Table of Contents

This slide deck consists of slides used in 3 lecture videos in Week 2. Below is a list of shortcut hyperlinks for you to jump into specific sections.

Login and Logout

Charles Severance
www.dj4e.com

https://samples.dj4e.com/authz/

https://github.com/csev/dj4e-samples/tree/master/authz

User authentication in Django

Django comes with a user authentication system. It handles user accounts, groups, permissions and cookie-based user sessions. The authentication system consists of:

- Users
- Permissions: Binary (yes/no) flags designating whether a user may perform a certain task.
- Groups: A generic way of applying labels and permissions to more than one user.
- A configurable password hashing system
- Forms and view tools for logging in users, or restricting content
- A pluggable backend system

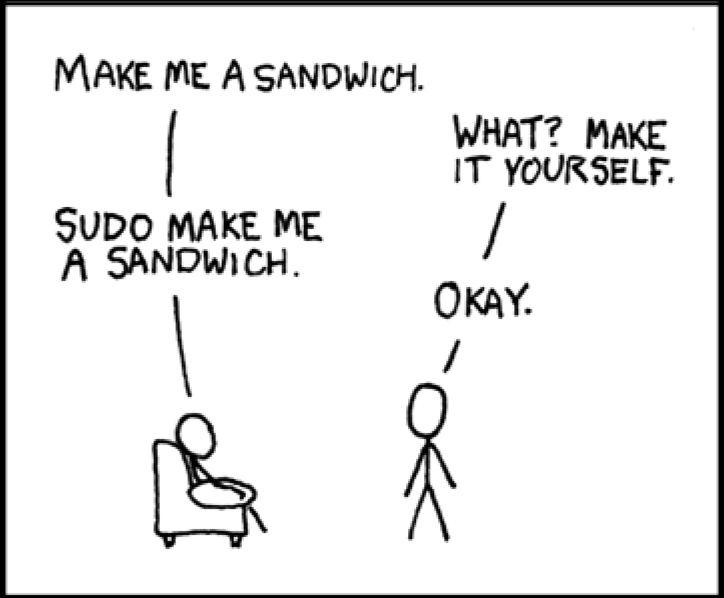
Authentication support is bundled as a Django **contrib** module in **django.contrib.auth**. By default, the required configuration is already included in **settings.py**.

https://docs.djangoproject.com/en/3.0/topics/auth/

Making the super user

- We need to "bootstrap" our system and make a user that can log into the admin page and make more users

```
dj4e-samples$ python3 manage.py createsuperuser
Username: dj4e-samples
Email address: csev@umich.edu
Password:
Password (again):
Superuser created successfully.
```



https://xkcd.com/149/

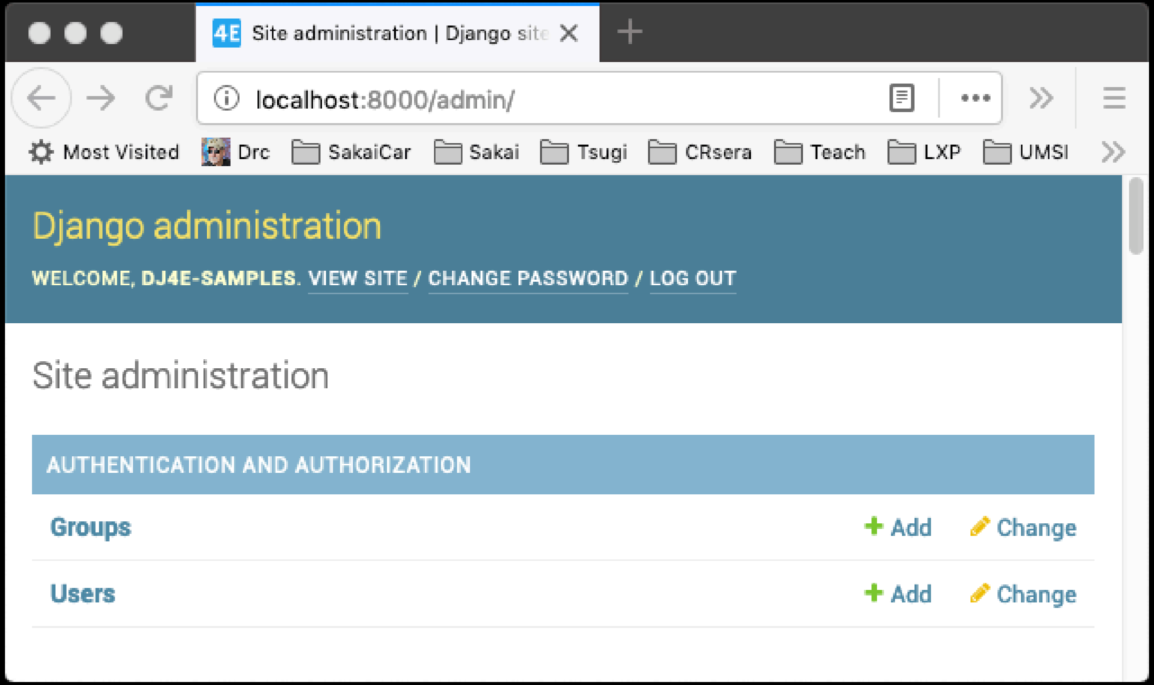
Wiping out your database

- Sometimes you want to clear out and re-initialize your db.sqlite3 file
- The super users and users are stored in the database so when you remove it, you need to re-create the super users.

```
dj4e-samples$ rm db.sqlite3
dj4e-samples$ python3 manage.py migrate
dj4e-samples$ python3 manage.py createsuperuser
Username: dj4e-samples
Email address: csev@umich.edu
Password:
Password (again):
Superuser created successfully.
```

Additional Users and Permissions

- Once you have a super user you can log into your application and create additional new users, associate them with groups, and give them permissions in the "/admin" user interface
- Many applications don't need to use the groups or permissions features of Django



Django administration

WELCOME, DJ4E-SAMPLES. VIEW SITE / CHANGE PASSWORD / LOG OUT

Site administration

AUTHENTICATION AND AUTHORIZATION

Groups	+ Add	Change
Users	+ Add	Change

Logging Users into Our Application

Sessions are not "logging in"

- A session is a way of marking a browser and storing data on the server which can be stored and retrieved across multiple request-response-cycles
- Sessions exist irrespective of whether or not the user is logged in
- When the user passes the login check, the server adds data to the session identifying the user
- When the user logs out, that information in the session is removed

- Sessions are required to implement login

Sessions, Users, Login, and Django

- Login functionality is built into Django and included in your **settings.py** by default

dj4e-samples/dj4e-samples/settings.py

```
INSTALLED_APPS = [
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
...
```

https://docs.djangoproject.com/en/3.0/topics/auth/default

Sessions, Users, Login, and Django

- We need to add a path to the code that gives us login and logout urls
- We can reverse lookup these urls using the 'login' and 'logout' view names

dj4e-samples/dj4e-samples/urls.py

```
urlpatterns = [
    path('', include('home.urls')),
    path('admin/', admin.site.urls),
    path('accounts/', include('django.contrib.auth.urls')),

...
```

https://docs.djangoproject.com/en/3.0/topics/auth/default dj4e-samples/authz/views.py

```
from django.urls import reverse

class DumpPython(View) :
    def get(self, req):
        resp = "<pre>\nUser Data in Python:\n\n"
        resp += "Login url: " + reverse('login') + "\n"
        resp += "Logout url: " + reverse('logout') + "\n\n"
```

You can get the login and logout urls using reverse()

https://samples.dj4e.com/authz/python

```
User Data in Python:


Login url: /accounts/login/
Logout url: /accounts/logout/
```

