

基于 Python+Django+Easyui 实现的接口自动化测试框架 V2.0

by:授客 QQ: 1033553122

博客: <https://www.cnblogs.com/shouke/>

欢迎加入软件测试交流 QQ 群: 7156436

目录

1、	开发环境.....	2
2、	框架功能简介.....	5
3、	服务端部署.....	6
a)	Apache 配置	6
	检查 Apache 版本是否正确.....	7
	修改 Apache 配置	7
	启动 Apache	8
	添加 mod_wsgi.so 模块.....	8
	再次修改 Apache 配置.....	9
b)	Django 配置	9
	Django 访问 IP 配置	9
	创建库配置.....	9
	修改 wsgi.py	10
c)	数据库的创建.....	10
d)	创建表.....	10
	执行初始化 sql.....	10
e)	重启 Apache 并启动 Django 应用	11
	浏览器验证.....	11
4、	客户端部署.....	11
a)	数据库配置.....	11
b)	https SLL、TSL 版本配置	12
c)	日志配置.....	12
5、	大致操作流程.....	12
a)	新增项目	12
b)	系统配置.....	12
c)	新增用例.....	13
d)	新增 API 计划	13
e)	关联 API 测试用例	13
f)	新增并执行“运行计划”	13
g)	查看测试报告	13
6、	用例步骤填写规范.....	13
a)	接口请求.....	13
	① 执行操作.....	13

② 请求头.....	14
③ URL/SQL	14
④ 输入参数.....	15
⑤ 检查响应.....	16
⑥ 校验规则及校验模式.....	16
⑦ 输出.....	16
⑧ 协议.....	16
⑨ 主机地址.....	16
⑩ 端口.....	16
b) 操作数据库.....	16
① 执行操作.....	16
② URL/SQL	17
③ 输入参数.....	17
④ 校验规则及校验模式.....	17
⑤ 输出.....	17
c) 执行函数.....	17
① 操作对象.....	17
② 输入参数.....	17
③ 校验规则及校验模式.....	17
④ 输出.....	17
d) 关于用例及步骤运行顺序说明.....	18
e) 导入用例步骤.....	18
7、 参数化.....	20
① 全局变量.....	20
② 局部变量.....	21
③ 变量的引用.....	21
④ 变量使用范围.....	21
⑤ 插件函数.....	23
8、 调试.....	24
① 项目级别的调试.....	24
② 用例(套件)级别的调试	24
9、 源码下载.....	25

1、 开发环境

服务端:

win7 64\Windows Server 2008 R2 x64

数据库

MariaDB 5.5.45, MySQL Server 5.7

Django-1.11.4.tar.gz

下载地址:

<https://pan.baidu.com/s/1hsc1V5y>

官方下载地址:

<https://www.djangoproject.com/download/>

Easyui 1.5.3

下载地址:

<https://pan.baidu.com/s/1i6E597R>

下载地址:

<http://www.jeasyui.com/download/v153.php>

Jquery-3.2.1.min.js

下载地址:

<https://pan.baidu.com/s/1bqP7jeZ>

下载地址:

<http://www.jq22.com/jquery/jquery-3.2.1.zip>

jquery-easyui-datagrid-dnd

下载地址:

<http://www.jeasyui.net/demo/193.html>

下载地址:

<https://pan.baidu.com/s/1EtiFA1W3-u3T5USCLNE0eQ>

mysqlclient-1.3.6-cp34-none-win_amd64.whl

下载地址:

<https://pan.baidu.com/s/1o9YDMrg>

官方下载地址: <https://pypi.python.org/pypi/mysqlclient/1.3.4>

参考链接:

<https://docs.djangoproject.com/en/1.11/ref/databases/>

httpd-2.4.23-win64.zip

下载地址 1: <https://www.apachelounge.com/download/VC10/>

下载地址 2: <https://pan.baidu.com/s/1hsclV5y>

Apache24-win64-VC10|mod_wsgi-py34-VC10.so

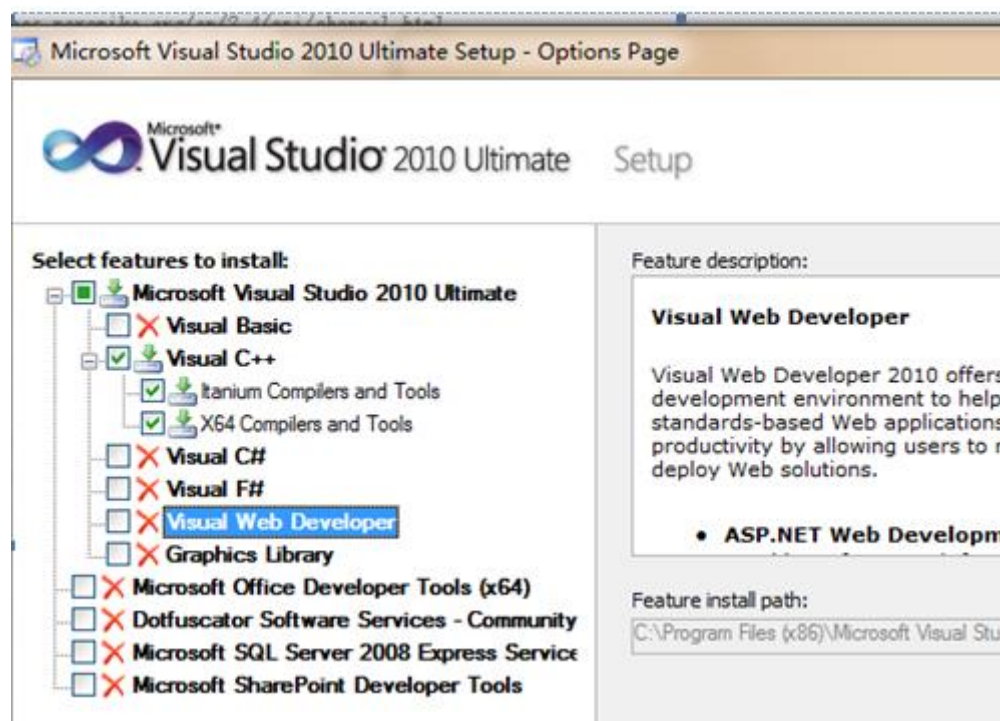
下载地址: <https://pan.baidu.com/s/1boEF7VD>(链接包含不同版本的集合包)

redis-2.10.6.tar.gz

下载地址: <https://pan.baidu.com/s/1gWEu4UFFXt9wggnbjlI1kQ>

官方下载地址: <https://pypi.python.org/pypi/redis>

软件包依赖 VS2010



客户端:

mysql-connector-python-2.0.5-py3.4.msi

mysql-connector-python-2.1.7-py3.4-windows-x86-64bit.msi

下载地址: <https://pan.baidu.com/s/1c3XCnGO>

官方下载地址: <http://dev.mysql.com/downloads/connector/python/>

chardet-2.3.0

下载地址 1: <https://pypi.python.org/pypi/chardet/>

下载地址 2: <http://pan.baidu.com/s/1nu7XzjN>

xlrd-1.1.0.tar.gz

下载地址: https://pan.baidu.com/s/1SgUwr_b3GTzWERjEuiAXAQ

下载地址: <https://pypi.python.org/pypi/xlrd/>

xlwt-1.3.0.tar.gz

下载地址: <http://pan.baidu.com/s/1nvaS43Z>

下载地址: <https://pypi.python.org/pypi/xlwt/>

公用部分:

python 3.4.0

setuptools-29.0.1.zip

下载地址: <http://pan.baidu.com/s/1mhMSAkK>

官方下载地址: <https://pypi.python.org/pypi/setuptools#downloads>

PyCharm 4.0.5

2、 框架功能简介

- 1、采用 **WEB** 页面, 可灵活管理测试项目, 测试计划, 测试用例(支持模块层级化组织用例, 支持自由拖拽分组), 运行计划
 - 2、支持一键复制单个测试用例, 也支持一键复制用例步骤, 禁用、启用用例步骤等(禁用的步骤将不被运行)
 - 3、通过项目关联计划, 计划关联用例的设计方式, 支持一次运行单个、多个测试计划
 - 4、支持调试模式, 支持按运行计划调试, 也支持单个用例、单个测试用例套件的调试运行
 - 5、支持 **HTTPS**, **HTTP**, **WebService** 协议; 支持 **POST**, **GET** 方法; 支持 **JSON**, 非 **JSON** 数据格式的请求, 支持多种形式的数据库级别的数据校验
 - 6、纯界面化, 无须写代码就可以实现如下操作:
 - a) 自定义变量存储 **web** 服务器、数据库服务器返回请求/查询结果
 - b) 根据自定义模式对 **web** 服务器返回结果进行自动校验, 支持多种模式的校验, 包含字符串, 不包含字符串, 键值提取, 包含成员, 不包含成员, 匹配/不匹配正则表达式, 完全匹配列表/元组/集合/字典
 - c) 根据界面输入的 **sql** 语句, 执行 **sql** 查询/更新操作, 针对只对返回单条记录的 **sql** 查询, 还支持对查询结果进行提取, 保存
 - d) 支持 **url**, 请求头, 输入参数的动态参数化, 支持全局动态参数, 非全局动态参数(如存储某个接口返回结果的自定义变量)
 - 7、针对脚本中已经支持的常见协议及常用数据格式, 且不需对接口执行结果进行数据库级别的逻辑校验, 支持界面直接增加用例而不需要改动脚本代码, 即不会编码的人也可以使用本框架
 - 8、支持不同编码(**utf8,ascii,gb2312**)的返回结果, 且可自由扩展
 - 9、支持文件、控制台的日志打印, 可分别控制开关
 - 10、可集成 **Jenkins** 自动运行脚本
- 参考文章:

“[为 Jenkins 添加 Windows Slave 远程执行 python 项目脚本](#)”

3、服务端部署

a) Apache 配置

项目文件结构

```

D:\AutotestPlatform 的目录
2017/10/24 03:07 <DIR> .
2017/10/24 03:07 <DIR> ..
2017/10/24 02:52 <DIR> .idea
2017/10/24 02:52 <DIR> AutotestPlatform
2017/09/19 23:35      2,598 layout.html
2017/08/06 23:07      836 manage.py
2017/10/24 03:07 <DIR> static
2017/08/07 22:43 <DIR> templates
2017/10/24 02:52 <DIR> website
2017/10/23 03:04 <DIR> __pycache__
                2 个文件      3,434 字节
                8 个目录 76,560,097,280 可用字节

D:\AutotestPlatform>dir AutotestPlatform
驱动器 D 中的卷是 常用软件
卷的序列号是 AA16-52FC

D:\AutotestPlatform\AutotestPlatform 的目录
2017/10/24 02:52 <DIR> .
2017/10/24 02:52 <DIR> ..
2017/10/25 01:34    4,410 settings.py
2017/09/06 23:39     919 urls.py
2017/10/25 01:54     572 wsgi.py
2017/08/06 23:07        0 __init__.py
2017/10/25 01:55 <DIR> __pycache__
                4 个文件      5,901 字节
                3 个目录 76,560,097,280 可用字节

D:\AutotestPlatform>dir website
驱动器 D 中的卷是 常用软件
卷的序列号是 AA16-52FC

D:\AutotestPlatform\website 的目录
2017/10/24 02:52 <DIR> .
2017/10/24 02:52 <DIR> ..
2017/08/06 23:19      66 admin.py
2017/08/06 23:33      93 apps.py
2017/10/21 14:27    3,480 common_views.py
半.

```

解压 httpd-2.4.23-win64.zip, 取出其中的目录(例中 Apache24), 放到目标路径(不能有空格等), 例中

D:/Apache24

检查 Apache 版本是否正确

```
cd /d D:/Apache24/bin
```

```
httpd.exe -V
```

```
Server version: Apache/2.4.23 (Win64)
```

.....

