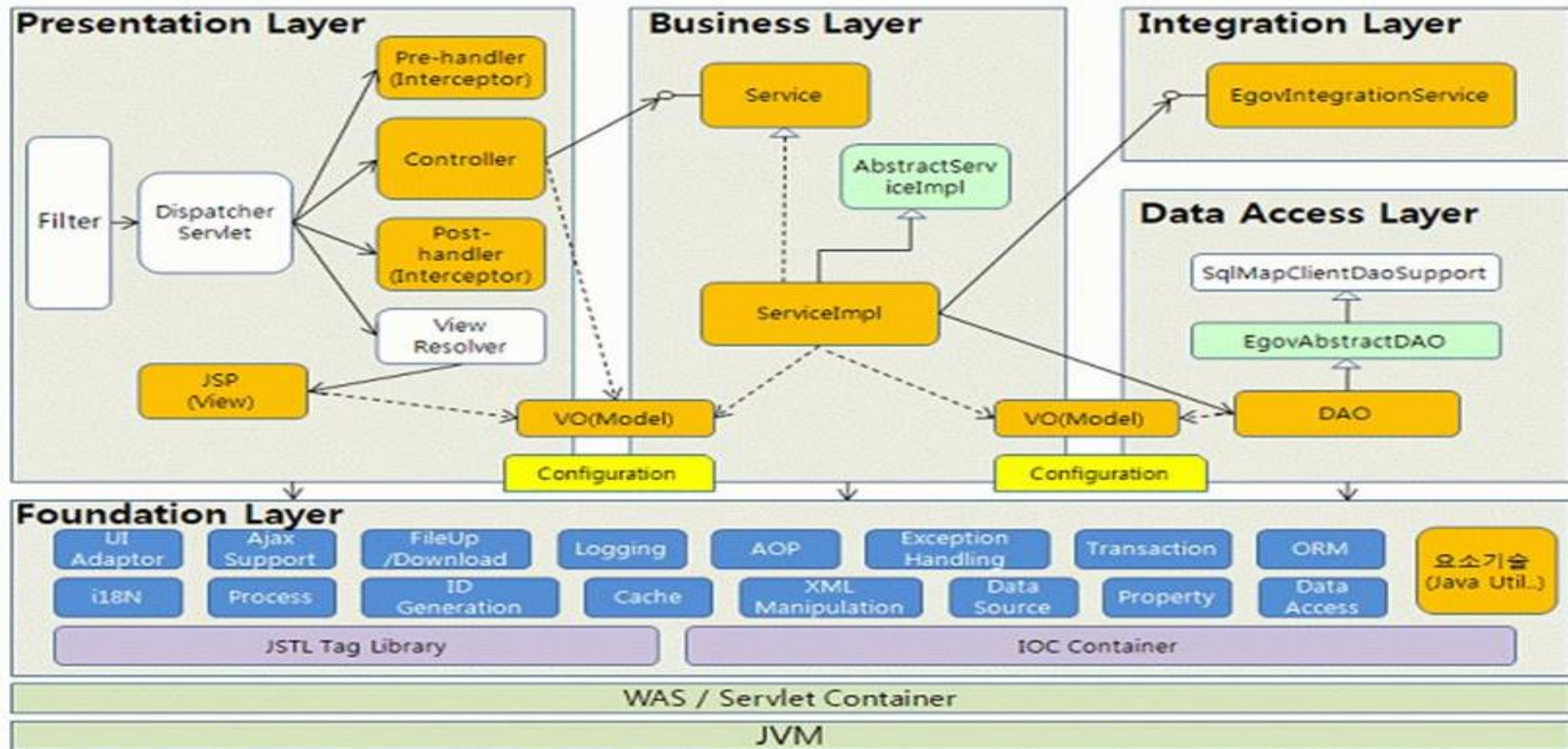


# Spring 게시판 만들기 (spring tool 3)

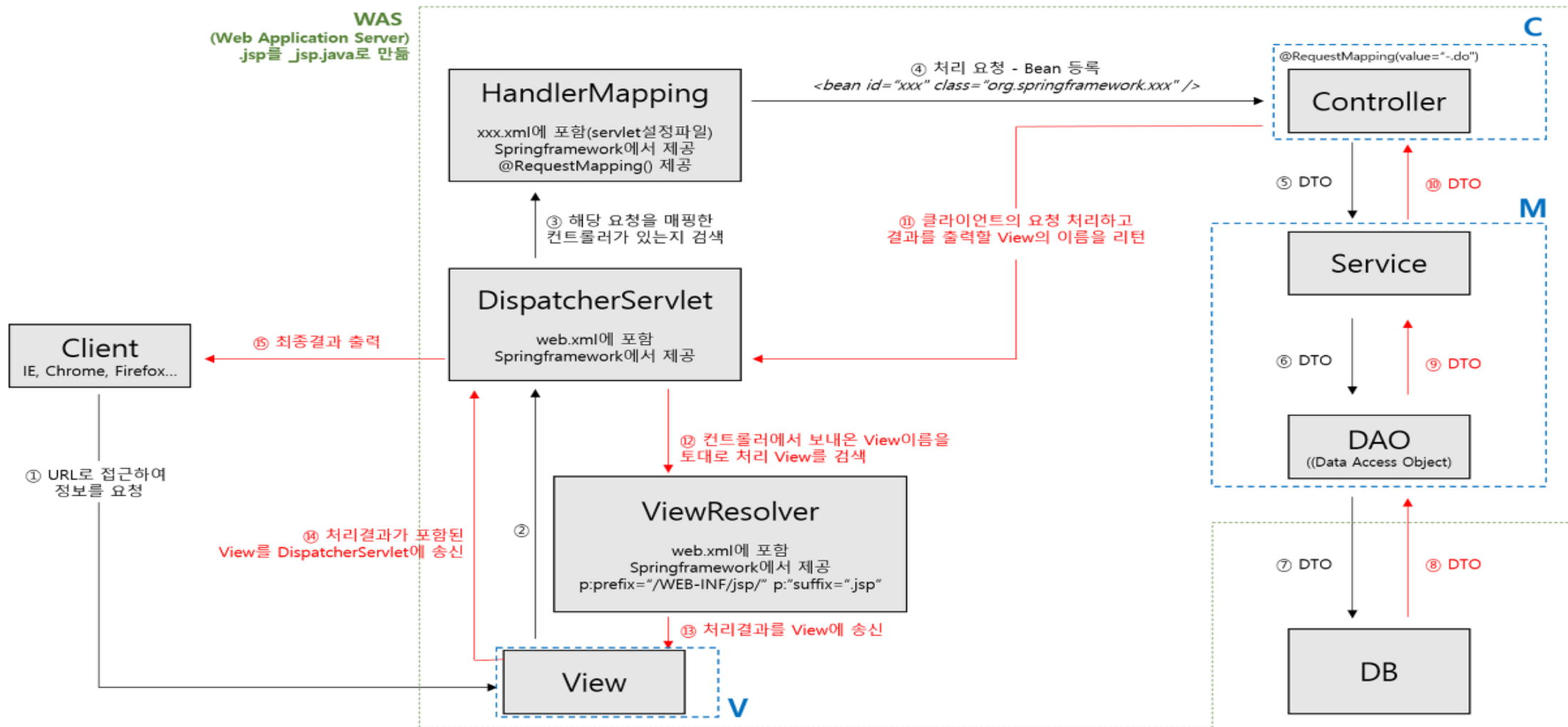
# 목차

- 작업환경 만들기
- 스프링의 구조
- Maven 구조
- Oracle
- Mybatis 연동하기
- CRUD 작성
- Restful API 서버 작성하기
- Ajax 호출하기

# 스프링 구조



# 스프링의 구조



# DI(Dependency Injection)

- 클라

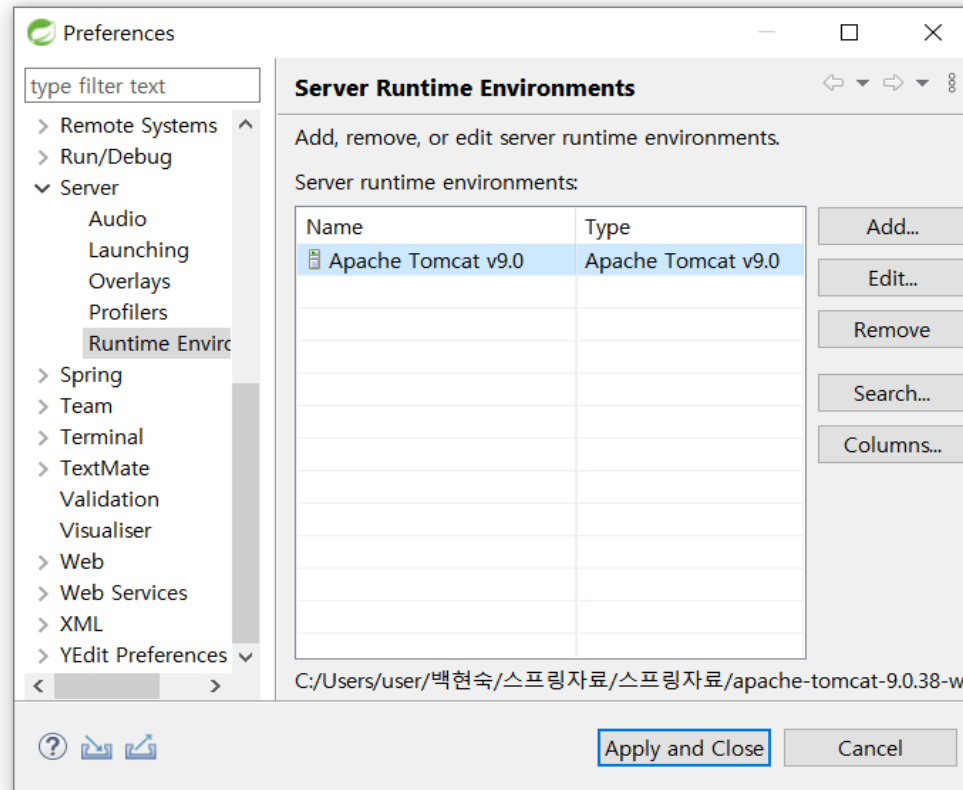
# **AOP(Aspect orientation project)**

# spring tool 3 다운로드

- <https://github.com/spring-projects/toolsuite-distribution/wiki/Spring-Tool-Suite-3>
  - win32-64.zip 버전을 다운받아 압축을 푼다
- 
- 아파치 톰캣을 다운받아 압축을 푼다
  - <https://tomcat.apache.org/download-90.cgi>

