【预备知识】

1．Pytest入门知识

课程是在WebUI自动化课上讲的，可忽略WebUI部分的技术，只看Pytest

腾讯课堂视频链接：https://ke.qq.com/webcourse/index.html?r=1665735759526#cid=2960022&term_id=105892940&taid=13563029982358166&type=3072&source=PC_COURSE_DETAIL&v

小鹅通视频链接：https://ffe.xet.tech/s/1wcOlZ

2. Allure报告入门到实战

课程是在WebUI自动化课上讲的，可忽略WebUI部分的技术，只看Allure

腾讯课堂视频链接：https://ke.qq.com/webcourse/index.html?r=1665735808994#cid=2960022&term_id=105892940&taid=13563094406867606&type=3072&source=PC_COURSE_DETAIL&v

小鹅通视频链接：https://ffe.xet.tech/s/gKcRO

3. 代码和课件资料链接

登陆课堂派

通过 加课码 KE7DYQ ， 加入课程，并找到下面2节课，下载资料

视频或者资料找不到就联系班班获取

一、华测自动化电商平台项目背景介绍

1.访问和注册

地址：http://shop-xo.hctestedu.com/

注册：可以随便填写账号密码完成注册

1.1 数据库连接串

主机（host，注意没有www）：

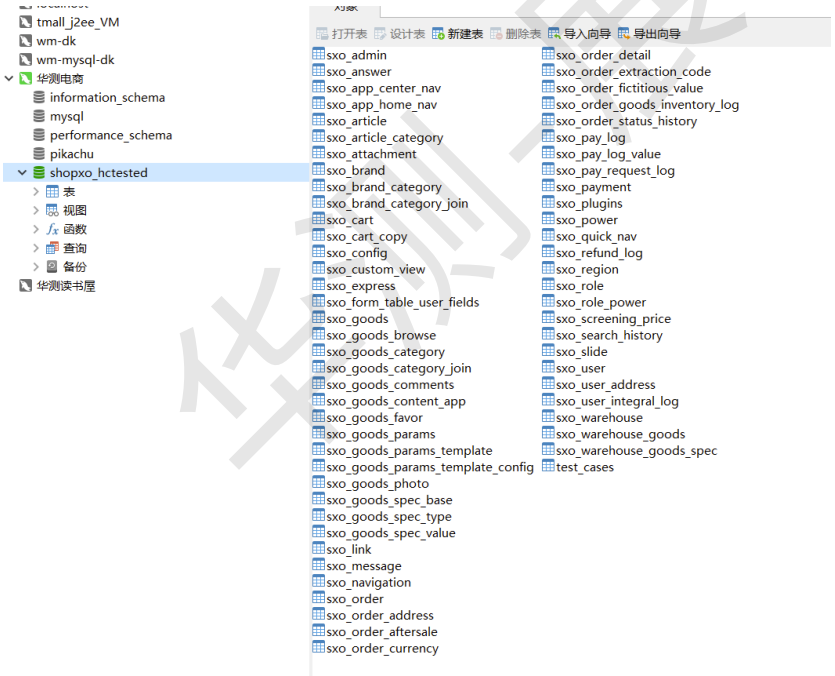
shop-xo.hctestedu.com

端口：3306

用户名：api_test

密码：Aa9999!

shopxo_hctested



2. 项目背景介绍

XX系统是一套基于ThinkPHP6框架研发的电商项目，由系统管理模块、用户端模块组成，包括商品管理、CMS、订单管理、支付管理、会员管理、后台管理等功能。

3. 项目接口及接口文档介绍

文档位置：

资料包/复习资料/商城接口API梳理 (改版byCF).pdf

账号直接在前台自行注册

测试环境前台地址：

http://shop-xo.hctestedu.com

请求URL示例（登陆）：

api/user/login

公共参数

参数名	是否必须	类型	默认值	描述
application	是	string	web	请求应用
application_client_type	是	string	pc	请求客户端
token	否	string		token

公共参数，通过url传递

```
params = {
  "application": "app",
  "application_client_type": "weixin",
  "token": token
}
```

请求应用详解 - application

参数值	描述
web	web 网页端
app	ios、android、小程序均为app

请求客户端详解 - application_client_type

参数值	描述
pc	web 网页端
h5	手机 H5端
ios	苹果手机 APP
android	安卓手机 APP
weixin	微信小程序
alipay	支付宝小程序
baidu	百度小程序
toutiao	头条/抖音小程序
qq	QQ 小程序

返回结构详解

正确判断状态使用 code 等于 0

正确状态下 data 并不一定存在数据

msg 仅业务描述、 code 等于负数情况下一般用于错误信息提示

参数值	描述
code	状态码、0正确、负数失败
msg	描述信息
data	返回数据

一个完整的请求示例(和文档中不一样)

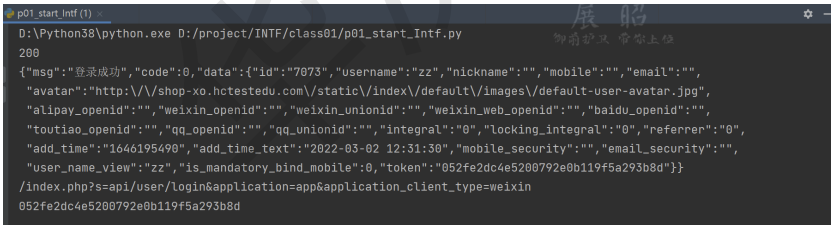
```
http://shop-xo.hctestedu.com/index.php?s=api/order/cancel&application=app&application_client_type=weixin&token=18b42b0b26ac17666638cfab43030482
```

Requests请求代码

```
# 01 DS_022 登陆-使用用户名能正确的登录用户
data = {
 "accounts": "zz",
 "pwd": "123456",
 "type": "username"
}
# app才会返回token
params = {
 "application": "app",
  "application_client_type": "weixin",
 }

r1 = requests.post(url="http://shop-xo.hctestedu.com/index.php?s=api/user/login", params=params, json=data)
print(r1.status_code)
print(r1.text)
print(r1.request.path_url)
token_list = jsonpath.jsonpath(r1.json(),'$..token')
token = token_list[0]
print(token)
```

响应报文：



4. Python接口测试急速入门

常规接口测试步骤：

1.接口URL：IP地址+端口+路径

2.接口请求参数：用户名 密码

3.接口请求方法：get post delete put...