Where to go after login / logout completion

- We want to transfer the user to a login page from many pages in our application and when they successfully log in, we want to bring them back to our page or some other page
- The "next=" parameter tells login or logout where to *redirect* the user after login

dj4e-samples/authz/urls.py

```
urlpatterns = [
    path('', TemplateView.as_view(template_name='authz/main.html')),
    path('open', views.OpenView.as_view(), name='open'),
    path('apereo', views.ApereoView.as_view(), name='apereo'),
    path('manual', views.ManualProtect.as_view(), name='manual'),
    path('protect', views.ProtectView.as_view(), name='protect'),
    path('python', views.DumpPython.as_view(), name='python'),
]
```

dj4e-samples/authz/templates/authz/main.html

```
<li>
        Go to <a href="{% url 'authz:open' %}">{% url 'authz:open' %}</a>
            (no login required)
    </li>
    <li>
        Go to <a href="{% url 'authz:apereo' %}">{% url 'authz:apereo' %}</a>
            (no login required)
    </li>
    <li>
        Go to <a href="{% url 'authz:manual' %}">{% url 'authz:manual' %}</a>
            (protected by user.is_authenticated)
    </li>
    <li>
        Go to <a href="{% url 'authz:protect' %}">{% url 'authz:protect' %}</a>
            (protected by LoginRequiredMixin)
    </li>
    <li>
        Go to <a href="{% url 'authz:python' %}">{% url 'authz:python' %}</a>
            dump request.user data in python
    </li>
```

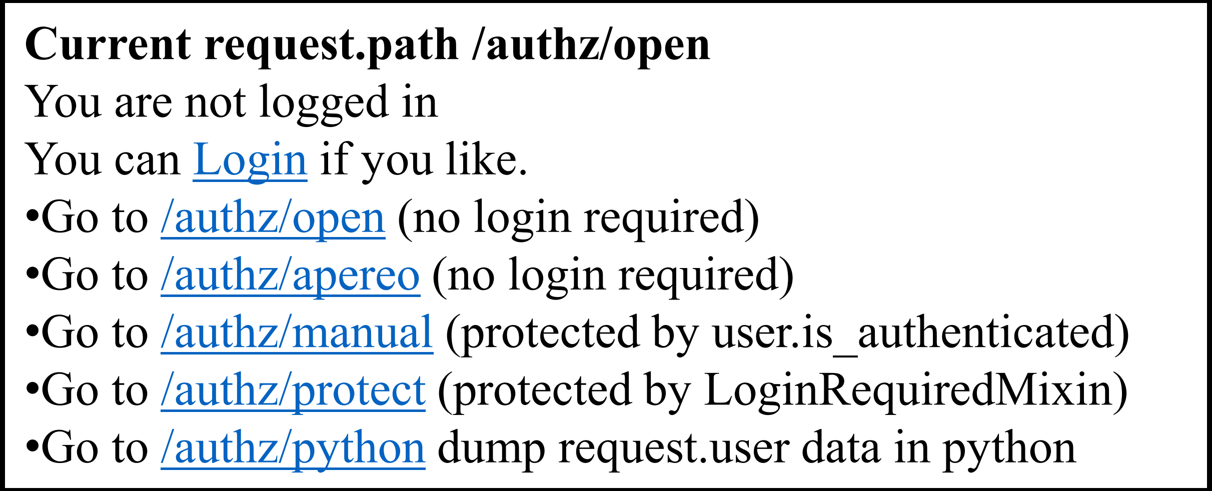
dj4e-samples/authz/templates/authz/main.html

```
<h1>Current request.path {{ request.path }}</h1>
{% if user.is_authenticated %}
<p>Authenticated as
<pre>
Name: {{ user.get_full_name }}
Email: {{ user.email }}
Id: {{ user.id }}
</pre>
</p>
<p>You can <a href="{% url 'logout' %}?next={% url 'authz:open' %}">Logout</a></p>
{% else %}
<p>You are not logged in</p>
<p>You can <a href="{% url 'login' %}?next={{ request.path }}">Login</a> if you
like.</p>
{% endif %}
```

```
dj4e-samples/authz/templates/authz/main.html
```

```
<p>You can <a href="{% url 'login' %}?next={{ request.path }}">
Login</a> if you like.</p>
```

```
https://samples.dj4e.com/authz/open
```



