Spring MVC表单标签库

业务场景：控制层返回模型数据到视图层，视图层需要使用EL表达式将模型数据绑定到HTML页面表单中。

Student:

```
package com.southwind.entity;

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

    public int getId() {
        return id;
    }

    public void setId(int 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;
    }

    public String getGender() {
        return gender;
    }

    public void setGender(String gender) {
        this.gender = gender;
    }
}
```

Handler:

```
package com.southwind.controller;

import com.southwind.entity.Student;
import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.web.bind.annotation.GetMapping;

@Controller
public class HelloHandler {

    @GetMapping("get")
    public String get(Model model){
        Student student = new Student();
        student.setId(1);
        student.setName("张三");
        student.setAge(22);
        student.setGender("男");
        model.addAttribute("student",student);
        return "show";
    }
}
```

jsp:

```
<%@ page contentType="text/html;charset=UTF-8" language="java" %>
<html>
<head>
    <title>Title</title>
</head>
<body>
    <h1>修改学生信息</h1>
    <form action="" method="post">
        学生编号：<input type="text" name="id"
value="${requestScope.student.id}"/><br/>
        学生姓名：<input type="text" name="name"
value="${requestScope.student.name}"/><br/>
        学生年龄：<input type="text" name="age"
value="${requestScope.student.age}"/><br/>
        学生性别：<input type="text" name="gender"
value="${requestScope.student.gender}"/>
    </form>
</body>
</html>
```

使用Spring MVC表单标签可以直接将模型数据绑定到HTML页面的表单中，非常简单。

- JSP页面导入Spring MVC标签库，与导入JSTL标签库的语法非常相似，前缀prefix可自定义，通常定义为form。

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

- 将form表单与模型数据进行绑定，通过modelAttribute属性完成绑定，将modelAttribute的值设置为控制器向model对象存值时的name即可。

```
<%@ page contentType="text/html;charset=UTF-8" language="java" %>
<%@ taglib prefix="form" uri="http://www.springframework.org/tags/form" %>
<html>
<head>
    <title>Title</title>
</head>
<body>
    <h1>修改学生信息</h1>
    <form:form modelAttribute="student">
        学生编号:<form:input path="id"></form:input><br/>
        学生姓名:<form:input path="name"></form:input><br/>
        学生性别:<form:input path="gender"></form:input><br/>
        学生年龄:<form:input path="age"></form:input><br/>
        <input type="submit" value="提交"/>
    </form:form>
</body>
</html>
```

- from表单与模型数据完成绑定之后，接下来就是将模型数据中的值取出绑定到不同的标签中，通过设置标签的path属性完成，将path属性的值设置为模型数据对应的属性名即可。

Spring MVC表单标签详细使用。

- form标签

```
<form:form modelAttribute="student" method="post">
```

渲染的是HTML中的

，通过modelAttribute属性绑定具体的模型数据。

- input标签

```
<form:input path="name" />
```

渲染的是HTML中的 ，form标签绑定的是模型数据，input标签绑定的就是模型数据中的属性值，通过path与模型数据的属性名对应，并支持及联属性。

Address

```
package com.southwind.entity;

public class Address {
    private int id;
    private String name;

    public int getId() {
        return id;
    }

    public void setId(int id) {
        this.id = id;
    }

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }
}
```

修改Student实体类，添加Address属性。

```
package com.southwind.entity;

public class Student {
    private int id;
    private String name;
    private int age;
    private String gender;
    private Address address;

    public Address getAddress() {
        return address;
    }

    public void setAddress(Address address) {
        this.address = address;
    }

    public int getId() {
        return id;
    }

    public void setId(int 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;
    }

    public String getGender() {
        return gender;
    }

    public void setGender(String gender) {
        this.gender = gender;
    }
}
```

模型数据添加Address属性

```
@GetMapping("get")
public String get(Model model){
    Student student = new Student();
    student.setId(1);
    student.setName("张三");
    student.setAge(22);
    student.setGender("男");
    Address address = new Address();
    address.setId(1);
    address.setName("科技路");
    student.setAddress(address);
    model.addAttribute("student",student);
    return "show2";
}
```

