接口测试实训教程

V3.0

目录

1. 接口测试基础

1.1. 什么是接口测试

(1) 接口的定义

计算机中，接口就是通常说的 API， 是一些预先定义好的函数，目的是提供应用程序以及开发人员基于某软件或硬件得以访问一组例程的能力。

接口是指系统模块与模块、系统与系统间进行交互的入口，一般我们用的多的是 HTTP 协议的接口、WebService 协议的接口。我们常说的接口一般指三种：

1) API：Application Programming Interface 应用程序编程接口
2) GUI：Graphical User Interface 图形用户界面（接口）
3) CLI:Command Line Interface 命令行接口

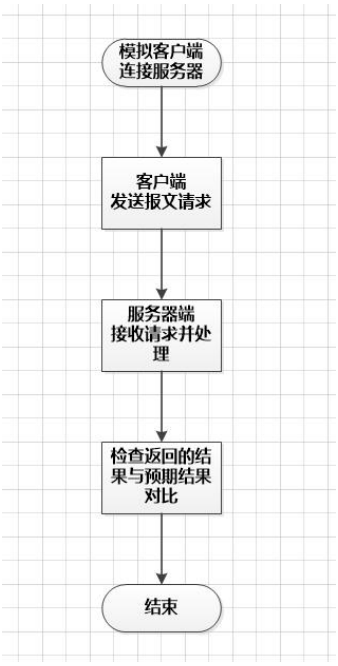
而我们测试的接口通常指的是 API 接口。

(2) 接口测试的定义：

接口测试是脱离前端页面，直接通过请求报文调用 API 接口，验证输入输出、业务逻辑正确性的功能测试，重点关注数据交换与传递的准确性；通常包括测试接口的参数检查、接口的参数传入及接口返回值是否正确，各接口间逻辑调用是否可以实现应用层功能，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。**接口测试的重要意义是**实现开发期并行测试，减少页面层测试的深度，缩短整个项目的测试周期。

(3) 接口测试的原理：

接口测试的原理是通过测试程序模拟客户端向服务器发送请求报文，服务器接收请求报文后对相应的报文做出处理，然后再把应答报文发送给客户端，客户端接收应答报文这一个过程。



1.2. 为什么要做接口测试

接口测试的核心作用是**提前发现**“模块/系统间数据交互”的**问题**，避免问题传递到前端或用户端，同时**降低整体测试成本、保障系统稳定性与安全性。**从成本、效率、稳定性、安全性多维度保障系统质量。

(1) 提前发现问题，降低修复成本

接口是数据流转的“通道”，若通道存在参数校验缺失、响应格式错误等问题，等前端开发完成后再发现，需同时修改“接口逻辑”和“前端适配代码”，修复成本高。而接口测试可在前端开发前（仅需接口文档和测试环境）执行，发现问题后仅需调整接口，大幅降低修复成本。

- 例：用户注册接口未校验“手机号格式”，接口测试阶段发现仅需修改接口校验逻辑；若上线后用户反馈，需改接口+前端提示，还可能产生无效账号的脏数据。

(2) 保障数据交互的正确性，避免业务异常

接口是业务逻辑的“核心载体”（如下单、支付、数据查询均依赖接口），测试可验证“请求数据是否被正确接收”“响应数据是否符合预期”，直接保障业务流程正常。

- 例：下单接口若未正确计算“商品单价×数量”，接口测试会发现响应的“订单金额”错误，避免用户支付时出现金额异常的业务故障。

(3) 覆盖边界与异常场景，提升系统稳定性

前端测试难以覆盖接口的异常场景（如参数缺失、非法值、高并发），而接口测试可针对性验证这些场景，防止系统因异常请求崩溃、超时或返回乱码，提升系统抗风险能力。

- 例：测试支付接口时，可模拟“不传支付金额”“传负数金额”，“短时间 1000 次并发请求”等场景，验证接口是否能正常拦截错误、稳定响应，避免线上系统卡死。

(4) 防止数据泄露或滥用，保障接口安全性

接口是外部（第三方系统、恶意用户）访问内部数据的“入口”，测试可验证接口的安全防护逻辑，避免越权访问、敏感数据泄露、SQL 注入等风险。

- 例：通过测试验证“用户信息查询接口”：未传身份 Token 时拒绝访问（防未授权）、用 A 用户 Token 无法查询 B 用户数据（防越权）、过滤请求中的 SQL 注入语句（防数据库攻击），保障数据安全。

(5) 提供复杂系统下的低成本高效率解决方案

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（数据模型推算，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；

(6) 持续集成测试，提高测试效率

接口测试不同于传统开发的单元测试，接口测试是站在用户的角度对系统接口进行全面高效持续的检测；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。

(7) 前后端分离，保障系统架构安全

现在很多系统前后端架构是分离的，从安全层面来说：

- 只依赖前端进行限制已经完全不能满足系统的安全要求（绕过前面实在太容易），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。
- 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，特别是涉及到用户的隐私信息，如身份证，银行卡等。

1.3. 接口测试分类

- **根据测试的层面不同分为：**
  - **代码层接口测试**：通常指的是系统模块间接口测试，即代码层面的接口调用。如:Java 中 UserService 接口调用 OrderService 接口（模块内）
  - **协议间接口测试** ：协议是指通信双方实现相同功能的相应层之间的交往规则 ，如:Web 中的 http 协议接口测试、socket 协议接口测试、telnet 协议接口测试 ；
  - **服务间接口测试** ：服务是一种应用程序类型，它在后台运行，服务应用程序通常可以在本地和通过网络为用户提供一些功能。如:电商系统调用支付宝支付接口（跨系统）
- **根据测试对象的不同：**
  - 外部接口：系统与系统之间的调用，如:电商项目中的订单支付需要调用第三方支付接口：网银支付、支付宝支付，微信支付等接口；
  - 内部接口：子系统模块间的调用，如:电商项目中订单生成时需要调用用户管理模块的用户查询接口、商品管理模块中的商品查询接口、库存管理模块的库存更新接口等；这里也是一个典型的多个接口逻辑之间有前后关联的例子，首先，需要先调用用户查询接口，如果查有此人，接口返回用户 id，然后，用户 id 作为参数请求第二个商品查询接口，接口返回商品 id，再次，商品 id 作为参数请求第三个接口库存管理接口，接口返回库存值。这里就串联成了一个流程，前一个接口返回值作为下一个接口请求参数。

1.4. 接口测试流程

接口测试的流程于功能测试流程基本一致，基本流程是“**接口需求分析→接口用例设计→准备测试环境→接口测试执行→测试报告分析**”，按标准化步骤落地可确保测试覆盖全面、结果可靠，具体拆解为 6 个关键阶段：

(1) 接口需求分析

接口测试前需要明确“测什么”，对接口文档进行需求分析，明确每个接口的功能、参数、协议、响应规则。

- 对接开发/产品，获取**接口文档**（如 Swagger、Apifox 文档），明确接口的“请求方法”（GET/POST）、“必填参数”（如用户 ID）、“参数格式”（如字符串/数字）、“预期响应”（如成功返回 200 状态码+用户信息）。
- 确认特殊规则，如“登录接口需传验证码”“订单接口需先登录（带 Token）”。
- **例**：从文档中明确“用户登录接口”：POST 方法，参数为 username（必填）、password（必填），成功响应为{"code":200,"data":{"token":"xxx"}}，失败响应为{"code":400,"msg":"账号或密码错误"}。

(2) 设计接口测试用例

接口需求明确后，针对每个接口灵活应用设计方法，进行场景覆盖，包括正常场景、异常场景、边界场景，避免遗漏。

- **接口用例设计维度：**
  - 用例编号、用例名称（如“登录-输入正确账号密码”）；
  - 请求信息（协议、URL、方法、参数）；
  - 前置条件（如“需先获取验证码”）；
  - 预期结果（如“响应状态码 200，返回 Token”）。
- **常见用例场景：**
  - 正常场景：输入合法参数（如正确的账号密码），验证功能正常；
  - 异常场景：参数缺失（如不传密码）、参数非法（如密码为空格）、权限不足（如用失效 Token 请求）；
  - 边界场景：参数长度极限（如密码 1 位/20 位）、数据不存在（如用不存在的账号登录）。

(3) 搭建测试环境与准备数据

接口测试执行前，需要确保测试有“可用的环境”和“对应的数据”，避免环境/数据问题影响测试结果。

- 搭建/获取测试环境：如连接开发提供的“测试服接口地址”（如 http://test-api.xxx.com/login），而非生产环境；
- 准备测试数据：如创建测试账号（test01/123456）、初始化数据库数据（如确保测试账号状态为“正常”，而非“冻结”）；
- 配置工具：若用 Postman，可导入接口文档自动生成请求，或配置“环境变量”（如将登录后的 Token 保存为变量，供后续接口使用）。

(4) 执行接口测试

接口测试执行借用合适的测试工具，按照设计好的用例发送请求，对比“实际响应”与“预期结果”，判断接口是否符合要求。

- 手动执行：用 Postman/Apifox，按用例逐个输入参数、发送请求，查看响应是否匹配预期（如校验状态码、响应字段）；
- 自动化执行：用 JMeter/Python+Requests，将用例写成脚本，批量执行（适合大量用例或回归测试）。
- 执行用例后，标记“通过/失败”：若实际响应与预期一致（如返回 200+Token），则用例通过；若不一致（如传正确密码却返回 400），则标记失败并记录现象。
- 记录“失败现象”：如“请求参数正确，但响应提示‘数据库连接超时’”，便于后续定位问题。

(5) 缺陷管理与回归测试

接口测试过程中发现的问题，同样需要提交给开发进行修复，使用缺陷管理工具禅道跟进 Bug 推动问题修复，并验证修复结果。

- 提交缺陷：将失败用例的“场景、请求信息、实际响应、截图”提交到缺陷管理工具（如禅道），指派给开发；
- 回归测试：开发修复后，重新执行“失败的用例”，确认问题已解决（如之前登录失败的用例，修复后能正常返回 Token）；
- 若回归失败（问题未修复），则重新跟进开发，直到问题闭环。

(6) 生成测试报告与总结

接口执行完成输出测试结果，生成测试报告，反馈接口质量，为项目上线提供依据。

- 整理测试数据：统计“用例总数、通过数、失败数、通过率”，列出未修复的缺陷；
- 生成测试报告：包含测试范围、测试环境、用例执行情况、缺陷分析（如失败主要原因是“参数校验缺失”）、风险提示（如“登录接口未做防刷限制，可能存在安全风险”）；
- 同步报告给开发/产品，确认接口是否满足上线条件。

1.5. 接口测试与其他测试关系

- **接口测试与单元测试**
  - 针对性：单元测试，一般来说是针对具体的**代码逻辑**进行测试，尽量减少这些功能单元集成起来出错的可能性，一般是由开发人员来完成；而接口测试，更注重从用户的角度设计用例，更偏向于功能测试，
  - 用例设计：单元测试设计测试用例的时候，可能更多的考虑是代码覆盖，而接口测试，则需要更多的考虑业务覆盖。
  - 测试人员：单元测试由开发人员来做，可以保证从代码角度来看是没有问题的，但服务保证业务角度来看也是没有问题的，而接口测试，则由测试人员通过业务的角度去设计测试用例，其实，也可以说是从更早的时候，以功能测试的方法，先保证项目的流程及功能是正常的，而不至于在页面开发完成后，又修改主要功能代码，导致项目赶工及一系列的重写。
- **接口测试与功能测试**

  功能测试用于测试系统功能是否满足业务逻辑；功能测试包含一个或多个接口测试；接口测试能涵盖一定的功能测试。
- **接口测试与自动化测试**

  自动化测试如 app 自动化测试、web 自动化测试，都是模拟人类行为的测试；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**

  自动化测试的模拟行为，测试效率较慢；接口测试可以直接与服务器进行快速交互，对接口进行性能、压力测试。
- **接口测试与安全测试**

  功能测试能一定程度上测试安全性，接口测试能大范围测试系统安全性，类似于模拟黑客攻击的行为。

1.6. 接口测试的策略与质量评估标准

- **接口测试策略**

在进行接口测试之前，首先要整理接口测试方案，分析接口测试要点，其中测试内容主要有：

- 接口设计检查

- 接口依赖关系检查
- 接口输入输出验证
- 测试每个参数类型不合法的情况(类型不合法容易遗漏 NULL 型)
- 测试每个参数取值范围不合法的情况
- 测试参数为空的情况
- 测试参数前后台定义的一致性
- 测试每个参数的上下限(这里容易出致命的 BUG，如果程序处理不当，可能导致崩溃)
- 如果两个请求有严格的先后顺序，需要测试调转顺序的情况

➢ **接口测试质量评估标准**

- 业务功能覆盖是否完整；
- 业务规则覆盖是否完整；
- 接口异常场景覆盖是否完整；
- 参数验证是否达到要求（边界、业务规则）
- 接口覆盖率是否达到要求
- 代码覆盖率是否达到要求
- 性能指标是否满足要求
- 安全指标是否满足要求

