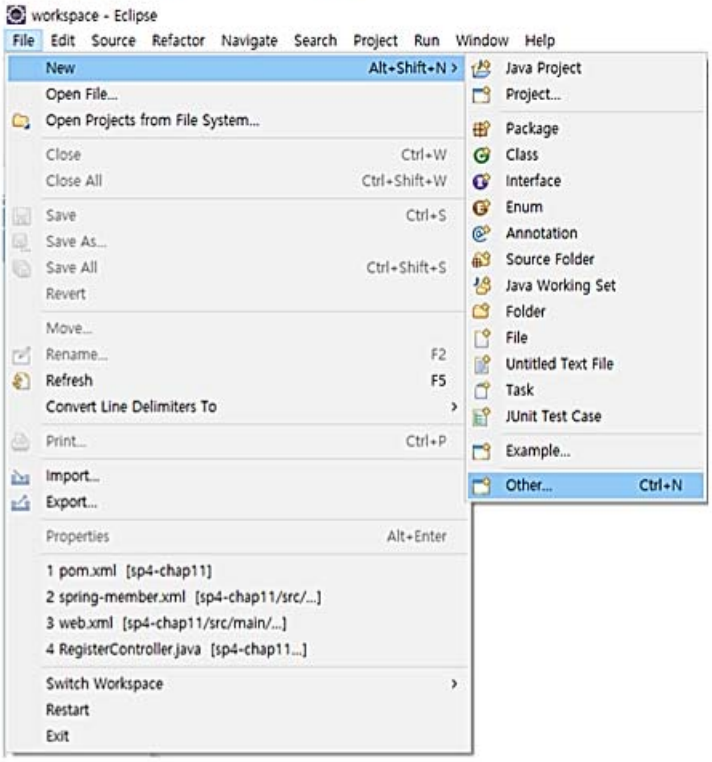
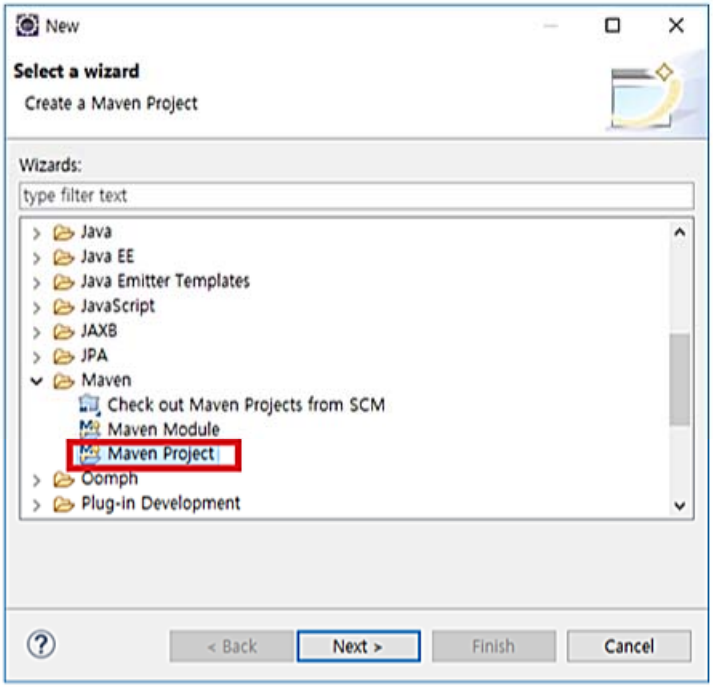
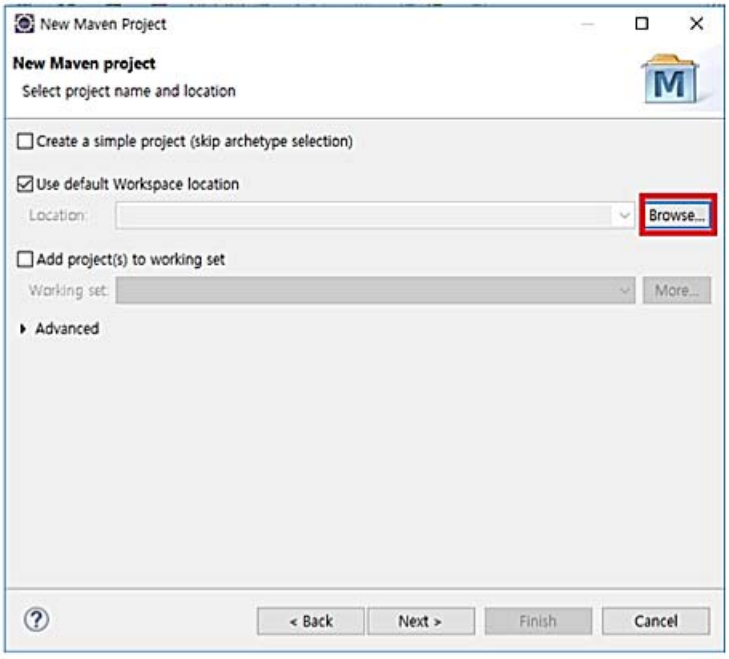
00.maven 웹 프로젝트 만들기



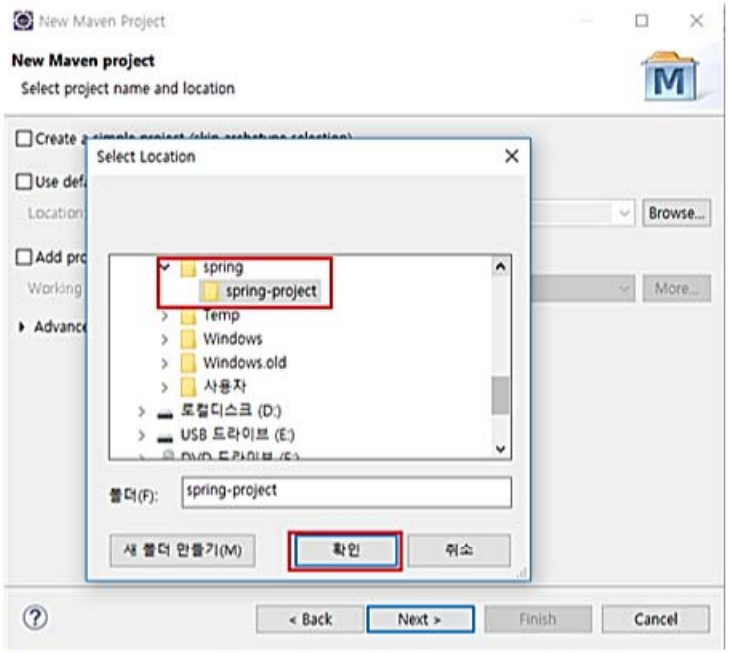
스프링으로 웹 프로젝트를 만들기 위해서는 [파일] - > [NEW] -> [Other]를 클릭한다.



