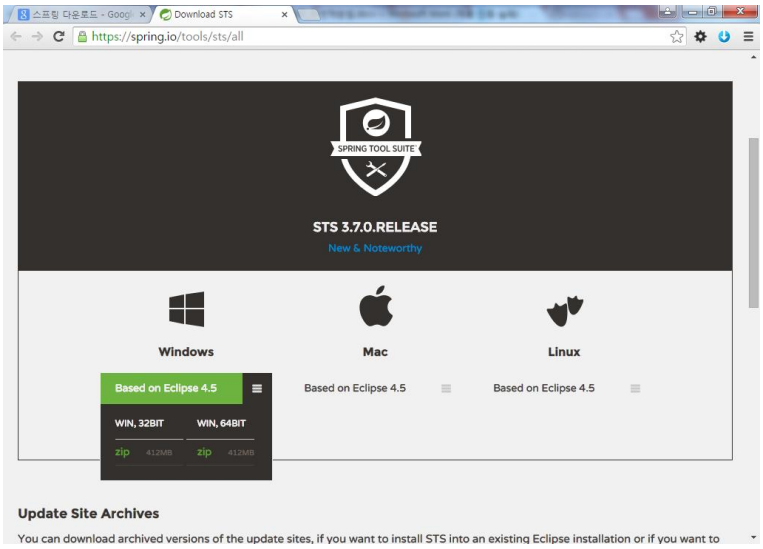
스프링(Spring) 개발 2

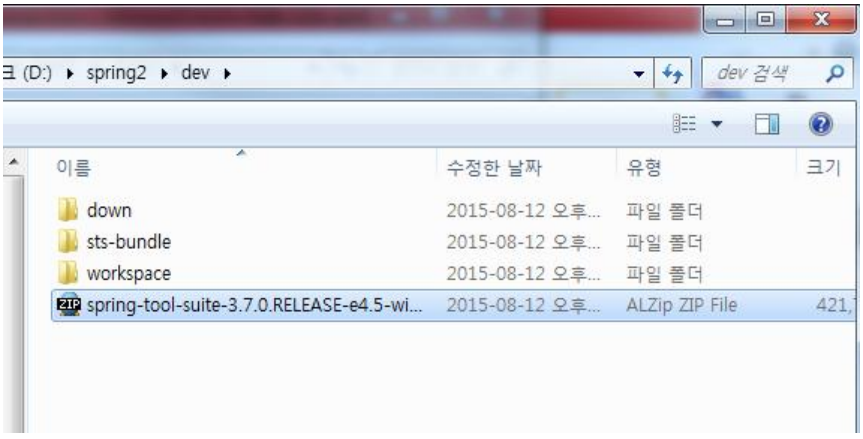
1.프로젝트 생성

(1). Spring Tool Suite(STS) 다운 설치

다운 사이트 : https://spring.io/tools/sts/all

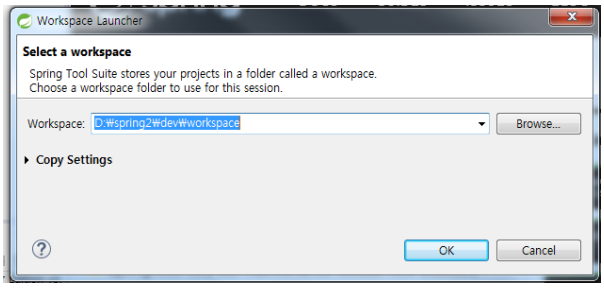


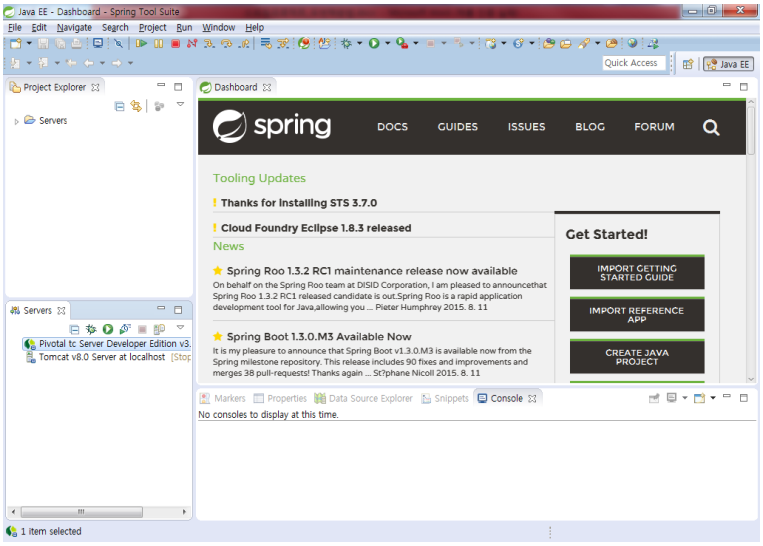
해당하는 OS 에 대한 ZIP 파일을 다운 받는다. 다운받은 파일을 압축을 풀고 sts-bundle 폴더 아래 D:₩spring2₩dev₩sts-bundle₩sts-3.7.0.RELEASE₩STS.exe 을 실행함.



실행을 하면 Spring Tool Suite 실행화면이 나타남.

이클립스처럼 workspace 를 묻는 화면이 나타나고 그 다음 아래의 실행 화면이 나타남.



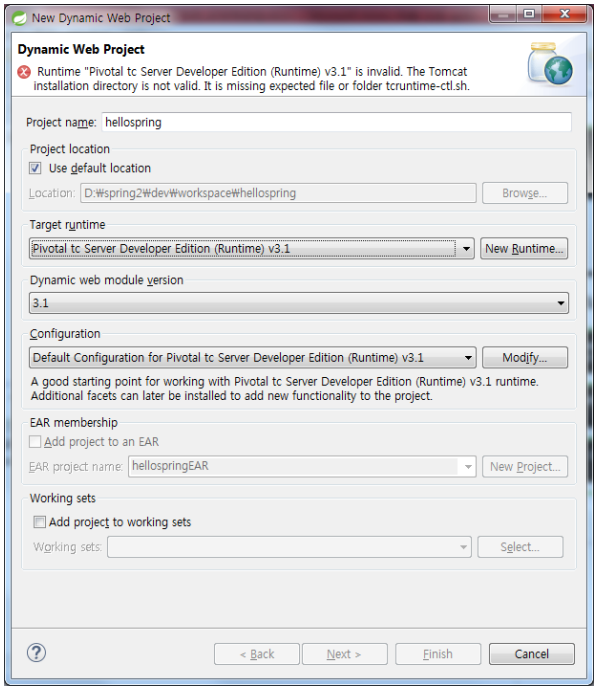


(2). Spring Hello 프로젝트 만들기

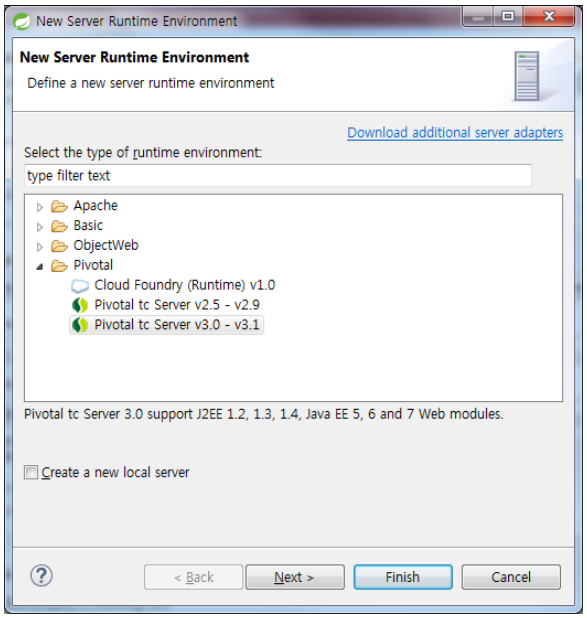
먼저, File ->New -> Dynamic Web Project 를 선택해서 웹 프로젝트를 생성합니다.

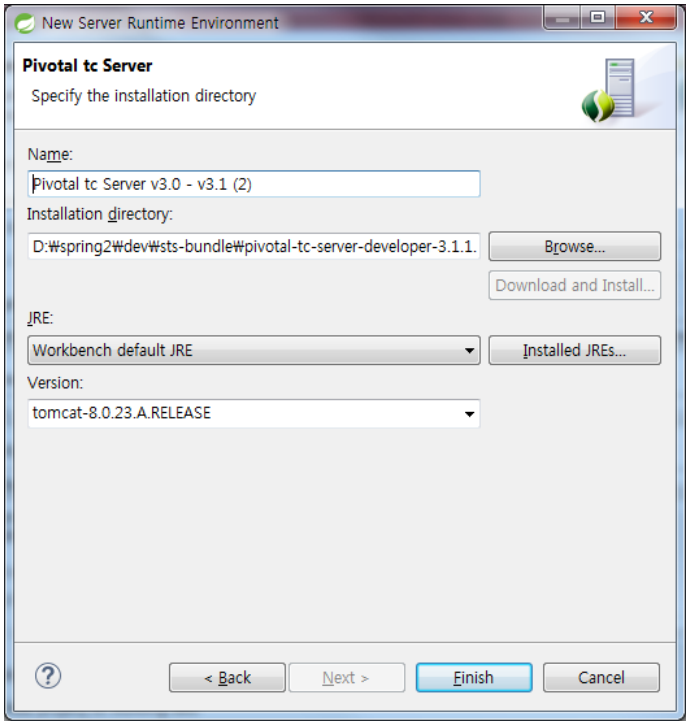
프로젝트 이름 :**hellospring**

Target runtime :**Pivotaktc Server Developer Edition (Runtime) v3.1** 지정함

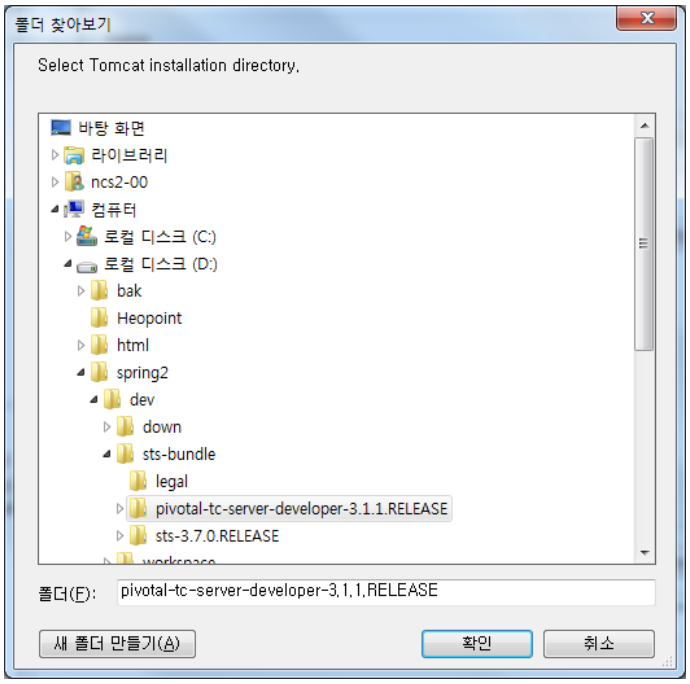


대화상자 상단에 **에러 메시지가**나타나면, Target runtime 오른쪽의 **'New Runtime...'** 버튼 클릭함 >> Pivotal 항목의 **Pivotal tc Server v3.0 –v3.1** 을 선택하고 Next 클릭함



