Tops Technology

Module 16)
Python DB and Framework

Presented By : Nandni Vala

MVT Pattern Architecture

1.Django’s MVT (Model-View-Template) architecture and how it handles request-response cycles.

- Django's **MVT (Model-View-Template)** architecture is a design pattern used to organize the structure of a Django application. It separates the logic of handling data, user interface, and user interactions into distinct components, making the application modular and maintainable. Here's an overview of the three components and how Django handles the request-response cycle:
- **MVT Components**
- **Model**:
    - Represents the data layer of the application.
    - Handles database interactions, such as fetching, saving, and updating data.
    - Defined using Django's Object-Relational Mapping (ORM), which allows you to interact with the database using Python code.

- **View**:
- Acts as the business logic layer.
- Processes incoming HTTP requests, fetches data from the model if needed, and decides what data to send to the template.
- Returns an HTTP response, often rendered using a template.
- In Django, views can be defined as functions or class-based views.
- **Template**:
- Handles the presentation layer.
- Defines how the data sent by the view should be displayed to the user.
- Written in HTML and enhanced with Django Template Language (DTL) for dynamic content.
- **Request-Response Cycle in Django MVT**
- **User Request**:
- The user interacts with the application (e.g., entering a URL or submitting a form).
- A web browser sends an HTTP request to the Django server.

- **URL Routing**:
- Django's URL dispatcher (urls.py) maps the incoming request URL to the corresponding view.
- If no match is found, Django returns a 404 error.
- **View Processing**:
- The mapped view function (or class-based view) is called.
- The view processes the request, performs any necessary business logic, and interacts with the model to retrieve or update data.
- **Model Interaction**:
- If data is required, the view interacts with the model to fetch or manipulate it.
- The model communicates with the database through Django ORM.
- **Template Rendering**:
- The view prepares the data to be presented and renders a template.
- The template dynamically generates an HTML response using the provided data.
- **Response to User**:
- The rendered template is converted into an HTTP response and sent back to the user's browser.
- The browser displays the resulting page.

➢ **Example :**

URL Configuration (urls.py):

```
from django.urls import path
from . import views
urlpatterns = [
   path('hello/', views.hello_world, name='hello_world'),
]
```

View (views.py):

```
from django.shortcuts import render
def hello_world(request):
    context = {'message': 'Hello, World!'}
    return render(request, 'hello.html', context)
```

Tamplate (hello.html):

```
<!DOCTYPE html>
<html>
<head>
    <title>Hello Page</title>
</head>
<body>
    <h1>{{ message }}</h1>
</body>
</html>
```

Key Advantages of Django's MVT

- **Separation of Concerns**: Makes the application modular and easier to maintain.
- **Rapid Development**: Django's built-in features like ORM and templating speed up development.
- **Scalability**: The architecture supports large-scale applications effectively.
