

## Day8; 20220914(Spring MVC)


📅 날짜	@2022년 9월 14일
📅 유형	@2022년 9월 14일
🏷 태그	

spring/src at main · u8yes/spring

Contribute to u8yes/spring development by creating an account on GitHub.

<https://github.com/u8yes/spring/tree/main/src>

u8yes/spring



1 Contributor

0 Issues

1 Star

0 Forks

### Inversion of Controller - DL, DI

DI - 생성자, setter 메서드, @annotation

IoC, DI, AOP 등은 이미 존재하고 있던 수학기론을 Spring에게 적용한 것이지, Spring에서 제공한 것이 아니다.

### 시점은? - Before, After, AfterReturning, AfterThrowing, Around

1. main(){}

Client에게 전달 받은 것을 Server에서 처리하는 것을 Controller.

화면에 보여주는 것을 **View**.

2. Business Logic - Service, DAO, VO, Database 총칭

3. Service는 구현하기 위해 처리하는 곳 = Service Layer, Model

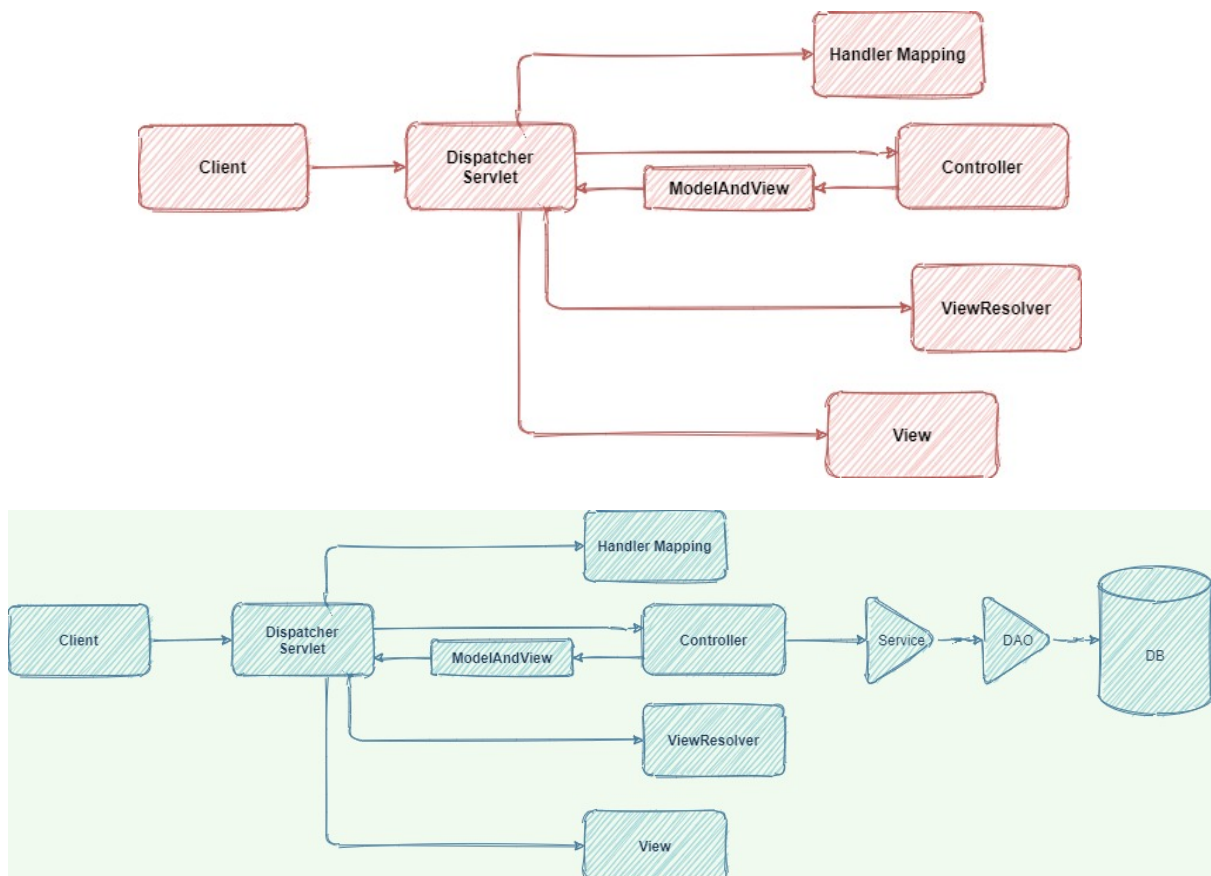
DAO, VO(DTO), Database = Persistence Layer

