项目开发的流程和注意事项 传递参数，如何排查错误的问题

> 选择有时候比努力更重要，但是放弃有时候比选择更重要。

springmvc传递参数

> *java.net.URL* 统一资源定位符，就如何通过请求请求网络的资源。
>
> *javaweb---jsp/servlet http*的认识*URL*

URL的组成

协议 （http/https）+ 端口（80/8080） + 上下文（contextPath /）+ 访问资源路径(/api/list) + 参数(?name=value&name1=value2) - -

不论是http请求，mysql: jdbc: 请求，还是redis请求，底层都是TCP/IP协议。特点：IP 和端口

它目的：都在找到电脑中各自服务，更具体执行是：在磁盘里对应的文件，因为操作系统中我们电脑中也只有：文件和文件夹---0/1

参数格式：以 ?name=value&name1=value2

总结

参数只有一种格式 ?name=value&name1=value2....

jquery / axios 传递参数是：{id:1,name:2}====内部会自动转成 ?id=1&name=2

```
var ajax = new XMLHttpRequest();
ajax.open("POST",url);
ajax.send("name=value&name1=value2")
```

请求方法

- HEAD,
- GET,
  - get请求特点 ：
    - 理论上GET请求数据长度没有限制的，真正起到限制的是浏览器对其长度进行了限制。参数暴露到访问地址

中，不安全的（就告诉我们不要把一些加密的数据用get去发送）

- 每个浏览器对请求地址URL的长度是有限制的，大部分浏览器的URL长度是8182字节，超过就剔除，这个不太适合做大文本的添加。

- POST
  - post请求的特点呢
    - 安全的，因为参数是放在请求体进行传输
    - 理论上tomcat服务器接收参数的时候，0代表不限制
    - ```
      web传输，前台的参数数据量过大【json格式的字符串】，可能达到几M，ajax调用后台方法时，无法传递
      问题分析：tomcat上默认post提交大小为2M，左右，超过这个大小了，就会传值不成功
      解决方法：修改post提交大小的限制大小，在server.xml上修改，如下：
      <Connector port="8080"
      protocol="HTTP/1.1"
      connectionTimeout="2000"
      redirectPort="8443"
      URIEncoding="UTF-8"
      maxThreads="3000"
      compression="on"
      compressableMimeType="text/html,text/xml"
      maxPostSize="10240"/>
      <Connection port="8009"
      enableLookups="false"
      redirectPort="8443" debug="0"
      protocol="AJP/1.3" />
      ```
      ```

```
12
13  其中参数maxPostSize="10240"是限制post请求参数的大小，tomcat7.0.63之前的版本设置为0和负数均可以代表不限制。但是7.0.63之后的版本只有设置为负数才代表不限制数据大小。
14
```

- PUT,
- PATCH,
- DELETE,
- OPTIONS,
- TRACE;

测试请求工具

- PostMan（客户端）（国外）
- ApiFox （客户端）（推荐--国产）
- 小幺鸡 (浏览器服务)
- Swagger (代码级别)
- idea安装rest client 插件

Get / Post请求传递 - params参数

get传递参数的格式是：http://localhost:9888/param1?id=

无@RequesetParam注解的版本

```
@GetMapping("/param1")
public String params1(String id,String name){
    System.out.println(id);
    System.out.println(name);
    return "success";
}
```

- 不传递参数直接请求：请求会进入方法，参数是null
- 传递参数直接请求：请求会进入，参数获取

问题如何参数必填怎么办？

- 自己控制（推荐做法）

```
@GetMapping("/param1")
    public String params1(String id,String name){
        Vsserts.isEmptyEx(id,new RuntimeException("请输入ID!!!"));
        Vsserts.isEmptyEx(name,new RuntimeException("请输入名字!!!"));
        System.out.println(id);
        System.out.println(name);
        return "success";
    }
```

推荐做法，为什么因为在程序中，我们可以自由意志去控制逻辑和参数。

- 用springmvc的提供参数的注解进行拦截处理 @RequesetParam

```
@GetMapping("/param2")
public String params2(@RequestParam(value = "id") String id, @RequestParam(value = "name") String name) {
    System.out.println(id);
    System.out.println(name);
    return "success";
}
```

  - 不传递参数直接请求：方法不进入，springMvc默认下会对请求方法中增加了@RequestParam的参数进行限制和校验，从而来满足用的所谓必填性和默认性。一句话：就可以去满足你程序中参数是否必填和给予默认值的机制。

如果不填，因为默认情况下@RequestParam修饰的参数，必填的required=true。所以如果你传递参数直接访问就直接报异常如下：

```
Required request parameter 'id' for method parameter type String is not present]
```

- 传递参数直接请求：请求会进入，参数获取 (正常访问)

Restful / SEO

Restful 对请求资源的一种约束规范：一句话最好遵循可以让请求更加清晰和命令，服务于seo搜索和收入。

飞哥送大家一句话：爱遵循就遵循，不遵循也无所谓。

我们知道在开发：CURD

- select - GET
- insert - PUT
- update - POST
- delete - DELETE

一般情况下：

- select - GET

- insert update delete - POST

query参数

语法：请求地址+"/参数值1/参数值2/....."

