Создаём своё первое приложение с Django, часть 1

Давайте учиться на примере.

Мы создадим проект, состоящий из простого приложения для голосования.

Проект будет состоять из двух частей:

- Публичный сайт, который позволяет людям видеть голосования и иголосовать.
- Интерфейс администратора, который позволяет вам добавлять, изменять и удалять голосования.

Мы предполагаем, что вы уже установили Django. Вы можете проверить это, проверив какая версия установлена, выполнив следующую команду в консоли (обозначена префиксом $):

```
$ python -m django --version
```

Если Django установлен, вы должны увидеть текущую версию. Иначе получите ошибку «No module named django».

Учебник написан для Django 3.0, которая поддерживает Python 3.6 и выше. Если версия Django отличается, вы можете обратиться к документации, соответствующей версии Django, или обновить Django до последней версии. Если вы все еще используете старую версию Python, обратитесь к Какие версии Python можно использовать с Django?, чтобы найти совместимую версию Django.

Смотрите Как установить Django, чтобы узнать как удалить старую версию Django и установить новую.

Где искать помощь:

При наличии проблем с данной инструкцией, пожалуйста, обратитесь к разделу FAQ Получение помощи.

Создание проекта

Если вы используете Django первый раз, вам придется позаботиться о некоторых первоначальных настройках. А именно, сгенерировать основу проекта Django – настройки проекта, базы данных, приложений и др.

Используя командную строку, перейдите (`cd`) в каталог, где вы хотите хранить код, и выполните следующую команду:

```
$ django-admin startproject mysite
```

Это создаст каталог `mysite` в текущем каталоге. Если нет, смотрите Трудности с запуском django-admin.

Примечание

Вы не должны использовать в качестве названия проекта названия компонентов Python или Django. Это означает, что проект не может называться `django` (что конфликтует с Django) или `test` (конфликтует со стандартным пакетом Python).

Где разместить этот код?

Если вы раньше использовали PHP (без использования современных фреймворков), то, наверное, привыкли размещать код проекта в корневом каталоге сайта на Web-сервере (например, `/var/www`). С Django вы не должны этого делать. Это плохая идея добавлять код проекта в корень Web-сервера, так как есть риск, что он будет доступен для просмотра. Не делайте этого в целях безопасности.

Разместите код в каталоге **вне** корневой директории сайта, например `/home/mycode`.

Давайте посмотрим, что было создано при помощи команды `startproject`:

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


Предыдущий раздел

Быстрое руководство по установке

Следующий раздел

Создаём своё первое приложение с Django, часть 2

Эта страница

- Исходный текст

Последнее обновление:

нояб. 23, 2021

Рассмотрим эти файлы:

- Внешний каталог `mysite/` – это просто контейнер для вашего проекта. Его название никак не используется Django и вы можете переименовать его во что угодно.
- `manage.py`: Скрипт, который позволяет вам взаимодействовать с проектом Django. Подробности о `manage.py` читайте в разделе django-admin and manage.py.
- Внутренний каталог `mysite/` - это пакет Python вашего проекта. Его название – это название пакета Python, которое вы будете использовать для импорта чего-либо из проекта (например, `mysite.urls`).
- `mysite/__init__.py`: Пустой файл, который указывает Python, что текущий каталог является пакетом Python. (Если вы новичок в Python, читайте о покетах в официальной документации Python.)
- `mysite/settings.py`: Настройки/конфигурация проекта. Раздел Настройки Django расскажет вам все о настройках проекта.
- `mysite/urls.py`: Конфигурация URL-ов для вашего проекта Django. Это «содержание» всех Django-сайтов. Вы можете прочитать о конфигурации URL-ов в разделе Менеджер URL-ов.
- `mysite/wsgi.py`: Точка входа вашего проекта для ASGI-совместимых веб-серверов. Подробности читайте в разделе How to deploy with ASGI.
- `mysite/wsgi.py`: Точка входа вашего проекта для WSGI-совместимых веб-серверов. Подробности читайте в разделе Развёртывание с WSGI.

Сервер для разработки

Давайте проверим, что все заработало. Перейдите во внешний каталог `mysite`, если вы этого еще не сделали, и выполните команду:

```
$ python manage.py runserver
```

Вы увидите следующий вывод:

```
Performing system checks...

System check identified no issues (0 silenced).

You have unapplied migrations; your app may not work properly until they are applied.
Run 'python manage.py migrate' to apply them.

ноября 23, 2021 - 15:50:53
Django version 3.0, using settings 'mysite.settings'
Starting development server at http://127.0.0.1:8000/
Quit the server with CONTROL-C.
```

Примечание

Игнорируйте предупреждения о невыполненных миграциях, мы разберёмся с базой данных чуть позже.

Только что вы запустили сервер для разработки Django, простой Web-сервер написанный на Python. Мы включили его в Django, чтобы вы сразу могли приступить к разработке, без дополнительной настройки боевого веб-сервера – например, Apache – пока вам это действительно не понадобится.

Следует заметить: НИКОГДА НЕ используйте этот сервер на «живом» сайте. Он создан исключительно для разработки. (Мы умеем делать Web-фреймворки, не Web-сервера.)

Теперь, когда сервер запущен, перейдите на страницу http://127.0.0.1:8000/ в браузере. Вы увидите страницу «Congratulations!» со взлетающей ракетой. Работает!

Поменять порт

По умолчанию, команда `runserver` запускает сервер для разработки на локальном IP используя порт 8000.

