Allure 报告

学习目标

1. 能够将项目生成 allure 报告
2. 能够在 allure 报告上添加测试步骤
3. 能够在 allure 报告上添加测试描述
4. 能够在 allure 报告上添加严重级别

一. Allure 的简介和使用

应用场景

我们自动化的结果一定是通过一个报告来进行体现。Allure 是一个独立的报告插件，生成美观易读的报告，目前支持 Java、PHP、Ruby、Python、Scala、C# 这些语言。生成的报告无论是帮我们定位问题，还是发送给领导看，都能快速上手。

帮助文档

> https://docs.qameta.io/allure/

步骤概述

最终我们会生成一个 html 格式的报告，中间我们需要操作两步来进行。

1. 生成xml
2. 将 xml 转成 html

1.1 生成 xml

安装

```
pip3 install pytest-allure-adaptor
```

使用步骤

1. 将 pytest 配置文件中的命令行参数加上如下代码

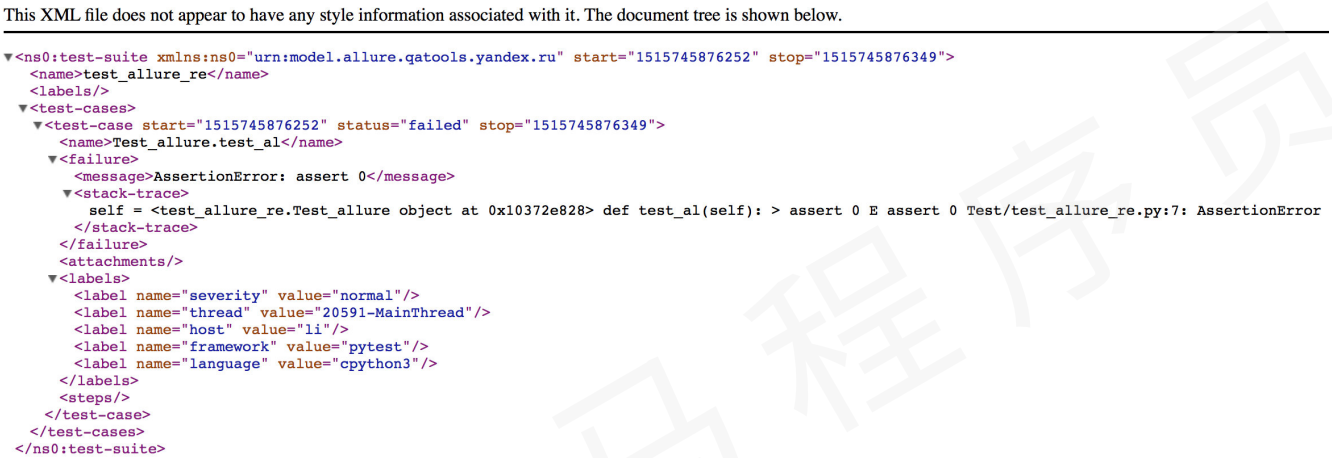
   ```
   --alluredir report
   ```

2. 编写好测试脚本后，正常的在命令行中运行 pytest 即可

> 示例
>
> pytest.ini

```
[pytest]
# 添加行参数
addopts = -s --alluredir report
# 文件搜索路径
testpaths = ./scripts
# 文件名称
python_files = test_*.py
# 类名称
python_classes = Test*
# 方法名称
python_functions = test_*
```

3. 程序运行结束后，我们会在项目中得到一个 xml 文件。



This XML file does not appear to have any style information associated with it. The document tree is shown below.

```
▼<ns0:test-suite xmlns:ns0="urn:model.allure.qatools.yandex.ru" start="1515745876252" stop="1515745876349">
   <name>test_allure_re</name>
   <labels/>
  ▼<test-cases>
   ▼<test-case start="1515745876252" status="failed" stop="1515745876349">
      <name>Test_allure.test_al</name>
     ▼<failure>
        <message>AssertionError: assert 0</message>
       ▼<stack-trace>
          self = <test_allure_re.Test_allure object at 0x10372e828> def test_al(self): > assert 0 E assert 0 Test/test_allure_re.py:7: AssertionError
        </stack-trace>
      </failure>
      <attachments/>
     ▼<labels>
        <label name="severity" value="normal"/>
        <label name="thread" value="20591-MainThread"/>
        <label name="host" value="li"/>
        <label name="framework" value="pytest"/>
        <label name="language" value="cpython3"/>
      </labels>
      <steps/>
    </test-case>
  </test-cases>
</ns0:test-suite>
```

