[illegible]

Dec 20, 2017

In this tutorial we'll learn how to configure login/logout functionality in Django 2.0 with the built-in user authentication system. Future tutorials cover how to implement signup as well as a password reset sequence. By the end of these tutorials you'll have a complete user authentication flow for your future Django projects.

This tutorial assumes you're already familiar with how to configure a new Django project. If you need help, please refer to Django for Beginners which covers the topic in more detail.

Note if you're looking for information on using a custom Django user model or adding social auth with django-allauth, I have separate tutorials on those too. And when you're ready to put it all together check out my Django Login Mega-Tutorial which shows how to combine a custom user model with social login with Gmail.

Complete source code can be found on Github.

Setup

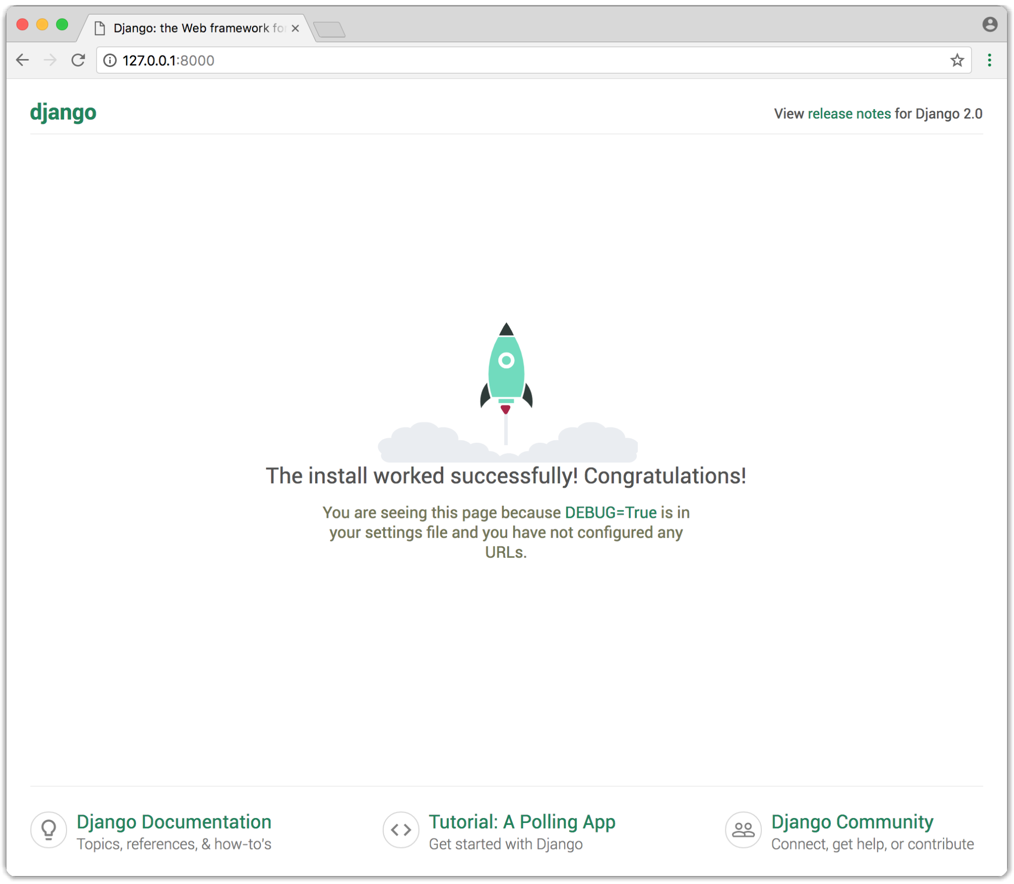
Start by creating a new Django project. We can do all of the normal configuration from the command line:

- create a new virtual environment called `accounts`
- install Django
- create a new Django project called `my_project`
- create a new Sqlite database with `migrate`
- run the local server

Here are the commands to run:

```
$ python3 -m venv ~/.virtualenvs/accounts
$ source ~/.virtualenvs/accounts/bin/activate
(accounts) $ pip install django
(accounts) $ django-admin.py startproject my_project .
(accounts) $ ./manage.py migrate
(accounts) $ ./manage.py runserver
```

If you navigate to http://127.0.0.1:8000 you'll see the Django welcome screen.


