Введение в Django (часть 2)

Создание приложения.

Когда вы создаете новое приложение в Django автоматически создаются несколько файлов. Каждый из них выполняет определенные функции в рамках работы вашего приложения.

__init__.py

Это пустой файл, который делает папку приложения пакетом Python. Благодаря этому файлу, Python воспринимает папку как модуль, и вы можете импортировать ее и другие файлы приложения в проекте.

admin.py

Этот файл отвечает за регистрацию моделей вашего приложения в Django Admin. В этом файле вы можете указать, какие модели будут доступны через административную панель Django, а также настроить их отображение.

```
from django.contrib import admin
from .models import Product

admin.site.register(Product)
```

apps.py

Этот файл содержит конфигурацию вашего приложения. Здесь указывается название приложения и другие параметры, такие как пути к файлам. Когда вы регистрируете приложение в файле settings.py основного проекта, Django использует этот файл для настройки приложения.

```
from django.apps import AppConfig

class ProductsConfig(AppConfig):
    default_auto_field = 'django.db.models.BigAutoField'
    name = 'products'
```

models.py

Это один из самых важных файлов. В нем вы определяете модели, которые представляют структуры данных вашего приложения (например, товары, категории). Модели в Django представляют собой таблицы базы данных, и каждый класс в models.py — это отдельная таблица с полями, соответствующими колонкам.

```
from django.db import models

class Product(models.Model):
    name = models.CharField(max_length=100)
    price = models.DecimalField(max_digits=10, decimal_places=2)
    description = models.TextField()

    def __str__(self):
        return self.name
```

tests.py

Здесь создаются тесты для вашего приложения. Это место, где вы пишете юнит-тесты, чтобы проверить корректность работы кода. Django поддерживает встроенную систему тестирования, и файл tests.py используется для написания тестов, которые проверяют, что ваши модели, представления (views) и формы работают правильно.

```
from django.test import TestCase
from .models import Product

class ProductTestCase(TestCase):
    def setUp(self):
        Product.objects.create(name="Test product", price=100.00)

    def test_product_name(self):
        product = Product.objects.get(name="Test product")
        self.assertEqual(product.name, "Test product")
```

views.py

Этот файл отвечает за логику отображения. Здесь вы создаете функции или классы, которые обрабатывают HTTP-запросы и возвращают HTTP-ответы (например, HTML-страницы, JSON-данные). Функции в views.py связывают модели с шаблонами и управляют отображением данных.

```
from django.shortcuts import render
from .models import Product

def product_list(request):
    products = Product.objects.all()
    return render(request, 'product_list.html', {'products': products})
```

migrations/

Папка migrations содержит файлы миграций. Миграции — это способ Django отслеживать изменения в моделях и синхронизировать их с базой данных. Когда вы создаете или изменяете модель в models.py, вы генерируете миграции с помощью команды python manage.py makemigrations, и они сохраняются в этой папке. Миграции помогают управлять изменениями в структуре базы данных без потери данных.

Назначение файлов:

- __init__.py — делает приложение модулем Python.
- admin.py — настраивает административную панель.
- apps.py — хранит конфигурацию приложения.
- models.py — описывает структуру данных (модели) приложения.
- tests.py — содержит тесты для проверки работоспособности приложения.
- views.py — управляет обработкой запросов и выводом данных.
- migrations/ — хранит файлы миграций для управления изменениями в БД.