This exercise is a part of the **"Flask Users, Sessions and Authentication"** course on **Pluralsight**, by **Mateo Prigl**.

Admin Interface: Assignments

There is more then one way to implement the administrator interface. We will build on top of the ADMIN Role we defined inside of the Role class.

- Create a new folder for the admin package (which will hold the admin blueprint)
- Create the *admin_required* decorator which will check if the user is admin
- Add this decorator on all of the routes from the admin blueprint
- Create two views (**gigs** and **users**) which will list all of the gigs and users from the site
- Create two POST views for deleting gigs and users (**delete_gig** and **delete_user**)
- Add buttons in some of the templates which you see fit to have this functionality of deleting users and gigs
- Only admins should be able to see admin account pages
- Create a new view for removing musician applications from gigs (**remove_gig_application**)

You can implement this functionality in your current state of the application. It shouldn't have any conflicts with the rest of the course.

Solution

This is one of the possible solutions for implementing the assignments above. This is how I did it, you could have a done it a little bit differently.
(*Source code is also available in the directory next to this pdf*)

Let's create a new blueprint called admin. We will not need any forms here, so I excluded the forms module.
- admin - - /templates - - **init**.py - - views.py

Just like we did before, define the blueprint in the views file and register it to the application. I will add the admin prefix to the routes.

---

app/admin/views.py

```
from flask import Blueprint, render_template, redirect, url_for, request, flash
from app.auth.views import current_user, login_required
from app.models import Gig, Role, User
from app import db

admin = Blueprint('admin', __name__, template_folder='templates')
```

---

app/init.py

```
...
from app.admin.views import admin
...
app.register_blueprint(admin, url_prefix="/admin")
...
```

To start off, let's create the admin required view decorator inside of the auth blueprint. This decorator will redirect the user from the view if he is not an administrator.

---

app/auth/views.py

```
def admin_required(f):
    @wraps(f)
    def _admin_required(*args, **kwargs):
        if not current_user.is_admin():
            flash("You need to be an administrator to access this page", "danger")
            return redirect(url_for("auth.login"))
        return f(*args, **kwargs)
    return _admin_required
```

Now we can import it inside of the admin views module.

---

app/admin/views.py

```
from app.auth.views import admin_required
```

I want this new decorator to decorate all of the views from the admin blueprint. I will use the before_request method from the blueprint class. This method will make sure that anything after it will be executed before each request. So before each request in the admin blueprint, check if the user is authenticated and if the user is an admin. Then I will define some placeholder function, which will just print out a simple message to the terminal.

---

app/admin/views.py

```
@admin.before_request
@login_required
@admin_required
def check_admin_in_each_view():
    # Print some kind of a log
    print("Admin " + current_user.username + " accessed " + request.url)
```

Now we can start creating views and we can be sure that each view in this blueprint can only be accessed by logged in administrators.
First I will create one view to show all of the gigs from the site and another to show all of the users.

---

app/admin/views.py

```
@admin.route("/gigs")
def gigs():
    gigs = Gig.query.all()
    return render_template("all_gigs.html", gigs=gigs)

@admin.route("/users")
def users():
    users = User.query.all()
return render_template("all_users.html", users=users)
```

Let's go to the navigation bar and add the link to these routes, but only if the current user is logged in and he is admin. I he's not an admin, render the my gigs page.
Include this under the account link, inside of the authenticated block.

---

app/templates/base.html

```
{% if current_user.is_admin() %}
<li class="nav-item {{ 'active' if active_page == 'gigs' }}">
<a href="{{ url_for('admin.gigs') }}" class="nav-link">All gigs</a>
</li>
<li class="nav-item {{ 'active' if active_page == 'users' }}">
<a href="{{ url_for('admin.users') }}" class="nav-link">All users</a>
</li>
{% else %}
<li class="nav-item {{ 'active' if active_page == 'my_gigs' }}">
<a href="{{ url_for('gig.my_gigs') }}" class="nav-link">My gigs</a>
</li>
{% endif %}
```

Since the my_gigs page is only available for non-admin users, let's make sure that my_gigs view redirects admins to their own gigs view. This condition needs to be added to the view in the gig views module.
Add this condition at the start of the **my_gigs** view.