Django: the Web framework fo
127.0.0.1:8000
django
View release notes for Django 2.0

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

The Django auth app

Django automatically installs the `auth` app when a new project is created. Look in the `settings.py` 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', # Yoohoo!!!!
    '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.

```
# my_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')),
]
```

On the second line we're also importing `include` from `django.urls`. And I've chosen to include the `auth` app at `accounts/` but you can use any url pattern you want.

The `auth` app we've now included provides us with several authentication 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 `registration` and the requisite `login.html` file within it. From the command line type `Control-c` to quit our local server and enter the following commands:

```
(accounts) $ mkdir templates
(accounts) $ mkdir templates/registration
(accounts) $ touch templates/registration/login.html
```

Then include this template code in our `login.html` file:

```
<!-- templates/registration/login.html -->
<h2>Login</h2>
<form method="post">
 {% csrf_token %}
 {{ form.as_p }}
 <button type="submit">Login</button>
</form>
```

This is a standard Django form using `POST` to send data and `{% csrf_token %}` tags 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.

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.

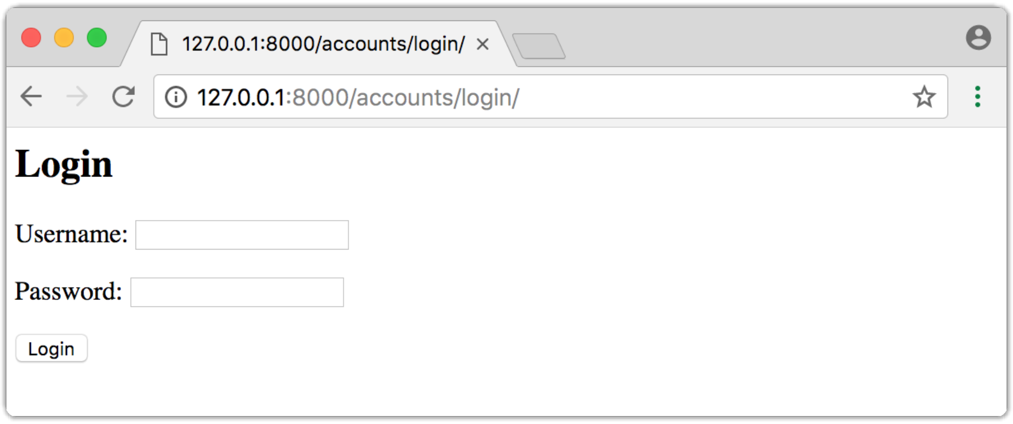
```
# my_project/settings.py
TEMPLATES = [
    {
        ...
        'DIRS': [os.path.join(BASE_DIR, 'templates')],
        ...
    },
]
```

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.

```
# my_project/settings.py
LOGIN_REDIRECT_URL = '/'
```

We're actually done at this point!

If you now start up the Django server again with `./manage.py runserver` and navigate to our login page at http://127.0.0.1:8000/accounts/login/ you'll see the following.