- v-router
  - this.$router.param.id === ?id=1&name=zhangsan
  - this.$router.query.id = / 1/2/

params参数和query参数的区别

params = http://localhost:9888/param7?id=1&name=zhangsan&age=35

query = http://localhost:9888/param7/1/zhangsan/35

在springmvc也支持query参数的获取，但是只能通过@PathVariable来获取。如下：

```
1 package com.kuangstudy.controller.params;
2
3 import com.kuangstudy.utils.fn.asserts.Vsserts;
4 import com.kuangstudy.vo.ParamsVo;
5 import lombok.extern.slf4j.Slf4j;
6 import org.springframework.stereotype.Controller;
7 import org.springframework.web.bind.annotation.*;
8
```

```java
/**
 * Description:
 * Author: yykk Administrator
 * Version: 1.0
 * Create Date Time: 2021/12/13 20:17.
 * Update Date Time:
 *
 * @see
 */

@RestController
@Slf4j
public class PararmeterController2 {


    // 推荐1
    @PostMapping("/param7/{id}/{name}/{age}")
    public String param7(@PathVariable("id") String id,
                         @PathVariable("name") String name,
                         @PathVariable("age") String age) {
        log.info("你传递的参数是：id:{},name:{},age：{}", id, name, age);
        return "success";
    }

    // 推荐2
    @PostMapping("/param9/{id}/{name}/{age}")
```

```
35      public String param9(@PathVariable("id")
    String userid,
36                           @PathVariable("name")
    String username,
37                           @PathVariable("age")
    String userage) {
38          log.info("你传递的参数是：id:{},name:
    {},age：{}", userid
39                  , username, userage);
40          return "success";
41      }
42
43
44      // 不要写
45      @PostMapping("/param8/{id}/{name}/{age}")
46      public String param8(@PathVariable String id,
47                           @PathVariable String
    name,
48                           @PathVariable String
    age) {
49          log.info("你传递的参数是：id:{},name:
    {},age：{}", id, name, age);
50          return "success";
51      }
52  }
53
```

状态码

- 405：参数有问题，参数类型，参的注入不匹配
- 400：请求的资源路径请求方式不对， post---axios.get
- 404：请求的地址写错了，你检查你请求地址 http://localhost:8080/api/xxxx/sdsd
  - 请求上下文增加了没有
  - 你路径名字 写错了，单词写错了吗？
  - 你资源是不是没被加载target目录下。

Post请求

没加： @RequestBody

- 默认情况下：get post请求也好，其实获取参数的方式是一模一样的。直接都能从url的后面的参数中获取到。

```
axios.get(url+"?pageNo=1&pageSize=123")
axios.get(url,{params:
{pageNo:1,pageSize:123}});

axios.post(url+"?pageNo=1&pageSize=123")
axios.post(url,{pageNo:1,pageSize:123});
```

加了 @RequestBody

- 如果你在请求方法中如果一定增加了 @RequestBody 就代表get/post参数的获取只能在请求体去存放。不能直接跟在url后方。也就说如果你一个参数或者对象用 @RequestBody修饰，这个参数只能在请求体中出现。
- 也同时告诉你，增加 @RequestBody说明你参数不能在直接写url后方，可能考虑到安全性

```
axios.post(url,{pageNo:1,pageSize:123});
axios.get(url,{params:{pageNo:1,pageSize:123}});
```

Query参数真的是参数吗?

不是。参数只有一种形态，?id=1&name=xxx。因为我们指定参数的存放在servlet是通过request.getParameter("name")如下：

```
System.out.println(request.getParameter("id"));
System.out.println(request.getParameter("name"));
System.out.println(request.getParameter("age"));
```

证明query不是参数:

```
@PostMapping("/param7/{id}/{name}/{age}")
    public String param7(@PathVariable("id") String id,
                         @PathVariable("name") String name,
                         @PathVariable("age") String age, HttpServletRequest request) {


System.out.println(request.getParameter("id"));

System.out.println(request.getParameter("name"));

System.out.println(request.getParameter("age"));
        log.info("你传递的参数是：id:{},name:{},age：{}", id, name, age);
        return "success";
    }

```

访问：

http://localhost:9888/param7/1/zhangsan/32

很清楚的看到

```
System.out.println(request.getParameter("id"));
System.out.println(request.getParameter("name"));
System.out.println(request.getParameter("age"));
```

全部是null, 说明它们不是参数，

总结

说明query只不过是restful遵循下方的一种类似于获取参数的一直机制。但是它不能通过request去获取。不是真正意义上所谓的参数，参数传递其实有两种方式：

- params
- query

不论你用那种方式，只要能解决你的业务其实就可以了。但是作为程序开发人员还要知道它们的区别也很重要。一句话：开发中不要太过于纠结你到底是要params的方式还是query 方式，只要能达到效果即可。但是如果你一定要在未来请求对象中获取到你的参数最好要使用params. 使用query有一个好处，简化你的url名字定义以及方便搜索引擎收录。仅此而已。底层是通过ongl正则表达式实现的一种字符串匹配。

总结

- 如果你一个参数用@RequestParam一定是必填的。---强制性的-
    - 如果不传递就会报错，影响程序的执行
    - 不容易去捕获错误
- 如果你不想填，最好不要增加@RequestParam

- 在程序开发过程中，如果你想自由意志的去控制程序和执行，最好的方式还自己来处理默认值和校验。统一不增加@RequestParam
- 如果参数是多个，建议使用对象 vo包装统一获取。一般超过3个以上建议直接用vo
- post请求是相对安全的，只能通过 form表单、工具才能访问，
- 所以访问页面只能是Get请求。
- 在程序开发过程中，不要纠结请求方式到底是get、post，它们接受参数的方式都是一模一样，只不过在未来的程序开发中，post请求，建议大家用@RquestBody请求体会更好一些。