postman使用

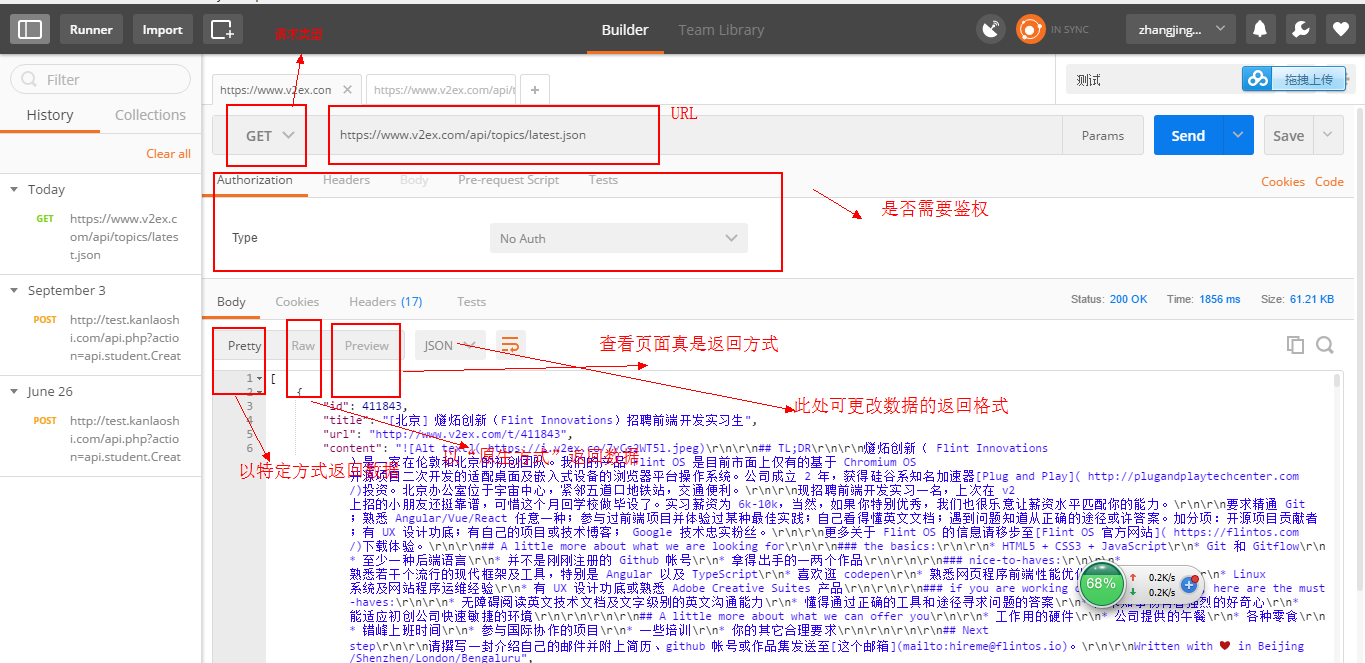
1. postman简介

当开发人员需要调试一个网页是否正常运行，并不是简简单单的调试网页的HTML、CSS等信息是否运行正常，更加重要的是网页能够正确处理各种HTTP请求，毕竟网页的HTTP请求是网站与用户之间进行交互的非常重要的一种方式，在动态网站中，用户的大部分数据都需要通过HTTP请求来与服务器进行交互

