

## 21장 스프링 MVC 기능

---

21.1 스프링 프레임워크 MVC 특징

21.2 SimpleUrlController 이용해 스프링 MVC 실습하기

21.3 MultiActionController 이용해 스프링 MVC 실습하기

21.4 MultiActionController 이용해 회원 정보 표시하기

21.5 요청명과 동일한 JSP 표시하기

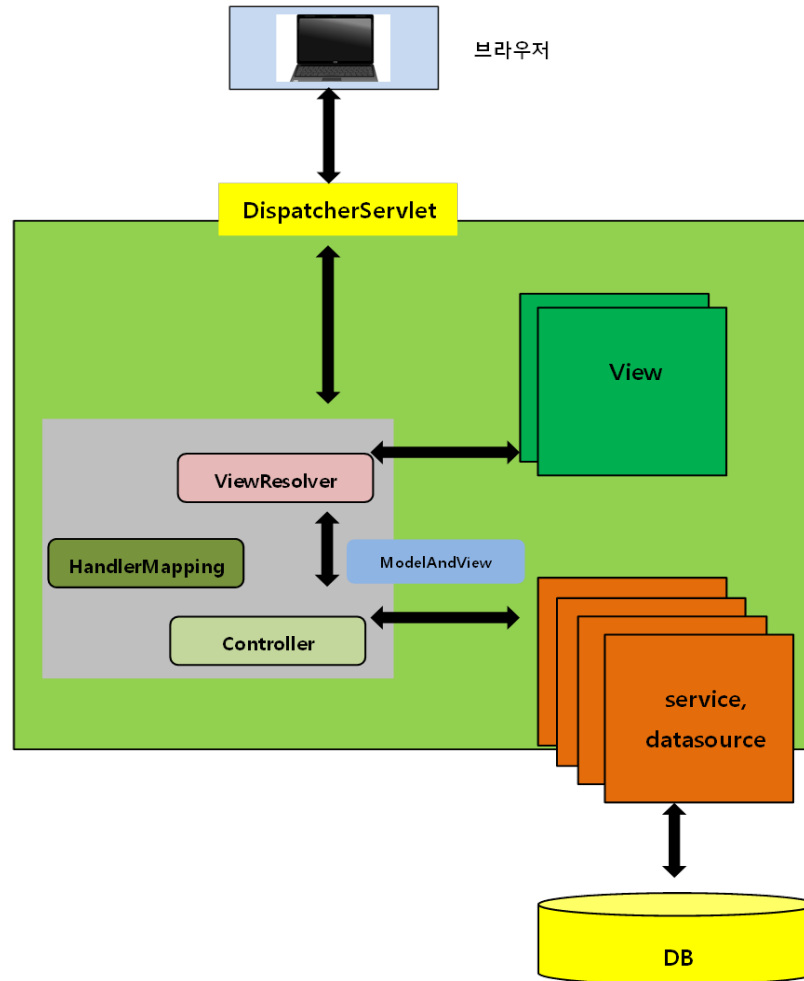
## 21.1 스프링 프레임워크 MVC 의 특징

### 스프링에서 지원하는 MVC 기능의 특징

- 모델2 아키텍처를 지원
- 스프링과 다른 모듈과의 연계가 용이
- 타일즈(tiles)나 사이트메시(sitemesh) 같은 View 기술과의 연계가 용이
- 태그 라이브러리를 통해 message 출력, theme 적용 그리고 입력 폼을 보다 쉽게 구현할 수 있음

