Django is a back-end server side web framework.

Django is free, not open source and written in Python.

Django makes it easier to build web pages using Python.

How does Django Work?

Django follows the MVT design pattern (Model View Template).

* Model - The data you want to present, usually data from a database.
* View - A request handler that returns the relevant template and content - based on the request from the user.
* Template - A text file (like an HTML file) containing the layout of the web page, with logic on how to display the data.
* Django Requires Python
* To check if your system has Python installed, run this command in the command prompt:
* python --version
* If Python is installed, you will get a result with the version number, like this
* Python 3.9.2
* If you find that you do not have Python installed on your computer, then you can download it for free from the following website: <https://www.python.org/>
* PIP
* To install Django, you must use a package manager like PIP, which is included in Python from version 3.4.
* To check if your system has PIP installed, run this command in the command prompt:
* pip --version
* If PIP is installed, you will get a result with the version number.
* For me, on a windows machine, the result looks like this:
* pip 20.2.3 from c:\python39\lib\site-packages\pip (python 3.9)
* Django - Create Virtual Environment
* Virtual Environment
* It is suggested to have a dedicated virtual environment for each Django project, and one way to manage a virtual environment is [venv](https://docs.python.org/3/tutorial/venv.html" \t "_blank), which is included in Python.
* The name of the virtual environment is your choice, in this tutorial we will call it myworld.
* Type the following in the command prompt, remember to navigate to where you want to create your project:
* Windows:
* py -m venv myworld
* Unix/MacOS:
* python -m venv myworld
* This will set up a virtual environment, and create a folder named "myworld" with subfolders and files, like this:
* myworld  
    Include  
    Lib  
    Scripts  
    pyvenv.cfg
* Then you have to activate the environment, by typing this command:
* Windows:
* myworld\Scripts\activate.bat
* Unix/MacOS:
* source myworld/bin/activate
* Once the environment is activated, you will see this result in the command prompt:
* Windows:
* (myworld) C:\Users\*Your Name*>
* Unix/MacOS:
* (myworld) ... $
* **Note:** You must activate the virtual environment every time you open the command prompt to work on your project.
* Install Django
* Now, that we have created a virtual environment, we are ready to install Django.
* **Note:** Remember to install Django while you are in the virtual environment!
* Django is installed using pip, with this command:
* Windows:
* (myworld) C:\Users\*Your Name*>py -m pip install Django
* Unix/MacOS:
* (myworld) ... $ python -m pip install Django
* Which will give a result that looks like this (at least on my Windows machine):
* Collecting Django  
    Downloading Django-4.0.3-py3-none-any.whl (8.0 MB)  
        |████████████████████████████████| 8.0 MB 2.2 MB/s  
  Collecting sqlparse>=0.2.2  
    Using cached sqlparse-0.4.2-py3-none-any.whl (42 kB)  
  Collecting asgiref<4,>=3.4.1  
    Downloading asgiref-3.5.0-py3-none-any.whl (22 kB)  
  Collecting tzdata; sys\_platform == "win32"  
    Downloading tzdata-2021.5-py2.py3-none-any.whl (339 kB)  
        |████████████████████████████████| 339 kB 6.4 MB/s  
  Installing collected packages: sqlparse, asgiref, tzdata, Django  
  Successfully installed Django-4.0.3 asgiref-3.5.0 sqlparse-0.4.2 tzdata-2021.5  
  WARNING: You are using pip version 20.2.3; however, version 22.3 is available.  
  You should consider upgrading via the 'C:\Users\*Your Name*\myworld\Scripts\python.exe -m pip install --upgrade pip' command.
* That's it! Now you have installed Django in your new project, running in a virtual environment!
* Windows, Mac, or Unix?
* You can run this project on either one. There are some small differences, like when writing commands in the command prompt, Windows uses py as the first word in the command line, while Unix and MacOS use python:
* Windows:
* py --version
* Unix/MacOS:
* python --version
* Check Django Version
* You can check if Django is installed by asking for its version number like this:
* (myworld) C:\Users\*Your Name*>django-admin --version
* If Django is installed, you will get a result with the version number:
* 4.1.2
* Django Create Project
* My First Project
* Once you have come up with a suitable name for your Django project, like mine: my\_tennis\_club, navigate to where in the file system you want to store the code (in the virtual environment), I will navigate to the myworld folder, and run this command in the command prompt:
* django-admin startproject my\_tennis\_club
* Django creates a my\_tennis\_club folder on my computer, with this content:
* my\_tennis\_club  
      manage.py  
      my\_tennis\_club/  
          \_\_init\_\_.py  
          asgi.py  
          settings.py  
          urls.py  
          wsgi.py
* These are all files and folders with a specific meaning, but for now, it is more important to know that this is the location of your project, and that you can start building applications in it.
* Run the Django Project
* Now that you have a Django project, you can run it, and see what it looks like in a browser.
* Navigate to the /my\_tennis\_club folder and execute this command in the command prompt:
* py manage.py runserver
* Which will produce this result:
* Watching for file changes with StatReloader  
  Performing system checks...  
    