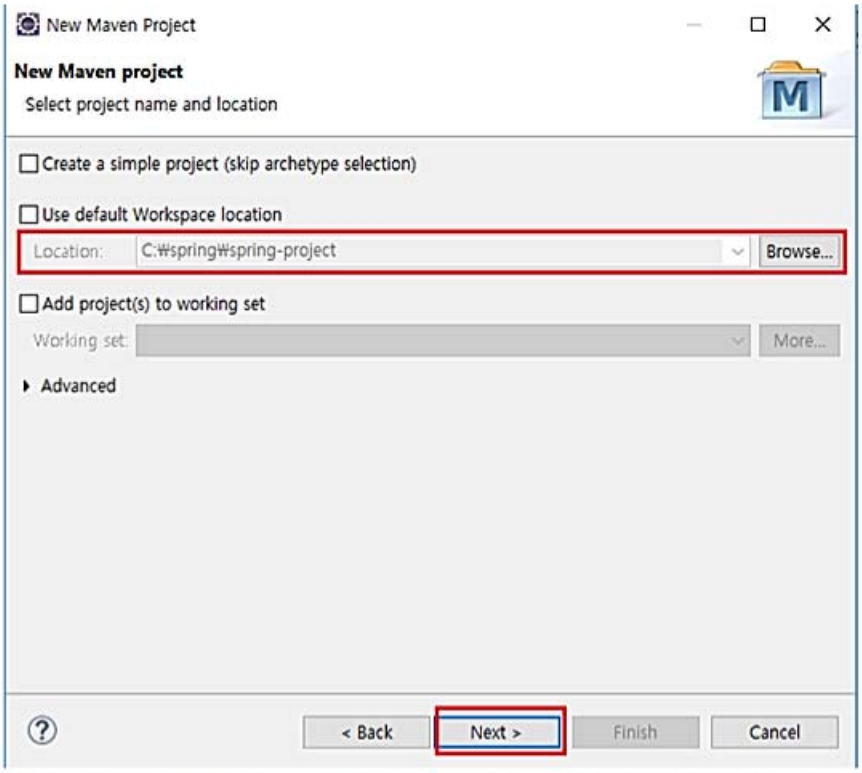
[NEW]창에서 Maven => Maven Project를 선택한다.



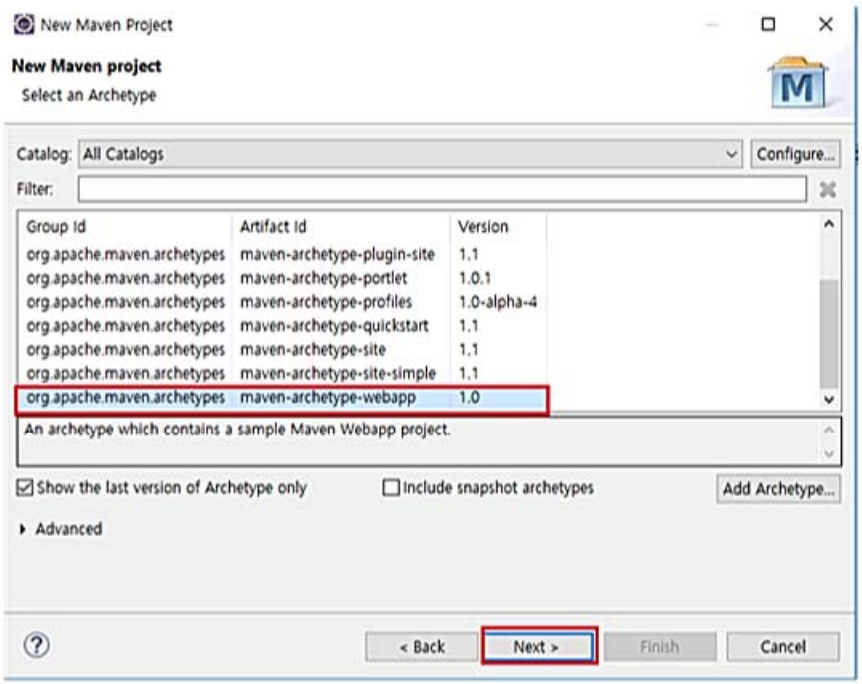
[NEW Maven Project]창에서 "Use default Workspace location"의 "Browse..."를 클릭한다.



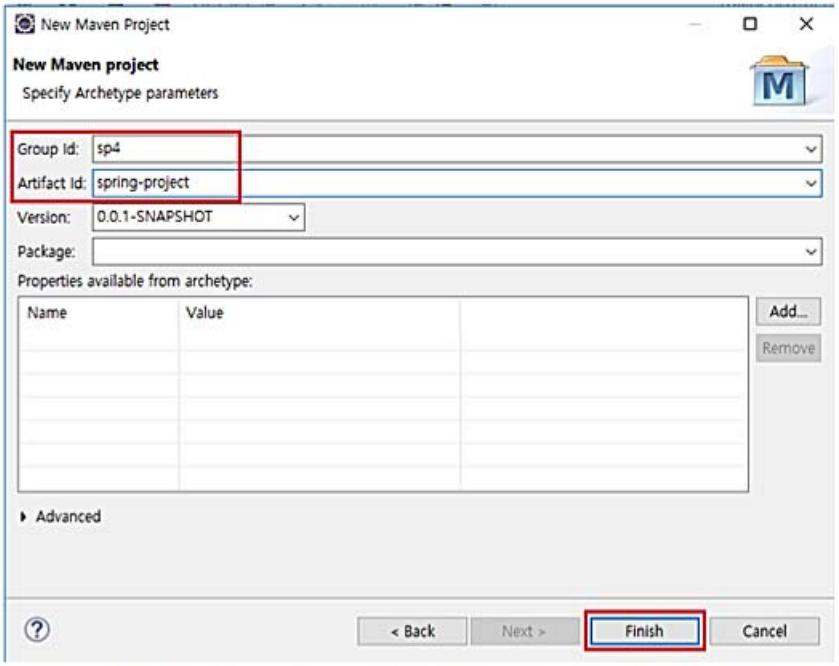
[Select Location]창에서 프로젝트를 할 폴더를 선택한다.
폴더를 새로 만들려면 새 폴더를 만들 폴더를 선택한 후 "새 폴더 만들기"를 클릭한 후 폴더 이름을 지정하고 선택한 후 [확인]버튼을 클릭한다.



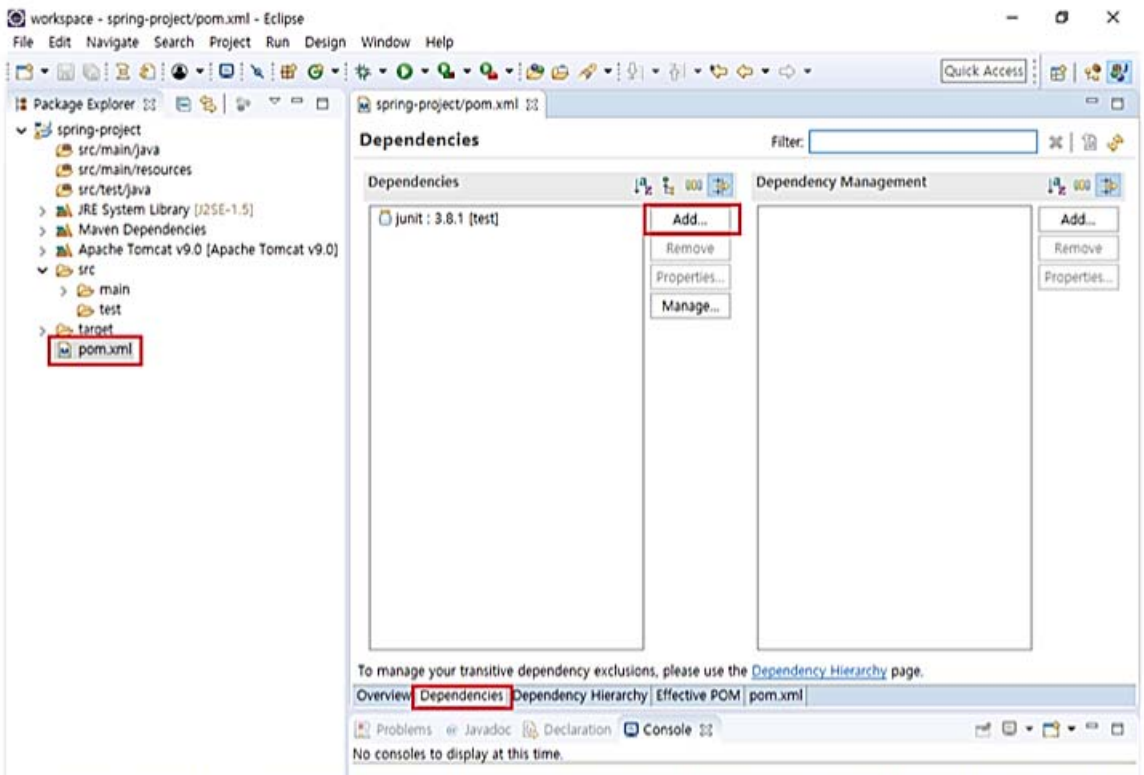
"Use default Workspace location"에 경로가 지정되었다면 [Next]를 클릭한다.