## 21.1 스프링 프레임워크 MVC 의 특징

스프링 프레임워크 의 MVC 구조도



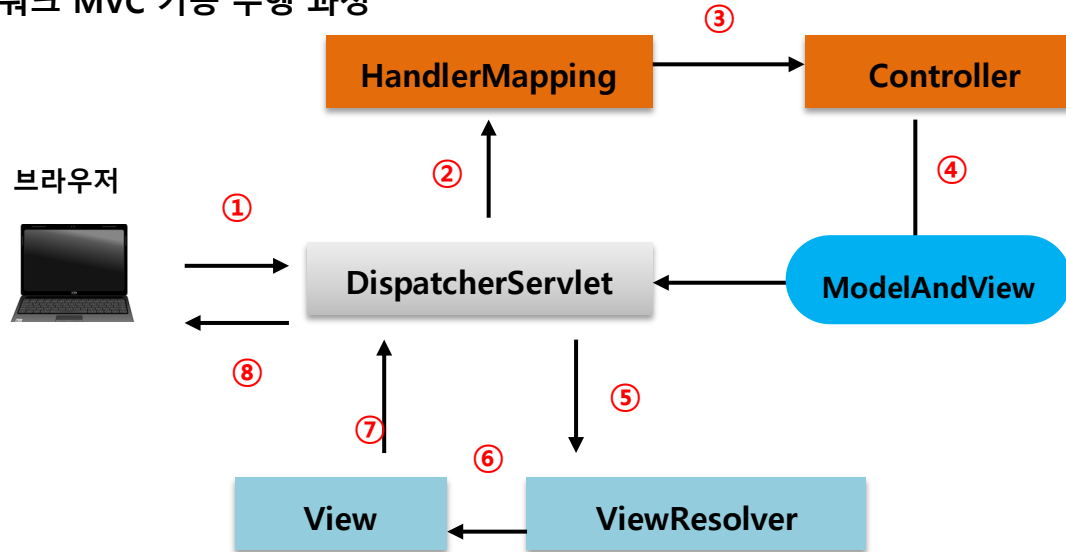
## 21.1 스프링 프레임워크 MVC 의 특징

### 스프링 프레임워크 MVC 구성 요소

구성 요소	설명
DispatcherServlet	클라이언트의 요청을 전달받아 해당 요청에 대한 컨트롤러를 선택하여 클라이언트의 요청을 전달합니다. 또한 컨트롤러가 반환한 값을 View에 전달하여 알맞은 응답을 생성합니다.
HandlerMapping	클라이언트가 요청한 URL을 처리할 컨트롤러를 지정합니다.
Controller	클라이언트의 요청을 처리한 후 그 결과를 DispatcherServlet에 전달합니다.
ModelAndView	컨트롤러가 처리한 결과 및 뷰 선택에 필요한 정보를 저장합니다.
ViewResolver	컨트롤러의 처리 결과를 전달할 뷰를 지정합니다.
View	컨트롤러의 처리 결과 화면을 생성합니다.

## 21.1 스프링 프레임워크 MVC 의 특징

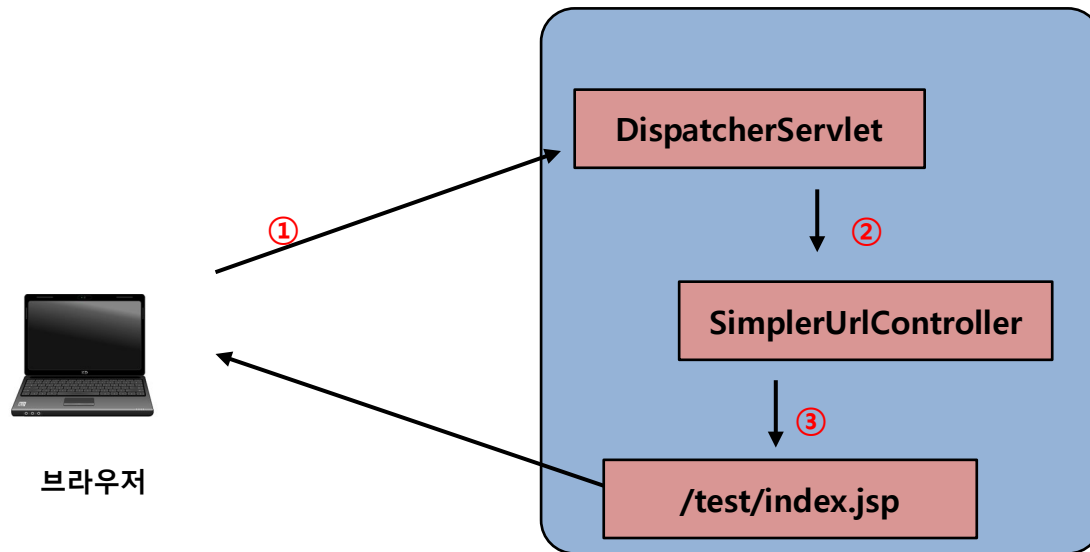
스프링 프레임워크 MVC 기능 수행 과정



- ① 브라우저가 DispatcherServlet에 URL로 접근하여 해당 정보를 요청합니다.
- ② 핸들러 매핑에서 해당 요청에 대해 매핑된 컨트롤러가 있는지 요청합니다.
- ③ 매핑된 컨트롤러에 대해 처리를 요청합니다.
- ④ 컨트롤러가 클라이언트의 요청을 처리한 결과와 View 이름을 ModelAndView에 저장해서 DispatcherServlet으로 반환합니다.
- ⑤ DispatcherServlet에서는 컨트롤러에서 보내온 View 이름을 ViewResolver로 보내 해당 View를 요청합니다.
- ⑥ ViewResolver는 요청한 View를 보냅니다.
- ⑦ View의 처리 결과를 DispatcherServlet으로 보냅니다.
- ⑧ DispatcherServlet은 최종 결과를 브라우저로 전송합니다.

