pytest插件与扩展

- pytest-assume：多重校验
- pytest-cov: 生成测试覆盖率报告
- pytest-html: 生成 html 报告
- pytest-instafail：修改pytest默认行为
- pytest-ordering：指定用例执行顺序
- pytest-rerunfailures: 失败重试
- pytest-repeat：重复执行
- pytest-timeout: 超时测试
- pytest-tld：省略无关信息
- pytest-sugar：控制台进度条显示
- pytest-pep8: 检测代码是否符合 PEP8 规范
- pytest-xdist：多线程执行用例
- pytest-forked：xdist的依赖模块
- pytest-randomly：测试顺序随机
- pytest-allure-adaptor：支持allure测试报告
- pytest-cache：重跑上次失败的用例，持续集成中很实用，提高分析效率
- pytest-selenium：集成selenium
- pytest-bdd：行为驱动

简易测试报告

pytest-html是pytest的测报告插件，功能简易。安装方法`pip install pytest-html`

运行命令：`pytest xxx.py --html=report.html`

- 执行失败，导入不了包怎么办?

在你们自己的`site-packages`下的mypython.pth添加项目路径和包的路径 即可

常用插件

多重校验

一般断言的话，如果前一个抛出异常，后面的则不进行断言，通过`pytest-assume`可以进行多重断言。安装：`pip install pytest-assume`

```python
import pytest


def test_add():
    pytest.assume(1 + 1 == 3)
    pytest.assume(1 + 1 == 3)
    pytest.assume(1 + 1 == 4)
```

用例执行顺序

安装：`pip install pytest-ordering`，通过装饰器`@pytest.mark.run(order=2)`指定执行顺序

```python
import pytest

# 这条后执行
@pytest.mark.run(order=2)
def test_order1():
    print("first test")
    assert True

# 这条先执行
@pytest.mark.run(order=1)
def test_order2():
    print("second test")
    assert True
```

用例超时

标记用例超时时间：`pip install pytest-timeout`

```python
# 用例执行超过10秒则报超时
@pytest.mark.timeout(10)
def test_01():
    pass
```

用例重试

用例执行失败后，进行错误重试：`pip install pytest-rerunfailures`

```python
# 最多重跑5次，重跑间隔1秒
@pytest.mark.flaky(reruns=5, reruns_delay=1)
def test_02()
    pass
```

还可以使用命令，从外部指定参数：`pytest --reruns 5 --reruns-delay 1`

用例重复执行

安装：`pip install pytest-repeat`

基本用法

使用：`py.test --count=10 xxx.py` 重复执行指定用例10次

指定范围

`--repaet-scope`可以指定参数：`session`，`moudule`，`class`，`function`。例如：`pytest xxx.py -s --count=5 --repeat-scope=session`重复执行整个session用例

装饰器指定

通过装饰器指定`@pytest.mark.repeat(5)`后，执行时不需要指定，即可重复执行

```
@pytest.mark.repeat(5)
def test_01(start, open_baidu):
    print("执行用例")
    assert 1 == 1
```

间歇性缺陷

比如需要测试间歇性bug，即：重复执行，直到断言失败。所以可以结合使用`-x`和`pytest-repeat`。例如：`py.test --count=1000 -x xxx.py`，执行1000次，直到失败才停止

多进程运行

安装：`pip install pytest-xdist`

1. 多进程运行：一般cpu几个核心，就启几个进程。运行命令：`pytest -n 4` 即可启动四个进程同时跑用例
2. 自动重新运行失败的测试：`pytest --looponfail`

随机运行

安装：`pip install pytest-randomly`

命令：`pytest --randomly-seed=123` 指定随机种子，即可打乱执行顺序

补充插件

对测试人员来说，用的相对较少的插件

覆盖率检查

单元测试中非常强大的插件！安装：`pip install pytest-cov`

运行：`pytest test_tsms_sign_api_create.py --cov=../../tsms_pytest_commons/` 可以检查针对tsms_pytest_commons模块的单元测试覆盖率

仅测试更改的文件

安装：`pip install pytest-picked`

运行：`pytest --picked` 对你已经修改但尚未提交 git 的代码运行测试

修改pytest默认行为

安装：`pip install pytest-instafail`

修改 pytest 的默认行为来立即显示失败和错误，而不是等到 pytest 完成所有测试。

其他

1. `pytest-django`
2. `django-test-plus`
3. `pytest-bdd`：行为驱动 BDD

花里胡哨插件

1. `pip install pytest-sugar`：控制台进度条，更美观
2. `pip install pytest-pep8`：强制pep8代码风格，否则用例算不通过。有代码洁癖者使用
3. `pip install pytest-tldr`：`tldr` 表示`too long ,don't read--太长了，不看`，可以省略很多无关信息，添加`-v` 可以重新补充详细输出

配置管理

pytest会从所有测试目录下寻找配置文件，配置文件有三种，优先级为：`pytest.ini` > `tox.ini` > `setup.cfg`

一般使用`pytest.ini`，放到项目的根目录即可