4.接口响应报文

5. 项目接口用例一览

用例编号	接口名	用例标题	参数 Content-Type	请求参数	期望结果	实际结果	设计人	编写时间	分类

DS_004	商品详情	验证输入的商品ID不存在提示用户	x-www-form-urlencoded	goods_id=12	提示错误信息：
					{
					"msg": "商品不存在或已删除",
					"code": -1,
					}
									作业

DS_005	商品详情	输入删除的商品ID提示 用户	x-www-form-urlencoded	goods_id=15	提示错误信息：
					{
					"msg": "商品不存在或已删除",
					"code": -1,
					}
									参考

DS_010	用户注册	用户名为不超过7位，注册成功	x-www-form-urlencoded	accounts:yulisa001
				pwd:123456
				type:username
					能够正确的注册，数据库插入数据，并提示：
					{
					"msg": "注册成功",
					"code": 0,
					}
									作业

DS_013	用户注册	验证当type输入不存在的类型提示错误信息	x-www-form-urlencoded	accounts:yulisa002
				pwd:123456
				type:phone
					提示错误信息：
					{
					"msg": "注册类型有误",
					"code": -1,
					}
						通过			参考

DS_022	登录接口	使用用户名能正确的登录用户	x-www-form-urlencoded	accounts:yulisa001
				pwd:123456
				type:username
					能够正确的登录并且显示对应的用户信息：
					{
					}
									老师

DS_025	登录接口	验证输入错误的用户名提示用户	x-www-form-urlencoded	accounts:yulisa001wu
				pwd:123456
				type:username
					提示错误信息：
					{
					"msg": "登录帐号不存在",
					"code": -3,
					}
									作业

DS_026	登录接口	验证用户名为空提示用户	x-www-form-urlencoded	accounts:[为空]
				pwd:123456
				type:username
					提示错误信息：
					{
					"msg": "登录账号不能为空",
					"code": -1,
					}
						通过			参考

DS_033	加入购物车	验证无规格值可以加入到对应商品	application/json	{
				"goods_id":"10",
				"spec":"",
				"stock":1
				}
					能够正确添加购物车，数据库插入数据，并提示：
					{
					"msg": "加入成功",
					"code": 0,
					}
									老师

DS_055	删除购物车	能删除当前用户对应的购物车数据（单个）	x-www-form-urlencoded	id:4863	能正确的删除数据，并且从购物车列表表中移除，提示用户：
					{
					"msg": "删除成功",
					"code": 0,
					}
									参考

DS_056	删除购物车	能删除当前用户对应的购物车数据（多个）	x-www-form-urlencoded	id:4867,5029	能正确的删除数据，并且从购物车列表表中移除，提示用户：
					{
					"msg": "删除成功",
					"code": 0,
					}
									作业

DS_094	提交订单	【线下支付】验证能正确的下单个商品	application/json	{
				"buy_type":"cart",
				"address_id":1129,
				"ids":"5102",
				"payment_id":"3"
				}
					能正确的确认订单，并且未支付的状态，提示用户：
					{
					"msg": "提交成功",
					"code": 0,
					"data": {
					"order_status":1
									老师

绿色：核心用例，老师讲课用，每堂课挑几个
黄色：5个参考学习用例，
蓝色：5个作业练习用例
文档位置：
资料包/复习资料/商电商接口测试用例_BY_展昭_V1.0.xlsx
代码位置：
资料包/代码包

6. 为什么用Pytest而不用postman、jmeter、rf 去做自动化测试

Postman支持csv数据文件的导入，**但是每次执行时都需要手工加载数据文件，不方便**（所以只能做半自动化）。然后它要通过Newman实现批量执行和保存结果。实际上，Postman比较适合做手工接口测试，因为简单，可以实现半自动化，一般用来做接口测试，用来发现BUG，验证后台程序。

Jmeter是个压测框架，他的整个体系是为压测设计的，对接口测试的支持只是最基础的，没有更多的灵活性支持。而且结合junit testng的测试用例比较吃力，也无法使用allure2这样的报告框架，无法拥有完整的测试管理能力。可以用它做接口测试，但它不是一个完整的测试框架，**在设计用例体系和流程的时候，会受限于他的执行模型，无法方便的编写用例。比如接口封装复用、灵活的断言支持、用例的套件编排等。**

rf是一个完整的测试框架，测试管理是优点，生态也比较齐全，但是他的生态自成一体，会导致使用体验问题。**比如他有自己的IDE，使用体验不好但是又无法离开**，不如使用原生的IDE强大灵活。他的那些**库多数都是从其他库里二次封装再引入的，维护和质量都不会太好。**使用它也会导致测试工程师的定制和二次封装出现困难，很多成熟的框架比如allure2就比较难以引入使用。他的关键字驱动方式维护用例也是非常体验不好的，所以BAT互联网巨头基本都没采用。

直接使用**Pytest**的好处是**简单，开放，可以复用已有的各种生态，比如各种强大的智能IDE，各种框架和扩展，用的是python语言所以招人维护也比较容易。**二次封装也都比较简单。使用**原生框架与原生语言**可以获得**最大的灵活性**，方便未来的**定制。**

还有一点，测试工程师需要做app、web、接口或者端到端的自动化测试。这些测试用例如果用同一套测试框架管理是最好的。目前jmeter这种模式肯定是做不了除了接口之外的其他的测试的。

二、接口概念与通信原理

1.什么是接口

接口测试主要用于外部系统与系统之间以及内部各个子系统之间的交互点，定义特定的交互点，然后通过这些交互点来，通过一些特殊的规则就是协议，来进行数据之间的交互。

SOS 摩斯密码
三长两短 协议

救命　　数据
无线电发报机　　交互点

2.接口都有哪些类型?

接口一般分为两种：1.程序内部的接口 2.系统对外的接口
1）系统对外的接口：比如你要从别的网站或服务器上获取资源或信息，别人肯定不会把数据库共享给你，他只能给你提供一个他们写好的方法来获取数据，你引用他提供的接口就能使用他写好的方法，从而达到数据共享的目的。

NGBSS系统：CRM+CBS+ESB+PROVINC
系统和系统之间通过接口进行交互：createSubscriber CO025
crm.log

2）程序内部的接口：方法与方法之间，模块与模块之间的交互，程序内部抛出的接口，
比如：bbs系统，有登陆模块，有发帖模块等等，那你要发帖就必须先登陆，那么这两个模块就有交互，我们在测试的时候都用通过工具才能调试和测试。

主流接口介绍：
1.webservice接口 2.http 接口
1）webservice接口是走soap协议通过http传输，请求报文和返回报文都是xml格式的，我们在测试的时候都要通过工具才能进行调用、测试（SOAPUI）
2）http接口是走http协议，通过路径来区分调用的方法，请求报文都是key-value形式的，返回报文一般都是json串，有get和post等方法，这也是最常用的两种请求方式。
json是一种通用的数据类型，所有的语言都认识它。（json的本质是字符串，他与其他语言无关，只是可以经过稍稍加工可以转换成其他语言的数据类型，比如可以转换成python中的字典，key-value的形式，可以转换成javaScript中的原生对象，可以转换成java中的类对象等。）

3.接口的本质及其工作原理是什么?

接口你可以简单的理解他就是URL，工作原理就是URL通过get或者post请求像服务器发送一些东西，然后得到一些相应的返回值。
本质就是数据的传输与接收。

4.什么是接口测试?