Если вы хотите изменить порт, укажите его как аргумент. Например, эта команда запускает сервер используя порт 8080:

```
$ python manage.py runserver 8080
```

Если вы хотите изменить IP сервера, передайте его вместе со значением порта. Чтобы слушать все публичные IP (полезно, если вы используете Vagrant, или хотите показать свою работу на других компьютерах), используйте:

```
$ python manage.py runserver 0:8000
```

0 короткая версия **0.0.0.0**. Смотрите полное описание команды `runserver`.

Автоматическая перезагрузка `runserver`

Dev-сервер самостоятельно перегружается при изменении Python файлов. Однако некоторые действия не перегружают сервер, например, добавление новых файлов. В таких случаях необходимо самостоятельно перегрузить сервер.

Создание приложения Polls

Теперь, после создания окружения (проекта), мы можем приступить к работе.

Каждое приложение Django состоит из пакета Python, который следует некоторым соглашениям. Django содержит команду, которая создает структуру для нового приложения, что позволяет вам сосредоточиться на написании кода, а не на создании каталогов.

Проекты или приложения

Какая разница между приложением и проектом? Приложение – это Web-приложение, которое предоставляет определенный функционал – например, Web-блог, хранилище каких-то записей или простое приложение для голосования. Проект – это совокупность приложений и конфигурации сайта. Проект может содержать несколько приложений. Приложение может использоваться несколькими проектами.

Ваше приложение может находиться где угодно в путях Python. В этом учебнике, мы будем создавать приложение голосования возле файла `manage.py`, и оно может быть импортировано как независимый модуль, а не подмодуль `mysite`.

Создавая приложение, убедитесь, что вы находитесь в том же каталоге, что и файл `manage.py`, и выполните команду:

```
$ python manage.py startapp polls
```

Эта команда создаст каталог `polls`:

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

Эти файлы являются частью приложения голосования.

Создадим свое первое представление

Давайте создадим свое первое представление. Откроем файл `polls/views.py` и добавим следующий код:

polls/views.py

```
from django.http import HttpResponse


def index(request):
    return HttpResponse("Hello, world. You're at the polls index.")
```

Это самое простое представление, которое можно создать на Django. Чтобы вызвать представление, нам нужно назначить его на какой-то URL - для этого нам нужна конфигурация URL-ов.

Чтобы добавить настройки URL-ов в приложение для голосования, создадим файл `urls.py`. Каталог приложения должен выглядеть следующим образом:

```
polls/
    __init__.py
    admin.py
    apps.py
    migrations/
        __init__.py
    models.py
    tests.py
    urls.py
    views.py
```

В файл `polls/urls.py` добавим следующий код:

polls/urls.py

```
from django.urls import path

from . import views

urlpatterns = [
    path('', views.index, name='index'),
]
```

Следующий шаг – добавить ссылку на `polls.urls` в главной конфигурации URL-ов. В ``mysite/urls.py`` добавим импорт `django.urls.include`, затем `include()` добавим в список `urlpatterns`. Вы должны получить следующий код:

mysite/urls.py

```
from django.contrib import admin
from django.urls import include, path

urlpatterns = [
    path('polls/', include('polls.urls')),
    path('admin/', admin.site.urls),
]
```

Функция `include()` позволяет ссылаться на другие конфигурации URL-ов. Когда Django встречает `include()`, обрезается часть URL, которая была распознана до этого момента, а оставшаяся часть отправляется в указанную конфигурация URL-ов для последующей обработки.

Идея `include()` в том, чтобы создать легко подключаемые URL-ы. Т.к. приложение для голосования содержит собственную конфигурацию URL-ов (`polls/urls.py`), они могут быть подключены в «/polls/», или «/fun_polls/», или «/content/polls/», или любой другой путь, и приложение будет работать.

Когда использовать `include()`

Вы всегда должны использовать `include()` при включении других шаблонов URL. Файл `admin.site.urls` является единственным исключением из этого.

Теперь вы связали представление `index` в настройках URL-ов. Проверим его работоспособность, запустив команду:

```
$ python manage.py runserver
```

Откройте в браузере http://localhost:8000/polls/, вы должны увидеть текст «*Hello, world. You're at the polls index.*», который вы указали в представлении `index`.

Страница не найдена?

Если вы получаете страницу с ошибкой, проверьте, что вы идёт на http://localhost:8000/polls/ , а не на http://localhost:8000/.

Функция `path()` принимает четыре аргумента, два из них обязательны: `route` и `view`, и два необязательных: `kwargs` и `name`. В этом месте лучше разобраться для чего предназначены эти аргументы.

path() аргумент: route

Аргумент `route` является строкой, содержащую шаблон URL. При обработке запроса Django начинает с первого шаблона в `urlpatterns` и проходит до конца списка, сравнивайся запрошенный URL с каждым шаблоном пока не найдёт совпавший.

Шаблоны не различают GET и POST параметры или доменные имена. Например, в запросе `https://www.example.com/myapp/`, URLconf будет искать `myapp/`. В запросе `https://www.example.com/myapp/?page=3`, URLconf будет также искать `myapp/`.

path() аргумент: view

Когда Django находит совпавший шаблон, он вызывает указанную функцию представления, передавая ей объект `HttpRequest` в качестве первого аргумента и все "захваченные" значения из `route` в виде именованных аргументов. Мы покажем это чуть позже.

url() аргумент: kwargs

В представление можно передать предопределенные именованные аргументы. Мы не будет использовать эту возможность в учебнике.

url() аргумент: name

Назначение имён вашим URL позволит вам однозначно ссылаться на низ из других мест проекта, особенно из шаблонов. Это позволяет вам вносить глобальные изменения в шаблоны URL вашего проекта, изменяя лишь один файл.

Изучив основы о запросах и ответах, приступим ко второй части учебника, чтобы начать работать с базой данных.