DAO, VO(DTO) - CRUD

Database

## Spring MVC 흐름

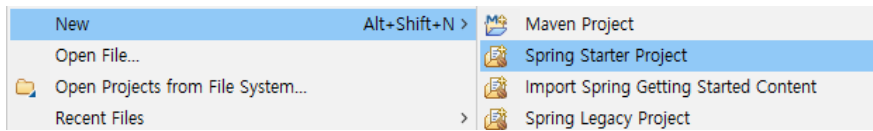
[https://s3-us-west-2.amazonaws.com/secure.notion-static.com/0d9e8890-2e78-433d-8789-33798b4285e6/20220914\\_Day8\\_Spring\\_MVC\\_%ED%9D%90%EB%A6%84.drawio](https://s3-us-west-2.amazonaws.com/secure.notion-static.com/0d9e8890-2e78-433d-8789-33798b4285e6/20220914_Day8_Spring_MVC_%ED%9D%90%EB%A6%84.drawio)



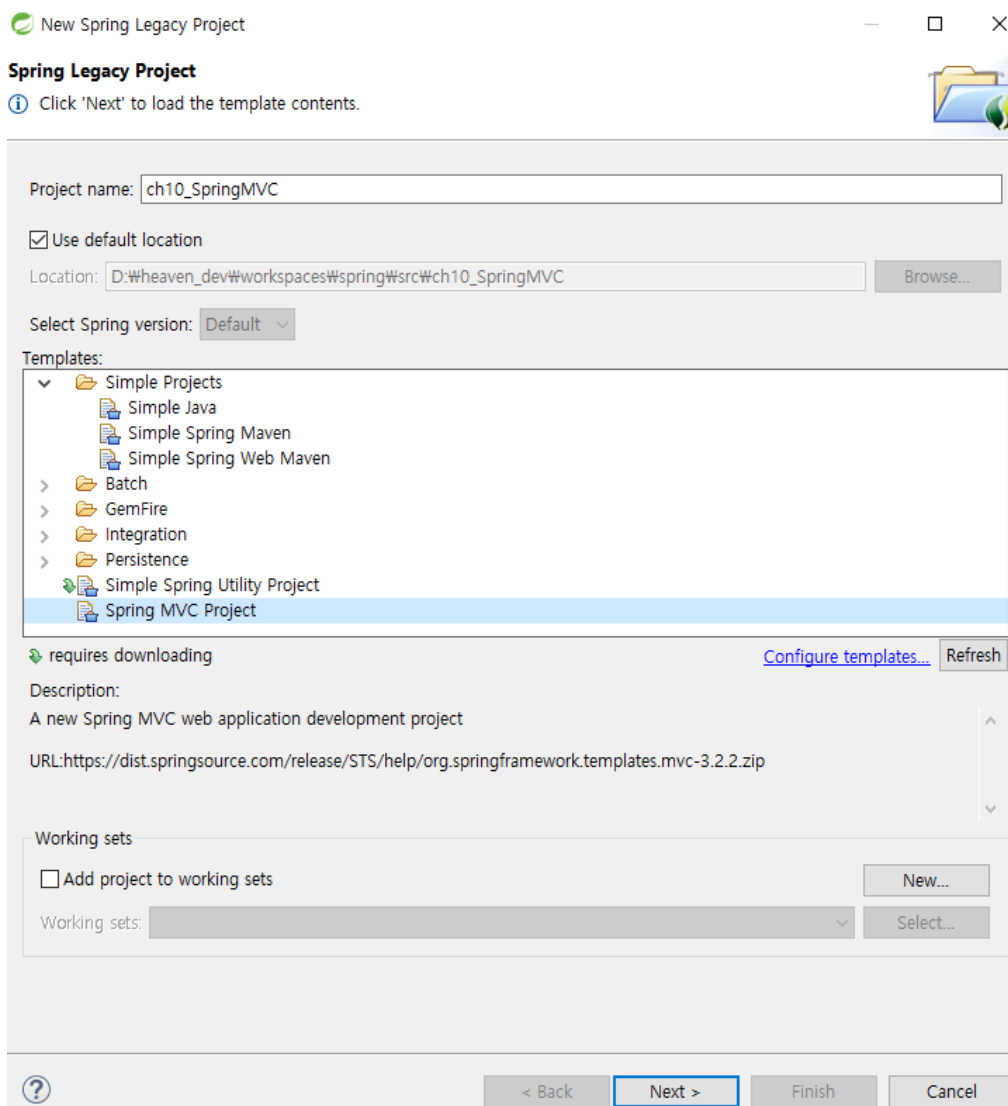
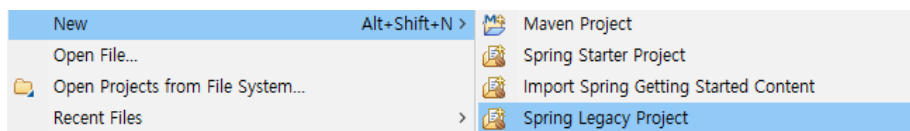
**Spring이 Dispatcher Servlet에서 알아서 다 처리해줌**

