

5장 스프링 MVC 기능

강사 김영석

A top-down view of a wooden desk. On the desk, there is a silver laptop with a black keyboard, a pair of black-rimmed glasses, a white coffee cup with a yellow handle, and a small green succulent in a dark pot. The word 'CONTENT' is written in large, white, sans-serif capital letters over the left side of the image.

CONTENT

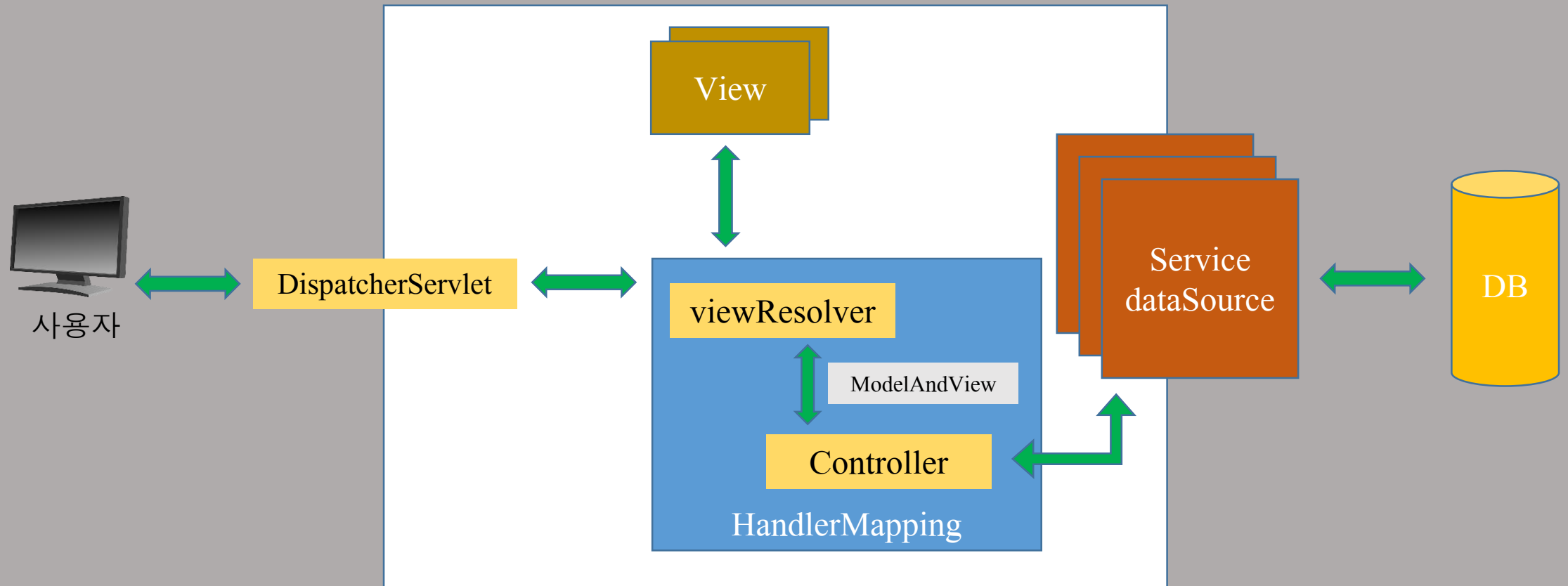
- 1 스프링 프레임워크 MVC 특징
- 2 SimpleUrlController 이용해 스프링 MVC 실습하기
- 3 MultiActionController 이용해 스프링 MVC 실습하기
- 4 요청명과 동일한 JSP 표시하기

1. 스프링 프레임워크 MVC 특징

✓ 스프링에서 지원하는 MVC 기능의 특징

- ① 모델2 아키텍처를 지원한다.
- ② 스프링과 다른 모듈과의 연계가 용이하다.
- ③ 타일즈(tiles)나 사이트 메시(sitemesh) 같은 View 기술과의 연계가 용이하다.
- ④ 태그 라이브러리를 통해 message 출력, theme 적용 그리고 입력 폼을 보다 쉽게 구현할 수 있다.

