Django

Basic & Advanced Interview Questions

BY – VIKAS MAURYA

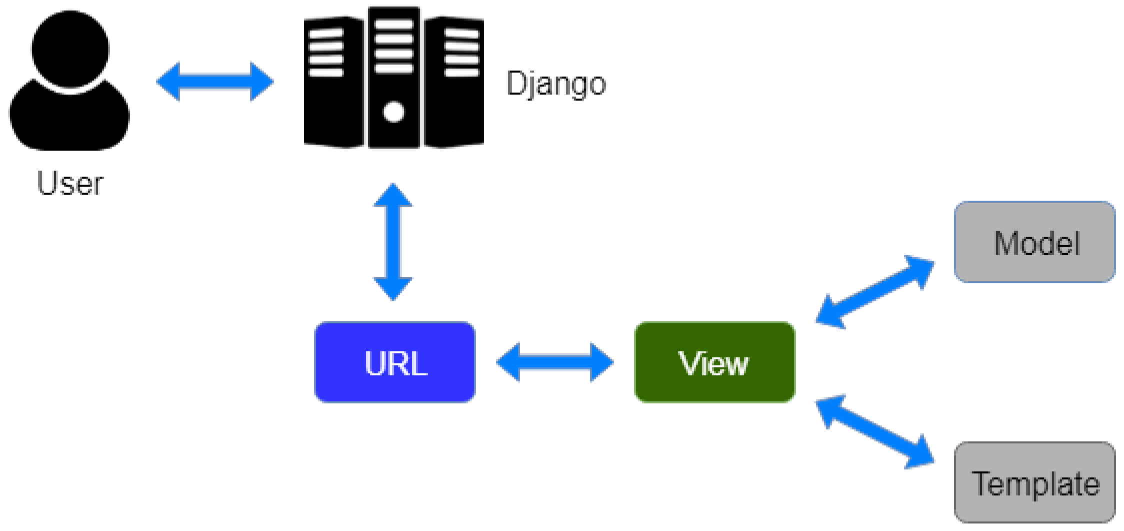
YOUTUBE Channel

1. CODE WITH VIKAS

2. VIKAS MAURYA ACADEMY

1.Explain Django Architecture?

Django follows the MVT (Model View Template) pattern which is based on the Model View Controller architecture. It’s slightly different from the MVC pattern as it maintains its own conventions, so, the controller is handled by the framework itself. The template is a presentation layer. It is an HTML file mixed with Django Template Language (DTL). The developer provides the model, the view, and the template then maps it to a URL, and finally, Django serves it to the user.



2. Explain the django project directory structure?

manage.py - A command-line utility that allows you to interact with your Django project __init__.py - An empty file that tells Python that the current directory should be considered as a Python package settings.py - Comprises the configurations of the current project like DB connections. urls.py - All the URLs of the project are present here wsgi.py - This is an entry point for your application which is used by the web servers to serve the project you have created.

3. What are models in Django?

A model in Django refers to a class that maps to a database table or database collection. Each attribute of the Django model class represents a database field. They are defined in app/models.py

Example:

from django.db import models

class SampleModel(models.Model):

field1 = models.CharField(max_length = 50)

field2 = models.IntegerField()

class Meta:

db_table = “sample_model”

Every model inherits

from django.db.models.Model

Our example has 2 attributes (1 char and 1 integer field), those will be in the table fields. The metaclass helps you set things like available permissions, singular and plural versions of the name, associated database table name, whether the model is abstract or not, etc.

4. What are templates in Django or Django template language?

Templates are an integral part of the Django MVT architecture. They generally comprise HTML, CSS, and js in which dynamic variables and information are embedded with the help of views. Some constructs are recognized and interpreted by the template engine. The main ones are variables and tags. A template is rendered with a context. Rendering just replaces variables with their values, present in the context, and processes tags. Everything else remains as it is. The syntax of the Django template language includes the following four constructs :

- Variables
- Tags
- Filters
- Comments

5. What are views in Django?

A view function, or "view" for short, is simply a Python function that takes a web request and returns a web response. This response can be HTML contents of a web page, or a redirect, or a 404 error, or an XML document, or an image, etc. Example: from django.http import HttpResponse def sample_function(request): return HttpResponse("Welcome to Django") There are two types of views: Function-Based Views: In this, we import our view as a function. Class-based Views: It's an object-oriented approach.

6. What is Django ORM?

This ORM (an acronym for Object Relational Mapper) enables us to interact with databases in a more pythonic way like we can avoid writing raw queries, it is possible to retrieve, save, delete and perform other operations over the database without ever writing any SQL query. It works as an abstraction layer between the models and the database.

