单元测试框架和自动化测试框架有什么关系？ (1)什么是自动化测试框架 它就是自动化测试组长针对一个项目开发的一个代码框架，这个框架封装了很多的基础模 块，报告模块等等。 (2)作用 1.提高测试效率，降低自动化用例的维护成本 2.减少人工干预，提高测试的准确性，增加代码的重用性。 3.核心思想是让不懂代码的人也能够通过这个框架去实现自动化测试。 (3)pytest单元测试框架和自动化测试框架的关系 pytest单元测试框架：只是自动化测试框架中的组成部分之一。 pom设计模式：只是自动化测试框架中的组成部分之一。 数据驱动:..... 关键字驱动 全局配置文件的封装 日志监控 selenium，requests二次封装 断言 报告邮件 更多.....

接口自动化引入用例管理框架 为什么需要使用用例管理框架，它主要完成了哪些事情？ 1.发现用例：从多个py文件收集并加载测试用例。 2.执行用例：按照一定的顺序执行。 3.判断结果：通过断言判断预期接口和实际的实际是否一致。 4.生成报告：统计测试进度，耗时，通过率等。

运行参数及其配置化

常用的配置 -v： 显示详细信息 -s : 关闭"输出捕获" -n X: 使用X个进程，并行化执行用例（多少个CPU，使用X就等于多少） -n auto： 自动选项进程数 进行执行 --html=Path： 生成HTML测试报告，并保存再Path路径 --self-contained-html : HTML文件自包含文件， --reruns X： 测试用例失败之后，重试X次

通过pytest.ini文件运行 注意： 1.不管是命令行方式还是主函数的方式都会自动的读取这个配置文件 2.pytest.ini文件可以改变pytest默认的测试用例的规则 3.这个文件一般是放在项目的根目录下

```
[pytest]
#配置参数
addopts = -vs
#配置测试用例文件夹
testpaths = ./testcases
#配置测试模块的规则
python_files = test_*.py
#配置测试类的规则
python_classes = Test*
#配置测试方法的规则
python_functions = test*
#配置接口测试的基础路径
base_url = http://127.0.0.1/
#给用例分组
markers =
    smoke:冒烟测试
    usermanage:用户管理
```

在用例上面加上： @pytest.mark.smoke @pytest.mark.usermanage 在执行时需要使用： -m 分组名 or 分组名

Pytest用例执行顺序 默认：是从上到下的顺序 可以通过如下标记改变测试用例的执行顺序 @pytest.mark.run(order=1)

Pytest前后置条件 在所有类，所有用例之前或之后

```
def setup_class(self):
    print("在类之前的操作")

def teardown_class(self):
    print("在类之后的操作")

def setup(self):
    print("在所有用例之前的前置操作")

def teardown(self):
    print("在所有用例之后的后置操作")
```

- 希望在部分用例之前或之后执行。使用Fixture

  Fixture装饰器完整结构如下：

  @pytest.fixture(scope="作用域",autouser="自动执行",params="数据驱动",ids="参数别名",name="fixture别名")

  scope：标记fixture的作用域

  function：函数级别（可以手动，也可以自动）

  class:类级别（一般是自动）

  module:模块级别（一般是自动）

  package/session:会话级别（一般是自动）：明天封装框架用法。

  autouser=True 自动执行

  params数据驱动

  ```
  # 新建一个fixture
  @pytest.fixture(scope="module",autouse=False,params=read_yaml())
  def execute_sql(request):
      print("执行数据库查询")
      yield request.param
      print("关闭数据库连接")
  # 这里的request参数和request.param值都是固定写法。
  ```

  ids参数别名

  ```
  #新建一个fixture
  @pytest.fixture(scope="module",autouse=False,params=read_yaml(),ids=['yz','tc'])
  def execute_sql(request):
      print("执行数据库查询")
      yield request.param
      print("关闭数据库连接")
  ```

  name表示fixture的别名

  当使用了name起别名之后，那么原来的fixture的名称就失效了。

- 一般情况下fixture会和conftest.py文件一起使用。

  注意：

  conftest.py是专门用于存放fixture的，是固定名称

  conftest.py文件的方法使用时不需要导包

  conftest.py文件可以有多个

pytest结合allure-pytest插件生成allure测试报告

1. 下载，解压，配置path路径。

   https://github.com/allure-framework/allure2/releases

   path路径配置：E:\allure-2.13.7\bin

   验证：allure --version

   问题：dos可以验证但是pycharm验证失败，怎么办，重启pycharm.
2. 加入命令生成json格式的临时报告。

   --alluredir ./temp
3. 生成allure报告

   os.system('allure generate ./temp -o ./report --clean')

   allure generate　　命令，固定的

   ./temp　　临时的json格式报告的路径

   -o　　输出output

   ./report　　生成的allure报告的路径

   --clean　　清空./report 路径原来的报告

插件

```
allure-pytest
pytest-html
pytest-ordering
pytest-rerunfailures
```