




제11장 스프링 웹 MVC 서비스 개발 및 응용(1)

- 웹 MVC 개발 및 응용
 - 웹 MVC 구조를 이용한 JDBC 연동
 - 스프링 웹 MVC 서비스 구현
- 

Spring Web MVC 개발을 위한 컨트롤러의 사용법(1)

[/test/t1.jsp] 파일

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>컨트롤러 테스트</title>
</head>
<body>
<h1>${cmd.title}</h1>
</body>
</html>
```

Spring Web MVC 개발을 위한 컨트롤러의 사용법(2)

[TestModel.java] 파일

```
package edu.hallym.kim;

public class TestModel {
    private String title;

    public TestModel(String title) { this.title = title; }
    public String getTitle() { return title; }
    public void setTitle(String title) { this.title = title; }

    @Override
    public String toString() {
        return "TestModel [title=" + title + "];"
    }
}
```

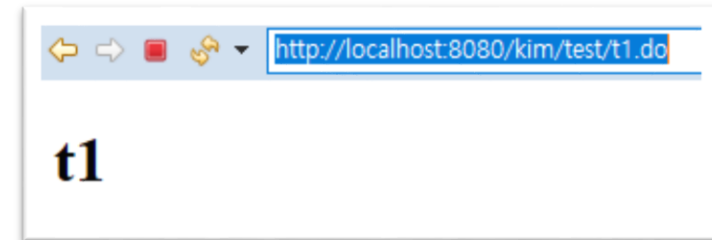
Spring Web MVC 개발을 위한 컨트롤러의 사용법(3)

[TestController.java] 파일

```
package edu.hallym.kim;

import org.springframework.http.HttpHeaders;

@Controller
public class TestController {
    @RequestMapping(value = "/test/t1.do", method = RequestMethod.GET)
    public String myTitle(Model model) {
        model.addAttribute("cmd", new TestModel("t1"));
        return "/test/t1";
    }
}
```



Spring Web MVC 개발을 위한 컨트롤러의 사용법(4)

[TestController.java] 파일에 추가한 내용

```
@RequestMapping(value = "/test/testmodelandview", method = RequestMethod.GET)
public ModelAndView myTitle() {
    ModelAndView ret = new ModelAndView();
    ret.setViewName("test/testmodelandview");
    ret.addObject("cmd", new TestModel("testmodelandview"));

    return ret;
}
```

```
testmodelandview.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>ModelAndView Test</title>
8 </head>
9 <body>
10 <h1>ModelAndView의 결과 : ${cmd.title}</h1>
11 </body>
12 </html>
```

http://localhost:8080/kim/test/testmodelandview

ModelAndView의 결과 : testmodelandview

Spring Web MVC 개발을 위한 컨트롤러의 사용법(5)

[TestController.java] 파일에 추가한 내용

```
@RequestMapping(value = "/test/myvoid", method = RequestMethod.GET)
public void mine(Model model) {
    model.addAttribute("cmd", new TestModel("myvoid"));
}
```

The screenshot shows a code editor on the left and a web browser on the right. The code editor displays the content of *myvoid.jsp, which is an HTML page. The browser shows the URL http://localhost:8080/kim/test/myvoid and the rendered output, which is the word 'myvoid' in a large, bold, black font.

```
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>컨트롤러 테스트</title>
8 </head>
9 <body>
10 <pre>
11 리턴형이 void인 경우..
12 RequestMapping의 value의 이름으로
13 파일명 대신 지정 : </pre>
14 <h1> ${cmd.title} </h1>
15 </body>
16 </html>
```

리턴형이 void인 경우..
RequestMapping의 value의 이름으로
파일명 대신 지정 :

myvoid

Spring Web MVC 개발을 위한 컨트롤러의 사용법(6)

[TestController.java] 파일에 추가한 내용

```
@RequestMapping(value = "/test/mymodel", method = RequestMethod.GET)
public TestModel myModel() {
    return new TestModel("mymodel");
}
```

mymodel.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>컨트롤러 테스트</title>
8 </head>
9 <body>
10 <h1>내가 만든 모델을 리턴형으로 지정할 경우 : ${testModel.title}</h1>
11 </body>
12 </html>
```

http://localhost:8080/kim/test/mymodel

내가 만든 모델을 리턴형으로 지정할 경우 : mymodel

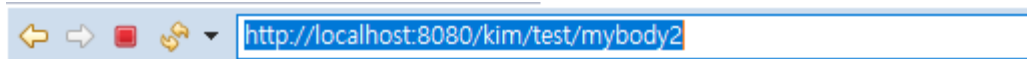
Test의 T를 소문자 t로 교체해야 함

Spring Web MVC 개발을 위한 컨트롤러의 사용법(7)

[TestController.java] 파일에 추가한 내용

```
@RequestMapping(value = "/test/mybody2", method = RequestMethod.GET)
@ResponseBody
public ResponseEntity<String> myBody2() {
    HttpHeaders h = new HttpHeaders();
    h.add("Content-Type", "text/html; charset=UTF-8");

    String html = "<h1>TestTestTest....지금은 테스트중임</h2>";
    return new ResponseEntity<String>(html, h, HttpStatus.OK); // 404에러 처리
}
```



TestTestTest....지금은 테스트중임

Spring Web MVC 응용(1)

