**Lab 4 Part (3)**

**User Accounts**

In this exercise we will look at how to authenticate users in our web application. Keep working on the project that you created in lab 4.

Django comes with a powerful, built-in user authentication system that we can use. Whenever you create a new project, by default Django installs the **auth** app, which

provides us with a **User** object containing:

* username
* password
* email
* first\_name
* last\_name

We will use this **User** object to implement log in, log out, and sign up in our blog

application.

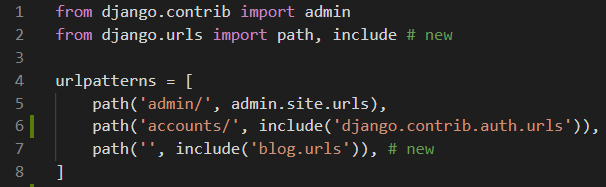
**Log in**

Django provides us with a default view for a log in page via **LoginView**. We need to make the following changes to our application to use this view:

* add a urlpattern for the auth system
* create a log in template
* make a small update to the settings.py file

We need to update the **blog\_project/urls.py** file. We will add a path to this file which defines the accounts/ URL where we will place our log in and log out pages.

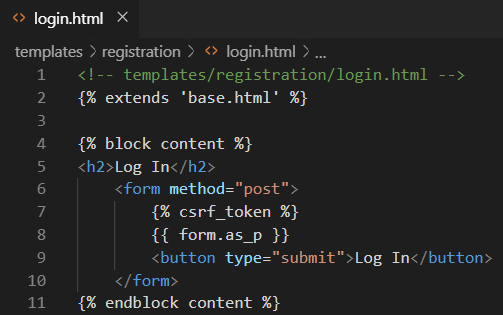
1. Add the line of code shown at line 6 below:

**Code**

By default, Django will look within a templates directory called registration for a file called **login.html** for a log in form. We need to create a new directory called registration and the requisite file within it.

1. In VS Code create a folder inside the templates folder called registration.
2. Create an empty file called **login.html** and add in the following code:

**Code**



Line 6: We use HTML **<form></form>** tags and specify the **POST** method since we are sending data to the server (we would use **GET** if we were requesting data, such as in a search engine form).

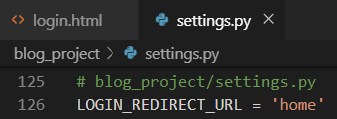
We add **{% csrf\_token %}** for security concerns, namely to prevent a XSS Attack. The form’s contents are outputted between paragraph tags thanks to **{{ form.as\_p }}** and then we add a “submit” button.

**Re Direct User**

The final step is to specify where to redirect the user upon a successful log in. We can set this with the **LOGIN\_REDIRECT\_URL** setting.

1. At the bottom of the settings.py file add the following:

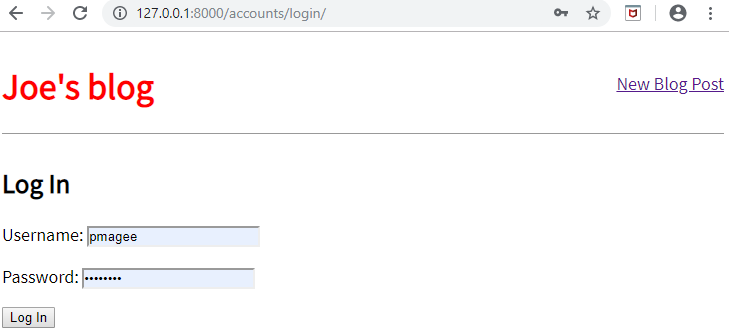
**Code**



Now the user will be redirected to the 'home' template which is our homepage.

1. Start up the Django server again with the command **python manage.py runserver** and navigate to our log in page: **http://127.0.0.1:8000/accounts/login/**

You will see the following:



1. Enter your user name and password for your superuser account and press the **Log In** button

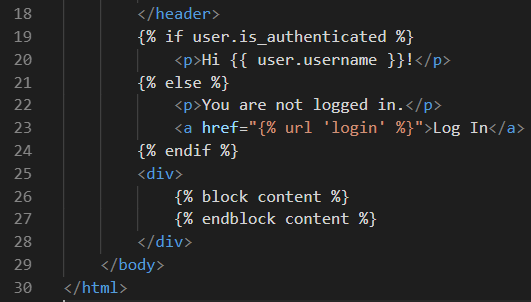
You will be redirected to the homepage. Notice that we didn’t add any view logic or create a database model because the Django auth system provided both for us automatically.

