Django's Authentication System

COMP 8347

Usama Mir

Usama.Mir@unwindsor.ca

Django Authentication

Topics

- Django's Authentication System
  - User Objects
  - Authentication
  - Login and Logout
- Permissions and Authorization

Authentication System

- Django's authentication system consists of:
  - ***User*** objects
  - A configurable password hashing system
  - Forms and view tools for logging in users, or restricting content.
  - **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.

Installation

- Add these 2 items in INSTALLED_APPS setting:
  - 'django.contrib.auth' : contains the core of the authentication framework, and its default models.
  - 'django.contrib.contenttypes': allows permissions to be associated with models you create.
- Add these 2 items in MIDDLEWARE_CLASSES setting:
  - SessionMiddleware: manages sessions across requests.
  - AuthenticationMiddleware: associates users with requests using sessions.
- By default: already included in settings.py.

User Objects

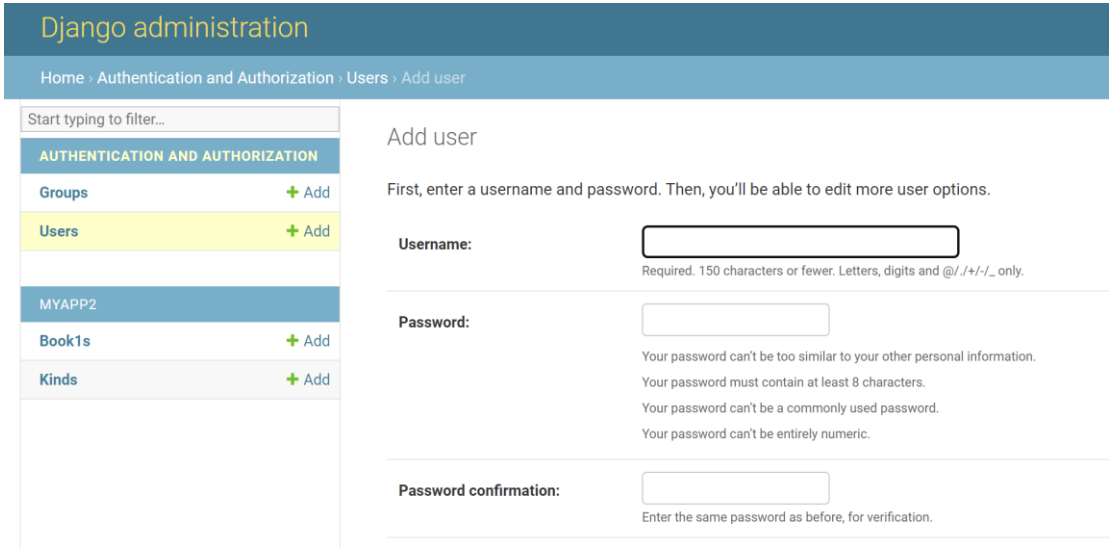
- User objects are the core of the authentication system.
  - Typically represent people interacting with your site.
  - Used to enable things like restricting access, registering user profiles etc.
  - Only one class of user exists in Django's authentication framework
    - Different user types e.g., 'superusers' or admin 'staff' users are just user objects with special attributes set
      - not different classes of user objects.

