



SPRING 02



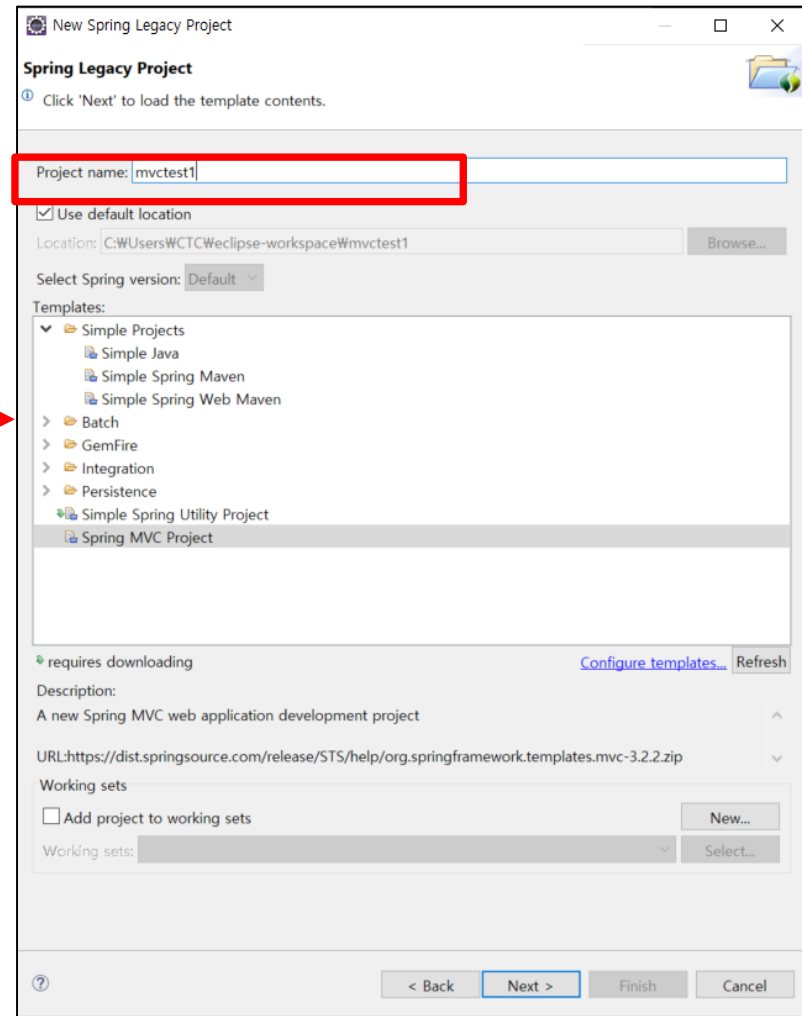
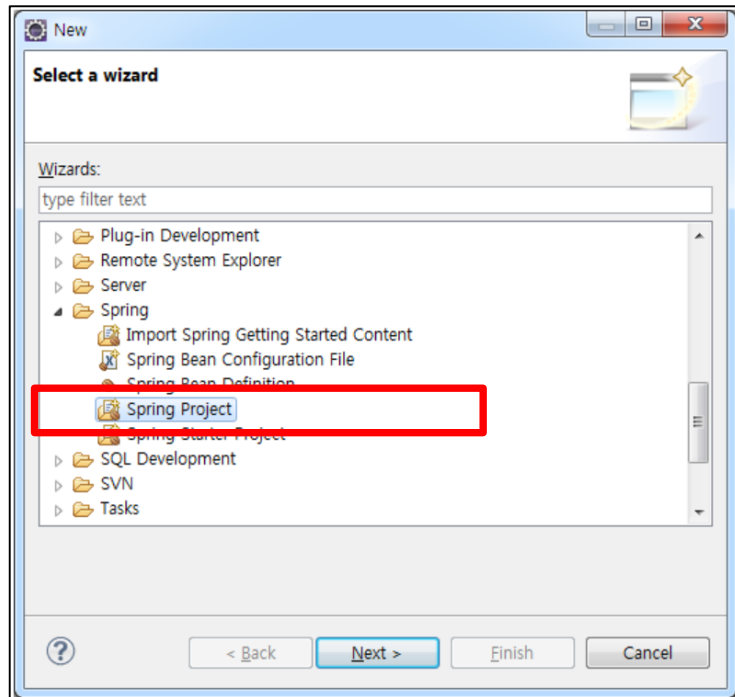
김규석 교수
(스프링프레임워크)

