**Django Admin**

Django provides an admin site to allow CRUD (Create Read Update Delete) operations on registered app model.

It is a built-in feature of Django that automatically generates interface for models.

We can see the url entry for admin in urls.py file, it is implicit and generated while creating a new project.

urlpatterns = [

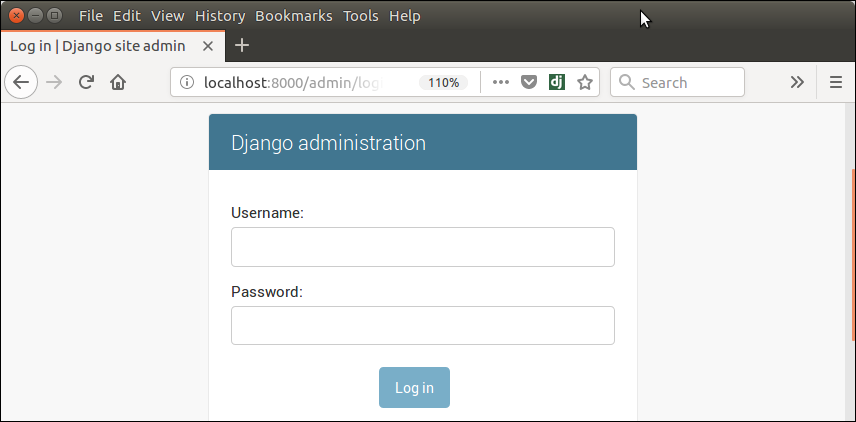
    path('admin/', admin.site.urls),

]

It can be easily accessed by after login from the admin panel

lets run the server **python manage.py runserver** and access it through the **localhost:8000/admin.**

A login form will be displayed, see the below.



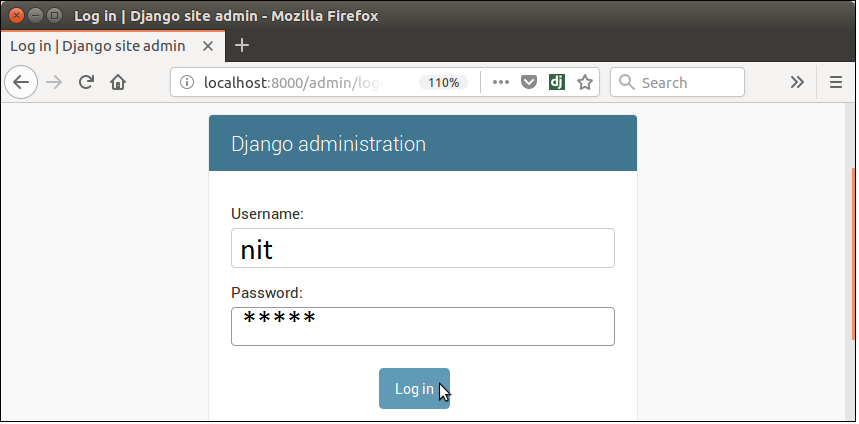
To login, first create admin (super user) user and provide password

Python manage.py createsuperuser

Username :

EmailAddress:

Password:



**// models.py**

from django.db **import** models

**class** Employee(models.Model):

    eid     = models.CharField(max\_length=20)

    ename   = models.CharField(max\_length=100)

    econtact = models.CharField(max\_length=15)

**class** Meta:

        db\_table = "employee"

**Register Django Model**

To register model in **admin.py** file. Use the **admin.site.register()** method and pass the Model name

**// admin.py**

from django.contrib **import** admin

from myapp.models **import** Employee

admin.site.register(Employee) # Employee is registered

**Django authentication and authorization**

**What is the purpose of authentication?**

Authentication is the very first step of a security system; it validates the identity of the user by verifying their credentials. Authorization must follow authentication in a system security environment. It grants or denies the access to different resources, actions or functions.

**The Django auth app**

Django automatically installs the auth app when a new project is created. Look in the django\_project/settings.py file under INSTALLED\_APPS and you can see auth is one of several built-in apps Django has installed for us.

INSTALLED\_APPS **=** [

"django.contrib.admin",

"django.contrib.auth",

"django.contrib.contenttypes",

"django.contrib.sessions",

"django.contrib.messages",

"django.contrib.staticfiles",

]

To use the auth app we need to add it to our project-level urls.py file. Make sure to add include on the second line. I've chosen to include the auth app at accounts/ but you can use any url pattern you want.

