Date : / /
Page No.

Django Assignment BCA-213

Q1. Introduction to Django :

- Provide a brief overview of Django and its features

Ans:→ Django is a backend server side web framework.
→ Django is a web application framework written in Python Programming language.
→ It is based on MVT (Model View Template) design pattern.
→ The Django is very demanding due to its rapid development feature.
→ It takes less time to build application after collecting client requirement.
→ The Django was design and developed by lawrence journal world in 2005.
→ Django was released on 21 July 2005 and current stable version is 2.0.3 which was released on 6 March 2018.

⇒ FEATURES :-

1) Rapid Development :- Django was designed with the intention to make a framework which takes less time to build web application.

2) Secure :- Django takes security seriously and helps developers to avoid many common security mistake such as SQL, injection, cross site etc.

Its user authentication system provides a secure way to manage user accounts and passwords.

3) Scalable :- Django is scalable in nature and has ability to quickly and flexibly switch from small to large scale application project.

4) Open Source :- Django is an open source web application framework. It is publicly available without any cost.

- Explain the adv: advantages of using Django for web development projects.

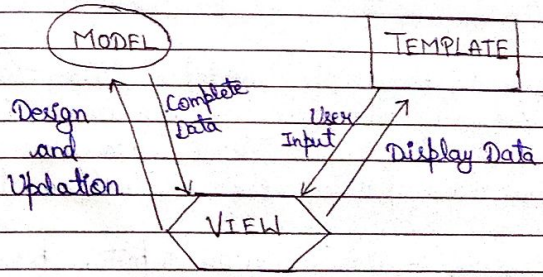
Ans:- Advantages :-

(i) Django is a framework and simple.
(ii) It works on python.
(iii) It is time effective.
(iv) It suits any kind of project.
(v) It is secure and upto-date.
(vi) It is backward compatible.
(vii) It has its own infrastructure.

- Discuss the role of Django in following the MVC / MVT architectural pattern.

Ans:- In the MVC / MVT architectural pattern, Django acts as the controller and view layer. The Model represents the data structure and business logic of the application, the view is responsible for presenting data of the user, and the controller handles user input and organize interactions between the Model and view. In Django's MVT pattern, the Model remains the same as in MVC, while the view is replaced by the Template, which represents the presentation layer. Django's URL dispatcher maps URLs to views functions, which serve as controllers in the MVC sense. This separation of concerns promotes code organization, maintainability, and scalability in web applications built with Django.

Figure :-



Date : / /
Page No.

Q2:- MVC vs. MTV:

- Define MVC (Model-View-Controller) architecture and explains its components (Model, View, Controller)

Ans:- MVC (Model-View-Controller) is a software architectural pattern used for developing user interface. It comprises three interconnected components :-

+ **MODEL** :- Represents the data and business logic of the application. It manages the data, defines rules, and responds to requests for information about its state.

+ **VIEW** :- Displays the model's data to the user and provides the interface for user interaction. It doesn't directly interact with the model but receives data from the controller to render.

+ **CONTROLLER** :- Handles user input, processes it, and interacts with both the model and the view. It updates the model's state based on user input and selects the appropriate view for presentation.

Date : / /
Page No.

- Compare and contrast MVC with Django's MTV (Model - Template - View) architecture.

Ans:-

	MVC	MTV
1) Model layer	MVC model represents the data and business logic of the application. It interacts with the database & manages the data.	MTV model is similar representing the data layer of the application, interaction with database & providing an interface to query and manipulate data.
2) View layer	MVC handle the presentation logic interpreting data from the model and ~~re~~ rendering it for the user interface.	MTV views represents the logic behind and returning appropriate response.
3) Controller	It manages user input updates the model and selects views to display.	MTV ~~the~~ the view handle both the logic processing and selects the appropriate template to render response.
4) Framework usage	MVC commonly associated with frameworks like Ruby and Rails.	MTV used by django a python web framework.
5) Flexibility & Easy of use	MVC provides a more traditional separation of concern, which can be beneficial for larger more complex application.	MTV simplifies development by reducing the no. of components and providing a clear structure.

Date: / /
Page No.

- Discuss how Django implements the MTV pattern and its benefits over traditional MVC.

Ans:- Django implements the MTV pattern by providing tools and conventions for organizing code and handling web requests:-

* MODEL :- Django provides an ORM (Object-Relational Mapping) to interact with the database, simplifying database operations and ensuring portability across different database engines.