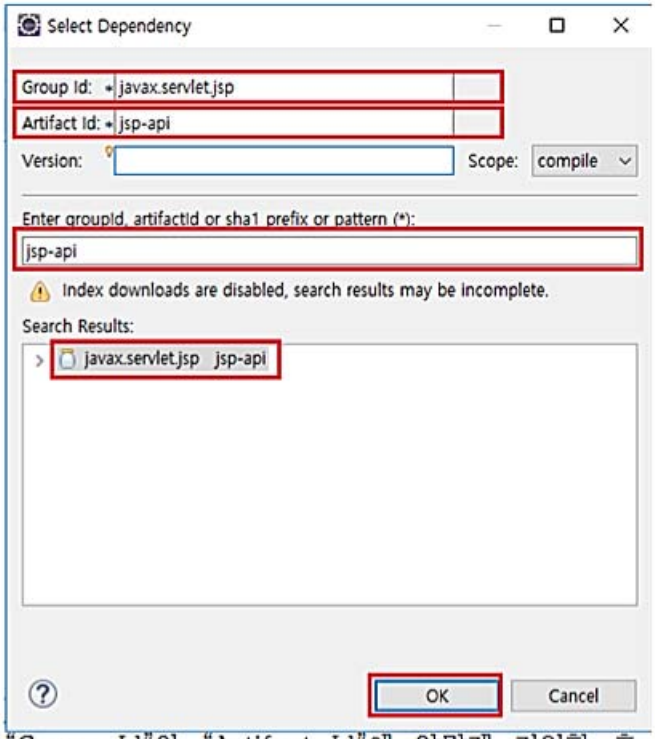
"Select an Archetype"에서 "Artifact Id"가 "Maven-archetype-webapp"를 선택한 후 [Next]를 클릭한다.



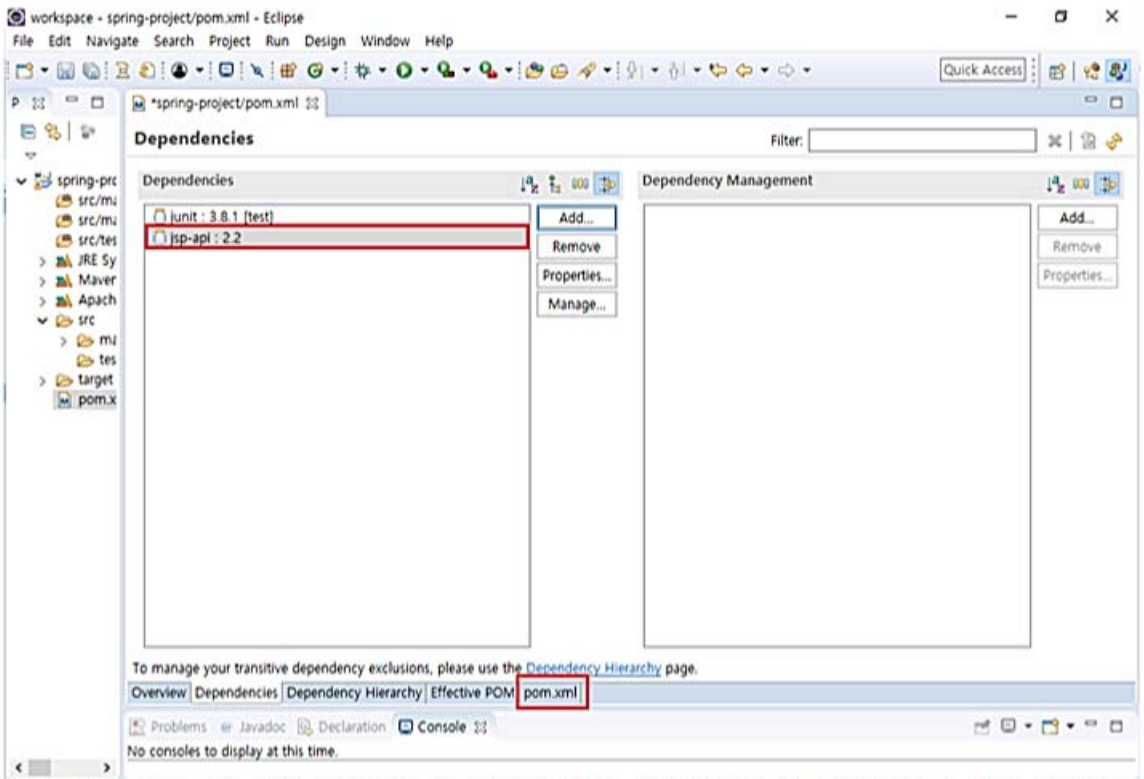
"Group Id"에 알맞은 이름을 지정해주고 "Artifact Id"에 프로젝트명을 기재해 주면 된다.



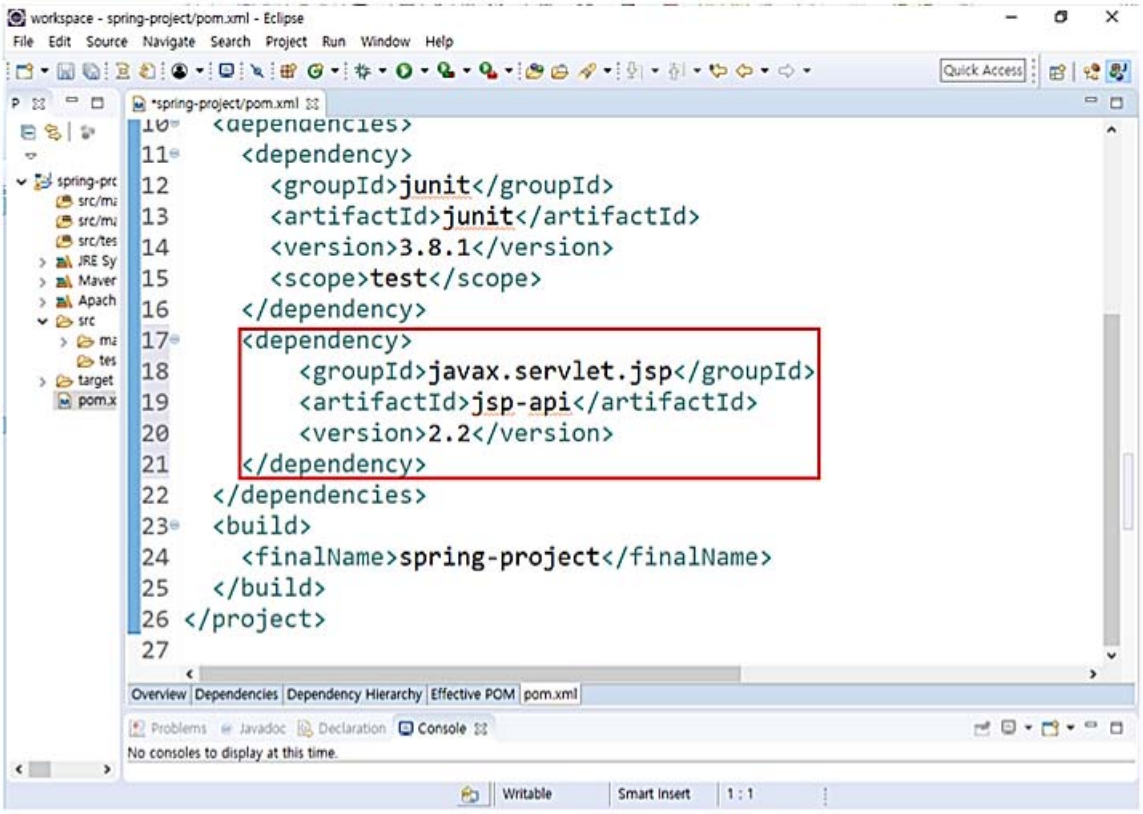
프로젝트 창에서 "pom.xml"을 더블클릭하면 오른 쪽창이 뜨면 오른쪽 창에 있는 하단 "Dependencies"를 클릭하고 오른쪽 창 상단 중앙에 있는 [Add]버튼을 클릭한다.