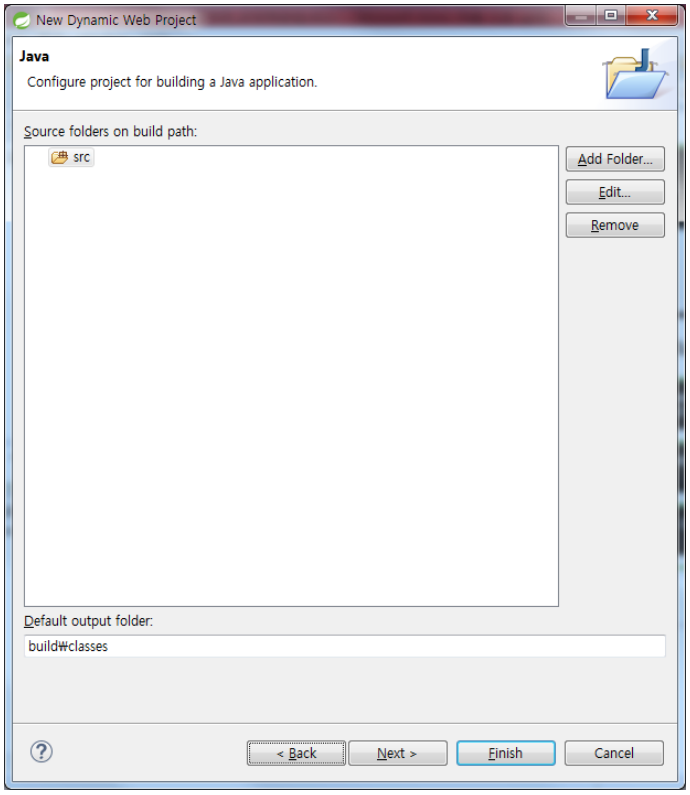


Installation directory 항목의 **Browse...** 클릭함

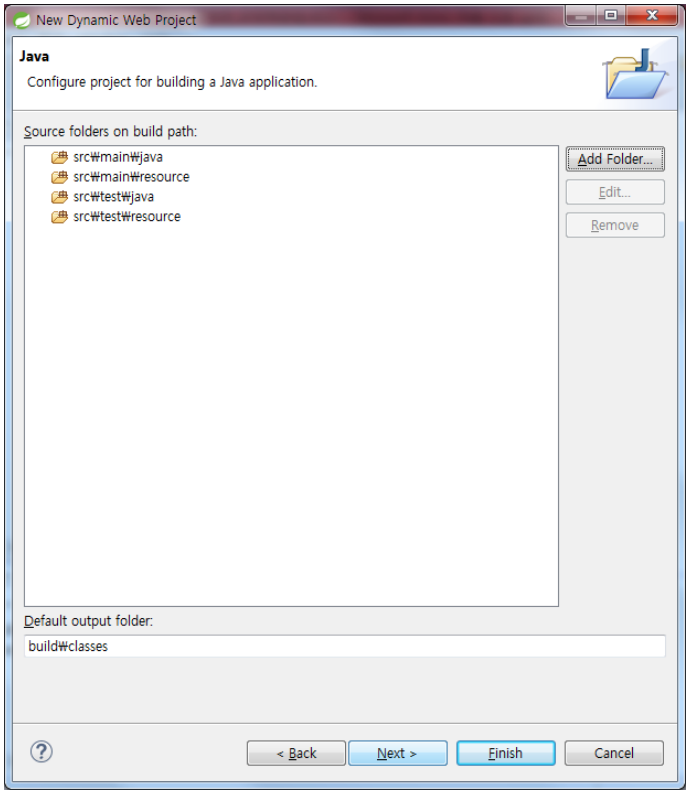


압축 푼 **sts-bundle** 폴더 아래의 **pivotal-tc-server-developer-3.1.1.RELEASE** 를 선택하고 '**확인**' 버튼 누름>>**finish** 버튼 누름

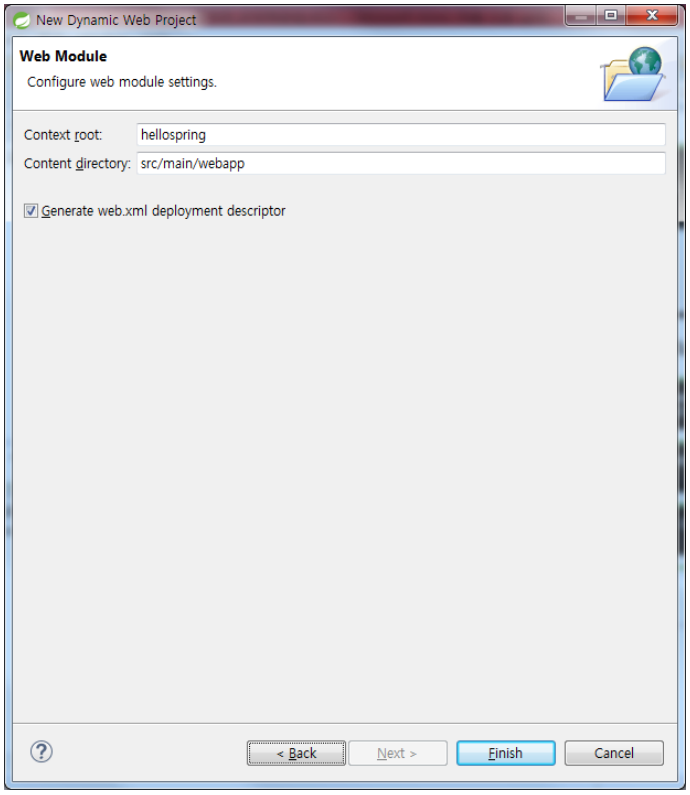
계속 Dynamic Web Project 에서 '**Next**' 누름



src 폴더를 **remove** 함



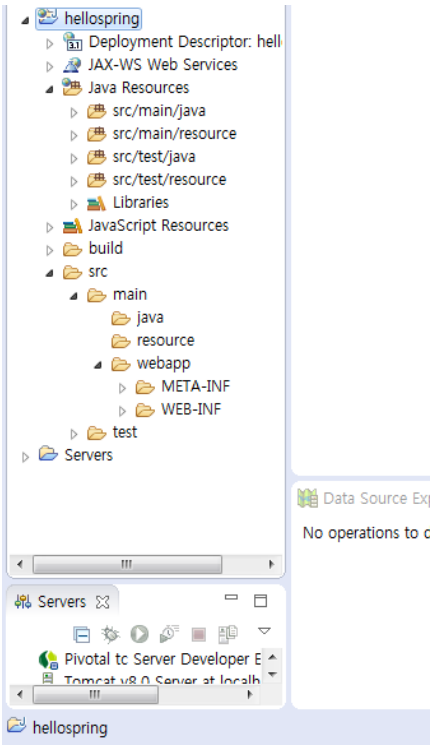
'Add Folder...' 누름 >> 위의 내용대로 **소스 폴더 추가함**>>**Next** 누름



Content directory :**src/main/sebapp**

Web.xml 생성 **체크**함 >>**Finish** 함

Project Explorer 뷰로 봤을 때의 구성 내용임.



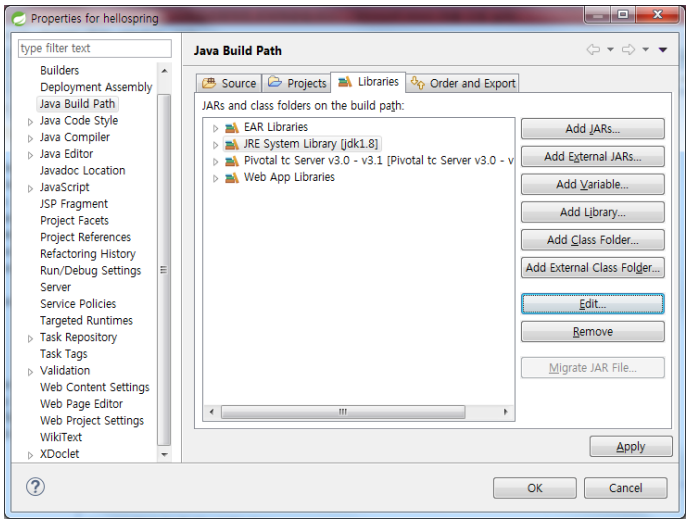
폴더 구성의 이유

main 의 경우 프로젝트로 배포할 때 사용되는 경로

test 의 경우 JUnit 의 단위 테스팅을 위한 경로

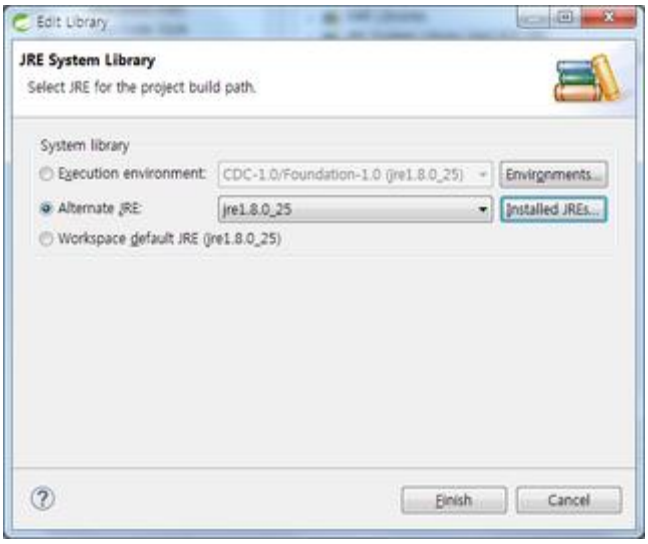
sac/main/webpps 는 실제 웹 실행시 index.jsp,borad.jsp 같은 html 뷰 파일들을 넣기 위한 경로

프로젝트명 ->마우스 우클릭 -> properties 를 클릭 => Java Build Path -> Libraries 탭 ->JRE System Library 가 jdk 로 지정되어 있는지 확인함



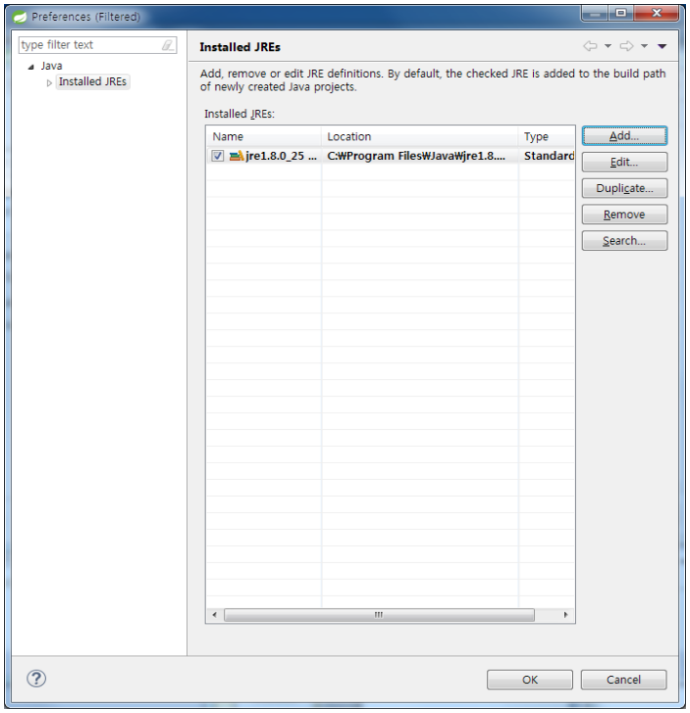
만약 jdk 로 지정되어 있지 않으면 JRE System Library 를 jdk 로 바꿔주는 작업을 해야 함.