● 학습목표

- ✓ SPRING 프로젝트 생성 및 실습
- ✓ MVC 패턴 이해

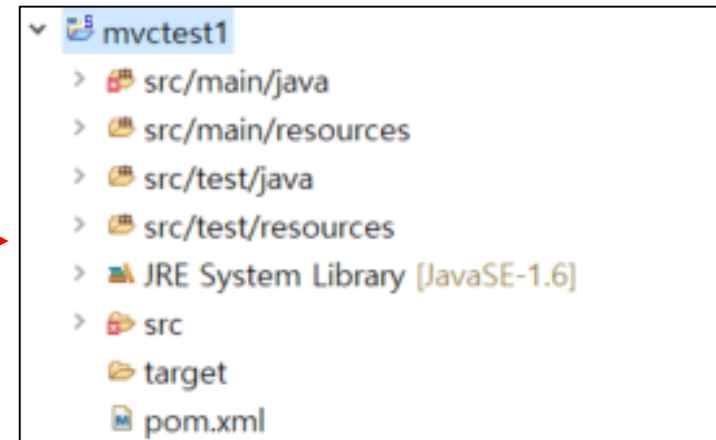
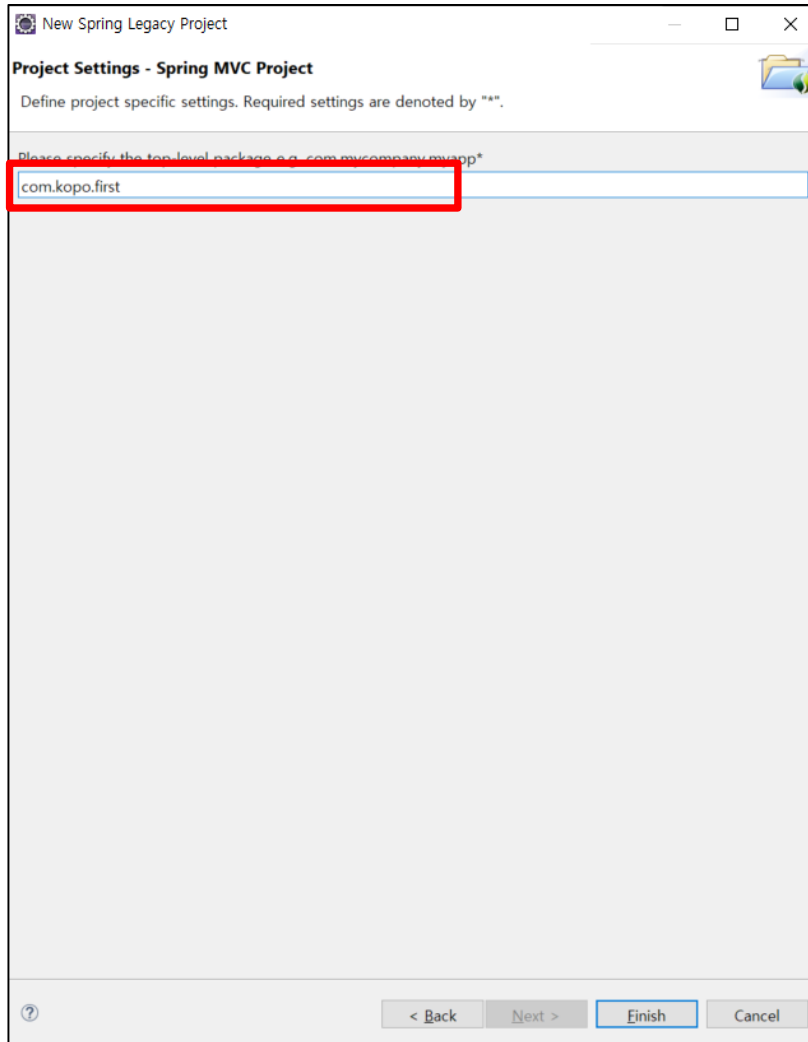
● Spring 프로젝트 생성 및 실습