1. 发送第一个API请求：

***发送api请求***

1. 选择发送方法
2. 添加URL
3. 点击send
4. 查看返回数据

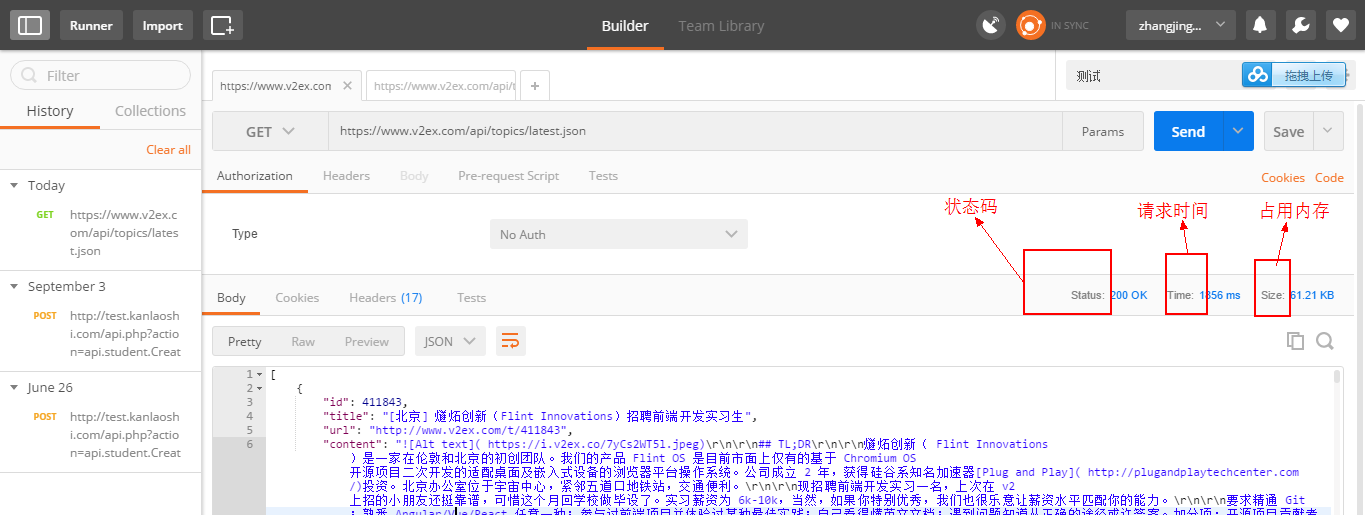


***查看返回结果：***

状态码：

请求时间：

占用内存大小：



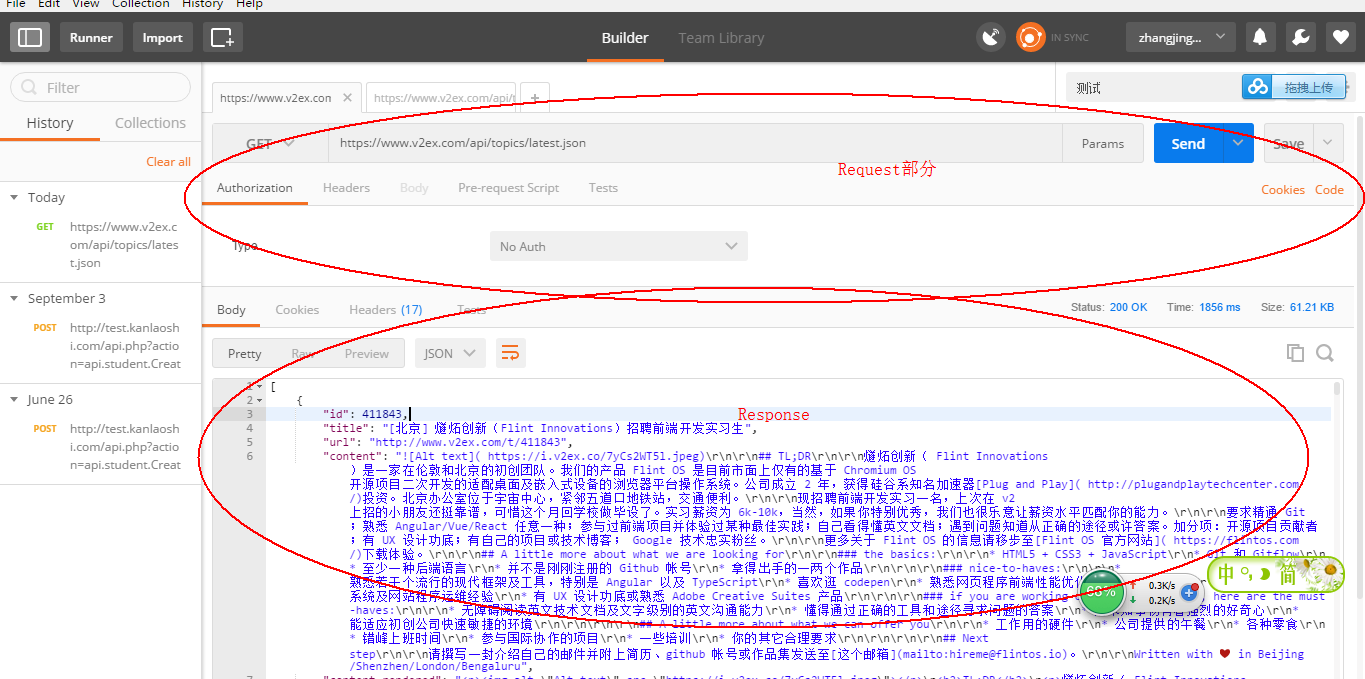
***postman界面详解：***

History：可以查看此前所有的请求历史

Collections：

request部分：方法+URL。由客户端发送用来触发一个服务器的操作

response部分：下方body部分。来自服务器的应答



1. 增加断言
2. 设置全局变量
3. 设置环境变量
4. 拿到并处理请求的响应
5. 定义测试检查点和断言

具体实现：

1. 填写请求方法&请求URL
2. 填写Tests(此处需要简单的js知识)

tests["状态码必须是 200"] = responseCode.code === 200;

//使用SNIPPETS直接添加状态码 的测试。此处意思为状态码必须是200

var res = JSON.parse(responseBody);

将responseBody部分使用json转换为全局变量

console.log(res.length);

//打印body的数组长度

tests["返回10条数据"] = res.length === 10;

//判断返回数组的长度

1. 点击发送
2. 查看response处的test结果

注意：如果不会js，可查看右侧的SNIPPETS

1. sandbox
2. 公共库
3. 环境变量与全局变量
4. 动态变量
5. 操作cookie
6. 读取数据文件
7. 查看请求与响应数据

上述断言可用下列语句替换：

pm.test('返回正确的状态码',function(){

pm.expect(pm.response).to.have.status(200);