User Attributes

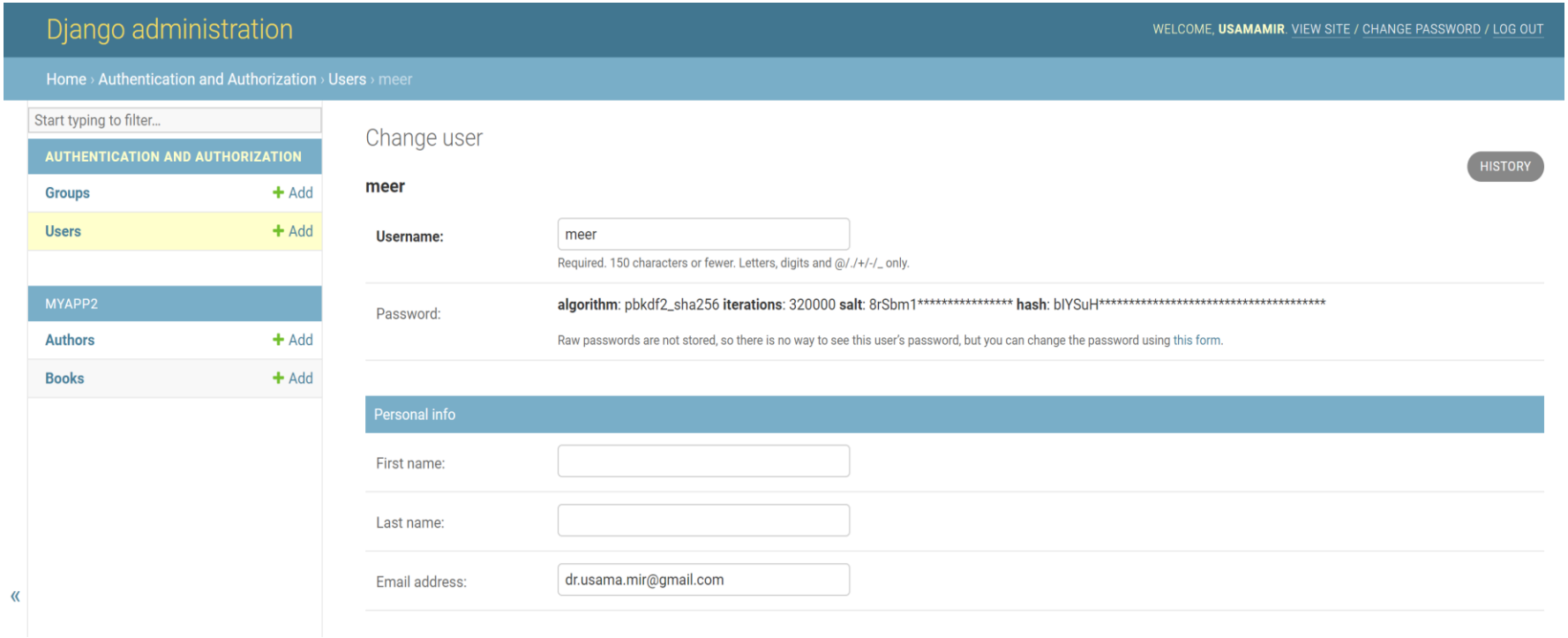
- The primary attributes of the default user are:
  - ***Username***: Required. 30 characters or fewer.
    - May contain alphanumeric, _, @, +, . and - characters.
  - *first_name*: Optional. 30 characters or fewer.
  - *last_name*: Optional. 30 characters or fewer.
  - *Email*: Optional. Email address.
  - ***Password***: Required.
    - A hash of, and metadata about, the password.
    - Django doesn't store the raw password.
    - <algorithm>$<iterations>$<salt>$<hash>
    - Algorithm = PBKDF2, Hash = SHA256
    - Iteration = 320000

Using Admin Interface

- Admin module can be used to view and manage users, groups, and permissions.
  - Both django.contrib.admin and django.contrib.auth must be installed.
  - The "Add user" admin page requires you to choose a username and password before allowing you to edit the rest of the user's fields.
  - User passwords are **not** displayed in the admin (nor stored in the database).
    - a link to a password change form allows admins to change user passwords



Using Admin Interface



Changing Passwords

- Django does not store raw (clear text) passwords on the user model,
  - It only stores a hash.
  - Do **not** manipulate the password attribute of the user directly.
    - **user.password = ‘new password' # Don’t do this!**
  - Passwords can be changed using **set_password()**

```
from django.contrib.auth.models import User
u = User.objects.get(username='john')
u.set_password('new password')
u.save()
```

Authenticating Users

- ***authenticate()***: Takes credentials in the form of keyword arguments:
  - For the default configuration this is username and password
    - Returns a User object if the password is valid for the given username.
    - Returns None if the credentials are invalid.
    - **authenticate(**request=***None,*** **credentials**)**

