Dec, 2023

Django overview

Google Developers Students Clubs
Addis Ababa Science and Technology University

Django overview

Today's Topics:

- What is Django?
- Folder Structure
- Django’s Fundamental Concepts
- Demo: Creating a simple Django App

What is Django?

- A high-level web framework written in python
- Developed to ease the process of building web application
- Follows the “Don’t Repeat Yourself” (DRY) and “Convention Over Configuration” principles
- Ideal for rapid development of web application
- Robust, secure, and scalable
- Django follows the Model-View-Template(MVT) architectural pattern

Why Django?

- Rapid development
  - Automates common web development tasks
  - Enable quick prototyping and deployment
- Clean and Pragmatic design
  - Organized structure with clear separation of concerns
- Built-in security features
- Extensive documentation and community support

Folder Structure of Django project

- **Project vs. App:**
  - **A project contains multiple apps**
  - **An app is a modular component of a project**

```
project/
        ‘manage.py’
        project/
                settings.py
                urls.py
                wsgi.py
                asgi.py
        app1/
                models.py
                views.py
                templates/
        app2/
                ...
```

The folder structure of django looks like this

Django’s Core concepts

1. Model-view-Template(MVT)
    a. Model: Defines the data structure (database schema)
    b. Views: handle user requests and return responses
    c. Templates: present data to the user dynamically
2. ORM: Advanced Django’s Object-Relational Mapping
3. Django’s URL Routing: maps URLs to views
4. Built-in Admin Interface
5. Middleware : process requests globally before the view.
6. Django Forms

Demo Project: Create your first project

Lets create a project called blogger which is going to be a blog project.

First activate the venv and navigate the folder where you want to create the project. Then use the command below;

```
> django-admin startproject blogger
```

This will create the project with following files containing

- manage.py: This is a command-line utility used to interact with your project.

Now lets see the files in blogger directory

- __init__.py an empty file that tells python to treat blogger as a python module.
- asgi.py : used to run our project as Asynchronous Server Gateway Interface application with ASGI - compatible web servers.
- settings.py : This indicates settings and configurations for the project.
- urls.py This is the place where our URLs patterns live.
- wsgi.py : This is the configuration to run our project as a Web Server Gateway Interface application with its relative compatible web server

manage.py Important commands

1. **runserver <host>:<port>** : used to run in the development server
2. **startapp <app_name>:** used to create a new django app.
3. **migrate** : used to create the database for the models for all apps
4. **Migrate <app_name>:** used to migrate for specific app
5. **makemigrations** : create new migration files based on changes
6. **createsuperuser** : used to create a super user for the admin panel
7. **shell** : open an interactive python shell with django env loaded

Project settings

Let's open the settings.py file and look at important configuration

DEBUG is the Boolean that turns the debug mode of the project on and off. If it is on then Django will display detail error pages when exception exists.

ALLOWED_HOSTS is used once you move your site to production and host on your domain/host site.

INSTALLED_APPS this setting tells Django which applications are active for this site.

MIDDLEWARE is a list that contains middleware to be executed.

DATABASE is a dictionary that contains the settings for all the databases to be used in the project

LANGUAGE_CODE defines the default language code for this Django site

USE_TZ tells Django to active/ deactive timezone support.

Creating an application

```
> python manage.py startapp blog
```

Let’s see the files in our blog app

__init__.py treat the blog directory as a python module

admin.py where models registered to include in the admin site

apps.py This include the main configuration of the blog app.

models.py this include the data model of the application

test.py this is where you can add tests for your application

views.py the logic of your application goes here

migrations is a directory that contain database migrations of the app

Creating the blog data models

Let’s create a post model for our blog app by adding attributes of title, slug, and body

```
from django.db import models
class Post(models.Model):
    title = models.CharField(max_length=250)
    slug = models.SlugField(max_length=250)
    body = models.TextField()
    def __str__(self):
        return self.title
```

Adding datetime fields and default order

Let’s add a time fields to the post model. First import timezone from django.utils. Then add the following code on the model

```
    body = models.TextField()
    publish = models.DateTimeField(default=timezone
    created = models.DateTimeField(auto_now_add=True
    updated = models.DateTimeField(auto_now=True)
```

To add the default order of the data create a class named Meta in the post class and add ordering attribute to [“-publish”]

```
    updated = models.DateTimeField(aut
    class Meta:
        ordering = ['-publish']
```

The demo will continue ....

Task

> create an application which is called comment

> create a model which includes content, created time, and modified time.