



# Компьютерные технологии

Лекция № 6.

**Web-программирование: Django**

**Часть 1.**

Нижний  
Новгород  
2023 г.

# Django

Django - это высокоуровневая веб-инфраструктура Python, которая позволяет быстро создавать безопасные и поддерживаемые веб-сайты.

Django (джанго) — бесплатный и свободный фреймворк для веб-приложений, написанный на Python.

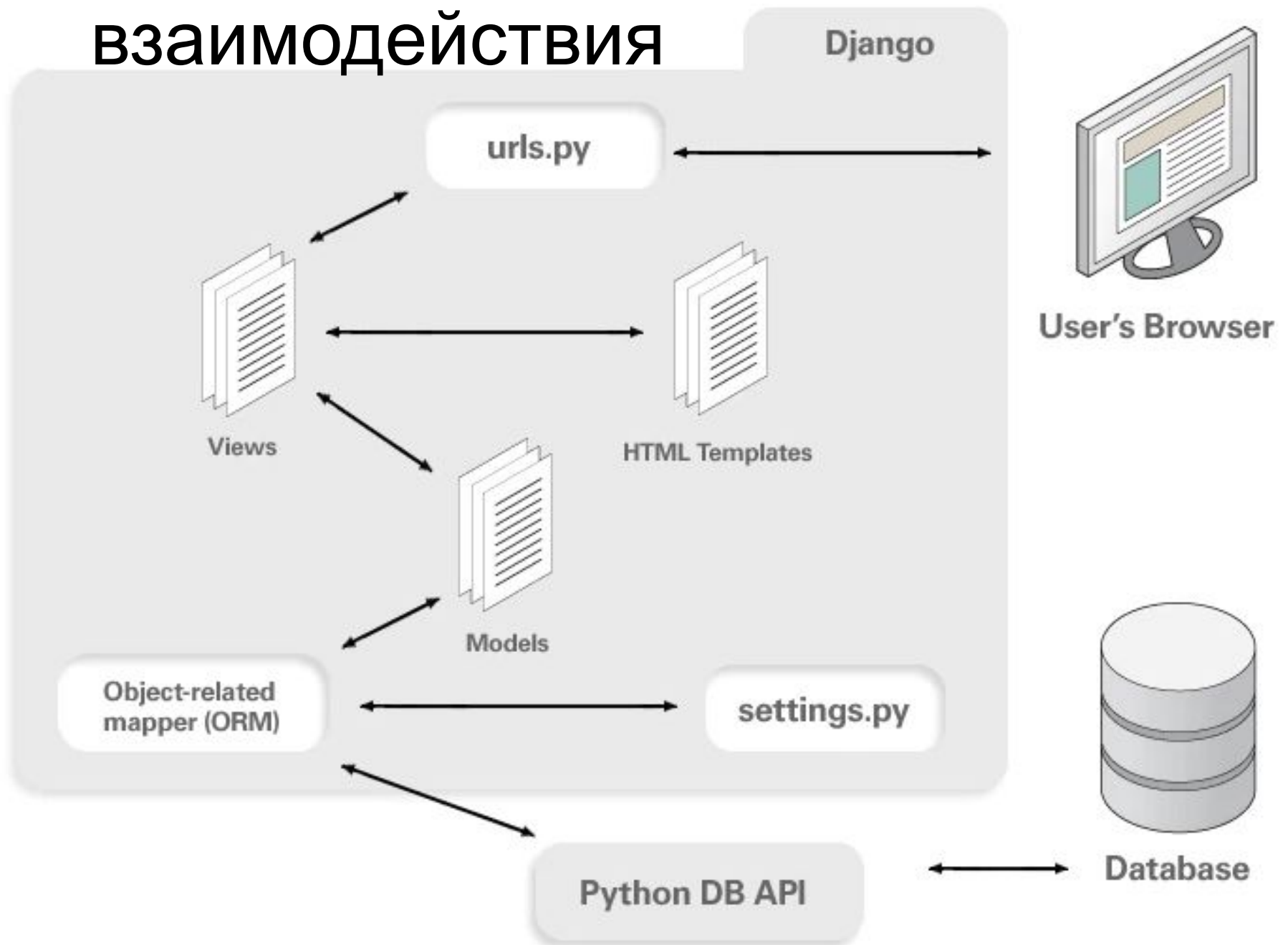
Фреймворк — это набор компонентов, которые помогают разрабатывать веб-сайты быстро и просто. Используют такие крупные сайты, как Disqus, Instagram, Knight Foundation, MacArthur Foundation, Mozilla, National Geographic, Open Knowledge Foundation, Pinterest и Open Stack.

# Что происходит, когда кто-то запрашивает веб-сайт у сервера?

Когда на сервер приходит запрос, он переадресуется Django, который пытается сообразить, что же конкретно от него просят. Для начала он берет адрес веб-страницы и пробует понять — что же нужно сделать. Эту часть процесса в Django выполняет `urlresolver` (адрес веб-сайта называется URL — Uniform Resource Locator — Единый указатель ресурсов, так что название `urlresolver`, `resolver` == определитель, имеет определенный смысл).

Он берет список шаблонов и пытается сопоставить их с URL. Django сверяет шаблоны сверху вниз и, если что-то совпадает, он переправляет запрос соответствующей функции (которая называется `view`).

