接口持续集成

postman+newman+jenkins持续集成

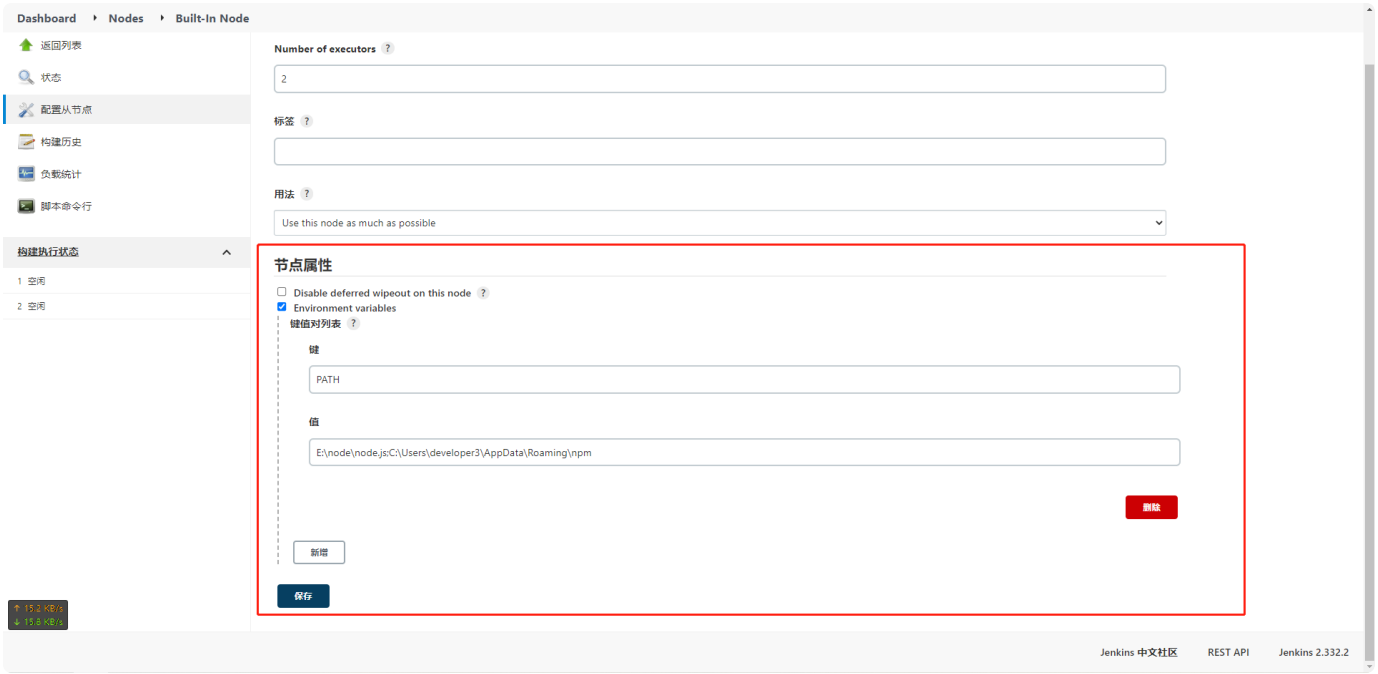
jenkins学习问题点

jenkins学习问题点(1)

当jenkins新建item构建运行newman脚本时,出现没有启动用户时(大部分可能会去查看jenkins进程PID其实jenkinsPID是启动状态),可能是Manage nodes and clouds内部没有设置节点属性

