DJNAGO STEPS

PINESPHERE

STEP 1. Create a folder as "empproject":

- Command: **`mkdir empproject`**
- Documentation: This command creates a directory named "empproject" in your current location.

STEP 2. Open terminal - change directory to "myproject":

Command: **cd myproject**

Documentation: Change the current working directory to the "myproject" directory.

STEP 3. Creating a Virtual Environment:

Command: **python -m venv envname**

Documentation: This command creates a virtual environment with the name "envname." A virtual environment is an isolated environment where you can install project-specific dependencies without interfering with the system-wide Python installation.

STEP 4. Activating the Virtual Environment:

Command: **envname\Scripts\activate**

Documentation: This command activates the virtual environment, allowing you to work within the isolated environment.

STEP 5. Django Installation:

Command: **pip install django==3.2**

Documentation: This command installs Django version 3.2 in the virtual environment. Django is a Python web framework for building web applications.

STEP 6.Djongo Installation:

Command: **pip install djongo**

Documentation: This command installs Djongo, which is a SQL to MongoDB query compiler for Django.

STEP 7. Pymongo Installation:

Command: **pip install pymongo==3.12.1**

Documentation: This command installs pymongo version 3.12.1, which is a Python driver for MongoDB.

STEP 8. Creating a Django Project:

Command: **django-admin startproject projectname**

Documentation: This command creates a new Django project with the specified name, "projectname."

STEP 9. Change directory to the project folder:

Command: **cd projectname**

Documentation: Change the current working directory to the newly created Django project folder.

STEP 10. Open your Visual Studio Code (VSCode) or preferred code editor.

Documentation: This step involves opening your code editor to work on your Django project.

STEP 11. Modify settings.py:

- In your project's settings.py file, modify the 78th and 79th lines by adding 'djongo' and your database name.

- Documentation:This step involves configuring the database settings in your Django project.

STEP 12. Create and apply database migrations:

- Command: **python manage.py makemigrations and python manage.py migrate**

- Documentation: These commands create and apply database migrations, which are used to update your database schema based on the changes you make to your Django models.

STEP 13. Creating a Django App:

- Command: **python manage.py startapp appname**

- Documentation: This command creates a new Django app within your project with the specified name, "appname."

STEP 14. Modify settings.py to include your app:

- In your project's settings.py file, add your app to the INSTALLED_APPS list.
- Documentation: This step registers your app with your Django project.

STEP 15. Configure URL routing:

- Edit your project's urls.py file to include your app's URLs.
- Documentation:This step allows you to define URL patterns for your app and link them to views.

STEP 16. Create urls.py in your app:

- Create a urls.py file within your app to define the URL patterns for your app's views.
- Documentation:This file defines the URL routing for your app..

STEP 17. Define models in models.py:

- In your app's models.py file, define the data models for your application.
- Documentation:This step involves defining the structure of your database tables using Django models.

STEP 18. Migrations:

- After defining models, you should create and apply migrations to update the database schema.
- Documentation: Migrations ensure that your database structure matches the defined models.

STEP 19. Register models in admin.py:

- In your app's admin.py file, register your models so you can manage them through the Django admin interface.

- Documentation:This step enables you to interact with your app's data through the Django admin panel.

STEP 20. Creating a superuser:

- Command: **python manage.py createsuperuser**

- Documentation: This command creates a superuser account, which allows you to access the Django admin panel and manage your app's data.

By following these steps, you will have set up a Django project, created an app, defined models, configured URL routing, and created a superuser account to manage your application.