Django Tutorial

# Introduction

This is a tutorial that follows Corey Schafer’s Django series videos.

Video series: <https://www.youtube.com/watch?v=UmljXZIypDc&list=PL-osiE80TeTtoQCKZ03TU5fNfx2UY6U4p>

Youtube Channel : <https://www.youtube.com/channel/UCCezIgC97PvUuR4_gbFUs5g>

# Basics

## Basic Django start commands:

django-admin startproject projectname # creates new project

python manage.py runserver # runs the server

python manage.py migrate # creates a a SQlite3 database for your project.

This will create a new django project.

Making a new app within our django project.

A app basically means a part of our site . For eg. I created a blog app that represent the blog part of our website:

We create a new app with the following way:

We navigate to our project root folder and type in:

**python manage.py startapp blog**



## How do we add paths to our website :

We navigate to our newly created app ‘views.py’ module and add the following:

Think of views as a module that represents what will be shown when you use the functions from it.

from django.shortcuts import render  
from django.http import HttpResponse

def home(request):  
 return HttpResponse('<h1>Blog Home Page</h1>')  
  
def about(request):  
 return HttpResponse('<h1>About Page</

These are functions that will run when we got on home and about page (if we assign those functions to them)

Next step is making a new .py module called ‘urls’ which will contain the path and the functions to the pages.

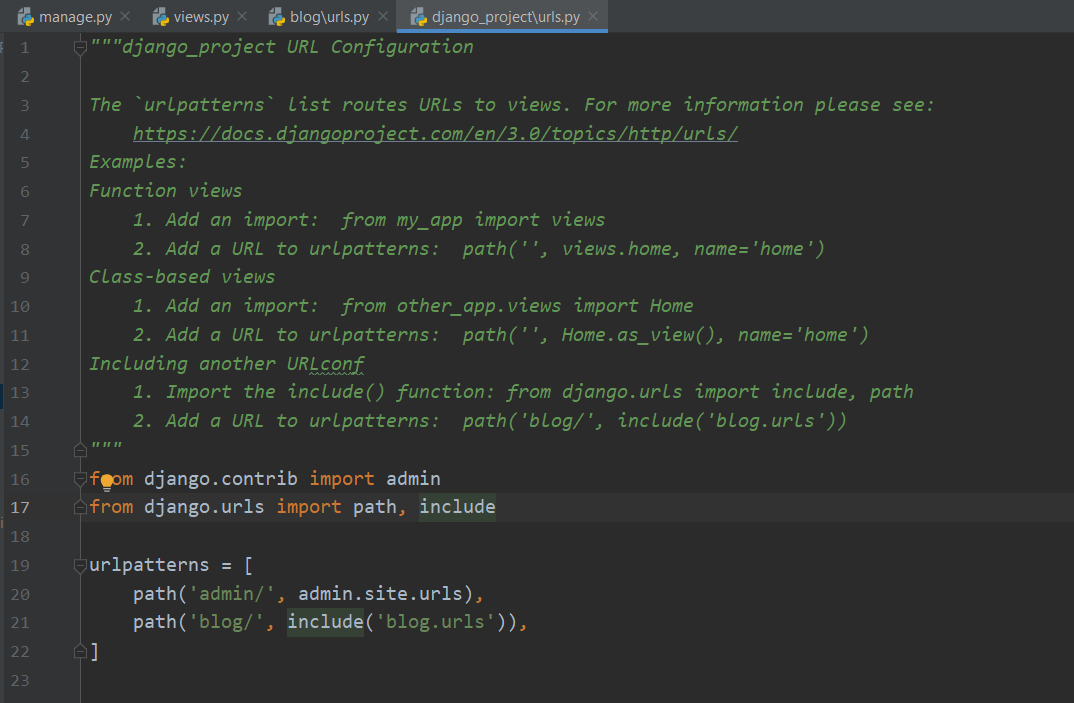
from django.urls import path # path function from django.urls  
from . import views # '.' means from current folder  
  
urlpatterns = [  
 path('', views.home, name='blog-home'),  
 path('about/', views.about, name='home-about')  
]

The upper code basically makes our website have two more ‘pages’:

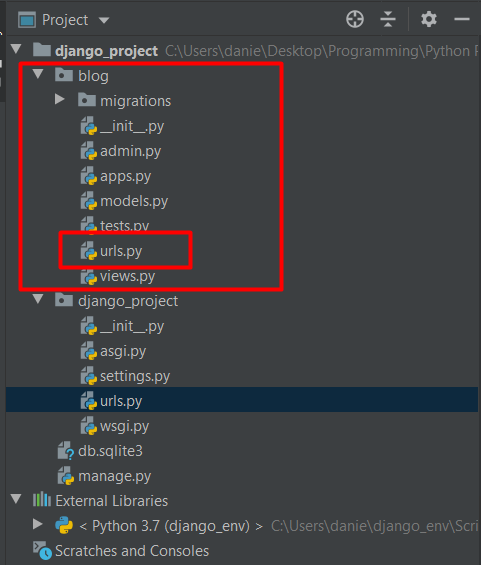
/blog/ # when it’s on blog homepage it will execute views.home function

/blog/about # when it’s on the about page it will execute views.about function

To make it work we have to add this path to the main project URL Module:



from django.contrib import admin  
from django.urls import path, include  
# The include() function basically references a python module in our case called blogs.url or in other words it MAPS it to our blogs.urls  
urlpatterns = [  
 path('admin/', admin.site.urls),  
 path('blog/', include('blog.urls')),  
]



# Templates

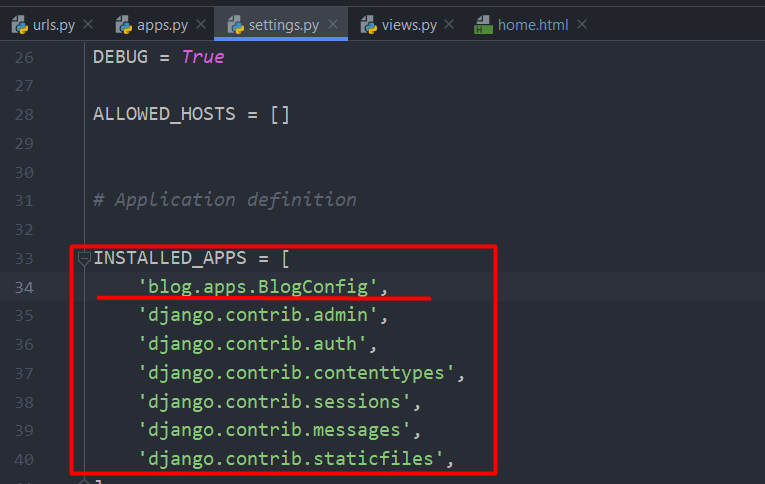
## Adding our app config class to the main projects settings module

First we need to add our apps config class to the main project settings.py module.

myapp/apps.py , down below is our class config file

*class* BlogConfig(AppConfig):  
 name = 'blog'

Now we add this to our main project settings.py module



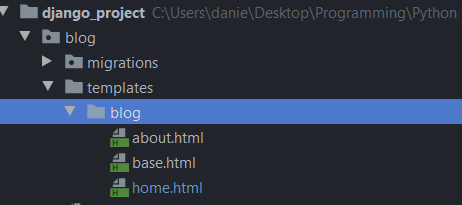
## Making templates

In order to make templates work , we have to add a templates folder inside the app we want to use the templates in our case ‘Blog’ and it should look something like this

eg. desiredapp/templates/desiredapp/

# Note : We have to make another folder in there with our apps name.

Here we are going to add the templates we want to use



Next we go to our views.py folder and make sure that we use render() on these templates

## Django render() function:

*from* django.shortcuts *import* render

*from* django.http *import* HttpResponse

*def* about(request):  
 *return* render(request, 'blog/about.html')

# Render includes HttpResponse in it’s function

# Basically here when we the user navigates to our blog page this function will handle the request and load about.html file

## Adding dummy data and using it in our templates:

Below we add some dummy data and use it in our blog/templates/blog/home.html file

*from* django.shortcuts *import* render  
*from* django.http *import* HttpResponse  
  
posts=[  
 {  
 'author': 'Ghirasim Daniel',  
 'title': 'Blog Post 1',  
 'content': 'First post content',  
 'date\_posted': '31 December, 2019'  
 },  
 {  
 'author': 'Jane Doe',  
 'title': 'Blog Post 2',  
 'content': 'Second post content',  
 'date\_posted': '01 January, 2020'  
 },  
  
]  
  
*def* home(request):  
 context = {  
 'posts': posts  
 }  
 *return* render(request, 'blog/home.html', context)

Breaking it down in steps:

1. We make a list of dictionaries containing some dummy data
2. We go into our home function and make a new dictionary called context where we make a key called ‘posts’ and the value we assign to it is our posts list of dictionaries
3. Then we add the context dict as an argument to our return statement

## Accessing data in our templates .html files:

This is how our home.html file looks:

<!DOCTYPE html>  
<html>  
<head>  
 <title>Title</title>  
</head>  
<body>  
 {% for post in posts%}  
 <h1>{{post.title}}</h1>  
 <p>By {{post.author}} on {{post.date\_posted}}</p>  
 <p>{{post.content}}</p>  
 {% endfor %}  
</body>  
</html>

### FOR LOOP in Django:

A Django for loop looks like this:

{% for post in posts%}

{% endfor %}

### VARIABLES in Django:

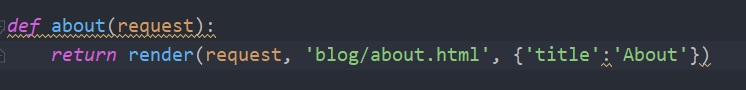
<h1>{{post.title}}</h1>

The double curly braces means that we want to access a variable

### IF ELSE statements in Django:

<head>  
 {% if title %}  
 <title>Django blog - {{ title }}</title>  
 {% else %}  
 <title>Django Blog</title>  
 {% endif %}  
</head>

We define title in our views.py:



With the IF ELSE statement we say:

If we have a title print ‘Django Blog – {title}’ else print ‘Django Blog’

And it will look like this if we have a title:



and if we don’t have one:



## Template Inheritance - Base Template – Making our life easier

Making a base template helps us reduce the amount of code we have to write . We can make a base.html file from which our other .html **will inherit from** so if we modify the base.html file the other ones will automatically get modified.

We add to this base.html file all the things that are going to be constant troughout all of our .html files for eg:

<head>  
 {% if title %}  
 <title>Django blog - {{ title }}</title>  
 {% else %}  
 <title>Django Blog</title>  
 {% endif %}  
</head>

We want this if statement for all of our website.

### How does a base template work

This is how our base.html looks after some editing:

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"  
 "http://www.w3.org/TR/html4/loose.dtd">  
<html>  
<head>  
 {% if title %}  
 <title>Django blog - {{ title }}</title>  
 {% else %}  
 <title>Django Blog</title>  
 {% endif %}  
</head>  
<body>  
 {% block content %}{% endblock %}  
</body>  
</html>

Between the block content code we are going to set each or .html files information.

for eg. home.html will look like this:

{% extends 'blog/base.html' %}   
{% block content %}  
 {% for post in posts%}  
 <h1>{{post.title}}</h1>  
 <p>By {{post.author}} on {{post.date\_posted}}</p>  
 <p>{{post.content}}</p>  
 {% endfor %}  
{% endblock content %}

We deleted everything that is found in base.html and are that we are left with is the actual code we want to run on the page.

{% extends 'blog/base.html' %}

This tells our home.html file that it **extends** base.html and after that we add:

{% block content %}  
 # We add our home.html code here  
{% endblock content

{% block content %} this refers to the CONTENT BLOCK we can have several blocks in a template file.

like {% block posts %} {% block pictures %} (have to verify this).

This is how our about page looks.

{% extends 'blog/base.html' %}  
{% block content %}  
 <h1>About Page!</h1>  
{% endblock content %}

Notice we reduced the code drastically by using template inheritance.

## Bootstrap

Bootstrap is a extremely popular library which makes it easy to add some nice styles to your website.

Basically we drag and drop the Bootstrap starter template into our base.html file . (It does some magic and it works).

#### Starter template:

<!doctype html>

<html lang="en">

<head>

<!-- Required meta tags -->

<meta charset="utf-8">

<meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no">

<!-- Bootstrap CSS -->

<link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/bootstrap/4.3.1/css/bootstrap.min.css" integrity="sha384-ggOyR0iXCbMQv3Xipma34MD+dH/1fQ784/j6cY/iJTQUOhcWr7x9JvoRxT2MZw1T" crossorigin="anonymous">

<title>Hello, world!</title>

</head>

<body>

<h1>Hello, world!</h1>

<!-- Optional JavaScript -->

<!-- jQuery first, then Popper.js, then Bootstrap JS -->

<script src="https://code.jquery.com/jquery-3.3.1.slim.min.js" integrity="sha384-q8i/X+965DzO0rT7abK41JStQIAqVgRVzpbzo5smXKp4YfRvH+8abtTE1Pi6jizo" crossorigin="anonymous"></script>

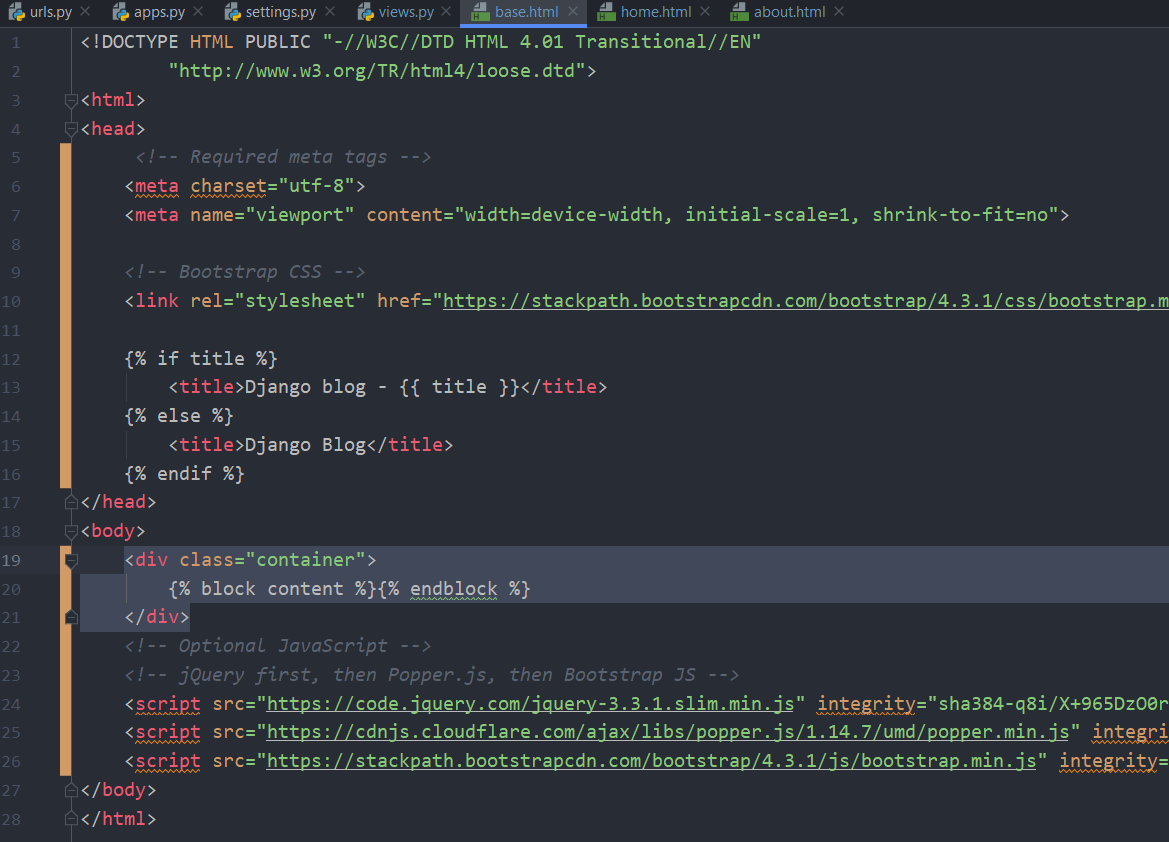
<script src="https://cdnjs.cloudflare.com/ajax/libs/popper.js/1.14.7/umd/popper.min.js" integrity="sha384-UO2eT0CpHqdSJQ6hJty5KVphtPhzWj9WO1clHTMGa3JDZwrnQq4sF86dIHNDz0W1" crossorigin="anonymous"></script>

<script src="https://stackpath.bootstrapcdn.com/bootstrap/4.3.1/js/bootstrap.min.js" integrity="sha384-JjSmVgyd0p3pXB1rRibZUAYoIIy6OrQ6VrjIEaFf/nJGzIxFDsf4x0xIM+B07jRM" crossorigin="anonymous"></script>

</body>

</html>

#### Adding bootstrap to our base.html



Notice we added our block content between <div> tags . I don’t really know the magic behind this but after the first div tag we add a class called ‘class=”container”’

After adding this I noticed the text moves slightly to the right.