---

app/gig/views.py

```
if current_user.is_admin():
    return redirect(url_for("admin.gigs"))
```

Administrator can remove gigs and users, so let's create two new views to allow this. The route will take in the id of the resource we want to remove and then remove it from the database.

app/admin/views.py

```python
@admin.route("/delete-gig/<int:gig_id>", methods=["POST"])
def delete_gig(gig_id):
    gig = Gig.query.get(gig_id)
    if gig:
        flash("The gig \"" + gig.title +  "\" is deleted.", "success")
        db.session.delete(gig)
        db.session.commit()

    if "gig/info" in request.referrer:
        return redirect(url_for("admin.gigs"))
    else:
        return redirect(request.referrer)

@admin.route("/delete-user/<int:user_id>", methods=["POST"])
def delete_user(user_id):
    user = User.query.get(user_id)
    if user:
        flash("The account \"" + user.username +  "\" is deleted.", "success")
        db.session.delete(user)
        db.session.commit()

    if "user/profile" in request.referrer:
        return redirect(url_for("admin.users"))
    else:
        return redirect(request.referrer)
```

Notice that I am checking if the "user/profile" or the "gig/info" are inside of the **request.referrer**. This is because I want to redirect the user to some other page if he is trying to delete the resource from its show page. If for example, an admin decides to delete the gig from the gig show page, then the **request.referrer** will redirect it back to the same show page, but since the gig is now deleted, the server will return 404.

Now we can create the templates for all_users and all_gigs. Nothing special here, just looping through all of the gigs. Each gig will have a form which will allow us to remove it with the delete_gig will created. Same thing goes for the users template, just replace this gig with user and gig id with the user_id. Save the templates inside of the admin templates directory.

app/admin/templates/all_gigs.html

```
{% extends 'base.html' %}
{% block title %}All gigs - {% endblock %}
{% set active_page = 'gigs' %}

{% block content %}
<div class="row">
  <div class="col-lg-10">
      <div class="card card-outline-secondary my-4">
        <div class="card-header">
          All gigs
        </div>
        <div class="card-body">
          {% if gigs %}
          {% for gig in gigs %}
          <h4>
        <a href="{{ url_for('gig.show', slug=gig.slug) }}">{{ gig.title }}</a>
      <h4>
          <small>{{ gig.description }}</small>
          <form method="POST" action="{{ url_for('admin.delete_gig', gig_id=gig.id) }}">
            <input class="btn btn-danger" type="submit"
        style="margin-top:10px" name="submit" value="Delete gig">
          </form>
          <hr class="form-border">
          {% endfor %}
          {% else %}
          Nothing to show
          {% endif %}
        </div>
      </div>
    </div>
  </div>
</div>
{% endblock %}
```

app/admin/templates/all_users.html

```
{% extends 'base.html' %}
{% block title %}All users - {% endblock %}
{% set active_page = 'users' %}

{% block content %}
<div class="row">
  <div class="col-lg-10">
      <div class="card card-outline-secondary my-4">
        <div class="card-header">
          All users
        </div>
        <div class="card-body">
          {% if users %}
          {% for user in users %}
          <h4>
            <a href="{{ url_for('account.show', username=user.username) }}">{{ user.username }}</a>
            <small>
              {% if user.is_role(Role.MUSICIAN) %}
              (musician)
              {% endif %}
              {% if user.is_role(Role.EMPLOYER) %}
              (employer)
              {% endif %}
              {% if user.is_admin() %}
              (admin)
              {% endif %}
            </small>
          <h4>
          <small>{{ user.description }}</small>
          <form method="POST" action="{{ url_for('admin.delete_user', user_id=user.id) }}">
            <input class="btn btn-danger" type="submit"
        style="margin-top:10px" name="submit" value="Delete user">
          </form>
          <hr class="form-border">
          {% endfor %}
          {% endif %}
        </div>
      </div>
    </div>
  </div>
</div>
{% endblock %}
```

Let's also add this delete button to the show account page. Add the form after each gig.

app/account/templates/show_account.html

```
{% if current_user.is_admin() %}
          <form  method="POST" action="{{ url_for('admin.delete_gig', gig_id=gig.id) }}">
            <input type="submit" class="btn btn-danger"
            style="margin-top:10px" value="Delete gig"
        onclick="return confirm('Are you sure that you want to premanently delete this gig?')">
          </form>
{% endif %}
```

While we are still in this template, let's also give the admin ability to remove the account. If the user is the current user, then put the regular update and remove account buttons. However, if this is not a logged in user, and the current_user is an admin, we will render the delete user button. This way admin can remove any account from the site.

