**接口测试**

接口测试：又称API测试，测试系统组件间接口，属于灰盒测试。通过脚本或者工具，模拟客户端对服务端接口进行调用。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，传递和控制管理过程， 以及系统间的相互逻辑依赖关系等。从接口层测试能更早的发现问题，从而提高测试效率，降低修复成本。

http/https协议学习,常见请求方法GET/Post,cookie和session学习,接口的基本概念,接口文档,接口测试用例的编写,接口测试工具postman,jmeter,SoupUI.

**分类：**模块之间的接口：如果订单支付时，需要去用户资金模块查询余额等。系统对外部系统提供的接口,内部子系统的服务之间的接口.

**场景：**web网站的API接口：如微信接口、新浪微博接口等。公司内部系统API：订单接口需要调用用户资金接口查询余额。APP，客户端软件等。

**流程：**分析接口文档和需求文档，编写接口测试计划与测试用例，接口测试执行，输出接口测试报告。

**工具：**postman、jmeter、soupUI、java+httpclient、robotframework+httplibrary等。

**接口测试与自动化测试的区别？**答:自动化测试如app自动化测试、web自动化测试：都是模拟人类的行为去测试，底层都是通过接口去和服务器进行交互,接口测试可以在底层模拟人类的行为去进行测试。

**接口测试与性能测试？**答: 自动化测试的模拟行为，测试效率较慢，接口测试可以直接和服务器进行快速交互，对接口进行性能、压力测试。

**接口测试的价值？**提升测试效率：底层的一个bug能够引发上层的8个左右bug，而且底层的bug很容易引起全网的宕机，接口测试能够提供系统复杂度上升情况下的低成本高效率的解决方案。能快速定位bug.

**接口文档:**

|  |  |
| --- | --- |
| 功能描述 | 处理查询某一时间段的文件请求 |
| 服务协议 | Http+Json |
| 服务地址 | http;// |
| 服务类型 | POST |

**接口测试用例：**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 编号 | 模块 | 测试点 | url | Header | Method | Body | 预期结果 | 实际结果 |
| Dll\_01 | 登录 | 正常登录 | <http://ip:port:api/login> | Content-Type=application/json | Post | Data:{“userName”:小明} | 1. 返回字段正确 2. 返回的字段内容正确 | Pass |
| Dll\_02 | 登录 | 登录名为空 | <http://ip:port:api/login> | Content-Type=application/json | Post | Username为空 | 提示用户名不为空 | pass |

**完整http的请求过程:**

浏览器(客户端)发起TCP连接请求,TCP三次握手建立连接,客户端将HTTP请求数据包发送给服务端,服务器端响应http请求,浏览器得到html代码,浏览器解析html代码,并请求html代码中的资源,渲染后呈现给用户.

**http中存在如下问题：**请求信息明文传输，容易被窃听截取。数据的完整性未校验，容易被篡改,没有验证对方身份，存在冒充危险.

**HTTPS 协议：** HTTP+SSL/TLS，通过 SSL证书来验证服务器的身份，并为浏览器和服务器之间的通信进行加密。

**SSL**：安全套接字层1994年为 Netscape 所研发，SSL 协议位于 TCP/IP 协议与各种应用层协议之间，为数据通讯提供安全支持。

**TLS：**传输层安全其前身是 SSL，它最初的几个版本（SSL 1.0、SSL 2.0、SSL 3.0）由网景公司开发，1999年从 3.1 开始被 IETF 标准化并改名.

**REST:**表现层状态转移是HTTP协议（1.0版和1.1版）的主要设计者Roy Fielding提出的的一种架构风格。REST 规范中有如下几个核心：REST中一切实体都被抽象成资源，资源通过URI来指定。使用HTTP1.1协议定义的通用动词GET/POST/PUT/DELETE，来实现资源的增删改查操作。服务端和客户端之间用JSON，XML传输文本，或者用JPG，WebP传输图片。

**RESTful API:**RESTful是一种接口设计的架构风格，它不是需要严格执行的标准，而只是提供了一组可用设计原则，通过约定的规范，便于开发、测试理解沟通。越来越多的互联网公司，已经要求接口设计必须按照这种规范执行。

这样的API格式，可以达到如下目标：通过URI，就知道需要操作什么资源,通过http请求方法，就可以知道对资源进行何种操作。通过http 返回状态码，就知道操作的结果。

