# 浦东审批项目自动化测试方案

## 相关术语

|  |  |
| --- | --- |
| 测试套件（TestSuite） | 一个事项 |
| 测试用例（TestCase） | 某个事项下某个样例 |
| 真值 | 样例人工标定结果 |
| 测试值 | 浦东审批项目程序输入样例图片输出的审批结果，包含样例图片分类结果，审批事项字段内容，审批事项的结果 |
| 样例 | 输入至浦东审批项目程序一个样例扫描得到的图片 |
|  |  |
|  |  |

## 引言

在浦东审批项目中，帮助政府机关实现繁琐的审批简化。多个事项的开发，自动化测试方案的布署可以额大大提高浦东审批项目的质量看护。

## 方案框架

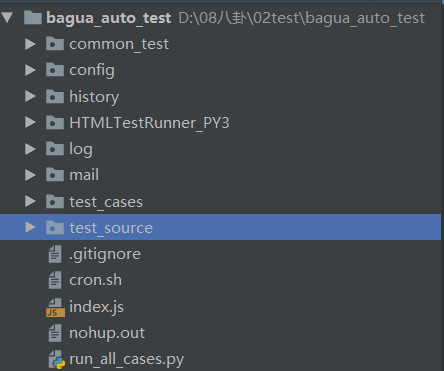
目前自动化测试框架主要由两个平台承载：

1. Gitlab实现test before merge 和daily test；
2. 10.40.12.46服务器cron脚本控制daily test；

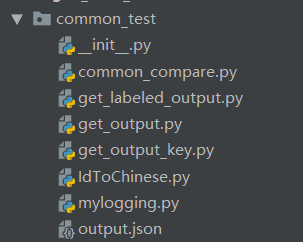
在本测试方案中，为了简化，每个事项组成一个测试套件，每个自动化测试用例完成事项下面一个sample的测试，每个测试用例完成一个sample程序输出的验证；

#### 2.1 测试框架文件结构

测试框架基于python中的unittest+ HTMLTestRunner完成，主要目录结果如下：



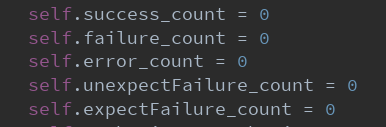
1. Common\_test: 提供测试公共接口



1. Config:自动化测试环境配置脚本
2. History:自动化测试报告汇总记录，以及部分自动化测试报告
3. HTMLTestRunner\_PY3:自动化测试报告模板类
4. Log：自动化测试框架log
5. Mail: 发送邮件脚本
6. Test\_case:测试用例脚本
7. Test\_source:测试值和真值存在的地方
8. Run\_all\_cases.py ：自动化测试用例调用主程序

#### 2.2 测试用例状态

unittest测试框架一共有5种测试用例状态：



目前的测试基于HTMLTestRunner + unittest框架，unittest提供了很多种测试结果，目前HTMLTestRunner只选取了这5种测试用例结果状态

|  |  |
| --- | --- |
| 用例状态 | 实际意义 |
| 通过 | 该用例没有bug |
| 失败 | 浦东审批程序有bug |
| 错误 | 测试程序运行有错误 |
| 不可忽略（非预期成功） | 浦东审批程序有未知bug |
| 可忽略（预期失败） | 浦东审批程序有已知bug |

#### 2.3 测试接口

测试用例脚本一共提供了10个测试接口

1. 文档比对不判断known\_issue

def docs\_sorted\_compare(label\_docs, output\_docs):  
 *"""* ***:param*** *label\_docs: 真值doc* ***:param*** *output\_docs: 测试值doc* ***:return****: error 是否有错误  
 """*

1. 文档比对判断known\_issue

def docs\_sorted\_compare\_knownissue(label\_docs, output\_docs):  
 *"""* ***:param*** *label\_docs: 真值doc* ***:param*** *output\_docs: 测试值doc* ***:return****: 是否可忽略  
 """*

1. 规则比对判断不判断known\_issure

def rules\_compare(label\_rules, output\_rules):  
 *"""* ***:param*** *label\_rules: 真值rule* ***:param*** *output\_rules: 测试值rule* ***:return****: 是否有error  
 """*

