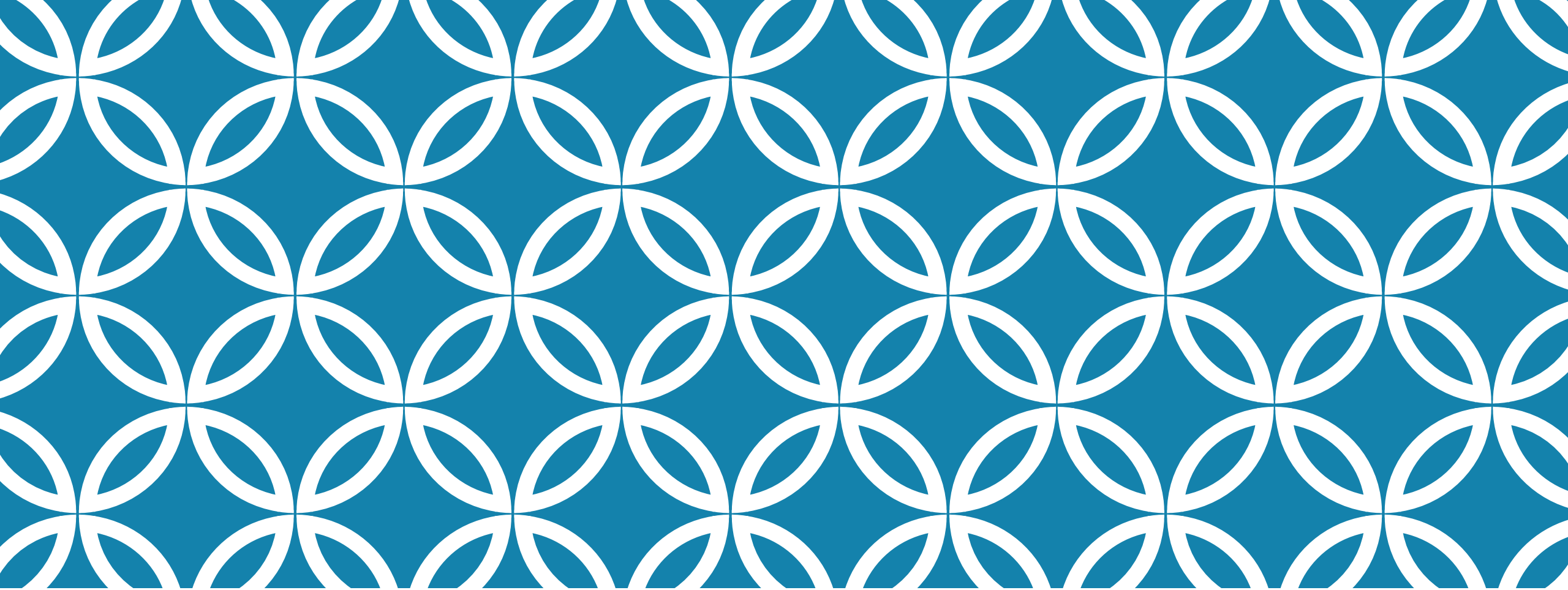


СОЗДАНИЕ ПЕРВОГО ПРОЕКТА

ПРИЛОЖЕНИЕ ДЛЯ ВЕДЕНИЯ БЛОГА

- Проект Django – приложения для ведения блога
- Создание модели данных
- Их синхронизация с базой данных

Создание изначальной файловой структуры проекта

```
django-admin startproject mysite
```

ПРИЛОЖЕНИЕ ДЛЯ ВЕДЕНИЯ БЛОГА

Сгенерированная структура проекта

```
mysite/
    manage.py
    mysite/
        __init__.py
        asgi.py
        settings.py
        urls.py
        wsgi.py
```

ПРИЛОЖЕНИЕ ДЛЯ ВЕДЕНИЯ БЛОГА

Внешний каталог mysite/ контейнером проекта. Он содержит файлы:

- **manage.py**: это утилита командной строки, используемая для взаимодействия с проектом. Редактировать этот файл не требуется;

- **__init__.py**: пустой файл, который сообщает Python, что каталог **mysite** нужно трактовать как модуль Python;

