



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



Suivez-nous sur LinkedIn  Twitter  @Adrar_Numerique

... et sur le web : www.adrar-numerique.com

VUES GÉNÉRIQUES

Suivez-nous sur LinkedIn  Twitter  @Adrar_Numerique

... et sur le web : www.adrar-numerique.com

Nous avons mis en places différentes vues que nous avons configurés nous même, une vue d'index permettant d'afficher nos sondages, une vue de détails de nos question et une vue de formulaire pour voter à nos sondages.

Django fournit une série de modèles de vues pour ces pages génériques. Nous avons besoin d'importer *generic* de *django.views* pour pouvoir utiliser nos modèles de vues.

```
def index(request):  
    latest_question_list = Question.objects.order_by('-pub_date')[:5]  
    context = {'question_list': latest_question_list}  
    return render(request, 'sondage/index.html', context)
```

```
def detail(request, question_id):  
    question = get_object_or_404(Question, pk=question_id)  
    return render(request, 'sondage/detail.html', {'question': question})
```

```
def results(request, question_id):  
    question = get_object_or_404(Question, pk=question_id)  
    return render(request, 'sondage/results.html', {'question': question})
```

```
from django.views import generic
```

```
class IndexView(generic.ListView):  
    template_name = 'sondage/index.html'  
    context_object_name = 'question_list'  
  
    def get_queryset(self):  
        return Question.objects.order_by('-pub_date')[:5]
```

```
class DetailView(generic.DetailView):  
    model = Question  
    template_name = 'sondage/detail.html'
```

```
class ResultsView(generic.DetailView):  
    model = Question  
    template_name = 'sondage/results.html'
```


Suivez-nous sur LinkedIn  Twitter  @Adrar_Numerique

... et sur le web : www.adrar-numerique.com

```
class IndexView(generic.ListView):  
    template_name = 'sondage/index.html'  
    context_object_name = 'question_list'  
  
    def get_queryset(self):  
        return Question.objects.order_by('-pub_date')[:5]
```

Nous utilisons ici la classe *generic.ListView* pour notre index afin de générer une vue affichant une liste d'objets.

Nous redéfinissons pour celle-ci deux variables : *template_name* et *context_object_name* pour indiquer le nom du template de la page ainsi que le nom de variable utilisé dans notre page pour récupérer les objets. Nous redéfinissons également une méthode *get_queryset()* afin de récupérer nos éléments à afficher.

Suivez-nous sur LinkedIn  Twitter  @Adrar_Numerique

... et sur le web : www.adrar-numerique.com

```
class DetailView(generic.DetailView):  
    model = Question  
    template_name = 'sondage/detail.html'
```

```
class ResultsView(generic.DetailView):  
    model = Question  
    template_name = 'sondage/results.html'
```

La classe *generic.DetailView* est une classe affichant une page détaillée d'un objet. Elle a besoin pour ça du nom de la classe de modèle, du nom du template ainsi que du contexte. Par défaut pour un modèle nommé *Question* django stock les données dans une variable nommée *question*, nous n'avons pas besoin de le modifier.

Suivez-nous sur LinkedIn  Twitter  @Adrar_Numerique

... et sur le web : www.adrar-numerique.com

Pour pouvoir utiliser nos classes modèles nous avons besoin de faire correspondre les informations dans notre fichier d'url.

```
path('', views.index, name='index'),
path('<int:question_id>/', views.detail, name='detail'),
path('<int:question_id>/results/', views.results, name='results'),
path('<int:question_id>/vote/', views.vote, name='vote'),
```

```
path('', views.IndexView.as_view(), name='index'),
path('<int:pk>/', views.DetailView.as_view(), name='detail'),
path('<int:pk>/results/', views.ResultsView.as_view(), name='results'),
path('<int:question_id>/vote/', views.vote, name='vote'),
```

Nous avons besoin de spécifier pour nos différentes vues basées sur des classes le nom de la vue en spécifiant la méthode *as_view()* pour transformer la classe en fonction de vue. Également nos vues modèles attendent la variable nommée *pk* pour indiquer l'id de la question souhaité