  System check identified no issues (0 silenced).  
    
  You have 18 unapplied migration(s). Your project may not work properly until you apply the migrations for app(s): admin, auth, contenttypes, sessions.  
  Run 'python manage.py migrate' to apply them.  
  October 27, 2022 - 13:03:14  
  Django version 4.1.2, using settings 'my\_tennis\_club.settings'  
  Starting development server at http://127.0.0.1:8000/  
  Quit the server with CTRL-BREAK.
* Open a new browser window and type [127.0.0.1:8000](http://127.0.0.1:8000/) in the address bar.
* The result:
* A screenshot of a computer

  Description automatically generated
* What's Next?
* We have a Django project!
* The next step is to make an app in your project.
* You cannot have a web page created with Django without an app.
* Django Create App
* What is an App?
* An app is a web application that has a specific meaning in your project, like a home page, a contact form, or a members database.
* In this tutorial we will create an app that allows us to list and register members in a database.
* But first, let's just create a simple Django app that displays "Hello World!".
* Create App
* I will name my app members.
* Start by navigating to the selected location where you want to store the app, in my case the my\_tennis\_club folder, and run the command below.
* If the server is still running, and you are not able to write commands, press [CTRL] [BREAK], or [CTRL] [C] to stop the server and you should be back in the virtual environment.
* py manage.py startapp members
* Django creates a folder named members in my project, with this content:
* my\_tennis\_club  
      manage.py  
      my\_tennis\_club/  
      members/  
          migrations/  
              \_\_init\_\_.py  
          \_\_init\_\_.py  
          admin.py  
          apps.py  
          models.py  
          tests.py  
          views.py
* These are all files and folders with a specific meaning. You will learn about most of them later in this tutorial.
* First, take a look at the file called views.py.
* This is where we gather the information we need to send back a proper response.
* Django Views
* Views
* Django views are Python functions that take http requests and return http response, like HTML documents.
* A web page that uses Django is full of views with different tasks and missions.
* Views are usually put in a file called views.py located on your app's folder.
* There is a views.py in your members folder that looks like this:
* my\_tennis\_club/members/views.py:
* from django.shortcuts import render
* # Create your views here.
* Find it and open it, and replace the content with this:
* my\_tennis\_club/members/views.py:
* from django.shortcuts import render
* from django.http import HttpResponse
* def members(request):
* return HttpResponse("Hello world!")
* **Note:** The name of the view does not have to be the same as the application.
* I call it members because I think it fits well in this context.
* This is a simple example on how to send a response back to the browser.
* But how can we execute the view? Well, we must call the view via a URL.
* Django URLs
* URLs
* Create a file named urls.py in the same folder as the views.py file, and type this code in it:
* my\_tennis\_club/members/urls.py:
* from django.urls import path
* from . import views
* urlpatterns = [
* path('members/', views.members, name='members'),
* ]
* The urls.py file you just created is specific for the members application. We have to do some routing in the root directory my\_tennis\_club as well. This may seem complicated, but for now, just follow the instructions below.
* There is a file called urls.py on the my\_tennis\_club folder, open that file and add the include module in the import statement, and also add a path() function in the urlpatterns[] list, with arguments that will route users that comes in via 127.0.0.1:8000/.
* Then your file will look like this:
* my\_tennis\_club/my\_tennis\_club/urls.py:
* from django.contrib import admin
* from django.urls import include, path
* urlpatterns = [
* path('', include('members.urls')),
* path('admin/', admin.site.urls),
* ]

* If the server is not running, navigate to the /my\_tennis\_club folder and execute this command in the command prompt:
* py manage.py runserver
* In the browser window, type [127.0.0.1:8000/members/](http://127.0.0.1:8000/members/) in the address bar.
* A screenshot of a computer

  Description automatically generated
* Django Templates

Templates

In the Django Intro page, we learned that the result should be in HTML, and it should be created in a template, so let's do that.

Create a templates folder inside the members folder, and create a HTML file named myfirst.html.