1.7. 常见接口协议及报文

➢ **常见的接口协议**

- HTTP
- WebService
- RPC
- Socket

随着互联网技术的多样化，各种项目都在走接口的模式，因而使用的对应的协议也是多样的，在众多的协议中又以 HTTP 协议的接口居多；

➢ **常见接口报文**

Xml 报文：参数名和参数值以自定义标签的形式存在，参数名为标签名，参数值在标签中；

```
<resp>
   <city>北京</city>
   <updatetime>15:18</updatetime>
   <wendu>-4</wendu>
   <fengli>3 级</fengli>
   <shidu>18%</shidu>
   <fengxiang>西南风</fengxiang>
   <sunrise_1>07:36</sunrise_1>
   <sunset_1>16:59</sunset_1>
   <sunrise_2/>
   <sunset_2/>
   <yesterday>
      <date_1>30 日星期一</date_1>
      <high_1>高温 -4℃</high_1>
```

```
      <low_1>低温 -12℃</low_1>
      <day_1>
         <type_1>晴</type_1>
         <fx_1>西北风</fx_1>
         <fl_1>4-5 级</fl_1>
      </day_1>
      <night_1>
         <type_1>晴</type_1>
         <fx_1>北风</fx_1>
         <fl_1>&lt;3 级</fl_1>
      </night_1>
   </yesterday>
</resp>
```

- Json 报文：参数名和参数值以键值对的形式存在，即”key” ：“value”；

```
{
   "ret": 200,
    "data": {
      "err_code": 0,
       "err_msg": "",
       "uuid": "8E4C366FB1185463C4BC5A466D097881",
        "token": "C8699F63270536578D32C3C05EBAEFFB716E69F22F0832ABAF3B7162F43FA7C3",
       "role": "user"
    },
    "msg": "欢迎登录接口 App.User.Login"
}
```

- 接口传递数据方式
  - Get 方式是从服务器上获取数据；在做数据查询时，建议用 Get 方式；如：公共服务接口、搜索接口、博客访客系统接口等
  - Post 方式是向服务器传送数据 ；在做数据添加、修改或删除时，建议用 Post 方式 ；如：微博图片上传图片接口、Picself API 接口等
  - Put 方式：这个方法比较少见。HTML 表单也不支持这个。本质上来讲， PUT 和 POST 极为相似，都是向服务器发送数据，但它们之间有一个重要区别，PUT 通常指定了资源的存放位置，而 POST 则没有，POST 的数据存放位置由服务器自己决定。
  - Delete：删除某一个资源。基本上这个也很少见。

1.8. 接口测试工具

- **常见接口测试工具**

典型开源工具:postman、 apifox、Jmeter、 Jsoup、HttpClient、Python 中 requests 库、Java 的 RestAssured 库等。典型商业工具：SoapUI、 LoadRunner；

- Postman

最主流的入门接口测试的工具， HTTP/HTTPS 等协议，可保存请求用例、批量执行、简单断言（如校验响应状态码、关键字段），还能通过 “环境变量” 实现接口关联（如用前一个接口的 “订单号” 作为后一个接口的参数）。
适用场景：个人学习、小型项目快速测试、接口调试。

- Apifox

国产工具，整合了 “接口文档管理 + 测试 + Mock” 功能，对中文用户更友好。支持自动生成接口文档，测试用例可直接关联文档，断言逻辑可视化配置，适合团队协作（如多人共享用例）。
适用场景：国内团队、需要兼顾文档与测试的场景。

- JMeter

Apache 开源工具，性能测试首选，也可做接口功能测试。支持 HTTP、TCP、JDBC（数据库）等多种协议；可模拟 thousands 级并发请求（测试接口在高压力下的稳定性）；通过 “断言”“逻辑控制器” 实现复杂场景（如 “登录成功后才执行下单接口”）。
适用场景：接口性能测试（如秒杀场景）、需要批量执行大量用例的场景。

- SoapUI

专注于 SOAP 协议接口（早期接口常用，现在更多是 REST），也支持 REST。支持复杂的接口自动化测试、数据驱动测试（用 Excel 等数据源批量传参），还能集成 JUnit 等框架做持续集成。
适用场景：测试 SOAP 接口、需要深度自动化校验的场景。

- Python + Requests 库

Requests 是 Python 的 HTTP 请求库,代码简洁(如 1 行代码发送 GET 请求),配合 pytest（测试框架）可实现用例管理、断言、报告生成；再结合 Allure 能生成美观的测试报告。
适用场景：需要自定义复杂逻辑（如接口参数加密、与数据库 / 缓存联动）、融入 CI/CD 流程（如 Jenkins 自动触发测试）。

- Java + RestAssured

Java 生态下的接口测试库，语法简洁，支持 BDD（行为驱动开发）风格编写用例（如 given-when-then 结构），适合 Java 技术栈的团队。
适用场景：Java 项目、需要与 JUnit/TestNG 等 Java 测试框架集成的场景。

- **接口测试工具的选择**

- 投入成本：确定是选择开源免费工具还是商业付费工具；
- 接口协议类型：根据项目选择 Http 协议还是 Webservice 协议；
- 接口功能：工具有专做功能的，也有功能性能都可做的；
- 测试人员技术能力：具备代码功底可以考虑使用代码来实现，否则考虑使用现有工具；
- 使用复杂度： 轻量级选择浏览器插件，重量级选择 Loadrunner 工具；
- 根据不同的需求场景，选择合适的工具

需求场景	推荐工具
新手入门、快速调试接口	Postman / Apifox

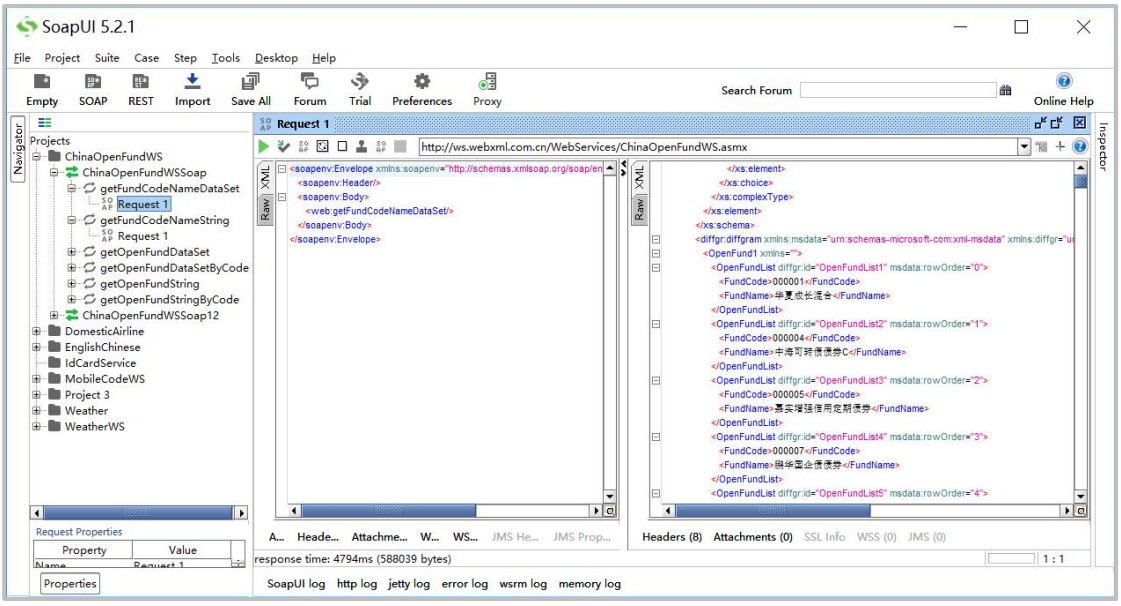
需求场景	推荐工具
性能测试、高并发场景	JMeter
SOAP 协议接口测试	SoapUI
代码化测试、复杂逻辑	Python+Requests / Java+RestAssured
国内团队、文档 + 测试一体	Apifox

- **常见接口抓包工具**

接口测试过程中通常也需要借助抓包工具分析接口参数和返回，分析接口存在的问题。常见的接口抓包工具有：Fiddler 、Charles，Httpwatch、Wireshark，也可以借助浏览器插件来进行抓包，比如：Chrome 浏览器 F12，Firefox 浏览器 Firebug 。

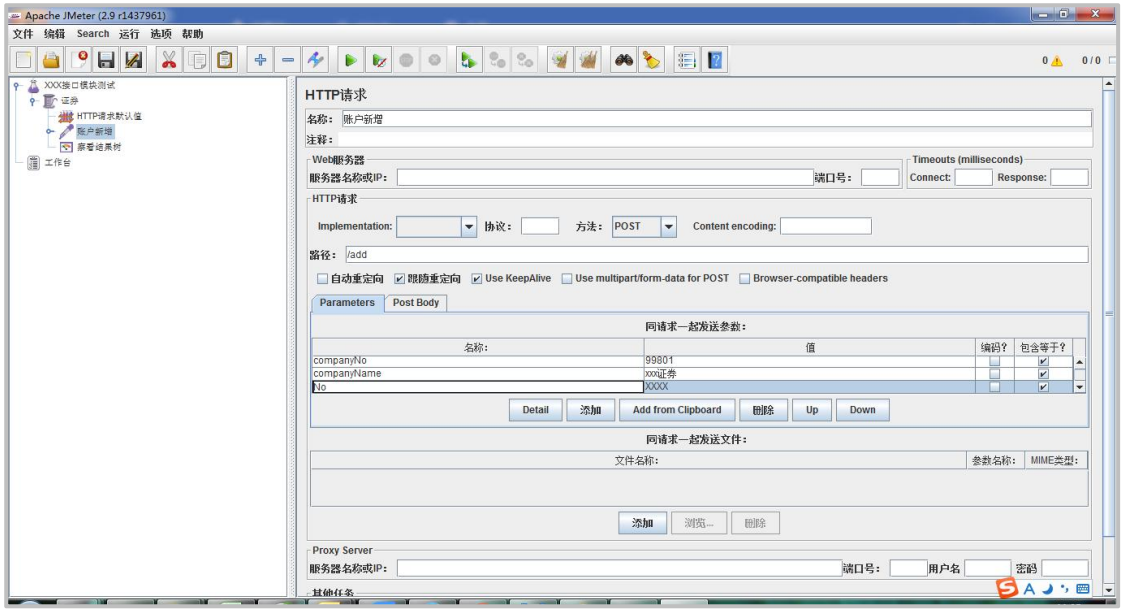
- SoapUI

SoapUI 是一个免费、开源、跨平台的功能测试解决方案。一个易于使用的图形界面，和企业级功能，让你轻松和 soapUI 迅速创建和执行自动化的功能，回归，合规性，测试和负载测试。



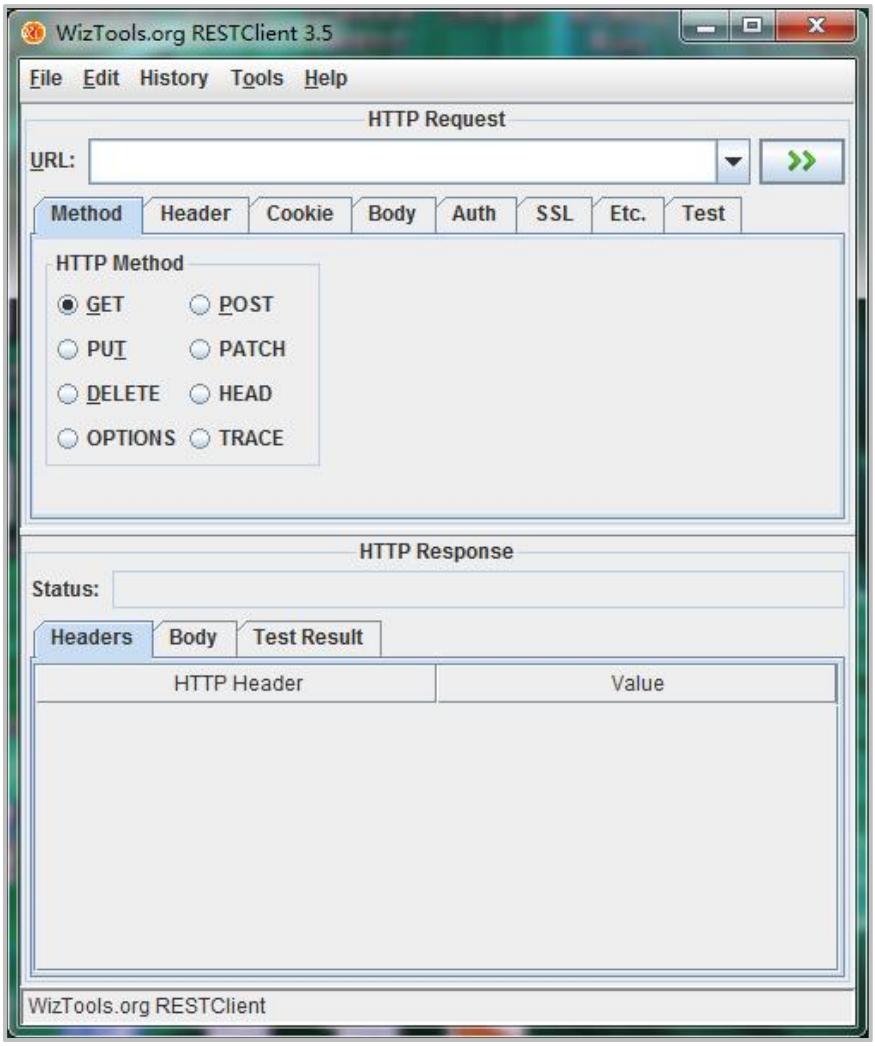
- JMeter

Apache JMeter 是 Apache 组织开发的基于 Java 的开源的测试工具， JMeter 可以用于对服务器、网络或对象模拟巨大的负载，来自不同压力类别下测试它们的强度和分析整体性能。另外，JMeter 能够对应用程序做功能/回归测试/接口测试，同时 JMeter+Ant+Jenkins 也可以搭建接口和性能的持续集成测试平台