✓ 스프링 프레임워크의 MVC 구조도



✓ 스프링 프레임워크 MVC 구성 요소

구성 요소	설명
DispatcherServlet	클라이언트의 요청을 전달받아 해당 요청에 대한 컨트롤러를 선택하여 클라이언트의 요청을 전달한다. 또한 컨트롤러가 반환한 값을 View 에 전달하여 알맞은 응답을 생성한다.
HandlerMapping	클라이언트가 요청한 URL을 처리할 컨트롤러를 지정한다.
Controller	클라이언트의 요청을 처리한 후 그 결과를 DispatcherServlet에 전달한다.
ModelAndView	컨트롤러가 처리한 결과 및 뷰 선택에 필요한 정보를 저장한다.
ViewResolver	컨트롤러의 처리 결과를 전달할 View를 지정한다.
View	컨트롤러의 처리 결과 화면을 생성한다.

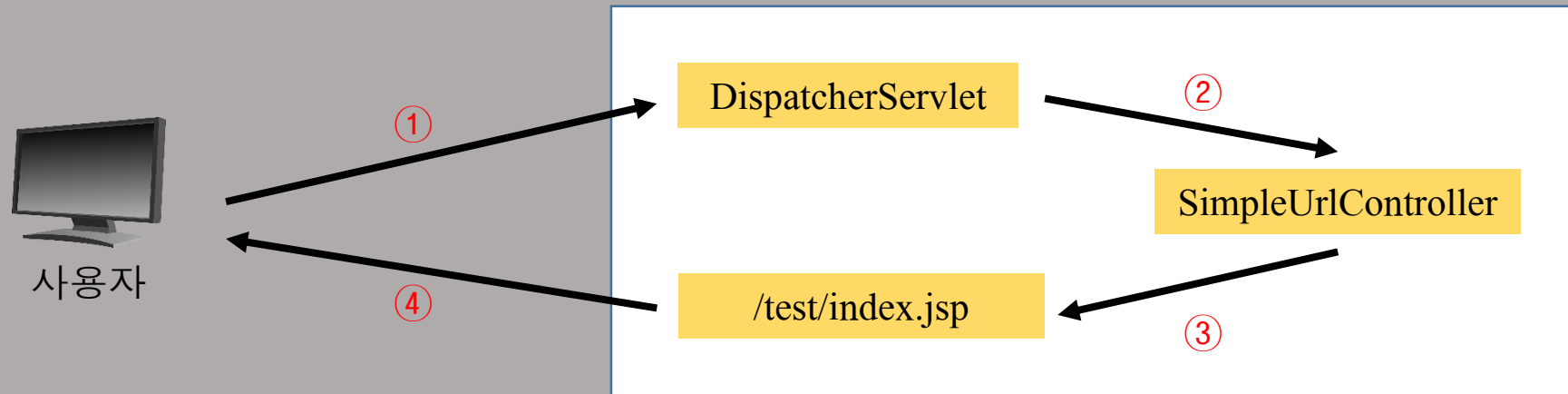


✓ 스프링 프레임워크 MVC 기능 수행 과정

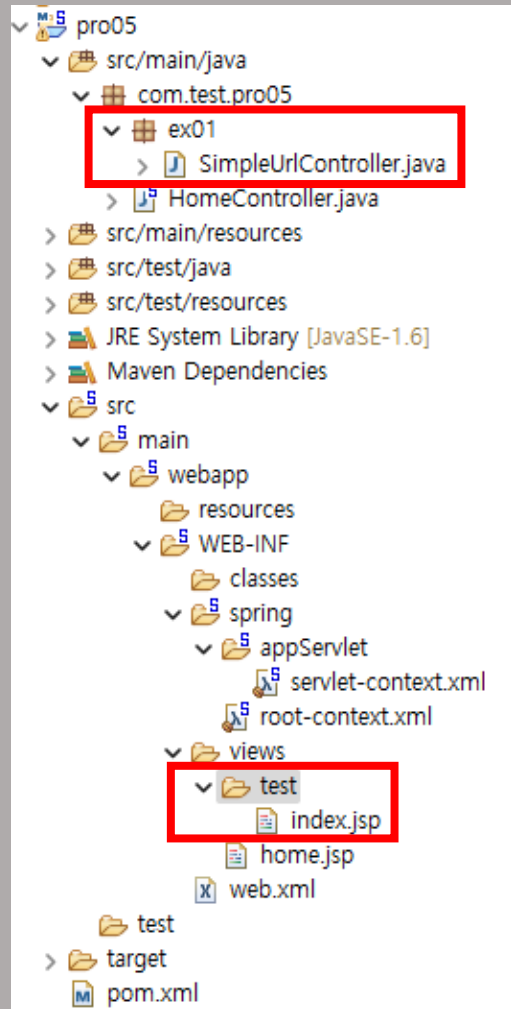
- ① 사용자가 DispatcherServlet에 URL로 접근하여 해당 정보를 요청한다.
- ② 핸들러 매핑에서 해당 요청에 대해 매핑된 컨트롤러가 있는지 요청한다.
- ③ 매핑된 컨트롤러에 대해 처리를 요청한다.
- ④ 컨트롤러가 클라이언트의 요청을 처리한 결과와 View 이름을 ModelAndView에 저장하여 DispatcherServlet으로 반환한다.
- ⑤ DispatcherServlet에서는 컨트롤러에서 보내온 View 이름을 ViewResolver로 보내 View를 요청한다.
- ⑥ ViewResolver는 요청한 View를 보낸다.
- ⑦ View의 처리 결과를 DispatcherServlet으로 보낸다.
- ⑧ DispatcherServlet은 최종 결과를 사용자로 전송한다.

2. SimpleUrlController 이용해 스프링 MVC 실습하기

✓ SimpleUrlController 실행 과정



- ① 사용자 브라우저에서 <http://localhost:8080/pro21/test/index.do>로 요청한다.
- ② DispatcherServlet은 요청에 대해 미리 action-servlet.xml에 매핑된 SimpleUrlController를 요청한다.
- ③ 컨트롤러는 요청에 대해 test 폴더에 있는 index.jsp를 브라우저로 전송한다.



파일	설명
web.xml	브라우저에서 *.do로 요청시 스프링의 DispatcherServlet 클래스가 요청을 받을 수 있게 서블릿 매핑을 한다.
