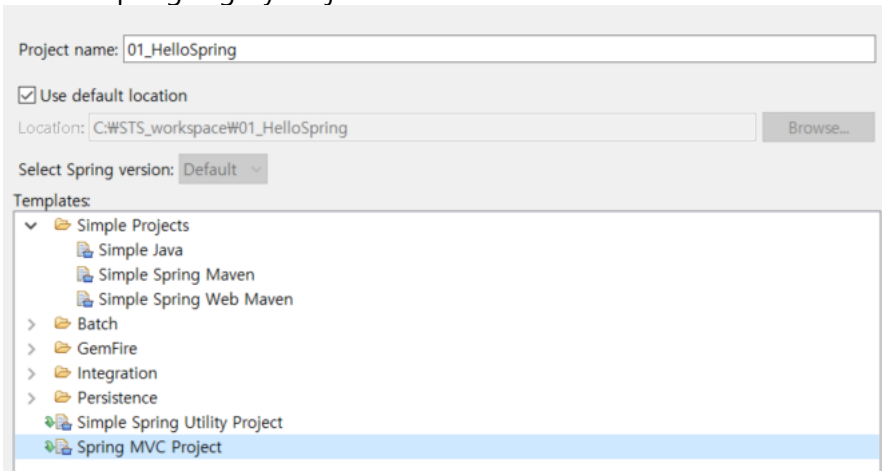
SPRING SETUP

Notebook:	First Notebook		
Created:	2019-10-02 오후 2:32	**Updated:**	2019-10-04 오후 4:29
Author:	qorhvkqorhvkqorhvk@gmail.com		
URL:	http://maven.apache.org/download.cgi		

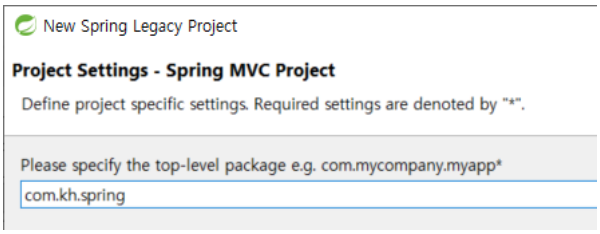
- 스프링 웹 구조
-
- 스프링 IDE 를 다운로드
  - STS PSRING TOOL SUIT -> MADE BY ECLIPSE
  - https://spring.io/tools/sts/all
  - then extract
- MAVEN 을 통해서 SPRING JAR FILES 들 다 받을거임
  - http://maven.apache.org/
  - apache-maven-3.6.2-bin.zip <----- download bin.zip file
-
-

- setting up the environment
  - encoding (changing to html environement to UTF-8):
  - 1. In the Preferences, go to General - Editors - Spelling - change to UTF-8
  - 2. Workspace to UTF-8 (the second one)
  - 3. JSON Files - UTF-8
  - 4. web - css file , htmlfile, jspfile to UTF-8
  - 5. workspace - preference XML Files

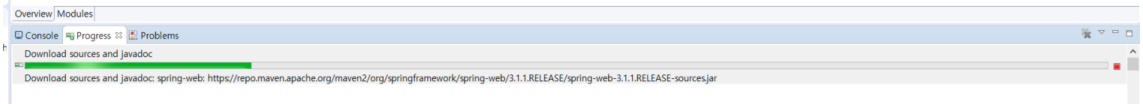
- Add server
  - check "serve modules without publishing"

