Maven对应Pom主要依赖配置

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-webmvc</artifactId>

<version>4.3.1.RELEASE</version>

</dependency>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-web</artifactId>

<version>4.3.1.RELEASE</version>

</dependency>

#### 基本数据类型

SpringMVC Controller各方法参数绑定首先支持Java所有基本类型（包括： byte、short、int、long、float、double、char、string、boolean），以及基本类型对应封装高级类（包括：StringBuilder、StringBuffer）。

代码示例如下所示（这里仅以int及Integer类型做例子，其他类型实现方式与此类同）：

@Controller

@RequestMapping("param")

public class ParamController {

@RequestMapping(value="/int", method=RequestMethod.POST)

@ResponseBody

public String requestInt(int param) {

return "Request successful. Post param : Int - " + param;

}

@RequestMapping(value="/integer", method=RequestMethod.POST)

@ResponseBody

public String requestInteger(Integer param) {

return "Request successful. Post param : Integer - " + param;

}

//......

}

#### 集合类型

* List

除了基本类型，Spring Controller各方法参数绑定还支持集合类型，这里拿常用的List和Map举例，如果直接拿集合类作为绑定参数，写法上与基本类型类似，不同的是需要在参数处增加注解@RequestParam()，具体代码示例如下所示：

@RequestMapping(value="/list", method=RequestMethod.POST)

@ResponseBody

public String requestList(@RequestParam("listParam[]") List<String> param) {

return "Request successful. Post param : List<String> - " + param.toString();

}

对应Postman的请求如下：

Postman Post请求及应答图示.png

对应Ajax的请求如下：

var strList = new Array();

strList.push("field1");

strList.push("field2");

function postList() {

$.ajax({

type:"POST",

url:"http://localhost:8080/Learn-Spring-MVC/list",

data:{"listParam" : strList},

dataType:"json",

success:function(result) {

//......

},

error:function(result) {

//......

}

});

}

关于List做为Controller参数，除了String本人还测试Integer以及Double类型，其他类型未作测试。这里面@RequestParam()必须增加**"listParam[]"**如果你的Web端是别的名字，换成相应名称即可，如果只用@RequestParam注解的话，请求会接到HTTP Status 400 - Required List parameter 'xxx' is not present的提示信息。

* Map

Map类型作为参数，只需要使用@RequestParam注解即可，这里Map对应的key-value基本是String类型，value可以为Object，代码示例如下所示：

@RequestMapping(value="/map", method=RequestMethod.POST)

@ResponseBody

public String requestList(@RequestParam Map<String, Object> param) {

return "Request successful. Post param : Map - " + param;

}

对应Postman的请求如下：

Postman Post请求及应答图示.png

对应Ajax的请求如下：

function postMap() {

$.ajax({

type:"POST",

url:"http://localhost:8080/Learn-Spring-MVC/map",

data:{field1:"field1",field2:1},

dataType:"json",

success:function(result) {

//......

},

error:function(result) {

//......

}

});

}

Map作为参数的话，会把Ajax请求中的全部数据转换为key-value存入Map对应中，具体可参见[Spring官方文档关于@RequestParam部分](https://link.jianshu.com?t=http:/docs.spring.io/spring/docs/current/spring-framework-reference/htmlsingle/#mvc-ann-requestparam)

#### 自定义类型

自定义People类，具体类内容如下代码所示：

public class People implements Serializable{

private static final long serialVersionUID = 1L;

private String name;

private short age;

private Map<String, String> relationship;

public String getName() {

return name;

}

public void setName(String name) {

this.name = name;

}

public short getAge() {

return age;

}

public void setAge(short age) {

this.age = age;

}

public Map<String, String> getRelationship() {

return relationship;

}

public void setRelationship(Map<String, String> relationship) {

this.relationship = relationship;

}

@Override

public String toString() {

return "People [name=" + name + ", age=" + age + ", relationship=" + relationship + "]";