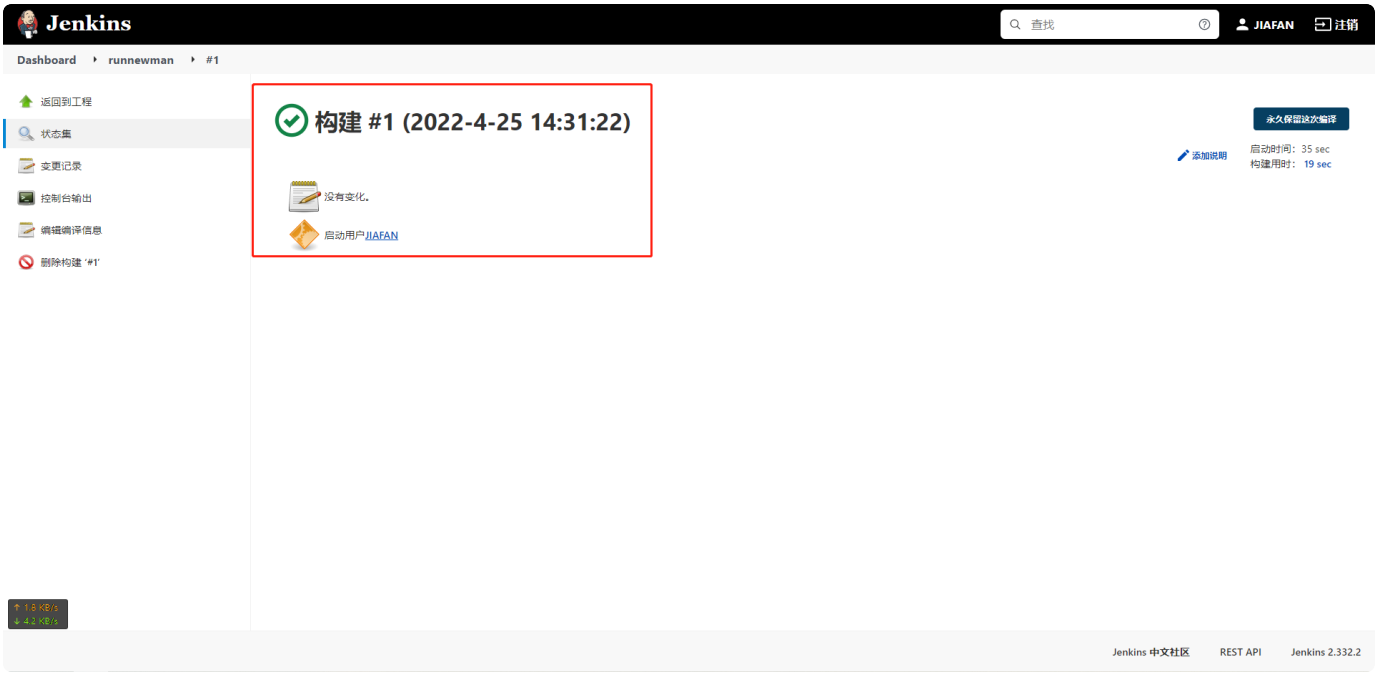
配置node安装路径:E:\node\node.js

以及系统内npm路径:C:\Users\developer3\AppData\Roaming\npm



点击保存即可关联Built-In Node

再次构建item时启动脚本运行即可



以上是jenkins问题点,亲测有效

postman学习问题点

1、postman手写内置函数使用JavaScript

（1）写至Pre-request Script内调用至请求参数处

生成数据 JavaScript | 复制代码

```javascript
var num ="";
for(var i =0;i<4;i++){
    num += Math.floor(Math.random()*10)
}
//生成用户名
var username = "test" + num;
//打印username
console.log(username);
pm.variables.set('username',username);

//==========================================================================
=====
//生成编号信息
var num ="";
for(var i =0;i<2;i++){
    num += Math.floor(Math.random()*10)
}
//编号信息
var workNumberid = "90" + num;
//打印workNumber
console.log(workNumberid);
pm.variables.set('workNumber_id',workNumberid);

//==========================================================================

//随机生成一个电话号码
//前三位生成"135","138","166","188","199"等等
var list=["135","138","166","188","199"];
//list[下标]取值,取列表中的值,下边是随机变化的
//求列表的长度lisi.lenght
//随机获取一个下标
var index = Math.floor(Math.random()*list.length)
var pre_phone3=list[index];
//生成后8位随机数字
var back_phone8 = "";
for(var i=0;i<8;i++){
    back_phone8 += Math.floor(Math.random()*10);
}
//电话号码
var phonenum = pre_phone3+back_phone8
//打印电话号码
console.log(phonenum)
//设置本地变量,请求直接{{phonenum}}调用
pm.variables.set("phonenum",phonenum)
```

