

Django is a Python web framework that encourages rapid development.

It takes care of much of the hassle of web development, so you can focus on writing your app without needing to reinvent the wheel. It’s free and open source.

We’ll write out backend with Django.

Important commands

* manage.py is stored in backend/habit\_tracker

|  |  |
| --- | --- |
| Run the Django server | python manage.py runserver |
| Creates new applications | python manage.py startapp habit\_tracker\_app |
| Creates new project | django-admin startproject habit\_tracker |
|  |  |
|  |  |
|  |  |

<https://dev.to/ericchapman/my-beloved-django-cheat-sheet-2056>

Setup venv

pip install virtualenvwrapper-win

mkvirtualenv habit-tracker

deactivate

workon habit-tracker

---

**Virtual Environment** [<https://pypi.org/project/virtualenvwrapper-win/>]

A tool for creating isolated Python virtual environments, each with their own libraries and site-packages.

---

Django installation

pip install django

Start a new Django project

django-admin startproject habit\_tracker

---

**django-admin** [<https://docs.djangoproject.com/en/4.1/ref/django-admin/>]

django-admin is Django’s command-line utility for administrative tasks

manage.py is automatically created in each Django project. It does the same thing as django-admin but also sets the DJANGO\_SETTINGS\_MODULE environment variable so that it points to your project’s settings.py file.

---

Directory structure:

backend

* project\_name
  + Text

    Description automatically generated project\_name
  + Text

    Description automatically generated app\_name
  + Shape

    Description automatically generated with low confidence db.sqlite3
  + Icon

    Description automatically generated manage.py

# Settings.py

Contains project settings (duh), such as:

* BASE\_DIR
* SECRET\_KEY (Move it after creation)
* DEBUG flag
* INSTALLED\_APPS (3rd party apps, and own apps)
* TEMPLATES (Directories that contains them and options)
* DATABASES (defaults to sqlite3)
* Etc…

To update settings, run **python manage.py migrate**

# Admin view

<http://127.0.0.1:8000/admin/login/?next=/admin/login>

Create mega user: python manage.py createsuperuser

The user is stored in the DB

# Models.py

Create classes there with attributes (for example, if the class is ‘Product’, the attributes might be name, price, stock,…)

class Product(models.Model):

name = models.TextField()

price = models.TextField(default=’30$’)

This file belongs to an app we created.

We can update the Settings.py file (Project settings) to contain said app (INSTALLED\_APPS), and then to apply changes in models.py, run

> python manage.py makemigrations

> python manage.py migrate

Something is stored in migrations/\_\_pycache\_\_

# Admin.py

Import the model

from .models import Product

admin.site.register(Product)

Now in the admin view (<http://127.0.0.1:8000/admin/login/?next=/admin/login>) we can add products and save them in the DB

Tahoma

# Create Objects in the Python Shell

> python manage.py shell

> from product.models import Product

> Product.objects.all()

> Product.objects.create(name=’new\_product’, price=’50$’)

> Product.objects.all()

# ANOTHER

Tahoma

# ANOTHER

Tahoma

# ANOTHER

Tahoma

# ANOTHER

Tahoma

# URL Routing and Django Apps

Start the Django application

cd habit\_tracker

python manage.py startapp habit\_tracker\_app

Explanation about some files

* C:\Code\git\habit-tracker\backend\habit\_tracker\**habit\_tracker\_app\urls.py**

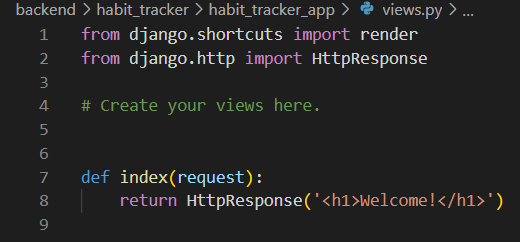
This file contains urls that will be used in our app.

Text

Description automatically generated

* C:\Code\git\habit-tracker\backend\habit\_tracker\**habit\_tracker\_app\views.py**

This file contains functions that will be used when the pages are viewed



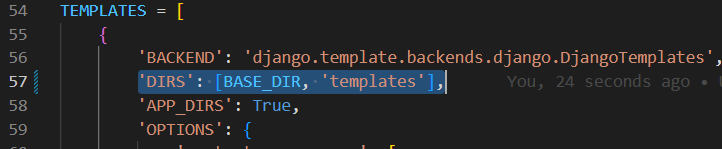
Graphical user interface, application

Description automatically generatedAnd when starting the app with python .\habit\_tracker\manage.py runserver, we get:

# Django Template Language

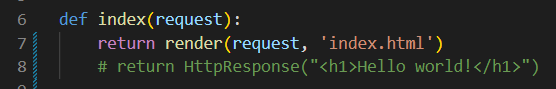
Create directory: backend/habit\_tracker/templates

Edit file backend/habit\_tracker/habit\_tracker/settings.py, add the newly created directory



Create file backend/habit\_tracker/templates/**index.html** and edit it to contain basic html code (<h1>…</h1>)

Update the index function (in the views.py file) to use that template



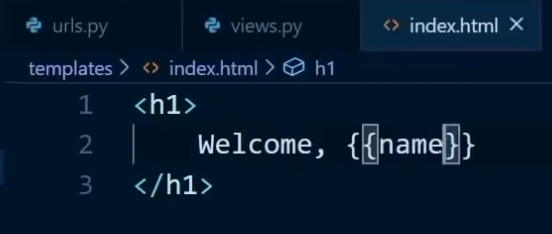
BUT WHAT IS A TEMPLATE?

A template contains the static parts of the desired HTML output as well as some special syntax describing how dynamic content will be inserted

<https://docs.djangoproject.com/en/4.1/topics/templates/>

# Sending Data to Template File

Update the function in views.py and the template such that data will be sent by the function as input to the template



Graphical user interface, text, application

Description automatically generated