– **asgi.py**: конфигурация для выполнения проекта в качестве приложения, работающего по протоколу интерфейса шлюза асинхронного сервера (ASGI) с ASGI-совместимыми веб-серверами. ASGI – это новый стандарт Python для асинхронных веб-серверов и приложений;

– **settings.py**: здесь указаны настроечные параметры и конфигурация проекта и содержатся изначальные параметры со значениями, используемыми по умолчанию;

– **urls.py**: место, где располагаются ваши шаблоны URL-адресов. Каждый URL-адрес, который определен здесь, соотносится с представлением;

– **wsgi.py**: конфигурация для выполнения проекта в качестве приложения, работающего по протоколу интерфейса шлюза веб-сервера (WSGI) 1 с WSGI-совместимыми веб-серверами.

ПРИМЕНЕНИЕ ПЕРВОНАЧАЛЬНЫХ МИГРАЦИЙ БАЗЫ ДАННЫХ

- файл **settings.py** содержит конфигурацию базы данных проекта в настроечном параметре **DATABASES;**

- settings.py также содержит настроечный параметр **INSTALLED_APPS** со списком, содержащим распространенные приложения Django, которые добавляются в проект по умолчанию;

- изначально конфигурацией предусматривается использование базы данных SQLite3. SQLite – это облегченная база данных, которую можно использовать с Django

(*Если планируете развернуть свое приложение в производственной среде, то следует использовать полнофункциональную базу данных, такую как PostgreSQL, MySQL или Oracle)*

ПРИМЕНЕНИЕ ПЕРВОНАЧАЛЬНЫХ МИГРАЦИЙ БАЗЫ ДАННЫХ

- для того чтобы завершить настройку проекта, необходимо создать таблицы, ассоциированные с моделями стандартных приложений Django, включенных в состав параметра INSTALLED_APPS

- выполните следующие ниже команды:

```
cd mysite
python manage.py migrate
```

ПРИМЕНЕНИЕ ПЕРВОНАЧАЛЬНЫХ МИГРАЦИЙ БАЗЫ ДАННЫХ

- результат:

```
Applying contenttypes.0001_initial... OK
Applying auth.0001_initial... OK
Applying admin.0001_initial... OK
Applying admin.0002_logentry_remove_auto_add... OK
Applying admin.0003_logentry_add_action_flag_choices... OK
Applying contenttypes.0002_remove_content_type_name... OK
Applying auth.0002_alter_permission_name_max_length... OK
Applying auth.0003_alter_user_email_max_length... OK
Applying auth.0004_alter_user_username_opts... OK
Applying auth.0005_alter_user_last_login_null... OK
Applying auth.0006_require_contenttypes_0002... OK
Applying auth.0007_alter_validators_add_error_messages... OK
Applying auth.0008_alter_user_username_max_length... OK
Applying auth.0009_alter_user_last_name_max_length... OK
Applying auth.0010_alter_group_name_max_length... OK
Applying auth.0011_update_proxy_permissions... OK
Applying auth.0012_alter_user_first_name_max_length... OK
Applying sessions.0001_initial... OK
```

ПРИМЕНЕНИЕ ПЕРВОНАЧАЛЬНЫХ МИГРАЦИЙ БАЗЫ ДАННЫХ

- результат:

```
Applying contenttypes.0001_initial... OK
Applying auth.0001_initial... OK
Applying admin.0001_initial... OK
Applying admin.0002_logentry_remove_auto_add... OK
Applying admin.0003_logentry_add_action_flag_choices... OK
Applying contenttypes.0002_remove_content_type_name... OK
Applying auth.0002_alter_permission_name_max_length... OK
Applying auth.0003_alter_user_email_max_length... OK
Applying auth.0004_alter_user_username_opts... OK
Applying auth.0005_alter_user_last_login_null... OK
Applying auth.0006_require_contenttypes_0002... OK
Applying auth.0007_alter_validators_add_error_messages... OK
Applying auth.0008_alter_user_username_max_length... OK
Applying auth.0009_alter_user_last_name_max_length... OK
Applying auth.0010_alter_group_name_max_length... OK
Applying auth.0011_update_proxy_permissions... OK
Applying auth.0012_alter_user_first_name_max_length... OK
Applying sessions.0001_initial... OK
```

