



## Présentation ADRAR Pôle Numérique

Suivez-nous... LinkedIn  et Twitter  @Adrar\_Numerique



# ADRAR PÔLE NUMERIQUE

PÔLE NUMERIQUE DU CENTRE DE FORMATION ADRAR

- > SUPPORT, ADMINISTRATION SYSTEMES & RESEAUX
- > DEVELOPPEMENT D'APPLICATIONS WEB & MOBILES
- > WEBDESIGN & DEVELOPPEMENT INTERFACES
- > TRANSFORMATION NUMERIQUE DES ENTREPRISES

<http://www.adrar-numerique.com>

Suivez-nous sur LinkedIn  Twitter  @Adrar\_Numerique

... et sur le web : [www.adrar-numerique.com](http://www.adrar-numerique.com)



Suivez-nous sur LinkedIn  Twitter  @Adrar\_Numerique

... et sur le web : [www.adrar-numerique.com](http://www.adrar-numerique.com)

# GESTION DES UTILISATEURS

Suivez-nous sur LinkedIn Twitter @Adrar\_Numerique

... et sur le web : [www.adrar-numerique.com](http://www.adrar-numerique.com)

Nous avons vus auparavant comment créer un utilisateur administrateur afin de manager notre site via la page d'administration. Nous avons pu faire ceci car django met en place automatiquement dans la base de donnée de votre projet une table pour les utilisateurs. Nous allons voir comment utiliser cette table afin de permettre aux visiteurs de notre site de créer un compte et de s'authentifier. Nous restreindrons ensuite la possibilité de voter aux utilisateurs enregistrés. Pour se faire nous allons tout d'abord rajouter sur notre page d'index les boutons *inscription*, *connexion* et *déconnexion* (en créant également les urls, vues et templates associés).

```
{% block contenue %}
    <a href="{% url 'sondage:login' %}">connexion</a>
    <a href="{% url 'sondage:register' %}">inscription</a>
    <a href="{% url 'sondage:logout' %}">déconnexion</a>
{% endblock %}
```

```
path('login/', views.my_login, name='login'),
path('register/', views.register, name='register'),
path('logout/', views.my_logout, name='logout'),
```

```
def my_login(request):
    return render(request, 'sondage/login.html')

def register(request):
    return render(request, 'sondage/register.html')

def my_logout(request):
    logout(request)
    return render(request, 'sondage/logout.html')
```



Suivez-nous sur LinkedIn Twitter @Adrar\_Numerique

... et sur le web : [www.adrar-numerique.com](http://www.adrar-numerique.com)

Pour le moments nos vues redirigent juste vers les templates de nos différentes pages. La page de connexion sera un formulaire permettant d'indiquer ses identifiants de connexion et redirigeant vers une page de bienvenue. La page d'inscription sera également un formulaire permettant de s'enregistrer et dont le bouton redirige vers de résumé des informations de connexion. La page de déconnexion afficher juste un message et disposera d'un liens vers notre index.

```
<form id="login" method="post" action="{% url 'sondage:welcome' %}">
  {% csrf_token %}
  <table class='table table-striped table-sm'>
    <tr>
      <td>
        <label for="username">Login</label>
        <input id="username" name="username">
      </td>
      <td>
        <label for="password">Mot de passe</label>
        <input id="password" name="password">
      </td>
    </tr>
    <tr>
      <td>
        <input type="submit" value="Connexion">
        <input type="hidden" name="next" value="{next}">
      </td>
    </tr>
  </table>
</form>

<form id="register" action="{% url 'sondage:registered' %}" method="post">
  {% csrf_token %}
  <table class='table table-striped table-sm'>
    <tr>
      <td>
        <label for="user_name">Nom</label>
        <input id="user_name" name="user_name">
      </td>
    </tr>
    <tr>
      <td>
        <label for="user_firstname">Prénom</label>
        <input id="user_firstname" name="user_firstname">
      </td>
    </tr>
    <tr>
      <td>
        <label for="user_email">Email</label>
        <input id="user_email" name="user_email">
      </td>
    </tr>
    <tr>
      <td>
        <label for="user_pwd">Mot de passe</label>
        <input id="user_pwd" name="user_pwd">
      </td>
    </tr>
    <tr>
      <td>
        <input type="submit" value="Enregistrer">
      </td>
    </tr>
  </table>
</form>

<p>A bientot !</p>
<a href="{% url 'sondage:index' %}">index</a>
```