* TEMPLATE :- Django's template engine allows developers to create dynamic HTML pages by embedding python code within HTML templates. This separation of logic and presentation promotes code readability and maintainability.

* VIEW :- Django's views encapsulate request-handling logic and business logic, promoting code reuse and modularization. Views are mapped to URL patterns, enabling clean and flexible URL routing.

⇒ Benefits of Django's MTV over traditional MVC include :-

* Rapid Development :- Django's built-in features and conventions streamline development, allowing developers to focus on building application logic rather than boiler plate code.

Date: / /
Page No.

* Scalability :- Django's MTV pattern facilities modular and maintainable code, making it easier to scale and extend applications as requirements evolve.

* Community Support :- Django has a large and active community, providing extensive documentation, third-party packages, and resources to support developers throughout the development process.

Q3:- Directory Structure :

- Explain the standard directory structure of a Django project.

Ans:- In Django the standard directory structure typically includes:

(i) Project Directory:- This directory is the main folder containing the entire django project.

(ii) manage.py :- A command line utility that lets you interact with the project, such as running development servers, creating migrations and more.

(iii) Project Name Directory :- This folder holds the setting URL configurations, and other configurations for the project.

(iv) settings.py :- This file contains all the project setting including database configuration, static files configuration, middleware, installed apps and more.

(v) urls.py :- This file defines the URL patterns for the project, mapping URLs to views.

(vi) wsgi.py :- A WSGI compatible web server gateway interface for deploying the application.

(vii) asgi.py :- A ASGI compatible web server gateway interface for asynchronous support.

(ix) App Directory :- This is where individual Django application reside. Each app usually contains its own models, views, templates and static files.

(x) Static Directory :- It contains static files such as CSS, Javascript and image files used by the project or apps.

(xi) Template Directory :- It stores HTML templates used by the project or apps.

(xii) Media Directory :- This is where user uploaded files are stored.

- Describe the purpose of each directory and its contents, including:

→ manage.py : Entry point for interacting with the Django project.
→ project_name/ : Directory containing project-specific settings and configurations.
→ app_name/ : Directory for individual Django applications.
→ static/ : Directory for static files such as CSS, JavaScript, and images.
→ templates/ : Directory for HTML templates used by Django views.

Ans:- (a) manage.py = Provides a command-line interface for various administrative tasks related to the Django project, such as creating applications, running the development server, and managing database migrations.

(b) project_name/ = Contains project-specific settings and configurations, including database settings, middleware configurations, installed apps, URL routing, and deployment configurations.

(c) app_name/ = Houses individual Django applications, each containing files and components specific to the functionality they provide. This directory helps modularize the project and maintain a separation of concerns.

(d) static/ = Stories static files required by the project, which are served directly to the

client by the web server. These files typically include CSS stylesheets, Javascript scripts, image, and other assets.

e. templates/ = Contains HTML template files used by Django views to generate dynamic HTML content. These templates can include placeholders for dynamic data and logic, allowing for the creation of dynamic web pages.

Q4:- Creating a Django Project:

- Provide step-by-step instructions on how to create a new Django project using the django-admin command.

Ans:- STEP 1:- Start the project
→ django-admin startproject my_first_project.

STEP 2:- Change the directory
→ cd my_firstproject

STEP 3:- Start the app.
→ py manage.py startapp testapp.

STEP 4:- Add application in the settings.py file
→ installed_app = ['testapp']

STEP 5:- Define views in views.py file from django shortcuts import render
→ from django.http import HttpResponse
def greeting_view(request):
return HttpResponse('<h1> Good Morning</h1>')

STEP 6:- ~~Start the Runserver~~
Define urls in the urls.py file
→ from testapp import views as v1
urlspattern = [path('myapp/', v1.greeting_view)]

STEP 7:- Start the Runserver
→ py manage.py runserver

- Explain the process of setting up project-level configurations in the settings.py file.

Ans:- The process of setting up project-level configura-tions in the 'settings.py' file:-

1.) Open the 'settings.py' file located inside the project directory ('project_name/').

2.) Configure the database settings by specifying the database engine, name, user, password, host, and port.

3.) Add the names of Django apps installed in your project to the 'INSTALLED_APPS' list.

4.) Configure the URLs for serving static files ('STATIC_URL') and media files ('MEDIA_URL').

