Pytest中fixture的范围

2024-08-28T20:33:52+08:00

pytest作为Python技术栈下最主流的测试框架，功能极为强大和灵活。其中Fixture夹具是它的核心。而且pytest中对Fixture的作用范围也做了不同区分，能为我们利用fixture带来很好地灵活性。

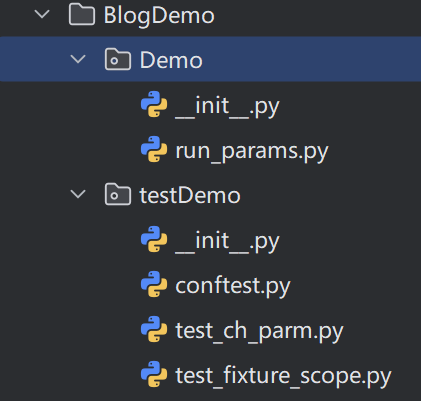
下面我们就来了解下这里不同scope的作用

## fixture的scope定义

首先根据官网的说明，Pytest中fixture的作用范围支持5种设定，分别是function(默认值), classs, module, package, session

| **作用范围** | **说明** |
| --- | --- |
| **function** | 默认值，对每个测试方法（函数）生效，生命周期在测试方法级别 |
| **class** | 对测试类生效，生命周期在测试类级别 |
| **module** | 对测试模块生效，生命周期在模块（文件）级别 |
| **package** | 对测试包生效，生命周期在测试包（目录）级别 |
| **session** | 对测试会话生效，生命周期在会话（一次pytest运行）级别 |

下面结合代码来说明，假设目前有这样的代码结构



run\_params是被测方法

def deal\_params(p):   
 print(f"input :{p}")   
 if type(p) is int:   
 return p\*10   
 if type(p) is str:   
 return p\*3   
 if type(p) in (tuple, list):   
 return "\_".join(p)   
 else:   
 raise TypeError

test\_ch\_param, test\_fixture\_scope中分别定义了参数化和在测试类中的不同测试方法

import pytest  
  
@pytest.mark.parametrize("param",[10, "城下秋草", "软件测试", ("示例", "代码")])   
def test\_params\_mark(param):   
 print(deal\_params(param))

import pytest   
   
class TestFixtureScope1:   
 def test\_int(self):   
 assert deal\_params(2) == 20   
   
 def test\_str(self):   
 assert deal\_params("秋草") == "秋草秋草秋草"   
   
class TestFixtureScope2:   
 def test\_list(self):   
 assert deal\_params(["城下","秋草"]) == "城下\_秋草"   
   
 def test\_dict(self):   
 with pytest.raises(TypeError):   
 deal\_params({"name": "秋草"})

在公共方法文件conftest.py中定义fixture: prepare， 设置了autouse=True，即会根据fixture的设置范围自动应用

@pytest.fixture(autouse=True, scope='function')   
def prepare():   
 print('-----some setup actions.....')   
 yield   
 print('-----some teardown actions!!')

这里我们分别调整prepare的scope为不同取值，然后得到对应的输出

### function

(.venv) C:\Chengxiaqiucao  
pytest  
======================================== test session starts =========================================  
collected 8 items   
  
BlogDemo/testDemo/test\_ch\_parm.py::test\_params\_mark[10] -----some setup actions.....  
input :10  
100  
PASSED-----some teardown actions!!  
  
BlogDemo/testDemo/test\_ch\_parm.py::test\_params\_mark[城下秋草] -----some setup actions.....  
input :城下秋草  
城下秋草城下秋草城下秋草  
PASSED-----some teardown actions!!  
  
BlogDemo/testDemo/test\_ch\_parm.py::test\_params\_mark[软件测试] -----some setup actions.....  
input :软件测试  
软件测试软件测试软件测试  
PASSED-----some teardown actions!!  
  
BlogDemo/testDemo/test\_ch\_parm.py::test\_params\_mark[param3] -----some setup actions.....  
input :('示例', '代码')  
示例\_代码  
PASSED-----some teardown actions!!  
  
BlogDemo/testDemo/test\_fixture\_scope.py::TestFixtureScope1::test\_int -----some setup actions.....  
input :2  
PASSED-----some teardown actions!!  
  
BlogDemo/testDemo/test\_fixture\_scope.py::TestFixtureScope1::test\_str -----some setup actions.....   
input :秋草  
PASSED-----some teardown actions!!  
  
BlogDemo/testDemo/test\_fixture\_scope.py::TestFixtureScope2::test\_list -----some setup actions.....   
input :['城下', '秋草']  
PASSED-----some teardown actions!!  
  
BlogDemo/testDemo/test\_fixture\_scope.py::TestFixtureScope2::test\_dict -----some setup actions.....   
input :{'name': '秋草'}  
PASSED-----some teardown actions!!  
  