применяемые веб-фреймворком Django миграции базы данных

ПРИМЕНЕНИЕ ПЕРВОНАЧАЛЬНЫХ МИГРАЦИЙ БАЗЫ ДАННЫХ

- результат:

```
Applying contenttypes.0001_initial... OK
Applying auth.0001_initial... OK
Applying admin.0001_initial... OK
Applying admin.0002_logentry_remove_auto_add... OK
Applying admin.0003_logentry_add_action_flag_choices... OK
Applying contenttypes.0002_remove_content_type_name... OK
Applying auth.0002_alter_permission_name_max_length... OK
Applying auth.0003_alter_user_email_max_length... OK
Applying auth.0004_alter_user_username_opts... OK
Applying auth.0005_alter_user_last_login_null... OK
Applying auth.0006_require_contenttypes_0002... OK
Applying auth.0007_alter_validators_add_error_messages... OK
Applying auth.0008_alter_user_username_max_length... OK
Applying auth.0009_alter_user_last_name_max_length... OK
Applying auth.0010_alter_group_name_max_length... OK
Applying auth.0011_update_proxy_permissions... OK
Applying auth.0012_alter_user_first_name_max_length... OK
Applying sessions.0001_initial... OK
```

применяемые веб-фреймворком Django миграции базы данных

ЗАПУСК И ВЫПОЛНЕНИЕ СЕРВЕРА РАЗРАБОТКИ

- Django идет в комплекте вместе с облегченным веб-сервером с целью быстрого выполнения вашего исходного кода без необходимости тратить время на настройку производственного сервера

Запуск сервера разработки

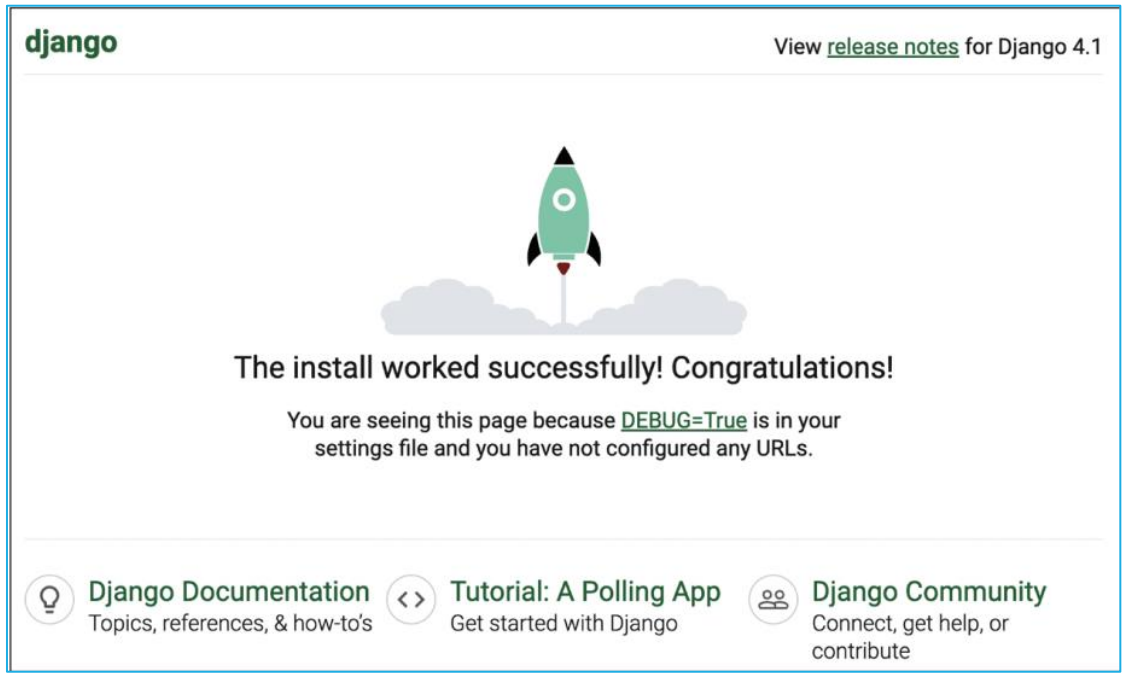
```
python manage.py runserver
```

Результат

```
Watching for file changes with StatReloader
Performing system checks...
System check identified no issues (0 silenced).
January 01, 2022 - 10:00:00
Django version 4.0, using settings 'mysite.settings'
Starting development server at http://127.0.0.1:8000/
Quit the server with CONTROL-C.
```