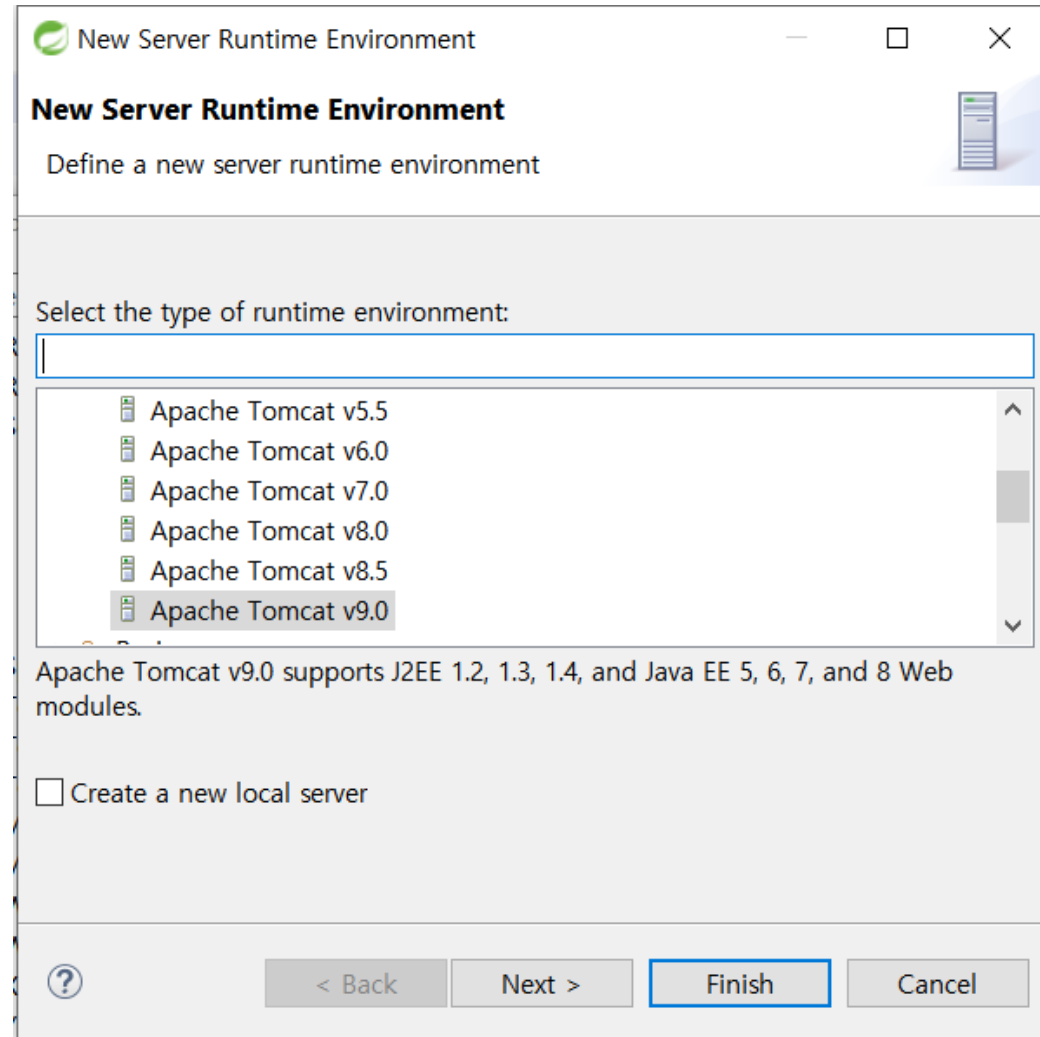
# 스프링 이클립스와 톰캣 연동하기

- windows - preferences - server - Runtime Environment
- Add 선택

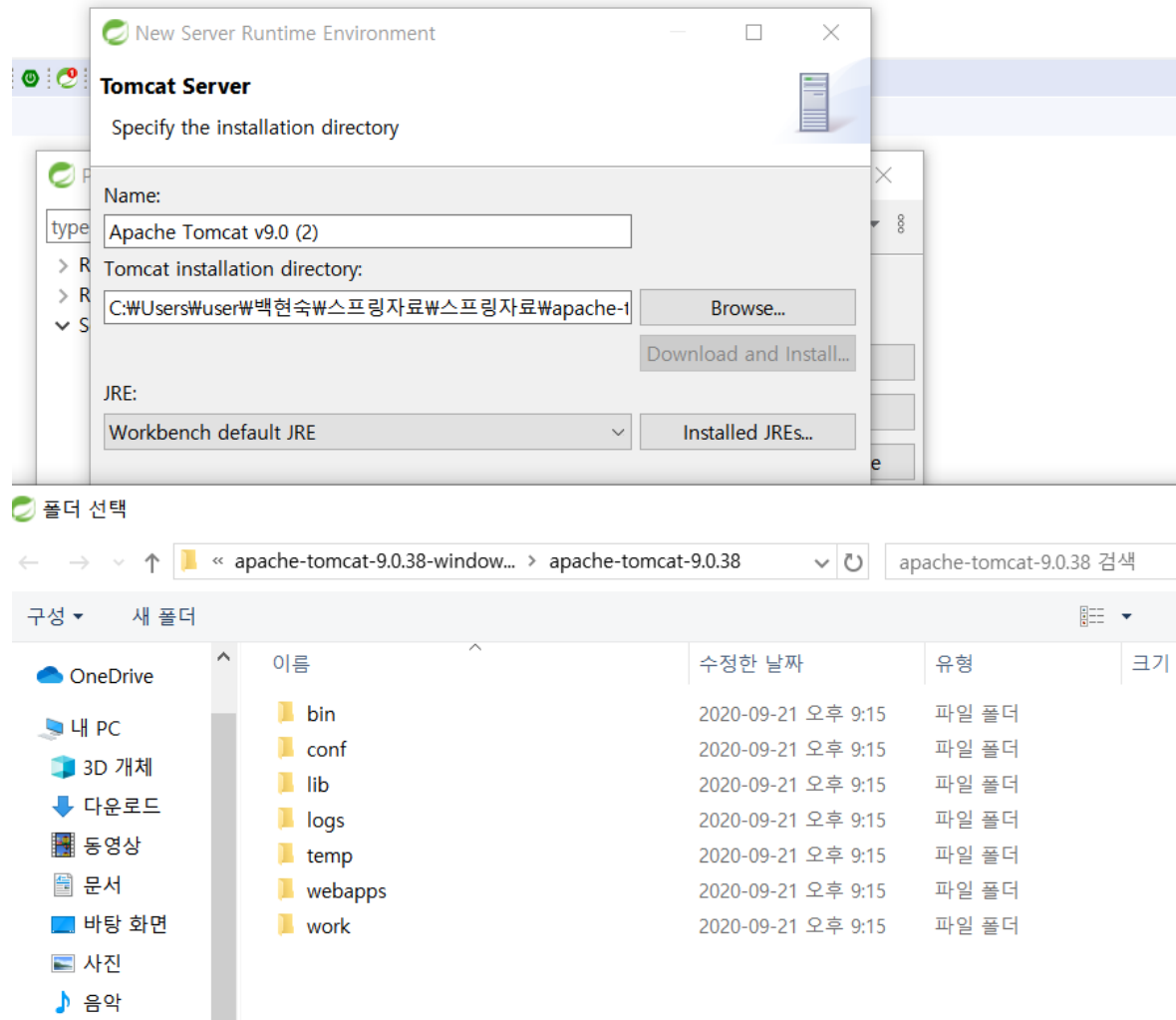




# 스프링 이클립스와 톰캣 연동하기



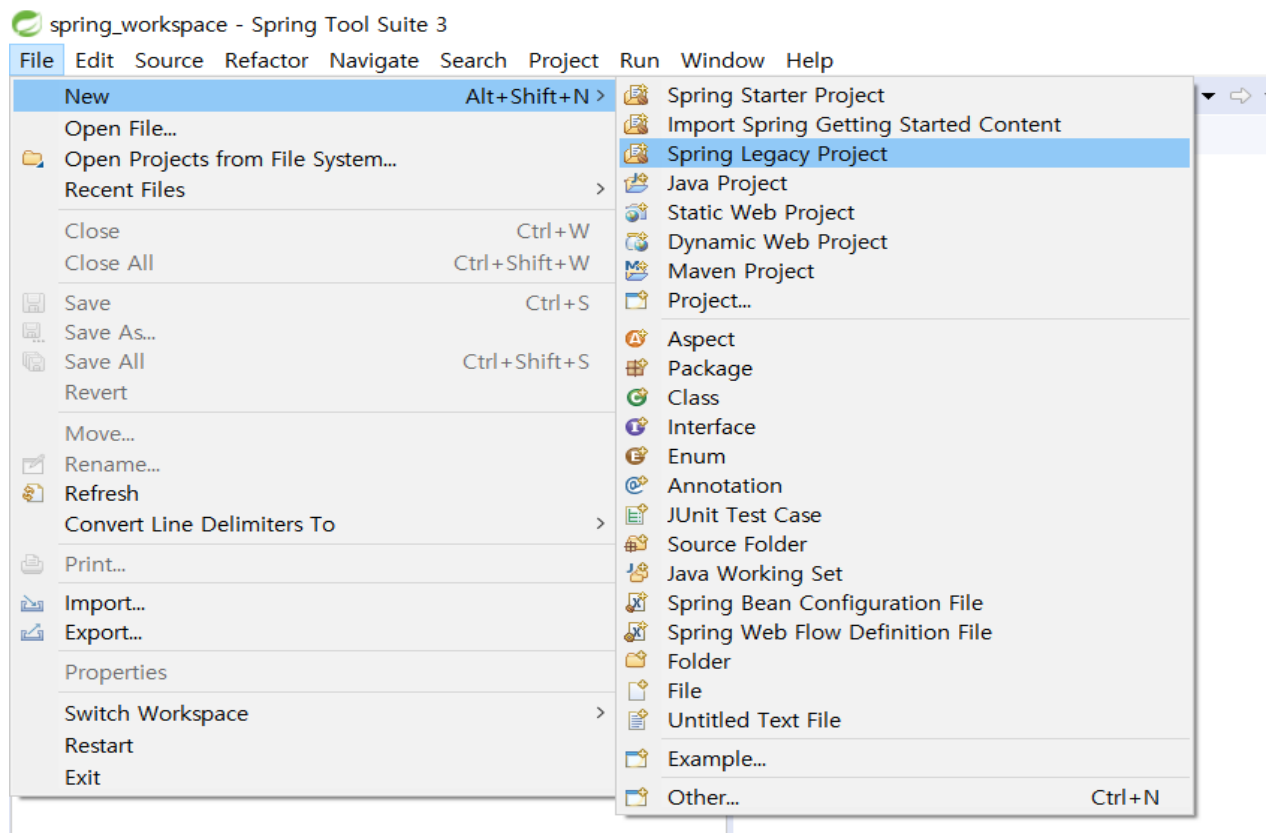
# 스프링 이클립스와 톰캣 연동하기



bin 폴더가 보여야 한다

# Spring MVC 프로젝트 작성하기

- File - New - Spring Legacy Project



# Spring MVC 프로젝트 작성하기

New Spring Legacy Project

**Spring Legacy Project**

Click 'Next' to load the template contents.

Project name: myhome1

☒ Use default location

Location: C:\Users\User\바탕화면\스프링자료\스프링자료\spring-tool-suite-3.9.12.RELEASE-e4.15.0-win32-x86\_64\spring\_workspace Browse...

Select Spring version: Default

Templates:

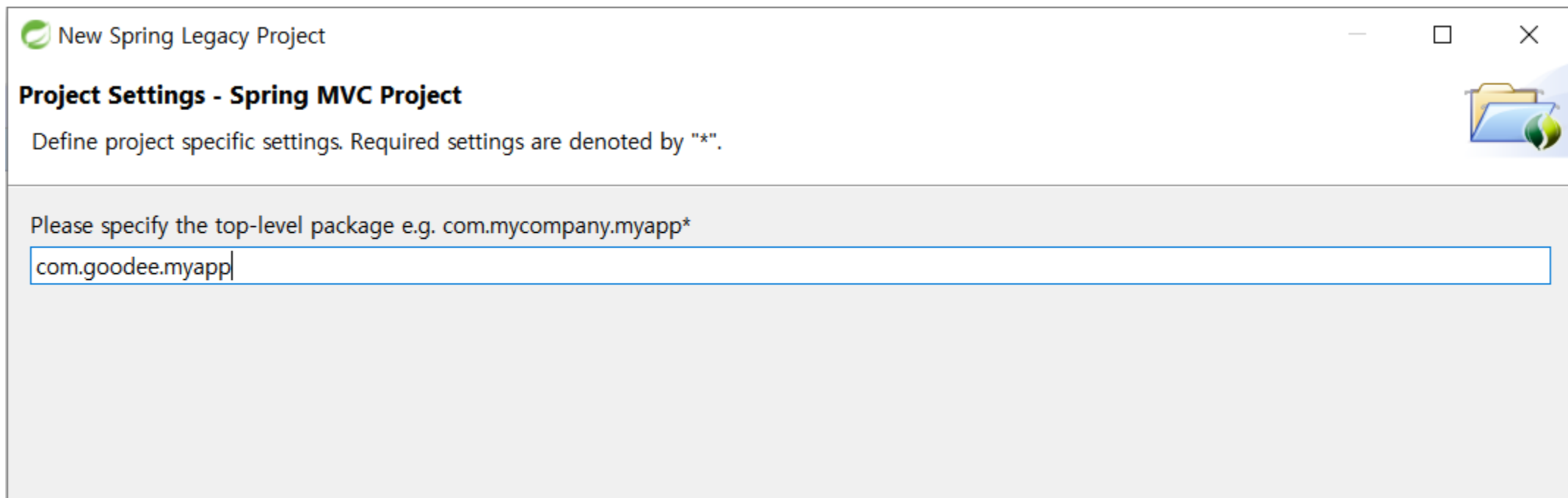
- Simple Projects
  - Simple Java
  - Simple Spring Maven
  - Simple Spring Web Maven
- Batch
- GemFire
- Integration
- Persistence
- Simple Spring Utility Project
- Spring MVC Project

requires downloading

[Configure templates...](#) Refresh

Description:  
A new Spring MVC web application development project

# Spring MVC 프로젝트 작성하기



New Spring Legacy Project

**Project Settings - Spring MVC Project**

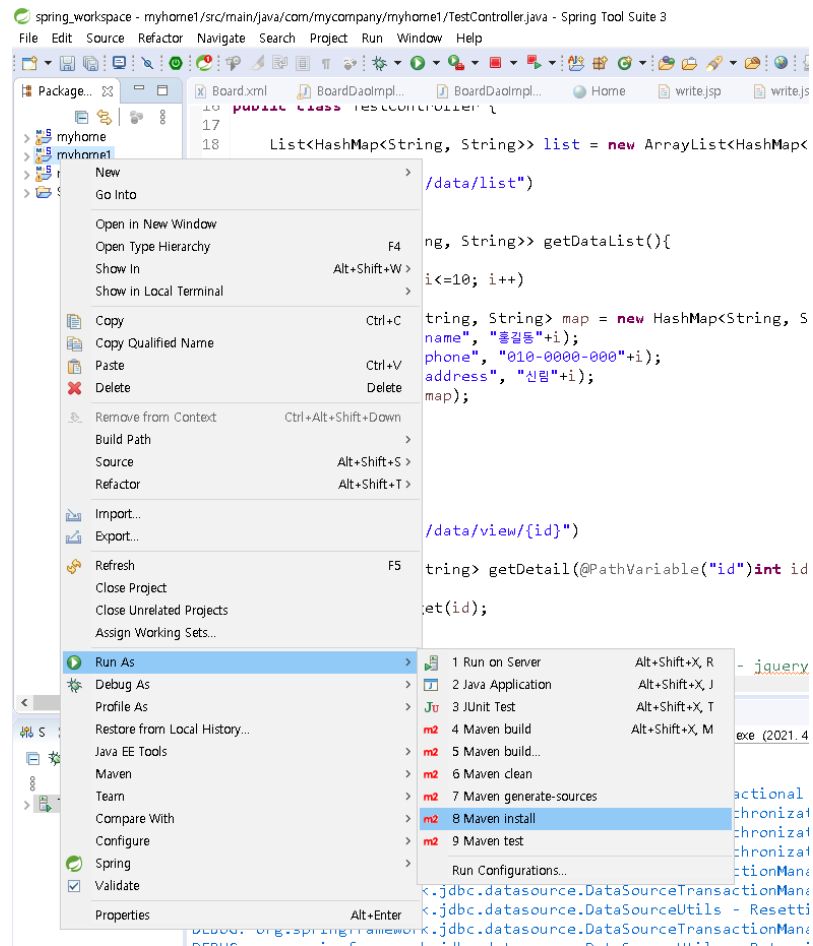
Define project specific settings. Required settings are denoted by "\*".

