**Your first Django project!**

Part of this chapter is based on tutorials by Geek Girls Carrots (<https://github.com/ggcarrots/django-carrots>).

Parts of this chapter are based on the [django-marcador tutorial](http://django-marcador.keimlink.de/" \t "_blank) licensed under the Creative Commons Attribution-ShareAlike 4.0 International License. The django-marcador tutorial is copyrighted by Markus Zapke-Gründemann et al.

We're going to create a small blog!

The first step is to start a new Django project. Basically, this means that we'll run some scripts provided by Django that will create the skeleton of a Django project for us. This is just a bunch of directories and files that we will use later.

The names of some files and directories are very important for Django. You should not rename the files that we are about to create. Moving them to a different place is also not a good idea. Django needs to maintain a certain structure to be able to find important things.

Remember to run everything in the virtualenv. If you don't see a prefix (myvenv) in your console, you need to activate your virtualenv. We explained how to do that in the **Django installation** chapter in the **Working with virtualenv** part. Typing myvenv\Scripts\activate on Windows or source myvenv/bin/activate on Mac OS X or Linux will do this for you.

**Create project: OS X or Linux**

In your Mac OS X or Linux console, you should run the following command. **Don't forget to add the period (or dot) . at the end!**

command-line

(myvenv) ~/djangogirls$ django-admin startproject mysite .

The period . is crucial because it tells the script to install Django in your current directory (for which the period . is a short-hand reference).

**Note** When typing the command above, remember that you only type the part which starts by django-admin. The (myvenv) ~/djangogirls$ part shown here is just example of the prompt that will be inviting your input on your command line.

**Create project: Windows**

On Windows you should run the following command. **(Don't forget to add the period (or dot) . at the end)**:

command-line

(myvenv) C:\Users\Name\djangogirls> django-admin.exe startproject mysite .

The period . is crucial because it tells the script to install Django in your current directory (for which the period . is a short-hand reference).

**Note** When typing the command above, remember that you only type the part which starts by django-admin.exe. The (myvenv) C:\Users\Name\djangogirls> part shown here is just example of the prompt that will be inviting your input on your command line.

django-admin.py is a script that will create the directories and files for you. You should now have a directory structure which looks like this:

djangogirls

├───manage.py

└───mysite

settings.py

urls.py

wsgi.py

\_\_init\_\_.py

**Note**: in your directory structure, you will also see your venv directory that we created before.

manage.py is a script that helps with management of the site. With it we will be able (amongst other things) to start a web server on our computer without installing anything else.

The settings.py file contains the configuration of your website.

Remember when we talked about a mail carrier checking where to deliver a letter? urls.pyfile contains a list of patterns used by urlresolver.

Let's ignore the other files for now as we won't change them. The only thing to remember is not to delete them by accident!

**Changing settings**

Let's make some changes in mysite/settings.py. Open the file using the code editor you installed earlier.

**Note**: Keep in mind that settings.py is a regular file, like any other. You can open it from inside the code editor, using the "file -> open" menu actions. This should get you the usual window in which you can navigate to your settings.py file and select it. Alternatively, you can open the file by navigating to the djangogirls folder on your desktop and right-clicking on it. Then, select your code editor from the list. Selecting the editor is important as you might have other programs installed that can open the file but will not let you edit it.

It would be nice to have the correct time on our website. Go to [Wikipedia's list of time zones](https://en.wikipedia.org/wiki/List_of_tz_database_time_zones" \t "_blank)and copy your relevant time zone (TZ) (e.g. Europe/Berlin).

In settings.py, find the line that contains TIME\_ZONE and modify it to choose your own timezone. For example:

mysite/settings.py

TIME\_ZONE = 'Europe/Berlin'

A language code consist of the language, e.g. en for english or de for german, and the country code, e.g. de for germany or ch for switzerland. You will want to add this if you want the default buttons and notifications from Django to be in your language. So you would have "Cancel" button translated into the language you defined here. [Django comes with a lot of prepared translations](https://docs.djangoproject.com/en/1.11/ref/settings/" \l "language-code" \t "_blank).

Change the language code by changing the following line:

mysite/settings.py

LANGUAGE\_CODE = 'de-ch'

We'll also need to add a path for static files. (We'll find out all about static files and CSS later in the tutorial.) Go down to the *end* of the file, and just underneath the STATIC\_URL entry, add a new one called STATIC\_ROOT:

mysite/settings.py

STATIC\_URL = '/static/'

STATIC\_ROOT = os.path.join(BASE\_DIR, 'static')

When DEBUG is True and ALLOWED\_HOSTS is empty, the host is validated against ['localhost', '127.0.0.1', '[::1]']. This won't match our hostname on PythonAnywhere once we deploy our application so we will change the following setting:

mysite/settings.py

ALLOWED\_HOSTS = ['127.0.0.1', '.pythonanywhere.com']

**Note**: If you're using a Chromebook, add this line at the bottom of your settings.py file:MESSAGE\_STORAGE = 'django.contrib.messages.storage.session.SessionStorage'

Also add .c9users.io to the ALLOWED\_HOSTS if you are using cloud9

**Set up a database**

There's a lot of different database software that can store data for your site. We'll use the default one, sqlite3.

This is already set up in this part of your mysite/settings.py file:

mysite/settings.py

DATABASES = {

'default': {

'ENGINE': 'django.db.backends.sqlite3',

'NAME': os.path.join(BASE\_DIR, 'db.sqlite3'),

}

}

To create a database for our blog, let's run the following in the console: python manage.py migrate (we need to be in the djangogirls directory that contains the manage.py file). If that goes well, you should see something like this:

command-line

(myvenv) ~/djangogirls$ python manage.py migrate

Operations to perform:

Apply all migrations: auth, admin, contenttypes, sessions

Running migrations:

Rendering model states... DONE

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 sessions.0001\_initial... OK

And we're done! Time to start the web server and see if our website is working!

**Starting the web server**

You need to be in the directory that contains the manage.py file (the djangogirls directory). In the console, we can start the web server by running python manage.py runserver:

command-line

(myvenv) ~/djangogirls$ python manage.py runserver

If you are on a Chromebook, use this command instead:

Cloud 9

(myvenv) ~/djangogirls$ python manage.py runserver 0.0.0.0:8080

If you are on Windows and this fails with UnicodeDecodeError, use this command instead:

command-line

(myvenv) ~/djangogirls$ python manage.py runserver 0:8000

Now all you need to do is check that your website is running. Open your browser (Firefox, Chrome, Safari, Internet Explorer or whatever you use) and enter this address:

browser

http://127.0.0.1:8000/

If you're using a Chromebook, you'll always visit your test server by accessing:

browser

https://django-girls-<your cloud9 username>.c9users.io

Congratulations! You've just created your first website and run it using a web server! Isn't that awesome?

While the web server is running, you won't see a new command-line prompt to enter additional commands. The terminal will accept new text but will not execute new commands. This is because the web server continuously runs in order to listen for incoming requests.

We reviewed how web servers work in the **How the Internet works** chapter.

To type additional commands while the web server is running, open a new terminal window and activate your virtualenv. To stop the web server, switch back to the window in which it's running and press CTRL+C - Control and C keys together (on Windows, you might have to press Ctrl+Break).

Ready for the next step? It's time to create some content!