Testing tool (pytest)

pytest and pytest-django seamlessly integrate with standard Django test suites as well as Nose test suites, providing a convenient and efficient testing framework for your Django project. Their compatibility allows you to effortlessly pick up and run existing tests with minimal configuration. In this section, we'll guide you through a quick and easy setup to get started with pytest and pytest-django in your Django project.

Features:

- Detailed info on failing assert **statements** (no need to remember self.assert* names)
- Auto-discovery of test modules and functions
- Modular fixtures for managing small or parametrized long-lived test resources
- Can run unittest (including trial)
- Python 3.8+ or PyPy 3
- Rich plugin architecture, with over 800+ external plugins and thriving community

Installation:

Installing pytest-django will also automatically install the latest version of pytest. pytest-django uses pytest's plugin system and can be used right away after installation there is nothing more to configure.

Poweshell >>

```
pip install pytest

pip install pytest pytest-django
```

Project Structure:

Organize project's test files in a way that follows the Django conventions. Typically, tests are placed in the "tests" *directory within each Django app.*

```
MY_project/
        MY_app/
              tests/
                   test_models.py
                   test_views.py
```

Running Tests:

When using pytest-django, django-admin.py or manage.py is not used to run tests. This makes it possible to invoke pytest and other plugins with all its different options directly.

Use the following command to run tests:

```
Pytest
```

Database Handling:

pytest-django provides useful fixtures for handling database transactions in tests. Utilize **django_db_setup** and **django_db_blocker** fixtures for effective database testing:

```
# test_views.py


import pytest

from your_app.views import YourView

@pytest.mark.django_db

def test_view_function(django_db_setup, django_db_blocker):

    # Your test logic here
```

The `pytest` framework makes it easy to write small, readable tests, and can scale to support complex functional testing for applications and libraries.