（2）手写内置函数放置Body请求参数内，请求传参数值标准“{{}}”

JSON JavaScript 复制代码

```
{
    "username":"{{username}}",
    "mobile":"{{phonenum}}",
    "workNumber":"{{workNumber_id}}",
    "timeOfEntry":"",
    "formOfEmployment":"1",
    "departmentId":"",
    "departmentName": "",
    "correctionTime":""
}
```

（3）postman中console用于调试接口请求过程

Test后置脚本提取内容以及tonken方式（快捷方式先打印，后提取至变量内）

提取id值设置至环境变量内

json提取 JavaScript 复制代码

```
//打印json内容
var user_Body = JSON.parse(responseBody);
console.log("这是json返回内容:",user_Body);
//打印target_id
var target_id = user_Body.data.id;
console.log("这是返回target_id的值:",target_id)
//提取target_id的变量
pm.globals.set("target_id",target_id)
```

打印token提取至环境变量内

提取token JavaScript | 复制代码

```javascript
//打印json内容
var json_data = JSON.parse(responseBody);
console.log("这是json的内容",json_data)
//打印token内容
var json_token = json_data.data
console.log("这是token内容",json_token)

//提取token
var jsonData = pm.response.json()
//提取data的值,拼接data的值,拼接Bearer空格 前缀
var token = "Bearer" + jsonData.data.token
//设置环境变量
pm.globals.set("IHRMtoken",jsonData.data);
```

postman动态参数使用方式

动态参数官方文档：https://learning.postman.com/docs/writing-scripts/script-references/variables-list/

哪些情况需要保证接口幂等性

哪些情况需要保证接口幂等?

对于业务中需要考虑幂等性的地方一般都是接口的重复请求，重复请求是指同一个请求因为某些原因被多次提交。导致这个情况会有几种场景

1.前端重复提交

用户在新增页面上快速点击多次，造成发了多次请求，后端重复保存了多条一模一样的数据。如用户提交订单。生成很多重复的订单;

1.消息重复消费

消息重复消费。一般指的是消息中间件。如RabbitMQ，由于网络抖动,MQ Broker将消息发送给消费斌消费，消费端进行了消费，在返回ack给MQ Broker时网络中断等原因，导致MQBrokeri认为消费端没能正常消费消息。这时候MQ Broker会重复将这条消息发给消费端进行消费，如果没有做幂等，就会造成客户端重复消费同一条消息。

1.页面回退再次提交

举个例子。用户购买商品的时候。如果第一次点击下单按钮后。提示下单成功。跳转到下单成功页面，这时候如果用户点击浏览器返回按钮，返阿上一个下单页面，重新点击下单按钮，这时候如果没有做幂等的话。也会造成重复下单的问题。

1.微服务互相调用

分布式系统中。服务之间的通信一般都通过RPC或者Feign进行调用，难免网络会出小问题，导致此次请求失败，这时候这些远程调用，如feign都会触发重试机制。所以我们也需要保证接口幂等。

什么是接口幂等

postman之postman newman jenkins实现持续集成

1、新建项目

2、设置自定义的工作空间

3、执行windows的批处理命令

4、执行系统单的Groovy脚本

5、生成的html的报告集成jenkins

（特别点）

构建触发器Build periodically

*(分) *(时) * * *构建触发器时间(定时器)

尽然有postman,jmeter等这些接口测试工具,那么为什么很多企业还要做接口自动化呢?

1.接口数量巨大,方便?有团队协作功能?

2.很多的功能比较死板,csv数据驱动

3.只支持http协议接口,微服务

4.找错和定位错误只能通过congsle

5.生成的报告有点丑

6.多接口的串联,数据库验证,日志监控

7.接口自动化和web自动化一起做

接口自动化最核心,接口自动的框架.

最终:投入和产出比,解放人力

框架调整

1.一个接口的项目中有多个基础路径,每一个模块一个基础路径.

