

How to use AdminLTE in Django

I have been doing Django Framework based design and development work for 5+ years. While the Django development is Python programming at its core, inevitably we will need the CSS/HTML/JavaScript for the presentation. When comes to the selection of CSS/JavaScript libraries, jQuery/Bootstrap combination seems to be the most popular choice. There are number of Django packages that support the jQuery and Bootstrap already. However, I have decided to go one step further that is to use a ready-made theming library. It took me awhile to find a suitable CSS/JavaScript theming library to work with our Django based project. What I had selected is AdminLTE (<https://adminlte.io/>). In my opinion, AdminLTE is one of the best Open Source Bootstrap 4 theming library. In this article, I will share how to use AdminLTE and Django Framework to build a website.

Dependencies:

1. Python 3.7
2. Django 2.2
3. AdminLTE 3.0.5 (Bootstrap 4)

In this tutorial, I am going to use [Visual Studio Code](#) as my main IDE.

Step 1. Create Django project

```
django-admin startproject mysite
```

Go to **mysite** folder and start project by running this command.

```
python manage.py runserver
```

By default, Django project runs on <http://127.0.0.1:8000/> . Opening browser with this url, we would expect this.



The install worked successfully! Congratulations!

You are seeing this page because `DEBUG=True` is in your settings file and you have not configured any URLs.



Django Documentation
Topics, references, & how-to's



Tutorial: A Polling App
Get started with Django



Django Community
Connect, get help, or contribute

Step 2. Create app

Run this command to create a new app called **polls**.

```
python manage.py startapp polls
```

In **mysite/settings.py**, add **polls** in **INSTALLED_APPS**.

```
INSTALLED_APPS = [  
    'django.contrib.admin',  
    'django.contrib.auth',  
    'django.contrib.contenttypes',  
    'django.contrib.sessions',  
    'django.contrib.messages',  
    'django.contrib.staticfiles',  
  
    # apps  
    'polls.apps.PollsConfig',  
]
```

In **polls/views.py**, add a simple view function.

```
from django.http import HttpResponse

def index(request):
    return HttpResponse("Hello, world. You're at the polls index.")
```

In **polls/urls.py**, add url pattern for index page.

```
from django.urls import path

from . import views

urlpatterns = [
    path('', views.index, name='index'),
]
```

In **mysite/urls.py**, add url pattern to map **polls/urls.py**.

```
from django.contrib import admin
from django.urls import include, path

urlpatterns = [
    path('polls/', include('polls.urls')),
    path('admin/', admin.site.urls),
]
```

Run server and check <http://127.0.0.1:8000/polls/> in browser. We will see “Hello, world. You're at the polls index.” printed on screen.

Boring content, is it? Let's add a fancy scheme!

Step 3. Download AdminLTE

As I am writing this article, AdminLTE is up to version 3.0.5. Download and unzip the source code. There are a lot of packages and template. We don't necessarily need everything inside. In this tutorial, we will use template **starter.html**.

<https://github.com/ColorlibHQ/AdminLTE/releases/tag/v3.0.5>

- .github
- dist
- pages
- .babelrc
- .editorconfig
- .eslintrc
- .npmignore
- index.html
- index3.html
- package.json
- README.md

- build
- docs
- plugins
- .browserslistrc
- .eslintignore
- .gitignore
- composer.json
- index2.html
- LICENSE
- package-lock.json
- starter.html

Step 4. Set up template folder

Create a folder called **templates** in **polls** folder. Within the **templates** folder we have just created, create another folder called **polls**, and within that create a file called **index.html**. Then copy and paste the content of **starter.html** from AdminLTE folder into **index.html** we have just created.


Modify **polls/views.py** so that it ensures that it responds rich content with html template.

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

def index(request):
    return render(request, 'polls/index.html')
```

Let's run server again.

-
- [Home](#)
- [Contact](#)

- [3](#)
-  [User Avatar](#)
- **Brad Diesel**

Hmm... It's not as rich as we thought. The reason is that we didn't set up static folder yet.

Step 5. Setup static folder

Create a folder called **static** in project folder. The **static** folder and **polls** folder should be in the same level. We are going to add all static file we need into this folder. Because they are global files which are very likely to be shared by multiple apps.

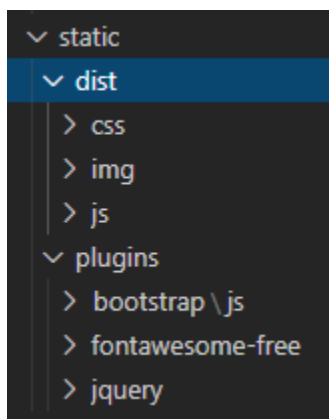
In addition, add **STATICFILES_DIRS** in **settings.py**.

```
STATICFILES_DIRS = [  
    os.path.join(BASE_DIR, "static"),  
]
```

Looking into **index.html**, we need to work on the followings.