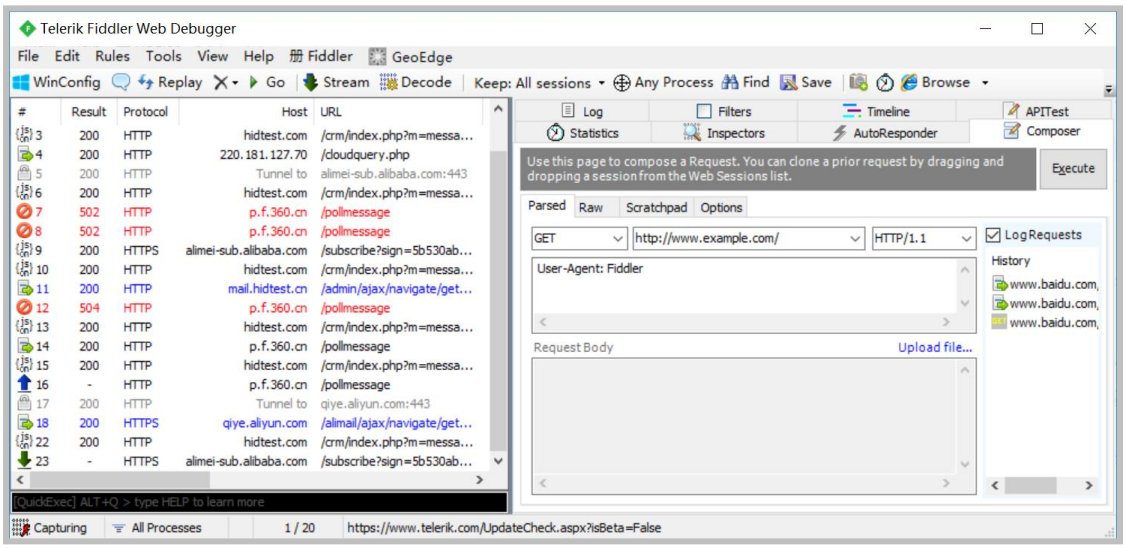
- RESTClient

  RESTClient 是用 java Swing 编写的基于 http 协议的接口测试工具，工具比较灵巧，便于做接口的调试,源码在官网上可以下载。



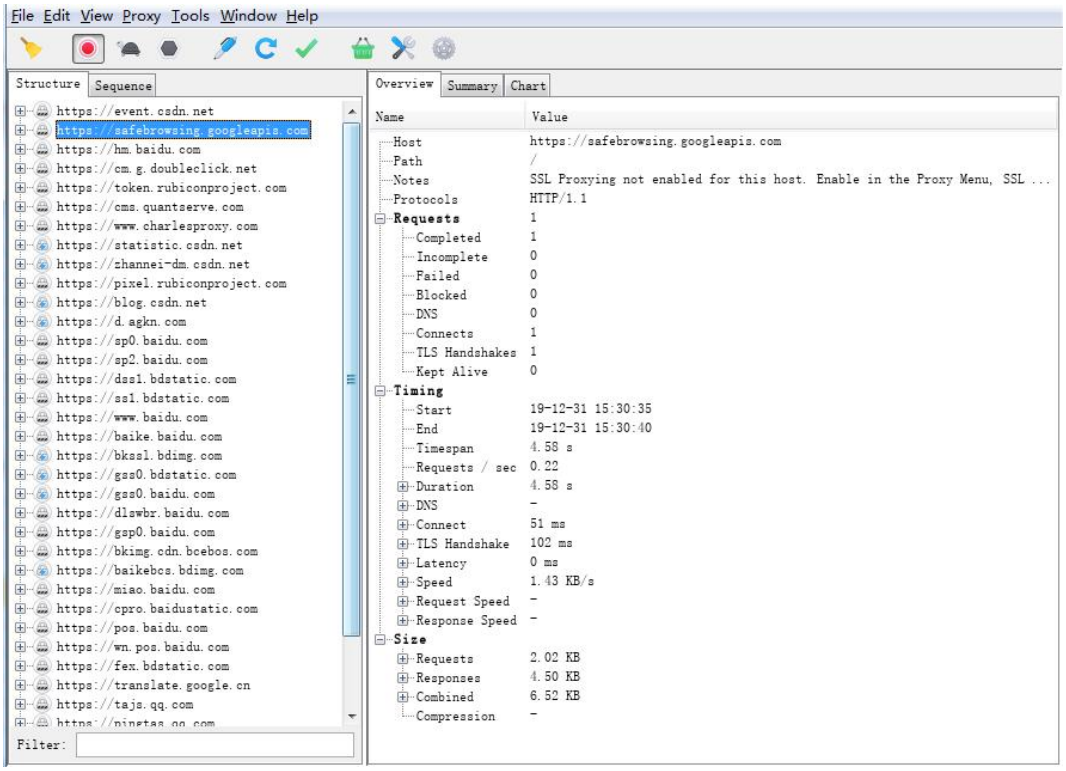
– Fiddler

Fiddler 是一个 http 协议调试代理工具，它能够记录并检查所有你的电脑和互联网之间的 http 通讯，设置断点，查看所有的“进出”Fiddler 的数据（指 cookie,html,js,css 等文件）。 Fiddler 要比其他的网络调试器要更加简单，因为它不仅仅暴露 http 通讯还提供了一个用户友好的格式。



- Charles

Charles 是一个 HTTP 代理服务器,HTTP 监视器,反转代理服务器，当浏览器连接 Charles 的代理访问互联网时，Charles 可以监控浏览器发送和接收的所有数据。它允许查看所有连接互联网的 HTTP 通信，包括 request，response 和 HTTP headers。



2. 接口测试用例设计

2.1. 接口测试文档

➢ 接口文档规范

接口文档可以包含很多信息，根据项目的要求进行调整。不过，下面几项内容必须有，这是我们使用接口中和测试接口的依据：

（1） 接口名称 ：标识各个接口的简单说明，如登录接口，获取项目详情接口等；

（2） 接口 URL：接口的调用地址，在测试环境下前面的域名可能不一样，不过接口名是不会变的；

（3） 调用方式：接口的调用方式：POST/GET/PUT/DELETE/HEAD 方式，决定了如何调用接口及传递参数；

（4） 输入参数：接口需要传递的参数，参数具体说明

（a） 参数类型：参数值要说明支持字母、数据、特殊字符或是字母数据混搭；

（b） 参数描述：参数的业务含义、字段描述；

（c） 参数长度：参数接收最大多少个的字符串，或是最大是多少的数值等。

（d） 参数取值范围：像枚举型的参数，只接收什么范围内的数据，如 1-5 等。

（e） 参数的配合：有些参数需要配合起作用的，如：offset 和 count 参数；

（f） 参数必选：参数是否是必选的；

（5） 返回值：接口的返回值说明需要包含正确和错误的情况，正确的情况下有哪些数据，

错误的情况下会有什么提示；

（6） 其他说明：除上述的通用说明外，还有其他的一些说明，如必须是登录状态调用，或是版本号等说明，在某些情况下也需要说明一下；

严格要求的公司会要求接口开发人员标明上面相关的接口说明，此时我们做接口测试的时候，就可以参照文档来转化我们的测试用例。但目前国内很多公司不太注重文档，接口信息不全是常有的事情。

➢ 接口文档格式

- **接口描述：**

  根据账号和 md5 后的密码进行会员登录

- 请求说明：

  请求 url：http://hn1.api.okayapi.com/?s=App.User.Login

  请求方法：get/post

- 接口参数：

参数名称	参数类型	是否必须	默认值	备注	参数说明
app_key	字符串	是		长度为 32 位	注册后自动生成
username	字符串	是		1-50 位之间	登录用户名
password	字符串	是		长度为 32 位	MD5 加密过后的密码

- 返回字段说明：

返回字段	类型	说明
ret	整型	接口状态码，200 表示成功，4xx 表示客户端非法请求，5xx 表示服务端异常，查看异常错误码
data.err_code	整型	操作码，0 登录成功，1 登录失败，账号不存在；2 登录失败，密码错误；3 会员已过期，请联系管理员或开发者；4 账号已被封号
data.err_msg	字符串	错误提示信息，err_code 非 0 时参考此提示信息
data.uuid	字符串	全局唯一 UUID，全局唯一用户 ID
data.token	字符串	登录凭证，一个月内有效
msg	字符串	提示信息，面向技术人员的帮助或错误提示

- ret 异常错误码：

错误码	错误类型	错误描述信息	解决方法
ret = 200	成功	请求成功	
ret = 401	客户端非法请求	用户未登录，或登录凭证已过期	1、如果用户未注册，请先用注册接口；2、如果注册未登录或会话过期，请先用登录接口
ret = 404	客户端非法请求	接口服务不存在	1、查看小白接口大全，确保接口服务名称拼写正确
ret = 500	服务端异常	如果出现此错误，请联系技术人员处理	

- 请求示例：

```
http://hn1.api.okayapi.com/?s=App.User.Login&app_key=C7630186182EF2A5462FEFE8A45D2390&username=test_user&password=e10adc3949ba59abbe56e057f20f883e
```

- 响应示例：

```
    {
```

```
        "ret": 200,
        "data": {
            "err_code": 0,
            "err_msg": "",
            "uuid": "8E4C366FB1185463C4BC5A466D097881",
            "token": "C8699F63270536578D32C3C05EBAEFFB716E69F22F0832ABAF3B7162F43FA7C3",
            "role": "user"
        },
        "msg": "欢迎访问接口 App.User.Login"
}
```


2.2. 接口测试用例设计

➢ 接口测试用例编写要点

1. 测试每个参数类型不合法的情况(类型不合法容易遗漏 NULL 型)
2. 测试每个参数取值范围不合法的情况
3. 测试参数为空的情况
4. 测试参数前后台定义的一致性
5. 测试每个参数的上下限(这里容易出致命的 BUG，如果程序处理不当，可能导致崩溃)
6. 如果两个请求有严格的先后顺序，需要测试调转顺序的情况

➢ 接口测试用例的参考点

- 参数测试：
  针对输入参数进行的测试，也可以说是假定接口参数的不正确性进行的测试，确保接口对任意类型的输入都做了相应的处理：输入参数合法（不合法），输入参数为空，为 null，输入参数超长等等；
- 功能测试：
  接口是否满足了所提供的功能，相当于正常情况测试，如果一个接口功能复杂时推荐对接口用例进行结构划分，这样子用例就有更好的可读性和可维护性；
- 逻辑测试：
  逻辑测试严格讲应为单元测试，单元测试应保持内部逻辑的正确性，可单元测试和接口测试的界限并不是那么清楚，所以我们也可以从给出的设计文档中考虑内部逻辑错误的分支情况和异常；
- 异常情况测试：
  接口实现是否对各种情况都进行了处理，接口输入参数虽然合法，但是在接口实现中，也会出现异常，因为内部的异常不一定是输入的数据造成的，而有可能是其他逻辑造成的，程序需要对任何异常都进行处理；

➢ 接口测试的主要内容

主要内容	主要考察点	子考察点

业务功能测试	场景设置	正常场景 异常场景
边界分析测试	业务规则边界分析 输入输出边界分析 参数组合测试	必输项参数、可选参数、参数有无，为空或为 null 参数类型、个数、顺序 参数取值范围、最大最小值、边界值、特殊字符 参数组合
异常情况测试	重复提交、并发测试、分布式测试	
性能测试	响应时间、吞吐量、并发数、服务器资源使用率	服务器 CPU、内存、IO、网络
安全测试	敏感信息加密、批量操作、SQL 注入	日志信息加密、数据传输加密、批量抽奖安全

- **接口测试用例模板**

接口测试用例的设计思路跟系统测试的用例设计一致，模板也跟功能测试是用例模板基本一致，常用模板包括以下字段：

1)项目名称
2)模块名　接口属于的功能模块；
3)用例编号：
4)接口名称：
5)用例标题 ： 用例描述
6)请求方式 ： GET/POST
7)请求 url ： url 地址
8)请求参数
9)前置条件 有依赖的时候，比如说要测登录失败 3 次的
10)结果验证 预期结果
11)请求报文
12)返回报文
13)测试结果 通过/失败
14)测试人员

