TABLE OF CONTENTS

CHAPTER 1

INTRODUCTION TO DJANGO

WHAT IS DJANGO USED FOR?

Django is an open-source python web framework used for rapid development, pragmatic, maintainable, clean design, and secures websites. A web application framework is a toolkit of all components need for application development.

IS DJANGO BACKEND OR FRONTEND?

Django is a framework, not a language. Python is the language in which Django is written. Django is a collection of Python libs allowing you to quickly and efficiently create a quality Web application and is suitable for both frontend and backend.

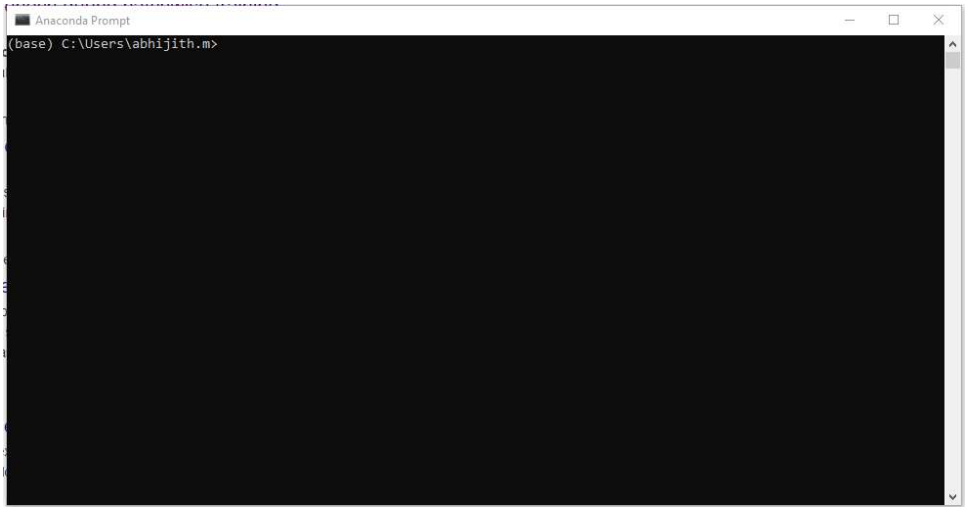
INSTALL DJANGO

- ✓ Open anaconda prompt
- ✓ Create a new environment in anaconda prompt
- ✓ conda create -n python36 python=3.6
- ✓ This command will create a new environment in anaconda. Name of environment is python36
- ✓ then activate your newly created environment using following command.
- ✓ activate python36
- ✓ Now the environment is set. You can do all of Django projects in this environment.
- ✓ Simply an environment is a folder containing all necessary packages of your project
- ✓ You should need to install Django package in this environment.
- ✓ $ pip install django

CREATE NEW PROJECT

First step is to activate environment in anaconda prompt

- ➢ Open anaconda prompt

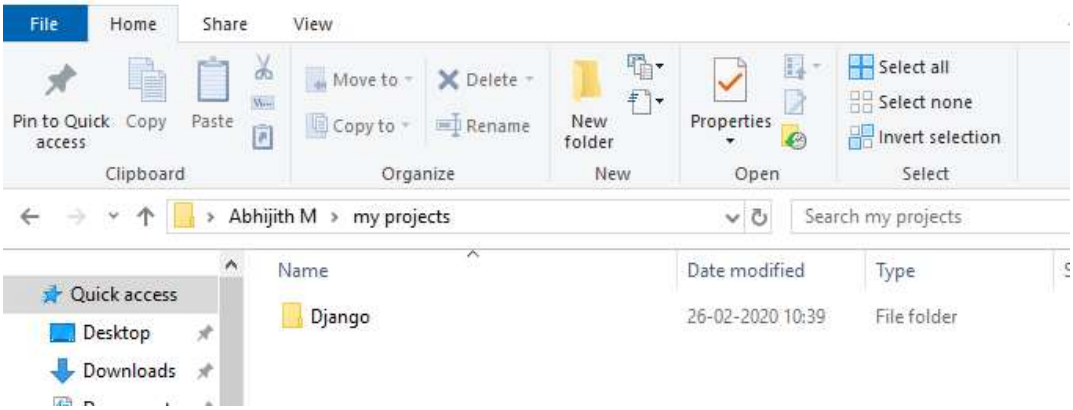


- activate python36

Anaconda Prompt

```
(base) C:\Users\abhijith.m>activate python36

(python36) C:\Users\abhijith.m>
```