# Схема взаимодействия



# Django. Элементы.

- **urls.py** URL-mapper используется для перенаправления HTTP-запросов в соответствующее представление на основе URL-адреса запроса. URL-mapper также может извлекать данные из URL-адреса в соответствии с заданным шаблоном и передавать их в соответствующую функцию в виде аргументов.
- **view:** Представление (view) – это функция обработчика запросов, которая получает HTTP-запросы и возвращает ответы. View имеет доступ к данным через модели (необходимым для удовлетворения запросов и делегирования ответа в шаблоны).
- **Models:** Модели представляют собой объекты Python, которые определяют структуру данных приложения и предоставляют механизмы для управления (добавления, изменения, удаления) и выполнения запросов в базу данных.
- **HTML Templates:** Template (шаблон) – это текстовый файл определяющий структуру или разметку страницы (например HTML страницы), с полями для подстановки используемыми для представления актуального содержимого.

# Настройка virtualenv

Virtualenv изолирует зависимости Python/Django для каждого отдельного проекта. Это значит, что изменения одного сайта никогда не затронут другие сайты.

- Создадим директорию, например *djangopracticum*.
- Создадим виртуальное окружение под именем *myvenv*. В общем случае команда будет выглядеть так:

***python3 -m venv myvenv***

- Запустим виртуальное окружение. (Префикс myvenv сообщит, что virtualenv запущено)

```
C:\Users\Olga>mkdir djangopracticum
C:\Users\Olga>cd djangopracticum
C:\Users\Olga\djangopracticum>python -m venv myvenv
C:\Users\Olga\djangopracticum>myvenv\Scripts\activate.bat
(myvenv) C:\Users\Olga\djangopracticum>
(myvenv) C:\Users\Olga\djangopracticum>
```

# Установка Django

Перед этим необходимо удостовериться, что установлена последняя версия pip.

*(myvenv) ~\$ pip install --upgrade pip*

Затем запускаем команду: *pip install django~=1.11.0* , чтобы установить Django.

```
(myvenv) C:\Users\Olga\django practicum>
(myvenv) C:\Users\Olga\django practicum>python -m pip install django~=1.11.0
Collecting django~=1.11.0
  Downloading Django-1.11.10-py2.py3-none-any.whl (6.9MB)
    100% |#####| 7.0MB 17kB/s
Collecting pytz (from django~=1.11.0)
  Downloading pytz-2018.3-py2.py3-none-any.whl (509kB)
    100% |#####| 512kB 201kB/s
Installing collected packages: pytz, django
Successfully installed django-1.11.10 pytz-2018.3
```

# Установка Git

Git — это «система управления версиями», используемая множеством программистов. Эта программа отслеживает изменения, происходящие с файлами, чтобы впоследствии можно было восстановить состояние кода на нужный момент времени.

- Необходимо загрузить Git с официального сайта [git-scm.com](https://git-scm.com). На всех этапах установки нажать далее, за исключением одного: на пятом шаге "Adjusting your PATH environment" выбрать "Use Git and optional Unix tools from the Windows Command Prompt". После окончания установки перезапустить командную строку.
- Перейти на [GitHub.com](https://github.com) и зарегистрировать новый бесплатный аккаунт.



# Создание учётной записи на PythonAnywhere

Приложения Django можно развернуть на хостинге **PythonAnywhere** (облачная платформа, предназначенная преимущественно для запуска приложений Python).

Необходимо создать бесплатный аккаунт уровня "Beginner" на PythonAnywhere: [www.pythonanywhere.com](http://www.pythonanywhere.com)

При выборе имени пользователя необходимо учесть, что URL сайта примет вид *yourusername.pythonanywhere.com*.

# Проект на Django

Первый шаг — создать новый проект Django. В сущности, это значит, что мы запустим несколько стандартных скриптов из поставки Django, которые создадут для нас скелет проекта (каталоги и файлы). Названия этих каталогов и файлов очень важны для Django (нельзя переименовывать или перемещать).

*(myvenv) C:\Users\Olga\django practicum> django-admin.exe startproject mysite .*

*django-admin.py* — это скрипт, который создаст необходимую структуру директорий и файлы.

# Изменяем настройки

Внесём изменения в `mysite/settings.py`:

- **Установим корректный часовой пояс:**

```
TIME_ZONE = 'Europe/Moscow'
```

- **Изменим язык, отредактировав следующую строку:**

```
LANGUAGE_CODE = 'ru-ru'
```

- **Добавим в настройки информацию о расположении статических файлов** (в конце файла и после переменной `STATIC_URL` добавим новую — `STATIC_ROOT`):

```
STATIC_URL = '/static/'
```

```
STATIC_ROOT = os.path.join(BASE_DIR, 'static')
```

- **Добавим разрешённый *host*** (имя пользователя на PythonAnywhere):

```
ALLOWED_HOSTS = ['127.0.0.1',
```

```
'<твое_имя_пользователя>.pythonanywhere.com']
```

# Настройка базы данных

Существует множество различных баз данных, которые могут хранить данные для твоего сайта. Мы будем использовать стандартную — sqlite3.

