Let's walk through creating a simple Django project and setting up PyTest for testing step by step. The project will be a basic web application with a single view that returns a "Hello, World!" message.

### 1. \*\*Set Up Your Django Project\*\*

1. \*\*Create a Virtual Environment\*\*:

- It's good practice to create a virtual environment for your project to manage dependencies.

```bash

python -m venv myenv

source myenv/bin/activate # On Windows, use `myenv\Scripts\activate`

```

2. \*\*Install Django\*\*:

- Install Django in your virtual environment.

```bash

pip install django

```

3. \*\*Create a Django Project\*\*:

- Create a new Django project named `simple\_project`.

```bash

django-admin startproject simple\_project

cd simple\_project

```

4. \*\*Create a Django App\*\*:

- Create a new app within your project called `hello`.

```bash

python manage.py startapp hello

```

### 2. \*\*Create a Simple View\*\*

1. \*\*Define the View\*\*:

- In `hello/views.py`, create a simple view that returns a "Hello, World!" message.

```python

from django.http import HttpResponse

def hello\_world(request):

return HttpResponse("Hello, World!")

```

2. \*\*Set Up URLs\*\*:

- In `hello/urls.py`, set up the URL configuration for the `hello\_world` view.

```python

from django.urls import path

from . import views

urlpatterns = [

path('', views.hello\_world, name='hello\_world'),

]

```

- Include the `hello` app URLs in the main `urls.py` of your project:

```python

from django.contrib import admin

from django.urls import include, path

urlpatterns = [

path('admin/', admin.site.urls),

path('hello/', include('hello.urls')),

]

```

3. \*\*Run the Server\*\*:

- Start the Django development server to ensure everything is working.

```bash

python manage.py runserver

```

- Visit `http://127.0.0.1:8000/hello/` in your browser. You should see "Hello, World!" displayed.

### 3. \*\*Set Up PyTest\*\*

1. \*\*Install PyTest\*\*:

- Install PyTest and the Django plugin for PyTest.

```bash

pip install pytest pytest-django

```

2. \*\*Configure PyTest\*\*:

- Create a `pytest.ini` file in the root of your project (same directory as `manage.py`).

```ini

[pytest]

DJANGO\_SETTINGS\_MODULE = simple\_project.settings

python\_files = tests.py test\_\*.py \*\_tests.py

```

### 4. \*\*Write Test Cases Using PyTest\*\*

1. \*\*Create a Test File\*\*:

- In the `hello` app directory, create a file named `test\_views.py`.

```bash

touch hello/test\_views.py

```

2. \*\*Write a Simple Test\*\*:

- In `hello/test\_views.py`, write a test for the `hello\_world` view.

```python

import pytest

from django.urls import reverse

@pytest.mark.django\_db

def test\_hello\_world(client):

url = reverse('hello\_world')

response = client.get(url)

assert response.status\_code == 200

assert response.content == b"Hello, World!"

```

### 5. \*\*Run the Tests\*\*

1. \*\*Run PyTest\*\*:

- Run the tests using PyTest.

```bash

pytest

```

2. \*\*Review the Results\*\*:

- PyTest will execute your test and display the results in the terminal. You should see that the test passed if everything was set up correctly.

### Summary

- \*\*Project Structure\*\*: You created a basic Django project with an app that serves a "Hello, World!" response.

- \*\*PyTest Setup\*\*: You set up PyTest for testing your Django views.

- \*\*Writing Tests\*\*: You wrote a simple test to ensure your view returns the correct response.

- \*\*Running Tests\*\*: Finally, you ran the tests and verified that they passed.

This guide gives you a foundational understanding of how to set up and test a Django project using PyTest.