<https://www.cnblogs.com/testdream/p/6184285.html>

实例：<https://www.cnblogs.com/wangxiaoqun/p/6924797.html>

HTTP接口功能测试是确保Web应用和H5应用**页面内容数据正确**的关键。

输入：根据接口描述构造不同的参数输入值

输出：json

**Web应用**（浏览器）：

1. 前台 - 使用JS框架加载后台返回的Json
2. 后台 - 提供接口调用返回Json对象

**H5应用**（移动端）：

动态获取内容的方式是采用ajax异步请求后台数据实时刷新，用GET/POST的HTTP请求后台接口，再将返回的数据（一般是json或xml格式）渲染在页面上。

1. 同步请求：发送方发出数据 后，等接收方发回响应以后才发下一个数据包的通讯方式。
   * 如果信息填写错误又要重新填写，再次发送请求，再次等待
2. 异步请求：发送方发出数据 后，不等接收方发回响应，接着发送下个数据包的通讯方式。
   * 如果邮箱填写重复了，请求已经发送给服务器，在等待返回结果期间还是可以操作页面，页面内容不会刷新不见，只是在结果返回时在邮箱输入框旁边给出提示信息，等到页面填写完成后一次性提交。

**HTTP接口功能测试的相关知识：**

1. http相关知识
   * HTTP URL的格式：http://host[:port][abs\_path]，其中port为空则使用缺省端口80。
   * 端口分为虚拟端口和物理端口。
     + 虚拟端口：计算机内部或交换机路由器内的端口，不可见。例如计算机中的80端口、21端口等
     + 物理端口：计算机背板的网口，交换机路由器集线器等端口，是可见的。
2. http协议的请求
   * 请求行：以一个方法符号开头，以空格分开，后面跟着请求的URI和协议的版本，格式如：Method Request-URI HTTP-Version CRLF，其中Method表示请求方法；Request-URI是一个统一资源标识符；HTTP-Version表示请求HTTP协议版本；CRLF表示回车和换行（除了作为结尾的CRLF外，不允许出现单独的CR或LF字符）。
     + 请求方法：
       - GET - 请求获取Request-URI所标识的资源
       - POST - 在Request-URI所标识的资源后附加新数据
   * 消息报头
   * 请求正文
3. http协议的响应
   * 状态行：状态行格式HTTP-Version Status-Code Reason-Phrase CRLF，其中HTTP-Version表示服务器HTTP协议的版本；Status-Code表示服务器发回的响应状态代码；Reason-Phrase表示状态代码的文本描述。
     + 状态代码：第一个数字定义了响应的类别，且有五种可能取值：
       - 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；
   * 消息报头
   * 响应正文：服务器返回的资源的内容

**JSON：**

两种结构：

1. “名称/值”对的集合，例如对象(object)，记录(record)，结构(struct)，字典(dictionary)，哈希表(hash table)，有键列表(keyed list)，或关联数组(associative array)
2. 值的有序列表，如数组(array)。举例：{ "people": [{ "firstName": "Brett", "lastName":"McLaughlin", "email": "aaaa" }, { "firstName": "Jason", "lastName":"Hunter", "email": "bbbb”}, { "firstName": "Elliotte", "lastName":"Harold", "email": "cccc" }]}

**实现方法**：

1. 使用Python语言驱动测试
2. 调用http接口采用pycurl模块
   * 安装pycurl模块
     + $ pip install pycur
     + Error:  Curl is configured to use SSL, but we have not been able to determine which SSL backend it is using. Please see PycURL documentation for how to specify the SSL backend manually.
     + Resolution:
       - download "<https://bootstrap.pypa.io/>“中的get-pip.py文件
       - $ cd  /Users/sherryzhang/Downloads
       - $  pip install pycurl==7.43.0
       - $ python2.7
       - Error:  ImportError: pycurl: libcurl link-time ssl backend (openssl) is different from compile-time ssl backend (none/other)
       - $ pip uninstall pycurl
       - $ export PYCURL\_SSL\_LIBRARY=openssl
       - $ pip install pycurl
3. 设置断言，对比实际返回结果和预期结果的正确性
4. 首次执行测试要人工检查输出的json文件是否正确，一旦正确将封存json文件，为后续回归测试的预期结果，如有错误要手工修正为预期文件。（只首次测试的时候才检查）
5. 按照逻辑写出test suite
6. 使用HTMLTestRunner模块生成测试报告

**python+requests接口自动化测试框架实例详解教程**

定义配置文件config.ini > 读取配置文件readConfig.py >

1. config.ini文件：类似于“仓库”所有一成不变的东西都可以放到这里
   * [EMAIL]
   * [HTTP]
   * [HEADERS]
   * [DATABASE]
2. readConfig.py文件
   * def get\_url() 在config.ini中没有找到URL分组???
   * os.path中的一些方法
     + \_\_file\_\_是指当前文件
     + os.path.split(path) 是将路径名称分成头和尾一对。
       - 返回值是元组，与列表相似，但元组的元素不能修改
       - 访问元组的元素，可以使用下标索引来访问元组中的值，下标索引从0开始
     + os.path.realpath(path)返回特定文件名的绝对路径
     + os.path.join(dir, filename) 将指定的路径与文件名拼接，用于找到指定的filename
   * def \_\_init\_\_ 是用来初始化实例的值，这些值一般要供其他方法调用，要求只初始化值，不要返回值（就是别用return）
   * data[:3] 取出data前三个字符，即0，1，2这三个索引位置的字符
   * BOM是用来判断文本文件是哪一种unicode编码的标记，UTF8的二进制字节是EF BB BF
   * codecs.open()可以避免中英文字符的编码问题
3. Log.py文件
   * def \_\_init\_\_(self)中定common的一些内容，如log的目录，命名格式以及log level等
   * 典型的日志记录的步骤：
     + 创建logger - logger可以看做是一个记录日志的人
     + 创建handler - handler可以看作是记录每个日志的一套规则，如formatter, level等
     + 定义formatter
     + 给handler添加formatter
     + 给logger添加handler
   * set resultPath - log文件所在的目录
   * log文件时间命名 - 日期格式转化为字符串格式的函数为datetime.now().strftime(“%Y%m%d%H%M%S")
   * logger.setLevel(loggin.INFO): 一般日志中调用最多的就是loggin.INFO
   * FileHandler是对文件类型输出日志，相对应的还有StreamHandler，针对控制台输出的日志
   * Formatter参数的格式化信息：
     + %(asctime)s - 字符串形式的当前时间
     + %(name)s - Logger的名字
     + %(levelname)s - 文件形式的日志级别
     + %(message)s - 用户输出的消息
   * 互斥锁概念：
     + python中引入互斥锁概念保证共享数据操作的完整性。每个对象对应于一个可称为“互斥锁”的标记，用来保证在任一时刻，只能有一个线程访问该对象。
     + mutex = threading.Lock()是定义了一个互斥锁变量
     + mutex.acquire() 争夺资源之前我们会抢占这把锁
     + mutex.release() 对资源使用完成之后会释放这把锁
   * 三个双引号：多行显示内容