Чтобы создать базу данных используем (директории, где расположен файл manage.py):

***python manage.py migrate***

```
(myvenv) C:\Users\Olga\djangopracticum>django-admin.exe startproject mysite .  
  
(myvenv) C:\Users\Olga\djangopracticum>python manage.py migrate  
Operations to perform:  
  Apply all migrations: admin, auth, contenttypes, sessions  
Running migrations:  
  Applying contenttypes.0001_initial... OK  
  Applying auth.0001_initial... OK  
  Applying admin.0001_initial... OK  
  Applying admin.0002_logentry_remove_auto_add... 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 sessions.0001_initial... OK
```

# Запуск веб-сервера

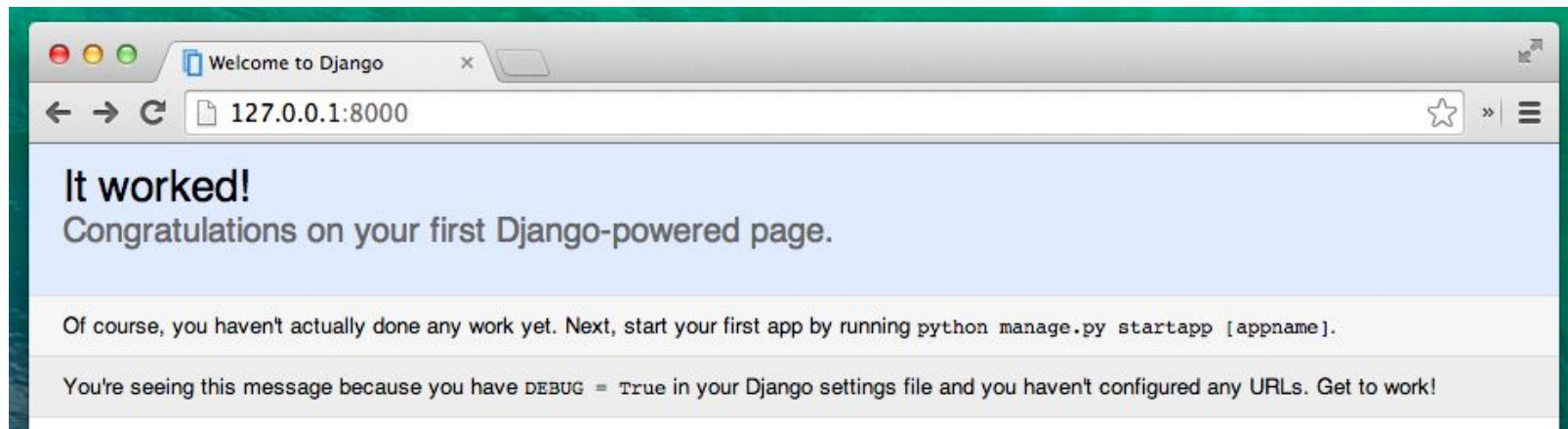
Запустим веб-сервер из командной строки:

***python manage.py runserver***

```
(myenv) C:\Users\Olga\django\practicum>python manage.py runserver
Performing system checks...

System check identified no issues (0 silenced).
February 24, 2018 - 20:11:32
Django version 1.11.10, using settings 'mysite.settings'
Starting development server at http://127.0.0.1:8000/
Quit the server with CTRL-BREAK.
[24/Feb/2018 20:11:57] "GET / HTTP/1.1" 200 1998
```

Теперь нужно проверить, работает ли веб-сайт — открыть браузер и набрать адрес: <http://127.0.0.1:8000/>

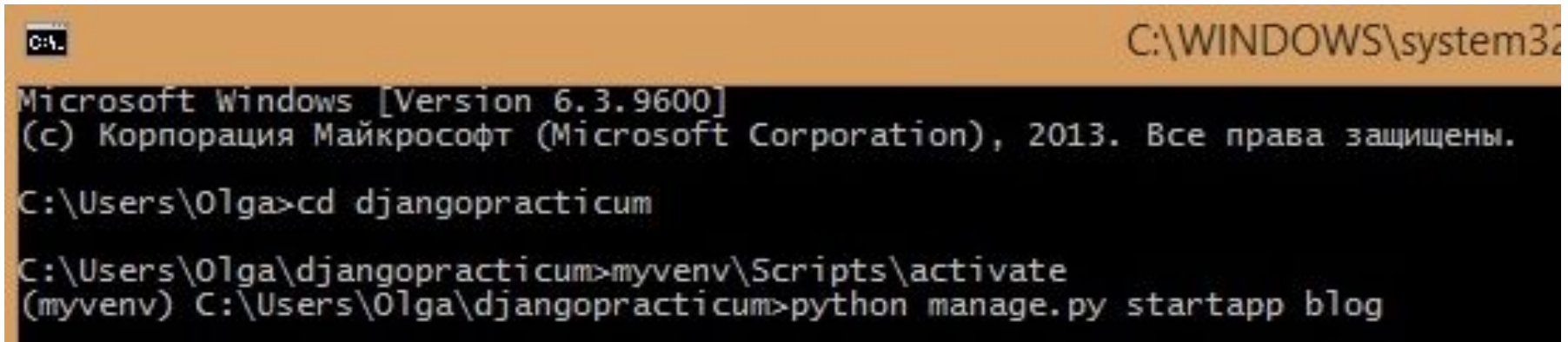


# Модели Django

Модель в Django — это **объект** определённого свойства, он хранится в базе данных (используем SQLite).

