Course Module 1: Course Unit 10: Django (Fundamentals)

Week 13

Django (Fundamentals)

Objectives:

- Classify Django fundamentals.
- Categorize Django fundamentals.
- Generate a Django model using Django fundamentals.

Django Fundamentals

Django Fundamentals

- **Django Architecture**
- Django MTV Architecture
- Django Project Layout and File Structure
- Django Models
- Django Views
- Django Templates
- Django URLs and URL Conf

Django Architecture

Django Architecture

For every website on the Internet, there are **3 Main components or code partitions**:

- **Input Logic** - the dataset and how the data gets to organize in the database. It just takes that input and sends it to the database in the desired format.
- **Business Logic** - the main controller which handles the output from the server in the HTML or desired format.
- **UI Logic** - the HTML, CSS and JavaScript pages.

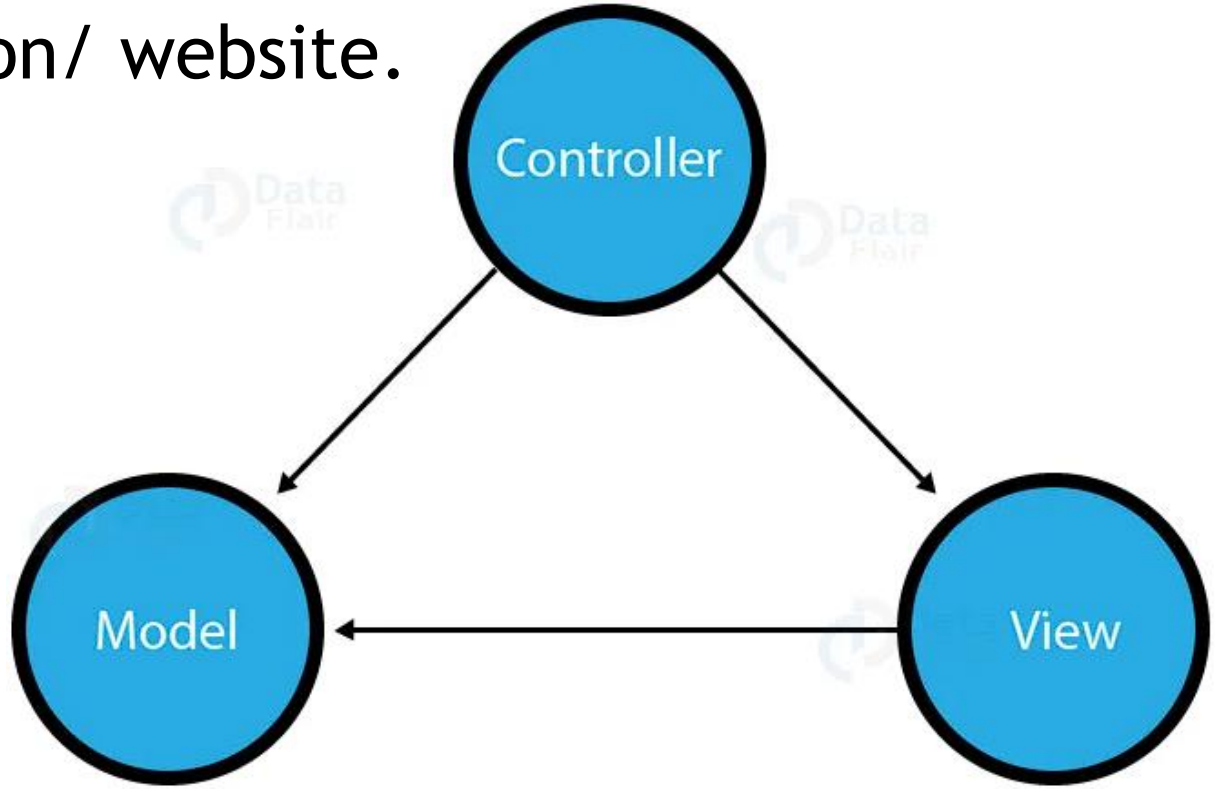
(Traditional Approach) – Written in a single file

Django Architecture

Model View Controller

MVC pattern is a Product Development Architecture. It solves the traditional approach's drawback of code in one file, i.e., that MVC architecture has different files for different aspects of our web application/ website.

