flybirds

发行版本 *v0.1.5*

trip_flight

2023 年 07 月 11 日

Contents:

CHAPTER 1

携程机票跨端跨框架 BDD UI 自动化测试方案 Flybirds

1.1 背景

Flybirds 是一套基于 BDD 模式的前端 UI 自动化测试框架，提供了一系列开箱即用的工具和完善的文档。

多端研发对于当今时代的前端开发来说是个绕不过去的话题，为了解决这些问题，行业内推出了很多开发方案，但是跨端 UI 自动化测试的解决方案并不多。

Flybirds 从 2022 年初开源至今，通过与社区内活跃用户的交流和反馈，推出了 v0.2 版本的跨端跨框架测试方案，一套脚本多端运行，插件化的架构设计，也方便社区开发者自由加入扩展，一起共建成长。

1.2 我们需要一个怎么样的多端测试方案

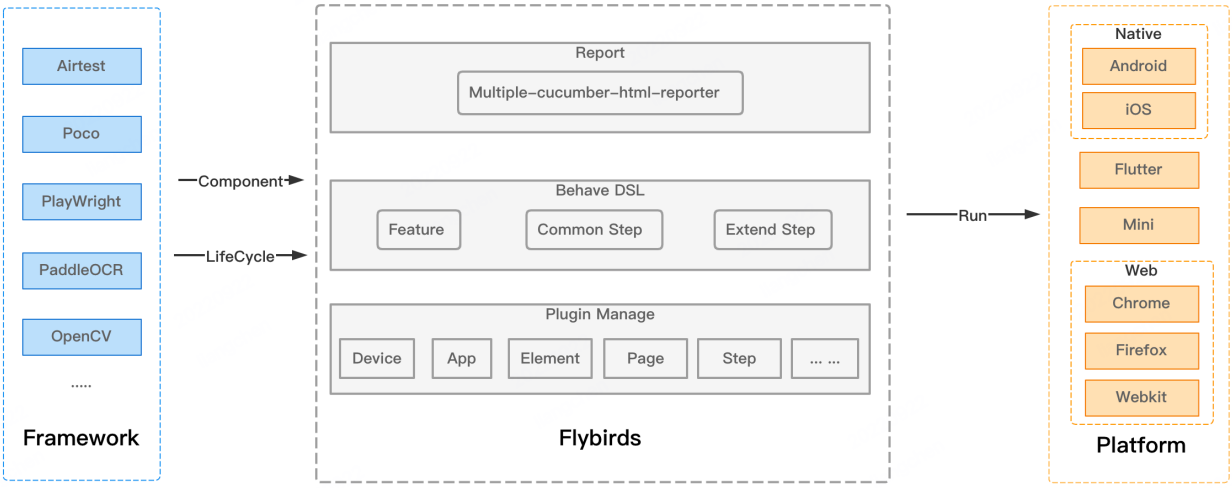
近几年，每隔一段时间就会有很多新的开发框架出现，带来了更好的开发体验和性能的同时，也给自动化测试创造了很多难题。

我们到底需要一个怎样的多端测试方案呢？从 Flybirds 的视角来说，我们希望多端测试不会成为研发流程中的障碍，特别是多端生态整体呈现欣欣向荣之时，自动化测试方案应和开发方案共同成长。

不论是 Web 、React Native 端，还是 Native 端，理想的方案应该进行多端适配，保留良好扩展，兼顾更多框架，由社区共同建设，促进整体生态繁荣，因此就有了 Flybirds 向社区提供的跨端跨框架测试方案。

1.3 插件化架构

插件化架构帮助我们将每一个端的能力拆分开, 插件提供运行时所需的组件、API 和配置，Flybirds 将它们分别注入对应的生命周期。



- 基于 Behave，实现 BDD 中“自然语言测试用例文档”和“自动化测试代码”关联所需要用到的支持 BDD 工具。
- 基于 Airtest，实现 BDD 中“测试用例能在自动化测试平台上执行”所需要用到的 UI 自动化测试框架。
- 基于 Playwright, 实现 BDD 中“测试用例能在自动化测试平台上执行”需要用到 Web 端 UI 自动化测试框架。
- 基于 PaddleOCR 和 OpenCV, 实现 BDD 中“测试用例能在自动化测试平台上执行”需要用到的 OCR 和图像识别能力。
- 基于 Multiple-cucumber-html-reporter，实现可视化的测试报告

1.4 文件结构

```
└
↪
├─ cli                                脚手架
├─ core
|   ├─ config_manage.py          配置管理
|   ├─ dsl
|   |    ├─ globalization        国际化处理
|   |    └─ step                 Step 列表
|   ├─ global_resource.py        全局配置
|   ├─ launch_cycle              生命周期管理
```

(续下页)

(接上页)

```
|   └─ plugin
|        ├─ event                事件管理
|        ├─ plugin_manager.py    插件管理
|        └─ plugins
|             ├─ android         Andriod 相关处理
|             ├─ ios             iOS 相关处理
|             └─ web             Web 相关处理
├─ report                        报告
├─ template                      模板处理
└─ utils
```

1.5 特性

使用 **Flybirds** 你能够完成大部分的手机端自动化操作，以下是一些帮助入门的特性描述：

- 基于 BDD 模式，类自然语言语法
- 支持自动化操作、表单提交、UI 元素校验、键盘输入、Deeplink 跳转等
- 支持 Android、iOS、React Native、Flutter、Web
- 支持多端脚本复用
- 支持 OCR 和 OpenCV
- 支持多浏览器渲染引擎：Chromium、WebKit 和 Firefox
- 支持多浏览器并发模式下的兼容性测试
- 默认支持英文、中文两种语言，支持更多语言扩展
- 插件式设计，支持用户自定义自动化扩展
- 提供 cli 脚手架，快速搭建项目
- 提供 html 报告

1.6 环境要求

- python(3.8-3.10)
- nodejs(12+)

1.7 环境搭建

1. **使用 `pip` 安装 flybirds 框架，过程中会自动安装所需的 依赖包**

```
pip3 install flybirds
```

在 Mac/Linux 系统下，需要手动赋予 adb 可执行权限

- for mac

```
cd {your_python_path}/site-packages/airtest/core/android/static/adb/mac
chmod +x adb
```

- for linux

```
cd {your_python_path}/site-packages/airtest/core/android/static/adb/linux
chmod +x adb
```

2. **使用脚手架创建项目**

```
flybirds create
```

```
Welcome to flybirds cli. Please enter any information to continue.
Please input your project name>>: my_first_project
Please input your test platform?(Android/IOS/Web): web
Please input your test browserType?(chromium/firefox/webkit): chromium
Do you want to launch browser in headless mode? [y/N]: y
Cloning into 'my_first_project'...
Processing  [####################################]  100%
Done it! Create Project my_first_project has success!
You can find it at: /Users/liangchen/webtest/my_first_project
```

1.8 运行前检查

1.8.1 Android、iOS

1. 请确保配置的测试设备能够正常连接
   - Android: 执行命令 `adb devices` , 检查设备列表中是否包含测试设备
   - iOS：以 `tidevice` 库举例，执行命令 `tidevice list`，检查设备列表中是否包含测试设备

Android 设备连接 Q&A

- 请先安装手机对应品牌的官方驱动，确保能使用电脑对手机进行 USB 调试
- 确保已经打开了手机中的”开发者选项”，并且打开”开发者选项”内的”允许 USB 调试”
- 部分手机需要打开”允许模拟位置”、”允许通过 USB 安装应用”
- 关闭电脑上已经安装的手机助手软件，能避免绝大多数问题，请务必在任务管理器中手工结束手机助手进程

iOS 设备连接 Q&A

- 请先准备一台 macOS ，使用 xcode 部署 iOS-Tagent 成功后，能够在 mac 或 windows 机器上连接到 iOS 手机。请点击链接下载项目代码到本地进行部署。
- mac 环境通过 Homebrew 安装 iproxy `brew install libimobiledevice`
- windows 环境安装itunes

2. 下载安装测试包
   - Android：框架会通过 `config` 中配置的 `packagePath` 自动下载测试包并安装（请确保手机已经打开”允许安装未知来源 “)。也可手动下载安装：下载地址
   - iOS:
     1. 请手动下载演示 APP 进行安装：下载地址
     2. 开 启 wdaproxy：`shell tidevice --udid $udid wdaproxy -B $web_driver_angnt_bundle_id -p $port`

1.8.2 Web

Web 项目，需安装浏览器

```
# 不带参数的运行将安装默认所有浏览器
playwright install
```

```
# 通过提供一个参数来安装特定的浏览器
playwright install webkit
```

```
# 查看支持安装的浏览器
playwright install --help
```

1.8.3 语言环境

检查执行环境中是否已安装了对应的字体，以 unbutu 系统为例，执行中文用例前，需安装对应的字库

```
sudo apt-get install ttf-wqy-microhei #文泉驿-微米黑
sudo apt-get install ttf-wqy-zenhei #文泉驿-正黑
sudo apt-get install xfonts-wqy #文泉驿-点阵宋体
```

1.9 运行

1.9.1 运行参数

在终端输入以下内容来查看 **flybirds** 运行项目时支持的操作

```
flybirds run --help
```

- **run**

  执行 features 目录下所有的 feature 文件

  ```
  cd {PATH_TO_PROJECT_FOLDER}
  flybirds run  # 运行所有feature
  flybirds run -P features/test/android  # 运行所有android feature
  flybirds run -P features/test/ios # 运行所有ios feature
  ```

- **–path, -P**

  指定需要执行的 feature 集合，可以是目录，也可以指定到具体 feature 文件，默认是‘**features**’目录.

  示例:

  ```
  flybirds run -P ./features/test/demo.feature
  ```

- **–tag, -T**

  运行有特定 tag 的场景，多个用逗号隔开，‘-’开头表示不运行包含此 tag 的场景

  ```
  flybirds run -T tag1,tag2,-tag3,tag4
  ```

- **–format, -F**

  指定生成测试结果的格式，默认是 json.

  示例:

```
#默认
flybirds run --format=json
```

- **–report, -R TEXT(可选)**

指定生成报告的地址，不指定时默认为 report 目录下随机生成的一个文件.

