MTSC2019
中国移动互联网测试开发大会

HttpRunner 2.0 技术架构与接口测试应用

李隆　debugtalk

主办方：TesterHome　腾讯课堂

自我介绍

- 李隆　Leo Lee
- ID：debugtalk
- 邮箱：mail@debugtalk.com

先后就职于：

- 工信部电子五所
- 阿里移动事业群（UC）
- 大疆创新
- 字节跳动

You can find me here:

- Blog: https://debugtalk.com
- GitHub: https://github.com/debugtalk
- TesterHome: https://testerhome.com/debugtalk
- 微信公众号：**DebugTalk**

主题大纲

1、HttpRunner 项目背景 （3min）

2、HttpRunner 核心特色 （7min）

3、HttpRunner 开发历程回顾（3min）

4、HttpRunner 2.0 新特性解析（7min）

5、现场实践演示（15min）

6、再看 HttpRunner，新思路及演进方向（5min）

7、加入 HttpRunner 开源项目（3min）

8、Q & A（2min）

期望达成目标

1、掌握 HttpRunner 功能特性和最佳实践，可在工作项目中立即投入使用；

2、理解 HttpRunner 设计思想和技术架构，掌握二次开发和平台化建设思路；

3、加入 HttpRunner 开源项目。

HttpRunner 项目背景

HttpRunner 项目背景

业务导向，快速落地
提高投入产出比

少投入

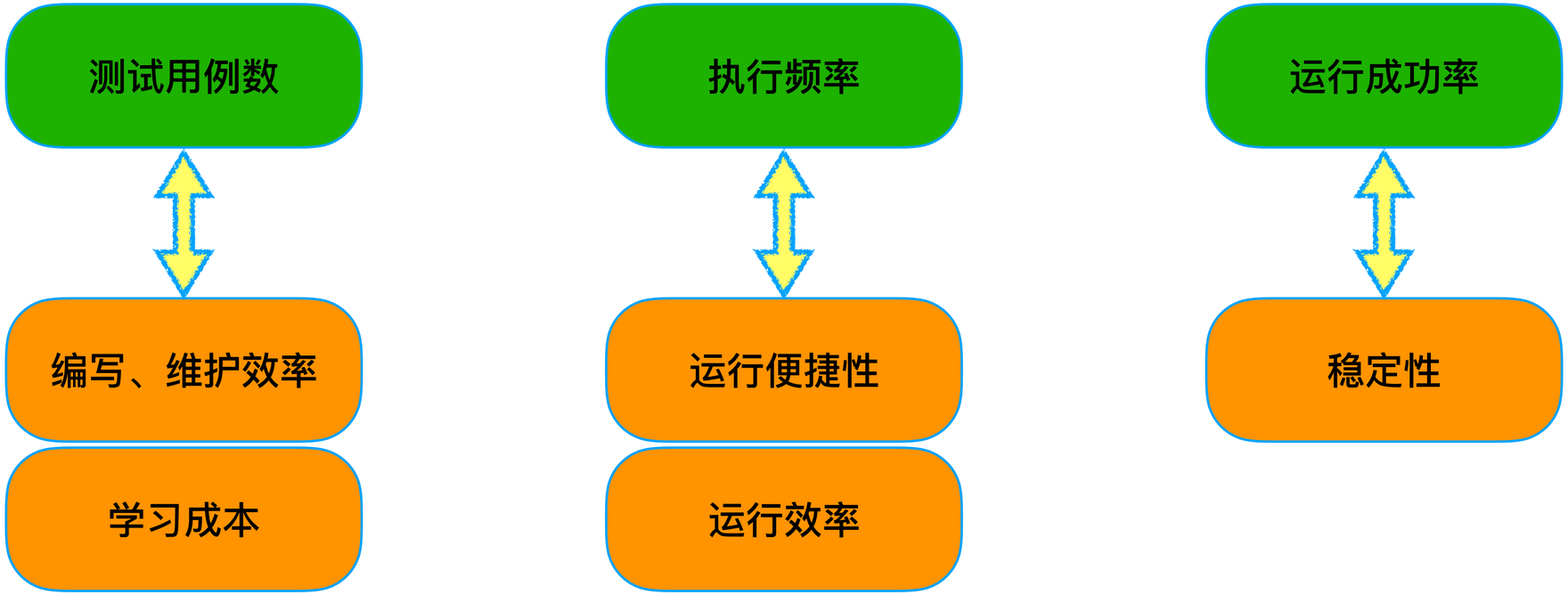
- 工具开发&维护
- 学习使用成本
- 编写&录入测试用例
- 测试用例管理&维护

高收益

- 实现自动化回归测试
- 性能测试脚本复用
- 兼具持续集成、线上监控
- 辅助手工测试：自定义生成特定业务数据

自动化测试核心指标

"If you cannot measure it, you cannot improve it. "



调研结果

- 工具很多，但都只能满足部分需求
- 投入产出比不能满足预期
- 缺乏最佳工程实践的有效融合

HttpRunner 项目背景

HttpRunner 核心特色

HttpRunner 核心特色

One-stop solution for HTTPS(S) testing.

GitHub, Inc. [US] | https://github.com/HttpRunner/HttpRunner

Search or jump to... / Pull requests Issues Marketplace Explore

HttpRunner / HttpRunner

Unwatch 116 | Star 1,380 | Fork 528

Code | Issues 185 | Pull requests 0 | Projects 3 | Wiki | Security | Insights | Settings

One-stop solution for HTTP(S) testing. https://cn.httprunner.org/

Edit

api-test locustio performance-testing locust httptest testrunner htmltestrunner load-testing Manage topics

999 commits | 2 branches | 18 releases | 1 environment | 6 contributors | Apache-2.0

HttpRunner

license Apache-2.0 | build passing | coverage 81% | pypi v2.1.3 | python 2.7 | 3.4 | 3.5 | 3.6 | 3.7

HttpRunner is an HTTP(S) protocol-oriented universal testing framework. Write testing scripts in `YAML/JSON` once, you can then achieve automated testing, performance testing, online monitoring, continuous integration and other testing needs.

