数据驱动

讲师：潘sir

ddt

产生背景

- 假如需要测试一百组登录场景，每组的过程都是一样，只是入参和预期结果不一样，如果一个一个写，则需要写100个用例，造成代码冗长，大量重复且难管理，所以引入数据驱动的思维，不需要关注代码，只需要关注测试数据

数据驱动

- 概念：用数据来控制测试的业务流。比如你测一个WEB程序，有很多页面，你可以通过一个数据来控制每次是在哪个页面下工作的（即通过数据来导航到相应的页面）。它是关键字驱动的低级版本，他控制的是函数级的，而关键字是控制动作级的
- 基于数据驱动的测试框架的特点：
    1. 数据(测试数据/预期结果)和驱动分离，每个独立的数据 + 一次驱动 = 一条测试用例
    2. 可应付大量重复逻辑的测试场景，比如：一百个不同类型的账号登录
    3. 数据驱动的数据，不仅仅是excel，一切数据皆可，如：ODBC源文件、Excel文件、Csv文件、ADO对象文件等，只是通常拿excel来做
    4. 测试脚本作为驱动存在，或者说是一个传送数据的机制
    5. 数据驱动不是框架，而是一种设计思想

ddt使用

- 这是一个python写的一个框架，运用的正是基于数据驱动的思想
- 安装：`pip install ddt==1.1.2` 最新版本有bug
- ddt传参实例：

```
import unittest
import ddt

# 测试数据
data = [
    {"user": "u1", "passwd": "p1", "exp": "结果1"},
    {"user": "u2", "passwd": "p2", "exp": "结果2"},
    {"user": "u3", "passwd": "p3", "exp": "结果3"},
    {"user": "u4", "passwd": "p4", "exp": "结果4"},
]


# 装饰类
```

```
@ddt.ddt
class TestDDT(unittest.TestCase):
    # 带星号，可以依次取出内部元素
    @ddt.data(*data)
    def test_01(self, testdata):
        # 通过testdata传递参数，参数名可随意命名
        print(testdata)
        user = 'panwj'
        passwd = '123'
        exp = '结果'  # 预期结果
        res = '结果'  # 实际结果
        self.assertEqual(exp, res)


if __name__ == '__main__':
    unittest.main(argv=['ignored', '-v'], exit=False)
```

```
test_01_1 (__main__.TestDDT) ... ok
test_01_2 (__main__.TestDDT) ... ok
test_01_3 (__main__.TestDDT) ... ok
test_01_4 (__main__.TestDDT) ...

{'user': 'u1', 'passwd': 'p1', 'exp': '结果1'}
{'user': 'u2', 'passwd': 'p2', 'exp': '结果2'}
{'user': 'u3', 'passwd': 'p3', 'exp': '结果3'}
{'user': 'u4', 'passwd': 'p4', 'exp': '结果4'}
```

```
ok

----------------------------------------------------------------------
Ran 4 tests in 0.004s

OK
```

