**Unittest单元测试框架说明文档**

**单元测试框架功能**

提供用例编写规范与批量执行，提供丰富的断言方法，例如断言相等/不相等、包含/不包含，True/Flase等，提供丰富的测试日志（失败信息、成功用例数、执行时间）。

常见自动化测试框架+工具：

1. 、HTTP接口自动化测试：unittest + Requests
2. 、Web UI自动化测试：unittest（TestNG） + Selenium
3. 、移动自动化测试：unittest + Appium

**测试用例编写**

**class** **GamesInfo(**unittest**.**TestCase**):**

””” 获取游戏列表 ”””

# https://apis.zhanqi.tv/static/v2.2/tv/games.json

**def** setUp**(**self**):**

self**.**url **=** 'https://apis.zhanqi.tv/static/v2.2/tv/games.json'

mylog**.**info**(**"接口:获取游戏列表，测试开始"**)**

**def** tearDown**(**self**):**

mylog**.**info**(**"接口:获取游戏列表，测试完成"**)**

**def** test\_gamesInfo\_success**(**self**):**

””” 获取成功 ”””

r **=** requests**.**get**(**self**.**url**)**

self**.**response\_data **=** r**.**json**()**

self**.**assertEqual**(**self**.**response\_data**[**'code'**],** 0**,** msg='ERROR:获取失败'**)**

self.assertEqual**(**self.response\_data**[**'message'**]**, 'OK', msg='ERROR:返回值错误'**)**

print**(**self.response\_data**)**

**def** test\_roomInfo\_games**(**self**):**

””” 直播间列表 ”””

r **=** requests**.**get**(**self**.**url**)**

self**.**response\_data **=** r**.**json**()**

self**.**assertTrue**(**isinstance**(**self**.**response\_data**[**'data'**],** list**),** msg**=**"ERROR:返回数据类型错误"**)**

**print(**self**.**response\_data**[**'data'**])**

# 直播间列表 - game结构

**def** test\_roomInfo\_img**(**self**):**

””” 游戏图 ”””

r **=** requests**.**get**(**self**.**url**)**

self**.**response\_data **=** r**.**json**()**

self**.**assertTrue**(**self**.**response\_data**[**'data'**][**0**][**'img'**],** msg**=**"ERROR:游戏图为空"**)**

**print(**self**.**response\_data**[**'data'**][**0**])**

首先，通过import导入unittest测试框架。创建GamesInfo类继承unittest.TestCase类。setUp()和tearDown()两个方法分别在每一个测试用例的开始和结束时执行。setUp()用于测试用例执行前的初始化工作，例如初始化变量、生成数据库测试数据、打开浏览器等。tearDown()方法用于测试用例执行之后的善后工作，例如清除数据库测试数据、关闭文件、关闭浏览器等。

unittest要求测试用例（方法）必须以“test”开头。例如test\_gamesInfo\_success、test\_roomInfo\_games等。

**执行测试用例**

方案一：unittest.main()方法会搜索该模块下所有以test开头的测试用例方法，并自动执行它们。执行顺序是命名顺序：先执行test\_case1，再执行test\_case2

unittest**.**main**()**

**方案二：**①、先构造测试集

suite**=**unittest**.**TestSuite**()** #实例化测试套件

②、将测试用例加载到测试套件中。执行顺序是安装加载顺序：先执行test\_case2，再执行test\_case1

suite**.**addTest**(**Test**(**'test\_case2'**))**

suite**.**addTest**(**Test**(**'test\_case1'**))**

③、执行测试用例

runner**=**unittest**.**TextTestRunner**()** #实例化TextTestRunner类

runner**.**run**(**suite**)** #使用run()方法运行测试套件（即运行测试套件中的所有用例）

**方案三：**①、构造测试集（简化了方案二中先要创建测试套件然后再依次加载测试用例），执行顺序同方案一：执行顺序是命名顺序：先执行test\_case1，再执行test\_case2

test\_dir **=** './'

discover**=** unittest**.**defaultTestLoader**.**discover**(**test\_dir**,** pattern**=**'test\_\*.py'**)**

②、执行测试用例

runner**=**unittest**.**TextTestRunner**()** #实例化TextTestRunner类

runner**.**run**(**discover**)** #使用run()方法运行测试套件（即运行测试套件中的所有用例）

**常用断言方法(assert\*)**

assertEqual**(**a**,** b**,** msg**=**'测试失败时打印的信息'**)** #断言a和b是否相等，相等则测试用例通过

assertNotEqual**(**a**,** b**,** msg**=**'测试失败时打印的信息'**)** #断言a和b是否相等，不相等则测试用例通过

assertTrue**(**x**,** msg**=**'测试失败时打印的信息'**)** #断言x是否True，是True则测试用例通过

assertFalse**(**x**,** msg**=**'测试失败时打印的信息'**)** #断言x是否False，是False则测试用例通过

assertIs**(**a**,** b**,** msg**=**'测试失败时打印的信息'**)** #断言a是否是b，是则测试用例通过

assertNotIs**(**a**,** b**,** msg**=**'测试失败时打印的信息'**)** #断言a是否是b，不是则测试用例通过

assertIsNone**(**x**,** msg**=**'测试失败时打印的信息'**)** #断言x是否None，是None则测试用例通过

assertIsNotNone**(**x**,** msg**=**'测试失败时打印的信息'**)** #断言x是否None，不是None则测试用例通过

assertIn**(**a**,** b**,** msg**=**'测试失败时打印的信息'**)** #断言a是否在b中，在b中则测试用例通过

assertNotIn**(**a**,** b**,** msg**=**'测试失败时打印的信息'**)** #断言a是否在b中，不在b中则测试用例通过

assertIsInstance**(**a**,** b**,** msg**=**'测试失败时打印的信息'**)** #断言a是是b的一个实例，是则测试用例通过

assertNotIsInstance**(**a**,** b**,** msg**=**'测试失败时打印的信息'**)** #断言a是是b的一个实例，不是则测试用例通过

**推荐文档**

[**Unittest单元测试框架总结**](https://www.cnblogs.com/yufeihlf/p/5707929.html#test0)

作者：曹飞

部门：PC端-测试部-项目管理中心

时间：2017/09/20