"Group Id"와 "Artifact Id"에 알맞게 기입한 후 "Enter groupId, artifactId or sha1 prefix or pattern(*);"에 "Group Id"또는 "Artifact Id"를 기입한 후 "Search Results"에 리스트가 뜨면 최신버전이나 필요한 항목을 선택 후 [OK]버튼을 클릭한다.



위와 같이 추가 된 것을 확인 할 수 있으며 계속 추가하려면 [Add]버튼을 눌러서 반복하면 된다.


workspace - spring-project/pom.xml - Eclipse
File Edit Source Navigate Search Project Run Window Help
Quick Access
*spring-project/pom.xml
<dependencies>
<dependency>
<groupId>junit</groupId>
<artifactId>junit</artifactId>
<version>3.8.1</version>
<scope>test</scope>
</dependency>
<dependency>
<groupId>javax.servlet.jsp</groupId>
<artifactId>jsp-api</artifactId>
<version>2.2</version>
</dependency>
</dependencies>
<build>
<finalName>spring-project</finalName>
</build>
</project>
Overview Dependencies Dependency Hierarchy Effective POM pom.xml
Problems Javadoc Declaration Console
No consoles to display at this time.
Writable Smart Insert 1 : 1

01. 프로젝트 생성

- src/main/java
- src/main/resources
- src/main/webapp : 웹 어플리케이션을 구현할 때 필요한 코드가 위치하는 곳
- src/main/webapp/WEB-INF : web.xml이 위치하는 곳
- src/main/webapp/WEB-INF/view

스프링 MVC 작업순서

1. DispatchServlet 설정 및 스프링 칸텍스트 설정
2. 컨트롤러 구현 및 설정추가
3. 설정파일에 ViewResolver 설정추가
4. 뷰 코드 구현
5. 실행

예제 1) sp4-chap09/pom.xml

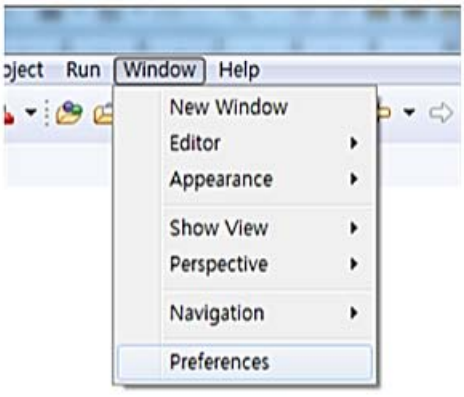
```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
                http://maven.apache.org/xsd/maven-4.0.0.xsd">
        <modelVersion>4.0.0</modelVersion>
        <groupId>sp4</groupId>
        <artifactId>sp4-chap09</artifactId>
        <version>0.0.1-SNAPSHOT</version>
        <packaging>war</packaging>

        <dependencies>
                <dependency>  <!-- 13행 -->
                        <groupId>javax.servlet.jsp</groupId>
                        <artifactId>jsp-api</artifactId>
                        <version>2.2</version>
                        <scope>provided</scope>
                </dependency>
                <dependency>
                        <groupId>javax.servlet</groupId>
                        <artifactId>javax.servlet-api</artifactId>
                        <version>3.0.1</version>
                        <scope>provided</scope>
                </dependency>
                <dependency>
                        <groupId>org.springframework</groupId>
```

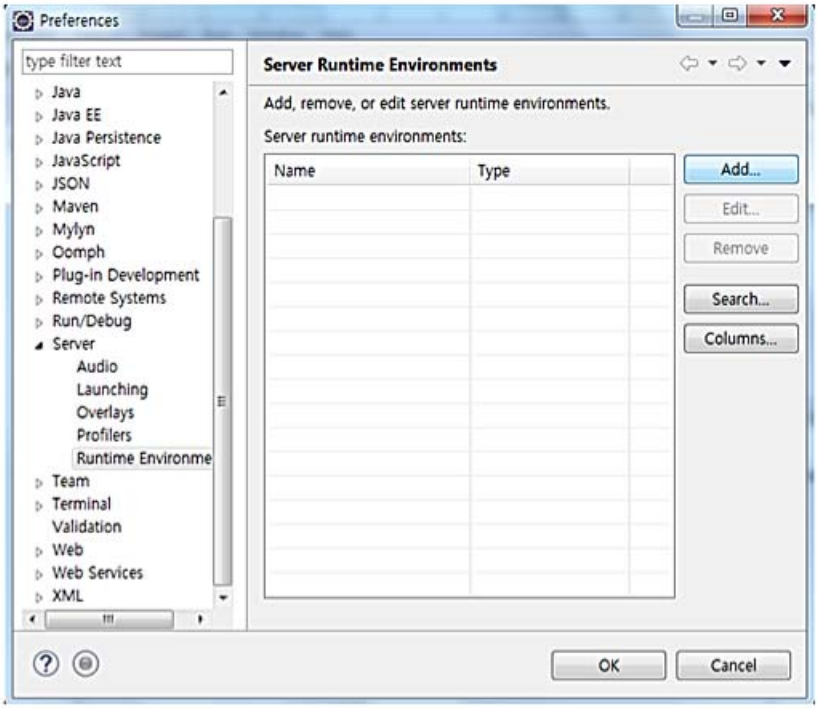
```
                <artifactId>spring-webmvc</artifactId>
                <version>4.1.0.RELEASE</version>
            </dependency> <!-- 29행 -->
        </dependencies>

        <build>
            <plugins>
                <plugin>
                    <artifactId>maven-compiler-plugin</artifactId>
                    <version>3.1</version>
                    <configuration>
                        <source>1.7</source>
                        <target>1.7</target>
                        <encoding>utf-8</encoding>
                    </configuration>
                </plugin>
            </plugins>
        </build>

</project>
```

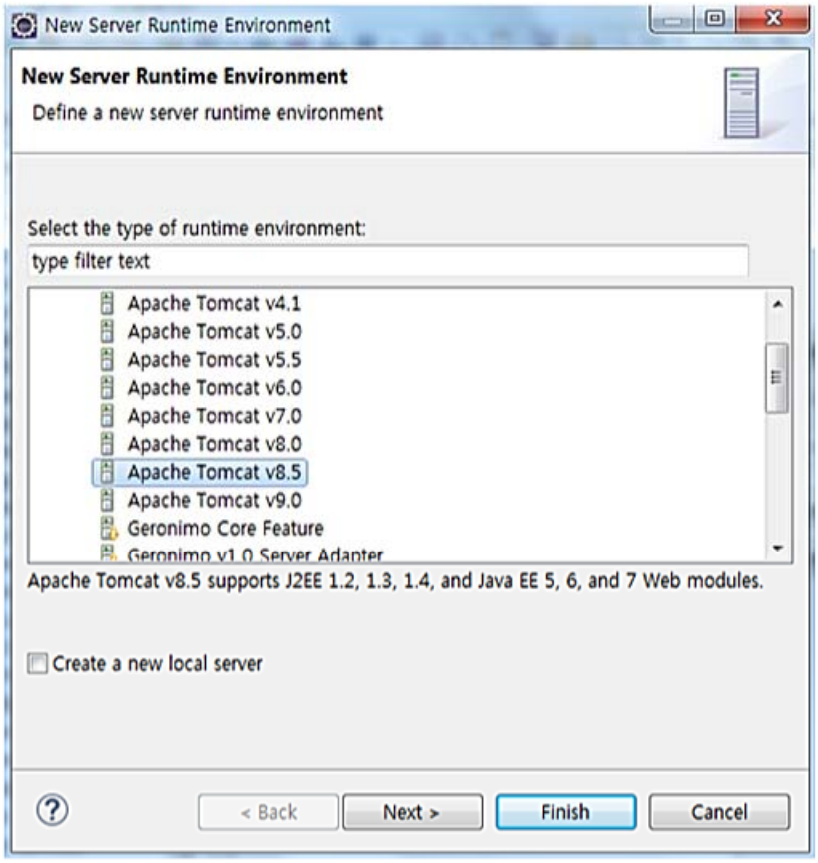
13~29행 : 스프링을 이용해서 웹 어플리케이션 개발하는데 필요한 의존 설정
JSP2.2와 서블릿3.0dp 대한 의존을 추가하고, 스프링 MVC를 사용하기 위해 spring-webmvc모듈에 대한 의존을 추가한다.