✓ 프로젝트 생성하기



● Spring 프로젝트 생성 및 실습

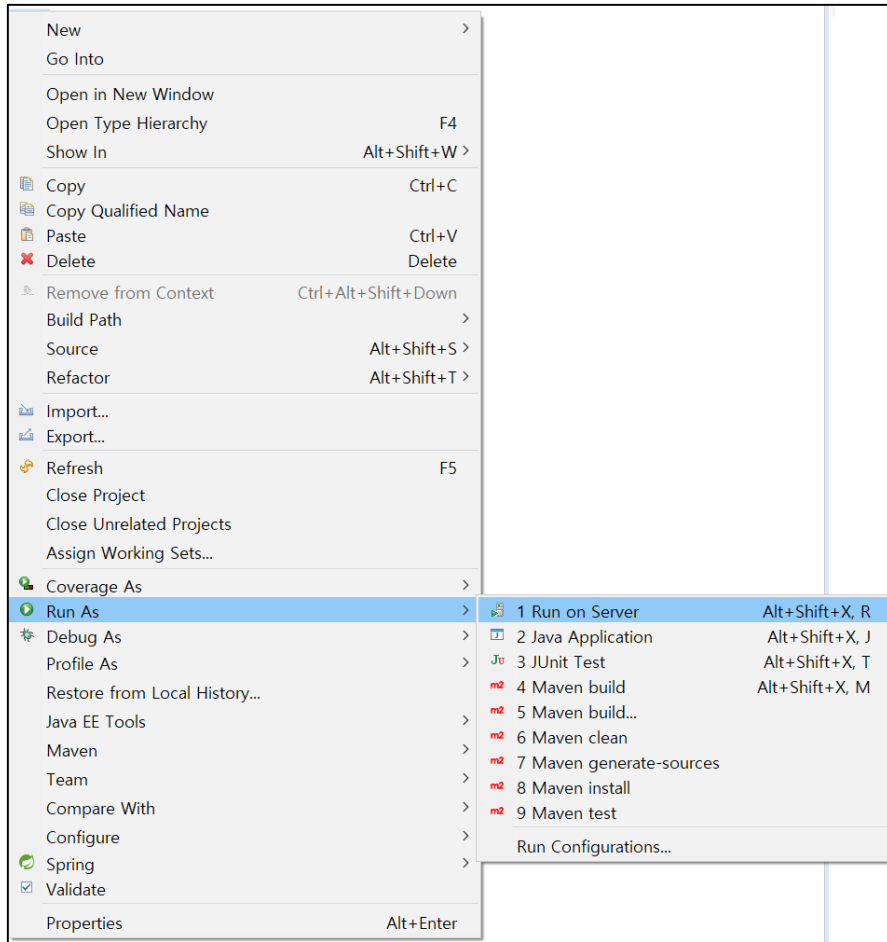
✓ 프로젝트 생성하기



● Spring 프로젝트 생성 및 실습

✓ 프로젝트 실행하기

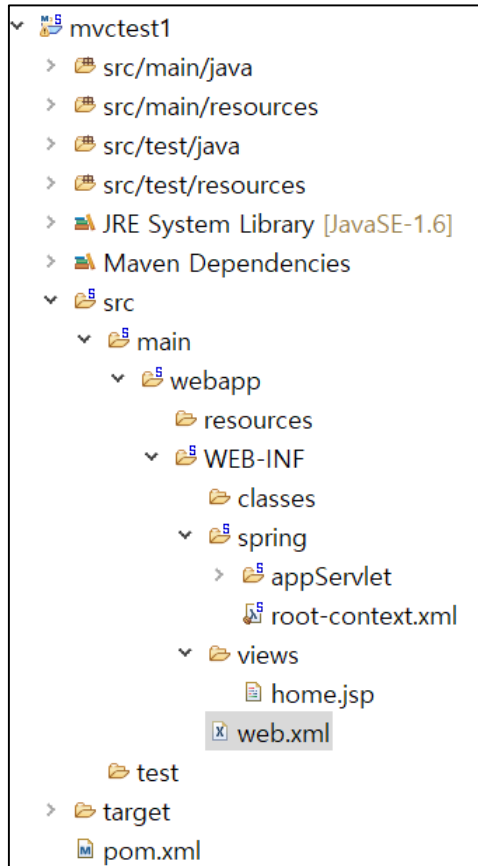
– 한글이 깨져서 보임



● Spring 프로젝트 생성 및 실습

✓ 프로젝트 실행하기

– web.xml 파일을 열어서 UTF-8로 Encoding 하는 코드 추가

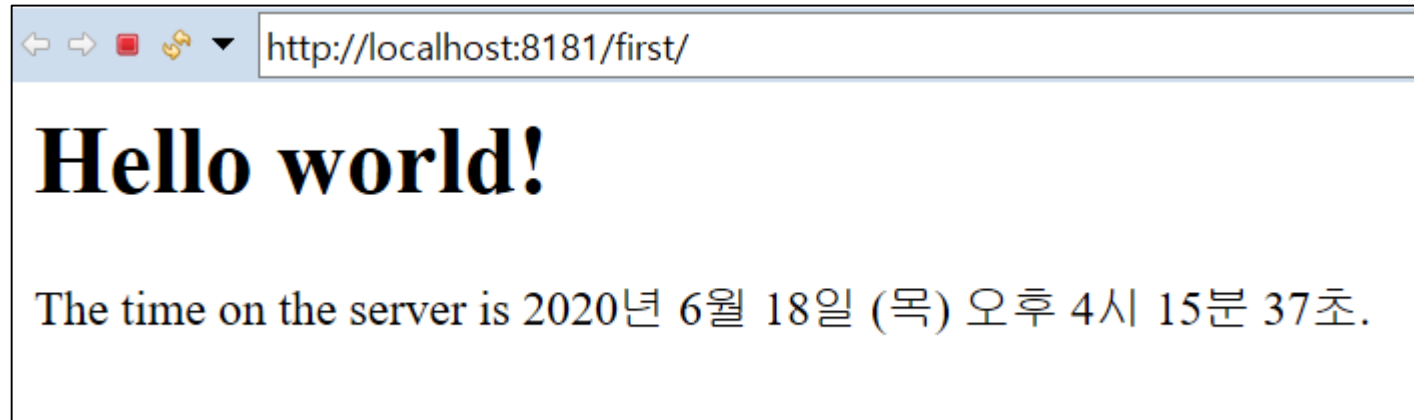


```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <web-app version="2.5" xmlns="http://java.sun.com/xml/ns/javaee"
3     xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
4     xsi:schemaLocation="http://java.sun.com/xml/ns/javaee https://java.sun.com/xml/ns/javaee/web-app_2_5.xsd">
5
6     <!-- The definition of the Root Spring Container shared by all Servlets and Filters -->
7     <context-param>
8         <param-name>contextConfigLocation</param-name>
9         <param-value>/WEB-INF/spring/root-context.xml</param-value>
10    </context-param>
11
12    <!-- Creates the Spring Container shared by all Servlets and Filters -->
13    <listener>
14        <listener-class>org.springframework.web.context.ContextLoaderListener</listener-class>
15    </listener>
16
17    <!-- Processes application requests -->
18    <servlet>
19        <servlet-name>appServlet</servlet-name>
20        <servlet-class>org.springframework.web.servlet.DispatcherServlet</servlet-class>
21        <init-param>
22            <param-name>contextConfigLocation</param-name>
23            <param-value>/WEB-INF/spring/appServlet/servlet-context.xml</param-value>
24        </init-param>
25        <load-on-startup>1</load-on-startup>
26    </servlet>
27
28    <servlet-mapping>
29        <servlet-name>appServlet</servlet-name>
30        <url-pattern>/</url-pattern>
31    </servlet-mapping>
32
33    <filter>
34        <filter-name>encodingFilter</filter-name>
35        <filter-class>org.springframework.web.filter.CharacterEncodingFilter</filter-class>
36        <init-param>
37            <param-name>encoding</param-name>
38            <param-value>UTF-8</param-value>
39        </init-param>
40        <init-param>
41            <param-name>forceEncoding</param-name>
42            <param-value>true</param-value>
43        </init-param>
44    </filter>
45    <filter-mapping>
46        <filter-name>encodingFilter</filter-name>
47        <url-pattern>/*</url-pattern>
48    </filter-mapping>
49 </web-app>
```