servlet-context.xml	스프링 프레임워크에서 필요한 번들을 설정한다.
SimpleUrlController.java	매핑된 요청에 대해 컨트롤러의 기능을 수행한다.
index.jsp	요청에 대해 컨트롤러가 브라우저로 전송하는 JSP 이다.



```
web.xml
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>*.do</url-pattern>
31 </servlet-mapping>
32
33 </web-app>
```



```
servlet-context.xml
16 <resources mapping="/resources/**" location="/resources/" />
17
18 <!-- Resolves views selected for rendering by @Controllers to .jsp resources in the /WEB-INF/views
19 <beans:bean class="org.springframework.web.servlet.view.InternalResourceViewResolver">
20     <beans:property name="prefix" value="/WEB-INF/views/" />
21     <beans:property name="suffix" value=".jsp" />
22 </beans:bean>
23
24 <context:component-scan base-package="com.test.pro05" />
25
26 <beans:bean id="simpleUrlController" class="com.test.pro05.ex01.SimpleUrlController"></beans:bean>
27 <beans:bean id="urlMapping"
28     class="org.springframework.web.servlet.handler.SimpleUrlHandlerMapping">
29     <beans:property name="mappings">
30         <beans:props>
31             <beans:prop key="/test/index.do">simpleUrlController</beans:prop>
32         </beans:props>
33     </beans:property>
34 </beans:bean>
35
36 </beans:beans>
```



