

03 SpringMVC-RequestMapping请求参数

SpringMVC

一：请求参数描述

1、注相应的注解（@PathVariable、@RequestParam、@RequestHeader 等）、Spring MVC 框架会将 HTTP 请求的信息绑定到相应的方法入参中，并根据方法的返回值类型做出相应的后续处理

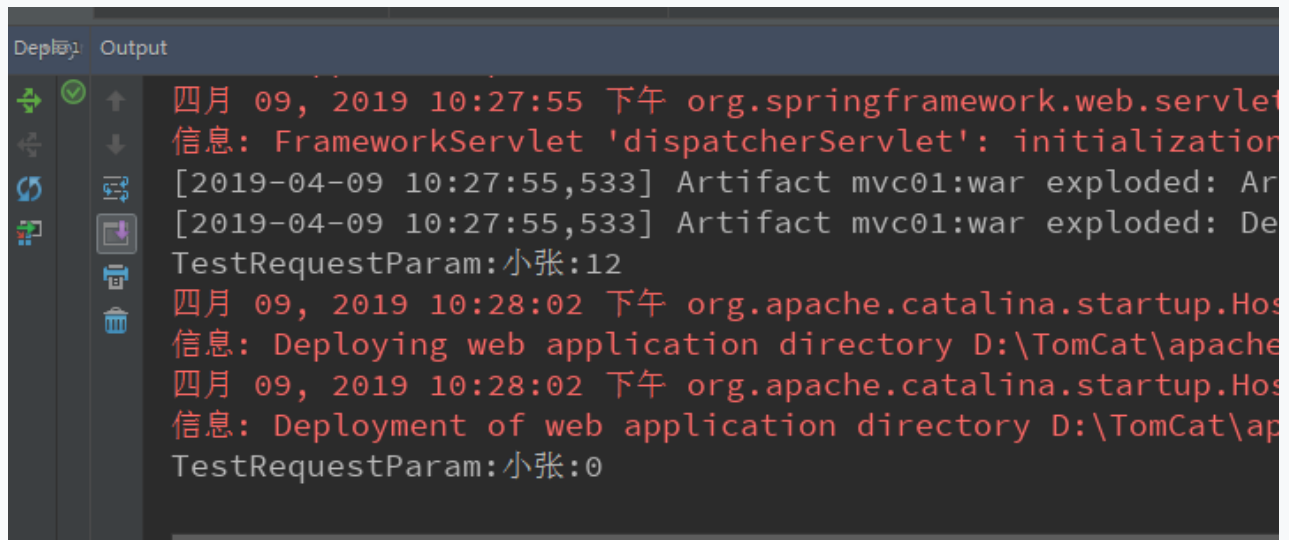
二：@RequestParam绑定请求参数值

1、在处理方法入参处使用 @RequestParam 可以把请求参数传递给请求方法

- value :
参数名
- required :
是否必须。默认为true,表示请求参数中必须包含对应的参数，反之将抛出异常
- defaultValue :
请求参数的默认值

```
1.      @RequestMapping(value = "/TestRequestParam")
2.      public String TestRequestParam(@RequestParam("name") String name
3.          ,@RequestParam(value = "age",required = false,defaultValue
4.      = "0") int age){
5.          System.out.println("TestRequestParam:"+name+": "+age);
6.          return "/success";
7.      }
8.
9.      =====  页面请求=====
10.      <a href="TestRequestParam?name=小张&age=12">TestRequestParam</a>
      <a href="TestRequestParam?name=小张">TestRequestParam</a>
```