接口测试是测试系统组件间接口的一种测试。接口测试主要用于检测外部系统与系统之间以及内部各个子系统之间的交互点。测试的重点是要检查数据的交换、传递和控制管理的过程，以及系统间相互逻辑依赖关系等。

简单的说就是通过URL向服务器或者其他模块等，传输我们想传输的数据，然后看看他们返回的是不是我们预期想要的。

5.为什么要做接口测试

1）越底层发现bug，它的修复成本是越低的。
2）前端随便变，接口测好了，后端不用变。前后端是两拨人开发的。
3）检查系统的安全性、稳定性，前端传参不可信，比如京东购物，前端价格是不可能传入-1元，但是通过接口可以传入-1元。
4）如今的系统复杂度不断上升，传统的测试方法成本急剧增加且测试效率大幅下降，接口测试可以提供这种情况下的解决方案。
5）接口测试相对容易实现自动化持续集成，且相对UI自动化也比较稳定，可以减少人工回归测试人力成本与时间，缩短测试周期，支持后端快速发版需求。接口持续集成是为什么能低成本高收益的根源。

举例：马来DiGi 局点，几千个接口，开发每次发布版本总会有大量缺陷未解决，或者解决问题导致新的问题。通过人力，非常耗时。引入自动化，几千秒出结果，1个人执行就可以了。crm.log

6）现在很多系统前后端架构师分离的，从安全层面来说：
　　a.只依赖前端进行限制已经完全不能满足系统的安全要求（绕过前端实在太容易），需要后端同样进行控制，在这种情况下就需要从接口层面进行验证。
　　b.前后端传输、日志打印等信息是否加密传输也是需要验证的，特别是设计到用户的隐式信息，如身份证，银行卡等。

6.怎么做接口测试

由于我们项目前后端调用主要是基于http协议的接口，所以测试接口时主要是通过工具或代码模拟http请求的发送与接收。工具有很多如：postman、jmeter、soapUI、java+httpclient、python+requests、robotframework+httplibrary等

也可以用接口自动化来实现，一般来讲通过代码实现，框架和UI自动化差不多，发送请求用断言来判断。

7.通信原理

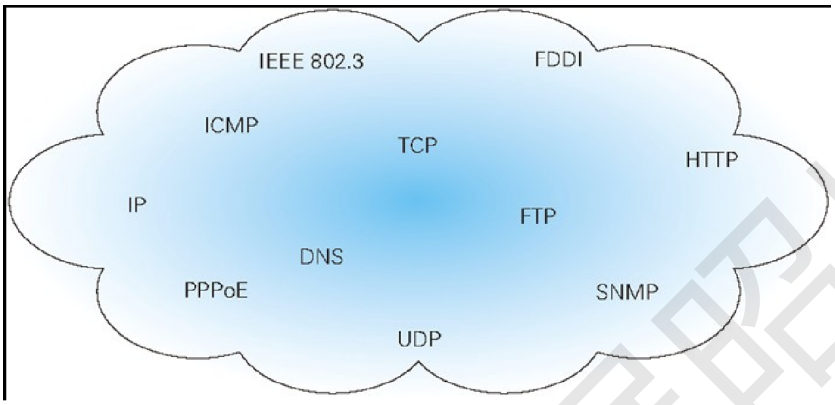
7.1 网络基础TCP/IP

通常使用的网络（包括互联网）是在TCP/IP协议族的基础上运作的。而HTTP属于它内部的一个子集。

7.1.1 TCP/IP 协议族

不同的硬件、操作系统之间的通信，所有的这一切都需要一种规则。而我们就把这种规则成为协议（protocol）

像这样把与互联网相关联的协议集合起来总称为 TCP/IP。也有说法认为，TCP/IP 是指 TCP 和 IP 这两种协议。还有一种说法认为，TCP/IP 是在 IP 协议的通信过程中，使用到的协议族的统称。



TCP/IP 是互联网相关的各类协议族的总称

7.1.2 TCP/IP 的分层管理

OSI七层模型	TCP/IP概念层模型	功能	TCP/IP协议族
应用层	应用层	文件传输，电子邮件，文件服务，虚拟终端	TFTP，HTTP，SNMP，FTP，SMTP，DNS，Telnet
表示层		数据格式化，代码转换，数据加密	没有协议
会话层		解除或建立与别的接点的联系	没有协议
传输层	传输层	提供端对端的接口	TCP，UDP
网络层	网络层	为数据包选择路由	IP，ICMP，RIP，OSPF，BGP，IGMP
数据链路层	链路层	传输有地址的帧以及错误检测功能	SLIP，CSLIP，PPP，ARP，RARP，MTU
物理层		以二进制数据形式在物理媒体上传输数据	ISO2110，IEEE802，IEEE802.2

TCP/ IP 协议族 按层次分别分为以下 4层： 应用层、传输层、网络层和数据链路层。

TCP/IP协议族各层的作用如下。

应用层

应用层决定了向用户提供应用服务时的通信的活动。

TCP/IP 协议族内预存了各类通用的应用服务。比如，FTP（File Transfer Protocol，文件传输协议）和DNS（Domain Name System，域名系统）服务就是其中两类。

HTTP 协议也处于该层。

传输层

传输层对上层应用层，提供处于网络链接中的两台计算机之间的数据传输。

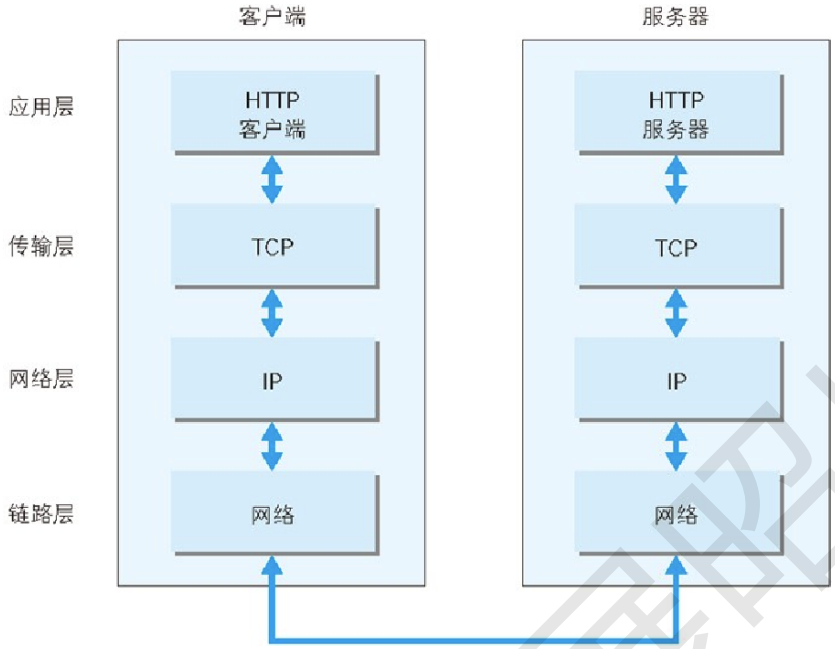
在传输层有两个性质不同的协议：TCP（Transmission Control Protocol，传输控制协议）和 UDP（User Data Protocol，用户数据报协议）。