2.增加了cookie关联

newman

三、newman（新男人,非GUI的方式运行）

postman为接口而生，newman为poatman而生

运行命令：newman run

常用参数：

–e 引入环境变量

–g 使用全局变量

–d 使用csv ， json数据驱动

–n 循环次数

–r cli.html.json .junit –––reporter–html–export 生成html报告

–r htmlextra 生成好看的html报告

应用步骤：

1.导出用例，环境变量，全局变量，数据文件，导出都为json格式

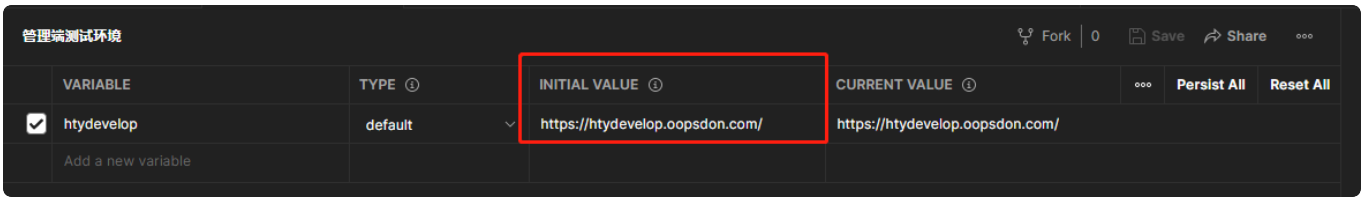
2.运行(cmd)

newman run E:\postman_json\IHRM人力资源管理系统接口.postman_collection.json –e E:\postman_json\IHRM测试 环境.postman_environment.json –g E:\postman_json\workspace.IHRM全局变量postman_globals.json

生成报告

newman run E:\postman_json\IHRM人力资源管理系统接口.postman_collection.json –e E:\postman_json\IHRM测试 环境.postman_environment.json –g E:\postman_json\workspace.IHRM全局变量postman_globals.json –r cli.html,json,junit ––reporter –html–export "e:\\newmanreport.html"(生成报告路径)

问题点:导出环境变量时 INITIAL VALUE 填写上参数,不然newman判断没有请求地址判断没有地址(jenkins持续集成时也会判断出错)



postman之接口加密解密

一、接口加密解密

目前的市面上有哪些加解密的方式

加密/解密网站：https://www.bejson.com/

1.对称式的加密方式，不常用DES和AES（已经被淘汰了 ）

Base64加密：

base64加密方式 JavaScript 复制代码

```
//base64加密方式
var username = CryptoJS.enc.Utf8.parse("admin");//转为utf-8的编码
var base64_username = CryptoJS.enc.Base64.stringify(username);//转为base64位加密
console.log(base64_username);
```

Base64解密：

//base64解密方式 JavaScript 复制代码

```
//base64解密方式:
var username = CryptoJS.enc.Base64.parse("YWRtaW4=");//转为base64位解密
var new_username = username.toString(CryptoJS.enc.Utf8);//转为utf-8
console.log(new_username);
```

2.非对称的加密方式（双钥（公钥和私钥）加密）：RSA加密方式

由一个密码生成的双钥，公钥做加密，私钥做解密。私钥加密，公钥解密

3.只加密不解密

1.md5加密（企业中实用最多）

md5加密 JavaScript 复制代码

```
//md5加密的32位
var new_username2 = CryptoJS.MD5("admin").toString().toUpperCase();
console.log(new_username2);
```

2.SHA1,SHA3,SHAN.....

特殊的加密：

快分期：贷款业务,加密都是自定义混合加密

admin,base64(YWRtaW4=**),** YWRtaW4=+DFJK/模块名首字母,MD5加密

二、接口签名（sign）-----了解

postman之Cookie鉴权

一、postman的Cookie鉴权

接口鉴权，接口加密，接口签名（金融项目，银行项目，信贷项目，特大型项目）

什么是cooike?

cookie是一段文本信息，客户端第一次访问服务器是，那么服务器不知道客户端的身份，所有就需要创建一个身份标识，这个身份标识就是cookie，以键值对的方式保存。格式key=value。

cookie鉴权的原理：

1、当客户端第一次访问服务器的时候，那么服务器就会生成cookie信息，并且在响应头的set-cookie里面吧生成的cookie信息发送给客户端，客户端接收到cookie之后就会保存起来！

