Lab 14

User Groups & Permissions

To do this exercise, you will need to have your project code from Lab 9.

Step 1: Create two groups in Django Admin

Open the project and run the server, log into Django Admin & create two groups **Customer** & **Manager**

Step 2: Update Accounts App

Copy-paste the code below into **accounts/views.py** and overwrite the existing SignUpView class:

```
from django.shortcuts import render, redirect
from django.contrib.auth.models import Group
from .models import CustomUser

class SignUpView(CreateView):
    form_class = CustomUserCreationForm
    template_name = 'registration/signup.html'

    def post(self, request, *args, **kwargs):
        form = CustomUserCreationForm(request.POST)
        if form.is_valid():
            form.save()
            username = form.cleaned_data.get('username')
            signup_user = CustomUser.objects.get(username=username)
            customer_group = form.cleaned_data.get('group')
            customer_group.user_set.add(signup_user)
            return redirect('login')
        else:
            return render(request, self.template_name, {'form' : form })
```

Open the **accounts/forms.py** file and add in the code shown below. This code will add a new drop-down field to the form so that the new use can select which group they wish to be assigned to. The drop-down field is populated from the **Group** database table.

```
accounts > forms.py > ...
1  from django import forms
2  from django.contrib.auth.forms import UserCreationForm, UserChangeForm
3  from .models import CustomUser
4  from django.contrib.auth.models import Group
5  class CustomUserCreationForm(UserCreationForm):
6      group = forms.ModelChoiceField(queryset=Group.objects.all(), required=True)
7      class Meta(UserCreationForm):
8          model = CustomUser
9          fields = ('username', 'email', 'age',)
```

Step 3: Create a custom TemplateTag

In the **base.html** template we want to show a new menu item for a user that belongs to the **Manager** group. This menu item will allow a manager to add a new book to the database. In the template we need to check if the user belongs to this group. To do this we will create a **template tag**.

In the **accounts** app create a new **folder** called **templatetags** and inside this folder create two files, one called **auth_extras.py** and the 2nd one called **__init__.py**.

```
templatetags
  __init__.py
  auth_extras.py
```

Open the **auth_extras.py** file and add in the following code:

```
accounts > templatetags > auth_extras.py > ...
1  from django import template
2  from django.contrib.auth.models import Group
3
4  register = template.Library()
5
6  @register.filter(name='has_group')
7  def has_group(user, group_name):
8      group = Group.objects.get(name=group_name)
9      return True if group in user.groups.all() else False
```

Line 4: To be a valid tag library, the module must contain a module-level variable named register that is a template.Library instance, in which all the tags and filters are registered.

Line 6: Once you have written your filter definition, you need to register it with your Library instance, to make it available to Django's template language

Lines 7-9: This is the code for the filter that will return true if the user is in the Manager group. Otherwise, False is returned.

Open the base.html file and add in the following code:

```
2  {% load auth_extras %}
```

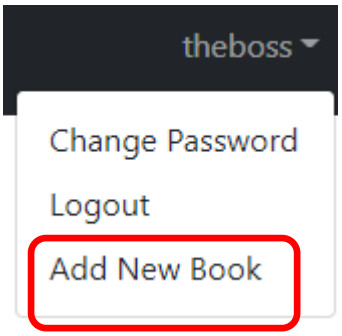
```
45                {% else %}
46                  <a href="{% url 'password_change'%}" class="dropdown-item">Change Password</a>
47                  <a href="{{% url 'logout' %}" class="dropdown-item">Logout</a>
48                {% endif %}
49                {% if user|has_group:"Manager" %}
50                  <a href="" class="dropdown-item">Add New Book</a>
51                {% endif %}
52              </div>
```

Line 2: We load our new template tag

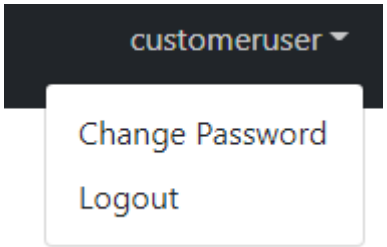
Lines 49-51: We check if the current user belongs to the “**Manager**” group using the custom filter that we wrote above.

