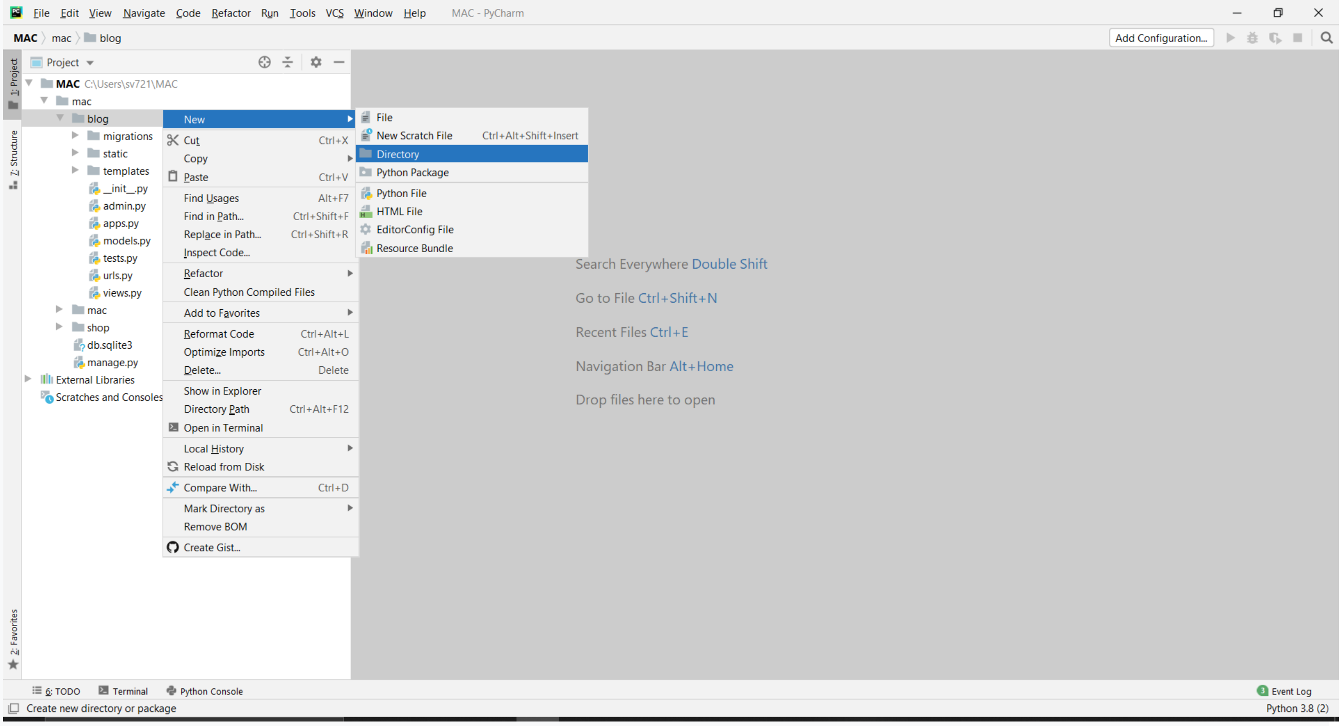
E-commerce Website: Adding Bootstrap, Templates & Static | Python Django Tutorials In Hindi #22

In the previous tutorial, we created two apps for our E-commerce website. In this tutorial, we will add Bootstrap to those apps, and we will also discuss the working of static files in Django. As discussed in one of our previous tutorials, we need templates to include Bootstrap on our website. So, without wasting time, let's create templates for our apps. Follow the steps given below:

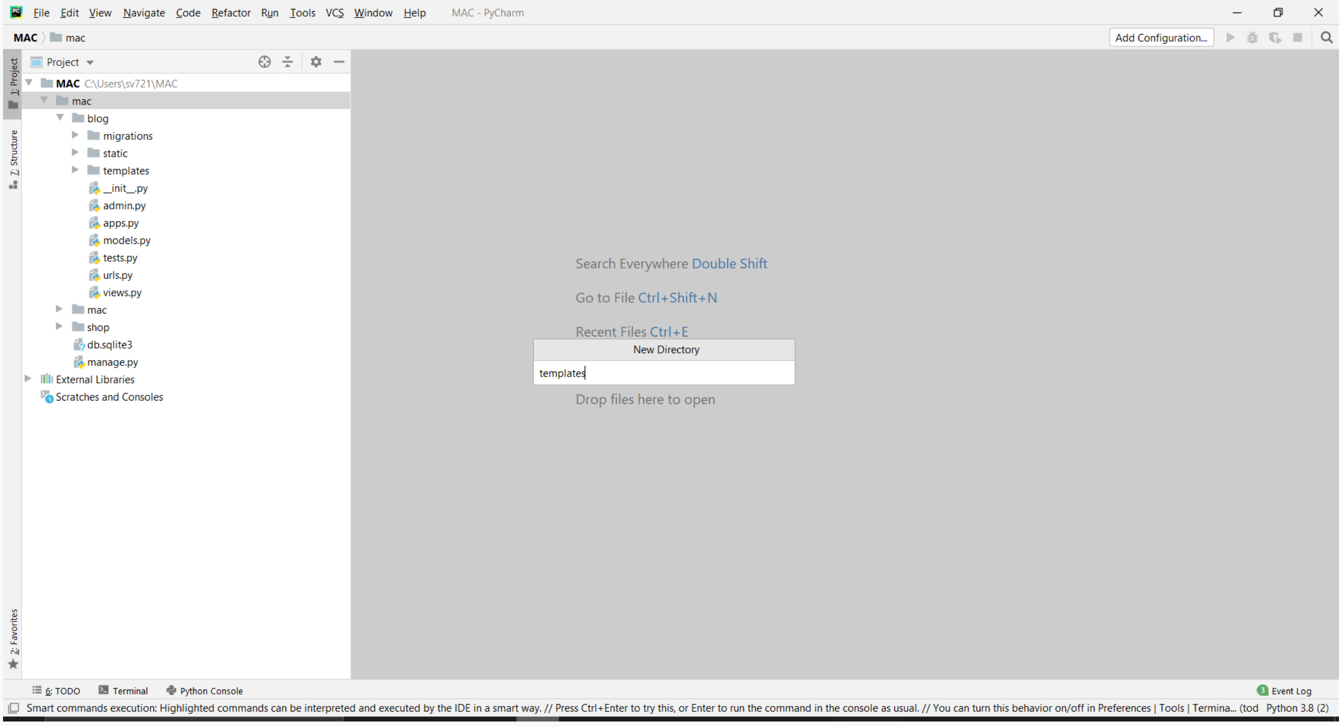
Step 1: Double click on the blog folder.

Step 2: Click on New and select Directory, as shown in the image below.



Create new directory or package

Step 3: After doing this, name the directory as templates and press enter.