ЗАПУСК И ВЫПОЛНЕНИЕ СЕРВЕРА РАЗРАБОТКИ

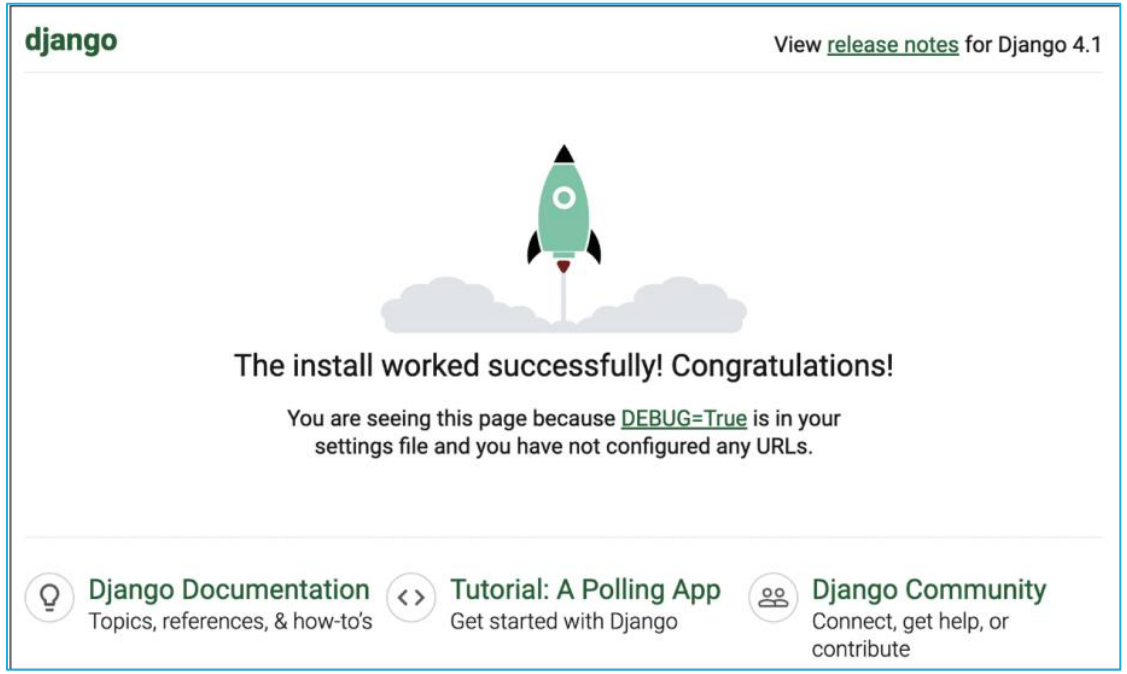
- пройти по URL-адресу http://127.0.0.1:8000/ в своем браузере



Домашняя страница сервера разработки

ЗАПУСК И ВЫПОЛНЕНИЕ СЕРВЕРА РАЗРАБОТКИ

- пройти по URL-адресу http://127.0.0.1:8000/ в своем браузере



Домашняя страница сервера разработки

Каждый HTTP-запрос регистрируется в консоли сервером разработки. Любая ошибка, возникающая во время работы сервера разработки, также будет появляться в консоли

```
[01/Jan/2022 17:20:30] "GET / HTTP/1.1" 200 16351
```

ЗАПУСК И ВЫПОЛНЕНИЕ СЕРВЕРА РАЗРАБОТКИ

```
python manage.py runserver 127.0.0.1:8001 --settings=mysite.settings
```

этот сервер предназначен только для разработки и не подходит для производственного использования.

для того чтобы развернуть Django в производственной среде, необходимо его выполнять как приложение на основе WSGI с использованием такого веб-сервера, как

Apache,

Gunicorn,

uWSGI, или

же как приложение на основе ASGI с использованием такого сервера, как Daphne или Uvicorn.