● Spring 프로젝트 생성 및 실습

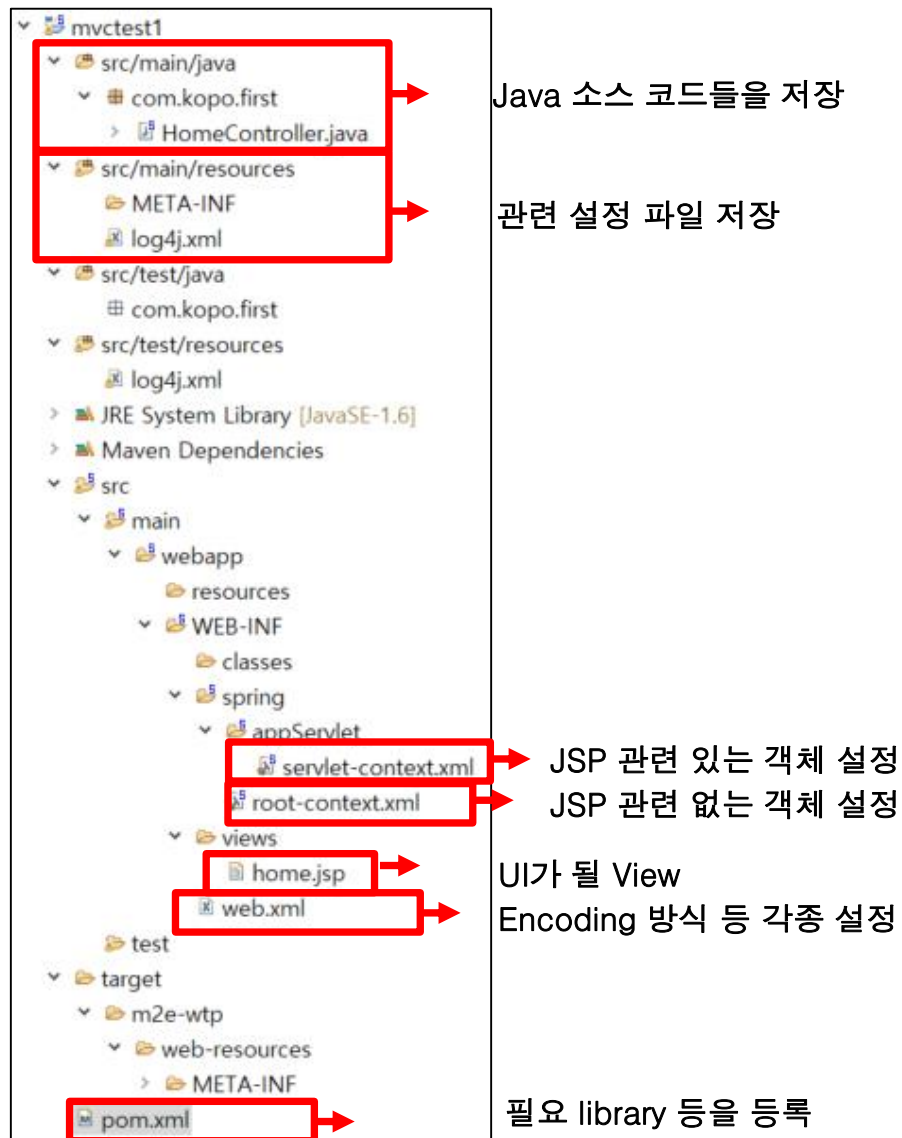
✓ 프로젝트 실행하기

- Encoding 하는 코드 추가 후 재실행하면 한글이 정상적으로 보여짐



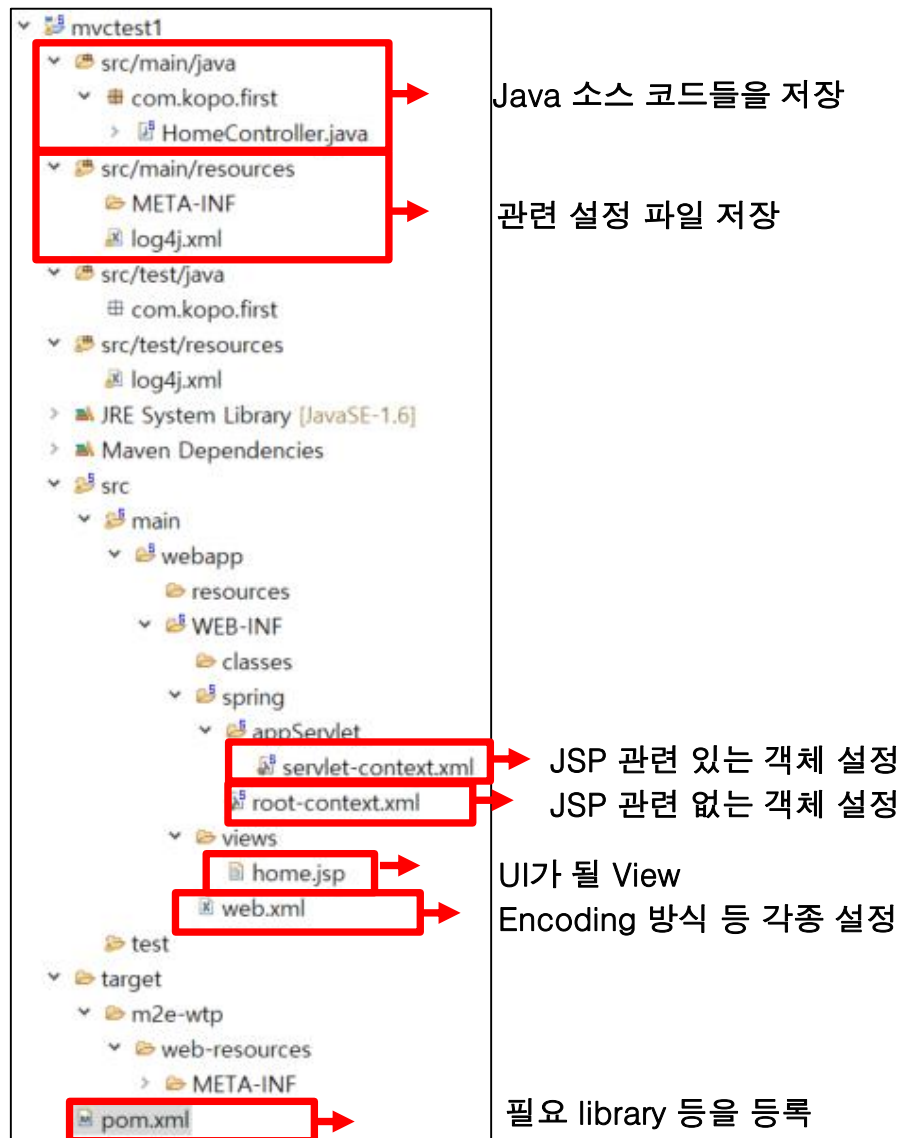
