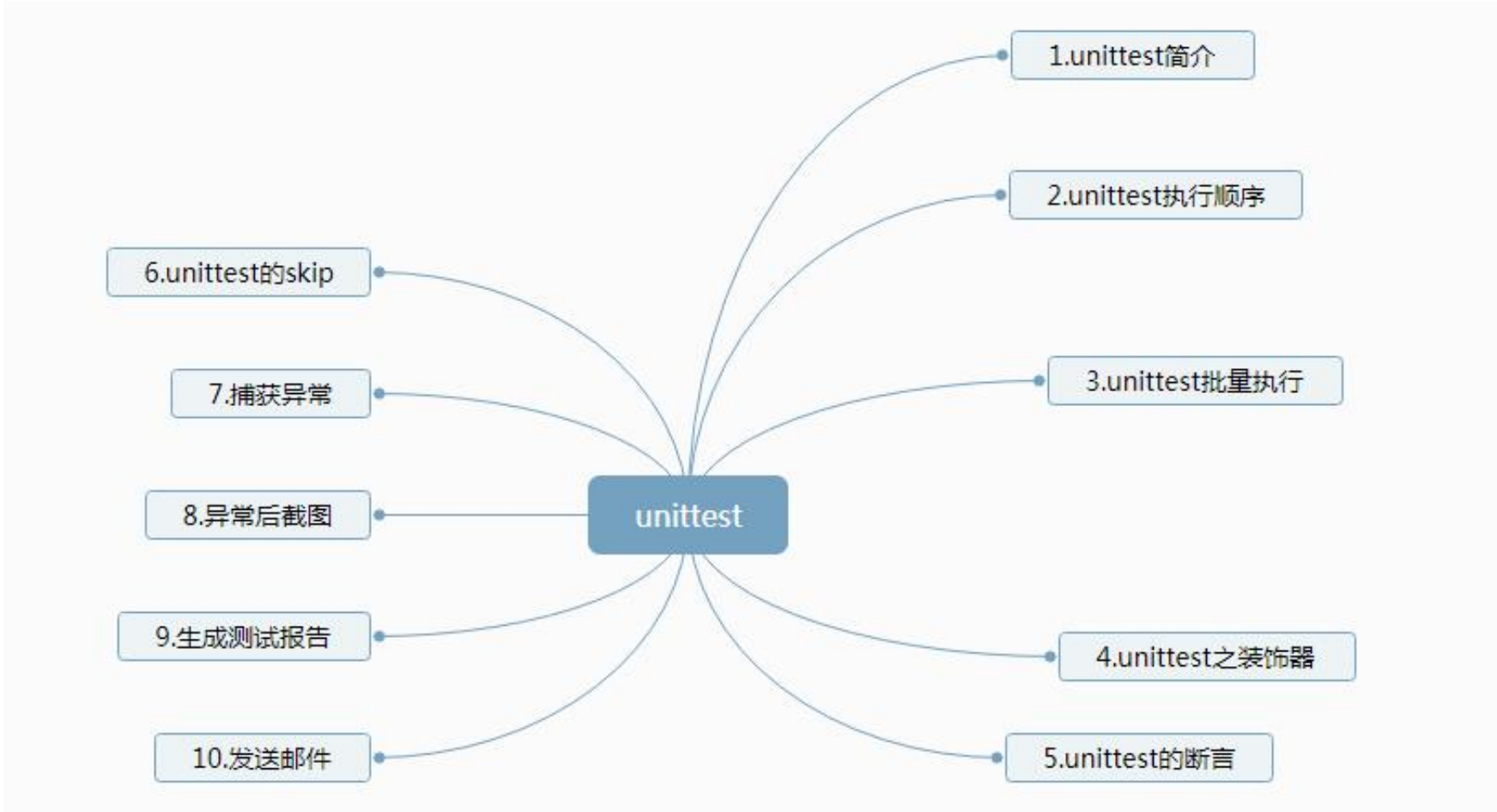
unittest单元测试框架

讲师：芳姐

金斧子 | 金斧子五周年 ANNIVERSARY

科 技 提 升 投 资 品 质

unittest



• 1.1 unittest的简介

- unittest是python的单元测试框架，有时候，也做叫做“PyUnit”,是Junit的Python语言版本。
- unittest支持测试自动化，共享测试用例中的初始化（setUp）和关闭退出（shutDown）代码块，在unittest中最小单元是test，也就是一个测试用例。集合所有的测试用例并且将测试结果独立地展示在报告框架中的特性，在一组测试中，通过unittest框架提供的类很容易支持它的这些特性。

