今日内容：用例管理框架pytest之fixtrue,conftest.py,allure报告以及logo定制（上）

@上课老师：码尚学院_百里老师

@老师邮箱：2971330037@qq.com

一、使用fixture实现部分前后置

setup/teardown

setup_class/teardown_class

语法：

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

scope="作用域"

functioin:在每个方法(测试用例)的前后执行一次。

class:在每个类的前后执行一次。

module：在每个py文件前后执行一次。

package/session:每个package前后执行一次。

1.function级别:在每个函数的前后执行

```
@pytest.fixture(scope="function")
def execute_sql():
 print("执行数据库的验证，查询数据库。")
 yield
 print("关闭数据库的连接")
```

调用：

```
def test_baili(self,execute_sql):
 print("百里老师"+execute_sql)
```

yield和return，都可以返回值，并且返回的值可以在测试用例中获取。

yield生成器，反复一个对象，对象中可以有多个值，yield后面可以接代码。

return 返回一个值，return后面不能接代码。

注意：如果加入autouse=True参数，那么表示自动使用，那么和setup、teardown功能一致。

2.class级别：在每个类的前后执行一次。

```
@pytest.fixture(scope="class")
def execute_sql():
 print("执行数据库的验证，查询数据库。")
 yield
 print("关闭数据库的连接")
```

调用

```
@pytest.mark.usefixtures("execute_sql")
class TestApi2:

 def test_duo_class(self):
 print("多个类的情况")
```

3.module级别：在每个模块的前后执行一次。和setup_module和teardown_module效果一样。

```
@pytest.fixture(scope="module",autouse=True)
def execute_sql():
 print("执行数据库的验证，查询数据库。")
 yield "1"
 print("关闭数据库的连接")
```

4.package、sesion级别，一般是和connftest.py文件一起使用。

autouse=True 自动调用

params=数据（list，tuple，字典列表，字典元祖）

```
def read_yaml():
 return ['甄子丹','成龙',"菜10"]

@pytest.fixture(scope="function",params=read_yaml())
def execute_sql(request):
 print(request.param)
 print("执行数据库的验证，查询数据库。")
 yield request.param
 print("关闭数据库的连接")
```

这里的params用于传输数据（list，tuple，字典列表，字典元祖），需要在夹具里面通过request（固定写法）接收，然后通过request.param（固定写法）获取数据，然后再通过yield把数据返回到测试用例中，然后使用。

ids参数：它要和params一起使用，自定义参数名称。意义不大。了解即可

```
def read_yaml():
 return ['甄子丹','成龙',"菜10"]

@pytest.fixture(scope="function",params=read_yaml(),ids=['zzd','cl','cyl'])
def execute_sql(request):
 print(request.param)
 print("执行数据库的验证，查询数据库。")
 yield request.param
 print("关闭数据库的连接")
```

name参数：对fixtrue固件取的别名。意义不大。了解即可，用了别名后，那么真名会失效，只能使用别名。

```
def read_yaml():
 return ['甄子丹','成龙',"菜10"]

@pytest.fixture(scope="function",params=read_yaml(),ids=['zzd','cl','cyl'],name="jj")
def execute_sql(request):
 print(request.param)
 print("执行数据库的验证，查询数据库。")
 yield request.param
 print("关闭数据库的连接")
```

二、当fixture的级别为package，session时，那么一般和conftest.py文件一起使用。

1.名称是固定的conftest.py，主要用于单独的存放fixture固件的。

2.级别为package，sesion时，那么可以在多个包甚至多个py文件里面共享前后置。

举例：登录。

模块：模块的共性。

3.发现conftest.py文件里面的fixture不需要导包可以直接使用。
4.conftest。py文件，可以有多个。

作用：出现重复日志，初始化一次日志对象。规避日志重复。连接数据库。关闭数据库。

注意：多个前置同时存在的优先级。
1.conftest.py为函数级别时优先级高于setup/teardown
2.conftest.py为class级别时优先级高于setup_class/teardown_class
3.conftest.py为session级别时优先级高于setup_module/teardown_module

三、pytest断言

使用的是python原生的assert

四、pytest结合allure-pytest实现生成allure报告

第一步
1.官网下载allure文件：https://github.com/allure-framework/allure2/releases
2.下载之后解压到非中文的目录
3.把bin路径配置到系统变量path中：E:\allure-2.13.7\bin （注意分号不要是中文的）

第二步：
安装allure报告：pip install allure-pytest
验证：allure --version
注意：可能需要重启pycharm。

第三步：
1.pytest.ini文件如下

```
[pytest]
addopts = -vs --alluredir=reports/temps --clean-alluredir
testpaths = testcases/
python_files = test_*.py
python_classes = Test*
python_functions = test_*
#鐢九緥鍒嗙粍
markers =
  smoke:冒烟测试
   productmanage:商品管理模块
```

加上--clean-alluredir表示：每执行一次把原来的清除。

2.all.py文件如下：

```
if __name__ == '__main__':
 pytest.main()
 time.sleep(3)
 os.system("allure generate reports/temps -o reports/allures --clean")
```

加上--clean表示：每执行一次把原来的清除。

第四步：实现logo定制。

1.修改E:\allure-2.13.7\config下的allure.yml配置文件，加入：自定义logo插件。最后一句。

```
plugins:
 - junit-xml-plugin
 - xunit-xml-plugin
 - trx-plugin
 - behaviors-plugin
 - packages-plugin
 - screen-diff-plugin
 - xctest-plugin
 - jira-plugin
  - xray-plugin
  - custom-logo-plugin
```

2.修改插件里面的图片和样式

```
/*
.side-nav__brand {
 background: url('custom-logo.svg') no-repeat left center !important;
 margin-left: 10px;
}
*/

.side-nav__brand{
 background: url('logo.png') no-repeat left center !important;
  margin-left: 20px;
  height: 90px;
  background-size: contain !important;
 }

```

```
.side-nav__brand-text{
 display: none;
}
```