2、当客户端第2-N访问服务器的时候，那么客户端就会在请求头的cookie带上cookie信息，从而实现鉴权。

cookie的分类

会话cookie：保存在内存当中，浏览器关闭后自动消失

持久cookie：保存在硬盘中，浏览器关闭之后不会消失，只有当持久化的时间到期了才会消失

数据格式：

name：cookie名称

value：cookie的值

domain：cookie作为的ip地址（访问服务器）

path：cookie所在服务器上面的项目路径

exprise in：失效时间

size：大小

凡是网页端（web）项目，95%以上都是存在cookie鉴权

说明：postman能够自动保存第一次访问服务器的cookie信息，并且能够在第2-N次的时候自动的带上cookie的信息

postman之必须带请求头的接口测试方式

一、必须带请求头的接口测试方式

请求头参数输入进headers内部

Accept: /

Accept-Encoding: gzip, deflate, br 客户端接收的数据压缩方式

Accept-Language: zh-CN,zh;q=0.9 客户端接收的数据编码格式

Cache-Control: no-cache

Connection: keep-alive 请求方式:保持活跃

Cookie: 请求的cookie信息
COOKIE_SESSION=34_0_6_8_6_6_1_0_5_6_1_0_1138_0_0_0_1650421790_0_1650603837%7C9%230_1_1650006885%7C1; WWW_ST=1650603842762

Host: www.baidu.com请求的主机地址

is_referer: https://www.baidu.com/

is_xhr: 1

Pragma: no-cache

Referer: 来源(避免被攻击sql注入)

sec-ch-ua: " Not A;Brand";v="99", "Chromium";v="100", "Google Chrome";v="100"

sec-ch-ua-mobile: ?0

sec-ch-ua-platform: "Windows"

Sec-Fetch-Dest: empty

Sec-Fetch-Mode: cors

Sec-Fetch-Site: same-origin

User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/100.0.4896.127 Safari/537.36 请求的客户端的类型(模拟浏览器的请求)

X-Requested-With: XMLHttpRequest 异步请求:长沙到北京,飞机,高铁,走路,局部刷新

Content-Type:

from-data: Content-Type:multipart/from-data表单文件上传

x-www-from-urlencoded: Content-Type: x-www-from-urlencoded:表单提交

raw: Content-Type:applicaton/json 根据你选择的数据类型来决定

binary: Content-Type:binary:二进制文件

二、接口mock

使用场景：前后端分离，后端的接口数据没有出来，前端需要调用后端的接口实现业务，为了保证前端能够正常的开发以及测试，那么就需要使用mock，模拟桩。（模拟一个接口供给前端使用）

预告:

加密接口,newman,jenkins持续集成

接口测试之用例批量执行

一、断言

使用公共断言Edit

设置公共断言

公共断言 JavaScript 复制代码

```
//公共断言
pm.test("Status code is 200", function () {
    pm.response.to.have.status(200);
});
```

检查json的key与value

检查JSON响应数据 JavaScript 复制代码

```
//检查json的数据
pm.test("Your test message", function () {
    var jsonData = pm.response.json();
    pm.expect(jsonData.message).to.eql("操作成功! ");
});
```

应有与返回的数据很少的情况下

断言等于字符串

业务断言 JavaScript 复制代码

```
//-------业务断言-----------
//断言返回字符串
pm.test("Body is correct", function () {
    pm.response.to.have.body('{"Message":"OK","ErrorLoginNum":0}');
});
```

三、postman用例的批量执行

数据驱动：csv.json

特别注意:在参数里面取数据文件的值和取全局变量一致，在断言内取数据文件的值使用data[名称]

csv文件编码格式必须为utf-8

postman之动态参数和断言

遗留问题:

创建标签的接口便签名和已存在的标签名重复

删除便签只能删除固定id的标签

增、改、查、删、接口流程（形成闭环）

删除便签只能删除固定id的标签

```JavaScript
//提取id的值
var res = JSON.parse(responseBody);
console.log(res.tag.id);
//设置全局变量
pm.globals.set("tag_id",res);
```

一、postman动态参数

1、postman内置动态参数

{{$timestamp}} 生成当前时间的时间戳(不是万能的)