RESTful API格式如下: http://服务器地址:端口号[/项目名称/版本]/资源集合[/单个资源]

**浏览器在使用HTTPS传输数据的流程:**

首先客户端通过URL访问服务器建立SSL连接。 服务端收到客户端请求后，会将网站支持的证书信息（证书中包含公钥）传送一份给客户端。客户端的服务器开始协商SSL连接的安全等级，也就是信息加密的等级。

客户端的浏览器根据双方同意的安全等级，建立会话密钥，然后利用网站的公钥将会话密钥加密，并传送给网站。

服务器利用自己的私钥解密出会话密钥。服务器利用会话密钥加密与客户端之间的通信。

**HTTPS的缺点:**HTTPS协议多次握手，导致页面的加载时间延长近50%；HTTPS连接缓存不如HTTP高效，会增加数据开销和功耗；申请SSL证书需要钱，功能越强大的证书费用越高。SSL涉及到的安全算法会消耗 CPU 资源，对服务器资源消耗较大。

**HTTPS和HTTP的区别：**https协议需要到CA申请证书，一般免费证书较少，使用需要一定费用。

http是超文本传输协议，信息是明文传输，https则是具有安全性的ssl加密传输协议。

http和https使用连接方式不同，默认端口也不一样，http是80，https是443。

HTTPS协议是SSL+HTTP协议构建的可进行加密传输、身份认证的网络协议，比http协议安全。

**Cookie的原理:** W3C组织提出的保存在客户端本地，方便下次访问服务端时直接放到请求报文头中，如登录的cookie，下次访问同一网站时，用户不必再次输入用户名和密码就已经直接登录，所有的主流浏览器如IE、Netscape、Firefox、Opera等都支持Cookie。

**session原理**Session保存在服务器上用来记录用户状态的机制。客户端浏览器访问服务器的时候，服务器把客户端信息以某种形式记录在服务器上。这就是Session。Session在用户第一次访问服务器的时候自动创建。客户端浏览器再次访问时只需要从该Session中查找该客户的状态就可以了。一般客户端和服务端通过一个SessionID来进行沟通，为了防止不同的用户之间出现冲突和重复，SessionID一般是一个32或者48个字节的随机字符串。

**cookie 和session 的区别：**存放位置不同：cookie数据存放在客户端（浏览器）；session数据放在服务器端，一般存储在内存中，但是SessionID存储在客户端cookie中。cookie由浏览器存储在本地，安全有风险，不宜存储敏感信息，如密码等。session会在一定时间内保存在服务器上，访问较多时，影响服务器性能。

**接口的常见类型：**

**webService接口：**它是简单对象访问协议（soap）通过http传输，SOAP=RPC+HTTP+XML，即采用HTTP作为通信协议，RPC（Remote Procedure Call Protocol 远程过程调用协议）作为一致性的调用途径，XML作为数据传送的格式，从而允许服务提供者和服务客户经过防火墙在Internet上进行通信交互，请求报文和返回报文都是xml格式，常使用测试调用工具有SoapUI、jmeter、loadrunner等。

**http api接口：**基于http协议，一般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，请求方法最常见的为get和post方法，常见的测试调用工具有postman、RESTClient、jmeter、loadrunner等。

**Fiddler：**Fiddler被称为抓包神器，它是一个http协议调试代理工具，它能够记录互联网之间的http协议通信，可以设置断点，查看所有“进出”Fiddler的数据（如cookie,html,js,css等文件）。Fiddler 要比其他的网络调试器要更加简单，操作便捷，格式显示清晰，这个工具也可以进行接口测试，但是不能写断言，只能用composer构造HTTP请求，需要人工确认返回结果的正确性。

**Postman：**是谷歌开发的一款接口测试软件，它使用简单，功能强大，能够发送任何类型的HTTP请求 (GET, HEAD, POST, PUT..)，可以进行文件上传、响应验证、变量管理、环境参数管理，还可以批量管理和运行用例，并支持用例导出、导入，可以进行接口的自动化测试。缺点是自动化断言不够强大，不能和jenkins、代码管理库进行持续集成测试。

**soapUI：**是一个开源测试工具，通过soap/http来检查、调用，实现Web Service的接口和性能测试。该工具既可作为一个单独的测试软件使用，也可利用插件集成到Eclipse，maven2.X，Netbeans 和intellij中使用。soapUI pro是soapUI的商业非开源版本，实现的功能较开源的soapUI更多，国外公司使用较多。