});

1. postman进行UI测试
2. 发请求去拿到网站的响应--->html
3. 解析HTML标签，判断是够真确显示
4. request method
5. 如果是要往后台传送一些数据，则在Header里面，Content-Type

必须定义为“application/json”.

1. collection
2. 组织业务逻辑
3. 导入导出
4. 其他功能

postman使用总结

1. 进行接口测试，可以完成各种方式的接口测试，post、get、request、put、delete、head等。（各种接口方法的区别）
2. 传参的多样性。
3. get。直接使用url后面的param进行添加.
4. post。 Body下的form-data使用表单方式提交（需在Header下更改提交的content-type）
5. post。Body下的X-www-form-urlencoded进行提交。将表单转换为键值对
6. post。Body下 的raw原始数据进行添加。可以使用json方式进行提交。或者数组
7. post。Body下的binary进行添加。可以上传文件
8. 进行环境配置。设置测试环境和正式环境，毕竟混乱
9. collection。便于组织业务逻辑，方便输入输出。将全部接口的变量及参数设置好之后，可以在回归测试时直接run，查看全部结果。
10. 使用断言，查看输出结果是否符合规范
11. 查看返回结果的数据。1
12. 返回数据格式多样性。Pretty可以看到格式化后的JSON，也可选择XML,HTML,TEXT等格式，Raw就是未经处理的数据，Preview可以预览HTML页面
13. 可以查看返回的响应状态与时间（各种状态码对应）
14. test校验结果，查看瞬间查看所有通过或者失败的数量。
15. cookie.
16. 鉴权：有些api访问之前必须登录。
17. 可以使用tokenname
18. 或者basic auth
19. 