- creating a project
  - new -> Spring Legacy Project
  - 

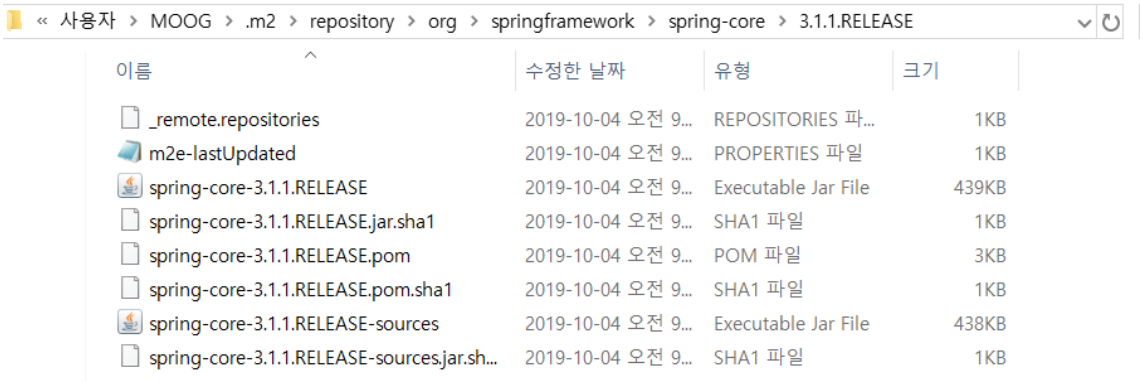
    - check spring MVC Project



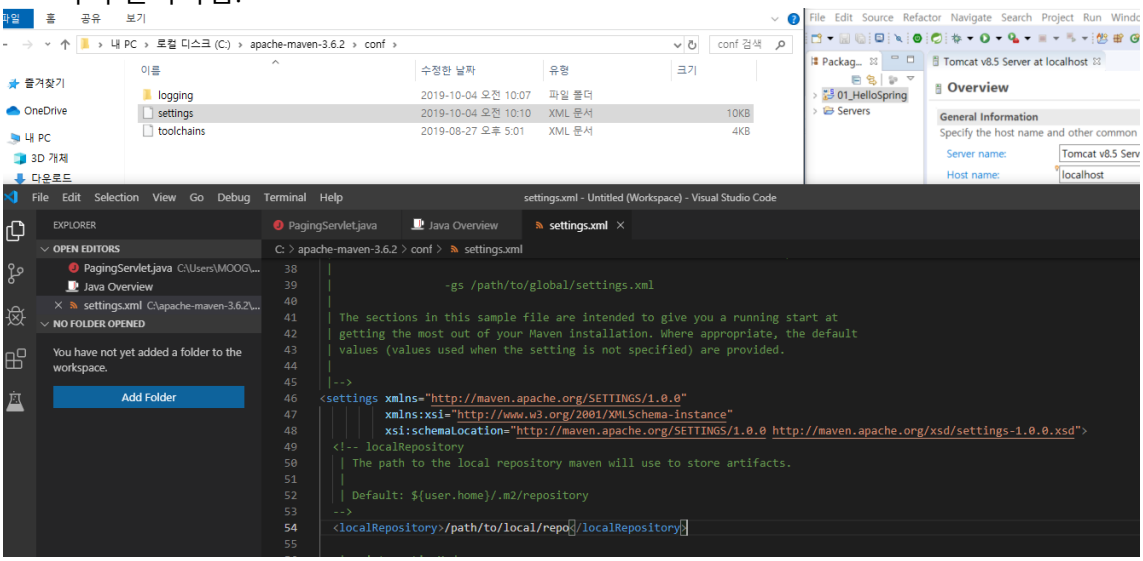
- write your package name and press Finish
- 맨 마지막 단어가 root context 가 된다



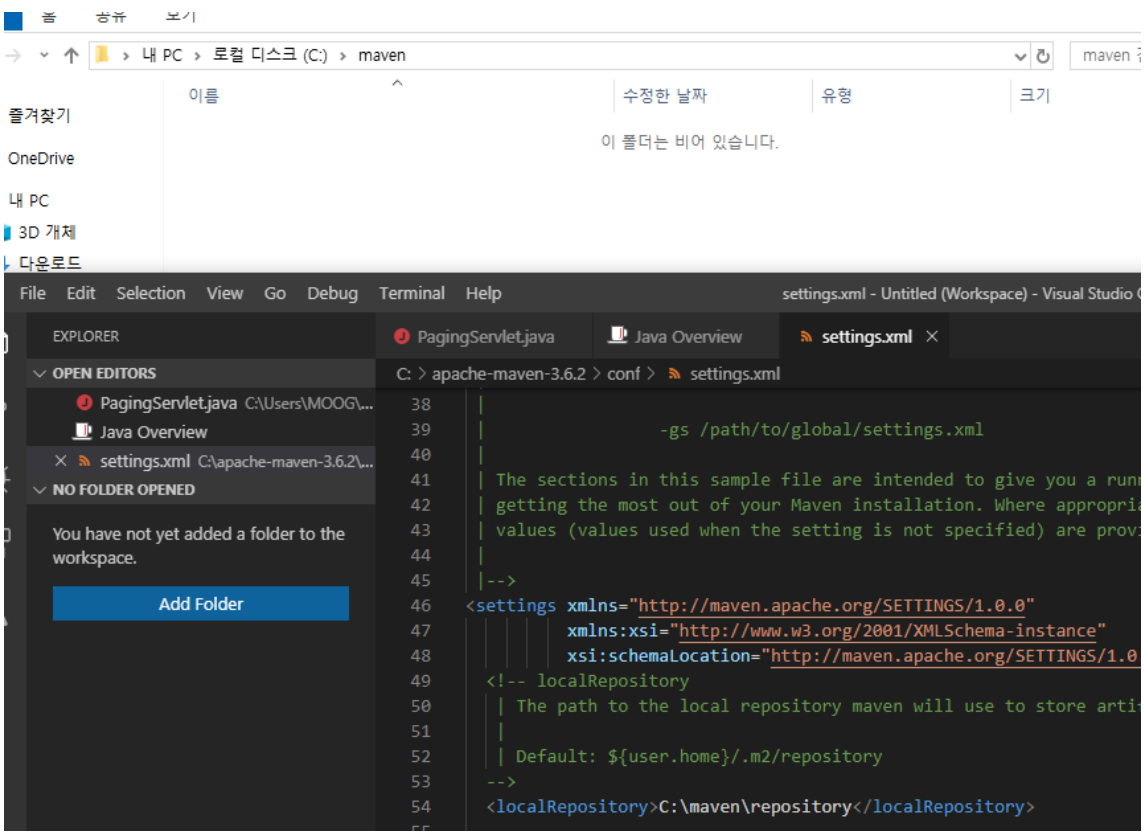
- Progress 를 보면 이렇게 받아오고 있다.