**Jmeter：**是一款100%纯Java编写的免费开源工具，主要用来做性能测试，但也可以做接口测试，配合后置处理器与断言，可以满足大部分的接口测试场景，JMeter提供了BeanShell编程能力，可以写出比较灵活的测试脚本，通过jmeter+ant+jenkins可以实现接口和性能自动化测试。

**代码实现**:java代码实现接口测试：java可以通过httpclient.jar包，利用JUnit或者TestNG这样的单元测试工具进行用例编写，结合jenkins实现接口持续集成测试。

**python代码实现接口测试：**python可以利用强大的第三方库Requests实现接口用例编写，结合unittest单元测试框架，生成测试报告，也可以和jenkins做持续集成测试。

**Postman的使用：**最常用的请求方法有GET，POST，PUT和DELETE，它们的区别如下：GET请求：用于查询获取数据。POST请求：创建或更新数据，PUT请求：更新数据，DELETE请求：删除数据，HTTP请求包含四个部分：URL，方法，header头部和body主体。

了解接口测试都包含哪些测试活动，如何整理接口测试需要的输入条件。合成接口测试技术和业务测试知识，让你形成接口测试思维。

会通过案例的方式教你如何从流水账式的接口测试脚本，一步步抽象出属于自己的接口测试框架。随着课程的深入，你会建立建立一套自己的接口测试框架。

在上一个模块的基础上，继续从场景切入，教你学习如何测试一个完全陌生的协议接口，以及如何在测试微服务接口时，用 Mock 梳理混乱的调用关系。当然，这些能力最后都会叠加到你的测试框架中。不断提升它的测试能力。测试框架是你强大的武器，框架数据层的封装则是你的弹药库。所以最后我会教你完成外部依赖解耦的方法和思路，保证你能顺利完成测试任务.

**接口测试的思维开始，教你把业务测试的思维和接口测试的技术结合到一起，合成接口测试的思维，最终让你拥有接口测试的能力，这种能力既包含了工具的使用、代码的编写同时也包含用例的设计.**

项目的影响来说，接口测试直接测试后端服务，更加接近服务器上运行的代码程序，也更能发现影响范围广泛的 Bug。

通过接口自动化完成接口回归测试，让自己的工作更轻松、更高效；

通过持续集成平台调用接口自动化测试，为流水线提供质量保障方法和手段，赋能研发。

接口测试必要性，再讲讲什么是接口、什么是接口测试.

测试三环节:单元测试,接口测试,界面测试.

单元测试并非测试工程师的本职工作，它属于开发工程师的工作范筹.但开发很少做单元测试,故采用智能化框架补充单元测试工作或是加大接口测试,增大接口测试的测试深度和测试广度，往下逐渐覆盖一些公共接口的单元测试内容，往上则逐渐覆盖应该由 UI 层保障的业务逻辑测试.

接口测试更容易和其他自动化系统相结合；相对于界面测试，接口测试可以更早开始，也可测试一些界面测试无法测试的范围，使“测试更早的投入”这句话变成现实；接口测试还可以保障系统的健壮性，使得被测系统更健壮。

接口测试其实就是模拟调用方，比如 Client 端，通过接口通信来检测被测接口的正确性和容错性。

**理想的提测项目:**

**包含前期参与的产品需求、原型设计，由产品经理来提供的；**产品需求。它描述系统的业务逻辑，通过这个文档，你才能知道怎么来设计测试用例；原型设计。它会更加直观地告诉你系统的使用逻辑，这对测试用例的设计、对系统的前期认知都是有辅助作用的。

**包含后端接口文档、代码单元测试脚本，由开发工程师提来供的。**接口文档。它详细地描述了后端接口的访问方式和参数说明，使用这个输入项才能开展接口测试用例的设计、测试脚本的准备和测试数据的构建。单元测试脚本。它是保障提测质量的必要环节，是研发工程师自测的一个有效手段，可以保障提测项目的提测质量。

**现实的提测项目:**一个项目没有接口文档.

拿到一个 SUT 环境的时候，首先就要进行接口测试，因为单元测试不是由测试工程师来完成的，而是由开发工程师编写、并由持续集成系统自动完成执行的。但开发没有给有价值的文档,要开始接口测试,可从工具辅助,分析问题,询问解惑三个步骤完成.

