

Tops Technology

Module 16)

Python DB and Framework

Presented By : Nandni Vala

URL Patterns and Template Integration

1. Setting up URL patterns in urls.py for routing requests to views

➤ **Steps to Set Up URL Patterns in urls.py**

➤ **Import Required Modules:**

- Import the path and/or re_path functions from django.urls.
- Import the views from your app.

➤ `from django.urls import path`

➤ `from . import views`

➤ **Define URL Patterns:**

➤ Use `path()` for simple patterns and `re_path()` for regex-based patterns.

➤ Associate a URL path with a view function or class-based view.

➤ `urlpatterns = [`

➤ `path('home/', views.home_view, name='home'), # Function-based view`

➤ `path('about/', views>AboutView.as_view(), name='about'), # Class-based view`

➤ `]`

Include Parameters in URLs:

- Define dynamic parts of a URL using angle brackets (< >).
 - Specify types like <int:id> or <str:username>.
 - urlpatterns = [
 - path('product/<int:id>/', views.product_detail, name='product_detail'),
 -]
 - **Namespace Your URLs** (Optional):
 - For apps with multiple URLs, use namespaces for better organization.
 - In the app's urls.py:
 - app_name = 'myapp'
 - urlpatterns = [
 - path('home/', views.home_view, name='home'),
 -]
 - In the project's urls.py:
 - from django.urls import include
 - urlpatterns = [
 - path('myapp/', include('myapp.urls')),
 -]
-

➤ **Route to Static and Media Files (Optional):**

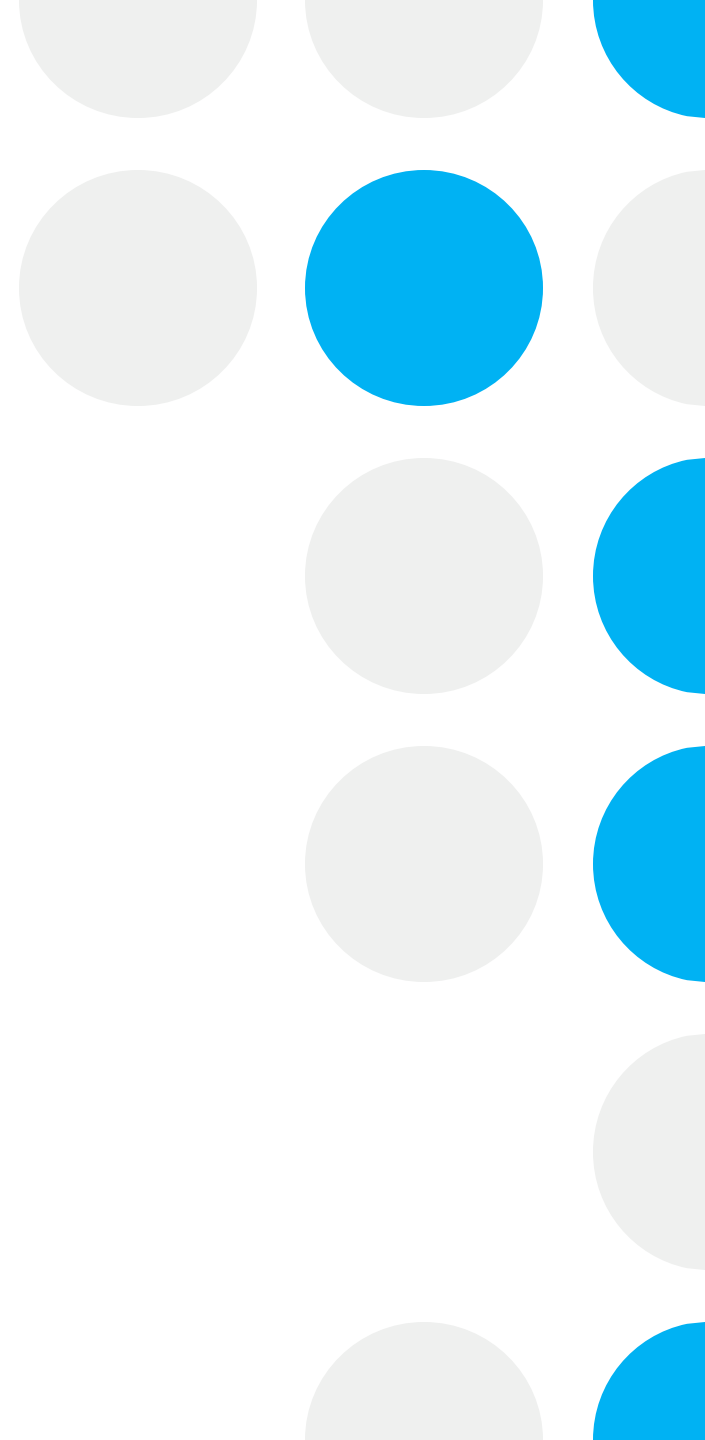
- During development, serve static and media files using `django.conf.urls.static`.
- `from django.conf import settings`
- `from django.conf.urls.static import static`
- `urlpatterns += static(settings.MEDIA_URL, document_root=settings.MEDIA_ROOT)`



