In this article, we went through the creation of the traditional 'Hello, world!' app with Django.

Ref: <https://djangocentral.com/create-a-hello-world-django-application/>

# **Create a Virtual Environment**

A virtual environment is a self-contained directory tree that contains dependencies required by different projects isolated to existing packages. By using virtual Python environments, applications can run in their own 'sandbox' in isolation of other Python applications. This way modules that won’t disrupt any of your other projects.

Let us create a virtual environment for our Hello World project.

*python -m venv virtualenv*

*cd virtualenv*

*python -m pip install --user virtualenv*

*Scripts\activate.bat*

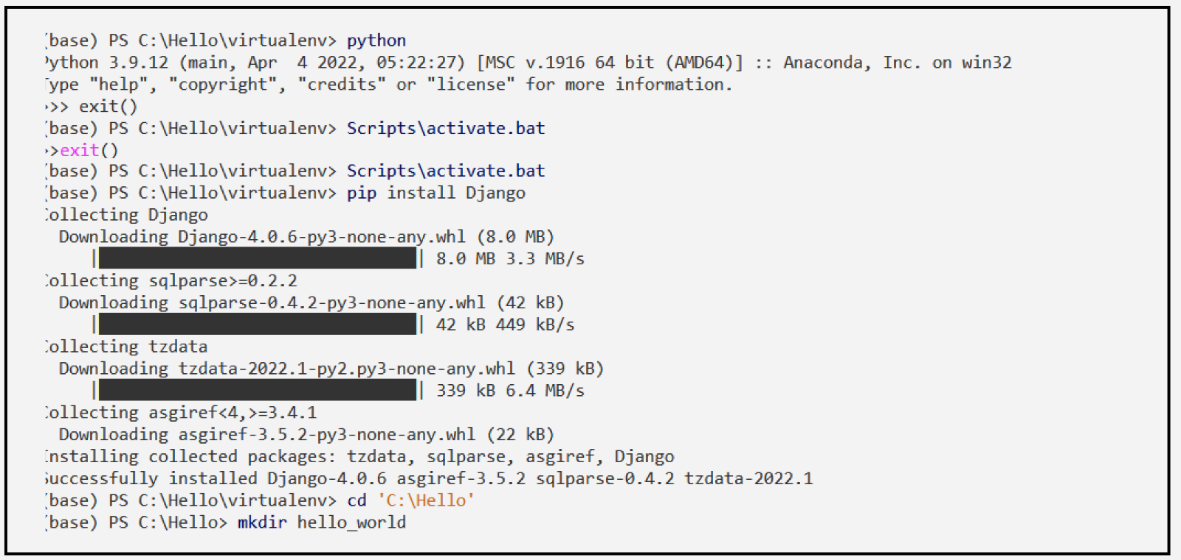


Now you should see (virtualenv) prefixed in your terminal, which indicates that the virtual environment is successfully activated, if not then go through the guide again.

# **Install Django in the virtual environment**

We need to Install Django on our virtual environment.

*pip install Django*



# **Create a Django project**

A Django project is a set of applications and configurations which combined make a full-fledged web application. Django apps are the sub-directories inside the Django project. The purpose of Django applications is to perform a particular task which in this case is to render 'Hello, World!'.

We will create the traditional "Hello, World!" app, which will basically display the string 'Hello, world!' in the browser.

First, create a directory named, hello\_world in the directory, C:\Hello and navigate into it.

*cd 'C:\Hello'*

*mkdir hello\_world*

*cd hello\_world*

*django-admin startproject hello\_world\_project*

Executing the command, ‘django-admin startproject hello\_world\_project’, will invoke the django-admin.py script which will set up a new Django project instance name hello\_world\_project in the hello\_world directory.

**hello\_world\_project**/

**manage.py** - *Command line utility lets you interact with your Django project.*

**hello\_world\_project/**

**\_\_init\_\_.py** - *a blank Python script whose presence indicates to the Python interpreter that the directory is a Python package.*