JRE System Libraries [jre1.8.025]를 클릭하고 **Edit** 를 클릭하면 아래와 같은 창이 뜹니다.

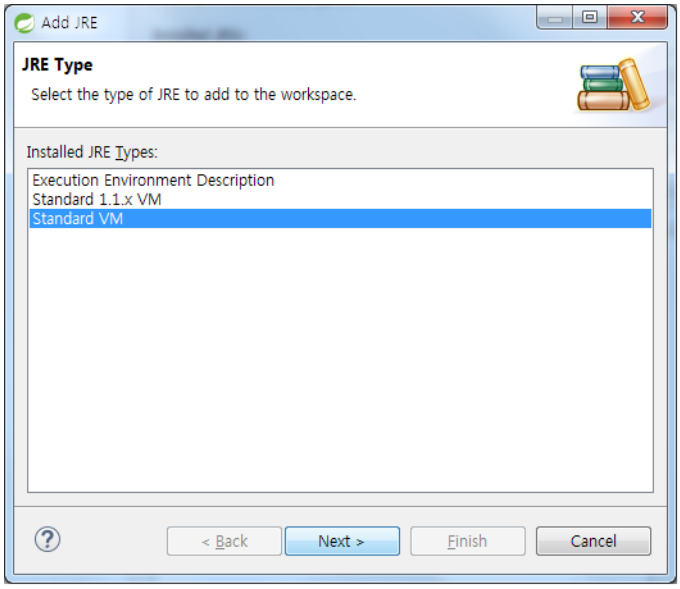


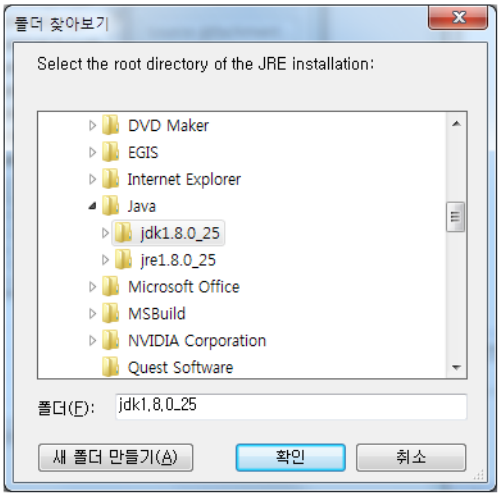
여기서 Installed JREs 를 눌러 직접 JDK 가 설정된 경로를 잡아주면 됨.

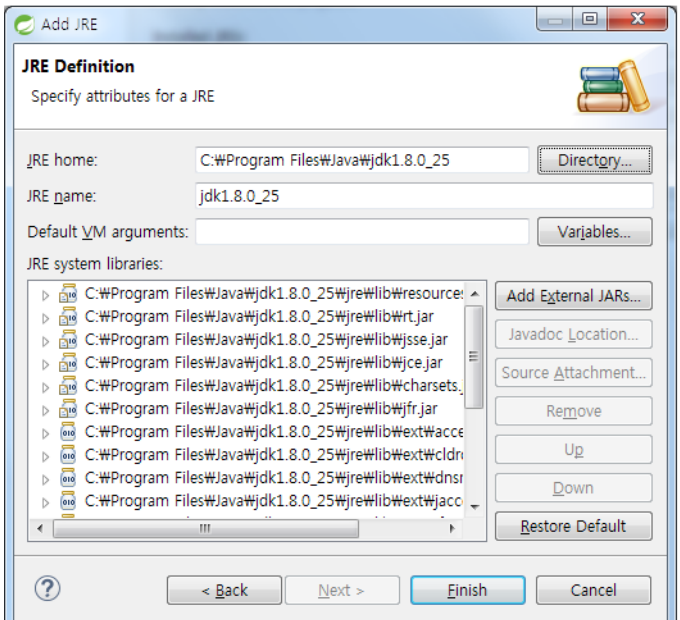
아래 창이 뜨면 Add 를 눌러서 jdk 경로를 직접 설정해 줌.

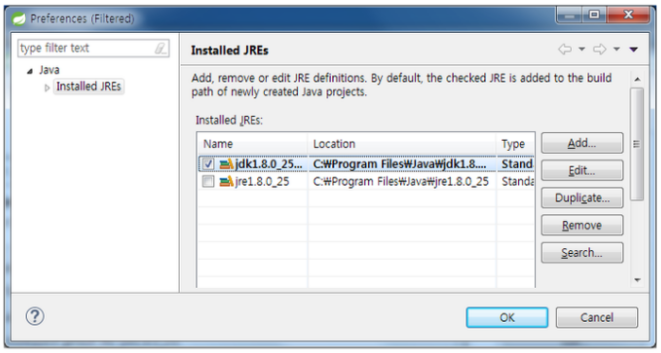


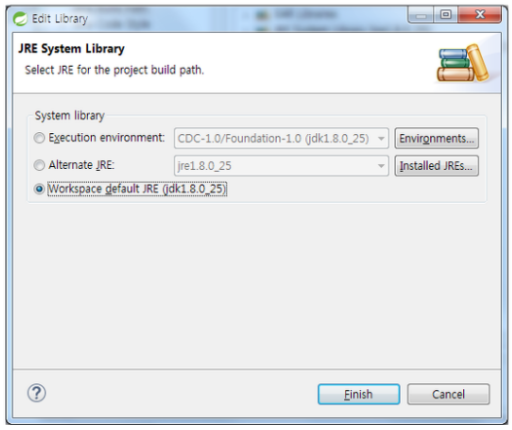
이렇게 경로도 설정해주면 알아서 라이브러리가 잡힘.

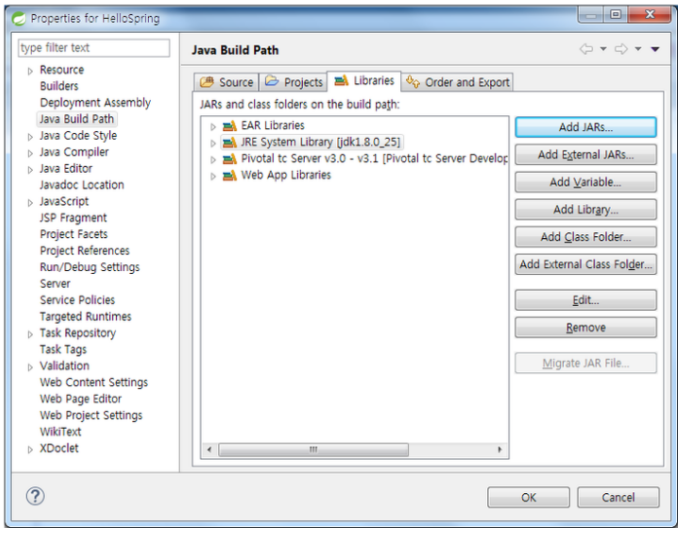

Add JRE
JRE Type
Select the type of JRE to add to the workspace.
Installed JRE Types:
Execution Environment Description
Standard 1.1.x VM
Standard VM
< Back
Next >
Finish
Cancel


폴더 찾아보기
Select the root directory of the JRE installation:
DVD Maker
EGIS
Internet Explorer
Java
jdk1.8.0_25
jre1.8.0_25
Microsoft Office
MSBuild
NVIDIA Corporation
Quest Software
폴더(F): jdk1.8.0_25
새 폴더 만들기(A)
확인
취소


Add JRE
JRE Definition
Specify attributes for a JRE
JRE home: C:₩Program Files₩Java₩jdk1.8.0_25
Directory...
JRE name: jdk1.8.0_25
Default VM arguments:
Variables...
JRE system libraries:
C:₩Program Files₩Java₩jdk1.8.0_25₩jre₩lib₩rt.jar
C:₩Program Files₩Java₩jdk1.8.0_25₩jre₩lib₩jsse.jar
C:₩Program Files₩Java₩jdk1.8.0_25₩jre₩lib₩jce.jar
C:₩Program Files₩Java₩jdk1.8.0_25₩jre₩lib₩jfr.jar
Add External JARs...
Javadoc Location...
Source Attachment...
Remove
Up
Down
Restore Default
< Back
Next >
Finish
Cancel







이로써, 기본적인 세팅은 끝났습니다.

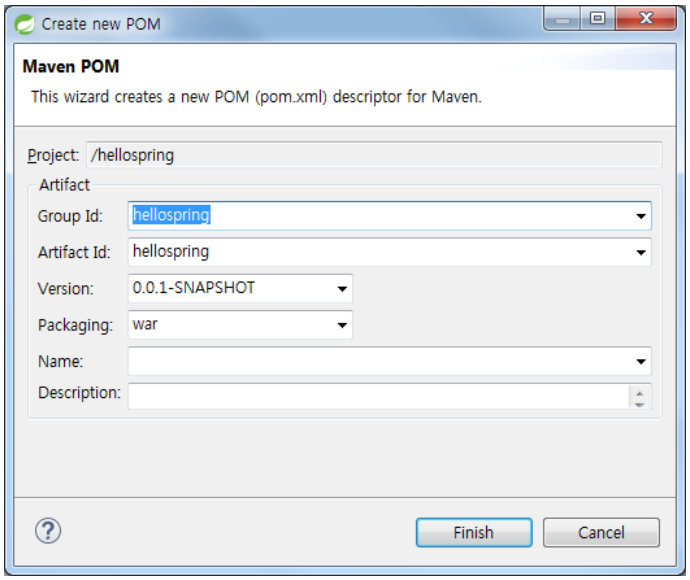
(3). Spring의 Model and View (MVC) 설정

Spring 은 Dependency Injection(DI)에 영향을 많이 받기 때문에, 의존성 패키지가 없으면 실행이 안되거나 프로젝트 설정이 어렵게 됨.

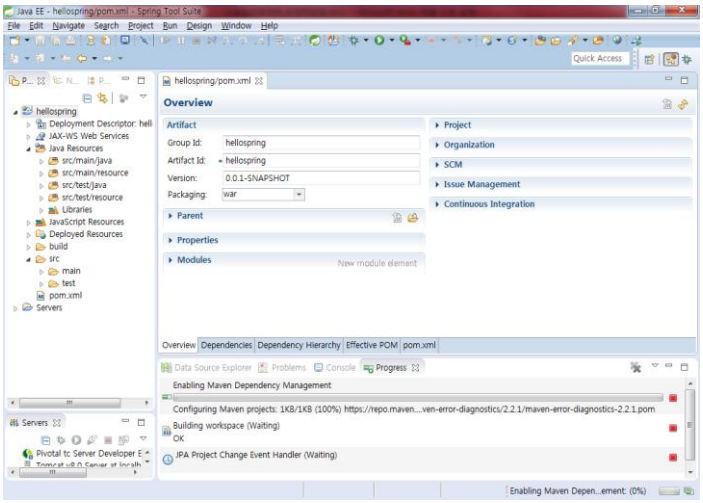
이를 해결하기 위해서는 Maven 이라는 툴을 사용함으로써 의존성 패키지를 자동으로 다운로드 하는게 가능함.

먼저 생성된 프로젝트를 Maven 프로젝트로 변경해본다.