НАСТРОЕЧНЫЕ ПАРАМЕТРЫ ПРОЕКТА

Открыть файл **settings.py** и взглянуть на конфигурацию проекта. Несколько настроечных параметров уже внесены в указанный файл веб-фреймворком Django, но это лишь часть всех имеющихся параметров:

• **DEBUG** – это булев параметр, который включает и выключает режим отладки проекта. Если его значение установлено равным True, то Django будет отображать подробные страницы ошибок в случаях, когда приложение выдает неперехваченное исключение. При переходе в производственную среду следует помнить о том, что необходимо устанавливать его значение равным False. Никогда не развертывайте свой сайт в производственной среде с включенной отладкой, поскольку вы предоставите конфиденциальные данные, связанные с проектом.

НАСТРОЕЧНЫЕ ПАРАМЕТРЫ ПРОЕКТА

• **ALLOWED_HOSTS** не применяется при включенном режиме отладки или при выполнении тестов. При перенесении своего сайта в производственную среду и установке параметра DEBUG равным False в этот настроечный параметр следует добавлять свои домен/хост, чтобы разрешить ему раздавать ваш сайт Django.

• **INSTALLED_APPS** – это параметр, который придется редактировать во всех проектах. Он сообщает Django о приложениях, которые для этого сайта являются активными.

НАСТРОЕЧНЫЕ ПАРАМЕТРЫ ПРОЕКТА

По умолчанию Django вставляет следующие ниже приложения:

– django.contrib.admin: сайт администрирования;

– django.contrib.auth: фреймворк аутентификации;

– django.contrib.contenttypes: фреймворк типов контента;

– django.contrib.sessions: фреймворк сеансов;

– django.contrib.messages: фреймворк сообщений;

– django.contrib.staticfiles: фреймворк управления статическими файлами.

НАСТРОЕЧНЫЕ ПАРАМЕТРЫ ПРОЕКТА

• MIDDLEWARE – подлежащие исполнению промежуточные программные компоненты.

• ROOT_URLCONF указывает модуль Python, в котором определены шаблоны корневых URL-адресов приложения.

• DATABASES – словарь, содержащий настроечные параметры всех баз данных, которые будут использоваться в проекте. Всегда должна существовать база данных, которая будет использоваться по умолчанию. В стандартной конфигурации используется база данных SQLite3, если не указана иная.

• LANGUAGE_CODE определяет заранее заданный языковой код этого сайта Django.

• USE_TZ сообщает Django, что нужно активировать/деактивировать поддержку часовых поясов. Django поставляется вместе с поддержкой дат и времен с учетом часовых поясов. Этот настроечный параметр получает значение True при создании нового проекта с помощью команды управления startproject.

СОЗДАНИЕ ПРИЛОЖЕНИЯ

Выполнить команду

```
python manage.py startapp blog
```

Она создаст базовую структуру приложения

```
blog/
    __init__.py
    admin.py
    apps.py
    migrations/
        __init__.py
    models.py
    tests.py
    views.py
```

СОЗДАНИЕ ПРИЛОЖЕНИЯ

Описание этих файлов:

• **__init__.py**: пустой файл, который сообщает Python, что каталог blog нужно трактовать как модуль Python;

• **admin.py**: здесь вы регистрируете модели, чтобы включать их в состав сайта администрирования – этот сайт используется опционально, по вашему выбору;
• apps.py: содержит главную конфигурацию приложения blog;

• **migrations**: этот каталог будет содержать миграции базы данных приложения. Миграции позволяют Django отслеживать изменения модели и соответствующим образом синхронизировать базу данных. Указанный каталог содержит пустой файл __init__.py;

СОЗДАНИЕ ПРИЛОЖЕНИЯ

Описание этих файлов:

• **models.py**: содержит относимые к приложению модели данных; все приложения Django должны иметь файл models.py, но его можно оставлять пустым;

• **tests.py**: здесь можно добавлять относимые к приложению тесты;

• **views.py**: здесь расположена логика приложения; каждое представление получает HTTP-запрос, обрабатывает его и возвращает ответ.