2、结果，

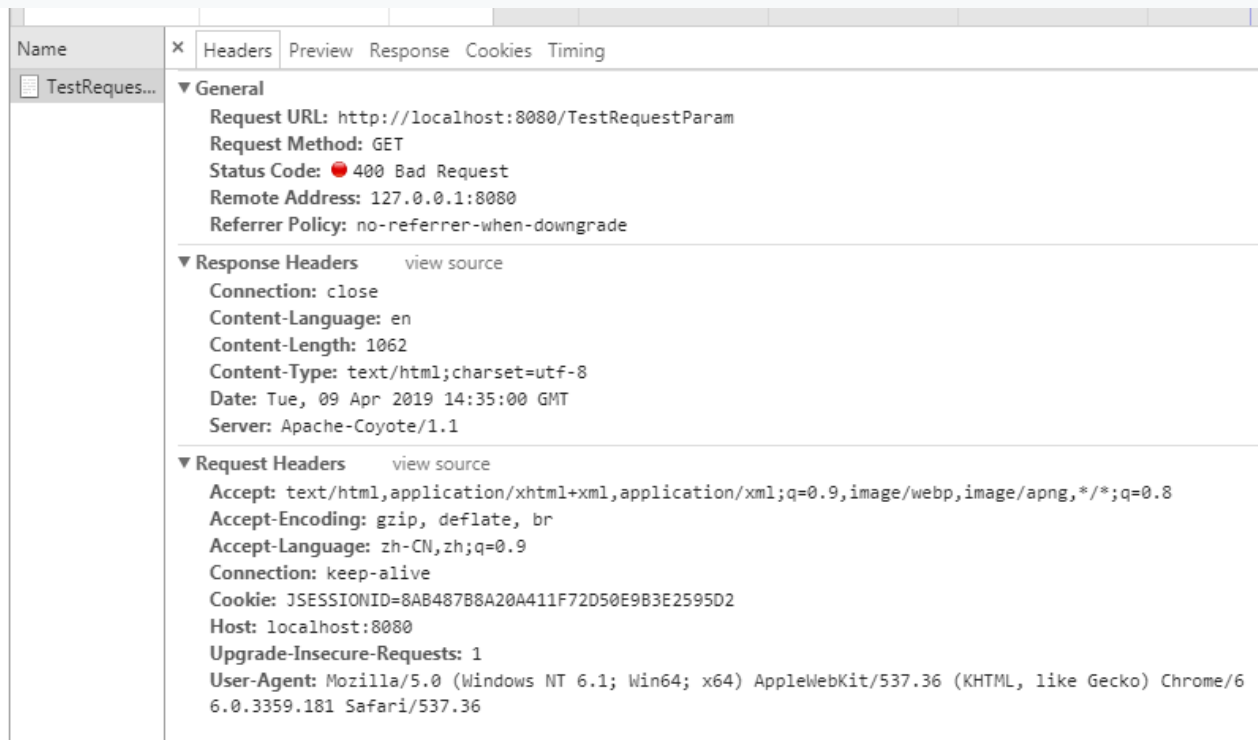


```
四月 09, 2019 10:27:55 下午 org.springframework.web.servlet
信息: FrameworkServlet 'dispatcherServlet': initialization
[2019-04-09 10:27:55,533] Artifact mvc01:war exploded: Ar
[2019-04-09 10:27:55,533] Artifact mvc01:war exploded: De
TestRequestParam:小张:12
四月 09, 2019 10:28:02 下午 org.apache.catalina.startup.Hos
信息: Deploying web application directory D:\TomCat\apache
四月 09, 2019 10:28:02 下午 org.apache.catalina.startup.Hos
信息: Deployment of web application directory D:\TomCat\ap
TestRequestParam:小张:0
```

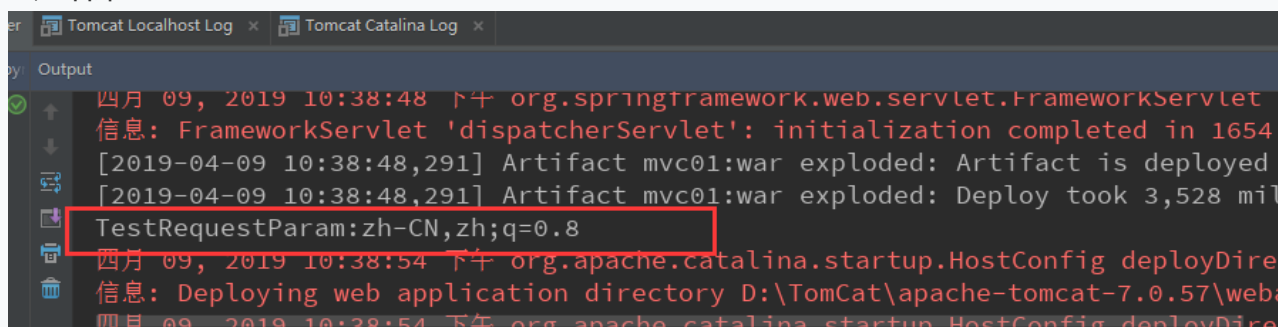
三：@RequestHeader 绑定请求头属性值

- 1、请求头包含了若干个属性，服务器可据此获知客户端的信息
- 2、通过 @RequestHeader 即可将请求头中的属性值绑定到处理方法的入参中

```
1.      @RequestMapping(value = "/TestRequestHeader")
2.      public String TestRequestHeader(
3.          //value的值必须与页面上显示的RequestHeader的key一样
4.          @RequestHeader(value = "Accept-Language"
5.              ,required = false,defaultValue = "没有获取到")
6.              String language){
7.          System.out.println("TestRequestParam:"+language);
8.          return "/success";
9.      }
10.      ===== 页面请求=====
11.      <a href="TestRequestHeader">TestRequestHeader</a>
```