1. 规则比对判断判断known\_issure

def rules\_compare\_knownissue(label\_rules, output\_rules):  
 *"""* ***:param*** *label\_rules: 真值rule* ***:param*** *output\_rules: 测试值rule* ***:return****: 是否预期失败  
 """*

1. 根据事项调用程序启动

def input\_url(input\_config, case\_path):  
 *"""* ***:param*** *inputConfig: dictionary  
 config = {"sid":"SH00PD\_963201101","description":"内资新设",audioRecordId:""}  
 file = eachFile(filepath)* ***:return****: audioRecordId  
 """*

1. 获取程序输出output.json

def output\_url(audioRecordId, result\_path):  
 *"""* ***:param*** *audioRecordId: 记录id* ***:param*** *result\_path: output.json存放目录* ***:return****: output 的json格式  
 """*

1. 获取真值json

class SampleSystemOutput:  
 *"""样例生成系统的json"""* def \_\_init\_\_(self, source\_path, case\_num):  
 *"""* ***:param*** *source\_path: 事项目录* ***:param*** *case\_num: 样本序号  
 """*

1. 获取git commit id

def get\_binary\_link(gitlog\_path):  
 *"""* ***:param*** *gitlog\_path: gitlog文件目录* ***:return****: 返回当前最新程序git commit id  
 """*

1. 更新history\_csv

def update\_history\_csv(his\_path, result ,binary\_link):  
 *"""* ***:param*** *his\_path: 存放history.csv路径* ***:param*** *result: 测试结果* ***:param*** *binary\_link: commit id* ***:return****: history pandas  
 """*

1. 发送邮件ReportMail.py

#### 2.4 测试用例

开始

配置当前事项测试值存放路径

输入样例图片目录和事项id，调用程序接口

根据记录id获取测试值output.json

获取真值json

开始进行doc分类和规则结果比对

返回是否有错误

结束

图1 测试用例执行流程图

开始

遍历所有测试用例

执行所有测试用例，生成dailytest测试结果html

获取当前程序git commit id

更新history.csv

输出history.html

发送邮件

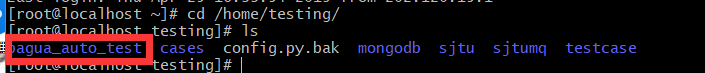
结束

#### 2.5 新增事项自动化测试用例

在新增事项时，首先在test\_source目录下新增该事项文件夹，将生成样例真值json放入，然后在test\_cases目录下新增该事项的测试脚本.py即可

#### 2.6 自动化测试环境布署

目前dailytest 测试环境布署在10.50.12.46服务器上，自动化测试脚本目录：/home/testing/bagua\_auto\_test



配置脚本如下：

Report\_path : /home/testing/testcase 此处为绝对路径，移动后请注意修改； 每天的测试报告html存在路径，可以直接连接，链接为：<http://10.50.12.46:5009/>