Please specify the top-level package e.g. com.mycompany.myapp\*

com.goodee.myapp

# maven install

- repository로부터 필요한 라이브러리를 다운받아 빌딩작업을 수행한다.
- 프로젝트명 - 마우스 오른쪽 - run as - maven install 선택하기

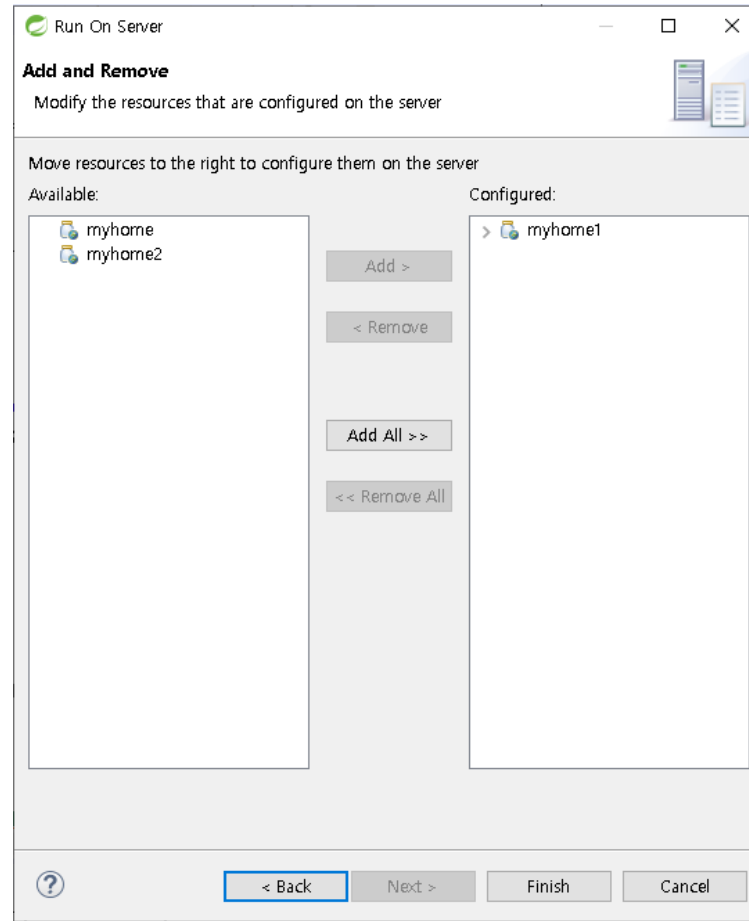
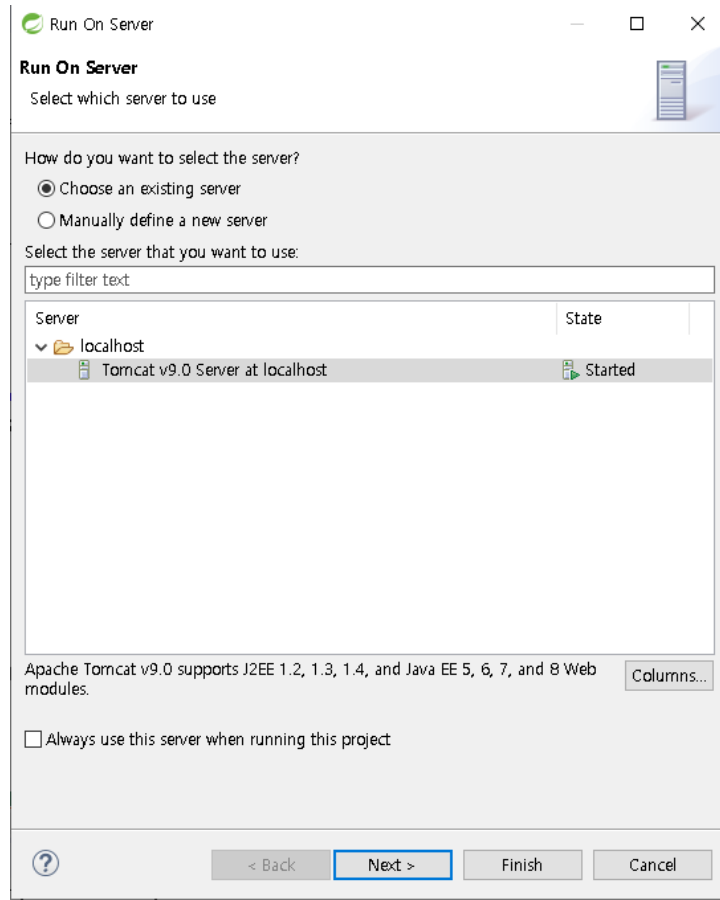


## maven - 라이브러리 위치

- 인터넷에 늦을 경우에 라이브러리가 미처 다 다운받지 않아 프로그램 실행시 오류가 발생할 수 있다
- c:/users/사용자계정/.m2/repository 폴더를 지우고 maven install을 다시 한다

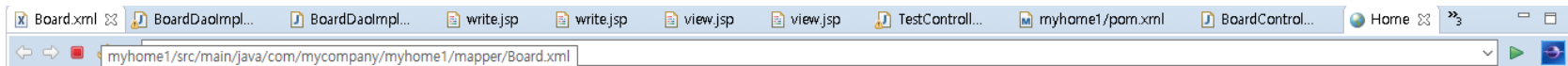
# 프로젝트 실행하기

- 프로젝트 - 마우스 오른쪽 - run as - run on server





# 실행화면

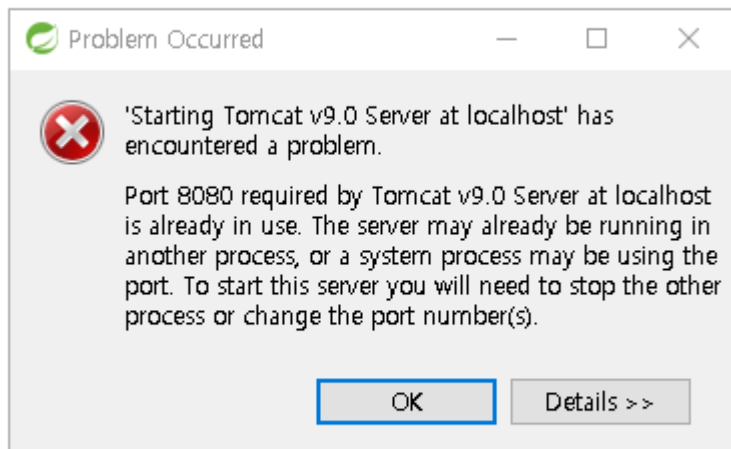


**Hello world!**

The time on the server is 2021년 4월 29일 (목) 오후 10시 17분 04초.

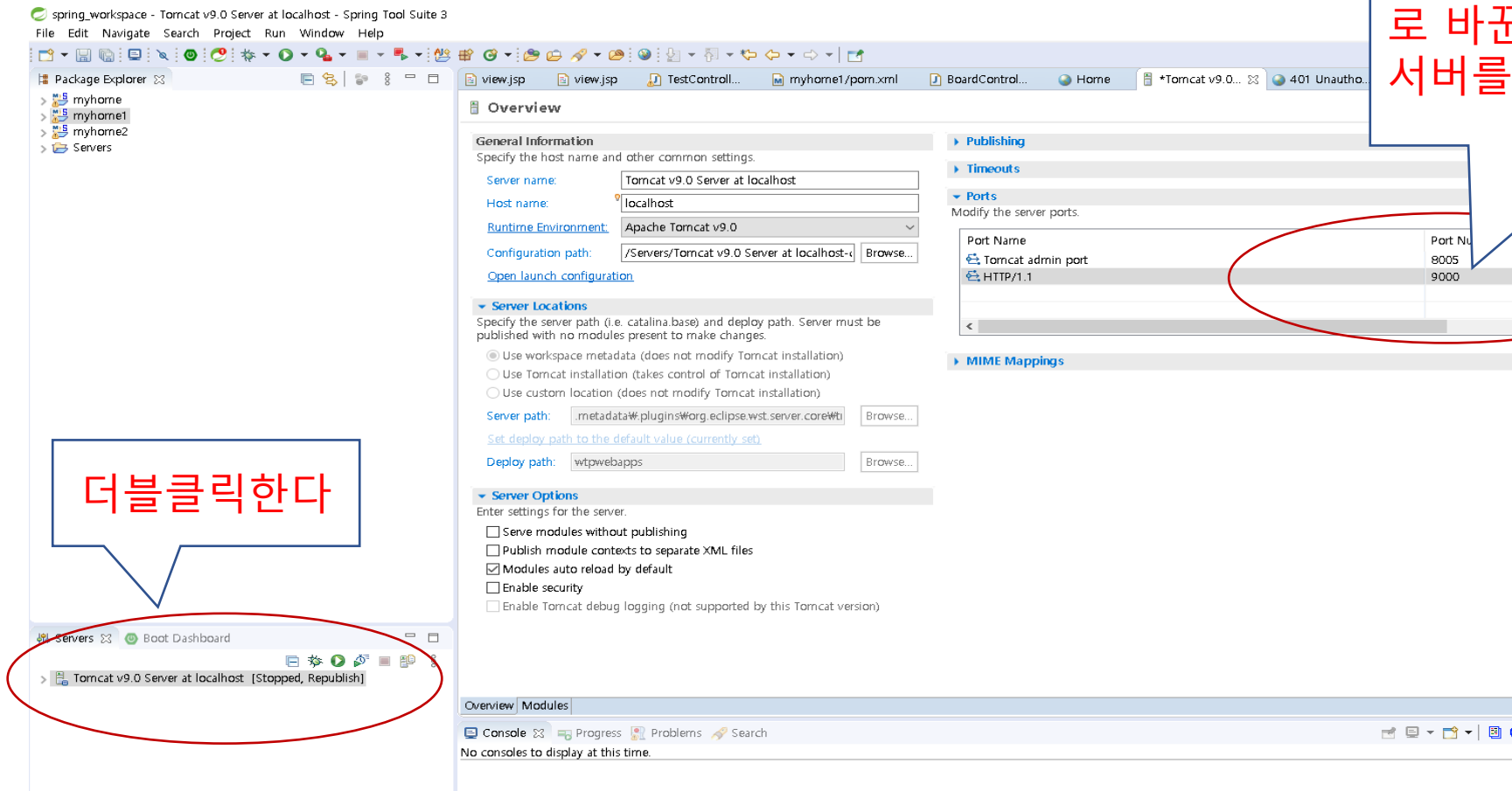
# 에러 처리

- 다음과 같은 에러가 뜰 경우에 이미 포트번호를 사용하고 있는 경우이다.