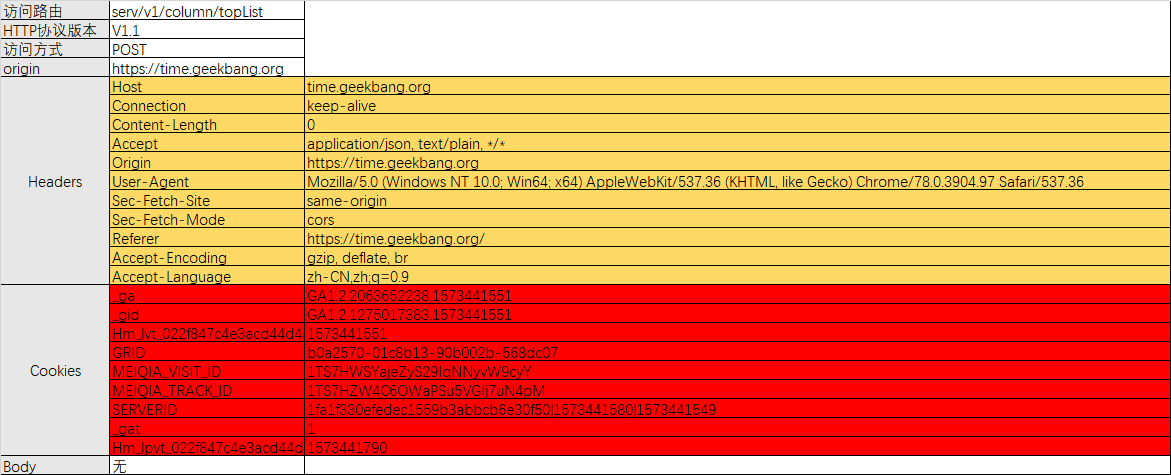
具体的工作模式：借助一些工具的辅助来完成接口分析；通过工具截获一些接口信息；通过分析接口的访问方式、参数等信息整理出一些问题，和研发工程师沟通这些问题，将一些不知道的参数含义、参数取值范围等问题问清楚。

**使用工具Fiddler:**是一个 HTTP 的调试代理， HTTP 协议的抓包工具，运行时会在本地建立一个代理服务，默认地址为 127.0.0.1:8888。当浏览器和被访问的服务之间发生交互，Request（请求）和 Response（响应）都会经由 Fiddler 代理，可截获全部的访问信息流。

先打开Fiddler(需配置抓取HTTPS的网址),然后启动浏览器访问被测项目地址,如<https://time.geekbang.org>,点击Fiddler的Inspectors的标签页,查看Request请求的内容和Response请求的内容.注意一下几个参数:

HOST，它表示指定访问的服务器域名；Connection 的值为 keep-alive，这表示需要持久连接；Accept，它表示客户端可以接受的内容类型为 application/json, text/plain, User-Agent，它说明请求是从什么浏览器发出去的；Sec-Fetch-Site 和 Sec-Fetch-Mode，它们是 JS 中对跨域的一些设置；Accept-Encoding 设置为 gzip、deflate、br，这表示可以支持的 Web 服务器返回内容压缩编码类型；Accept-Language，它表示接受的语言。

其中的 Cookie 的内容，特别重要，因为 Cookie 中传递的参数很多都是用来确认用户身份、鉴定角色权限等需要的参数。通过上面的分析绘制如下的表格.



在表格中，被标注为白色背景的部分，是这次访问的基本信息；被标注为黄色背景的部分，是访问的头信息，同时也是我们已知的内容；被标注为红色背景的部分，就是 Cookie 信息，是我们未知的内容。同时你可以看到，本次消息访问的 body 是空的，是没有内容的。

再看请求的Response信息，这些返回值包含了很多参数，也需要关注这些参数，很多时候，一个接口的返回值会是另外一个接口的入参，也就是我常说的串联业务逻辑上下文的参数。