Current request.path /authz/open

You are not logged in

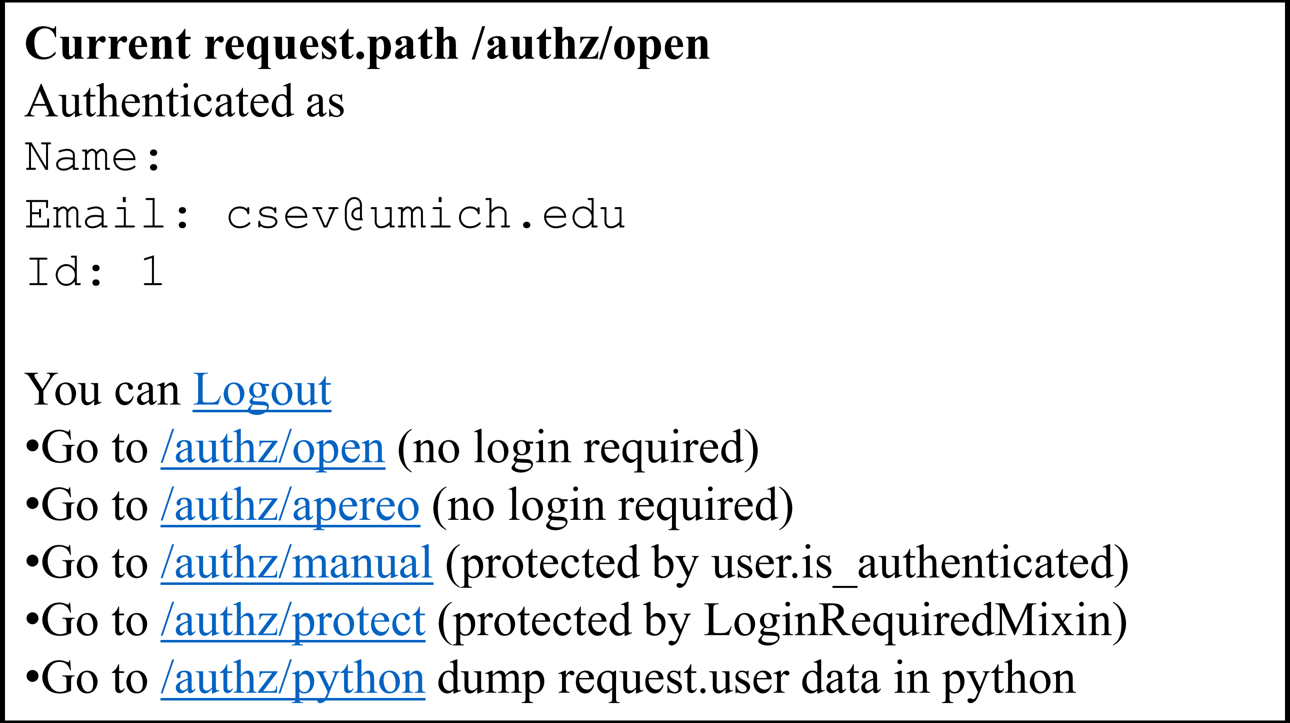
You can Login if you like.

- Go to /authz/open (no login required)
- Go to /authz/apereo (no login required)
- Go to /authz/manual (protected by user.is_authenticated)
- Go to /authz/protect (protected by LoginRequiredMixin)
- Go to /authz/python dump request.user data in python

```
<p>You can <a href="/accounts/login/?next=/authz/open">Login</a> if you like.</p>
```

```
<p>You can <a href="{% url 'logout' %}?next={% url 'authz:open' %}">
Logout</a></p>
```

```
https://samples.dj4e.com/authz/open
```



Current request.path /authz/open

Authenticated as

```
Name:
Email: csev@umich.edu
Id: 1
```

You can Logout

- Go to /authz/open (no login required)
- Go to /authz/apereo (no login required)
- Go to /authz/manual (protected by user.is_authenticated)
- Go to /authz/protect (protected by LoginRequiredMixin)
- Go to /authz/python dump request.user data in python

When logging out, make sure to set next to a url that does not require login. If you do – the user will be in a frustrating logout / login loop.