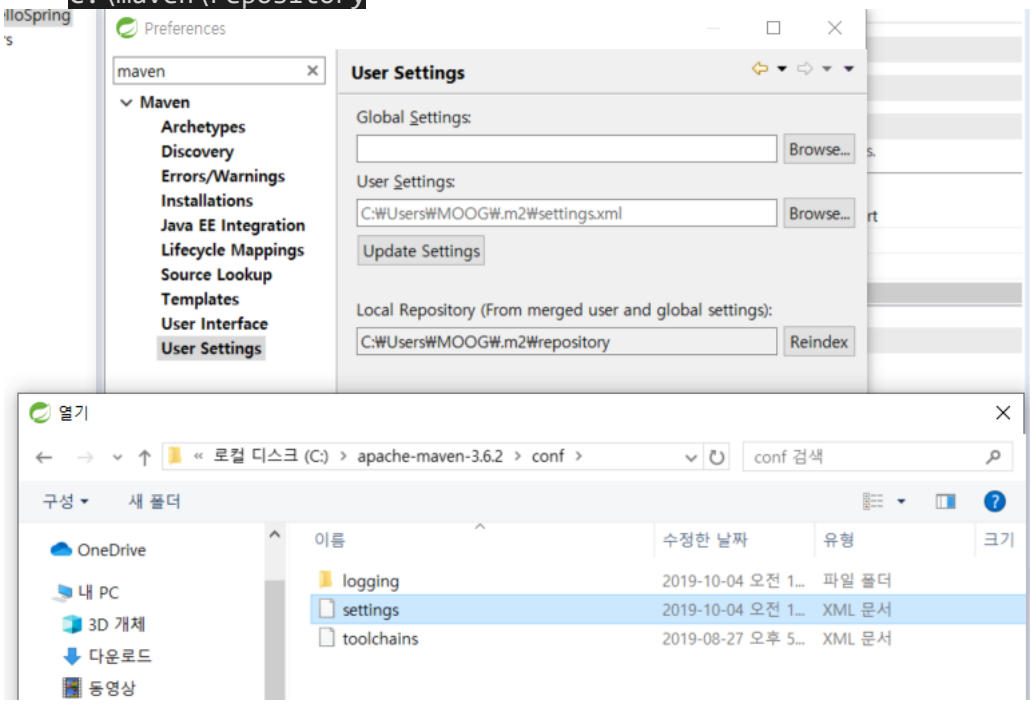
- 여기 있는 파일들 중에 0kb 가 아닌지 학인 해야하고 만약에 0kb가 있다면 폴더를 다 다시 받아야함.



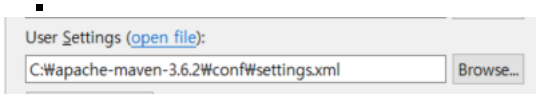
- extract maven zip file, open settings file on VS Code and remove the activate <localRepository> line



- create a maven folder in C drive and change the path on VS code to "`C:\maven\repository`"

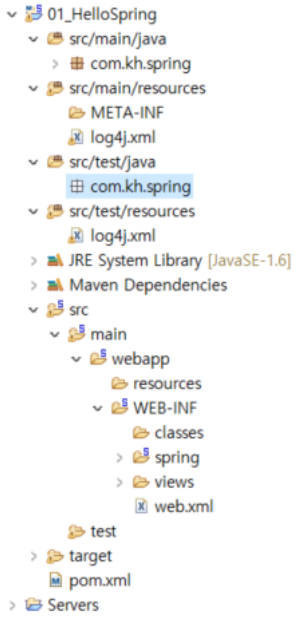


- In STS, open Window - Preferences - Maven - User Settings and change "User Settings" to the path where settings.xml file is located

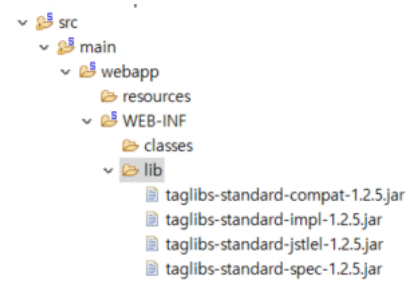


- Like this
- And Press Apply. You'll see the Progress downloading the jar files again

- Like this



- 이렇게 나오면 됨.
- \src\main\java - 여기는 java srouce files 들이 들어간다.
- \src\test\java - test 용 files 들이 여기에 들어간다.
- src.main.webapp.resrouces 안에 js, css files 들어간다
- serverlet-context.xml - servlet 을 위한 .xml
- root-context.xml - 모든 프로젝트를 위한 .xml
- Note the view folder is inside the WEB-INF folder
- pom.xml - 추가적으로 더 필요한 라이브러리를 받아오면 여기에 알아서 저장됨. 즉, jar 파일들을 받아올수 있게 하는 파일



- create a lib folder on this location.
- Import the JSTL files (the 4 jars)..

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
```

- Ad that line on home.jsp (the top)

- Start the server on this linke " http://localhost:9090/spring/" -----> This is the root

- pom.xml 안에 java 버전 이랑 스프링 프레임워크 다 들어가 있음

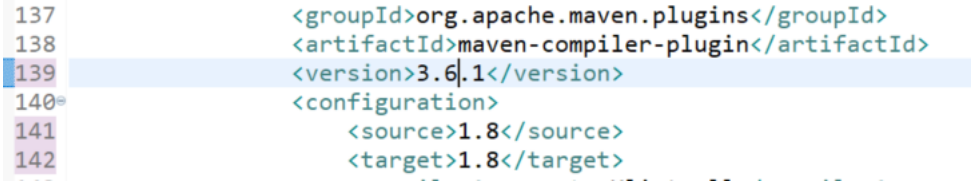
  - Maven 이 있어서 이렇게 pom.xml 안에 알아서 다 넣어줌
  -
```
<properties>
    <java-version>1.8</java-version>
    <org.springframework-version>5.0.6.RELEASE</org.springframework-version>
```

  - 스프링 업데이트 5.0.6.RELEASE 로 바꾸기