示例：

```
#mac 自定义生成报告地址
flybirds run --report report/curent/report.json

#windows 自定义生成报告地址
flybirds run --report report\curent\report.json
```

- **–define, -D TEXT(可选)**

传入用户自定义的参数:

作用：覆盖 config 配置文件中的相应配置的值，比如：

```
# 通过参数切换执行平台Android、iOS、Web
flybirds run --define platform=web
```

- **–rerun /–no-rerun (可选)**

指定失败的场景是否需要重新运行，默认是‘True’, 失败后会自动重跑。

示例：

```
#失败场景不重跑
flybirds run --no-rerun
```

- **–html/–no-html (可选)**

指定 case 执行完成后是否生成 html 测试报告，默认是‘True’, 执行完成后自动生成结果测试报告。

示例：

```
#不生成测试报告
flybirds run --no-html
```

- **–processes, -p INTEGER(可选)**

  指定并发执行时开启进程的最大数量。默认是 4 。

  注意：此命令只在 **web** 平台执行时有效。

示例：

```
flybirds run --path features -p 5
```

1.9.2 运行演示

为了帮助使用，项目创建时，会生成中英文的 Android、iOS、Web 演示 feature，方便用户参考。

```
features/test/
features/test/android
features/test/android/cn/everything.feature
features/test/android/en/everything.feature
features/test/ios
features/test/ios/cn/everything.feature
features/test/ios/en/everything.feature
```

以“Android”为例

1. 执行命令 `adb devices`，检查设备列表中是否包含测试设备
2. 检查 `flybirds_config` 中配置的 `deviceId` 和 `packageName` 是否正确
3. 如你选择手动安装测试包，需检查测试设备中，测试包是否已正确安装
4. 开始运行

```
cd {PATH_TO_PROJECT_FOLDER}
flybirds run -P features/test/android
```

框架会通过 `flybirds_config` 中配置的 `packagePath` 自动下载测试包并安装（请确保手机已经打开”允许安装未知来源“）
运行结果如下

```
11 features passed, 0 failed, 0 skipped, 0 untested
23 scenarios passed, 0 failed, 0 skipped, 0 untested
117 steps passed, 0 failed, 0 skipped, 0 undefined, 0 untested
Took 5m21.300s
================================================================================
    Multiple Cucumber HTML report generated in:

    /Users/test/my_first_project/report/7eb9162a-9d42-4fde-a5d7-d8d4bca7a8d8/index.
↪html
================================================================================
```

1.10 项目细节

1.10.1 项目结构

- config：配置文件
- features：测试用例 feature 文件
- pscript：自定义扩展
- report：测试报告

1.10.2 features 目录

基础目录结构如下

- test：存放 feature 文件，这些文件使用自然语言编写，最好由软件项目中的非技术业务、产品人员参与者编写。
- steps：存放场景中使用的 step 语句实现，“steps.py”中加载了所有的 step 语句模版

```
features/
features/test/
features/test/everything.feature
features/steps/
features/steps/steps.py
```

复杂些的目录结构参考如下

```
features/
features/test/
features/test/list.feature
features/test/buy.feature
features/test/detail.feature
features/steps/
features/steps/steps.py
```

1.10.3 feature 文件

> feature 文件包含用户动作，行为特征描述及预期结果的文本，行为特征部分使用 Gherkin 语言编写。

feature 文件，也称为功能文件，有两个目的：文档和自动化测试。

以关键字开头（“功能”、“场景”、“场景大纲”、“当”、“而且”、“那么”……），文件中的任何位置都允许使用注释行。

功能 (Feature) 是被测试功能的一些合理的描述性标题，由场景组成。他们可以选择有一个描述、一个背景和一组标签。

背景 (Background) 由一系列类似于场景的步骤组成。它允许您向功能的场景添加一些上下文。在此功能的每个场景之前执行。

场景 (Senario) 标题应该是被测试场景的合理描述性标题，由一系列给定条件的步骤组成

场景大纲 (Senario Outline) 包含功能的详细描述，可以有一组预期条件和结果来配合您的场景步骤

以下是中文 feature 例子

```
# language: zh-CN
 功能: 首页机票模块、搜索框
       1. 进入首页后能够展示机票按钮和搜索框
       2. 点击机票按钮后显示特价机票模块
       3. 在搜索框中输入上海，点击搜索后，能够展示上海机票信

 背景:
     当   启动APP[ctrip.android.view]
     而且 页面渲染完成出现元素[text=首页]

  场景: 验证首页机票按钮功能
     当   存在元素[text=机票]
     那么 点击[text=机票]
     那么 页面渲染完成出现元素[text=特价机票]
     那么 全屏截图
     那么 关闭APP


   场景: 验证首页搜索框功能
     当   开始录屏
     那么 在[text=搜索]中输入[上海]
     而且 点击[label=sbtn]
     那么 页面渲染完成出现元素[text=上海机票]
     那么 结束录屏
     那么 关闭APP
```

以下是英文 feature 例子

```
# language: en
 Feature: Homepage ticket module, search box
          1. After entering the homepage, the ticket button and search box can be displayed
          2. Click the ticket button to display the special ticket module
          3. Input "Shanghai" in the search box and click search to display the “Shanghai ticket”
information

 Background:
     When start app [ctrip.android.view]
     And  page rendering complete appears element [text=Homepage]

  Scenario: verify the function of the homepage ticket button
     When exist element [text=ticket]
     And  click [text=ticket]
     Then page rendering complete appears element [text=SpecialTicket]
     Then screenshot
     Then close app

   Scenario: verify the function of the homepage search box
     When start record
     Then in [text=search] input [shanghai]
     And  click [label=sbtn]
     Then page rendering complete appears element [text=ShangHaiTicket]
     Then stop record
     Then close app
```

1.10.4 配置参数

必须配置项

进行**移动端**测试时，必须配置项：`deviceId` 、`packageName`。在 IOS 设备上测试时，必须额外配置 `webDriverAgent`。

flybirds_config.json

- `packageName`

  app 的 packageName, 示例为 ctrip 的 packageName, 此配置必须填写

- `packagePath`

  下载安装 app 的地址，项目运行时会自动从该地址下载并安装在测试设备中，为空时不下载

- `overwriteInstallation`

  是否每次运行前覆盖安装测试包，默认：” True”

- `uniqueTag`

  app 的唯一性标识，比如:clientId, 默认：” 000”

- `defaultUser`

  运行前需要全局登录时会使用此用户名, 默认：“null”

- defaultPassword

  运行前需要全局登陆时会用此密码, 默认：“null”

- ocrLang

  ocr 扫描语言，默认：”ch”

- deviceId

  示例为设备的序列号 Android 设备使用“adb devices”获取, 默认：”10.5.170.85:5555”

- platform

  项目 case 执行的平台，目前支持 android 、ios 和 web, 不填时默认为：android

- webDriverAgent

  设备里 WebDriverAgent 的 BundleID，可通过 tidevice applist 命令查看。连接 IOS 设备时必填。

- headless

  浏览器的运行模式，为 true 时表示浏览器将以**无头**方式运行。platform=web 时必填。默认为：true

- browserType

  支持的浏览器类型：chromium, firefox and webkit。platform=web 时必填。支持同时配置多个值，

  如：”browserType”：[“firefox”,”chromium”,”webkit”]。默认为：”browserType”：[“chromium”]。

- requestInterception

  开启请求拦截。platform=web 时必填。默认为：true

- ignoreOrder

请求报文比对时忽略报文节点的顺序或重复的列表差异。默认为：false。仅在 requestInterception=true 时有效。

- abortDomainList

请求拦截时，终止路由的域名列表。如：”abortDomainList”：[“google.com”]。仅在 requestInterception=true 时有效。

- emulatedDevice

  模拟设备的名称, 如：” emulatedDevice”：“iPhone 6”。仅在 platform=web 时有效。默认为：” emulatedDevice” : null。支持模拟的设备列表：https://github.com/microsoft/playwright/blob/main/packages/playwright-core/src/server/deviceDescriptorsSource.json

- userAgent

  浏览器的网络代理，默认为 null

- width

  浏览器的宽度，仅在 platform=web 时有效。如：”width”：“1080”。默认为：null

- `height`

  浏览器的高度，仅在 `platform=web` 时有效。如：”height”： “1920”。默认为：null

- `locale`

  浏览器的语言，仅在 `platform=web` 时有效。如：”locale”： “de-DE”。默认为：null

- `timezone`

  浏览器的时区，仅在 `platform=web` 时有效。如：”timezone”： “Europe/Berlin”。默认为：null

- `geolocation`

  浏览器的地理位置，仅在 `platform=web` 时有效。如：”geolocation”： “” longitude”：41.890221, “latitude”：12.492348”。默认为：null

- `beforeRunPage`

  在开始测试前对 app 的行为配置，默认时“restartApp”保证测试时页面处于大首页，还有 startApp(启动 app)，stopApp(关闭 app)、None(无任何操作), 默认：”restartApp”

- `scenarioFailPage`

  在测试用例执行失败后对 app 的配置行为，默认是“restartApp”保证当前内存中不会积压太多历史页面，还有 backupPage(返回上一页)，stopApp(停止 app)、None(无任何操作), 默认：“restartApp”

- `scenarioSuccessPage`

  测试用例执行成功后对 app 的配置行为，默认是“None”无任何操作，还有 restartApp(重启 app)，backupPage(返回上一页)，stopApp(停止 app), 默认：”None”

- `beforeRunLogin`

  开始测试前是否需要登陆, 默认：”false”

- `failScreenRecord`

  失败后是否需要关联用例的执行录屏文件到测试报告中, 默认：”true”

- `scenarioScreenRecordTime`

  failScreenRecord 为 ture 开启失败录屏时，录制屏幕的最大时长, 默认：180

- `failRerun`

  失败后是否重新运行, 默认：true

- `maxFailRerunCount`

  失败重新运行所满足的失败个数, 默认：1

- `maxRetryCount`

  失败重试次数, 默认：2

- `waitEleTimeout`

  页面中查找元素的超时时间, 默认：15

- `waitEleDisappear`

  页面中指定元素消失的超时时间, 默认：10

- `clickVerifyTimeout`

  点击操作的判断渲染完成的超时时间, 默认：15

- `useSwipeDuration`

  使用全局配置的滑动时间, 默认：”false”

- `swipeDuration`

  当 useSwipeDuration 为 ture 时，滑动操作的时间为该值, 默认：6

- `usePocoInput`

  输入是否使用 poco 的输入方法，默认使用 airtest, 默认：false

- `afterInputWait`

  输入框输入后的等待时间, 默认：1

- `useSearchSwipeDuration`

  滑动查找中是否使用全局的滑动时间, 默认：false

- `searchSwipeDuration`

  useSearchSwipeDuration 为 ture 时，滑动查找中的所有滑动时间由该值全局设置, 默认：1

- `swipeSearchCount`

  滑动查找元素的最大滑动次数, 默认：5

- `swipeSearchDistance`

  滑动查找中每次滑动的距离, 默认：0.3

- `pageRenderTimeout`

  等待页面渲染完成的时间，语句“页面渲染完成出现元素 [选择器 {, path=false, multiSelector=false, timeout=10}]”中的 timeout 参数的全局配置时间, 默认：35

- `appStartTime`

  APP 启动后等待时间, 默认：6

- `swipeReadyTime`

  滑动开始前的等待时间, 默认：3

- `verifyPosNotChangeCount`

  判断元素位置未发生改变的最大判断次数, 默认：5

- `screenRecordTime`

  录屏时间, 默认：60

- `useSnap`

  是否使用快照查找文案, 默认：true

- `useAirtestRecord`

  使用 airtest 录屏, 默认：true

schema_url.json

用于对多端页面的 `schema` 访问地址进行统一配置

示例：

- 多端页面的 `schema` 访问地址相同：

  “首页”：“ctrip://homepage”

  示例为携程 APP 首页

- 多端页面的 `schema` 访问地址不同

  示例为携程 android、ios、web 端的列表页地址

```
"列表页": {
  "android": "/rn_test/ctrip_list_android/",
  "ios": "/rn_test/ctrip_list_ios/",
  "web": "https://ctrip.test/list"
}
```

ele_locator.json

用于对多端元素的定位方式进行统一配置

示例：

- 多端元素的定位方式相同：

  “元素 1”：“text= 帮助中心”

  示例为【元素 1】的定位方式

- 多端元素的定位方式不同

  示例为【元素 2】在 android、ios、web 端的定位方式

```
"元素2": {
  "android": "text=机票",
  "ios": "label=机票",
  "web": "#s-top-loginbtn"
}
```

1.10.5 Hooks

用户可在以下文件中定义 hooks

```
pscript/dsl/step/hook.py
```

- **before_step_extend(context, step), after_step_extend(context, step)**

  在每个步骤 (step) 之前和之后运行

- **before_scenario_extend(context, scenario), after_scenario_extend(context, scenario)**

  在每个场景 (senario) 之前和之后运行

- **before_feature_extend(context, feature), after_feature_extend(context, feature)**