项目名称	模块名	用例编号	接口名称	用例标题	请求方式	请求URL	前置条件	请求参数	请求报文	返回报文	结果验证	测试结果	测试人员
接口测试	会员模块	ST_001	会员登录	合法用户名密码，登录成功	POST	http://hnl.api.okayapi.com/?s=App.User.Login	无	1.app_key="C7630186182EF2A5462FEFE8A45D2390" 2.username="test_user" 3.password="e10adc394	{ "app_key": "C7630186182EF2A5462FEFE8A45D2390", "username": "test_user", "password":	{ "ret": 200, "data": { "err_code": 0, "err_msg": "", "uuid": "8E4C366FB1185463C4BC5A466D097881", "token":	ret=200 err_code=0		

3. 接口测试实践及应用

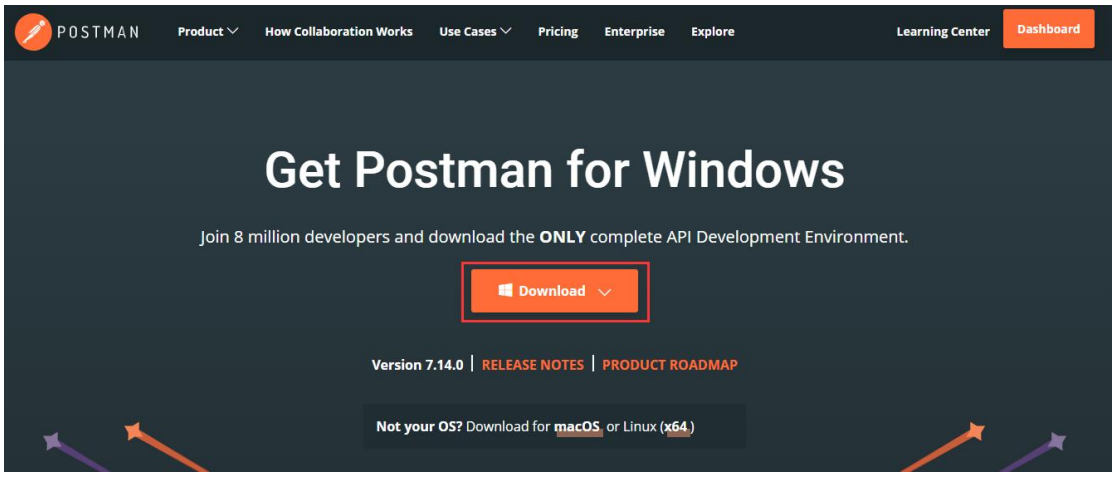
3.1. Postman 接口实战

Postman 是一款功能超级强大的用于发送 http 请求的工具，常用于 web 开发、接口测试，

使用非常方便。

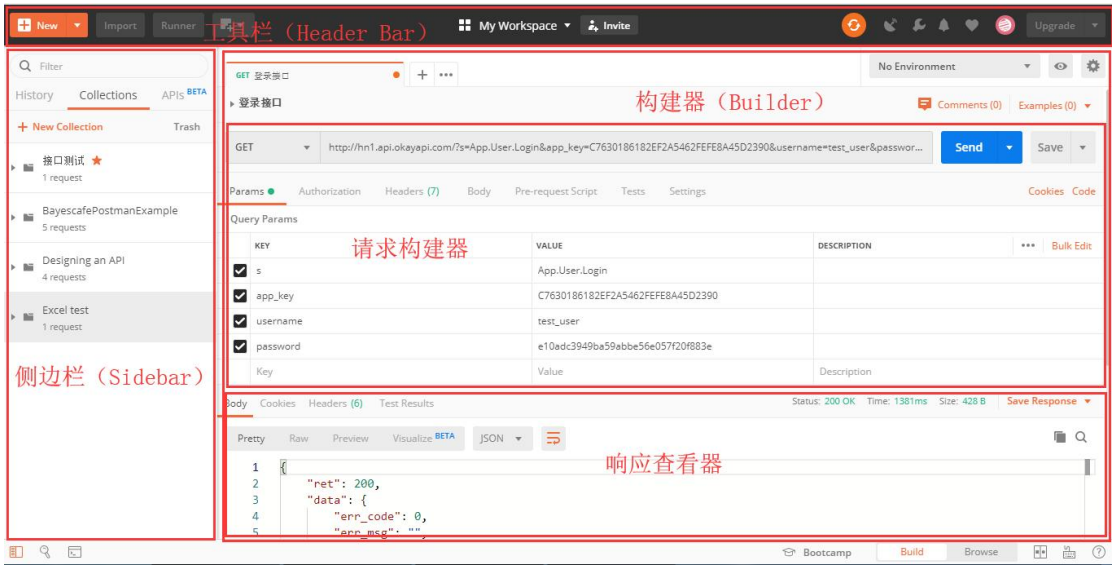
3.1.1 Postman 安装

在 windows 上安装 postman，到 postman 官方网站(https://www.getpostman.com)下载对应版本的安装包，双击该安装包文件进行安装。



3.1.2 Postman 界面导航

Postman 提供了一个多窗口和多标签界面。Postman 努力保持清洁灵活的需求。为你提供尽可能多的空间用于你的 API，导航界面很简单。Postman 主界面可分为三部分侧边栏、标题工具栏、构建器。



➢ 工具栏(Header Bar)

工具栏包含常用操作快捷按钮以及 Postman 用户相关设置：

new 按钮：创建请求，集合，环境，文档，模拟服务器和监视器

import 按钮：使用文件、链接或原始文本将 Postman 集合，环境等数据导入 Postman

runner 按钮：打开测试集运行界面

➢ 侧边栏(Sidebar)

Postman 边栏使您可以查找和管理请求和集合。侧边栏有两个主要标签 History 和 Collections。

History：Postman 会将你发送的每个请求都保存在 History 选项卡中

Collections：Collection 类似文件夹，可以把同一个项目的请求放在一个 Collection 里方便管理和分享，Collection 里面也可以再建文件夹

➢ 构建器(Builder)

Postman 提供了一个标签式布局，用于在构建器中发送和管理接口请求。上半部分是请求构建器，下半部分是响应查看器 。

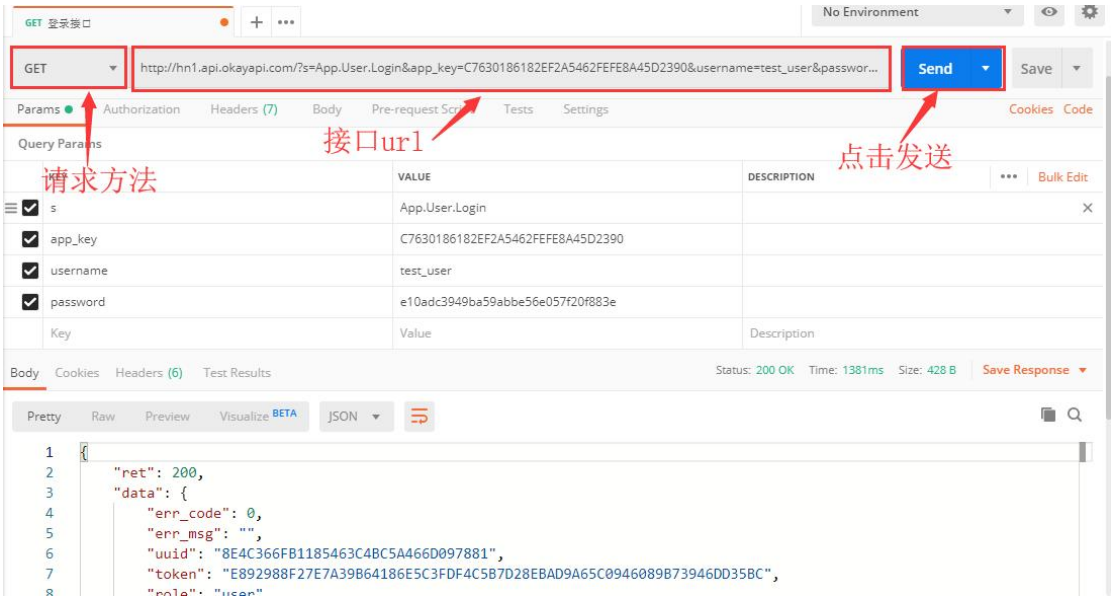
➢ 控制台(Console)

点击主窗口左下角的 Console 图片即可打开控制台窗口，控制台可用于查看当前会话的所有 http 请求和响应以及用户想要记录的所有消息，这些信息包括网络、请求头、响应头、响应体。

3.1.3Postman 请求与响应

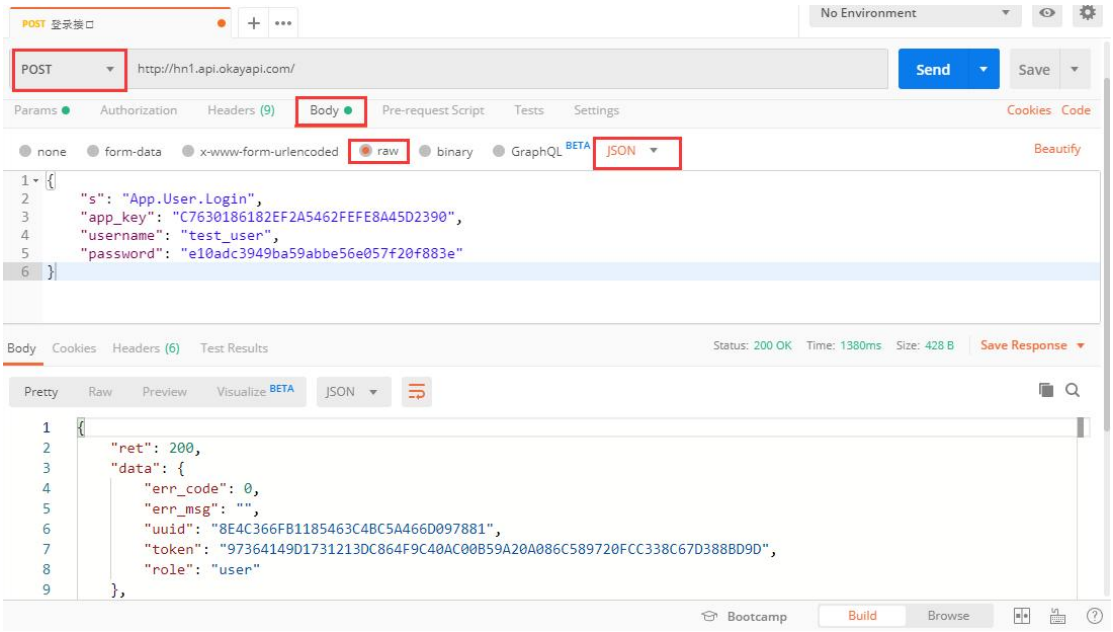
● 创建请求

在构建器中创建请求，一个 http 请求可能包含多个内容，这些内容决定了 Postman 将发送到接口的数据。但至少你需要输入 URL 并选择一种请求方法。



当请求需要携带参数的时候，参数可以直接附加到请求 url 的末尾，并以键值对的形式列出，多个键值对之间用&符号隔开，语法为：?key1=value1&key2=value2。也可以将键值对填写在构建器的 Params 分页中，Postman 会自动帮你附加在 url 末尾。

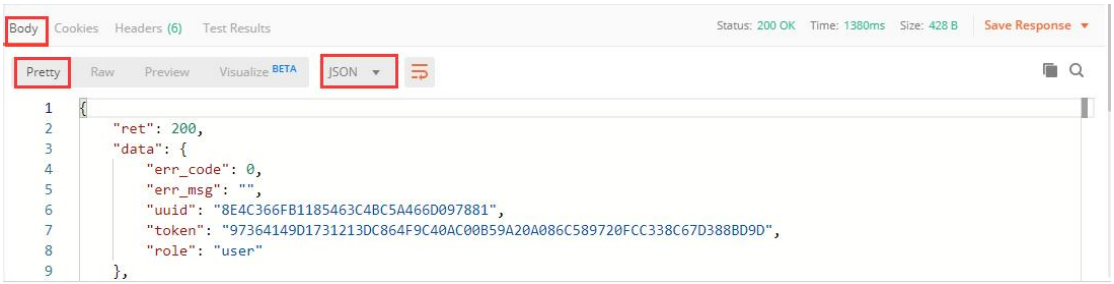
如果使用 post 请求，那么需要发送 body 数据，在 body 分页中选择对应的请求体数据类型即可，常用数据类型有：表单数据，raw，binary 等等。



- 查看响应

Postman 响应界面主要包括响应体、响应头、test 结果等内容。默认展示响应体分页。

响应体分页提供三种查看内容的方式：Pretty、Raw、Preview。通常情况下使用 Pretty 方式查看内容，它会自动格式化 Json 或 xml 数据，更便于查看。