Step 4: Try it out

Restart the server, go to the home page, and click the **Sign-Up** button to register a new user of type “**Manager**”. Log in with the new user details and note the new menu item in the drop-down menu:



Click the **Log out** button to log out and click the **Sign-Up** button again to register a new user of type “**Customer**”. Log in with the new user details and note the **Add New Book** menu item is missing from the drop-down menu:



Step 5: Update Books App

We will now update our books app to include a form for the **Create Book** functionality. Open **books/views.py** and add in the following code:

```
from django.views.generic.edit import CreateView
```

```
class BookCreateView(CreateView):
    model = Book
    fields = ('title','author','price','date_publication','cover')
    template_name = 'books/book_new.html'
```

Create a new template called **book_new.html** in the **templates/books** folder and copy-paste the following code. Note the **enctype** that is included in this form because we are uploading an image as part of the new book details.

```
{% extends 'base.html' %}

{% load crispy_forms_tags%}

{% block content %}

<h1>New Book</h1>
<form action="" method="post" enctype="multipart/form-data">{% csrf_token %}
    {{ form|crispy }}
    <button class="btn btn-success ml-2" type="submit">Save</button>
</form>

{% endblock content %}
```

Add the following code to **books/urls.py** & don't forget the comma on line 7:

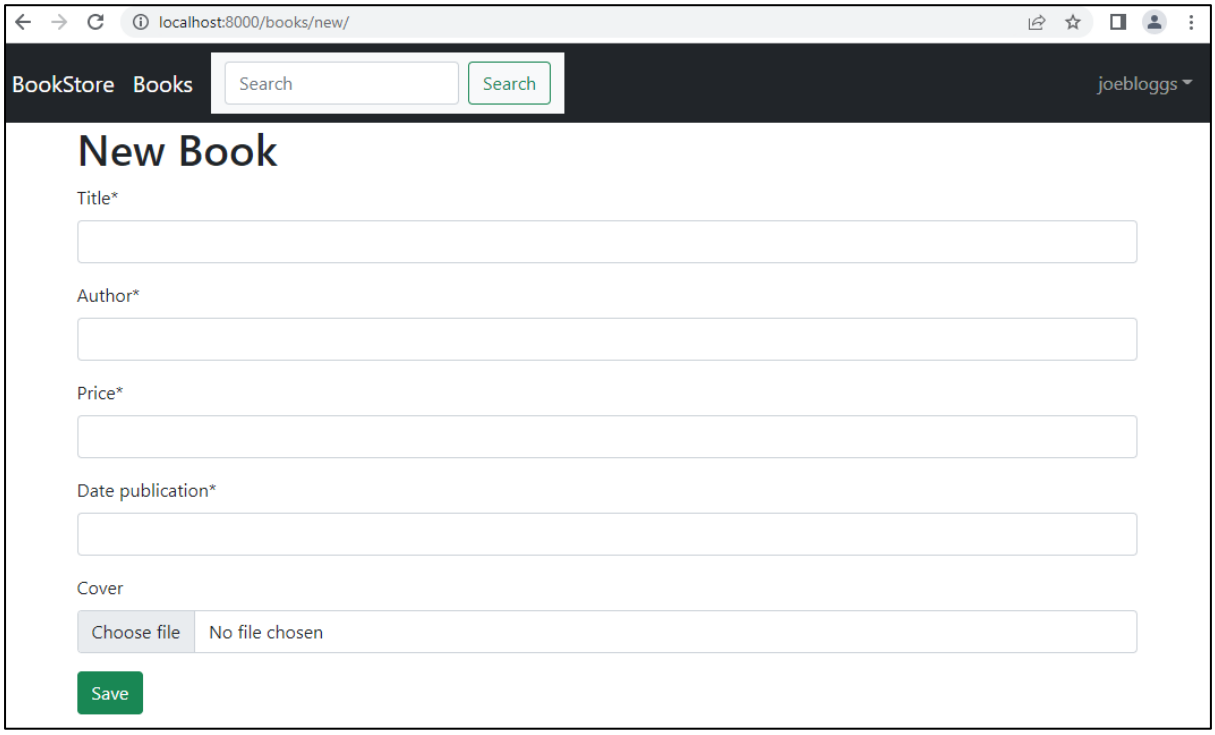
```
books > urls.py > ...
1  from django.urls import path
2  from .views import BookListView, BookDetailView, SearchResultsListView, BookCreateView
3
4  urlpatterns = [
5      path('', BookListView.as_view(), name='book_list'),
6      path('<uuid:pk>', BookDetailView.as_view(), name='book_detail'),
7      path('search/', SearchResultsListView.as_view(), name='search_results'),
8      path('new/', BookCreateView.as_view(), name='book_create'),
9  ]
```

Open **base,html** and add in the following url for the **Create Book** menu item:

```
49                  {% if user|has_group:"Manager" %}
50                    <a href="{% url 'book_create' %}" class="dropdown-item">Add New Book</a>
51                  {% endif %}
```

Step 4: Try it out

Run the server and log in as a **Manager** user type and try to add a new book.

