

스프링 MVC

- 기본 구조 -

스프링 MVC의 주요 구성 요소 및 처리 흐름

❖ MVC 패턴

○ 모델-뷰-컨트롤러 패턴

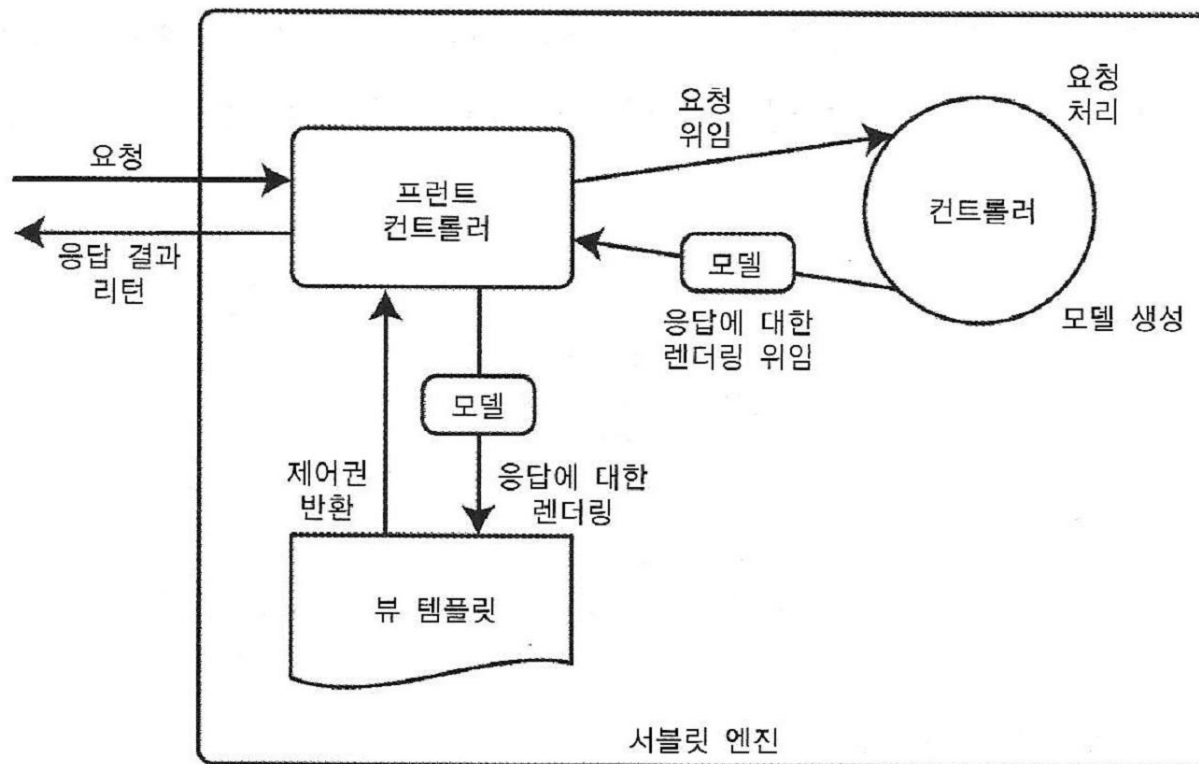
- 프리젠테이션 레이어를 여러 컴포넌트로 분리해 각 컴포넌트가 특정 기능을 담당
 - 뷰는 모델을 사용해 렌더링하고, 사용자의 액션을 보고 컨트롤러에 이를 전달
 - 컨트롤러는 모델을 업데이트
 - 모델은 뷰에게 렌더링을 요청
- 관심사의 분리

컴포넌트	설명
모델	뷰가 렌더링하는 데 필요한 데이터이다. 예를 들어 사용자가 요청한 책 목록이나 주문 내역이 이에 해당한다.
뷰	웹 어플리케이션에서 뷰는 실제로 보이는 부분이며, 모델을 사용해 렌더링한다. 뷰는 JSP나 JSF, PDF, XML 등으로 웹 페이지를 표현한다.
컨트롤러	폼 전송이나 링크 클릭 같은 사용자의 액션에 응답하는 컴포넌트다. 모델을 업데이트하고(주문을 처리하는 서비스 메서드 호출 등) 다른 액션을 수행한다.

스프링 MVC의 주요 구성 요소 및 처리 흐름

❖ MVC 패턴

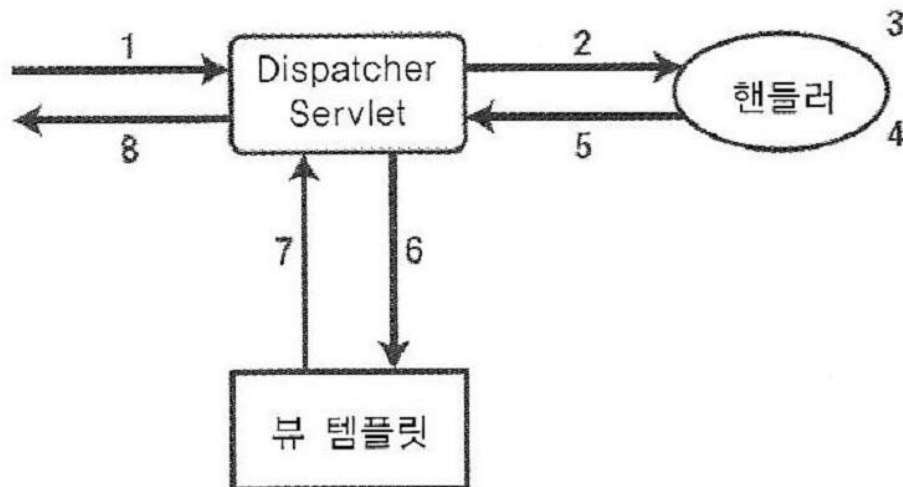
- 웹 어플리케이션을 위한 MVC 패턴(모델2)
 - 프론트 컨트롤러를 두어 서버로 들어오는 요청을 다른 컴포넌트로 전달
 - `javax.servlet.Servlet`으로 구현
 - 요청을 전달받은 컨트롤러는 해당 요청을 처리하기 위해 모델을 만들고 뷰를 선택