7. Define static files and explain their uses?

Websites generally need to serve additional files such as images. Javascript or CSS. In Django, these files are referred to as "static files", Apart from that Django provides django.contrib.staticfiles to manage these static files. 8. What is Django Rest Framework(DRF)? Django Rest Framework is an open-source framework based upon Django which lets you create RESTful APIs rapidly.

9. What is django-admin and manage.py and explain its commands?

django-admin is Django's command-line utility for administrative tasks. In addition to this, a manage.py file is also automatically created in each Django project. Not only does it perform the same purpose as the django-admin but it also sets the DJANGO_SETTINGS_MODULE environment variable to point to the project's settings.py file.

- django-admin help - used to display usage information and a list of the commands provided by each application.
- django-admin version - used to check your Django version.
- django-admin check - used to inspect the entire Django project for common problems.
- django-admin compilemessages - Compiles .po files created by makemessages to .mo files for use with the help of built-in gettext support.
- django-admin createcachetable - Creates the cache tables for use in the database cache backend.
- django-admin dbshell - Runs the command-line client for the database engine specified in your ENGINE setting(s), with the connection parameters (USER, PASSWORD, DB_NAME, USER etc.) specified settings file.
- django-admin diffsettings - Shows the difference between the existing settings file and Django's default settings.
- django-admin dumpdata - Used to the dumpdata from the database.
- django-admin flush - Flush all values from the database and also re-executes any post-synchronization handlers specified in the code.
- django-admin inspectdb - It generates django models from the existing database tables.
- django-admin loaddata - loads the data into the database from the fixture file.
- django-admin makemessages - Used for translation purpose and it generates a message file too.
- django-admin makemigrations - Generates new migrations as per the changes detected to your models.
- django-admin migrate - Executes SQL commands

10. What is Jinja templating?

Jinja Templating is a very popular templating engine for Python, the latest version is Jinja2. Some of its features are: Sandbox Execution - This is a sandbox (or a protected) framework for automating the testing process HTML Escaping - It provides automatic HTML Escaping as , & characters have special values in templates and if using a regular text, these symbols can lead to XSS Attacks which Jinja deals with automatically. Template Inheritance Generates HTML templates much faster than default engine Easier to debug as compared to the default engine.

11. What are Django URLs?

URLs are one of the most important parts of a web application and Django provides you with an elegant way to design your own custom URLs with help of its module known as URLconf (URL Configuration). The basic functionality of this python module is to You can design your own URLs in Django in the way you like and then map them to the python function (View function).

These URLs can be static as well as dynamic. These URLs as present in the urls.py where they are matched with the equivalent view function.

Basic Syntax:

from django.urls import path

from . import views

urlpatterns = [path('data/2020/', views.data_2020), path('data//', views.data_year)]

12. What is the difference between a project and an app in Django?

In simple words Project is the entire Django application and an app is a module inside the project that deals with one specific use case. For eg, payment system(app) in the eCommerce app(Project).

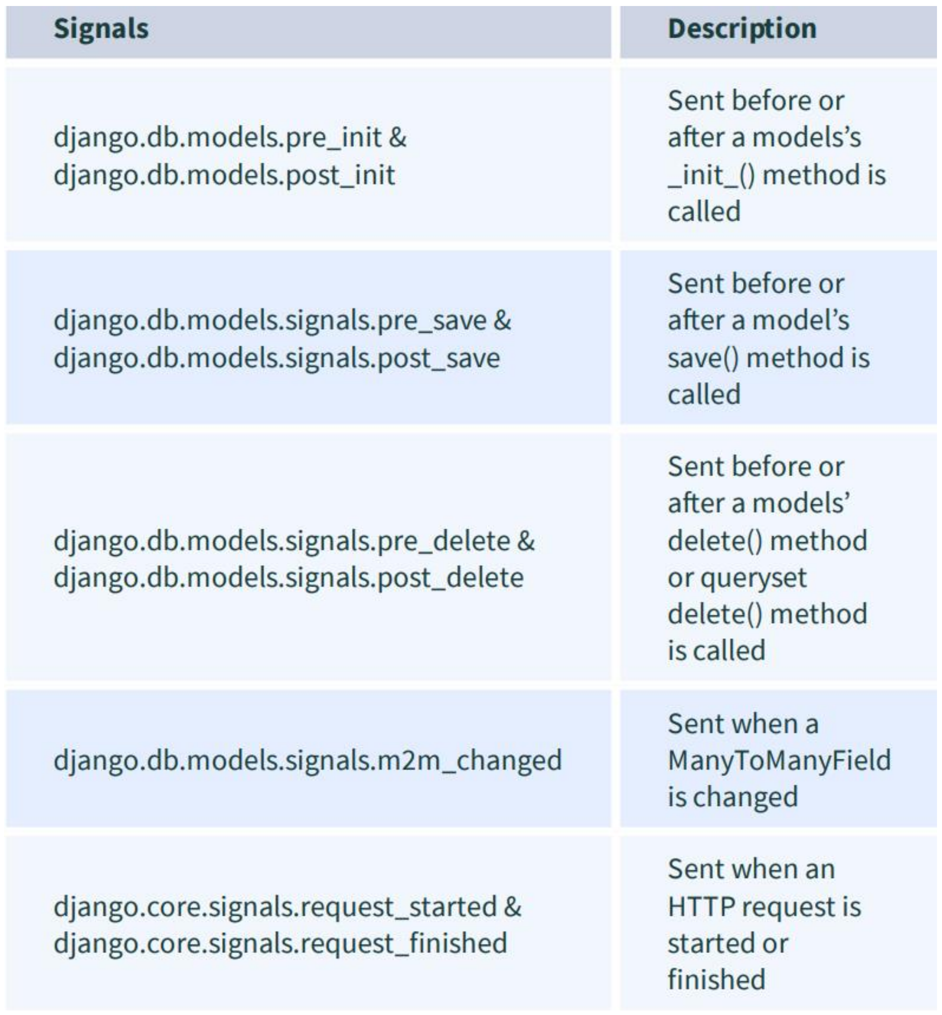
13. What are different model inheritance styles in the Django?

- Abstract Base Class Inheritance: Used when you only need the parent class to hold information that you don’t want to write for each child model.
- Multi-Table Model Inheritance: Used when you are subclassing an existing model and need each model to have its own table in the database.
- Proxy Model Inheritance: Used when you want to retain the model's field while altering the python level functioning of the model.

14. What are Django Signals?

Whenever there is a modification in a model, we may need to trigger some actions. Django provides an elegant way to handle these in the form of signals. The signals are the utilities that allow us to associate events with actions. We can implement these by developing a function that will run when a signal calls it.

List of built-in signals in the models:



Signals	Description
django.db.models.pre_init & django.db.models.post_init	Sent before or after a models's _init_() method is called
django.db.models.signals.pre_save & django.db.models.signals.post_save	Sent before or after a model's save() method is called
django.db.models.signals.pre_delete & django.db.models.signals.post_delete	Sent before or after a models' delete() method or queryset delete() method is called
django.db.models.signals.m2m_changed	Sent when a ManyToManyField is changed
django.core.signals.request_started & django.core.signals.request_finished	Sent when an HTTP request is started or finished

15. Explain the caching strategies in the Django?

Caching refers to the technique of storing the output results when they are processed initially so that next time when the same results are fetched again, instead of processing again those already stored results can be used, which leads to faster accessing as well us less resource utilization. Django provides us with a robust cache system that is able to store dynamic web pages so that these pages don't need to be evaluated again for each request.

16. Explain user authentication in Django?

Django comes with a built-in user authentication system, which handles objects like users, groups, user-permissions, and few cookie-based user sessions. Django User authentication not only authenticates the user but also authorizes him. The system consists and operates on these objects:

- Users
- Permissions
- Groups
- Password Hashing System
- Forms Validation

- A pluggable backend system

17. How to configure static files?

Ensure that django.contrib.staticfiles is added to your INSTALLED_APPS In your settings file.

define STATIC_URL

for ex. STATIC_URL = '/static/'

In your Django templates, use the static template tag to create the URL for the given relative path using the configured STATICFILES_STORAGE.

{% load static %}

<img src="{% static 'my_sample/abcxy.jpg' %}" alt="ABC image">

Store your static files in a folder called static in your app. For example my_sample/static/my_sample/abcxy.jpg

18. Explain Django Response lifecycle?

Whenever a request is made to a web page, Django creates an HttpRequest object that contains metadata about the request.

1. First of the Django settings.py file is loaded which also contain various middleware classes (MIDDLEWARES)

2. The middlewares are also executed in the order in which they are mentioned in the MIDDLEWAREST

3. From here on the request is now moved to the URL Router, who simply gets the URL path from the request and tries to map with our given URL paths in the urls.py.

4. As soon as it has mapped, it will call the equivalent view function, from where an equivalent response is generated

5. The response also passes through the response middlewares and send back to the client/browser.