- ddt登录实例

```
import unittest
import ddt
from tsms.tsms_login_cls import TestLogin

# 测试数据
data = [
    {"user": "panwj", "passwd": "123", "exp": True},
    {"user": "u2", "passwd": "p2", "exp": False},
    {"user": "u3", "passwd": "p3", "exp": False},
    {"user": "u4", "passwd": "p4", "exp": False},
]
```

```python
# 装饰类
@ddt.ddt
class TestDDT(unittest.TestCase):
    def setUp(self) -> None:
        # 实例化
        self.t = TestLogin()

    def tearDown(self) -> None:
        self.t.close()

    # 带星号，可以依次取出内部元素
    @ddt.data(*data)
    def test_01(self, testdata):
        '''验证登录'''
        # 获取参数
        user = testdata["user"]
        passwd = testdata["passwd"]
        # 登录操作
        self.t.login_c(user, passwd)
        # 获取预期结果
        exp = testdata["exp"]
        # 获取实际结果
        res = self.t.is_login(user)
        self.assertEqual(exp, res)


if __name__ == '__main__':
    unittest.main(argv=['ignored', '-v'], exit=False)
```

```
test_01_1 (__main__.TestDDT)
验证登录 ... ok
test_01_2 (__main__.TestDDT)
验证登录 ... ok
test_01_3 (__main__.TestDDT)
验证登录 ... ok
test_01_4 (__main__.TestDDT)
验证登录 ...

获取页面token:
ImEyYWNlYTk4NmYwZTZkYWMyNTBhOTZjNTliNWI3ZjRlZmQ4N2IwMTAi.EIqZiw.GyYI0d1u1JJKxJ
kwgjYk81pfiMg
登录成功
获取页面token:
ImIwNWVhNGQwYjQ4MzJiN2ViMDk2ZTM1ZjA5YjdkN2MxMmI1NzllODAi.EIqZjA.iIF0cdT_cOWb1z
lXXtMD1m7nINE
登录失败
```

```
获取页面token:
IjBkZDM2OTM4MWFhNWZiZTliYzM5ZmIwMjdhNzQxY2U5ZTc5NDY0NjYi.EIqZjA.E-
EDeFtrdmLrmM1dLVvsv174zOM
登录失败
获取页面token:
IjIyOWIwMjkxNjM4ZjZmODg5OTIyYmY3ZWVlODUzNGYyNjA3NzYwZTIi.EIqZjA.5rOUuJcN3cThT1
W2fgIg39V0B40
登录失败
```

```
ok

----------------------------------------------------------------------
Ran 4 tests in 0.261s

OK
```

excel读数据

读excel数据

- 使用python模块：`pip install xlrd`
- 读取excel数据实例：

```python
import xlrd


class ExcelUtil():
    def __init__(self, excelPath, sheetName):
        self.data = xlrd.open_workbook(excelPath)
        self.table = self.data.sheet_by_name(sheetName)
        # 获取第一行作为key值
        self.keys = self.table.row_values(0)
        # 获取总行数
        self.rowNum = self.table.nrows
        # 获取总列数
        self.colNum = self.table.ncols

    def dict_data(self):
        if self.rowNum <= 1:
            print("总行数小于1")
        else:
            r = []
            # 表示从第二行开始取
            j = 1
            # 21行，要取20个，range(0, 20)
            for i in range(self.rowNum - 1):
                # 声明一个空字段
```

```
                s = {}
                # 从第二行取对应的values值
                values = self.table.row_values(j)
                # 遍历这行的列
                for x in range(self.colNum):
                    # self.keys[x] => list(user, passwd, exp) [0] => user
                    # values[x] => list(dcs, 123, True)[0] => dcs
                    s[self.keys[x]] = values[x]
                # 把字段添加到列表
                r.append(s)
                # 序号增加1
                j += 1
            return r


if __name__ == '__main__':
    filePath = 'test_data.xlsx'
    sheetName = 'Sheet1'
    data = ExcelUtil(filePath, sheetName)
    print(data.dict_data())
```

```
[{'user': 'dcs', 'passwd': '123', 'exp': ''}, {'user': 'dcs1', 'passwd':
'124', 'exp': 'FALSE'}, {'user': 'dcs2', 'passwd': '125', 'exp': 'FALSE'},
{'user': 'dcs3', 'passwd': '126', 'exp': 'FALSE'}, {'user': 'dcs4', 'passwd':
'127', 'exp': 'FALSE'}, {'user': 'dcs5', 'passwd': '128', 'exp': 'FALSE'},
{'user': 'dcs6', 'passwd': '129', 'exp': 'FALSE'}, {'user': 'dcs7', 'passwd':
'130', 'exp': 'FALSE'}, {'user': 'dcs8', 'passwd': '131', 'exp': 'FALSE'},
{'user': 'dcs9', 'passwd': '132', 'exp': 'FALSE'}, {'user': 'dcs10', 'passwd':
'133', 'exp': 'FALSE'}, {'user': 'dcs11', 'passwd': '134', 'exp': 'FALSE'},
{'user': 'dcs12', 'passwd': '135', 'exp': 'FALSE'}, {'user': 'dcs13',
'passwd': '136', 'exp': 'FALSE'}, {'user': 'dcs14', 'passwd': '137', 'exp':
'FALSE'}, {'user': 'dcs15', 'passwd': '138', 'exp': 'FALSE'}, {'user':
'dcs16', 'passwd': '139', 'exp': 'FALSE'}, {'user': 'dcs17', 'passwd': '140',
'exp': 'FALSE'}, {'user': 'dcs18', 'passwd': '141', 'exp': 'FALSE'}, {'user':
'dcs19', 'passwd': '142', 'exp': 'FALSE'}]
```