**settings.py -** *Contains the configuration settings for the Django project.*

**urls.py -** *Contains URL patterns for the Django project.*

**wsgi.py -** *Contains WSGI configuration properties for the Django project.*

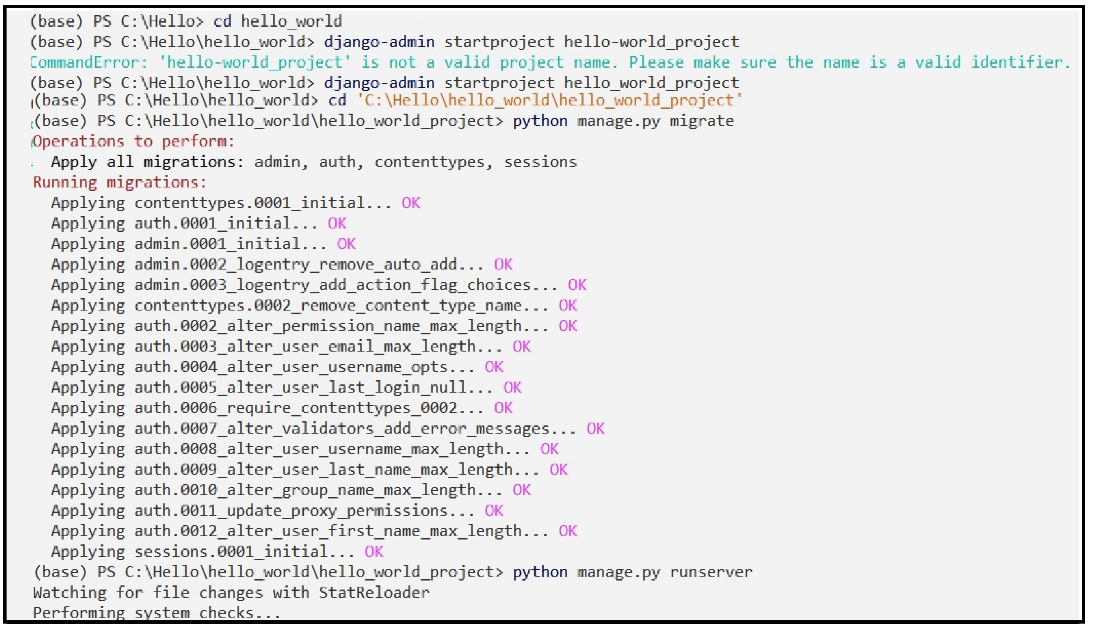
Now, let's apply migrations and test our project. Navigate into the Base directory (i.e., the outer directory) and run these commands.

*cd 'C:\Hello\hello\_world\hello\_world\_project'*

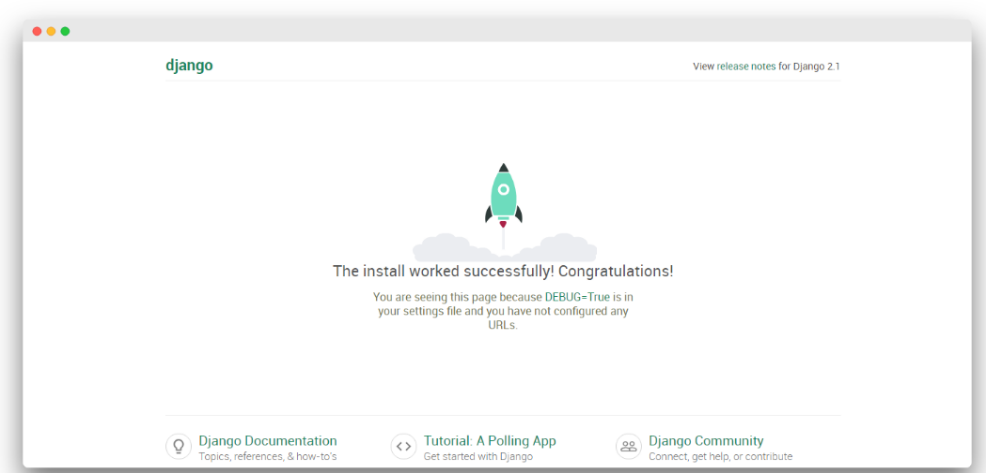
*python manage.py migrate\**

*python manage.py runserver*

Django python \*manage.py migrate executes those SQL commands in the database file. So, after executing migrate all the tables of your installed apps are created in your database file.



This will start the Django's built-in server now open your preferred browser and navigate to this address http://127.0.0.1:8000/ if everything went well you should see the default Django's welcome page.