Formerly known as `ApiTestEngine`.

Design Philosophy

- Take full reuse of Python's existing powerful libraries: `Requests`, `unittest` and `Locust`.
- Convention over configuration.
- Pursuit of high rewards, write once and achieve a variety of testing needs

Key Features

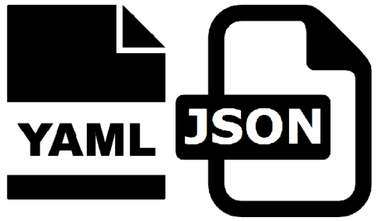
- Inherit all powerful features of `Requests`, just have fun to handle HTTP(S) in human way.
- Define testcases in YAML or JSON format in concise and elegant manner.
- Record and generate testcases with `HAR` support. see `har2case`.
- Supports `function` / `variable` / `extract` / `validate` mechanisms to create full test scenarios.
- Supports perfect hook mechanism.
- With `debugtalk.py` plugin, module functions can be auto-discovered in recursive upward directories.
- Testcases can be run in diverse ways, with single testcase, multiple testcases, or entire project folder.
- Test report is concise and clear, with detailed log records.
- With reuse of `Locust`, you can run performance test without extra work.

核心特色

- 充分复用开源项目
- 约定大于配置
- 配置文件组织测试用例（YAML/JSON）
- 一次投入，多处复用
- 高度可扩展性
- etc.

HttpRunner 核心特色

充分复用开源项目



HttpRunner

HttpRunner 核心特色

借鉴优秀（开源）项目

- debugtalk.py ← pytest（conftest.py）
- 测试用例模板语言（JSON/YAML） ← Jinja
- 参数化机制（parameters） ← Loadrunner
- 环境变量管理（.env） ← POSTMAN
- 测试用例分层管理 ← Katalon Intelligent Test Automation

HttpRunner 核心特色

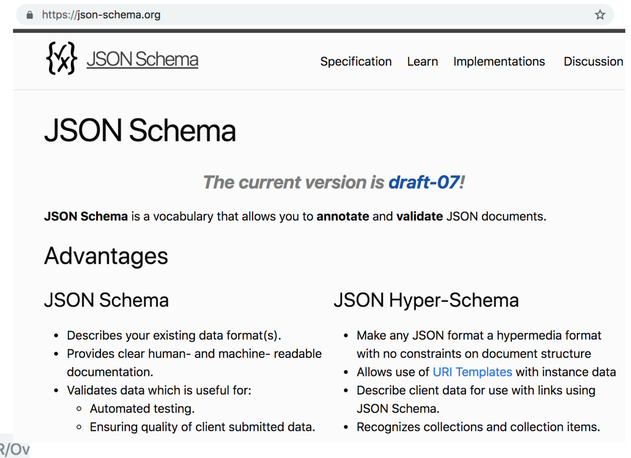
借力行业标准

HTTP Archive (HAR) format

Editor's Draft August 14, 2012

This version:
https://w3c.github.io/web-performance/specs/HAR/Overview.html
Latest version:
https://w3c.github.io/web-performance/specs/HAR/Overview.html
Latest Editor's Draft:
https://w3c.github.io/web-performance/specs/HAR/Overview.html
Editors:
Jan Odvarko, <honza@softwareishard.com>
Arvind Jain, Google Inc., <arvind@google.com>
Andy Davies, <andy@andydavies.me>

Copyright © 2012 W3C® (MIT, ERCIM, Keio, Beihang), All Rights Reserved.



Main Interface

All of Requests' functionality can be accessed by these 7 methods. They all return an instance of the **Response** object.

`requests.request`(*method, url, **kwargs*) [source]

Constructs and sends a **Request**.

Parameters:
- **method** – method for the new **Request** object.
- **url** – URL for the new **Request** object.
- **params** – (optional) Dictionary or bytes to be sent in the query string for the **Request**.
- **data** – (optional) Dictionary or list of tuples `[(key, value)]` (will be form-encoded), bytes, or file-like object to send in the body of the **Request**.
- **json** – (optional) A JSON serializable Python object to send in the body of the **Request**.
- **headers** – (optional) Dictionary of HTTP Headers to send with the **Request**.
- **cookies** – (optional) Dict or CookieJar object to send with the **Request**.

HttpRunner 核心特色

export

HAR

har2case

YAML

postman2case

swagger2case

JSON

自动化平台录入/手工编写

整合生态工具链

JSON Schema

HttpRunner 核心特色


```python
import unittest
import requests

class FirstTestcase(unittest.TestCase):

    def test_get_token(self):
        url = "http://127.0.0.1:5000/api/get-token"
        method = "POST"
        headers = {
            "user_agent": "iOS/10.3",
            "device_sn": "9TN6O2Bn1vzfybF",
            "os_platform": "ios",
            "app_version": "2.8.6"
        }
        json = {
            "sign": "19067cf712265eb5426db8d3664026c1ccea02b9"
        }

        resp = requests.request(method, url, headers=headers, json=json)
        status_code = resp.status_code
        token = resp.json()["token"]

        assert status_code == 200
        assert len(token) == 16

    def test_create_user_which_does_not_exist(self):
        url = "http://127.0.0.1:5000/api/users/1000"
        method = "POST"
        headers = {
            "device_sn": "9TN6O2Bn1vzfybF",
            "token": "kEoKIu6SRPTX3IZA"
        }
        json = {
            "name": "user1",
            "password": "123456"
        }

        resp = requests.request(method, url, headers=headers, json=json)
        status_code = resp.status_code
        success = resp.json()["success"]

        assert status_code == 201
        assert success is True
```