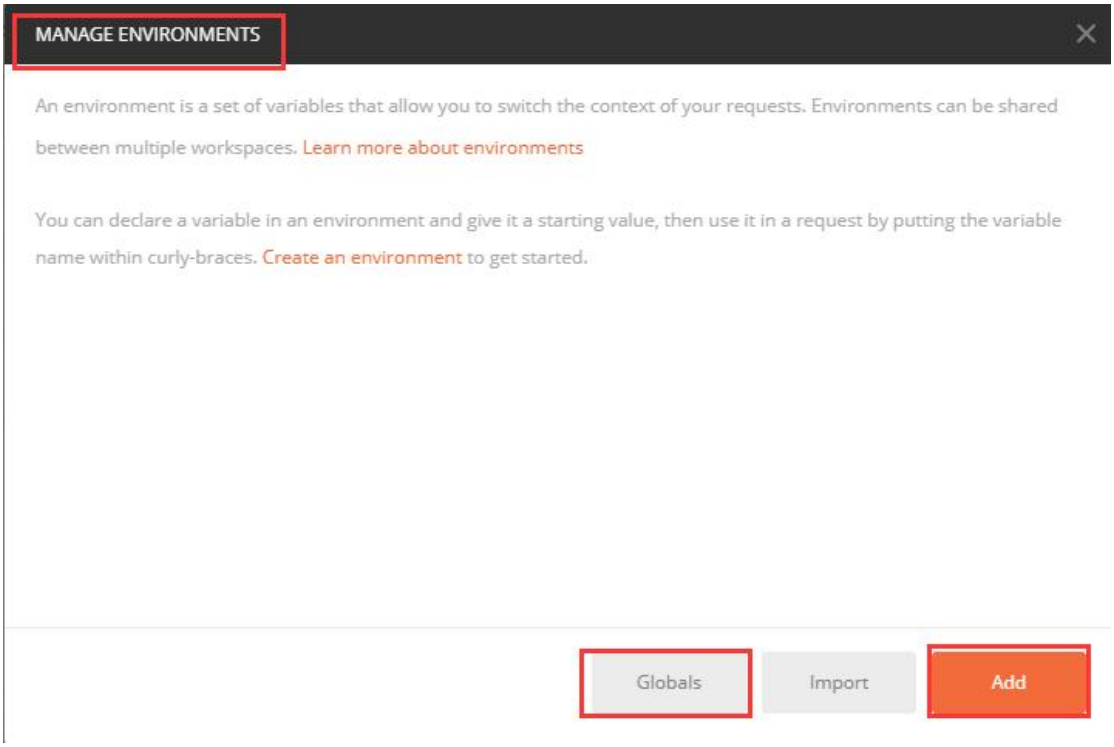


3.1.4 Postman 变量

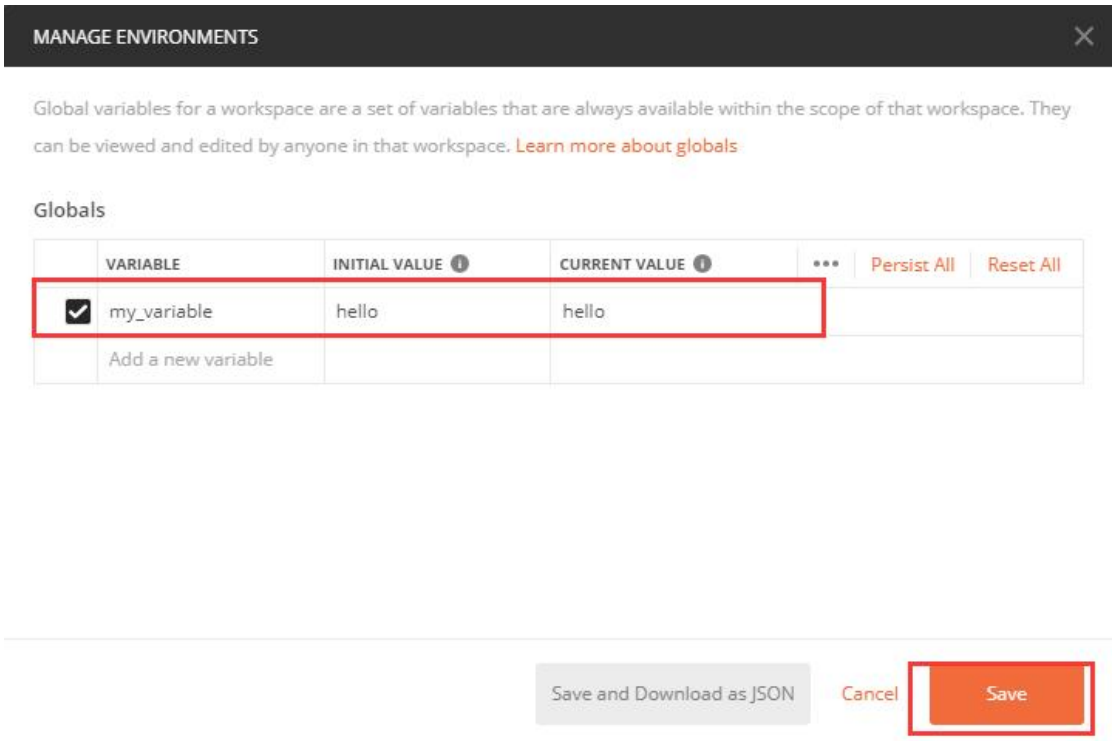
变量使你可以在请求和脚本中存储和重用值。通过将值存储在变量中，可以在整个集合和请求中引用它，如果需要更新值，则只需在一个位置进行更改。使用变量可以提高有效工作的能力，并最大程度地减少出错的可能性。

- 设置变量

单击 Postman 有上角的设置图标即可进入管理环境变量窗口，此处可以设置全局变量与环境变量。通常多环境测试时需要进行环境切换，需要设置环境变量。接口的一些通用参数设置需要色孩子全局变量。

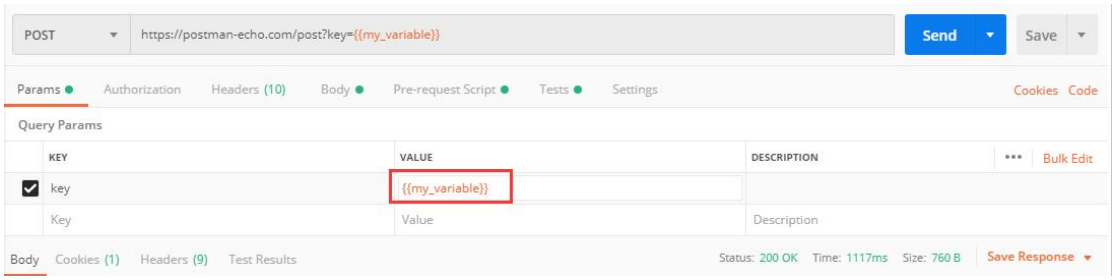


单击 Globals 按钮，即可进入全局变量设置界面，输入变量名和值保存即可。



- 访问变量

在 Postman 用户界面中使用双花括号来引用变量。语法格式：{{variable_name}}，当发送请求时，Postman 将自定解析该变量并替换成当前值。可以在 url、Params、 Headers、Authorization、 Body 中使用。



- 数据文件

数据文件是非常强大的方式，使用不同的测试数据来测试我们的接口，以检查它们是否在各种情况下都能正常运行。数据文件是“Collection Runner”中每个请求的参数。当我们运行测试集的时候，可以选择用例的数据直接从文件中读取，postman 默认将文件数据保存到 data 变量中，postman 支持 JSON 和 CSV 两种格式的文件。

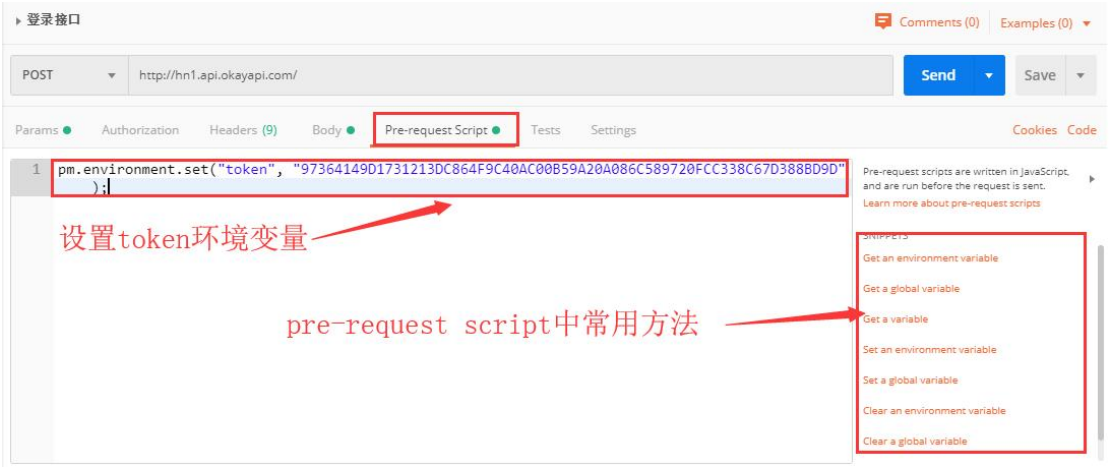
3.1.5 Postman 脚本

Postman 包含一个基于 Node.js 的强大运行时，该运行时可让你向请求和集合中添加动态行为。这样，你就可以编写测试套件，构建可以包含动态参数的请求，在请求之间传递数据等等。整个请求过程中有两个地方可以添加脚本：Pre-request scripts、Tests。

- Pre-request scripts

如果在请求之前需要额外做一些操作，可以将这些操作写在 Pre-request scripts 分页中，比如创建、获取变量的值等等。

例如：在请求之前先设置所需要的 token，可以先将 token 保存到变量中。



Pre-request script 中常用方法介绍：

pm.environment.get("variable ");	获取环境变量 variabl 的值
pm.globals.get("variable ");	获取全局变量 variabl 的值
pm.environment.set("variable ", " value");	设置环境变量 variabl 的值为 value

pm.sendRequest("https://postman-echo.com/get", function (err, response) {
console.log(response.json());
});
	请求接口：
	https://postman-echo.com/get 并打印响应体到控制台。

- **Tests&Test Results**

  构建请求的 Tests 分页中的脚本会在得到响应之后运行，通常用来判断得到的响应是否满足预期结果，比如判断响应状态码、响应体中是否包含某些数据等等。

tests 中常见方法介绍：

pm.test("Status code is 200", function () {
pm.response.to.have.status(200);
});
	判断响应状态码为 200

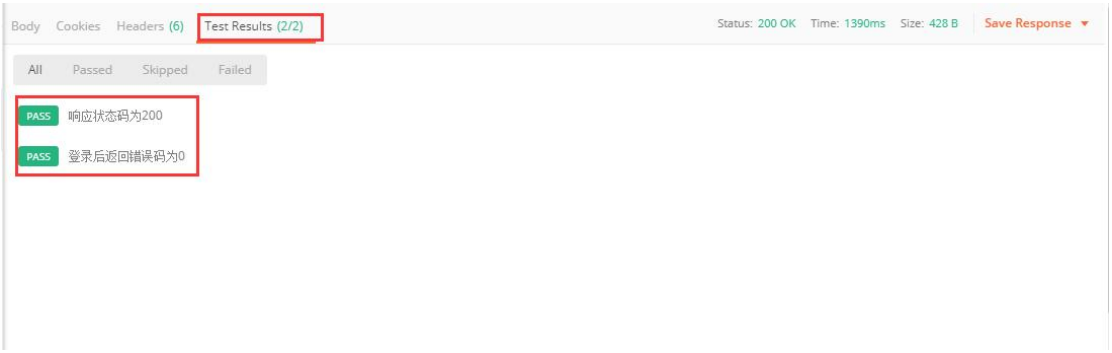
pm.test("Body matches string", function () {
pm.expect(pm.response.text()).to.include("string ");
});
	判断响应体中包含字符串 string

pm.test("Your test name", function () {	判断响应体中的某个 age 的

var jsonData = pm.response.json();
pm.expect(jsonData.age).to.eql(100);
});
	值为 100

pm.test("Response time is less than 200ms", function () {
pm.expect(pm.response.responseTime).to.be.below(200);
});
	判断响应时间少于 200 毫秒