*# django\_project/urls.py*

from django.contrib import admin

from django.urls import path, include

urlpatterns **=** [

path("admin/", admin**.**site**.**urls),

path("accounts/", include("django.contrib.auth.urls")),

]

The auth app we've now included provides us with several [authentication views](https://docs.djangoproject.com/en/4.0/topics/auth/default/#module-django.contrib.auth.views) and URLs for handling login, logout, and password management.

The URLs provided by auth are:

accounts/login/ [name='login']

accounts/logout/ [name='logout']

accounts/password\_change/ [name='password\_change']

accounts/password\_change/done/ [name='password\_change\_done']

accounts/password\_reset/ [name='password\_reset']

accounts/password\_reset/done/ [name='password\_reset\_done']

accounts/reset/<uidb64>/<token>/ [name='password\_reset\_confirm']

accounts/reset/done/ [name='password\_reset\_complete']

There are associated auth views for each URL pattern, too. That means we only need to create a *template* to use each!

**Login Page**

Let's make our login page! Django by default will look within a templates folder called registration for auth templates. The login template is called login.html.

Create a new directory called templates and within it another directory called registration.

Then create a templates/registration/login.html file with your text editor and include the following code:

*<!-- templates/registration/login.html -->*

<h2>Log In</h2>

<form method**=**"post">

{% csrf\_token %}

{{ form.as\_p }}

<button type**=**"submit">Log In</button>

</form>

Next update the settings.py file to tell Django to look for a templates folder at the project level. Update the DIRS setting within TEMPLATES as follows. This is a one-line change.

*# django\_project/settings.py*

TEMPLATES **=** [

{

**...**

'DIRS': [BASE\_DIR **/** "templates"],

**...**

},

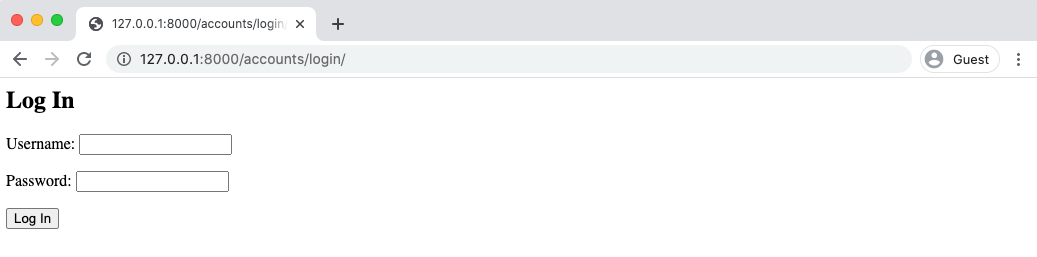
]

Our login functionality now works but to make it better we should specify *where* to redirect the user upon a successful login. In other words, once a user has logged in, where should they be sent on the site? We use the LOGIN\_REDIRECT\_URL setting to specify this route. At the bottom of the settings.py file add the following to redirect the user to the homepage.

*# django\_project/settings.py*

LOGIN\_REDIRECT\_URL **=** "/"

We're actually done at this point! If you now start up the Django server again with python manage.py runserver and navigate to our login page at http://127.0.0.1:8000/accounts/login/ you'll see the following.



**Create users**

But there's one missing piece: **we haven't created any users yet**. Let's quickly do that by making a superuser account from the command line.  python manage.py createsuperuser. Answer the prompts and note that your password will not appear on the screen when typing for security reasons.

python manage.py createsuperuser

Username :

Email address:

*Password:*

Password (again):

Superuser created successfully.

Now spin up the server again with python manage.py runserver and refresh the page at http://127.0.0.1:8000/accounts/login/. Enter the login info for your just-created user.

**Create a homepage**

We want a simple homepage that will display one message to logged out users and another to logged in users. Create two new files with your text editor: templates/base.html and templates/home.html. Note that these are located within the templates folder but *not* within templates/registration/ where Django auth looks by default for user auth templates.

*<!-- templates/base.html -->*

**<!DOCTYPE html>**

<html>

<head>

<meta charset**=**"utf-8">

<title>{% block title %}Django Auth Tutorial{% endblock %}</title>

</head>

<body>

<main>

{% block content %}

{% endblock %}

</main>

</body>

</html>

*<!-- templates/home.html -->*

{% extends 'base.html' %}

{% block title %}Home{% endblock %}

{% block content %}

{% if user.is\_authenticated %}

Hi {{ user.username }}!