3、结果

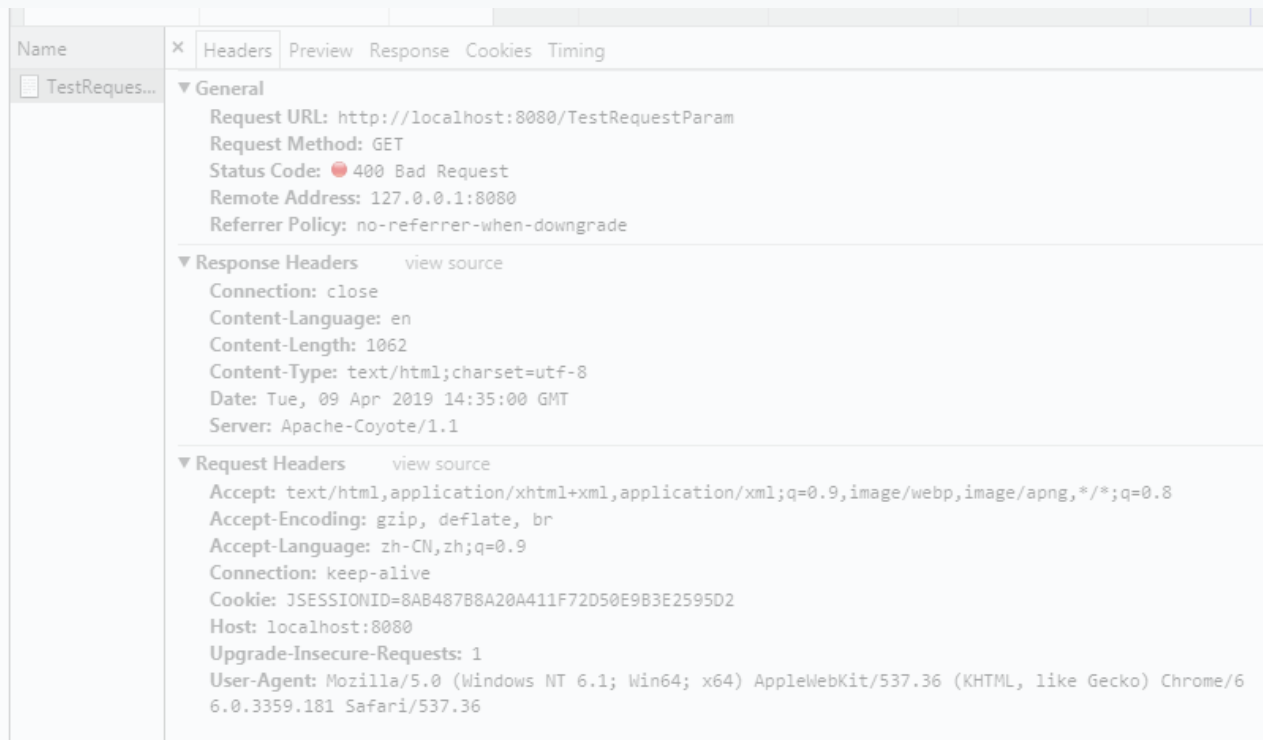


四：@CookieValue 绑定请求中的 Cookie 值

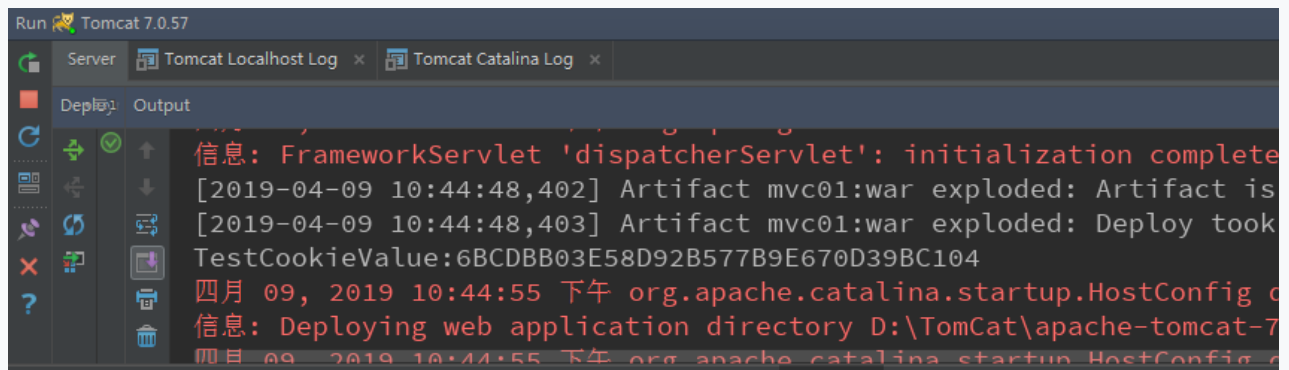
1、可让处理方法入参绑定某个 Cookie 值

```
1. @RequestMapping(value = "/TestCookieValue")
2. public String TestCookieValue(
3.     //value的值必须与页面上显示的RequestHeader的key一样
4.     @CookieValue(value = "JSESSIONID"
5.         ,required = false,defaultValue = "没有获取到")
6.     String jsessionid){
7.     System.out.println("TestCookieValue:"+jsessionid);
8.     return "/success";
9. }
```

```
10.      ===== 页面请求 =====
11.      <a href="TestCookieValue">TestCookieValue</a>
```



