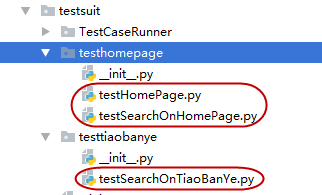
**Uninttest 如何管理和执行测试用例**

在前面一节, 我们介绍了unittest中如何在同一个class文件写很多test开头的测试用例。利用test fixture中的setUpClass() 和tearDownClass() 这两个类方法，来确定，同一个class下很多个test开头的用例只需要调用一次setUpClass() 和tearDownClass()。这样，我们就掌握了，在一个页面或者一个模块写单独一个类文件，一个类里面可以写很多测试用例。

接下来，我们来学习如何利用unittest中TestSuite()来管理我们实际项目中不同包结构下的自动化测试脚本文件。



我圈起来的两个包下的三个.py文件，每个文件里面至少有两个test开头的脚本用例。我们如何去调用不同包或者模块下的脚本文件。从前面文章我们知道，TestCase是最小单元，TestCase需要被添加到TestSuite中去，然后开始执行，执行完后，需要有报告，如何生成报告，我们稍后讲，本文讨论，如何把不同TestCase添加到TestSuite，从而被顺利执行。

1. **构建测试套件-- addTest()**

当有多个或者几百测试用例的时候，我们必须需要一个集合，来把测试用例放在该集合中，来进行执行，幸运的是，unittest模块中提供了TestSuite类来生成测试套件，使用该类的构造函数可以生成一个测试套件的实例，该类提供了addTest来把单元测试用例加入到测试套件中，测试代码使用TestSuite的addTest进行更改，具体看如下的测试脚本：

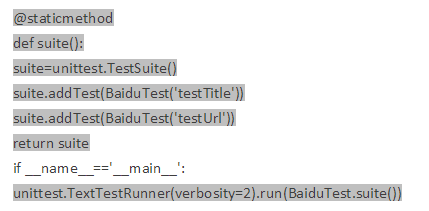
*# coding= utf-8***from** selenium **import** webdriver  
**import** unittest  
  
  
**class** BaiduTest(unittest.TestCase):  
 **def** setUp(self):  
 self.driver=webdriver.Firefox()  
 self.driver.maximize\_window()  
 self.driver.get(**'http://www.baidu.com'**)  
 self.driver.implicitly\_wait(10)  
  
 **def** tearDown(self):  
 self.driver.quit()  
  
 **def** testTitle(self):  
 self.assertEqual(**u'百度一下，你就知道'**, self.driver.title)  
  
 **def** testUrl(self):  
 self.assertEqual(**'https://www.baidu.com/'**, self.driver.current\_url)  
  
**if** \_\_name\_\_==**'\_\_main\_\_'**:  
 suite=unittest.TestSuite()  
 suite.addTest(BaiduTest(**'testTitle'**))  
 suite.addTest(BaiduTest(**'testUrl'**))  
 unittest.TextTestRunner(verbosity=2).run(suite)

**分析：**

上面一个class文件里有两个测试用例（test开头的函数就是测试用例）：

1. 先初始化一个suite，这个是TestSuite()的一个实例对象。
2. 通过addTest(class\_name(testcasename))来添加到suite，注意testcasename不需要() 这小括号。
3. 通过TextTestRunner().run(suite)来执行用例

上面的addTest()的好处就是会按照你添加case的顺序去执行，我们可以把构建TestSuite的过程写到一个方法里，然后去调用静态方法。



仔细一看，很明显这个方法有一个缺陷，如果一个类里面有十个或者上百个测试用例，如果通过addTest()来添加，会产生很大垃圾代码，写脚本的人也会受不了。我们需要用少量的代码去构建TestSuite(),不去关心这个类里有多少个test开头的用例，接下来我们来学makeSuite()方法