1.2 将 xml 转成 html

安装

1. https://bintray.com/qameta/generic/allure2 下载 allure-2.6.0.zip
2. 解压缩到一个目录（不经常动的目录）
3. 将压缩包内的 bin 目录配置到 path 系统环境变量
4. 在命令行中敲 allure 命令，如果提示有这个命令，即为成功

使用步骤

在保证项目中的 report 目录下有 xml 文件的时候，执行以下步骤。

1. 进入 report 上级目录执行命令

```
allure generate report/ -o report/html --clean
```

2. report 目录下会生成 html 文件夹，html 下会有一个 index.html ，右键用浏览器打开即可。

1.3 参数和命令详解

应用场景

修改 xml 所在的目录名称和 index.html 所在的目录名称。

疑问和解答

1. `addopts = -s --alluredir report` 中的 `--alluredir report` 是什么意思?
    - --alluredir 后面的 report 为 xml 输出的目录名
    - 如果希望目录名叫 result 那么可以将命令行参数改为 `--alluredir result`
2. `allure generate report/ -o report/html --clean` 是什么意思?
    - report/ 表示 xml 所在的目录
    - -o 表示 output 输出
    - report/html 表示将 index.html 报告生成到哪个文件夹

二. Allure 与 pytest 结合

项目准备

使用 po 模式 + pytest 框架 制作程序，实现《设置》程序搜索 “xiaoming” 和 “hello1” 的文字

> 不作为此知识点的重点，能够理解代码和展示效果即可。

示例代码

scripts/test_search.py

```
from base.base_driver import init_driver
from page.page import Page
import pytest
import time
```

```
class TestSearch:

    def setup(self):
        self.driver = init_driver()
        self.page = Page(self.driver)

    @pytest.mark.parametrize("args", ["hello1", "xiaoming"])
    def test_search(self, args):
        self.page.setting.click_search()
        self.page.search.input_key_word(args)
        time.sleep(3)
        self.page.search.click_back()

    def teardown(self):
        time.sleep(3)
        self.driver.quit()
```

page/page.py

```
from page.search_page import SearchPage
from page.setting_page import SettingPage


class Page:

    def __init__(self, driver):
        self.driver = driver

    @property
    def setting(self):
        return SettingPage(self.driver)

    @property
    def search(self):
        return SearchPage(self.driver)
```

page/search_page.py

```
import allure
from selenium.webdriver.common.by import By

from base.base_action import BaseAction


class SearchPage(BaseAction):

    search_edit_text = By.ID, "android:id/search_src_text"

    back_button = By.CLASS_NAME, "android.widget.ImageButton"

    def input_key_word(self, text):
```

```
        self.input(self.search_edit_text, text)

    def click_back(self):
        self.click(self.back_button)
```

page/setting_page.py

```
import allure
from selenium.webdriver.common.by import By

from base.base_action import BaseAction


class SettingPage(BaseAction):

    search_button = By.ID, "com.android.settings:id/search"

    def click_search(self):
        self.click(self.search_button)
```

2.1 添加测试步骤

应用场景

一套登录流程需要至少三个步骤，输入用户名，输入密码，点击登录。我们可以通过添加测试步骤，让这些步骤在报告中进行体现

使用方式

在 page 中的所有方法上加上 `@allure.step(title="测试步骤001")` 装饰器即可

核心代码

page/setting_page.py

```
    @allure.step(title="点击搜索")
    def click_search(self):
        self.click(self.search_button)
```

page/search_page.py

```
    @allure.step(title="输入文字")
    def input_key_word(self, text):
        self.input(self.search_edit_text, text)

    @allure.step(title="点击返回")
    def click_back(self):
        self.click(self.back_button)
```

