WEB APPLICATION DEVELOPMENT WITH DJANGO

TUTORIAL BY NALUGYA VANESSA.

Steps to follow by windows users;

By the end of this tutorial, you'll have the skills and knowledge to build your own dynamic and powerful web applications using the Django framework.

Django is a fantastic tool that empowers developers like you to bring your creative ideas to life. Whether you're new to programming or already have some experience, this tutorial is designed to be accessible and informative, guiding you step by step through the process of setting up your development environment, creating models, views, and templates, and ultimately deploying your application.

Thank you for choosing to learn with me, and I'm excited to be your guide on this journey. Let's get started and create something amazing together!

Nb: You should have python installed on your device so that you can continue with the rest of the commands.

The first order of business is to download the latest version of Python. The

version that is available at the time of writing this tutorial is Python 3.11.4 so we

will be using that. For any future versions, there is a good chance all of the

commands and features will work exactly the same unless the community

decides to change the way the language works radically.

Let us begin by opening up our web browser and going straight to the source.

In the address bar, type in www.python.org, and you will be greeted by a

simplistic website.

Hover the mouse cursor over ‘Downloads’, and the website should be able to

detect your platform and present the corresponding version accordingly

automatically. Click on the button to commence the download.

If it doesn’t, simply click on the name of the platform you are using to be taken to the downloads page. Here, click on the first option that says “Latest

Python 3 Release – Python 3.11.4” and download that.

Once the file has been downloaded, simply run it to install Python within

your system. This should not take more than a minute or so. Now run the

command prompt/terminal once again and type in ‘python’ to see the version

of Python installed within your machine. This, therefore, confirms that the

installation went well.

a. Configuration of the virtual environment.

These commands are to be run from the command prompt.

- We create a folder where all our Django projects are going to be, and this can be anywhere you want stored by a command **"mkdir folder name"**
- Installing a virtual environment with command **"pip install virtualenv".**
- You create a sub folder within the main folder by a command **"mkdir then folder name".**
- Then cd into that sub folder and specify the name of the virtualenv by a command **"virtualenv venv".**
- Activate the virtualenv by a command **"venv\Scripts\activate".**
- If you want to deactivate you use **"venv\Scripts\deactivate.bat"**

b. Configuration of Django.

- ✓ We use a command **"pip install django==(version) .e.g 3.0.8"** to install django.
- ✓ We then create the project with a command **"django-admin startproject name of the project"** e.g., spare part.
- ✓ We then cd into that project created e.g cd spare part.
- ✓ We then make migrations by **"python manage.py migrate"** This helps us to bring tables that come with django into our database.

- ✓ We then create a superuser(admin) by a command **"python manage.py createsuperuser".**
- ✓ Here we create an application by a command **"python manage.py startapp name of the app"** e.g berealapp.
- ✓ **N.b You can have one or more applications in a project.**
- ✓ After appending the application run the server to see if our installations are working very well by **"python manage.py runserver"**.
- ✓ We install crispy forms which helps us to create forms in our project by **"pip install django-crispy-forms".**
- ✓ We install bootstrap4 for displaying crispy forms by **"pip install crispy-bootstrap4."**
- ✓ We then install bootstrap5 by **"pip install django-bootstrap-v5".** For displaying our bootstrap content.
- ✓ We also install filters for activating the search fields by **"pip install django-filter".**

c. Settings adjustments.

- We go to settings in installed apps to append the app we created.
- We go to settings to append to the downloads e.g **crispy_forms.**

- We go to settings to append to the downloads e.g **crispy_bootstrap4.**
- We go to settings to append to the downloads e.g **bootstrap5.**
- We go to settings to append to the downloads e.g **django_filters.**
- **STATIC_ROOT** for the project static directory
- **STATICFILES_DIR** for the application static directory.
- We also put there redirections for our logging in and logging out.

d. Configuring our templates and static folders.

- We create our templates folder in the project folder (they should be on the same line as the duplicated folder). This stores our html files.
- We create our static folder in the project folder (they should be on the same line as the duplicated folder).
- We create another static folder in our application where we going to store our images, CSS and js folders with their files.

e. Configuring master(project) urls.(urls.py)

- We go to the list of urls patterns to include a path for our application urls to be accessed.
- **N.b** don't forget to add **"include"** after the word **"path".**

f. Configuring our application.

- We create a urls.py file in our application.
- We edit our urls.py file by importing and then defining our paths for our url patterns [] to be accessed by the views.
- We also create a filter.py file to specify our fields to be filtered.
- We also create our forms.py in our application.

g. Configuring our views in our application.

- The views handle the **request** and then gives us back a **response** from the templates. E.g 200 (ok),404(page not found),303,503 etc.

- We define views as a function, so a view is a function that handles a request to give back a response.

h. Configuring models.py

N.b Models help us to create tables where our data is going to be stored in the database.

- ★ We import some packages from django e.g time zone
- ★ We then create classes e.g class Country(models.Model).
- ★ We then give it attributes e.g name.
- ★ We then define a function __str__ to enable other classes we create to access attributes for that particular class as foreign key.
- ★ We then makemigrations and then migrating in the command line.
- ★ This is done by **python manage.py makemigrations** and **python manage.py migrate** respectively.

i. Configuring admin.py

- ❑ We register the models we use in admin.py so that they can be reflected on the admin interface and the database.
- ❑ We import our models in admin.

Notes to remember: A browser cannot communicate with the database directly.

And a server cannot talk directly to the database because it's dumb.

So, a back-end language as Django creates a bridge between the database and the server to communicate.

A browser cannot post data into the database directly, so we use a form to post data.

And the crsf **(Cross-site Request Forgery)** helps us to give security to our data in forms so that it can't be tempered with, and it comes with django.

Thank you for following the tutorial and I strongly know that if you follow the tutorial very well, web development with django will be a walk over.