网络层（又叫网络互连层）

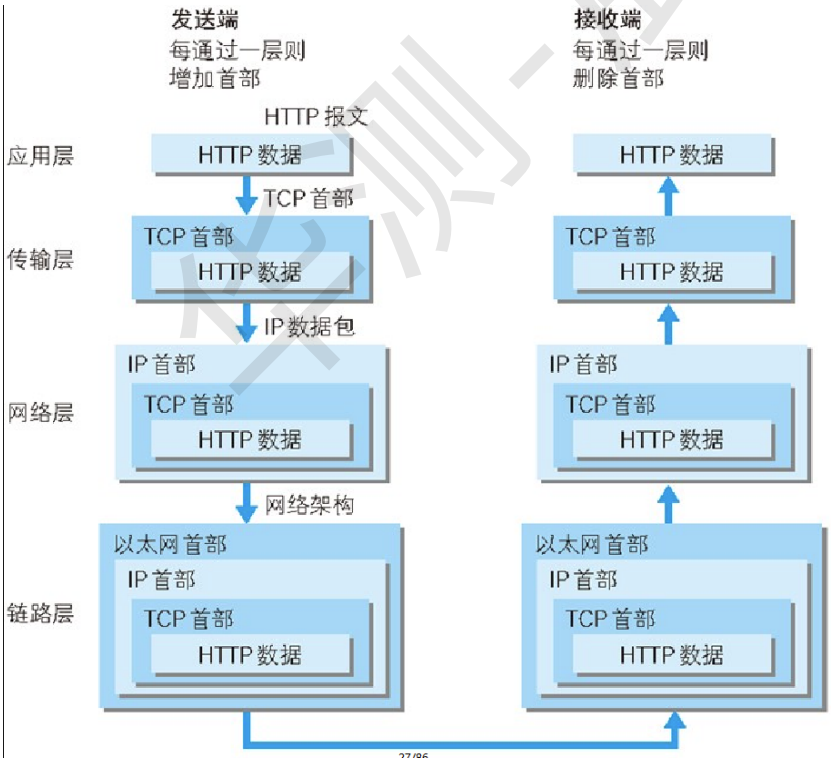
网络层用来处理在网络上流动的数据包。数据包是网络传输的最小数据单位。该层规定了通过怎样的路径（所谓的传输路线）到达对方计算机，并把数据包传送给对方。
与对方计算机之间通过多台计算机或网络设备进行传输时，网络层所起的作用就是在众多的选项内选择一条传输路线。

链路层（又名数据联络层，网络接口层）
用来处理连接网络的硬件部分。包括控制操作系统、硬件的设备驱动、NIC（Network Interface Card，网络适配器，即网卡），及光纤等物理可见部分（还包括连接器等一切传输媒介）。硬件上的范畴均在俩路层的作用范围之内。

7.1.3 TCP/IP 通信传输流



利用 TCP/IP 协议族进行网络通信时，会通过分层顺序与对方进行通信。发送端从应用层往下走，接收端则从网络层往应用层走。



我们用 HTTP 举例来说明，首先作为发送端的客户端在应用层（HTTP 协议）发出一个想看某个 Web 页面的 HTTP 请求。

接着，为了传输方便，在传输层（TCP 协议）把从应用层处收到的数据（HTTP 请求报文）进行分割，并在各个报文上打上标记序号及端口号后转发给网络层。
在网络层（IP 协议），增加作为通信目的地的 MAC 地址后转发给链路层。这样一来，发往网络的通信请求就准备齐全了。
接收端的服务器在链路层接收到数据，按序往上层发送，一直到应用层。当传输到应用层，才能算真正接收到由客户端发送过来的 HTTP 请求。

发送端在层与层之间传输数据时，每经过一层时必定会被打上一个该层所属的首部信息。反之，接收端在层与层传输数据时，每经过一层时会把对应的首部消去。
这种把数据信息包装起来的做法称为封装（encapsulate）。

三、HTTP网络协议详解

1.URL和URI

URI（统一资源标识符）
URL（Uniform Resource Locator，统一资源定位符）网页地址：http://www.baidu.com

URI 用字符串标识某一互联网资源，而 URL 表示资源的地点（互联网上所处的位置）。可见 URL 是 URI 的子集。

“RFC3986：统一资源标识符（URI）通用语法”中列举了几种 URI 例子，
ftp://ftp.is.co.za/rfc/rfc1808.txt
http://www.ietf.org/rfc/rfc2396.txt
ldap://[2001:db8::7]/c=GB?objectClass?one
mailto:John.Doe@example.com
news:comp.infosystems.www.servers.unix
tel:+1-816-555-1212
telnet://192.0.2.16:80/
urn:oasis:names:specification:docbook:dtd:xml:4.1.2

RFC：有一些用来制定 HTTP 协议技术标准的文档，它们被称为 RFC（Request for Comments，征求修正意见书）。

2.URL详解

百度搜索的一个url地址：
https://www.baidu.com/s?wd=%E4%B8%8A%E6%B5%B7%E6%82%A0%E6%82%A0%E5%8D%9A%E5%AE%A2&rsv_spt=1&rsv_iqid=0x91baaabd00070ba2

1）http/https:　　协议类型
2）host：主机地址或域名
--192.168.x.xx:8080　地址+端口号
--www.xxx.com　　域名
--localhost:8080　　localhost是本机地址127.0.0.1

3）port:端口号　　（默认端口是80可以省略）
4）path:　　请求的路径（host之后，问号？之前）
5）？：　　问号是分割符号
6）参数:　　name=value
7）&　：　　多个参数用&隔开

3.请求方法

根据HTTP标准，HTTP请求可以使用多种请求方法。
HTTP1.0定义了三种请求方法： GET, POST 和 HEAD方法。
HTTP1.1新增了五种请求方法：OPTIONS, PUT，DELETE, TRACE 和 CONNECT 方法。

GET 请求指定的页面信息，并返回实体主体。
HEAD 类似于get请求，只不过返回的响应中没有具体的内容，用于获取报头
POST 向指定资源提交数据进行处理请求（例如提交表单或者上传文件）。数据被包含在请求体中。POST请求可能会导致新的资源的建立和/或已有资源的修改。
PUT 从客户端向服务器传送的数据取代指定的文档的内容。
DELETE 请求服务器删除指定的页面。
CONNECT HTTP/1.1协议中预留给能够将连接改为管道方式的代理服务器。
OPTIONS 允许客户端查看服务器的性能。
TRACE 回显服务器收到的请求，主要用于测试或诊断。

3.1 get 和 post方法的区别：

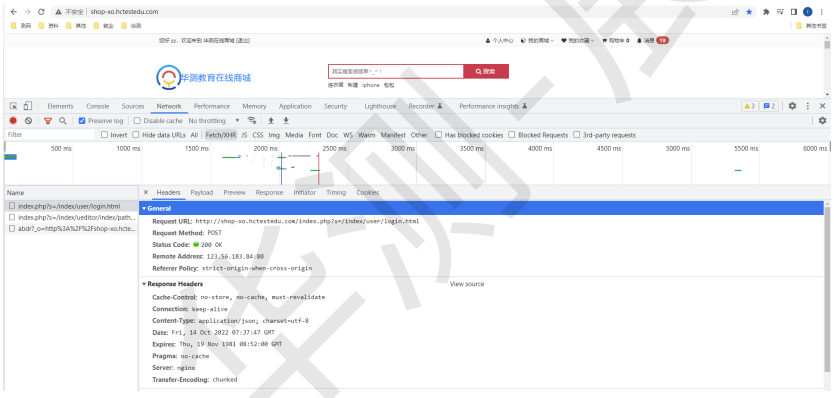
GET和POST的区别主要表现在如下方面。
（1）GET提交的数据会放在URL之后，以问号（？）分割URL和传输数据，参数之间以&相连，如EditPosts.aspx?name=test1&id=123456。POST方法是把提交的数据放在HTTP包的Body中。
（2）GET提交的数据大小有限制（因为浏览器对URL的长度有限制），而POST方法提交的数据大小没有限制。
（3）GET 方式需要使用 Request.QueryString 来取得变量的值，而 POST 方法通过Request.Form来获取变量的值。
（4）GET 方式提交数据会带来安全问题，比如一个登录页面通过 GET 方式提交数据时，用户名和密码将出现在 URL 上，如果页面可以被缓存或者其他人可以访问这台机器，就可以从历史记录获得该用户的账号和密码。