# 에러처리

- 포트번호 수정하기 - 서버를 찾아서 더블클릭한다



포트번호를 8080에서 9000으로 바꾼후 저장한다  
서버를 재가동한다

더블클릭한다

# pom.xml파일

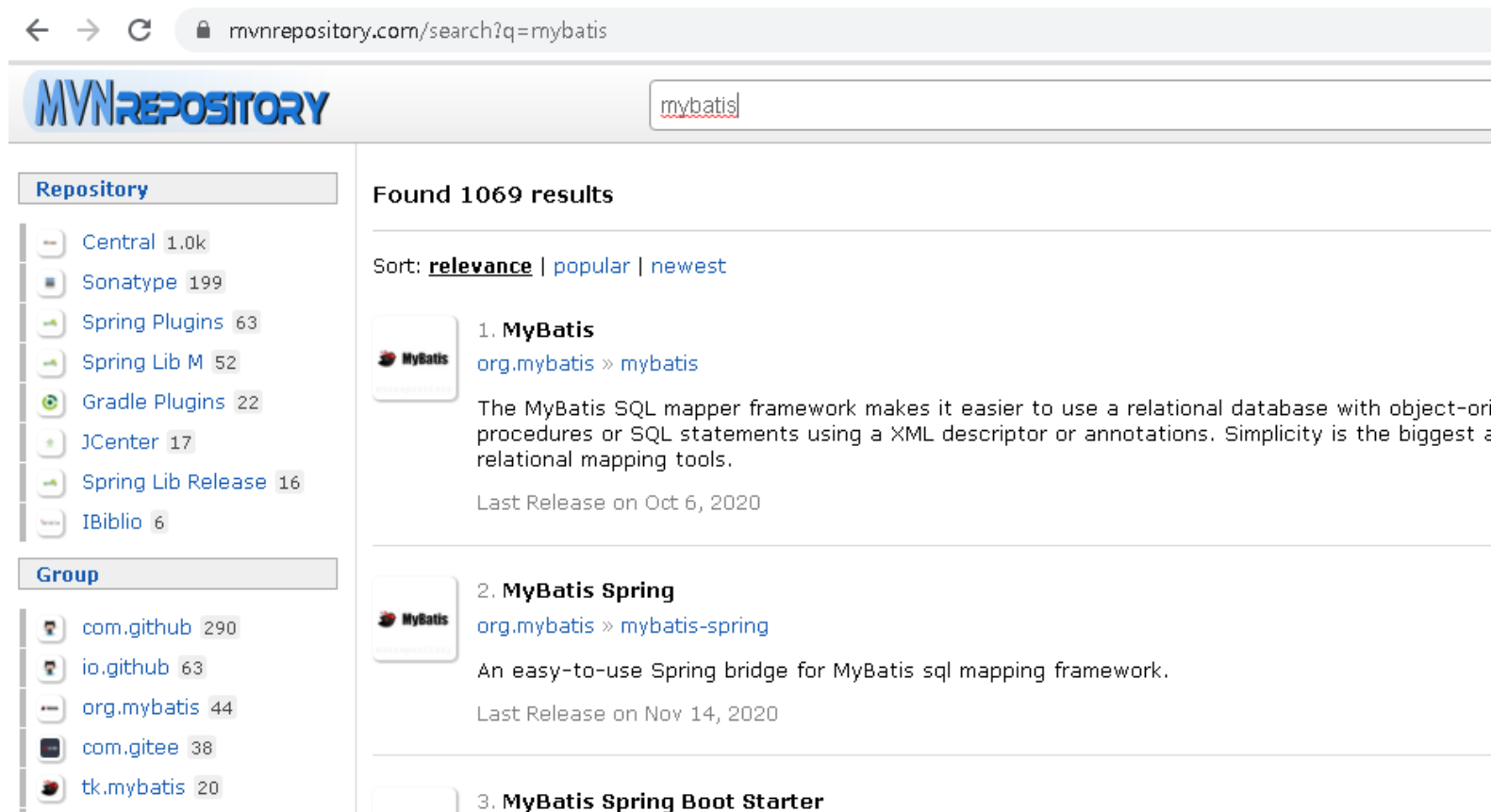
- 필요한 라이브러리 정보를 기술하면 인터넷을 이용해 다운받아 온다
- 기술방법

```
<dependencies>
  <!-- Spring -->
  <dependency>
    <groupId>org.springframework</groupId>
    <artifactId>spring-context</artifactId>
    <version>${org.springframework-version}</version>
    <exclusions>
      <!-- Exclude Commons Logging in favor of SLF4j -->
      <exclusion>
        <groupId>commons-logging</groupId>
        <artifactId>commons-logging</artifactId>
      </exclusion>
    </exclusions>
  </dependency>
```

..... 형태로 필요한 라이브러리를 배치한다

# maven repository

- <https://mvnrepository.com/>



The screenshot shows the Maven Repository website with a search for 'mybatis'. The left sidebar lists various repositories and groups. The main content area shows 1069 results, sorted by relevance. The top three results are:

- 1. MyBatis**  
[org.mybatis » mybatis](https://org.mybatis.org/mybatis)  
The MyBatis SQL mapper framework makes it easier to use a relational database with object-oriented procedures or SQL statements using a XML descriptor or annotations. Simplicity is the biggest advantage of relational mapping tools.  
Last Release on Oct 6, 2020
- 2. MyBatis Spring**  
[org.mybatis » mybatis-spring](https://org.mybatis.org/mybatis-spring)  
An easy-to-use Spring bridge for MyBatis sql mapping framework.  
Last Release on Nov 14, 2020
- 3. MyBatis Spring Boot Starter**

**Repository**

- Central 1.0k
- Sonatype 199
- Spring Plugins 63
- Spring Lib M 52
- Gradle Plugins 22
- JCenter 17
- Spring Lib Release 16
- IBiblio 6

**Group**

- com.github 290
- io.github 63
- org.mybatis 44
- com.gitee 38
- tk.mybatis 20

# maven repository

- 검색창에 찾고자하는 라이브러리 명을 입력한다.(예시: mybatis) - 클릭한다

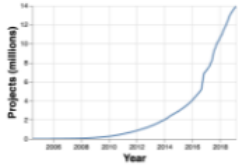
← → ↺

mvnrepository.com/artifact/org.mybatis/mybatis

MVNREPOSITORY

Search for groups, artifacts, categories

Indexed Artifacts (20.3M)



Popular Categories

Aspect Oriented

Actor Frameworks

Application Metrics

Build Tools

Bytecode Libraries

Command Line Parsers

Cache Implementations

Cloud Computing

Code Analyzers

Collections


Configuration Libraries

Core Utilities

Date and Time Utilities

Dependency Injection

Home » org.mybatis » mybatis

 **MyBatis**

The MyBatis SQL mapper framework makes it easier to use a relational database with object-oriented applications. MyBatis uses SQL statements using a XML descriptor or annotations. Simplicity is the biggest advantage of the MyBatis relational mapping tools.

License

Apache 2.0

Categories

Object/Relational Mapping

Tags

persistence

relational

mapping

Used By

1,025 artifacts

Central (34)

EBIPublic (1)

ICM (1)

Alfresco (1)

	Version	Repository	Count
3.5.x	3.5.6	Central	72
	3.5.5	Central	83
	3.5.4	Central	66
	3.5.3	Central	106
	3.5.2	Central	61
	3.5.1	Central	65
	3.5.0	Central	53

# maven repository

- 원하는 버전을 선택한다

**Indexed Artifacts (20.3M)**

**Popular Categories**

- Aspect Oriented
- Actor Frameworks
- Application Metrics
- Build Tools
- Bytecode Libraries
- Command Line Parsers
- Cache Implementations
- Cloud Computing
- Code Analyzers
- Collections
- Configuration Libraries
- Core Utilities
- Date and Time Utilities
- Dependency Injection
- Embedded SQL Databases

Home » org.mybatis » mybatis » 3.5.6

**MyBatis » 3.5.6**

The MyBatis SQL mapper framework makes it easier to use a relational database with procedures or SQL statements using a XML descriptor or annotations. Simplicity is the relational mapping tools.

License	Apache 2.0
Categories	Object/Relational Mapping
HomePage	<a href="http://www.mybatis.org/mybatis-3">http://www.mybatis.org/mybatis-3</a>
Date	(Oct 06, 2020)
Files	<a href="#">jar (1.7 MB)</a> <a href="#">View All</a>
Repositories	Central
Used By	1,025 artifacts

**Maven** | Gradle | SBT | Ivy | Grape | Leiningen | Buildr

```
<!-- https://mvnrepository.com/artifact/org.mybatis/mybatis -->
<dependency>
  <groupId>org.mybatis</groupId>
  <artifactId>mybatis</artifactId>
  <version>3.5.6</version>
</dependency>
```

☒ Include comment with link to declaration

이 부분을 복사해서  
pom.xml파일에 붙여넣  
기 한다

<dependencies>....  
</dependencies>  
사이에 끼워 넣는다

# 필요 설정파일(pom.xml)

```
<!-- MyBatis용 -->
<!-- https://mvnrepository.com/artifact/org.mybatis/mybatis -->
<dependency>
    <groupId>org.mybatis</groupId>
    <artifactId>mybatis</artifactId>
    <version>3.4.6</version>
</dependency>

<!-- mybatis랑 spring 연동 라이브러리 -->
<!-- https://mvnrepository.com/artifact/org.mybatis/mybatis-spring -->
<dependency>
    <groupId>org.mybatis</groupId>
    <artifactId>mybatis-spring</artifactId>
    <version>1.3.3</version>
</dependency>

<!-- https://mvnrepository.com/artifact/org.springframework/spring-jdbc -->
<dependency>
    <groupId>org.springframework</groupId>
    <artifactId>spring-jdbc</artifactId>
    <version>${org.springframework-version}</version>
</dependency>

<!-- https://mvnrepository.com/artifact/org.apache.commons/commons-dbcp2 -->
<dependency>
    <groupId>org.apache.commons</groupId>
    <artifactId>commons-dbcp2</artifactId>
    <version>2.7.0</version>
</dependency>
```



# 사이트 설정파일( web.xml )

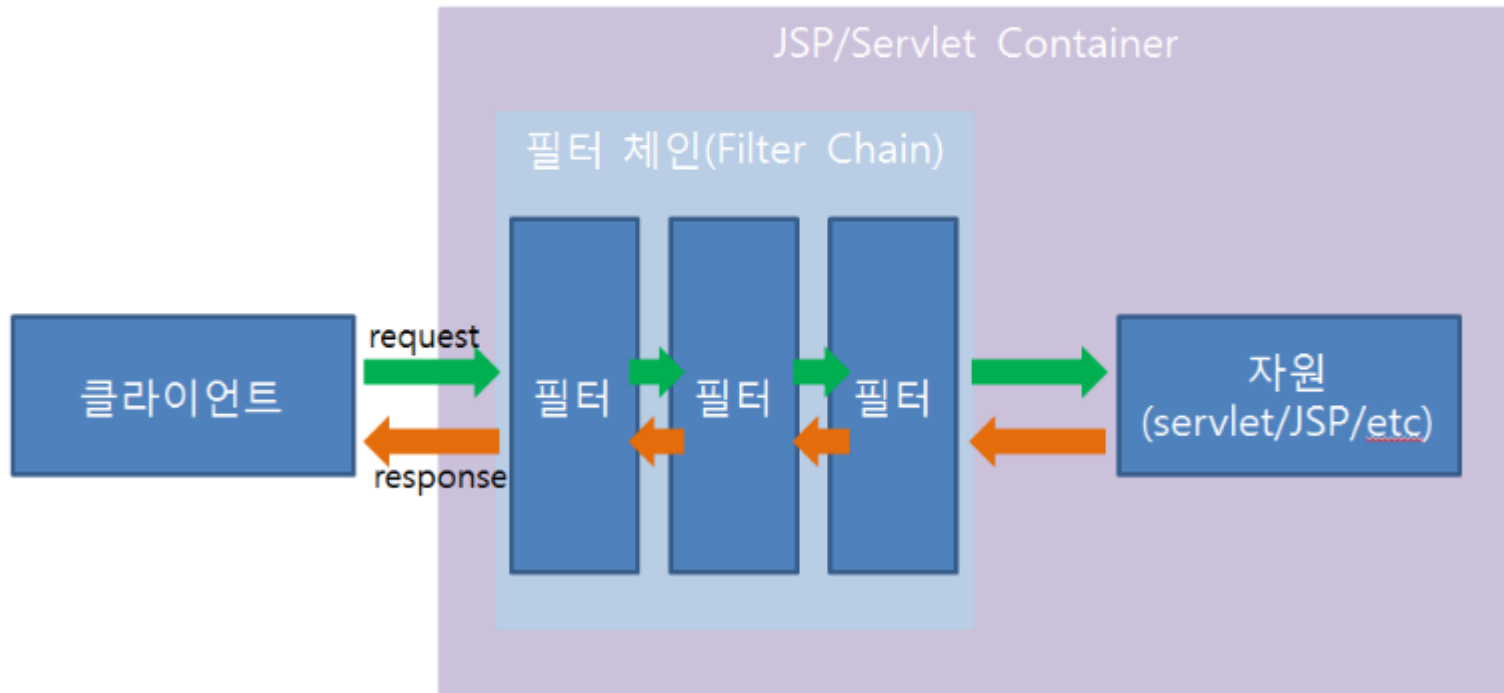
- 프로젝트명 - src - main - webapp - WEB-INF - web.xml 파일
  - 필터 설정
  - 리스너 설정
  - 서블릿 설정

# 필터란

- 필터는 객체의 형태로 존재하며 클라이언트로부터 오는 요청(request)과 최종자원(서블릿/JSP/기타 문서) 사이에 위치하며 클라이언트의 요청 정보를 알맞게 변경할 수 있으며, 또한 필터는 최종 자원과 클라이언트로 가는 응답(response) 사이에 위치하여 최종 자원의 요청 결과를 알맞게 변경할 수 있다.
- 보통 문자셋을 처리할 때 많이 사용한다. 한글이나 영어권 국가가 아닌 나라들의 경우에 문자들이 깨지는걸 방지하기 위해서 보통 utf-8을 사용하는데 jsp 에서 값을 주고 받을때마다 request.setCharacterEncoding("utf-8") 과 response 객체에서도 동일한 작업을 해줘야 하는데 필터를 통하면 주고받는 데이터 모두를 인코딩해서 전달하기 때문에 별도의 인코딩 작업이 필요 없다.