Suivez-nous sur LinkedIn Twitter @Adrar\_Numerique

... et sur le web : [www.adrar-numerique.com](http://www.adrar-numerique.com)

Intéressons nous d'abord à la partie inscription. Créons la vue *registered*, vers laquelle nous redirigeons notre formulaire d'enregistrement, ainsi que son url et son template. Pour interagir avec la table d'utilisateurs de django nous aurons besoin d'importer le model *User*.

```
from django.contrib.auth.models import User
```

```
def registered(request):
    name = request.POST['user_name']
    firstname = request.POST['user_firstname']
    pwd = request.POST['user_pwd']
    email = request.POST['user_email']
    username = firstname[0].lower() + "." + name.lower()
    user = User.objects.create_user(username, email, pwd)
    user.last_name = name
    user.first_name = firstname
    user.save()
    context = {'user':user}
    return render(request, 'sondage/registered.html', context)
```

Ici nous récupérons donc les éléments de l'attribut *POST* de la requête, nous créons les éléments de notre utilisateur et nous utilisons la méthode *create\_user* du model pour créer l'utilisateur. Nous n'oublions pas de sauvegarder notre utilisateur dans la base de données.

Nous chargeons ensuite le template *registered.html* avec dans le contexte notre utilisateur.

```
<h1>Bienvenue !</h1>
<p>Votre nom d'utilisateur est {{user.username}}</p>

<a href="{% url 'sondage:login' %}">Login</a>
```

Suivez-nous sur LinkedIn Twitter @Adrar\_Numerique

... et sur le web : [www.adrar-numerique.com](http://www.adrar-numerique.com)

Voyons maintenant la connexion d'un utilisateur. Pour ça nous avons créé un formulaire afin d'authentifier notre visiteur et qui redirige vers la vue *welcome*. Rédigeons donc cette vue, ajoutons son url et créons son template. Nous allons également créer une page dans le cas où l'authentification s'est mal passé. Pour authentifier notre utilisateur nous aurons besoin d'importer la méthode *login* ainsi que la méthode *authenticate*.

```
from django.contrib.auth import authenticate, login,
```

```
def welcome(request):
    username = request.POST['username']
    password = request.POST['password']
    user = authenticate(request, username=username, password=password)
    context = {'user': user}
    if user is not None:
        login(request, user)
        return render(request, 'sondage/welcome.html', context)
    else:
        return render(request, 'sondage/error_log.html')
```

```
path('welcome/', views.welcome, name='welcome'),
```

Encore une fois nous récupérons les informations présentes dans l'attribut *POST* de la requête. Deux possibilité ici, soit la méthode à retourné un objet utilisateur et nous pouvons le connecter sur le site et afficher la page *welcome.html*, soit elle n'a rien retourné et nous affichons la page d'erreur.

```
<h1>Bienvenue {{user.username}} !</h1>
<p>Connexion réussi</p>
<a href="{% url 'sondage:index' %}">Page d'accueil</a>
```

```
<p>Identifiant ou mot de passe incorrecte</p>
<a href="{% url 'sondage:login' %}">Login</a>
```

Suivez-nous sur LinkedIn Twitter @Adrar\_Numerique

... et sur le web : [www.adrar-numerique.com](http://www.adrar-numerique.com)

Enfin, voyons notre fonctionnalité de déconnexion en modifiant notre vue *my\_logout*. Nous allons rajouter en plus de la redirection vers notre page *logout.html* un appel à la méthode *logout* de django, que nous devons importer au préalable.

```
from django.contrib.auth import authenticate, login, logout
```

```
def my_logout(request):
    logout(request)
    return render(request, 'sondage/logout.html')
```

```
<p>A bientot !</p>
<a href="{% url 'sondage:index' %}">index</a>
```

Voyez qu'ici c'est sur notre objet *request* que nous appliquons la méthode *logout*. C'est parce-que cet objet contient toutes les informations de la requête http et donc également la connexion de l'utilisateur.