Authenticating Users

```
from django.contrib.auth import authenticate
 user = authenticate(username='john', password='secret')
 if user is not None: # password verified for the user
       if user.is_active:
           print("User is valid, active and authenticated")
       else:
            print("The credentials are valid, but the account has been disabled!")
else: # unable to verify the username and password
       print(“username and password did not match.")
```

Login

- ***login()*** *function*: used to attach an *authenticated* user to the current session.
  - It takes an HttpRequest object and a User object.
    - Associates the **user** with the current **request** object
    - Ex. login(request, user)
  - Any data set during the anonymous session is retained in the session after a user logs in.
  - ***login()*** function can be called from a view.
- Normally, authenticate() is used before login().

Logout

- Use django.contrib.auth.logout() within your view.
  - It takes an HttpRequest object and has no return value.
  - logout() does not throw any errors if the user wasn't logged in.
  - Cleans out the session data for the current request

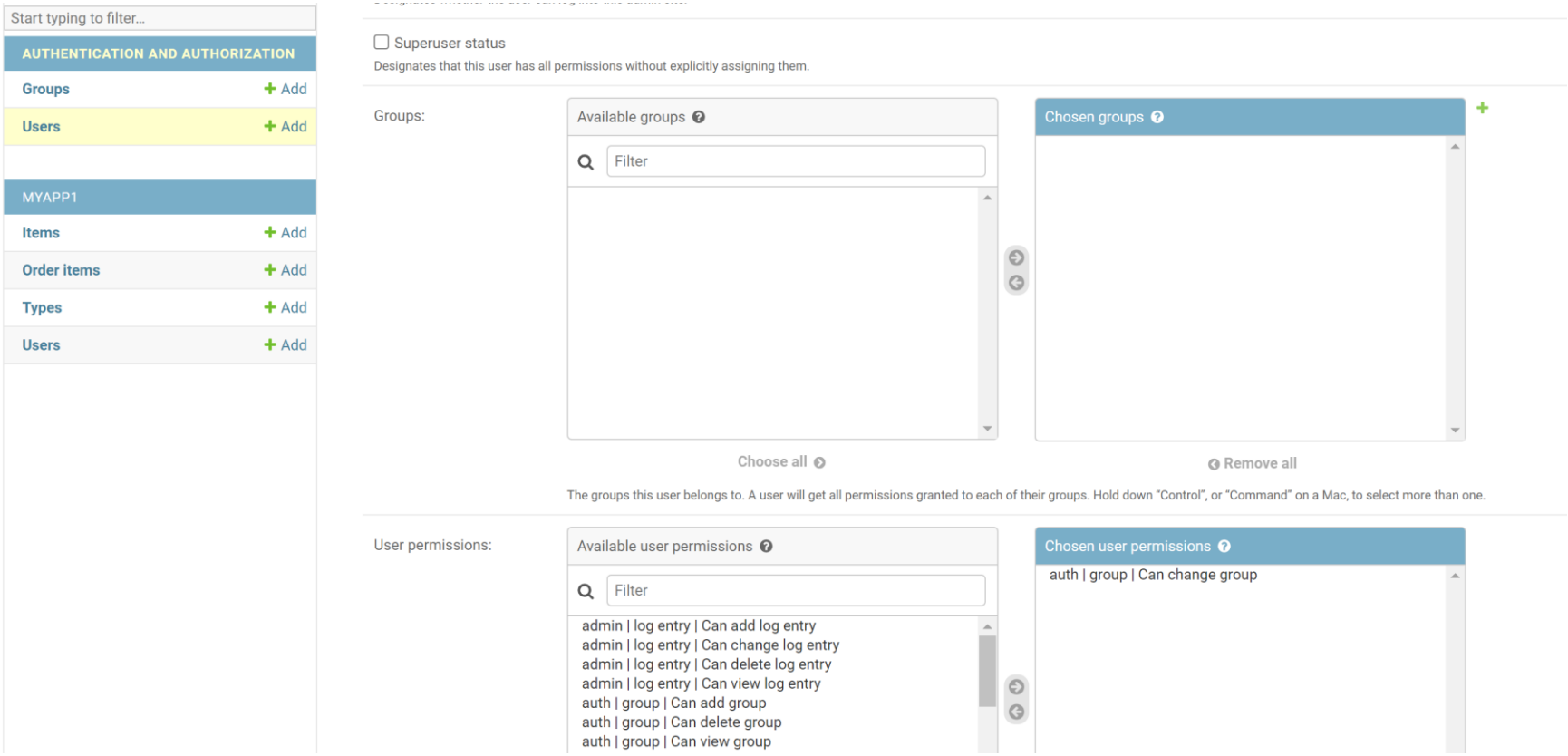
```
from django.contrib.auth import logout

def logout_view(request):
    logout(request)
    # Redirect to a success page.
```