配置文件组织测试用例
(YAML/JSON)

信息量等价

重复

迥异

简洁

规范

```yaml
- test:
    name: get token
    request:
        url: http://127.0.0.1:5000/api/get-token
        method: POST
        headers:
            user_agent: "iOS/10.3",
            device_sn: "9TN6O2Bn1vzfybF",
            os_platform: "ios",
            app_version: "2.8.6"
        json:
            sign: 19067cf712265eb5426db8d3664026c1ccea02b9
    extract:
        - token: content.token
    validate:
        - eq: [status_code, 200]
        - len_eq: [content.token, 16]

- test:
    name: create user which does not exist
    request:
        url: http://127.0.0.1:5000/api/users/1000
        method: POST
        headers:
            device_sn: "9TN6O2Bn1vzfybF",
            token: "kEoKIu6SRPTX3IZA"
        json:
            name: "user1"
            password: "123456"
    validate:
        - eq: [status_code, 201]
        - eq: [content.success, true]
```

HttpRunner 核心特色

数据驱动

实现命令行（CLI）调用

Jenkins

生成测试数据

自动化测试
YAML/JSON

持续集成

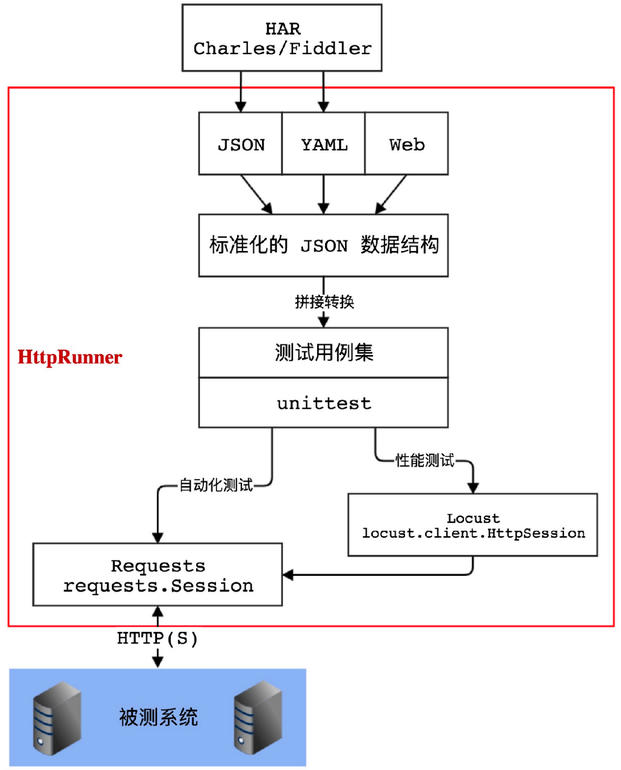
线上监控

引入并发机制

LOCUST

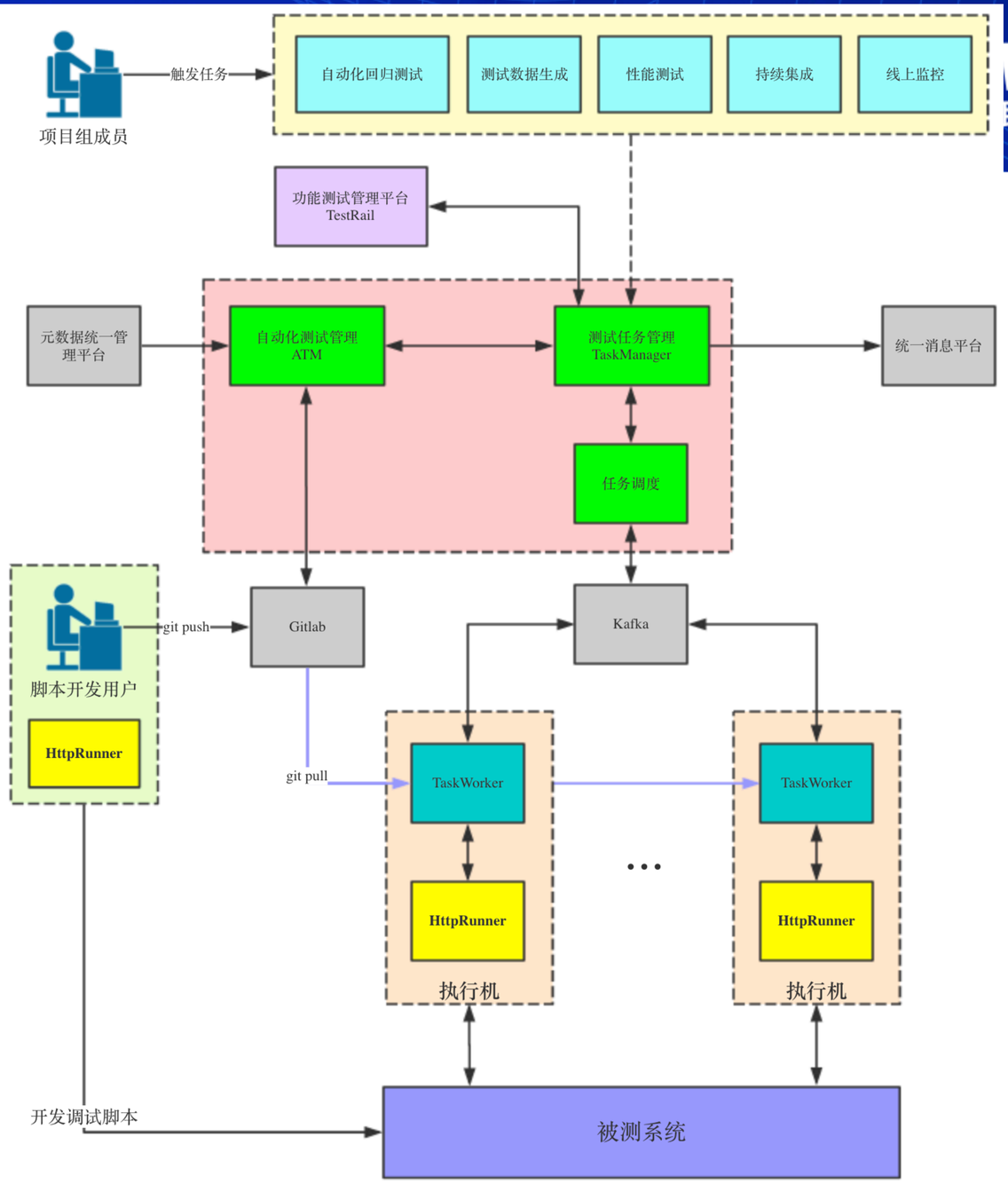
性能测试

一次投入，多处复用



HttpRunner 核心特色

高度可扩展性



HttpRunner 核心特色

更多特性

- 抓包录制 & 生成用例
- *Postman* 转换生成用例
- *Swagger* 转换生成用例
- *YAML/JSON* 用例格式
- *JSON Schema*
- 数据驱动机制
- 测试用例分组执行控制
- *setup & teardown hooks*
- 热加载机制*(debugtalk.py)*
- 性能测试
- *CLI* 调用
- 可扩展测试报告*(Jinja2)*
- 测试用例分层机制
- 加载环境变量*(.env)*
- ...

HttpRunner 核心特色

不仅仅是自动化测试工具

通用的自动化测试解决方案

融合最佳工程实践

打造接口自动化测试生态

不止于此

基于 HttpRunner 的开源接口测试平台：

- HttpRunnerManager
- FastRunner
- SECO
- testcenter
- ApiTestWeb

HttpRunner 开发历程回顾

HttpRunner 开发历程回顾