Создадим отдельное приложение в нашем проекте, используя ***python manage.py startapp blog***

*(Перед этим необходимо запустить виртуальное окружение.)*

A screenshot of a Windows command prompt window. The title bar is orange and shows the path 'C:\WINDOWS\system32'. The window content is black with white text. It shows the command prompt interface with the following text: 'Microsoft Windows [Version 6.3.9600] (c) Корпорация Майкрософт (Microsoft Corporation), 2013. Все права защищены. C:\Users\Olga>cd.djangopracticum C:\Users\Olga\djangopracticum>myenv\Scripts\activate (myenv) C:\Users\Olga\djangopracticum>python manage.py startapp blog'.

```
C:\WINDOWS\system32
Microsoft Windows [Version 6.3.9600]
(c) Корпорация Майкрософт (Microsoft Corporation), 2013. Все права защищены.
C:\Users\Olga>cd.djangopracticum
C:\Users\Olga\djangopracticum>myenv\Scripts\activate
(myenv) C:\Users\Olga\djangopracticum>python manage.py startapp blog
```

# Модели Django

После того, как приложение создано, нам нужно сообщить Django, что теперь он должен его использовать.

Сделаем это с помощью файла `mysite/settings.py`.

Нужно найти **INSTALLED\_APPS** и добавить к списку **'blog'**, прямо перед `]`. Конечный результат должен выглядеть следующим образом:

```
INSTALLED_APPS = [  
    'django.contrib.admin',  
    'django.contrib.auth',  
    'django.contrib.contenttypes',  
    'django.contrib.sessions',  
    'django.contrib.messages',  
    'django.contrib.staticfiles',  
    'blog',  
]
```

# Создание модели записи в блоге

В файле `blog/models.py` определяем все модели (удалить все и добавить).

```
from django.db import models
from django.utils import timezone

class Post(models.Model):
    author = models.ForeignKey('auth.User', on_delete=models.CASCADE)
    title = models.CharField(max_length=200)
    text = models.TextField()
    created_date = models.DateTimeField(
        default=timezone.now)
    published_date = models.DateTimeField(
        blank=True, null=True)

    def publish(self):
        self.published_date = timezone.now()
        self.save()

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



# Создаем таблицы моделей в базе данных

Далее добавим модель в базу данных. Сначала мы должны дать Django знать, что сделали изменения в нашей модели.

*python manage.py makemigrations blog*

(команда создаст файл с миграцией для базы данных)

```
(myvenv) C:\Users\Olga\django\practicum>python manage.py startapp blog

(myvenv) C:\Users\Olga\django\practicum>python manage.py makemigrations blog
Migrations for 'blog':
  blog\migrations\0001_initial.py
    - Create model Post

(myvenv) C:\Users\Olga\django\practicum>python manage.py migrate blog
Operations to perform:
  Apply all migrations: blog
Running migrations:
  Applying blog.0001_initial... OK

(myvenv) C:\Users\Olga\django\practicum>_
```

# Администрирование Django

Чтобы добавлять, редактировать и удалять записи используем панель управления администратора Django.

**Импортируем и регистрируем модель Post:** Откроем файл `blog/admin.py` и заменим его содержимое на (далее можно зайти на <http://127.0.0.1:8000/admin/>):

```
from django.contrib import admin
```

```
from .models import Post
```

```
admin.site.register(Post)
```

Далее необходимо создать супер пользователя:

```
python manage.py createsuperuser
```

```
(myvenv) C:\Users\Olga\django\practicum>python manage.py createsuperuser
Username (leave blank to use 'olga'): admin
Email address: admin@admin.com
Password:
Password (again):
This password is too short. It must contain at least 8 characters.
This password is too common.
This password is entirely numeric.
Password:
Password (again):
Superuser created successfully.