● Spring 프로젝트 생성 및 실습

✓ Spring 프로젝트 구조



● Spring 프로젝트 생성 및 실습

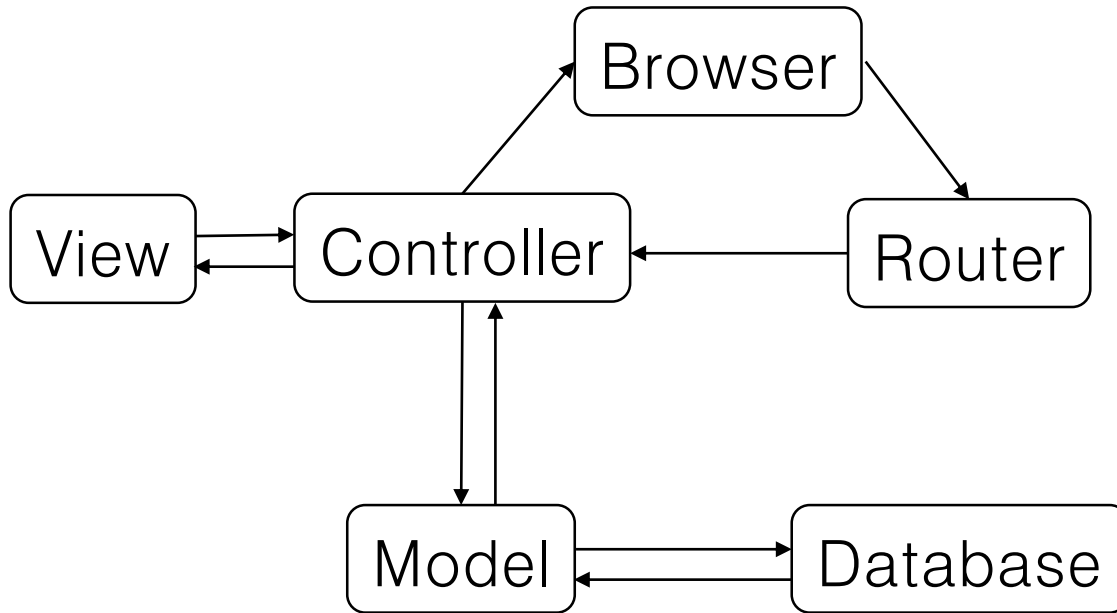
✓ web.xml 설정



● MVC

✓ MVC란[1]?