[stdinput.jsp] 파일

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

<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>
<%@ taglib uri="http://www.springframework.org/tags/form" prefix="form" %>
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>회원정보등록화면</title>
</head>
<body>
    <h1>회원 정보 등록 화면</h1>
    <form:form method="POST" action="/kim/stdinput.do" commandName="msg">
    <!-- <form action="stdinput" method="post" > --%>
    <table>
        <tr><td>아이디 </td><td><input type="text" value="" name="id" id="id"/> </td></tr>
        <tr><td>이름</td><td><form:input path="name"/> </td></tr>
        <tr><td>나이</td><td><form:input path="age"/> </td></tr>
        <tr><td></td><td><input type="submit" value="전송" /> </td></tr>
    </table>
    </form:form>    <!-- </form> --%>

</body>
</html>
```

Spring Web MVC 응용(2)

[result.jsp] 파일

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

<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>등록된 내용</title>
</head>
<body>
    <h1>등록된 내용</h1>

    <table>
        <tr><td>아이디 : </td><td>${msg.id}</td></tr>
        <tr><td>이름 : </td><td>${msg.name} </td></tr>
        <tr><td>나이 : </td><td>${msg.age} </td></tr>
    </table>
</body>
</html>
```

Spring Web MVC 응용(3)

[Student.java] 파일

```
package edu.hallym.kim;

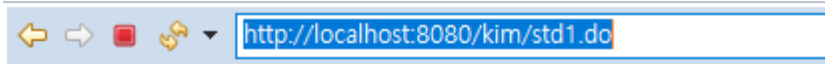
public class Student {
    private String id;
    private String name;
    private int age;

    public String getId() { return id; }
    public void setId(String id) { this.id = id; }
    public String getName() { return name; }
    public void setName(String name) { this.name = name; }
    public int getAge() { return age; }
    public void setAge(int age) { this.age = age; }

    @Override
    public String toString() {
        return "Student [id=" + id + ", name=" + name + ", age=" + age + "]";
    }
}
```

Spring Web MVC 응용(4)

[StdController.java] 파일



회원 정보 등록 화면

아이디
이름
나이

```
package edu.hallym.kim;

import org.springframework.stereotype.Controller;

@Controller
public class StdController {

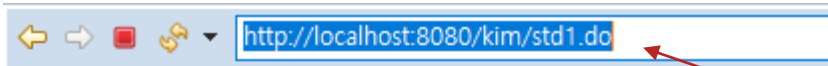
    @RequestMapping(value="/std1.do", method=RequestMethod.GET)
    public String eprocess1(Model model)
    {
        model.addAttribute("msg", new Student());
        return "/stdinput";
    }

}
```

Spring Web MVC 응용(5)

[StdController.java] 파일

- 첫번째 방법론 : 리턴형으로 String타입을 지정하여 view이름을 지정함



회원 정보 등록 화면

아이디

이름

나이

```
package edu.hallym.kim;

import org.springframework.stereotype.Controller;

@Controller
public class StdController {

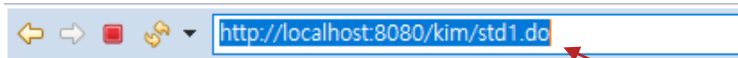
    @RequestMapping(value="/std1.do", method=RequestMethod.GET)
    public String eprocess1(Model model)
    {
        model.addAttribute("msg", new Student());
        return "/stdinput";
    }

}
```

Spring Web MVC 응용(6)

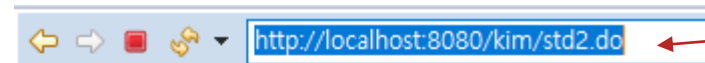
[StdController.java] 파일

- 두번째 방법론 : 리턴형으로 ModelAndView타입을 지정하여 view이름을 지정함



회원 정보 등록 화면

아이디
이름
나이



회원 정보 등록 화면

아이디
이름
나이

```
@Controller
public class StdController {

    @RequestMapping(value="/std1.do", method=RequestMethod.GET)
    public String eprocess1(Model model)
    {
        model.addAttribute("msg", new Student());
        return "/stdinput";
    }

    @RequestMapping(value="/std2.do", method = RequestMethod.GET)
    public ModelAndView eprocess2() {
        ModelAndView k= new ModelAndView("/stdinput","msg",new Student());
        return k;
        //return new ModelAndView("/stdinput","msg",new Student());
    }
}
```

Spring Web MVC 응용(7)

[StdController.java] 파일

- 세번째 방법론 : 입력 [stdinput.jsp] view파일에서 [전송]버튼을 클릭했을 때, 입력한 값을 받아 [result.jsp] view파일에 전달하는 매핑 과정으로 컨트롤러 설정

```
@RequestMapping(value="/stdinput.do", method = RequestMethod.POST)
public String eprocess3(@ModelAttribute Student ret, Model model)
{
    model.addAttribute("msg", ret);
    return "/result";
}
```

http://localhost:8080/kim/std2.do

회원 정보 등록 화면

아이디

이름

나이

http://localhost:8080/kim/std2.do

회원 정보 등록 화면

아이디

이름

나이

http://localhost:8080/kim/stdinput.do

등록된 내용

아이디 : kim
이름 : 김한국
나이 : 20