{{$randomInt}} 生成0–1000的随机数

{{$guid}} 生成随机的guid字符串

动态参数官方文档：https://learning.postman.com/docs/writing-scripts/script-references/variables-list/

2、自定义动态参数

(1)手动制造一个时间时间戳

```JavaScript
//创建时间戳
var times = Date.now();
//提取到全局变量内
pm.globals.set("times",times);
```

(2)等待3秒

```javascript
//等待3秒
console.log("请求之前")
const sleep = (milliseconds) => {
const start = Date.now()
while (Data.now() <= start + milliseconds){}
};
sleep(3000);
console.log("请求之后")
```

一、断言

Ststus code:Code is 200 断言返回状态码为200

Response body: Contains string 断言返回的结果中包含一个指定的字符串

Response body:JSON value check 检查json中其中一个字段的值

检查json的key与value

```javascript
//检查json的数据
pm.test("Your test message", function () {
    var jsonData = pm.response.json();
    pm.expect(jsonData.message).to.eql("操作成功! ");
});
```

应有与返回的数据很少的情况下

Response body: is equal to a string 断言返回的结果是否等于一个字符串

```javascript
//--------业务断言-------------
//断言返回字符串
pm.test("Body is correct", function () {
    pm.response.to.have.body('{"Message":"OK","ErrorLoginNum":0}');
});
```

Response headers: Content-Type header check 检查是否有Content-Type响应头

Response time is less than 200ms 断言接口请求的时间少于200毫秒

Status code: Successful POST request 断言响应码在一个列表内

Status code: Code name has string 断言响应的信息为一个指定的字符串

```javascript
//------------断言---------------
//状态断言
pm.test("Status code is 200", function () {
    pm.response.to.have.status(200);
});
//业务断言
pm.test("Body matches string", function () {
    pm.expect(pm.response.text()).to.include("message");
});
```

特别注意：在Tsets断言里面不能使用{{}}取全局变量,只能通过pm.globals.get('times')或者globals['times']或者globals.times这种方式取全局变量

这种方式获取断言中的动态参数的值

第一种写法:

```javascript
//------------断言---------------
//状态断言
pm.test("Status code is 200", function () {
    pm.response.to.have.status(200);
});
//业务断言
pm.test("断言返回的结果中包含pid的值", function () {
    pm.expect(pm.response.text()).to.include(pm.globals.get("times"));
});

```

第二种写法:

```javascript
//-----------断言---------------
//状态断言
pm.test("Status code is 200", function () {
    pm.response.to.have.status(200);
});
//业务断言
pm.test("断言返回的结果中包含pid的值", function () {
    pm.expect(pm.response.text()).to.include(globals.times);
});

```

第三种写法(全部一起判断断言结果):

```javascript
// 开始断言
pm.test("断言返回结果", function () {
    var jsonData = pm.response.json();
    pm.expect(jsonData.Records[0].IsWhite).to.eql(true);
    pm.expect(jsonData.HscodeType).to.not.eql("HS")
})
```

判断接口返回为空

```javascript
pm.test('判断Records是否有值',function(){
var josnData = pm.response.json();
var Records = josnData.Records;
pm.expect(!!(Records)).not.to.eql(false);
});
```

postma之接口测试实战

六、全局变量和环境变量

全局变量:能够在所有接口请求里使用的变量叫全局变量

环境变量:环境变量是能够让我们的代码在多个不同环境下去执行,环境变量其实也是全局变量

http默认端口是80

https默认端口是443

七、接口关联

面试官的问题:

接口测试里面，接口关联是如何实现的?

接口测试当中上一个接口的返回值作为下一个接口的参数?