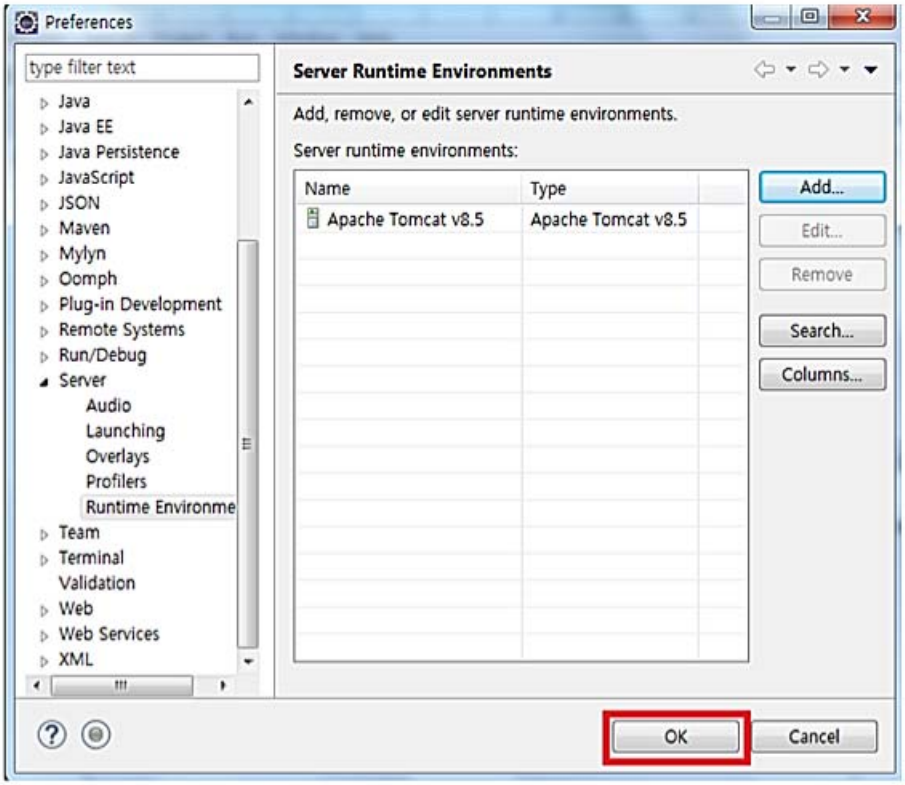
02. 이클립스 톰캣 설정

- [Window]-[Preference] 메뉴 실행
- Server/Runtime Environments 선택
- [Add]버튼을 눌러 톰캣 서버를 등록한다.




Preferences
type filter text
Java
Java EE
Java Persistence
JavaScript
JSON
Maven
Mylyn
Oomph
Plug-In Development
Remote Systems
Run/Debug
Server
Audio
Launching
Overlays
Profilers
Runtime Environme
Team
Terminal
Validation
Web
Web Services
XML
Server Runtime Environments
Add, remove, or edit server runtime environments.
Server runtime environments:
Name
Type
Add...
Edit...
Remove
Search...
Columns...
OK
Cancel


New Server Runtime Environment
New Server Runtime Environment
Define a new server runtime environment
Select the type of runtime environment:
type filter text
Apache Tomcat v4.1
Apache Tomcat v5.0
Apache Tomcat v5.5
Apache Tomcat v6.0
Apache Tomcat v7.0
Apache Tomcat v8.0
Apache Tomcat v8.5
Apache Tomcat v9.0
Geronimo Core Feature
Geronimo v1.0 Server Adapter
Apache Tomcat v8.5 supports J2EE 1.2, 1.3, 1.4, and Java EE 5, 6, and 7 Web modules.
Create a new local server
< Back
Next >
Finish
Cancel



03. 스프링 MVC를 위한 설정

- 스프링 MVC의 주요 설정(HanderMapping, ViewResolver 등)
- 스프링의 DispatcherServlet 설정

3.1 스프링 MVC설정

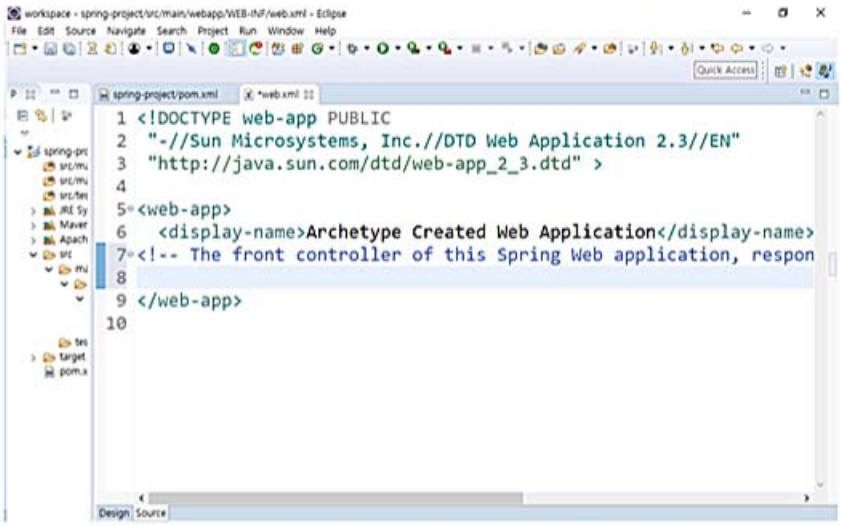
예제 2) src\main\resources\spring-mvc.xml

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

<beans xmlns="http://www.springframework.org/schema/beans"
        xmlns:mvc="http://www.springframework.org/schema/mvc"
        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
                http://www.springframework.org/schema/mvc
                http://www.springframework.org/schema/mvc/spring-mvc.xsd">

        <mvc:annotation-driven />              //11행

        <mvc:default-servlet-handler />        // 13행
```

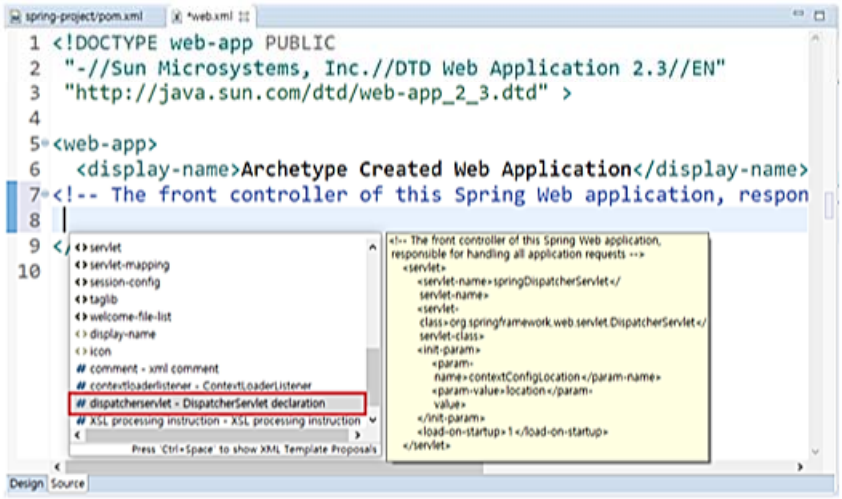
```
        <mvc:view-resolvers>
                <mvc:jsp prefix="/WEB-INF/view/" />    // 16행
        </mvc:view-resolvers>