This difference between components helps the developer to focus on one aspect of the web-app and therefore, better code for one functionality with better testing, debugging and scalability.



Django Architecture

Django Architecture

Model

- the part of the web-app which acts as a mediator between the website interface and the database.
- In technical terms, it is the object which implements the logic for the application's data domain.
- There are times when the application may only take data in a particular dataset, and directly send it to the view (UI component) without needing any database then the dataset is considered as a model.
- The Model is the component which contains Business Logic in Django architecture.

Django Architecture

Model

For example:

When you sign up on any website you are actually sending information to the controller which then transfers it to the models which in turn applies business logic on it and stores in the database.

Django Architecture

View

- This component contains the UI logic in the Django architecture.
- View is actually the User Interface of the web-application and contains the parts like HTML, CSS and other frontend technologies.
- Generally, this UI creates from the Models component, i.e., the content comes from the Models component.

For example:

When you click on any link or interact with the website components, the new webpages that website generates is actually the specific views that stores and generates when we are interacting with the specific components.

Django Architecture

Controller

- The controller as the name suggests is the main control component. What that means is, the controller handles the user interaction and selects a view according to the model.
- The main task of the controller is to select a view component according to the user interaction and also applying the model component.
- This architecture has lots of advantages and that’s why Django is also based on this architecture. It takes the same model to an advanced level.

Django Architecture

Controller

For example:

When we combine the two previous examples, then we can very clearly see that the component which is actually selecting different views and transferring the data to the model's component is the controller.

Django Fundamentals

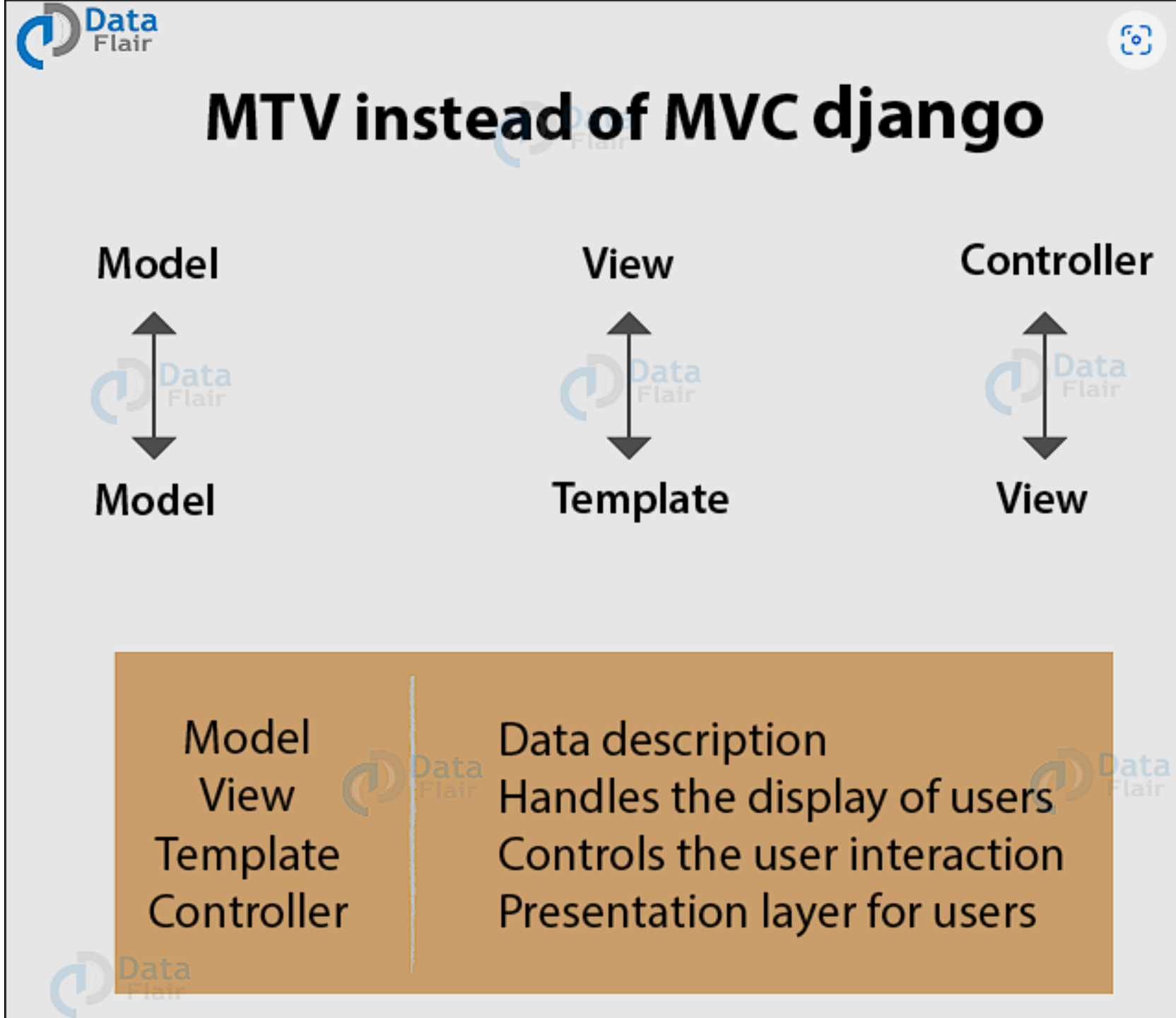
- Django Architecture
- **Django MTV Architecture**
- Django Project Layout and File Structure
- Django Models
- Django Views
- Django Templates
- Django URLs and URL Conf

