Restful接口即http接口，

1前端客户端(后端用httpclient)可以通过ajax发送json,类型或者参数类型，后端（服务端）用controller接收(写服务可以返回json对象给客户端（可以是前端，也可后端）处理，还可以返回普通map对象交给jsp渲染 )

2controler用法

@ResponseBody注解再方法上表示服务端返回的是jsonobject类型

@RequestBody 接收参数表明客户端发送的jsonObject类型，前端ajax发送的contentType必须application/json格式，具体可以转化为实体类，（参数一一对应）或josnObject,

@RequestMapping("/callback")默认不填写method时，可以接收任何类型请求方式，还可以指定post,get，接收指定的请求类型@RequestMapping(value="/callback",method = RequestMethod.*POST*)，（get请求暴露请求参数，不安全，post放在了body里，是安全的）

@RequestMapping("/testSinglePointStations")

例子：TestJob实体类  
@ResponseBody  
public String testSinglePointStations(@RequestBody TestJob testJob) {  
 String protocolVersion = testJob.getProtocolVersion();  
  
 if (testJob.getStations() == null || testJob.getShelfNum() <= 0) {  
 return "Error:" + JSONUtil.*toJSONString*(testJob);  
 }  
 System.*out*.println(testJob.getShelfNum());  
 System.*out*.println(testJob.getStations());  
 List<Integer> stations = testJob.getStations();  
 int shelfNum = testJob.getShelfNum();  
 List<String> shelfCodelist = baseInfoService.getAllShelfCodeList();  
 if(shelfCodelist.size()<=0){  
 return "Error,货架数量不够";  
 }

例2：接收jsonObject

@RequestMapping("/callback")  
@ResponseBody  
public String testSinglePointStations(@RequestBody JSONObject resJson) {  
 String backMessage=null;  
 JSONObject callbackBodyJson=resJson.getJSONObject("request").getJSONObject("body");  
 JSONObject callbackHeaderJson=resJson.getJSONObject("request").getJSONObject("header");  
 CallbackHeader callbackHeader=messageConvert.Json2CallbackHeader(callbackHeaderJson);  
 MapEntity robotTaskCallbackBody=messageConvert.Json2MapEntity(callbackBodyJson);  
 if(robotTaskCallbackBody!=null){  
 *logger*.info("http callback message,header:{},body:{}", callbackHeader.toString(),robotTaskCallbackBody.toString());  
 APIMessageHandler.*addRobotCallBackTask*(robotTaskCallbackBody);  
 backMessage = "{ \"id\": \"\*\", \"msgType\": \"com.geekplus.athena.api.msg.callback.RobotTaskCallbackResponseMsg\", \"response\": { \"header\": { \"responseId\": \""+callbackHeaderJson.getString("requestId")+"\", \"code\": 0, \"msg\": \"success\" } } }";  
 }  
 return backMessage;

Ajax发送接口header中的几种文本类型contentType(**x-www-form-urlencoded,** **multipart/form-data,** **application/json**)如下

1. **x-www-form-urlencoded,** 可以通过form提交，但是不能有file类型的控件，为get请求时浏览器会把form数据转换成一个字串（name1=value1&name2=value2…），

**form.serialize()就把form的内容name 和value转化成name1=value1&name2=value2键值对了**

可在浏览器地址输入url，后台controller用@RequestParam接收参数RequestParam可以指定控件name,多个用逗号隔开，例如:

\*@RequestParam ("QueueSize") string queueSiz, @RequestParam ("name") string namre.。也可以将所有键值对批量一次性加载到map里:

\*@RequestParam Map<String, Integer> shelfHeatList,缺点必须类型一致，一般就string ，int，可以取到后转换

**2.multipart/form-data**

跟**x-www-form-urlencoded区别在于可以发送file文件，其他跟x-www-form-urlencoded用法一样**

\*

**var** json1 = $('#uploadForm1').serialize();  
  
  
$.ajax({  
 url: "http://" + location.hostname + ":61118/map/QueueSize",  
 type: "post",  
 cache: **false**,  
 //  
 data: json1,  
 // processData: false,  
 //用于对参数进行参数化处理，默认true  
 // processData: true,  
 //不用指定类型 form 得entype已经指定  
 // contentType: 'application/json;charset=utf-8',  
 contentType: 'application/x-www-form-urlencoded',  
 success: **function** (result) {  
 console.log(result);  
 // console.log(result.msg);  
 /\* document.getElementById("div1").value=\*/  
  
  
 // alert(result.msg);  
   
 **var** responseText = document.getElementById("responseText");  
 responseText.value = result.mapText.toString();  
  
 }

3

当为json类型时，ajax两个属性如下

data: JSON.stringify({robotNum: robotNum, server: server, mode: mode,version:version}),

contentType: ' "application/json;charset=UTF-8",

controller接收参数用@RequestBody接收json格式的数据，可以用bean接收，可以用jsonObject接收解析；

public Map<String, String> config(@RequestBody TestJob testJob) throws IOException, DocumentException {

',