The file structure should be like this:

my\_tennis\_club

manage.py

my\_tennis\_club/

members/

templates/

myfirst.html

Open the HTML file and insert the following:

my\_tennis\_club/members/templates/myfirst.html:

<!DOCTYPE html>

<html>

<body>

<h1>Hello World!</h1>

<p>Welcome to my first Django project!</p>

</body>

</html>

Modify the View

Open the views.py file and replace the members view with this:

my\_tennis\_club/members/views.py:

from django.http import HttpResponse

from django.template import loader

def members(request):

template = loader.get\_template('myfirst.html')

return HttpResponse(template.render())

Change Settings

To be able to work with more complicated stuff than "Hello World!", We have to tell Django that a new app is created.

This is done in the settings.py file in the my\_tennis\_club folder.

Look up the INSTALLED\_APPS[] list and add the members app like this:

my\_tennis\_club/my\_tennis\_club/settings.py:

INSTALLED\_APPS = [

'django.contrib.admin',

'django.contrib.auth',

'django.contrib.contenttypes',

'django.contrib.sessions',

'django.contrib.messages',

'django.contrib.staticfiles',

'members'

]

Then run this command:

py manage.py migrate

Which will produce this output:

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 admin.0003\_logentry\_add\_action\_flag\_choices... 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 auth.0009\_alter\_user\_last\_name\_max\_length... OK

Applying auth.0010\_alter\_group\_name\_max\_length... OK

Applying auth.0011\_update\_proxy\_permissions... OK

Applying auth.0012\_alter\_user\_first\_name\_max\_length... OK

Applying sessions.0001\_initial... OK

(myworld) C:\Users\Your Name\myworld\my\_tennis\_club>

Start the server by navigating to the /my\_tennis\_club folder and execute this command:

py manage.py runserver

In the browser window, type 127.0.0.1:8000/members/ in the address bar.

The result should look like this:

A screenshot of a computer

Description automatically generated

Django Models

Up until now in this tutorial, output has been static data from Python or HTML templates.

Now we will see how Django allows us to work with data, without having to change or upload files in the process.

In Django, data is created in objects, called Models, and is actually tables in a database.

Create Table (Model)

To create a model, navigate to the models.py file in the /Inventory/ folder.

Open it, and add a Member table by creating a Member class, and describe the table fields in it:

DjangoProject/Inventory/models.py:

from django.db import models

class Mytable(models.Model):

address = models.CharField(max\_length=255)

pincode = models.CharField(max\_length=255)

The first field, firstname, is a Text field, and will contain the first name of the members.

The second field, lastname, is also a Text field, with the member's last name.

Both firstname and lastname is set up to have a maximum of 255 characters.

SQLite Database

When we created the Django project, we got an empty SQLite database.

It was created in the DjangoProject root folder, and has the filename db.sqlite3.

By default, all Models created in the Django project will be created as tables in this database.

Migrate

Now when we have described a Model in the models.py file, we must run a command to actually create the table in the database.

Navigate to the /DjangoProject/ folder and run this command:

python manage.py makemigrations Inventory

Which will result in this output:

Migrations for 'members':  
  Inventory\migrations\0001\_initial.py  
    - Create model Member  
  
(myworld) C:\Users\*Your Name*\myworld\my\_tennis\_club>

Django creates a file describing the changes and stores the file in the /migrations/ folder:

DjangoProject/InVENTORY/migrations/0001\_initial.py:

# Generated by Django 5.1.7 on 2025-03-20 11:39

from django.db import migrations, models

class Migration(migrations.Migration):

initial = True

dependencies = [

]

operations = [

migrations.CreateModel(

name='Member',

fields=[

('id', models.BigAutoField(auto\_created=True, primary\_key=True, serialize=False, verbose\_name='ID')),

('firstname', models.CharField(max\_length=255)),

('lastname', models.CharField(max\_length=255)),

],

),

]

Note that Django inserts an id field for your tables, which is an auto increment number (first record gets the value 1, the second record 2 etc.), this is the default behavior of Django, you can override it by describing your own id field.

The table is not created yet, you will have to run one more command, then Django will create and execute an SQL statement, based on the content of the new file in the /migrations/ folder.

Run the migrate command:

python manage.py migrate

Which will result in this output:

Operations to perform:  
  Apply all migrations: admin, auth, contenttypes, members, sessions  
Running migrations:  
  Applying members.0001\_initial... OK  
  
(myworld) C:\Users\*Your Name*\myworld\my\_tennis\_club>

Now you have a Member table in you database!

View SQL

As a side-note: you can view the SQL statement that were executed from the migration above. All you have to do is to run this command, with the migration number:

python manage.py sqlmigrate Inventory 0001

Which will result in this output:

BEGIN;  
--  
-- Create model Member  
--  
CREATE TABLE "members\_member" ("id" integer NOT NULL PRIMARY KEY AUTOINCREMENT, "firstname" varchar(255) NOT NULL, "lastname" varchar(255) NOT NULL);  
COMMIT;