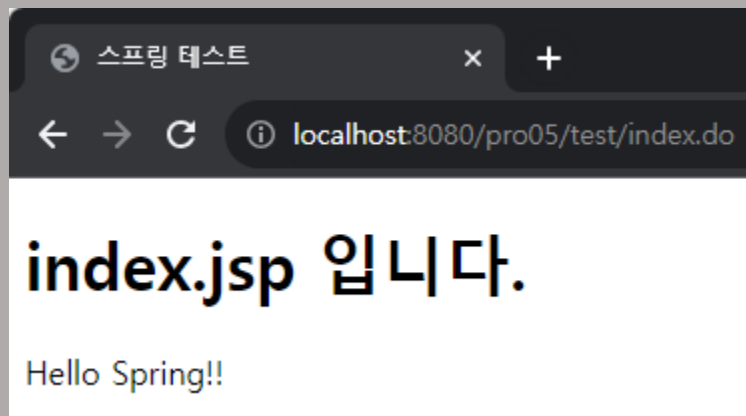
```
package com.test.pro05.ex01;

import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

import org.springframework.web.servlet.ModelAndView;
import org.springframework.web.servlet.mvc.Controller;

public class SimpleUrlController implements Controller{
    public ModelAndView handleRequest(HttpServletRequest request, HttpServletResponse response)
        throws Exception {
        return new ModelAndView("test/index");
    }
}
```

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>스프링 테스트</title>
</head>
<body>
    <h1>index.jsp 입니다.</h1>
    <p>Hello Spring!!</p>
</body>
</html>
```

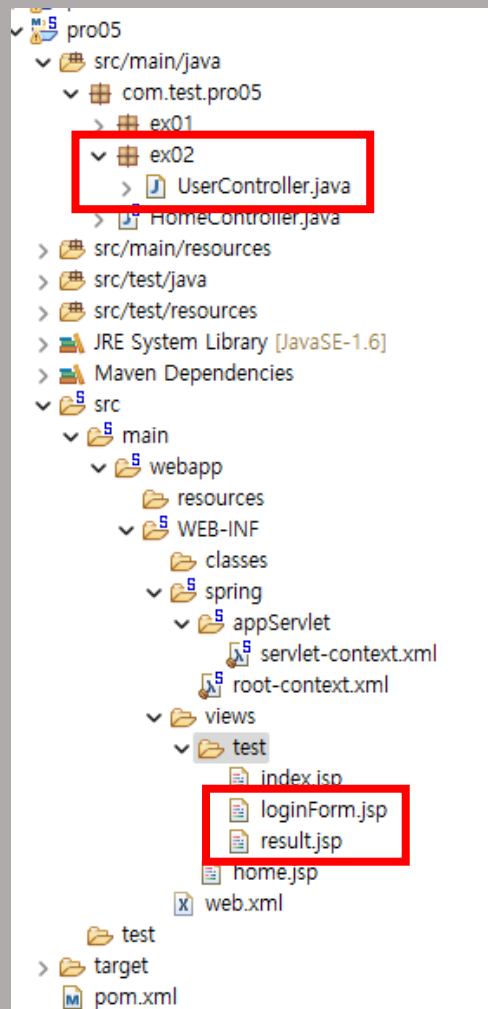


3. MultiActionController 이용해 스프링 MVC 실습하기

- MultiActionController를 이용하면 여러 요청명에 대해 한 개의 컨트롤러에 구현된 각 메서드로 처리할 수 있다.

클래스	설명
MultiActionController	URL 요청명으로 바로 컨트롤러를 지정해서 사용할 수 있다.
PropertyMethodNameResolver	URL 요청명으로 컨트롤러의 설정 파일에서 미리 설정된 메서드를 바로 호출해서 사용할 수 있다.
InternalResourceViewResolver	JSP나 HTML 파일과 같이 웹 애플리케이션의 내부 자원을 이용해 뷰를 생성하는 기능을 제공한다. 기본적으로 사용하는 View 클래스이며 prefix와 suffix 프로퍼티를 이용해 경로를 지정할 수 있다.

파일	설명
UserController.java	매핑된 요청에 대해 컨트롤러의 기능을 수행한다.
loginForm.jsp	로그인창이다.
result.jsp	로그인 결과를 보여주는 JSP 이다.