修改 Apache 配置

打开 conf/httpd.conf 文件, 编辑, 修改服务器根目录

修改 SRVROOT "/Apache24" 为 SRVROOT "d:/Apache24"

修改 ServerRoot "c:/Apache24" 改成 ServerRoot "d:/Apache24"

然后查找所有的 "c:/Apache24", 全部改成 "d:/Apache24"

修改监听端口(可选, 根据实际需要)

```
Listen 80 改成 Listen 8000
```

修改服务器名称(建议)

```
#ServerName www.example.com:80 改成 ServerName 192.168.1.101:80
```

注: 这里我没有注册域名, 直接改成了本机 ip

去掉#注释, 打开访问日志, 当然也可以保持注释状态, 服务器磁盘写操作(推荐)

```
CustomLog "logs/access.log" common
```

修改#LoadModule rewrite_module modules/mod_rewrite.so 为如下:

```
LoadModule rewrite_module modules/mod_rewrite.so
```

找到如下配置

```
<Directory />
```

```
    AllowOverride none
```

```
    Require all denied
```

```
</Directory>
```

修改为

```
<Directory />
```

```
    AllowOverride ALL
```

```
    Require all granted
```

```
</Directory>
```

说明: 配置更改, 以防止出现如下情形:

Forbidden

You don't have permission to access / on this server.

启动 Apache

1) httpd.exe -k install -n Apache2.4

Installing the 'Apache2.4' service

The 'Apache2.4' service is successfully installed.

Testing httpd.conf....

Errors reported here must be corrected before the service can be started.

注: install:把 Apache 注册为 Windows 服务, 反之 uninstall, -n 接服务名称

2) httpd.exe -k start

注: start 启动, stop 停止

浏览器访问



It works!

添加 mod_wsgi.so 模块

把 mod_wsgi-py34-VC10.so 重命名为 mod_wsgi.so, 放入 D:\Apache24\modules 目录下。

再次修改 Apache 配置

打开 `conf/httpd.conf` 文件, 编辑, 在末尾添加一下内容:

```
LoadModule wsgi_module modules/mod_wsgi.so

WSGIScriptAlias / D:/AutotestPlatform/AutotestPlatform/wsgi.py
WSGIPythonHome "D:/Program Files (x86)/python34"
WSGIPythonPath D:/AutotestPlatform/AutotestPlatform/website

Alias /static/ D:/AutotestPlatform/website/static/
<Directory D:/AutotestPlatform/website/static>
    Require all granted
</Directory>

<Directory D:/AutotestPlatform/AutotestPlatform/website/>
<Files wsgi.py>
    Require all granted
</Files>
</Directory>
```

b) Django 配置

Django 访问 IP 配置

修改应用的 `settings.py`(例中为 `D:\AutotestPlatform\AutotestPlatform\settings.py`), 编辑, 找到 `ALLOWED_HOSTS` 修改为如下值, 其中 `192.168.1.101` 是 Django 所在主机 ip, 也就是客户端浏览器访问用的 IP

```
ALLOWED_HOSTS = ['localhost', '127.0.0.1', '192.168.1.101']
```

创建库配置

修改应用的 `settings.py`(例中为 `D:\AutotestPlatform\AutotestPlatform\settings.py`)

```

DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.mysql',
        'NAME': 'testplatform',
        'USER': 'testacc',
        'PASSWORD': 'test1234',
        'HOST': '192.168.1.102',
        'PORT': '3306',
        'OPTION': {
            'init_command': 'SET default_storage_engine=INNODB'
        }
    }
}

```

如上, 设置蓝色部分的 key 值, 分别表示数据库用户名, 密码, 服务器 ip, 端口

修改 wsgi.py

如下, 新增带背景色内容

```

from django.core.wsgi import get_wsgi_application

import sys

sys.path.append('D:/AutotestPlatform/AutotestPlatform')

sys.path.append('D:/AutotestPlatform')

os.environ.setdefault("DJANGO_SETTINGS_MODULE", "AutotestPlatform.settings")

application = get_wsgi_application()

```

c) 数据库的创建

```
CREATE DATABASE IF NOT EXISTS `testplatform` DEFAULT CHARACTER SET utf8;
```

d) 创建表

```

cd /d D:\AutotestPlatform

python manage.py makemigrations website

python manage.py migrate

```

执行初始化 sql

详情见“初始化 sql.sql”

e) 重启 Apache 并启动 Django 应用

```
D:\Apache24\bin>httpd.exe -k stop
```

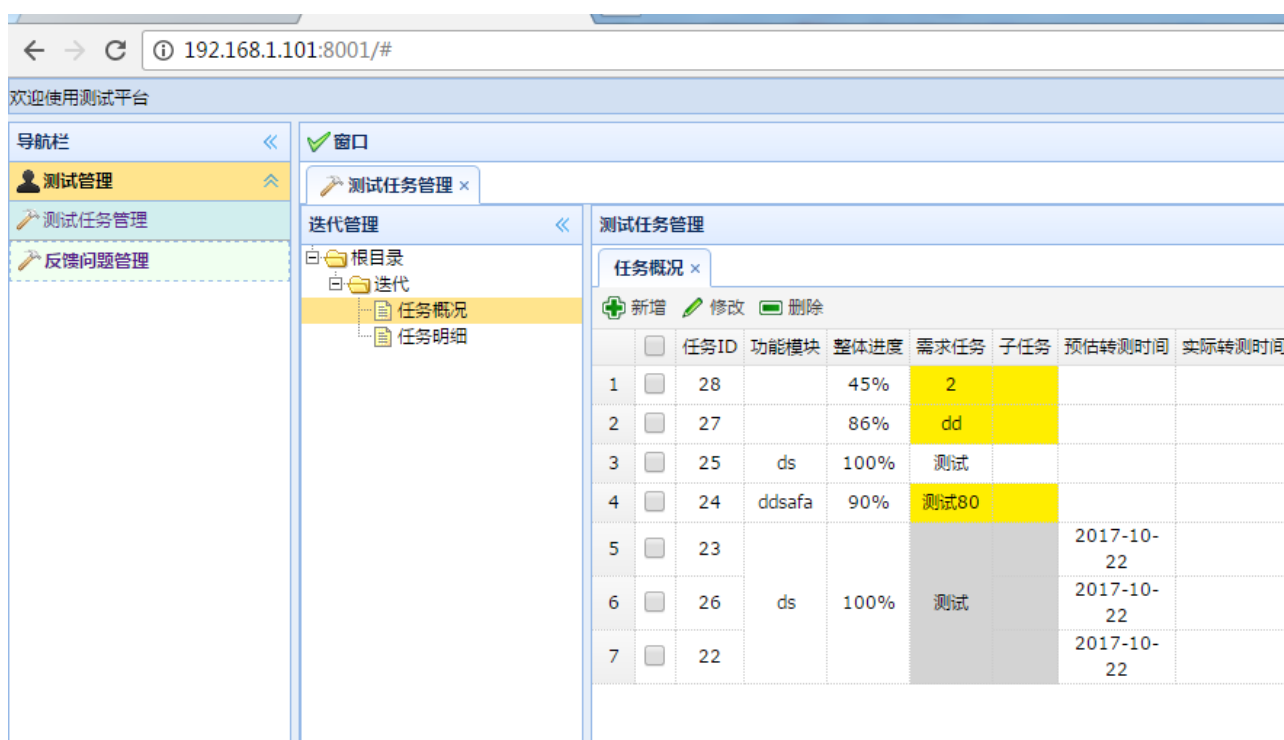
The 'Apache2.4' service is stopping.

