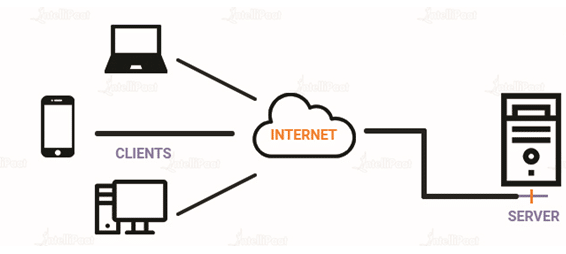


* Django/flask is a web framework not a library.
* Framework is a structure intended to serve as a guide for building applications that expands the structure into something useful.
* Framework is an application uses our code to build an application.
* Library is a predefined code that we use to build an application.
* Django is a python based free and open source web framework which follows the Model View Template (MVT) architectural pattern.
* Django is maintained by Django software foundation.
* Django based applications are YouTube, Dropbox, Mozilla, Spotify, Instagram, etc.
* Features of Django are Fast Development, Provide web server for development and testing, scalability, Loaded with apps, Secure, etc.

**Client Server Architecture :**



**Django Installation :**

* Check whether python is installed or not (python –version or python).
* Check the pip command (pip –version or pip).
* Install Django (pip install django).
* Check the Django (django-admin –version or python –m django --version).

All applications in Django are called as web app and All projects in Django are called as website.

Django is a framework. We can create our project with simple command.

**Django Project Creation :**

* Create a workspace directory.
* Create a first project (Website).

1. django-admin startproject firstproject or python -m django startproject firstproject

* \_\_init\_\_.py is an empty file its presence make the folder as python package.
* Setting.py contains project related setting statements.
* Urls.py contains url pattern of web pages.
* Wsgi.py is a webserver gateway interface, it is a special file, no modification in this file is required.
* Asgi.py is an asynchronous server gateway interface, like wsgi.py, it describes a common interface between a python web app and the web server.
* Manage.py is a program to do some initial stuff, never edit this file.

1. Come in your project directory (cd firstproject).
2. Run the web server (python manage.py runserver).

When project is not creating then create virtual environment.

1. Create virtual environment (py –m venv myworld).
2. Activate virtual environment (myworld\scripts\activate.bat).
3. Go to your environment or workspace (cd myworld).
4. Create project (django-admin startproject firstproject).
5. Change directory (cd firstproject).
6. Run server (python manage.py runserver).

Create a web app (python manage.py startapp myapp).

Update setting.py in child myproject directory (in installed apps).

Define the view in myapp view.py file :

|  |
| --- |
| from django.http import HttpResponse  def greeting(request):  s = "Hello and welcome to my firstproject"  return HttpResponse(s)  def myclass(request):  c = "This is my AIDS class"  return HttpResponse(c)  def myname(request):  n = "Nikhlesh Shukla  return HttpResponse(n) |

Set url in myproject urls.py file.

|  |
| --- |
| from myapp import views  urlpatterns = [  path('hello/',views.greeting),  path('class/',views.myclass),  path('name/',views.myname),  path('admin/', admin.site.urls),  ] |

Request and Response :

**Multiple Apps Multiple Views :** Steps to perform.

* Create project (django-admin startproject firstproject or python -m django startproject firstproject).
* Create apps (python manage.py startapp myapp) for one app.
* Update settings.py file.
* Define views in views.py file.
* Set urls in urls.py file.

**App Level urls :** for them we have to create a new urls file.

* Click app name.
* Click new file.
* Assign name urls.py
* First app content in urls.py file :

from django.urls import path

from third1app import views

urlpatterns = [

path('home/',views.home),

path('about/',views.about),

path('contact/',views.contact)

]

* Second app content in urls.py file :

from django.urls import path

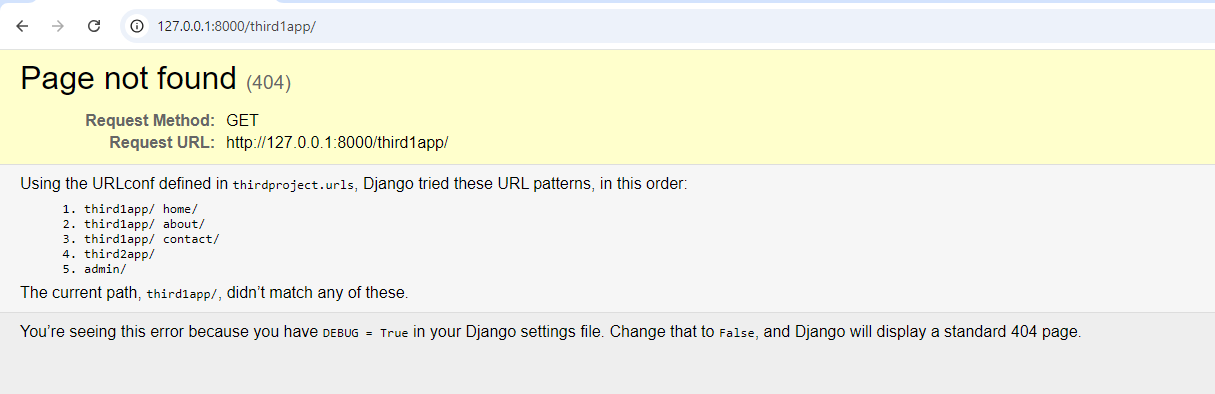
from third2app import views

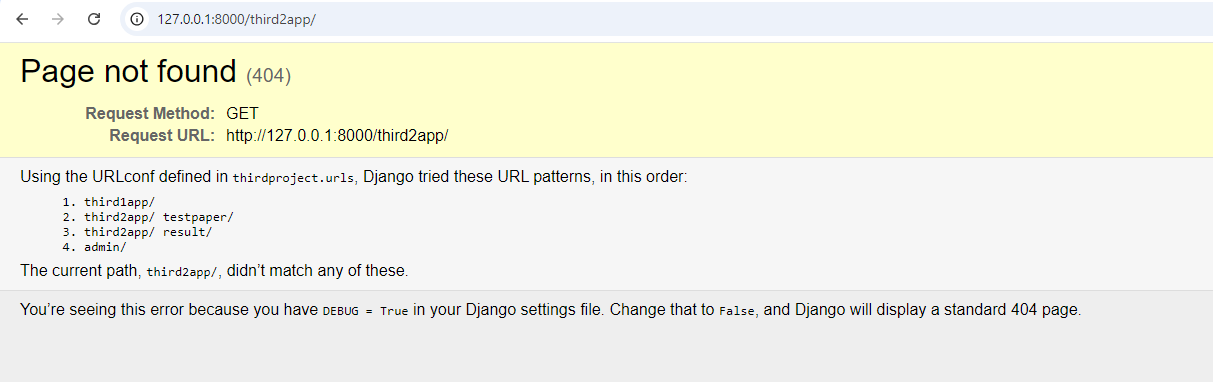
urlpatterns = [

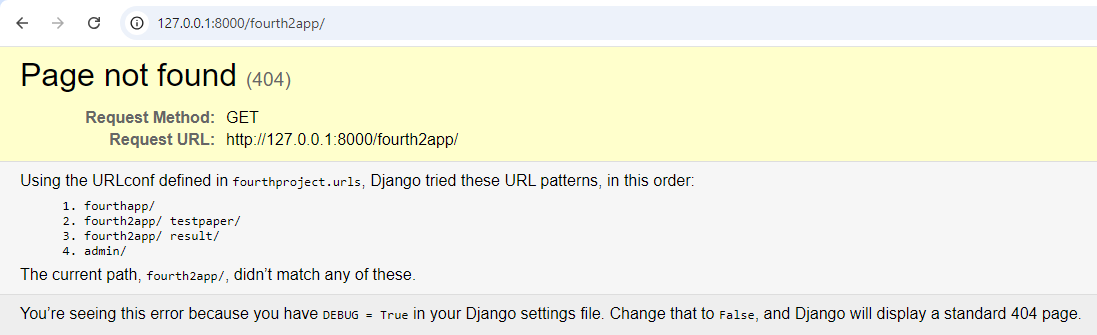
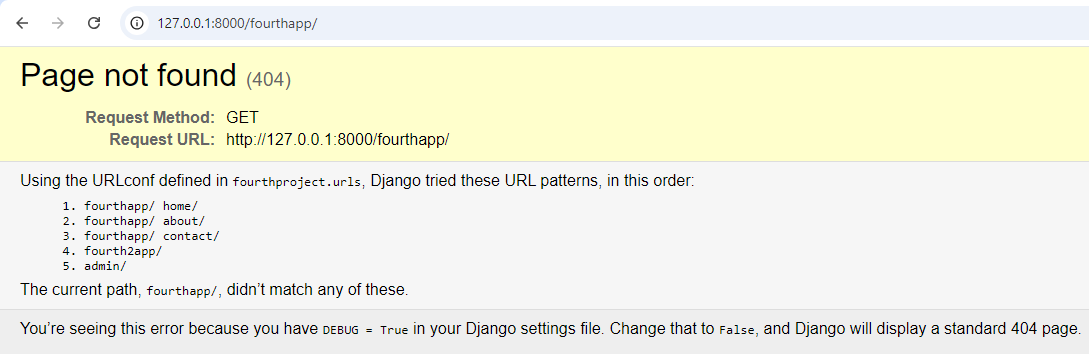
path('testpaper/', views.testpaper),

path('result/', views.result)

]

* After running server :





**Dynamic Content in Template :**

Jinja2 technology is used to implement this concept in html file which is created in app.

Usually we cannot write language ingredients like variable decision and iterative controls in html, but with the help of Jinja2, it become possible.

There are three things related to templates you need to explore :

* Template Tags
* Template Variable
* Template filter

**Template Variable :**

Template variable can be used in html using {{variable}} (see quiz\_eg project).

**Template Tags :**

We can perform programming logic in html using template tags.

There are several built-in tags and we can define our own custom template tag.

Example : {% Template Tag %}

Some built-in template tags are :

* block : Define a block that can be overwritten by child template.
* csrf\_token : This is used for CSRF (Cross Site Request Forgiries) Protection.
* for : Loop.
* extends : Signals that this template extends a parent template.
* if : Decision control.
* load : Load a custom template tag.
* now : Display current date and time or both.

**Template Filters :**

Filters are used to transform the values of variables and tags arguments.

There are several built-in template filters and we have an option to define our own, known as custom template filters.

* add : {{ variable | add : “2” }}
* capfirst : {{ text | capfirst }}
* cut : {{ text | cut : “ ” }} (Cut arguments from the text).
* dictsort : {{ value | dictsort : “key” }} (Sort list of dict by given key).
* dictsortreversed :
* join : Join a list with a string.
* length : Count length.
* lower :
* upper :
* random : Returns a random item from the list.
* wordcount :
* static :

**Extending Templates & Static Folder :**