• 1.2 unittest框架的4个重要概念

- **测试固件（test fixture）**

- 一个test fixture代表一个或多个测试执行前的准备动作（setUp）和测试结束后的清理动作（shutDown），例如：打开浏览器输入url，关闭浏览器

- **测试用例（test case）**

- 一个test case就是一个最小测试单元，也就是一个完整的测试流程，针对一组特殊的输入进行特殊的验证和响应。通过继承unittest提供的测试基类（TestCase），可以创建新的测试用例。一个测试用例中，测试固件可以不写，但是至少有一个已test开头的函数。unittest会自动化识别test开头的函数是测试代码，如果你写的函数不是test开头，unittest是不会执行这个函数里面的脚本的

➢ 测试套件（test suite）

一个test suite就是一组测试用例，一组测试套件或两者共同组成的集合，它的作用是将测试用例集合到一起，然后一次执行集合中所有的测试用例。

➢ 测试运行器（test runner）

一个test runner由执行设定的测试用例和将测试结果提供给用户两部分功能组成

- 示例1

1. 新建一个testfour.py的文件

2. 导入unittest模块

3. 当前测试类继承unittest.TestCase，相当于当前利用unittest创建了一个test case，这个test case是能够被unittest直接识别。

4. 写setUP(),主要是打开浏览器和打开站点

5. 写一个test_login（）用例用于登录的代码

6. 写tearDown(),主要是浏览器退出操作

相关脚本代码如下：

- import unittest
- import time
- from selenium import webdriver

- class testfour(unittest.TestCase):

- def setUp(self):
- '''测试固件setUp的代码，主要用于测试的前期准备工作'''
- self.driver = webdriver.Chrome()
- self.driver.maximize_window()
- self.driver.implicitly_wait(8)
- self.driver.get("https://qa1-erp.jfz.com")

```
    def tearDown(self):
        '''测试结束之后的清理，一般是关闭浏览器'''

        self.driver.quit()
    def test_login(self):
        '''这里一定要以test开头'''

        self.driver.find_element_by_xpath("//*[@name='username']").send_keys("defang2")
        self.driver.find_element_by_xpath("//*[@name='password']").send_keys("123")
        self.driver.find_element_by_xpath("//*[@class='submit_wrap']").click()
        self.assertEqual(self.driver.title,'深圳市金斧子网络科技有限公司-ERP',msg='访问erp失败')

        print("标题"+ self.driver.title)

        time.sleep(5)
        assert '德芳客服' in  self.driver.page_source
        time.sleep(5)

if __name__ == '__main__':
    unittest.main()
```

1.2.3 单元测试加载方法

在unittest单元测试框架中，提供了两种单元测试的加载方法

（1）直接通过unittest.main()方法加载单元测试的测试模块，这是一种最简单的加载方法，所有的测试方法执行顺序都是按照方法名的字符串表示的ASCII码升序排序。

