### Java 请求参数

1. @RequestBody使用详解

@RequestBody主要用来接收前端传递给后端的json字符串中的数据的(请求体中的数据的)；GET方式无请求体，所以使用@RequestBody接收数据时，前端不能使用GET方式提交数据，而是用POST方式进行提交。在后端的同一个接收方法里，@RequestBody与@RequestParam()可以同时使用，@RequestBody最多只能有一个，而@RequestParam()可以有多个。

注：一个请求，只有一个RequestBody；一个请求，可以有多个RequestParam。

注：当同时使用@RequestParam（）和@RequestBody时，@RequestParam（）指定的参数可以是普通元素、 数组、集合、对象等等(即:当，@RequestBody 与@RequestParam()可以同时使用时，原SpringMVC接收参数的机制不变，只不过RequestBody 接收的是请求体里面的数据；而RequestParam接收的是key-value 里面的参数，所以它会被切面进行处理从而可以用普通元素、数组、集合、对象等接收)。

       即：如果参数时放在请求体中，传入后台的话，那么后台要用@RequestBody才能接收到；如果不是放在请求体中的话，那么后台接收前台传过来的参数时，要用@RequestParam来接收，或则形参前什么也不写也能接收。

注：如果参数前写了@RequestParam(xxx)，那么前端必须有对应的xxx名字才行(不管其是否有值，当然可以通过设置该注解的required属性来调节是否必须传)，如果没有xxx名的话，那么请求会出错，报400。

注：如果参数前不写@RequestParam(xxx)的话，那么就前端可以有可以没有对应的xxx名字才行，如果有xxx名的话，那么就会自动匹配；没有的话，请求也能正确发送。

       追注：这里与feign消费服务时不同；feign消费服务时，如果参数前什么也不写，那么会被默认是@RequestBody的。

如果后端参数是一个对象，且该参数前是以@RequestBody修饰的，那么前端传递json参数时，必须满足以下要求：

后端@RequestBody注解对应的类在将HTTP的输入流(含请求体)装配到目标类(即：@RequestBody后面的类)时，会根据json字符串中的key来匹配对应实体类的属性，如果匹配一致且json中的该key对应的值符合(或可转换为)，这一条我会在下面详细分析，其他的都可简单略过，但是本文末的核心逻辑代码以及几个结论一定要看！ 实体类的对应属性的类型要求时,会调用实体类的setter方法将值赋给该属性。

json字符串中，如果value为""的话，后端对应属性如果是String类型的，那么接受到的就是""，如果是后端属性的类型是Integer、Double等类型，那么接收到的就是null。

json字符串中，如果value为null的话，后端对应收到的就是null。

如果某个参数没有value的话，在传json字符串给后端时，要么干脆就不把该字段写到json字符串中；要么写value时， 必须有值，null  或""都行

**2、Get与Post区别**

  Get和Post都是Http协议的组件，所以底层都是使用tcp链接。

Get的请求方式是将http的header和data一并发往服务端，也就是一条tcp数据包发送，这就会有两个问题：数据量有限，依赖于Tcp负载能力，所以携带的数据量很大的情况下，容易造成重发。所有的携带的数据只能接受转化成ASCII字符。

  Post不一样，post使用两步走，先发送http的header，然后再传输data。数据类型也不受限制。而且数据隐秘性比较好。

三、Get方式参数获取

  get请求方式参数是拼接在url后，所以限制了可以发送的长度。Get不支持使用http Body获取参数，他只支持params，也就是URL拼接参数。

1、默认方式

@RequestMapping(value = "/getUserInfo")

public User getUserInfo(Integer id){

}

 默认情况下会从Param（就是URL后面的拼接参数）获取名字是id的项，自动转化成Integer类型，其他的类型也是类似。

2、使用@RequestParam

@RequestMapping(value = "/getUserInfo")

public User getUserInfo(@RequestParam(value = "id",required = false) Integer id){

}

3、使用HttpServletRequest

@RequestMapping(value = "/getUserInfo")

public User getUserInfo(HttpServletRequest request){

Integer id = Integer.parseInt(request.getParameter("id"));

if (id.intValue() == 0) {

return null;

}

}

  这个方法是获取整个URL的信息，然后手动获取和分离参数，和类型转化。这个里面带的内容很多，不仅Param还有Header，Cookies等。

四、Post方式参数获取

  post可以传递参数可以大致分成两种，一种是表单：在sevlet实现中mutipart/form-data和application/x-www-form-urlencoded会被特殊处理，请求参数将被放置于request.paramter，解析成map。第二种，已application/json，参数是存放在json中的，参数必须要用@RequestBody才能解析出来。

1、使用@RequestBody

@RequestMapping(value = "/getUserInfo")

public User getUserInfo(@RequestBody Integer id){

}

  这种方式传递方式必须使用Content-Type=application/json。这个不仅可以指定id等具体对象，也可以用Map，直接把所有对象拿到手，然后再(Integer) map.get(“id”)转化。

2、使用@RequestParam

@RequestMapping(value = "/getUserInfo")

public User getUserInfo(@RequestParam(value = "id",required = false) Integer id){

}

这种方式只用在Content-Type=mutipart/form-data和

Content-Type=application/x-www-form-urlencoded这种情况下才能使用，sevlet将Body中的key-value转成Param。

  如果是这种方式，URL后面拼接Param，也就是类似Get请求的方式，这样的post请求，@RequestParam是可以用的。能够获取后面的参数

  有一个有趣的现象，如果Content-Type=mutipart/form-data，Body中加入参数和URL后面拼接参数一起做Post请求，都可以被加载到Param中，如果是同名的，只取用form-data中的。

  如果Content-Type=application/x-www-form-urlencoded和URL拼接的一起，如果是String类型，则两个值会被拼接，其他类型取的是URL拼接的参数。

2、通用HttpServletRequest

  和Get方式一样，这是个通用的方式。这个也可以和URL拼接的一起搭，但是没有RequestParam的String类型值被拼接问题，优先级 form-data高于URL拼接高于x-www-form-urlencoded。