(myvenv) C:\Users\Olga\django\practicum>_
```

# Администрирование Django

Вернемся в браузер и войдем в систему при помощи имени пользователя и пароля для супер пользователя (попадаем в панель управления Django).



Необходимо создать несколько постов с различной датой публикации.

# Развертывание

Это процесс публикации приложения в сети (до настоящего момента сайт был доступен только для локального просмотра).

Будем использовать ***GitHub*** – сервис хостинга кода и ***PythonAnywhere*** в качестве сервера.

Общая последовательность действий при работе с **Git**:

## На локальной станции:

1. **git status** (проверяем есть ли изменения)
2. **git add --all .** (добавляем изменения в текущую ветку)
3. **git commit** (сохраняем изменения)
4. **git push** (отправляем изменения на сервер)

## На сервере:

1. **git pull** (забираем изменения с GitHub)

# Развертывание

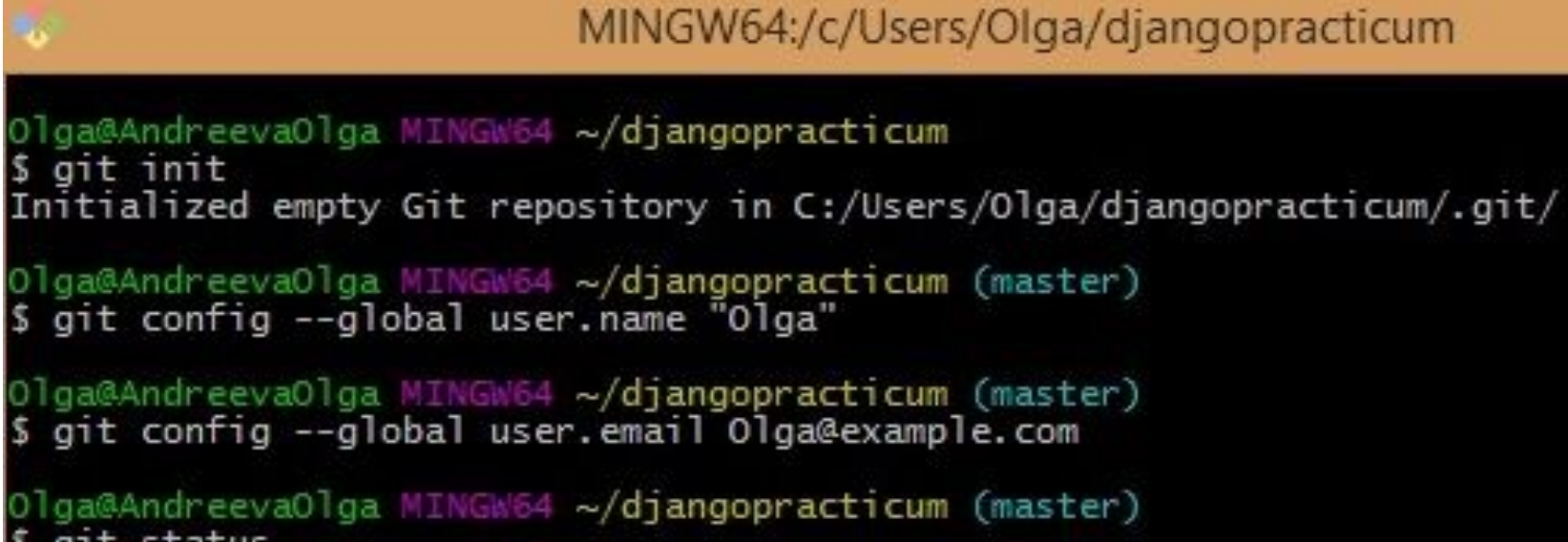
**Создаём Git-репозиторий** (только один раз за проект)

Git отслеживает изменения определенного набора файлов, который называется репозиторием. Из директории проекта выполняем:

```
$ git init
```

```
$ git config --global user.name "Your Name"
```

```
$ git config --global user.email you@example.com
```



The screenshot shows a Windows command prompt window with the title bar "MINGW64:/c/Users/Olga/djangopracticum". The prompt is "Olga@AndreevaOlga MINGW64 ~/djangopracticum". The commands and their outputs are as follows:

```
Olga@AndreevaOlga MINGW64 ~/djangopracticum
$ git init
Initialized empty Git repository in C:/Users/Olga/djangopracticum/.git/

Olga@AndreevaOlga MINGW64 ~/djangopracticum (master)
$ git config --global user.name "Olga"

Olga@AndreevaOlga MINGW64 ~/djangopracticum (master)
$ git config --global user.email Olga@example.com

Olga@AndreevaOlga MINGW64 ~/djangopracticum (master)
$ git status
```

# Развертывание

Git будет отслеживать изменения всех файлов и каталогов в заданной директории, однако некоторые из них нам лучше игнорировать. Для этого нужно создать файл **.gitignore** в корневом каталоге репозитория. Открываем редактор и создаем новый файл со следующим содержанием:

```
*.рус
```

```
*~
```

```
__pycache__
```

```
myvenv
```

```
db.sqlite3
```

```
/static
```

```
.DS_Store
```

Сохраняем его как **.gitignore** в корневом каталоге.

# Развертывание

Далее выполняем команды `git status`, `git add --all .` и `git commit -m "first commit"`

```
Olga@AndreevaOlga MINGW64 ~/djangopracticum (master)
$ git status
On branch master

Initial commit

Untracked files:
  (use "git add <file>..." to include in what will be committed)

        .gitignore
        blog/
        manage.py
        mysite/

nothing added to commit but untracked files present (use "git add" to track)

Olga@AndreevaOlga MINGW64 ~/djangopracticum (master)
$ git add --all .