СОЗДАНИЕ МОДЕЛЕЙ БЛОГА

Объект Python – это набор данных и методов.

Класс – это концептуальная схема, которая объединяет данные и функциональности в единое целое.

Создание нового класса влечет новый тип объекта, позволяя формировать экземпляры этого типа.

Модель Django – это источник информации и поведения данных. Она состоит из класса Python, который является подклассом **django.db.models.Model.**

СОЗДАНИЕ МОДЕЛЕЙ БЛОГА

Каждой модели ставится в соответствие одна таблица базы данных, где каждый атрибут класса соотносится с полем базы данных.

Когда вы будете создавать модель, Django будет предоставлять практичный API, чтобы легко запрашивать объекты в базе данных.

Сначала мы определим модели баз данных для приложения blog. Затем сгенерируем для этих моделей миграции базы данных, чтобы создать соответствующие таблицы базы данных.

При применении миграций Django будет создавать таблицу по каждой модели, определенной в файле **models. py** приложения

СОЗДАНИЕ МОДЕЛЕЙ БЛОГА

Каждой модели ставится в соответствие одна таблица базы данных, где каждый атрибут класса соотносится с полем базы данных.

Когда вы будете создавать модель, Django будет предоставлять практичный API, чтобы легко запрашивать объекты в базе данных.

Сначала мы определим модели баз данных для приложения blog. Затем сгенерируем для этих моделей миграции базы данных, чтобы создать соответствующие таблицы базы данных.

При применении миграций Django будет создавать таблицу по каждой модели, определенной в файле **models. py** приложения

СОЗДАНИЕ МОДЕЛЕЙ ПОСТА

Определить модель Post, которая позволит хранить посты блога в базе данных.

Добавить следующие ниже строки в файл **models.py** приложения **blog**.

Новые строки выделены жирным шрифтом:

```
from django.db import models

class Post(models.Model):
    title = models.CharField(max_length=250)
    slug = models.SlugField(max_length=250)
    body = models.TextField()
    def __str__(self):
        return self.title
```

СОЗДАНИЕ МОДЕЛЕЙ ПОСТА

Это модель данных для постов блога. Посты будут иметь заголовок, короткую метку под названием slug и тело поста. Поля указанной модели:

• title: поле заголовка поста. Это поле с типом CharField, которое транслируется в столбец VARCHAR в базе данных SQL;

• slug: поле SlugField, которое транслируется в столбец VARCHAR в базе данных SQL.

Слаг – это короткая метка, содержащая только буквы, цифры, знаки подчеркивания или дефисы. Пост с заголовком «*Django Reinhardt: A legend of Jazz*» мог бы содержать такой заголовок: «django-reinhardtlegend-jazz».

СОЗДАНИЕ МОДЕЛЕЙ ПОСТА

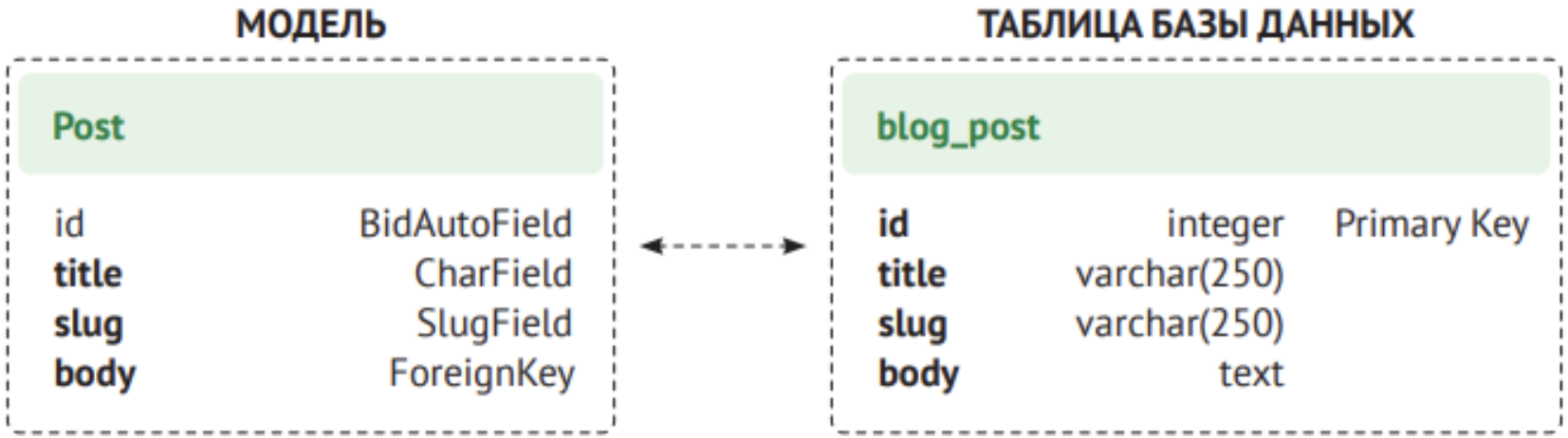
• body: поле для хранения тела поста. Это поле с типом TextField, которое транслируется в столбец Text в базе данных SQL.

