geeksforgeeks.org

Django Basics

GeeksforGeeks

5–6 minutes

Last Updated : 12 Jul, 2025

Improve

Django is a Python-based web framework which allows us to quickly develop robust web application without much third party installations. It comes with many built-in features—like user authentication, an admin panel and form handling—that help you build complex sites without worrying about common web development tasks.

Key Features of Django

- **Rapid Development**: Build fully featured web applications quickly.
- **Built-In Admin Interface**: Manage your app's data easily through an automatically generated admin panel.
- **Database Flexibility**: Easily switch between databases like SQLite, MySQL or PostgreSQL.
- **Extensible**: Thousands of additional packages are available to extend Django's capabilities.
- **Scalability**: Designed to grow with your application.

> For more insight on when to use Django, check out: When to Use

> Django? Comparison with other Development Stacks ?

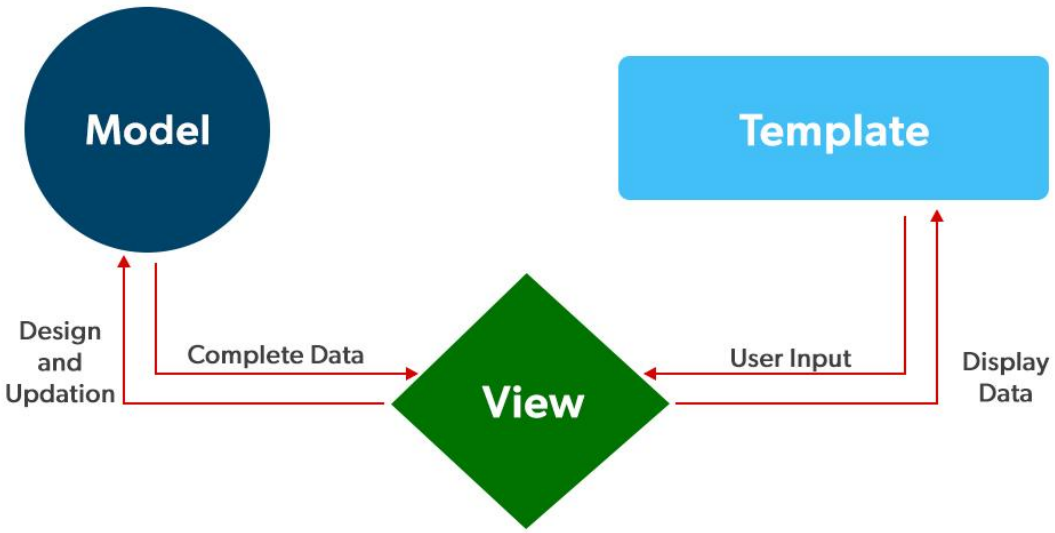
Django Architecture: The MVT Pattern

Django is based on MVT (Model-View-Template) architecture. MVT is a software design pattern for developing a web application. It's structure has the following three parts :

1. Model: Acts as the **data interface**. It defines the structure of your data and is usually backed by a database (e.g., MySQL, PostgreSQL).

2. View: A **Python function** or **class** that handles **web requests**. It interacts with the Model and renders a response, typically by passing data to a Template.

3. Template: Contains static HTML mixed with Django's templating syntax. Templates are used to display dynamic data on the web page.



Django MVT

> To check more about Django's architecture, visit Django Project MVT Structure

Installing Django

Follow these steps to set up Django on your system:

1. Install Python 3:

Download and install the latest Python 3 version from the official website.

2. Install pip:

Pip comes with recent versions of Python. Open the command prompt and run:

```
pip --version
```

3. Set Up a Virtual Environment:

This isolates your project's dependencies.

```
python -m virtualenv env_site
```

4. Activate the Environment:

Windows:

```
cd env_site\Scripts\activate
```

Mac/Linux:

```
source env_site/bin/activate
```

5. Install Django:

With your virtual environment active, run:

```
pip install django
```

Create a Django Project and App

1. Create a Django Project

Open the terminal and run:

```
django-admin startproject projectName
cd projectName
```

This creates a new folder named **projectName** containing your project settings and then changes the current working directory to it..

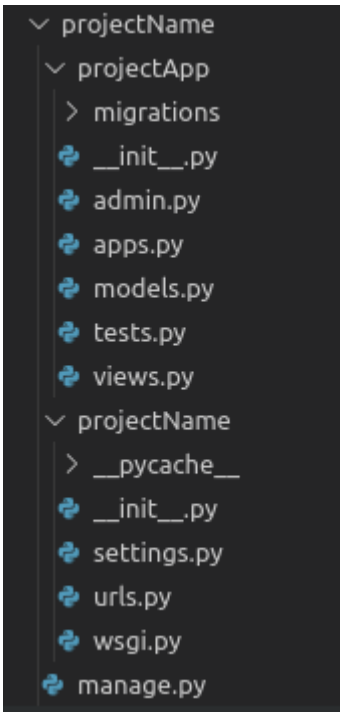
2. Create a Django App

Inside the project directory (where manage.py is located), run:

```
python manage.py startapp projectApp
```

This creates a new app named projectApp.

Now you can see your directory structure as under :



3. Update INSTALLED_APPS in Settings

In projectName/settings.py, add your app to the INSTALLED_APPS list:

```
# Application definition
```

```
INSTALLED_APPS = [
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',
    'projectApp'
]
```

We have finally created an app but to render it using urls we need to include the app in our main project so that urls redirected to that app can be rendered.

4. Include App URLs in the Main Project

Edit **projectName/urls.py** to include your app's URLs:

```
from django.contrib import admin
from django.urls import path, include

urlpatterns = [
    path('admin/', admin.site.urls),
    path('', include("projectApp.urls")),  #
Routes all other URLs to projectApp
]
```

Make sure you create a urls.py file inside **projectApp** to define your app's routes.

Using Django's MVT Model

Once your project is set up:

- **Models**: Define your data structure in **projectApp/models.py**.

- **Views**: Create functions or class-based views in projectApp/views.py to handle requests.
- **Templates**: Create HTML templates in a templates folder (either in the app or at the project level) and use Django's templating language to render dynamic content.