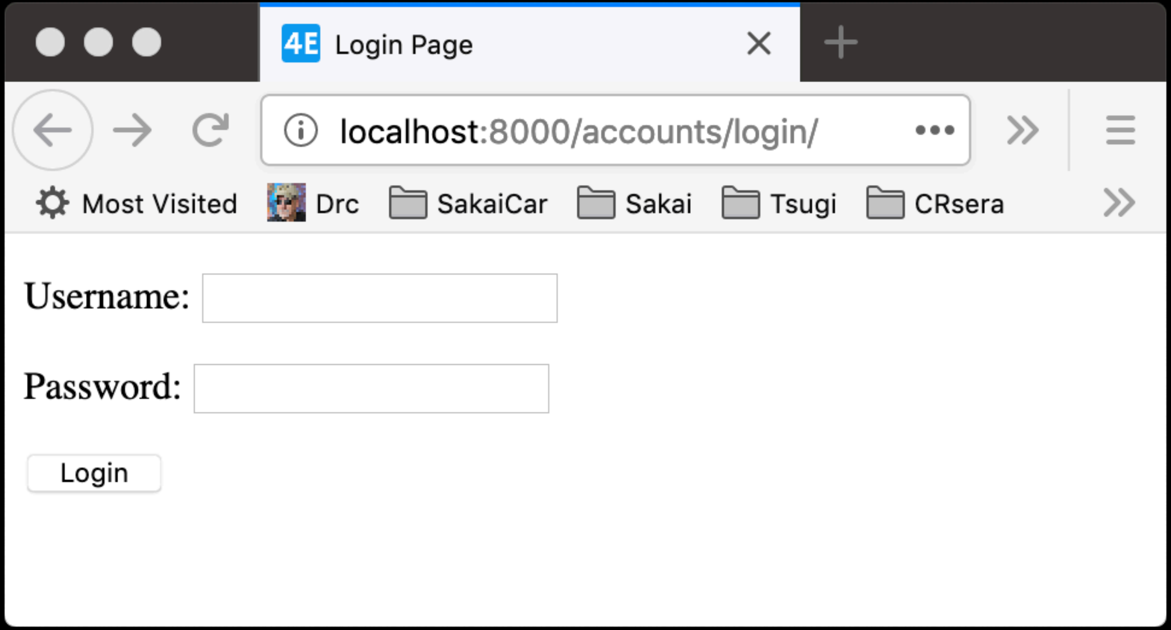
```
<p>You can <a href="/accounts/logout/?next=/authz/open">Logout</a></p>
```

The Login Page

Look and Feel - Login Template

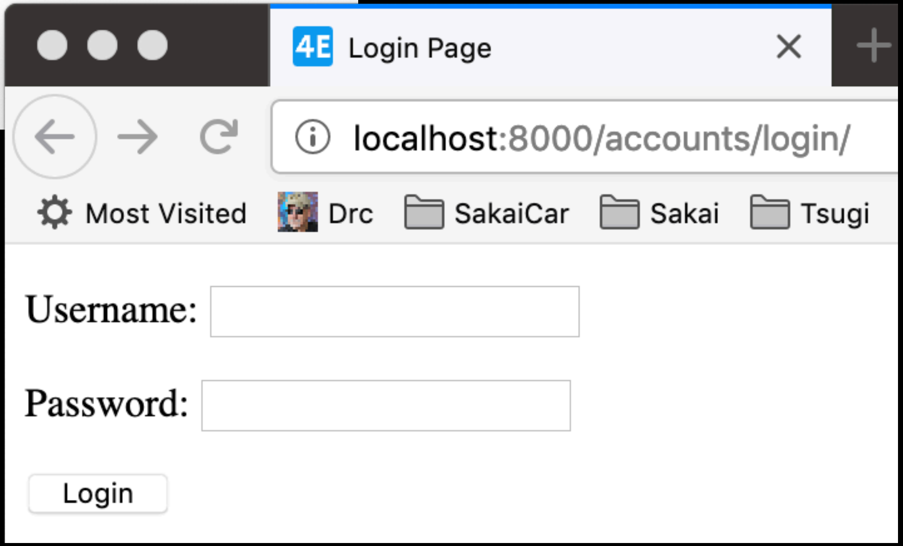
- To allow us to control the look and feel of the login page we must provide a template called "registration/login.html"
- Django describes what needs to be in this template
- We can put this in any of our application templates folders

`https://samples.dj4e.com/accounts/login`



https://docs.djangoproject.com/en/3.0/topics/auth/default/#django.contrib.auth.views.LoginView dj4e-samples/home/templates/registration/login.html

```
<html>
    <title>Login Page</title>
</head>
<body>
<form method="post" action="{% url 'login' %}">
{% csrf_token %}
{{ form.as_p }}
<input type="submit" class="btn btn-primary" value="Login" />
<input type="hidden" name="next" value="{{ next }}" />
</form>
</body>
```


4E Login Page

localhost:8000/accounts/login/

Most Visited Drc SakaiCar Sakai Tsugi

Username:

Password:

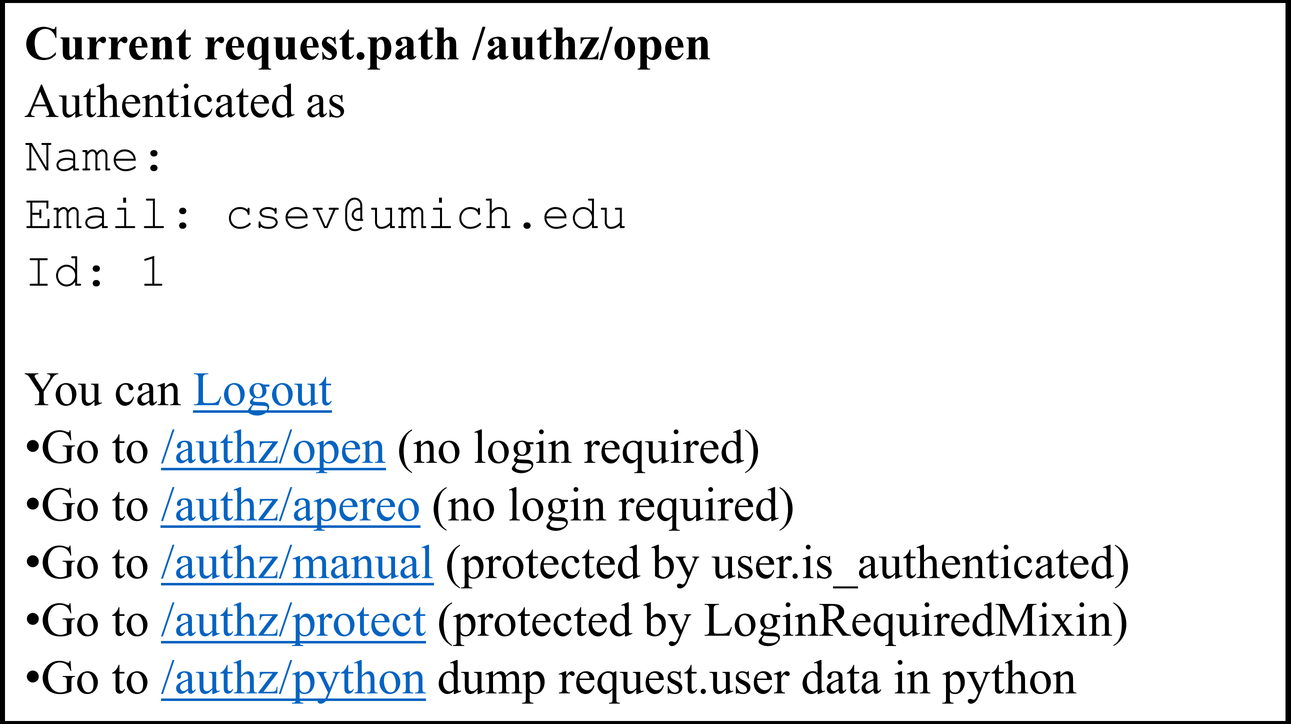
Login

https://samples.dj4e.com/accounts/login

Data for the logged in user

https://samples.dj4e.com/authz/open

```
{% if user.is_authenticated %}
<p>Authenticated as
<pre>
Name: {{ user.get_full_name }}
Email: {{ user.email }}
Id: {{ user.id }}
</pre>
</p>
{% else %}
<p>You are not logged in</p>
{% endif %}
```



Current request.path /authz/open

Authenticated as

```
Name:
Email: csev@umich.edu
Id: 1
```

You can Logout

- •Go to /authz/open (no login required)
- •Go to /authz/apereo (no login required)
- •Go to /authz/manual (protected by user.is_authenticated)
- •Go to /authz/protect (protected by LoginRequiredMixin)
- •Go to /authz/python dump request.user data in python

dj4e-samples/authz/templates/authz/main.html

Accessing user data in Python

https://samples.dj4e.com/authz/python

dj4e-samples/authz/views.py

```
User Data in Python:

Login url: /accounts/login/
Logout url: /accounts/logout/

User: dj4e-samples
Email: csev@umich.edu
```

```
class DumpPython(View) :
    def get(self, req):
        resp = "<pre>\nUser Data in Python:\n\n"
        resp += "Login url: " + reverse('login') + "\n"
        resp += "Logout url: " + reverse('logout') + "\n\n"
        if req.user.is_authenticated:
            resp += "User: " + req.user.username + "\n"
            resp += "Email: " + req.user.email + "\n"
        else:
            resp += "User is not logged in\n"

        resp += "\n"
        resp += "</pre>\n"
        resp += """<a href="/authz">Go back</a>"""
        return HttpResponse(resp)
```

Views that require a logged in user