## 21.2 SimpleUrlController 이용해 스프링 MVC 실습

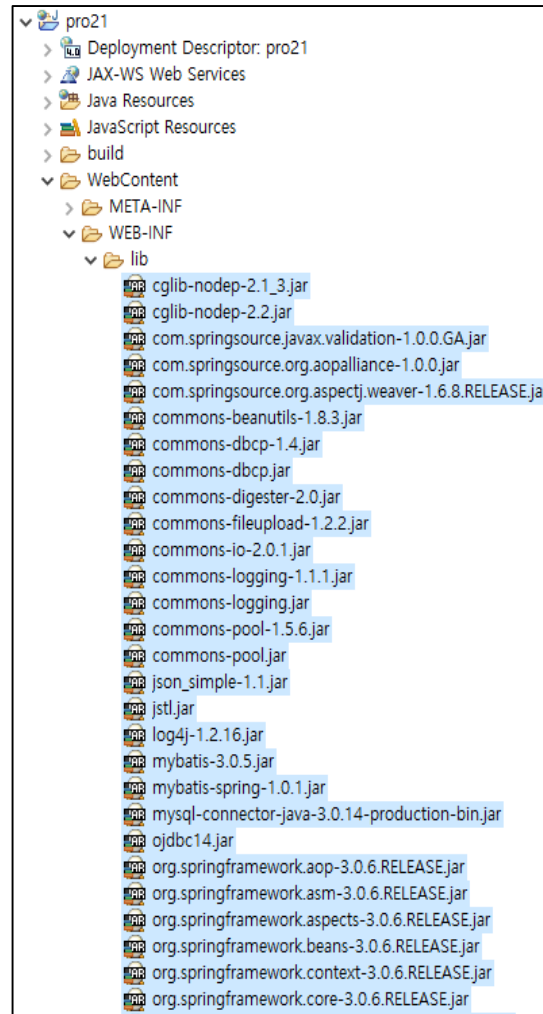
### SimpleUrlController 실행 과정



- ① 브라우저에서 `http://localhost:8090/pro21/test/index.do`로 요청합니다.
- ② DispatcherServlet은 요청에 대해 미리 `action-servlet.xml`에 매핑된 SimpleUrlController를 요청합니다.
- ③ 컨트롤러는 요청에 대해 `test` 폴더에 있는 `index.jsp`를 브라우저로 전송합니다.

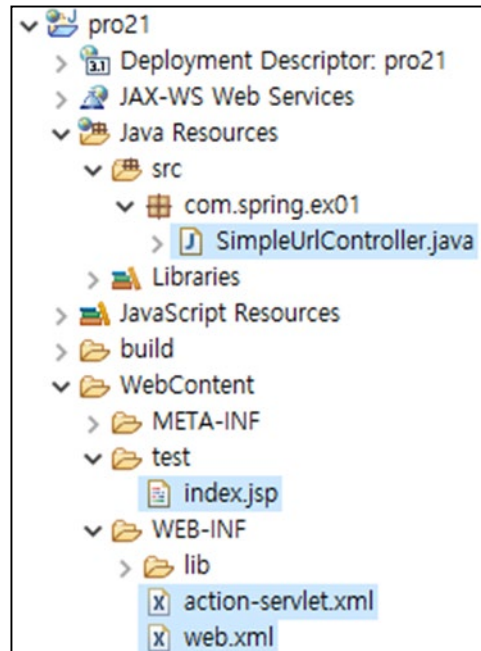
## 21.2 SimpleUrlController 이용해 스프링 MVC 실습

1. 새 프로젝트 pro21을 만들고 이 책과 함께 제공하는 스프링 3.0 라이브러리 파일들을 복사해 /WebContent/WEB-INF/lib 폴더에 붙여 넣습니다.



## 21.2 SimpleUrlController 이용해 스프링 MVC 실습

2. com.spring.ex01 패키지과 test 폴더를 만들고 요청 처리에 사용할 파일인 web.xml, actionservlet.xml, SimpleUrlController.java, index.jsp를 각각 생성합니다.





## 21.2 SimpleUrlController 이용해 스프링 MVC 실습

### 실습에 필요한 여러 가지 파일에 관한 설명

파일	설명
web.xml	브라우저에서 *.do로 요청 시 스프링의 DispatcherServlet 클래스가 요청을 받을 수 있게 서블릿 매핑을 합니다.
action-servlet.xml	스프링 프레임워크에서 필요한 빈들을 설정합니다.
SimpleUrlController.java	매핑된 요청에 대해 컨트롤러의 기능을 수행합니다.
index.jsp	요청에 대해 컨트롤러가 브라우저로 전송하는 JSP입니다.

## 21.2 SimpleUrlController 이용해 스프링 MVC 실습

3. web.xml을 다음과 같이 작성합니다.

**코드 21-1** pro21/WebContent/WEB-INF/web.xml

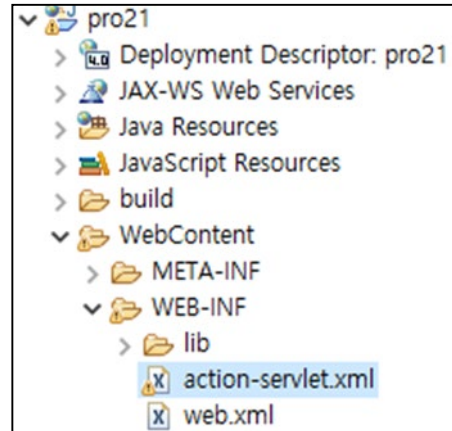
```
<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns="http://xmlns.
jcp.org/xml/ns/javaee" xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/javaee
http://xmlns.jcp.org/xml/ns/javaee/web-app_3_1.xsd" id="WebApp_ID" version="3.1">
  <servlet>
    <servlet-name>action</servlet-name>
    <servlet-class>org.springframework.web.servlet.DispatcherServlet</servlet-class>
    <load-on-startup>1</load-on-startup>
  </servlet>

  <servlet-mapping>
    <servlet-name>action</servlet-name>
    <url-pattern>*.do</url-pattern>
  </servlet-mapping>
</web-app>
```

값이 1 이상이면 톰캣 실행 시 DispatcherServlet을  
미리 메모리에 로드합니다(지정하지 않거나 음수로  
지정하면 브라우저에서 요청 시 메모리에 로드합니다).