4. HTTP协议Header介绍

HTTP请求和HTTP响应中有很多Header，HTTP请求方法和HTTP Header配合工作，共同决定客户端和服务器能做什么事情。我们需要掌握每一个Header的用法。
Header翻译成中文，叫“首部”或者“头域”。为了避免混乱，我们直接称之为Header。

4.1 HTTP Header介绍

HTTP请求中有Header，HTTP响应中也有Header。
Header的语法格式是“key：value”，一行一个Header。每一个Header都有特殊的作用，在谷歌浏览器开发者工具中可以查看完整的Header，如下图所示（登陆后抓包报文）：



4.2 Cache相关的Header

HTTP请求和HTTP响应中都有很多用于缓存的Header。
HTTP缓存是指当Web请求抵达缓存时，如果本地有“已缓存的”副本，就可以从本地存储设备而不是从原始服务器中提取这个文档。
本书第6章会专门讲缓存，并对Cache相关的Header进行详细讲解。

4.3 Cookies

Cookie是一种HTTP Header，是HTTP中非常重要的内容。它由key=value的形式组成，比如ip_country=CN。
浏览器把Cookie通过HTTP请求中的“Cookie: header”发送给Web服务器，Web服务器通过HTTP响应中的“Set-Cookie: header”把Cookie发送给浏览器。

4.4 Accept

Accept表示浏览器客户端可以接受的媒体类型。
例如，Accept: text/html代表浏览器可以接受服务器返回html，也就是我们通常说的html文档。
通配符代表任意类型，例如Accept: text/html,/*;q=0.8代表浏览器可以处理所有的类型。一般浏览器客户端给Web服务器发送的都是这个。

4.5 Accept-Encoding

Accept-Encoding跟压缩有关，浏览器发送HTTP请求给Web服务器，HTTP请求中的Header有Accept-Encoding: gzip, deflate（告诉服务器，浏览器支持gzip压缩）。

4.6 Accept-Language

Accept-Language的作用是浏览器声明自己接受的语言。
语言跟字符集的区别在于：中文是语言，中文有多种字符集，比如big5、gb2313、gbk等。示例如下：
Accept-Language: en-US,en;q=0.8,zh-CN;q=0.6,zh;q=0.4,zh-TW;q=0.2

4.7 User-Agent

User-Agent的作用是浏览器用来告诉服务器，客户端使用的操作系统及版本、CPU类型、浏览器及版本、浏览器渲染引擎、浏览器语言、浏览器插件等。

```
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/105.0.0.0
Safari/537.36
```

这个代表客户端用的是64位win10系统，Chrome是105.0.0.0版本。
假如用手机的APP访问网站，APP中的HTTP请求会包含如下的User-Agent：

```
User-Agent:Dalvik/2.1.0 (Linux; U; Android 6.0; Redmi Note 4 MIUI/ V8.5.2.0.MBFCNED)
```

这个User-Agent表示客户端用的是红米Note 4，Android 6.0版本。
如果我们想模拟各种不同的客户端，只需要修改User-Agent，就可以伪装成各种客户端。

4.8 Referer

HTTP协议头中的Referer主要用来让服务器判断来源页面，即用户是从哪个页面来的。网站通常用其来统计用户来源，看用户是从搜索页面来的，还是从其他网站链接过来的，或是从书签等访问的，以便合理定位网站。
Referer有时也被用作防盗链，即下载时判断来源地址是不是在网站域名之内，否则就不能下载或显示。很多网站，如天涯就是通过Referer页面来判断用户是否能够下载图片的。

4.9 Connection

从HTTP/1.1起，系统默认都开启了Connection:Keep-Alive，保持连接特性。
HTTP协议是基于TCP协议的。当一个网页完全打开后，客户端和服务器之间用于传输HTTP数据的TCP连接不会关闭；如果客户端再次访问这个服务器上的网页，将会继续使用这一条已经建立的连接。
Keep-Alive不会永久保持连接，它有一个保持时间，可以在不同的服务器软件（如Apache）中设定这个时间。

4.10 Host

Host这个Header是必需的，它的作用是指定被请求的主机和端口号，它通常从HTTP URL中提取出来。
实例：我们在浏览器中输入https://www.cnblogs.com/tankxiao/，浏览器发送的HTTP请求中就会包含Host的Header，例如Host: www.cnblogs.com。此处使用了默认端口80。
如果指定了端口号，例如我们在浏览器中输入http://tankapi.vicp.io:15375/，则Header变为Host: tankapi.vicp.io:15375。

5.状态码

状态代码有三位数字组成，第一个数字定义了响应的类别，共分五种类别:
1xx：指示信息--表示请求已接收，继续处理
2xx：成功--表示请求已被成功接收、理解、接受
3xx：重定向--要完成请求必须进行更进一步的操作

4xx：客户端错误--请求有语法错误或请求无法实现
5xx：服务器端错误--服务器未能实现合法的请求

常见状态码：
200 OK//客户端请求成功
400 Bad Request//客户端请求有语法错误，不能被服务器所理解
401 Unauthorized//请求未经授权，这个状态代码必须和WWW-Authenticate报头域一起使用
403 Forbidden//服务器收到请求，但是拒绝提供服务
404 Not Found//请求资源不存在，eg：输入了错误的URL
500 Internal Server Error//服务器发生不可预期的错误
503 Server Unavailable//服务器当前不能处理客户端的请求，一段时间后可能恢复正常

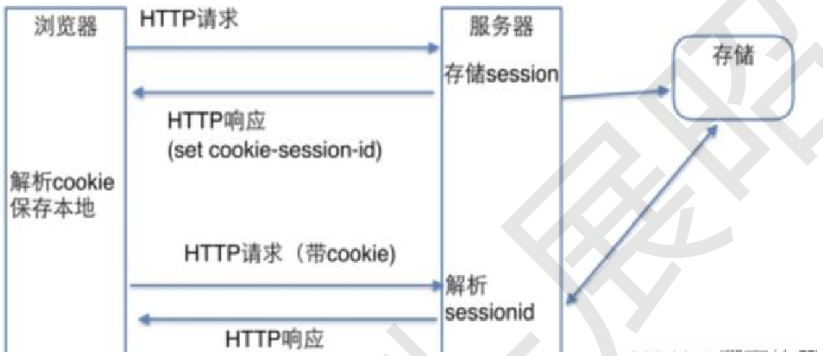
四、接口鉴权机制详解 （重点）

1 Cookie鉴权机制

最开始，使用Cookie单独实现用户的鉴权，userid分别保存在客户端的Cookie，以及服务器的数据库里。服务器数据库里的userid是固定的，这样一旦客户端的Cookie信息泄露，就可以伪装成用户进行操作，很不安全。

