Dec 20, 2017

Previously we added login and logout pages to our Django app. In this tutorial we'll create a signup page so users can register for a new account.

The Django auth app provided us with built-in url and views for login and logout. All we needed to do was add a template for login. But to create a sign up page *we will* need to make our own view and url. Let's begin!

Complete source code can be found on Github.

Users app

Since we're making our own view and url for registration, we need to create a dedicated app. Let's call it `accounts` .

```
(accounts) $ ./manage.py startapp accounts
```

Make sure to add the new app to the `INSTALLED_APPS` setting in our `my_project/settings.py` file:

```python
# my_project/settings.py
INSTALLED_APPS = [
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',
    'accounts',
]
```

Then add a project-level url for the `accounts` app **above** our included Django `auth` app. Django will look top to bottom for url patterns so when it sees a url route within our `accounts` app that matches one in the built-in `auth` app, it will choose the `accounts` route first.

```
# my_project/urls.py
from django.contrib import admin
from django.urls import path, include
from django.views.generic.base import TemplateView

urlpatterns = [
    path('', TemplateView.as_view(template_name='home.html'), name='home'),
    path('admin/', admin.site.urls),
    path('accounts/', include('accounts.urls')),
    path('accounts/', include('django.contrib.auth.urls')),
]
```

Create a new `urls` file in our `accounts` app. Note that we are importing a view called `SignUp` which we'll implement in the next section.

```
(accounts) $ touch accounts/urls.py
```

```
# accounts/urls.py
from django.urls import path

from . import views


urlpatterns = [
    path('signup/', views.SignUp.as_view(), name='signup'),
]
```

Now for the `views.py` file:

```
# accounts/views.py
from django.contrib.auth.forms import UserCreationForm
from django.urls import reverse_lazy
from django.views import generic


class SignUp(generic.CreateView):
    form_class = UserCreationForm
    success_url = reverse_lazy('login')
    template_name = 'signup.html'
```

We're subclassing the generic class-based view CreateView in our `SignUp` class. We specify the use of the built-in `UserCreationForm` and the *not-yet-created* template at `signup.html`. And we use reverse_lazy to redirect the user to the `login` page upon successful registration.

Why use `reverse_lazy` instead of `reverse` I hope you're asking? The reason is that *for all generic class-based views* the urls are not loaded when the file is imported, so we have to use the lazy form of reverse to load them later when they're available.

Ok final step. Create a template `signup.html` within the existing project-level `templates` folder.

```
(accounts) $ touch templates/signup.html
```