## <복습>New

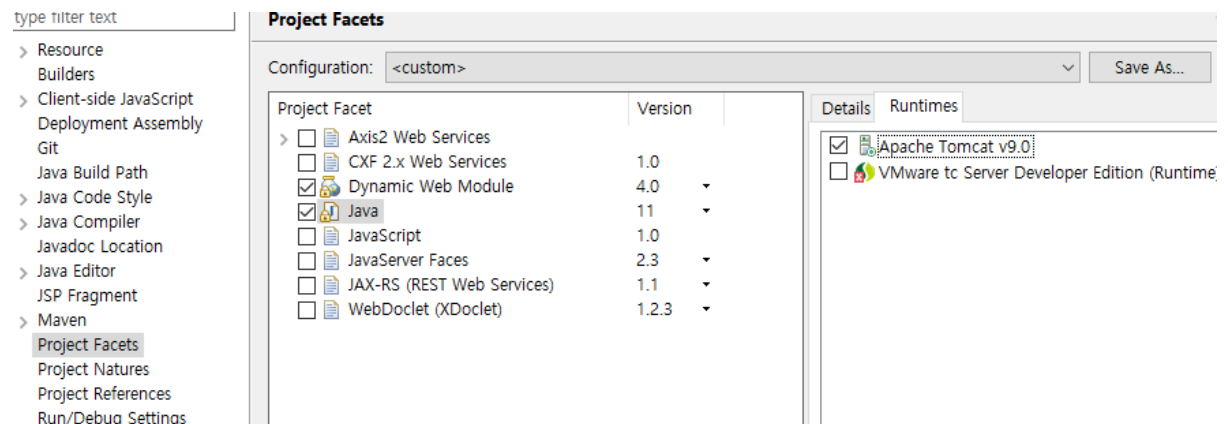
Spring Starter Project(Spring Boot) - 모든 것을 다 갖춘 개발 환경.



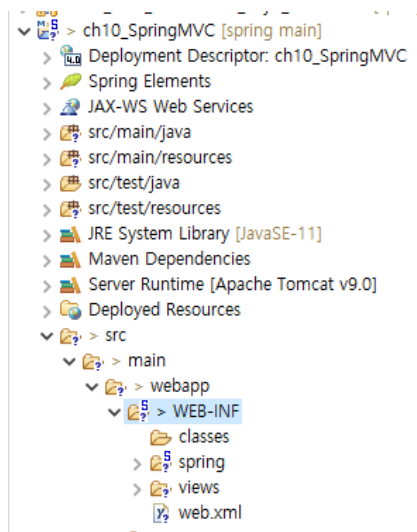
## Spring Legacy Project.