结果

scripts.test_search.TestSearch.test_search[hello1]

通过 test_search.TestSearch.test_search[hello1]

总览 历史 重试次数

优先级: normal

耗时: 3ms

执行

测试步骤

1ms
0s
0s

2.2 添加测试描述

2.2.1 文字描述

应用场景

我们在输入文字的时候想要在报告中展示输入的内容，可以使用添加测试描述的方式

> 不可以使用添加步骤，因为步骤使用的是装饰器，在传递过来的参数之上，无法使用

使用方式

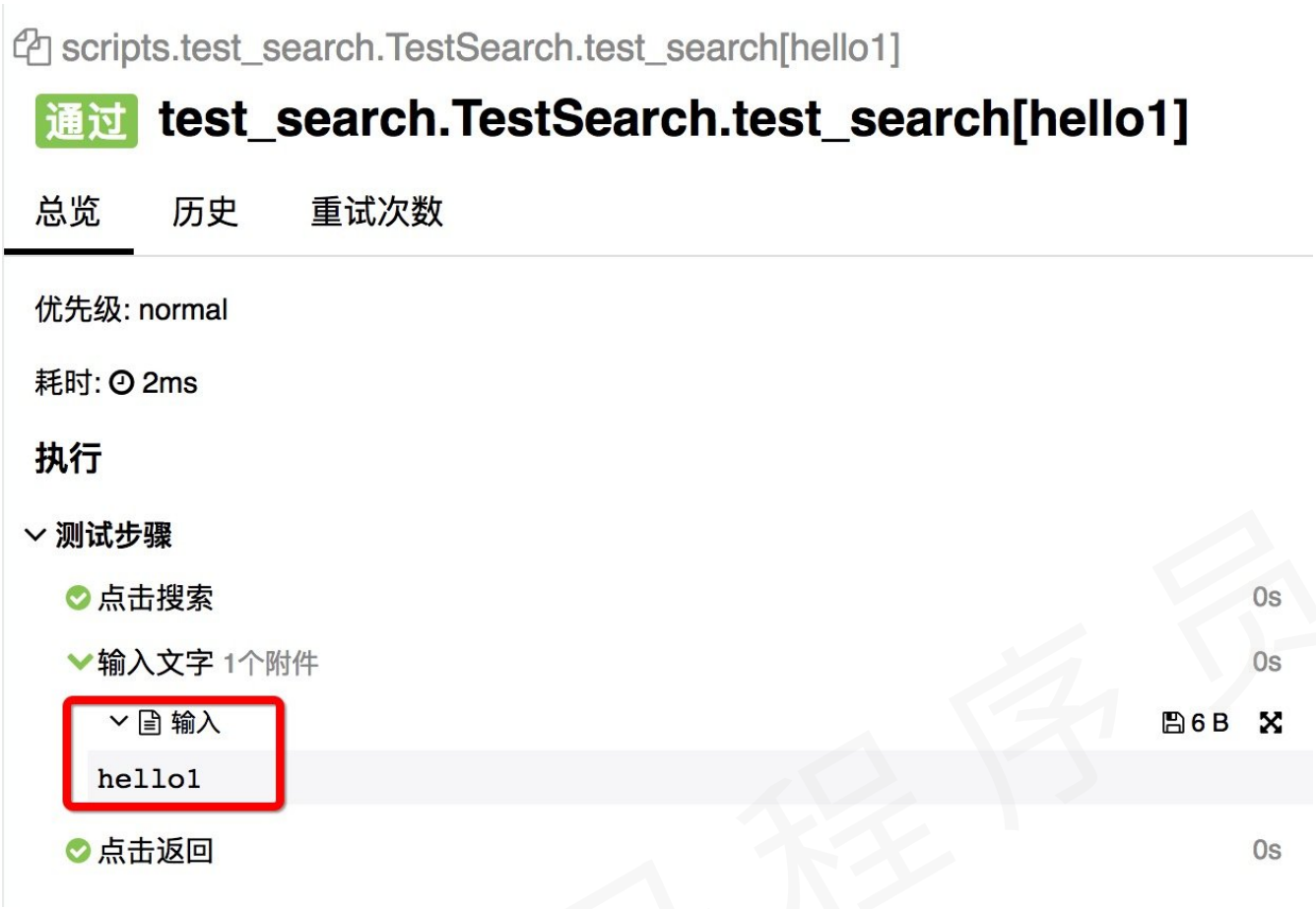
在需要增加描述的方法之前，调用 `allure.attach('描述', '我是测试步骤001的描述~~~', allure.attach_type.TEXT)`