```
package com.test.pro05.ex02;

import javax.servlet.http.HttpServletRequest;

public class UserController extends MultiActionController{

    public ModelAndView login(HttpServletRequest request, HttpServletResponse response)
        throws Exception {
        String userId = "";
        String passwd = "";
        ModelAndView mav = new ModelAndView();
        request.setCharacterEncoding("utf-8");
        userId = request.getParameter("userId");
        passwd = request.getParameter("passwd");
        mav.addObject("userId", userId);
        mav.addObject("passwd", passwd);
        mav.setViewName("result");
        return mav;
    }

    public ModelAndView loginForm(HttpServletRequest request, HttpServletResponse response)
        throws Exception {
        return new ModelAndView("loginForm");
    }
}
```



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

```
<beans:bean id="userUrlMapping"
  class="org.springframework.web.servlet.handler.SimpleUrlHandlerMapping">
  <beans:property name="mappings">
    <beans:props>
      <beans:prop key="/test/*.do">userController</beans:prop>
    </beans:props>
  </beans:property>
</beans:bean>
<beans:bean id="userController" class="com.test.pro05.ex02.UserController">
  <beans:property name="methodNameResolver">
    <beans:ref bean="userMethodNameResolver"/>
  </beans:property>
</beans:bean>
<beans:bean id="userMethodNameResolver"
  class="org.springframework.web.servlet.mvc.multiaction.PropertiesMethodNameResolver">
  <beans:property name="mappings">
    <beans:props>
      <beans:prop key="/test/login.do">login</beans:prop>
      <beans:prop key="/test/loginForm.do">loginForm</beans:prop>
    </beans:props>
  </beans:property>
</beans:bean>
```

```
loginForm.jsp result.jsp
1 <%@ page language="java" contentType="text/html; charset=UTF-8"
2   pageEncoding="UTF-8"%>
3 <!DOCTYPE html>
4 <html>
5 <head>
6 <meta charset="UTF-8">
7 <title>로그인창</title>
8 </head>
9 <body>
10 <form method="post" action="/pro05/test/login.do">
11 <table border="1" width="80%" align="center">
12 <tr align="center">
13 <td>아이디</td>
14 <td>비밀번호</td>
15 </tr>
16 <tr align="center">
17 <td><input type="text" name="userId" size="20"></td>
18 <td><input type="password" name="passwd" size="20"></td>
19 </tr>
20 <tr>
21 <td colspan="2" align="center">
22 <input type="submit" value="로그인">
23 <input type="reset" value="다시 입력">
24 </td>
25 </tr>
26 </table>
27 </form>
28 </body>
29 </html>
```