스프링 MVC의 주요 구성 요소 및 처리 흐름

❖ DispatcherServlet

- `org.springframework.web.servlet.DispatcherServlet`
- 프론트 컨트롤러 역할 수행
 - 사용자로부터 요청을 받아 이를 처리할 핸들러에 넘김
 - 핸들러가 처리한 결과를 받아 사용자에게 응답 결과를 보여줌



1. 요청
2. 요청을 핸들러에게 전달
3. 요청 처리
4. 모델 준비와 뷰 선택
5. `org.springframework.web.servlet.ModelAndView` 리턴
6. 모델을 사용해 `org.springframework.web.servlet.View` 렌더링
7. 서블릿에 제어권 넘김
8. 클라이언트에 응답 결과 리턴

스프링 MVC의 주요 구성 요소 및 처리 흐름

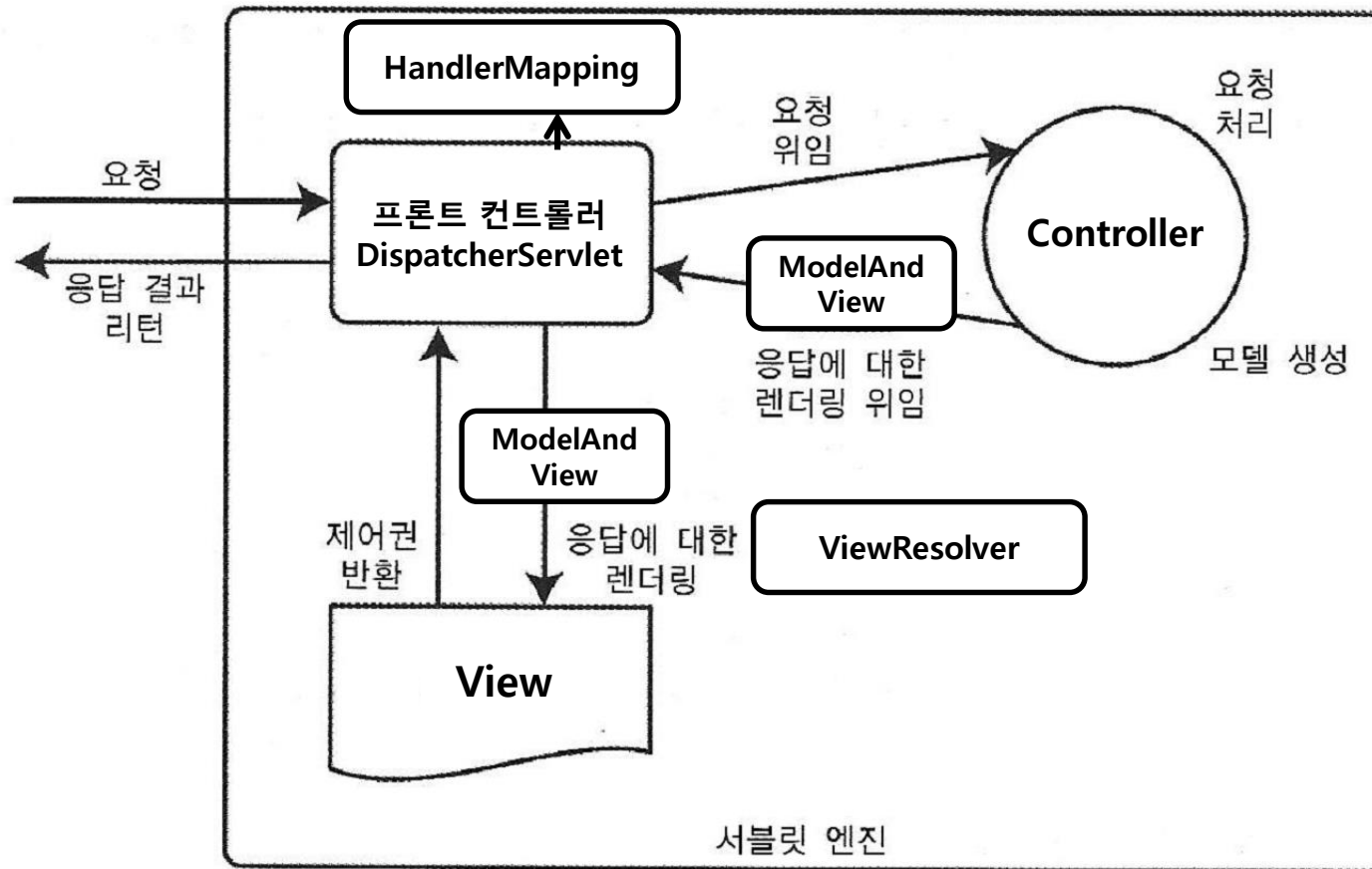
❖ 스프링 MVC의 주요 구성요소

- 핵심은 DispatcherServlet

구성요소	설명
DispatcherServlet	클라이언트의 요청을 전달 받음. 컨트롤러에게 클라이언트의 요청을 전달하고, 컨트롤러가 리턴한 결과값을 View에 전달하여 알맞은 응답을 생성하도록 함
HandlerMapping	클라이언트의 요청 URL을 어떤 컨트롤러가 처리할지 결정
컨트롤러	클라이언트의 요청을 처리한 뒤, 그 결과를 DispatcherServlet에 알려줌
ModelAndView	컨트롤러가 처리한 결과 정보 및 뷰 선택에 필요한 정보를 담음
ViewResolver	컨트롤러의 처리 결과를 생성할 뷰를 결정
뷰(View)	컨트롤러의 처리 결과 화면을 생성 JSP나 Velocity 템플릿 파일 등.

