



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, TEMPLATES ET GABARITS

Suivez-nous sur LinkedIn  Twitter  @Adrar_Numerique

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

Nous allons maintenant rédiger quelques vues afin d'afficher de faire vire un peu notre projet. Une page pour afficher une question en particulier avec un formulaire de vote, une page affichant les votes d'une question, et une page permettant de gérer le vote. Rajoutons ces vues à notre fichier `sondage/views.py` :

```
def detail(request, question_id):  
    return HttpResponse("Voici la question numéro %s." % question_id)  
  
def results(request, question_id):  
    response = "Voici les votes de la question numéro %s."  
    return HttpResponse(response % question_id)  
  
def vote(request, question_id):  
    return HttpResponse("Vous votez pour la question numéro %s." % question_id)
```

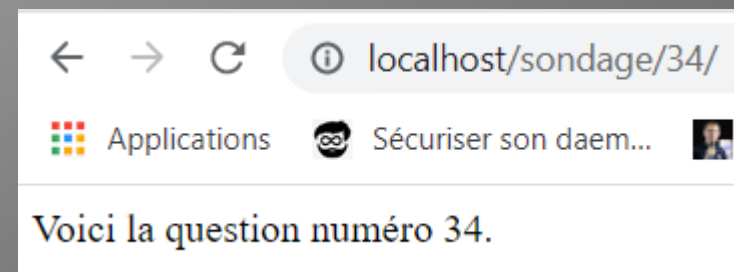
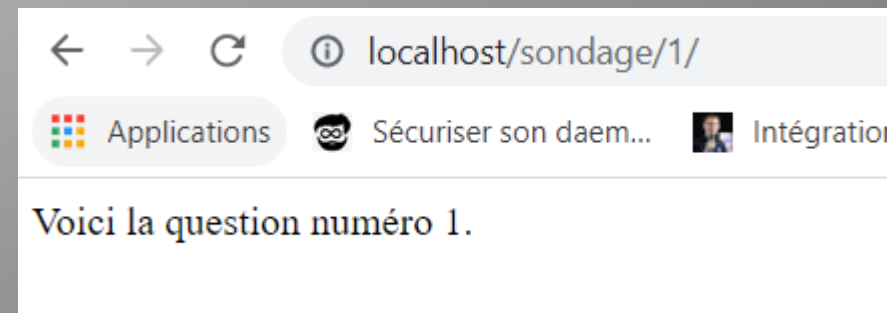
Ces vues sont déclarées avec comme paramètre `question_id` qui sera l'id de la question voulues. Le passage de ce paramètre se fera par l'objet `request` qui est généré en fonction de la requête HTTP. C'est donc dans l'url qu'il faudra passer cet ID en paramètre. Voyons maintenant le fichier `sondage/urls.py`.

Suivez-nous sur LinkedIn Twitter @Adrar_Numerique

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

```
urlpatterns = [
    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'),
]
```

Nous rajoutons une url par vue que nous avons créée qui auront toute comme chemin de départ la variable `<int:question_id>`. Quand nous accèderons à la page `http://localhost/sondage/1` django appellera la vue nommé détail et mettra dans le paramètre `question_id` de la fonction la valeur 1 passé dans l'url et nous. `question_id` n'est qu'une variable et n'est en rien relié à notre base de donnée. C'est pour ça que vous pouvez accéder à la page `http://localhost/sondage/34` alors qu'il n'existe pas de question 34 dans votre base de données



Suivez-nous sur LinkedIn Twitter @Adrar_Numerique

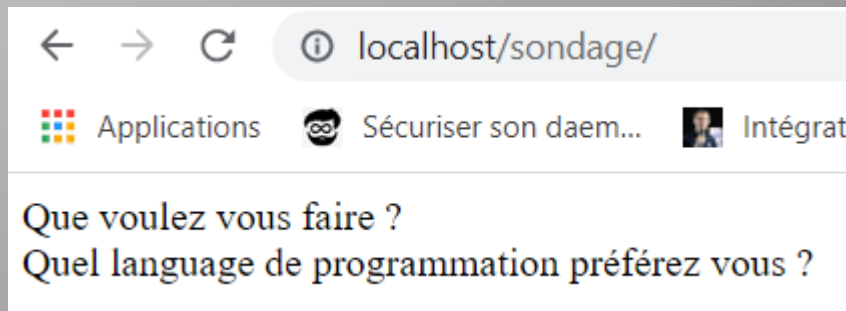
... et sur le web : www.adrar-numerique.com

Changeons maintenant nos vues pour qu'elle fasse vraiment quelque chose. Nous allons afficher sur l'index les 5 questions les plus récentes. Modifiez votre vue de cette manière :

```
from .models import Question, Choice

def index(request):
    latest_question_list = Question.objects.order_by('-pub_date')[:5]
    output = '<br> '.join([q.question_text for q in latest_question_list])
    return HttpResponse(output)
```

Nous créons tout d'abord un objet *QuerySet* de nos 5 premières questions triées par ordre décroissant de nos dates de publications. Nous créons ensuite une chaîne de caractère des éléments de notre variable que nous affichons sur la page. N.B: vue que nous envoyons le résultat sur une page web nous générons le retour à la ligne par un `
` et non par un `\n`.



Suivez-nous sur LinkedIn  Twitter  @Adrar_Numerique

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

Vous pourriez faire toutes vos pages avec le css inclus de cette manière mais ce n'est pas comme ça que django est prévu et surtout ça ne permettrait pas de modification faciles.

Nous allons plutôt utiliser des templates HTML dans lesquels nous allons insérer les données voulus et que nous allons faire appeler par nos vues. Nous allons donc encore une fois modifier notre vue index pour cette fois envoyer la réponse http vers un template que nous écrirons après. Pour se faire vous aurez besoin d'importer *loader* de *django.template*.

```
from django.http import HttpResponse
from django.template import loader

from .models import Question, Choice

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

Ici nous chargeons dans notre variable *template* le template index.html qui sera situé dans le répertoire *sondage* du répertoire de template de notre application.

Nous définissons ensuite un contexte afin de pouvoir utiliser notre variable dans notre code html.

Nous envoyons la réponse http à notre template en lui indiquant la requête initiale ainsi que le contexte de variables à utiliser.

Suivez-nous sur LinkedIn Twitter @Adrar_Numerique

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

Nous avons indiqués dans notre vues que le template *index.html* se situait dans le répertoire *sondage*. En l'occurrence django va chercher tous les templates dans un dossier templates de l'application. *index.html* se trouve donc en fait dans *sondage/templates/sondage*. Créez donc cette arborescence, *templates/sondage*, dans votre application et créez dedans le fichier *index.html* dans lequel vous y insérez le code suivant (en replaçant le commentaire « HTML de base » par la base d'un fichier html):

```
<!-- HTML de base -->
<body>
  {% if question_list %}
    <ul>
      {% for question in question_list %}
        <li>
          <a href="/sondage/{{ question.id }}/">{{ question.question_text }}</a>
        </li>
      {% endfor %}
    </ul>
  {% else %}
    <p>Pas de sondage disponible.</p>
  {% endif %}
</body>
```

Ce qu'il y a à l'intérieur des balises {% %}, appelés gabarits, sont des instructions de codes qui seront interprétées.

Les balises {{ }} permettent d'utiliser des variables passées au préalable. En l'occurrence nous avons indiqué dans notre vue que la variable *latest_question_list* de la vue sera utilisable par le nom *question_list* dans notre template.

Suivez-nous sur LinkedIn Twitter @Adrar_Numerique

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

Django fournit une série de modules pour faciliter la syntaxe de beaucoup d'éléments. Revoyons notre vue `index()` :

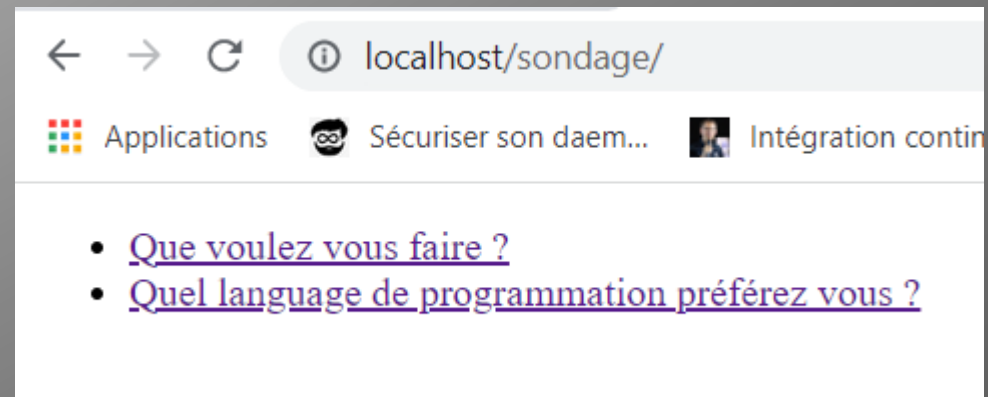
```
from django.http import HttpResponse
from django.template import loader

from .models import Question, Choice

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

```
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)
```

Nous avons remplacé *HttpResponse* par la fonction *render* qui prend en paramètres la requête initiale, le template et le contexte. N'oubliez pas d'importer *render* de *django.shortcuts* si il n'est pas déjà importé. Voici le rendu de notre vue `index`:



Suivez-nous sur LinkedIn Twitter @Adrar_Numerique

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

Modifions maintenant notre vue détail pour qu'elle affiche le détail de la question sur laquelle nous aurons cliqué à l'index de notre projet en implémentant une gestion d'erreur. Créons également un template simple.

```
def detail(request, question_id):
    try:
        question = Question.objects.get(pk=question_id)
    except Question.DoesNotExist:
        raise Http404("La question n'existe pas")
    return render(request, 'sondage/detail.html', {'question': question})
```

```
<!-- HTML de base -->
<h1>Question numéro {{question.id}}</h1>
<p>{{question}}</p>
```

Encore une fois django nous facilite la tâche avec une Exception si on n'essaye d'accéder à un élément de notre base qui n'existe pas et une Exception pour afficher une erreur de requête. Cette Exception, *Http404*, est à importer depuis *django.http*.

Suivez-nous sur LinkedIn Twitter @Adrar_Numerique

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

Une fois n'est pas coutume, django nous fournit encore un raccourci permettant de faciliter la gestion d'un code erreur 404 d'une requête get. Voici notre vue détail terminée en version courte : (pensez bien à importer la fonction)

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

Et voici le résultat

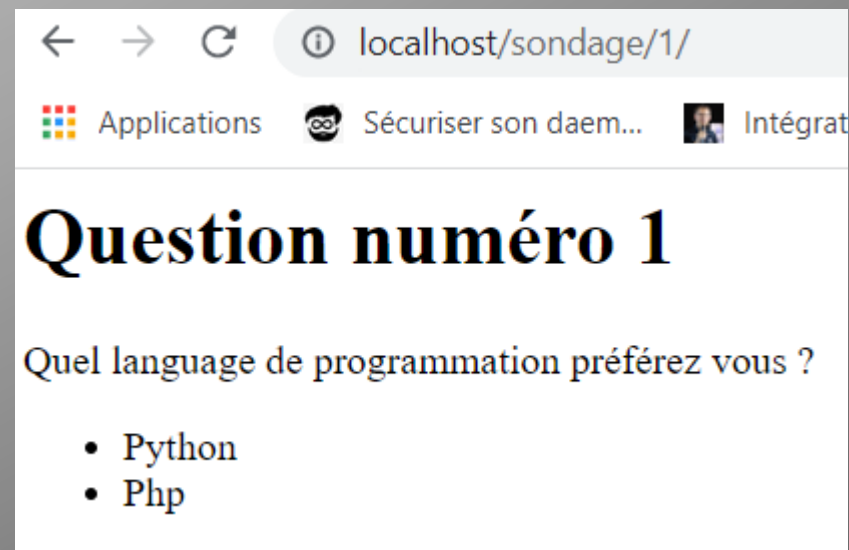


Suivez-nous sur LinkedIn Twitter @Adrar_Numerique

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

Personnalisons un peu plus notre template *detail.html* pour qu'il affiche également les choix possibles de notre question. Rappelez vous que nos question possèdent une méthode *choice_set* permettant de générer un *QuerySet* de nos clefs étrangères.

```
<!-- HTML de base -->
<h1>Question numéro {{question.id}}</h1>
<p>{{question}}</p>
<ul>
    {% for choice in question.choice_set.all %}
    <li>{{choice}}</li>
    {% endfor %}
</ul>
```



Suivez-nous sur LinkedIn Twitter @Adrar_Numerique

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

Nos vues et nos templates fonctionnent en l'état mais le fait d'avoir notre url en dur dans le code de nos pages html peut se retrouver fastidieux à modifier lorsqu'on se retrouve avec beaucoup de pages.

```
<a href="/sondage/{{ question.id }}/">{{ question.question_text }}</a>
```

Pour palier à ça django permet de modifier la syntaxe en faisant des références à notre page d'url en créant des espaces de noms pour nos applications. Voici à quoi cette modification ressemble dans notre script *urls.py* ainsi que dans notre code html.

```
<a href="{% url 'sondage:detail' question.id %}">{{ question.question_text }}</a>
```

```
app_name = 'sondage'
urlpatterns = [
    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'),
]
```

sondage ici fait référence à l'espace de nom définit dans la variable *app_name* de notre fichier *urls.py* et *detail* fait référence au paramètre *name* renseigné dans notre path.