0.x 版本

- 单元测试框架
- 项目基础框架
- 确定测试用例（集）描述方式（v1）
- YAML/JSON 测试用例中实现 Python 调用
- 集成 Locust 实现性能测试
- 实现热加载机制（debugtalk.py）
- 结果校验机制
- 结果校验解析器（JSON/XML/HTML）
- 测试用例分层机制（第一版）
- 通过 skip 机制实现对测试用例的分组执行控制
- 实现参数化数据驱动机制（第一版）

1.x 版本

- 参数化数据驱动机制（第二版）
- 实现 hook 机制
- 重构 locusts

2.x 版本

- 规范化版本管理（Semantic Versioning）
- 测试用例分层机制（第二版）
- 测试用例（集）格式（v2）
- 模板化脚手架
- 惰性解析器
- 模块化拆分（Pipeline）
- 脚本参数异常检测机制

HttpRunner 2.0 新特性解析

重点升级

- 规范化版本管理（Semantic Versioning）
- 测试用例（集）格式（v2）
- 测试用例分层机制（第二版）
- 模板化脚手架
- 函数惰性解析机制
- 模块化拆分（Pipeline）
- 脚本参数异常检测机制
- etc.

规范化

易用性

性能优化

规范化版本管理
Semantic Versioning

HttpRunner 将采用 MAJOR.MINOR.PATCH 的版本号机制：

- MAJOR: 重大版本升级并出现前后版本不兼容时加 1
- MINOR: 大版本内新增功能并且保持版本内兼容性时加 1
- PATCH: 功能迭代过程中进行问题修复（bugfix）时加 1

Release History

2.2.2 (2019-06-26)

Features

- `extract` is used to replace `output` when passing former teststep's (as a testcase) export value to next teststep
- `export` is used to replace `output` in testcase config

2.2.1 (2019-06-25)

Features

- add demo api/testcase/testsuite to new created scaffold project
- update default `.gitignore` of new created scaffold project
- add demo content to `debugtalk.py` / `.env` of new created scaffold project

Bugfixes

- fix extend with testcase reference in format version 2
- fix ImportError when locustio is not installed
- fix YAMLLoadWarning by specify yaml loader

2.2.0 (2019-06-24)

Features

- support testcase/testsuite in format version 2

Bugfixes

- add wheel in dev packages
- fix exception when teststep name reference former extracted variable

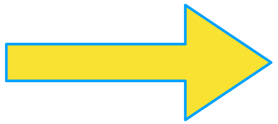
2.1.3 (2019-04-24)

