ui自动化测试day10

一、 pytest常用插件

2、控制用例执行顺序

- unittest测试用例执行顺序是根据测试方法名称的assicc码值的大小来的，值越小排在前面(a-z)
- pytest 正常情况下是根据测试方法的从上到下的顺序来执行

  可以通过 pytest-ordering 插件来控制pytest测试方法执行的顺序。
- 安装:
  - 在线安装： pip install pytest-ordering
  - 离线安装: 下载对应的离线安装包，解压后，并进入到对应的目录，执行 python setup.py install
  - pycharm
- 使用

  @pytest.mark.run(order=x) # x 表示的是整数， (既可以是负数也可以是正数)
  - 全为负数或者正数时，值越小，优先级越高
  - 既有正数，又有负数，那么正数优先级高
  - 没有确定执行顺序的用例优先于负数

```
@pytest.mark.last     # 设置用例最后执行
```

3、失败重试

- pytest-rerunfailures 安装

    - 在线安装
    - 离线安装
    - pycharm
- 使用

  在addopts参数行中增加对应的参数项：--reruns 3

  当重复执行成功时，就不会再重复执行。

二、 pytest高级用法

1、跳过

@pytest.mark.skipif(condition, reason=None)

condition 表示是跳过的条件

这里面reason参数名称必填。

@pytest.mark.skip(reason=None)

reason表示的是跳过的原因

可以在测试类和测试方法上使用

```
import pytest


def add(x, y):
    return x+y

version = 21

class TestAdd:
```

```
    # @pytest.mark.last    # 设置用例最后执行
    def test_add_01(self):
        result = add(1, 2)
        assert 3 == result

    @pytest.mark.skipif(version > 20, reason="大于
2.0的版本不需要再执行此用例")
    # @pytest.mark.skip("版本已更新，不需要再进行测试")
    @pytest.mark.run(order=0)
    def test_add_02(self):
        result = add(2, 2)
        assert 4 == result

    @pytest.mark.run(order=-2)
    def test_add_03(self):
        result = add(3, 2)
        assert 5 == result
```

2、数据的参数化

- pytest参数化实现： @pytest.mark.parameterize(argnames, argvalues)
  - argnames 表示是 参数名字，是一串字符， 多个参数之间由逗号隔开 "username，password"
  - argvalues 表示的是参数化的数据 [("13700001111","123124"),("13800011111","123456")]

    argname的参数个数要与argvalues里面的测试数据的个数要相同，否则会报错。

```
import pytest


def add(x, y):
    return x+y


class TestAdd:
    @pytest.mark.parametrize("x,y,expect", [(1, 2, 3), (2, 2, 4), (3, 2, 5)])
    def test_add_01(self, x, y, expect):
        result = add(x, y)
        assert expect == result
```

三、 PO模式

1、V1版本介绍

V1版本缺点：

2、V2版本实现（通过unittest管理用例）

V2版本缺点：

3、V3版本

3.1 方法封装

3.2 V3版本代码实现

4、V4版本实现

4.1 PO模式介绍

4.2 PO模式优点

4.3 PO模式实现

V4版本缺点：