```
<!-- Servlet -->
<dependency>
    <groupId>javax.servlet</groupId>
    <artifactId>javax.servlet-api</artifactId>
    <version>3.1.0</version>
    <scope>provided</scope>
</dependency>
<dependency>
    <groupId>javax.servlet.jsp</groupId>
    <artifactId>javax.servlet.jsp-api</artifactId>
    <version>2.3.1</version>
    <scope>provided</scope>
</dependency>
```

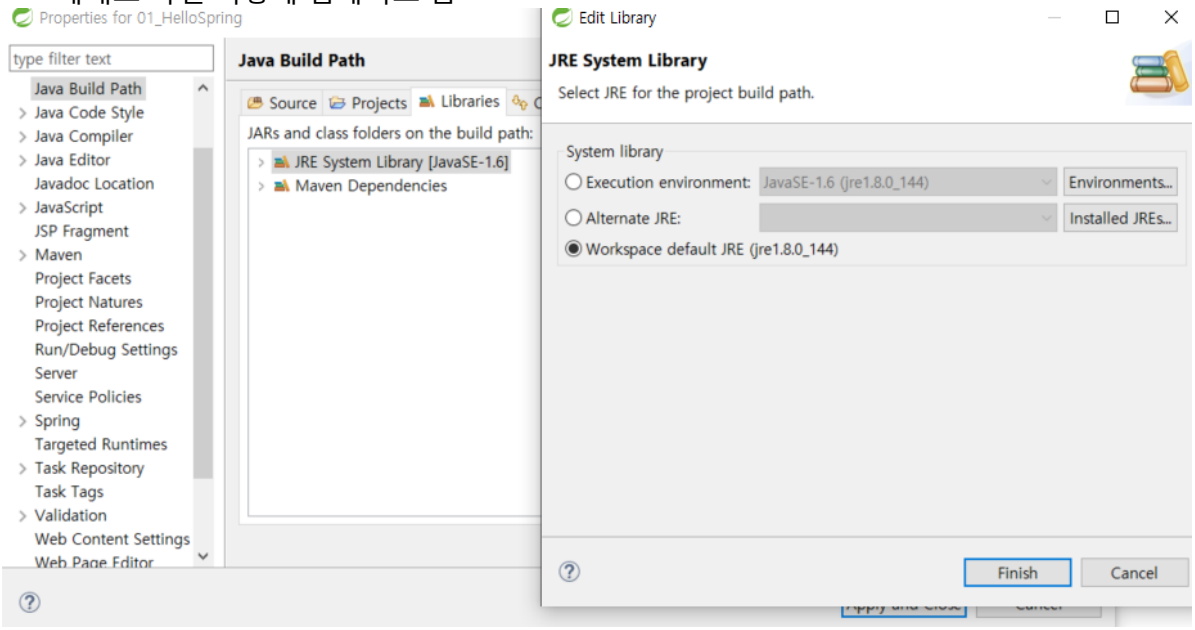
  - 서블렛 업데이트
  - 이건 꼭 이렇게 바꾸라. If you are confused, just copy this