**Updated homepage**

We will update our **base.html** template so we display a message to users whether they are logged in or not. We can use the is\_authenticated attribute for this.

1. Update the **base.html** file with the following code starting beneath the closing **</header>** tag.

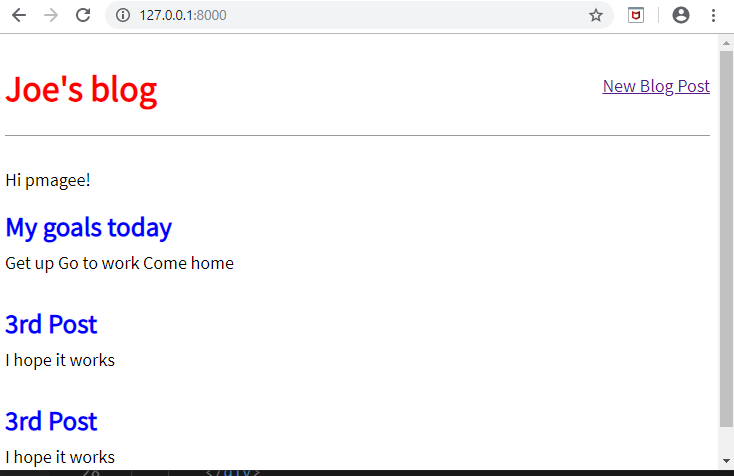
**Code**



If the user is logged in, we say hello to them by name, if not we provide a link to a

newly created log in page.

1. Run the server and access the site at: **http://127.0.0.1:8000/accounts/login/**



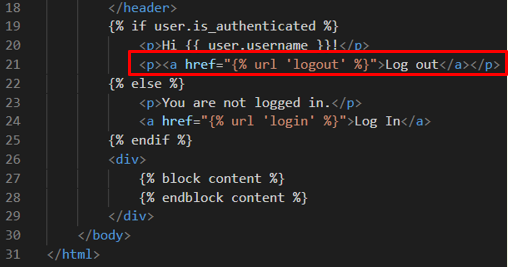
It worked! My superuser name is pmagee so that’s what I see on the page.

**Log out link**

Next, we will add a log out link that redirects the user to the homepage.

1. In the **base.html** file add a one-line {% url 'logout' %} link for logging out just below our user greeting.

**Code**

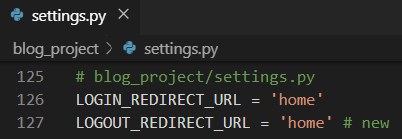


That’s all we need to do as the necessary view is provided to us by the Django auth

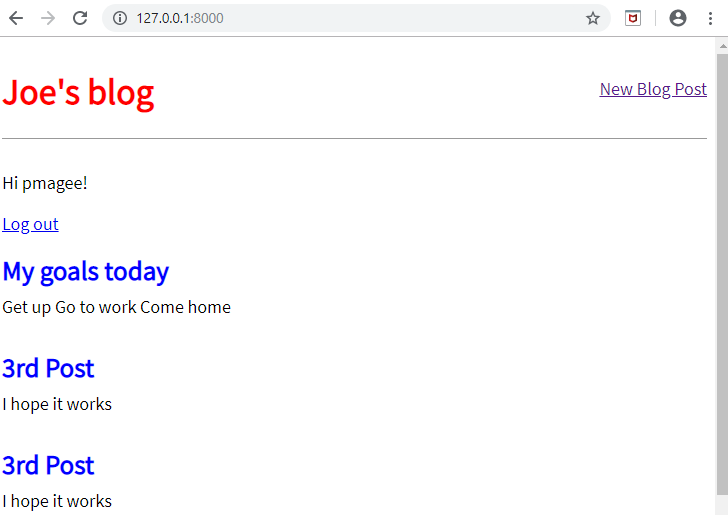
app. We do need to specify where to redirect a user upon log out though.

