Spring's Form Tag

Spring의 폼 태그는 다른 폼 태그 라이브러리와는 다르게 Spring Web MVC와 연동 될 수 있다. 이는 태그에서 command 객체, controller 참조 데이터로의 접근이 가능하다는 것이다. Spring Form tag의 사용방법은 매우 간단하며 예제를 중심으로 각 tag에 대한 내용을 살펴본다.

configuration

spring 폼 태그를 사용하기 위해서는 spring-form.tld파일이 필요하고 이는 spring-webmvc-2.5.2.jar 파일에 포함되어 있다. 이 폼 태그를 사용하기 위해서는 JSP 페이지에 taglib을 추가해줘야한다.

```
<%@ taglib prefix="form" uri="http://www.springframework.org/tags/form" %>
```

form tag

'form'태그는 데이터 바인딩을 위해 태그 안에 바인딩 path를 지정해 줄 수 있다. path에 해당되는 값은 도메인 모델의 Bean 객체를 의미한다. 사용예는 다음과 같다.

```
<form:form commandName="user">
userId : <form:input path="userId" />
</form:form>
```

또한 spring form 태그를 이용하기 위해서는 각각의 입력 path값에 매칭될 도메인 객체를 지정해 줘야 하는데 form태그 안에 commandName 속성으로 다음과 같이 지정해 줄 수 있다.

```
<% request.setAttribute("user", sample.services.UserVO())%>
```

이러한 commandName의 기본값은 "command"이며 input값들과 매칭될 도메인 객체를 request값으로 세팅해 줘야한다. 이 값은 SimpleFormController를 사용할 경우 FormBackingObject()메소드에서 지정해 줄 수도있다.

```
protected Object formBackingObject(HttpServletRequest request)
        throws Exception {
        UserVO vo=new UserVO();
         request.setAttribute("user",vo);
        return new UserVO();
}
```

input tag

HTML의 input태그의 value가 text인 것을 기본 value로 갖는다. 이 태그의 예는 위의 form태그 예에서 볼수있다. 또한 다음과 같은 화면을 출력하게 된다.

checkbox tag

다음은 checkbox tag의 예이다. 마찬가지로 path에 도메인 객체의 bean name을 매핑시켜주고 label속성을 이용하면 jsp페이지로 보여질 이름을 설정할 수 있다.

```
<form:checkbox path="hobby" value="listeningMusic" label="음악감상"/>
<form:checkbox path="hobby" value="study" label="공부"/>
```

checkboxes tag

위의 checkbox tag는 각각의 항목에 대해 작성해줘야 하지만 checkboxes tag를 사용하면 items속성을 이용해서 한줄로 나타내줄 수 있다. 이러한 items에 들어갈 값은 컨트롤러의 formBackingObject()메소드에서 Array, List, Map형태의 것들로 넘겨 줄 수 있다. Map의 key와 value쌍으로 넘겨줄 경우 key는 태그의 value값이 되고 value는 label명이 된다. (단, Array나 List로 넘길 경우 label은 value와 같은 값을 가지게 된다.) 다음은 그 예이다.

```
protected Object formBackingObject(HttpServletRequest request)
        throws Exception {
        UserVO vo=new UserVO();
        Map interest =new HashMap();
         interest.put("reading", "독서");
         interest.put("listeningMusic", "음악감상");
         interest.put("study", "공부");
         request.setAttribute("interest",interest);
        request.setAttribute("user",vo);
return new UserVO();
}
```

```
<tr>
        <td>hobby :</td>
        <td><form:checkboxes path="hobby" items="${interest}" /></td>
</tr>
```

radiobutton tag

다음은 radiobutton tag의 예이다. radiobutton tag 또한 label 속성을 이용하여 label명을 설정해 줄 수 있다.

```
<tr>
    <td>Sex:</td>
    <td>Male: <form:radiobutton path="sex" value="M" label="남자"/> <br/>
        Female: <form:radiobutton path="sex" value="F" label="여자"/> </td>
</tr>
```

radiobuttons tag

다음은 radiobuttons tag의 예이다. items 속성의 사용방법은 위의 checkboxes tag와 동일하다.

```
<tr>
    <td>Sex:</td>
    <td><form:radiobuttons path="sex" items="${sexOptions}"/></td>
</tr>
```

password tag

다음은 password tag의 예이다.

```
<tr>
        <td>password :</td>
        <td><form:password path="password" /></td>
</tr>
```

select tag

select tag도 위의 checkboxes tag나 radiobuttons tag 처럼 items 속성을 이용하여 formBackingObject에서 넘겨주는 값으로 자동 매핑 시켜줄 수 있다.

```
protected Object formBackingObject(HttpServletRequest request)
        throws Exception {
        UserVO vo=new UserVO();
        Map address =new HashMap();
         address.put("seoul","서울");
         address.put("daegu","대구");
         address.put("busan","부산");
         request.setAttribute("address",address);
        request.setAttribute("user",vo);
return new UserVO();
}
```

```
<tr>
        <td>주소</td>
```

```
        <td><form:select path="address" items="${address}" /></td>
</tr>
```