```
*result.jsp
1 <%@ page language="java" contentType="text/html; charset=UTF-8"
2     pageEncoding="UTF-8"%>
3 <!DOCTYPE html>
4 <html>
5 <head>
6 <meta charset="UTF-8">
7 <title>로그인 결과창</title>
8 </head>
9 <body>
10     <form method="post" action="/pro05/test/login.do">
11         <table border="1" width="80%" align="center">
12             <tr align="center">
13                 <td>아이디</td>
14                 <td>비밀번호</td>
15             </tr>
16             <tr align="center">
17                 <td>${userId}</td>
18                 <td>${passwd }</td>
19             </tr>
20         </table>
21     </form>
22 </body>
23 </html>
```

로그인창

localhost:8080/pro05/test/loginForm... 시크릿 모드

아이디	비밀번호
hong
<div>로그인 다시 입력</div>	

로그인 결과창

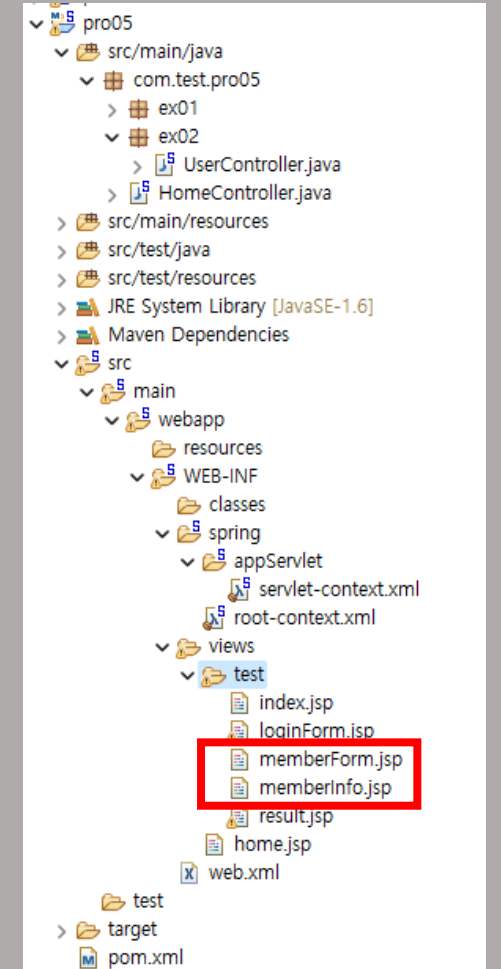
localhost:8080/pro05/test/login.do 시크릿 모드

아이디	비밀번호
hong	1234

Quiz


✓ 아래와 같이 구현해 보세요.

- 회원 정보 입력창에 회원정보를 입력한 후 전송된 회원 정보 보기
- 정보는 id, pwd, name, email 로 지정한다.
- memberForm.do 와 memberInfo.do 로 접근해서 JSP 에 접근한다.
- 메서드명과 요청명은 같은 이름으로 지정한다.



4. 요청명과 동일한 JSP 표시하기

```
private String getViewName(HttpServletRequest request) {  
    String contextPath = request.getContextPath();  
    String uri = (String)request.getAttribute("javax.servlet.include.request_uri");  
    if(uri == null || uri.trim().equals("")) {  
        uri = request.getRequestURI();  
    }  
    int begin = 0;  
    if ((!(contextPath == null) || ("".equals(contextPath)))) {  
        begin = contextPath.length();  
    }  
    int end;  
    if(uri.indexOf(";") != -1) {  
        end = uri.indexOf(";");  
    } else if(uri.indexOf("?") != -1) {  
        end = uri.indexOf("?");  
    } else {  
        end = uri.length();  
    }  
    String fileName = uri.substring(begin, end);  
    if(fileName.lastIndexOf(".") != -1) {  
        fileName = fileName.substring(0, fileName.lastIndexOf("."));  
    }  
    if(fileName.lastIndexOf("/") != -1) {  
        fileName = fileName.substring(fileName.lastIndexOf("/"), fileName.length());  
    }  
    return fileName;  
}
```



```
public ModelAndView memberInfo(HttpServletRequest request, HttpServletResponse response)
    throws Exception {
    ModelAndView mav = new ModelAndView();
    request.setCharacterEncoding("utf-8");
    String id = request.getParameter("id");
    String pwd = request.getParameter("pwd");
    String name = request.getParameter("name");
    String email = request.getParameter("email");
    mav.addObject("id", id);
    mav.addObject("pwd", pwd);
    mav.addObject("name", name);
    mav.addObject("email", email);
    String viewName = getViewName(request);
    mav.setViewName(viewName);
    return mav;
}
```

A photograph of a server room with rows of server racks on both sides of a central aisle. The racks have blue indicator lights. The ceiling has three long, rectangular light fixtures. The text "수고하셨습니다." is overlaid in the center.

수고하셨습니다.