1. Update settings.py to provide a redirect link which is called, appropriately, **LOGOUT\_REDIRECT\_URL**. Add this line of code just beneath the log in redirect as shown below:

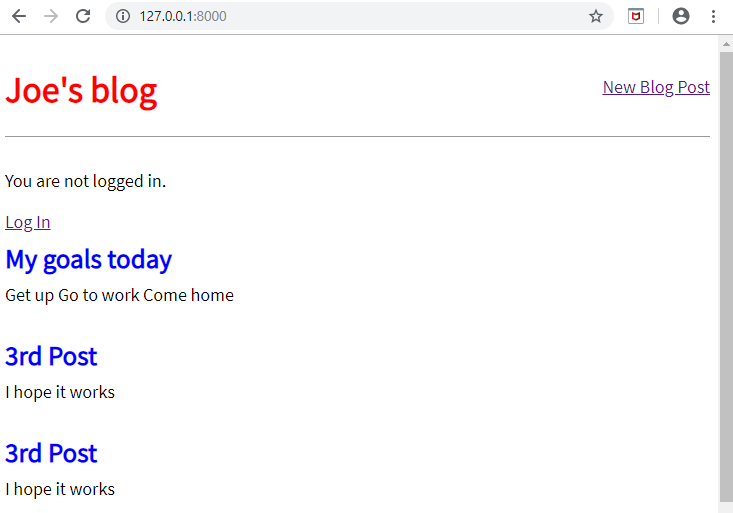
**Code**



1. Refresh the homepage & you will see it now has a “log out” link for logged in users.



1. Click the logout link and see that it takes you back to the homepage with a login link.
2. Try logging in and out a few times with your user account



**Sign up**

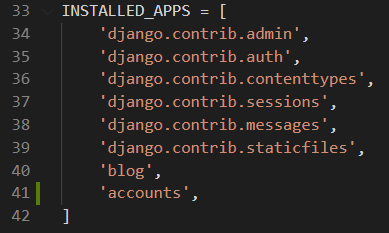
We need to write our own view for a sign up page to register new users, but Django provides us with a form class, UserCreationForm, to make things easier. By default, it comes with three fields: username, password1, and password2.

There are many ways to organize your code and URL structure for a user authentication system. Here we will create a dedicated new app called accounts, for our sign up page.

At the command line, create a new app called accounts.

1. Add the following line of code to the **INSTALLED\_APPS** setting in our **settings.py** file.

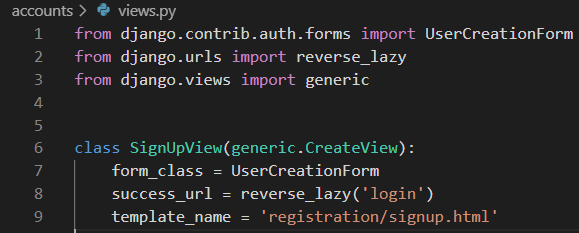
**Code**



**Create the view**

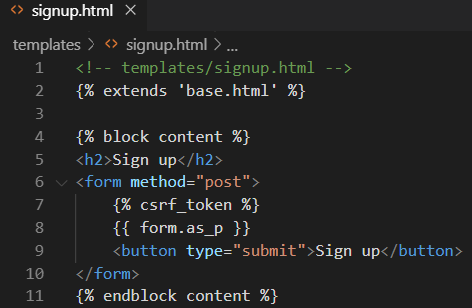
1. Open the accounts/views.py file
2. Delete the import at the top of the file
3. Add the following code to the file. This new view uses the built-in UserCreationForm and generic CreateView.

**Code**



1. Create a new file called **signup.html** in the **templates/registration** directory & populate it with the following code:

**Code**



This format is very similar to what we have done before. We extend our base template at the top, place our logic between **<form></form>** tags, use the csrf\_token for security, display the form’s content in paragraph tags with form.as\_p, and include a **submit** button.

**Configure the URLs**

1. Add the following line of code which adds a new URL path in blog\_project/urls.py pointing to this new app directly below where we include the built-in auth app.

