

Web Development using Django

MSc(IT) SEM.: 05

LJ Institute of computer applications

LJ University

What is Django?

- In 2003, Django was an internal project started by Adrian Holovaty and Simon Willison.
- The first version of Django was released in July 2005, and the name is given Django based on the name of guitarist Django Reinhardt
- A high-level Python web framework allows rapid development
- Provides easily maintainable and scalable websites
- Allows developers to build robust, scalable, and secure web applications quickly and efficiently
- Follows MVT model (model-view-template)
- "Don't Repeat Yourself" (DRY) and "Convention over Configuration"






Pre requisites



- ✓ Python Core Programming
- ✓ DBMS fundamentals
- ✓ Object Oriented Programming
- ✓ Familiar with web development concepts
- ☐ IDE: Visual Studio Code (preferred)
- ☐ DB: SQL (MySQL or Xamp)
- ☐ Version control: Git and GitHub (optional)



Outcome

- ✓ Strong backend development skills
 - ✓ MVC/MVT logic building
 - ✓ Able to work with real life applications
- 

Django vs Laravel

Feature / Aspect	Django	Laravel
Language	Python 	PHP 
Architecture	MVT (Model-View-Template)	MVC (Model-View-Controller)
Performance	Slightly faster (Python advantage)	Good, but can be slower under load
Learning Curve	Easier (Python syntax)	Moderate (needs PHP basics)
Built-in Features	More batteries-included (ORM, Admin)	Lots of built-in helpers + CLI tools
Admin Interface	Auto-generated & powerful	Needs third-party packages (e.g., Voyager)
Database ORM	Django ORM (built-in)	Eloquent ORM (intuitive, fluent)
Community Support	Large (Python + Django devs)	Very large (PHP is older and widespread)
Template Engine	DTL (Django Template Language)	Blade (clean, powerful)
Authentication	Built-in, extendable	Built-in, with easy role-based control
Frontend Integration	Good with React, Vue, HTMX	Good with Vue, InertiaJS, Livewire
Job Market	Data-heavy roles, AI/ML-ready	CMS, enterprise sites, agencies
Ideal Use Case	Data-driven apps, dashboards, APIs	Websites, eCommerce, CMSs, ERP systems



Features of using Django

- Batteries included which allows built-In security mechanism such as CSRF token, Session management, User Authentication and many more
- Customizable built-in Admin Interface
- ORM-Object relational mapper makes easier transactions with database
 - ORM(Object relational mapping) Let's you interact with the database using Python objects rather than raw SQL queries.
- Easy to build, develop and flexible architecture

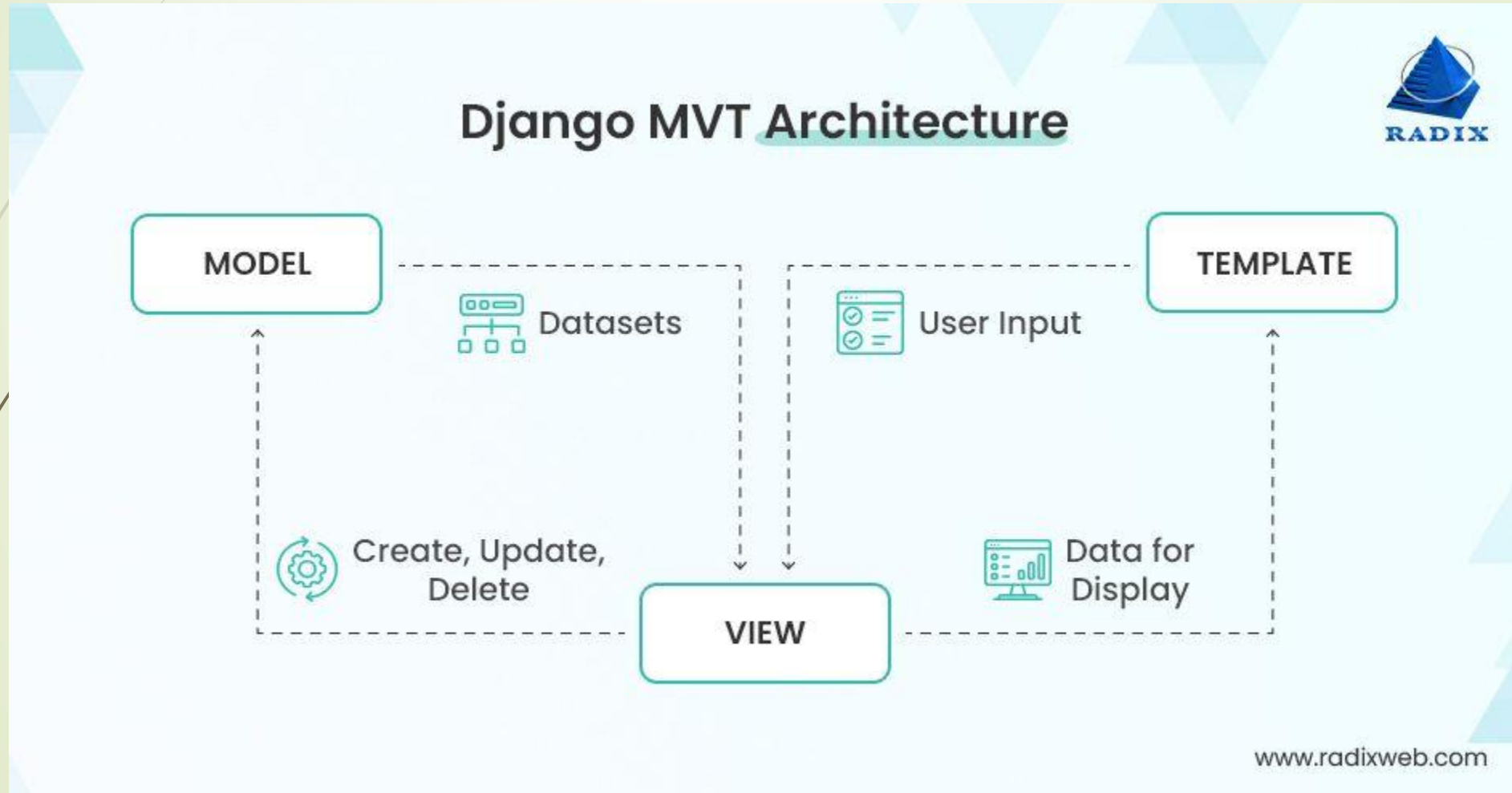


Applications



- Social network applications such as blogs, social media platforms (even Instagram originally used Django as their initial backend)
- Widely used for CMS and booking engines (hotels, tickets, etc...)
- Backend for mobile apps
- Popular for inventory and CRM platforms
- Scalable for enterprise applications
- Flexible for creating visuals and dynamic dashboards

Architecture of Django



Let's setup Django !

- Needs python installed (recommended python 3.11+)
- If not installed then install PIP

https://www.w3schools.com/django/django_create_virtual_environment.php

■ Set up Virtual environment:

Create a virtual environment

> python -m venv venv

Activate it:

Windows:

> venv\Scripts\activate

macOS/Linux:

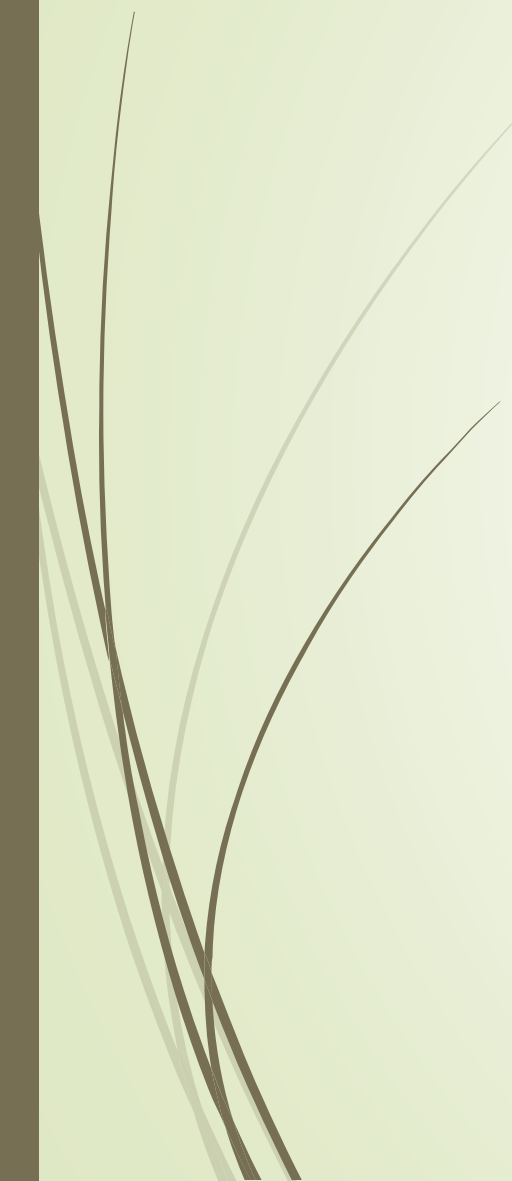
source venv/bin/activate

Install and create project

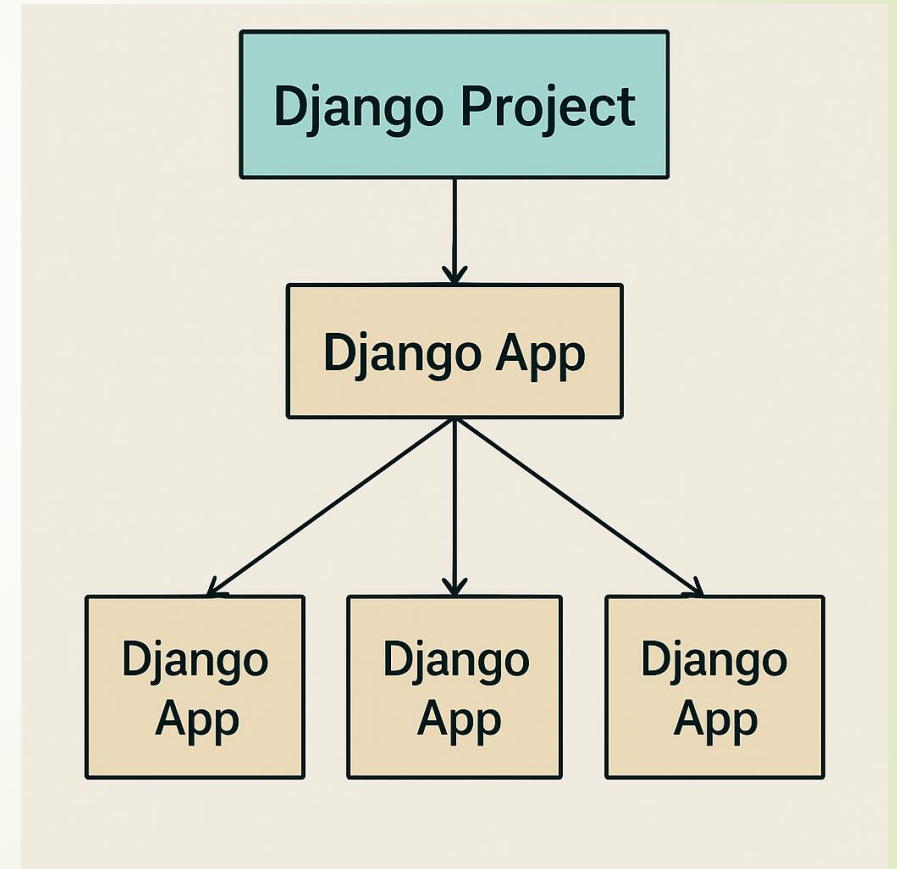
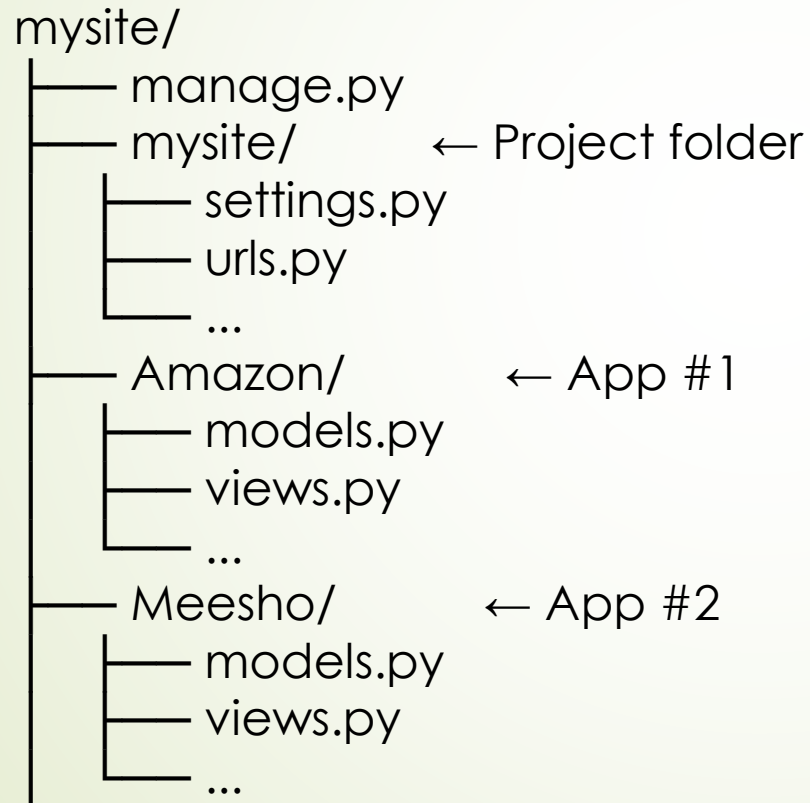
- Install Django
- > (myworld) ... \$ `python -m pip install Django`
- Check installation
- > (myworld)... \$ `django-admin --version`
- Create project using CLI
- > `django-admin startproject <project_name>`
- Start project
- > `py manage.py runserver`



App/Module in Django

- Django allows modular approach to build websites
 - App is used by Django project
 - Multiple apps could be built inside a single Django project
 - Apps or modules simplifies the complex structure and organizes large file systems used by Django
 - Provides reusability and makes scalable system design
- 

App structure example





Common & useful commands

- Create app
 - > *python manage.py startapp <appName>*
- Create project
 - > *Django-admin startproject <projectname>*
- Run server
 - > *py manage.py runserver*
- Migrate database
 - > *py manage.py migrate*
 - > *py manage.py makemigrations*
- Install third party packages
 - > *pip install <pkgname>*

Let's understand the project file structure

- myproject/
- |— manage.py
- |— myproject/
- | |— __init__.py
- | |— settings.py
- | |— urls.py
- | |— wsgi.py
- |— myapp/
- | |— admin.py
- | |— models.py
- | |— views.py
- | |— urls.py
- |— templates/

➤ myproject/ (Root Folder)

This is your main project folder (same name as your project). It contains “manage.py”, a command-line utility for managing your project (runserver, migrations, etc.)

➤ Inside myproject

__init__.py

Marks this folder as a Python package

settings.py

Main config file (database, apps, static files, security, etc.)

urls.py

Root URL routing for the entire project; includes app-level urls.py

wsgi.py

Entry point for WSGI servers (for deployment)

(Optional) asgi.py

Used when deploying with ASGI for async support

➤ Django app folder

File

admin.py

models.py

views.py

urls.py

templates/

Purpose

Register models to Django Admin

Define your database tables using Django ORM

Define the logic for what to show when a URL is visited

URL patterns specific to this app (you link this from the project urls.py)

Store HTML templates (inside templates/myapp/)

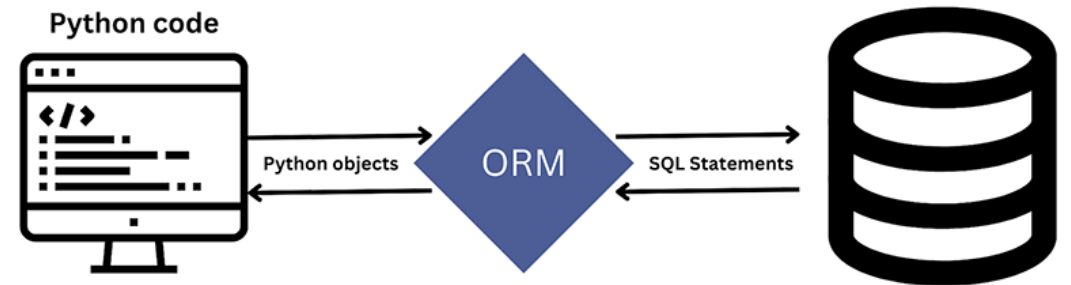


Working of Databases

- Most used databases with Django
 - SQLite (default)
 - MYSQL (preferred)
 - Postgresql (most used)
 - MongoDB(NoSQL – not suitable)

Django ORM

The Django ORM (Object-Relational Mapper) is a core component of the Django web framework that facilitates interaction with relational databases using Python code, rather than writing raw SQL.





MYSQL setup with Django

1. Install MYSQL server
 - <https://dev.mysql.com/downloads/>
 - Install mysql workbench for GUI
 - Note: remember your password for root user
2. Install MySQL Client for Python
 - `pip install mysqlclient`
3. Create MySQL database
4. Configure database in settings.py file
5. Run migrations



ALL THE BEST