- Create a folder for Django projects in user directory



- Move to project directory
- To change directory in command prompt use the following code
- cd “path to your project directory”

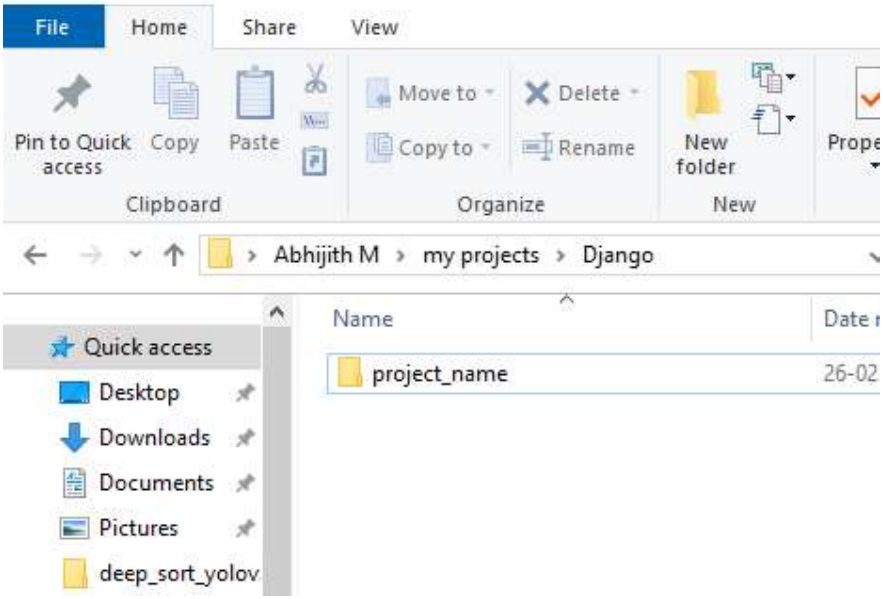
Anaconda Prompt

```
(base) C:\Users\abhijith.m>activate python36

(python36) C:\Users\abhijith.m>cd "my projects\Django"
```

- (python36) C:\Users\abhijith.m\Django>***django-admin startproject project_name***
- “*django-admin startproject project_name* “command will create a new project with name *project_name*. A new folder named “*project_name*” will create to the current directory.

Anaconda Prompt

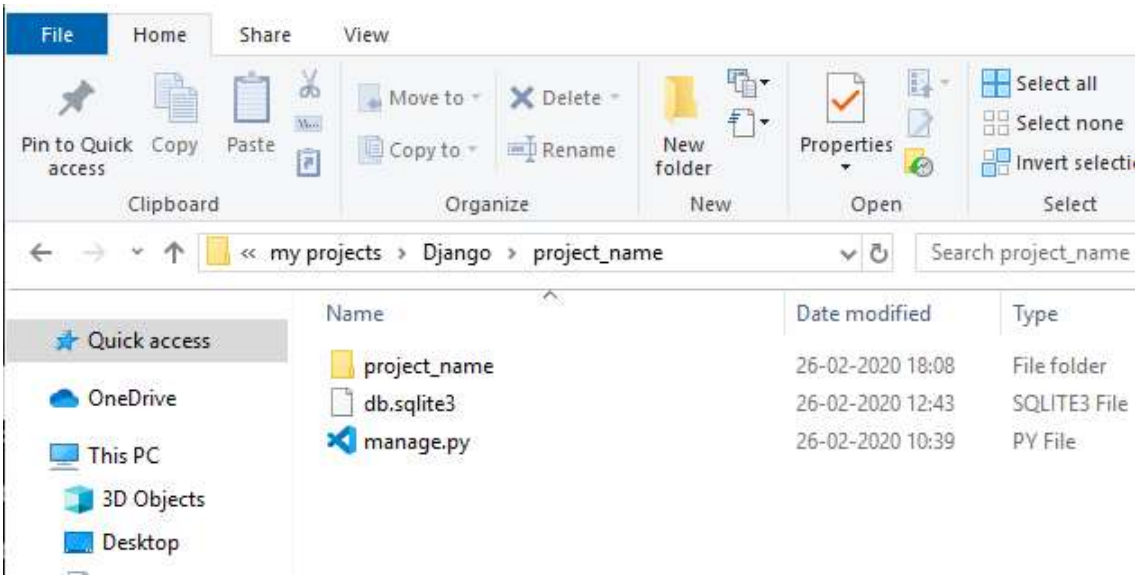
```
(base) C:\Users\abhijith.m>activate python36

(python36) C:\Users\abhijith.m>cd "my projects\Django"

(python36) C:\Users\abhijith.m\my projects\Django>django-admin startproject project_name
```



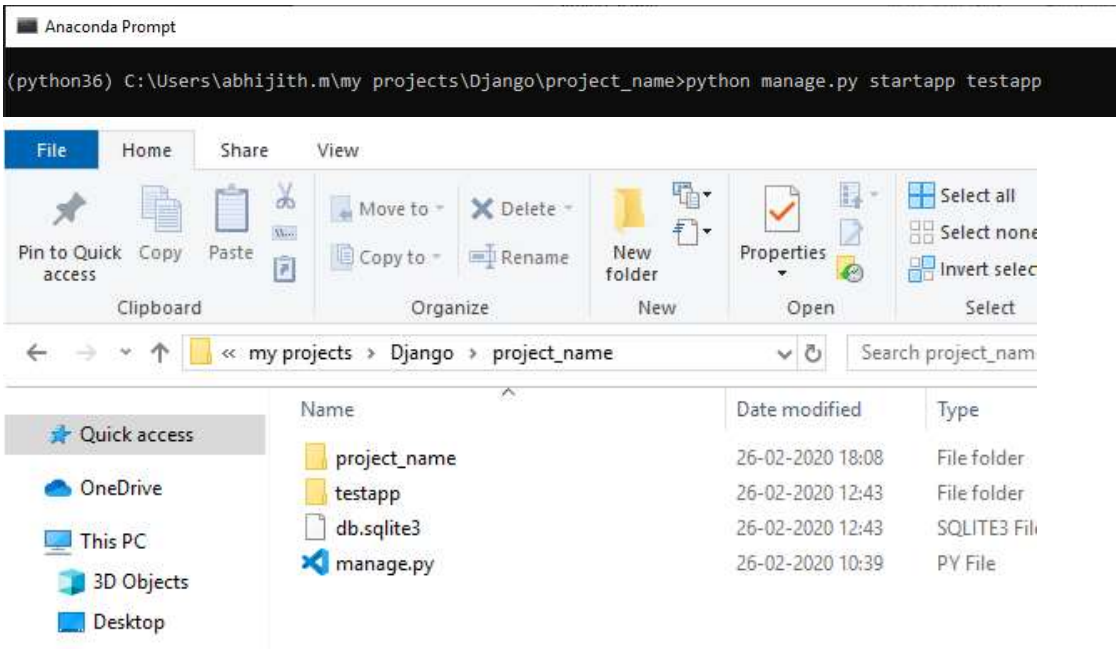
- (python36) C:\Users\abhijith.m\Django>***cd project_name***
- *cd project_name* will change your current directory in prompt to *project_name*

```
Anaconda Prompt
(base) C:\Users\abhijith.m>activate python36

(python36) C:\Users\abhijith.m>cd "my projects\Django"

(python36) C:\Users\abhijith.m\my projects\Django>django-admin startproject project_name

(python36) C:\Users\abhijith.m\my projects\Django>cd project_name
```

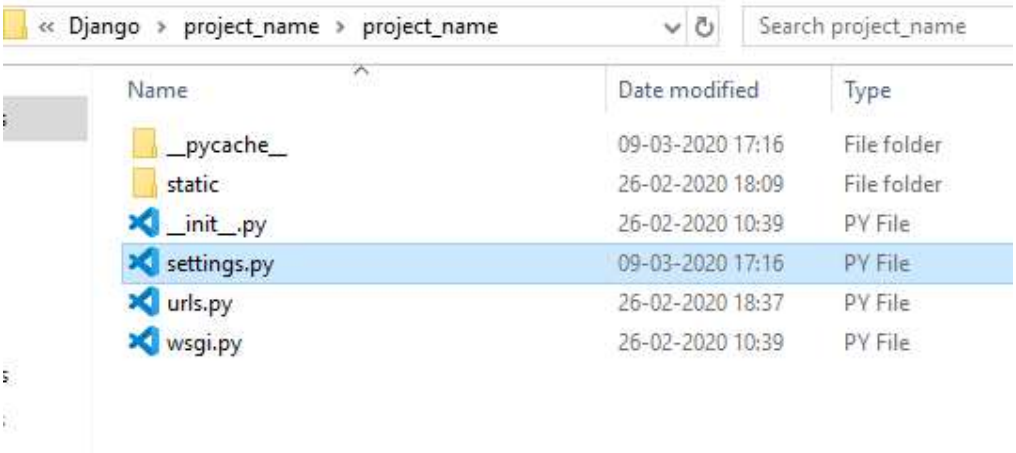
- When we first initialize the project, these files are automatically created in project folder.



- (python36) C:\Users\abhijith.m\Django\withoutrest>***python manage.py startapp testapp***
- *Manage.py* is a Django controller file. "*python manage.py startapp testapp*" this command will create a new app with name "*testapp".*



- Open project folder - ***../project_name/project_name/***
- Open ***settings.py*** file



- Add your app name to **INSTALLED_APPS**

```
# Application definition

INSTALLED_APPS = [
    'count_people.apps.CountPeopleConfig',
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',
    'testapp', # <-- your app name
]
```

- Now the project creation is almost completed. But the view is not created. For that we should need some website design knowledge. Let's discuss this in Chapter 2

CHAPTER 2

WEBSITE DESIGN BASICS

HTML

CSS

JS

CHAPTER 3

WEB TEMPLATES

WHAT IS JINJA TEMPLATE IN DJANGO?

Template Engines. Template engines take in tokenized strings and produce rendered strings with values in place of the tokens as output. Templates are typically used as an intermediate format written by developers to programmatically produce one or more desired output formats, commonly HTML, XML or PDF.

WHAT IS TEMPLATE ENGINE IN DJANGO?

Django's template engine provides a powerful mini-language for defining the user-facing layer of your application, encouraging a clean separation of application and presentation logic. Templates can be maintained by anyone with an understanding of HTML; no knowledge of Python is required.

WHAT IS JINJA TEMPLATE IN DJANGO?

Jinja is a web template engine for the Python programming language and is licensed under a BSD License created by Armin Ronacher. It is similar to the Django template engine but provides Python-like expressions while ensuring that the templates are evaluated in a sandbox.

DOES DJANGO NEED HTML?

But when you ask about Django it is a web development framework, So you should learn html, css and javascript is very essential. ... But when you ask about Django it is a web development framework, So you should learn html, css and javascript is very essential.

WHAT IS A TEMPLATE USED FOR?

A template is a form, mold, or pattern used as a guide to making something. ... A document in which the standard opening and closing parts are already filled in is a template that you can copy and then fill in the variable parts.

HOW DO I ADD A TEMPLATE TO DJANGO PROJECT?

Step 1: Create the Templates Folder. In this step we'll create the templates directory.

Step 2: Create new html file.

Step 3: Render your template to Django app.

After creating a new project do the following steps to create new template.

Step 1: Create the Templates Folder. In this step we'll create the templates directory.

- Open the project folder *(../project_name)*
- open "*testapp*" folder
- In this folder you need to create new folder named "*templates*" (location:- *../project_name/testapp/)*

Step 2: Create new html file.

2.1 Create a new "*index.html*" file in *(../project_name/testapp/templates)*
- *index.html* file

```
<html>
  <head>
  </head>
  <body>
    <h1>Django template</h1>
  </body>
</html>
```

Step 3: Render your template to Django app.

- To show this *index page* in website do the following steps,

3.1 Create new view in views.py (location:- *../project_name/testapp/views.py*)

```
from django.shortcuts import render

# Create your views here.
def MyView(request):
    return render(request,template_name="index.html", context={})
```

3.2 Open *urls.py* file (location:- *../project_name/project_name/urls.py)*

- Import views to this file

```
from testapp import views
```

- Add url path to map you view
- *urls.py* file >

```
from django.contrib import admin
from django.urls import path
from testapp import views

urlpatterns = [
    path('admin/', admin.site.urls),
    path('mytemplate/', views.MyView),
]
```

The final step is to start server

- open anaconda prompt
- move to project folder
- *(python36) C:\Users\abhijith.m > cd my projects\\Django\\project_name*
- This is my project location you can replace this location with your project path
- Then start running your server
- *(python36)C:\Users\abhijith.m\my projects\Django\project_name>python manage.py runserver*

```
Watching for file changes with StatReloader
Performing system checks...

System check identified no issues (0 silenced).

You have 17 unapplied migration(s). Your project may not work properly until you apply t
he migrations for app(s): admin, auth, contenttypes, sessions.
Run 'python manage.py migrate' to apply them.
February 26, 2020 - 12:43:57
Django version 2.2.3, using settings 'project_name.settings'
Starting development server at http://127.0.0.1:8000/
Quit the server with CTRL-BREAK.
```

- Open your web browser
- Search this url (http://127.0.0.1:8000/mytemplate/) in browser

PASSING CONTEXT TO TEMPLATES

WHAT IS CONTEXT IN DJANGO?

When you use a Django Template, it is compiled once (and only once) and stored for future use, as an optimization. A template can have variable names in double curly braces, such as *{{ myvar1 }}* and *{{ myvar2 }}*.

A Context is a dictionary with variable names as the key and their values as the value. Hence, if your context for the above template looks like: *{myvar1: 101, myvar2: 102}*, when you pass this context to the template render method, *{{ myvar1 }}* would be replaced with *101* and *{{ myvar2 }}* with *102* in your template. This is a simplistic example, but really a Context object is the context in which the template is being rendered.

WHAT ARE VIEWS IN DJANGO?

Django views are a key component of applications built with the framework. At their simplest they are a Python function or class that takes a web request and return a web response. Views are used to do things like fetch objects from the database, modify those objects if needed, render forms, return HTML, and much more.

WHAT IS RENDER IN PYTHON?

Combines a given template with a given context dictionary and returns an *HttpResponse* object with that rendered text. *render()* is the same as a call to *render_to_response()* with a context_instance argument that forces the use of a *RequestContext*.

WHY WE USE RENDER IN DJANGO?

render() combines a given template with a given context dictionary and returns an *HttpResponse* object with that rendered text. Django does not provide a shortcut function which returns a *TemplateResponse* because the constructor of *TemplateResponse* offers the same level of convenience as *render()* .

HOW TO PASS CONTEXT TO TEMPLATE?

- views.py >

```
from django.shortcuts import render

# Create your views here.
def MyView(request):
    myvar1 = "id1"
    myvar2 = "id2"
    context = {myvar1: 101, myvar2: 102}
    return render(request,template_name="index.html", context=context)
```

- Here we passing two variables *myvar1* and *myvar2*. Value of *myvar1* is "*id1*" so in template page we use *id1* for accessing the value of *id1.* The *{{ varname }}* is the syntax used to read the value of Django variable.
- In *index.html* page update the code >

```
<html>
    <head>
    </head>
    <body>
        <h1>Django template</h1>
        {{id1}}, {{id2}}
    </body>
</html>
```

- Open your browser and refresh the webpage (http://127.0.0.1:8000/mytemplate/)

CREATE A BASE TEMPLATE

WHAT IS BASE TEMPLATE IN DJANGO?

A base template is the most basic template that you extend on every page of your website. You used the template tag *{% block %}* to make an area that will have HTML inserted in it. That HTML will come from another template that extends this template (base. html).

WHAT IS TEMPLATE INHERITANCE IN DJANGO?

Template inheritance. The most powerful – and thus the most complex – part of Django's template engine is template inheritance. Template inheritance allows you to build a base “skeleton” template that contains all the common elements of your site and defines blocks that child templates can override.

HOW CAN I CREATE BASE TEMPLATE IN DJANGO?

1. Create a new *base.html* page in template folder

```
project_name
└────testapp
    └────templates
         base.html
         index.html
```

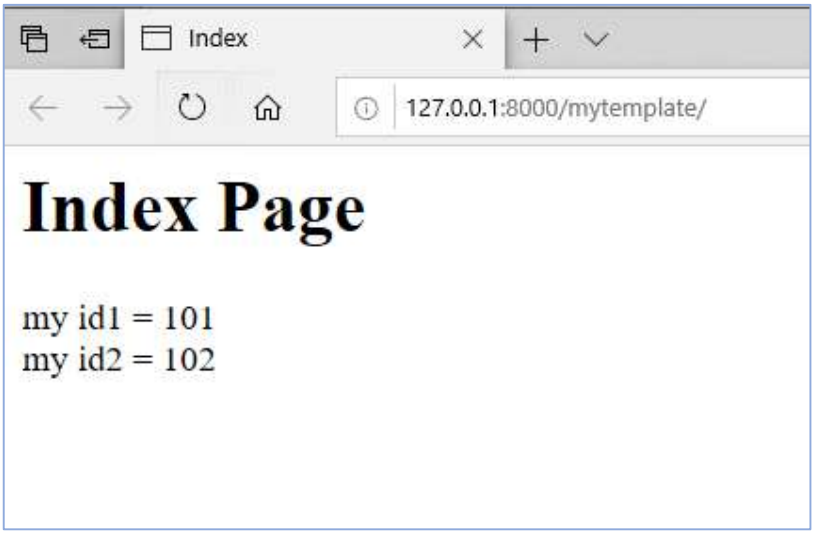
2. In *base.html* page add the following code.

```
<html>
  <head>
    <title> {% block title %} {% endblock %}</title>
  </head>

  <body>
    {% block index_page %} {% endblock %}
  </body>
</html>
```

3. In *index.html* update the code >

```
{% extends 'base.html' %}

{% block title %}
Index
{% endblock %}
{% block index_page %}
   <h1>Index Page</h1>
   my id1 = {{id1}} <br>
   my id2 = {{id2}}
{% endblock %}
```

4. Refresh browser



Let's create a django project *musicproject*. Inside the project create an app called *music.*

- (music/templates/music) > create a new file *'base.html'* inside music folder.

```
<!DOCTYPE html>
<html>
   <head>
<title>{% block title %} {% endblock %}</title>
   </head>
   <body>
      <nav class="navbar navbar-inverse">…
      </nav>
      {% block body %}
      {% endblock %}
   </body>
</html>
```

- (music/templates/music/index.html) > create a new file *'index.html'* inside music folder.

```
{% extends 'music/base.html' %}

{% block title %} Album title {% endblock %}
{% block body %}
   {% if all_albums %}
         # statement(s)
   {% endif %}
{% endblock %}
```

- (music/templates/music/details.html) > also create *details.html.*

GENERIC VIEW

WHAT IS GENERIC DJANGO?