响应查看器中的 Test Results 分页，展示的是请求后 tests 的结果。

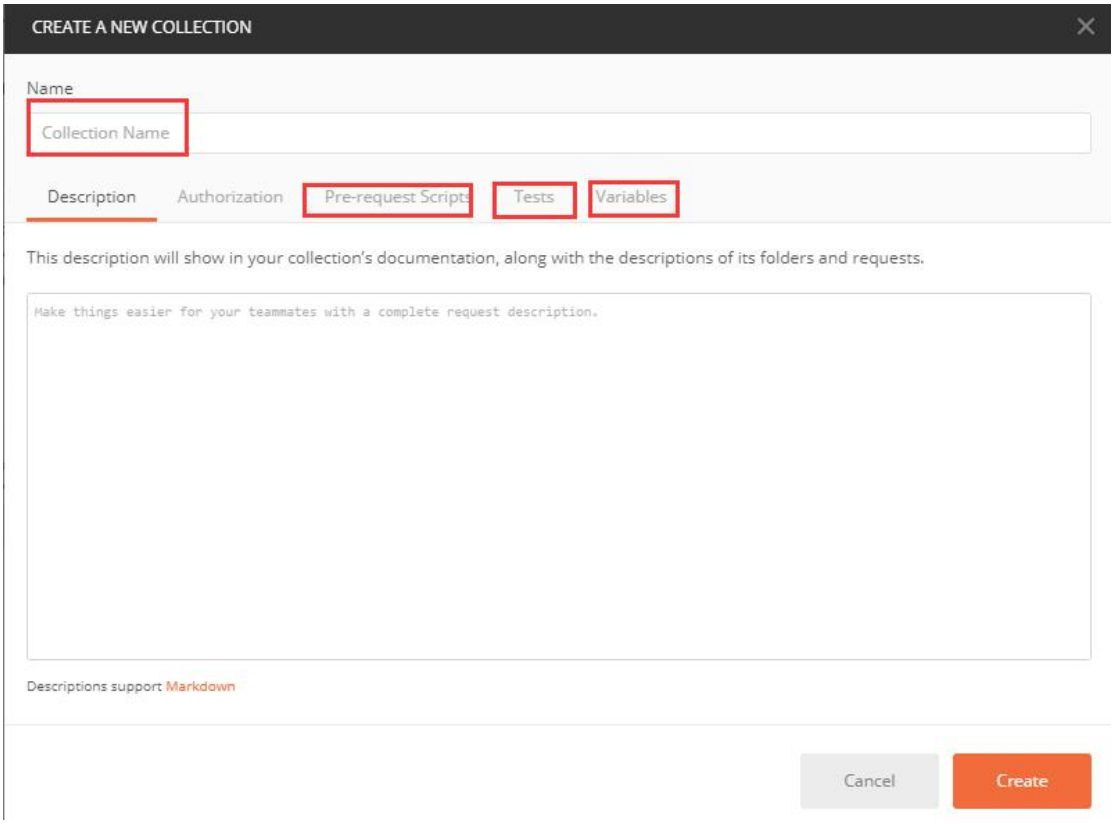


3.1.6 Postman 集合

集合可以将同一个接口或者相同模块的接口按需组织在文件夹中，方便管理请求。

- 创建集合

在侧边栏 Collections 分页中，点击 New Collection 按钮，进入创建新集合界面，输入集合名称、Pre-request scripts，Tests，Variables 后点击创建即可。

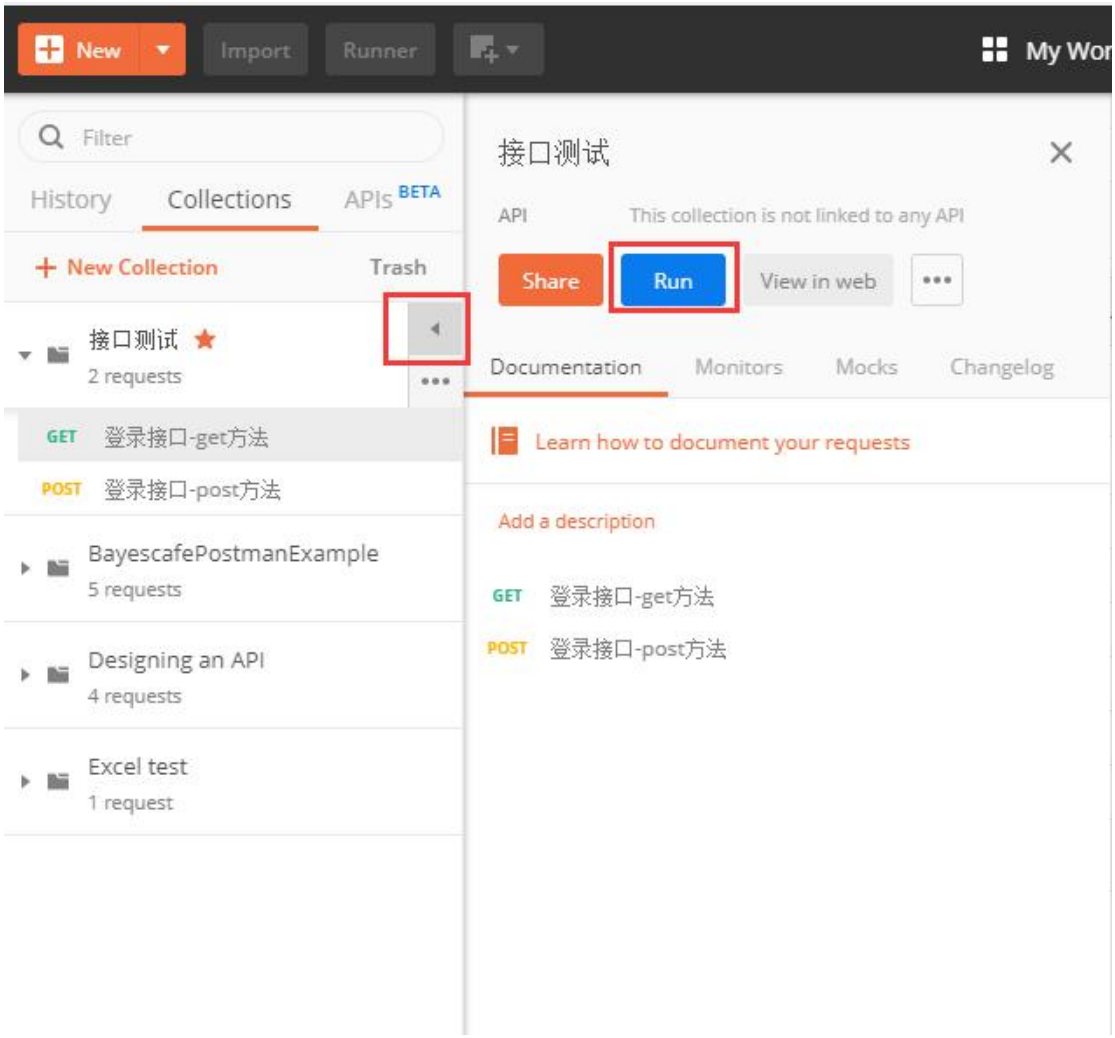


集合创建完成后，可以通过 History 分页中请求记录，或者构建器中的请求记录，将请求加入集合中。

- 运行集合

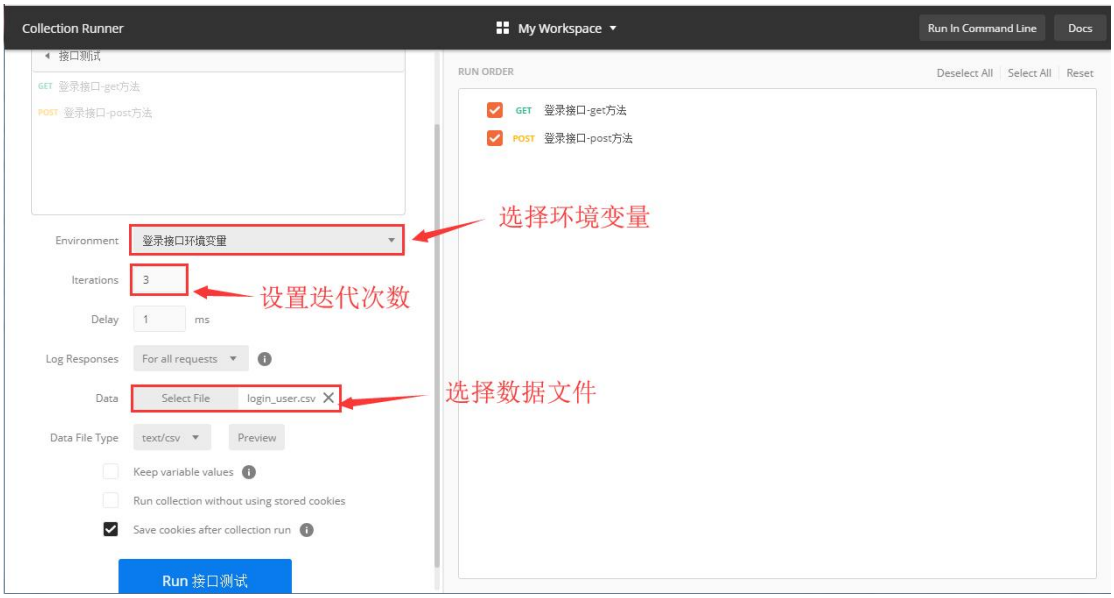
集合是一组请求，可以在对应的环境下作为一系列请求一起运行。

在集合名称右边点击展开，然后点击 Run 按钮可进入运行集合界面。



在运行集合界面，可以设置此次运行集合所需要的环境，运行次数，需要加载的数据，也可以选择只运行集合中的某一些请求。

例如：运行接口测试这个集合，所需要环境变量，迭代次数，数据文件直接在运行结合界面选择即可。



运行完成后会去到 Run Result 分页，此界面查看集合的执行过程，运行结果，也可以将结果导出。

3.1.7 Newman 执行

Newman 是 Postman 的命令行 Collection Runner。它可以直接从命令行运行和测试 Postman Collection。它在构建时考虑了可扩展性，因此您可以轻松地将其与连续集成服务器集成并构建系统。Newman 与 Postman 保持功能对等，并允许您以在 Postman 应用程序中的集合运行器中执行集合的方式运行集合。

- 安装 Newman

  首先下载(https://nodejs.org/zh-cn/download/)并安装 Node.js，Node.js 安装完成之后只需要一条命令即可安装 Newman。

```
npm install -g newman
```

- Newman 运行集合

  Newman 常用选项介绍：

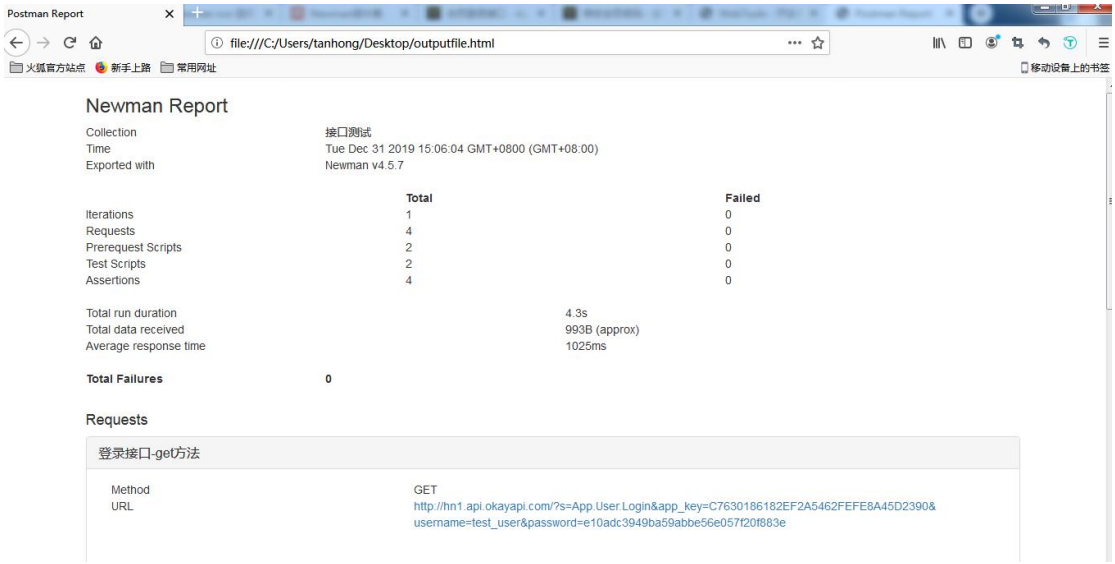
-n	指定迭代次数
-d	设置需要使用的数据文件，格式可以为 json,csv,txt
-e	指定环境变量文件
--reporters	可生成测试报告的格式
--reporter-htm-export	指定生成报告的名称
-h	查看帮助信息

Newman 完全支持集合的运行，将运行所需要的集合、环境、数据文件准备好执行一下命令即可，例如：

```
newman run mycollection.json -n 10 -d datas.csv - e environment.json --reporters html --reporter-html-export outputfile.html
```

使用 Newman 在命令行中执行 mycollection.json 集合，迭代 10 次，执行过程中集合需要的数据从 datas.csv 中读取，需要的环境变量从 environment.json 中读取，执行完成之后生成名为 outputfile.html 的 html 格式的报告。

测试报告界面如下，包含此次运行用例条数，断言次数，失败个数，测试时长等信息：



3.1.8 Postman 接口练习

举例：物流接口-查询快递接口

接口 URL：

http://www.kuaidi100.com/query?type=快递公司代号&postid=快递单号

接口参数：

参数 1 type=快递公司编码

申通="shentong" EMS="ems" 顺丰="shunfeng" 圆通="yuantong"

中通="zhongtong" 韵达="yunda" 天天="tiantian" 汇通="huitongkuaidi"

全峰="quanfengkuaidi" 德邦="debangwuliu" 宅急送="zhaijisong"