3、结果



五：使用POJO对象绑定请求参数值

- 1、Spring MVC 会按请求参数名和 POJO 属性名进行自动匹配，自动为该对象填充属性值。支持级联属性

1、实体对象

1、Person对象

```
1. public class Person {
2.     private String pname;
3.     private Integer page;
4.     //对应的get set toString方法
5. }
```

2、Users对象

```
1. public class Users {
2.     private String name;
3.     private Integer age;
4.     private Person person;
5.     //对应的get set toString方法
6. }
```

2、Controller层

1、对应的controller请求

```
1. @RequestMapping("/TestPojo")
2. public String TestPojo(Users users) {
3.     System.out.println("users:"+users);
4.     return "success";
5. }
6. =====页面=====
7. <form ACTION="TestPojo" METHOD="post">
8.     name:<INPUT TYPE="text" NAME="name"><br/>
9.     age:<INPUT TYPE="text" NAME="age"><br/>
10.    person.pname:<INPUT TYPE="text" NAME="person.pname"><br/>
11.    person.page:<INPUT TYPE="text" NAME="person.page"><br/>
12.    <INPUT TYPE="submit" VALUE="提交"><br/>
13. </form>
```

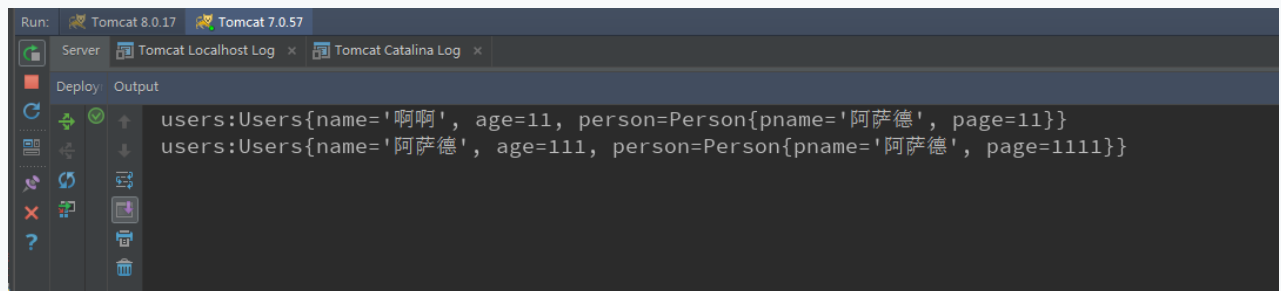
2、xml文件的设置字符集

```

1.     <filter>
2.         <filter-name>characterEncodingFilter</filter-name>
3.     <filter-
class>org.springframework.web.filter.CharacterEncodingFilter</filter-c
lass>
4.         <init-param>
5.             <param-name>encoding</param-name>
6.             <param-value>UTF-8</param-value>
7.         </init-param>
8.         <init-param>
9.             <param-name>forceEncoding</param-name>
10.            <param-value>true</param-value>
11.        </init-param>
12.    </filter>
13.    <filter-mapping>
14.        <filter-name>characterEncodingFilter</filter-name>
15.        <url-pattern>/*</url-pattern>
16.    </filter-mapping>

```

3、执行结果



六：Servlet API 作为定请求参数值

1、controller层代码

```

1.     @RequestMapping("/TestServletAPI")
2.     public String TestServletAPI (HttpServletRequest httpServletRequest
, HttpServletResponse httpServletResponse) {
3.         System.out.println("httpServletRequest:"+httpServletRequest.get
Parameter("name"));
4.         System.out.println("httpServletResponse:"+httpServletResponse);
5.         return "success";
6.     }

```

7. =====页面请求=====
8. `TestServletAPI`

2、执行结果