  在每个功能文件 (feature) 之前和之后运行

- **before_tag_extend(context, tag), after_tag_extend(context, tag)**

  在用给定名称标记 (tag) 的部分之前和之后运行

- **before_all_extend(context), after_all_extend(context)**

  在所有执行之前和之后运行

1.10.6 标签 (Tags)

可以使用 tag 标记不同的场景，方便有选择性的运行。

下面是一个例子

```
# language: zh-CN
 功能: 首页机票模块

  @p1 @flight
  场景: 验证首页机票按钮功能
      当  启动APP[ctrip.android.view]
      而且 页面渲染完成出现元素[text=首页]
      那么 存在元素[text=机票]
      那么 点击[text=机票]
      那么 页面渲染完成出现元素[text=特价机票]
      那么 全屏截图
      那么 关闭APP
```

运行有特定 tag 的场景，多个用逗号隔开

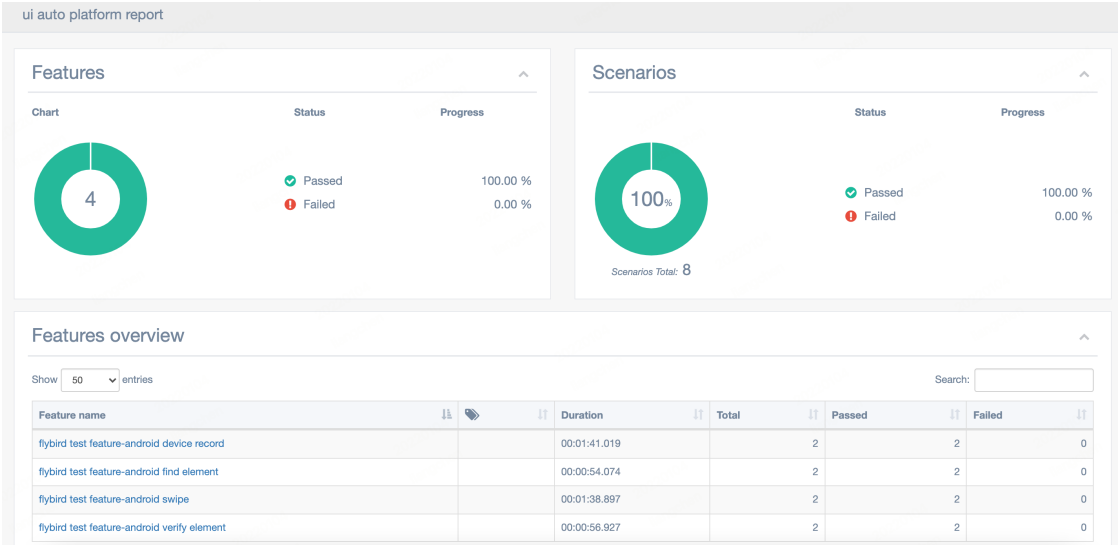
```
flybirds run -T tag1,tag2
```

‘-’开头表示运行不包含某 tag 的场景

```
flybirds run -T -tag
```

1.10.7 报告 (report)

报告包含汇总 Summary 和功能 (feature)、场景 (senario) 的执行结果，对于失败的场景 (senario)，报告中会展示当时的屏幕图像和视频，下面是一个例子。



1.10.8 自定义 DSL step

在编写 Feature 的过程中，可能会遇到提供的公共语句不能满足自身项目的需求，需要自定义语句。比如：需要对接某个内部工具 API，此时需要用到自定义语句功能。

自定义语句功能会用到 python，如果你不了解这门编程语言，也不必要太担心，因为只会使用到最基础的 python 语法，这并不会太难。

使用方法

1. 进入项目目录”psscript/dsl/steps”
2. 新建.py 文件来编写自定义语句
3. 在 feature/steps/steps.py 中 import 该.py 文件

示例代码如下

```
# -*- coding: utf-8 -*-

from behave import step


@step("Create new milestone[{param1}]and[{param2}]")
def create_milestone_with_title(context, param1, param2):
    param1 = param1.strip()
    param2 = param2.strip()
    print(f'I create milestone with title {param1} and {param2}!')
```

对于团队内部通用的自定义功能，可以考虑创建一个 extend package，flybirds 支持动态加载，package 命名包含“-flybirds-plugin”即可。

1.10.9 自定义脚本

项目目录 `psscript/custom_handle`: 存放 python 语言自定义的脚本包括自定义 step 语句，对接如 mock 等其他平台，自定义 schema 跳转逻辑，登陆登出，behave 运行时各种钩子函数的扩展，参数处理方法等等

- **custom_handle/operation.py**： 定义一些 app、web 特有的行为。比如，app 的 schema 跳转协议的拼接，登陆，登出，跳转到首页；web 的创建自定义 BrowserContext，获取 MockData 等。

  示例一：web 端自定义 BrowserContext

```
def create_browser_context(browser):
    """
    custom creates a new browser context.
    :param browser: the browser instance
    """
    # For example, adding parameter when create, locale: language, viewport:␣
↪screen size
    context = browser.new_context(record_video_dir="videos",
                                  ignore_https_errors=True,
                                  locale="en",
                                  viewport={"width": 800, "height": 800})
    return context
```

自定义全局变量

1. 项目中自定义全局 key

```
from flybirds.core.global_context import GlobalContext

def set_global_value():
# 设置全局key
   GlobalContext.set_global_cache("order_id", "just a test")
```

2. 在 pscript 包下面的 operation.py 中添加如下方法, 自定义全局 key 获取方法

```
from flybirds.core.global_context import GlobalContext

def get_global_value(v):
"""
 replace with global cache
 """
# order_id 为通过其他step 向global cache 中设置的值
 order_id = GlobalContext.get_global_cache("order_id")
if order_id is not None:
# 替换参数 v 中自定义全局缓存key（规则自己设定，框架不做限制）
 return v.replace("@order_id@", order_id)
# 返回为None 表示不做替换
 return None
```

3. 在 case 中使用自定义的全局 key：order_id

```
场景: 输入
假如 跳转页面到[百度]
而且 在[#kw]中输入[@order_id@]
而且 等待[3]秒
那么 全屏截图
```

1.11 DSL step

1.11.1 DSL step 列表

上面例子中的关键字“当”、“而且”和“那么”部分构成了测试用例的操作步骤，这些操作步骤框架中已经通过 python 实现。

当然在这个架构中, 各端略有不同，主要是各端的平台差异性导致，以下是各端具体支持的 DSL step 列表, 大部分 step 能够适用于多端

DSL step	语义
跳转到 []	跳转到指定的 url 地址
等待 [] 秒	等待一段时间
页面渲染完成出现元素 []	进入新的页面时检查指定元素是否渲染完成
页面扫描完成出现元素 []	进入新的页面时使用 OCR 检查指定元素是
点击 []	点击指定属性的元素
点击文案 []	点击指定文案的元素
点击扫描文案 []	使用 OCR 点击指定文案的元素
点击屏幕位置 [][]	点击屏幕指定位置
在 [] 中输入 []	在指定选择器中输入字符串
在扫描文字 [] 中输入 []	使用 OCR 在指定文字选择器中输入字符串
在 [] 中清空并输入 []	在指定选择器中清空并输入字符串
向 [] 查找 [] 的元素	向指定方向查找指定属性的元素
全屏向 [] 滑动 []	全屏向指定方向滑动指定距离
[] 向 [] 屏幕滑动 []	在指定区域内向指定方向屏幕滑动指定距离
存在 [] 的文案	检查页面中存在指定的字符串
不存在 [] 的文案	检查页面中不存在指定的字符串
存在元素 []	检查页面中存在指定属性的元素
扫描存在 [] 的文案	使用 OCR 检查页面中存在指定的文案
扫描不存在 [] 的文案	使用 OCR 检查页面中不存在指定的文案
扫描包含 [] 的文案	使用 OCR 检查页面中包含指定的文案
切换 OCR 语言 []	切换指定的 OCR 扫描语言
元素 [] 消失	检查页面中指定属性的元素在指定时间内消
文案 [] 消失	检查页面中指定的字符串在规定时间内从页
文案 [] 的属性 [] 为 []	检查页面中指定文案的指定属性为指定值
元素 [] 的属性 [] 为 []	检查页面中指定元素的指定属性为指定值
元素 [] 位置 [] 秒内未变动	检查页面中指定元素的位置在指定时间内未
[] 的文案为 []	检查页面中指定元素的文案等于指定值
[] 的文案包含 []	检查页面中指定元素的文案包含指定值
回到首页	回到首页
全屏截图	保存当前屏幕图像
开始录屏	开始录制视频
开始录屏超时 []	开始录屏并设置超时时间
结束录屏	结束录制视频
连接设备 []	连接测试设备
安装 APP[]	安装 APP
删除 APP[]	删除 APP
启动 APP[]	启动 APP
重启 APP	重启 APP

表 1 – 接上页

DSL step	语义
关闭 App	关闭 App
登录账号 [] 密码 []	使用账号密码进行登录
退出登录	退出系统登录
返回上一页	返回上一页面
在 [] 中向 [] 查找 [] 的元素	在指定选择器的元素内向指定方向滑动查找
在 [] 中选择 []	在 web 页面下拉框元素中选择指定值
存在 [父选择器] 的 [子选择器] 的元素	存在某个父元素，并且该父元素下存在某个
[父选择器] 的 [子选择器] 的文案为 []	存在某个父元素，并且该父元素下某个子元
缓存服务请求 [operation[,operation …]]	缓存该服务的最后一次请求报文到本地。注
移除请求缓存 [operation[,operation …]]	从本地清除该服务的请求缓存报文。注意：
移除所有请求缓存	移除所有请求的缓存报文
监听服务 [operation[,operation …]] 绑定 MockCase[mockCaseId[,mockCaseId …]]	监听相关 operation 的请求并拦截，用 mock
移除服务监听 [operation[,operation …]]	移除 operation 的请求监听注意：operation 是
移除所有服务监听	移除所有请求监听
验证服务请求 [operation] 与 [target_data_path] 一致	比对相关 operation 的缓存报文与 target_data
验证服务非 json 请求 [operation] 与 [target_data_path] 一致	比对相关 operation 的非 json 类型的缓存报
验证服务 [operation] 的请求参数 [target_json_path] 与 [expect_value] 一致	检查相关 operation 的缓存报文体对应参数的
—–	—–

1.11.2 连接设备 []

连接设备 [{param}]

- 支持平台：Android 、IOS
- 语义：连接测试设备
- 例子：连接设备 [10.21.37.123:5555]

1.11.3 安装 APP[]

安装 APP[{param}]

- 支持平台：Android
- 语义：安装 APP
- 例子：安装 APP[/Users/xxx/xxx.apk]

1.11.4 删除 APP[]

删除 APP[{param}]

- 支持平台：Android
- 语义：删除 APP
- 例子：删除 APP[package name]

1.11.5 启动 APP[]

启动 APP[{param}]

- 支持平台：Android、iOS
- 语义：启动 APP
- 例子：启动 APP[package name]

1.11.6 重启 APP[]

重启 app

- 支持平台：Android、iOS
- 语义：重新启动 app

1.11.7 回到首页 []

回到首页

- 语义：跳转到首页
- 注：用户自定义实现, 在 pscript/app/operation.py 文件中实现 to_home() 方法

1.11.8 登录账号 [] 密码 []

登录账号 [{param1}] 密码 [{param2}]

- 语义：指定用户名、密码登录
- 注：用户自定义实现, 在 pscript/app/operation.py 文件中实现 login(user_name, password) 方法