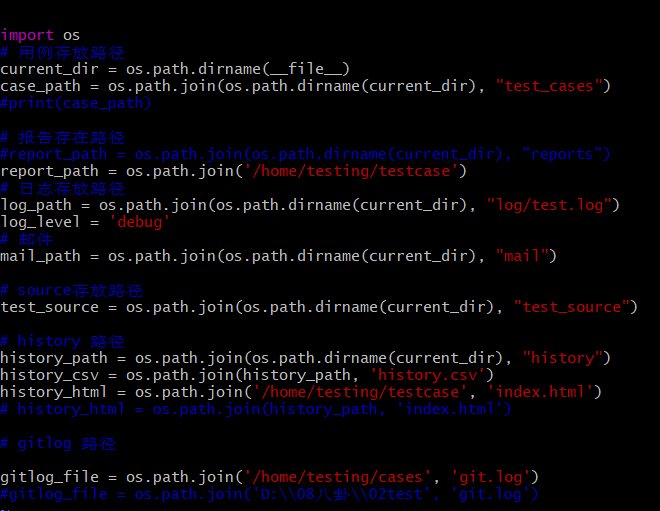
Log\_path:测试程序运行log存在路径

Mail\_path:发送邮件的脚本存放路径

Test\_source:测试值和真值存放路径，相对路径（无需修改）

History\_csv：每天测试概况后台存放数据路径

History\_html：测试历史数据概况html存放路径，可直接访问，链接为：<http://10.50.12.46:5009/>

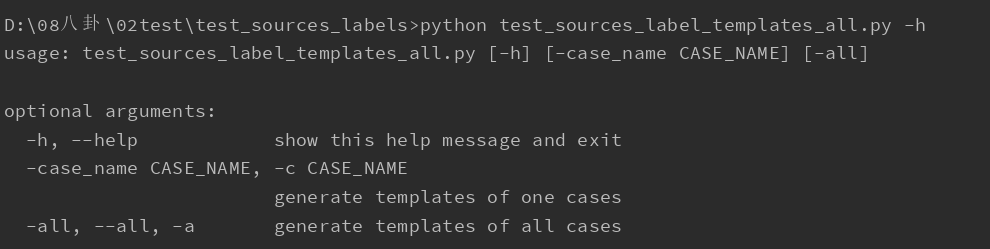


## 样例真值标定脚本

为了方便真值标定，这边提供了真值标定的脚本和流程：

1. 生成事项人工标定的模板

支持单个事项生成和批量生成，需要手动修改事项需求目录和模板输出目录，具体操作如下：



1）单个事项生成人工标定模板

a) 修改test\_sources\_label\_templates.py 里面事项目录和模板输出目录，都用'"/"地址格式

\_basedir = os.path.abspath("D:/08八卦/02test")   //business\_cases所在目录  
\_base\_output\_dir = os.path.abspath("D:/08八卦/02test")// 人工标定模板输出的目录   
​

b）使用命令python test\_sources\_label\_templates.py [-case\_name/-c] case\_name case\_name: 项目编号

c）如果输出目录路径下已经有生成模板的话，会提示已经存在模板，请先删除再生成新的模板

2）所有事项生成人工标定模板

a) 修改test\_sources\_label\_templates.py 里面事项目录和模板输出目录，都用'"/"地址格式

\_basedir = os.path.abspath("D:/08八卦/02test")   //business\_cases所在目录  
\_base\_output\_dir = os.path.abspath("D:/08八卦/02test")// 人工标定模板输出的目录   
​

b）使用命令python test\_sources\_label\_templates.py [-all/--all/-a] case\_name: 项目编号

c）如果输出目录路径下已经有生成模板的话，会提示已经存在模板，请先删除再生成新的模板

3）填写人工标定的模板后check模板填写是否正确

使用脚本：python check\_source\_labels.py 模板.xlsx

示例：

python check\_source\_labels.py D:/08八卦/02test/310101-007-02人工标定模板.xlsx

4）根据样本填写对应的模板后生成样本的expect\_json，并放到自动化测试目录

使用脚本：

python generate\_expected\_json.py  
​

a)修改脚本中的对应模板的路径

file = 'D:/08八卦/02test/310101-007-03人工标定模板.xlsx'

b)运行脚本，生成json文件在运行目录下

## 相关建议

1. 关于known\_issue:

目前known\_issue实现的方案是提供了一个加\_knownissue后缀的接口，通过unittest expectFailure装饰器将测试用例至为可忽略，但是如果至为可忽略有两种情况，一种是只有已知bug，一种是没有bug；

理论上没有bug的场景应该至为通过状态，所以建议后期重新搜索dailytest 结果的html，如果是通过状态，请在详情里面打印“通过”字样，所有当前测试用例的std缓存是否存在“通过”来确定该用例是否为通过。

以上方案仅供参考

1. 关于validation点：

目前的测试用例是以事项样例为单位进行，建议后期收敛验证点之后，将测试用例以验证点为单位管理；

## 附录

真值标定的脚本git 仓库：http://wudan@10.50.5.59:8888/bagua/test\_sources\_labels.git

自动化测试脚本git仓库：http://wudan@10.50.5.59:8888/bagua/bagua\_auto\_test.git