Then populate it with this code that looks almost exactly like what we used for `login.html`.

```
<!-- templates/signup.html -->
{% extends 'base.html' %}

{% block title %}Sign Up{% endblock %}

{% block content %}
  <h2>Sign up</h2>
  <form method="post">
    {% csrf_token %}
    {{ form.as_p }}
    <button type="submit">Sign up</button>
  </form>
{% endblock %}
```
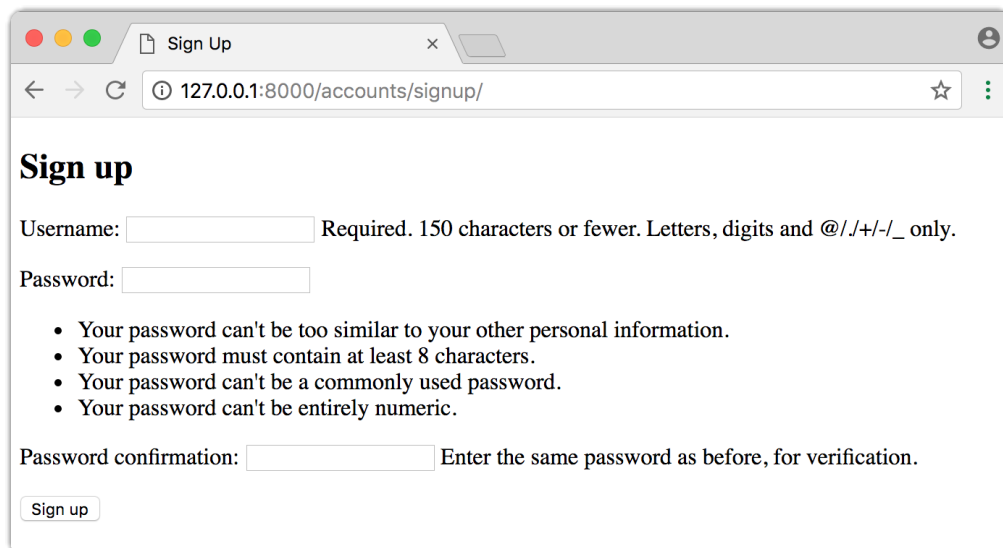
And we're done! To confirm it all works, spin up our local server with `./manage.py runserver` and navigate to http://127.0.0.1:8000/accounts/signup/.
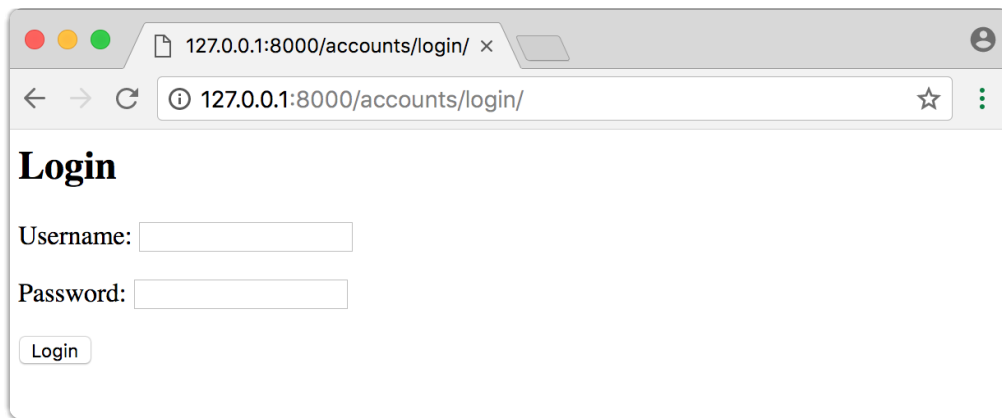
The extra text with tips on usernames and passwords comes from Django. We can customize that too but it requires a little more work and is beyond the scope of this tutorial.

Sign up for a new account and hit the "Sign up" button. You will be redirected to the login page http://127.0.0.1:8000/accounts/login/ where you can log in with your new account.
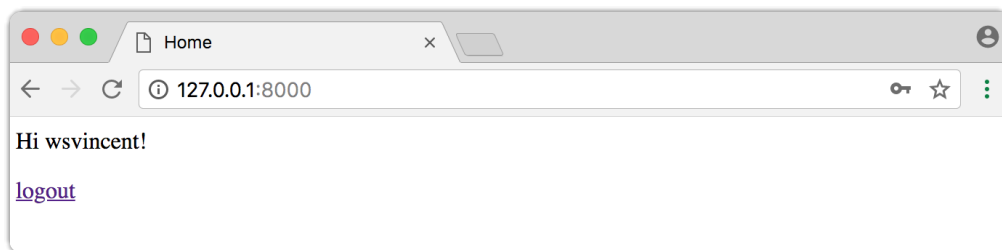
Login

Username: [   ]

Password: [   ]

[Login]

And then after a successful login you'll be redirect to the homepage with a personalized "Hi **username**!" greeting.

Hi wsvincent!

logout

Next Steps

We've successfully created a new signup functionality to go alongside our existing login and logout. There's only one thing missing: add the ability for users to reset their passwords. We'll cover this in part 3, Django Password Reset Tutorial.

If you'd like to learn more about Django and build step-by-step multiple web applications, check out the free online book I wrote Django For Beginners.

Subscribe for more articles on Python & JavaScript

[Your Email   ]

[**Sign Up**]

© William S. Vincent

will@wsvincent.com