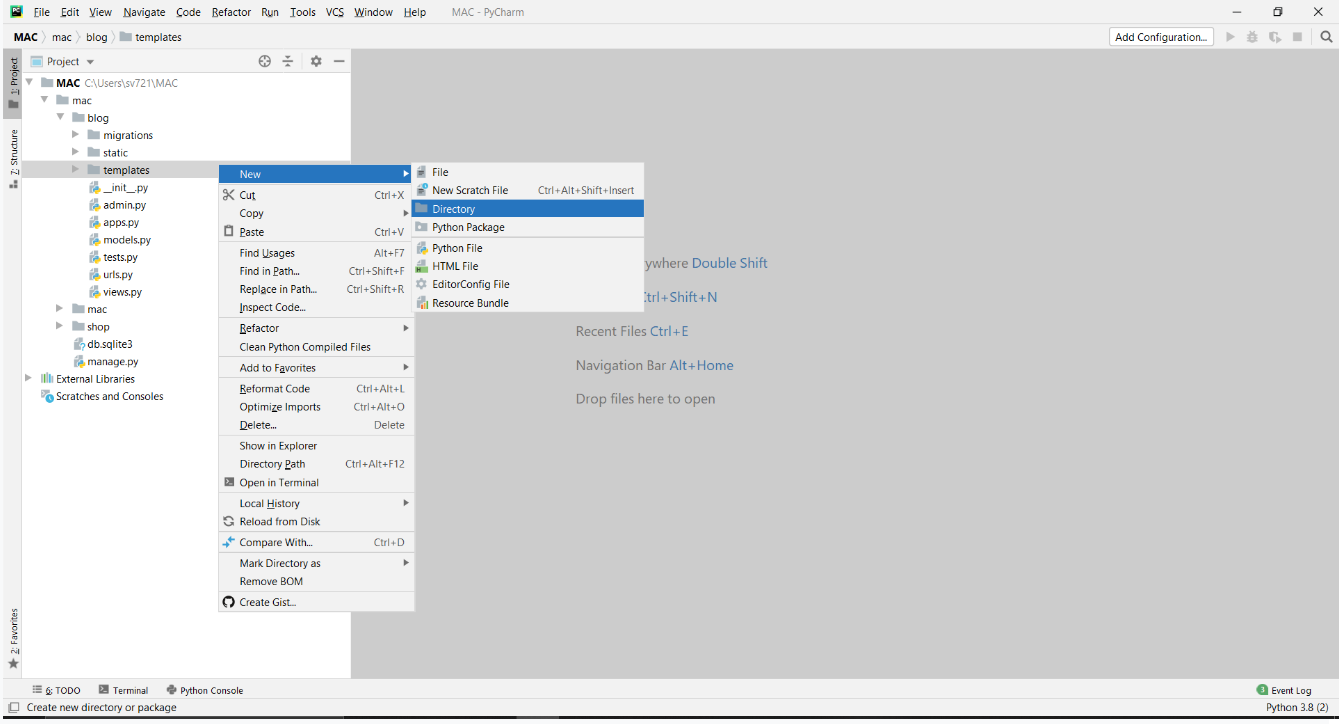


New Directory

templates

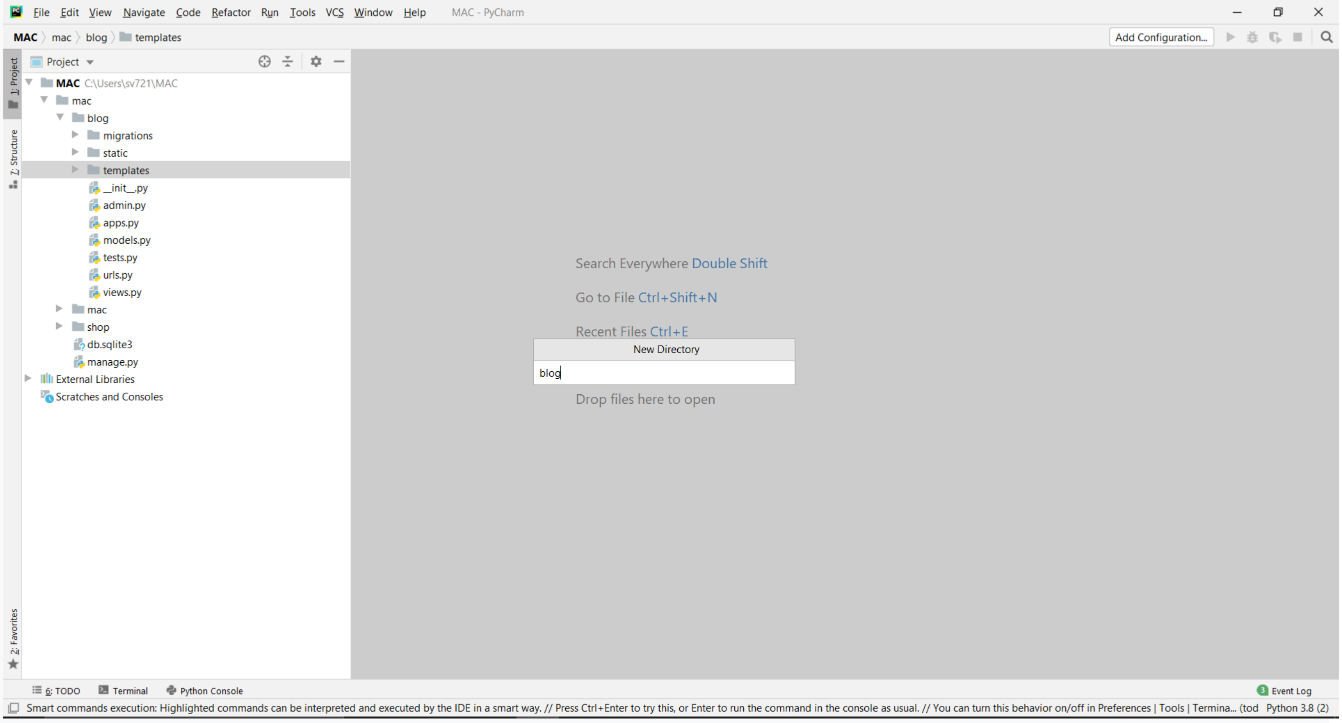
Smart commands execution: Highlighted commands can be interpreted and executed by the IDE in a smart way. // Press Ctrl+Enter to try this, or Enter to run the command in the console as usual. // You can turn this behavior on/off in Preferences | Tools | Termina... (tod

Step 4: Now, we will create a directory called "blog" inside the blog app's template directory. Double click on the template folder -->click on New-->Select Directory.



Create new directory or package

Step 5: Name the directory as blog, as shown in the image given below.