프로젝트 이름 위에서 ->마우스 우클릭 ->**Configure** ->**Convert to Maven Project** 선택



-> Finish 를 하면 자동으로 Maven 프로젝트로 바뀌줌.

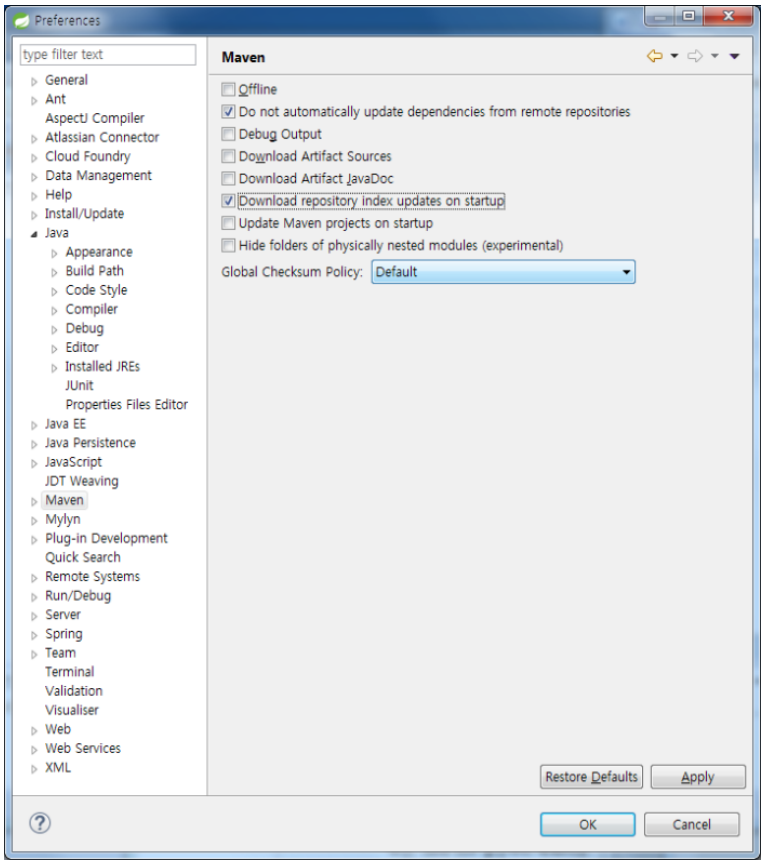


화면이 위와 같이 바뀌고, 프로젝트 아이콘 왼쪽 위에 'M' 이 표시된 것을 확인함.

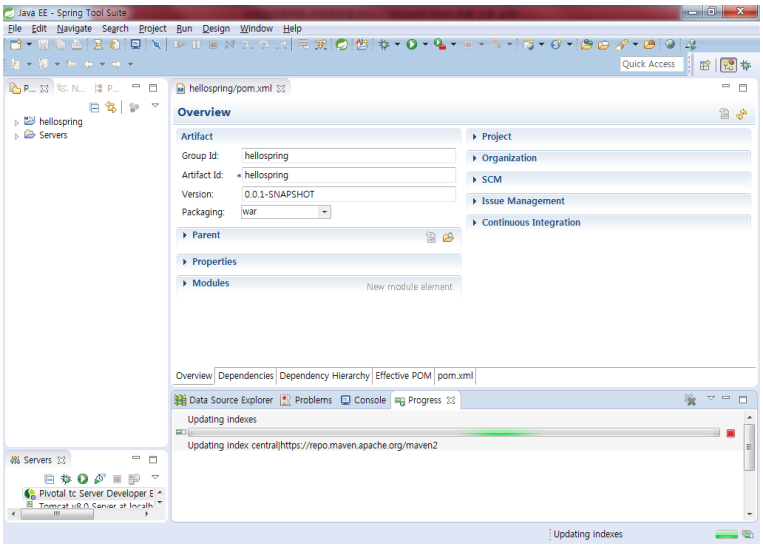
의존성 패키지를 설치하기 위해서는, 패키지들을 자동으로 다운받기 위한 설정을 해주어야 함.

windows->preferences->Maven->Download repository index updates on startup 체크함.

의존성 패키지 검색이 가능합니다. 설정 후 STS 를 restart 합니다.



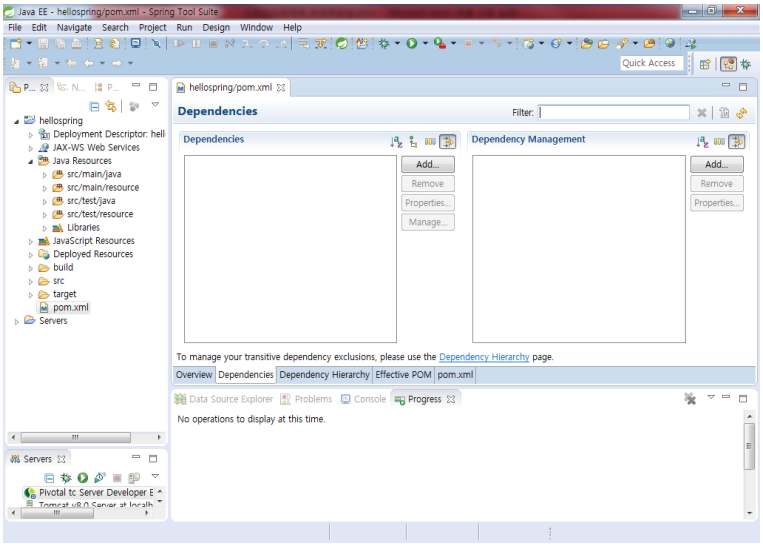
progress 뷰에서 updating 다운로드가 진행되는 것을 볼 수 있음.



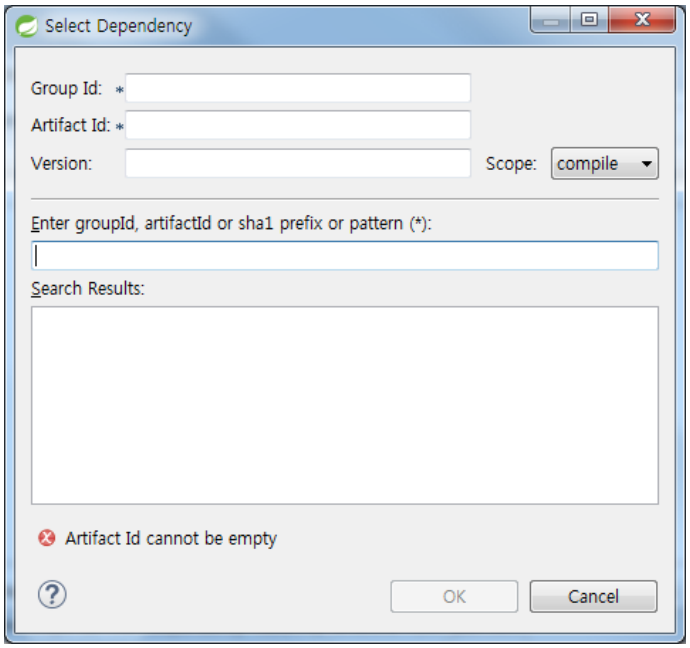
다운이 완료되면 의존성 검색을 하기 위한 세팅이 완료되었음.

기본적인 Spring framework 를 사용하기 위한 의존성 패키지를 주입(Injection)

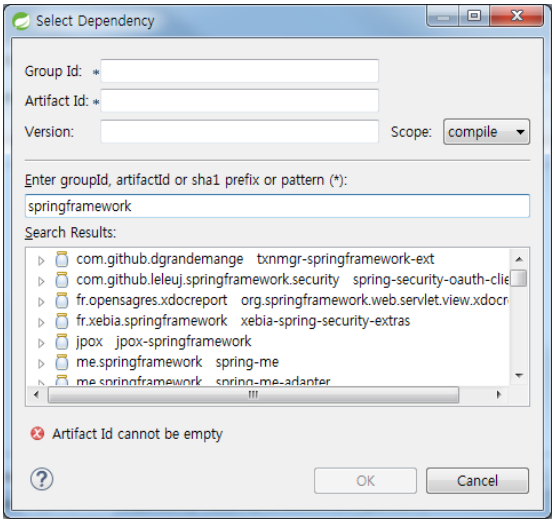
hellospring/pom.xml 화면에서 -> Dependencies 탭 클릭 -> Add 를 클릭합니다.



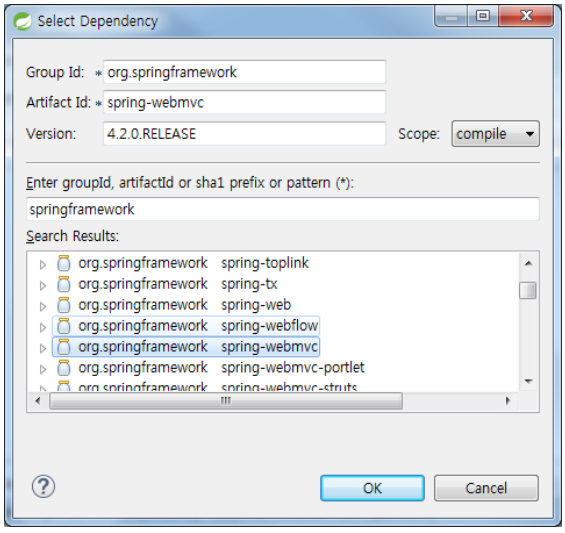
나타난 대화상자에서



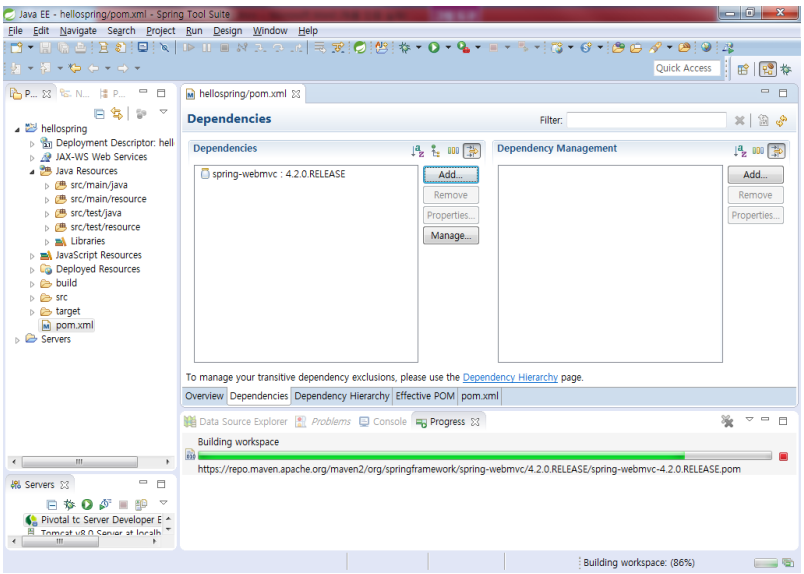
springframework 를 입력하고 엔터해서 검색이 되면,



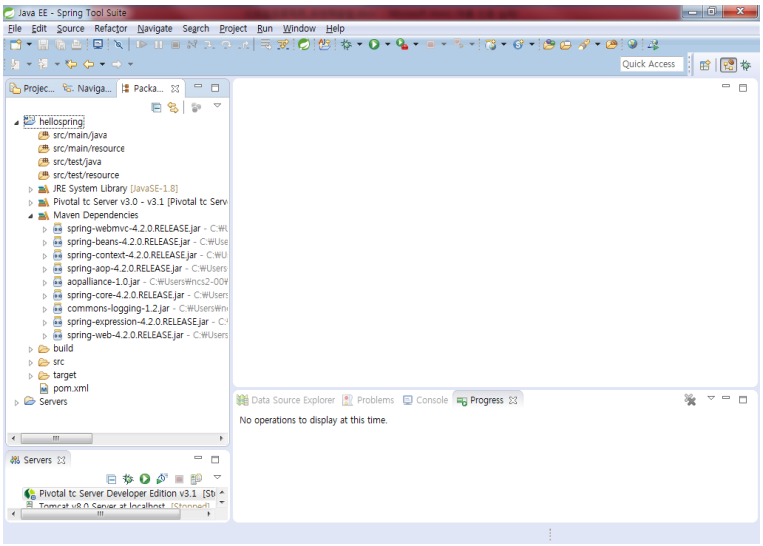
org.springframework spring-web mvc 를 선택하고 ok 함.



pom.xml 를 저장하면 STS 가 필요한 패키지를 알아서 다운로드 받는다



package explorer 뷰에서 보면, 아래와 같이 라이브러리가 추가되었음을 확인할 수 있음.