Suivez-nous sur LinkedIn  Twitter  @Adrar\_Numerique

... et sur le web : [www.adrar-numerique.com](http://www.adrar-numerique.com)

Avec ces 3 fonctionnalités nous avons géré l'inscription, la connexion et la déconnexion d'utilisateurs. Nous allons maintenant faire en sorte que certaines fonctionnalités de notre site ne soient accessible qu'aux utilisateurs authentifiés. Par exemple nous ne voulons pas qu'un visiteur non connecté puisse cliquer sur le lien de déconnexion. Cela va se faire via l'utilisation de gabarits.

```
{% block contenue %}
  {% if not request.user.is_authenticated %}
    <a href="{% url 'sondage:login' %}">connexion</a>
    <a href="{% url 'sondage:register' %}">inscription</a>
  {% else %}
    <a href="{% url 'sondage:logout' %}">déconnexion</a>
  {% endif %}
```

Via notre objet *request* nous pouvons vérifier si le visiteur de la page est un utilisateur authentifié ou non. Cela simplifie grandement les choses pour gérer nos pages. Nous affichons donc les boutons de connexion et d'inscription uniquement pour un utilisateur authentifié. Nous pouvons procéder de la même manière pour notre page de détail.

Suivez-nous sur LinkedIn Twitter @Adrar\_Numerique

... et sur le web : [www.adrar-numerique.com](http://www.adrar-numerique.com)

```
{% if request.user.is_authenticated %}
    <form action="{% url 'sondage:vote' question.id %}" method="post">
    {% csrf_token %}
    <fieldset>
        <legend><h1>{{ question.question_text }}</h1></legend>
        {% if error_message %}<p><strong>{{ error_message }}</strong></p>{% endif %}
        {% for choice in question.choice_set.all %}
            <input type="radio" name="choice" id="choice{{ forloop.counter }}" value="{{ choice.id }}">
            <label for="choice{{ forloop.counter }}">{{ choice.choice_text }}</label><br>
        {% endfor %}
    </fieldset>
    <input type="submit" value="Vote">
</form>
{% else %}
    <a href="{% url 'sondage:login' %}"><i>Vous devez être connecté pour pouvoir voter</i></a>
{% endif %}
```

Le formulaire ne s'affiche que si l'utilisateur est authentifié. Sinon la page affiche un lien vers la page de connexion

Suivez-nous sur LinkedIn  Twitter  @Adrar\_Numerique

... et sur le web : [www.adrar-numerique.com](http://www.adrar-numerique.com)

Nous allons maintenant voir comment ajouter des champs à notre profil d'utilisateurs. Pour être plus précis nous n'allons pas modifier le modèle *User* de base, nous allons créer un modèle contenant les champs supplémentaires voulus et nous allons lier ce modèle au modèle de base par une relation 1 pour 1. Nous pourrions par la suite, au même titre que pour une relation 1 pour N ou N pour N, accéder aux champs de notre nouveau modèle par notre objet issu du modèle *User*.

```
class Citizen(models.Model):
    user = models.OneToOneField(User, on_delete=models.CASCADE)
    electeur = models.BooleanField(default=False)
```

```
def registered(request):
    name = request.POST['user_name']
    firstname = request.POST['user_firstname']
    pwd = request.POST['user_pwd']
    email = request.POST['user_email']
    username = firstname[0].lower() + "." + name.lower()
    user = User.objects.create_user(username, email, pwd)
    citizen = Citizen(user=user)
    user.last_name = name
    user.first_name = firstname
    user.save()
    citizen.save()
    context = {'user': user}
    return render(request, 'sondage/registered.html', context)
```

Nous avons donc notre nouveau model *Citizen* qui a une relation 1 pour 1 (*OneToOneField*) avec le modèle *User*. Nous définissons un attribut *electeur*, qui est un booléen à *False* par défaut permettant de dire si l'utilisateur a le droit de voter ou non.