1.11.9 退出登录

退出登录

- 语义: 退出当前登录
- 注：用户自定义实现, 在 pscript/app/operation.py 文件中实现 logout() 方法

1.11.10 存在 [] 的文案

存在 [字符串 {, fuzzyMatch=false, timeout=10}] 的文案

- 支持平台：Android、iOS、Web
- 语义：页面中存在指定的字符串
- timeout 查找的超时时间，优先级：默认值 < flybirds_config.json 中的“waitEleTimeout” < 语句中指定

```
存在[机票]的文案
存在[机票, timeout=10]的文案
存在[.?票, fuzzyMatch=true]的文案
```

1.11.11 不存在 [] 的文案

不存在 [字符串 {, fuzzyMatch=false}] 的文案

- 支持平台：Android、iOS、Web
- 语义：页面中不存在指定的文案

```
不存在[机票]的文案
不存在[.?票, fuzzyMatch=true]的文案
```

1.11.12 文案 [] 消失

文案 [字符串 {, fuzzyMatch=false, timeout=10}] 消失

- 支持平台：Android、iOS、Web
- 语义：指定的字符串在规定时间内从页面消失 timeout 等待消失的超时时间，优先级：默认值 < flybirds_config.json 中的“waitEleDisappear” < 语句中指定

```
文案[机票]消失
文案[.?票, fuzzyMatch=true, timeout=20]消失
```

1.11.13 存在元素 []

存在 [选择器 {, path=false, multiSelector=false, timeout=10}] 的元素

- 支持平台：Android、iOS、Web
- 语义：页面中存在指定选择器的元素
- timeout 查找的超时时间，优先级：默认值 < flybirds_config.json 中的“waitEleTimeout” < 语句中指定

```
存在元素[center_content_layout]
存在元素[text=经济舱]
```

1.11.14 元素 [] 消失

元素 [选择器 {, path=false, multiSelector=false, timeout=10}] 消失

- 支持平台：Android、iOS
- 语义：指定的选择器在规定时间内从页面消失
- timeout 等待消失的超时时间，优先级：默认值 < flybirds_config.json 中的“waitEleDisappear” < 语句中指定

```
元素[center_content_layout]消失
元素[text=机票]消失
元素[机票→第1个兄弟节点, path=true, timeout=15]消失
```

1.11.15 [] 的文案为 []

[选择器 {, path=false, multiSelector=false, timeout=10}] 的文案为 [字符串 {, dealMethod=name}]

- 支持平台：Android、iOS、Web
- 语义：指定选择器的元素的文案为指定的字符串
- timeout 查找的超时时间，优先级：默认值 < flybirds_config.json 中的“waitEleTimeout” < 语句中指定

```
[text=机票]的文案为[机票]
[textMatches=.?经济舱, timeout=15]的文案为[经济舱, dealMethod=trim_prefix]
[textMatches=.?经济舱并且visible=True, multiSelector=true, timeout=15]的文案为[经济舱,
↪ dealMethod=trim_prefix]
```

1.11.16 [] 的文案包含 []

[选择器 {, path=false, multiSelector=false, timeout=10}] 的文案包含 [字符串 {, dealMethod=name}]

- 支持平台：Android、iOS、Web
- 语义：指定选择器的元素的文案包含指定的字符串
- timeout 查找的超时时间，优先级：默认值 < flybirds_config.json 中的“waitEleTimeout” < 语句中指定

```
[text=机票]的文案包含[票]
[textMatches=.?经济舱, timeout=15]的文案包含[经济舱]
[textMatches=.?经济舱并且visible=True, multiSelector=true, timeout=15]的文案包含[经济,
↪ dealMethod=trim_prefix]
```

1.11.17 页面渲染完成出现元素 []

页面渲染完成出现元素 [选择器 {, path=false, multiSelector=false, timeout=10}]

- 支持平台：Android、iOS、Web
- 语义：进入新的页面时通过指定选择器的元素出现在页面上来判断页面渲染完成
- timeout 查找的超时时间，优先级：默认值 < flybirds_config.json 中的“pageRenderTimeout” < 语句中指定

```
页面渲染完成出现元素[text=机票]
页面渲染完成出现元素[center_content_layout, timeout=15]
页面渲染完成出现元素[center_content_layout, timeout=40]
```

1.11.18 点击文案 []

点击文案 [字符串 {, fuzzyMatch=false, timeout=10, verifyEle=null, verifyIsPath=false, verifyIsMultiSelector=false, verifyTimeout=10, verifyAction=null}]

- 支持平台：Android、iOS、Web
- 语义：点击页面上指定的字符串
- timeout 查找“字符串”的超时时间，优先级：默认值 < flybirds_config.json 中的“waitEleTimeout” < 语句中指定
- verifyEle 点击后如果有局部渲染，使用该属性指定的选择器代表的元素的相关信息判断
- verifyIsPath：指定 verifyEle 是否是 path 类型的选择器
- verifyIsMultiSelector：指定 verifyEle 是否是多属性类型的选择器
- verifyTimeout 判断点击操作的渲染是否完成的超时时间，优先级：默认值 < flybirds_config.json 中的“clickVerifyTimeout” < 语句中指定

- verifyAction： verifyEle 代表的元素发生特定类型的变化时表示点击后的渲染完成，
- position/text/appear/disappear: 位置发生变化/文案发生变化/出现在页面上/从页面消失

```
点击文案[机票]
点击文案[.?票, fuzzyMatch=true, timeout=15]
点击文案[机票, verifyEle=center_content_layout, verifyAction=position]
点击文案[机票, verifyEle=text=筛选并且type=textView, verifyIsMultiSelector=true,␣
↪verifyAction=position]
```

1.11.19 点击 []

点击 [选择器 {, path=false, multiSelector=false, timeout=10, verifyEle=null, verifyIsPath=false, verifyIsMultiSelector=false, verifyTimeout=10, verifyAction=null}]

- 支持平台：Android、iOS、Web
- 语义: 点击页面上指定选择器的元素
- timeout 查找“字符串”的超时时间，优先级：默认值 < flybirds_config.json 中的“waitEleTimeout” < 语句中指定
- verifyEle 点击后如果有局部渲染，使用该属性指定的选择器代表的元素的相关信息判断
- verifyIsPath：指定 verifyEle 是否是 path 类型的选择器
- verifyIsMultiSelector：指定 verifyEle 是否是多属性类型的选择器
- verifyTimeout 判断点击操作的渲染是否完成的超时时间，优先级：默认值 < flybirds_config.json 中的“clickVerifyTimeout” < 语句中指定
- verifyAction：verifyEle 代表的元素发生特定类型的变化时表示点击后的渲染完成，
- position/text/appear/disappear: 位置发生变化/文案发生变化/出现在页面上/从页面消失

```
点击[text=机票]
点击[textMatches=.?票, timeout=15]
点击[center_content_layout, verifyEle=center_content_layout, verifyAction=position]
点击[testId, verifyEle=text=筛选并且type=textView, verifyIsMultiSelector=true,␣
↪verifyAction=position]
```

1.11.20 点击屏幕位置 []

点击屏幕位置 [{x},{y}]

```
点击屏幕位置[200,100]
```

文案 [字符串 {, fuzzyMatch=false, timeout=10}] 的属性 [属性名 {, dealMethod=name}] 为 {属性值}

- 支持平台：Android、iOS、Web
- 语义：页面中指定字符串对应的元素的指定的属性的值为指定的值
- timeout 查找“字符串”的超时时间，优先级：默认值 < flybirds_config.json 中的“waitEleTimeout” < 语句中指定

```
文案[机票]的属性[text]为机票
文案[机票, timeout=15]的属性[text, dealMethod=trim_last]为机
```

1.11.21 元素 [] 的属性 [] 为 []

元素 [选择器 {, path=false, multiSelector=false, timeout=10}] 的属性 [属性名 {, dealMethod=name}] 为 {属性值}

- 支持平台：Android、iOS、Web
- 语义：页面中指定选择器的元素的指定的属性的值为指定的值
- timeout 查找“字符串”的超时时间，优先级：默认值 < flybirds_config.json 中的“waitEleTimeout” < 语句中指定

```
元素[text=机票]的属性[text]为机票
元素[text=机票, timeout=15]的属性[text, dealMethod=trim_last]为机
```

1.11.22 在 [] 中输入 []

在 [选择器 {, path=false, multiSelector=false, timeout=10}] 中输入 [文案 {, pocoInput=false, afterInputWait=1}]

- 支持平台：Android、iOS、Web
- 语义：在指定选择器的元素中输入指定的文案
- timeout 查找“字符串”的超时时间，优先级：默认值 < flybirds_config.json 中的“waitEleTimeout” < 语句中指定
- pocoInput: 是否使用 poco 的输入方法，默认使用的是 airtest 的输入方法，优先级：默认值 < flybirds_config.json 中的“usePocoInput” < 语句中指定

- afterInputWait: 输入完成的休眠时间，优先级：默认值 < flybirds_config.json 中的“afterInputWait” < 语句中指定

```
在[inputEleId]中输入[上海]
在[type=InputView]中输入[用户名, pocoInput=true, afterInputWait=5]
```

1.11.23 [] 向 {上/下/左/右} 屏幕滑动 []

[选择器 {, path=false, multiSelector=false, timeout=10}] 向 {上/下/左/右} 屏幕滑动 [滑动距离 {, startX=0.5, startY=0.5, duration=null, readyTime=null}]

- 支持平台：Android、iOS、Web
- 语义：在指定选择器的滑动容器内向指定方向滑动指定距离
- **注意：Android、iOS，是模拟手指滑动方向，举例：向上滑动，屏幕滚动轴向下，此时可以看到屏幕下方的内容**
- timeout 查找“字符串”的超时时间，优先级：默认值 < flybirds_config.json 中的“waitEleTimeout” < 语句中指定
- startX: 在容器中滑动起始坐标的 X 轴的坐标值，<=1 代表百分比，>1 代表像素点
- startY: 在容器中滑动起始坐标的 Y 轴的坐标值，<=1 代表百分比，>1 代表像素点
- duration: 每次滑动的时间，优先级：默认值 < flybirds_config.json 中的“swipeDuration” < 语句中指定
- readyTime: 滑动开始前的等待时间，优先级：默认值 < flybirds_config.json 中的“swipeReadyTime” < 语句中指定

```
[containerEleId]向左屏幕滑动[0.1]
[containerEleId]向上屏幕滑动[100, duration=5, readyTime=3]
[containerEleId]向上屏幕滑动[100, startX=0.2, startY=0.4, duration=5, readyTime=3]
```

1.11.24 全屏向 {上/下/左/右} 滑动 []

全屏向 {上/下/左/右} 滑动 [滑动距离 {, startX=0.5, startY=0.5, duration=null, readyTime=null}]

- 支持平台：Android、iOS、Web
- **注意：Android、iOS，是模拟手指滑动方向，举例：向上滑动，屏幕滚动轴向下，此时可以看到屏幕下方的内容**
- 语义：以全屏为容器向指定方向滑动指定距离
- startX: 在全屏中滑动起始坐标的 X 轴的坐标值，<=1 代表百分比，>1 代表像素点
- startY: 在全屏中滑动起始坐标的 Y 轴的坐标值，<=1 代表百分比，>1 代表像素点
- duration: 每次滑动的时间，优先级：默认值 < flybirds_config.json 中的“swipeDuration” < 语句中指定

- readyTime: 滑动开始前的等待时间，优先级：默认值 < flybirds_config.json 中的“swipeReadyTime” < 语句中指定

```
全屏向上滑动[0.05]
全屏向下滑动[0.4, readyTime=3, duration=2]
```

1.11.25 在 [] 中向下查找 [] 的元素

在 [选择器 {, path=false, multiSelector=false, timeout=10}] 中向 {上/下/左/右} 查找 [选择器 {, path=false, multiSelector=false, swipeCount=5, startX=0.5, startY=0.5, distance=0.3, duration=null}] 的元素