```
<!-- Servlet -->
        <dependency>
            <groupId>javax.servlet</groupId>
            <artifactId>javax.servlet-api</artifactId>
            <version>3.1.0</version>
            <scope>provided</scope>
        </dependency>
        <dependency>
            <groupId>javax.servlet.jsp</groupId>
            <artifactId>javax.servlet.jsp-api</artifactId>
            <version>2.3.1</version>
            <scope>provided</scope>
        </dependency>
        <dependency>
            <groupId>javax.servlet</groupId>
            <artifactId>jstl</artifactId>
            <version>1.2</version>
        </dependency>
```

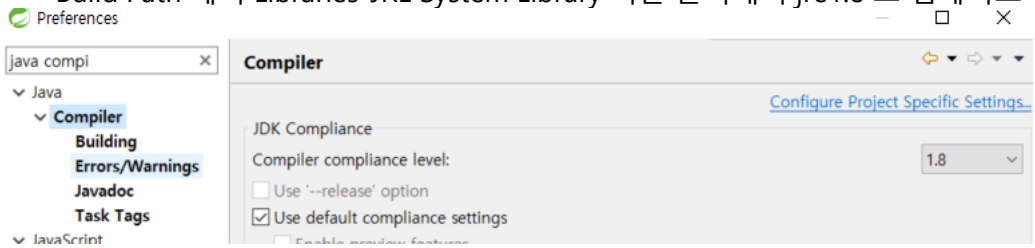


  - 컴파일러 업데이트
  -

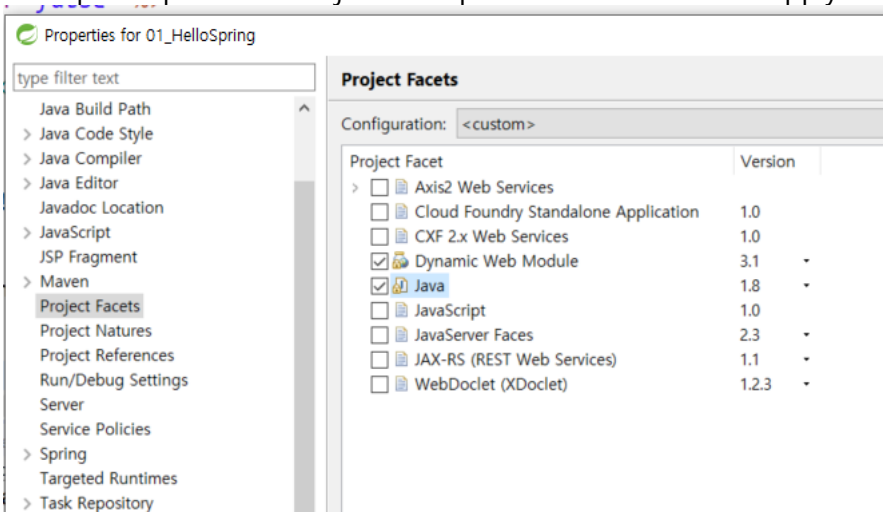
- ◦ 제대로 하면 이렇게 업데이트 됨

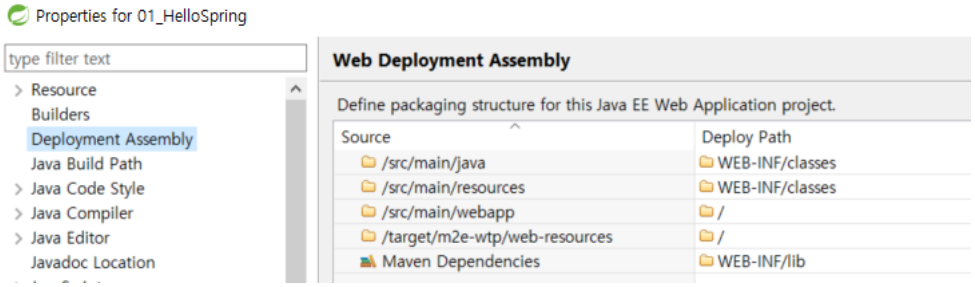


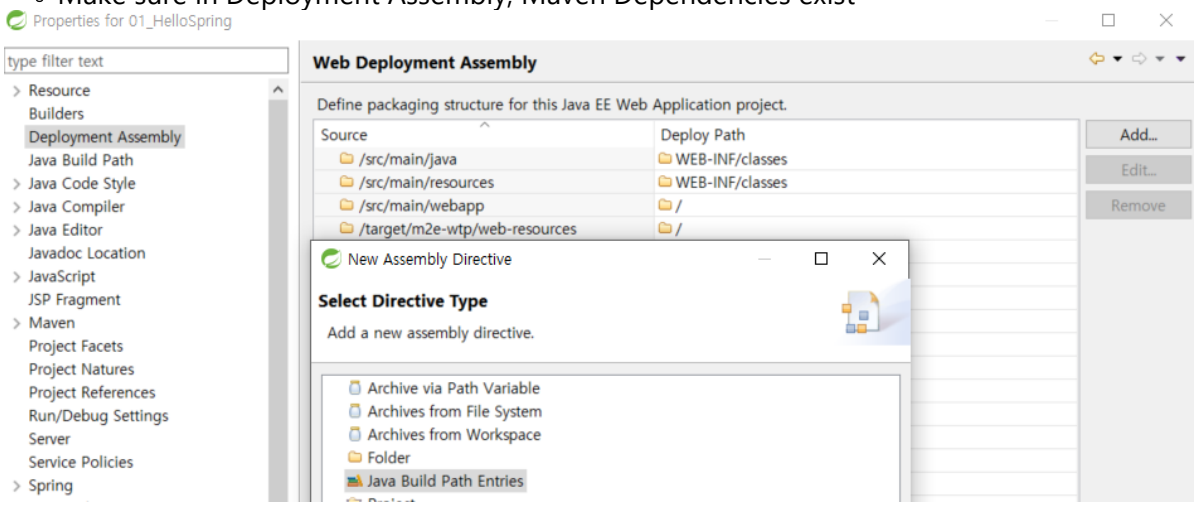
- ◦ Build Path 에서 Libraries-JRE System Library 더블 클릭해서 jre1.8 로 업데이트 함



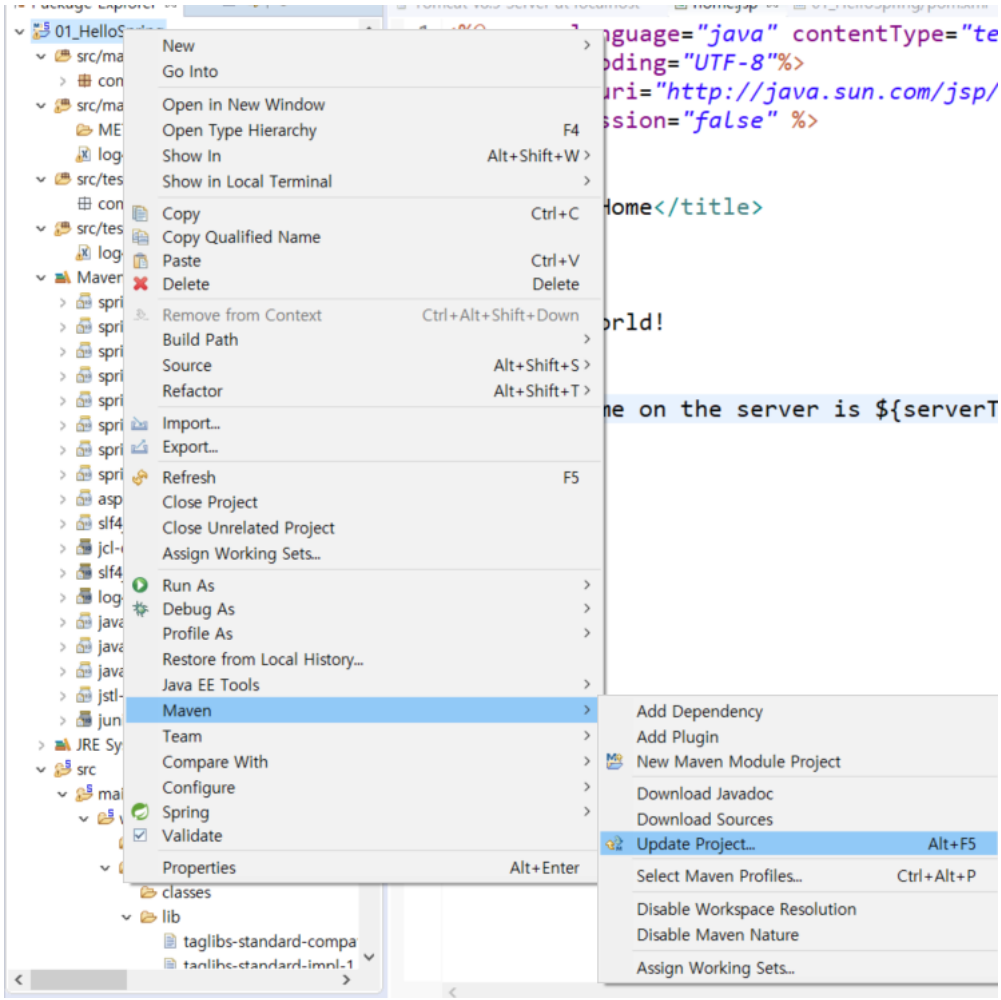
- workspace - preferences - java- compiler - JDK to 1.8 and hit apply



- right click on the project - properties - project facets
  - Dynamic Web Module - 3.1
  - Java - 1.8
  -
- 

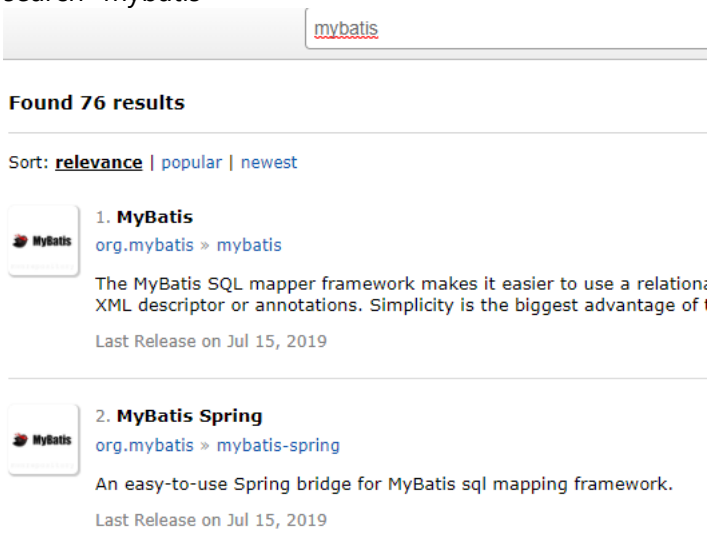
  - Make sure in Deployment Assembly, Maven Dependencies exist
- 

  - 만약 Maven Dependencies 가 없으면 Add - Java Build Path Entries 누르면 있음



- -그런 다음 우클릭 - Maven - Update Project 해야됨

- https://mvnrepository.com/ - 얘가 jar file 들을 복사해서 가져가야됨. 수시로 쓴다고 함
  - search "mybatis"

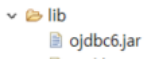


  - 위에 있는 두개 다 받아야함
  - 3.4.6 Central 169
  - 이렇게 인기 많은걸로 받음

- 이거 복사해서 .xml 안에 넣어줌