Dynamic Web Module 4버전이 Servlet을 말하는 것이고 Tomcat9버전과 호환이 된다.



변경해줘야 하는 것들.



## web.xml



## pom.xml

java version 11로

framework를 5.2.22.로

```

<packaging>war</packaging>
<version>1.0.0-BUILD-SNAPSHOT</version>
<properties>
  <java-version>11</java-version>
  <org.springframework-version>5.2.22.RELEASE</org.springframework-version>
  <org.aspectj-version>1.6.10</org.aspectj-version>
  <org.slf4j-version>1.6.6</org.slf4j-version>
</properties>
<dependencies>

```

아래 부분을 아예 전체 변경해줌.

화면에는 변경된 후를 보여주고 있다.

```

<!-- servlet -->
<!-- https://mvnrepository.com/artifact/javax.servlet/javax.servlet-api -->
<dependency>
  <groupId>javax.servlet</groupId>
  <artifactId>javax.servlet-api</artifactId>
  <version>4.0.1</version>
  <scope>provided</scope>
</dependency>
</dependencies>

```

```

<!-- servlet -->
<!-- https://mvnrepository.com/artifact/javax.servlet/javax.servlet-api -->
<dependency>
  <groupId>javax.servlet</groupId>
  <artifactId>javax.servlet-api</artifactId>
  <version>4.0.1</version>
  <scope>provided</scope>
</dependency>

```

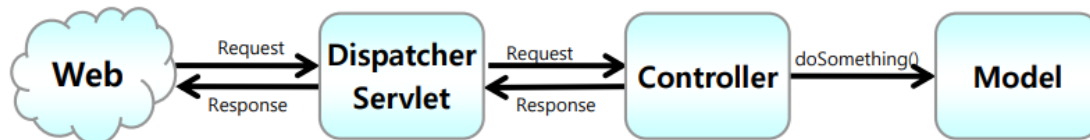
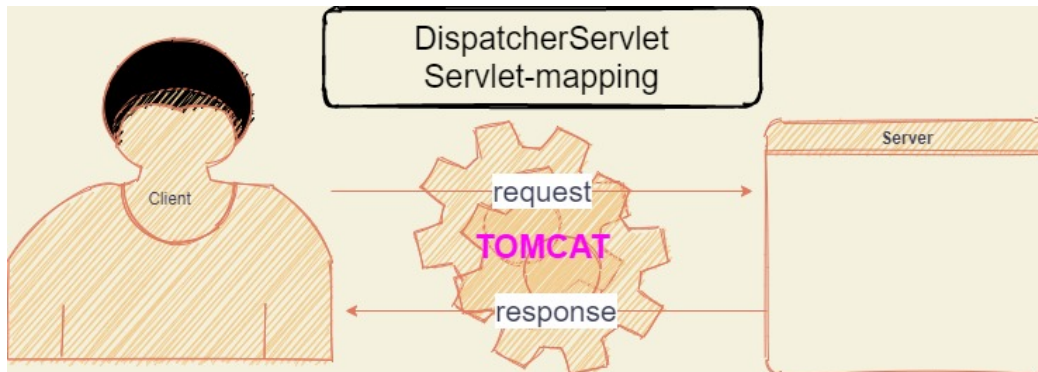
```

<plugin>
  <groupId>org.apache.maven.plugins</groupId>
  <artifactId>maven-compiler-plugin</artifactId>
  <version>2.5.1</version>
  <configuration>
    <source>11</source> <!-- 11로 변경했음 -->
    <target>11</target> <!-- 11로 변경했음 -->
    <compilerArgument>-Xlint:all</compilerArgument>
    <showWarnings>true</showWarnings>
    <showDeprecation>true</showDeprecation>
  </configuration>
</plugin>
</plugins>