Default Permissions

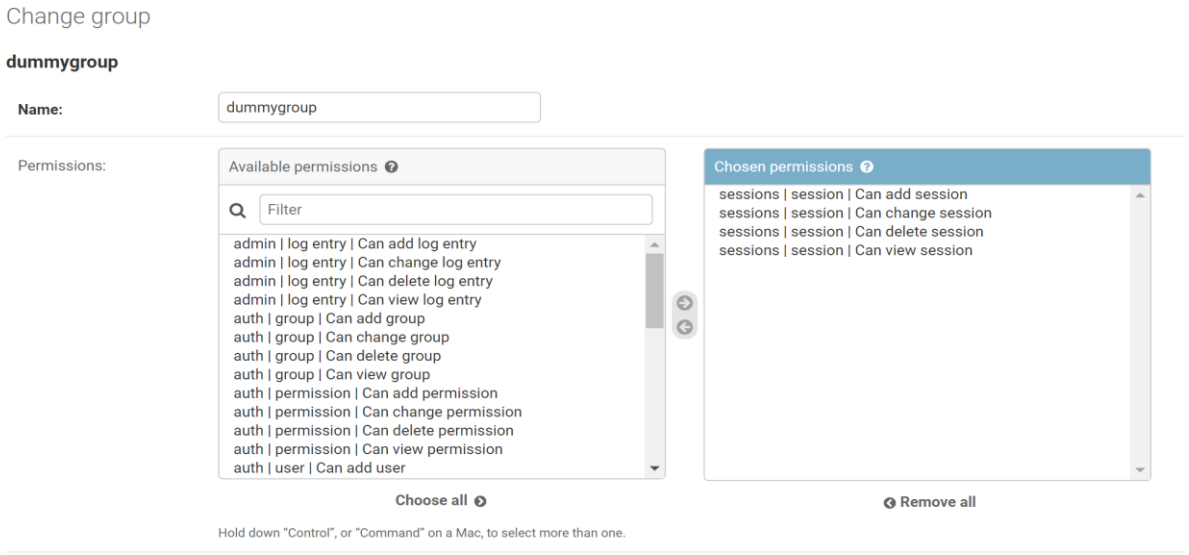
- 4 default permissions created for each Django model defined in one of your installed apps :
  - *Add*: Access to view the "add" form and add an object
    - limited to users with "add" permission for that type of object.
  - *Change*: Access to view the change list, view the "change" form and change objects
    - limited to users with the "change" permission for that type of object.
  - *Delete*: Access to delete objects
    - limited to users with the "delete" permission for that type of object.
  - *View:* Access to view objects