# 필터의 기본 구조

## [필터의 기본 구조]



출처 : [https://twofootdog.github.io/Spring-%ED%95%84%ED%84%B0\(Filter\)%EB%9E%80-%EB%AC%B4%EC%97%87%EC%9D%B8%EA%B0%80/](https://twofootdog.github.io/Spring-%ED%95%84%ED%84%B0(Filter)%EB%9E%80-%EB%AC%B4%EC%97%87%EC%9D%B8%EA%B0%80/)

# 필터 적용하기

```
<filter>
  <filter-name>CharacterEncodingFilter</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>
  <init-param>
    <param-name>forceEncoding</param-name>
    <param-value>true</param-value>
  </init-param>
</filter>

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

모든 url에 요청에 대해서 org.springframework.web.filter.CharacterEncodingFilter을 사용하여 UTF-8로 강제로 인코딩 하라는 의미임

# 리스너

- 웹어플리케이션의 주요 변화를 감지하여 특정 이벤트가 발생했을 때 특별한 작업을 처리하도록 할 수 있습니다.
- 스프링 컨텍스트가 로딩될때 기본적인 작업을 진행하도록 하고자 할 때 사용합니다
- 보통 스프링에서는 데이터베이스와 연결작업을 진행할 때 주로 사용합니다

# 리스너 예시

```
<context-param>
  <param-name>contextConfigLocation</param-name>
  <param-value>/WEB-INF/spring/root-context.xml</param-value>
</context-param>
<listener>
  <listener-
    class>org.springframework.web.context.ContextLoaderListener</listener-class>
</listener>.
```

- 스프링 컨테이너가 로드될때 ContextLoaderListener를 동작시킨다. ContextLoaderListener는 /WEB-INF/spring/root-context.xml에 있는 기본설정을 읽어들이어 컨테이너의 환경설정을 한다

# 서블릿

- url 요청을 받아서 처리를 하는 모듈
- 여러개의 서블릿을 만들어서 각각의 url요청을 처리할 수 있다.
- 스프링은 DispatcherServlet 클래스 하나가 모든 url 요청을 처리한다

# 서블릿 지정

```
<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>

<servlet-mapping>
    <servlet-name>appServlet</servlet-name>
    <url-pattern>/</url-pattern>
</servlet-mapping>
```

모든 url요청에 대해서 DispatcherServlet클래스가 처리를 할것이고 환경설정은 / WEB-INF/spring/appServlet/servlet-context.xml 에 있는 파일을 사용한다는 의미이다.



# web.xml 예

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app version="2.5" 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 https://java.sun.com/xml/ns/javaee/web-app_2_5.xsd">

  <!-- 여기서부터 추가하기 시작 -->
  <!-- web.xml 파일 : 폼 입력 항목에 한글을 입력할 수 있도록 한다 -->
  <filter>
    <filter-name>CharacterEncodingFilter</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>
    <init-param>
      <param-name>forceEncoding</param-name>
      <param-value>true</param-value>
    </init-param>
  </filter>

  <filter-mapping>
    <filter-name>CharacterEncodingFilter</filter-name>
    <url-pattern>/*</url-pattern>
  </filter-mapping>

  <!-- 여기까지 붙여 넣는다 -->
```

# web.xml 예

```
<!-- 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>

<!-- Processes application requests -->
<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>

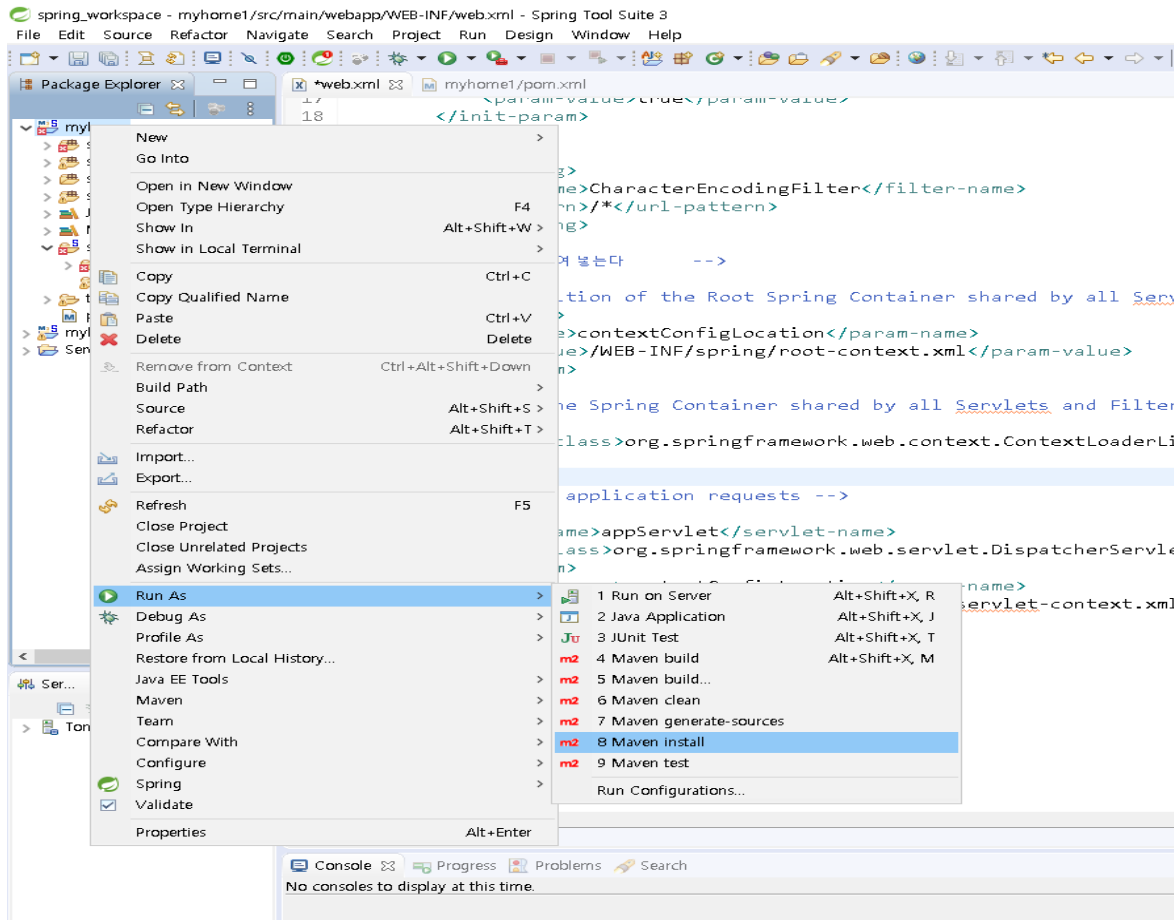
<servlet-mapping>
  <servlet-name>appServlet</servlet-name>
  <url-pattern>/</url-pattern>
</servlet-mapping>

</web-app>
```

# 프로젝트 빌딩하기

- 프로젝트 - 마우스 오른쪽 버튼 - run as - maven install

필요한 라이브러리를 다운받는다

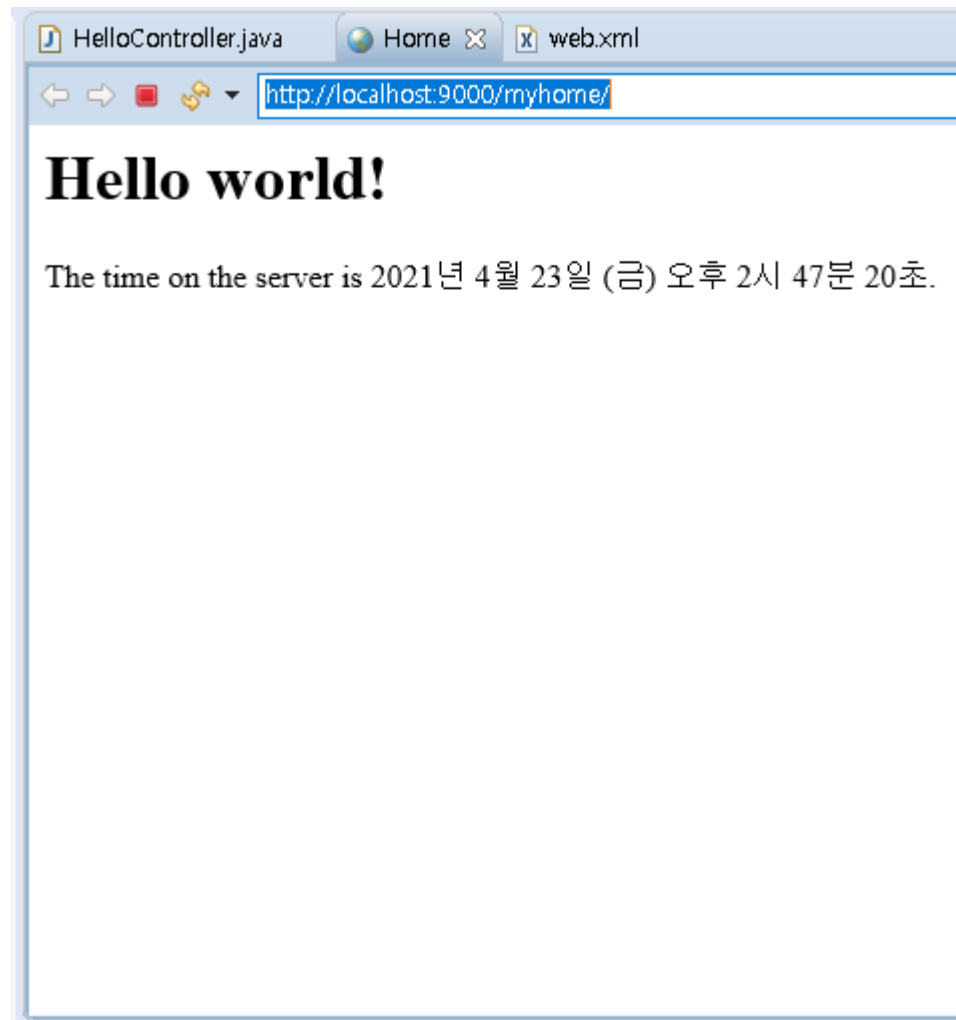


# BUILD SUCCESS 확인

```
[INFO] --- maven-war-plugin:2.2:war (default-war) @ myhome ---
[INFO] Packaging webapp
[INFO] Assembling webapp [myhome] in [C:\spring3\spring_workspace\myhome\target\myhome-1.0.0-BUILD-SNAPSHOT]
[INFO] Processing war project
[INFO] Copying webapp resources [C:\spring3\spring_workspace\myhome\src\main\webapp]
[INFO] Webapp assembled in [483 msecs]
[INFO] Building war: C:\spring3\spring_workspace\myhome\target\myhome-1.0.0-BUILD-SNAPSHOT.war
[INFO] WEB-INF\web.xml already added, skipping
[INFO]
[INFO] --- maven-install-plugin:2.4:install (default-install) @ myhome ---
[INFO] Installing C:\spring3\spring_workspace\myhome\target\myhome-1.0.0-BUILD-SNAPSHOT.war to C:\Users\user\.m2\repository\com\woori\my
[INFO] Installing C:\spring3\spring_workspace\myhome\pom.xml to C:\Users\user\.m2\repository\com\woori\myhome\1.0.0-BUILD-SNAPSHOT\myhor
[INFO] -----
[INFO] BUILD SUCCESS
[INFO] -----
[INFO] Total time: 4.301 s
[INFO] Finished at: 2021-04-23T11:01:23+09:00
[INFO] -----
```

# 실행하기

- 



# HomeController.java

**@Controller 가 있어야 객체가 만들어진다.**

```
@Controller
public class HomeController {

    private static final Logger logger = LoggerFactory.getLogger(HomeController.class);

    /**
     * @RequestMapping : url과 함수를 맵핑한다
     */
    @RequestMapping(value = "/", method = RequestMethod.GET)
    public String home(Locale locale, Model model) {
        logger.info("Welcome home! The client locale is {}.", locale);

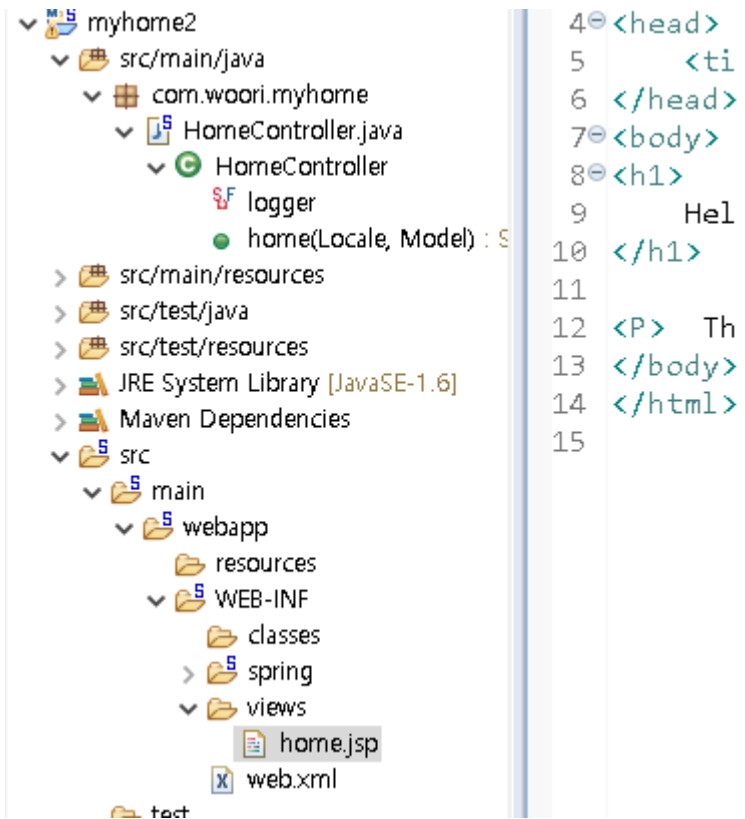
        Date date = new Date();
        DateFormat dateFormat = DateFormat.getDateInstance(DateFormat.LONG, DateFormat.LONG, locale);

        String formattedDate = dateFormat.format(date);

        model.addAttribute("serverTime", formattedDate );

        return "home";
    }
}
```

# home.jsp



The image shows a screenshot of an IDE with two panels. The left panel displays a project structure for 'myhome2'. The right panel shows the content of the 'home.jsp' file.