В модельный класс также добавлен метод __str__(). Это метод Python, который применяется по умолчанию и возвращает строковый литерал с удобочитаемым представлением объекта.

Django будет использовать этот метод для отображения имени объекта во многих местах, таких как его сайт администрирования.

СОЗДАНИЕ МОДЕЛЕЙ ПОСТА

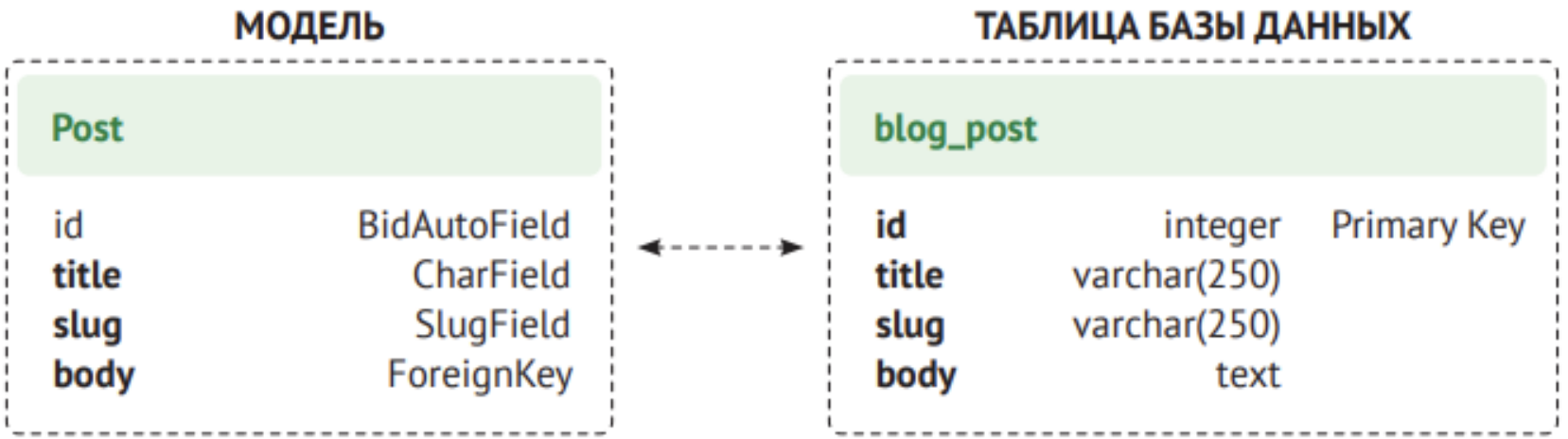
модель Post и соответствующая таблица базы данных, которую Django создаст, когда мы синхронизируем модель с базой данных



Соответствие изначальной модели Post и таблицы базы данных

СОЗДАНИЕ МОДЕЛЕЙ ПОСТА

Мы расширим модель Post дополнительными полями и поведением. После завершения мы синхронизируем ее с базой данных, создав миграцию в базе данных и применив ее



Соответствие изначальной модели Post и таблицы базы данных

Мы расширим модель Post дополнительными полями и поведением. После завершения мы синхронизируем ее с базой данных, создав миграцию в базе данных и применив ее