Olga@AndreevaOlga MINGW64 ~/djangopracticum (master)
$ git commit -m "First commit"
[master (root-commit) 8ab382a] First commit
14 files changed, 252 insertions(+)
create mode 100644 .gitignore
create mode 100644 blog/__init__.py
create mode 100644 blog/admin.py
create mode 100644 blog/apps.py
create mode 100644 blog/migrations/0001_initial.py
create mode 100644 blog/migrations/__init__.py
create mode 100644 blog/models.py
create mode 100644 blog/tests.py
create mode 100644 blog/views.py
create mode 100644 manage.py
create mode 100644 mysite/__init__.py
create mode 100644 mysite/settings.py
create mode 100644 mysite/urls.py
create mode 100644 mysite/wsgi.py
```



# Развертывание. Загружаем код в репозиторий GitHub

На GitHub создаем новый репозиторий "my-first-blog". Не выбираем опцию "initialise with a README", не создаем файл .gitignore и оставляем лицензию None.

The screenshot shows the GitHub repository creation interface. At the top, the 'Owner' is 'hjwp' and the 'Repository name' is 'my-first-blog', which is marked with a green checkmark. Below this, a hint suggests repository names should be short and memorable, with an example 'ducking-octo-tyrion'. The 'Description' field is optional and empty. Under the 'Visibility' section, 'Public' is selected by default, with a description: 'Anyone can see this repository. You choose who can commit.' The 'Private' option is also available. The 'Initialize this repository with a README' option is unchecked, and a red arrow points to it. Below this, there are two dropdown menus: 'Add .gitignore: None' and 'Add a license: None', both with red arrows pointing to them. At the bottom, there is a green 'Create repository' button.

Owner: hjwp / Repository name: my-first-blog ✓

Great repository names are short and memorable. Need inspiration? How about [ducking-octo-tyrion](#).

Description (optional)

☒ **Public**  
Anyone can see this repository. You choose who can commit.

☐ **Private**  
You choose who can see and commit to this repository.

☐ **Initialize this repository with a README**  
This will let you immediately clone the repository to your computer. Skip this step if you're importing an existing repository.

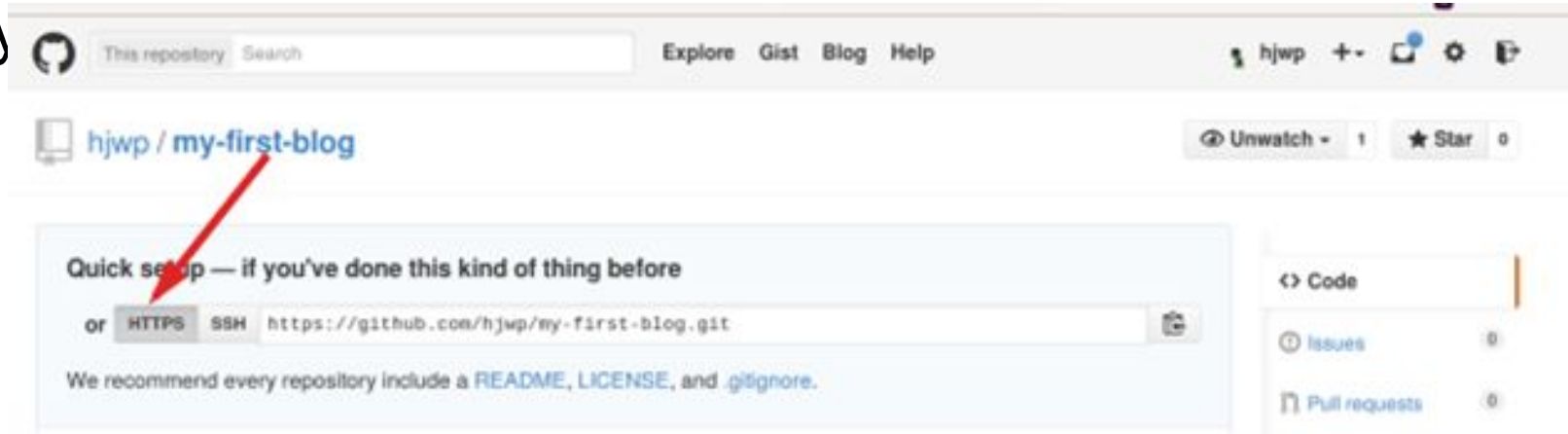
Add .gitignore: **None** Add a license: **None** ⓘ

**Create repository**



# Развертывание. Загружаем код в репозиторий GitHub

На следующем экране выбираем вариант "HTTPS" и копируем



Далее связываем локальный репозиторий с репозиторием на GitHub:

```
$ git remote add origin https://github.com/<your-github-username>/my-first-blog.git
```

\$ git push -u origin master

```
Olga@AndreevaOlga MINGW64 ~/djangopracticum (master)
$ git remote add origin https://github.com/OlgaAndreeva/my-first-blog.git

Olga@AndreevaOlga MINGW64 ~/djangopracticum (master)
$ git push -u origin master
Counting objects: 17, done.
Delta compression using up to 4 threads.
Compressing objects: 100% (16/16), done.
Writing objects: 100% (17/17), 4.09 KiB | 0 bytes/s, done.
Total 17 (delta 0), reused 0 (delta 0)
To https://github.com/OlgaAndreeva/my-first-blog.git
 * [new branch]      master -> master
Branch master set up to track remote branch master from origin.