**Project Structure (Left Panel):**

- myhome2
  - src/main/java
    - com.woori.myhome
      - HomeController.java
        - HomeController
          - logger
            - home(Locale, Model) : S
  - src/main/resources
  - src/test/java
  - src/test/resources
  - JRE System Library [JavaSE-1.6]
  - Maven Dependencies
  - src
    - main
      - webapp
        - resources
        - WEB-INF
          - classes
          - spring
          - views
            - home.jsp
      - web.xml
  - test

```
4 <head>
5     <ti
6 </head>
7 <body>
8 <h1>
9     Hel
10 </h1>
11
12 <P>  Th
13 </body>
14 </html>
15
```

# HelloController 추가

- src/main/java/com/woori/myhome/HelloController 추가하기

**New Java Class**

Java Class  
Create a new Java class.

Source folder: myhome2/src/main/java Browse...

Package: com.woori.myhome Browse...

☐ Enclosing type: Browse...

Name: HelloController

Modifiers: ☒ public ☐ package ☐ private ☐ protected  
☐ abstract ☐ final ☐ static

Superclass: java.lang.Object Browse...

Interfaces: Add...  
Remove

Which method stubs would you like to create?

☐ public static void main(String[] args)  
☐ Constructors from superclass  
☒ Inherited abstract methods

Do you want to add comments? (Configure templates and default value [here](#))  
☐ Generate comments

? Finish Cancel



# /test url추가

```
package com.woori.myhome;

import org.springframework.stereotype.Controller;

@Controller
public class HelloController {

    @RequestMapping(value="/test")
    String test() {

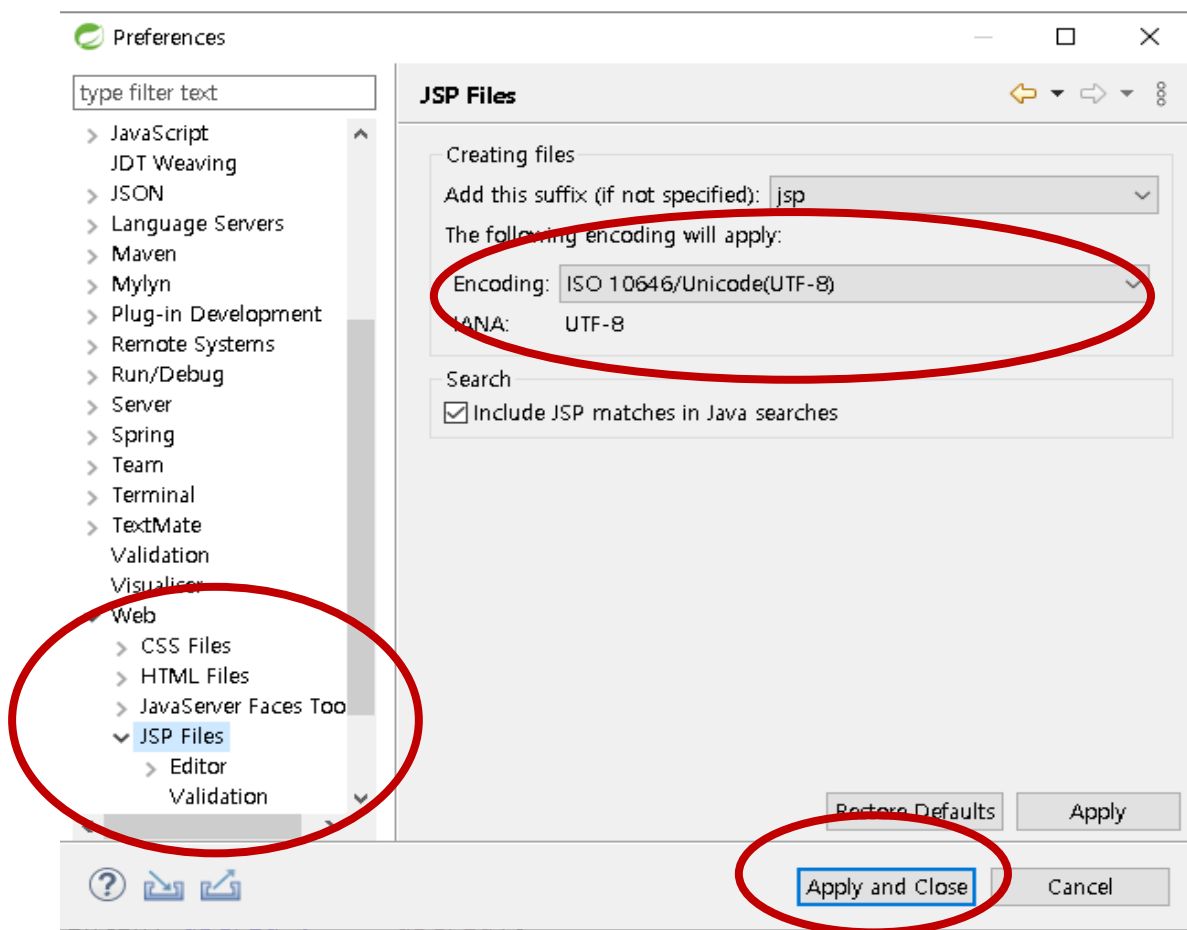
        System.out.println("test-----");

        //return 구문에서 보내는 문자열 jsp페이지 이름이 된다.
        //WEB-INF/views/test.jsp 가 존재해야한다

        return "test";
    }
}
```

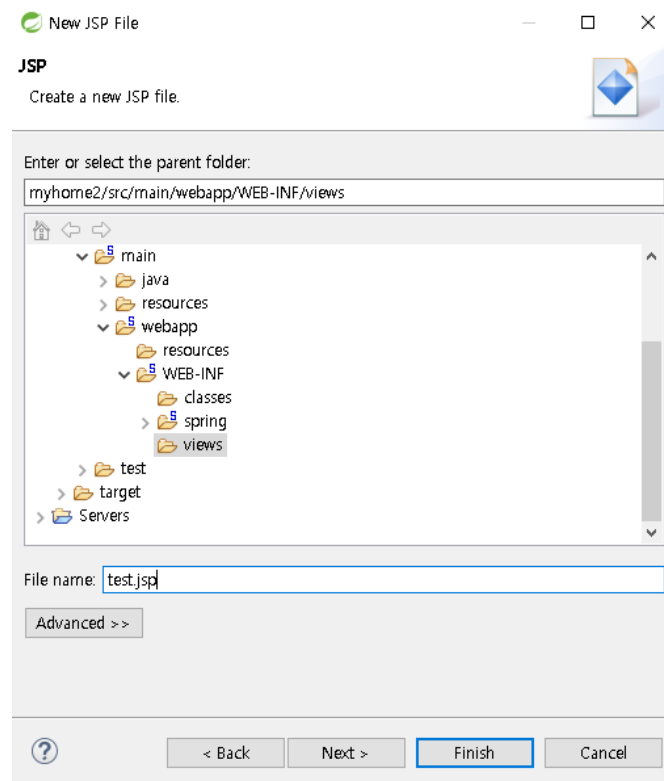
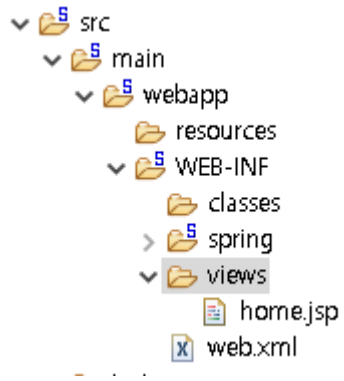
# jsp 문자셋 설정하기

- window - preferences - Web - JSP files - Encoding ISO 10646/Unicode(UTF-8) - 적용하기 누르기



# test.jsp

- src - main - webapp - WEB-INF - views - test.jsp 추가하기



# test.jsp

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>Insert title here</title>
</head>
<body>
<h2>테스트 페이지입니다</h2>

<p>모든 요청은 DispatcherServlet가 처리하여 컨트롤러를 호출하고 컨트롤러의 @RequestMapping 을 이용하여 해당 함수가 호출된다 </p>
<p>Controller 함수내에서 return 구문에 호출할 jsp 파일명을 기술한다 </p>
</body>
</html>
```

# 결과

← → ↻ ⓘ localhost:9000/myhome/test

## 테스트 페이지입니다

모든 요청은 `DispatcherServlet`가 처리하여 컨트롤러를 호출하고 컨트롤러의 `@RequestMapping` 을 이용하여 해당 함수가 호출된다

`Controller` 함수내에서 `return` 구문에 호출할 `jsp` 파일명을 기술한다

# 파라미터 처리하기

```
//파라미터 처리하기
//http://localhost:9000/myhome/test2?userid=test&pwd=1234
@RequestMapping(value="/test2")
String test2(Model model, String userid, String pwd)
{
    System.out.println("userid : " + userid);
    System.out.println("pwd : " + pwd);

    //Model - request 객체 대신에

    model.addAttribute("userid", userid);
    model.addAttribute("pwd", pwd);

    //request.setAttribute("userid", userid);
    //request.setAttribute("pwd", pwd);

    return "test2";
}
```

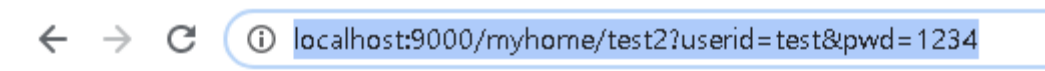
# test2.jsp

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>Insert title here</title>
</head>
<body>
<h2>request.getAttribute 활용</h2>
<%
String userid = (String)request.getAttribute("userid");
String pwd = (String)request.getAttribute("pwd");
%>
<%=userid%><br/>
<%=pwd %><br/>
<br/>

<h2>표현식 활용</h2>
${userid} <br/>
${pwd} </br>
</body>
</html>
```

# 결과 확인

- `http://localhost:9000/myhome/test2?userid=test&pwd=1234`



## `request.getAttribute` 활용

test

1234

## 표현식 활용

test

1234



# get방식 전달

```
//http://localhost:9000/myhome/add?x=10&y=20
@RequestMapping(value="/add")
String add(Model model, HttpServletRequest req)
{
    int x = Integer.parseInt(req.getParameter("x").toString());
    int y = Integer.parseInt(req.getParameter("y").toString());

    System.out.println("x : " + x);
    System.out.println("y : " + y);

    //Model - request 객체 대신에

    model.addAttribute("x", x);
    model.addAttribute("y", y);
    model.addAttribute("add", x+y);

    return "add";
}
```

# add.jsp

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>Insert title here</title>
</head>
<body>
<h3>${x} + ${y} = ${add}</h3>
</body>
</html>
```

# HATEOAS

- Hypermedia As The Engine Of Application State 의 약자로 REST 아키텍처의 한 구성요소이다

```
//http://localhost:9000/myhome/sub/10/20
@RequestMapping(value="/sub/{x}/{y}")
String sub(Model model, @PathVariable("x")int x, @PathVariable("y")int y)
{
    System.out.println("x : " + x);
    System.out.println("y : " + y);
    System.out.println("x-y : " + (x-y));

    model.addAttribute("x", x);
    model.addAttribute("y", y);
    model.addAttribute("sub", x-y);

    return "add";
}
```

# sub.jsp

- <http://localhost:9000/myhome/sub/5/6>

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>Insert title here</title>
</head>
<body>
    <h3>${x} - ${y} = ${sub}</h3>
</body>
</html>
```