参数 2 postid=快递单号： 667543064565

实例返回结果：

http://www.kuaidi100.com/query?type= shunfeng &postid=667543064565

3.2. SoapUI 接口实战

3.2.1SoapUI 软件安装

SOAPUI 分开源版和商业版 SOAPUI pro，都可以进行使用，这里以开源版为例进行讲解；

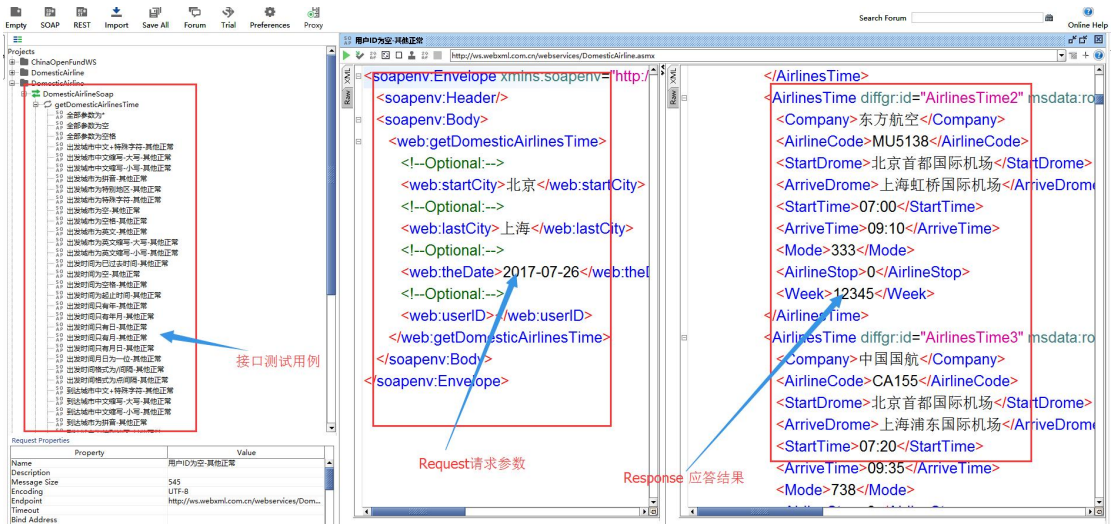
3.2.2 接口实例分析航班查询

- 接口名称：getDomesticAirlinesTime 获得航班时刻表 DataSet
- 输入参数：
  - startCity = 出发城市（中文城市名称或缩写、空则默认：上海）；
  - lastCity = 抵达城市（中文城市名称或缩写、空则默认：北京）；
  - theDate = 出发日期（String 格式：yyyy-MM-dd，如：2024-07-02，空则默认当天）；
  - userID = 商业用户 ID（免费用户不需要）；
- 返回数据：

  DataSet，Table(0)结构为 Item(Company)航空公司、Item(AirlineCode)航班号、Item(StartDrome)出发机场、Item(ArriveDrome)到达机场、Item(StartTime)出发时间、Item(ArriveTime)到达时间、Item(Mode)机型、Item(AirlineStop)经停、Item(Week)飞行周期（星期）
- 接口名称： getDomesticCity

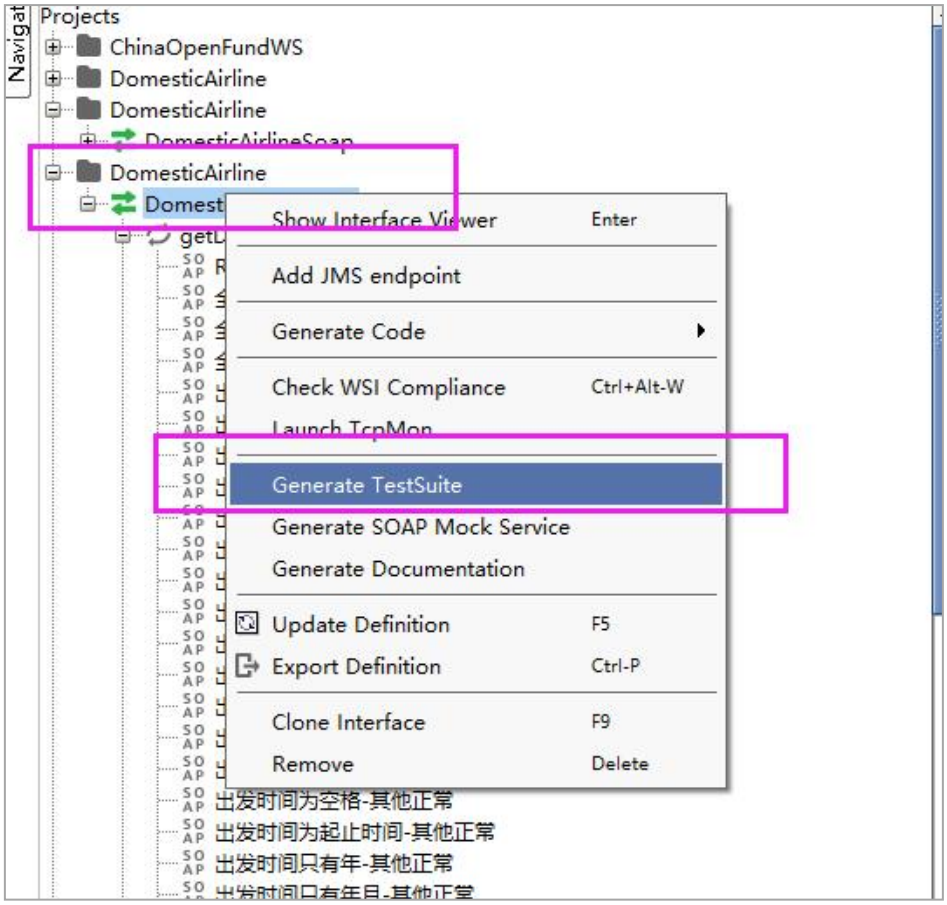
  获得这国内飞机航班时刻表 Web Services 支持的全部城市中英文名称和缩写 DataSet
- 输入参数：无；
- 返回数据：

  结构为 Item(enCityName)城市英文名称、Item(cnCityName)城市中文名称、Item(Abbreviation)缩写，按城市英文名称升序排列

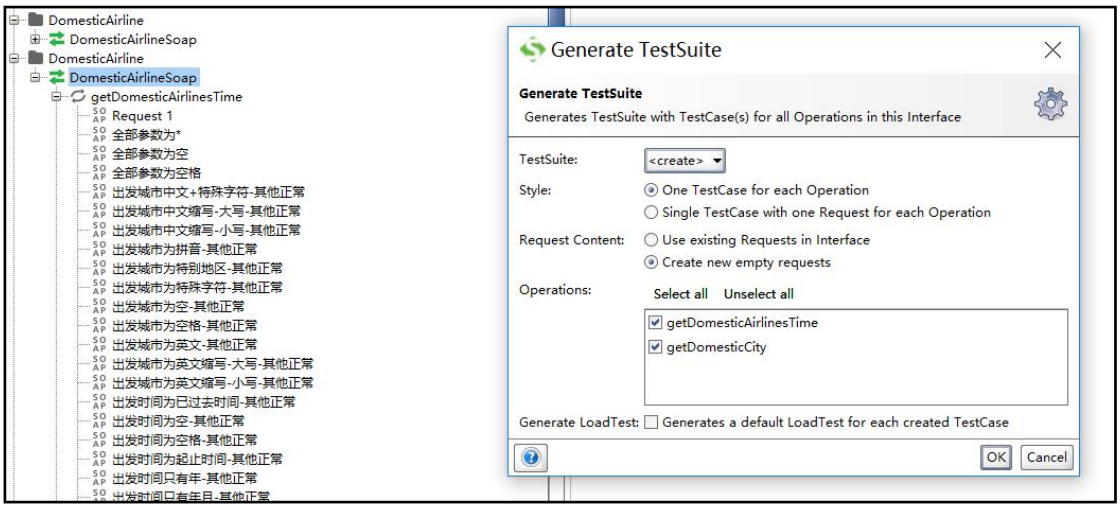


➢ **生成 TestSuit**

选中接口的服务名称，右键【Generate TestSuit】生成测试用例集合，如下图：

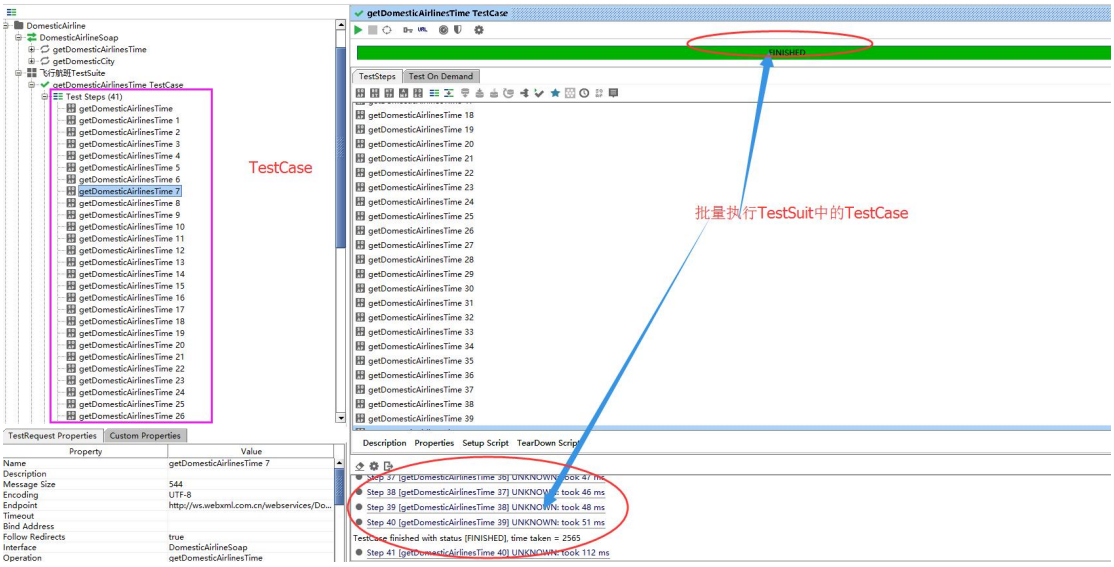


生成 TestSuit 界面，默认选中所有需要生成用例集合的接口，也可选择部分接口生成对应的接口用例集合，如下图：



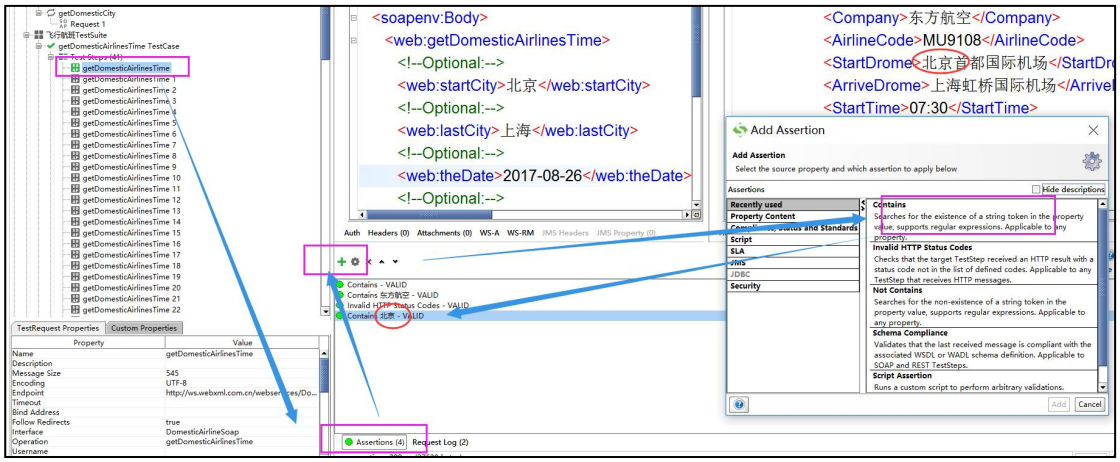
➢ 批量执行 TestSuit

在生成的测试用例集合中进行添加对应的 TestCase，若要执行某接口下的所有用例，则在 TestSuit 中选择对应的接口名，双击打开批量执行用例界面，如下图：