스프링 MVC의 주요 구성 요소 및 처리 흐름

❖ 스프링 MVC의 클라이언트 요청 처리 과정



스프링 MVC의 주요 구성 요소 및 처리 흐름

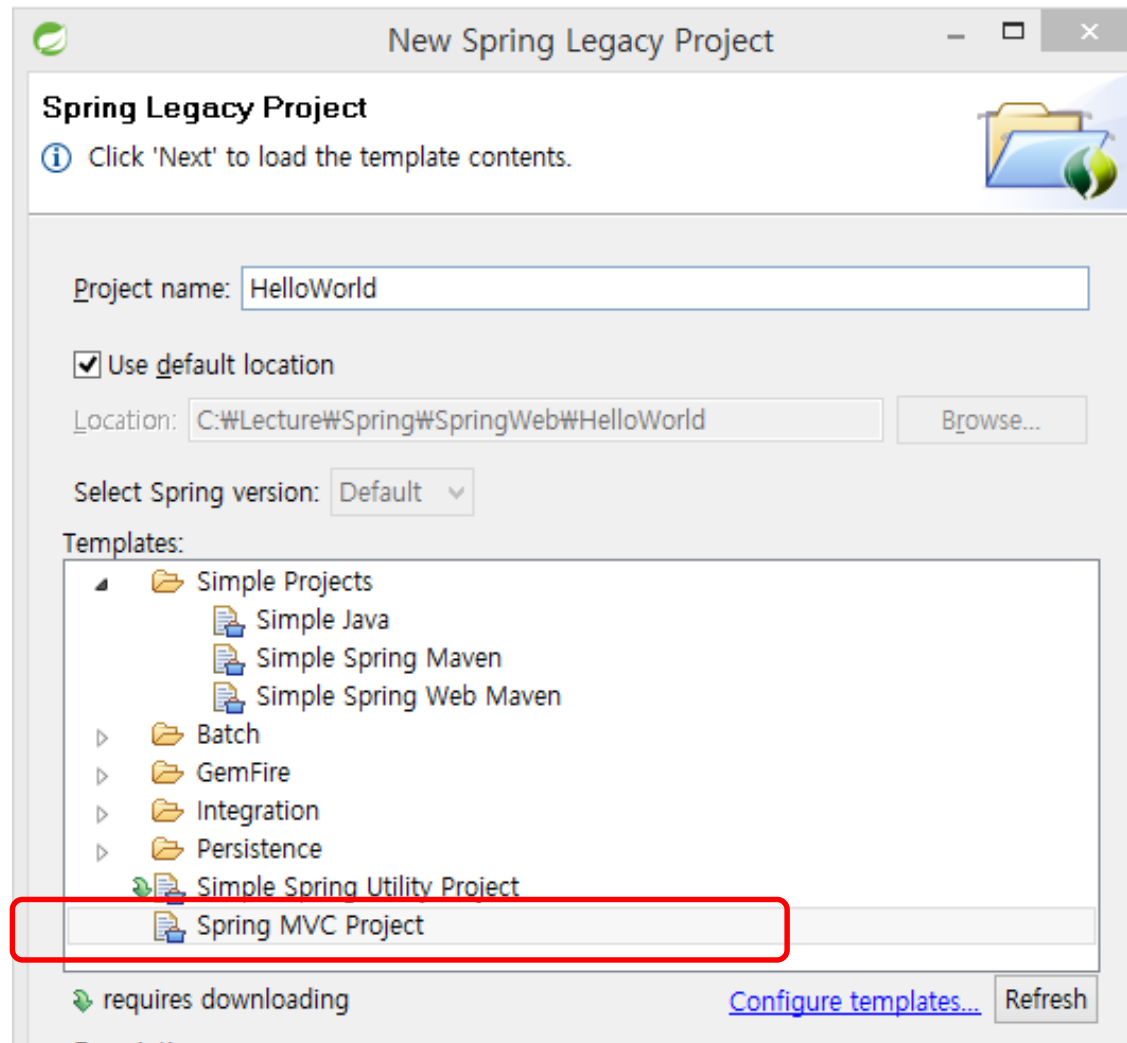
❖ 스프링 MVC의 클라이언트 요청 처리 과정

- 클라이언트의 요청이 DispatcherServlet에 전달
- DispatcherServlet은 HandlerMapping을 사용하여 클라이언트 요청을 처리할 컨트롤러 객체를 구함
- DispatcherServlet은 컨트롤러 객체를 이용해서 클라이언트의 요청을 처리
- 컨트롤러는 클라이언트의 요청 처리 결과 정보를 담은 ModelAndView 객체를 리턴
- DispatcherServlet은 ViewResolver로부터 응답 결과를 생성할 뷰 객체를 구함
- 뷰는 클라이언트에 전송할 응답을 생성
- 개발자 담당 부분
 - 컨트롤러
 - 응답 결과 화면을 전송할 JSP