# POST 전송

Postman

File Edit View Help

Home Workspaces Reports Explore

Search Postman

Upgrade

Overview POST http://localhost:9000/myhome/test2

No Environment

Save

POST http://localhost:9000/myhome/test2

Send

Params Authorization Headers (10) Body Pre-request Script Tests Settings

none form-data x-www-form-urlencoded raw binary GraphQL

<input checked="" type="checkbox"/>	userid	test	
<input checked="" type="checkbox"/>	pwd	1234	
	Key	Value	Description

Body Cookies (1) Headers (6) Test Results

Status: 200 OK Time: 14 ms Size: 437 B Save Response

Pretty Raw Preview Visualize HTML

```
<html>
  <head>
  </head>
  <body>
    <h2>request.getAttribute 활용</h2>
    test<br/>
    1234<br/>
    <br/>
    <h2>표현식 활용</h2>
  </body>
</html>
```

Find and Replace Console

```
userid : test
pwd : null
```

Bootcamp Runner Trash

# 과제 1

다음 형태로 url 전달시 경과가 나오도록 작성할것 (사각형 면적 구하기)

클래스명 : ShapeController

두개의 url모두 만들기

<http://localhost:9000/myhome/rect/width=5&height=7>

<http://localhost:9000/myhome/rect/5/7>

출력형태

가로 : 5   세로 : 7   면적 : 35

## 과제2

다음 형태로 url 전달시 경과가 나오도록 작성할것 (구구단 출력하기)

클래스명 : GuguController

<http://localhost:9000/myhome/gugu/3>

출력결과

$$3 \times 1 = 3$$

$$3 \times 2 = 6$$

$$3 \times 3 = 9$$

$$3 \times 4 = 12$$

$$3 \times 5 = 15$$

...

$$3 \times 9 = 27$$

## 과제3

다음 형태로 url 전달시 결과가 나오도록 작성할것 ()

클래스명 : Controller

```
List<String> flowerList=new ArrayList<String>();
```

```
list.add("장미");
```

```
list.add("목단");
```

```
list.add("수련");
```

```
list.add("백합");
```

```
list.add("은방울꽃");
```

```
list.add("제비꽃");
```

위 배열을 컨트롤러에 넣어놓고 파라미터로 인덱스가 전달되면 해당 인덱스의 꽃이름을 출력하고 만일 인덱스를 초화하면 해당하는 꽃이 없다고 출력해야 합니다.

예) http://localhost:9000/myhome/flower/0

장미

http://localhost:9000/myhome/flower/3

백합

http://localhost:9000/myhome/flower/10

해당하는 꽃이 없음



게시판 만들기

# 게시판 만들기

- Controller
  - url을 받아서 처리한다, 여러개의 서비스를 소유한다
  - @Controller
- Service
  - Service 클래스, 여러개의 Dao를 소유할 수 있고, 트랜잭션 처리를 담당한다
  - @Service가 지정되어야 한다
  - 인터페이스를 만들어야 한다
- Dao
  - Data Access Object ( 데이터베이스와 직접 접촉하여 데이터 액세스를 한다
  - @Repository(객체명)
  - 인터페이스를 만들어야 한다
- Dto
  - Data Transfer Object(보통 테이블에 하나씩 Dto를 만든다. 각 필드의 값을 저장할 클래스,
  - join해서 가져올 경우에는 두개이상의 테이블 필드를 가질 수 도 있다)

# 전체 파일

- jsp 파일
  - board\_list.jsp
  - board\_write.jsp
  - board\_view.jsp
- 클래스파일
  - BoardDto.java
  - BoardDao.java -interface
  - BoardDaoImpl - BoardDao 인터페이스 구현
  - BoardService.java - interface
  - BoardServiceImpl.java - BoardService 인터페이스 구현
  - BoardController.java

# BoardDto.java

- class 안에서 마우스 오른쪽 누르고 generate - source , constructor 생성하기

```
package com.mycompany.myhome1.board;

public class BoardDto {
    private String id="";
    private String title="";
    private String contents="";
    private String writer="";
    private String wdate="";

    public BoardDto() {
        super();
        // TODO Auto-generated constructor stub
    }

    public BoardDto(String id, String title, String contents, String writer, String wdate) {
        super();
        this.id = id;
        this.title = title;
        this.contents = contents;
        this.writer = writer;
        this.wdate = wdate;
    }
}
```

# BoardDao.java

```
package com.mycompany.myhome1.board;

import java.util.List;

public interface BoardDao {
    List<BoardDto> getList();
    BoardDto getView(String id);
    void insert(BoardDto dto);
    void update(BoardDto dto);
    void delete(String id);
}
```

# BoardDaoImpl.java

```
@Repository("boardDao")
public class BoardDaoImpl implements BoardDao{

    List<BoardDto> list = new ArrayList<BoardDto>();

    public BoardDaoImpl()
    {
        for(int i=1; i<=10; i++)
        {
            list.add( new BoardDto(""+i,"제목"+i, "내용"+i, "작성자"+i, "작성일"+i));
        }
    }

    @Override
    public List<BoardDto> getList() {
        return list;
    }

    @Override
    public void insert(BoardDto dto) {
        list.add(dto);
    }

    @Override
    public BoardDto getView(String id) {

        int nID= Integer.parseInt(id);
        return list.get(nID);
    }

    @Override
    public void update(BoardDto dto) { }

    @Override
    public void delete(String id) { }

}
```

# BoardService.java

```
package com.mycompany.myhome1.board;
```

```
import java.util.List;
```

```
public interface BoardService {  
    List<BoardDto> getList();  
    void insert(BoardDto dto);  
    BoardDto getView(String id);  
    void update(BoardDto dto);  
    void delete(String id);  
}
```

# BoardServiceImpl

```
@Service("boardService")
public class BoardServiceImpl implements BoardService{

    @Resource(name="boardDao") //boardDao라는 아이디를 갖는 객체를 찾아서 참조를 전달한다
    BoardDao boardDao;

    @Override
    public List<BoardDto> getList() {
        // TODO Auto-generated method stub
        return boardDao.getList();
    }

    @Override
    public void insert(BoardDto dto) {
        boardDao.insert(dto);
    }

    @Override
    public BoardDto getView(String id) {

        return boardDao.getView(id);
    }

    @Override
    public void update(BoardDto dto) {
        boardDao.update(dto);
    }

    @Override
    public void delete(String id) {
        boardDao.delete(id);
    }
}
```



# BoardController.java

```
@Controller
public class BoardController {

    // @Autowired - 특정객체를 고를 수 없어서 그냥 타입이 맞으면 시스템에 알아서 객체를 주입시켰음
    //              동일한 인터페이스를 상속받았을때 .....

    @Resource(name="boardService")
    BoardService boardService;

    @RequestMapping(value="/board/list")
    public String board_list(Model model)
    {
        List<BoardDto> list = boardService.getList();

        model.addAttribute("boardList", list); // 데이터 읽어와서 request 객체에 저장 - Model 클래스

        return "/board/list";
    }

    @RequestMapping(value="/board/view")
    public String board_view(Model model, String id, HttpServletRequest request, BoardDto dto )
    {
        // 파라미터의 이름이 중요하지 순서는 전혀 상관없다.
        String iid = request.getParameter("id");

        System.out.println("id(spring) : " + id);
        System.out.println("iid(old) : " + iid);

        System.out.println("id(dto) : " + dto.getId());

        model.addAttribute("boardDto", boardService.getView(id));
        return "/board/view";
    }
}
```

# BoardController.java

```
//write.jsp 로 이동하기
@RequestMapping(value="/board/write")
public String board_write(Model model, String id, HttpServletRequest request, BoardDto dto )
{
    model.addAttribute("boardDto", new BoardDto());

    return "/board/write";
}

@RequestMapping(value="/board/save")
public String board_save(Model model, String id, HttpServletRequest request, BoardDto dto )
{
    boardService.insert(dto);
    return "redirect:/board/list";
}

@RequestMapping(value="/board/modify")
public String board_modify(Model model, String id, BoardDto dto )
{
    model.addAttribute("boardDto", boardService.getView(id));

    return "/board/write";
}

@RequestMapping(value="/board/update")
public String board_update(Model model, BoardDto dto )
{
    boardService.update(dto);
    return "redirect:/board/list";
}

@RequestMapping(value="/board/delete")
public String board_delete(Model model, BoardDto dto, String id )
{
    boardService.delete(dto.getId());
    return "redirect:/board/list";
}
}
```

# 디비 스키마

# MyBatis

# MyBatis 연동에 필요한 라이브러리

```
<!-- MyBatis용 ***** -->
<!-- https://mvnrepository.com/artifact/org.mybatis/mybatis -->
<dependency>
  <groupId>org.mybatis</groupId>
  <artifactId>mybatis</artifactId>
  <version>3.4.6</version>
</dependency>

<!-- mybatis랑 spring 연동 라이브러리 -->
<!-- https://mvnrepository.com/artifact/org.mybatis/mybatis-spring -->
<dependency>
  <groupId>org.mybatis</groupId>
  <artifactId>mybatis-spring</artifactId>
  <version>1.3.3</version>
</dependency>

<!-- https://mvnrepository.com/artifact/org.springframework/spring-jdbc -->
<dependency>
  <groupId>org.springframework</groupId>
  <artifactId>spring-jdbc</artifactId>
  <version>${org.springframework-version}</version>
</dependency>

<!-- db connection pool -->
<!-- https://mvnrepository.com/artifact/org.apache.commons/commons-dbcp2 -->
<dependency>
  <groupId>org.apache.commons</groupId>
  <artifactId>commons-dbcp2</artifactId>
  <version>2.7.0</version>
</dependency>
```

# 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"
  xmlns:aop="http://www.springframework.org/schema/aop"
  xmlns:context="http://www.springframework.org/schema/context"
  xmlns:jdbc="http://www.springframework.org/schema/jdbc"
  xmlns:mybatis-spring="http://mybatis.org/schema/mybatis-spring"
  xmlns:tx="http://www.springframework.org/schema/tx"
  xmlns:mvc="http://www.springframework.org/schema/mvc"
  xsi:schemaLocation="http://www.springframework.org/schema/jdbc http://www.springframework.org/schema/jdbc/spring-jdbc-4.3.xsd
    http://mybatis.org/schema/mybatis-spring http://mybatis.org/schema/mybatis-spring-1.2.xsd
    http://www.springframework.org/schema/mvc http://www.springframework.org/schema/mvc/spring-mvc-4.3.xsd
    http://www.springframework.org/schema/beans http://www.springframework.org/schema/beans/spring-beans.xsd
    http://www.springframework.org/schema/context http://www.springframework.org/schema/context/spring-context-4.3.xsd
    http://www.springframework.org/schema/aop http://www.springframework.org/schema/aop/spring-aop-4.3.xsd
    http://www.springframework.org/schema/tx http://www.springframework.org/schema/tx/spring-tx-4.3.xsd">

  <!-- db와 관련된건은 반드시 여기에 -->
  <bean id="dataSource"
    class="org.apache.commons.dbcp2.BasicDataSource" destroy-method="close">
    <property name="driverClassName">
      <value>oracle.jdbc.driver.OracleDriver</value>
    </property>
    <property name="url">
      <value>jdbc:oracle:thin:@127.0.0.1:1521:xe</value>
    </property>
    <property name="username">
      <value>user01</value>
    </property>
    <property name="password">
      <value>1234</value>
    </property>
  </bean>
```

# myBatis 설정파일

src

└ resource

└ mybatis

└ mybatis-config.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE configuration PUBLIC "-//mybatis.org//DTD Config 3.0//EN"
"http://mybatis.org/dtd/mybatis-3-config.dtd">

<configuration>

<!-- 원래 클래스명은 패키지까지 포함하기때문에 너무 길어서 짧은 별명을 만들어서 접근하기 위해 기술한다 -->
<typeAliases>
  <typeAlias alias="GuestbookDto" type="com.mycompany.myapp.guestbook.GuestbookDto"/>
</typeAliases>

<mappers>
  <mapper resource= "/com/mycompany/myapp/mapper/Guestbook.xml"/>
</mappers>

</configuration>
```

# Guestbook.xml파일 만들기

src

└ main

└ java

└ com

└ mycompany

└ myhome

└ mapper

└ Guestbook.xml파일



# Guestbook.xml 예시

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE mapper
  PUBLIC "-//mybatis.org//DTD Mapper 3.0//EN"
  "http://mybatis.org/dtd/mybatis-3-mapper.dtd">