디스패처서블릿을 사용하기 위한 web.xml 파일 수정

이제 mvc 가 참조하게 될 dispatcherServlet.xml 를 사용하기 위한 경로 설정을 해야 하며, dispatcherServlet.xml 를 연결하기 위하여 web.xml 파일을 수정해야 함.

소스는 아래와 같다.

webapp/WEB-INF/web.xml

```
1  <?xmlversion="1.0"encoding="UTF-8"?>
2  <web-appxmlns: xsi="http://www.w3.org/2001/XMLSchema-instance"
3   xmlns="http://xmlns.jcp.org/xml/ns/javaee"
4   xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/javaee
5   http://xmlns.jcp.org/xml/ns/javaee/web-app_3_1.xsd" id="WebApp_ID"version="3.1">
6
7   <display-name>HelloSpring</display-name>
8   <welcome-file-list>
9   <welcome-file>index.html</welcome-file>
10  <welcome-file>index.htm</welcome-file>
11  <welcome-file>index.jsp</welcome-file>
12  <welcome-file>default.html</welcome-file>
13  <welcome-file>default.htm</welcome-file>
14  <welcome-file>default.jsp</welcome-file>
15  </welcome-file-list>
16
```

```
17  <servlet>
18  <servlet-name>dispatcherServlet</servlet-name>
19  <servlet-class>
20  org.springframework.web.servlet.DispatcherServlet
21  </servlet-class>
22  <init-param>
23  <param-name>contextConfigLocation</param-name>
24  <param-value>/WEB-INF/config/spring/dispatcherServlet.xml</param-value>
25  </init-param>
26  </servlet>
27
28  <servlet-mapping>
29  <servlet-name>dispatcherServlet</servlet-name>
30  <url-pattern>/</url-pattern>
31  </servlet-mapping>
32
33  <filter>
34  <filter-name>encodingFilter</filter-name>
35  <filter-class>
36  org.springframework.web.filter.CharacterEncodingFilter
37  </filter-class>
38
39  <init-param>
40  <param-name>encoding</param-name>
41  <param-value>UTF-8</param-value>
42  </init-param>
43  </filter>
44
45  <filter-mapping>
46  <filter-name>encodingFilter</filter-name>
47  <url-pattern>/*</url-pattern>
48  </filter-mapping>
49 </web-app>
```

여기서 prefix 는 /WEB-INF/view/의 고정된 패스입니다. 앞으로 /HelloSpring/view 경로에서 hello.jsp 를 매핑할 겁니다.

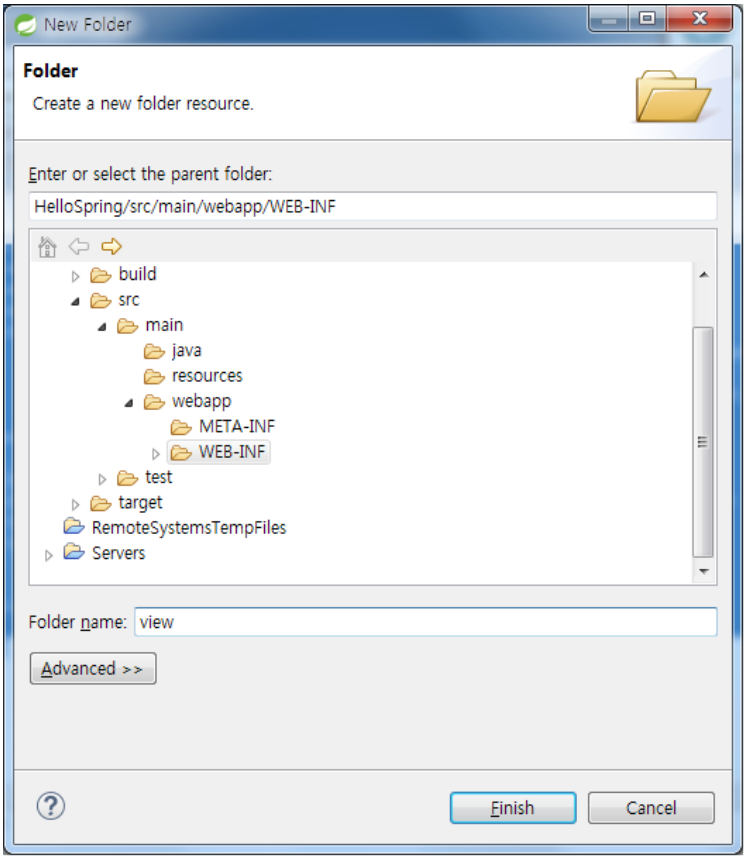
(4). MVC 를 사용한 모델/뷰 처리

A. 서블릿 URI 를 매핑할 hello

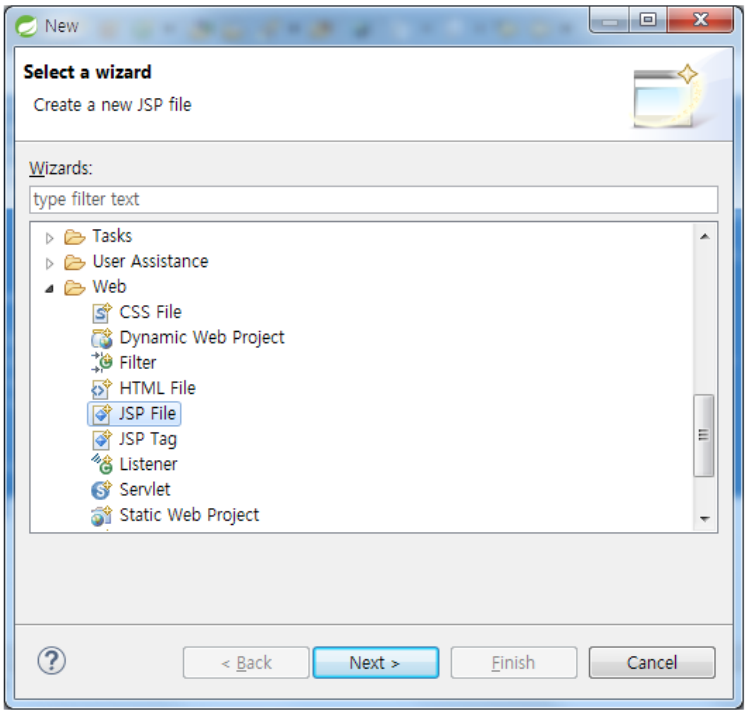
B. 이를 표시할 hello.jsp 가 필요합니다.

test.jsp 를 만들어보겠습니다.

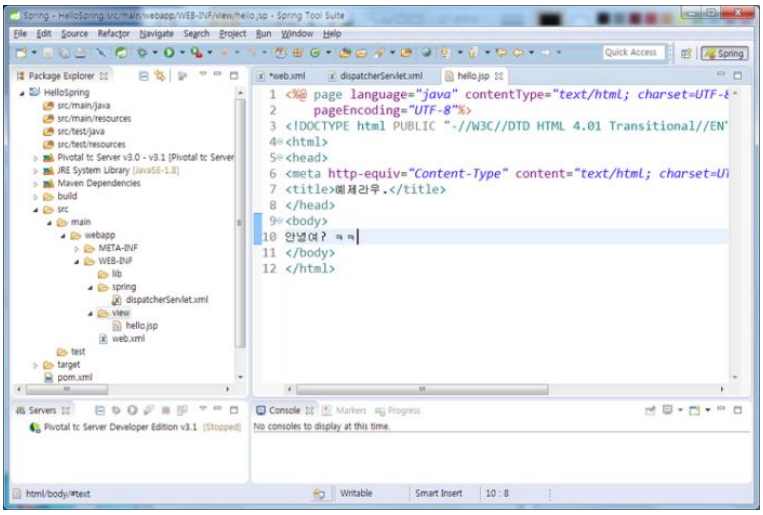
WEB-INF/view 폴더를 먼저 만들고 거기에다가 hello.jsp 을 생성합니다.



webapp/view/hello.jsp 생성



코딩 ㅋ

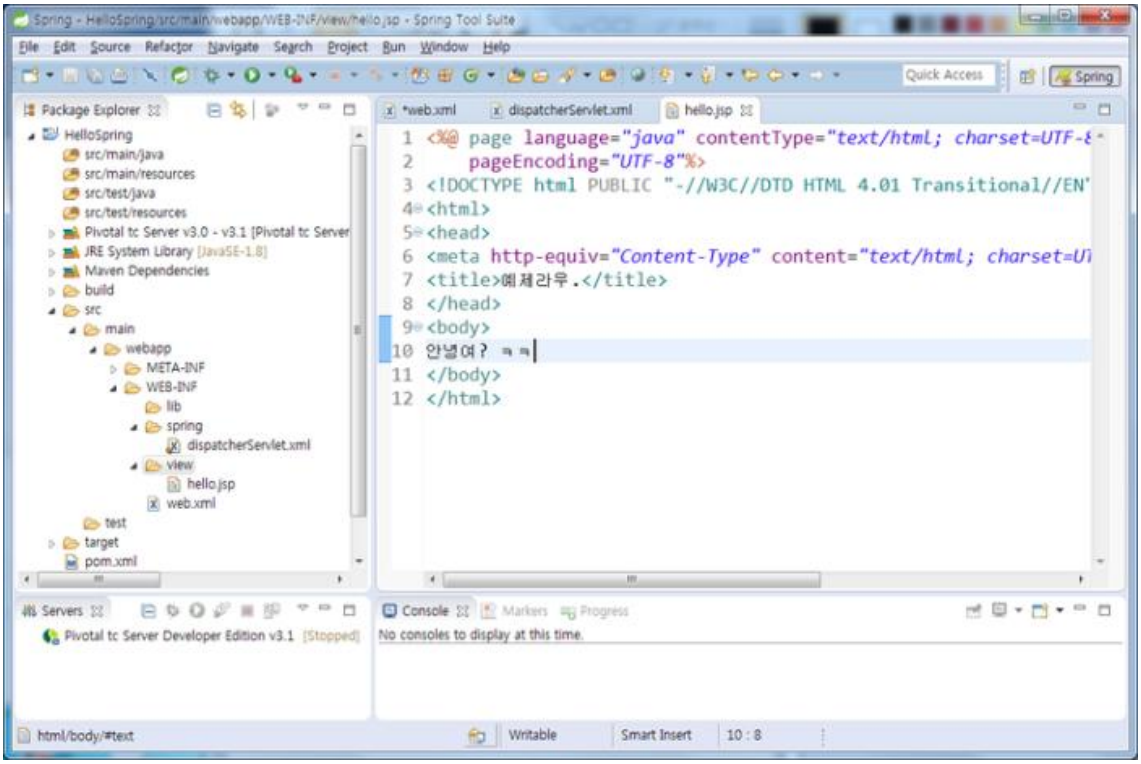


다음에는 /hello 라는 경로로 매핑을 해보겠습니다. 간단한 코딩으로 시작합니다.

src/main/java 에 패키지를 하나 만듭니다.