5) Specify the directories where Django should look for HTML templates using the 'TEMPLATES' setting.

6) Optionally, set the 'DEBUG' setting to 'TRUE' for development and debugging purposes (ensure it's set to 'FALSE' in production).

- Demonstrate how to create a new Django application within the project using the manage.py command.

Ans:- To create a new Django application within your project using the 'manage.py' command follow these steps:

STEP1:- Open the terminal

STEP2:- Run the manage.py command
by manage.py startapp <app name>
by manage.py startapp called myapp

STEP3:- Add the application to Installed_apps.
INSTALLED_APPS = [
'myapp',
]

STEP4:- Define the view in the views.py file.

STEP5:- Define the urls in the urls.py file.

STEP6:- Start the server → by manage.py runserver.
by

Q5:- Understanding Views and Templates:

- Define views and templates in the context of Django.

Ans:- * VIEWS :-

→ Views in django are python functions or classes that receive web request and return web response. They encapsulate the logic to process and respond to incoming HTTP.

→ Views typically interact with models to retrieve data, platform actions and pass data to templates for rendering.

→ In django, views are typically defined as functions or methods within django app and are mapped to specific URL pattern using the URL dispathcher.

→ It can return various types of responses such as HTML content, JSON, data or error responses.

* TEMPLATES :-

→ Template is nothing but it is collection of HTML, CSS, bootstrap, XML.

→ It is used to render information to views function and send views function send that information to user.

→ Template allow you to separate the presentation layer from the business logic and are typically used by views to generate the final output sent to the client's web browser.

→ Django's template engine provides features like template inheritance template tags, filters and more facilitate dynamic content generation.

- Explain the role of views in handling user requests and generating responses.

Ans→ Views in Django play a crucial role in handling user requests by processing incoming HTTP requests, accessing data from the database or other sources, and generating appropriate HTTP responses.

→ When a user makes a request to a specific URL pattern defined in the project's URL configuration, Django's URL dispatcher routes the request to the corresponding view functions or class.

→ Views then execute the necessary business logic, such as querying the database, processing form submissions, or performing any other required operations.

→ Finally, views return an HTTP response, which can be a rendered HTML page, a JSON response, a redirect, or any other valid HTTP response.

- Discuss the purpose of templates in rendering HTML content dynamically.

Ans→ Templates in Django are used to generate dynamic HTML content by combining static HTML markup with dynamic data.

→ Templates contain placeholders and template tags, which allow for the insertion of dynamic content, control structures, and template inheritance.

→ Django's template engine processes these templates at runtime, replacing placeholders with actual data retrieved from views, and generates HTML output that is sent to the client's browser.

→ Templates facilitate the separation of presentation logic from application logic, making it easier to maintain and update the user interface without modifying underlying business logic.

Date : / /
Page No.

- Provide examples of defining views and templates within a Django application.

Ans:

Step 1:-
Start the project
⇒ django.admin startproject templateproject1

Step 2:-
Change the directory
⇒ cd templateproject1

Step 3:-
Start the app
⇒ by manage.py startapp testapp.

Step 4:-
Add the application in the settings.py file.
⇒ installed_app = ['testapp',]

Step 5:-
Create a template folder inside root project folder and introduce that folder into settings.py file
⇒ import os
TEMPLATE_DIR = os.path.join(BASE_DIR, 'template')
TEMPLATES = [
DIR = [TEMPLATE_DIR]
]

Date : / /
Page No.

Step 6:-
Right click on the template folder and create a HTML file.
course.html
⇒ <html>
<head></head>
<body bgcolor = "yellow">
 Welcome
<h1> Welcome html </h1>
<h2> course : {{ cname }}, duration : {{ duration }},
Fees : {{ fee }} </h2>
</body>
</html>

Step 7:-
Define views in views.py file from django shortcuts import render
⇒ from django.shortcuts import render
from django.http import HttpResponse
def course_view(request):
course = "Django"
duration = "two months"
Fees = 12000
course_details = {'cname': course,
'duration': duration,
'Fees': Fees}
return render(request, 'course.html', course_details)

Date : / /
Page No.

Step 8:-
Define urls in the urls.py file

⇒ From testapp import views as v1
urlspattern = [path('temp/', v1.course_view),
]

Step 9:-
Start the server
⇒ by manage.py runserver