# 一、Java+testng+HttpClient接口自动化测试框架

## 框架介绍

* HttpClient库： 使用HttpClient发送Http请求，并统一接口response返回值为String
* Java+Testng单元测试框架：构建测试框架，包括测试脚本构成，测试脚本的运行，测试脚本集成，测试报告生成
* 数据驱动设计：使用TestNG中的@DataProvider读取Excel中存储的自动化测试用例或者properties文件。
* 持续集成、持续交付、自动构建与测试

## **框架执行流程图**

测试执行的流程(代码层)：

1.测试执行的入口为TestNG的XML文件

2.获取测试用例的脚本，测试用例执行分为测试前(@BeforeTest)，测试中(@Test)，测试后(@AfterTest)

3.@BeforeTest：获取/解析测试数据(Excel文件)，接口关联参数化，Cookie装载等

4.@Test：初始化httpclient对象，执行请求，获取响应并转换成String等

5.@AfterTest：jsonpath检查点，对于接口正确/错误请求的逻辑处理

6.生成在线ExtentReports报告/日志

7.Email报告，测试结果通知测试相关人员

## **测试用例设计**

测试用例设计：

1.通过TestNG的@DataProvider获取Excel测试数据

2.将数据的表头与数据存储在map中，具体使用时方便

3.jsonpathPoint：jsonpath检查点，框架特点之一，极大的方便断言的编写

4.前后缀的使用：通过前后缀使用来区分统一作用的数据，如请求参数：params

是否运行/检查控制：通过编写测试用例时就可以控制该用例是否运行，是否检查

1. 不足之处：一个接口的测试数据对应一个Excel文件/Sheet表，后期数据量太大

# 二、如何搭建HTTPclient接口自动化测试框架

## HTTPclient插件的安装

在maven项目的pom.xml中引用HTTPclient包，如下

<dependencies>

<dependency>

<groupId>org.apache.httpcomponents</groupId>

<artifactId>httpclient</artifactId>

<version>4.5.6</version>

</dependency></dependencies>

添加完后，maven项目会自动加载相关依赖包

## 创建一个get请求

**public** **class** MyHttpClient {

@Test

**public** **void** atest() **throws** IOException{

//用来存放测试结果

String result;

//创建一个get实例，并指定请求url

HttpGet get = **new** HttpGet("http://www.baidu.com");

//创建一个client对象，是用来执行get方法的

CloseableHttpClient client = HttpClients.*createDefault*();

//用客户端执行get实例，并把响应结果保存在response对象中

HttpResponse response =client.execute(get);

//response.getEntity()用来获取整个响应的实例，即获取整个响应内容

//EntityUtils对象是org.apache.http.util下的一个工具类，用官方的解释是为HttpEntity对象提供的静态帮助类

//把响应内容转成字符串，转换时使用编码为utf-8

result = EntityUtils.*toString*(response.getEntity(), "utf-8");

System.***out***.println(result);

}

}

## 创建一个post请求

@Test(description = "post请求——Form表单形式(数据格式)")

**public** **void** btest() **throws** IOException, JSONException {

String url = "https://api.apishop.net/communication/phone/getLocationByPhoneNum";

//用哈希图准备请求头部信息

HashMap<String, String> hashHead = **new** HashMap<String, String>();

hashHead.put("Content-Type", "application/x-www-form-urlencoded");

//用NameValuePair的list来添加请求主体参数

List params = **new** ArrayList();

params.add(**new** BasicNameValuePair("apiKey", "nMke6NK29c40b1d\*\*\*\*\*\*b3eec8aa0808389b16c4"));

params.add(**new** BasicNameValuePair("phoneNum", "1861196136"));

//传参发送post请求并接收反馈

String str=(String) sendPost.*sendPost*(url,hashHead,params);

System.***out***.println("post请求的返回结果是："+str);

JSONObject jsonStr = **new** JSONObject(str);

String jsonStr1 =jsonStr.getString("statusCode"); //取出json数组中的某一个属性

System.***out***.println("取出json数组中的某一个属性是："+jsonStr1);

JSONArray jsonArray = jsonStr.getJSONArray("result");

System.***out***.println("post请求的返回的内容包含的数组是："+jsonArray);

}

@Test(description = "post请求——Json形式(数据格式)")

**public** **void** ctest() **throws** IOException, JSONException {

String url = "https://api.apishop.net/communication/phone/getLocationByPhoneNum";

//用哈希图准备请求头部信息

HashMap<String, String> hashHead = **new** HashMap<String, String>();

hashHead.put("Content-Type", "application/json");