Assign User Permissions

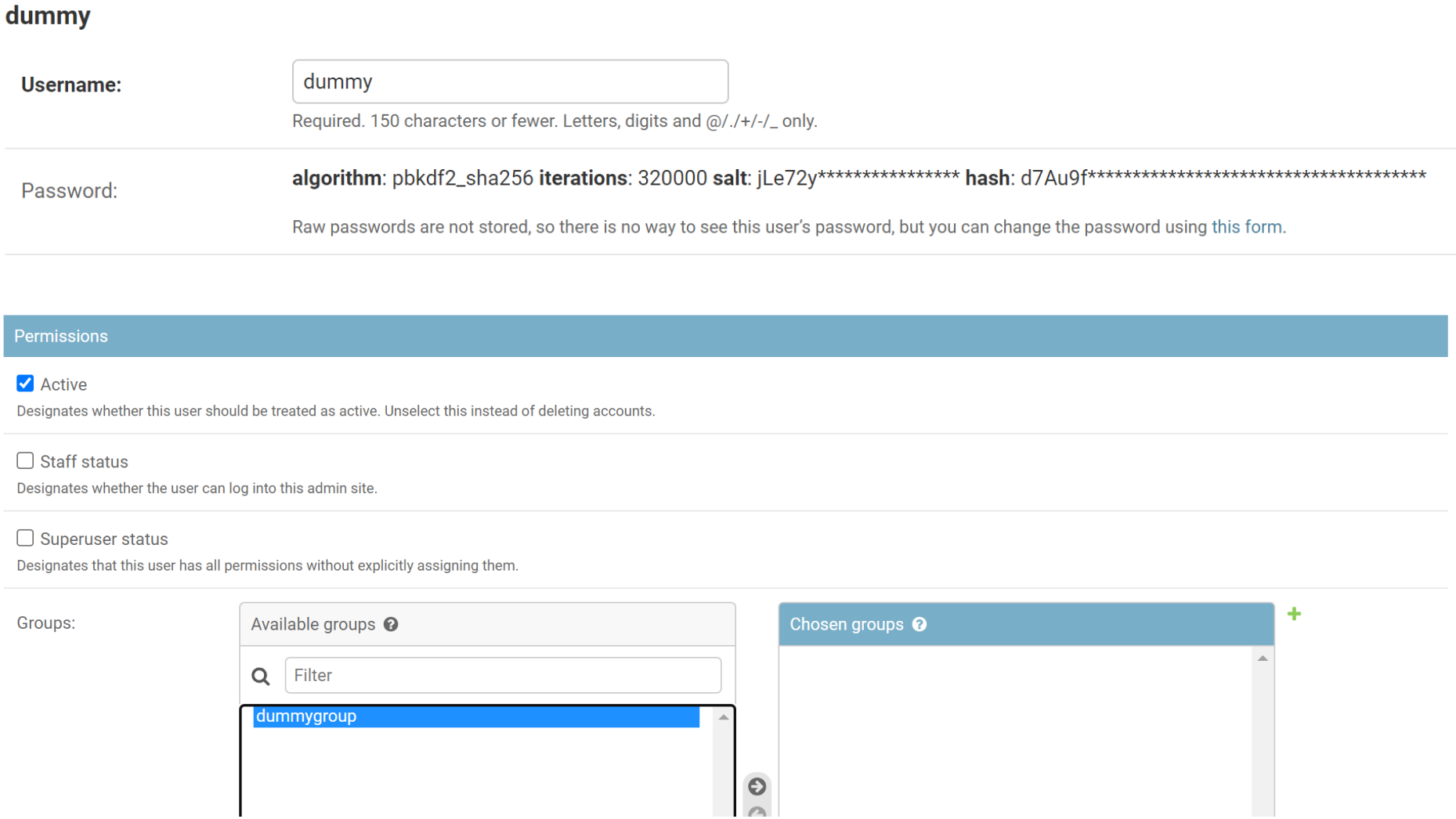


Groups

- Groups allow you to apply permissions to a group of users.
  - A user in a group automatically has the permissions granted to that group.
  - Also a convenient way to categorize users to give them some label, or extended functionality.



Create Users and Assign Them to Groups



References

- https://docs.djangoproject.com/en/4.2/topics/auth/
- https://docs.djangoproject.com/en/4.2/topics/auth/passwords/
- https://docs.djangoproject.com/en/4.2/topics/auth/customizing/
- https://docs.djangoproject.com/en/4.2/ref/contrib/contenttypes/
- https://www.youtube.com/watch?v=eBsc65jTKvw
- https://www.youtube.com/watch?v=dBctY3-Z5hY
- Python Web Development with Django, by J. Forcier et al.
- Slides from Dr. Arunita and Dr. Saja