일반적인 option tag와 함께 아래와 같이 사용할 수도 있다.

option tag

다음은 option tag의 사용 예이다.

```
<tr>
        <td>주소</td>
        <td><form:select path="address">
                <form:option value="seoul" label="서울" />
                <form:option value="daegu" label="대구" />
                <form:option value="busan" label="부산" />
        </form:select></td>
</tr>
```

options tag

다음은 options 태그의 사용예이다.

```
<tr>
        <td>주소</td>
        <td><form:select path="address">
                <form:options items="${address}" />
        </form:select></td>
</tr>
```

textarea tag

다음은 textarea tag의 사용 예이다.

```
<td>Note :</td>
<td><form:textarea path="comment" rows="3" cols="20"></form:textarea></td>
```

hidden tag

다음은 hidden tag의 사용 예이다.

```
<form:hidden path="userId" />
```

errors tag

Spring은 validatior에서 얻어진 메시지를 JSP페이지에서 쉽게 출력할 수 있도록 spring form 태그의 form:errors태그를 제공한다. 이는 생성한 validator를 통해 입력값의 유효성 체크 후 에러 메시지를 출력해주는데 자세한 사항은 본 매뉴얼 Spring MVC validator - form:errors 태그 사용 방법을 참고한다.

sample

- **tag.jsp(입력화면)**

```
<body>
<form:form commandName="user">
  <table>
     <tr>
        <td>userId :</td>
        <td><form:input path="userId" /></td>
     </tr>
     <tr>
        <td>password :</td>
        <td><form:password path="password" /></td>
     </tr>
     <tr>
        <td>sex :</td>
        <td><form:radiobutton path="sex" value="M" label="남자" />
            <form:radiobutton path="sex" value="F" label="여자" /></td>
     </tr>
     <tr>
        <td>주소</td>
        <td><form:select path="address" items="${address}">
            </form:select></td>
     </tr>
     <tr>
        <td>hobby :</td>
        <td><form:checkboxes path="hobby" items="${interest}" /></td>
     </tr>
     <tr>
        <td>Note :</td>
        <td><form:textarea path="comment" rows="3" cols="20"></form:textarea></td>
     </tr>
   </table>
   <input type="submit" value="submit"/>
</form:form>
</body>
```

- **FormTagTestController.java**

```
public class FormTagTestController extends SimpleFormController {
        private static Log log = LogFactory.getLog(FormTagTestController.class);

        public FormTagTestController() {
                setCommandName("user");
                setCommandClass(UserVO.class);
                setFormView("tag");
                setSuccessView("formtagafter");
        }

        protected ModelAndView onSubmit(HttpServletRequest request,
                        HttpServletResponse response, Object command,
                        BindException exception) throws Exception {

                System.out.println(command.toString());
```

```
                // TODO Auto-generated method stub
                UserVO vo = (UserVO) command;
                return new ModelAndView(getSuccessView(),"vo", vo);
        }
         protected Object formBackingObject(HttpServletRequest request)
         throws Exception {
                UserVO vo=new UserVO();

                Map interest =new HashMap();
                interest.put("reading", "독서");
                interest.put("listeningMusic", "음악감상");
                interest.put("study", "공부");

                Map address =new HashMap();
                address.put("seoul","서울");
                address.put("daegu","대구");
                address.put("busan","부산");

                request.setAttribute("address",address);
                request.setAttribute("interest",interest);

         request.setAttribute("user",vo);

         return new UserVO();
         }
}
```

- **formtagafter.jsp(출력화면)**

```
<body>
userId = ${vo.userId}<br/>
password = ${vo.password}<br/>
address = ${vo.address}<br/>
sex = ${vo.sex}<br/>
hobby= "${vo.hobby}"<br/>
note= "${vo.comment}"<br/>
</body>
```

위의 JSP 코드처럼 Expression Language(JSP 2.0에서 지원)를 사용하면 간단하게 컨트롤러에서 넘겨준 객체의 빈 값을 출력할 수 있다.