读取时需要注意：

1. 文件路径，实际使用时应该使用相对路径
2. 数字格式，读取出来是浮点型，需要设置单元格格式，特别注意：要先设置单元格格式为文本格式，再编辑才可生效

excel数据驱动用例

- 从外部(excel)读取数据，再进行测试

```
import unittest
import ddt
from tsms.tsms_login_cls import TestLogin
```

```
from tsms.read_excel import ExcelUtil

filePath = 'test_data.xlsx'
sheetName = 'Sheet1'
data = ExcelUtil(filePath, sheetName)
test_data = data.dict_data()
print(test_data)

# 装饰类
@ddt.ddt
class TestDDT(unittest.TestCase):
    def setUp(self) -> None:
        # 实例化
        self.t = TestLogin()

    def tearDown(self) -> None:
        self.t.close()

    # 带星号，可以依次取出内部元素
    @ddt.data(*test_data)
    def test_01(self, a):
        '''验证登录'''
        # 获取参数
        user = a["user"]
        passwd = a["passwd"]
        # 登录操作
        self.t.login_c(user, passwd)
        # 获取预期结果
        exp = eval(a["exp"])
        print("exp is", exp, a["exp"])
        # 获取实际结果
        res = self.t.is_login(user)
        self.assertEqual(exp, res)


if __name__ == '__main__':
    unittest.main(argv=['ignored', '-v'], exit=False)
#     a = "False"
#     print(bool(a))
```