Nous ajoutons ensuite à notre vue de création d'utilisateurs l'enregistrement de l'utilisateur en tant qu'instance de la classe *Citizen* en lui passant en paramètre de construction l'objet *User* créé.

Nous n'oublions pas de sauvegarder l'enregistrement de l'utilisateur.

Suivez-nous sur LinkedIn Twitter @Adrar\_Numerique

... et sur le web : [www.adrar-numerique.com](http://www.adrar-numerique.com)

Avec ces modifications, chaque visiteur qui se crée un compte est automatiquement enregistré en tant que *citizen* mais avec le booléen *electeur* à *False*. Il nous faut donc donner la possibilité à l'utilisateur de devenir citoyen. Pour cela nous allons rajouter une page *profil* affichant l'username de l'utilisateur et lui proposant un lien pour devenir citoyen si il ne l'est pas.

```
path('profil/', views.profil, name='profil'),

def profil(request):
    citizen = Citizen.objects.get(user=request.user)
    context = {'citizen': citizen}
    return render(request, 'sondage/profil.html', context)
```

```
{% extends 'sondage/index.html' %}
{% load static %}
{% block contenu %}
    <h1>{{request.user.username}}</h1>
    {% if citizen.electeur == False %}
        <a href="{% url 'sondage:become' %}">Devenir electeur</a>
    {% endif %}
{% endblock %}
```

Pour accéder à notre objet *Citizen* nous pouvons tout d'abord le récupérer avec une méthode *get*, en lui passant en paramètre de filtrage l'user de la requête, puis l'envoyer à notre page par le contexte.

Il sera donc ensuite utilisable par nos gabarits.

Voyons maintenant la vue *become* permettant de gérer le passage d'un utilisateur vers un compte autorisé à voter.



Suivez-nous sur LinkedIn Twitter @Adrar\_Numerique

... et sur le web : [www.adrar-numerique.com](http://www.adrar-numerique.com)

```
path('profil/become/', views.become, name='become')
```

```
def become(request):
    request.user.citizen.electeur = True
    request.user.citizen.save()
    return IndexView.as_view()(request)
```

Ici nous pouvons voir une autre méthode pour accéder à notre objet *Citizen* en passant directement par l'objet *User*. Etant donnée qu'il y a une relation 1 pour 1 entre *User* et *Citizen* nous pouvons accéder à l'attribut *electeur* de notre objet *Citizen* par l'utilisateur stocké dans la requête via *request.user.citizen.electeur*. Encore une fois nous sauvegardons la modifications.

Ici nous voyons aussi comment depuis une vue appeler une autre vue. Il suffit de return la fonction de cette vue en lui passant le paramètre *request*. Etant donné que nous utilisons une classe de vues la syntaxe est légèrement différentes d'un appel de fonction classique. Si nous avons gardé notre vue *index* (décrite dans la fonction *def index(request)*) nous aurions simplement indiqué en retour de fonction :

```
return index(request)
```

Nous pouvons maintenant modifier notre page *detail.html* pour qu'elle n'autorise le vote que pour des citoyens

Suivez-nous sur LinkedIn Twitter @Adrar\_Numerique

... et sur le web : [www.adrar-numerique.com](http://www.adrar-numerique.com)

```
{% if request.user.is_authenticated %}
    {% if request.user.citizen.electeur %}
        <form action="{% url 'sondage:vote' question.id %}" method="post">
            {% csrf_token %}
            <fieldset>
                <legend><h1>{{ question.question_text }}</h1></legend>
                {% if error_message %}<p><strong>{{ error_message }}</strong></p>{% endif %}
                {% for choice in question.choice_set.all %}
                    <input type="radio" name="choice" id="choice{{ forloop.counter }}" value="{{ choice.id }}">
                    <label for="choice{{ forloop.counter }}">{{ choice.choice_text }}</label><br>
                {% endfor %}
            </fieldset>
            <input type="submit" value="Vote">
        </form>
    {% else %}
        <i>Seul les électeurs peuvent voter</i>
    {% endif %}
{% else %}
    <a href="{% url 'sondage:login' %}"><i>Vous devez être connecté pour pouvoir voter</i></a>
{% endif %}
```