



Travaux Pratiques n°6

(Login/Logout/inscription)

I. Login/Logout

Dans ce TP, nous allons vous montrer comment autoriser les utilisateurs à se connecter à votre site avec leurs propres comptes, et comment contrôler ce qu'ils peuvent faire et voir en fonction de leur connexion ou non et de leurs *autorisations*.

1. Django admet une application '**auth**' intégrée qui permet de gérer les authentifications des utilisateurs. Il suffit d'ajouter le path correspondant au fichier **urls.py** du projet.

```
from django.contrib import admin, auth
from django.contrib.auth import views as auth_views

urlpatterns = [
    path('admin/', admin.site.urls),
    path('magasin/', include('magasin.urls')),
    path('', views.home, name='home'),
    .....
```

```
path('login/', auth_views.LoginView.as_view(), name='login'),
path('logout/', auth_views.LogoutView.as_view(), name='logout'),
```

```
path('password_change/', auth_views.PasswordChangeView.as_view(),
name='password_change'),
path('password_change/done/', auth_views.PasswordChangeDoneView.as_view(),
name='password_change_done'),
```

```

path('password_reset/',auth_views.PasswordResetView.as_view(),
name='password_reset'),

path('password_reset/done/',auth_views.PasswordResetDoneView.as_view(),
name='password_reset_done'),

path('reset/<uidb64>/<token>/',auth_views.PasswordResetConfirmView.as_view(),
name='password_reset_confirm'),

path('reset/done/',auth_views.PasswordResetCompleteView.as_view(),
name='password_reset_complete'),

]+ static(settings.MEDIA_URL, document_root= settings.MEDIA_ROOT)

```

Django étant open-source, on peut voir la liste des chemins liés aux vues du système d'authentification par défaut dans le code source, répertorié dans [GitHub](#). Les **templates** liés à ces vues et chemins seront personnalisés et placés dans un répertoire dédié.

2. Nous commençons par ajouter les liens de connexion/deconnexion/inscription dans le fichier base.html. Ajouter le code suivant dans votre navbar ainsi :

```

<ul class="navbar-nav ml-auto">
    {% if user.is_authenticated %}
    <li class="nav-item active">
        <a class="nav-link" href="#">{{user}},</a>
    </li>
    <li class="nav-item active">
        <a class="nav-link" href="#">Déconnexion</a>
    </li>
    {% else %}
    <li class="nav-item active">
        <a class="nav-link" href="#">Connexion</a>
    </li>
    <li class="nav-item active">
        <a class="nav-link" href="#">Inscription</a>
    </li>
    {% endif %}
</ul>

```

3. Il faut aussi spécifier l'url de connexion de notre choix et la redirection vers quel url après connexion dans le fichier de configuration setting.py comme ceci :

```
LOGIN_URL = 'login'
LOGIN_REDIRECT_URL = 'home'
```

4. Dans le répertoire `templates`, il faudra donc créer un second répertoire que nous nommerons `registration` et dans lequel seront ajoutés les différents templates liés aux vues du système d'authentification.

Nous créons dans `registration` deux fichiers `login.html` et `logout.html`

Login.html

```
{% extends "magasin/base.html" %}
{% block content %}
<h2>Se connecter </h2>
<div>
    <form method="post">
        {% csrf_token %}
        {% if form.non_field_errors %}
            {% for error in form.non_field_errors %}
                <p class="alert alert-danger">{{ error }}</p>
            {% endfor %}
        {% endif %}
        <table>
            {% for field in form %}
                <tr>
                    <td>{{ field.label_tag }}</td>
                    <td>{{ field }}</td>
                </tr>

            {% endfor %}
        </table>
```

```

        <button type="submit">se connecter</button>
    </form>
</div>

<div>
    <small>
        Pas encore inscrit.e?
        <a href="#">
            S'inscrire
        </a>
    <br>
        Mot de passe oublié ?
        <a href="#">
            Renouveler mon mot de passe
        </a>
    </small>
</div>
{% endblock %}

```

Logout.html

```

{% extends "magasin/base.html" %}
{% block content %}
<h2>Vous êtes déconnecté. </h2>
    <a href="{% url 'login' %}">Se reconnecter</a>
{% endblock %}

```

5. Gérer les permissions

Dans cette dernière section, voyons comment on peut gérer la sécurité des accès aux données. Nous allons apporter les modifications nécessaires à notre vue `home()` pour qu'elle traite que la personne authentifiée. Cela veut dire qu'il faut être connecté en premier lieu pour accéder à la page d'accueil. Pour ce faire, on importe ce qu'on appelle un décorateur ("decorator" en

anglais), comme cela est montré en ligne 1, puis on l'inscrit juste avant la fonction à modifier (ligne 2).

```
from django.contrib.auth.decorators import login_required
@login_required
def home(request):
    context={'val':"Menu Accueil"}
    return render(request, 'home.html', context)
```

Dans le cas présent nous utiliserons un décorateur d'authentification `login_required` qui va sécuriser et restreindre l'accès à la page d'accueil selon qu'un utilisateur est authentifié ou non. Ainsi, si un utilisateur tente d'accéder à la page d'accueil pour la première fois, il sera automatiquement redirigé vers la page de connexion, dont l'url `LOGIN_URL` est définie dans le fichier de configuration `settings.py`.

II. L'inscription

l'inscription d'un utilisateur ne fait pas partie du système d'authentification par défaut de Django. Nous pouvons cependant créer notre page d'inscription à l'aide du formulaire `personnalisé`.

1. Définir un formulaire personnalisé de création d'un nouvel utilisateur

Pour avoir un formulaire d'inscription personnalisé, il nous faut ajouter dans le fichier `forms.py` de l'application `magasin` un nouveau formulaire « `UserRegistrationForm` » qui hérite du formulaire `UserCreationForm` qui est formulaire, fourni par Django et facilite l'inscription d'un nouvel utilisateur. comme ceci :

```
from django import forms
```

```

from django.contrib.auth.models import User
from django.contrib.auth.forms import UserCreationForm

class UserRegistrationForm(UserCreationForm):
    first_name = forms.CharField(label='Prénom')
    last_name = forms.CharField(label='Nom')
    email = forms.EmailField(label='Adresse e-mail')

    class Meta(UserCreationForm.Meta):
        model = User
        fields = UserCreationForm.Meta.fields + ('first_name', 'last_name', 'email')

```

La vue qui gère l'inscription d'un nouvel utilisateur s'appelle `register()` à définir dans le fichier `views.py` de `magasin`.

2. Définir la vue de la page d'inscription Django

Dans le fichier `views.py` de l'application `magasin`, nous ajoutons la vue qui gère l'inscription d'un nouvel utilisateur s'appelle `register()` comme suit :

```

from .forms import ProduitForm, FournisseurForm, UserRegistrationForm

from django.contrib.auth import login, authenticate

from django.contrib import messages

def register(request):
    if request.method == 'POST' :
        form = UserCreationForm(request.POST)

        if form.is_valid():
            form.save()

            username = form.cleaned_data.get('username')

```

```

        password = form.cleaned_data.get('password1')

        user = authenticate(username=username, password=password)

        if user is not None :

            login(request,user)

            messages.success(request, f'Coucou {username}, Votre compte
a été créé avec succès !')

        else :

            return redirect('home')

    else :

        form = UserCreationForm()

    return render(request,'registration/register.html',{'form' : form})

```

3. Créer l'url correspondant pour l'inscription dans le fichier urls.py de l'application magasin ainsi :

```

urlpatterns=[

    ....

    path('register/',views.register, name = 'register'), ]

]

```

4. Construire le template de la page d'inscription

Définissons ensuite le template `register.html` sous le dossier **registration** :

```

{% extends "base.html" %}

{% block content %}

```

```

<h2>S'inscrire</h2>

<form method="post">

    {% csrf_token %}

    <table>

        {% for field in form %}

            <tr>

                <td>{{ field.label_tag }}</td>

                <td>{{ field }}</td>

            </tr>

            {% for error in field.errors %}

                <tr><td></td><td class="alert alert-danger">{{ error }}</td></tr>

            {% endfor %}

        {% endfor %}

    </table>

    <button type="submit">s'inscrire</button>

</form>

<div>

    <small>

        Déjà inscrit.e?

        <a href="{% url 'login' %}">Se connecter</a>

    </small>

</div>

{% endblock %}

```


5. Modifier le template de la page connexion login.html et aussi le lien inscription dans le navbar dans base.html

```
<div>
  <small>
    Pas encore inscrit.e?
    <a href="{% url 'register' %}"> S'inscrire  </a>
  ...
```