</beans>
```

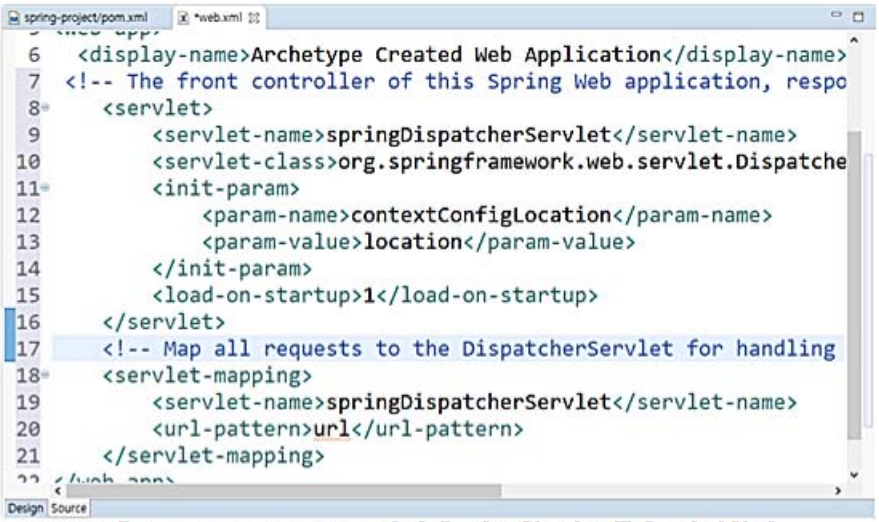
11행 : @Controller 애노테이션을 이용한 컨트롤러를 사용하기 위한 설정.
13행 : Dispatcherservlet의 매핑경로를 '/'로 주었을 때, JSP/HTML/CSS등을 올바르게 처리하기 위한 설정
16행 : JSP를 이용해서 컨트롤러의 실행결과를 보여주기 위한 설정
<mvc:annotation-driven />은 내부적으로 다양한 빈 설정을 추가해주기 위해서 사용한다.

3.2 web.xml 파일에 DispatcherServlet 설정
스프링 MVC가 웹 요청을 처리하면 DispatcherServlet을 통해서 웹 요청을 받아야한다.
이를 위해 web.xml파일에 DispatcherServlet을 등록한다.



DispatcherServlet 설정하고자 하는 자리에 커서를 놓고 ctrl+space를 누른다.





web.xml에 DispatcherServlet설정에 필요한 정보들을 기입한다.

예제 3) src\main\webapp\WEB-INF\web.xml

ctrl+space를 누른다.

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xmlns="http://java.sun.com/xml/ns/javaee"
        xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
                http://java.sun.com/xml/ns/javaee/web-app_3_0.xsd"
        version="3.0">

        <servlet>
                <servlet-name>dispatcher</servlet-name>
                <servlet-class>
                        org.springframework.web.servlet.DispatcherServlet    //11행
                </servlet-class>
                <init-param>
                        <param-name>contextConfigLocation</param-name>  14행
                        <param-value>
                                classpath:spring-mvc.xml
                                classpath:spring-controller.xml
                        </param-value>
                </init-param>
                <load-on-startup>1</load-on-startup>
        </servlet>

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

08~21행 : DispatcherServlet을 등록하는데, 각 행은 다음의 의미를 갖는다.
09~11행 : DispatcherServlet을 Dispatcher라는 이름으로 등록한다.
13행 ~ 18행 : contextConfigLocation 초기화 파라미터의 값을 지정한다. 이파라미터에는 스프링 설정 파일의 목록을 지정한다. 각 설정 파일의 경로는 별도 줄로 구분하거나 콤마로 구분한다.
20행 : 톰캣과 같은 컨테이너가 웹 어플리케이션을 구동할 때 이 서블릿을 함께 실행하도록 설정한다.

DispatcherServlet은 초기화 과정에서 contextConfigLocation 초가화 파라미터에 지정한 설정 파일을 이용해서 스프링 컨테이너를 초기화한다.
위 예제의 설정은 클래스패스에 위치한 spring-mvc.xml 파일과 spring-controller.xml 파일을 이용해서 스프링 컨테이너를 생성한다.

23~26행 : 모든 요청을 DispatcherServlet이 처리하도록 서블릿 매핑을 설정했다.
28~41행 : HTTP요청 파라미터의 인코딩 처리를 위한 서블릿 필터를 등록한 것이다.
스프링은 인코딩 처리를 위한 필터인 CharacterEncodingFilter 클래스를 제공하고 있다.
33~36행 : encoding 초기화 파라미터를 이용해서 HTTP요청 파라미터를 읽어 올 때 사용할 인코딩을 지정한다.

04. 코드 구현

- 클라이언트의 요청을 알맞게 처리할 컨트롤러
- 처리결과를 보여줄 JSP

4.1 컨트롤러 구현

예제 4) src\main\java\chap09\HelloController.java

```
package chap09;

import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestParam;

@Controller                                                    // 8행
public class HelloController {

        @RequestMapping("/hello")                              // 11행
        public String hello(Model model,
            @RequestParam(value = "name", required = false) String name) {// 13행
                model.addAttribute("greeting", "안녕하세요, " + name);
                return "hello";
        }
}
```

08행 : @Controller 애노테이션을 적용한 클래스는 스프링 MVC에서 컨트롤러를 사용된다.
11행 : @RequestMapping 애노테이션은 메서드가 처리할 요청경로를 지정한다. 위 코드의 경우 "/hello"경로로 들어온 요청을 hello() 메서드를 이용해서 처리한다고 설정한다.
12행 : Model 파라미터 - 컨트롤러의 처리 결과를 뷰에 전달할 때 사용된다.
13행 : @RequestParam 애노테이션 - http 요청 파라미터 값을 메서드의 파라미터로 전달할 때 사용된다.
14행 : "greeting" 이라고 모델 속성에 값을 설정한다. 값으로는 "안녕하세요. "와 name 파라미터 값을 연결한 문자열을 사용한다. 뒤에서 작성한 JSP 코드는 이 속성을 이용해서 값을 사용한다.
15행 : 컨트롤러의 처리 결과를 보여줄 뷰의 이름으로 "hello"를 사용한다.

스프링 컨트롤러로 사용될 클래스는 @Controller 애노테이션을 가져야 하며, @RequestMapping 애노테이션과 요청 URL간의 관계, 그리고 @RequestParam 애노테이션과 요청 파라미터와의 관계는 아래와 같다.

http://localhost:8088/sp4-chap09/hello? name=bk

@RequestMapping("/hello")

```
        public String hello(Model model,
            @RequestParam(value = "name", required = false) String name) {
                model.addAttribute("greeting", "안녕하세요, " + name);
                return "hello";
        }
```

톰켓의 경우 webapp\sp4-chap09폴더는 웹브라우저에서 http://host/sp4-chap09 경로에 해당하는데, 이때 sp4-chap09가 컨텍스트 경로가 된다.
@RequestParam 애노테이션은 HTTP요청파라미터를 메서드의 파라미터로 바로 전달 받을 수 있게 해준다.
@RequestParam 애노테이션의 value속성은 HTTP요청 파라미터 이름을 , required속성은 필수여부를 지정한다.
name 요청파라미터의 값인 "bk"가 hello()메서드의 name파라미터에 전달된다.

Model객체의 addAttribute()메서드를 실행하고 있는데, 이는 뷰에 전달할 데이터를 지정하기 위해 사용된다.