- 支持平台：Android、iOS、Web
- 语义：在指定选择器的元素内向指定方向滑动查找指定选择器的元素
- timeout 查找“字符串”的超时时间，优先级：默认值 < flybirds_config.json 中的“waitEleTimeout” < 语句中指定
- swipeCount: 滑动查找最大滑动次数，超过这个值的滑动操作后还未在页面中找到指定元素则失败，优先级：默认值 < flybirds_config.json 中的“swipeSearchCount” < 语句中指定
- startX: 在全屏中滑动起始坐标的 X 轴的坐标值，<=1 代表百分比，>1 代表像素点
- startY: 在全屏中滑动起始坐标的 Y 轴的坐标值，<=1 代表百分比，>1 代表像素点
- duration: 每次滑动的时间，优先级：默认值 < flybirds_config.json 中的“searchSwipeDuration” < 语句中指定
- distance：滑动查找中每次滑动的距离，优先级：默认值 < flybirds_config.json 中的“swipeSearchDistance” < 语句中指定

```
在[containerId]中向下查找[text=机票]的元素
在[containerId]中向下查找[testId, distance=0.5, duration=2, swipeCount=8]
```

1.11.26 向 {上/下/左/右} 查找 [] 的元素

向 {上/下/左/右} 查找 [选择器 {, path=false, multiSelector=false, swipeCount=5, startX=0.5, startY=0.5, distance=0.3, duration=null}] 的元素

- 支持平台：Android、iOS、Web
- 语义：在全屏向指定方向滑动查找指定选择器的元素
- swipeCount: 滑动查找最大滑动次数，超过这个值的滑动操作后还未在页面中找到指定元素则失败，优先级：默认值 < flybirds_config.json 中的“swipeSearchCount” < 语句中指定
- startX: 在全屏中滑动起始坐标的 X 轴的坐标值，<=1 代表百分比，>1 代表像素点
- startY: 在全屏中滑动起始坐标的 Y 轴的坐标值，<=1 代表百分比，>1 代表像素点

- duration: 每次滑动的时间，优先级：默认值 < flybirds_config.json 中的 “searchSwipeDuration” < 语句中指定
- distance：滑动查找中每次滑动的距离，优先级：默认值 < flybirds_config.json 中的 “swipeSearchDistance” < 语句中指定

```
向下查找[text=机票]的元素
向下查找[testId, distance=0.5, duration=2, swipeCount=8]
```

1.11.27 元素 [] 位置 [] 秒内未变动

元素 [选择器 {, path=false, multiSelector=false, timeout=10}] 位置 [time{, verifyCount=5}] 秒内未变动

- 支持平台：Android、iOS
- 语义：指定选择器的元素在指定时间位置未发生变化，目的是判断页面未处于滑动状态
- timeout 查找 “字符串” 的超时时间，优先级：默认值 < flybirds_config.json 中的 “waitEleTimeout” < 语句中指定
- verifyCount: 最大判断次数，优先级：默认值 < flybirds_config.json 中的 “verifyPosNotChangeCount” < 语句中指定

1.11.28 开始录屏超时 []

开始录屏超时 [time]

- 支持平台：Android、iOS
- 语义：开始录制屏幕，到超时时间未停止则停止录屏

1.11.29 开始录屏

开始录屏

- 支持平台：Android、iOS
- 语义：开始录制屏幕，使用默认的超时时间（在配置文件中配置）

1.11.30 结束录屏

结束录屏

- 支持平台：Android、iOS、Web
- 语义：结束录制屏幕，并将视频文件关联到报告中

1.11.31 等待 [] 秒

等待 [time] 秒

- 支持平台：Android、iOS、Web
- 语义：执行暂停指定时间

1.11.32 全屏截图

全屏截图 *

- 支持平台：Android、iOS、Web
- 语义：截取当前屏幕快照并关联到报告中

1.11.33 跳转到 []

跳转到 [页面名称]

- 支持平台：Android、Web
- 语义：通过 schema 跳转到指定页面，页面名称在 config/schema_url.json 中以 “页面名称: 页面 schemaUrl” 的形式维护

```
跳转到[首页]
```

1.11.34 缓存服务请求 []

缓存服务请求 [operation[,operation …]]

- 支持平台：Web
- 语义：缓存该服务的最后一次请求报文到本地。注意：operation 是 url 最后一个\ 和？中间的字符串，即请求名

```
//示例1:
缓存服务请求[getRecommendHotelList]
//示例2:
缓存服务请求[getRecommendHotelList,writecookie]
```

1.11.35 移除请求缓存 []

移除请求缓存 [operation[,operation …]]

- 支持平台：Web
- 语义：从本地清除该服务请求的缓存报文。注意：operation 是 url 最后一个\ 和？中间的字符串，即请求名

```
//示例1:
移除请求缓存[getRecommendHotelList]
//示例2:
移除请求缓存[getRecommendHotelList,writecookie]
```

1.11.36 移除所有请求缓存

移除所有请求缓存

- 支持平台：Web
- 语义：移除所有请求的缓存报文

1.11.37 监听服务 [] 绑定 MockCase []

监听服务 [operation[,operation …]] 绑定 MockCase[mockCaseId[,mockCaseId …]]

- 支持平台：Web
- 语义：监听相关 operation 的请求并拦截，用 mockCaseId 的返回报文进行替换。注意：operation 是 url 最后一个\ 和？中间的字符串
- **MockCase 配置**：服务监听 `step` 语句的 `mock` 数据支持通过 2 种方式来获取：**json 文件配置**和 **函数调用**。
  - **json 文件配置**：如以下示例一。具体设置方式及格式可以参考 **Demo** 项目 **mockCaseData** 目录下的 json 文件。此方式需要注意，对应 `response` 的 `mockCaseId`（json key，如示例一中的 `4245512`）在整个 `mockCaseData` 目录下需要是唯一的，否则该 mock 数据会被其他具有相同 `key` 的数据覆盖掉。
  - **函数调用**：自定义处理与获取 `MockData`。此种方式需要在 **Demo** 项目的 **pscript/custom_handle/operation.py** 文件中实现 `get_mock_case_body(mock_case_id)`

扩展方法。MockCase 绑定的报文优先以自定义扩展方法的返回结果为主。当自定义扩展方法返回结果为 None 时，框架会尝试查找项目 **mockCaseData** 目录下的所有 json 文件，并返回 json 文件中 mock_case_id 对应的 mock 数据。**Mock 数据配置示例一：json 文件配置**

```
{
  "4245512": {
    "count": 101,
    "results": [
      {
        "id": 10,
        "name": "test-狮子王",
        "alias": "The Lion King",
        "cover": "https://p0.meituan.net/movie/
↪27b76fe6cf3903f3d74963f70786001e1438406.jpg@464w_644h_1e_1c",
        "categories": [
          "动画",
          "歌舞",
          "冒险"
        ],
        "published_at": "1995-07-15",
        "minute": 89,
        "score": 9.0,
        "regions": [
          "美国"
        ]
      }
    ]
  }
}
```

语法使用示例:

```
//示例1:
监听服务[movie]绑定MockCase[4245512]
//示例2:
监听服务[movie,testList]绑定MockCase[4245512,123456]
```

1.11.38 移除服务监听 []

移除服务监听 [operation[,operation …]]

- 支持平台：Web
- 语义：移除 operation 的请求监听注意：operation 是 url 最后一个\ 和？中间的字符串

```
//示例1:
移除服务监听[movie]
//示例2:
移除服务监听[movie,testList]
```

1.11.39 移除所有服务监听

移除所有服务监听

- 支持平台：Web
- 语义：移除所有请求监听

1.11.40 验证服务请求 [] 与 [] 一致

验证服务请求 [operation] 与 [target_data_path] 一致

- 支持平台：Web
- 语义：比对相关 operation 的缓存报文与 target_data_path 对应的文件内容注意：operation 是 url 最后一个\ 和？中间的字符串

```
验证服务请求[getRecommendHotelList]与[compareData/getRecommendHotelList.json]一致
```

1.11.41 验证服务非 json 请求 [] 与 [] 一致

验证服务非 json 请求 [operation] 与 [target_data_path] 一致

- 支持平台：Web
- 语义：比对相关 operation 的非 json 类型的缓存报文与 target_data_path 对应的文件内容注意：operation 是 url 最后一个\ 和？中间的字符串

```
验证服务非json请求[writecookie]与[compareData/writecookie.txt]一致
```

1.11.42 验证服务 [] 的请求参数 [] 与 [] 一致

验证服务 [operation] 的请求参数 [target_json_path] 与 [expect_value] 一致

- 支持平台：Web
- 语义：检查相关 operation 的缓存报文体对应参数的值与给定的期望值是否一致注意：operation 是 url 最后一个\ 和？中间的字符串

```
验证服务[getRecommendHotelList]的请求参数[head.syscode]与[PC]一致
```

- **配置忽略节点：** 服务报文比对支持设置**忽略节点**，包括**具体路径**和**正则表达式**。具体设置方式及格式可以参考 **Demo** 项目 **interfaceIgnoreConfig** 目录下的 json 文件。**示例：**

```
{
  "getRecommendHotelList": [
    "head.cid",
    "regex: root\\['head'\\]\\['extension'\\]\\[\\d+\\]\\['value'\\]"
  ],
  "writecookie": [
    "token"
  ]
}
```

说明：

- `getRecommendHotelList`、`writecookie` 为服务请求的请求名 `operation`
- `head.cid` 为服务 `getRecommendHotelList` 请求体的具体节点路径。与文件进行报文比对时，该节点会被忽略。
- `regex: root\\['head'\\]\\['extension'\\]\\[\\d+\\]\\['value'\\]` 是一个正则表达式。`getRecommendHotelList` 请求体中所有匹配该路径的节点在比对时都将被忽略。**注意：** 配置正则表达式时，请在字符串前用 `regex:` 进行标注申明。

1.11.43 设置 cookie 名称 [] 值 [] 网址 []

设置浏览器 cookie

- 支持平台：Web
- 语义：设置浏览器 cookie

```
设置cookie 名称[str] 值[str] 网址[str]
```

1.11.44 切换目标页面标题 [{title}] 链接 [{url}]

设置浏览器 tab

- 支持平台：Web
- 语义：设置浏览器 tab

```
切换目标页面标题[{title}]链接[{url}]
```

1.11.45 执行 js[{param}]

执行 js

- 支持平台：Web
- 语义：执行 js

```
执行js[{param}]
```

1.11.46 对比目标元素的链接 [{target_url}] 与文本内容 [{target_ele}] 和比较元素的链接 [{compared_url}] 与文本匹配内容 [{compared_ele}]

dom 元素文本对比

- 支持平台：Web
- 语义：dom 元素文本对比

```
对比目标元素的链接[{target_url}]与文本内容[{target_ele}]和比较元素的链接[{compared_
↪url}]与文本匹配内容[{compared_ele}]
```

1.11.47 对比目标图片 [{target_picture_path}] 和比较图片 [{compared_picture_path}]

图像对比

- 支持平台：Web
- 语义：图像对比

```
对比目标图片[{target_picture_path}]和比较图片[{compared_picture_path}]
```

1.11.48 鼠标悬浮 [{selector}]