1.接口关联关联第一种方式(常用json代码)

json提取 JavaScript 复制代码

```javascript
1.//获取响应的正文主体
var resbody = responseBody
console.log(resbody)
2.//返回的是json的字符串, 可以转换成json的对象然后解析
JSON.parse(responseBody);
console.log(jsonobj.access_token);
3.//使用正则表达式提取
var value = responseBody.match(new Regexp("access_token": "(.*?)"));
console.log(value[1]);
```

接口关联第二种方式（正则表达式）

正则表达式 JavaScript 复制代码

```javascript
//1获取响应主体
var resbody = responseBody;
console.log(resbody);
//2使用正则表达式提取match匹配new的正则表达式
var value = responseBody.match(new RegExp('"access_token":"(.8?)"'));
console.log(value[1]);
//把提取的toke放刀全局变量内
pm.globals.set("access_token",value[1]);
```

其他方法:

1.地方

获取头部和cookie JavaScript 复制代码

```javascript
//获取响应Headers的值,需要从响应头取值时
var types = postman.getResponseHeader('Content-Type');
console.log(types);
//获取响应的cookie信息
var cookie_id = postman.getResponseCookie('ASP.NET_SessionId')
console.log(cookie_id['value']);
```

遗留问题:

创建标签的接口便签名和已存在的标签名重复

删除便签只能删除固定id的标签

postman接口测试流程

json：数据类型

主要包含：

键值对：用{}表示

列表：用[]表示

字典: (对象)

要包含：

键值对：用{}表示

列表：用[]表示

发现问题:

1.access_token鉴权码需要手动输入

2.标签名 不能和已存在的标签名重复

3.环境是写死的没有办法让用例在不同环境下执行

1.接口关联(常用json代码)

JavaScript | 复制代码

```
1.//获取响应的正文主体
var resbody = responseBody console.log(resbody)
2.返回的是json的字符串, 可以转换成json的对象然后解析
JSON.parse(responseBody);
console.log(jsonobj.access_token);
3.使用正则表达式提取
var value = responseBody.match(new Regexp("access_token": "(.*?)")); consol
e.log(value[1]);
```

接口测试里面, 接口关联如何实现?

postman之什么是接口测试

一、什么是接口测试

接口测试分两种

测试外部接口：系统和外部之间的接口（第三方）如：电商网站：支付宝支付、微信支付

测试内部接口：系统内部的模块之间的联调，或者子系统之间的数据交互

测试重点：测试接口参数传递的正确性，接口功能的正确性。输出结果的正确性，以及对各种异常情况的容错性和权限控制

测试接口过程?

老师：洗脚服务。

接口请求过程

1、接口地址

2、get，post接口请求方式

3、指定请求头

4、请求参数

接口响应过程：

1、响应状态码

2、响应信息

3、响应头

4、响应主体

二、接口测试工具

poatman+newman+jenkins+git/svn

jmeterant+jenkins+git/svn

postman简介：

postman一款功能强大的接口测试工具，专为接口而生

安装：

两个版本：

postman Chorme app（chorme浏览器，不推荐）

postman native app 客户端安装方式

注册,登录.只有登录才可以使用postman中的云服务功能

三、接口测试流程

1、拿到接口文档。（抓包：F12、fiddler、Charles）熟悉接口业务接口地址、请求信息、请求头信息、请求方式、鉴权方式

2、编写接口用例以及评审*

3、使用接口测试工具执行接口测试

4、输出接口测试报告（jmeter可以做出输出测试报告）

四、接口测试执行

鉴权码:是否有访问次接口的权限的一个字符串码

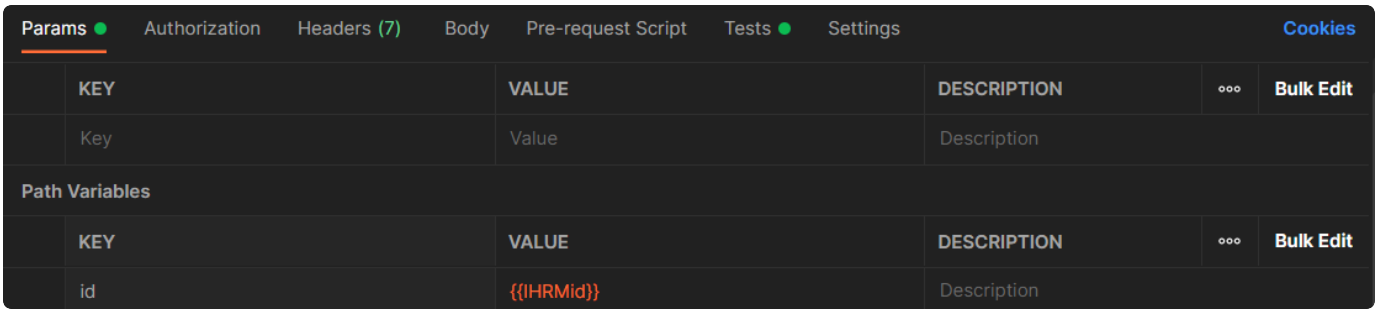
获取鉴权码的方式:

1、有一个专门获取token的鉴权码的接口

2、登录之后自动生成token的鉴权码

get请求以问号的方式传参,多个参数之间用&(并且)分隔

请求的功能页签:



Params:get请求传参

Authorization:是验证是否拥有从服务器访问所需数据的权限

Headers:请求头部

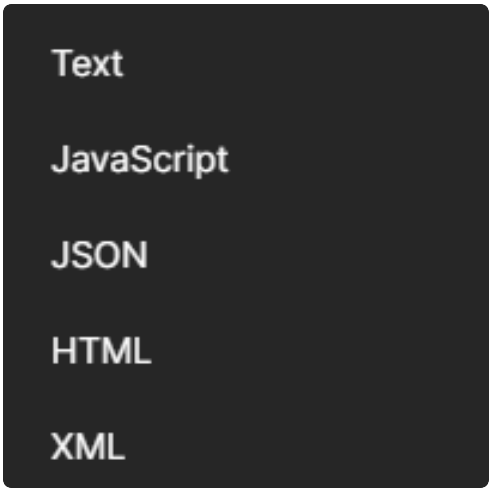
Body:

1.none:无

2.from_data:用于表单文件上传: text:键值对 还有file文件上传

3.x_www_from_urlencoded:表单""键值对

4.raw:传各种其他类型的参数,比如:



5.binary:用于上传二进制文件

Pre-request Script:接口请求之前的js脚本

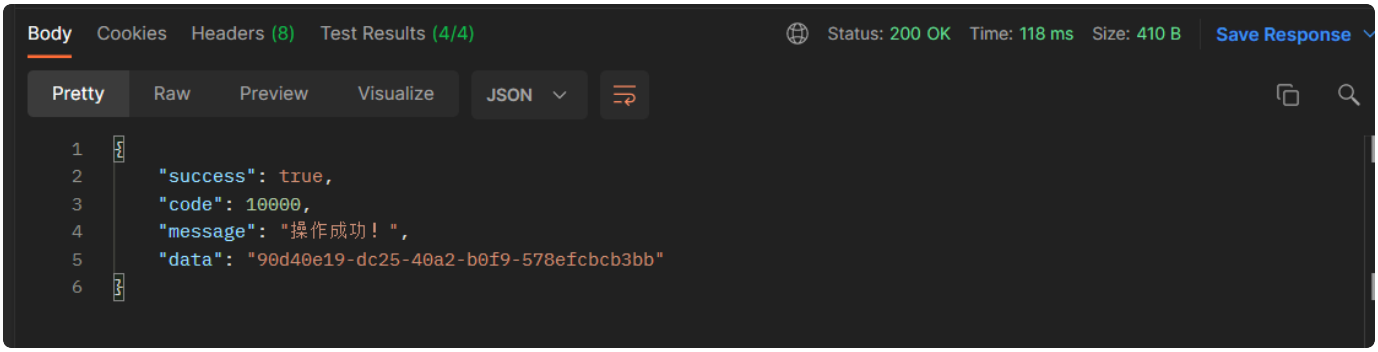
Tests:接口请求之后的断言(断言接口是否请求成功)

Settings:对请求头的设置

---

cookie:是postman用于自动管理cookie关联

响应功能页签



Body：响应内容

1.Pretty：json格式 ,raw:文本格式 Preview:网页格式

Cookies:响应的cookie信息

Headers:响应的四要素之一,响应头

Test Results:断言结果

响应状态码:200

响应信息:ok

响应时间和响应的字节数

Save Response:保存响应

Console :控制台,用于接口测试调试

面试题: get请求和post请求的区别是什么?

1.get请求一般是获取资源,post请求是提交数据

2.get请求是通过在地址栏中以?的形式传参, post请求是通过表单去传参

3.post请求比get请求安全