```

Tomcat에게 알려주는 설정파일 - web.xml

[https://s3-us-west-2.amazonaws.com/secure.notion-static.com/6edf2a7b-6b19-4301-b41f-6271e6ba41ae/20220914\\_Day8\\_JSP%EA%B8%B0%EC%A4%80\\_WebApplicationService.drawio](https://s3-us-west-2.amazonaws.com/secure.notion-static.com/6edf2a7b-6b19-4301-b41f-6271e6ba41ae/20220914_Day8_JSP%EA%B8%B0%EC%A4%80_WebApplicationService.drawio)



```
<!-- Processes application requests -->
<servlet>
  <servlet-name>appServlet</servlet-name>

  <!-- 모든 요청을 클래스로 만들어져있는 DispatcherServlet에게 서버에게 처리해달라고 넘겨주는 것 -->
  <servlet-class>org.springframework.web.servlet.DispatcherServlet</servlet-class>
  <init-param>
    <param-name>contextConfigLocation</param-name> <!-- 파라미터를 초기화해주는 이름 -->
    <param-value>/WEB-INF/spring/appServlet/servlet-context.xml</param-value> <!-- servlet-context.xml 파일을 읽어가라 명령함. -->
  </init-param>
  <load-on-startup>1</load-on-startup>
</servlet>

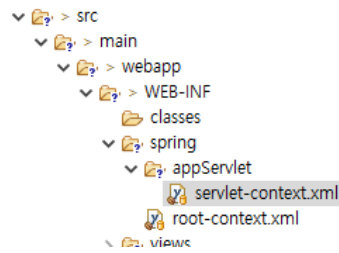
<servlet-mapping>
  <servlet-name>appServlet</servlet-name>
  <url-pattern>/</url-pattern> <!-- /표시는 모든 요청이다. -->
</servlet-mapping>
```

```
<!-- Processes application requests -->
<servlet>
  <servlet-name>appServlet</servlet-name>

  <!-- 모든 요청을 클래스로 만들어져있는 DispatcherServlet에게 서버에게 처리해달라고 넘겨주는 것 -->
  <servlet-class>org.springframework.web.servlet.DispatcherServlet</servlet-class>
  <init-param>
    <param-name>contextConfigLocation</param-name> <!-- 파라미터를 초기화해주는 이름 -->
    <param-value>/WEB-INF/spring/appServlet/servlet-context.xml</param-value> <!-- servlet-context.xml 파일을 읽어가라 명령함. -->
  </init-param>
  <load-on-startup>1</load-on-startup>
</servlet>

<servlet-mapping>
  <servlet-name>appServlet</servlet-name>
  <url-pattern>/</url-pattern> <!-- /표시는 모든 요청이다. -->
</servlet-mapping>
```

servlet-context.xml 파일을 읽어가라 명령함.



