本规范试用于Python+unittest+requests的接口测试项目

# 一、项目书写规范

一级目录（要进行测试的项目名称）

README.md（对项目进行描述）

二级目录（具体案例、数据等）

cases（测试案例）

comm（通用方法、函数等）

datas（预埋数据）

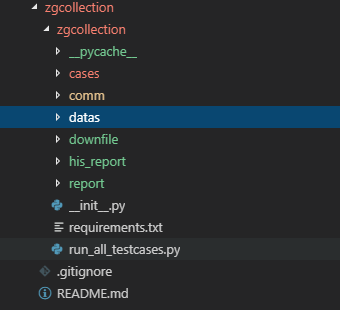
downfile（上传、现在文件存放）

his\_report（历史报告）

report（最新执行报告）

requirements.txt（项目依赖的第三方包）

run\_all\_testcases.py（项目执行文件）



# 二、案例文件命名规范

案例文件名称要标识清楚该案例的测试点，并考虑到不同案例执行顺序问题（书写规范请参照（附录））

test\_执行顺序\_模块名称\_执行顺序\_案例名称.py

例：

test\_a\_login.py

test\_b\_abs\_a\_project.py

test\_b\_abs\_b\_directory.py

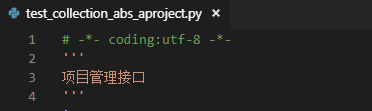
test\_c\_tb1\_a.py

test\_c\_tb2\_b.py

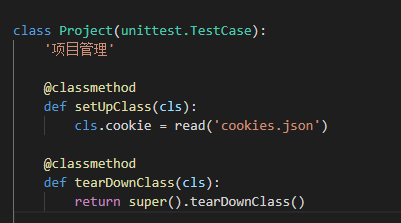
# 三、案例文件书写规范

## 3.1注释书写规范

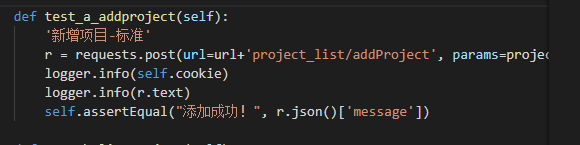
文件注释



类注释



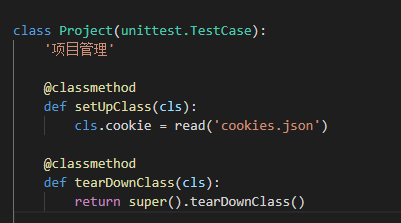
案例注释



## 3.2案例命名规范

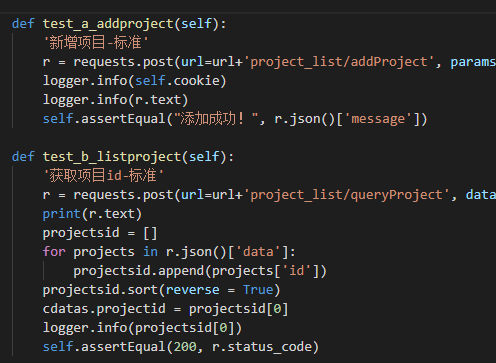
类的命名

使用测试的模块或者接口名称进行命名（书写规范请参照（附录））



案例命名

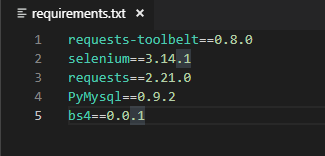
使用test\_执行顺序\_接口名称(self) 的形式命名



# 四、案例书写规范

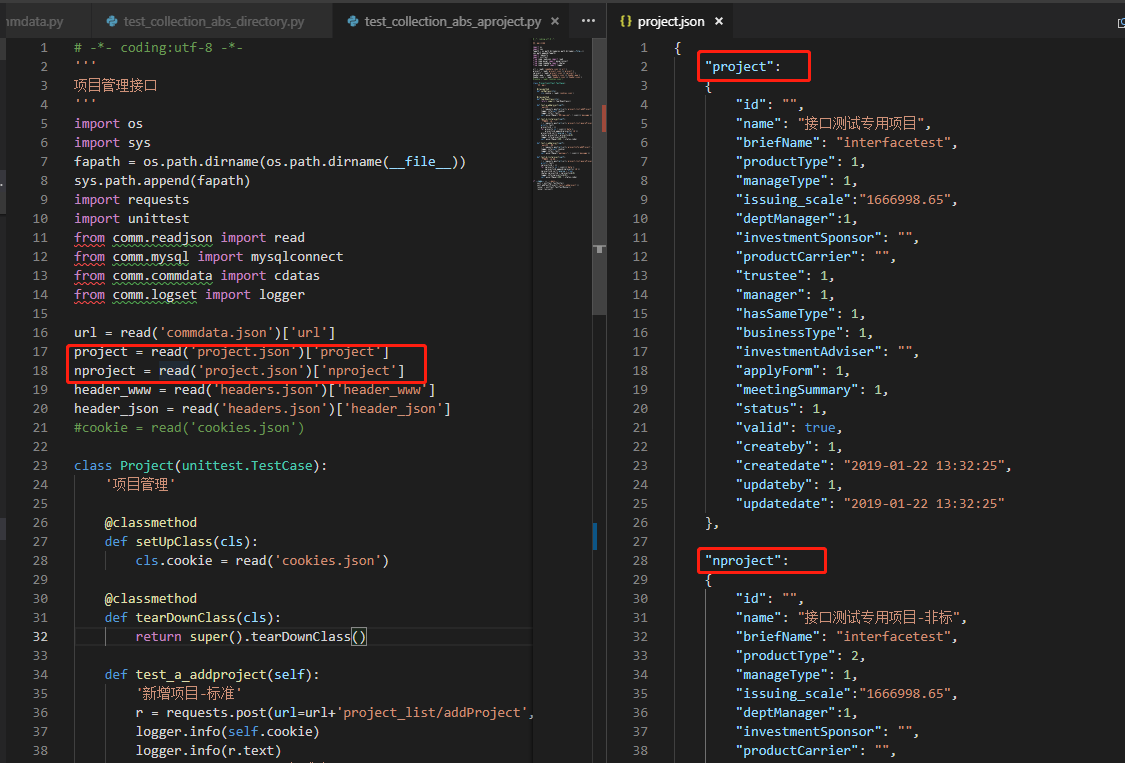
## 4.1第三方使用规范

如需使用第三方包，需按照现有格式将包的名称和版本写入到 [requirements.txt](http://192.168.6.32/interface/zgcollection/blob/master/zgcollection/requirements.txt)，并在项目执行服务器上安装此包。



## 4.2数据预埋规范

1. 一个案例对应一个json数据文件，json文件中可以定义多个节点；



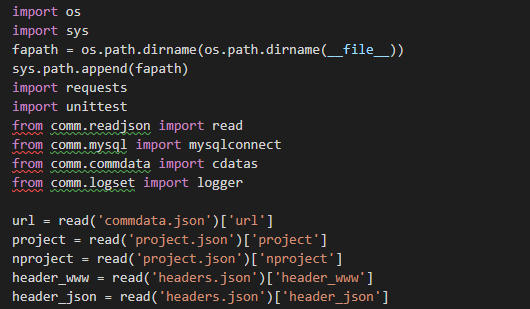
1. json文件名称最好与案例名称一致，方便案例和数据文件的对应；
2. 切勿随意定义多个数据文件，不利于后期的维护；

## 4.3公共方法使用规范

1、按照已有的方式进行引入和使用公共方法；

2、若修改已存在公共方法，尽量不影响现有功能；

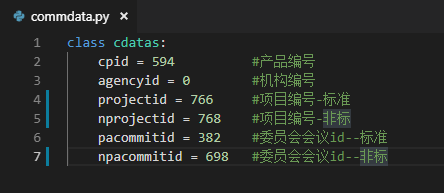
3、若定义新的公共方法，不要影响项目的执行；



## 4.4公共变量书写规范

公共变量定义在commdata.py文件中，保证变量名称清晰，注释准确

不要修改已存在的公共变量



## 4.5用例准备/清理规范

使用setUp()/ setUpClass() (用例准备)及tearDown()/ tearDownClass() (用例清理)

在用例准备中进行数据处理、数据库连接等工作

在用例清理中进行数据清理，数据库关闭等工作

class FileTree(unittest.TestCase):

'文档管理'

@classmethod

def setUp(self):

self.db = mysqlconnect('zgcollection')

@classmethod

def tearDownClass(self):

self.db.closedb()