}

}

对应Controller方法代码如下所示，主要的不同点就是需要在自定义类前添加@ModelAttribute注解：

@RequestMapping(value="/people", method=RequestMethod.POST)

@ResponseBody

public String requestPeople(@ModelAttribute People people) {

return "Get request is successful. Post param : User Class - " + people.toString();

}

对应Ajax的请求如下：

var people = {

name:"Heacewalker",

age:25,

relationship:{"boss":"BigBoss"}

}

function postClick() {

$.ajax({

type:"POST",

url:"http://localhost:8080/Learn-Spring-MVC/people",

data:people,

dataType:"json",

success:function(result) {

//......

},

error:function(result) {

//......

}

});

}

对应Postman的请求如下：

Postman Post请求及应答图示.png

这里再说一下，如果自定义类中包含集合的话，要如何处理参数的问题，首先定义包含集合的类，具体代码如下所示：

public class Address {

private String code;

private String address;

public String getCode() {

return code;

}

public void setCode(String code) {

this.code = code;

}

public String getAddress() {

return address;

}

public void setAddress(String address) {

this.address = address;

}

@Override

public String toString() {

return "Address [code=" + code + ", address=" + address + "]";

}

}

public class Link {

private List<Address> address;

public List<Address> getAddress() {

return address;

}

public void setAddress(List<Address> address) {

this.address = address;

}

@Override

public String toString() {

return "Link [address=" + address + "]";

}

}

最后是Controller部分的代码，示例如下：

@RequestMapping(value="/link", method=RequestMethod.POST)

@ResponseBody

public String requestLink(@ModelAttribute Link link) {

return "Get request is successful. Post param : User Class - " + link.toString();

}

Postman Post请求及应答图示.png

对应Postman的请求如下：

Postman Post请求及应答图示.png

对应Ajax的请求如下：

<form onsubmit="return PostData()">

<input name="address[0].address" value="AKB"/>

<input name="address[0].code" value="48"/>

<input name="address[1].address" value="HKT"/>

<input name="address[1].code" value="48"/>

<input type="submit" value="提交"/>

</form>

......

function PostData() {

$.ajax({

type:"POST",

url:"http://localhost:8080/Learn-Spring-MVC/link",

data:'',

success:function(result) {

//......

},

error:function(result) {

//......

}

});

}

通过URL路径传递参数并接收，具体Controller代码如下所示：

@RequestMapping(value="/path/{key}/{value}", method=RequestMethod.POST)

@ResponseBody

public String requestPath(@PathVariable String key, @PathVariable String value) {

return "Get request is successful. Path param : key - " + key + "; value - " + value;

}

对应Postman的请求如下：

Postman Post请求及应答图示.png

最后，说一下用HttpServletRequest接收请求参数的方法，Controller代码如下所示，Controller接收到的参数值均为String类型：

@RequestMapping(value="/request", method=RequestMethod.POST)

@ResponseBody

public String request(HttpServletRequest arg0, HttpServletResponse arg1) {

return "Get request is successful. Post param : param1 - " + arg0.getParameter("param1") +

"; param2 - " + arg0.getParameter("param2") +

"; param3 - " + arg0.getParameter("param3");

}

对应Ajax请求：

var idList = new Array();

idList.push("1");

idList.push("2");

idList.push("3");

function postTest() {

$.ajax({

type:"POST",

url:"http://localhost:8080/Learn-Spring-MVC/request",

data:{"param1":"test","param2":2,"param3":JSON.stringify(idList)},

dataType:"json",

success:function(result) {

//......

},

error:function(result) {

//......

}

});

}

这里idList是集合类型，如果不是用JSON.stringify()进行转换的话，到Controller层接收到的参数值对应则为null。

作者：何平行者  
链接：https://www.jianshu.com/p/ed44e89a6f79  
來源：简书  
著作权归作者所有。商业转载请联系作者获得授权，非商业转载请注明出处。