xml 작업을 가능하게 해주는 정보

```
<!-- 톰캣에게 알려주는 정보를 여기에 작업함 -->

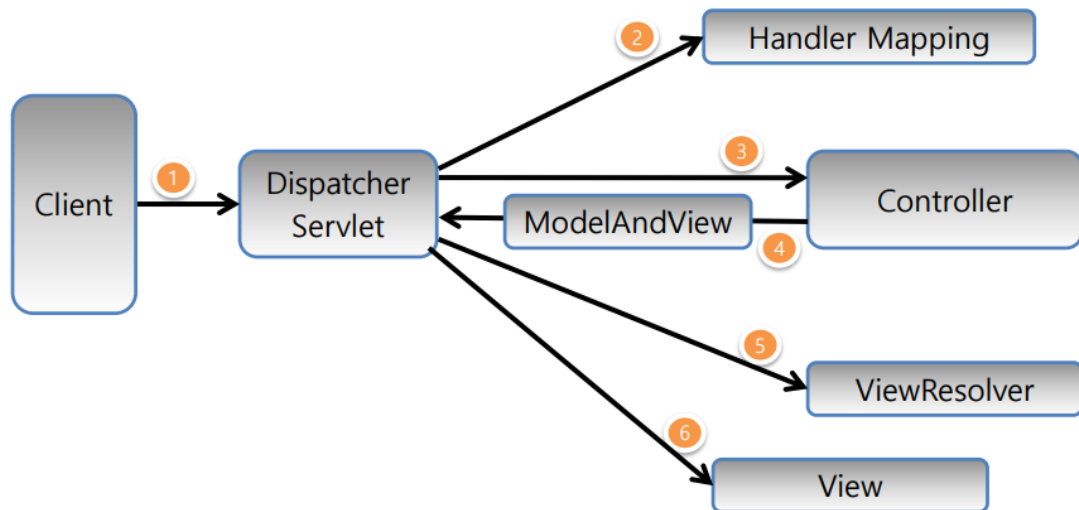
<!-- The definition of the Root Spring Container shared by all Servlets and Filters -->
<context-param> <!-- -->
  <!-- setContextConfigLocation 작동. -->
  <param-name>contextConfigLocation</param-name>
  <!-- 비즈니스 로직을 처리·관리할 파일, 일반 xml파일이 여기에서 구동되게 해줌. -->
  <param-value>/WEB-INF/spring/root-context.xml</param-value>
</context-param>

<!-- Creates the Spring Container shared by all Servlets and Filters -->
<!-- 컨테이너를 작동시키게 하는 엘리먼트가 Listener이고 ContextLoaderListener가 작동된다. 비즈니스 로직이 활성화되어진다. -->
<listener>
  <listener-class>org.springframework.web.context.ContextLoaderListener</listener-class>
</listener>
```

**동시에 Dispatcher New 생성자, Handler Mapping New 생성자, Controller New 생성자 .... 만들어지면서 자동으로 호출한다.**

# Spring MVC 흐름 (1/2)

- Spring MVC
  - MVC 패턴 기반 웹 개발 프레임워크



2

'/'가 아니라 board로 바꾸니 404 에러가 나옴.

```
</servlet>
<servlet-mapping>
  <servlet-name>appServlet</servlet-name>
  <url-pattern>*.board</url-pattern> <!--
</servlet-mapping>
```

web.xml

## HTTP 상태 404 – 찾을 수 없음

타입 상태 보고

설명 Origin 서버가 대상 리소스를 위한 현재의 representation을 찾

Apache Tomcat/9.0.65

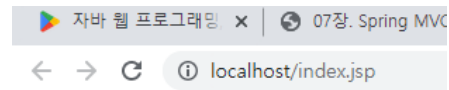


localhost로 사용하기 위해 톰캣 포트번호를 '80' 변경해줌.

modify the server ports.

Port Name	Port Number
Tomcat admin port	8005
HTTP/1.1	80

#### MIME Mappings



## Main HomePage

index.jsp에서 hello.do로 바로 다시 보냄.

```
</head>
<body>
    <% response.sendRedirect("./hello.do"); %>
</body>
</html>
```

.do 파일 실행되면 (tomcat이 아닌)Spring에서 작업하라고 설정

```
<servlet-mapping>
    <servlet-name>appServlet</servlet-name>
    <url-pattern>*.do</url-pattern> <!-- '
</servlet-mapping>

</web-app>
```

servlet을 처리할 때 servlet-context.xml도 같이 호출

```

<!-- Processes application requests -->
<servlet>
    <servlet-name>appServlet</servlet-name>

    <!-- 모든 요청을 클래스로 만들어져있는 DispatcherServlet에게 서버에게 처리해달라고 넘겨주는 것 -->
    <servlet-class>org.springframework.web.servlet.DispatcherServlet</servlet-class>
    <init-param>
        <param-name>contextConfigLocation</param-name> <!-- 파라미터를 초기화해주는 이름 -->
        <param-value>/WEB-INF/spring/appServlet/servlet-context.xml</param-value> <!-- servlet-context.xml 파일을 읽어가라 명령함. -->
    </init-param>
    <load-on-startup>1</load-on-startup>
</servlet>

<servlet-mapping>
    <servlet-name>appServlet</servlet-name>
    <url-pattern>*.do</url-pattern> <!-- '/'표시는 모든 요청이다. tomcat이 아니라 Spring이 직접 처리하겠다는 선언 |-->
</servlet-mapping>