## 4.6cookie使用规范

准备

@classmethod

def setUpClass(cls):

cls.cookie = read('cookies.json')

使用

def test\_a\_addproject(self):

'新增项目-标准'

r = requests.post(url=url+'project\_list/addProject', params=project, headers=header\_json, cookies=self.cookie)

logger.info(self.cookie)

logger.info(r.text)

self.assertEqual("添加成功！", r.json()['message'])

## 4.7断言书写规范

每个测试用例必须添加有效断言

self.assertEqual("新增成功", r.json()['message'])

不能使用无效断言或者无断言

self.assertEqual("1","1")

## 4.8单文件执行规范

单个文件中的案例执行，必须使用如下方法：

if \_\_name\_\_ == '\_\_main\_\_':

suit = unittest.TestSuite()

suit.addTest(Project('test\_c\_addproject'))

runner = unittest.TextTestRunner()

runner.run(suit)

或者

if \_\_name\_\_ == '\_\_main\_\_':

unittest.main()

## 4.9数据清理规范

单个案例或者整个项目执行完毕，要进行测试数据的清理，防止接口自动化数据污染测试环境。

1. 案例执行结束后使用删除接口或者数据库删除（不影响剩余案例执行）；
2. 项目执行完成后，使用数据库删除测试数据；

# 五、代码提交规范

1、提交到gitlab前，先将本地pull到最新（gitlab机制）；

2、项目his\_report和report目录禁止提交；

3、run\_all\_testcases.py中的all\_case()方法禁止随意提交；

# 附录（python编码规范）

| **Type** | **Public** | **Internal** |
| --- | --- | --- |
| Modules | lower\_with\_under | \_lower\_with\_under |
| Packages | lower\_with\_under |  |
| Classes | CapWords | \_CapWords |
| Exceptions | CapWords |  |
| Functions | lower\_with\_under() | \_lower\_with\_under() |
| Global/Class Constants | CAPS\_WITH\_UNDER | \_CAPS\_WITH\_UNDER |
| Global/Class Variables | lower\_with\_under | \_lower\_with\_under |
| Instance Variables | lower\_with\_under | \_lower\_with\_under (protected) or \_\_lower\_with\_under (private) |
| Method Names | lower\_with\_under() | \_lower\_with\_under() (protected) or \_\_lower\_with\_under() (private) |
| Function/Method Parameters | lower\_with\_under |  |
| Local Variables | lower\_with\_under |  |

其他规范参照：http://www.runoob.com/w3cnote/google-python-styleguide.html