鼠标悬浮

- 支持平台：Web
- 语义：对指定元素做悬浮操作

```
悬浮[{selector}]
```

1.12 Android 端例子

使用 cli 脚手架创建项目后，可在目录 `{your project}/features/test/android/`中找到以下例子

1.12.1 点击

```
# language: zh-CN
功能: flybirds功能测试-android click

  场景: 验证点击--点击屏幕位置
    当  启动APP[ctrip.android.view]
    而且 点击屏幕位置[580,1200]
    而且 等待[5]秒
    那么 全屏截图
    那么 关闭App


  场景: 验证点击--点击元素
    当  启动APP[ctrip.android.view]
    而且 页面渲染完成出现元素[text=机票]
    而且 点击[text=机票]
    那么 全屏截图


  场景: 验证点击--点击并输入
    当  启动APP[ctrip.android.view]
    而且 页面渲染完成出现元素[text=搜索]
    而且 在[text=搜索]中输入[flybirds]
    而且 等待[10]秒
    那么 全屏截图
```

1.12.2 查找元素

```
# language: zh-CN
功能: flybirds功能测试-android find element

  场景: 验证元素操作--在页面中查找
     当  启动APP[ctrip.android.view]
     而且 页面渲染完成出现元素[text=机票]
     那么 在[android.widget.LinearLayout]中向下查找[text=机票]的元素


   场景: 验证元素操作--在全屏查找
     当  启动APP[ctrip.android.view]
     而且 页面渲染完成出现元素[text=机票]
     那么 向下查找[text=租车]的元素
```

1.12.3 录屏

```
# language: zh-CN
功能:  flybirds功能测试-android device record

   场景: 验证设备录屏--录屏
     当  启动APP[ctrip.android.view]
     而且 页面渲染完成出现元素[text=机票]
     而且 开始录屏
     而且 点击[text=机票]
     而且 等待[10]秒
     那么 结束录屏
     那么 关闭App


   场景:验证设备录屏--录屏超时
     当  启动APP[ctrip.android.view]
     而且 页面渲染完成出现元素[text=机票]
     而且  开始录屏超时[50]
     而且  点击文案[机票]
     而且  等待[10]秒
     那么 结束录屏
     那么 关闭App
```

1.12.4 滑动

```
# language: zh-CN
功能: flybirds功能测试-android swipe

  场景: 验证滑动--元素滑动
    当 启动APP[ctrip.android.view]
    而且 页面渲染完成出现元素[text=租车]
    而且 [text=租车]向上滑动[600]
    而且 等待[5]秒
    那么 元素[text=租车]位置[5]秒内未变动
    那么 存在元素[text=搜索]
    那么 存在[搜索]的文案


  场景: 验证滑动--全屏滑动
    当 启动APP[ctrip.android.view]
    而且 页面渲染完成出现元素[text=租车]
    而且 全屏向上滑动[600, readyTime=3, duration=2]
    那么 不存在[text=机票]的元素
    那么 不存在[升级攻略]的文案
    那么 元素[text=机票]消失
    那么 文案[机票]消失
```

1.12.5 验证元素

```
# language: zh-CN
功能: flybirds功能测试-android verify element

  场景: 验证元素--元素文案
    当 启动APP[ctrip.android.view]
    而且 页面渲染完成出现元素[text=机票]
    那么 [text=机票]的文案为[机票]
    那么 [text=机票]的文案包含[机]


  场景: 验证元素--元素属性
    当 启动APP[ctrip.android.view]
    而且 页面渲染完成出现元素[text=机票]
    那么 元素[text=机票]的属性[text]为机票
```

1.13 iOS 端例子

使用 cli 脚手架创建项目后，可在目录 `{your project}/features/test/ios/`中找到以下例子

1.13.1 点击

```
# language: zh-CN
功能： flybirds功能测试-ios click

  场景: 验证点击--点击屏幕位置
    当  启动APP[com.ctrip.inner.wireless]
    而且 点击屏幕位置[580,1200]
    而且 等待[5]秒
    那么 全屏截图
    那么 关闭App


  场景: 验证点击--点击元素
    当  启动APP[com.ctrip.inner.wireless]
    而且 页面渲染完成出现元素[label=机票]
    而且 点击[label=机票]
    那么 全屏截图


  场景: 验证点击--点击并输入
    当  启动APP[com.ctrip.inner.wireless]
    而且 页面渲染完成出现元素[label=搜索]
    而且 在[label=搜索]中输入[flybirds]
    而且 等待[10]秒
    那么 全屏截图
```

1.13.2 查找元素

```
# language: zh-CN
功能: flybirds功能测试-ios find element

  场景: 验证元素操作--在页面中查找
    当  启动APP[com.ctrip.inner.wireless]
    而且 页面渲染完成出现元素[label=机票]
    那么 在[ScrollView]中向下查找[label=租车]的元素
```

(续下页)

(接上页)

```
  场景: 验证元素操作--在全屏查找
    当  启动APP[com.ctrip.inner.wireless]
    而且 页面渲染完成出现元素[label=机票]
    那么 向下查找[label=租车]的元素
```

1.13.3 滑动

```
# language: zh-CN
功能: flybirds功能测试-ios swipe

  场景: 验证滑动--元素滑动
    当  启动APP[com.ctrip.inner.wireless]
    而且 页面渲染完成出现元素[label=租车]
    而且 [label=租车]向上滑动[600]
    而且 等待[5]秒
    那么 元素[label=租车]位置[5]秒内未变动
    那么 存在元素[label=租车]
    那么 存在[租车]的元素


  场景: 验证滑动--全屏滑动
    当  启动APP[com.ctrip.inner.wireless]
    而且 页面渲染完成出现元素[label=租车]
    而且 全屏向上滑动[600, readyTime=3, duration=2]
    那么 不存在[label=机票]的元素
    那么 不存在[升级攻略]的文案
    那么 元素[label=机票]消失
    那么 文案[搜索]消失
```

1.13.4 验证

```
 # language: zh-CN
功能: flybirds功能测试-ios verify element

  场景: 验证元素--元素文案
    当  启动APP[com.ctrip.inner.wireless]
    而且 页面渲染完成出现元素[label=机票]
    那么 [label=机票]的文案为[机票]
    那么 [label=机票]的文案包含[机]
```

(续下页)

(接上页)

```
场景: 验证元素--元素属性
  当  启动APP[com.ctrip.inner.wireless]
  而且 页面渲染完成出现元素[label=机票]
  那么 元素[label=机票]的属性[label]为机票
```

1.14 OCR & OpenCV 使用例子

- 基于 OCR 进行页面元素识别、定位，解决了部分技术栈（如 Flutter）下使用 Poco 无法对 Android、iOS 进行元素操作的问题, 支持 Android、iOS 端使用
- 基于 OpenCV 进行页面图像级识别和定位

1.14.1 OCR DSL

全屏扫描

基于默认配置对屏幕图像进行 OCR 扫描，获取屏幕中的文案

- 支持平台：Android、iOS
- 语义：基于默认配置对屏幕图像进行 OCR 扫描，获取屏幕中的文案，对屏幕内容进行自动分区

```
全屏扫描
```

全屏扫描 []

全屏扫描 [right_gap_max=, left_gap_max=, height_gap_max=, skip_height_max=]

- 支持平台：Android、iOS
- 语义：在指定的配置下对屏幕图像进行 OCR 扫描和分区
- right_gap_max 右侧元素的最大间距，默认值：0.3, 0 < right_gap_max < 1, 计算时会使用 right_gap_max * 屏幕宽度像素点
- left_gap_max 左侧元素的最大间距，默认值：0.3, 0 < left_gap_max < 1, 计算时会使用 left_gap_max * 屏幕宽度像素点
- height_gap_max 元素上下的最大间距，默认值：0.07, 0 < height_gap_max < 1, 计算时会使用 height_gap_max * 屏幕高度像素点
- skip_height_max 屏幕头部忽略的最大高度，默认值：0.12, 0 < skip_height_max < 1, 计算时会使用 skip_height_max * 屏幕高度像素点

```
全屏扫描[height_gap_max=0.05]
全屏扫描[height_gap_max=0.05, skip_height_max=0.1]
```

点击扫描文案 []

点击扫描文案 [文案内容, fuzzyMatch=]

- 支持平台：Android、iOS
- 语义: 点击页面上指定内容的文案
- fuzzyMatch 基于正则表达式查找文案，默认值：False

```
点击扫描文案[搜索]
点击扫描文案[.票, fuzzyMatch=true]
```

点击区域 []

点击区域 [id=]

- 支持平台：Android、iOS
- 语义: 点击页面上指定的扫描分区, 区域 id 可通过全屏扫描生成的图像查看

```
点击区域[id=1]
```

点击区域 [] 中扫描文案 []

点击区域 [id=] 中扫描文案 [文案内容, fuzzyMatch=]

- 支持平台：Android、iOS
- 语义: 点击页面上指定区域内的文案，区域 id 可通过全屏扫描生成的图像查看
- fuzzyMatch 基于正则表达式查找文案，默认值：False

```
点击区域[id=1]中扫描文案[搜索]
点击区域[id=2]中扫描文案[.票, fuzzyMatch=true]
```

在扫描文字 [] 中输入 []

在扫描文字 [文字内容] 中输入 [文字内容]

- 支持平台：Android、iOS
- 语义: 在指定文字的输入框中，输入文案，输入框中原先的文案会被清空

```
在扫描文字[出发城市]中输入[上海]
```

扫描存在 [] 的文案

扫描存在 [文案内容] 的文案

- 支持平台：Android、iOS
- 语义：检查屏幕中存在指定内容的文案，使用绝对匹配，忽略空格

```
扫描存在[接送机/租车]的文案
```

扫描不存在 [] 的文案

扫描不存在 [文案内容] 的文案

- 支持平台：Android、iOS
- 语义：检查屏幕中不存在指定内容的文案，使用绝对匹配，忽略空格

```
扫描不存在[大兴机场]的文案
```

扫描包含 [] 的文案

扫描包含 [文案内容] 的文案

- 支持平台：Android、iOS
- 语义：基于正则表达式检查屏幕中存在指定内容的文案

```
扫描包含[通知.*]的文案
```

扫描区域 [] 中存在 [] 的文案

扫描区域 [id=] 中存在 [指定内容] 的文案

- 支持平台：Android、iOS
- 语义：检查屏幕指定区域中存在指定内容的文案，使用绝对匹配，忽略空格

```
扫描区域[id=4]中存在[经济舱]的文案
```

扫描区域 [] 中包含 [] 的文案

扫描区域 [id=] 中包含 [指定内容] 的文案

- 支持平台：Android、iOS
- 语义：检查屏幕指定区域中存在指定内容的文案，使用正则表达式匹配

```
扫描区域[id=4]中包含[我的..]的文案
```

页面扫描完成出现文字 []

页面扫描完成出现文字 [指定文案]

- 支持平台：Android、iOS
- 语义：检查屏幕中存在指定的文案，自动循环等待 20 秒，基于正则表达式检查

```
页面扫描完成出现文字[机票]
页面扫描完成出现文字[经济舱..]
```

向 [] 扫描 [] 的文案

向 [上/下/左/右] 扫描 [指定内容] 的文案

- 支持平台：Android、iOS
- 语义：在全屏向指定方向滑动查找指定选择器的元素，基于正则表达式检查

```
向[上]扫描[机票]的文案
向[下]扫描[南方航.*]的文案
```

切换 OCR 语言 []