- Model-View-Controller의 약자로 애플리케이션을 구현하는 패턴 중 한 가지



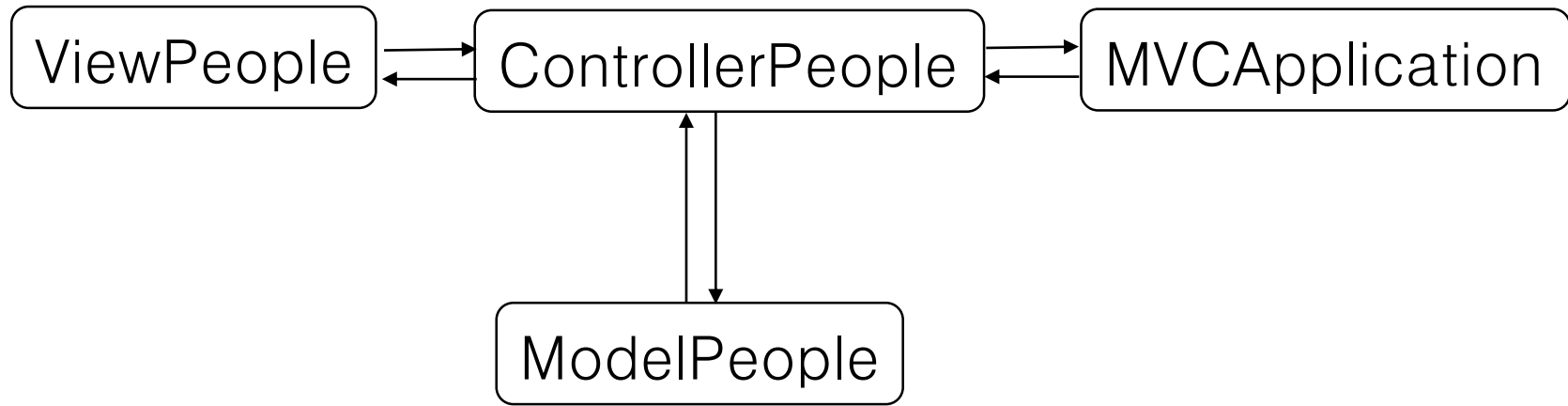
- Model : 어떠한 동작을 수행하는 코드. DAO, DTO로 분류할 수 있음.
- View : Model에게 Query를 하여 사용자에게 보여주는 부분. UI 부분.
- Controller : 사용자의 입력처리와 흐름 제어를 담당. Model과 View를 연결해주는 역할

[1] <https://ko.wikipedia.org/wiki/%EB%AA%A8%EB%8D%B8-%EB%B7%B0-%EC%BB%A8%ED%8A%B8%EB%A1%A4%EB%9F%AC>

● MVC

✓ MVC의 간단 예제

– 아래와 같이 Java Class 4개를 구성하시오.



```
> ControllerPeople.java
> ModelPeople.java
> MVCApplication.java
> ViewPeople.java
```

● MVC

✓ 실습 #1(cont'd)

- Java 프로젝트를 생성하고,
- 아래와 같이 Model class를 구성하시오.

```
public class ModelPeople {  
    private String name;  
    private String hobby;  
  
    public String getName() {  
        return name;  
    }  
  
    public void setName(String name) {  
        this.name = name;  
    }  
  
    public String getHobby() {  
        return hobby;  
    }  
  
    public void setHobby(String hobby) {  
        this.hobby = hobby;  
    }  
}
```

● MVC

✓ 실습 #1(cont'd)

– 아래와 같이 View Class를 구성하시오.

```
public class ViewPeople {  
    public void printPeopleProfile(String name, String hobby) {  
        System.out.println("Name : " + name);  
        System.out.println("Hobby : " + hobby);  
    }  
}
```

● MVC

✓ 실습 #1(cont'd)

– 아래와 같이 Controller Class를 구성하시오

```
public class ControllerPeople {  
    private ModelPeople model;  
    private ViewPeople view;  
  
    public ControllerPeople(ModelPeople model, ViewPeople view) {  
        this.model = model;  
        this.view = view;  
    }  
  
    public void setPeopleName(String name) {  
        model.setName(name);  
    }  
  
    public String getPeopleName() {  
        return model.getName();  
    }  
  
    public void setPeopleHobby(String hobby) {  
        model.setHobby(hobby);  
    }  
  
    public String getPeopleHobby() {  
        return model.getHobby();  
    }  
  
    public void updateView() {  
        view.printPeopleProfile(model.getName(), model.getHobby());  
    }  
}
```

● MVC

✓ 실습 #1(cont'd)

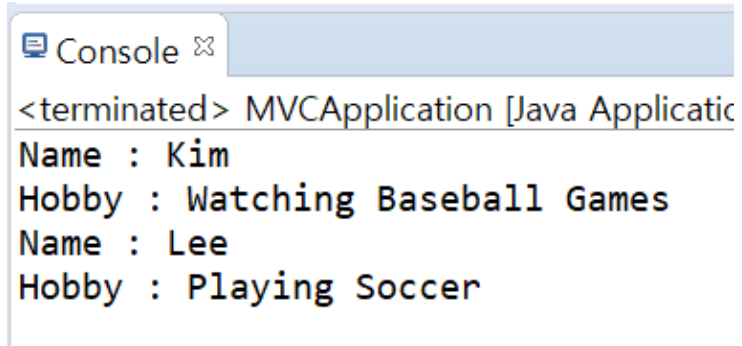
– 아래와 같이 Application Class를 구성하시오.

```
public class MVCApplication {  
  
    public static void main(String[] args) {  
        // TODO Auto-generated method stub  
        ModelPeople model = retrievePeople(); // get related information and set them to model  
        ViewPeople view = new ViewPeople();  
  
        ControllerPeople controller = new ControllerPeople(model, view);  
        controller.updateView(); // show the updated information  
  
        controller.setPeopleName("Lee"); // update the variable  
        controller.setPeopleHobby("Playing Soccer"); // update the variable  
        controller.updateView(); // show the updated information again  
    }  
  
    private static ModelPeople retrievePeople() {  
        ModelPeople model = new ModelPeople();  
        model.setName("Kim");  
        model.setHobby("Watching Baseball Games");  
        return model;  
    }  
}
```