- Many of your views need to make sure that someone is logged in before performing some operation that depends on the request.user data being set
  - request.user.id
  - request.user.email
- You could check user.is_authenticated at the beginning of each view and if the user is not logged, redirect them to reverse('login') with the appropriate next= parameter

https://docs.djangoproject.com/en/3.0/topics/auth/default/#the-loginrequired-mixin dj4e-samples/authz/views.py

```python
from django.utils.http import urlencode

class ManualProtect(View) :
    def get(self, request):
        if not request.user.is_authenticated :
            loginurl = reverse('login')+'?'+urlencode({'next': request.path})
            return redirect(loginurl)
        return render(request, 'authz/main.html')


from django.contrib.auth.mixins import LoginRequiredMixin

class ProtectView(LoginRequiredMixin, View) :
    def get(self, request):
        return render(request, 'authz/main.html')
```

https://docs.djangoproject.com/en/3.0/topics/auth/default/#the-loginrequired-mixin

dj4e-samples/authz/templates/authz/main.html

```
<h1>Current request.path {{ request.path }}</h1>
{% if user.is_authenticated %}
<p>Authenticated as
<pre>
Name: {{ user.get_full_name }}
Email: {{ user.email }}
Id: {{ user.id }}
</pre>
</p>
<p>You can <a href="{% url 'logout' %}?next={% url 'authz:open' %}">Logout</a></p>
{% else %}
<p>You are not logged in</p>
<p>You can <a href="{% url 'login' %}?next={{ request.path }}">Login</a> if you
like.</p>
{% endif %}
```

Summary - Setting up login

- Add django.contrib.auth entries to INSTALLED_APPS and urlpatterns
- Create a template named 'registration/login.html'
- Get urls for login and logout using reverse, reverse_lazy, or the url template tag
- Add the "next=" parameter to those URLs to bring the user back to a page after successful login or logout
- Add LoginRequiredMixin to views that can only be accessed by a logged in user

Acknowledgements / Contributions

Initial Development: Charles Severance, University of Michigan School of Information

Insert new Contributors and Translators here including names and dates

Continue new Contributors and Translators here