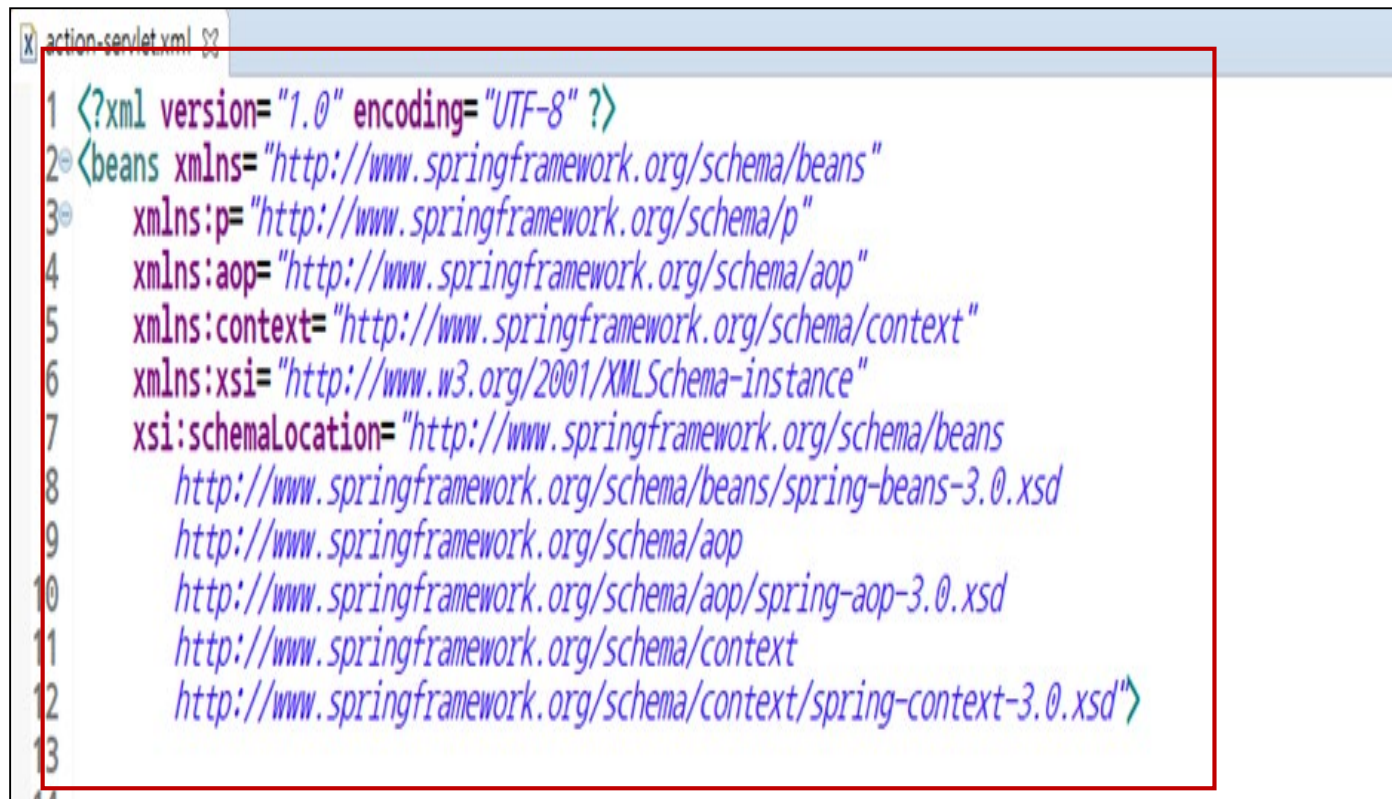
## 21.2 SimpleUrlController 이용해 스프링 MVC 실습

4. 프로젝트의 WEB-INF 아래 action-servlet.xml을 생성합니다.



## 21.2 SimpleUrlController 이용해 스프링 MVC 실습

5. action-servlet.xml을 열고 이 책에서 제공하는 action-servlet.xml에서 <beans> 태그 부분을 복사해 붙여 넣습니다.



```
1 <?xml version="1.0" encoding="UTF-8" ?>
2 <beans xmlns="http://www.springframework.org/schema/beans"
3       xmlns:p="http://www.springframework.org/schema/p"
4       xmlns:aop="http://www.springframework.org/schema/aop"
5       xmlns:context="http://www.springframework.org/schema/context"
6       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
7       xsi:schemaLocation="http://www.springframework.org/schema/beans
8                           http://www.springframework.org/schema/beans/spring-beans-3.0.xsd
9                           http://www.springframework.org/schema/aop
10                          http://www.springframework.org/schema/aop/spring-aop-3.0.xsd
11                          http://www.springframework.org/schema/context
12                          http://www.springframework.org/schema/context/spring-context-3.0.xsd">
13
```

## 21.2 SimpleUrlController 이용해 스프링 MVC 실습

6. action-servlet.xml에 필요한 빈들을 다음과 같이 설정합니다.

**코드 21-2** pro21/WebContent/WEB-INF/action-servlet.xml

```
<?xml version="1.0" encoding="UTF-8" ?>
<beans xmlns="http://www.springframework.org/schema/beans"
  xmlns:p="http://www.springframework.org/schema/p"
  xmlns:aop="http://www.springframework.org/schema/aop"
  xmlns:context="http://www.springframework.org/schema/context"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://www.springframework.org/schema/beans
...