스프링 MVC HelloWorld

❖ 단계1 : 스프링 프로젝트 생성

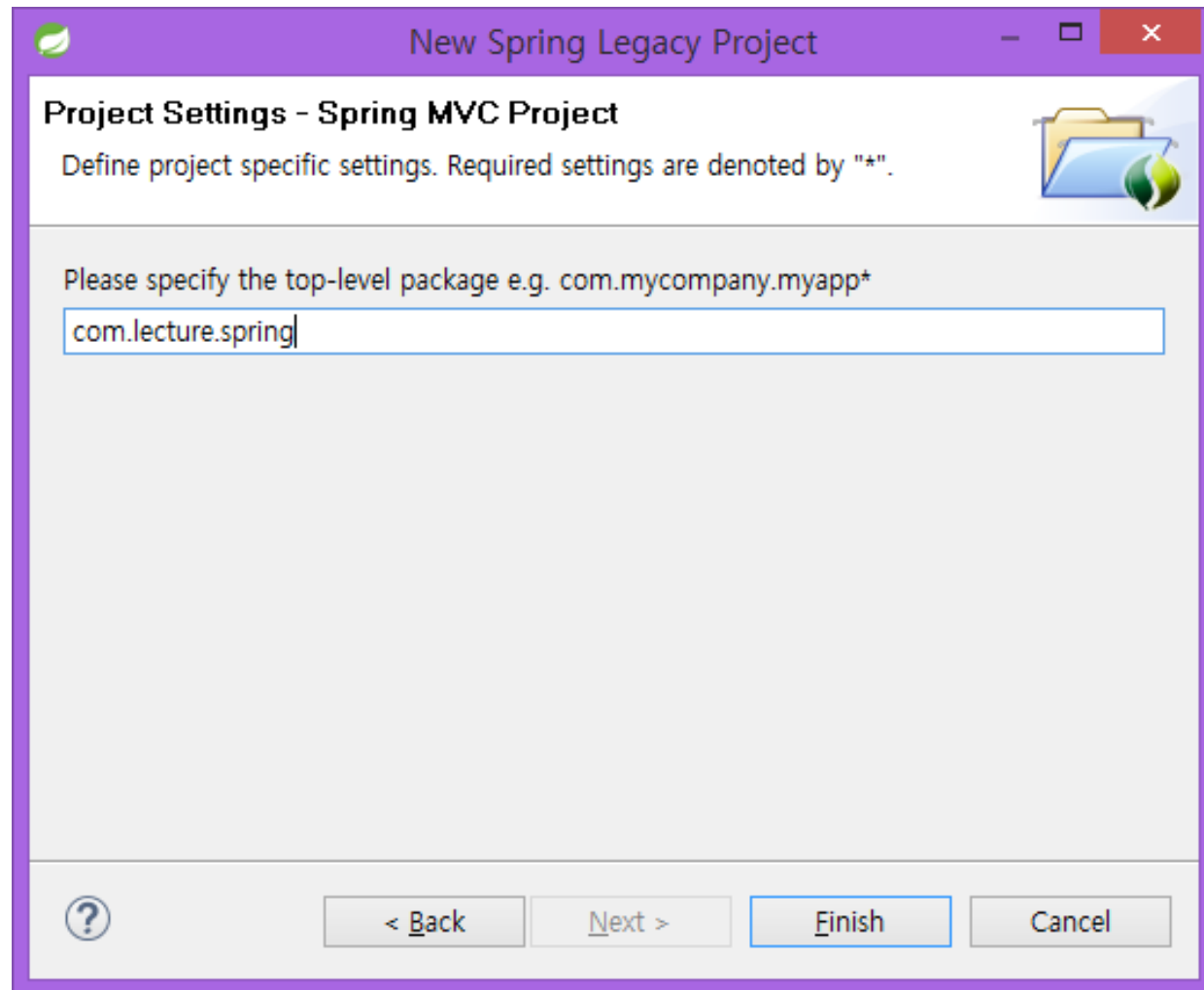
- New > Spring Legacy Project > Spring MVC Project
 - Project name : HelloWorld



스프링 MVC HelloWorld

❖ 단계1 : 스프링 프로젝트 생성

- 최상위 패키지명 설정
 - `com.lecture.spring`



스프링 MVC HelloWorld



❖ 단계1 : 스프링 프로젝트 생성

- 프로젝트 기본 폴더 구성

스프링 MVC HelloWorld

❖ 단계1: DispatcherServlet 설정 및 스프링 컨텍스트 설정

- WEB-INF/web.xml
 - 공통으로 사용할 어플리케이션 컨텍스트 설정

```
<web-app version="3.0"
  xmlns="http://java.sun.com/xml/ns/javaee"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
    http://java.sun.com/xml/ns/javaee/web-app_3_0.xsd"
  metadata-complete="true">

  <!-- 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>
```

스프링 MVC HelloWorld

❖ 단계 1: DispatcherServlet 설정 및 스프링 컨텍스트 설정

- WEB-INF/web.xml
 - 클라이언트의 모든 요청을 DispatcherServlet이 처리(Front Controller)
 - 디폴트 스프링 설정 파일 설정
 - 예) /WEB-INF/[서블릿이름]-servlet.xml → dispatcher-servlet.xml

```
<servlet>
  <servlet-name>dispatcher</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>
```

스프링 설정 파일 직접 지정

```
<servlet-mapping>
  <servlet-name>dispatcher</servlet-name>
  <url-pattern>/</url-pattern>
</servlet-mapping>
</web-app>
```

스프링 MVC HelloWorld

❖ 단계 1: DispatcherServlet 설정 및 스프링 컨텍스트 설정

- /webapp/WEB-INF/spring/root-context.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       xsi:schemaLocation="http://www.springframework.org/schema/beans
                           http://www.springframework.org/schema/beans/spring-beans.xsd">

    <!-- Root Context: defines shared resources visible to all other web
    components -->

</beans>
```

- 여러 스프링 설정의 부모 설정 역할
 - 공통으로 적용할 사항을 기술

스프링 MVC HelloWorld

❖ 단계 1: DispatcherServlet 설정 및 스프링 컨텍스트 설정

- /webapp/WEB-INF/spring/appServlet/servlet-context.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<beans:beans ...>
```

```
<!-- DispatcherServlet Context: defines this servlet's request-
processing infrastructure -->
```

```
<!-- Enables the Spring MVC @Controller programming model -->
```

```
<annotation-driven />
```

스프링 어노테이션 처리

```
<!-- Handles HTTP GET requests for /resources/** by efficiently serving
up static resources in the ${webappRoot}/resources directory -->
```

```
<resources mapping="/resources/**" location="/resources/" />
```

/resources/ 하위 요청에 대한 위치 지정
뷰와 관련 없는 파일 배치

스프링 MVC HelloWorld

❖ 단계 1: DispatcherServlet 설정 및 스프링 컨텍스트 설정

- /webapp/WEB-INF/spring/appServlet/servlet-context.xml

```
<!-- Resolves views selected for rendering by @Controllers to .jsp resources in the /WEB-INF/views directory -->
```

JSP 뷰의 이름을 처리할 모듈
InternalResourceViewResolver

```
<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>
```

어노테이션으로 자동 빈 등록을 위한 스캔 패키지 지정

```
<context:component-scan base-package="com.lecture.spring" />
```

```
</beans:beans>
```