Olga@AndreevaOlga MINGW64 ~/djangopracticum (master)
```

# Настройка блога на PythonAnywhere.

## Загружаем код на PythonAnywhere.

После регистрации на PythonAnywhere на странице «Consoles» выбираем опцию старта консоли «Bash» — это версия консоли PythonAnywhere, аналогичная локальному терминалу



Загружаем код из GitHub на PythonAnywhere, создав «клон» репозитория. Введем команду в консоли на PythonAnywhere:

\$ git clone https://github.com/<your github username>/my-first-blog.git

```
18:10 ~ $ git clone https://github.com/OlgaAndreeva/my-first-blog.git
Cloning into 'my-first-blog'...
remote: Counting objects: 17, done.
remote: Compressing objects: 100% (16/16), done.
remote: Total 17 (delta 0), reused 17 (delta 0), pack-reused 0
Unpacking objects: 100% (17/17), done.
Checking connectivity... done.
18:11 ~$ tree my-first-blog.git
```

# Настройка блога на PythonAnywhere.

## Загружаем код на PythonAnywhere.

Команда загружает копию кода с GitHub на PythonAnywhere.  
Проверяем это, набрав `tree my-first-blog`:

```
18:11 ~ $ tree my-first-blog
my-first-blog
├── blog
│   ├── __init__.py
│   ├── admin.py
│   ├── apps.py
│   ├── migrations
│   │   └── 0001_initial.py
│   │       └── __init__.py
│   ├── models.py
│   ├── tests.py
│   └── views.py
├── manage.py
└── mysite
    ├── __init__.py
    ├── settings.py
    ├── urls.py
    └── wsgi.py

3 directories, 13 files
18:12 ~ $
```

# Создаём виртуальное окружение на PythonAnywhere

По аналогии с локальной версией создаем виртуальное окружение на PythonAnywhere:

```
18:12 ~ $ cd my-first-blog/
18:13 ~/my-first-blog (master)$ virtualenv --python=python3.6 myvenv
Running virtualenv with interpreter /usr/bin/python3.6
Using base prefix '/usr'
New python executable in /home/OlgaAndreeva/my-first-blog/myvenv/bin/python3.6
Also creating executable in /home/OlgaAndreeva/my-first-blog/myvenv/bin/python
Installing setuptools, pip, wheel...done.
18:13 ~/my-first-blog (master)$ source myvenv/bin/activate
(myvenv) 18:13 ~/my-first-blog (master)$ pip install django~=1.11.0
Collecting django~=1.11.0
  Downloading Django-1.11.10-py2.py3-none-any.whl (6.9MB)
    100% |#####| 7.0MB 56kB/s
Collecting pytz (from django~=1.11.0)
  Downloading pytz-2018.3-py2.py3-none-any.whl (509kB)
    100% |#####| 512kB 1.5MB/s
Installing collected packages: pytz, django
Successfully installed django-1.11.10 pytz-2018.3
(myvenv) 18:15 ~/my-first-blog (master)$
```



# Создаём базу данных на PythonAnywhere

Одно отличие локального компьютера и сервера — они используют разные базы данных. Таким образом, пользовательские аккаунты и записи в блоге на сервере и локальном компьютере могут отличаться друг от друга. Необходимо инициализировать базу данных— с помощью команд

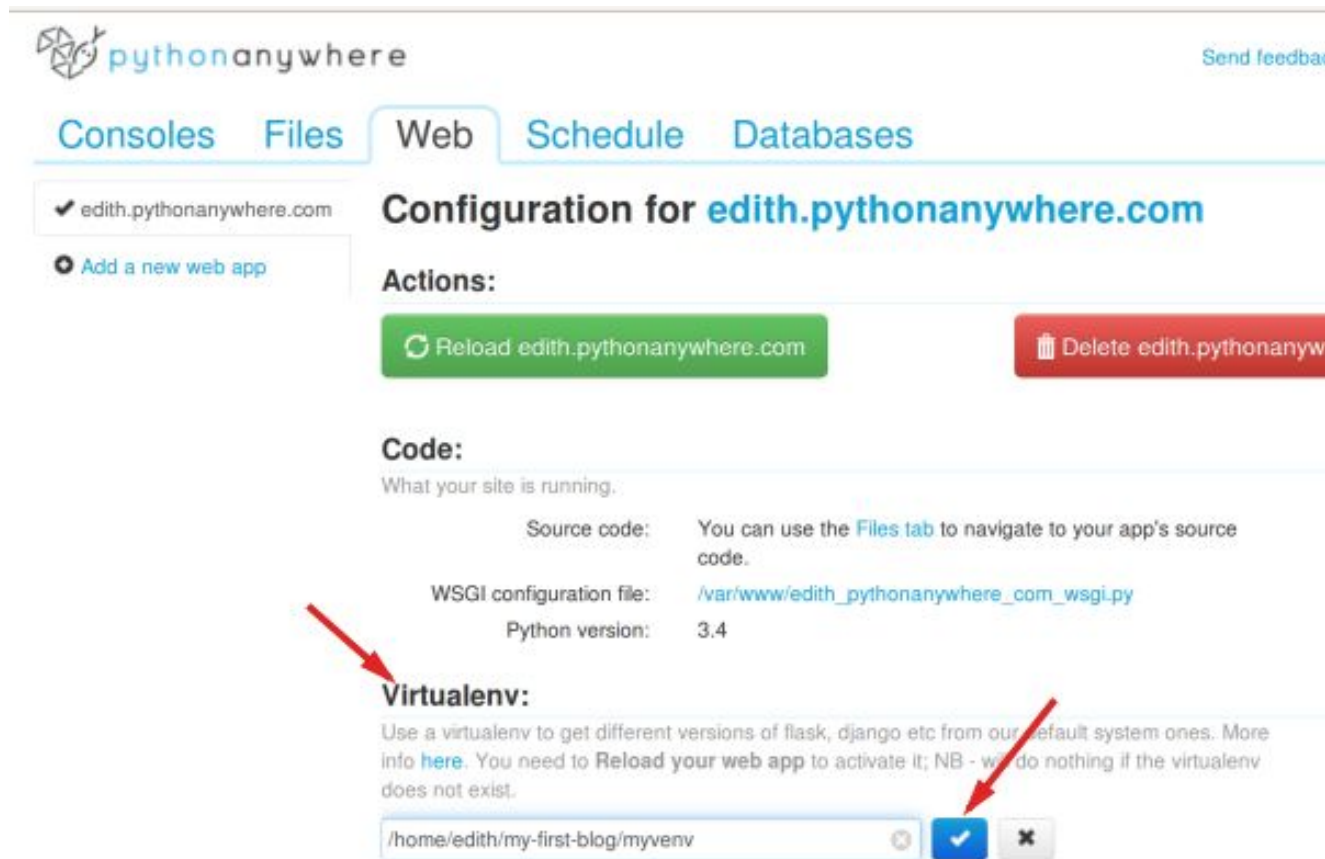
```
(myvenv) 18:16 ~/my-first-blog (master)$ python manage.py migrate
Operations to perform:
  Apply all migrations: admin, auth, blog, contenttypes, sessions
Running migrations:
  Applying contenttypes.0001_initial... OK
  Applying auth.0001_initial... OK
  Applying admin.0001_initial... OK
  Applying admin.0002_logentry_remove_auto_add... 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 blog.0001_initial... OK
  Applying sessions.0001_initial... OK
(myvenv) 18:16 ~/my-first-blog (master)$ python manage.py createsuperuser
Username (leave blank to use 'olgaandreeva'): admin
Email address: admin@admin.com
Password:
```

# Публикация нашего блога как веб-приложения

Возвращаемся в панель управления PythonAnywhere, нажав на лого в верхнем левом углу, затем переключаемся на вкладку **Web** и нажимаем кнопку **Add a new web app**. После подтверждения доменного имени выбираем **Manual configuration** (не «Django»!) в диалоговом окне. Затем выбираем **Python 3.6** и завершаем работу мастера.

# Настройка виртуального окружения

На странице настройки приложения в секции "Virtualenv" кликаем по красному тексту "Enter the path to a virtualenv" и набираем `/home/<your-username>/my-first-blog/myvenv/`. Нажимаем на синий прямоугольник с галочкой, чтобы сохранить изменения, прежде чем двигаться дальше.



pythonanywhere [Send feedback](#)

[Consoles](#) [Files](#) [Web](#) [Schedule](#) [Databases](#)

✓ edith.pythonanywhere.com [Add a new web app](#)

## Configuration for edith.pythonanywhere.com

**Actions:**

[Reload edith.pythonanywhere.com](#) [Delete edith.pythonanyw](#)

**Code:**  
What your site is running.

Source code: You can use the [Files tab](#) to navigate to your app's source code.

WSGI configuration file: `/var/www/edith_pythonanywhere_com_wsgi.py`

Python version: 3.4

**Virtualenv:**  
Use a virtualenv to get different versions of flask, django etc from our default system ones. More info [here](#). You need to **Reload your web app** to activate it; NB - we do nothing if the virtualenv does not exist.

`/home/edith/my-first-blog/myvenv` [✓](#) [✕](#)

# Настройка файла WSGI

Django использует протокол WSGI, стандартный протокол для обслуживания веб-сайтов, использующих Python, который поддерживается PythonAnywhere. Используя файл настроек WSGI, мы позволим PythonAnywhere распознать наш Django блог.

Кликаем по ссылке "**WSGI configuration file**" (в секции «Code» наверху страницы — она будет выглядеть следующим образом: `/var/www/<your-username>_pythonanywhere_com_wsgi.py`)

Удаляем все содержимое и заменяем его на:



# Настройка файла WSGI