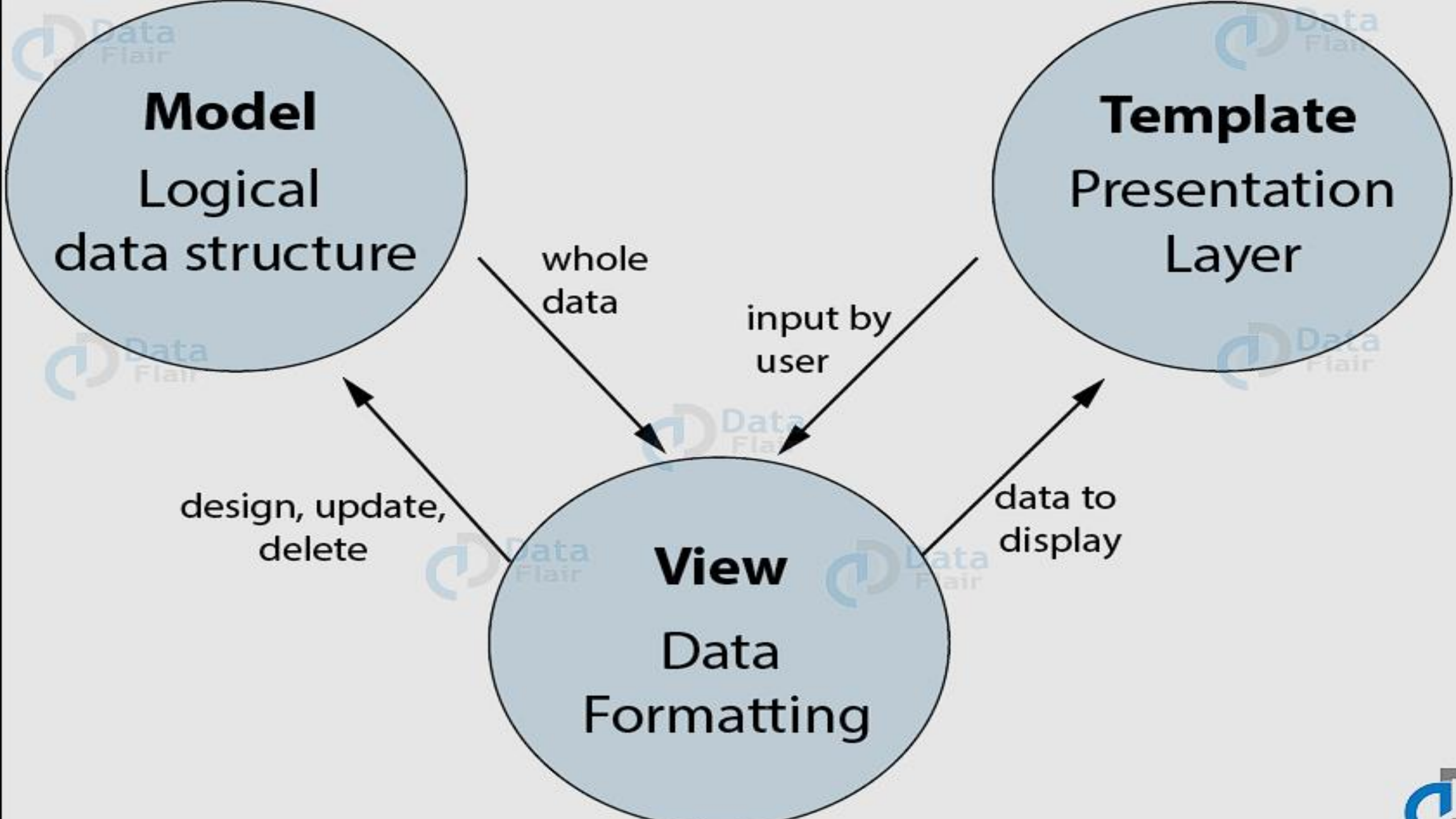
Django MTV Architecture

Django is mainly an MTV (Model-Template-View) framework. It uses the terminology Templates for Views and Views for Controller.

Template relates to the View in the MVC pattern as it refers to the presentation layer that manages the presentation logic in the framework and essentially controls the content to display and how to display it for the user.

Thus our **Python code** will be in views and models and HTML code will be in templates.




Model
Logical
data structure
Template
Presentation
Layer
whole
data
input by
user
design, update,
delete
View
Data
Formatting
data to
display

An easier real-life working of above functioning would be –

When you login in a website (Django based), you open the login page. It again happens without the need of the Model. It is because Views will process the request and send it to the URL of the login page. Then, it will be a response by the server, from there to the browser.

After that, you enter your credentials in the given Template, HTML form. From there the data is again sent to the view, this time this request rectifies and the model is given data. Then the Model reads and verifies the data that user provides within the connected database.

If the user data matches it will send the relevant user data like profile image, name and (other things depending on the type of website) to the Views. It will then format the same in desired response and will transmit the same to the client.

Otherwise, the Model will send a negative result to the Views. In turn, it will rout it to the login page again alongside an error message.

That's how the Django MTV architecture is actually working.

Django Fundamentals

- Django Architecture
- Django MTV Architecture
- **Django Project Layout and File Structure**
- Django Models
- Django Views
- Django Templates
- Django URLs and URL Conf

Django Project Layout and File Structure

django Project Layout & Files Structure

When you **create a Django project**, the Django framework itself creates a root directory of the project with the project name on it. That contains some files and folder, which provide the very basic functionality to your website and on that strong foundation you will be building your full scaled website.

Files in the Django Project Root Directory

By root directory, we mean about the directory which contains your **manage.py** file. Additional files like **db.sqlite**, which is a database file may be present when we will be migrating our project.

Django root directory is the default app which Django provides you. It contains the files which will be used in maintaining the whole project.

The name of Django root directory is the same as the project name you mentioned in django-admin startproject [projectname]. This root directory is the project's connection with Django.

manage.py

- This file is the command line utility of the project and we will be using this file only to deploy, debug and test with the project.
- The file contains the code for starting the server, migrating and controlling the project through command-line.
- This file provides all the functionality as with the django-admin and it also provides some project specific functionalities.

FREQUENTLY USED COMMANDS:

- **runserver** is a command to start the test server provided by Django framework and that is also one of the advantages of Django over other frameworks.
- **makemigrations** is the command for integrating your project with files or apps you have added in it. This command will actually check for any new additions in your project and then add that to the same.
- **migrate,** the last command is to actually add those migrations you made in the last command with the whole project. You can get the idea as the former command is used for saving the changes in the file and later one to actually apply that change to the whole project then the single file.

my_project

my_project is the python package of the project.
It has the configuration files of project settings.

i. __init__.py
ii. settings.py
iii. urls.py
iv. wsgi.py

```
    Directory: C:\Users\HP\documents\dataflair\dataflairdjango\dataflair\dataflair


Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
d-----        06-03-2019     23:42                __pycache__
-a----        06-03-2019     23:15           3217 settings.py
-a----        06-03-2019     23:15            772 urls.py
-a----        06-03-2019     23:15            411 wsgi.py
-a----        06-03-2019     23:15              0 __init__.py
```

i. __init__.py

The __init__.py file is empty and it exists in the project for the sole purpose of telling the python interpreter that this directory is a package.
That's one of the standard rules of **python packages**.
Although we won't be doing anything on this file.

```
    Directory: C:\Users\HP\documents\dataflair\dataflairdjango\dataflair\dataflair


Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
d-----        06-03-2019     23:42                __pycache__
-a----        06-03-2019     23:15           3217 settings.py
-a----        06-03-2019     23:15            772 urls.py
-a----        06-03-2019     23:15            411 wsgi.py
-a----        06-03-2019     23:15              0 __init__.py
```

ii. settings.py

- The settings.py is the main file where we will be adding all our applications and middleware applications.
- As the name suggests this is the main settings file of the Django project. This file contains the installed applications and middleware information which are installed on this Django project.
- Every time you install a new app or custom application you will be adding that in this file.

iii. urls.py

- urls.py file contains the project level URL information.
- URL is universal resource locator and it provides you with the address of the resource (images, webpages, web-applications) and other resources for your website.
- The main purpose of this file is to connect the web-apps with the project. Anything you will be typing in the URL bar will be processed by this urls.py file. Then, it will correspond your request to the designated app you connected to it.

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

urlpatterns = [
    path('admin/', admin.site.urls),
]
```

Here this file by default adds one URL to the admin app. The **path ()** takes two arguments.

- the URL to be searched in the URL bar on the local server
- the file you want to run when that URL request is matched, the admin is the pre-made application and the file is URL's file of that app.

This file is the map of your Django project.

iv. wsgi.py

- Django is based on python which uses WSGI (Web Server Gateway Interface) server for web development.This file is mainly concerned with that and we will not be using this file much.
- wsgi is still important though if you want to deploy the applications on Apache servers or any other server because Django is still backend and you will need its support with different servers.

But you need not to worry because for every server there is a Django middleware out there which solves all the connectivity and integration issues and you just have to import that middleware for your server, it's very simple.

Applications???

Any questions???

REFERENCES:

- Django Tutorial. (2022). https://data-flair.training/blogs/django-tutorial/
- Django Tutorial. (2020). https://www.geeksforgeeks.org/django-tutorial/
- Django Tutorial. (2022). https://developer.mozilla.org/en-US/docs/Learn/Server-side/Django/Introduction
- Django Tutorial. (n.d.). https://www.tutorialspoint.com/django/index.htm
- Django Tutorial. (n.d.). https://www.w3schools.com/django/index.php
- Getting started with Django. (n.d.). https://www.djangoproject.com/start/