  
========================================= 8 passed in 0.27s ==========================================

fixture运行了8次

### class

(.venv) C:\Chengxiaqiucao  
pytest  
======================================== test session starts =========================================  
collected 8 items   
  
BlogDemo/testDemo/test\_ch\_parm.py::test\_params\_mark[10] -----some setup actions.....  
input :10  
100  
PASSED-----some teardown actions!!  
  
BlogDemo/testDemo/test\_ch\_parm.py::test\_params\_mark[城下秋草] -----some setup actions.....  
input :城下秋草  
城下秋草城下秋草城下秋草  
PASSED-----some teardown actions!!  
  
BlogDemo/testDemo/test\_ch\_parm.py::test\_params\_mark[软件测试] -----some setup actions.....  
input :软件测试  
软件测试软件测试软件测试  
PASSED-----some teardown actions!!  
  
BlogDemo/testDemo/test\_ch\_parm.py::test\_params\_mark[param3] -----some setup actions.....  
input :('示例', '代码')  
示例\_代码  
PASSED-----some teardown actions!!  
  
BlogDemo/testDemo/test\_fixture\_scope.py::TestFixtureScope1::test\_int -----some setup actions.....  
input :2  
PASSED  
BlogDemo/testDemo/test\_fixture\_scope.py::TestFixtureScope1::test\_str input :秋草  
PASSED-----some teardown actions!!  
  
BlogDemo/testDemo/test\_fixture\_scope.py::TestFixtureScope2::test\_list -----some setup actions.....   
input :['城下', '秋草']  
PASSED  
BlogDemo/testDemo/test\_fixture\_scope.py::TestFixtureScope2::test\_dict input :{'name': '秋草'}  
PASSED-----some teardown actions!!  
  
  
========================================= 8 passed in 0.27s ==========================================

test\_ch\_param中的测试方法，因为直接定义在文件中，也属于类级别，所以每次赋值参数，fixture也被调用。 而 test\_fixture\_scope中明确定义了两个测试类，所以运行了2次

### module

(.venv) C:\Chengxiaqiucao  
pytest  
======================================== test session starts =========================================  
collected 8 items   
  
BlogDemo/testDemo/test\_ch\_parm.py::test\_params\_mark[10] -----some setup actions.....  
input :10  
100  
PASSED  
BlogDemo/testDemo/test\_ch\_parm.py::test\_params\_mark[城下秋草] input :城下秋草  
城下秋草城下秋草城下秋草  
PASSED  
BlogDemo/testDemo/test\_ch\_parm.py::test\_params\_mark[软件测试] input :软件测试  
软件测试软件测试软件测试  
PASSED  
BlogDemo/testDemo/test\_ch\_parm.py::test\_params\_mark[param3] input :('示例', '代码')  
示例\_代码  
PASSED-----some teardown actions!!  
  
BlogDemo/testDemo/test\_fixture\_scope.py::TestFixtureScope1::test\_int -----some setup actions.....  
input :2  
PASSED  
BlogDemo/testDemo/test\_fixture\_scope.py::TestFixtureScope1::test\_str input :秋草  
PASSED  
BlogDemo/testDemo/test\_fixture\_scope.py::TestFixtureScope2::test\_list input :['城下', '秋草']  
PASSED  
BlogDemo/testDemo/test\_fixture\_scope.py::TestFixtureScope2::test\_dict input :{'name': '秋草'}  
PASSED-----some teardown actions!!  
  
  
========================================= 8 passed in 0.27s ==========================================

修改为module范围后，可以看到，每个模块文件调用了一次fixture

### package

(.venv) C:\Chengxiaqiucao  
pytest  
======================================== test session starts =========================================  
collected 8 items   
  
BlogDemo/testDemo/test\_ch\_parm.py::test\_params\_mark[10] -----some setup actions.....  
input :10  
100  
PASSED  
BlogDemo/testDemo/test\_ch\_parm.py::test\_params\_mark[城下秋草] input :城下秋草  
城下秋草城下秋草城下秋草  
PASSED  
BlogDemo/testDemo/test\_ch\_parm.py::test\_params\_mark[软件测试] input :软件测试  
软件测试软件测试软件测试  
PASSED  
BlogDemo/testDemo/test\_ch\_parm.py::test\_params\_mark[param3] input :('示例', '代码')  
示例\_代码  
PASSED  
BlogDemo/testDemo/test\_fixture\_scope.py::TestFixtureScope1::test\_int input :2  
PASSED  
BlogDemo/testDemo/test\_fixture\_scope.py::TestFixtureScope1::test\_str input :秋草  
PASSED  
BlogDemo/testDemo/test\_fixture\_scope.py::TestFixtureScope2::test\_list input :['城下', '秋草']  
PASSED  
BlogDemo/testDemo/test\_fixture\_scope.py::TestFixtureScope2::test\_dict input :{'name': '秋草'}  
PASSED-----some teardown actions!!  
  