스프링 MVC HelloWorld

❖ 단계2: 컨트롤러 구현 및 설정 추가

- 컨트롤러 클래스 추가
 - New > Class
 - 패키지 : `com.lecture.spring.controller`
 - 클래스명 : `HelloController`

스프링 MVC HelloWorld

❖ 단계2: 컨트롤러 구현 및 설정 추가

- @Controller 어노테이션을 이용한 컨트롤러 구현
- @RequestMapping 어노테이션을 이용해서 클라이언트 요청을 처리할 메서드 지정

@Controller

```
public class HelloController {  
    private static final Logger logger =  
        LoggerFactory.getLogger(HomeController.class);
```

@RequestMapping("/hello")

```
public ModelAndView hello() {  
    logger.info("/hello 요청 처리");  
  
    ModelAndView mav = new ModelAndView();  
    mav.setViewName("test/hello");    // 모델의 이름 설정  
    mav.addObject("greeting", getGreeting());    // 모델의 데이터 설정
```

```
    return mav;  
}
```

```
private String getGreeting() {  
    return "안녕하세요";  
}
```

```
}
```

스프링 MVC HelloWorld

❖ 단계2: 컨트롤러 구현 및 설정 추가

○ Logger

▪ SLF4J 인터페이스 (구현체 : LOG4J)

- info() : 일반 정보 추력
- debug() : 디버그 정보 출력
- warn() : 경고 정보 출력
- error() : 에러 정보 출력

▪ 로그 레벨

- src/main/resources/log4j.xml에서 로그 레벨 및 출력 장치 설정 가능

스프링 MVC HelloWorld

❖ 단계2: 컨트롤러 구현 및 설정 추가

- Logger 설정 : src/main/resources/log4j.xml

```
<?xml version="1.0" encoding="UTF-8"?>
:

<!-- Appenders -->
<appender name="console" class="org.apache.log4j.ConsoleAppender">
  <param name="Target" value="System.out" />
  <layout class="org.apache.log4j.PatternLayout">
    <param name="ConversionPattern" value="%-5p: %c - %m%n" />
  </layout>
</appender>

<!-- Application Loggers -->
<logger name="com.lecture.spring">
  <level value="info" />
</logger>

:

</log4j:configuration>
```

스프링 MVC HelloWorld

❖ 단계3: 설정 파일에 ViewResolver 설정 추가

- ModelAndView 객체에 뷰의 이름 지정

```
public ModelAndView hello() {  
    :  
    ModelAndView mav = new ModelAndView();  
    mav.setViewName("test/hello");        // 모델의 이름 설정  
    mav.addObject("greeting", getGreeting());    // 모델의 데이터 설정  
  
    return mav;  
}
```

- 뷰의 이름 : test/hello
- DispatcherServlet은 뷰의 이름으로 뷰 구현체를 검색
 - ViewResolver가 수행

스프링 MVC HelloWorld

❖ 단계3: 설정 파일에 ViewResolver 설정 추가

- JSP를 뷰 기술로 사용하는 경우 InternalResourceViewResolver 구현체를 빈으로 등록

```
<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>
```