```
import os
import sys

path = os.path.expanduser('~ /my-first-blog')
if path not in sys.path:
    sys.path.append(path)

os.environ['DJANGO_SETTINGS_MODULE'] = 'mysite.settings'

from django.core.wsgi import get_wsgi_application
from django.contrib.staticfiles.handlers import StaticFilesHandler
application = StaticFilesHandler(get_wsgi_application())
```

Задача данного файла — сказать PythonAnywhere, где находится наше веб-приложение и как называется файл настроек Django.

***StaticFilesHandler*** нужен для обработки наших CSS. Она происходит автоматически во время разработки при запуске runserver.

Нажимаем Save и переключаемся на вкладку Web.

# Советы по отладке

Если появились ошибки при попытке посетить сайт, для получения отладочной информации необходимо просмотреть **журнал ошибок**.

Это могут быть:

- Пропуск одного из шагов в консоли: создание `virtualenv`, её активация, установка Django в виртуальное окружение, инициализация базы данных.
- Ошибка в пути к `virtualenv` — рядом должно появиться небольшое предупреждение, если PythonAnywhere не может найти виртуальное окружение по указанному адресу.
- Ошибка в файле настроек WSGI — правильно указан путь к директории `my-first-blog`?
- Выбрана одна и та же версия Python для `virtualenv` и для веб-приложения? Обе должны быть 3.6.

Также можно посмотреть общие советы по отладке на вики PythonAnywhere: