



DJANGO - PYTHON - Introduction

www.infomaxacademy.com

Introduction to Django

- Django is a high-level, open-source web framework written in Python that encourages rapid development and clean, pragmatic design.
- It follows the Model-View-Controller (MVC) architectural pattern, but in Django, it's often referred to as Model-View-Template (MVT).
- Django's primary goal is to enable developers to build web applications quickly and efficiently, emphasizing the "Don't Repeat Yourself" (DRY) and "Convention over Configuration" principles.

Key components of Django

- Models
- Views
- Templates
- Controller (URLconf)
- Django ORM
- Middleware
- Admin Site
- Forms

Models

- Django provides an Object-Relational Mapping (ORM) system that allows you to define your data models using Python classes.
- These models represent database tables, and Django takes care of the underlying SQL queries for you.

Views

- Views handle the logic behind processing user requests and returning appropriate responses.
- They interact with models to retrieve or store data and pass it to templates

Templates

- Templates define how the data is presented to the user.
- They use Django's own template language, which allows for the dynamic generation of HTML.

Controller (URLconf)

- The URLconf (URL configuration) is responsible for mapping URL patterns to view functions.
- It helps Django determine which view should handle a particular URL

Forms

- Django provides a form handling system that simplifies the process of collecting and validating user input.
- It helps in creating HTML forms and processing form submissions.

Admin Site

- Django includes a built-in admin interface that allows developers to manage database records through a web-based interface automatically generated from model definitions.

Middleware

- Middleware is a way to process requests globally before they reach the view.
- It can be used for various purposes, such as authentication, security, or custom request processing.

Django ORM

- Django's ORM system abstracts away the database management, allowing developers to interact with the database using Python instead of raw SQL.

Install Django

- Install Django

  pip install django.

- Check the version of Django

  python -m django –version

- To run the server

  python manage.py runserver

- to run app in different port
  - Python manage.py runserver <portnumber>

Create a Project

- Use the django-admin startproject projectname command to create a new Django project.
- Ex: django-admin startproject mysite

```
mysite/
    manage.py
    mysite/
        __init__.py
        settings.py
        urls.py
        asgi.py
        wsgi.py
```

Project Structure

- The outer mysite/ root directory is a container for your project. Its name doesn't matter to Django; you can rename it to anything you like.
- manage.py: A command-line utility that lets you interact with this Django project in various ways.
- The inner mysite/ directory is the actual Python package for your project. Its name is the Python package name you'll need to use to import anything inside it (e.g. mysite.urls).
- mysite/__init__.py: An empty file that tells Python that this directory should be considered a Python package.
- mysite/settings.py: Settings/configuration for this Django project. Django settings will tell you all about how settings work.

- mysite/urls.py: The URL declarations for this Django project; a “table of contents” of your Django-powered site. You can read more about URLs in URL dispatcher.
- mysite/asgi.py: An entry-point for ASGI-compatible web servers to serve your project. See How to deploy with ASGI for more details.
- mysite/wsgi.py: An entry-point for WSGI-compatible web servers to serve your project. See How to deploy with WSGI for more details.

THANK YOU

Address: G. R. Complex Preetam Nagar, Dhoomanganj
Prayagraj Uttar Pradesh 211011

Mobile No.: 9598948810,8874588766,9532878816
Website : www.infomaxacademy.com