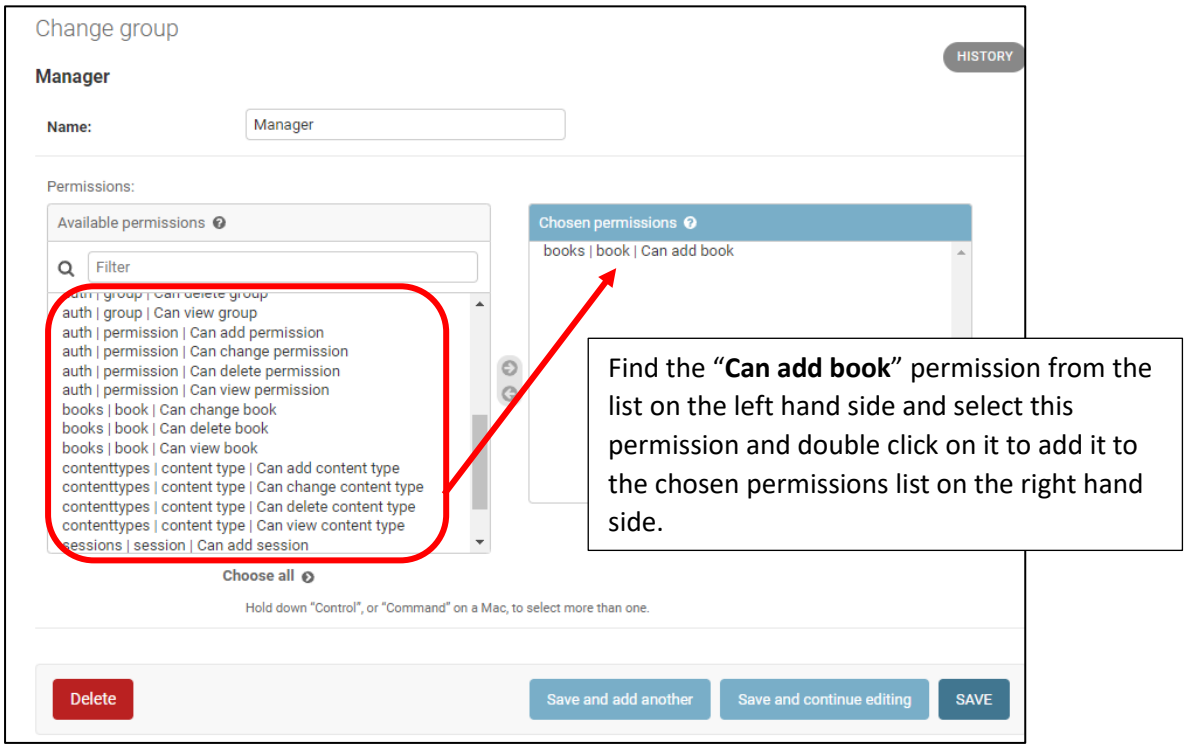


The new book should display on the page after you click the **Save** button.

Step 5: Try to add a new book as a Customer

Log out of the website and log back in as a **Customer** user type. The menu item is not available to you to add a new book but try and access the view using the url: http://localhost:8080/books/new/ and you are taken to the **New Book** page. Fill in the details for a new book and click the Save button. A user in the Customer group can create a book which should not be allowed.

In Django Admin, log in as a superuser and assign this permission to the "**Manager**" group as shown below:

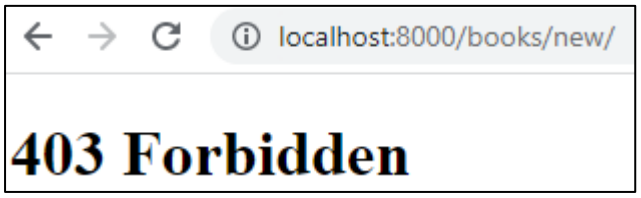


Permissions can be tested in a class-based view using the **PermissionRequiredMixin**. Add the following code to **books/views.py**:

```
from django.contrib.auth.mixins import PermissionRequiredMixin
```

```
class BookCreateView(PermissionRequiredMixin,CreateView):
    permission_required = 'books.add_book'
    model = Book
    fields = ('title','author','price','date_publication','cover')
    template_name = 'books/book_new.html'
```

Save the project, run the server. Log in as a **Manager** and you should be able to add a new book. Log out and log in as a **Customer** and you should see the following:



Stop the server and run the following git commands to update the local and remote repositories:

```
git add -A

git commit -m “lab 9 Groups commit”

git push -u origin main
```