The 'Apache2.4' service has stopped.

```
D:\Apache24\bin>httpd.exe -k start
```

说明:到这一步,已经可以浏览器访问了

浏览器验证



注:截图有点错误,端口应该是 8001

4、客户端部署

a) 数据库配置

编辑 conf/db.conf 文件

```
[TESTPLATFORM]
```

```
host = 192.168.1.102
```

```
port = 3306
```

```
user = testacc
```

```
passwd = test1234
```

```
db = testplatform
```

```
charset = utf8
```

说明：这里配置的数据库，即为部署服务端时使用的数据库。

b) https SLL、TSL 版本配置

编辑 conf/https.conf 文件

[HTTPS]

SSL_OR_TLS_PROTOCOL = V2

[README]

#SSL_OR_TLS_PROTOCOL 可选值 V1、 V2、 V23、 V3，不区分大小写

#V1 - TLSv1

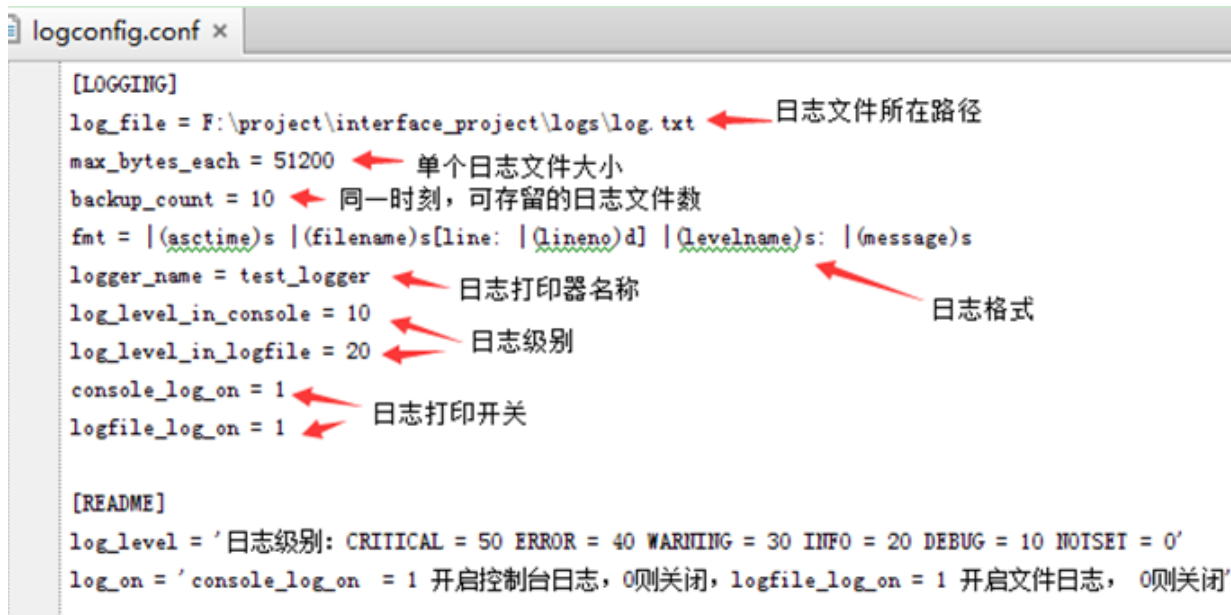
#V2 - SSLv2

#V23 - SSLv23

#V3 - SSLv3

c) 日志配置

编辑 conf/log.conf 文件



```

[LOGGING]
log_file = F:\project\interface_project\logs\log.txt ← 日志文件所在路径
max_bytes_each = 51200 ← 单个日志文件大小
backup_count = 10 ← 同一时刻，可存留的日志文件数
fmt = |(asctime)s |(filename)s[line: |(lineno)d] |(levelname)s: |(message)s
logger_name = test_logger ← 日志打印器名称
log_level_in_console = 10 ← 日志级别
log_level_in_logfile = 20 ← 日志级别
console_log_on = 1 ← 日志打印开关
logfile_log_on = 1 ← 日志打印开关

[README]
log_level = '日志级别: CRITICAL = 50 ERROR = 40 WARNING = 30 INFO = 20 DEBUG = 10 NOTSET = 0'
log_on = 'console_log_on = 1 开启控制台日志, 0则关闭, logfile_log_on = 1 开启文件日志, 0则关闭'
  
```

5、大致操作流程

a) 新增项目

项目管理-API 项目配置

b) 系统配置

数据库配置

操作配置(如果没有初始化配置的话)

断言配置(如果没有初始化配置的话)

c) 新增用例

测试用例管理-API 测试用例管理, 支持嵌套管理

d) 新增 API 计划

测试计划管理-API 测试计划

e) 关联 API 测试用例

可先关联要最先运行的用例, 关联后也可做适当的调整。

f) 新增并执行“运行计划”