//添加请求主体参数

JSONObject params = **new** JSONObject();

params.put( "wsf",GetProperty.*getProperty*("param01"));//参数化

params.put("TENANT\_ID", "123");

params.put("NM", "张三");

params.put("BRTH\_DT", "1983-01-20");

params.put("GND\_CODE", "1");

JSONArray params2 = **new** JSONArray();

JSONObject param3 = **new** JSONObject();

param3.put("DOC\_TP\_CODE", "10100");

param3.put("DOC\_NBR", "100200198301202210");

param3.put("DOC\_CUST\_NM", "test");

params2.put(param3);

params.put("DOCS", params2);

String json = params.toString();

System.***out***.println("json格式post请求的主体内容是:"+json);

//传参发送post请求并接收反馈

String str=(String) sendPost.*sendPost\_json*(url,hashHead, json);

System.***out***.println("post请求的返回结果是:"+str);

}

/---------------------------处理表单格式的post请求

public class sendPost {

//获取post请求的反馈

public static String sendPost(String url, HashMap<String, String> headers,List params) throws ClientProtocolException, IOException{

//创建post请求对象 //NameValuePair 简单名称值对节点类型：这个代码多处用于Java像url发送Post请求。在发送post请求时用该list来存放参数

HttpPost post= new HttpPost(url);

//设置头部信息

Set<String> set = headers.keySet();

for(Iterator<String> iterator = set.iterator(); iterator.hasNext();){

String key = iterator.next();

String value = headers.get(key);

post.addHeader(key, value);

}

//设置请求主体格式

post.setEntity(new UrlEncodedFormEntity(params, "UTF-8"));

CloseableHttpClient client = HttpClients.createDefault();

HttpResponse Response = (HttpResponse) client.execute(post);

if (Response.getStatusLine().getStatusCode() == 200) {

HttpEntity entity = Response.getEntity();

String json = EntityUtils.toString(entity, "UTF-8");

return json;

}

else {

return null;

}

}

//---------------------------处理json格式的post请求

public static String sendPost\_json(String url,HashMap<String, String> headers,String strjson) throws ClientProtocolException, IOException{

//创建post请求对象

HttpPost Post = new HttpPost(url);

//设置头部信息

Set<String> set = headers.keySet();

for(Iterator<String> iterator = set.iterator(); iterator.hasNext();){

String key = iterator.next();

String value = headers.get(key);

Post.addHeader(key, value);

}

//设置请求主体json格式

StringEntity poststr=new StringEntity(strjson);

Post.setEntity(poststr);

CloseableHttpClient client = HttpClients.createDefault();

HttpResponse Response = (HttpResponse) client.execute(Post);

if (Response.getStatusLine().getStatusCode() == 200) {

HttpEntity entity = Response.getEntity();

String json = EntityUtils.toString(entity, "UTF-8");

return json;

}

else {

return null;

}

}

}

## 接口-数据驱动（参数化）

## **（1）利用excel文件**

# 三、Pycharm +unitest+suds接口自动化

1. 安装pycharm
2. 在pycharm里导入unitest 模块
3. 在pycharm里导入suds模块
4. 可以使用pip install suds 安装第三方库，默认安装在C盘Unitest框架
5. Unitest框架

unittest测试框架：它是python自带的单元测试框架，用来管理自动化测试用例

调用unittest方法，添加测试套件、执行用例等：

1、TestCase测试用例：测试方法，以test\_开头，否则不会执行

2、TestFixture测试固件：运行前后数据的处理，setUp()和tearDown()方法

3、TestSuite测试套件：通过addTest()的手动方法，将测试用例放在测试套件中，批量执行测试方法；也可以使用TestLoader自动添加测试用例到测试套件中

4、TextTestRunner：执行测试方法，把测试套件跑起来

5、TextTestResult：保存测试结果

6、TestReport：基于html的测试报告

注意：unitest不支持多线程，testng支持

# 四、Pycharm +unitest+request接口自动化

1. 安装pycharm
2. 在pycharm里导入unitest 模块
3. 在pycharm里导入request模块
4. 可以使用pip install selenium 安装第三方库

import flask  
from flask import request

## 框架介绍

* Requests：Requests模块发送给予http的网络请求，主要包含了post，get，PUT， DELETE， HEAD
* python+unittest单元测试框架：构建测试框架，包括测试脚本构成，测试脚本的运行，测试脚本集成，测试报告生成（HTMLTestRunner）
* 数据驱动设计：excel文件
* 持续集成、持续交付、自动构建与测试