2. Integrating templates with views to render dynamic HTML content.

➤ **Create a Template**

- Store HTML files in a templates directory within your app or in a central templates folder for the project.
 - Example: app_name/templates/app_name/template_name.html
 - `<!-- templates/myapp/home.html -->`
 - `<!DOCTYPE html>`
 - `<html>`
 - `<head>`
 - `<title>{{ title }}</title>`
 - `</head>`
 - `<body>`
 - `<h1>Welcome, {{ user_name }}!</h1>`
 - ``
 - `{% for item in items %}`
 - `{{ item }}`
 - `{% endfor %}`
 - ``
 - `</body>`
 - `</html>`
-



➤ **Configure Template Directory**

➤ In settings.py, ensure Django knows where to find templates:

➤ `TEMPLATES = [`

➤ `{`

➤ `'BACKEND': 'django.template.backends.django.DjangoTemplates',`

➤ `'DIRS': [BASE_DIR / 'templates'], # Add your custom templates
directory`

➤ `'APP_DIRS': True,`

➤ `'OPTIONS': {`

➤ `'context_processors': [`

➤ `'django.template.context_processors.request',`

➤ `'django.contrib.auth.context_processors.auth',`

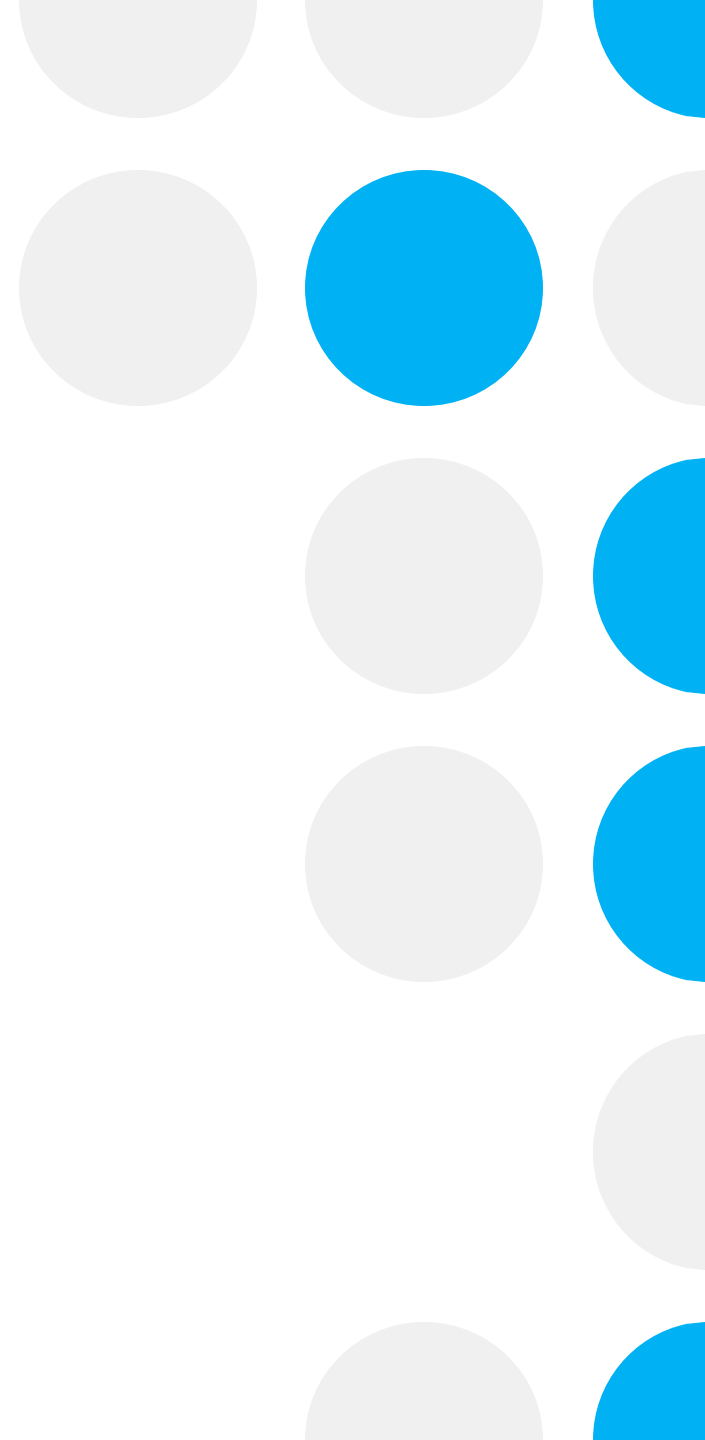
➤ `],`

➤ `},`

➤ `},`

➤ `]`

- **. Create a View**
 - Use `render()` to combine the template with the data.
 - Example (Function-Based View):
 - `from django.shortcuts import render`
 - `def home_view(request):`
 - `context = {`
 - `'title': 'Home Page',`
 - `'user_name': 'John Doe',`
 - `'items': ['Apples', 'Bananas', 'Cherries']`
 - `}`
 - `return render(request, 'myapp/home.html', context)`
 - Update `urls.py`
 - Route the URL to the appropriate view.
 - `from django.urls import path`
 - `from . import views`
 - `urlpatterns = [`
 - `path("", views.home_view, name='home'), # Function-Based View`
 - `]`
-



➤ How It Works

➤ View:

- Prepares the data (context) and sends it to the template.

➤ Template:

- Dynamically renders the HTML using the context variables.

➤ Browser:

- Displays the rendered HTML to the user.