➢ Assertion 断言

若需要对每个用例进行验证，则可为对应的 TestCase 添加断言 Assertion，添加方法如下图，在对应的 case 中



3.3. Fiddler 抓包实战

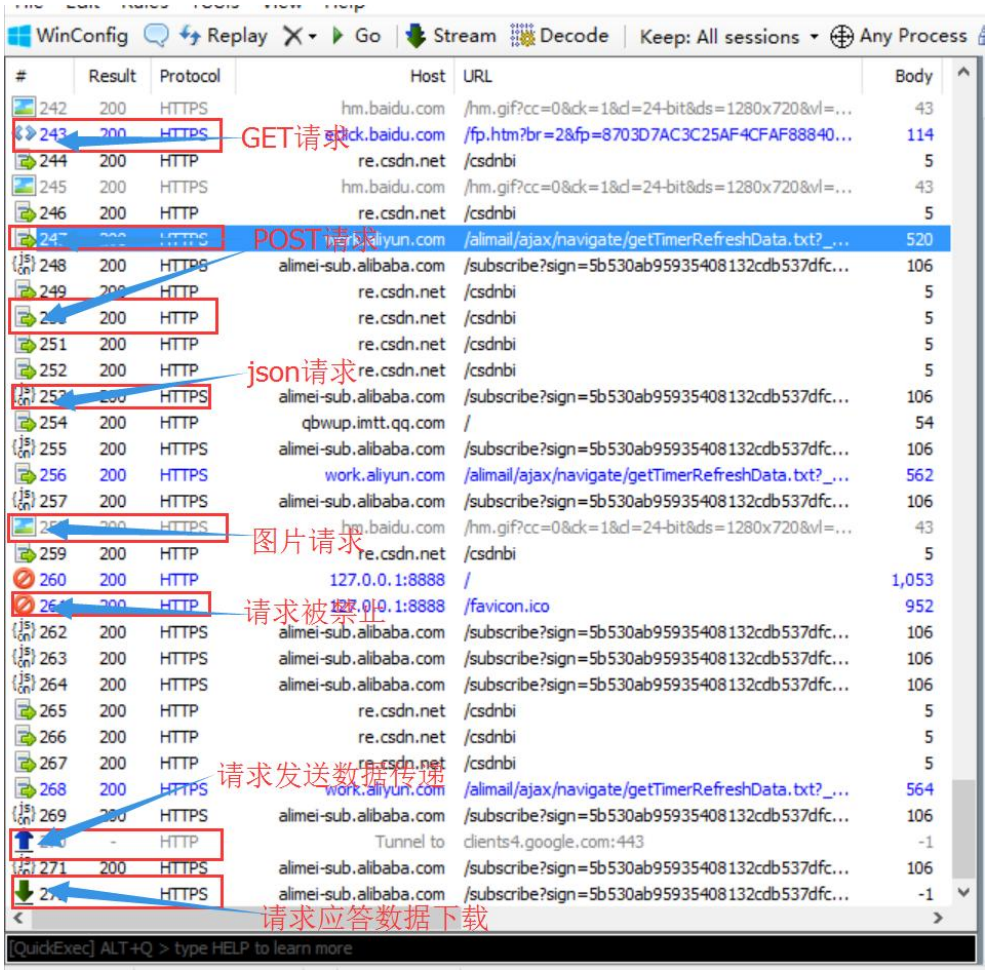
- 软件安装

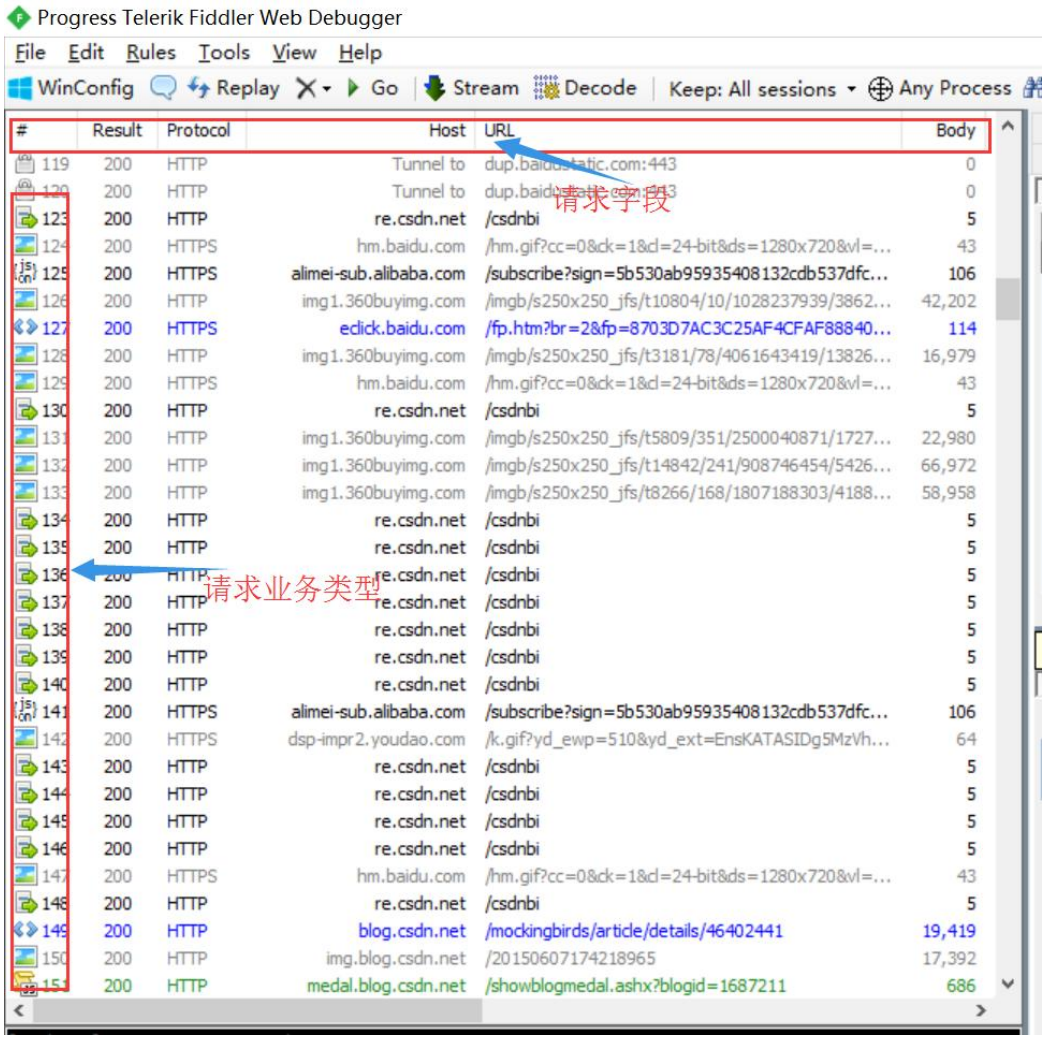
  直接下载 Fiddler4 安装即可；

  HTTPS 配置步骤：打开 Fiddler→Tools→Options→HTTPS→勾选 “Capture HTTPS CONNECTs” 和 “Decrypt HTTPS traffic”→安装证书；

  断点调试示例：选中请求→右键 “Break on Response”→修改响应报文后放行，验证前端表现。

3.3.1Fiddler 介绍


WinConfig
Replay
Go
Stream
Decode
Keep: All sessions
Any Process
GET请求
POST请求
json请求
图片请求
请求被禁止
请求发送数据传递
请求应答数据下载
[QuickExec] ALT+Q > type HELP to learn more


Progress Telerik Fiddler Web Debugger
File Edit Rules Tools View Help
WinConfig Replay Go Stream Decode Keep: All sessions Any Process
Result Protocol Host URL Body
请求字段
请求业务类型

字段含义：

名称	含义
#	抓取 HTTP Request 的顺序，从 1 开始，以此递增
Result	HTTP 状态码，如 200、302、403、404、502
Protocol	请求使用的协议，如 HTTP/HTTPS/FTP 等
Host	请求地址的主机名
URL	请求资源的位置
Body	该请求的大小
Caching	请求的缓存过期时间或者缓存控制值
Content-Type	请求响应的类型
Process	发送此请求的进程：进程 ID
Comments	允许用户为此回话添加备注
Custom	允许用户设置自定义值

常见图标含义

图标	含义
	请求已经发往服务器
	已从服务器下载响应结果
	请求从断点处暂停
	响应从断点处暂停
	请求使用 HTTP 的 GET 方法
	请求使用 HTTP 的 POST 方法
	请求使用 HTTP 的 HEAD 方法，即响应没有内容（Body）
	请求使用 HTTP 的 CONNECT 方法，使用 HTTPS 协议建立连接隧道
	响应是 HTML 格式
	响应是一张图片
	响应是脚本格式
	响应是 CSS 格式
	响应是 XML 格式
	响应是 JSON 格式
	响应是一个音频文件
	响应是一个视频文件
	响应是一个 SilverLight
	响应是一个 FLASH
	响应是一个字体
	普通响应成功

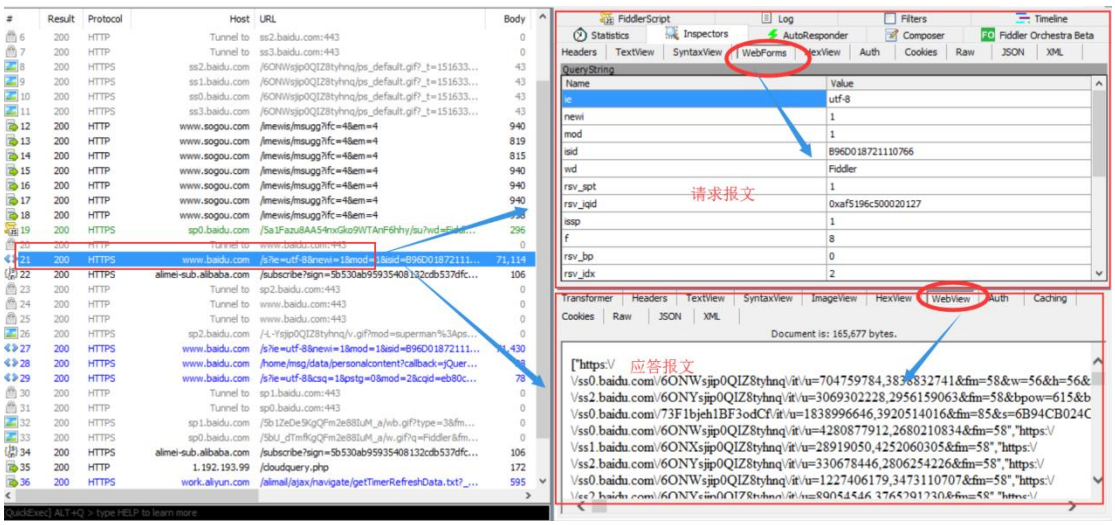
	响应是 HTTP/300、301、302、303 或 307 重定向
	响应是 HTTP/304（无变更）：使用缓存文件
	响应需要客户端证书验证
	服务端错误
	会话被客户端、Fiddler 或者服务端终止

3.3.2Fiddler 抓包报文分析

用浏览器访问百度页面，并搜索关键字“Fiddler”发送请求到百度服务器，搜到结果后显示在浏览器上，过程如下图：

如何分析捕获这次请求报文

浏览器发送的这次请求，被 Fiddler 捕获，在左侧找到 Host 为 www.baidu.com 的请求，双击后在右侧即显示该次报文的请求和应答报文，如下图：



4. 接口测试框架

- **为何要做接口自动化测试的持续集成？**
  - 接口相对稳定，改动少，比起 GUI 自动化测试来说性价比更加高些，减少脚本维护的成本；
  - 接口测试比较简单，一个规范的接口，测试只需要按照接口扩展测试用例就行，覆盖上较方便，易于后期接口的维护；
  - 利用持续集成的优势，可以在开发写好接口后直接做好接口测试的持续集成，部署到 jenkins 上，使问题尽早发现尽早解决；

- **测试框架的选择及技术选型**

  接口测试在某种程度上来说，还是属于灰盒测试，测试框架的选择与系统本身的语言并无太大的关系。如果测试框架与被测试系统的语言一致，有些基础文件可能还可以复用一下；如果不一致，则要重新的构建数据类型，重新的分析业务逻辑，这样对测试系统本身也是有极大的好处的。

  测试框架的选择，有一个最重要的点，就是要让大多数的测试人员都能接受的语言，且有大量的资料可以查阅的语言，这样才能快速的搭建框架及后期快速的编写接口测试用例，比如接口测试中 JAVA、Python 都用的比较多。

- **接口自动化框架框架介绍**

当下接口自动化是一种趋势，接口自动化框架的形成也是多样性的，可以借助现有的工具有效的组合或者自己开发出适合公司业务的自动化框架；

基于功能自动化框架： Selenium + Java+Testng+Maven+Jenkins

基于接口自动化框架： Postman +Newman +Jenkins

Jsoup+java+Jenkins

Jmeter+Ant+Jenkins

Python+requests+unittest+Jenkins

- 数据驱动 - ddt 做数据驱动

- 结构抽象 -
- 断言扩展 - unittest 框架自带断言体系
- 日志体系 - logger 模块做日志记录
- 测试报告 - BeautifulReport
- 持续集成 - Jenkins 持续集成

- **接口测试过程中常见问题及解决方案**

  1，接口测试或者自动化接口测试的过程中，上下游接口有数据依赖如何处理？

  2，依赖于第三方数据的接口如何进行测试？

  3，当一个接口出现异常时候，是如何分析异常的？

  4，如何分析一个 bug 是前端还是后端的？

  5，接口测试常见 bug（权限控制，逻辑校验，状态处理，边界溢出）。

  6，接口测试没有接口文档该如何处理？