（2）将所有的单元测试用例（Test Case)添加到测试套件（Test suite)集合中，然后一次性加载所有测试对象。

1.3 unittest执行顺序

示例1：

testdemo.py

结果分析

1.4 uittest 批量执行

示例： testfour文件夹

结果分析

1.5 uittest 的断言

用unittest组件测试用例的时候，断言的方法很多，下面介绍几种常用的断言方法：assertEqual、assert

1.5.1 示例

demo6.py

1.5 uittest 的断言

用unittest组件测试用例的时候，断言的方法很多，下面介绍几种常用的断言方法：assertEqual、assert

1.5.1 示例

demo6.py

1.5.2 unittest常用的断言方法

1）.assertEqual(self, first, second,msg=None)
--判断两个参数相等：first == second
2）.assertNotEqual(self, first, second,msg=None)
--判断两个参数不相等：first ！= second
3）.assertIn(self, member, container,msg=None)
--判断是字符串是否包含：member in container
4）.assertNotIn(self, member,container, msg=None)
--判断是字符串是否不包含：member not in container
5）.assertTrue(self, expr, msg=None)
--判断是否为真：expr is True
6）.assertFalse(self, expr, msg=None)
--判断是否为假：expr is False
7）.assertIsNone(self, obj, msg=None)
--判断是否为None：objis None
8）.assertIsNotNone(self, obj,msg=None)
--判断是否不为None：obj is not None

1.6 unittest的装饰器(@classmethod)

1.6.1 用setUp与setUpclass区别

- setUp()：每个测试case运行前执行
- teardown()：每个测试case运行完后执行
- setUpClass():必须使用@classmethod装饰器，所有case运行前只运行一次
- teardownClass()：必须使用@classmethod装饰器，所有case运行完后只运行一次

1.6.2 @是修饰符，classmethod是Python里的类方法

1.6.3 执行示例

1.7 unittest 的skip

当测试用例比较多的时候，有些模块有改动时候，会影响部分用例的执行，这个时候希望暂时跳过这些用例，或者前面某个功能运行失败了，后面的几个用例是依赖于这个功能的用例，如果第一步就失败了，后面的用例也就没有必要去执行了，直接跳过就行，节省用例执行时间。

1.7.1 装饰器

skip装饰器一共有四个：

- **@unittest.skip(reason)　#无条件跳过用例，reason是说明原因**
- **@unittest.skipIf（condition，reason）　# condition为true的时候跳过**
- **@unittest.skipUnless（condition，reason）　#condition为False的时候跳过**
- **@unittest.expectedFailure　　#断言的时候跳过**

1.7.2 执行案例 demo3.py

1.7.3 跳过整个测试类

```python
class demo(unittest.TestCase):

    @unittest.skip("无条件跳过此用例")
    def test_01(self):
        print("测试无条件跳过此用例")

    @unittest.skipIf(True,"为True的时候跳过")
    def test_02(self):
        print("测试为True的时候跳过")

    @unittest.skipUnless(False,"为False的时候跳过")
    def test_03(self):
        print("测试为False的时候跳过")

    @unittest.expectedFailure
    def test_04(self):
        print("进行断言的时候跳过")
        self.assertEqual(5,8,msg="判断相等")

if __name__ == '__main__':
    unittest.main()
```

1.8 捕获异常

1.8.1 发生异常

1.8.2 捕获异常

```
 try :


except

        try:

            self.assertEqual(self.driver.title,'test',msg='访问erp失败')
        except AssertionError as msg:
            print('msg:%s' % msg)
```

demo4.py

1.8.3 selenium常见异常

- **NoSuchElementException ： 没有找到元素**
- **NoSuchFrameException：没有找到iframe**
- **NoSuchWindowException：没有找到窗口句柄handle**
- **NoAlertPresentException：没有找到alert弹出框**
- **NoSuchAttributeException：属性错误**
- **ElmentNotVisibleException：元素不可见**
- **ElmentNotSelectableException:元素没有被选中**
- **TimeoutException：查找元素超时**
- **AssertionError ：断言**

1.9 异常后截图

1.9.1 截图方法

- get_screenshot_as_file(self,filename)

这个方法是获取当前Windows的截图，出现IOError时候返回False，截图成功返回True
filename参数是保存文件的路径

```
driver.get_screenshot_as_file('/exceptionpictures/2017-12-20/15-57-01-668427.png')
```

- get_screenshot_as_base64(self)

这个方法也是获取屏幕截图，保存的是base64的编码格式，在HTML界面输出截图的时候，会用到。

```
driver.get_screenshot_as_base64()
```

- **get**_screenshot_as_png(self)