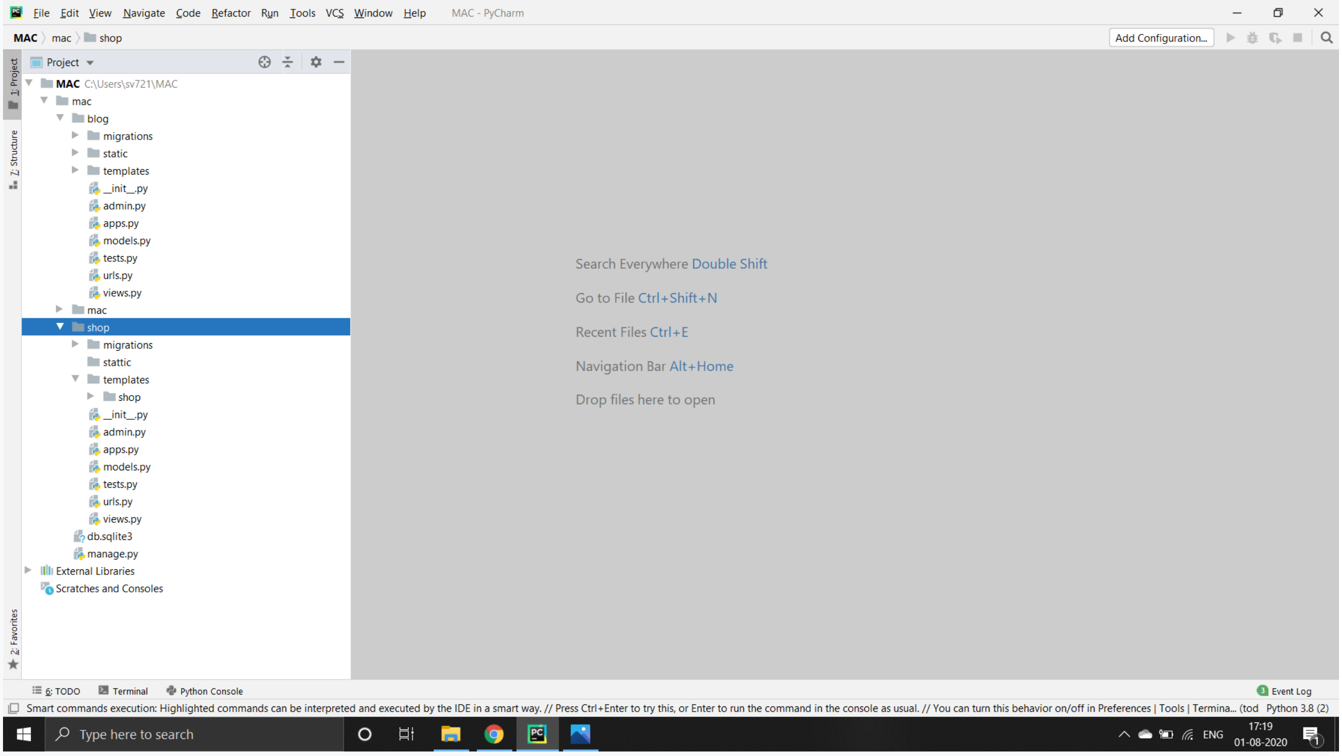


New Directory

blog

Smart commands execution: Highlighted commands can be interpreted and executed by the IDE in a smart way. // Press Ctrl+Enter to try this, or Enter to run the command in the console as usual. // You can turn this behavior on/off in Preferences | Tools | Termina... (tod

You can see in the image given below that we have followed the same procedure for the shop app.



Smart commands execution: Highlighted commands can be interpreted and executed by the IDE in a smart way. // Press Ctrl+Enter to try this, or Enter to run the command in the console as usual. // You can turn this behavior on/off in Preferences | Tools | Termina... (tod Python 3.8 (2)

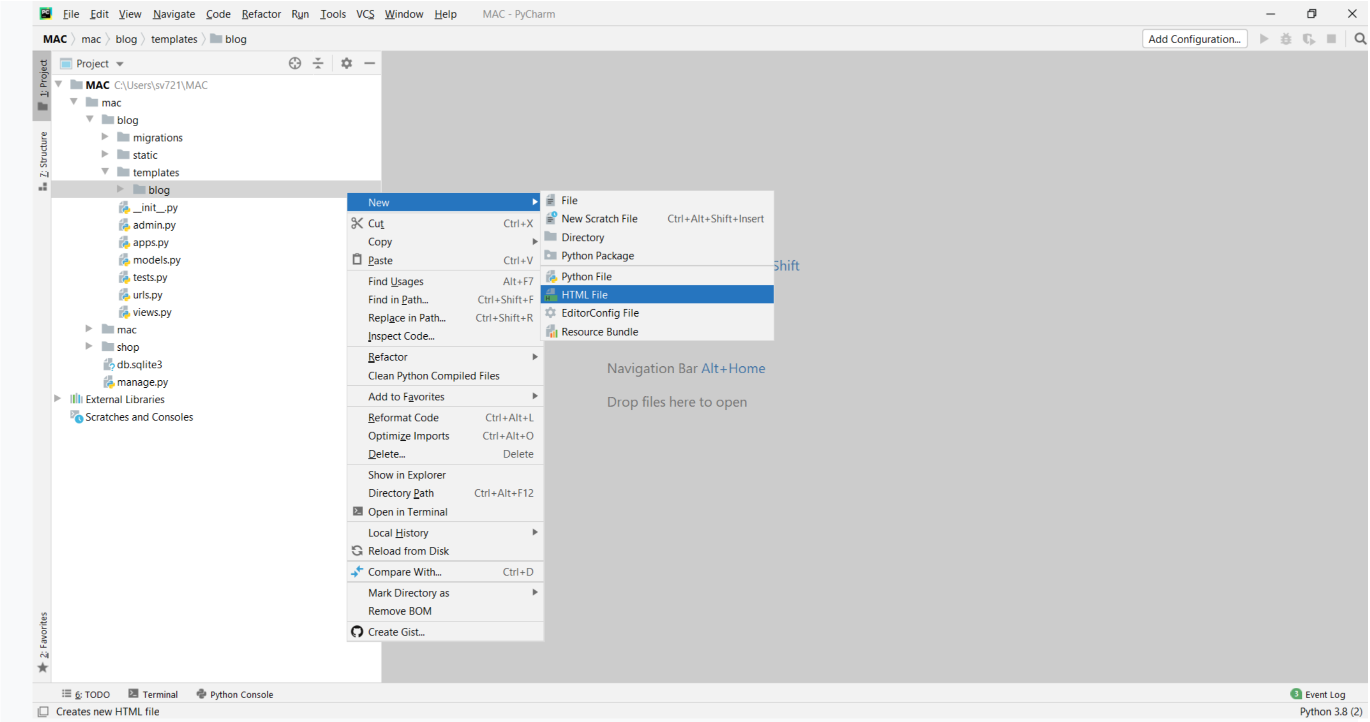
Django requires us to include the installed apps in the settings of the main project. So, open the settings.py file of the mac and write the names of the apps under the INSTALLED_APPS section:

```
# Application definition
INSTALLED_APPS = [
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',
     'blog',
     'shop'
]
```

We need an HTML file to add Bootstrap, so let's create "index.html." Follow the below steps:

Step 1: Blog-->templates-->blog.

Step 2: Double click on the blog and select the HTML file.



Step 3: Name the HTML file as "index.html."

Step 4: Head over to the starter page of Bootstrap and copy the stater template.

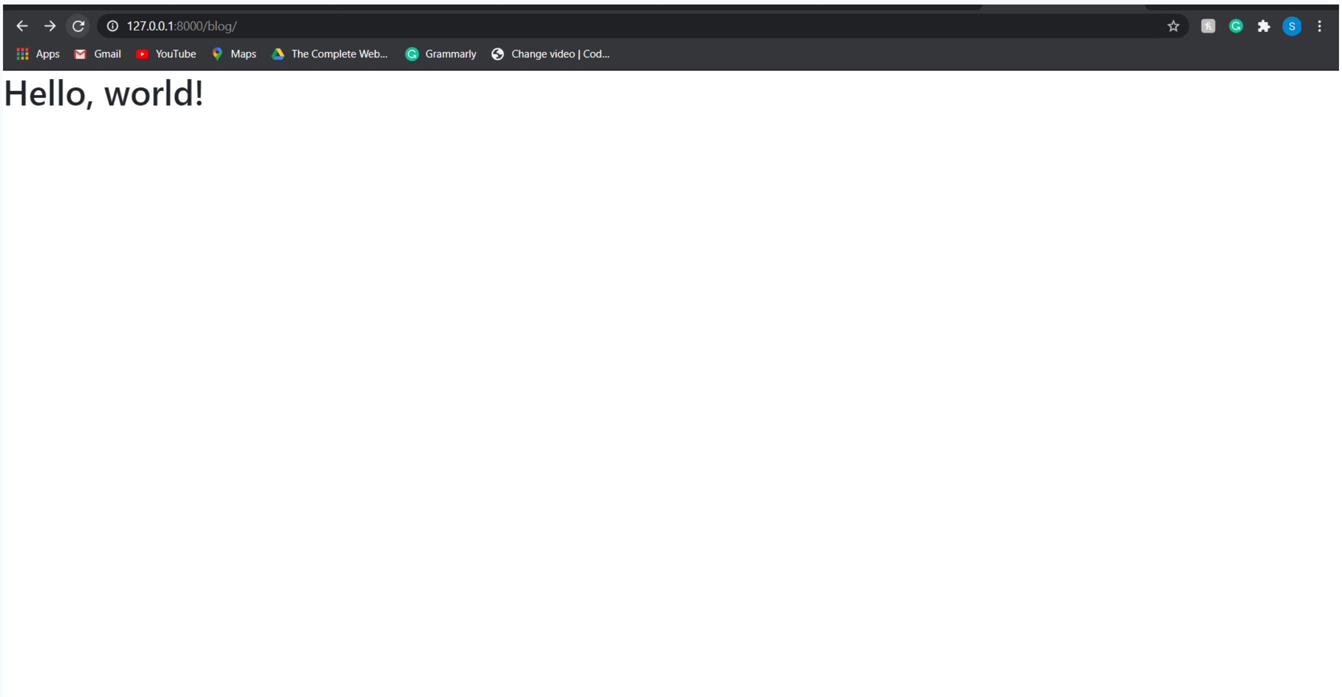
Step 5: Paste this starter template in the "index.html" file.

Step 6: Now, open the views.py file and type the following code :

```
from django.shortcuts import render
from django.http import HttpResponse
def index(request):
    return render(request,"index.html")
```

With this, we have successfully added Bootstrap to our blog app. Now, let's check that Bootstrap is added correctly or not. So, restart the development server and go to http://127.0.0.1:8000/blog/.