- 支持平台：Android、iOS
- 语义：切换 OCR 扫描所使用的语言模型库

```
切换OCR语言[fr]
切换OCR语言[ch]
```

OCR 支持的语种及缩写

语种	描述	缩写		语种	描述	缩写
中文	chinese and english	ch		保加利亚文	Bulgarian	bg
英文	english	en		乌克兰文	Ukranian	uk
法文	french	fr		白俄罗斯文	Belarusian	be
德文	german	german		泰卢固文	Telugu	te
日文	japan	japan		阿巴扎文	Abaza	abq
韩文	korean	korean		泰米尔文	Tamil	ta
中文繁体	chinese traditional	chinese_cht		南非荷兰文	Afrikaans	af
意大利文	Italian	it		阿塞拜疆文	Azerbaijani	az
西班牙文	Spanish	es		波斯尼亚文	Bosnian	bs
葡萄牙文	Portuguese	pt		捷克文	Czech	cs
俄罗斯文	Russia	ru		威尔士文	Welsh	cy
阿拉伯文	Arabic	ar		丹麦文	Danish	da
印地文	Hindi	hi		爱沙尼亚文	Estonian	et
维吾尔	Uyghur	ug		爱尔兰文	Irish	ga
波斯文	Persian	fa		克罗地亚文	Croatian	hr
乌尔都文	Urdu	ur		匈牙利文	Hungarian	hu
塞尔维亚文（latin)	Serbian(latin)	rs_latin		印尼文	Indonesian	id
欧西坦文	Occitan	oc		冰岛文	Icelandic	is
马拉地文	Marathi	mr		库尔德文	Kurdish	ku
尼泊尔文	Nepali	ne		立陶宛文	Lithuanian	lt
塞尔维亚文（cyrillic)	Serbian(cyrillic)	rs_cyrillic		拉脱维亚文	Latvian	lv
毛利文	Maori	mi		达尔瓦文	Dargwa	dar
马来文	Malay	ms		因古什文	Ingush	inh
马耳他文	Maltese	mt		拉克文	Lak	lbe
荷兰文	Dutch	nl		莱兹甘文	Lezghian	lez
挪威文	Norwegian	no		塔巴萨兰文	Tabassaran	tab
波兰文	Polish	pl		比尔哈文	Bihari	bh
罗马尼亚文	Romanian	ro		迈蒂利文	Maithili	mai
斯洛伐克文	Slovak	sk		昂加文	Angika	ang

续下页

表 2 – 接上页

语种	描述	缩写		语种	描述	缩写
斯洛文尼亚文	Slovenian	sl		孟加拉文	Bhojpuri	bho
阿尔巴尼亚文	Albanian	sq		摩揭陀文	Magahi	mah
瑞典文	Swedish	sv		那格浦尔文	Nagpur	sck
西瓦希里文	Swahili	sw		尼瓦尔文	Newari	new
塔加洛文	Tagalog	tl		保加利亚文	Goan Konkani	gom
土耳其文	Turkish	tr		沙特阿拉伯文	Saudi Arabia	sa
乌兹别克文	Uzbek	uz		阿瓦尔文	Avar	ava
越南文	Vietnamese	vi		阿瓦尔文	Avar	ava
蒙古文	Mongolian	mn		阿迪赫文	Adyghe	ady

新增配置项

- “ocrLang” // ocr 扫描语言，默认为 ch 简体中文
- paddle_fix.json // 对 PaddleOCR 扫描错误的结果进行补偿矫正

```
{
"{original value}":"{replace value}",
"原始值":"替换值"
}
```

自动区块划分

解决屏幕中存在多个相同文案，导致识别错误的问题，对扫描结果进行区块划分，方便检查、操作指定区块内的文案。

区域 id 和区域内文案，会展示在扫描结果图片中，多个文案以逗号分隔，如：0：机票, 酒店，度假。

获取扫描结果 DSL：全屏扫描，可在报告中获取到。

0: <,北京一上海v
1: △ • 测试ELF商旅配置 • 及时更新缓存测试,周六,周日,周-,周二,周,菌,24,25,27,12-26,2:,¥270,¥2,查价,查价,¥300
2: 筛选 √,仅看直飞,大机型,下午出发,免费托运,当日低价,20:35,23:00,¥300,大兴国际,浦东T2,经济舱1.9折,南方航空 | 中机型,此航班还有厦航共享售卖,展开
3: 超值优选,09:00,11:05,□310,虹桥T2,大兴国际,经济舱2折,南方航空 | 大机型,此航班还有厦航、吉祥共享售卖,+,展开
4: 中转 • 高性价比,1h30m,11:20,16:45,450,转,转,武汉,大兴国际,浦东T2,经济舱,?,推荐排序,时间排序,价格排序

示例如下:

OCR 例子

```
# language: zh-CN
功能: flybirds功能测试-android ocr

   场景: 验证扫描文字
     当   启动APP[ctrip.android.view]
     那么 全屏扫描
     当 页面扫描完成出现文字[机票]
     那么 扫描存在[酒店]的文案
     那么 扫描不存在[机票机票]的文案
     那么 向下扫描[热门城市]的文案
```

(续下页)

(接上页)

```
场景: 验证区域内扫描文字
  当  启动APP[ctrip.android.view]
  那么 全屏扫描
  当  页面扫描完成出现文字[机票]
  那么 扫描区域[id=0]中存在[旅游地图]的文案
  那么 扫描区域[id=3]中包含[接送机.*]的文案

 场景: 验证点击扫描文字
  当  启动APP[ctrip.android.view]
  那么 全屏扫描
  当 页面扫描完成出现文字[机票]
  那么 点击扫描文案[机票]
  而且 等待[1]秒
  那么 页面扫描完成出现文字[阿克苏]
  那么 点击扫描文案[阿克苏]
  而且 等待[1]秒
  而且 全屏向上滑动[300]
  那么 页面扫描完成出现文字[阿克苏]
  那么 点击扫描文案[阿克苏]
  那么 全屏截图

场景: 验证点击区域内扫描文字
  当  启动APP[ctrip.android.view]
  那么 全屏扫描
  当 页面扫描完成出现文字[机票]
  那么 点击区域[id=4]
  而且 等待[3]秒
  而且 全屏扫描[height_gap_max=0.05, skip_height_max=0.1]
  当  扫描区域[id=1]中存在[上海]的文案
  那么 点击区域[id=1]中扫描文案[上海]
  那么 页面扫描完成出现文字[经济舱]

场景: 验证输入
  当  启动APP[ctrip.android.view]
  那么 全屏扫描
  当  页面扫描完成出现文字[目的地/酒店/景点/关键字/航班号]
  那么 在扫描文字[目的地/酒店/景点/关键字/航班号]中输入[上海]
  那么 全屏截图
```

1.14.2 OpenCV DSL

存在图像 []

存在图像 [图片路径]

- 支持平台：Android、iOS
- 语义：检查屏幕中是否存在指定的图像，基于项目目录下的相对路径

```
存在图像[img/ne.png]
```

不存在图像 []

不存在图像 [图片路径]

- 支持平台：Android、iOS
- 语义：检查屏幕中是否存在指定的图像，图片路径基于项目目录下的相对路径

```
不存在图像[img/ne.png]
```

点击图像 []

点击图像 [图片路径]

- 支持平台：Android、iOS
- 语义：检查屏幕中是否存在指定的图像，图片路径基于项目目录下的相对路径

```
点击图像[img/button.png]
```

向 [] 查找 [] 的图像

向 [上/下/左/右] 扫描 [指定内容] 的文案

- 支持平台：Android、iOS
- 语义：在全屏向指定方向滑动查找指定的图像，图片路径基于正则表达式检查

```
向[下]查找[img/fly.png]的图像
```

图像识别

1. 在项目根目录下，创建图像存放目录，如:”img”
2. 将待检测的图像放入该目录

3. 编写用例

OpenCV 例子

```
# language: zh-CN
功能:首页图像信息校验

@p1
场景: 图像检查
  当 启动APP[ctrip.english.debug]
  那么 页面扫描完成出现文字[Discover]
  那么 存在图像[img/exist.png]
  那么 不存在图像[img/ne.png]
  那么 点击图像[img/click.png]
  那么 向下扫描[[img/city.png]的图像
```

1.15 Web 端例子

使用 cli 脚手架创建项目后，可在目录 `{your project}/features/test/web/`中找到以下例子

1.15.1 点击

```
# language: zh-CN
功能: web点击

    场景: 点击元素
    假如 跳转页面到[百度]
    而且 点击[#s-top-loginbtn]
    而且 等待[3]秒
    那么 全屏截图


    场景: 点击文案
    假如 跳转页面到[百度]
    而且 点击文案[新闻]
    而且 等待[3]秒
    那么 全屏截图


    场景: 点击屏幕位置
    假如 跳转页面到[百度]
```

(续下页)

(接上页)

```
    而且 点击屏幕位置[720,400]
    而且 等待[3]秒
    那么 全屏截图
```

1.15.2 查找元素

```
# language: zh-CN
功能: web查找元素

    场景: 在全屏查找
    假如 跳转页面到[百度]
    而且 页面渲染完成出现元素[text=新闻]
    那么 向下查找[text=关于百度]的元素


    场景: 在父元素中查找子元素
    假如 跳转页面到[百度]
    那么 存在[#hotsearch-content-wrapper]的[li.hotsearch-item.odd[data-index="2
↪"]]的元素
    那么 [.s-bottom-layer-content]的[text=帮助中心]文案为[帮助中心]
```

1.15.3 输入

```
# language: zh-CN
功能: 输入操作

    场景: 输入
    假如 跳转页面到[百度]
    而且 在[#kw]中输入[flybirds]
    而且 等待[3]秒
    那么 全屏截图


    场景: 清空并输入
    假如 跳转页面到[百度]
    而且 在[#kw]中输入[flybirds]
    而且 等待[3]秒
    那么 在[#kw]中清空并输入[input test]
    那么 全屏截图
```

1.15.4 页面操作

```
# language: zh-CN
功能：页面操作

    场景：返回上一页
    假如 跳转页面到[列表页]
    而且 页面渲染完成出现元素[text=霸王别姬 - Farewell My Concubine]
    而且 点击[text=霸王别姬 - Farewell My Concubine]
    而且 等待[3]秒
    而且 返回上一页
    而且 等待[2]秒
    那么 结束录屏


    场景：判断当前页面
    假如 跳转页面到[列表页]
    而且 页面渲染完成出现元素[text=霸王别姬 - Farewell My Concubine]
    而且 点击[text=霸王别姬 - Farewell My Concubine]
    而且 等待[3]秒
    那么 当前页面是[列表详情页]
```

1.15.5 录屏

```
# language: zh-CN
功能： web录屏

    场景:录屏
    假如 跳转页面到[百度]
    而且 页面渲染完成出现元素[text=新闻]
    而且 点击[#s-top-loginbtn]
    而且 等待[10]秒
    那么 结束录屏
```

1.15.6 下拉框选择

```
# language: zh-CN
功能：下拉框

    场景：下拉框选择
    假如 跳转页面到[携程]
```

(续下页)

(接上页)

```
    而且 在[#J_roomCountList]中选择[6间]
    而且 等待[5]秒
    那么 结束录屏
```

1.15.7 滑动

```
# language: zh-CN
功能: web滑动

    场景: 滑动
    假如 跳转页面到[列表页]
    而且 等待[2]秒
    而且 [text=霸王别姬 - Farewell My Concubine]向下滑动[600]
    而且 等待[5]秒
    那么 存在元素[text=肖申克的救赎 - The Shawshank Redemption]
    那么 存在[肖申克的救赎 - The Shawshank Redemption]的文案


    场景:全屏滑动
    假如 跳转页面到[列表页]
    而且 等待[2]秒
    而且 全屏向下滑动[600]
    而且 等待[5]秒
    那么 不存在[text=测试]的元素
    那么 不存在[测试]的文案
```

1.15.8 验证

```
# language: zh-CN
功能: web元素属性验证

    场景: 验证元素文案
    假如 跳转页面到[百度]
    那么 [text=新闻]的文案为[新闻]
    那么 [text=新闻]的文案包含[新]


    场景: 验证元素属性
    假如 跳转页面到[百度]
    那么 元素[#kw]的属性[name]为wd
    那么 元素[#su]的属性[value]为百度一下
```