```

id가 simpleUrlController인 빈을 생성합니다.

```
<bean id="simpleUrlController" class="com.spring.ex01.SimpleUrlController"/>
<bean id="urlMapping"
  class="org.springframework.web.servlet.handler.SimpleUrlHandlerMapping">
  <property name="mappings">
    <props>
      <prop key="/test/index.do">simpleUrlController</prop>
    </props>
  </property>
</bean>
</beans>
```

SimpleUrlHandlerMapping 클래스를 이용해 /test/index.do로 요청 시 simpleUrlController를 호출하도록 매핑합니다.

## 21.2 SimpleUrlController 이용해 스프링 MVC 실습

### ❖ 주의

설정 파일 이름은 반드시 web.xml의 서블릿 매핑 시 사용했던 <servlet-name> 태그 값인 action으로 시작해야 합니다(pro21에서는 web.xml에서 태그 값을 action으로 설정했으므로 action1-servlet.xml로 지정하면 톰캣 실행 시 오류가 발생합니다).

7. SimpleUrlController 클래스를 다음과 같이 작성합니다.

**코드 21-3** pro21/src/com/spring/ex01/SimpleUrlController.java

```
package com.spring.ex01;
...
public class SimpleUrlController implements Controller {
    @Override
    public ModelAndView handleRequest(HttpServletRequest request,
    HttpServletResponse response) throws Exception {
        return new ModelAndView("index.jsp");
    }
}
```

스프링에서 제공하는 Controller 인터페이스를 반드시 구현합니다.

작업을 마친 후 뷰이름을 ModelAndView에 index.jsp로 설정하여 반환합니다.

## 21.2 SimpleUrlController 이용해 스프링 MVC 실습

8. 컨트롤러에서 ModelAndView의 인자로 설정된 index.jsp를 화면에 출력하도록 설정합니다.

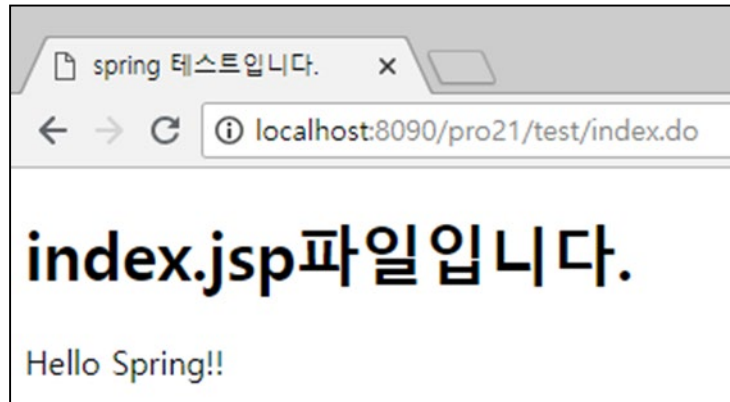
**코드 21-4** pro21/WebContent/WEB-INF/test/index.jsp

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"
    isELIgnored="false" %>
<!DOCTYPE html>
<html>
<head>
    <meta charset="UTF-8">
    <title>spring 테스트입니다.</title>
</head>
<body>
    <h1>index.jsp파일입니다.</h1>
    <p>Hello Spring!!</p>
</body>
</html>
```

---

## 21.2 SimpleUrlController 이용해 스프링 MVC 실습

9. <http://localhost:8080/pro21/test/index.do>로 요청하여 실행 결과를 확인합니다.





## 21.3 MultiActionController 이용해 스프링 MVC 실습

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

### 실습에 사용되는 스프링 클래스들

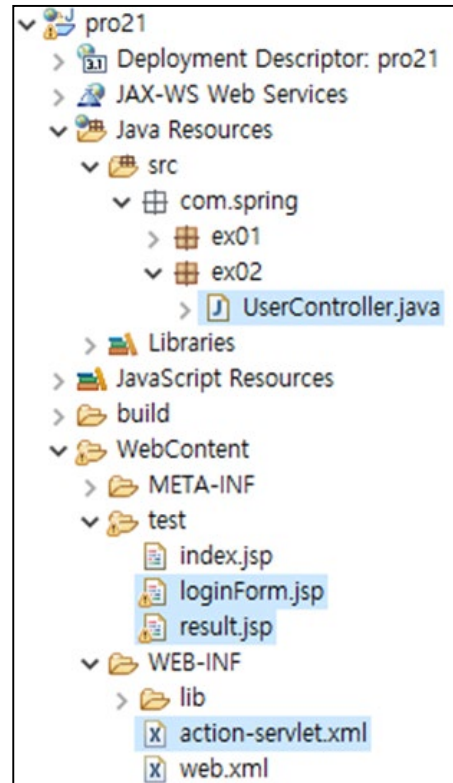
클래스	설명
MultiActionController	URL 요청명으로 바로 컨트롤러를 지정해서 사용할 수 있습니다.
PropertiesMethodNameResolver	URL 요청명으로 컨트롤러의 설정 파일에서 미리 설정된 메서드를 바로 호출해서 사용할 수 있습니다.
InternalResourceViewResolver	JSP나 HTML 파일과 같이 웹 애플리케이션의 내부 자원을 이용해 뷰를 생성하는 기능을 제공합니다. 기본적으로 사용하는 View 클래스이며 prefix와 suffix 프로퍼티를 이용해 경로를 지정할 수 있습니다.

### MultiActionController 실습에 사용되는 파일들

파일	설명
web.xml	브라우저에서 *.do로 요청하면 스프링의 DispatcherServlet 클래스가 요청을 받을 수 있게 서블릿 매핑을 설정합니다.
action-servlet.xml	스프링에서 필요한 빈들을 설정합니다.
UserController.java	매핑된 요청에 대해 컨트롤러의 기능을 수행합니다.
loginForm.jsp	로그인창입니다.
result.jsp	로그인 결과를 보여주는 JSP입니다.

## 21.3 MultiActionController 이용해 스프링 MVC 실습하기

1. com.spring.ex02 패키지를 만들고 UserController 클래스를 추가합니다. 그리고 loginForm.jsp, result.jsp 등 필요한 파일을 준비합니다.



## 21.3 MultiActionController 이용해 스프링 MVC 실습

2. action-servlet.xml을 다음과 같이 수정합니다.

**코드 21-5** pro21/WebContent/WEB-INF/action-servlet.xml

