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

Pytest plugins

Other pytest articles:

Why testing is important

Types of tests

Test driven Development

Hello, World!

Selecting tests with pytest

Testing HTTP client with pytest

Testing database with pytest

Advanced fixtures with pytest

Pytest plugins

There are lot of plugins in pytest ecosystem. Some of the widely used are listed here All the plugins can be installed with pip and invoked by providing an argument to pytest executable.

pytest-cov

This plugin calculates test coverage - how much of our code is covered by test. Since every project is unique it's difficult to tell an universally acceptable coverage (https://www.artima.com/forums/flat.jsp?forum=106&thread=204677)

If we have all out source files in `source/` folder and test in `test/` then we can run pytest with this plugin by

Copy

```
pytest --cov="source" --cov-report=term-missing test/
```

cov-report argument will determine what report to generate. `term-missing` report will tell exactly which lines are not covered by tests.

The output report looks like this

Copy

```
--- coverage: python 3.6.2-final-0 ---
Name                        Stmts   Miss  Cover   Missing
------------------------------------------------
source/__init__.py              0      0   100%
source/api.py                  12      4    67%   10-11, 15-16
source/database.py             15      0   100%
source/exceptions.py            2      0   100%
source/nnormalize.py           14      4    71%   6-9
source/number_checker.py        8      0   100%
------------------------------------------------
TOTAL                          51      8    84%
```

pytest-xdist

This plugin will parallelize test execution. It will speed up executing of large test sets.

Copy

```
pytest -n 5 test/
```

Where 5 is a number of workers. It can also accept `auto` as a parameter. The pytest output is not changed when using this plugin (except some hint that it's running multiple processes)

pytest-bdd

This is a less technical plugin of the three I wanted to cover. This one allows to convert feature files written in Gherkin format to pytest cases.

We start working with pytest-bdd by creating a feature file in a Gherkin format. This is a highly readable form to describe a feature or a user story and to fix business-side requirements.

Copy

```
Feature: Phone number validation application

    Scenario: validate a number
        Given I have a number +311234555
        When I run number validation application
        Then Number is normalized
        And Cache is checked for presence of this number
        And External API is called if cache doesn't contains the number
```

With this file we can generate our test case:

Copy

```
pytest-bdd generate validation.feature >> test_number_validation.py
```

Test file appears to be pretty long, but here is some main points:

- there is a functions decorated with `given`, `when`, `then` decorators from pytest-bdd like

Copy

```
@given('I have a number +311234555')
def i_have_a_number_311234555():
    """I have a number +311234555."""
    raise NotImplementedError
```

- the decorator input string is what we have written in a feature file
- functions decorated with @given are recognized as fixtures, so they can be passed to a last function, decorated with `@then`

Copy

```
@given('I have a number +311234555')
def i_have_a_number_311234555():
    """I have a number +311234555."""
    return '+311234555'
```

```
@when('I run number validation application')
def i_run_number_validation_application(i_have_a_number_311234555):
    # do stuff and use
    # i_have_a_number_311234555 holding out test number
    raise NotImplementedError
```

This might look like an unnecessary extra step, but some organizations find it useful to fix business requirements.

This is the last unit in this course. Thanks for staying with me and I hope that you find this content valuable.

Similar articles:

- Advanced fixtures with pytest
- Hello, World!
- Selecting tests with pytest
- Test driven Development
- Testing HTTP client with pytest