1. **构建测试套件--** **makeSuite()**

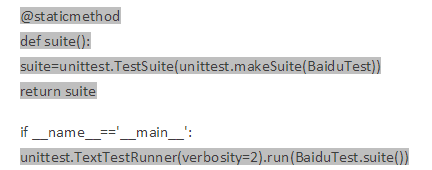
**代码如下：**

*# coding= utf-8***from** selenium **import** webdriver  
**import** unittest  
  
  
**class** BaiduTest(unittest.TestCase):  
 **def** setUp(self):  
 self.driver=webdriver.Firefox()  
 self.driver.maximize\_window()  
 self.driver.get(**'http://www.baidu.com'**)  
 self.driver.implicitly\_wait(10)  
  
 **def** tearDown(self):  
 self.driver.quit()  
  
 **def** testTitle(self):  
 self.assertEqual(**u'百度一下，你就知道'**, self.driver.title)  
  
 **def** testUrl(self):  
 self.assertEqual(**'https://www.baidu.com/'**, self.driver.current\_url)  
  
**if** \_\_name\_\_==**'\_\_main\_\_'**:  
 suite = unittest.TestSuite(unittest.makeSuite(BaiduTest))  
 unittest.TextTestRunner(verbosity=2).run(suite)

分析：

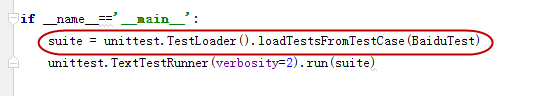
Unittest.makeSuite(class name), 利用这个可以执行这个类下所有的测试用例， 显然这个比第一种方法有了很大进步。但是还是有缺陷，如果在实际脚本开发中，特别是POM，会根据不同模块新建不同的class，有几十个或者上百个class都有可能，所以，我们需要更加智能去加载许多class，例如，按照，一次性加载一个模块或者路径下所有的class中的脚本用例。

If \_\_name\_\_ 部分可以重构如下：



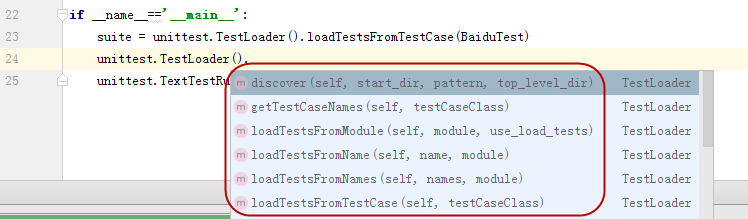
1. **用例加载TestLoader()**

TestLoader用于创建类和模块的测试套件，一般的情况下，使用TestLoader().loadTestsFromTestCase(TestClass)来加载测试类。见如下使用TestLoader调整的测试脚本（selenium部分代码和上面相同）：



分析：

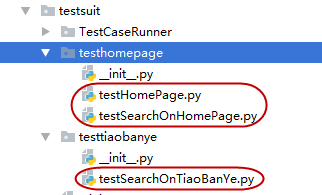
这个loadTestsFromTestCase(classname)和上面maikeSuite(classname)一样，都是把单个class文件加载到suite中去。我们看看TestLoader()还提供了哪些常用的方法：



1. **常用方法discover**

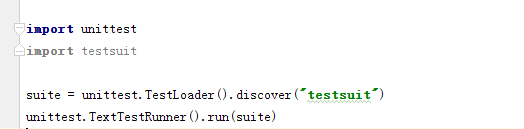
discover是通过递归的方式到其子目录中从指定的目录起始，找到所有测试模块并返回一个包含他们对象的TestSuite，然后进行加载与模式匹配唯一的测试文件。

回到我们文章开始的实际项目场景：



如果我们利用discover就很简单，只需要运行”testsuit”这个包下面的所有脚本文件。

我新建一个run.py文件，代码如下



只需要运行run.py就可以运行我项目中testsuit包下所有子包内的class文件中的test开头的测试用例。不管我testsuit下目录结构如何变化，只有有case都会被执行。

总结，最后一种方法是运行项目一个目录下所有的脚本，如果需要运行一部分，可以使用loadTestsFromModule(modelename)，如果是单元测试，少数几个方法，可以用第一个addTest().