Django offers an easy way to set those simple views that is called generic views. Unlike classic views, generic views are classes not functions. Django offers a set of classes for generic views in Django. views. generic, and every generic view is one of those classes or a class that inherits from one of them.

WHAT ARE CLASS-BASED VIEWS?

Class-based views. A view is a callable which takes a request and returns a response. This can be more than just a function, and Django provides an example of some classes which can be used as views. These allow you to structure your views and reuse code by harnessing inheritance and *mixins*.

WHAT IS GET AND POST METHOD IN DJANGO?

Django's login form is returned using the POST method, in which the browser bundles up the form data, encodes it for transmission, sends it to the server, and then receives back its response. GET , by contrast, bundles the submitted data into a string, and uses this to compose a URL.

WHAT IS AS_VIEW() IN DJANGO?

as_*view()* is the function(class method) which will connect my *MyView* class with its url. From django docs: classmethod as_view(**initkwargs) Returns a callable view that takes a request and returns a response: You just can't use class-based views like you could in normal function-based views.

HOW CAN I CREATE A GENERIC VIEW IN DJANGO?

- Open *views.py*, delete *MyView* function and create new class-based view
- ***from django.views.generic import View*** – is a default function in Django for Generic view.

```
from django.shortcuts import render
from django.views.generic import View

# Create your views here.
class MyGenericView(View):
    def get(self, request, *args, **kwargs):
        myvar1 = "id1"
        myvar2 = "id2"
        context = {myvar1: 101, myvar2: 102}

        return render(request, template_name="index.html", context=context)
```

- Now the function name is changed so we need to map this function in *urls.py* page
- In *urls.py* update path

```
from django.contrib import admin
from django.urls import path
from testapp import views

urlpatterns = [
    path('admin/', admin.site.urls),
    path('mytemplate/', views.MyGenericView.as_view()),
]
```

- Refresh browser. No changes will be found. But the code works fine.

FOR LOOP IN TEMPLATE

ADDING STATIC FILES

- Create a '*static*' folder in parent directory

```
project_name
└─────static
└─────testapp
    └─────templates
          base.html
          index.html
```

- Open *settings.py* (location:- **../project_name/project_name/settings.py**)
- Add this code to the bottom of *setting.py* file

```
STATIC_URL = '/static/'

# Add these new lines
STATICFILES_DIRS = (
 os.path.join(BASE_DIR, 'static'),
)

STATIC_ROOT = os.path.join(BASE_DIR, 'staticfiles')
```

- In static folder create folder css and js
- Inside this folder you can add your javascript and css files

```
project_name
└─────static
    └─────css
          style.css
    └─────js
          myscript.js
```

- *{% load static %}* will load static file link to html page

```
{% load static %}
<link rel="stylesheet" type="text/css" href="{% static 'css/style.css' %}" >
```

- Complete code of base.html page

```
<html>
   <head>
      <title> {% block title %} {% endblock %}</title>
      {% load static %}
      <link rel="stylesheet" type="text/css" href="{% static 'css/style.css' %}" >
   </head>

   <body>
      {% block index_page %} {% endblock %}
   </body>
</html>
```

- style.css (location:- ../project_name/static/css/style.css)

```
body {
   background-color: antiquewhite;
}
```

- refresh your browser

URL

WHAT IS URL NAMING IN DJANGO?

Django offers a way to name urls so it's easy to reference them in view methods and templates. The most basic technique to name Django urls is to add the name attribute to url definitions in *urls.py* .

HOW DO I NAME A URL IN DJANGO?

- Add the name attribute to url definitions in urls.py .

```
from django.contrib import admin
from django.urls import path
from testapp import views

urlpatterns = [
    path('admin/', admin.site.urls),
    path('mytemplate/', views.MyGenericView.as_view(), name="mytemplate"
),
]
```

USING {% URL ??? %} IN DJANGO TEMPLATES

- If you add a name attribute to the path, you can refer this name in html page.

```
<form action="{% url 'logout_view' %}">
```