JSP设置及联

```
<%@ page contentType="text/html;charset=UTF-8" language="java" %>
<%@ taglib prefix="form" uri="http://www.springframework.org/tags/form" %>
```

```
<html>
<head>
    <title>Title</title>
</head>
<body>
    <h1>修改学生信息</h1>
    <form:form modelAttribute="student">
        学生编号:<form:input path="id"></form:input><br/>
        学生姓名:<form:input path="name"></form:input><br/>
        学生性别:<form:input path="gender"></form:input><br/>
        学生年龄:<form:input path="age"></form:input><br/>
        学生地址:<form:input path="address.name"></form:input><br/>
        <input type="submit" value="提交"/>
    </form:form>
</body>
</html>
```

- password标签

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

渲染的是HTML中的 <input type="password" />，通过path与模型数据的属性名对应，password标签的值不会在页面中显示。

- checkbox标签

```
<form:checkbox path="hobby" value="读书"/>读书
```

渲染的是HTML中的 <input type="checkbox" />，通过path与模型数据的属性名对，可以绑定boolean、数组和集合。

如果绑定的是boolean类型的数据，该变量值为true，表示选中，false表示未选中。

```
student.setFlag(true);
checkbox:<form:checkbox path="flag"></form:checkbox><br/>
```

如果绑定数组或者集合类型，集合中的元素等于checkbox的value值，则该项选中，否则不选中。

```
List<String> hobby = new ArrayList<>();
hobby.add("读书");
hobby.add("看电影");
hobby.add("打游戏");

学生爱好:<form:checkbox path="hobby" value="读书"></form:checkbox>读书
<form:checkbox path="hobby" value="看电影"></form:checkbox>看电影
<form:checkbox path="hobby" value="旅行"></form:checkbox>旅行
<form:checkbox path="hobby" value="打游戏"></form:checkbox>打游戏
<form:checkbox path="hobby" value="听音乐"></form:checkbox>听音乐
```

- checkboxes

```
<form:checkboxes items="${student.hobby}" path="selectHobby" />
```

渲染的是HTML中的一组☐，这里需要结合items和path两个属性来使用，items绑定被遍历的集合或数组，path绑定被选中的集合或数组，可以这样理解，items为全部选型，path为默认选中的选型。

```
List<String> hobby = new ArrayList<>();
hobby.add("读书");
hobby.add("看电影");
hobby.add("打游戏");
hobby.add("旅行");
hobby.add("听音乐");
List<String> selectHobby = new ArrayList<>();
selectHobby.add("读书");
selectHobby.add("看电影");
selectHobby.add("旅行");
student.setHobby(hobby);
student.setSelectHobby(selectHobby);

学生爱好:<form:checkboxes path="selectHobby"
items="${requestScope.student.hobby}"></form:checkboxes>
```

需要注意的是path可以直接绑定模型数据的属性，items则需要通过EL表达式从域对象中取值，不能直接写属性名。

- radiobutton标签

```
<form:radiobutton path="radioId" value="0"/>
```

渲染的是HTML中的一个◯，绑定的数据与标签的value值相等则为选中状态，否则为未选中状态。

```
student.setRadioId(1);
学生性别:<form:radiobutton path="radioId" value="1"></form:radiobutton>男
<form:radiobutton path="radioId" value="0"></form:radiobutton>女 <br/>
```

- radiobuttons标签

```
<form:radiobuttons items="${student.garde}" path="selectGrader" />
```

渲染的是HTML页面中的一组◯，这里需要结合items和path两个属性来使用，items绑定被遍历的集合或数组，path绑定被选中的值，可以这样理解，items为全部选型，path为默认选中的选型，用法与form:checkboxes/一致。

```
Map<Integer,String> gradeMap = new HashMap<>();
gradeMap.put(1,"一年级");
gradeMap.put(2,"二年级");
gradeMap.put(3,"三年级");
gradeMap.put(4,"四年级");
gradeMap.put(5,"五年级");
gradeMap.put(6,"六年级");
student.setGrade(gradeMap);
student.setSelectGrade(3);

<form:radiobuttons path="selectGrade" items="${requestScope.student.grade}">
</form:radiobuttons><br/>
```

path可以直接绑定模型数据的成员变量，items必须使用EL来取值。