app/account/templates/show_account.html

```
{% if user == current_user %}
        <div class="button-actions">
            <a href="{{ url_for('account.edit') }}" class="btn btn-primary">Update account</a>
            <form class="delete-form" method="POST" action="{{ url_for('account.delete') }}">
                {{ deleteAccountForm.submit(class="btn btn-danger",
        onclick="return confirm('Are you sure that you want to premanently delete your account?')") }}
            </form>
            </div>
        {% else %}
          {% if current_user.is_admin() %}
            <div class="button-actions">
              <form class="delete-form" method="POST"
        action="{{ url_for('admin.delete_user', user_id=user.id) }}">
                <input type="submit" class="btn btn-danger"
            value="Delete user"
        onclick="return confirm('Are you sure that you want to premanently delete this account?')">
              </form>
            </div>
           {% endif %}
         {% endif %}
```

The send email button will only be shown if the user is a musician and the logged in user is an employer. However, I also want this button to appear if the user is an admin and this is not his own account. This way an admin can send an email to any user, even to employers.

app/account/templates/show_account.html

```
{% if user.is_role(Role.MUSICIAN) and current_user.is_role(Role.EMPLOYER)
    or current_user.is_role(Role.ADMIN) and current_user != user %}
        <div class="button-actions">
          <a href="mailto:{{ user.email }}" class="btn btn-success" style="color:white">Send email</a>
        </div>
{% endif %}
```

Inside of the show account view I will add a condition which will check if the user account belongs to an admin. If the account you want to see is an administrator account and you are not an administrator, then you won't be able to see it. Only admins can see admin accounts. Don't forget to import abort from flask.

app/account/views.py

```
from flask import abort


# Add this at the start of the show view, after the user declaration
if user.is_admin() and not current_user.is_admin():
    abort(404)
```

Let's also add the delete gig form to the show gig template. If the current user is admin we will render the delete gig form.

app/gig/templates/show_gig.py

```
 {% if current_user.is_admin() %}
        <div class="button-actions">
          <form class="delete-form" method="POST"
        action="{{ url_for('admin.delete_gig', gig_id=gig.id) }}">
            <input type="submit" class="btn btn-danger"
        value="Delete gig"
        onclick="return confirm('Are you sure that you want to premanently delete this gig?')">
          </form>
        </div>
 {% endif %}
```

Now we can refactor the home page for the authenticated admins. Add this code at the end of the *_authenticated.html* partial.

---

app/main/templates/home/_authenticated.html

```
{% if current_user.is_admin() %}
<h1 class="my-4">
  Hello admin "{{ current_user.username }}"!
</h1>
<a href="{{ url_for('admin.gigs') }}" class="nav-link">All gigs >></a>
<a href="{{ url_for('admin.users') }}" class="nav-link">All users >></a>
{% endif %}
```

One last thing to do is to create the view for removing musician application from gigs. I will create a special route for this, remove_gig_applications. It will take in the id of the gig and also the id of the applied user. Then we can remove the application from the musician.

---

app/admin/views.py

```
@admin.route('/remove_apply/<int:gig_id>/<int:musician_id>', methods=["POST"])
def remove_gig_application(gig_id, musician_id):
    gig = Gig.query.get(gig_id)
    musician = User.query.get(musician_id)

    if gig and musician:
        musician.remove_application(gig)
        db.session.commit()

        flash("Application of the user \""
                + musician.username + "\" is removed from the gig \""
                + gig.title + "\"", "success")
        return redirect(request.referrer)

    flash("Something went wrong.", "danger")
    return redirect(request.referrer)
```

Of course, we need to create this remove_application method in the user model class. If the user is applied to the gig, we will remove the gig from the applied_gigs.

---

app/models.py

```
def remove_application(self, gig):
    if self.is_applied_to(gig):
            self.applied_gigs.remove(gig)
            db.session.add(self)
```

Now we can add this remove application button to each musician on the show gig page. Add it under the musicians name. This button will be shown only if the user is admin.

---

app/gig/templates/show_gig.html

```
{% if current_user.is_admin() %}
        <form method="POST" action="{{ url_for('admin.remove_gig_application',
                        gig_id=gig.id, musician_id=musician.id) }}">
          <input class="btn btn-warning" style="color:white" type="submit"
            name="submit" value="Remove application">
        </form>
{% endif %}
```

To test this out you need to create a new admin user directly from the flask shell. Inside of the terminal type **flask shell** and enter the shell.

Inside of the shell create the admin user

```
>> u = User("admin", "admin@mail.com", "password123", "nowhere", "no")
>> db.session.add(u)
>> db.session.commit()
>> exit()
```

After that you can just login with this email and password.

Thanks for watching the course!
Mateo