Suivez-nous sur LinkedIn  Twitter  @Adrar_Numerique

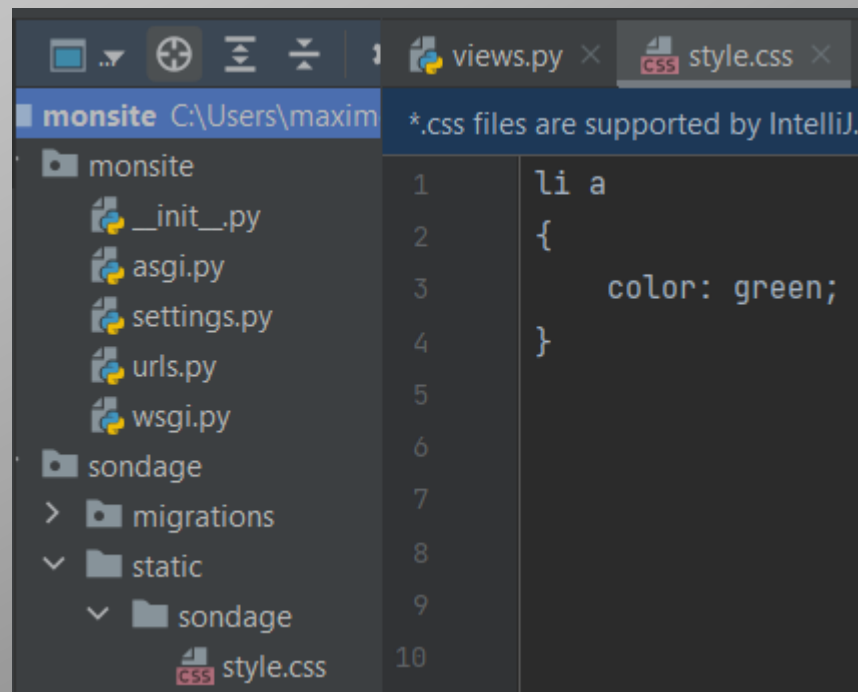
... et sur le web : www.adrar-numerique.com

FICHIERS STATIQUES ET GABARITS

Suivez-nous sur LinkedIn Twitter **@Adrar_Numerique**

... et sur le web : **www.adrar-numerique.com**

Comme pour les templates, les fichiers statiques (feuilles de style, scripts et images) doivent être stockés à un endroit précis pour que django les retrouvent. Ils doivent se trouver dans *sondage/static/sondage*. Créez cette arborescence et placez y un fichier *style.css* dans lesquels vous mettez un css de base.



Suivez-nous sur LinkedIn Twitter @Adrar_Numerique

... et sur le web : www.adrar-numerique.com

Pour l'intégrer à notre page html il faut tout d'abord le charger, via une balise de gabarit *load* puis l'indiquer comme n'importe quel fichier css mais en utilisant une syntaxe de gabarit pour le liens :

```

<!DOCTYPE html>
{% load static %}
<html>
  <head>
    <meta charset="utf-8">
    <title>Mon sondage</title>
    <link rel="stylesheet" type="text/css" href="{% static 'sondage/style.css' %}">
  </head>

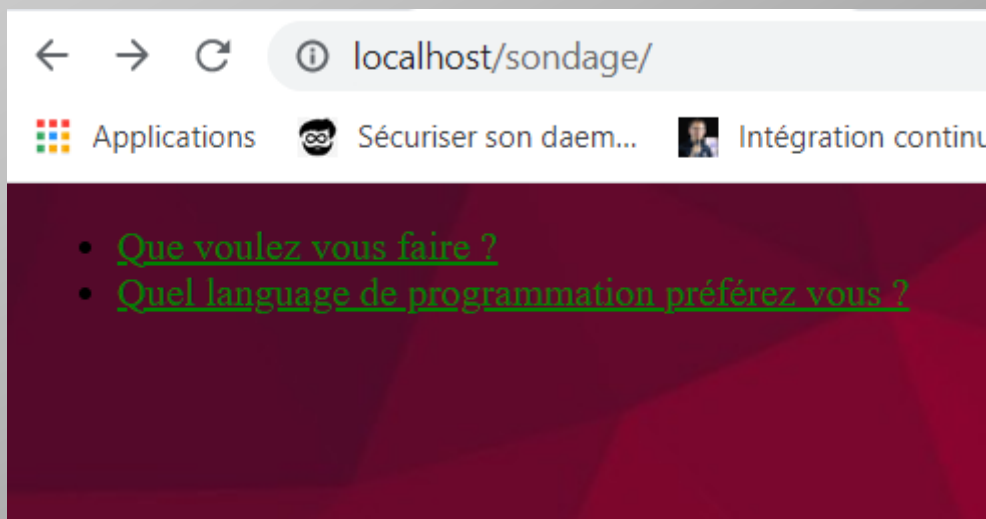
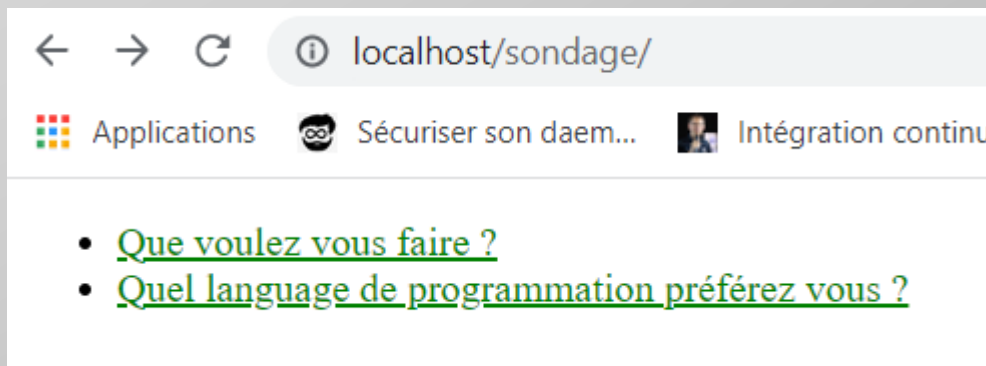
```