核心代码

page/search_page.py

```
    @allure.step(title="输入文字")
    def input_key_word(self, text):
        allure.attach("输入", text, allure.attach_type.TEXT)
        self.input(self.search_edit_text, text)
```

结果



2.2.2 图片描述

应用场景

和文字描述的显示位置相同，如果我们想将某些操作过后的结果展现在报告上，可以使用添加图片描述的方法。

使用方式

和文字描述的方法是一样的，但是参数不同。在操作完成之后增加代码 `allure.attach("截图", self.driver.get_screenshot_as_png(), allure.attach_type.PNG)`

核心代码

page/search_page.py

```
    @allure.step(title="输入文字")
    def input_key_word(self, text):
        allure.attach("输入", text, allure.attach_type.TEXT)
        self.input(self.search_edit_text, text)
        allure.attach("截图", self.driver.get_screenshot_as_png(), allure.attach_type.PNG)
```

结果

2.3 添加严重级别

应用场景

在工作中，我们会向开发人员提交很多 bug ，不同的 bug 优先级也应当不同，打开程序就崩溃，和关于软件的页面错了一个字。就这两者而言，崩溃肯定更严重，也更需要开发人员优先修复。那么，我可以将这些 bug 的优先级展示在报告当中。

使用方式

在测试脚本中，增加装饰器 `@pytest.allure.severity(pytest.allure.severity_level.BLOCKER)`

参数有五个，也对应不同的优先级，只需要将最后一个词替换即可。

1. BLOCKER 最严重
2. CRITICAL 严重
3. NORMAL 普通
4. MINOR 不严重
5. TRIVIAL 最不严重

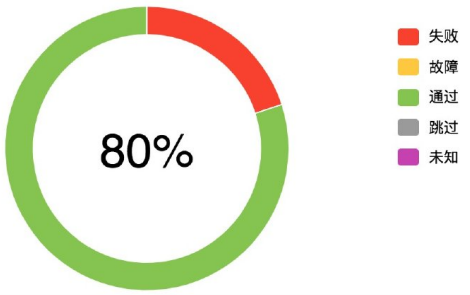
示例

scripts/test_search.py

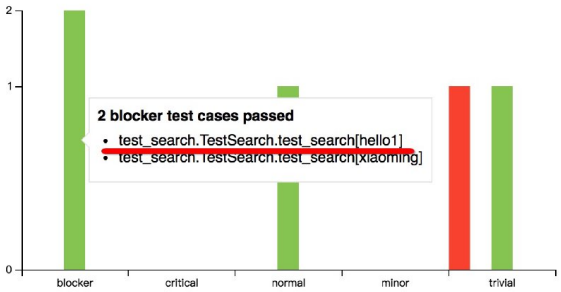
```
    @pytest.allure.severity(pytest.allure.severity_level.BLOCKER)
    @pytest.mark.parametrize("args", ["hello1", "xiaoming"])
    def test_search(self, args):
        self.page.setting.click_search()
        self.page.search.input_key_word(args)
        self.page.search.click_back()
```

结果

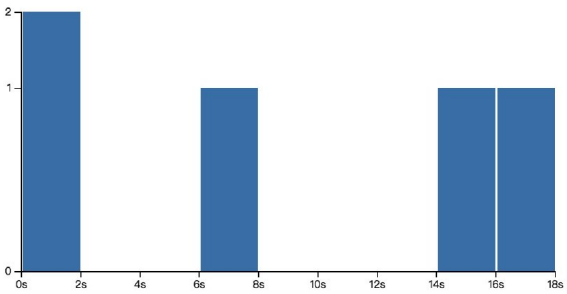
状态



优先级



耗时



DURATION TREND