```
<?xml version="1.0" encoding="UTF-8" ?>
<beans xmlns="http://www.springframework.org/schema/beans"
```

...

```
<bean id="viewResolver"
      class="org.springframework.web.servlet.view.InternalResourceViewResolver">
  <property name="viewClass"
            value="org.springframework.web.servlet.view.JstlView"/>
  <property name="prefix" value="/test/" />
  <property name="suffix" value=".jsp" />
</bean>
```

```
<bean id="userUrlMapping"
```

컨트롤러에서 ModelAndView 인자로 뷰이름이 반환되면  
InternalResourceViewResolver의 프로퍼티 prefix 속성에 지정된  
/test 폴더에서 ModelAndView 인자로 넘어온 뷰이름에 해당되  
는 JSP를 선택해 DispatcherServlet으로 보냅니다.

## 21.3 MultiActionController 이용해 스프링 MVC 실습

```

class="org.springframework.web.servlet.handler.SimpleUrlHandlerMapping">
<property name="mappings">
  <props>
    <prop key="/test/*.do">userController</prop>
  </props>
</property>
</bean>

<bean id="userController" class="com.spring.ex02.UserController">
  <property name="methodNameResolver">
    <ref local="userMethodNameResolver"/>
  </property>
</bean>

<bean id="userMethodNameResolver"
      class="org.springframework.web.servlet.mvc.multiaction.
              PropertiesMethodNameResolver">
  <property name="mappings">
    <props>
      <prop key="/test/login.do">login</prop>
    </props>
  </property>
</bean>
</beans>

```

URL 요청명이 /test/\*.do로 시작되면  
userController를 요청합니다.

methodNameResolver 프로퍼티에  
userMethodNameResolver를 주입  
해서 지정한 요청명에 대해 직접 메서  
드를 호출하게 합니다.

PropertiesMethodNameResolver를 이용해  
/test/login.do로 요청하면 userController의  
login 메서드를 호출합니다.

## 21.3 MultiActionController 이용해 스프링 MVC 실습

3. UserController 클래스를 다음과 같이 작성합니다.

**코드 21-6** pro21/src/com/spring/ex02/UserController.java

```
package com.spring.ex02;
...
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;
    }
}
```

설정 파일의 userMethodNameResolver 프로퍼티를 사용하려면 반드시 MultiActionController를 상속받아야 합니다.

ModelAndView에 로그인 정보를 바인딩합니다.

ModelAndView 객체에 포워딩할 JSP 이름을 설정합니다.

## 21.3 MultiActionController 이용해 스프링 MVC 실습

4. loginForm.jsp를 다음과 같이 작성합니다. 로그인창에서 ID와 비밀번호를 입력하고 로그인을 클릭하면 /test/login.do로 DispatcherServlet에 요청합니다.

**코드 21-7** pro21/WebContent/test/loginForm.jsp

```
...
<form name="frmLogin" method="post" action="${contextPath}/test/login.do">
  <table border="0" width="80%" align="center" >
    <tr align="center" >
      <td>아이디</td>
      <td>비밀번호</td>
    </tr>
    <tr align="center" >
      <td><input type="text" name="userID" value="" size="20"></td>
      <td><input type="password" name="passwd" value="" size="20"></td>
    </tr>
  </table>
</form>
</body>
</html>
```

/test/login.do로 DispatcherServlet에 요청합니다.

## 21.3 MultiActionController 이용해 스프링 MVC 실습

5. result.jsp를 다음과 같이 작성합니다.

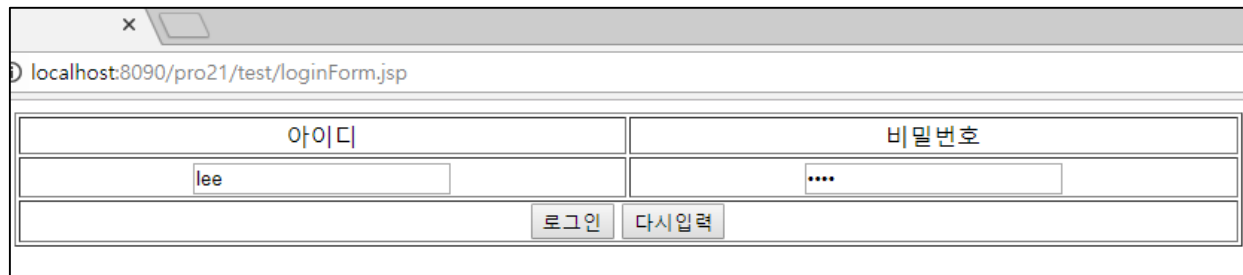
**코드 21-8** pro21/WebContent/test/result.jsp