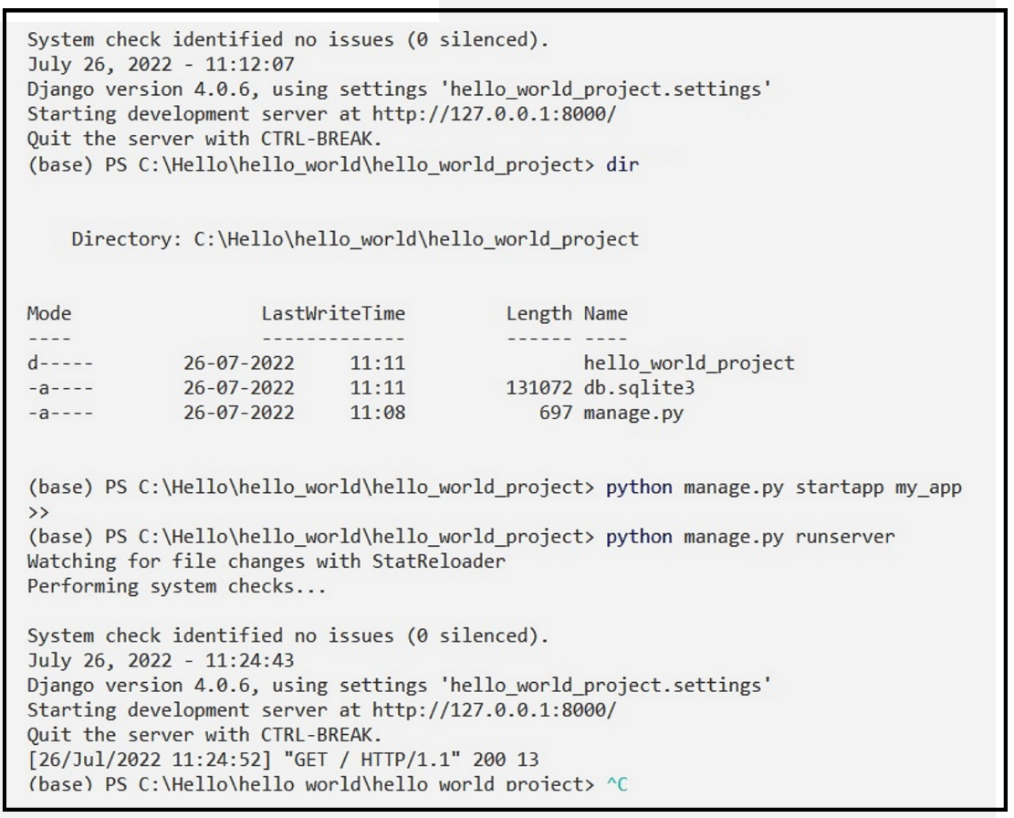
Now press ctrl+c in the terminal window, to stop the server.

# **Creating A Django App**

A Django project is a set of applications and configurations which combined make a full-fledged web application. Django apps are the sub-directories inside the Django project. The purpose of Django applications is to perform a particular task which in this case it to render 'Hello World'.

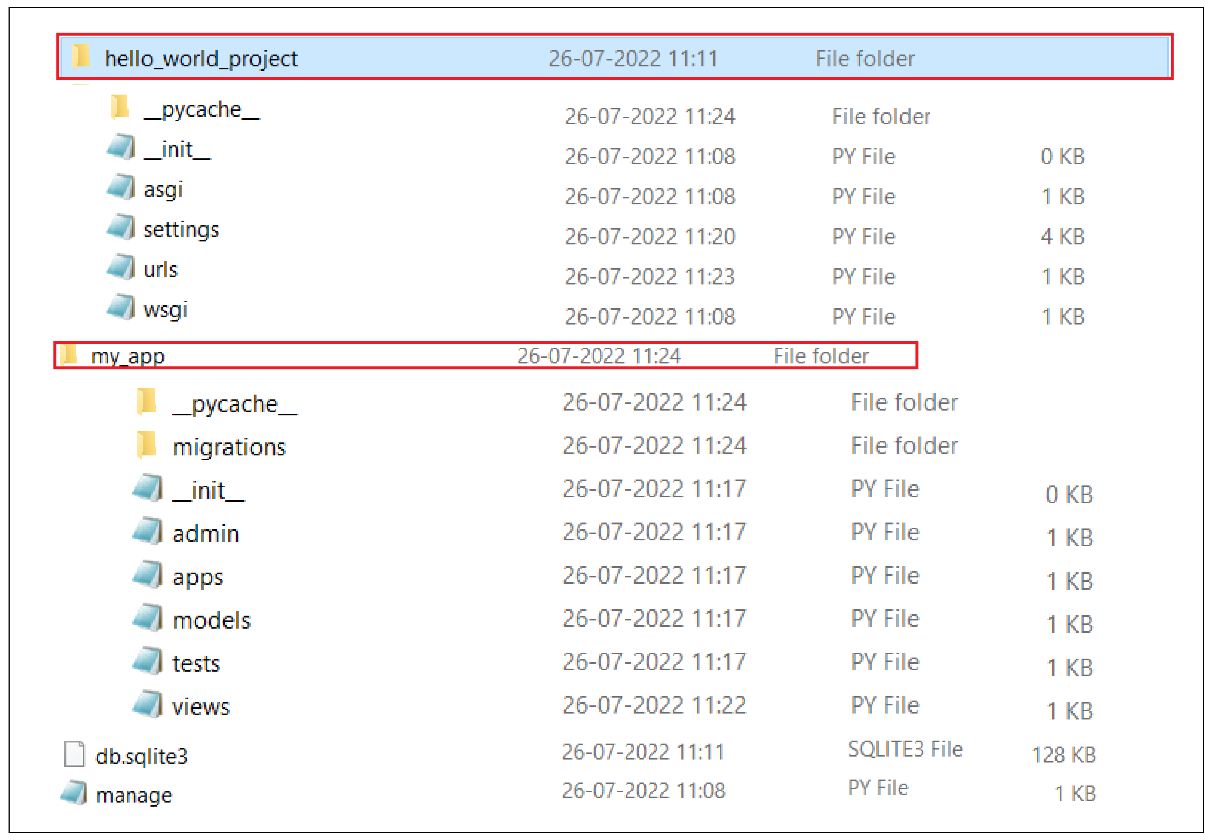
Make sure you are at the outer directory (C:\Hello\hello\_world\hello\_world\_project) where manage.py is and run the following:

*python manage.py startapp my\_app*



Browse at 'http://127.0.0.1:8000/' to see the message "The install worked successfully! Congratulations!"

This will create another directory inside the project called my\_app, now the project should look something like this.



There are lots of new files in the apps which we haven't seen before in the project,

**admin.py** - File with admin definitions for the app - such definitions are needed to access model class instances from the Django admin

**apps.py** - File with configuration parameters for the app.

**models.py** - File with database definitions (i.e., model classes) for the app.

**tests.py** - File with test definitions for the app.

**views.py** - File with view definitions (i.e., controller methods) for the app.

**migrations** - The directory that contains migrations applied to the app’s database definitions (i.e., model classes).

Now we need to add this app into the Installed apps list in Django's settings so that Django can know about the app. Open your preferred text editor and open the settings.py file and scroll to the INSTALLED\_APPS section. There you should see the list of built-in Django apps.

Open the settings.py file and scroll to the INSTALLED\_APPS section.

Add 'my\_app' below the preinstalled apps and save it.

# Application definition

INSTALLED\_APPS = [

'django.contrib.admin',

'django.contrib.auth',

'django.contrib.contenttypes',

'django.contrib.sessions',

'django.contrib.messages',

'django.contrib.staticfiles',

'my\_app',

]

## **Creating Web App**

Till now everything was about the configuration which is needed to be done for any web app, now it's time to actually design the app. Suppose you were creating a hello world app without a framework you'd simply type Hello world into a text file, call it hello.html, and upload it to a directory on a web server somewhere. Notice in this process you've specified two key pieces of information about that web page: its contents (the string Hello world) and its URL (for example, <http://www.example.com/hello.html>).

With Django, you specify those same two things, but in a different manner. The view function produces the contents of the page in the views.py file and the URL is specified in urls.py file.

Let's create our first view, open views.py file of my\_app and add the below lines.

###

*from django.http import HttpResponse*

*def index(request):*

*return HttpResponse('Hello, World!')*

###

First, we imported the HttpResponse class from django.http module then we made a function that takes in a request and returns a HttpResponse object i.e. the string 'Hello, World!'. Note that every view function must take atleast one parameter by convention called request.

In order to see this view in our browser, we need to map this view in our URL configurations. We need to edit urls.py file of the main project to instruct to Django explicitly that we need to activate the view for a particular URL. Now, the urls.py should look like this.

###

*from django.contrib import admin*

*from django.urls import path*

*# imported views*

*from my\_app import views*

*urlpatterns = [*

*path('admin/', admin.site.urls),*

*# configured the url*

*path('',views.index, name="homepage")*

*]*

###

First, we imported the views from my\_app directory then in the URL patterns we added the path for the view which is the homepage hence blank string denoted with ' ' than we mapped this URL to our index view, and at last the optional argument name which we assign to homepage. This implies every request to the homepage should return the 'Hello, world!' string.

Now let's test out our and run the development server.

*python manage.py runserver*

Now visit http://127.0.0.1:8000/ you should see Hello, World! written there.