- 练习：利用数据驱动，测试模版接口的异常场景，要求excel驱动字段指定：data/user/passwd/status/text
- 注意：
    1. json格式的转换问题
    2. excel的格式问题，建议使用文本格式

```
import unittest, json
import ddt
```

```
from tsms.tsms_login_cls import TestLogin
from tsms.read_excel import ExcelUtil
from tsms.tsms_base import Tsmstest
from tsms.tsms_decorator import logging
ts = Tsmstest()
filePath = 'sign_fail_data.xlsx'
sheetName = 'Sheet1'
data = ExcelUtil(filePath, sheetName)
test_data = data.dict_data()
# print(test_data)

@ddt.ddt
class Testsignddt(unittest.TestCase):

    @ddt.data(*test_data)
    def test_temp_fail(self, a):
        data = a["data"]
        user = a["user"]
        passwd = a["passwd"]
        status = a["status"]
        text = a["text"]
        ts.req_post('temp', data=data,user=user, passwd=passwd)
        # 实际结果 == 预期结果
#         print(ts.status_code)
        logging.info("{} {}".format(ts.status_code, status))
        assert str(ts.status_code) == str(status)
        if text:
            assert ts.json == json.loads(text)

if __name__ == '__main__':
    unittest.main(argv=['ignored', '-v'], exit=False)
```

```
test_temp_fail_1 (__main__.Testsignddt) ... 2019-10-19 22:31:08,342 INFO
[当前被调用方法是]: req_post
2019-10-19 22:31:08,343 INFO     [url_type is]: temp
2019-10-19 22:31:08,344 INFO     [data is]: {"name": "名称", "template": "创建签
名1", "type": 1, "description": "谢谢", "sign_id": 1}
2019-10-19 22:31:08,345 INFO     [user is]: dcs
2019-10-19 22:31:08,346 INFO     [passwd is]: 123
2019-10-19 22:31:08,347 INFO     [当前请求的地址是]:
http://127.0.0.1:5001/v1/template
2019-10-19 22:31:08,347 INFO     [发送内容是]: {'name': '名称', 'template': '创建
签名1', 'type': 1, 'description': '谢谢', 'sign_id': 1} <class 'dict'>
2019-10-19 22:31:08,514 INFO     [返回码是:] 403
2019-10-19 22:31:08,515 INFO     [返回内容是]: {'error': 'ER:0023', 'message':
'sign id is not belong to user'}
```

```
2019-10-19 22:31:08,516 INFO     [执行结果为]: {'error': 'ER:0023', 'message':
'sign id is not belong to user'}
2019-10-19 22:31:08,516 INFO     403 403
ok
test_temp_fail_2 (__main__.Testsignddt) ... 2019-10-19 22:31:08,519 INFO
[当前被调用方法是]: req_post
2019-10-19 22:31:08,519 INFO     [url_type is]: temp
2019-10-19 22:31:08,520 INFO     [data is]: {"name": "名称", "template": "创建签
名2", "type": 1, "description": "谢谢", "sign_id": null}
2019-10-19 22:31:08,521 INFO     [user is]: dcs
2019-10-19 22:31:08,522 INFO     [passwd is]: 123
2019-10-19 22:31:08,522 INFO     [当前请求的地址是]:
http://127.0.0.1:5001/v1/template
2019-10-19 22:31:08,523 INFO     [发送内容是]: {'name': '名称', 'template': '创建
签名2', 'type': 1, 'description': '谢谢', 'sign_id': None} <class 'dict'>
2019-10-19 22:31:08,662 INFO     [返回码是:] 500
2019-10-19 22:31:08,663 INFO     [执行结果为]: <!DOCTYPE HTML PUBLIC "-
//W3C//DTD HTML 3.2 Final//EN">
<title>500 Internal Server Error</title>
<h1>Internal Server Error</h1>
<p>The server encountered an internal error and was unable to complete your
request. Either the server is overloaded or there is an error in the
application.</p>

2019-10-19 22:31:08,663 INFO     500 500
ok
test_temp_fail_3 (__main__.Testsignddt) ... 2019-10-19 22:31:08,665 INFO
[当前被调用方法是]: req_post
2019-10-19 22:31:08,666 INFO     [url_type is]: temp
2019-10-19 22:31:08,667 INFO     [data is]: {"name": "名称", "template": "创建签
名3", "type": 1, "description": "谢谢", "sign_id": ""}
2019-10-19 22:31:08,668 INFO     [user is]: dcs
2019-10-19 22:31:08,668 INFO     [passwd is]: 123
2019-10-19 22:31:08,669 INFO     [当前请求的地址是]:
http://127.0.0.1:5001/v1/template
2019-10-19 22:31:08,670 INFO     [发送内容是]: {'name': '名称', 'template': '创建