```
...  
<table border="1" width="50%" align="center" >  
  <tr align="center">  
    <td>아이디</td>  
    <td>비밀번호</td>  
  </tr>  
  <tr align="center">  
    <td>${userID}</td>  
    <td>${passwd}</td>  
  </tr>  
</table>  
...
```

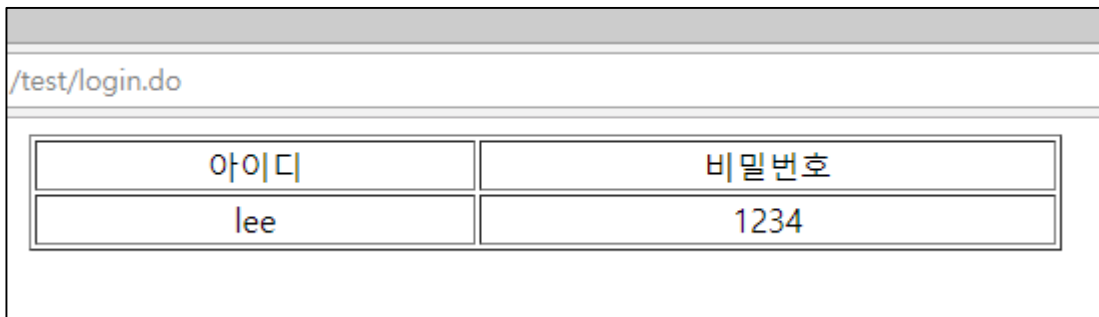
컨트롤러에서 바인딩해 넘어온  
회원 정보를 출력합니다.

## 21.3 MultiActionController 이용해 스프링 MVC 실습

6. <http://localhost:8090/pro21/test/loginForm.jsp>로 요청합니다. 로그인창에서 ID와 비밀번호를 입력하고 로그인을 클릭하면 </test/login.do>로 요청하여 결과를 출력합니다



The screenshot shows a web browser window with the address bar displaying `localhost:8090/pro21/test/loginForm.jsp`. The page contains a login form with two input fields: "아이디" (ID) and "비밀번호" (Password). The ID field contains the text "lee" and the password field contains four dots "....". Below the input fields are two buttons: "로그인" (Login) and "다시입력" (Re-enter).



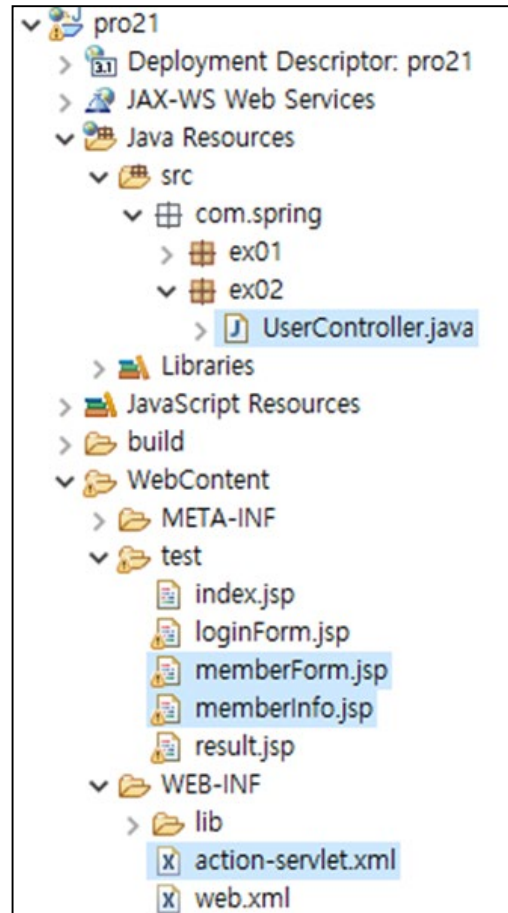
The screenshot shows a web browser window with the address bar displaying `/test/login.do`. The page displays the login result in a table-like structure:

아이디	비밀번호
lee	1234



## 21.4 MultiActionController 이용해 회원 정보 표시

회원 정보 입력창에서 회원 정보를 입력한 후 요청 시 전송된 회원 정보를 표시하기



## 21.4 MultiActionController 이용해 회원 정보 표시

1. 회원 정보를 표시하기 위해 action-servlet.xml을 다음과 같이 수정합니다.

**코드 21-9** pro21/WebContent/WEB-INF/action-servlet.xml

...