```
        <!-- DB 설정 jar dependency -->
        <!-- https://mvnrepository.com/artifact/org.mybatis/mybatis -->
        <dependency>
            <groupId>org.mybatis</groupId>
            <artifactId>mybatis</artifactId>
            <version>3.4.6</version>
        </dependency>
        <!-- https://mvnrepository.com/artifact/org.mybatis/mybatis-spring -->
        <dependency>
            <groupId>org.mybatis</groupId>
            <artifactId>mybatis-spring</artifactId>
            <version>1.3.2</version>
        </dependency>
```

  - 이렇게 두개

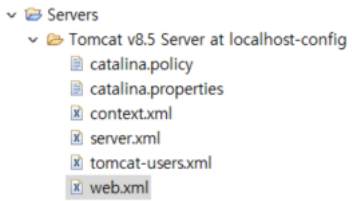


- ojdbc6 은 lib 안에 그냥 넣음

- Maven repository 가서 이거 다운받음. 1.4
  - https://mvnrepository.com/artifact/commons-dbcp/commons-dbcp

- 주의해야 할 점은 아무 버전이나 막 받으면 안 되고 degrade 하면서 jar 파일을 맞춰야 되는 경우도 있다고 한다.

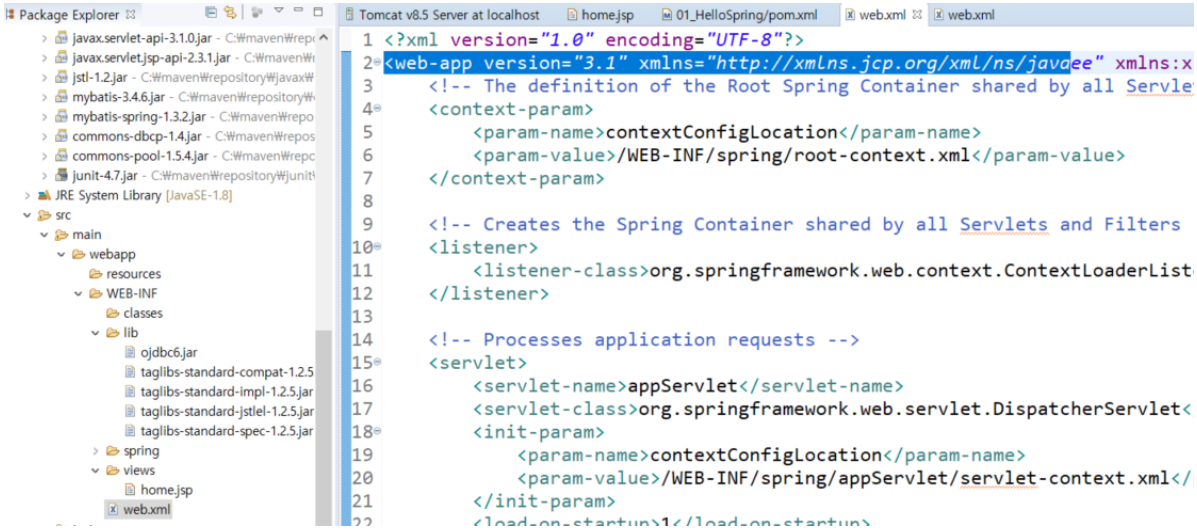
web.xml 설정하기



- Server 폴더에 있는 web.xml 열고

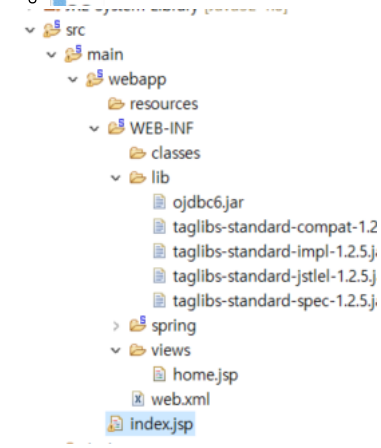
```
on --><web-app version="3.1" xmlns="http://xmlns.jcp.org/xml/ns/javaee" xmlns:x
```

  - line 17번에 있는거 복사

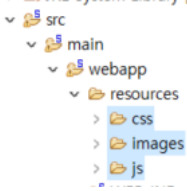


-
  - copy and paste it on the project's web.xml line 2

- in the project's web.xml file - add this
  -
```
30     <welcome-file-list>
31         <welcome-file>index.jsp</welcome-file>
32     </welcome-file-list>
```

-


  - webapp 폴더 안에 index.jsp 넣기 (rclass 에서 받음)

-


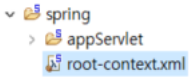
  - css/ images/ js files all go into the resources folder like this

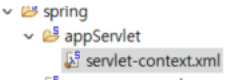
- 필터 등록하기
  - web.xml 다시 들어간다.