Bugfixes

- replace eval mechanism with builtins to prevent security vulnerabilities

测试用例（集）
格式（v2）

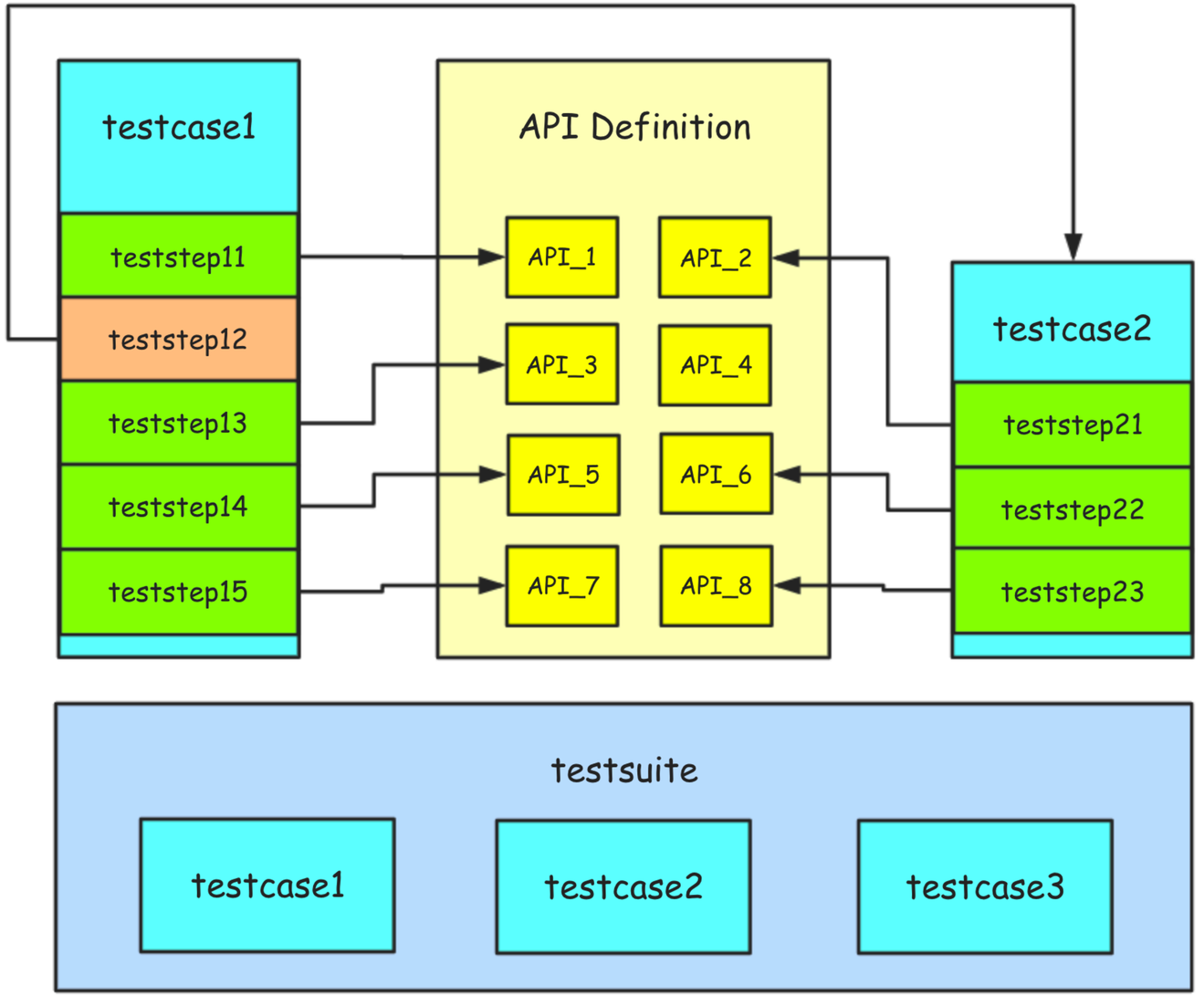
```
[
  {
    "config": {
      "id": "create_user",
      "variables": {
        "device_sn": "TESTCASE_CREATE_XXX",
        "uid": 9001
      },
      "export": [
        "session_token"
      ],
      "base_url": "http://127.0.0.1:5000",
      "name": "create user and check result."
    }
  },
  {
    "test": {
      "testcase": "testcases/setup.yml",
      "export": [
        "session_token"
      ],
      "name": "setup and reset all (override) for $device_sn."
    }
  },
  {
    "test": {
      "testcase": "testcases/deps/check_and_create.yml",
      "variables": {
        "token": "$session_token"
      },
      "name": "create user and check result."
    }
  }
]
```



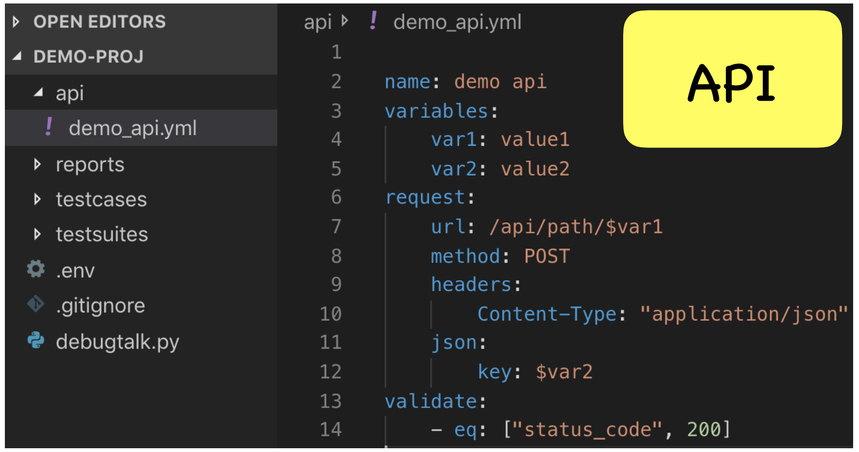
```
{
  "config": {
    "id": "create_user",
    "variables": {
      "device_sn": "TESTCASE_CREATE_XXX",
      "uid": 9001
    },
    "export": [
      "session_token"
    ],
    "base_url": "http://127.0.0.1:5000",
    "name": "create user and check result."
  },
  "teststeps": [
    {
      "testcase": "testcases/setup.yml",
      "export": [
        "session_token"
      ],
      "name": "setup and reset all (override) for $device_sn."
    },
    {
      "testcase": "testcases/deps/check_and_create.yml",
      "variables": {
        "token": "$session_token"
      },
      "name": "create user and check result."
    }
  ]
}
```