签名3', 'type': 1, 'description': '谢谢', 'sign_id': ''} <class 'dict'>
2019-10-19 22:31:08,816 INFO     [返回码是:] 500
2019-10-19 22:31:08,817 INFO     [执行结果为]: <!DOCTYPE HTML PUBLIC "-
//W3C//DTD HTML 3.2 Final//EN">
<title>500 Internal Server Error</title>
<h1>Internal Server Error</h1>
<p>The server encountered an internal error and was unable to complete your
request. Either the server is overloaded or there is an error in the
application.</p>

2019-10-19 22:31:08,818 INFO     500 500
ok
```

```
test_temp_fail_4 (__main__.Testsignddt) ... 2019-10-19 22:31:08,819 INFO
[当前被调用方法是]: req_post
2019-10-19 22:31:08,820 INFO     [url_type is]: temp
2019-10-19 22:31:08,821 INFO     [data is]: {"name": "名称", "template": "创建签
名4", "type": 4, "description": "谢谢", "sign_id": 424}
2019-10-19 22:31:08,821 INFO     [user is]: dcs
2019-10-19 22:31:08,822 INFO     [passwd is]: 123
2019-10-19 22:31:08,823 INFO     [当前请求的地址是]:
http://127.0.0.1:5001/v1/template
2019-10-19 22:31:08,823 INFO     [发送内容是]: {'name': '名称', 'template': '创建
签名4', 'type': 4, 'description': '谢谢', 'sign_id': 424} <class 'dict'>
2019-10-19 22:31:08,966 INFO     [返回码是:] 403
2019-10-19 22:31:08,967 INFO     [返回内容是]: {'error': 'ER:0022', 'message':
'template type fail'}
2019-10-19 22:31:08,968 INFO     [执行结果为]: {'error': 'ER:0022', 'message':
'template type fail'}
2019-10-19 22:31:08,968 INFO     403 403
ok
test_temp_fail_5 (__main__.Testsignddt) ... 2019-10-19 22:31:08,969 INFO
[当前被调用方法是]: req_post
2019-10-19 22:31:08,970 INFO     [url_type is]: temp
2019-10-19 22:31:08,971 INFO     [data is]: {"name": "名称", "template": "创建签
名5", "type": 0, "description": "谢谢", "sign_id": 424}
2019-10-19 22:31:08,971 INFO     [user is]: dcs
2019-10-19 22:31:08,972 INFO     [passwd is]: 123
2019-10-19 22:31:08,973 INFO     [当前请求的地址是]:
http://127.0.0.1:5001/v1/template
2019-10-19 22:31:08,973 INFO     [发送内容是]: {'name': '名称', 'template': '创建
签名5', 'type': 0, 'description': '谢谢', 'sign_id': 424} <class 'dict'>
2019-10-19 22:31:09,127 INFO     [返回码是:] 500
2019-10-19 22:31:09,127 INFO     [执行结果为]: <!DOCTYPE HTML PUBLIC "-
//W3C//DTD HTML 3.2 Final//EN">
<title>500 Internal Server Error</title>
<h1>Internal Server Error</h1>
<p>The server encountered an internal error and was unable to complete your
request. Either the server is overloaded or there is an error in the
application.</p>

2019-10-19 22:31:09,128 INFO     500 500
ok
test_temp_fail_6 (__main__.Testsignddt) ... 2019-10-19 22:31:09,129 INFO
[当前被调用方法是]: req_post
2019-10-19 22:31:09,130 INFO     [url_type is]: temp
2019-10-19 22:31:09,131 INFO     [data is]: {"name": "名称", "template": "创建签
名6", "type": 1, "description": "谢谢", "sign_id": 424}
2019-10-19 22:31:09,132 INFO     [user is]: dcs
2019-10-19 22:31:09,132 INFO     [passwd is]: 1234
2019-10-19 22:31:09,133 INFO     [当前请求的地址是]:
http://127.0.0.1:5001/v1/template
```

```
2019-10-19 22:31:09,133 INFO     [发送内容是]: {'name': '名称', 'template': '创建
签名6', 'type': 1, 'description': '谢谢', 'sign_id': 424} <class 'dict'>
2019-10-19 22:31:09,269 INFO     [返回码是:] 400
2019-10-19 22:31:09,269 INFO     [返回内容是]: {'error': 'ER:0001', 'message':
'auth not pass'}
2019-10-19 22:31:09,270 INFO     [执行结果为]: {'error': 'ER:0001', 'message':
'auth not pass'}
2019-10-19 22:31:09,271 INFO     400 400
ok

----------------------------------------------------------------------
Ran 6 tests in 0.931s

OK
```

数据驱动的优缺点

- 优点
    1. 精简重复代码
    2. 专注于测试数据，针对接口数据编写用例，提高效率
    3. 对于某些验证性测试特别适用，如：验证一批账户的登录权限
- 缺点
    1. 数据动态化困难
    2. 只能测试特定场景，并非所有测试场景都可以数据驱动
    3. 即用关键字驱动，又用数据驱动，不便统一管理