```
    <!-- 필터 등록하기 -->
    <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> <!-- /* 은 모오든 내용 -->
    </filter-mapping>
```

- Just copy and paste this

```
    <!-- 필터 등록하기 -->
    <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> <!-- /* 은 모오든  내용 -->
    </filter-mapping>
```

- 이젠 스프링 환경설정 파일 한다
- 

  - root-context.xml 은 스프링 돌아가는데 필요한거 다 담고 있다
  - 스프링은 객체들에 의해 돌아간다.
  - 여기에 디비에 대한 설정이 들어간다.
  - 

  - servlet-context.xml 에는

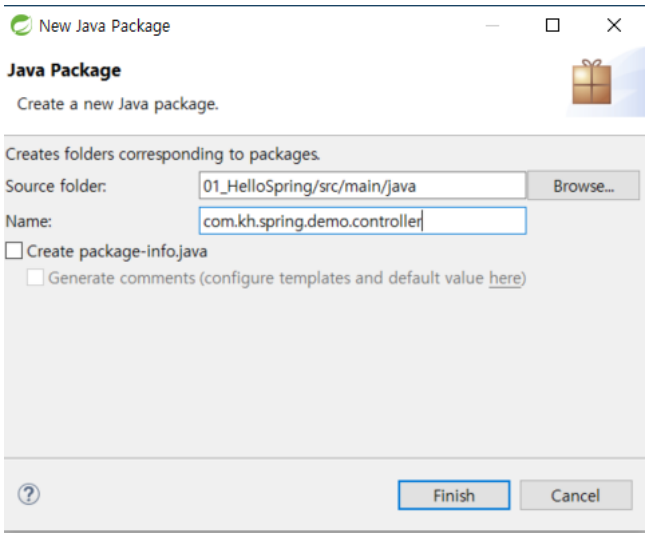
```
    <!-- Enables the Spring MVC @Controller  programming model -->
    <annotation-driven /> <!-- 이게 잇어야 annotation  쓸수 있음 -->
    <!-- Handles HTTP GET requests for /resources/**  by efficiently serving
up static resources in the  ${webappRoot}/resources directory -->
    <resources mapping="/resources/**"  location="/resources/" /> <!-- 소스파
일에 대한 경로 설정  -->
```

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

	<context:component-scan  base-package="com.kh.spring" /><!--
.controller,  .service, .dao 가 뒤에 붙음 -->
```

이런 게 들어가있다.

- package 만들때는 이렇게 해야한다.



- There's no servlet in spring as it's already integrated.
- Thus, you only create classes in the controller package.

스프링 bean 으로 등록하기 (DemoController.java 클래스 확인)