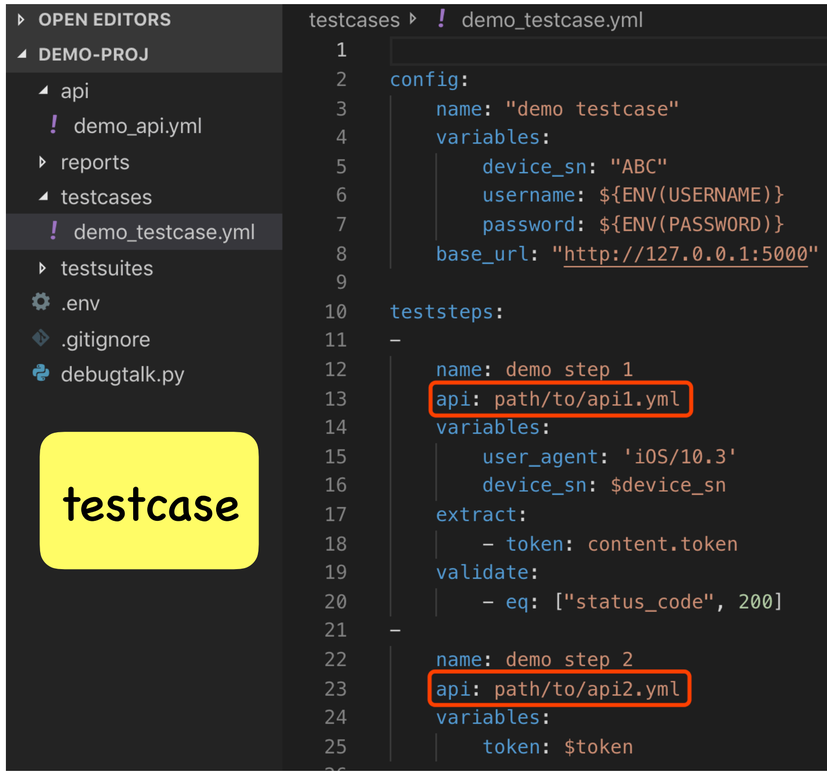
测试用例分层机制(第二版)

- 测试用例（testcase）应该是完整且独立的，每条测试用例应该是都可以独立运行的
- 测试用例是测试步骤（teststep）的 有序 集合，每一个测试步骤对应一个 API 的请求描述
- 测试用例集（testsuite）是测试用例的 无序 集合，集合中的测试用例应该都是相互独立，不存在先后依赖关系的；如果确实存在先后依赖关系，那就需要在测试用例中完成依赖的处理



测试用例分层机制(第二版)





模板化脚手架

```
➜ MTSC2019 hrun --startproject demo-proj
Start to create new project: demo-proj
CWD: /Users/debugtalk/Documents/MTSC2019

created folder: demo-proj
created folder: demo-proj/api
created folder: demo-proj/testcases
created folder: demo-proj/testsuites
created folder: demo-proj/reports
created file: demo-proj/api/demo_api.yml
created file: demo-proj/testcases/demo_testcase.yml
created file: demo-proj/testsuites/demo_testsuite.yml
created file: demo-proj/debugtalk.py
created file: demo-proj/.env
created file: demo-proj/.gitignore
```

```
➜ MTSC2019 tree demo-proj -a
demo-proj
├── .env
├── .gitignore
├── api
│   └── demo_api.yml
├── debugtalk.py
├── reports
├── testcases
│   └── demo_testcase.yml
└── testsuites
    └── demo_testsuite.yml

4 directories, 6 files
```

函数惰性解析机制

```yaml
config:
    name: test upload file with httpbin
    base_url: ${get_httpbin_server()}

teststeps:
-
    name: upload file
    variables:
        file_path: "data/test.env"
        multipart_encoder: ${multipart_encoder(file=$file_path)}
    request:
        url: /post
        method: POST
        headers:
            Content-Type: ${multipart_content_type($multipart_encoder)}
        data: $multipart_encoder
    validate:
        - eq: ["status_code", 200]
        - startswith: ["content.files.file", "UserName=test"]
```

```python
class LasyParser(object):

    def __init__(self, raw_string, functions_mapping):
        self.raw_string = raw_string
        self.__pre_parse(raw_string)

    def __pre_parse(self, raw_string):
        # prepare parser work
        pass

    def to_value(self, variables_mapping):
        # parse lazy data with evaluated variables mapping.
        pass
```

遵循 Unix 哲学

"Write programs that do one thing and do it well."
"Write programs to work together."

模块化拆分
（Pipeline）

prepare

initializer

loader

runner

parser

report

现场实践演示

案例介绍

网址：https://mubu.com

覆盖功能场景：

- 登录系统
- 创建文档
- 设置文档标题

现场实践演示

- 安装 HttpRunner
- 脚手架生成项目
- 录制接口，生成 HAR 文件
- 使用 HAR 生成 YAML/JSON 测试用例
- 环境变量配置
- 处理参数关联机制，成功运行脚本
- 参数化配置
- 测试用例分层
- 数据驱动
- 性能测试演示
- 持续集成配置（Jenkins）
- 测试数据生成（命令行、Jenkins）

