello/test8



← → ↻ ⓘ localhost:8081/pjt02/hello/test8

message : test8

✓ Spring Web MVC - Controller

- Controller Handler return value - redirect
- redirect을 위해서 사용함
- view 이름에 “**redirect:**” 접두어를 붙이면 redirect 된다.

```
@RequestMapping("/test9")  
public String handler9(Model model) {  
    model.addAttribute("message", "test8");  
    return "redirect:test8";  
}
```

http://localhost:8081/pjt02/hello/test9

요청URL

http://localhost:8081/pjt02/hello/test8

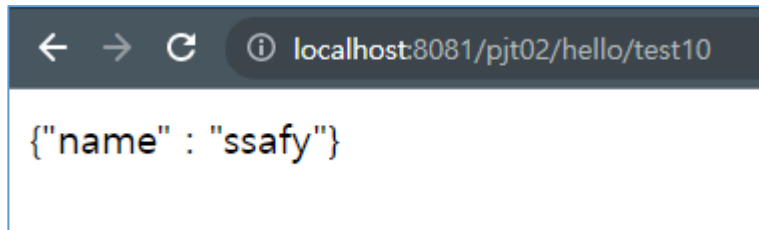
redirect url

✓ Spring Web MVC - Controller

- Controller Handler return value - @ResponseBody
- View 렌더링이 아니라, 데이터(XML, JSON 등)를 응답하기 위해서 사용
- @RestController 와 함께 사용된다. (추후 학습)

```
@RequestMapping("/test10")
@ResponseBody
public String handler10() {
    return "{\"name\" : \"ssafy\"}";
}
```

view가 아닌 응답할 데이터를 작성한다.(예) JSON



다음 방송에서 만나요!

삼성 청년 SW 아카데미