지금은 com.ktds.christof_kim.mvc.web 패키지에다가 컨트롤러를 만들었습니다.

(참고로, WEB-INF/config/spring/dispatcher.xml 에 위치함)



생성하고 @Controller 라는 어노테이션 표시해줘야 하며, 필요한 패키지를 설정해 줘야 합니다.

그런데 우린 이미 DI 에서 필요한 패키지를 알아서 받았기 때문에 별도의 설정은 필요 없습니다.

(만약 안뜨신 분은 spring-mvc library 버전이 1.3.6 인가 확인해주세요.)

그냥 import 해서 쓰면 됩니다.

pacakage :src/main/java/com.ktds.christof_kim.mvc.web

```
1        packagecom.ktds.christof_kim.mvc.web;
2
3        importorg.springframework.stereotype.Controller;
4        importorg.springframework.web.bind.annotation.RequestMapping;
5        importorg.springframework.web.servlet.ModelAndView;
6
7        @Controller
8        publicclassPrintHelloController {
9             @RequestMapping("/hello")
10           publicModelAndViewhelloView(){
11               ModelAndView view = newModelAndView();
12               view.setViewName("/hello");
13               returnview;
14           }
```

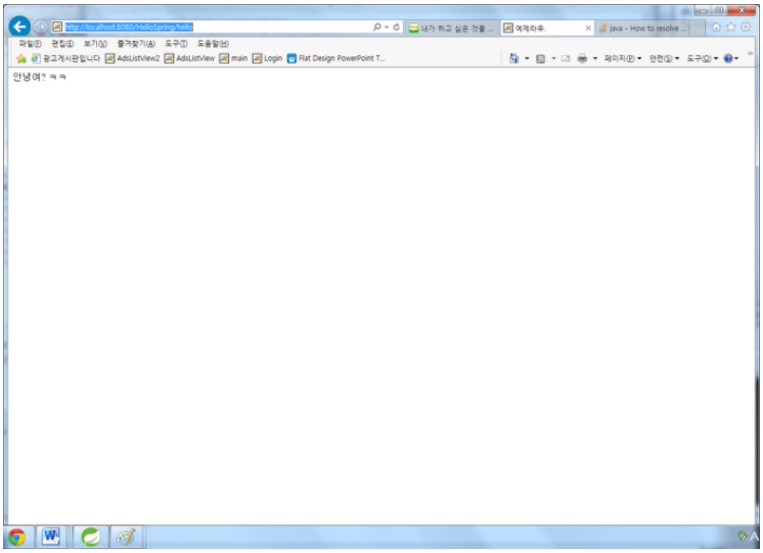
마지막으로 dispatcherServlet.xml 에 표시할 뷰를 bean 에 추가합니다.

(빈 추가 설정은 Spring bean configuration file 를 이용해서 추가합니다.)

webapp/WEB-INF/config/spring/dispatcherServlet.xml

```
1  <?xmlversion="1.0"encoding="UTF-8"?>
2  <beansxmlns="http://www.springframework.org/schema/beans"
3   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
```

```
4   xmlns:mvc="http://www.springframework.org/schema/mvc"
5   xsi:schemaLocation="http://www.springframework.org/schema/beans
6   http://www.springframework.org/schema/beans/spring-beans.xsd
7   http://www.springframework.org/schema/mvc
8   http://www.springframework.org/schema/mvc/spring-mvc-4.1.xsd">
9
10<mvc:annotation-driven/>
11
12<beanid="viewResolver"
13 class="org.springframework.web.servlet.view.InternalResourceViewResolver">
14  <propertyname="prefix"value="/WEB-INF/view/"/>
15  <propertyname="suffix"value=".jsp"/>
16</bean>
17
18<!--TODO:Add Controller -->
19<beanid="printHelloController"class="com.ktds.christof_kim.mvc.web.PrintHelloController"/>
20
21</beans>
```



잘 실행됩니다.

(5). 결론

1. 설정에 필요한 건 결국 메이븐 프로젝트를 통해 해결한다.!

2. xml 설정 dispatcherServlet.xml / pom.xml / 등등등...)

3. 폴더 설정(src/main/java, src/main/resources 등등등...)

```
@@ -0,0 +1,34 @@
+<?xml version="1.0" encoding="UTF-8"?>
+<classpath>
+	<classpathentry kind="src" output="target/classes" path="src/main/java">
+		<attributes>
+			<attribute name="optional" value="true"/>
+			<attribute name="maven.pomderived" value="true"/>
+		</attributes>
```

```
+	</classpathentry>
+	<classpathentry including="**/*.java" kind="src" path="src/main/resources"/>
+	<classpathentry kind="src" output="target/test-classes" path="src/test/java">
+		<attributes>
+			<attribute name="optional" value="true"/>
+			<attribute name="maven.pomderived" value="true"/>
+		</attributes>
+	</classpathentry>
+	<classpathentry including="**/*.java" kind="src" path="src/test/resources"/>
+	<classpathentry kind="con" path="org.eclipse.jst.server.core.container/com.springsource.
+		<attributes>
+			<attribute name="owner.project.facets" value="jst.web"/>
+		</attributes>
+	</classpathentry>
+	<classpathentry kind="con" path="org.eclipse.jdt.launching.JRE_CONTAINER">
+		<attributes>
+			<attribute name="maven.pomderived" value="true"/>
+		</attributes>
+	</classpathentry>
+	<classpathentry kind="con" path="org.eclipse.m2e.MAVEN2_CLASSPATH_CONTAINER">
+		<attributes>
+			<attribute name="maven.pomderived" value="true"/>
+			<attribute name="org.eclipse.jst.component.dependency" value="/WE
```

```
+                    </attributes>
+            </classpathentry>
+            <classpathentry kind="output" path="target/classes"/>
++</classpath>
```

View

42 ΣΣΣΣΣ HelloSpring/.project

```
@@ -0,0 +1,42 @@
++<?xml version="1.0" encoding="UTF-8"?>
++<projectDescription>
+            <name>HelloSpring</name>
+            <comment></comment>
+            <projects>
+            </projects>
+            <buildSpec>
+                    <buildCommand>
+                            <name>org.eclipse.wst.jsdt.core.javascriptValidator</name>
+                            <arguments>
+                            </arguments>
+                    </buildCommand>
+                    <buildCommand>
+                            <name>org.eclipse.jdt.core.javabuilder</name>
+                            <arguments>
```

```
+                        </arguments>
+                </buildCommand>
+                <buildCommand>
+                        <name>org.eclipse.wst.common.project.facet.core.builder</name>
+                        <arguments>
+                        </arguments>
+                </buildCommand>
+                <buildCommand>
+                        <name>org.eclipse.wst.validation.validationbuilder</name>
+                        <arguments>
+                        </arguments>
+                </buildCommand>
+                <buildCommand>
+                        <name>org.eclipse.m2e.core.maven2Builder</name>
+                        <arguments>
+                        </arguments>
+                </buildCommand>
+        </buildSpec>
+        <natures>
+                <nature>org.eclipse.m2e.core.maven2Nature</nature>
+                <nature>org.eclipse.jem.workbench.JavaEMFNature</nature>
+                <nature>org.eclipse.wst.common.modulecore.ModuleCoreNature</nature>
+                <nature>org.eclipse.wst.common.project.facet.core.nature</nature>
```

```
+                    <nature>org.eclipse.jdt.core.javanature</nature>
+                    <nature>org.eclipse.wst.jsdt.core.jsNature</nature>
+          </natures>
+</projectDescription>
```

View

12 ΣΣΣΣΣ HelloSpring/.settings/.jsdtscope

```
@@ -0,0 +1,12 @@
+<?xml version="1.0" encoding="UTF-8"?>
+<classpath>
+          <classpathentry kind="src" path="src/main/webapp"/>
+          <classpathentry kind="con" path="org.eclipse.wst.jsdt.launching.JRE_CONTAINER"/>
+          <classpathentry kind="con" path="org.eclipse.wst.jsdt.launching.WebProject">
+                    <attributes>
+                              <attribute name="hide" value="true"/>
+                    </attributes>
+          </classpathentry>
+          <classpathentry kind="con" path="org.eclipse.wst.jsdt.launching.baseBrowserLibrary"/>
+          <classpathentry kind="output" path=""/>
+</classpath>
```

View

8 ΣΣΣΣΣ HelloSpring/.settings/org.eclipse.jdt.core.prefs

```
@@ -0,0 +1,8 @@
```

		+eclipse.preferences.version=1
		+org.eclipse.jdt.core.compiler.codegen.inlineJsrBytecode=enabled
		+org.eclipse.jdt.core.compiler.codegen.targetPlatform=1.8
		+org.eclipse.jdt.core.compiler.compliance=1.8
		+org.eclipse.jdt.core.compiler.problem.assertIdentifier=error
		+org.eclipse.jdt.core.compiler.problem.enumIdentifier=error
		+org.eclipse.jdt.core.compiler.problem.forbiddenReference=warning
		+org.eclipse.jdt.core.compiler.source=1.8

View

4 ΣΣΣΣΣ HelloSpring/.settings/org.eclipse.m2e.core.prefs

		@@ -0,0 +1,4 @@
		+activeProfiles=
		+eclipse.preferences.version=1
		+resolveWorkspaceProjects=true
		+version=1

View

10 ΣΣΣΣΣ HelloSpring/.settings/org.eclipse.wst.common.component

		@@ -0,0 +1,10 @@
		+<?xml version="1.0" encoding="UTF-8"?><project-modules id="moduleCoreId" project-version=
		+ <wb-module deploy-name="HelloSpring">
		+ <wb-resource deploy-path="/" source-path="/target/m2e-wtp/web-resources"/>
		+ <wb-resource deploy-path="/" source-path="/src/main/webapp" tag="defaultRootSou

```
+            <wb-resource deploy-path="/WEB-INF/classes" source-path="/src/main/java"/>
+            <wb-resource deploy-path="/WEB-INF/classes" source-path="/src/main/resources"/>
+            <property name="context-root" value="HelloSpring"/>
+            <property name="java-output-path" value="/HelloSpring/build/classes"/>
+      </wb-module>
+</project-modules>
```

View

10 ΣΣΣΣΣ HelloSpring/.settings/org.eclipse.wst.common.project.facet.core.xml

```
@@ -0,0 +1,10 @@
+<?xml version="1.0" encoding="UTF-8"?>
+<faceted-project>
+   <runtime name="Pivotal tc Server Developer Edition (Runtime) v3.1"/>
+   <fixed facet="jst.web"/>
+   <fixed facet="java"/>
+   <fixed facet="wst.jsdt.web"/>
+   <installed facet="java" version="1.8"/>
+   <installed facet="jst.web" version="3.1"/>
+   <installed facet="wst.jsdt.web" version="1.0"/>
+</faceted-project>
```