Output :



Hello, world!

In the above image, you can see that the "Hello, world!" page is opened, which means Bootstrap is added correctly to the blog app. Follow the same procedure and add Bootstrap to the shop app as well. Now, let's start our discussion on static files in Django.

Static files:

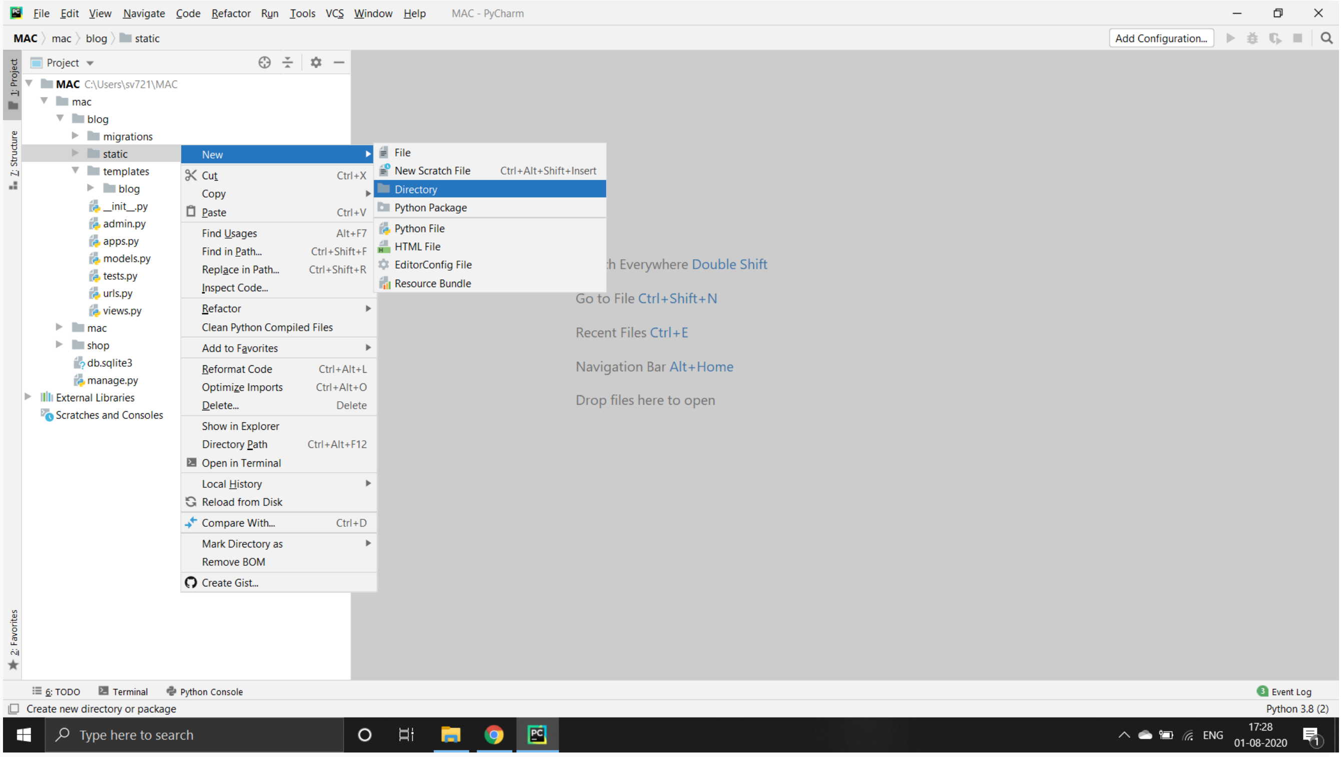
- Our E-commerce website will not only contain simple or plain text; it will also contain additional images, CSS, or javascript. These additional files are known as static files in Django.

Steps to create static files:

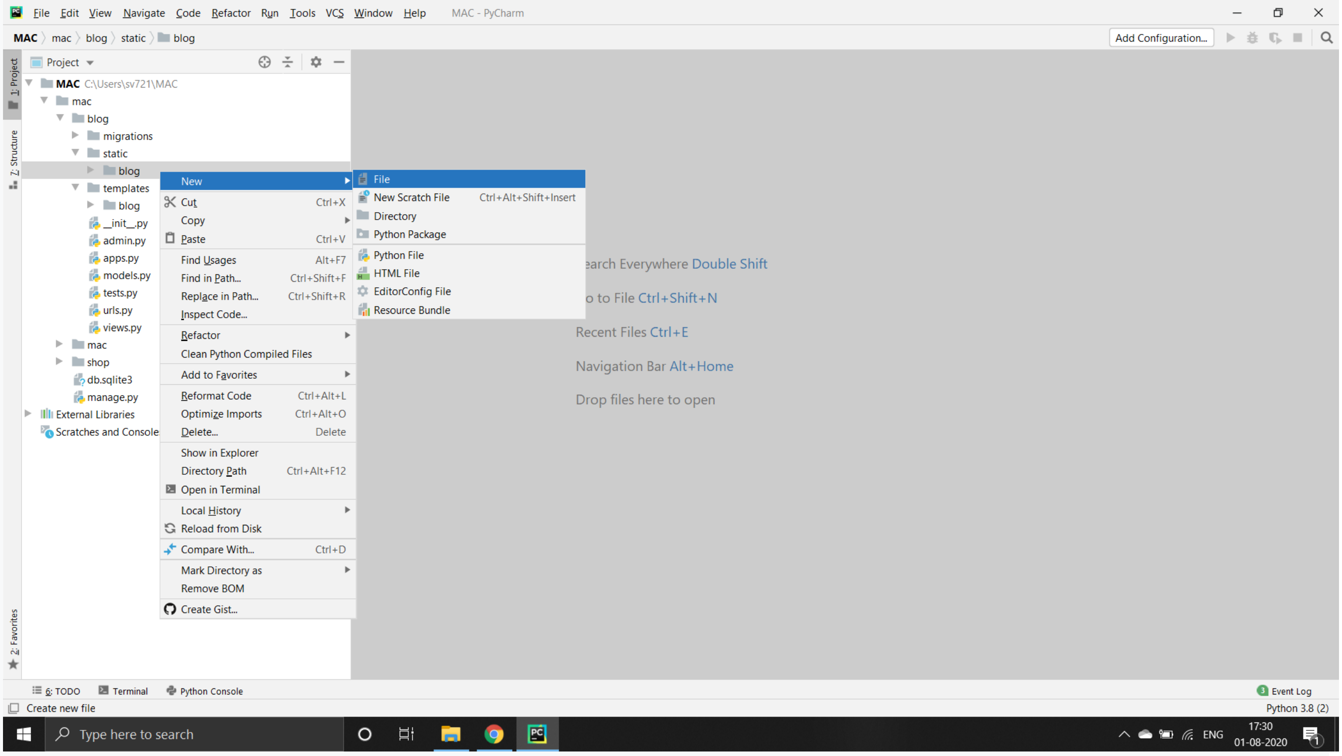
Step 1: Double click on the blog folder and create a new Directory.

Step 2: Name the directory as static.

Step 3: Click on the static folder and create a new directory called blog. All the static files will be contained under this blog folder.



Step 4: Create a new file called mystatic.txt inside the blog folder that we created in the last step. Type "This is my static file in blog" in the mystatic.txt file.

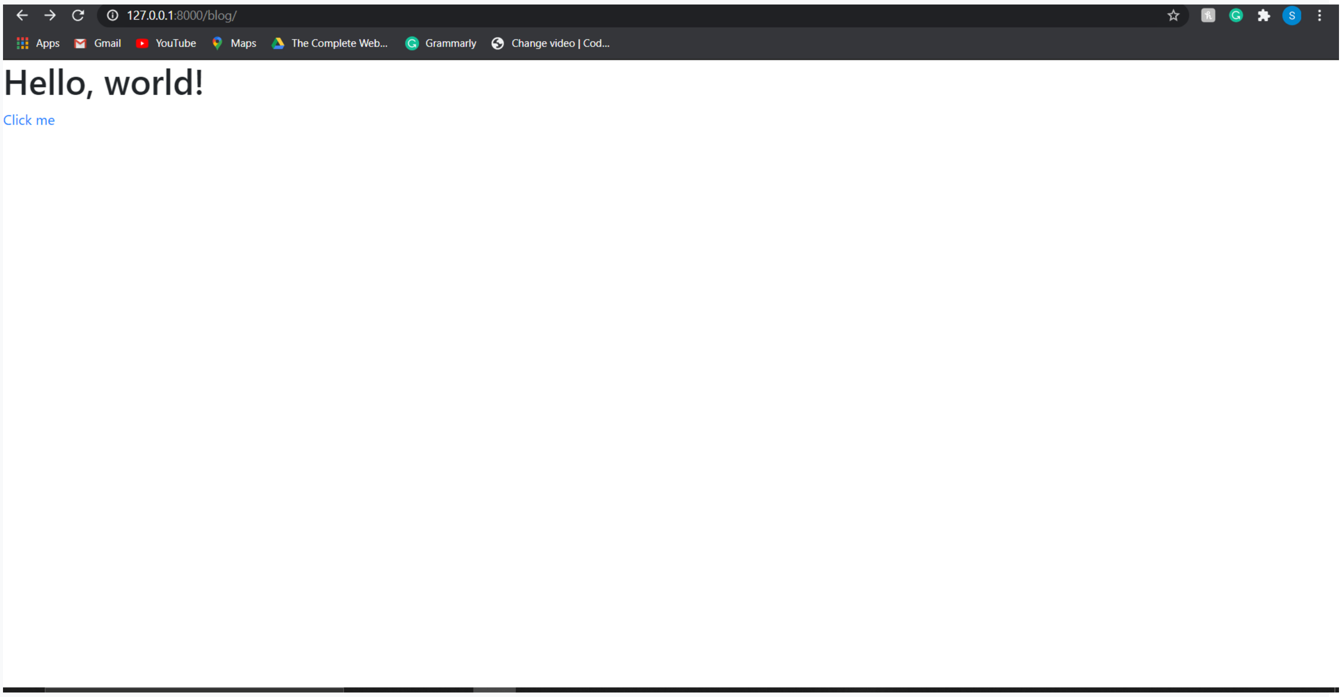


Accessing static file on the blog:

To access the static file, first of all, we need to load the static file in index.html. So, open the blog/index.html file and type the following code in <body>:

```
 {% load static %}
<a href="{%static 'blog/mystatic.txt' %}" >Click me </a>
```

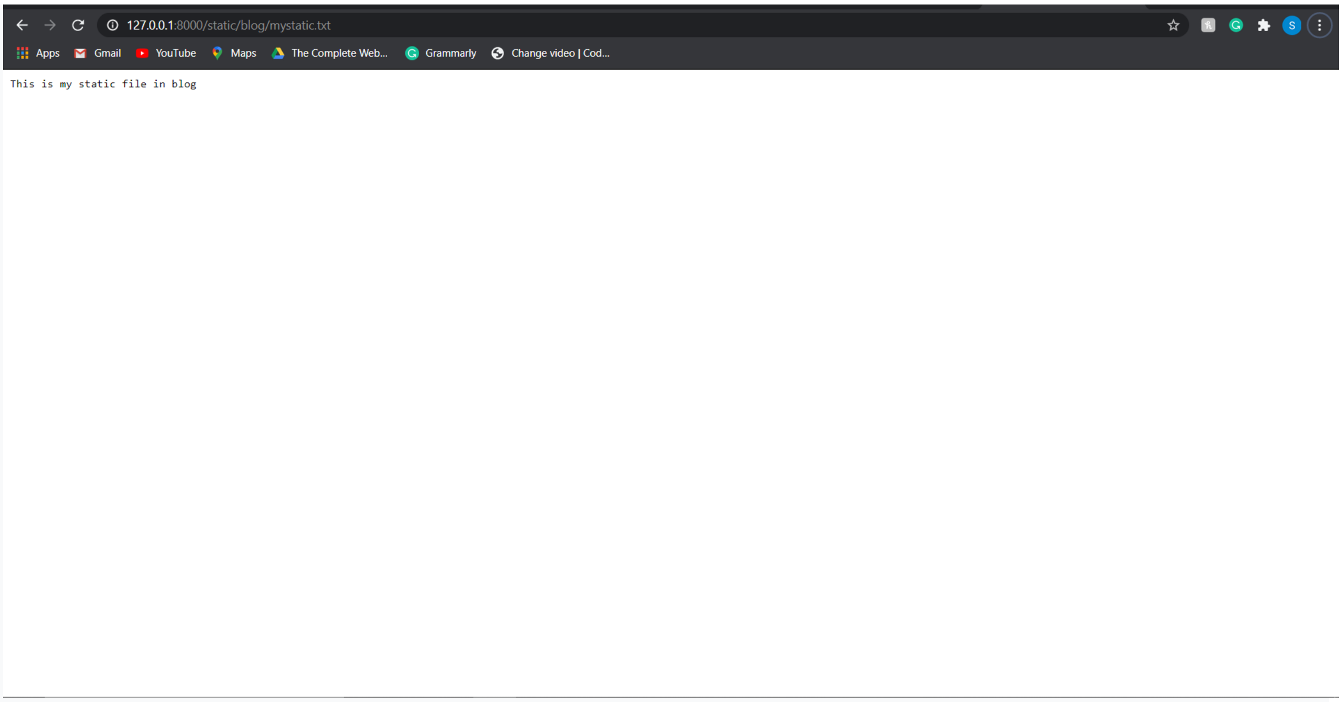
The above code is the syntax to load static files in Django templates. Now, restart the server and go to the blog page. If you have followed all the steps correctly, then you will see a screen like this :



Hello, world!

Click me

Screen after clicking on the "Click me" button:



```
This is my static file in blog
```

So, that's how you can easily include static files on a Django website. This tutorial ends here, and I will see you in the next tutorial.

Template code as described/written in the video

```
<!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/4.2.1/css/bootstrap.min.css" integ

    <title>Hello, world!</title>
  </head>
  <body>
    <h1>Hello, world! shop</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+965DzO0rT7abK41JStQIA
    <script src="https://cdnjs.cloudflare.com/ajax/libs/popper.js/1.14.6/umd/popper.min.js" integrity="sha384-wH
    <script src="https://stackpath.bootstrapcdn.com/bootstrap/4.2.1/js/bootstrap.min.js" integrity="sha384-B0Ugl
  </body>
</html>
```