2 Cookie+Session鉴权机制

为了解决安全性的问题，将Session和Cookie搭配在一起使用。



3 Token发展史及其鉴权机制

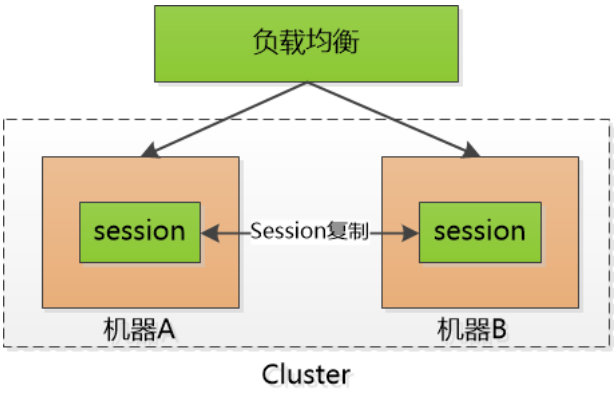
1. 很久很久以前，Web 基本上就是文档的浏览而已， 既然是浏览，作为服务器， 不需要记录谁在某一段时间里都浏览了什么文档，每次请求都是一个新的HTTP协议， 就是请求加响应， 尤其是服务器不用记住是谁刚刚发了HTTP请求， 每个请求对服务器来说都是全新的。这段时间很嗨皮

2. 但是随着交互式Web应用的兴起，像在线购物网站，需要登录的网站等等，马上就面临一个问题，那就是要管理会话，必须记住哪些人登录系统， 哪些人往自己的购物车中放商品， 也就是说服务器必须把每个人区分开，这就是一个不小的挑战，因为HTTP请求是无状态的，所以想出的办法就是给大家发一个会话标识(session id), 说白了就是一个随机的字串，每个人收到的都不一样， 每次大家向服务器发起HTTP请求的时候，把这个字符串给一并捎过来， 这样服务器就能区分开谁是谁了

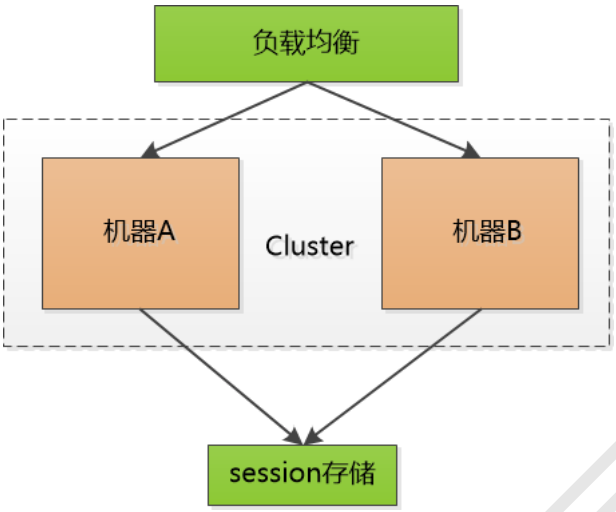
3. 这样大家很嗨皮了，可是服务器就不嗨皮了，每个人只需要保存自己的session id，而服务器要保存所有人的session id！ 如果访问服务器多了， 就得由成千上万，甚至几十万个。这对服务器说是一个巨大的开销， 严重的限制了服务器扩展能力， 比如说服务器用两个机器组成了一个集群， 小F通过机器A登录了系统， 那session id会保存在机器A上， 假设小F的下一次请求被转发到机器B怎么办？ 机器B可没有小F的 session id啊。

有时候会采用一点小伎俩： session sticky， 就是让小F的请求一直粘连在机器A上， 但是这也不管用， 要是机器A挂掉了， 还得转到机器B去。

那只好做session 的复制了， 把session id 在两个机器之间搬来搬去， 快累死了。



后来有个叫Memcached的支了招： 把session id 集中存储到一个地方， 所有的机器都来访问这个地方的数据， 这样一来，就不用复制了， 但是增加了单点失败的可能性， 要是那个负责session 的机器挂了， 所有人都得重新登录一遍， 估计得被人骂死。



也尝试把这个单点的机器也搞出集群，增加可靠性， 但不管如何， 这小小的session 对服务器来说是一个沉重的负担

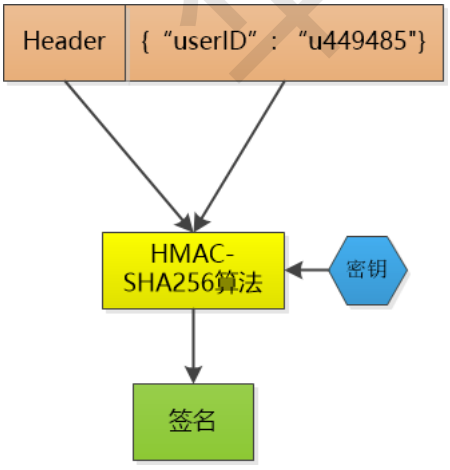
4.于是有人就一直在思考， 服务器为什么要保存这可恶的session呢， 只让每个客户端去保存该多好？可是如果不保存这些session id，怎么验证客户端发给服务器的session id 的确是服务器生成的呢？ 如果不去验证，我们都不知道他们是不是合法登录的用户，那些不怀好意的家伙们就可以伪造session id，为所欲为了。

嗯，对了，关键点就是验证！

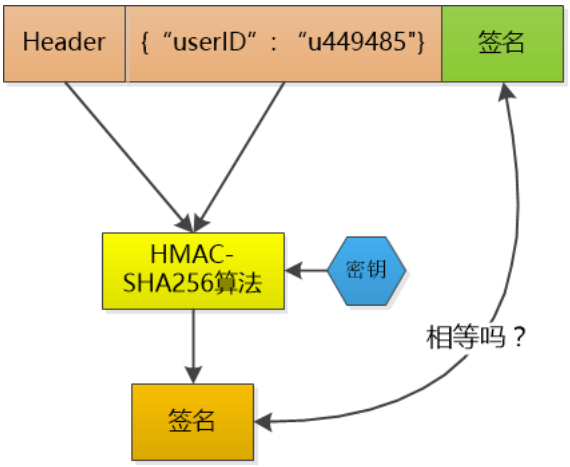
比如说， 小F已经登录了系统， 服务器给他发一个令牌(token)， 里边包含了小F的 user id， 下一次小F 再次通过Http 请求访问服务器的时候， 把这个token 通过Http header 带过来不就可以了。

不过这和session id没有本质区别啊， 任何人都可以可以伪造， 所以服务器得想点儿办法， 让别人伪造不了。

那就对数据做一个签名吧， 比如说服务器用HMAC-SHA256 算法，加上一个只有服务器才知道的密钥， 对数据做一个签名， 把这个签名和数据一起作为token， 由于密钥别人不知道， 就无法伪造token了。



这个token 服务器不保存， 当小F把这个token 给服务器发过来的时候，服务器再用同样的HMAC-SHA256 算法和同样的密钥，对数据再计算一次签名， 和token 中的签名做个比较， 如果相同， 服务器就知道小F已经登录过了，并且可以直接取到小F的user id，如果不相同， 数据部分肯定被人篡改过， 服务器就告诉发送者： 对不起，没有认证。