演示流程

完整示例代码

再看 HttpRunner，新思路及演进方向

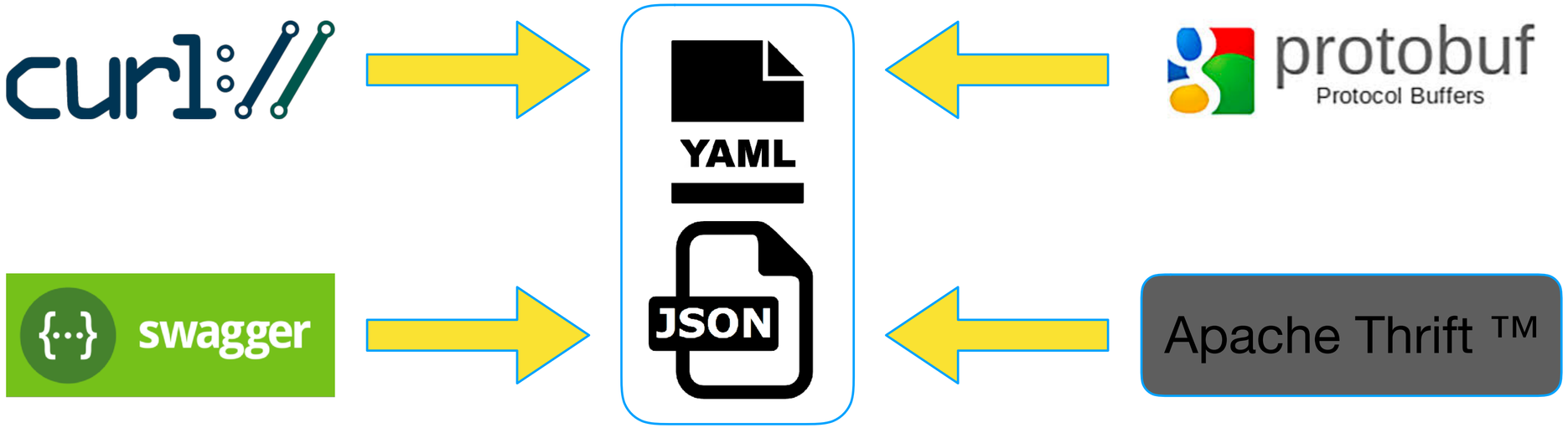
再看 HttpRunner，新思路及演进方向

- 更多的测试用例生成方式
- 扩展支持更多协议
- 测试用例 Tag 分组执行控制
- 整合更多测试类型（参考 QTA）
- 测试用例（脚本）单步调试功能
- 支持更多编程语言
- 运行环境前端化
- 支持更多配置化语言
- etc.