  
========================================= 8 passed in 0.27s ==========================================

修改为package， 这是因为两个测试文件位于同一个package内， 所以运行了一次

### session

(.venv) C:\Chengxiaqiucao  
pytest  
======================================== test session starts =========================================  
collected 8 items   
  
BlogDemo/testDemo/test\_ch\_parm.py::test\_params\_mark[10] -----some setup actions.....  
input :10  
100  
PASSED  
BlogDemo/testDemo/test\_ch\_parm.py::test\_params\_mark[城下秋草] input :城下秋草  
城下秋草城下秋草城下秋草  
PASSED  
BlogDemo/testDemo/test\_ch\_parm.py::test\_params\_mark[软件测试] input :软件测试  
软件测试软件测试软件测试  
PASSED  
BlogDemo/testDemo/test\_ch\_parm.py::test\_params\_mark[param3] input :('示例', '代码')  
示例\_代码  
PASSED  
BlogDemo/testDemo/test\_fixture\_scope.py::TestFixtureScope1::test\_int input :2  
PASSED  
BlogDemo/testDemo/test\_fixture\_scope.py::TestFixtureScope1::test\_str input :秋草  
PASSED  
BlogDemo/testDemo/test\_fixture\_scope.py::TestFixtureScope2::test\_list input :['城下', '秋草']  
PASSED  
BlogDemo/testDemo/test\_fixture\_scope.py::TestFixtureScope2::test\_dict input :{'name': '秋草'}  
PASSED-----some teardown actions!!  
  
  
========================================= 8 passed in 0.27s ==========================================

最后，当设置为session时，也就是运行pytest的一次执行会话，才会触发一次fixture调用

所以可以看到，我们通过fixture的不同scope定义，可以根据需要，来确定我们编写的fixture夹具的作用范围。有很好的灵活性

## 复杂fixture的scope灵活定义

有时在实际使用的时候，特别是我们的一些fixture初始化工作比较复杂但同时在不同作用范围下都可能会用到，这时如果仅仅因为针对不同的作用范围，就要编写多个不同的fixture，代码就显得比较冗余。这时可以怎么处理呢？ 其实可以利用上下文contextmanager来灵活实现

比如我们再编写一个fixture的基本代码上下文:

@contextmanager   
def fixture\_base():   
 print('~~~~~base fixture setup actions.....')   
 yield   
 print('~~~~~base fixture teardown actions!!')

然后针对不同的fixture，我们就可以根据不同的scope来定义不同的fixture并调用这里的context实现。 比如我们再定义一个scope为package的fixture

@pytest.fixture(autouse=False, scope='package')   
def fixture\_module():   
 """   
 对于复杂的fixture但希望灵活处理scope，可以将公共代码放到一个contextmanager中，   
 再针对不同scope定义相关对应fixture   
 """ with fixture\_base() as result:   
 yield result

👍👍这个方法来自pytest的社区总结，[原始问题链接](https://github.com/pytest-dev/pytest/issues/3425)

## 不同scope的执行顺序

上面例子中我们其实看到package和session的执行效果，因为测试方法都在同一个package中，所以效果上没什么差异。但其实不同scope也是有执行顺序的

顺序总结如下：

session > package > module > class > function

这里增加到两个fixture以后，执行的结果：

(.venv) C:\Chengxiaqiucao  
pytest  
======================================== test session starts =========================================  
collected 8 items   
  
BlogDemo/testDemo/test\_ch\_parm.py::test\_params\_mark[10] -----some setup actions.....  
~~~~~base fixture setup actions.....  
input :10  
100  
PASSED  
BlogDemo/testDemo/test\_ch\_parm.py::test\_params\_mark[城下秋草] input :城下秋草  
城下秋草城下秋草城下秋草  
PASSED  
BlogDemo/testDemo/test\_ch\_parm.py::test\_params\_mark[软件测试] input :软件测试  
软件测试软件测试软件测试  
PASSED  
BlogDemo/testDemo/test\_ch\_parm.py::test\_params\_mark[param3] input :('示例', '代码')  
示例\_代码  
PASSED  
BlogDemo/testDemo/test\_fixture\_scope.py::TestFixtureScope1::test\_int input :2  
PASSED  
BlogDemo/testDemo/test\_fixture\_scope.py::TestFixtureScope1::test\_str input :秋草  
PASSED  
BlogDemo/testDemo/test\_fixture\_scope.py::TestFixtureScope2::test\_list input :['城下', '秋草']  
PASSED  
BlogDemo/testDemo/test\_fixture\_scope.py::TestFixtureScope2::test\_dict input :{'name': '秋草'}  
PASSED~~~~~base fixture teardown actions!!  
-----some teardown actions!!  
  
  
========================================= 8 passed in 0.27s ==========================================

可以看到 session 比 package 更早执行，同时更晚被销毁。

那么以上就是关于pytest scope作用范围的总结

推广下我的测试课程，感兴趣的小伙伴可以通过以下链接了解下哦

❤️❤️❤️❤️ [城下秋草的测试职业进阶提升课](https://coding.imooc.com/class/chapter/885.html#Anchor) ❤️❤️❤️❤️