

三日精通：自动化必备之Pytest测试框架训练营

第三天：接口自动化框架封装实战

主讲：北凡老师

13. 测试框架要封装什么

封装：

- 隐藏细节
- 增加功能
- 优化功能

接口自动化封装：

- 使用yaml作为用例，降低自动化门槛
- 自动请求接口、断言接口
- 自动在日志记录HTTP报文
- 自动生成allure测试报告

14. YAML文件格式

一句话：YAML完全兼容JSON格式，并且支持Python相似写法

重点：

1. YAML完全兼容JSON
2. 是数据格式，不是变成语言
3. 像Python一样容易编辑和阅读

1. 安装yaml模块

```
pip install pyyaml
```

2. 编写yaml文件

1. # 作为注释符号
2. 缩进：使用2个空格
3. 成员表示
 - - 表示列表成员
 - : 表示字典成员
4. 兜底：完全兼容JSON

```
# 这是我的第一个yaml文件
```

```
数字：
```

```
- 1  
- -1  
- 1.1
```

```
字符串：
```

```
- '1231231231'  
- "ajklasdjas"  
- asdasda
```

```
空值：null # JSON写法
```

```
列表：[ 1, 2, 3, ] # JSON写法
```

```
字典：{ "a": 1, "b": 2 } # JSON写法
```

3. 加载yaml文件

```
import yaml

def load_yaml(path):
    f = open(path, encoding="utf-8") # 打开文件
    s = f.read() # 读取文件内容

    data = yaml.safe_load(s)

    return data
```

15. 接口测试用例

1. 设计用例内容

1. 名字：请求首页数据接口
2. 标记【可选】
3. 步骤
 1. 请求接口：GET https://www.baidu.com
 2. 响应断言：status_code == 200
 3. 提取变量：json()['code']

2. YAML表示用例

```
name: 登陆成功用例
steps:
  - request: # 发送请求
    method: POST
    url: http://116.62.63.211/shop/api.php?
application=app&application_client_type=weixin
    params:
      s: user/login
    json: {
      "accounts": "beifan_1105",
      "pwd": "beifan_1105",
      "type": "username"
    }

  - response: # 断言响应
    status_code: 200
```

```
    json:
      code: 0
      msg: 登录成功
      data:
        username: beifan_1105

- extract: # 提取变量
  token: [ json, $.data.token]
```

16. 封装接口自动化框架

1. 请求接口

外部工具: requests

从HTTP协议抓包角度, 请求由三部分组成:

- 行: 方法+地址 (必填)
- 头: 请求头 (键值对)
- 体: 参数内容

```
import requests

### 1. 方法
url = 'http://116.62.63.211/shop/api.php'

# GET方法
requests.get(url)
# POST方法
requests.post(url)

# 任意方法
requests.request('MOVE', url)

### 2. 头
method = 'post'
url = 'http://116.62.63.211/shop/api.php'

requests.request(
    method, url,
    headers={ # 字符串字典
        "1": '1',
        "2": "b",
    }
}
```

```

)

### 3. 参数
method = 'post'
url = 'http://116.62.63.211/shop/api.php'

requests.request(
    method, url,
    json={ # 任意内容字典
        "a": 1,
        "b": [1,2,3],
        "c": {},
    }
)

```

2. 断言响应

1. 响应里有什么
2. 响应如何断言

从HTTP协议抓包角度，响应由三部分组成：

- 行：状态码
- 头：响应头（键值对）
- 体：响应内容

```

# resp 就是响应
# 获取响应中的内容
print(resp.status_code) # 状态码
print(resp.headers) # 响应头
print(resp.text) # 响应正文
print(resp.json()) # 响应正文转成JSON

# 断言单个内容是否正确
assert resp.status_code == 200
assert '美酒' in resp.text
assert resp.json()['data']['banner_list'][0]['name'] == "美酒"

# 断言全部的内容
from responses_validator import validator

validator(
    resp,

```

```
status_code=200,
text='*美酒*',
json={
    "data":{
        "banner_list":[{"name":"美酒"}]
    }
}
)
```

3. 变量提取

基本原则：

- JSON: JSONPATH
- HTML: XPATH
- 字符串: RE

RE 可以兜底

```
from extract_utils import extract

var = extract(resp, 'json', '$..banner_list[0].name')

print(var)
```

4. 框架落地封装

```
runner_utils.py x  pytest.log x
my_pytest E:\PyProject\my_pytest
> .venv library root
> commons
> data
> logs
> report
> temps
> tests
> conftest.py
> main.py
> pytest.ini
External Libraries
Scratches and Consoles

INFO 2024-11-15 21:37:09 [pytest_result_log:122] : -----Start:
INFO 2024-11-15 21:37:09 [beifan:14] : 1. 正在发送请求...
INFO 2024-11-15 21:37:09 [beifan:15] : {'method': 'POST', 'url': 'http://116.6:
INFO 2024-11-15 21:37:09 [beifan:18] : 2. 正在断言响应...
INFO 2024-11-15 21:37:09 [beifan:19] : {'status_code': 200, 'json': {'code': 0,
INFO 2024-11-15 21:37:09 [responses_validator:20] : resp=<Response [200]>, rais
INFO 2024-11-15 21:37:09 [responses_validator:20] : is_valid = True, with []
INFO 2024-11-15 21:37:09 [beifan:22] : 3. 正在提取变量...
INFO 2024-11-15 21:37:09 [beifan:25] : token = 0be55e73bfb0e81a2948493b67c
INFO 2024-11-15 21:37:09 [pytest_result_log:190] : test status is PASSED (tests
INFO 2024-11-15 21:37:09 [pytest_result_log:128] : -----End:
```

进一步完善：

1. YAML用例测试文件上传？
2. YAML用例进行数据去掉测试？
3. YAML用例进行自定义的断言
4. YAML用例进行数据库查询？

两个方向：

1. 自动化工程师
2. 测试开发工程师

直播：自动化测试 4个月左右

快速就业跳槽：15-30天

学习：

- 找班主任
- 找技术北凡

工作：

- 技术难题
- 跳槽