{% load static %} permet donc de charger le répertoire de fichiers statiques et *{% static 'sondage/style.css' %}* permet d'indiquer que le fichier à charger est *sondage/static/sondage/style.css*

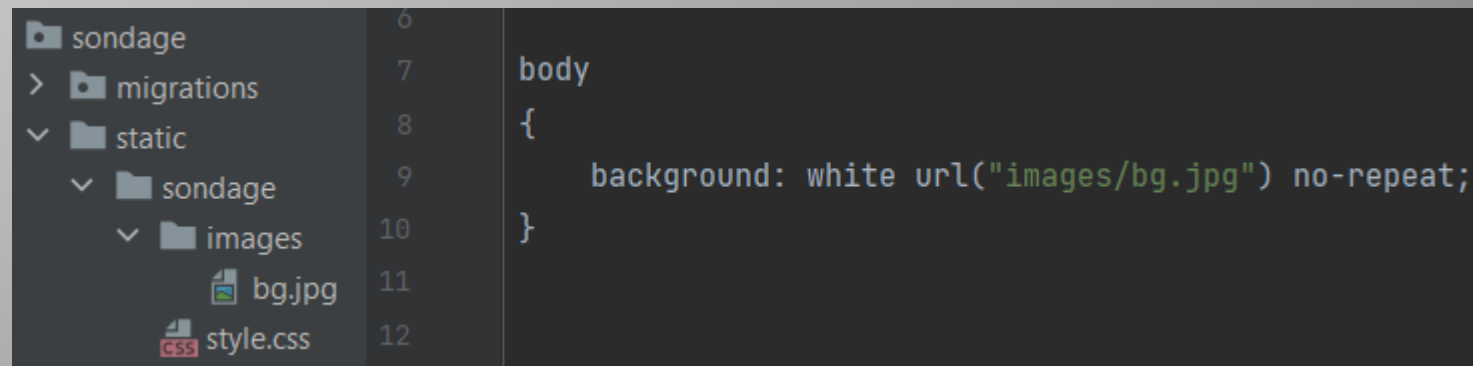
Suivez-nous sur LinkedIn Twitter @Adrar_Numerique

... et sur le web : www.adrar-numerique.com

Voici le résultat :



Cela fonctionne de la même manière pour les images



/!\ Vous ne pouvez pas utiliser la balise de gabarits `{% load static %}` dans un fichier de style. Vous devez utiliser un chemin relatif

Suivez-nous sur LinkedIn Twitter @Adrar_Numerique

... et sur le web : www.adrar-numerique.com

Vous pouvez inclure une page HTML dans une autre afin de ne pas avoir à réécrire le code qui est similaire d'une page à une autre. Il faudra définir des blocks dans votre page modèle que vous pourrez surcharger dans vos autres pages.

```
<!DOCTYPE html>
{% load static %}
<html>
  <head>
    <meta charset="utf-8">
    <title>Mon sondage</title>
    <link rel="stylesheet" type="text/css" href="{% static 'sondage/style.css' %}">
  </head>
  <!-- HTML de base -->
  <body>
    {% block contenue %}
      {% if question_list %}
        <ul>
          {% for question in question_list %}
            <li>
              <a href="{% url 'sondage:detail' question.id %}">{{ question.question_text }}</a>
            </li>
          {% endfor %}
        </ul>
      {% else %}
        <p>Pas de sondage disponible.</p>
      {% endif %}
    {% endblock %}
  </body>
</html>
```

Lorsqu'on inclura notre page index.html a notre page détail nous n'aurons qu'à spécifier ou démarre le bloque *contenue* et y indiquer dedans ce que nous voulons

Suivez-nous sur LinkedIn Twitter @Adrar_Numerique

... et sur le web : www.adrar-numerique.com

Il suffira dans notre page détail d'indiquer quelle page il faut inclure, et ce qu'il faut mettre dans les blocks que l'on veut remplacer. Tout bloc qui n'est pas ré écrit dans notre nouvelle page ne sera pas modifié. Il faudra toutefois faire les appels de chargement externe (`{% load xxxx %}`) dans chaque page

```
{% extends "sondage/index.html" %}
{% load static %}
{% block contenu %}
<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>
{% endblock %}
```


Suivez-nous sur LinkedIn  Twitter  **@Adrar_Numerique**

... et sur le web : **www.adrar-numerique.com**

Voici à quoi ressemble notre page *detail.html* en ayant chargé l'entête de notre page *index.html*.

N.B: nous aurions très pu charger des éléments tels que un menu de navigation, un footer ou un header.

Quel langage de programmation préférez vous ?

- ☐ Python
- ☐ Php

Vote