新增运行计划后, 可直接点击对应运行计划的“运行按钮”,

也可以直接客户端传参运行

```
cd /d E:\interface_project_for_dev
python main.py rop 1517487657
```

说明:

1、这里的数字 1517487657 即为运行计划编码

2、rop, 大小写不敏感 -> run one project 运行单个项目

g) 查看测试报告

所在模块: 测试报告-API 测试报告

6、用例步骤填写规范

a) 接口请求

针对 http,https 请求

① 执行操作

test_api_for_urlencode

针对请求参数为以下形式的:

```
arg1=value1&arg2=value2&...&argN
```

test_api_for_json

针对请求参数为以下形式(json 格式)的:

```
{
  "key1": "value1",
  "key2": "value2",
  .....,
  "keyN": "valueN"
}
```

test_api_for_xml

针对请求参数为以下形式(xml 形式)的:

```
<ArrayOfString xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns="http://WebXml.com.cn/">
  <string>阿尔及利亚,3320</string>
  <string>阿根廷,3522</string>
  <string>阿曼,3170</string>
  <string>阿塞拜疆,3176</string>
  <string>埃及,3317</string>
  <string>埃塞俄比亚,3314</string>
  <string>爱尔兰,3246</string>
  <string>奥地利,3237</string>
  <string>澳大利亚,368</string>
  <string>巴基斯坦,3169</string>
  <string>巴西,3580</string>
  <string>保加利亚,3232</string>
  <string>比利时,3243</string>
</ArrayOfString>
```

② 请求头

选填, 必须符合 json 格式, 形如以下

```
{
  "User - Agent": "Mozilla / 5.0(Windows NT 6.1; WOW64) AppleWebKit / 537.36(KHTML, like
  Gecko) Chrome / 49.0 .2623 .112 Safari / 537.36",
  "Cache-Control": "no-cache"
}

{"Content-Type":"application/x-www-form-urlencoded","charset":"utf-8"}
```

注: 这里添加的请求头, 不会覆盖“全局请求头”, 而是在全局请求头的基础上新增请求头域及对应值

③ URL/SQL

绝对 URL, 形如以下

```
/pages/nodeTree
/pages/nodeTree?
/pages/nodeTree?page=1&size=10
```

注:

- 1、url 打头可以携带/, 也可以不带/, 如 pages/nodeTree
- 2、如果参数携带中文, 要放到 url 中, 则必先 urlencode, 形如:
/sug?code=utf-8&q=%E7%94%B5%E5%8A%A8%E8%BD%A6&callback=cb

例中, E7%94%B5%E5%8A%A8%E8%BD%A6 decode 后为“电动车”

④ 输入参数

选填

如果执行操作为 `test_api_for_json`, 输入参数的必须符合 json 格式, 形如以下

```
{
  "arg1": "value1",
  "arg2": "value2",
  .....,
  "argN": "valueN"
}
```

如果执行操作为 `test_api_for_urlencode`, 输入参数可以写成 json 格式的(包含两个及以上的不同参数名时不支持), 也可以如下形式的(推荐):

`arg1=value1&arg2=value2&arg3=value3&.....&argN=valueN`

`arg1=value1&arg2=value2&arg3=value3&.....&argN=valueN` 安全模式 xxx

说明:

安全模式 xxx: 指定不需要进行 url 编码的字符, 默认字符为 & 和 =

如果不想指定“安全模式字符”则附加值写“安全模式无”

例:

json:

```
{
  "userName": "dataviewer",
  "userPwd": "f0NfsZjEy65NSc0qBOB4Vzef-NZ0B6n2cXfGvHmXESa-OuzwD3-Q4T0kq0kKdoB9tuCr3N1",
  "captcha": 5555,
  "channelType": "dt",
  "remark": ""
}
```

urlencode:

`custNo=全部&describe=全部&waybillNo[]=198130045411&waybillNo[]=198129993774`

执行操作为 `test_api_for_xml`, 输入参数的必须符合 XML 规范, 形如以下

```
<ArrayOfString xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns="http://WebXml.com.cn/">
  <string>阿尔及利亚,3320</string>
  <string>阿根廷,3522</string>
  <string>阿曼,3170</string>
  <string>巴西,3580</string>
```

```
<string>保加利亚,3232</string>
<string>比利时,3243</string>
</ArrayOfString>
```

⑤ 检查响应

支持 **body**, **header**, **code**, 含义如下:

body - 针对响应体执行断言、从响应体中提取目标值

header - 针对响应头执行断言、从响应头中提取目标值

code - 针对响应代码执行断言、从响应代码中提取目标值

⑥ 校验规则及校验模式

选填, 参考文档“结果断言和提取服务器返回结果.txt”

⑦ 输出

选填, 参考文档“结果断言和提取服务器返回结果.txt”

⑧ 协议

选填, 如果步骤请求使用的协议和项目统一配置不一样, 则可在此处做变更, 仅支持 **https, http**

⑨ 主机地址

选填, 如果步骤请求使用的主机地址和项目统一配置不一样, 则可在此处填写

⑩ 端口

选填, 如果步骤请求使用的主机地址和项目统一配置不一样, 则可在此处填写

b) 操作数据库

① 执行操作

目前支持以下几种操作

select_one_record

针对查询结果返回单条记录的查询, 如果返回了多条, 只会取第一条记录

update_record

执行数据库 update 操作

delete_record

执行数据库删除记录操作

truncate_table

执行数据库清空表操作

call_proc

执行数据库存储过程

insert_record

往数据库插入记录

② URL/SQL

必填, 形如

```
SELECT phone FROM user WHERE user_name = 'dataviewer';  
UPDATE user SET qq=1033553122 WHERE user_name='shouke';
```

注意: where 后面跟的“参数值”(例中的'dataviewer')如果包含单引号', 程序会自动转换为双引号 "

③ 输入参数

选填, 如果有多个参数, 则用逗号分隔, 其中字符串类型的值, 必须用双引号, 否则会报错。

例子: 携带输入参数的 SQL

URL/SQL:

```
SELECT phone FROM ddt_user WHERE qq= %s AND user_name="%s";
```

说明: 这里的%s 占位符和以下输入参数中值, 从左到右, 一一对应。

注意: 如果%s 占位符对应的是一个字符串类型的值, 一定要加双引号, 形如: "%s"