<mapper namespace="Guestbook">
  <!-- MyBatis ${필드명} 맵핑이 먼저 이루어진다 #{필드명} 맵핑이 나중에 이루어진다 쿼리를 콘솔에서 보면 $가 붙은건
  ? 가 아니라 실제 값으로 보여요 #이 붙은거는 ? 로 보인다 나중에 실제 값하고 결합한다 #붙은건 데이터 타입에 따라서 자동으로
  '를 붙여준다 $는 그냥 그 위치에 문자열을 내보낼 뿐이라서 '가 필요하다면 직접 붙여줘야 한다 $는 데이터를 문자열을 줘도 그대로
  옮겨놓는다 테이블명이나 필드명이 추가되거나 할때는 ${필드명} 또는 ${테이블명} -->
  <select id="Guestbook_getList" parameterType="GuestbookDto" resultType="GuestbookDto">

    select * from
    (
    select
      A.id
      ,A.title
      ,A.writer
      ,A.contents --내용
      ,to_char(A.wdate, 'yyyy-mm-dd') wdate
      ,row_number() over (order by A.id desc) as num
      ,ceil(row_number () over (order by A.id)/10-1) as pg
    from guestbook A
    )where pg=#{pg}
```

# MyBatis 연동에 필요한 라이브러리

Ajax

# ajax1.jsp

```
><body>
    <div id="demo">여기에 결과가</div>
    <button id="btnCall" onclick="loadDoc()">ajax call</button>
</body>
</html>
><script>

function loadDoc() {
    var xhttp = new XMLHttpRequest();
    xhttp.onreadystatechange = function() {
        if (this.readyState == 4 && this.status == 200) {
            document.getElementById("demo").innerHTML =
                this.responseText;
        }
    };
    xhttp.open("GET", "ajax_info.jsp", true);
    xhttp.send();
}
</script>
```

# ajax2.jsp

```
<body>
  <form name="myform" id="myform">
    <input type="text" name="userid" id="userid" >
    <input type="text" name="password" id="password" >
    <button id="btnCall" onclick="loadDoc()" type="button">ajax call</button>
  </form>

  <div id="demo"></div>
</body>
</html>
<script>

function loadDoc() {
  var userid=document.getElementById("userid").value;
  var password=document.getElementById("password").value;

  var url = "receive1.jsp?userid="+userid+"&password="+password;
  console.log(url);

  var xhttp = new XMLHttpRequest();
  xhttp.onreadystatechange = function() {
    if (this.readyState == 4 && this.status == 200) {
      document.getElementById("demo").innerHTML =
        this.responseText;
    }
  };
  xhttp.open("GET", url, true);
  xhttp.send();
}
</script>
```

# ajax3.jsp

```
<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.5.1/jquery.min.js"></script>
</head>
<body>
    <div id="demo"></div>
    <button id="btnCall" type="button">ajax call</button>
</body>
</html>
<script>
$(function(){
    $("#btnCall").click( ()=>{
        $.ajax({
            url: "ajax_info.txt",
        }).done(function(data) {
            $("#demo" ).html( data );
        }).fail(function(error){
            console.log(error);
        });
    });
});
</script>
```

# ajax4.jsp

```
<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.5.1/jquery.min.js"></script>
</head>
<body>
  <form name="myform" id="myform">
    <input type="text" name="userid" id="userid" >
    <input type="text" name="password" id="password" >
    <button id="btnCall" type="button">ajax call</button>
  </form>

  <div id="demo"></div>
</body>
</html>
<script>

$(()=>{
  $("#btnCall").click( ()=>{

    $.ajax({
      url: "receive1.jsp",
      data:{"userid":$("#userid").val(), password:$("#password").val()},
      method:"POST",
      dataType:"text"
    }).done(function(data) {
      console.log(data);
    }).fail(function(jqXHR, textStatus){
      console.log(jqXHR + textStatus);
    });
  });
});
</script>
```

# ajax5.jsp

```
<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.5.1/jquery.min.js"></script>
</head>
<body>
  <form name="myform" id="myform">
    <input type="text" name="userid" id="userid" >
    <input type="text" name="password" id="password" >
    <button id="btnCall" type="button">ajax call</button>
  </form>

  <div id="demo"></div>
</body>
</html>
<script>

$(()=>{
  $("#btnCall").click( ()=>{

    $.ajax({
      url: "receive2.jsp",
      data:{ "userid":$("#userid").val(), password:$("#password").val()},
      method:"POST",
      dataType:"json"
    }).done(function(data) {
      console.log(data);
    }).fail(function(jqXHR, textStatus){
      console.log(jqXHR + textStatus);
    });
  });
});
</script>
```



# ajax6.jsp

```
<body>
  <div class="container">
    <h2>게시판</h2>
    <p></p>
    <table class="table" id="mytable">
      <thead class="thead-dark">
        <tr>
          <th>번호</th>
          <th>제목</th>
          <th>작성자</th>
          <th>조회수</th>
        </tr>
      </thead>
      <tbody id="tbody">
      </tbody>
    </table>

  </div>
</body>
</html>
<script>

$(()=>{

  $.ajax({
    url:"board_list.jsp",
    data:{"page":1},
    method:"POST",
    dataType:"json"
  }).done(function(data) {
    console.log( data );
    // jquery를 사용한 방법

    for(var i =$("#mytable tr").length-2;i>-1;i--){
      $("#mytable tr:last").remove();
    }

    data.forEach(function(e){
      $('#mytable > tbody:last').append('<tr><td>'+e.id+ '</td><td>' + e.title
        + '</td><td>' + e.writer+ '</td><td>' + e.hit + '</td></tr>');
    });
  }).fail(function(jqXHR, textStatus){
    console.log(jqXHR + textStatus);
  });
});

</script>
```

# receive1.jsp

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
<%
String userid=request.getParameter("userid");
String password=request.getParameter("password");

%>
아이디 : <%=userid%> <br>
패스워드 : <%=password%>
```

---

# receive2.jsp

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
<%
String userid=request.getParameter("userid");
String password=request.getParameter("password");

%>
{"userid":"<%=userid%>","password":"<%=password%>"}
```

--

--

MyBatis 연동

# Mybatis 라이브러리(pom.xml에 추가)

```
<!-- mybatis랑 spring 연동 라이브러리 -->
<!-- https://mvnrepository.com/artifact/org.mybatis/mybatis-spring -->
<dependency>
    <groupId>org.mybatis</groupId>
    <artifactId>mybatis-spring</artifactId>
    <version>1.3.3</version>
</dependency>
<!-- https://mvnrepository.com/artifact/org.springframework/spring-jdbc -->
<dependency>
    <groupId>org.springframework</groupId>
    <artifactId>spring-jdbc</artifactId>
    <version>${org.springframework-version}</version>
</dependency>
<!-- https://mvnrepository.com/artifact/org.apache.commons/commons-dbcp2 -->
<dependency>
    <groupId>org.apache.commons</groupId>
    <artifactId>commons-dbcp2</artifactId>
    <version>2.7.0</version>
</dependency>
<!-- ***** -->
```

# ojdbc처리하기

- WEB-INF 폴더 아래에 lib 폴더 추가
- ojdbc6



Rest ful API 서버

# pom.xml 파일에 다음 추가

```
<dependency>  
    <groupId>org.codehaus.jackson</groupId>  
    <artifactId>jackson-mapper-asl</artifactId>  
    <version>1.9.13</version>  
</dependency>  
  
<dependency>  
    <groupId>com.fasterxml.jackson.core</groupId>  
    <artifactId>jackson-databind</artifactId>  
    <version>2.8.5</version>  
</dependency>
```

# TestController

```
import java.util.HashMap;

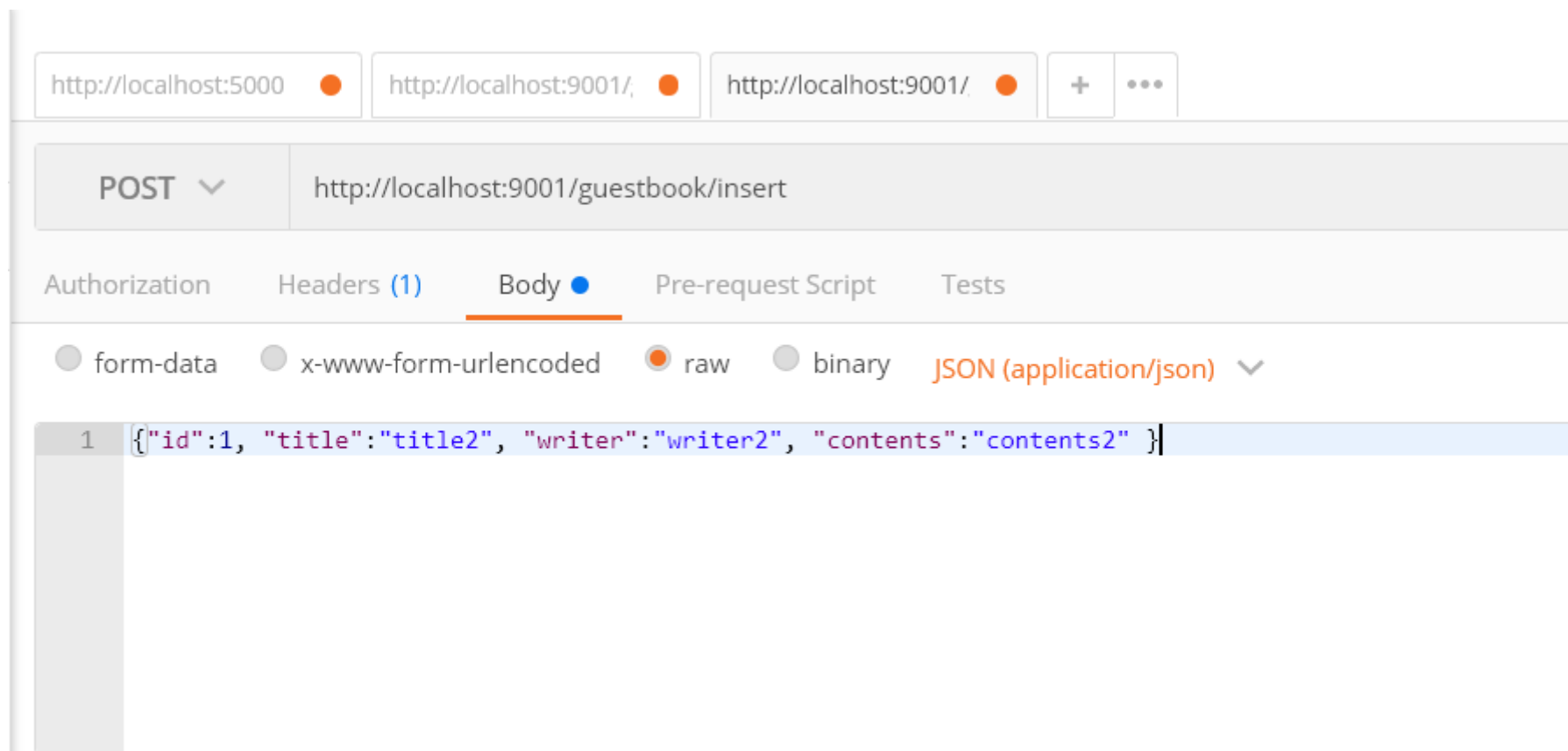
import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.RequestBody;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.ResponseBody;

@Controller
public class TestController {
    @RequestMapping("/list")
    @ResponseBody
    HashMap<String, String > dataList(){
        System.out.println("*****");
        HashMap<String, String > map = new HashMap<String, String >();
        map.put("name", "홍길동");
        map.put("age", "23");
        map.put("phone", "010-0000-000");

        return map;
    }

    @RequestMapping("/insert")
    @ResponseBody
    String insert(@RequestBody HashMap<String, String>data){
        System.out.println("*****");
        System.out.println(data.get("title"));
        System.out.println(data.get("writer"));
        System.out.println(data.get("contents"));
        return "success";
    }
}
```

# json통신 테스트



# form 태그 전송시

POST ▾

http://localhost:9001/guestbook/insert2

Authorization

Headers (1)

Body ●

Pre-request Script

Tests

☐ form-data

☒ x-www-form-urlencoded

☐ raw

☐ binary

	Key	Value
<input checked="" type="checkbox"/>	id	1
<input checked="" type="checkbox"/>	title	dddd
<input checked="" type="checkbox"/>	writer	eee
<input checked="" type="checkbox"/>	contents	rrrr
	New key	Value

Body

Cookies

Headers (7)

Test Results

Pretty

Raw

Preview

Text ▾

1	success
---	---------

# ajax 호출

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>Insert title here</title>
<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.5.1/jquery.min.js"></script>
</head>
<body>
    <form name="myform" id="myform">
        <input type="text" name="userid" id="userid" >
        <input type="text" name="password" id="password" >
        <button id="btnCall" type="button">ajax call</button>
    </form>

    <div id="demo"></div>
</body>
</html>|
<script>

$(function(){
    $("#btnCall").click(function(){

        $.ajax({
            url: "receive1.jsp",
            data:{"userid":$("#userid").val(), password:$("#password").val()},
            method:"POST",
            dataType:"text"
        }).done(function(data) {
            console.log(data);
        }).fail(function(jqXHR, textStatus){
            console.log(jqXHR + textStatus);
        });
    });
})
</script>
```