```
<!-- Font Awesome Icons -->  
<link rel="stylesheet" href="plugins/fontawesome-free/css/all.min.css">  
<!-- Theme style -->  
<link rel="stylesheet" href="dist/css/adminlte.min.css">  
  
  
  
<!-- jQuery -->  
<script src="plugins/jquery/jquery.min.js"></script>  
<!-- Bootstrap 4 -->  
<script src="plugins/bootstrap/js/bootstrap.bundle.min.js"></script>  
<!-- AdminLTE App -->  
<script src="dist/js/adminlte.min.js"></script>
```

We only pick the packages that are required by this template. Move them over into **static** folder. This is what it looks like eventually.



Step 6. Modify template

We must modify **index.html** so that Django template engine is able to recognize the path of static file.

Add `{% static %}` template tag at the beginning of the index.html.

```
{% load static %}
```

Sample of change on stylesheet path

```
<!-- Font Awesome Icons -->
<link rel="stylesheet" href="{% static 'plugins/fontawesome-free/css/all.min.css' %}">
<!-- Theme style -->
<link rel="stylesheet" href="{% static 'dist/css/adminlte.min.css' %}">
```

Sample of change on image path

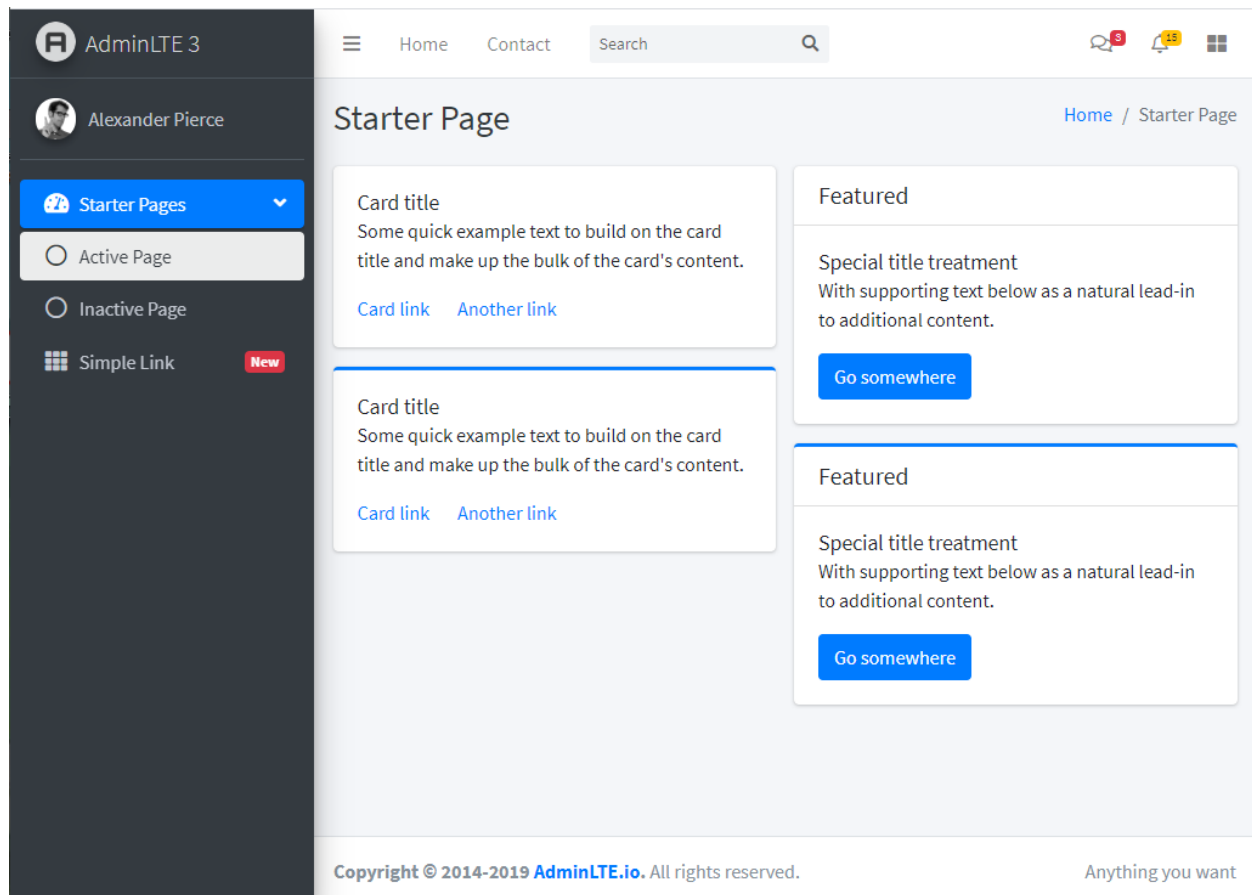
```

```

Sample of change on Javascript path

```
<!-- jQuery -->
<script src="{% static 'plugins/jquery/jquery.min.js' %}"></script>
<!-- Bootstrap 4 -->
<script src="{% static 'plugins/bootstrap/js/bootstrap.bundle.min.js' %}"></script>
<!-- AdminLTE App -->
<script src="{% static 'dist/js/adminlte.min.js' %}"></script>
```

OK. We have everything setup. Let's restart server and check out! We should expect this.



For anyone who is interested in looking into the real project, I have created a tutorial project on my GitHub account. The repository name is: [DjangoAndAdminLTE](https://github.com/slow999/DjangoAndAdminLTE). The repository URL is: <https://github.com/slow999/DjangoAndAdminLTE>.

If you would like to add Google ReCaptcha support in your website, you may check my previous article [here](#). Stay tuned.