```
<bean id="userMethodNameResolver"
      <class="org.springframework.web.servlet.mvc.multiaction.
                                   PropertiesMethodNameResolver">
```

```
<property name="mappings">
```

```
<props>
```

```
<prop key="/test/login.do">login</prop>
```

```
<prop key="/test/memberInfo.do">memberInfo</prop>
```

```
</props>
```

```
</property>
```

```
</bean>
```

...

URL 요청명과 메서드 이름을 동일하게  
하여 회원 정보를 요청합니다.

## 21.4 MultiActionController 이용해 회원 정보 표시

2. UserController 클래스를 다음과 같이 수정합니다.

**코드 21-10** pro21/src/com/spring/ex02/UserController.java

```
...  
public ModelAndView memberInfo(HttpServletRequest request,  
                                HttpServletResponse response)throws Exception {  
    request.setCharacterEncoding("utf-8");  
    ModelAndView mav=new ModelAndView();  
    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);  
    mav.setViewName("memberInfo");  
    return mav;  
}  
...
```

회원 가입창에서 전송된 회원 정보를 addObject() 메서드를 이용해 ModelAndView 객체에 바인딩합니다.

memberInfo.jsp로 포워딩합니다.

## 21.4 MultiActionController 이용해 회원 정보 표시

3. 회원 가입창에서 회원 정보를 입력한 후 /test/memberInfo.do로 요청하도록 action 속성을 설정합니다

**코드 21-11** pro21/WebContent/WEB-INF/test/memberForm.jsp

```
...  
<form method="post" action="${contextPath}/test/memberInfo.do">  
  <h1 class="text_center">회원 가입창</h1>  
  <table align="center">  
    <tr>  
      <td width="200"><p align="right">아이디</p>  
      <td width="400"><input type="text" name="id"></td>  
    </tr>  
  </table>  
...
```

---

## 21.4 MultiActionController 이용해 회원 정보 표시

4. memberInfo.jsp를 다음과 같이 작성합니다. UserController의 memberInfo() 메서드에서 바인딩해 전달된 회원 정보를 출력합니다.

코드 21-12 pro21/WebContent/WEB-INF/test/memberInfo.jsp

```
...  
<table border="1" align="center" width="100%" >  
  <tr align=center bgcolor="lightgreen">  
    <td><b>아이디</b></td>  
    <td><b>비밀번호</b></td>  
    <td><b>이름</b></td>  
    <td><b>이메일</b></td>  
  </tr>  
  <tr align="center">  
    <td>${id}</td>  
    <td>${pwd}</td>  
    <td>${name}</td>  
    <td>${email}</td>  
  </tr>  
</table>  
...
```

---

## 21.4 MultiActionController 이용해 회원 정보 표시

5. <http://localhost:8090/pro21/test/memberForm.jsp>로 요청하여 새 회원 정보를 입력하고 가입하기를 클릭하면 /test/memberInfo.do로 요청한 후 전송된 회원 정보를 출력합니다.

### 회원 가입창

아이디

비밀번호

이름

이메일

회원 정보 출력창			
아이디	비밀번호	이름	이메일
cha	1234	차범근	cha@test.com

## 21.4 MultiActionController 이용해 회원 정보 표시

MultiActionController와 PropertiesMethodNameResolver를 이용한 메서드 이름 설정 방법

요청명	호출 메서드 이름
/test/login.do	login()
/test/memberInfo.do	memberInfo()



- MultiActionController와 PropertiesMethodNameResolver를 이용하면 요청명과 같은 이름으로 메서드를 설정할 수 있어 코드 가독성이 좋아짐

## 21.5 요청명과 동일한 JSP 표시하기

1. 21.3절에서 사용했던 UserController 클래스를 다음과 같이 수정합니다.

코드 21-13 pro21/src/com/spring/ex02/UserController.java

```
...
public class UserController extends MultiActionController {
    public ModelAndView login(HttpServletRequest request,
                               HttpServletResponse response) throws Exception {
        String userID = "";
        String passwd = "";
        String viewName=getViewName(request);
        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");
        mav.setViewName(viewName);
        System.out.println("ViewName:"+viewName);
        return mav;
    }
}
```

getViewName() 메서드를 호출해 요청명에서 확장명 .do를 제외한 뷰이름을 가져옵니다.

뷰이름을 지정합니다.



## 21.5 요청명과 동일한 JSP 표시하기

request 객체에서 URL 요청명을 가져와  
.do를 제외한 요청명을 구합니다.

```
private String getViewName(HttpServletRequest request) throws Exception {
    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();
    }
}
```

## 21.5 요청명과 동일한 JSP 표시하기

```
String fileName=uri.substring(begin,end);
if(fileName.indexOf(".")!=-1){
    fileName=fileName.substring(0,fileName.lastIndexOf("."));
}
if(fileName.lastIndexOf("/")!=-1){
    fileName=fileName.substring(fileName.lastIndexOf("/"),fileName.length());
}
return fileName;
}
}
```

---

## 21.5 요청명과 동일한 JSP 표시하기

2. <http://localhost:8090/pro21/test/loginForm.jsp>로 요청한 후 아이디와 비밀번호를 입력하고로그인을 클릭합니다.  
그러면 /test/login.do로 요청하므로 login.jsp에 결과를 출력합니다.

아이디	비밀번호
<input type="text" value="hong"/>	<input type="password" value="...."/>
<input type="button" value="로그인"/> <input type="button" value="다시입력"/>	

아이디	비밀번호
hong	1234