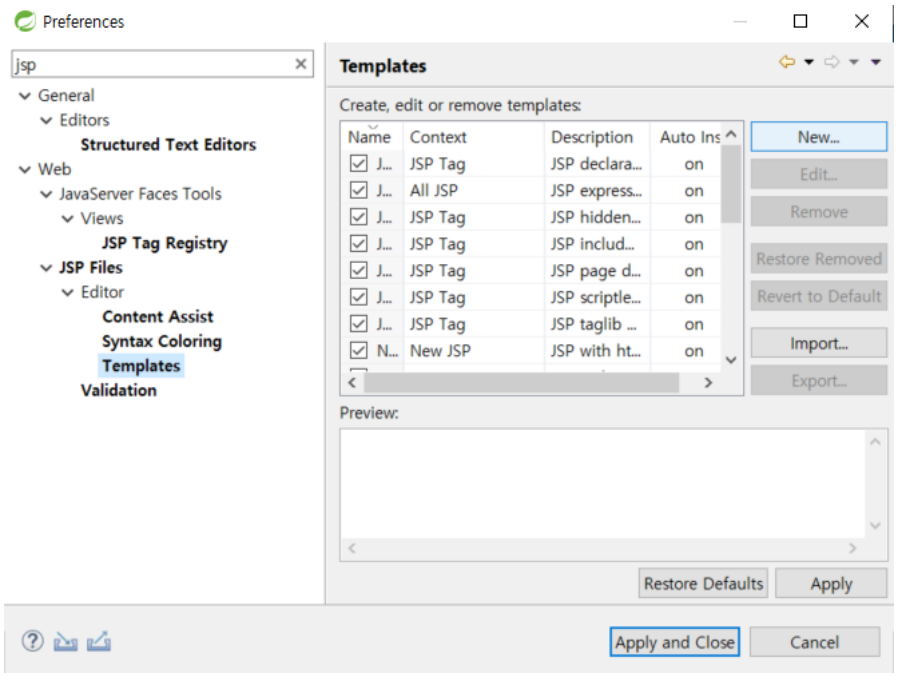
```
@Controller  // annotation 표시
public class DemoController {

	@RequestMapping("/demo/demo.do") // 매핑값
	public String jangwon() {
		System.out.println("/demo/demo.do 컨트롤러  호출");
		return "demo/demo"; // 얘를 resolve 한테 넘김.  servlet-context.xml 에
있는. 얘는 그냥 Dispatcher 라고  생각하면 됨
	}
}
```

- @Controller 를 쓰고 임포트 하면 file 옆에 S 가 생긴다.
- 즉, 스프링이 관리하는 파일로 바뀐다.

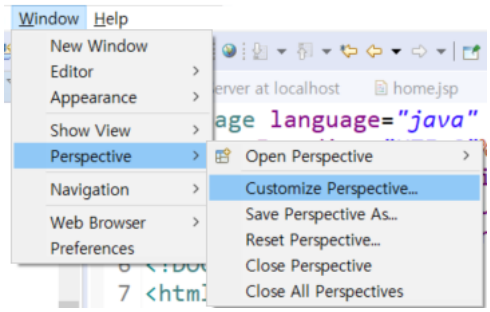
jsp template 만들기

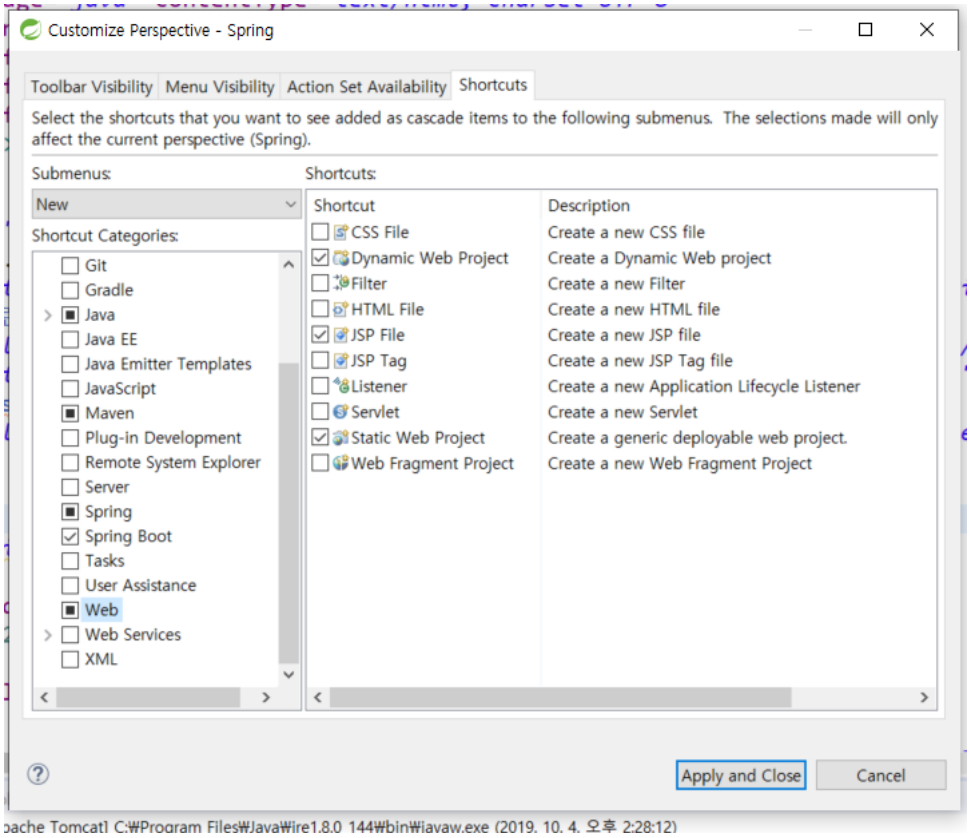
- window - pref - jsp - Templates



- 꼭 new JSP 로 설정하고 만들면 된다. (**)

- JSP 편하게 쓰기 (이거 보면 대충 알거라고 생각)





.

컨트롤러 메소드가 받을수 잇는 Parameter

- HttpServletRequest
- HttpServletResponse
- HttpSession
- java.util.Locale -> 위치값 확인할때
- InputStream, Reader : 요청에 대한 입력 스트림
- OutputStream, Writer : 요청에 대한 출력 스트림
- @PathhVariable
- @RequestParam
- @RequestHeader
- @CookieValue
- @RequestBody : Ajax 전송시 Json 객체를 받는 파라미터 타입!

Map, Model, ModelMap : view 한테 보낼 데이터를 보관하는 전용객체
Command 객체 : Vo 파라미터로 넘어오는 값을 자동으로 Vo 에 대입