

**INSTITUTE OF TECHNOLOGY,**

**NIRMA UNIVERSITY**

**2CSDE86  Application Development Framework**

**PRACTICAL 1**

**Name :** Jaimik S. Chauhan

**Roll No. :** 20BCE503

**Semester :** 7th

**Teacher’s Signature :**

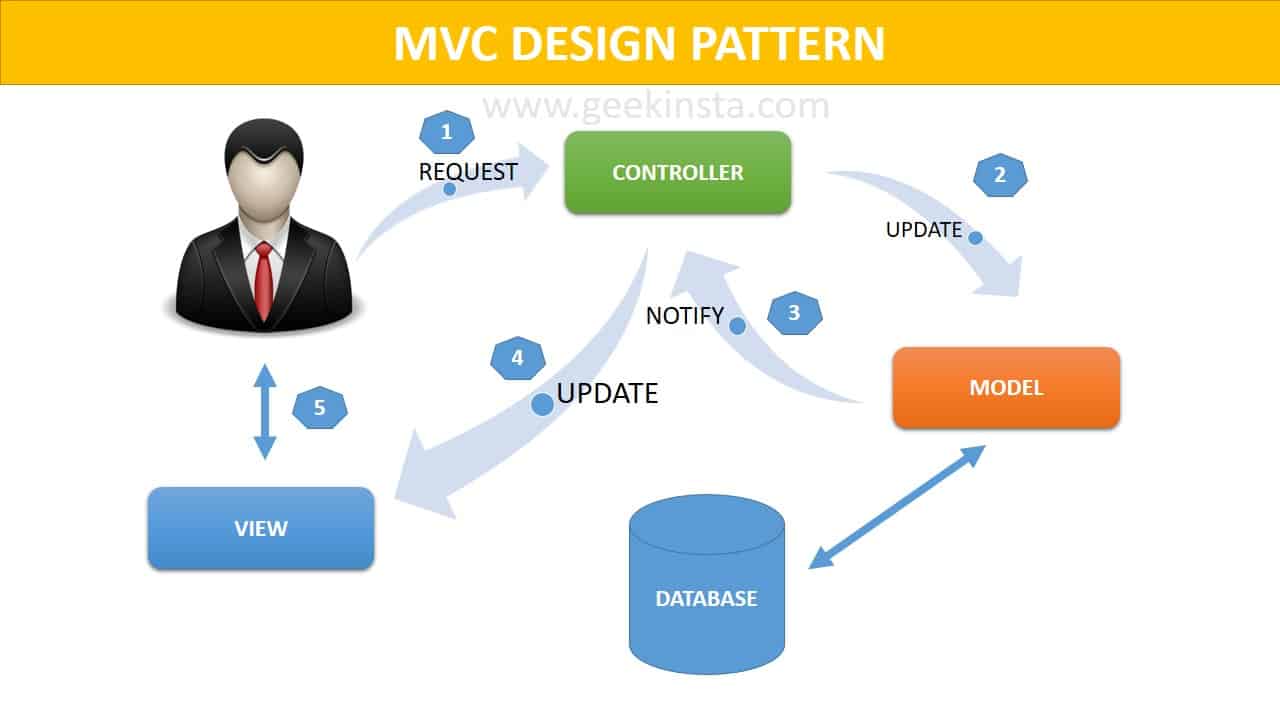
**Aim :** Introduction to MVC and MVT architecture. Comparative Study on MVC and MVT.

### MVC (Model View Controller)

* It’s software design pattern that is used to implement user interfaces and gives emphasis on separating data representation from the components which interacts and process the data.

It has 3 components and each has specific approach :

* Model is the central component of this architecture and manages the data, logic as well as other constraints of the application.
* View deals with how the data will be displayed to the user and provides various data representation components.
* Controller manipulates the Model and renders the view by acting as a bridge between both of them.



#### Advantages:

* Makes it easy to develop large applications
* Easy for multiple developers to collaborate and work together.

#### Disadvantages:

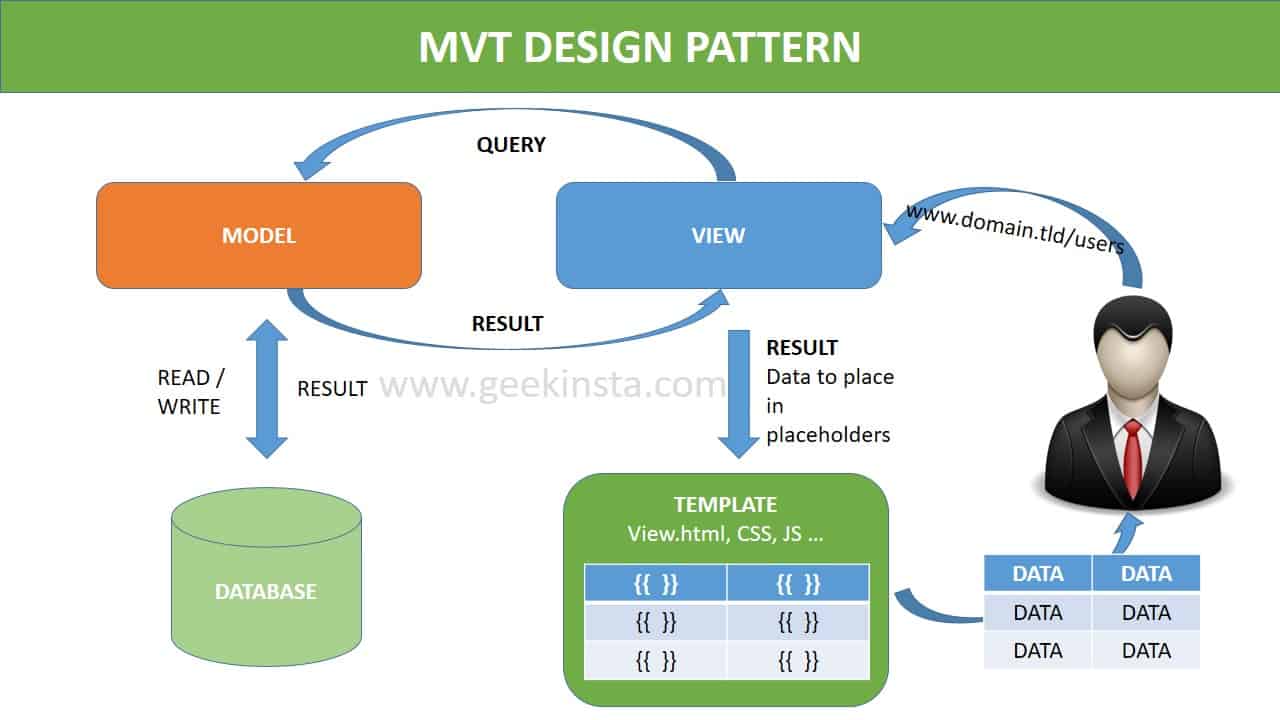
* View is controlled by Model and Controller
* Not suitable for small applications

### MVT (Model View Template)

* Similar to MVC, but in contrast to MVC the controller part is taken care for us by the framework itself.

It has 3 components :

* Model similar to MVC acts as an interface for your data and is basically the logical structure behind the entire web application which is represented by a database such as MySql, PostgreSQL.
* View Executes the business logic and interacts with the model and renders the template. It accepts HTTP request and then return HTTP responses.
* Templates is the component which makes MVT different from MVC. Templates acts as the presentation layer and are basically HTML code that renders the data. Content in these files can be either static on dynamic.



#### Advantages:

* Less coupled.
* Suitable for small to large-scale applications.
* Easy to Modify.

#### Disadvantages:

* Sometimes, understanding the flow can be confusing
* Modification of models/views should be done carefully without affecting templates.

## **MVC V/S MVT**

|  |  |
| --- | --- |
| Model View Controller (MVC) | Model View Template (MVT) |
| MVC has controller that drives both model and view | MVT has views for receiving HTTP request and returning HTTP response |
| View tells how the user data will be presented | Templates are used in MVT for that purpose |
| Highly coupled | Loosely coupled |
| Modifications are difficult | Modifications are easy |
| Suitable for development of large applications but not for small applications | Suitable for both small and large applications |
| Flow is clearly defined thus easy to understand | Flow is sometimes harder to understand as compared to MVC |
| It doesn’t involve mapping of URLs | URL pattern mapping takes place |
| Ex: Spring MVC | Ex: Django |

### **Explore Django Application Structure**

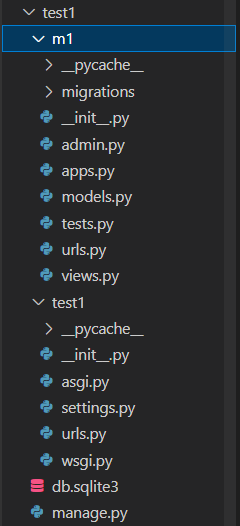
1. Create Django Project Using Following Command

Django-admin startproject [projectName]

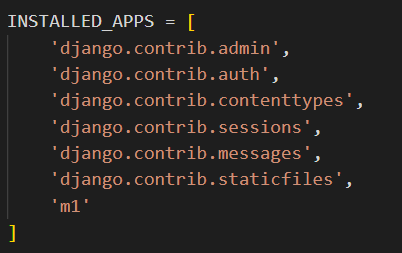
1. Create App/Module Using Following Command

Python manage.py startapp [app/module Name]

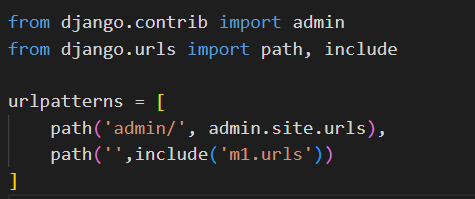
1. Project Structure



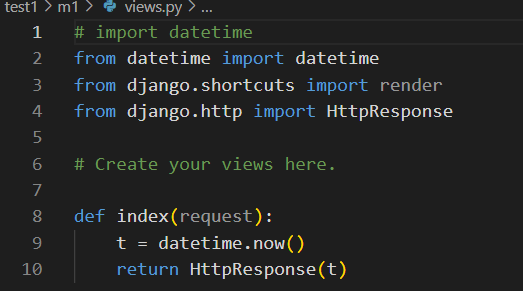
1. Register an app with Installed\_APPS.py file



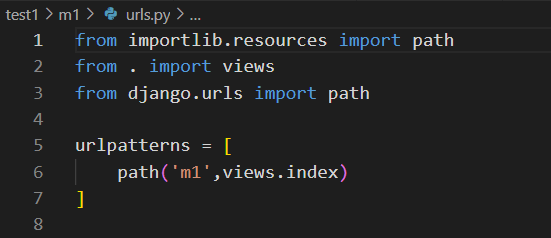
1. Register module/app urls to project Url file



1. Inside App/module edit views.py file



1. Import views inside urls.py file



1. Start the server using following command

Python manage.py runserver



**Manage.py :**

The manage.py file provides a command-line utility for a Django project. You will use this command-line utility to perform various operations related to debugging, running, and deploying a Django web application.

The Files are present in project :

**\_\_init\_\_.py**

It is an empty Python file. The \_\_init\_\_.py file tells the Python interpreter that the directory project is a Python package.

**Asgi.py & wsgi.py**

asgi stands for **Asynchronous Server Gateway Interface** and wsgi stands for **Web Server Gateway Interface.**

After your development process is completed, you will move to production and hosting. For hosting you will use asgi or wsgi compatible servers. According to the type of server you use, you have to import middleware accordingly.

asgi.py enables ASGI compatible servers and wsgi.py enables WSGI compatible servers to serve your Django web app.

**Settings.py**

This is the main configuration file for a Django project. This is the main settings file and here you will configure all the apps and middleware for your project.

This file also handles the database settings. By default Django uses sqlite3. But if you use a different database, which you will most probably do, you will configure it in settings.py.

settings.py also handles templates, static files, and media files related settings.

**Urls.py**

URLs are different endpoints of your website. urls.py contains the URL configurations for your website. By default, urls.py comes with the URL pattern for the admin panel. You will create other endpoints for your web app in this file.

**App Structure :**

**\_\_init\_\_.py**

The \_\_init\_\_.py file in an app is no different than the \_\_init\_\_.py file in a project. This empty Python file is telling the interpreter that app is a Python package.

**admin.py**

This file is used to register the models in your app in the Django administration. You will use this file to display the models of your app in the admin panel.

**apps.py**

It is a common configuration file for all Django apps. You can configure the attributes for your app using this file. However, the default configuration is sufficient for most cases. So, adding app configuration is a rare case.

**views.py**

In this file, you write the business logic for your app. A view can be either function-

based or class-based. You decide if you want to write your views using functions or

classes.