# Sessions Utilisateur/Admin dans Flask

## 1 Session

La session permet à Flask de se souvenir d’un utilisateur connecté entre les pages. Flask-Login facilite la gestion de la connexion, des rôles (admin/utilisateur), des accès et de la déconnexion.

## 2. Création de session

Quand l’utilisateur envoie ses identifiants via le formulaire de connexion, ils sont vérifiés. Si valides, ma fonction login\_user(user) crée une session. Flask utilize donc un cookie et montre les infos via current\_user.

## Extrait de code : login

@main.route('/login', methods=['POST'])  
def login\_submit():  
 username = request.form.get('username')  
 password = request.form.get('password')  
 user = Utilisateur.authenticate(username, password)  
 if user:  
 login\_user(user)  
 if user.admin:  
 return redirect(url\_for('main.menu'))  
 else:  
 return redirect(url\_for('main.menu\_public'))  
 else:  
 error = 'Identifiants invalides'  
 return render\_template('auth/login.html', error=error)

## 3. Utilisation dans les templates (Jinja2)

Une fois connecté, current\_user permet d’afficher des infos selon le rôle :

## Extrait de template Jinja2

{% if current\_user.is\_authenticated %}  
<div class='mb-2 text-center'>  
 <div class='fw-semibold mb-1'>Connecté en tant que :</div>  
 {% if current\_user.admin %}  
 <span class='badge bg-danger fs-6'>Admin</span>  
 {% else %}  
 <span class='badge bg-primary fs-6'>Utilisateur</span>  
 {% endif %}  
</div>  
{% if current\_user.admin %}  
 <a href='{{ url\_for("main.menu") }}' class='btn btn-warning w-100 mb-2'>Menu Admin</a>  
{% endif %}  
<a href='{{ url\_for("auth.logout") }}' class='btn btn-outline-danger w-100'>Déconnexion</a>  
{% else %}  
 <!-- Formulaire de connexion ici -->  
{% endif %}

## 4. Actions

- Connexion : via login\_user(user)  
- Vérifier si connecté : via current\_user.is\_authenticated  
- Vérifier le rôle : via current\_user.admin  
- Déconnexion : via logout\_user()

## 5. Modèle Utilisateur (exemple)

class Utilisateur(UserMixin):  
 def \_\_init\_\_(self, id, username, admin=False):  
 self.id = id  
 self.username = username  
 self.admin = admin  
  
 @staticmethod  
 def authenticate(username, password):  
 user = Utilisateur.query.filter\_by(username=username).first()  
 if user and check\_password\_hash(user.mdpuser, password):  
 return user  
 return None