● MVC

✓ 실습 #1

- Console 창에 결과물은 아래와 같아야 함

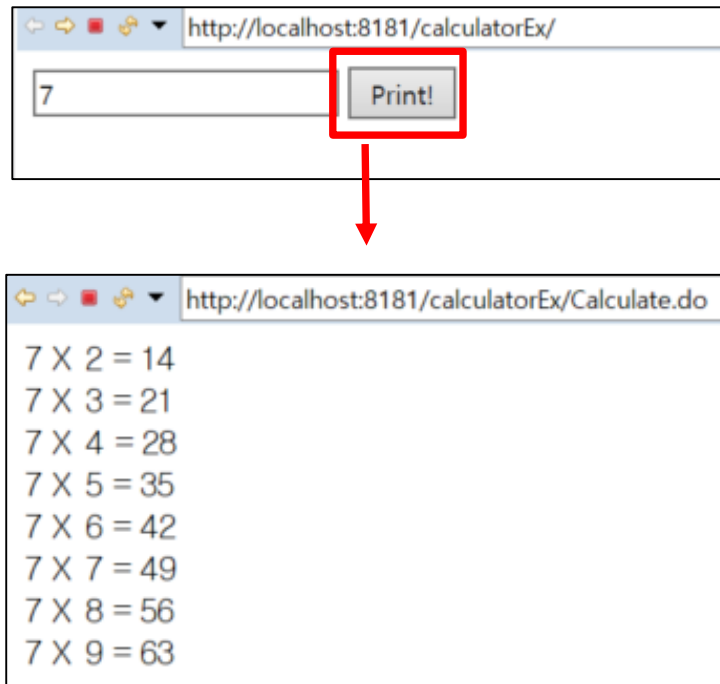


```
Console ✕  
<terminated> MVCApplication [Java Applicati  
Name : Kim  
Hobby : Watching Baseball Games  
Name : Lee  
Hobby : Playing Soccer
```


● 과제

✓ 실습 #2

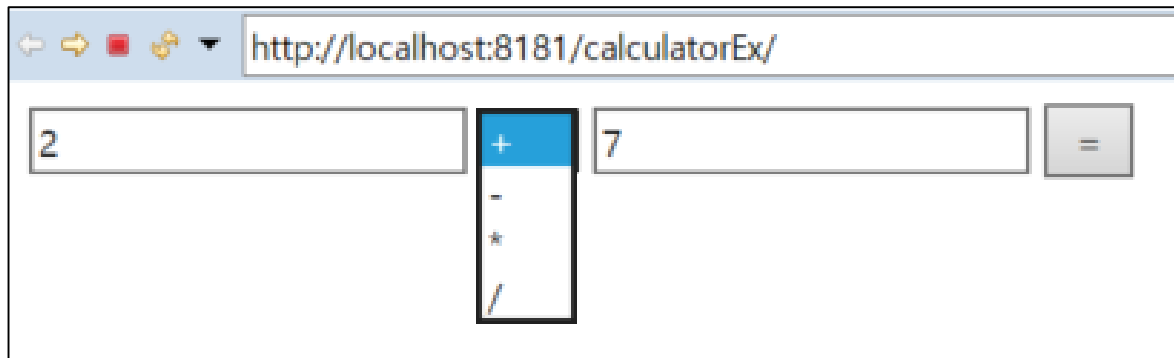
– MVC 패턴을 기반으로 아래와 같이 구구단 출력 프로그램 작성하기



● MVC

✓ 실습 #3

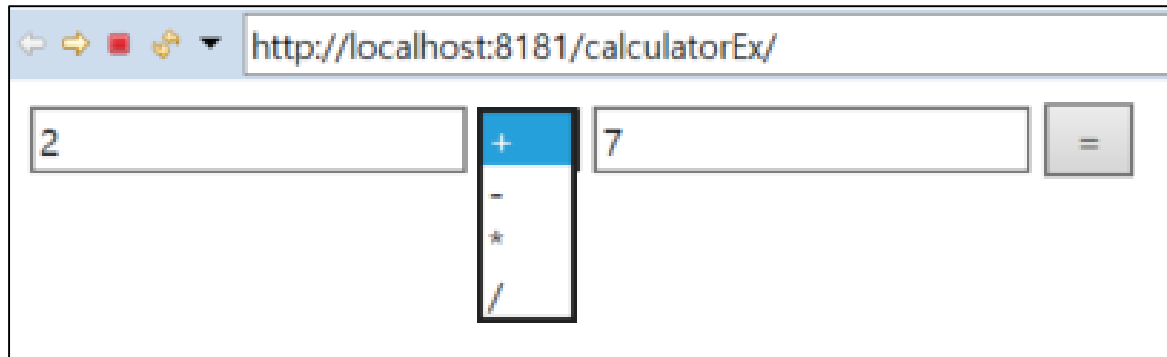
– 아래 그림처럼 사칙연산이 가능하도록 구현하기



● MVC

✓ 실습 #4

- 실습 #3의 코드를 MVC 패턴으로 Refactoring 하기



● MVC

✓ 실습 #5(cont'd)

– Spring 기반의 계산기 구현하기

Very Simple
Half-MVC

The screenshot displays an IDE interface with two main components:

- Project Explorer (Left):** Shows the project structure for 'calculatorEx'. The 'src' directory is expanded, revealing the following hierarchy:
 - src
 - main
 - java
 - com
 - kopo
 - calculatorEx
 - calculatorController.java
 - resources
 - webapp
 - resources
 - WEB-INF
 - classes
 - spring
 - appServlet
 - servlet-context.xml
 - root-context.xml
 - views
 - main.jsp
 - result.jsp
 - test
 - target
 - pom.xml

- Web Browser (Right):** Displays the running application at 'http://localhost:8181/calculatorEx/'. The page shows a simple calculator interface with two input fields containing '10' and '20', a '+' operator, and an '=' button.