- 뷰 이름이 hello인 경우 JSP 파일명 : prefix + 뷰이름 + suffix
 - - /WEB-INF/views/hello.jsp

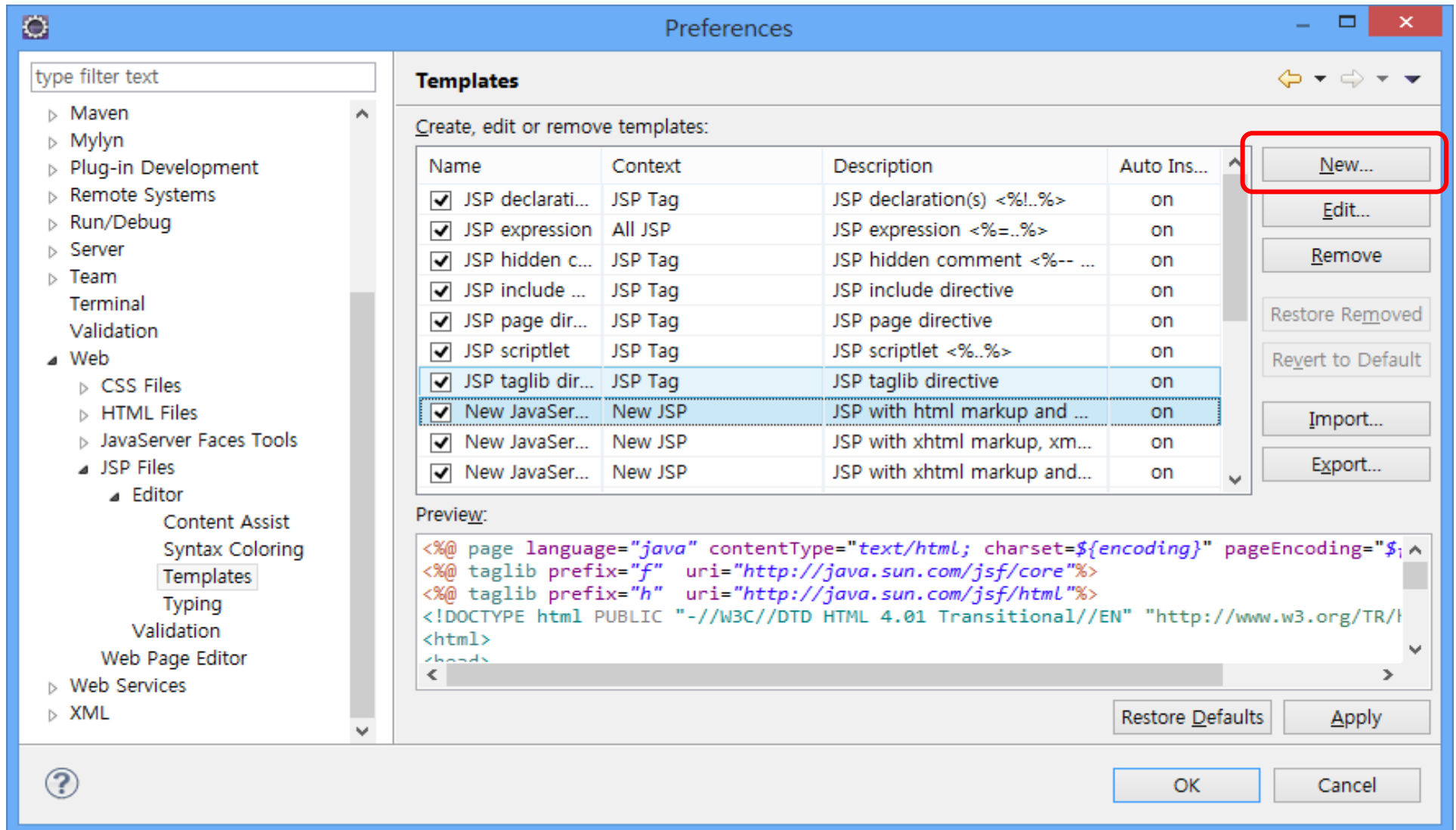
스프링 MVC HelloWorld

❖ JSP 템플릿 작성

- jsp 파일 템플릿에 HTML5 템플릿이 없음
- 기본 HTML5 템플릿 추가
- JQuery, CSS, JS 가 추가된 템플릿 추가

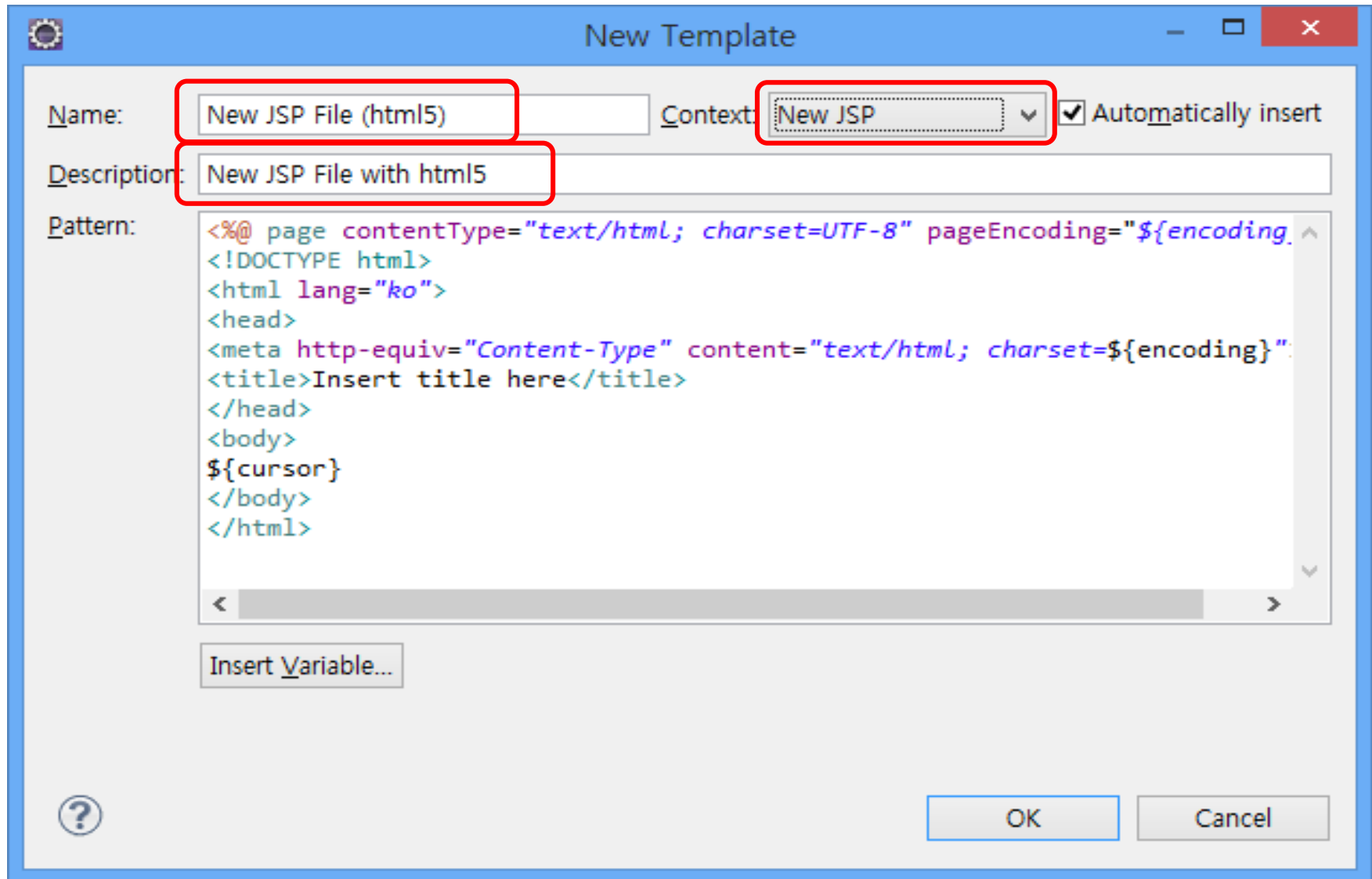
스프링 MVC HelloWorld

❖ Window > Preference > Web > JSP File > Editor > Templates



스프링 MVC HelloWorld

❖ 템플릿 작성



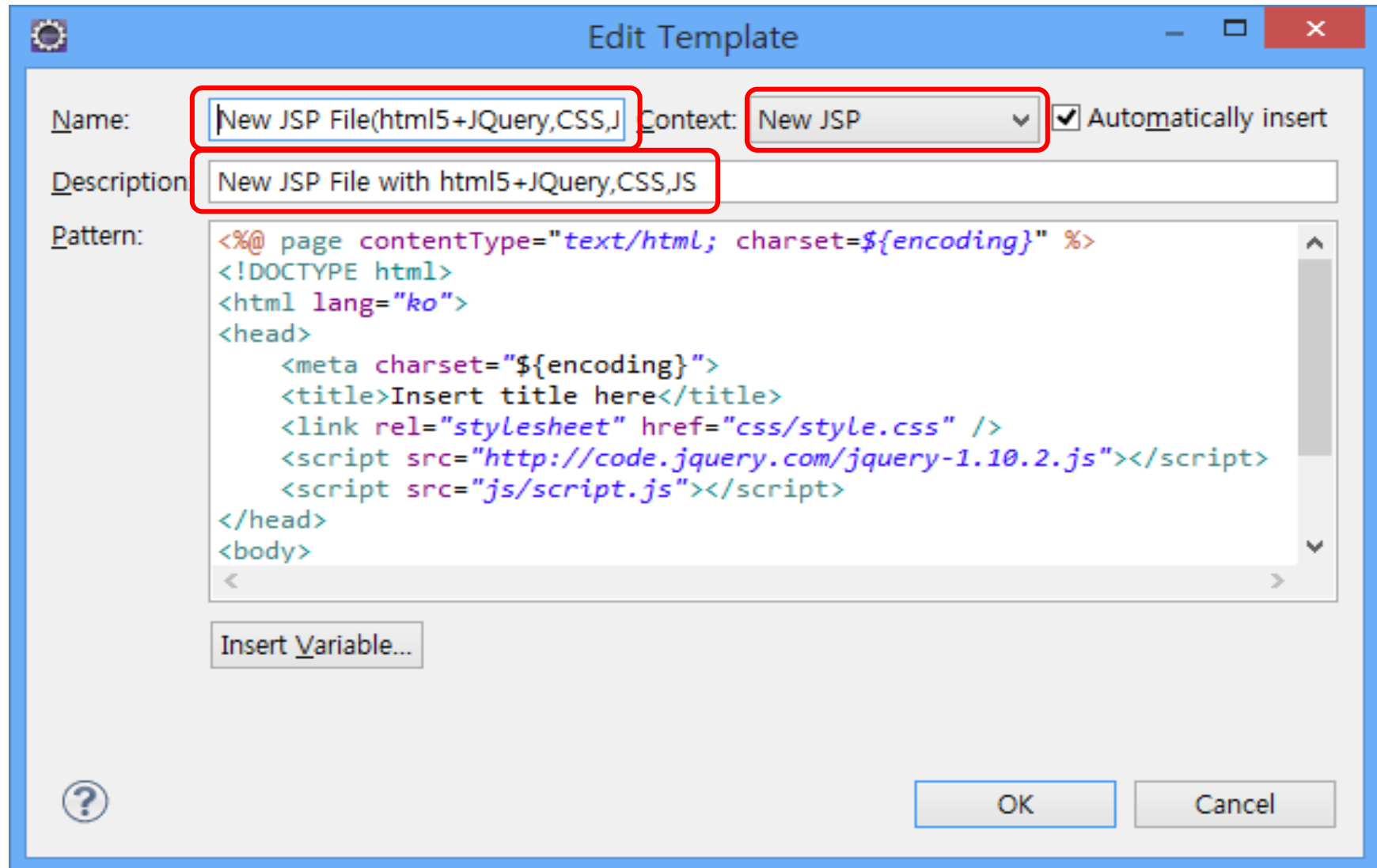
스프링 MVC HelloWorld

❖ JSP with HTML5의 Pattern 작성

```
<%@ page contentType="text/html; charset=${encoding}" %>
<!DOCTYPE html>
<html lang="ko">
<head>
    <meta charset="${encoding}">
    <title>Insert title here</title>
</head>
<body>
    ${cursor}
</body>
</html>
```

스프링 MVC HelloWorld

❖ JQuery, CSS, Js가 추가된 템플릿 추가



스프링 MVC HelloWorld

❖ JQuery, CSS, Js가 추가된 템플릿 추가

```
<%@ page contentType="text/html; charset=${encoding}" %>
<!DOCTYPE html>
<html lang="ko">
<head>
<meta charset="${encoding}">
<title>Insert title here</title>
<link rel="stylesheet" href="css/style.css" />
<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.2.1/jquery.min.js">
</script>
</head>
<body>
${cursor}
</body>
</html>
```

스프링 MVC HelloWorld

❖ 단계 4: 뷰 코드 구현

- webapp/WEB-INF/views/hello.jsp

```
<%@ page language="java" contentType="text/html; charset=UTF-8"%>
<!DOCTYPE html>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; UTF-8">
<title>인사</title>
</head>
<body>
인사말 : ${greeting}
</body>
</html>
```

컨트롤러에서 저장한 모델명

❖ 단계 5: 실행

- localhost/HelloWorld/hello


- 컨트롤러

```
@RequestMapping("/hello.do")
public ModelAndView hello() {
    ModelAndView mav = new ModelAndView();
    mav.setViewName("hello");
    mav.addObject("greeting", getGreeting());

    return mav;
}
```

