
# BackEnd Workshop-1

Clarusway

# Subject: First App (Welcome to BackEnd)

## Learning Goals

- Practice creating a basic Django project and app.

## Introduction

Let's get our hands dirty! Create a Django project from scratch.

## Practice Using the `IDE` in Lesson

It's much more easy to work with an IDE, we will use VSCode. But, it's ok to use any other you are familiar.

## Nice to have VSCode Extentions:

Djaneiro - Django Snippets

## Needs

- Python
- pip
- virtualenv

# Code Along

# Part 1 - Create first project and app

1. Create a working directory, cd to new directory.

2. Create virtual environment as a best practice:

```
python3 -m venv env # for Windows or
python -m venv env # for Windows
virtualenv env # for Mac/Linux or;
virtualenv yourenv -p python3 # for Mac/Linux
```

3. Activate scripts:

```
.\env\Scripts\activate  # for Windows, may need to switch powershell on
this operation.
source env/bin/activate  # for MAC/Linux
```

See the (env) sign before your command prompt.

4. Install django:

```
pip install django  # or
py -m pip install django
```

5. See installed python packages:

```
pip freeze

# you will see:
#
# asgiref==3.3.4
# Django==3.2.4
# pytz==2021.1
# sqlparse==0.4.1

# If you see lots of things here, that means there is a problem with your
virtual env activation.
# Activate scripts again!
```

6. Create requirement.txt same level with working directory, send your installed packages to this file, requirements file must be up to date:

```
pip freeze > requirements.txt
```

7. Create project:

```
django-admin startproject clarusway
django-admin startproject clarusway .
# With . it creates a single project folder.
# Avoiding nested folders.

# Another naming as "main":
django-admin startproject main .
```

Various files has been created inside project folder "clarusway"!

- manage.py - A command-line utility that allows you to interact with your Django project
- __init__.py - An empty file that tells Python that the current directory should be considered as a Python package
- settings.py - Comprises the configurations of the current project like DB connections.
- urls.py - All the URLs of the project are present here
- wsgi.py - This is an entry point for your application which is used by the web servers to serve the project you have created.

**What is pycache?**

When you run a program in python, the interpreter compiles it to bytecode first (this is an oversimplification) and stores it in the __pycache__ folder. If you look in there you will find a bunch of files sharing the names of the .py files in your project's folder, only their extensions will be either .pyc or .pyo. These are bytecode-compiled and optimized bytecode-compiled versions of your program's files, respectively.

As a programmer, you can largely just ignore it... All it does is make your program start a little faster. When your scripts change, they will be recompiled, and if you delete the files or the whole folder and run your program again, they will reappear (unless you specifically suppress that behavior).

When you're sending your code to other people, the common practice is to delete that folder, but it doesn't really matter whether you do or don't. When you're using version control (git), this folder is typically listed in the ignore file (.gitignore) and thus not included.

8. (Optional) If you created your project wihthout ".", change the name of the project main directory as src to distinguish from subfolder with the same name:

```
# optional
# Be careful to use your own folder names.
mv .\clarusway\ src
```

9. Lets create first application:

Go to the same level with manage.py file:

```
cd .\src\
```

Start app

```
# Using app name as "firstapp"
python manage.py startapp firstapp  # or
py -m manage.py startapp firstapp

# Another naming:
python manage.py startapp home
py -m manage.py startapp home
```

10. Go to clarusway/settings.py and add under INSTALLED_APPS:

```
'firstapp.apps.FirstappConfig',  # or shortly;
# 'firstapp',
```

11. Run our project:

```
python manage.py runserver  # or
py -m manage.py runserver
```

Go to http://localhost:8000/home/ in your browser, and you should see Django rocket, which means you successfully created project.

The install worked successfully! Congratulations!

## Part 2 - Add Simple View and Say Hello World!

12. Go to views.py under "firstapp" directory, and create first view as "home" by adding:

```
from django.http import HttpResponse

def home(request):
    return HttpResponse("<h1>Hello world!</h1>")
```

13. Include URL path of the new app to the project url list, go to urls.py and add:

```
from django.urls import include

# and:
path("", include("firstapp.urls"))
# to the urlpatterns
```

14. Add urls.py file under firstapp directory and add:

```
from django.urls import path
from . import views

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

15. Run our project again to see our view:

```
python manage.py runserver
py -m manage.py runserver
```

16. Go to http://localhost:8000/home/ in your browser, and you should see the text “Hello, world.”, which you defined in the home view.

To stop the server use "CTRL + C"

☺ **Thanks for Attending** ✍

Clarusway