1.15.9 缓存服务请求

```
# language: zh-CN
功能: 缓存服务请求

    场景:验证缓存服务请求
    假如 缓存服务请求[getRecommendHotelList]
    假如 缓存服务请求[writecookie]
    而且 跳转页面到[携程官网]
    那么 等待[5]秒
    而且 移除请求缓存[getRecommendHotelList]
    而且 移除所有请求缓存


    场景:验证缓存服务请求--同时传入多个参数
    假如 缓存服务请求[getRecommendHotelList,writecookie]
    而且 跳转页面到[携程官网]
    那么 等待[5]秒
    而且 移除请求缓存[getRecommendHotelList,writecookie]
```

1.15.10 验证服务请求

```
# language: zh-CN
功能: 验证比较服务请求

    场景: json类型服务请求比对
    假如 缓存服务请求[getRecommendHotelList]
    而且 跳转页面到[携程官网]
    那么 等待[5]秒
    而且 验证服务[getRecommendHotelList]的请求参数[head.syscode]与[PC]一致
    而且 验证服务[getRecommendHotelList]的请求参数[$.cityId]与[2]一致
    而且 验证服务[getRecommendHotelList]的请求参数[cityId]与[2]一致
    而且 验证服务请求[getRecommendHotelList]与[compareData/getRecommendHotelList.
↪json]一致


    场景: 非json类型服务请求比对
    假如 缓存服务请求[writecookie]
    而且 跳转到[携程官网]
    那么 等待[5]秒
    而且 验证服务非json请求[writecookie]与[compareData/writecookie.txt]一致
```

1.15.11 服务监听 Mock

```
# language: zh-CN
功能：监听并mock服务请求

    场景：服务监听与mock
    假如 监听服务[movie]绑定MockCase[4245512]
    当    跳转页面到[列表页]
    那么 等待[10]秒
    而且 移除服务监听[movie]
    而且 移除所有服务监听
```

1.16 多端应用例子

1.16.1 测试用例

```
功能：乘机人模块

@p1 @android @web
场景:外露乘机人_选择列表页乘机人
   当    跳转页面到[单程填写页]
   那么 页面渲染完成出现元素[已选乘机人姓名]
   那么 [选择乘机人文案]的文案为[选择乘机人]
   那么 [已选乘机人姓名]的文案为[李易峰]
   那么 [已选乘机人证件类型]的文案为[护照]
   那么 [已选乘机人证件号]的文案为[YHE77]
   那么 存在[乘客类型标签儿童]的元素
   那么 返回上一页
```

1.16.2 页面对象管理

多端项目中的页面对象管理，是通过 json 文件进行统一管理，通常存在以下两种情况

1. 各端相同时，参考以下配置

```
// 元素定位配置 ele_locator.json
{
  "选择乘机人文案": “testid=passger_check”,
  "已选乘机人姓名": “testid=passger_name_checked”,
  "已选乘机人证件类型": “testid=passger_ct_checked”,
```

(续下页)

(接上页)

```
  "已选乘机人证件号": “testid=passger_cn_checked”
}
```

2. 各端不同时，通过 android、ios、web 区分

```
// scheme配置 schema_url.json
{
  "单程填写页": {
    "android": "urlschemel://auth_activity",
    "ios": "urlschemel://ios_auth_activity",
    "web": "https://address"
  }
}

// 元素定位配置 ele_locator.json
{
  "乘客类型标签儿童": {
    "android": "textid=passger_type_child",
    "ios": "lableid=passger_type_child",
    "web": "xpath=//html/body/div"
  }
}
```

1.17 数据驱动参数化

实际项目中，大部分的自动化测试都是基于数据驱动参数化，因此还需要搭配「场景大纲 + 例子」一起使用，这里我们对上面的例子进行改造：

```
功能: 乘机人模块

@p1 @android @web
场景大纲:  外露乘机人_选择列表页乘机人
     当   跳转页面到[单程填写页]
     那么 页面渲染完成出现元素[已选乘机人姓名]
     那么 <element>的文案为<title>
     那么 存在[乘客类型标签儿童]的元素
     那么 返回上一页

     例子:
       |  element          |  title      |
       |  选择乘机人文案     |  选择乘机人   |
```

(续下页)

(接上页)

```
|    已选乘机人姓名    |    李易峰    |
|    已选乘机人证件类型    |    护照    |
|    已选乘机人证件号    |    YHE77    |
```

1.18 多浏览器并发

依托 PlayWright 的跨浏览器能力，Flybirds 支持所有的现代渲染引擎，包括 Chromium、WebKit 和 Firefox。

Flybirds 支持多浏览器并发模式，方便高效的进行浏览器兼容性测试

1.18.1 配置参数

```
// browserType: 配置浏览器内核
  "web_info": {
    "headless": true,
    "browserType": ["firefox","chromium","webkit"],
    "defaultTimeout": 30
  },
```

1.18.2 执行命令

```
# 通过参数指定web执行平台启动的浏览器(多个浏览器时用半角逗号进行分隔)
flybirds run -D browserType=webkit,firefox
```

1.19 自定义框架扩展

Flybirds 的插件式设计模式，保留了良好的扩展，未来我们会开放更多。

修改扩展

如果你希望在项目中修改当前扩展，你可以用本地文件替换 plugin 下面的(app,device,element,app,step,screen,screen_record)，并在“plugin_info.json”中做相应配置。

比如你希望修改 web 中 screen.py 文件:

```
1. 在本地创建一个py文件命名为 screen.py
2. 在plugin_info.json 的web中添加如下配置:
```

```
  "screen":  {
  "path": "{local_path}/screen.py",
  "ns": "screen.plugin.myextend"
  }
```

{local_path} 为本地路径,” ns” 为包名, 注意包名的唯一性 (以上包名只是例子不做强制限制)

内部增强包

对于团队内部通用的自定义功能，可以考虑创建一个 extend package，Flybirds 支持动态加载，package 命名包含 “-flybirds-plugin” 即可。携程机票内部，针对 DevOps 的各类工具，增强包中都进行了对接，安装后就可以使用。注意：扩展包是一个独立的 pip 包，包命名包含 “-flybirds-plugin” 即可，安装后 flybirds 就可以加载扩展包内容，具体 pip 包的制作方法可自行百度。

下面是一个扩展包基础目录结构的例子:

```
␣
↪
├─ trip-flybirds-plugin
|   ├─ core_apis                trip内部核心API处理
|   ├─ dsl
|   |    └─ step                Step 列表
|   ├─ utils                    工具方法
```

下面是一个在扩展包中自定义 dsl 的例子:

在 step 目录中增加一个自定义方法 mock.py，用于获取内部 mock 数据，step 中定义的方法可以在 feature 中直接调用

```
from behave import step

import trip_flybirds_plugin.core_apis.mock as mock_api
from trip_flybirds_plugin.core_apis.get_client import get_client_id, get_client_id_
↪domain
import flybirds.utils.flybirds_log as logger

@step('绑定MockSuite[{param1}]并开启Case[{param2}]')
def bind_flight_mock_suite_with_cases(context, param1, param2):
    client_id = get_client_id(context)
    suite_id = int(param1.strip())
    case_id_str_array = param2.strip().split(',')
```

(续下页)

(接上页)

```
    case_id_array = [int(case_id) for case_id in case_id_str_array]
    mock_api.bind_suite(client_id, suite_id, case_id_array)
```

1.20 其他语种支持

flybirds 可以支持 40 几种语言，在以下文件中增加公共方法的语言配置即可。

```
flybirds/core/dsl/globalization/i18n.py
```

示例代码如下

```
step_language = {
    "zh-CN": {
        "install app[{param}]": ["安装APP[{param}]"],
        "delete app[{param}]": ["删除APP[{param}]"],
        "start app[{param}]": ["启动APP[{param}]"],
        "restart app": ["重启App"],
        "close app": ["关闭App"],
        "init device[{param}]": ["设备初始化[{param}]"],
        "connect device[{param}]": ["连接设备[{param}]"],
        "start recording timeout[{param}]": ["开始录屏超时[{param}]"],
        "start record": ["开始录屏"],
        "stop record": ["结束录屏"],
        "go to url[{param}]": ["跳转到[{param}]", "跳转页面到[{param}]"],
        "return to previous page": ["返回上一页"],
        "go to home page": ["回到首页"],
        "logout": ["退出登陆", "退出登录"],
        "wait[{param}]seconds": ["等待[{param}]秒"],
        "screenshot": ["全屏截图"],
        "click[{param}]": ["点击[{param}]"],
        "click text[{param}]": ["点击文案[{param}]"],
        "click position[{x},{y}]": ["点击屏幕位置[{x},{y}]"],
        "in[{param1}]input[{param2}]": ["在[{param1}]中输入[{param2}]"],
        "slide to {param1} distance[{param2}]": ["全屏向{param1}滑动[{param2}]"],
        "exist text[{param}]": ["存在[{param}]的文案"],
        "not exist text[{param}]": ["不存在[{param}]的文案"],
        "text[{param}]disappear": ["文案[{param}]消失"],
        "exist[{param}]element": ["存在[{param}]的元素"],
        "not exist element[{param}]": ["不存在[{param}]的元素"],
        "element[{param}]disappear": ["元素[{param}]消失"],
        "page rendering complete appears element[{param}]": [
            "页面渲染完成出现元素[{param}]"],
        "existing element[{param}]": ["存在元素[{param}]"],
    },
}
```

1.21 持续集成

cli 提供的命令行执行模式，可以非常方便加入各种持续集成工具.

以 Jenkins 为例：

```
# Inside the jenkins shell command
cd {PATH_TO_PROJECT_FOLDER}
# Run
flybirds run -P ./features/test/everything.feature
cp -R reports $WORKSPACE
```

1.22 常见问题

安装中提示当前 python 版本不支持

- 当前支持的 python 版本为 >=3.7,<=3.9

安装的 flybirds 版本不对

- 使用了 pip 数据源中的缓存版本导致，可添加参数，不使用缓存
- `pip3 install flybirds --no-cache-dir`

pip 下载包过慢或总是失败

- pip 默认会使用海外的镜像源，可尝试切换到国内的源
- `pip3 install -U flybirds -i http://mirrors.aliyun.com/pypi/simple/ --trusted mirrors.aliyun.com`

flybirds command not found

出现这个的原因一般有两个：

- 安装未成功，执行 `pip3 show flybirds` 检查
- 安装成功了，但是没有配置 $PATH 环境变量如果是第二个原因，此时 `echo $PATH` 查看 flybirds 的安装目录是否在 PATH 中，如果没有，
- 在🄵/ .bash_profile 中中添加 export PATH=$PATH:/usr/local/bin（假设安装目录为在/usr/local/bin）中然后 source ~/.bash_profile 使之生效。

1.23 发版计划

我们将按照 SemVer 版本控制规范进行发版。逐步新增功能和代码优化，非常欢迎您加入到我们的共建计划中，在 GitHub 上提出您的宝贵建议，以及在使用时遇到的一切问题，我们也会对此每周进行一次小版本的迭代。您也可以在这里给我们精神支持，点上一颗 Star。

1.24 参考链接

- GitHub 地址：https://github.com/ctripcorp/flybirds
- PyPI 地址：https://pypi.org/project/flybirds
- Pages：https://ctripcorp.github.io/flybirds/
- PlayWright: https://github.com/microsoft/playwright-python
- Airtest: https://github.com/AirtestProject/Airtest
- Behave: https://github.com/behave/behave
- 欢迎在 GitHub issues 和Discussions区提问
- 支持邮箱：flybirds_support@trip.com

1.25 技术交流

CHAPTER 2

Indices and tables

- genindex
- modindex
- search