输入参数:

1033553122, "dataviewer"

④ 校验规则及校验模式

选填, 参考文档“结果断言和提取服务器返回结果.txt”

⑤ 输出

选填, 参考文档“结果断言和提取服务器返回结果.txt”

c) 执行函数

① 操作对象

目前支持函数:

死等待

② 输入参数

根据参数样例填写

③ 校验规则及校验模式

选填, 参考文档“结果断言和提取服务器返回结果.txt”

④ 输出

选填, 参考文档“结果断言和提取服务器返回结果.txt”

d) 关于用例及步骤运行顺序说明

1、用例执行顺序：已计划用例关联列表中的用例顺序升序，从上往下执行

2、用例步骤执行顺序：按用例步骤列表中的步骤顺序降序，从下往上执行。

e) 导入用例步骤

为了更方便用例设计，可在 web 端用例树中添加好用例后，在 excel 中填写对应用例的执行步骤，然后批量导入。

1、配置项目及项目环境

编辑 conf/project.conf

```
[PROJECTINFO]
```

```
project_id = 1
```

```
environment = 测试环境
```

```
[README]
```

```
project_id = 项目 ID
```

environment = 项目关联的环境，一定要正确填写，可选枚举值：'测试环境'，'预发布环境'，'生产环境'

2、配置平台数据库

编辑 conf/db.conf

```
[TESTPLATFORM]
```

```
host = 10.118.59.84
```

```
port = 3307
```

```
user = testacc
```

```
passwd = test1234
```

```
db = testplatform
```

```
charset = utf8
```

2、打开 case_steps_importer\datafile 目录下“用例步骤填写模板.xls”，按规范填写用例步骤。

一个 datafile 目录下支持同时存放多个.xls 文件，一个.xls 文件支持多个 sheet

3、进入到工程跟目录下，运行 python

```
cd d:\case_steps_importer
```

```
python main.py
```

4、查看结果

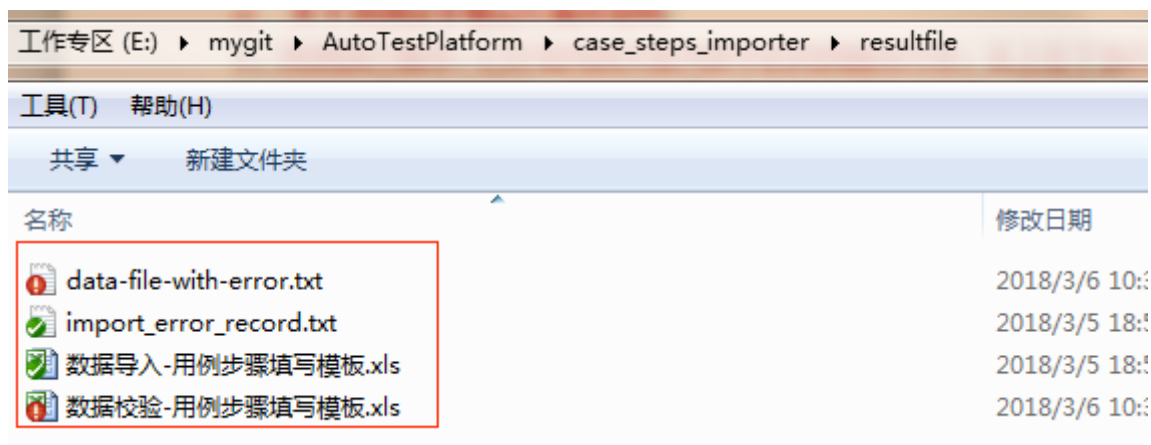
目前程序实现是这样的：

先解析.xls 文档并校验用例步骤否填写正确，并记录解析结果；如果校验通过，则开始导入对应 excel 表中的记录。

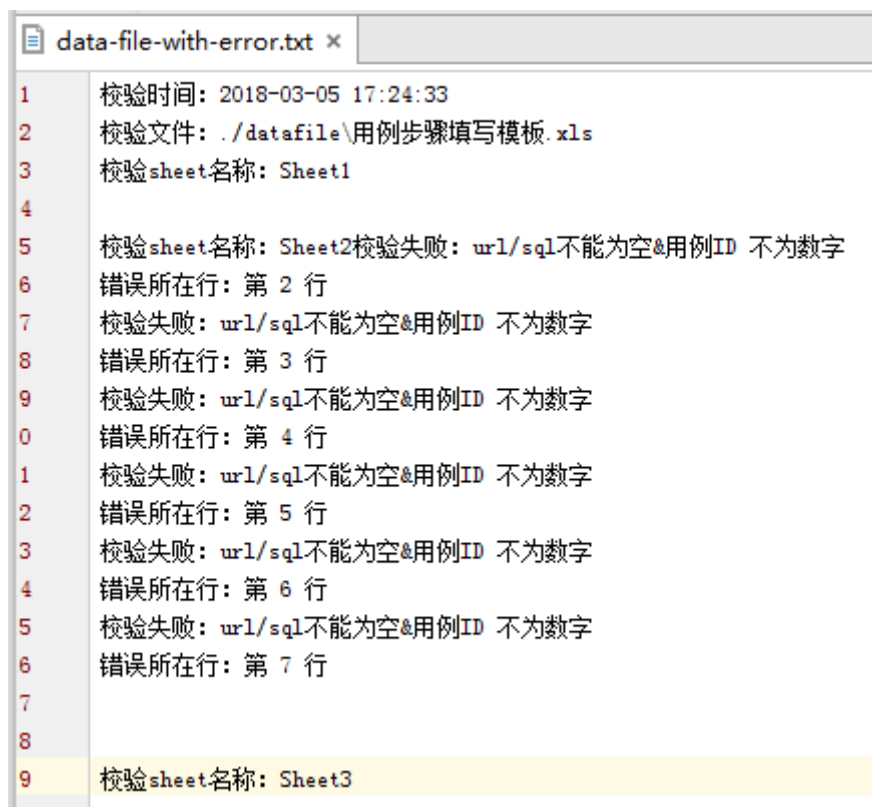
需要注意的是，整个 excel 表，只要有任何一条记录校验失败，则都整个 excel 文件的记录都不会被导入；如果都校验通过，则执行该 excel 表的导入操作，并记录导入结果。