拿着这张接口信息表，进入了第三步，询问解惑。去搞懂这些参数是干什么，起什么作用，怎样算搞懂，判断如下：

1. 你要搞清楚每一个参数的含义，也就是这个参数对应的实际自然语言的名字，通过记录每一个参数的中文语义，也会让更你容易记住这个函数是干什么的。同时，你也要知道这个参数的赋值是从哪里来的，是从其他页面的返回值中得到的？还是 JS 生成的？如果是其他页面或者接口返回的，那么，是哪一个接口返回的哪个字段？这样，当你开始做接口测试的时候，你就知道去哪里拿到这个参数的赋值了。如果是另一个接口的返回字段，那么，你还需要维护一份返回该参数接口的接口信息文档，以便于自己下一次创建对应的参数，如果不可以创建，那么你就要知道这个参数的生成规则，也要知道如何手动构造它。
2. 参数的作用域：参数的作用域指的是这个参数在这个接口中是做什么用的，它在哪一个访问周期里是一直存在的，它是否导致了业务逻辑分支等。比如说，这个参数是用来验证用户权限吗？它的验证算法是什么？之所以要搞清楚这些内容，是为了你在做接口测试的时候，可以设计更小的参数组合来覆盖更多的业务逻辑，这是测试用例去除冗余的一个很好的方法。
3. 返回值的含义：针对上面一大串的返回 JSON，你要搞清楚在返回值中，每一个 JSON 的 Key 所对应的含义，这样，当你需要和这个接口产生交互的时候，就可以快速地拿到对应参数的含义，完成业务逻辑上下文的参数串联了。

以上借助工具、通过分析问题明确未知参数，也通过询问解决未知参数的中文含义、作用域，以及对应返回参数的中文含义，现在，即使面对没有接口文档的提测项目，也能收集明确的、足够的信息。接下来进行业务逻辑的接口测试。

多个接口之间并不是随意组合的，而是按照业务逻辑、通过数据传递来完成的。要想使用接口测试完成业务逻辑，要制作一个流程中所有接口的接口信息表，同时理清每一个流程的数据流程，数据流程驱动业务流处理，才能开始业务逻辑的接口测试。

问题1：公司项目的单接口、串联接口都已经在做 但是好多时候接口已经测试完 然后开发改了部分接口，导致接口再次自动化回归的时候就又有问题了 这样很被动，请问下如何解决此类问题？

答：变更频繁这个问题目前只能在项目内部解决，服务端工程师应该设计接口，所有相关人评审后就不能变更，要变更需要有人承担成本。这个可以通过引入敏捷模式来进行规避，可以关注一下对应内容。

**如何通过一个实际方法落地接口测试呢？这里面就涉及到怎么做单接口的接口测试，怎么完成业务逻辑接口测试，以及用什么手段来完成接口测试等问题。接下来我会为你详细解答这些问题。**

工具使用Postman（HTTP协议客户端工具）

明确被测系统：xxx。先正确的跑遍流程 “正确”指的是“正确接受合法 Request 入参，正确拒绝非法 Request 入参”。

在postman中选择请求类型，输入请求地址，在body中的raw输入Request构造参数点击发送，查看Response内容。

然后设计测试用例，设计用例偏向于单元测试，即走通流程的用例。接下来是持续集成，使用工具Newman等，将postman的业务逻辑接口测试脚本导出后，push到本地的git仓库中，持续集成平台就可以通过pull对应的接口测试脚本然后通过newman执行，完成持续集成平台的赋能。

**“在接口测试中，我们通过单个接口测试完成了全部异常状态的覆盖；而在业务流程中，我们更需要关心业务流和数据流的关系，并不需要再过度关心如何用业务流的方法覆盖更多的代码逻辑异常，这也是分层测试中为什么在单元测试和界面测试之间要加入一层接口测试的主要原因之一”。**

**开发自己的测试框架：**

在win10环境配好python,先pip install requests,然后敲代码设计用例，见相关文件。再新建文件夹,新建一个common.py(见下图),然后一个个设计测试用例,模板使用unittest.



**测试框架支持RESTful风格接口：**

RESTful风格接口使用JSON格式进行数据交换.使框架支持可以借助python的相关支持库或自己封装.

**测试平台：**用工具和框架结合搭建平台.

**使用post工具设计脚本(在code里面,可读性差要改写),放到自己的框架中,使用github保存.**

**测试未知协议：**

接口测试最快速的方法不是去看协议的说明文档，而是直接去看开发实现的客户端代码.

**API 测试的基本步骤:**

**准备测试数据（这是可选步骤，不一定所有 API 测试都需要这一步）,** **通过 API 测试工具，发起对被测 API 的 request,** **证返回结果的 response。**

**API 测试工具，**比如常见的命令行工具 cURL、图形界面工具 Postman 或者 SoapUI、API 性能测试的 JMeter 等.