```
model.addAttribute("greeting", "안녕하세요, " + name);
```

addAttribute()메서드의 첫 번째 파라미터는 데이터를 식별하는데 사용되는 속성이름이고 두 번째 파라미터는 속성 이름에 해당하는 값이다.

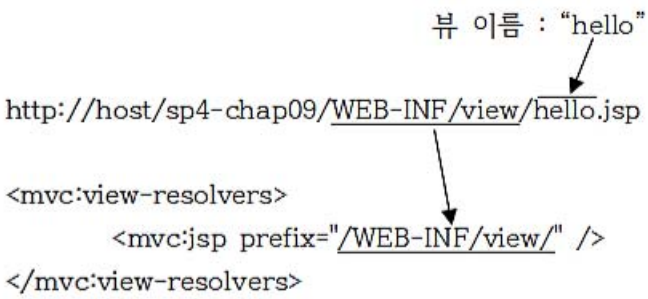
예제 5) 컨트롤러를 스프링 빈에 등록한다. src\main\resources\spring-controller.xml

```
<?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">

        <bean class="chap09.HelloController" />

</beans>
```

HollowController의 hello()메서드가 리턴한 뷰 이름은 "hello"였는데, JSP파일의 이름을 보면 "hello.jsp"이다. 뷰의 이름과 JSP 파일과의 연결은 spring-mvc.xml 파일을 통해 이루어진다.

```
        <mvc:view-resolvers>
                <mvc:jsp prefix="/WEB-INF/view/" />
        </mvc:view-resolvers>
```

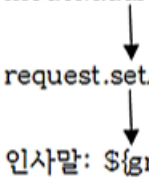
<mvc:jsp>는 JSP구현을 뷰 구현으로 사용할 수 있도록 해주는 설정이다.
prefix속성은 JSP경로를 찾을 때 사용할 접두어다. 별도의 설정이 없는 경우 접미사로 “.JSP"를 사용한다.



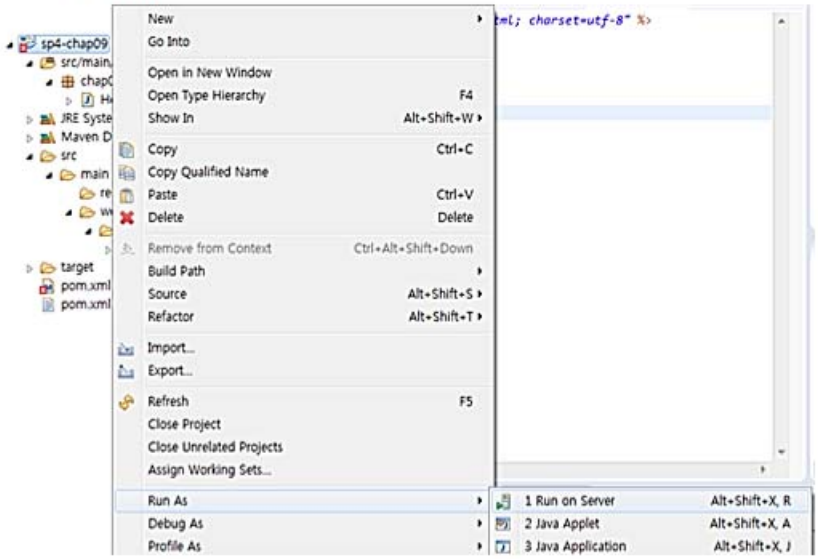
예제 6) JSP구현
컨트롤러가 생성한 결과를 뷰 코드를 만든다.
뷰 코드는 JSP를 이용해서 구현한다.
src\main\webapp\WEB-INF 밑에 view폴더를 만든다.
src\main\webapp\WEB-INF\view\hello.jsp