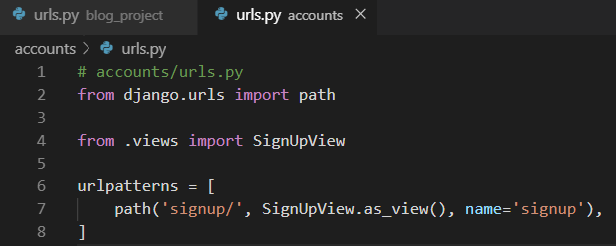
**Code**



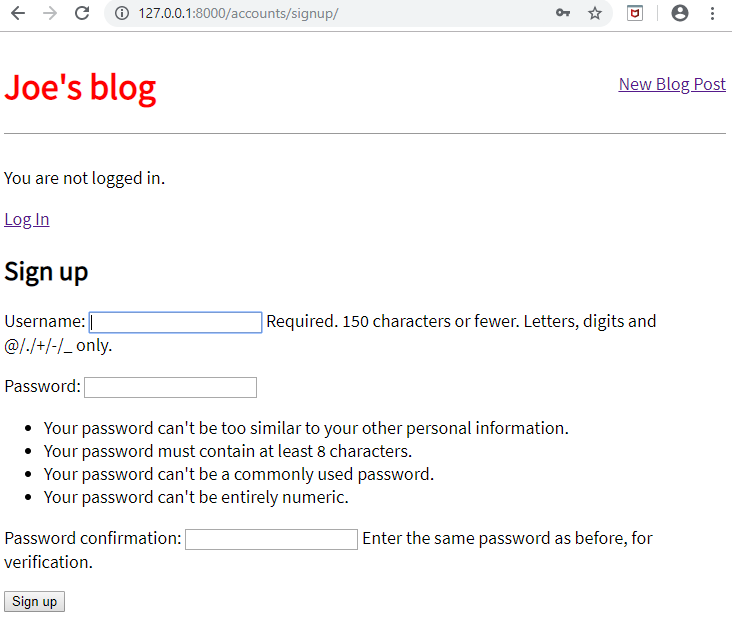
The order of our urls matters here because Django reads this file from top-to-bottom. Therefore, when we request the /accounts/signup url, Django will first look in auth, not find it, and then proceed to the accounts app.

1. In VS Code, create a new file inside the accounts folder called urls.pyand add the following code into it.

**Code**

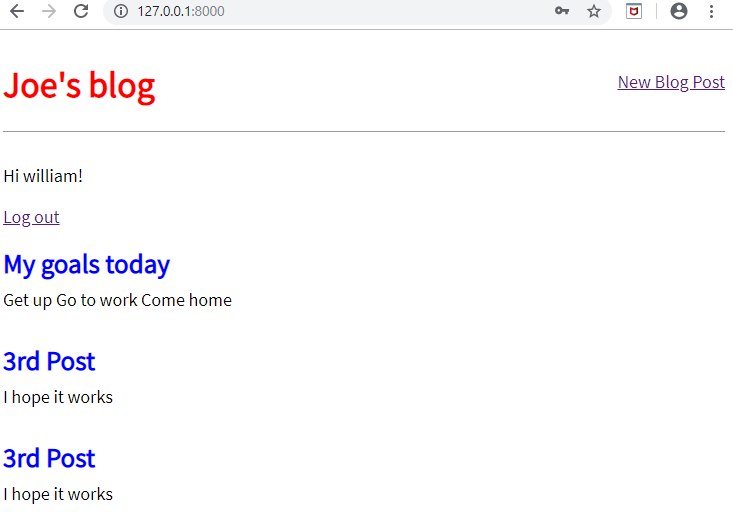


1. Start up the local server with the command **python manage.py runserver** and navigate to the newly created page: <http://127.0.0.1:8000/accounts/signup/>



Notice there is a lot of extra text that Django includes by default. We can customize this using something like the built-in messages framework but for now try out the form.

1. Create a new user and click **Sign up** which will redirect you to the log in page. Then after logging in successfully with my new user and password, you will be redirected to the homepage with our personalized “Hi username” greeting.



Run the following git commands to update the lab 4 local and remote repositories:

(env) djangoprojects\lab4>git add -A

(env) djangoprojects\lab4>git commit -m “lab 4 part 3 commit”

(env) djangoprojects\lab4>git push -u origin master