looking for an IT consultant?
aws
• Creating of a chatbot
• Integration with payment providers
• Setting up a CI/CD pipeline
• Help with Serverless stack
• Integration with messaging services
• Building Web scrapers or automation
• Custom development using Python
• Python or cloud coaching

September 6, 2020

#Python | #pytest | #blog

Advanced fixtures with pytest

Other pytest articles:

Now let’s create another test - it will test integration between our 2 components that talk to external systems - API and database cache. Let’s test that when we query a number twice - we call API only once that the result is saved to the database and fetched from it on the second call.

I'm going to use responses for mocking API and in memory sqlite for database, So I will have to fixtures passed - sessions and responses patch. I'm going to use some number for that test. First I'm going to check if the number is not in the database cache yet. Then there are going to be two call to some function we are yet to write. After those two calls I'll check that the number is in the database. And also I'll check that API was called only once.

Copy

```
import pytest
from database import CacheService
from number_checker import check_number

@pytest.mark.usefixtures("setup_db") # 1
def test_integration_external_calls(session, api_response): # 2
    number = "+311234555"
    cache = CacheService(session)
    assert not cache.get_status(number)


    check_number(number, cache)
    check_number(number, cache)

    assert cache.get_status(number)
    assert len(api_response.calls) == 1
```

1. Use setup DB fixture automatically
2. Some test case to check whole app

Now a number of our tests use this setup_db fixture. We can modify it so pytest will invoke it automatically

Copy

```
@pytest.fixture(autouse=True) # 1
def setup_db(session):
    session.execute('''CREATE TABLE numbers
                          (number text, existing boolean)''')
    session.execute('INSERT INTO numbers VALUES ("+3155512345", 1)')
    session.connection.commit()
```

1. Use this fixture for every test case

Now we can get rid of `@pytest.mark.usefixtures("setup_db")`

But pytest will invoke for every test case, which is a waste of time if we use this fixture to load some data in database. We can control the scope of the session with the scope parameter.

With this version we invoke create_db only once per testing session:

Copy

```
@pytest.fixture(autouse=True, scope='session')
def setup_db(session):
    ...
```

In that example session fixture needs to have a matching scope:

Copy

```
@pytest.fixture(scope='session') # 1
def session():
    ...
```

1. This fixture will be invoked once per pytest run

Possible values for scope are: function, class, module, package or session. Session - once per pytest run package, module (in our case test_*.py file) or class (test in pytest can be grouped in a class) - once in the corresponding unit function - default, get new session for each test_* function.

Fixtures can be also parametrized. For example, we have a sample database already in data.db file. We can create 2 session fixtures like so:

Copy

```
@pytest.fixture(scope='session', params=[':memory:', 'data.db']) # 1
def session(request):
    print("using db "+request.param)
    connection = sqlite3.connect(request.param)
    db_session = connection.cursor()
    yield db_session
    connection.close()
```

1. This will produce 2 test cases for each parameter

We provide list of parameters in the decorator, pass a spacial `request` parameter to the fixture and access the data with `request.param`.

Now when I run pytest it gives generated 2 test cases

Copy

```
~ pytest test_integration.py -vs
=== test session starts ================================================
collected 2 items

test_integration.py::test_integration_external_calls[:memory:] using db :memory:
PASSED
test_integration.py::test_integration_external_calls[data.db] using db data.db
PASSED

=== 2 passed in 0.06s ===
```

In this unit you've learned advanced usages of pytest fixtures, in the next we'll cover pytest plugins.

Similar articles:

- Hello, World!
- Pytest plugins
- Selecting tests with pytest
- Test driven Development
- Testing HTTP client with pytest