Django项目编写单元测试用例

《Python unittest模块实现单元测试》一节我们主要讲解了Python 标准库模块 unittest 的基本使用方法，本节讲解如何在 Django 项目中编程单元测试代码，其实当我们使用 startapp 命令创建 app 应用的时候，你就会发现有一个 tests.py 文件，这个文件就是 Django 提供给开发者做单元测试的，在这个文件中给出了测试类需要继承的基类 TestCase，其中 django.test.TestCase 是 unittest.TestCase 的一个子类，实现了数据库访问以及 HTTP 请求等测试功能。

针对一个项目而言，因为它的模块众多，所以在做测试的时候为了更加直观、方便，我们通常在应用下创建一个测试目录用来承载不同模块的测试用例，按照模块的类别为不同的功能函数定义测试代码，如 test_models.py、test_vierws.py 分别指的是模层与视图层，以这样的命名方式来对文件命名。本节为了方便演示，我们只在 tests.py 中编写测试单元代码。

1.项目单元测试常用方法

接下来介绍不同场景下 Django 项目单元测试的实现方法以及执行测试用例的命令。我们以 index 应用为例，介绍三类最常见的测试场景：基础功能测试、模型测试、视图测试。

1) 基础功能测试

基础功能测试即不涉及 Django 模块的逻辑功能，它和《Python unittest模块实现单元测试》一节中使用 unittest 模块实现的测试用例基本是类似的，只不过我们在这里继承的是 Django 中的测试模块即 unittest.TestCase 的子类 django.test.TestCase 。当然如果你愿意也可以继承 unittest.TestCase，不过不推荐。我们在 index/tests.py 中编写如下代码，实现两数加和的测试用例：

```
1. from django.test import TestCase
2. class ExampleTest(TestCase):
3.     def test_addition(self):
4.         def addition(x,y):
5.             return x+y
6.         self.assertEqual(addition(1,1),2,'ass is failed') #断言函数加和运算
```

2) 模型测试

模型测试就是对 Model 的增删改查进行测试，测试类必须继承自 django.test.TestCase，它会在执行测试用例之前创建数据库，并在执行测试用例之后销毁。下面我们进行简单的测试用例代码说明，如下所示：

```
1. from index.models import Book, PubName
2.     def test_model(self):
3.         pub1=PubName.objects.create(pubname="程序帮出版社") #创建pubname实例,
4.         book=Book.objects.create(title='Servlet',price='35.00', retail_price='35.00',pub=pub1)
5.         self.assertTrue(book is not None)
6.         self.assertNotEqual(Book.objects.count(),8)#使用断言判断
7.         self.assertEqual(Book.objects.count(),9)
```

上述代码，我门做一下简单剖析，我们定义了 test_model 函数，并创建了普通用户以及书籍类实例对象，之后还是和之前操作一样，我们是使用断言的方式对今天的测试代码进行了有效的测试。

虽然这里涉及到了数据库操作，但是该操作并不会影响数据库中原有数据。这些测试用例是相互隔离的，每一个测试用例都运行在一个事务中。

3) 视图层测试

视图层测试相比其他的项目单元测试方法略显复杂一点点，在这里需要引入测试客户端，它提供了 get、post 等方法实现了对视图的访问，测试客户端被封装在了如下模块中即 django.test.Client。测试类的每一个测试方法都可以直接使用测试客户端 self.client。每一个测试方法都会新建一个测试客户端，并且彼此之间互不影响，下面我们编写如下示例：

```
1. from index.models import Book, PubName
2.     def test_view(self):
3.         pub1=PubName.objects.create(pubname="机械工业出版社")
4.         book=Book.objects.create(title='Jsp',price='25.00', retail_price='25.00',pub=pub1)
5.         response=self.client.get('/index/update_book/%d/'% book.id)
6.         response['X-Token']='C语言中文网' #自定义响应头
7.         self.assertEqual(response.status_code,200)
8.         self.assertEqual(response[X-Token],'C语言中文网','it is not same')
```

我们使用 self.client.get 方法实现了对相应视图函数的访问，然后使用断言的方法，并调用 response 响应对象的方法或者属性进行了相关的测试。

2. 项目中执行测试用例命令

Django 提供了住专门的命令用来执行测试，命令如下所示：

python manage.py test
我们可以在项目的 manage.py 文件目录下直接执行此命令，但是该命令会将所有的测试结果都输出，如果你的每个应用下都做了单元测试，这显然会给你造成很错乱的感觉，所以我们可以使用下面的命令进行相关的测试输出：

- 执行 index 应用下的所有测试用例： python manage.py test index。
- 执行 index应用下 tests 模块下定义的测试用例： python manage.py test.index.tests。
- 直接执行 tests.py 文件下测试类：python manage.py test index.tests.ExampleTest。
- 直接执行测试类下某个测试方法：python manage.py test index.tests.ExampleTest.test_view。

由此可见 test 命令之后可以提供指定的参数来执行测试用例，比如一个 Python 包、模块、测试类或者一个测试方法。我们还可以加上 -v 参数来更加详细的显示测试过程，不仅如此可以使用数字 0、1、2、3 来指定详细程度，数字越大表示输出越详细，只有这 4 个级别可选则，命令如下所示：

python manage.py test -v 1 index.tests
如何你启动测试命令后，出现如下错误：

django.db.utils.InternalError: (1366, "Incorrect...
这就们还需要在 settings.py 的 DATABASES 选项中添加如下配置：

```
'TEST': {
        'CHARSET': 'utf8',
         'COLLATION': 'utf8_general_ci'}
```

最后我们执行测试命令可得到如下的输出结果：

```
C:¥Users¥Administrator¥Book¥BookStore>python manage.py test index.tests
Creating test database for alias 'default'...
System check identified no issues (0 silenced).
.PubName object (1)
增加了新的书籍
F增加了新的书籍
F
======================================================================
FAIL: test_model (index.tests.ExampleTest)
----------------------------------------------------------------------
Traceback (most recent call last):
  File "C:¥Users¥Administrator¥Book¥BookStore¥index¥tests.py", line 26, in test_model
```

```
    self.assertEqual(Book.objects.count(),9)
AssertionError: 1 != 9
======================================================================
FAIL: test_view (index.tests.ExampleTest)
----------------------------------------------------------------------
Traceback (most recent call last):
  File "C:¥Users¥Administrator¥Book¥BookStore¥index¥tests.py", line 34, in test_view
    self.assertEqual(response['X-Token'],'C语言中文网','it is not same')
AssertionError: '=?utf-8?b?Q+ivreiogOS4reaWh+e9kQ==?=' != 'C语言中文网'
- =?utf-8?b?Q+ivreiogOS4reaWh+e9kQ==?=
+ C语言中文网
: it is not same
----------------------------------------------------------------------
Ran 3 tests in 0.033s

FAILED (failures=2)
Destroying test database for alias 'default'...
```

Django 项目的单元测试到这里就讲解完毕了，我们通过三节知识的讲解，让大家认识了什么是单元测试、单元测试模块 unittest 是如何应用的以及 Python unittest 模块如何在 Django 中进行使用，希望大家通过这些介绍掌握知识的精髓。