● MVC

✓ 실습 #5(cont'd)

- Spring 기반의 계산기 구현하기

Very Simple
Half-MVC

```
package com.kopo.calculatorEx;

import java.text.DateFormat;

/**
 * Handles requests for the application home page.
 */
@Controller
public class calculatorController {
    @RequestMapping(value = "/", method = RequestMethod.GET)
    public String home(Locale locale, Model model) {
        return "main";
    }

    @RequestMapping("Calculate.do")
    public ModelAndView printCalculatedResult(String number1, String operation, String number2){
        ModelAndView modelAndView = new ModelAndView();
        int resultData = 0;

        resultData = Integer.parseInt(number1) + Integer.parseInt(number2);

        modelAndView.setViewName("result");
        modelAndView.addObject("resultData", resultData);

        return modelAndView;
    }
}
```

● MVC

✓ 실습 #5(cont'd)

– Spring 기반의 계산기 구현하기

Very Simple
Half-MVC

```
<%@ page language="java" contentType="text/html; charset=EUC-KR"
    pageEncoding="EUC-KR"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=EUC-KR">
<title>Calculator</title>
</head>
<body>
    <form action="Calculate.do" method="post">
        <input type = "text" name = "number1" width = 500>
        +
        <input type = "text" name = "number2">
        <input type="submit" value="=">
    </form>
</body>
</html>
```

● MVC

✓ 실습 #5(cont'd)

– Spring 기반의 계산기 구현하기

Very Simple
Half-MVC

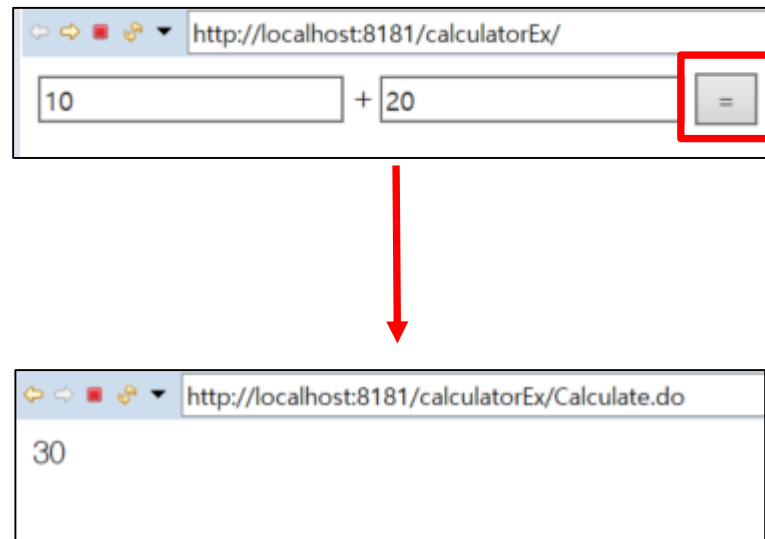
```
<%@ page language="java" contentType="text/html; charset=EUC-KR"
    pageEncoding="EUC-KR"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=EUC-KR">
<title>Calculated Result</title>
</head>
<body>
    ${resultData}
</body>
</html>
```

● MVC

✓ 실습 #5

- Spring 기반의 계산기 구현하기

Very Simple
Half-MVC

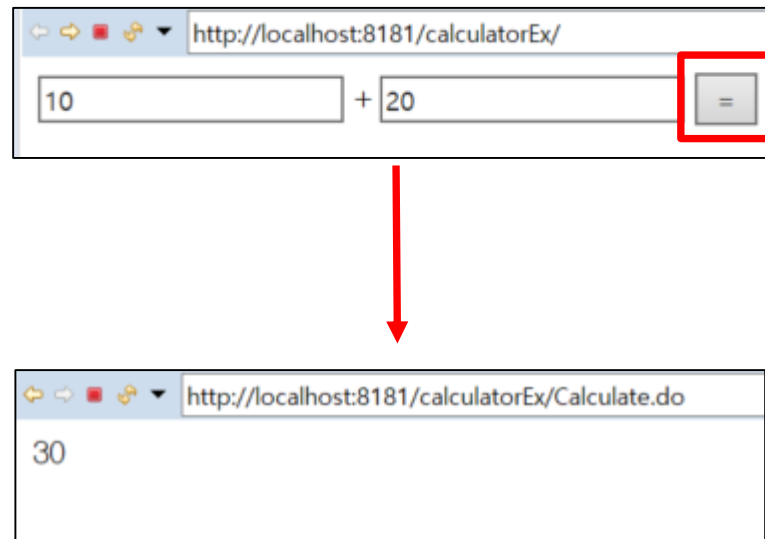


● MVC

✓ 실습 #5

- Spring 기반의 계산기 구현하기

Very Simple
Half-MVC



● 과제 #1

✓ 어제 했던 기술 용어 조사하는 페이지에 아래 내용들도 추가하기

(면접장에서 대답을 한다 생각하고 이해하면서 정리하기)

1. DAO(Data Access Object)
2. DTO(Data Transfer Object)
3. VO(Value Object)
4. JPA(Java Persistence API)
5. ORM(Object-Relation Mapping)

● 과제 #2

- ✓ 어제보다 업데이트 된 프로젝트 기획서 v0.5 작성