Token 中的数据是明文保存的（虽然服务器会用Base64做下编码， 但那不是加密）， 还是可以被别人看到的， 所以服务器不能在其中保存像密码这样的敏感信息。

当然， 如果一个人的token 被别人偷走了， 那服务器也没办法， 服务器也会认为小偷就是合法用户， 这其实和一个人的session id被别人偷走是一样的。

这样一来， 服务器就不保存session id 了， 服务器只是生成token , 然后验证token ， 服务器用服务器的CPU计算时间获取了服务器的session 存储空间 ！

解除了session id这个负担， 可以说是无事一身轻， 服务器的机器集群现在可以轻松地做水平扩展， 用户访问量增大， 直接加机器就行。 这种无状态的感觉实在是太好了！

4 Cookie 、Session、Token的概念

Cookie

cookie 是一个非常具体的东西，指的就是浏览器里面能永久存储的一种数据，仅仅是浏览器实现的一种数据存储功能。

cookie由服务器生成，发送给浏览器，浏览器把cookie以 K-V 形式保存到某个目录下的文本文件内，下一次请求同一网站时会把该cookie发送给服务器。由于cookie是存在客户端上的，所以浏览器加入了一些限制确保cookie不会被恶意使用，同时不会占据太多磁盘空间，所以每个域的cookie数量是有限的。

Session

session 从字面上讲，就是会话。这个就类似于你和一个人交谈，你怎么知道当前和你交谈的是张三而不是李四呢？对方肯定有某种特征（长相等）表明他就是张三。

session 也是类似的道理，服务器要知道当前发请求给自己的是谁。为了做这种区分，服务器就要给每个客户端分配不同的“身份标识(Session Id)”，然后客户端每次向服务器发请求的时候，都带上这个“身份标识(Session Id)”，服务器就知道这个请求来自于谁了。至于客户端怎么保存这个“身份标识(Session Id)”，可以有很多种方式，对于浏览器客户端，大家都默认采用 cookie 的方式。

服务器使用session把用户的信息临时保存在了服务器上，用户离开网站后session会被销毁。这种用户信息存储方式相对cookie来说更安全，可是session有一个缺陷：如果web服务器做了负载均衡，那么下一个操作请求到了另一台服务器的时候session会丢失。

cookie和session的区别

session是存储服务器端，cookie是存储在客户端，所以session的安全性比cookie高。

获取session里的信息是通过存放在会话cookie里的session id获取的。而session是存放在服务器的内存中里，所以session里的数据不断增加会造成服务器的负担，所以会把很重要的信息存储在session中，而把一些次要东西存储在客户端的cookie里。

cookie确切的说分为两大类：会话cookie和持久化cookie。

会话cookie是存放在客户端浏览器的内存中，他的生命周期和浏览器是一致的，当浏览器关闭会话cookie也就消失了

持久化cookie是存放在客户端硬盘中，持久化cookie的生命周期是我们在设置cookie时候设置的那个保存时间

session的信息是通过sessionid获取的，而sessionid是存放在会话cookie当中的，当浏览器关闭的时候会话cookie消失，所以sessionid也就消失了，但是session的信息还存在服务器端，只是查不到所谓的session但它并不是不存在。

session在服务器关闭的时候，或者是session过期，又或者调用了invalidate()，再或者是session中的某一条数据消失调用session.removeAttribute()方法，才会消失。

session通过调用session.getsession来创建的。

Token
Token俗称为“令牌”，它的构成是：

uid：用户唯一身份标识

timestamp：当前时间戳

sign：签名字符串，防止第三方伪造数据；签名密钥是存储在服务器端的，其它人无法知道

在Web领域基于Token的身份验证随处可见。在大多数使用Web API的互联网公司中，tokens 是多用户下处理认证的最佳方式。
以下几点特性会让你在程序中使用基于Token的身份验证

1. 无状态、可扩展
2. 支持移动设备
3. 跨程序调用
4. 安全

那些使用基于Token的身份验证的大佬们
大部分你见到过的API和Web应用都使用tokens。例如Facebook, Twitter, Google+, GitHub等。

五、不同种类的接口文档理解

有没有听说过以前还有人用过 word 来做接口文档，但在前后端分离推行的早期，确实没有那么多趁手好用的接口文档工具。
互联网发展到现在，接口文档也经历了从简单的word到html，到swagger等逐步进化的方式，变得越来越美观，越来越规范，也支持越来越多的功能辅助调试。
接下来给大家盘点一下这些年接口文档的进化历程。
1 接口文档的幼年体：word

一开始是前后端分离，后端得告知前端接口的各项信息，方便前端调用。那需要提供的信息也就那些，就用word 写吧，于是就有了这样的接口文档：

问题似乎解决了，但项目嘛，是不停修改和迭代的，这样会导致：

这份接口文档是随着项目进行频繁改动的。每更新一次，就需要给项目成员分发一次新的接口文档

于是：

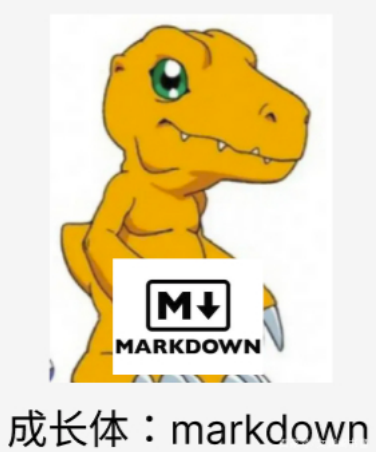
A.每改动一次就要新建一份，复制给项目组里很多人，这样一个文档复制来复制去，项目组里这么多人，谁都不知道拿到的是不是全新版本；

B.改一点就要生成一个新文档，于是文件夹里的接口文档可能就是这种形式：

是谁哭了我不说。

这些痛点促成了接口文档的第一次进化：从word版的接口文档进化成网页版的接口文档。

2 接口文档的成长体：Markdown



网页版的接口文档多完美，只需要分发一个链接给项目成员保存起来。

这样，如果后端修改了接口，直接在网页里修改，就保证大家看到的都是最新版本的，也不用每次一有改动就发一份新文档给大家。

这样一个由markdown生成的静态的html 页面，一个接口的关键要素全都有

文档简介

获取用户公开信息的接口文档

GetUserInfo

scope: user_info 需要用户授权

描述：获取用户的[redacted]公开资料，包括头像、昵称、性别和地区

- Domain : https://[redacted].com
- Method : GET
- URL : /openapi/user_info
- Param :