View

1 ΣΣΣΣΣ HelloSpring/.settings/org.eclipse.wst.jsdt.ui.superType.container

```
@@ -0,0 +1 @@
```

		+org.eclipse.wst.jsdt.launching.baseBrowserLibrary

View

1 ΣΣΣΣΣ HelloSpring/.settings/org.eclipse.wst.jsdt.ui.superType.name

		@@ -0,0 +1 @@
		+Window

View

2 ΣΣΣΣΣ HelloSpring/.settings/org.eclipse.wst.validation.prefs

		@@ -0,0 +1,2 @@
		+disabled=06target
		+eclipse.preferences.version=1

View

33 ΣΣΣΣΣ HelloSpring/pom.xml

		@@ -0,0 +1,33 @@
		+<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLS 4.0.0.xsd">
		+ <modelVersion>4.0.0</modelVersion>
		+ <groupId>HelloSpring</groupId>
		+ <artifactId>HelloSpring</artifactId>
		+ <version>0.0.1-SNAPSHOT</version>
		+ <packaging>war</packaging>
		+ <build>
		+ <plugins>

```
+            <plugin>
+                <artifactId>maven-compiler-plugin</artifactId>
+                <version>3.1</version>
+                <configuration>
+                    <source>1.8</source>
+                    <target>1.8</target>
+                </configuration>
+            </plugin>
+            <plugin>
+                <artifactId>maven-war-plugin</artifactId>
+                <version>2.4</version>
+                <configuration>
+                    <failOnMissingWebXml>false</failOnMissingWebXml>
+                </configuration>
+            </plugin>
+        </plugins>
+    </build>
+    <dependencies>
+            <dependency>
+                        <groupId>org.springframework</groupId>
+                        <artifactId>spring-webmvc</artifactId>
+                        <version>4.1.6.RELEASE</version>
+            </dependency>
```

```
+    </dependencies>
+</project>
```

View

15 ΣΣΣΣΣ HelloSpring/src/main/java/com/ktds/christof_kim/mvc/web/PrintHelloController.java

```
@@ -0,0 +1,15 @@
+package com.ktds.christof_kim.mvc.web;
+
+import org.springframework.stereotype.Controller;
+import org.springframework.web.bind.annotation.RequestMapping;
+import org.springframework.web.servlet.ModelAndView;
+
+@Controller
+public class PrintHelloController {
+	@RequestMapping("/hello")
+	public ModelAndViewhelloView(){
+		ModelAndView view = new ModelAndView();
+		view.setViewName("/hello");
+		return view;
+	}
+}
```

View

3 ΣΣΣΣΣ HelloSpring/src/main/webapp/META-INF/MANIFEST.MF

```
@@ -0,0 +1,3 @@
+Manifest-Version: 1.0
+Class-Path:
+
```

View

21 ΣΣΣΣΣ HelloSpring/src/main/webapp/WEB-INF/config/spring/dispatcherServlet.xml

```
@@ -0,0 +1,21 @@
+<?xml version="1.0" encoding="UTF-8"?>
+<beans          xmlns="http://www.springframework.org/schema/beans"
+               xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
+               xmlns:mvc="http://www.springframework.org/schema/mvc"
+               xsi:schemaLocation="http://www.springframework.org/schema/beans
+                                                                  http://www.springframework.org/sch
+                                                                  http://www.springframework.org/sch
+                                                                  http://www.springframework.org/sch
+
+<mvc:annotation-driven/>
+
+<bean id="viewResolver"
+               class="org.springframework.web.servlet.view.InternalResourceViewResolver">
+               <property name="prefix"  value="/WEB-INF/view/" />
+               <property name="suffix"  value=".jsp" />
```

```
+</bean>
+
+<!--TODO:Add Controller   -->
+<bean id="printHelloController"   class="com.ktds.christof_kim.mvc.web.PrintHelloController"/>
+
+</beans>
```

View

12 ΣΣΣΣΣ HelloSpring/src/main/webapp/WEB-INF/view/hello.jsp

```
@@ -0,0 +1,12 @@
+<%@ page language="java" contentType="text/html; charset=UTF-8"
+       pageEncoding="UTF-8"%>
+<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/htr
+<html>
+<head>
+<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
+<title>예제라우.</title>
+</head>
+<body>
+안녕여? ㅋㅋ
+</body>
+</html>
```

View

47 ΣΣΣΣΣ HelloSpring/src/main/webapp/WEB-INF/web.xml

```
@@ -0,0 +1,47 @@
+<?xml version="1.0" encoding="UTF-8"?>
+<web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns="http://xmlns.jcp.or
app_3_1.xsd" id="WebApp_ID" version="3.1">
+   <display-name>HelloSpring</display-name>
+   <welcome-file-list>
+      <welcome-file>index.html</welcome-file>
+      <welcome-file>index.htm</welcome-file>
+      <welcome-file>index.jsp</welcome-file>
+      <welcome-file>default.html</welcome-file>
+      <welcome-file>default.htm</welcome-file>
+      <welcome-file>default.jsp</welcome-file>
+   </welcome-file-list>
+
+   <servlet>
+         <servlet-name>dispatcherServlet</servlet-name>
+         <servlet-class>
+               org.springframework.web.servlet.DispatcherServlet
+         </servlet-class>
+         <init-param>
+               <param-name>contextConfigLocation</param-name>
+               <param-value>/WEB-INF/config/spring/dispatcherServlet.xml</param-value>
```

```
+		</init-param>
+	</servlet>
+
+	<servlet-mapping>
+		<servlet-name>dispatcherServlet</servlet-name>
+		<url-pattern>/</url-pattern>
+	</servlet-mapping>
+
+	<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>
		+
		+</web-app>
		+

View

1 ΣΣΣΣΣ HelloSpring/target/classes/.gitignore

		@@ -0,0 +1 @@
		+/com/

View

1 ΣΣΣΣΣ HelloSpring/target/m2e-wtp/web-resources/.gitignore

		@@ -0,0 +1 @@
		+/META-INF/

Please sign in to comment.

https://github.com/taehun3718/SpringPractice 에서 HelloSpring

Spring 에서 파라미터 전송

Spring Param 전송

본 포스팅은 KTDS Consotium 에서 배운 내용을 포스팅하였습니다.

문제가 있을 경우 바로 비공개 혹은 삭제처리하겠습니다.

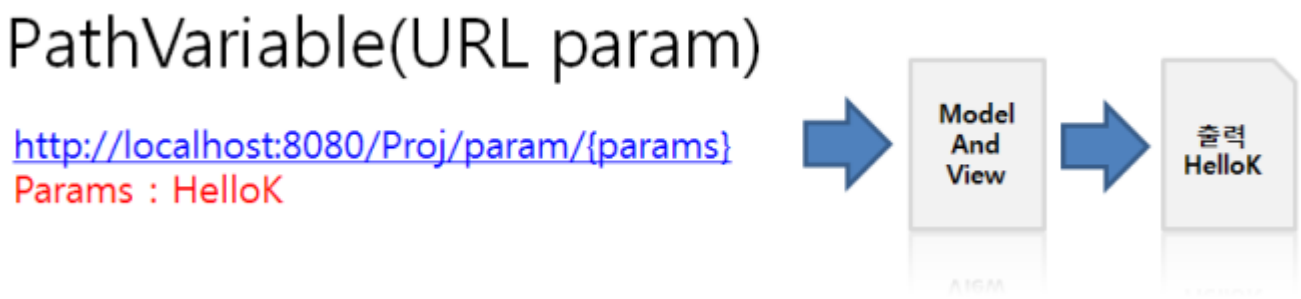
2015-04-20

파라미터를 전송 받는 방법은 크게 4 가지로 나뉩니다.

1. PathVariable(URL param) : URL를 통해서 데이터를 전송받는 방법입니다.
2. HttpServletRequestparam :HttpServletRequest의 request 객체를 이용하여 데이터를 전송받는 방법입니다.
3. @RequestMappingparam :HttpServlet과 비슷하게 request객체를 이용하는것처럼 보이지만, 이 방법은 애노테이션(@)를 이용하여 데이터를 전송받는 방법입니다.
4. Command object param : command 객체를 이용하여 데이터를 전송받는 방법입니다. 이 방법은 클래스 객체를 통해 데이터를 지정하고, 이를 전송 받게 됩니다.

대표적인 4 가지의 파라미터가 존재합니다. 이에 대해서 설명합니다.

1. PathVariable(URL param)



PathVariable 는 상위 그림과 같이 URL 에 직접 전송될 데이터를 적음으로써 데이터가 전송됩니다.

예를 들어 HelloK 라는 데이터를 전송할 경우

- http://localhost:8080/Proj/param/HelloK

이렇게 적을 경우 되면 실제 Html Document 출력에서는 HelloK 라는 데이터를 전송 받을 수 있습니다.

실제 ModelAndView 의 소스는 아래와 같습니다.

패키지 import 및 클래스 이름은 소스에서 생략합니다.

```
1        /**
2         * 실무에서 표준이 되다 싶은 파라미터 전송 방법. <br>
3         * URL에 직접 Id를 요청함으로써 파라미터를 전송할 수 있음<br>
4         * @paramparamId
5         * @return 뷰
6         */
7        @RequestMapping("/params/{paramId}")
8        publicModelAndViewexampleOf_URIParam(@PathVariableString paramId) {
9
10         ModelAndView view = newModelAndView();
11         view.setViewName("/paramViewOne");
12         view.addObject("welcomeMessage", paramId + "님 환영합니다.");
13
14         returnparamViewOne;
15       }
```

2. HttpServletRequestparam

HttpServletRequest 는 상위 그림과 같이 Param Key 값과 Value 를 전달하면

request.getParameter("id); 코딩을 통해 전송된 파라미터를 받을 수 있습니다.

request 는 GET method 및 POST method 을 통해서 데이터를 전송받을 수 있으며, 명시를 하지 않는 이상

Default 는 POST 혹은 GET method 로 데이터를 전송 받을 수 있습니다.

가령 상위 예제를 get method 로 받는다고 할 때

- http://localhost:8080/Proj/ExHttpServlet?id=kth

혹은 post method 로 받는다고 할 때

```
1        <formname="exampleTransmitForm"
2
3          id="exampleTransmitForm"
4
5          method="post"
6
7          action="http://localhost:8080/Proj/ExHttpServletPost>
8
9          <inputtype="hidden"name="id"value="kth"/>
10
11         <inputtype="submit"id="btnSubmit"value="전송!"/>
12
13       </form>
```

프로그램에서 받을 때에는 String id = request.getParameter("id); 로 받으면 되겠습니다.

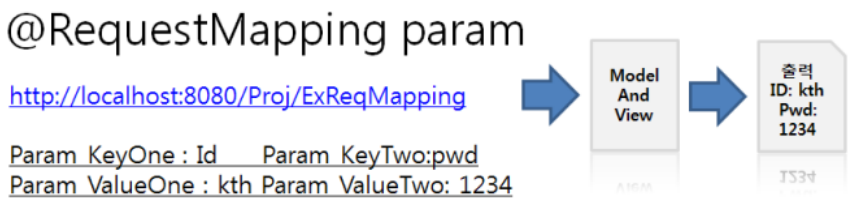
(예제에는 파라미터를 1 개만 넘겨줬지만 실제로는 1 개 이상의 다수의 파라미터를전송하는것이 가능합니다.)