另外，步骤顺序：从上往下依次递增，如果指定用例下已经存在对应的用例步骤，则取最大用例步骤的顺序，在此基础上递增。

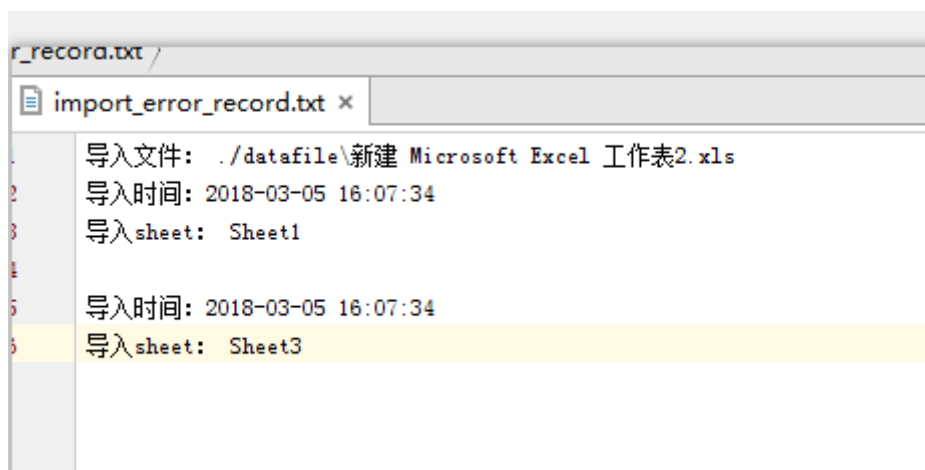
如下, 运行程序后, 会在 `resultfile` 目录下生成对应的文件。



`data-file-with-error.txt`: 存放校验文件发现的问题



`import_error_record.txt`: 存放导入数据时发现的出错, 正常导入则如下, 没有错误提示。



数据校验-*.xls: 和被校验的用例步骤文件类似, 不同的是增加了一列“备注”, 用于展示对应记录行校验不通过的原因

数据导入-*.xls: 和被校验的用例步骤文件类似, 不同的是增加了一列“备注”, 用于展示对应记录行导入失败的原因

7、参数化

① 全局变量

目前没页面化, 可手动在数据库表 `website_global_variable_setting` 中增加

name	value	environment	project_name	project_id
global_headers	{"10.202.95.88":{"Accept-Encoding":"gzip, deflate"},"10.202.95.87":{"Accept-Encoding":"gzip"}}	测试环境	数据灯塔API	1
global_host	10.202.64.168	测试环境	数据灯塔API	1
global_token	asfdacfdasfasfa	测试环境	数据灯塔API	1
global_url_arg	0.5185151614698615	测试环境	数据灯塔API	1
(NULL)	(NULL)	(NULL)	(NULL)	(NULL)

注意:

1、这里的全局变量, 和环境及项目两个因素紧密关联

2、全局请求头, 名称固定, 必须为小写的 `global_headers`, 注意, 这里的请求头和用例步骤定义的请求头是“并集”关系。

格式, 形如以下, 必须符合 json 格式:

```
{
  "host1":{"header_field1":"value1", "header_field2":"value2", ...,
  "header_fieldN":"valueN"},
  "host2":{"header_field1":"value1", "header_field2":"value2", ...,
  "header_fieldN":"valueN"},
  ...
}
```

3、为防止变量值被覆盖, 全局变量的名称, 必须以 `global_` 打头(大小写不限), `global_` 之后的部分大小写敏感

4、修改全局变量, 会通过正则匹配的方式, 替换其关联的老项目中对应用例, 对应步骤中的全局变量

② 局部变量

在用例步骤自定义的变量。

注意：

- 1、局部变量，名称不能包含“空格，-，tab 符”，且不能以 global_、GLOBAL 开头
- 2、局部变量，大小写敏感

③ 变量的引用

`${变量名}`，比如 `${global_headers}`

④ 变量使用范围

支持字段“请求头，URL/SQL，输入参数，校验模式， 主机地址”

注意：当前步骤，输出中的定义的“变量”可用只可用于当前步骤的“校验模式”及其它步骤中上述字段，但是不可用于当前步骤中的“请求头，URL/SQL，输入参数，主机地址”

样例：

请求头	请求方法	URL/SQL
<code>{"Token": "\${global_token\$}"}</code>	GET	<code>/res/rpatchca.png?\${global_url_arg\$}</code>

校验规则	校验模式	输出
键值相等	<code>[[{"模式": {"userName": "\${userName\$}"}, "消息": "FAILURE#用户名不为data viewer"}]]</code>	<code>{"dic": {"userName": {"userName": "value"}, "atten": {"atten": "value"}}</code>
匹配正则表达式	<code>[[{"模式": "\\qq\\ "(.+?)\\ """, "消息": "FAILURE#不符合断言时返回的消息"}]]</code>	<code>{"re": {"userName2": "userName\\ "(.+?)\\ ""}}</code>

注意：截图为旧版本的截图，新版本中变量引用已经改成 `${varName}`，以下不再赘述

校验规则	校验模式	输出
db列值相等	<pre>[{"模式":["\$user_name\$","dataviewer"], "消息":"FAILURE#姓名不为dataviewer"}, {"模式":["\$phone\$","18159001414"], "消息":"FAILURE#手机号不为18189001424"}]</pre>	
db列值相等	<pre>[{"模式":["\$user_name\$","dataviewer"], "消息":"FAILURE#姓名不为dataviewer"}, {"模式":["\$phone\$","18159001414"], "消息":"FAILURE#手机号不为18189001424"}]</pre>	<pre>{"db":{"user_name":2,"phone":1}}</pre>
xpath断言	<pre>[{ "模式": { "//return": "dataviewer" }, "消息": "FAILURE#获取的实体集团卡号错误" }, { "模式": { "//return": "\$groupCode_1\$" }, "消息": "FAILURE#获取的实体集团卡号错误" }]</pre>	<pre>{"xpath":{"groupCode": "//return"}}</pre>