```

mapping해서 컨트롤러를 호출해줌.

```

<!-- Handler Mapping 등록 -->
<beans:bean class="org.springframework.web.servlet.handler.SimpleUrlHandlerMapping">
    <beans:property name="mappings">
        <beans:props>
            <beans:prop key="/hello.do"></beans:prop>
        </beans:props>
    </beans:property> <!-- setter 메서드 따라서 setMappings를 호출해줘 -->
</beans:bean>

```

## ViewResolver

```

<!-- Resolves views selected for rendering by @Controllers to .jsp resources in the /WEB-INF/views directory -->
<beans:bean class="org.springframework.web.servlet.view.InternalResourceViewResolver">
    <beans:property name="prefix" value="/WEB-INF/views/" /> <!-- property는 set 메서드 넣어줄 value -->
    <beans:property name="suffix" value=".jsp" />
</beans:bean>

```

## HelloController implements Controller

```

public class HelloController implements Controller{
    @Override
    public ModelAndView handleRequest(HttpServletRequest request, HttpServletResponse response) throws Exception {
        //System.out.println("/hello.do 요청 처리 메서드()");

        ModelAndView mav = new ModelAndView(); // 자료형을 사용하려면 먼저 생성자 호출해야 한다.

        mav.addObject("greeting", "Hello Spring"); // greeting이라는 키값으로 Hello Spring value값을 저장.
        mav.setViewName("hello"); // /WEB-INF/views/hello.jsp ////////// Dispatcher Servlet에 보낼 이름

        return mav;
    }
}

```

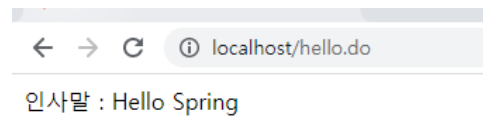
## Hello.jsp

```

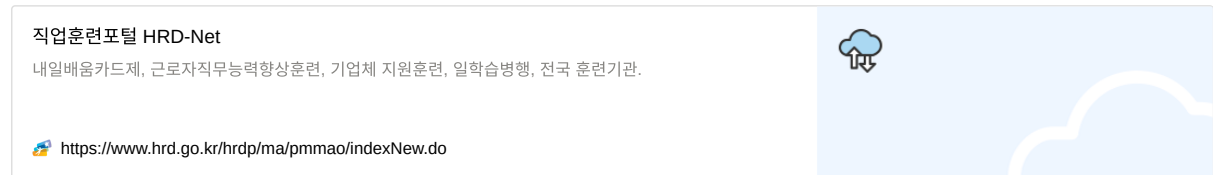
1 <%@ page language="java" contentType=
2     pageEncoding="EUC-KR"%>
3
4 <!DOCTYPE html>
5 <html>
6 <head>
7 <meta charset="EUC-KR">
8 <title>hello 요청 페이지</title>
9 </head>
10 <body>
11
12     인사말 : ${ greeting }
13
14 </body>
15 </html>

```

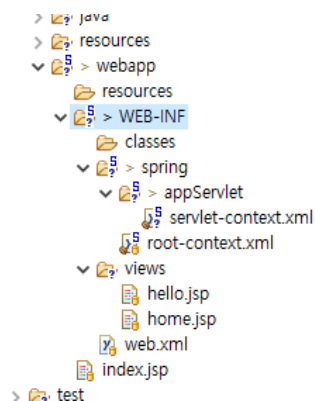
## 출력



hrd-net 홈페이지 들어갈 때도 jsp에서 바로 redirect 넘겨주는 것이 보임.



tomcat관리하는 파일은 webapp까지만 관리하고, 나머지 WEB\_INF 영역 안에서는 Spring만 관리한다.



**.do 파일만 Spring이 관리 관할임.**

---