```
1       /**
2        * HttpServletRequest파라메터를 이용하여 직접 id를 얻는 방법. get/post가능
3        * @param request
4        * @return paramViewTwo
5        */
6       @RequestMapping("/ExHttpServlet")
7       publicModelAndViewexampleOf_HttpServletRequest(HttpServletRequest request) {
8
9          String id = request.getParameter("id");
10
11         ModelAndView view = newModelAndView();
12         view.setViewName("/paramViewTwo");
13         view.addObject("userId", id);
14
15         returnview;
16      }
17      /**
18       * HttpServletRequest파라메터를 이용하여 직접 id를 얻는 방법. GET만가능
19       * @param request
20       * @return paramViewTwo
21       */
22      @RequestMapping(value="/paramHttpServletRequestGet", method=RequestMethod.GET)
```

```
23      publicModelAndViewexampleOf_HttpServletRequestGet(HttpServletRequest request) {
24
25          String id = request.getParameter("id");
26
27          ModelAndView view = newModelAndView();
28          view.setViewName("/paramViewTwo");
29          view.addObject("userId", id);
30
31          returnview;
32      }
33      /**
34       * HttpServletRequest파라메터를 이용하여 직접 id를 얻는 방법. POST만만가능
35       * @param request
36       * @return paramViewTwo
37       */
38      @RequestMapping(value="/paramHttpServletRequestPOST", method=RequestMethod.POST)
39      publicModelAndViewexampleOf_HttpServletRequestPOST(HttpServletRequest request) {
40
41          String id = request.getParameter("id");
42
43          ModelAndView view = newModelAndView();
44          view.setViewName("/paramViewTwo");
```

```
45          view.addObject("userId", id);
46
47          returnview;
48      }
```

3. RequestMappingparam



RequestMappingparam 은 상위 그림과 같이 Param Key 값과 Value 를 전달하면 전달된 파라미터를 그대로 받을 수 있습니다.

가령, @RequestParam("id") String id, @RequestParam("pwd")일 경우

```
1  <formname="exampleTransmitForm"
2
3   id="exampleTransmitForm"
4
5   method="post"
6
7   action="http://localhost:8080/Proj/exampleOf_RequestParamPost>
8
9   <inputtype="hidden"name="id"value="kth"/>
10
11 <inputtype="hidden"name="pwd"value="1234"/>
```

```
12

13 <inputtype="submit"id="btnSubmit"value="전송!"/>

14

15</form>
```

String myId = id;
String myPwd = pwd; 로 받을 수 있습니다.

GET method 및 POST method 을 통해서 데이터를 전송받을 수 있으며, 명시를 하지 않는 이상 Default 는 POST 혹은 GET method 로 데이터를 전송 받을 수 있습니다.

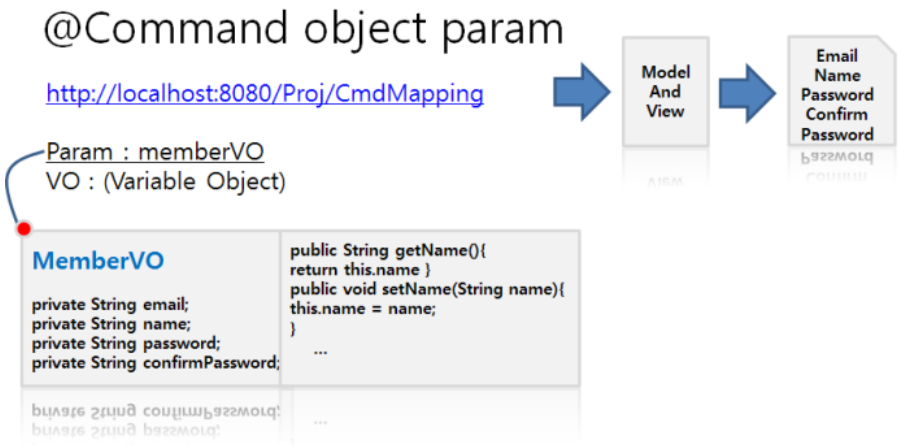
(1 개 이상의 다수의 파라미터를전송하는것이 가능합니다.)

```
1 /**

2  * RequestParam어노테이션을 이용하여 직접 id를 얻는 방법. get/post가능

3  * @return paramViewTwo

4  */

5 @RequestMapping("/exampleOf_RequestParam")

6 publicModelAndViewexampleOf_RequestParam(@RequestParam("id") String id) {

7

8  ModelAndView view = newModelAndView();

9  view.setViewName("/paramViewTwo");

10 view.addObject("userId", id);

11

12 returnview;

13}

14
```

```
15/**
16 * RequestParam어노테이션을 이용하여 직접 id pwd를 얻는 방법. POST만 가능
17 * @return paramViewTwo
18 */
19@RequestMapping(value="/exampleOf_RequestParamPost",
  method=RequestMethod.POST)
20
  publicModelAndViewexampleOf_RequestParamPost(@RequestParam("id") String id
21
   ,@RequestParam("pwd") String pwd) {
22

23
     ModelAndView view = newModelAndView();
24
     view.setViewName("/paramViewTwo");
25
     view.addObject("userId", id);
26
     view.addObject("pwd", pwd);
27

28
     returnview;
29
  }
30

31
  /**
32
   * RequestParam어노테이션을 이용하여 직접 id를 얻는 방법. GET만 가능
33
   * @return paramViewTwo
34
   */
35
  @RequestMapping(value="/exampleOf_RequestParamGet",
36
  method=RequestMethod.POST)
```

```
37publicModelAndViewexampleOf_RequestParamGet(@RequestParam("id") String id
38 ,@RequestParam("pwd") String pwd) {
39
40    ModelAndView view = newModelAndView();
41    view.setViewName("/paramViewTwo");
42    view.addObject("userId", id);
43    view.addObject("pwd", pwd);
44
45    returnview;
   }
```

4. Command object param



Command object param 은 상위 그림과 같이 객체(object)에 데이터를 넣어 전송됩니다.

사용자가 email, name, 패스워드에 대한 데이터를 전송할 경우

MemberVO 객체에 데이터를 넣고 전송합니다.

객체를 전달하기 위해서는 SpringFramework 에서 제공하는 form 테그를 이용합니다.

(이 SpringFramework 는 객체 Validation check 를 지원하기 때문에 사용하였습니다.>

테그라이브러리 부가적 요구사항

- <%@ taglib prefix="form" uri="http://www.springframework.org/tags/form” %>

```
1  <form:formcommandName="memberRegisterVO"
2
3  method="post"
4
5  action="http://localhost:8080/Proj/paramHttpServletRequest>
6
7  email:<inputtype="text"name="email"value="${memberRegisterVO.email }"/><br/>
8  name:<inputtype="text"name="name"value="${memberRegisterVO.name}"/><br/>
9  password:<inputtype="text"name="password"value="${memberRegisterVO.password }"/><br/>
10confirmPassword:<inputtype="text"name="confirmPassword"
11
12value="${memberRegisterVO.confirmPassword }"/><br/>
13<inputtype="submit"id="btnSubmit"value="전송!"/>
14
15</form:form>
```

<commandName 은 VO 클래스의 이름입니다. 관례로 이름은 같게 적습니다. 가령 MemberRegisterVO 는 memberRegisterVO 로 적습니다.>

Command 객체는 <form>테그로 전송해도 됩니다.

다만 받을 때 객체의 name 은 Command 객체의 name 과 동일해야 들어갑니다.

예를 들어

classMemberVO {

private String name;

private String pwd;

}

일 경우 form 테그에 전송되는 데이터는

<form>

<input type="text" name="name"/>

<input type="text" name="pwd"/>

</form>

로 클래스의 멤버변수는 name 과 동일하게 해야 합니다.

```
1    //실제 데이터를 전송하기 위한 폼을 보여주려는 뷰
2    @RequestMapping("/transmitDataEx03")
3       publicModelAndViewtransmitDataViewThree() {
4       ModelAndView view = newModelAndView();
5       view.setViewName("/transmitDataToCommandObject");
6       returnview;
7     }
8
9    /**
10    * Command 객체를 이용해 폼 전송 처리
11    * @param id
12    * @return paramViewThree
13    */
14    @RequestMapping(value="/paramHttpServletRequest3", method=RequestMethod.POST)
```

```
15    publicModelAndViewexampleOf_CommandParam(MemberRegisterRequestVOmemberRegister
      VO) {
16
17
        ModelAndView view = newModelAndView();
18
        view.addObject("memberRegisterVO", memberRegisterVO);
        view.setViewName("/paramViewThree");


        returnview;
      }
```

5. Command object Validation Check

사용자는 email, name, 패스워드에 대한 필드를 모두 채우지 않고 등록를 합니다.

이는 Null Point Exception 등 보안에 대해서 해킹을 당할 수 있기 때문에 Validation Check 은 필수입니다.

Command 객체에 대해서 Validation Check 를 하기 위해서는 아래와 같은 패키지를 요구합니다.

pom.xml

- hibernate-validator

스프링에서는 아래와 같은 Validation Check 을 제공합니다.

```
  @RequestMapping(value="/paramHttpServletRequestValidChk",
1 method=RequestMethod.POST)
2    publicModelAndViewexampleOf_CommandParamValidCheck(
3    @ValidMemberRegisterRequestVOmemberRegisterVO, Errors errors) {
4      ModelAndView view = newModelAndView();
5
```

```
6       if(errors.hasErrors()){
7      //에러가 생겼을 경우, 처리.
8          System.out.println(errors.getAllErrors());
9          System.out.println("error");
10         view.setViewName("/transmitDataToCommandObject");
11         view.addObject("memberRegisterVO", memberRegisterVO);
12         returnview;
13     } else{
14         view.setViewName("/paramViewThree"); //      /view/paramViewThree.jsp
15         view.addObject("memberRegisterVO", memberRegisterVO);
16         returnview;
17     }
18 }
19
20//paramViewThree.jsp
21<form:formcommandName="memberRegisterRequestVO"
22            method="post"
23            action="/ExampleOfParamTransmit/paramHttpServletRequestValidChk">
24   <table>
25     <tr>
26       <td>email:</td>
27       <td>
```

```
28          <input type="text"
29                  name="email"
30                  value="${memberRegisterVO.email }"/>
31          <form:errors path="email"></form:errors>
32      </td>
33    </tr>
34
35    <tr>
36      <td>name:</td>
37      <td>
38          <input type="text"
39                  name="name"
40                  value="${memberRegisterVO.name}"/>
41          <form:errors path="name"></form:errors>
42      </td>
43  </tr>
44
45  <tr>
46      <td>password:</td>
47      <td>
48          <input type="password"
49                  name="password"
```

```
50                 value="${memberRegisterVO.password }"/>
51           <form:errors path="password"></form:errors>
52       </td>
53  </tr>
54    <tr>
55        <td>confirmPassword:</td>
56        <td>
57           <input type="password"
58                  name="confirmPassword"
59                  value="${memberRegisterVO.confirmPassword }"/>
60          <form:errors path="confirmPassword"></form:errors>
61       </td>
62    </tr>
63  </table>
64<input type="submit"value="전송!"/>
65</form:form>
```

실행 화면

커맨드 객체를 이용한 데이터 전송 예제

email: dasd
name:
password:
confirmPassword:
전송!

Validation Check
email: dasd email을 입력해주세요.
name: 이름을 입력해주세요.
password: 패스워드를 입력해주세요.
confirmPassword: 패스워드를 입력해주세요.
전송!