校验规则	校验模式	输出
键值相等	<pre>[{"模式": {"userName": "\$UserName\$"}, "消息": "FAILURE#用户名不为dataviewer"}]</pre>	<pre>{ "dic": { "userName": "userName", "value": "value", "atten": { "atten": "value" } } }</pre>

请求方法	请求URL	请求头	请求体	响应体
POST	/ws/ddtDimCustGroupMapping		dataviewer ;	<pre><soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/" xmlns:ser="http://service.rpt.ddtm.platform.ddt.sf.com/"> <soapenv:Header></soapenv:Header> <soapenv:Body> <ser:findByAccountCompany> <!--Optional:--> <!--Optional:--> <!--Optional:--> <userName>\$UserName\$</userName> <!--Optional:--> <accountCompany>2017112815</accountCompany><vaildFlag>1 </vaildFlag><isReplace>0</isReplace> </ser:findByAccountCompany> </soapenv:Body> </soapenv:Envelope></pre>

注意:

截图中的 FAILURE# 已经改成了 fail#,不区分大小写, 比如也可以 FAIL#,这个是给旧版程序用的, 新版本填不填写都无所

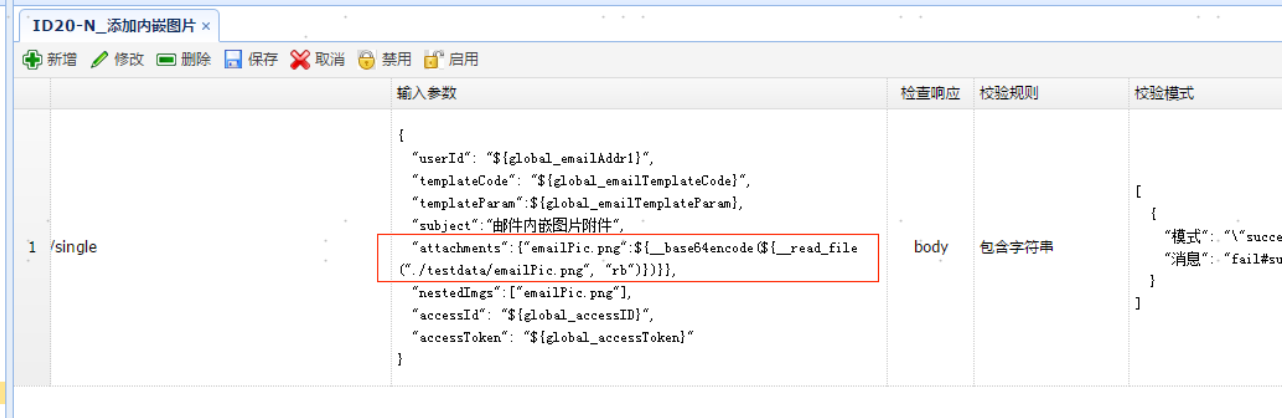
⑤ 插件函数

语法: `${__functionName()}、${_functionName()}`

说明:

- 1、函数名称不包含空格部分(函数名称同`${ }`之间可以存在空白字符, 形如 `${ __myfunction() }`), 以双、单下划线打头,
- 2、支持函数嵌套
- 3、所有字符串参数, 均用“英文双引号”引起来

样例:



支持的函数列表:

1) 读取文件

`${__read_file(file_path, mode, encoding=None)}`

参数说明:

file_path: 为文件路径, 文件存放地址为 interface_project_for_dev\testdata, 编写规范为 ./testdata/file_full_name

mode: 文件打开方式, 支持 r, r+, rw, rb, rb+,

encoding: 如果 mode 为 rb,rb+, 则不填写, 或者填写 None

函数返回值

如果读取文件成功, 如果 mode 不为 rb,rb+则返回字符串数据, 否则返回字节数据, 如果读取文件失败, 则返回 None

2) base64 编码

`${__base64encode(bytest, altchars=None)}`

参数说明:

bytest: 字节数据

函数返回值

如果编码成功, 返回 base64 编码后的数据, 如果读取编码失败, 则返回 None

8、调试

① 项目级别的调试

```
python main.py rop 1517487657 debug
```

说明:

- 1、这里 1517487657 为运行计划编码
- 2、开启调试模式, 不会往数据库记录相关数据

② 用例(套件)级别的调试

```
python main.py debug 项目 ID, 形如: python main.py debug 1
```

说明: debug 大小写不敏感

程序运行后, 按要求输入会要求测试用例、用例套件的 ID, 回车即可查看运行结果

The screenshot shows the 'API测试用例管理' (API Test Case Management) interface on the left and a terminal window on the right. In the interface, a tree view under '数据灯塔API' (Data Beacon API) shows several test cases. A red circle highlights a directory containing multiple test cases: 'ID2-登录页面' (Login Page), 'ID7-根据月结和账号获取集团卡号' (Get Group Card Number by Month Statement and Account), 'ID3-查找用户信息' (Find User Information), 'ID6-灯塔用户正常登录' (Normal Login of Beacon User), and 'ID5-请求验证码' (Request Verification Code). A red arrow points from this directory to the terminal window. The terminal window shows the command 'python main.py debug 2' being executed, where '2' is the ID of the highlighted directory. The terminal output shows the execution of the test cases in the directory, with the first case 'ID2-登录页面' being executed successfully.

类似这种包含多个用例的目录, 即为一个测试用例套件, 注意: 用例套件的调试, 不支持目录层级的嵌套, 如果套件下包含非用例目录, 则会被程序自动忽略

注意:

- 1、用例套件的调试, 不支持目录层级的嵌套, 如果套件下包含非用例目录, 会被自动跳过。
- 2、套件下用例的运行顺序: 从下往上, 顺序执行。

3、调试之前，需要开启并配置好项目相关信息

9、源码下载

下载地址:

下载后解压，用 `pycharm` 导入项目即可，必要时可能需要修改 `.idea/workspace.xml` 中 `python` 程序所在路径、或者删除 `.idea` 整个文件夹目录后倒入