Cookies: -

Cookies is a small packet of data. When user visit a website, webserver transfers a small packet of data in the form of cookies with that request. Basically the cookies are designed to remember the information about your website visits and activities.

Object Relational mapper: -

ORM is used to enable the application to interact with databases. It automatically creates a database schema from defined classes or models. It generate SQL from python code that means we don’t need to write SQL code.

Queryset: -

It can be defined as the list containing all those objects that we created using django model.

Explain Django Architecture?

Dajngo follows the MVT pattern which is based on the MVC architecture. It is slightly different from the MVC as it handle the controller itself. The templates are basically an HTML file mixed with Django Template Language (DTL). When user make a request to django, django act as a controller and looks for the resource in the urls. Once the resource is matched then it will go to the view. View takes the data from the models and perform some action on it. Then it gives data to the template and render it on UI.

What are models in Django?

Model class is the class which represents a table in the database. Each attribute in the model class represent a field in the database table. We can create our own model class inside our application in the models.py file. Our own model class inherits the python’s model class.

Rules for defining field:-

1. A field name cannot be a python reserved keyword. It throws python syntax error.
2. A field name cannot contain more than one underscore.
3. A field name cannot end with an underscore.

# **Migration: -**

It is a way of applying changes that we have made to a model into the database schema. Django create a migration file inside the migration folder for each model to create a table schema.

Django provides various command to perform migrations related tasks:

1. Makemigrations => Responsible for creating new migration based on the changes. (convert class into sql query => create file in migrations folder)
2. Migrate => Responsible for applying and unapplying migrations. (executes the sql statement to create table)
3. Sqlmigrate => display the sql statement for migration.

(syntax: python manage.py sqlmigrate student 0001)

1. Showmigrations => It list out all the migrations and their status.

What are Templates in Django or Django template language?

Template is an integral part of Django’s MVT architecture. It consist of HTML, CSS and JS in which dynamic variables and information are embedded with the help of views. A template is rendered with a context. Rendering just replaces the variables with their values and other remains as it is.

What are views in Django?

A view function is simply a python function which takes a web request from user and returns a web response. There are two types of views;

1. Function based view
2. Class based view => Object oriented approach

What are staticfiles and explain their uses?

Websites generally needs to serve some additional files such as images, javascript or css. In django these files are called static files.

**Signals: -**

Whenever we saved the data to the database, there may be require some processing of data before just saving and after just saving the data. This we can easily achieved by using Django Signals. This signals works on the concept of senders and receivers. Signals are used to perform any action on modifcations of model instances. We can define a function which is called a receiver function and it is called when senders sends the signal to that particular receivers function. There are three types of signals;

1. pre\_save/post\_save : These signal works before or after the method save().

2. pre\_delete/post\_delete : These signal works before or after the method delete().

3. pre\_init/post\_init : These signals works before or after instantiating a model. That is before or after \_\_init\_\_ method.

**Model Manager: -**

It is the interface through which database query operations are provided to the Django models. At least one manager exists for every models in a django application. By default django adds a manager with the name objects to every django model class. We can change it’s name with the help of this syntax;

manager\_name = models.Manager()

**Model Relationships: -**

Django provides three types of database relationships;

1. One to one

2. Many to one

3. Many to Many

One to One: -

It is a relationship where record in one table is associated with exactly one record in another table. To define one to one relationship we used OneToOneField.

Syntax:

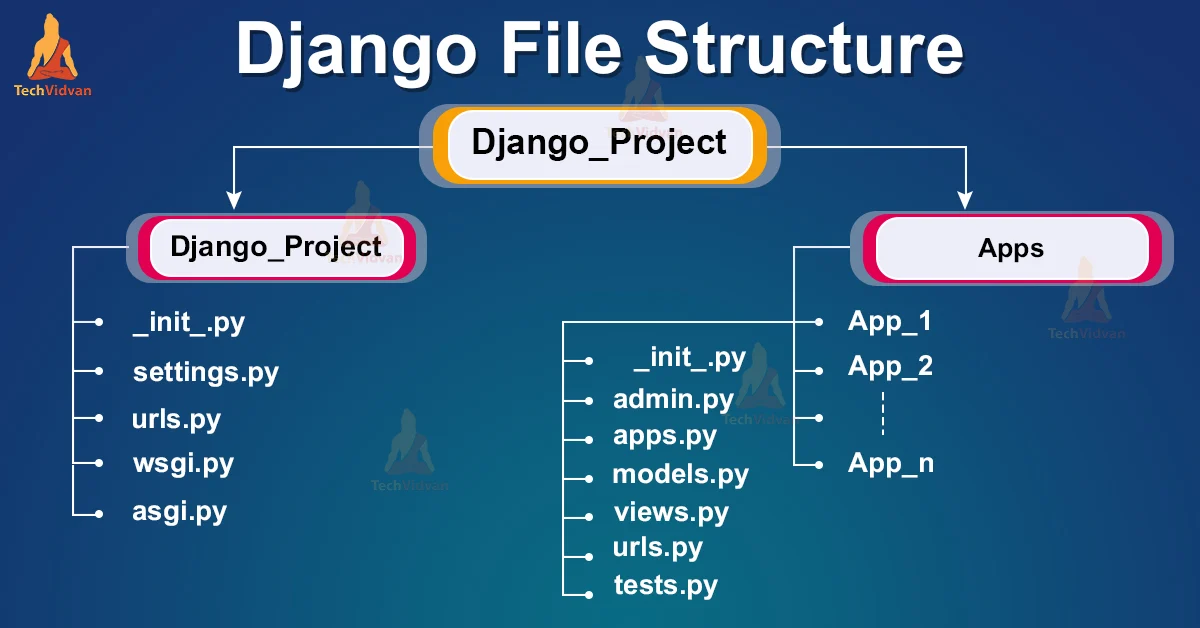
OneToOneField(to,on\_delete)

to => the class to which the model is related

**Django Intro:-**

It is free, open source, python based high level web framework. It follows the MVT pattern.

**Django Project Directory structure:-**



1. \_\_init\_\_.py ==> The folder which contain the \_\_init\_\_.py file is considered as python package.

2. wsgi.py ==> (Web Server Gateway Interface)

It is a specification that describes how a web server communicates with web application. WSGI provides a standard for synchronous python apps.

3. asgi.py ==> (Asynchronous Server Gateway interface)

It is a successor to WSGI. ASGI provides standard for both asynchronous and synchronous apps.

4. settings.py ==> contains all informations and data about project settings.

5. urls.py ==> It contains informations of urls attached with the applications.

6. Manage.py ==> This file is used as a command-line utility for our projects. We will use this file for debugging, deploying, and running our web applications.

**URL Dispatcher:-**

In order to design urls for our app, we need to create python module name urls.py. This module contains purely python code and it maps between url path and view functions.

**Extends tag:-**

this tag is used to inherit the templates. It has no end tag. It must be the first tag in the specific template, otherwise it will not work.

**Block tag:-**

This tag is used to overide the specific part of the template.

{% block blockname %}

..........

..........

{% endblock blockname %}

=> We can’t multiple tags with the same name.

=> If we need content of the block from parent template we used {{block.super}}

**Include tag:-**

This tag loads the template and renders it with current context. This is a way to include other template into current template.

Syntax:

{% include “templatename.html” %}

=> we can pass additional context to the template using “with” keyword.

{% include “templatename.html” with p=”php” %}

**Widget:-**

It represents how our html elements looks like. The widget handles the extraction of data from a GET or POST dictionary.

**Model Relationships: -**

Django provides three types of database relationships;

1. One to one

2. Many to one

3. Many to Many

One to One: -

It is a relationship where record in one table is associated with exactly one record in another table. To define one to one relationship we used OneToOneField.

Syntax:

OneToOneField(to,on\_delete)

to => the class to which the model is related