```
<%@ page contentType="text/html; charset=utf-8" %>
<!DOCTYPE html>
<html>
<head>
<title>Hello</title>
</head>
<body>
인사말: ${greeting}
</body>
</html>
```

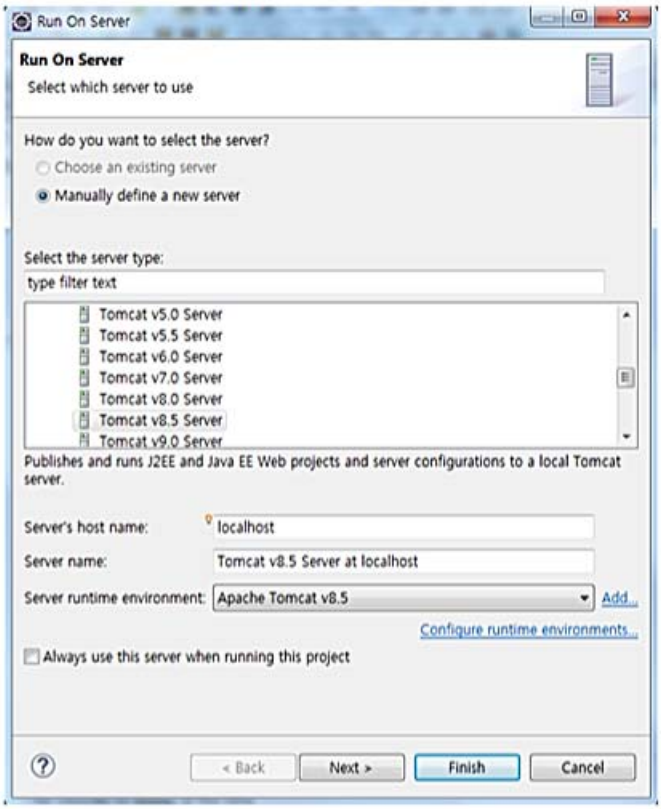
인사말: ${greeting}
이표현식의 “greeting”은 컨트롤러 구현에서 Model에 추가한 속성의 이름인 “greeting”과 동일하다. 모델에 추가한 속성을 JSP코드에서 접근할 수 있게 HttpServletRequest에 옮겨준다.



05. 실행하기

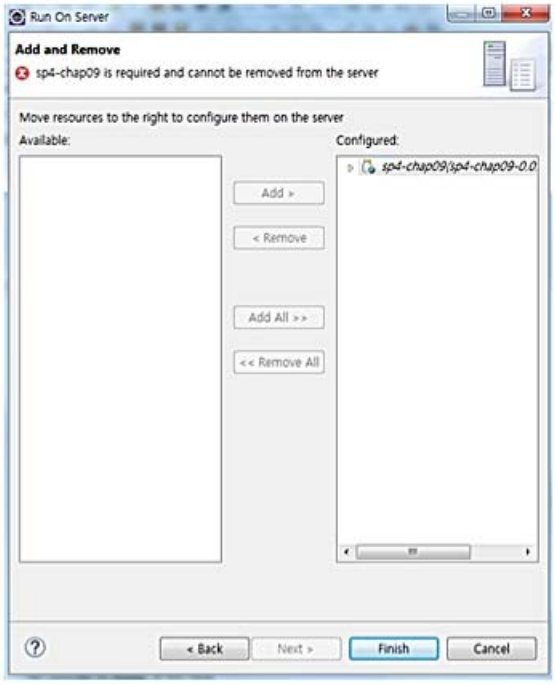


[Run on Server]메뉴를 실행한다.



최초에 서버를 선택할 때에는 "Manually define a new server"처럼 신규 서버를 정의하는 메뉴가 선택되며, 두 번째부터는 "Choose an existing server"로 기존에 사용한 서버를 선택할 수 있다.

[NEXT>]를 누른다.



'Configured' 목록에 'sp4-chap09'프로젝트를 위치시킨다.
다른 프로젝트가 'Configured'목록에 보인다면 [Remove]버튼을 사용해거 프로젝트를 왼쪽 'Available'로 이동시킨다.

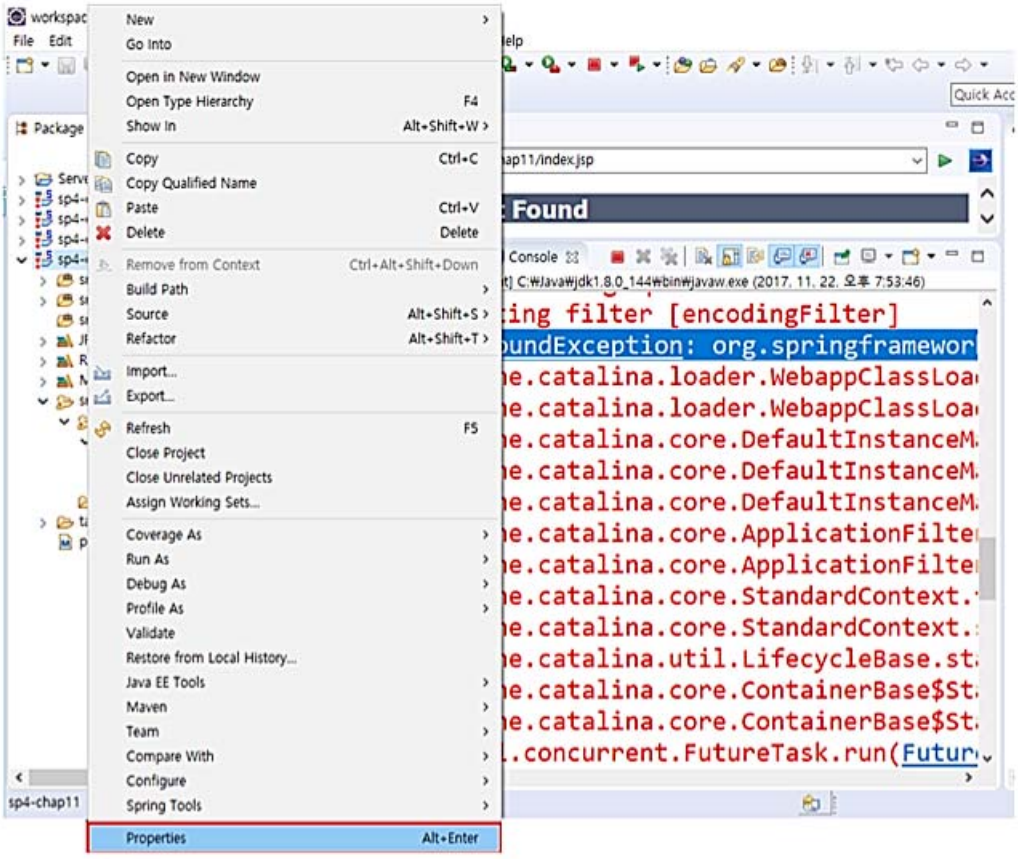
[Finish]버튼을 누른다.

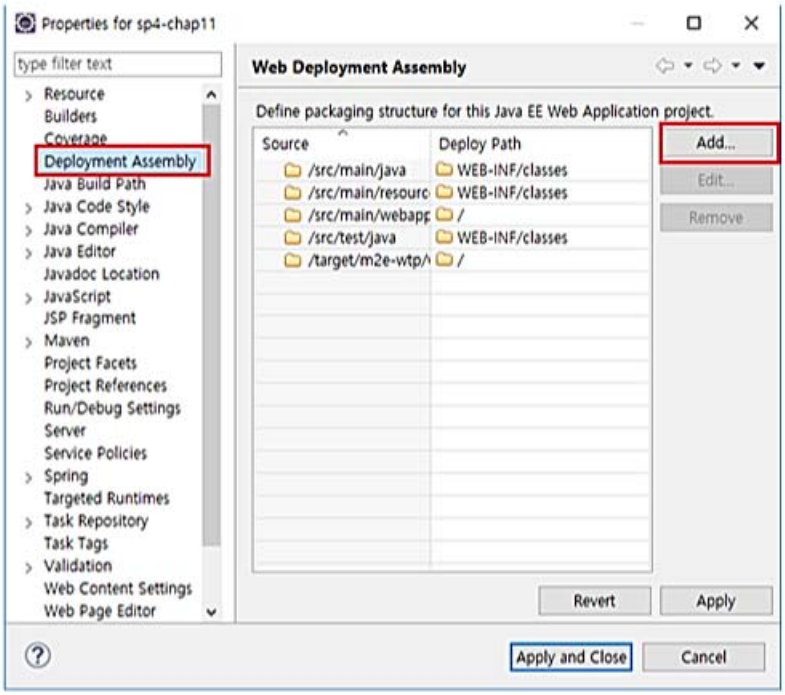
주소를 입력
http://localhost:8088/sp4-chap09/hello?name=bk

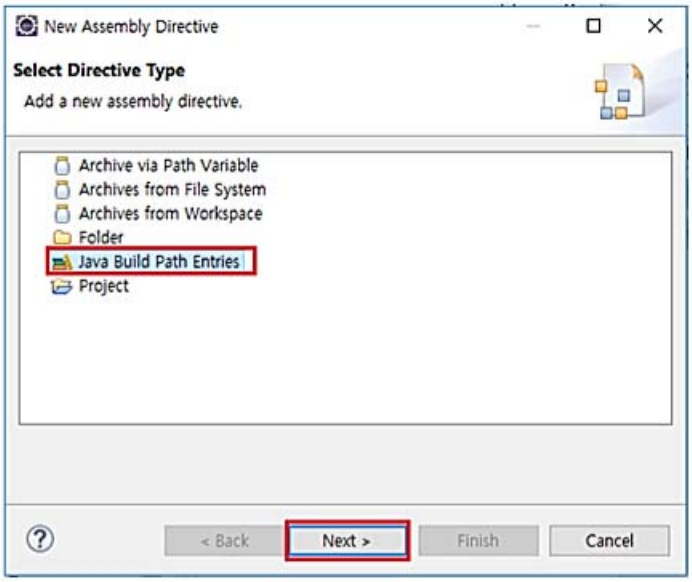
java.lang.ClassNotFoundException:

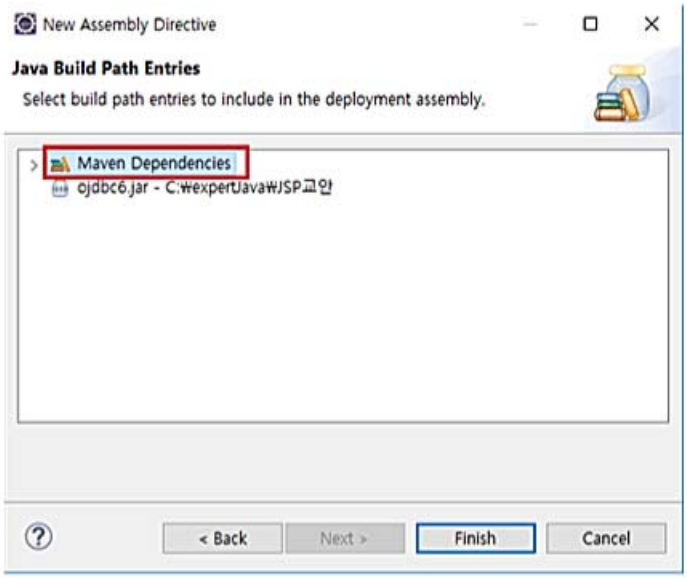
org.springframework.web.filter.CharacterEncodingFilter

오류 시 적용 방법






New Assembly Directive
Select Directive Type
Add a new assembly directive.
Archive via Path Variable
Archives from File System
Archives from Workspace
Folder
Java Build Path Entries
Project
< Back
Next >
Finish
Cancel


New Assembly Directive
Java Build Path Entries
Select build path entries to include in the deployment assembly.
Maven Dependencies
ojdbc6.jar - C:₩exper₩Java₩JSP교안
< Back
Next >
Finish
Cancel