再看 HttpRunner，新思路及演进方向

更多的测试用例生成方式

扩展支持更多协议

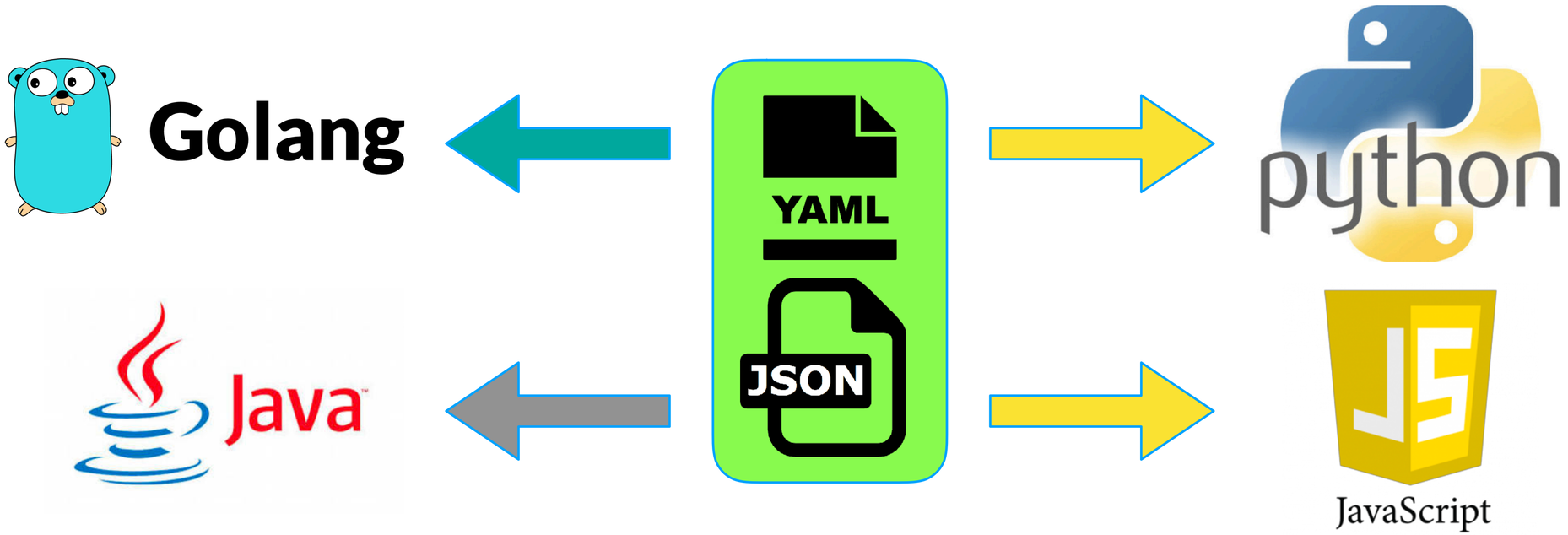


RPC

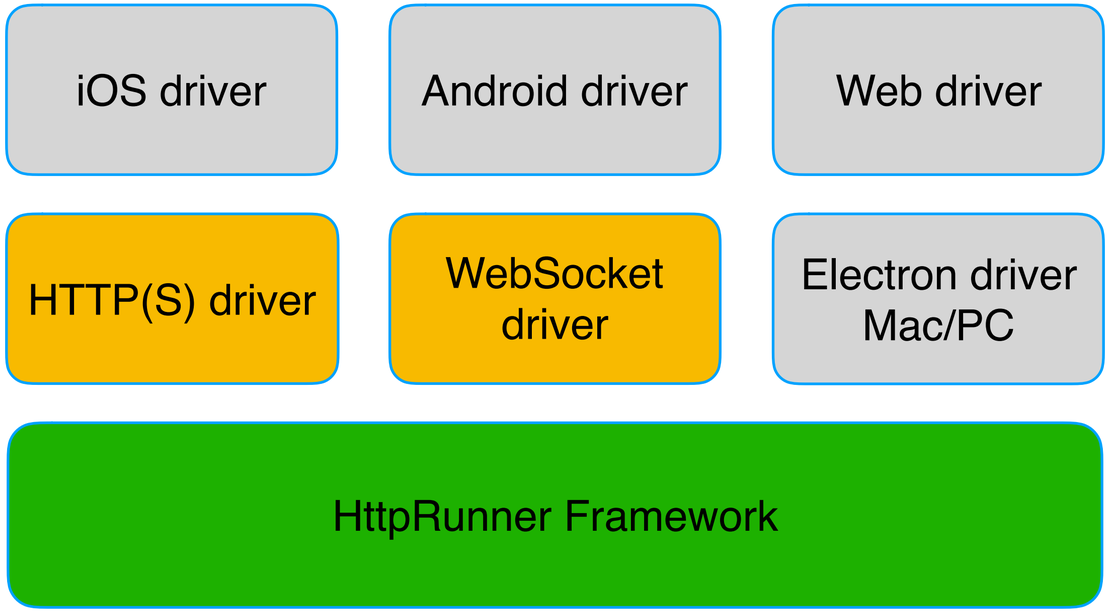
测试用例 Tag 分组执行控制

	testcase 1 tag A tag B	testcase 2 tag A tag C	testcase 2 tag B tag C
$ hrun --tags A	✓	✓	✗
$ hrun --tags B	✓	✗	✓

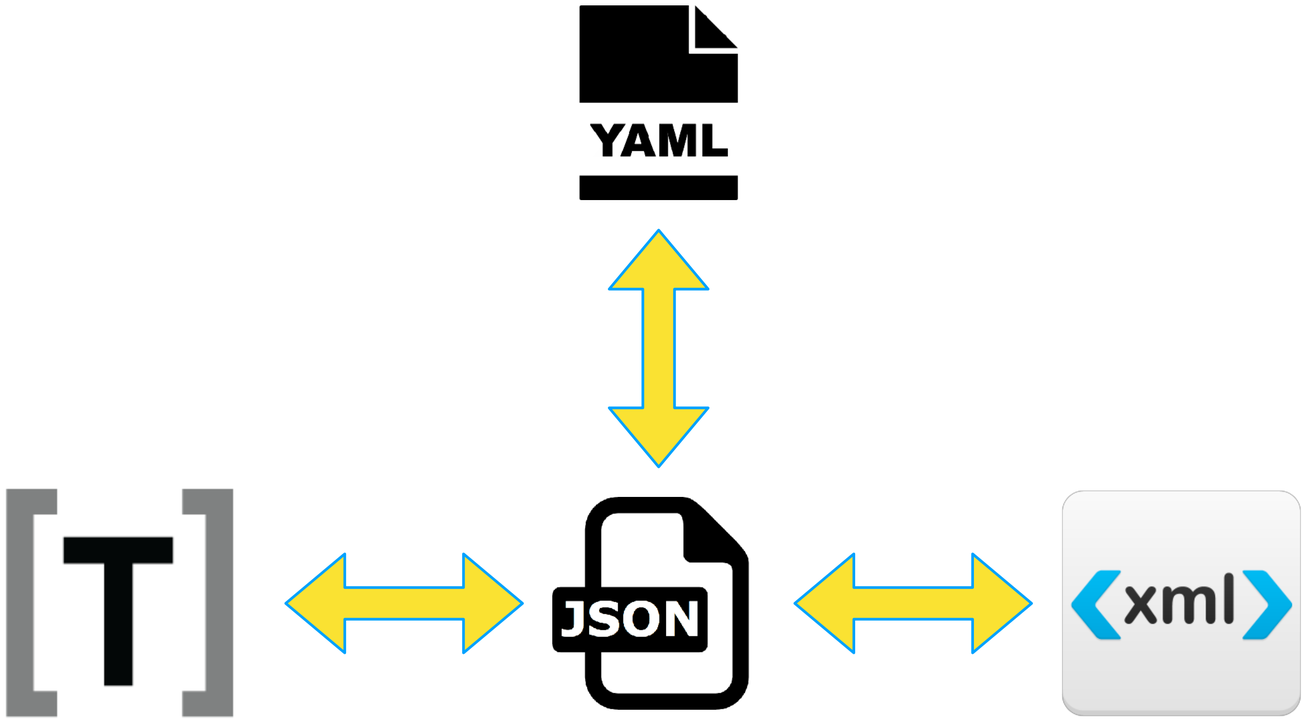
支持更多编程语言



整合更多测试类型


iOS driver
Android driver
Web driver
HTTP(S) driver
WebSocket driver
Electron driver Mac/PC
HttpRunner Framework

支持更多配置化语言



加入 HttpRunner 开源项目

TTF 项目孵化

招募核心开发者和项目成员

加入 HttpRunner 开源项目

进阶路线

重度用户 → 参与 issue 讨论 → 提交 bugfix PR → 新特性开发 → 参与项目规划设计

收获成长

更广阔的视角理解接口测试 → 以码会友，提升代码能力 → 亲手打造行业级测试框架

Q & A

DebugTalk

HttpRunner

Thanks