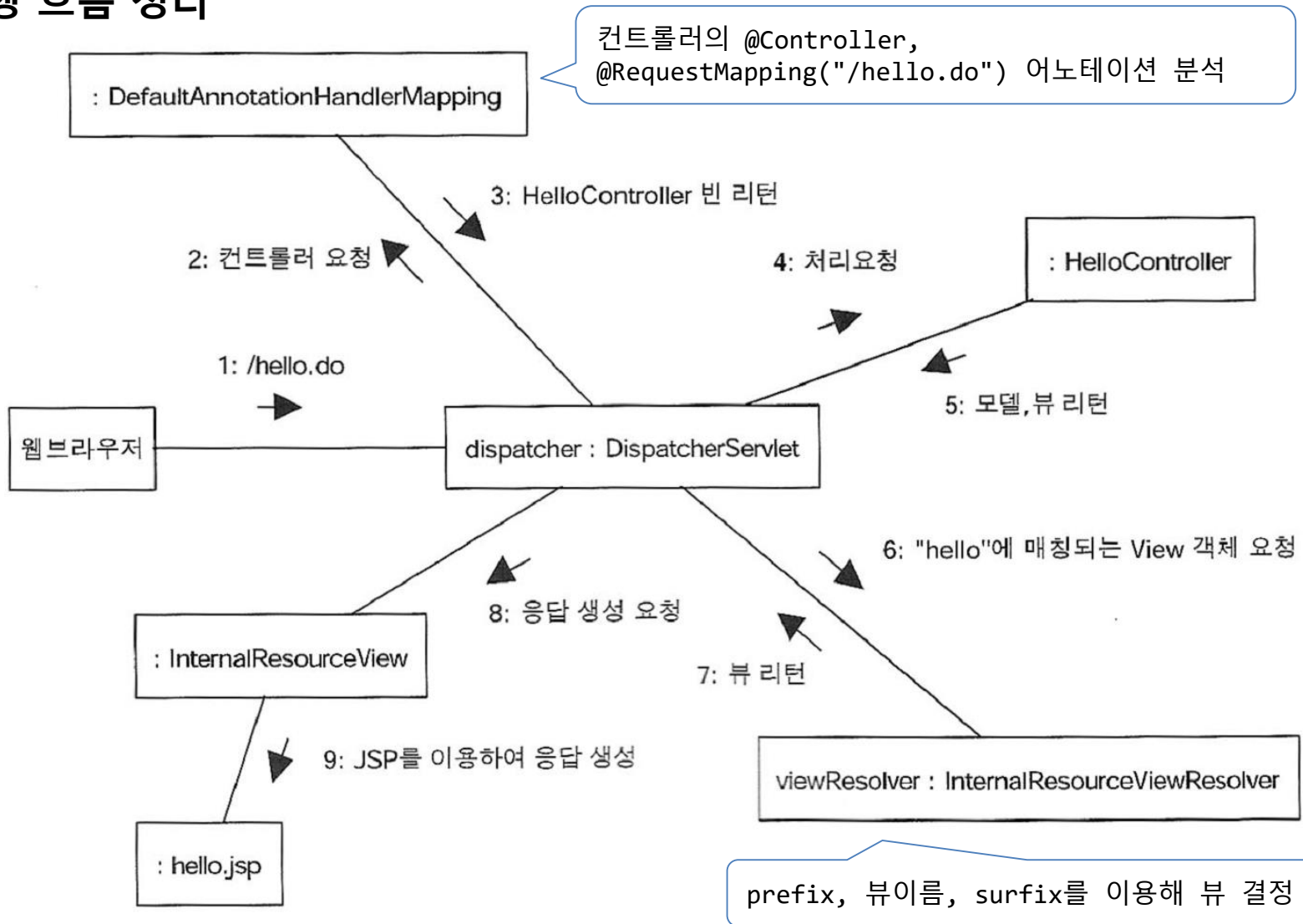
- 뷰 JSP

```
<body>
인사말 : ${greeting}
</body>
```



스프링 MVC HelloWorld

❖ 실행 흐름 정리



DispatcherServlet 설정과 ApplicationContext의 관계

❖ DispatcherServlet 설정

- DispatcherServlet을 위한 스프링 설정 파일
 - 디폴트 : /WEB-INF/[서블릿이름]-servlet.xml
 - 변경시 contextConfigLocation 프로퍼티로 스프링 설정 파일 경로 설정
 - 파일이 여러 개 인경우 콤마, 화이트문자로 구분해서 나열
- web.xml

```
<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>
```

서블릿 이름

서블릿 구현체 클래스

스프링 설정 파일 설정 파라미터 명

스프링 설정 파일 경로

DispatcherServlet 설정과 ApplicationContext의 관계

❖ DispatcherServlet 설정

- 서블릿 설정 파일 복수 지정 가능
 - contextConfigLocation 초기화 파라미터의 값에 지정
- web.xml

```
<servlet>
  ...
  <init-param>
    <param-name>contextConfigLocation</param-name>
    <param-value>
      /WEB-INF/spring/appServlet/database.xml
      /WEB-INF/spring/appServlet/servlet-context.xml
    </param-value>
  </init-param>
  <load-on-startup>1</load-on-startup>
</servlet>
```

DispatcherServlet 설정과 ApplicationContext의 관계

❖ 웹 어플리케이션을 위한 ApplicationContext 설정

- 복수개의 DispatcherServlet을 운영하는 경우 공통 설정
 - ContextLoaderListener를 사용하여 공통으로 사용될 빈을 설정
 - ContextLoaderListener를 ServletListener로 등록
 - contextConfigLocation 파라미터를 이용하여 공통으로 사용될 빈 정보를 담고 있는 설정 파일 목록을 지정

```
<context-param>
  <param-name>contextConfigLocation</param-name>
  <param-value>
    /WEB-INF/spring/root-context.xml
    classpath:config/service.xml
    classpath:persistence.xml
  </param-value>
</context-param>

<listener>
  <listener-class>
    org.springframework.web.context.ContextLoaderListener
  </listener-class>
</listener>
```


캐릭터 인코딩 처리를 위한 필터 설정

❖ 요청 파라미터의 캐릭터 인코딩 설정

- `response.setCharacterEncoding("utf-8");`
 - 모든 컨트롤러에서 실행해야 함
- `web.xml`에서 일괄 적용
 - `encodingFilter` 등록 (POST 전송에 대한 처리)

```
<filter>
  <filter-name>encodingFilter</filter-name>
  <filter-class>
    org.springframework.web.filter.CharacterEncodingFilter
  </filter-class>
  <init-param>
    <param-name>encoding</param-name>
    <param-value>utf-8</param-value>
  </init-param>
</filter>

<filter-mapping>
  <filter-name>encodingFilter</filter-name>
  <url-pattern>/*</url-pattern>
</filter-mapping>
```

외부접속 허용

❖ 이클립스 톱켓 연동

- 디폴트로 해당 컴퓨터에서의 접속만 허용
- server.xml

```
<Connector
  port="8080"
  protocol="HTTP/1.1"
  connectionTimeout="20000"
  redirectPort="8443"
  :
  address="0.0.0.0" />
```