这个是获取屏幕截图，保存的是二进制数据，很少用到

1.9.2 异常后截图

示例demo5.py

2.0 unittest生成测试报告HTMLTestRunner

2.0.1 导入htmlTestRunner

2.0.2 示例

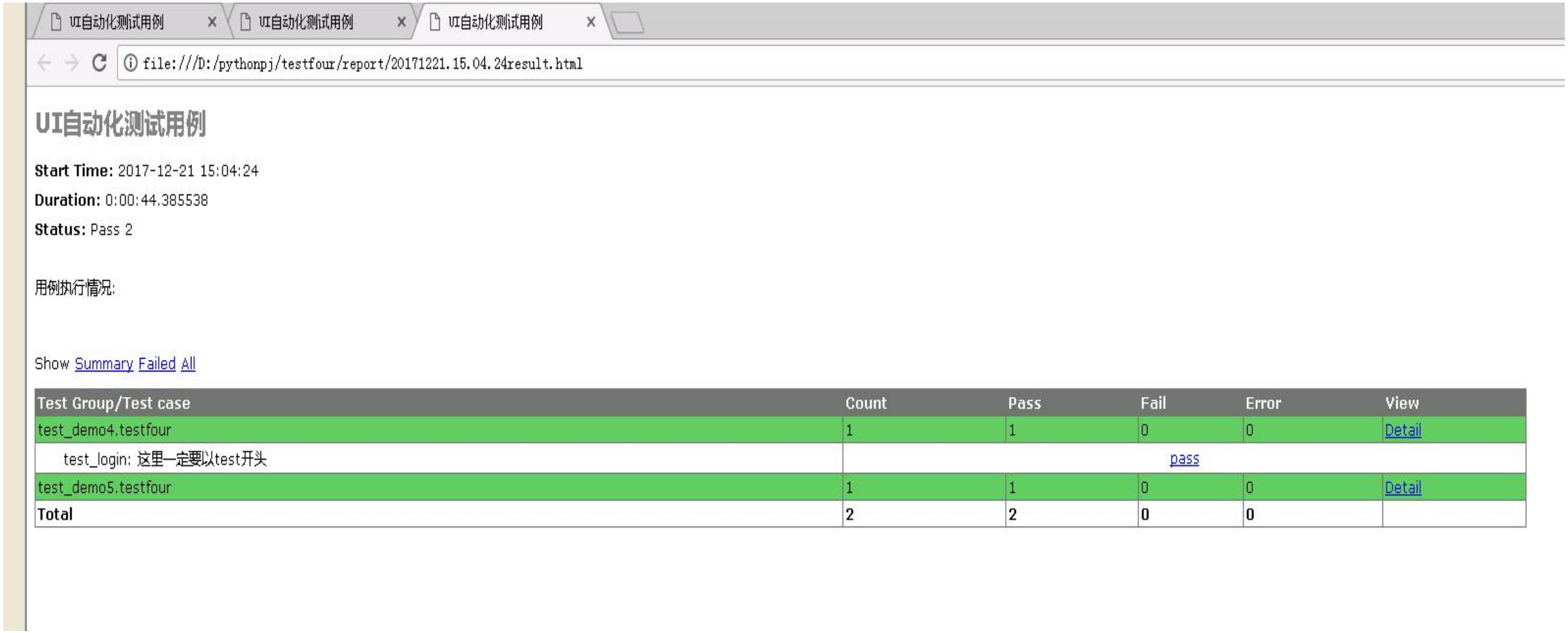
```
fp = open(filename, 'wb')
#定义测试报告
runner = HTMLTestRunner(stream=fp, title='UI自动化测试用例', description='用例执行情况:', verbosity=2)
runner.run(all_case())   #运行测试用例
fp.close()  #关闭报告文件
```

2.0.3 生成HTML报告

- stream=fp：测试报告写入文件的存储区域
- title='UI自动化测试用例',
- description='用例执行情况: 测试报告的描述

report_path：是存放测试报告的地址

2.0.4 测试报告详情



2.1 发送邮件

示例：run_all_case.py

作业

1、unittest的使用，比如（执行顺序，批量执行，装饰器，断言，skip，捕获异常，截图）
2、生成HTML报告
3、发送邮件

金斧子
聪明你的投资
JFZ.COM
金斧子五周年
ANNIVERSARY

Thanks!

科 技 提 升 投 资 品 质