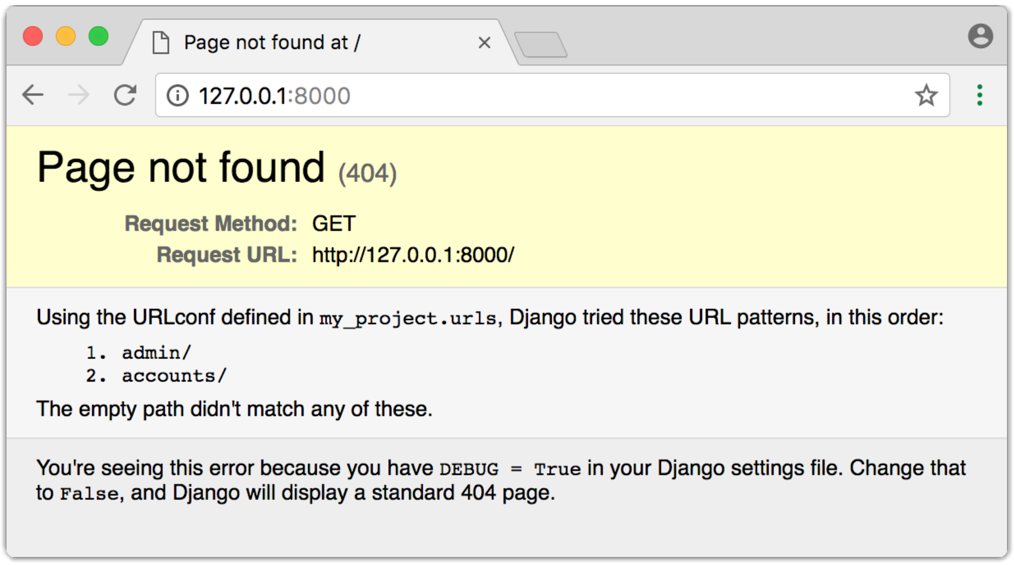

127.0.0.1:8000/accounts/login/
127.0.0.1:8000/accounts/login/
Login
Username:
Password:
Login

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. Quit the server with `Control-c` and then run the command `./manage.py createsuperuser`. Answer the prompts and note that your password will not appear on the screen when typing for security reasons.

```
(accounts) $ ./manage.py createsuperuser
Username (leave blank to use 'wsv'):
Email address: will@wsvincent.com
Password:
Password (again):
Superuser created successfully.
```

Now spin up the server again with `./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.


Page not found at /

127.0.0.1:8000

Page not found (404)

Request Method: GET
Request URL: http://127.0.0.1:8000/

Using the URLconf defined in `my_project.urls`, Django tried these URL patterns, in this order:

```
1. admin/
2. accounts/
```

The empty path didn't match any of these.

You're seeing this error because you have `DEBUG = True` in your Django settings file. Change that to `False`, and Django will display a standard 404 page.

We know that our login worked because we were redirected to the homepage, but we haven't created it yet so we see the error *Page not found*. Let's fix that!

Create a homepage

We want a simple homepage that will display one message to logged out users and another to logged in users.

First quit the local server with `Control-c` and then create new `base.html` and `home.html` files. Note that these are located within the `templates` folder but *not* within `templates/registration/` where Django auth looks by default for user auth templates.

```
(accounts) $ touch templates/base.html
(accounts) $ touch templates/home.html
```

Add the following code to each:

```
<!-- 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' %}">login</a>
{% endif %}
{% endblock %}
```

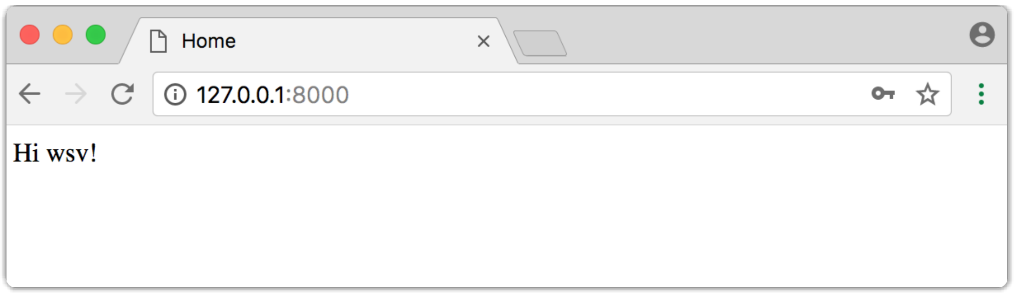
While we're at it, we can update `login.html` too to extend our new `base.html` file:

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

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

{% block content %}
<h2>Login</h2>
<form method="post">
  {% csrf_token %}
  {{ form.as_p }}
  <button type="submit">Login</button>
</form>
{% endblock %}
```

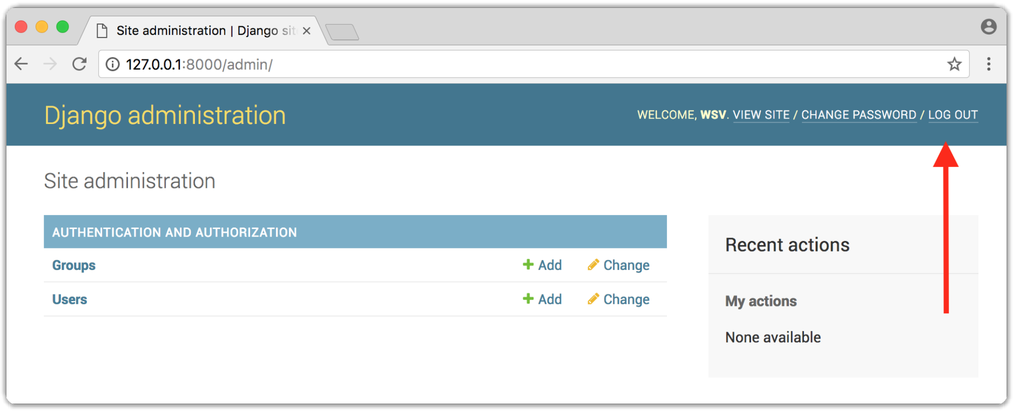
Now update our `urls.py` file so we display the homepage. On the third line, import `TemplateView` and then add a urlpattern for it.

```
# 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('django.contrib.auth.urls')),
]
```

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


Hi wsv!

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.


Django administration

WELCOME, WSV. VIEW SITE / CHANGE PASSWORD / LOG OUT

Site administration

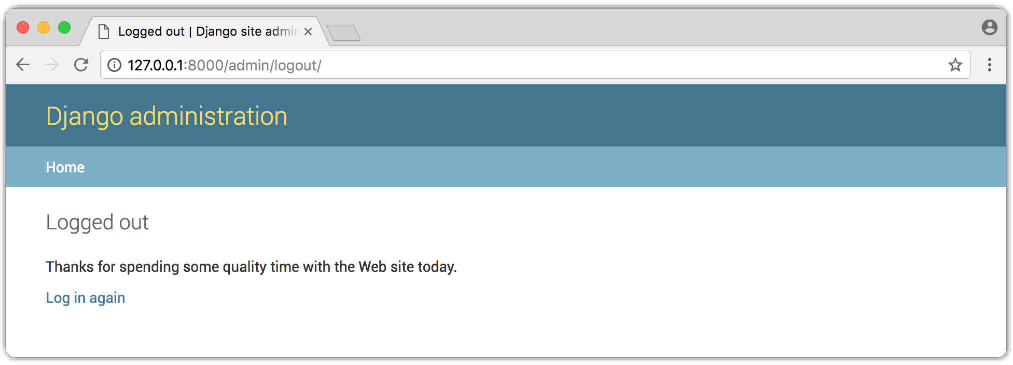
AUTHENTICATION AND AUTHORIZATION		
Groups	+ Add	Change
Users	+ Add	Change

Recent actions

My actions

None available

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


Django administration

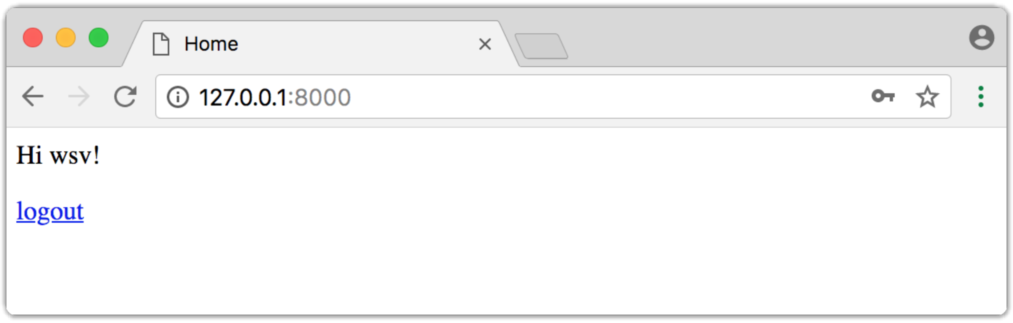
Home

Logged out

Thanks for spending some quality time with the Web site today.

Log in again

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.


Hi wsv!

logout

Logout link

Let's add a logout link to our page so users can easily toggle back and forth between the two states. Fortunately the Django `auth` app already provides us with a built-in url and view for this. And if you think about it, we don't need to display anything on logout so there's no need for a template. All really we do after a successful "logout" request is redirect to another page.

So let's first add a link to the built-in `logout` url in our `base.html` file:

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

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

{% block content %}
{% if user.is_authenticated %}
  Hi {{ user.username }}!
  <p><a href="{% url 'logout' %}">logout</a></p>
{% else %}
  <p>You are not logged in</p>
  <a href="{% url 'login' %}">login</a>
{% endif %}
{% endblock %}
```

Then update `settings.py` with our redirect link which is called `LOGOUT_REDIRECT_URL`. Add it right next to our login redirect so the bottom of the `settings.py` file should look as follows:

```
# my_project/settings.py
LOGIN_REDIRECT_URL = '/'
LOGOUT_REDIRECT_URL = '/'
```

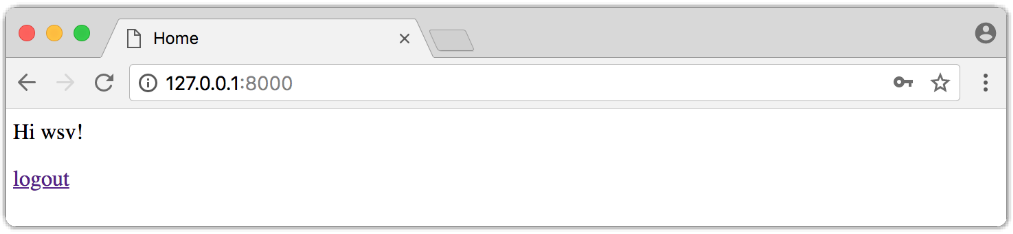
Actually, now that we have a homepage view we should use that instead of our current hardcoded approach. What's the url name of our homepage? It's `home`, which we named in our `my_project/urls.py` file:

```
# my_project/urls.py
    ...
    path('', TemplateView.as_view(template_name='home.html'), name='home'),
    ...
```

So we can replace `'/'` with `home` at the bottom of the `settings.py` file:

```
# my_project/settings.py
LOGIN_REDIRECT_URL = 'home'
LOGOUT_REDIRECT_URL = 'home'
```

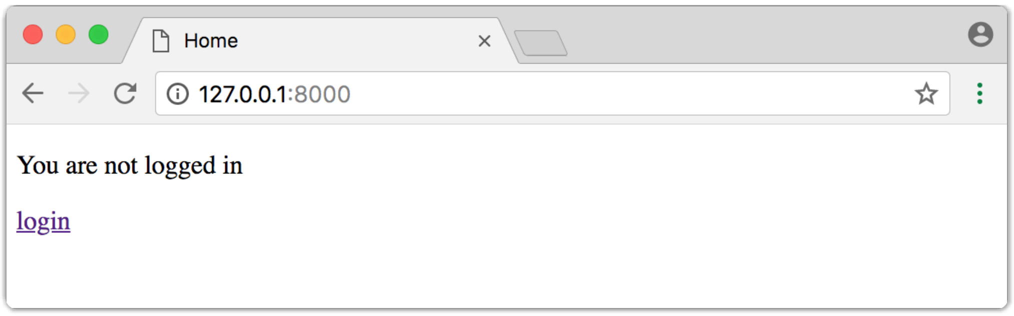
Now if you revisit the homepage and login you'll be redirected to the new hompage that has a "logout" link for logged in users.



Hi wsv!

logout

Clicking it takes you back to the homepage with a "login" link.



You are not logged in

login

Conclusion

With very little code we have a robust login and logout authentication system. It probably feels a bit like magic since the `auth` app did much of the heavy lifting for us. One of the nice things about Django is while it provides a lot of functionality out-of-the-box, it's designed to be customizable too.

In the next post, Django Signup Tutorial, we'll learn how to add a signup page to register new users.

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.

will@wsvincent.com