{% else %}

<p>You are not logged in</p>

<a href**=**"{% url 'login' %}">Log In</a>

{% endif %}

{% endblock %}

While we're at it, we can update login.html too to extend our new base.html file:

*<!-- templates/registration/login.html -->*

{% extends 'base.html' %}

{% block title %}Login{% endblock %}

{% block content %}

<h2>Log In</h2>

<form method**=**"post">

{% csrf\_token %}

{{ form.as\_p }}

<button type**=**"submit">Log In</button>

</form>

{% endblock %}

Now update our urls.py file so we can display the homepage. Normally I would prefer to create a dedicated pages app for this purpose, but we don't have to and for simplicity, we'll just add it to our existing django\_project/urls.py file. Make sure to import TemplateView on the third line and then add a urlpattern for it at the path ''.

*# django\_project/urls.py*

from django.contrib import admin

from django.urls import path, include

from django.views.generic.base import TemplateView *# new*

urlpatterns **=** [

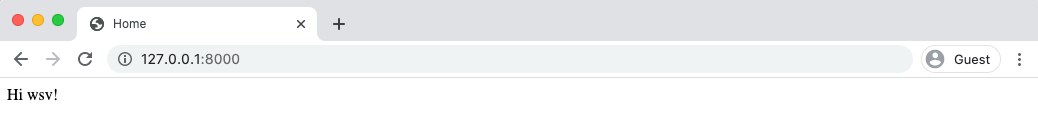
path('admin/', admin**.**site**.**urls),

path('accounts/', include('django.contrib.auth.urls')),

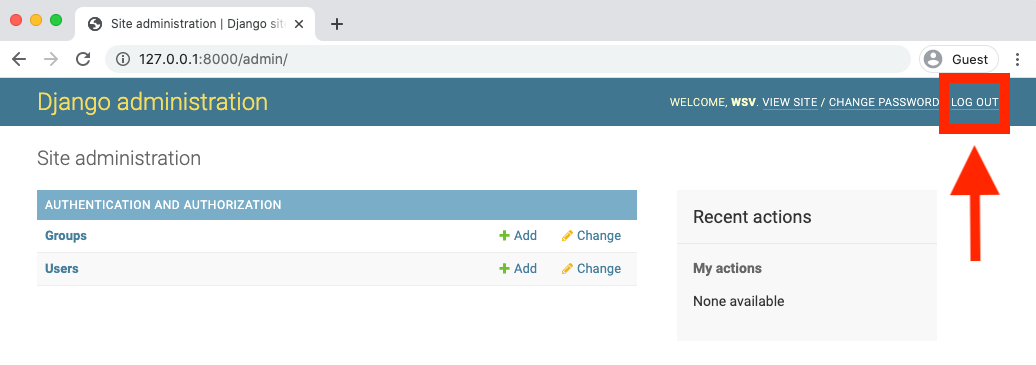
path('', TemplateView**.**as\_view(template\_name**=**'home.html'), name**=**'home'), *# new*

]

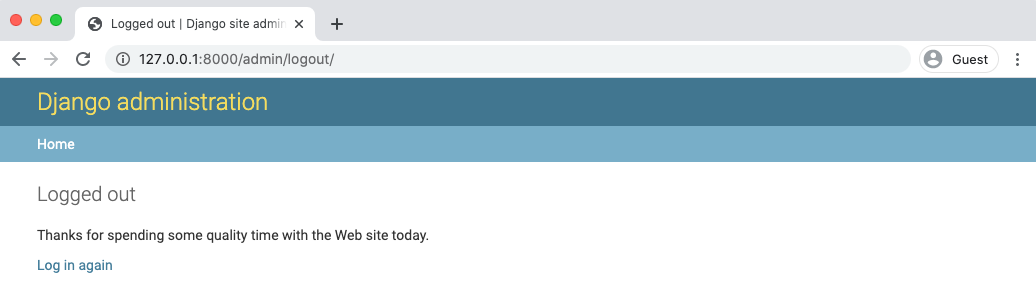
If you start the Django server again with python manage.py runserver and navigate to the homepage at http://127.0.0.1:8000/ you'll see the following:



It worked! But how do we logout? The only option currently is to go into the admin panel at http://127.0.0.1:8000/admin/ and click on the "Logout" link in the upper right corner.



This will log us out as seen by the redirect page:



If you go to the homepage again at http://127.0.0.1:8000/ and refresh the page, we can see we're logged out.

