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 отображает пользователям голосования и позволяет им голосовать.
- Интерфейс администратора, который позволяет вам добавлять, изменять и удалять голосования.

Мы предполагаем, что вы уже установили Django. Вы можете проверить это, запустив консоль Python и выполнив следующие команды:

```
$ python -c "import django; print(django.get_version())"
```

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

Учебник написан для Django 1.9 и Python 3.4 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 Python 2.7, ваш код может отличаться от приведенного в этом учебнике, и вы должны немного изменить, как это будет указано в комментариях.

Создание проекта

Если вы используете Django первый раз, вам придется позаботиться о некоторых первоначальных настройках. А именно, сгенерировать основу проекта (Project) - пакет Python, то есть директория с исходными кодами, настроенная для работы с Django - содержит параметры подключения к базе данных, конфигурацию конкретной версии Django и разрабатываемого приложения. Django – настройки проекта, базы данных, приложений и др.

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

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

Это создаст каталог mysite в текущем каталоге.

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, 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
        wsgi.py
```

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;
- 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;
- mysite/settings.py: Настройки/конфигурация проекта;
- mysite/urls.py: Конфигурация URL-ов для вашего проекта Django. Это “содержание” всех Django-сайтов;
- mysite/wsgi.py: Точка входа вашего проекта для WSGI-совместимых веб-серверов.

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.

June 05, 2017 - 15:50:53
Django version 1.9, using settings 'mysite.settings'
Starting development server at http://127.0.0.1:8000/
Quit the server with CONTROL-C.
```

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 с “Welcome to Django”. Работает!

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 через конфигурацию 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.conf.urls import url

from . import views

urlpatterns = [
    url(r'^$', views.index, name='index'),
]
```

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

```
mysite/urls.py
from django.conf.urls import include, url
from django.contrib import admin

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

Вы добавили index в настройки URL-ов. Давайте проверим, что он работает, 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.

Функция url() принимает четыре аргумента, два обязательных: regex и view, и два необязательных: kwargs и name. Давайте изучим эти аргументы.

url() argument: regex

Термин “regex” обычно используется как короткий вариант “regular expression”(регулярное выражение), который являет шаблоном для распознавания строк, или в нашем случае URL-ов. Django начинает с первого регулярного выражение и идет дальше по списку, сравнивая запрошенный URL с каждым регулярным выражением пока не будет найдено совпадение.

Обратите внимание, регулярные выражения не проверяют GET и POST аргументы, или доменное имя. Например, при запросе на https://www.example.com/myapp/, будет проверяться myapp/. При запросе на https://www.example.com/myapp/?page=3, так же будет проверяться myapp/.

Если вам нужна помощь с регулярными выражениями, почитайте статью в Википедии и документацию модуля re. Также книга O’Reilly “Mastering Regular Expressions”, написанная Jeffrey Friedl, просто фантастична. На практике, однако, вам не нужно быть экспертом в регулярных выражениях, т.к. в основном вам нужно знать, как составлять простые регулярные выражения. На практике сложные регулярные выражения могу влиять на производительность, поэтому не следует полагаться на их широкие возможности.

Замечание о производительности: эти регулярные выражения компилируются при первом импорте модуля с конфигурацией URL-ов. Они работают очень быстро (до тех пор, пока они достаточно просты, как уже замечалось выше).

url() argument: view

когда Django находит подходящее регулярное выражение, Django вызывает указанную функцию представления, передавая первым аргументом объект HttpRequest и все распознанные регулярным выражением значения как остальные аргументы. Если регулярное выражение использует простое распознание значений, они передаются как позиционные аргументы, если используются именованное распознание, значения передаются как именованные аргументы. Позже мы рассмотрим несколько примеров.

url() argument: kwargs

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

url() argument: name

Добавив имя вашему URL-у, вы можете обратиться к нему по имени из любой точки проекта, особенно шаблонов. Такой подход позволяет делать глобальные изменения в настройках URL-ов, меняя минимум файлов.

Изучив основы о запросах и ответах, приступим к изучению следующей главе: «*Создаём своё первое приложение с Django, часть 2*», чтобы начать работать с базой данных.