Name	Type	Must	Description
app_id	string	yes	应用唯一标识

问题解决！

但——新的问题又产生了。

这个接口文档是能用了，但又没那么好用，比如说：

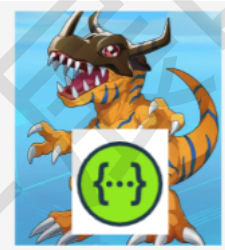
1.写接口挺麻烦的，完全纯手工写，没有任何辅助工具，非常花时间

2.接口写完还不能立刻看到生成的接口文档的效果，写错了还要重新回去调

3.没有接口规范约束，接口文档怎么写，哪些参数要写，哪些不写，呈现形式怎么样全凭开发人员本身的业务水准。

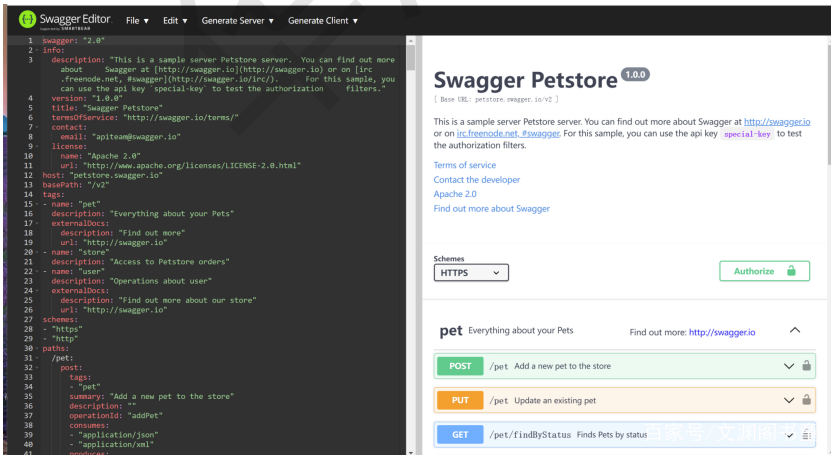
于是就开始了新的一轮的进化——有人开发了一个工具，专门就用来写接口文档。

3 接口文档的完全体：swagger



完全体：swagger

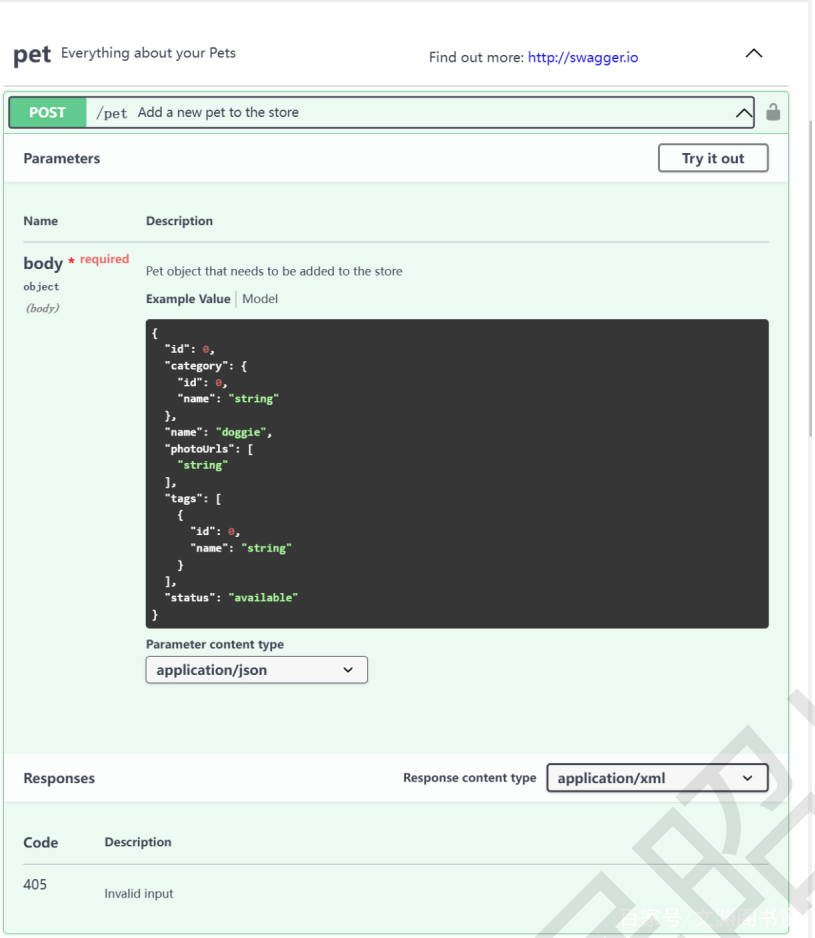
怎么写？在swagger editor里编写符合swagger 语法的接口文档，来生成接口文档，编写完的接口文档可以在swagger editor的右侧实时预览：



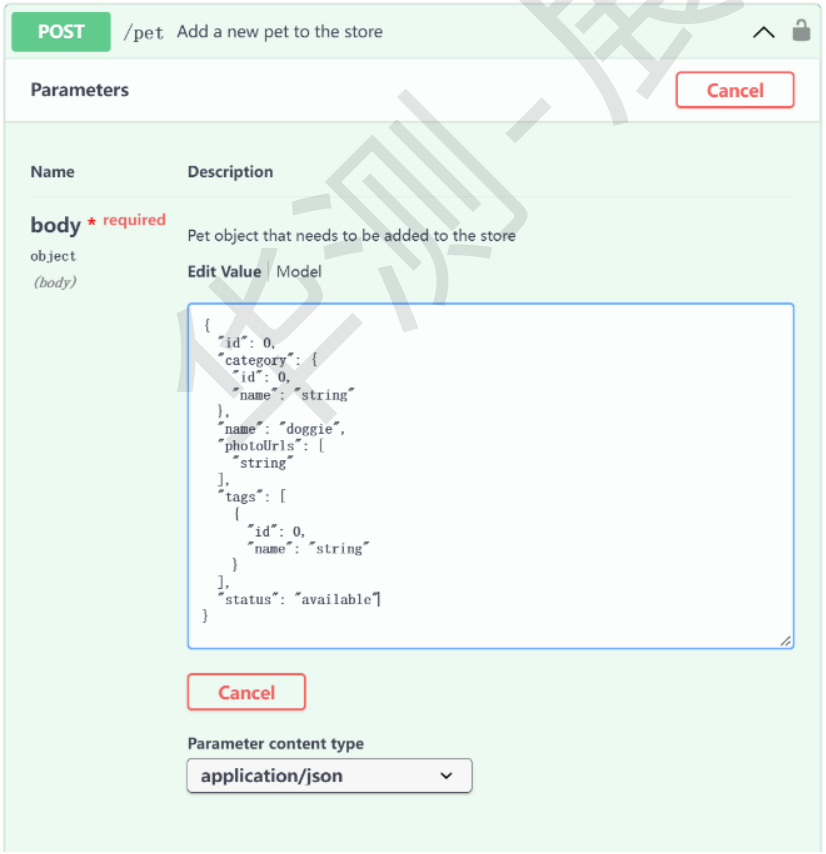
于是，进化到这个完全体阶段的接口文档工具已经实现了如下功能：

1.网页版接口文档支持的在线查看功能，当然他也有，而且这个接口文档的样式是符合open api3.0规范的，如果写得不符合语法，swagger editor 还会报错来纠正你。

一个标准的接口所应具备的信息：接口方法、接口路径、请求和响应参数，都能按照固定的格式呈现出来。



2.它还具备了初步、简单的调试功能，就是接口请求参数为空格，填写参数、发送请求，就能返回响应参数



六、作业

本次作业提交方式为：截图

1.简述cookie、session、token的异同点
2.在本地跑通老师的登陆接口代码

代码写完截图发到课堂派
作业是为了检查大家的学习情况
碰到任何问题，可以微信群里/私聊老师沟通！

要求：尽量下节课上课当天的16点之前提交，这样老师会有比较充足的时间仔细评审