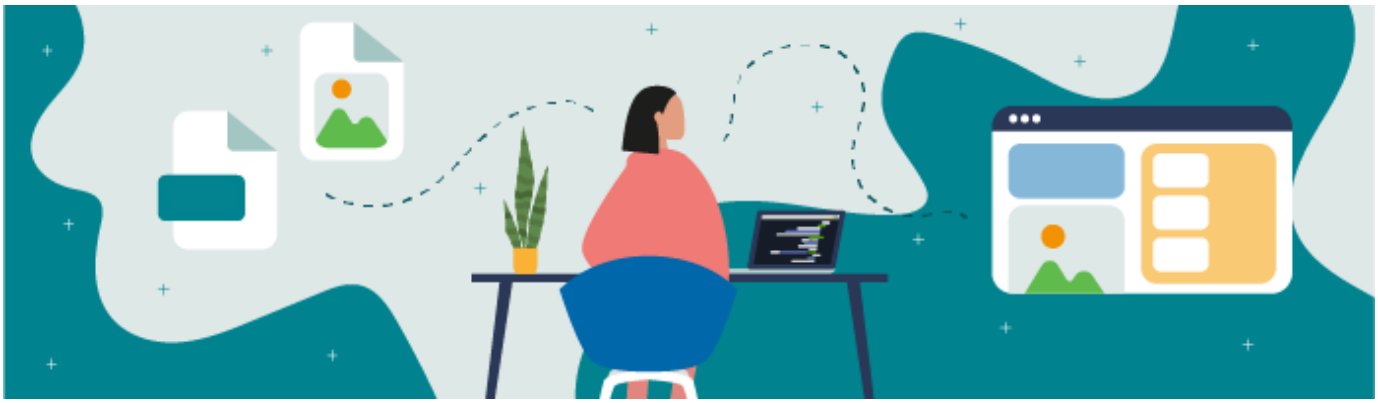
Débutez avec le framework Django

12 heures Moyenne

Mis à jour le 28/06/2023

Servez du contenu à l'aide d'une vue



Affichez une nouvelle page en 5 lignes de code !

Nous allons créer une nouvelle page sur notre site en écrivant seulement 5 lignes de code. Ce sera une page très basique, certes, mais ce sera tout de même une page !

Commençons par ouvrir le fichier listings/views.py. Pour l'instant, cela ressemble à ceci :

python

```python
# ~/projects/django-web-app/merchex/listings/views.py

from django.shortcuts import render

# Create your views here.
```

Supprimons le commentaire et remplaçons-le par une simple fonction. Nous allons également ajouter une déclaration d'import supplémentaire en haut :

python

```python
# ~/projects/django-web-app/merchex/listings/views.py

```

```
from django.http import HttpResponse
from django.shortcuts import render

def hello(request):
    return HttpResponse('<h1>Hello Django!</h1>')
```

Nous venons de créer une vue, l'un des blocs constitutifs de l'architecture MVT.

Une **vue** a pour fonction de répondre à la visite d'un utilisateur sur le site en renvoyant une page que l'utilisateur peut voir.

Ou en termes de Python :

Une vue est une fonction qui accepte un objet `HttpRequest` comme paramètre et retourne un objet `HttpResponse` .

Dans notre exemple de vue, nous renvoyons une réponse HTTP avec un contenu HTML simple : un titre H1 disant « Hello Django ! ».

Ouvrons donc le fichier merchex/urls.py. Actuellement, le code (sans les commentaires) ressemble à ceci :

python

```python
# ~/projects/django-web-app/merchex/merchex/urls.py

from django.contrib import admin
from django.urls import path

urlpatterns = [
path('admin/', admin.site.urls),
]
```

Importons le module views que nous avons édité à l'étape précédente, en ajoutant une déclaration d'import. Puis nous ajouterons un nouvel élément à la liste `urlpatterns` , où nous ferons référence à la fonction vue que nous venons de créer.

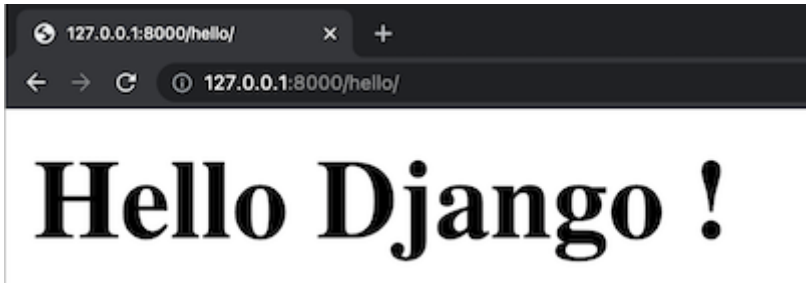
python

```python
# ~/projects/django-web-app/merchex/merchex/urls.py

from django.contrib import admin
from django.urls import path
from listings import views

urlpatterns = [
path('admin/', admin.site.urls),
path('hello/', views.hello)
]
```

Nous venons de créer un modèle d'URL et de le lier à notre vue. Un **modèle d'URL**, c'est la façon dont nous indiquons à Django qu'il doit être à l'écoute d'une requête pour une URL donnée, puis appeler une vue spécifique pour générer une page.

Maintenant, chaque fois que quelqu'un visite l'URL `hello/` sur notre site, la vue `hello` va générer une page pour nous. C'est exactement ce que nous allons essayer maintenant.

Ouvrez votre navigateur et tapez l'URL http://127.0.0.1:8000/hello/.



La page « hello ».

Voilà votre première page Django ! C'était rapide et simple, n'est-ce pas ?

Maintenant que vous avez essayé de votre côté, regardez-moi faire le processus pour vérifier que vous avez bien compris :

01:51

Nous allons ensuite ajouter une autre vue à notre application web, afin que nos utilisateurs puissent visiter une page de plus. Cette fois, nous allons examiner plus en détail ce qui se passe à chaque étape. Puis, à la fin du chapitre, vous ajouterez deux autres vues de votre côté, afin de mettre en pratique ce que vous avez appris.

Comprenez que tout commence par une URL

Toute interaction avec un site web commence par une URL. Vous pouvez la taper directement dans votre barre d'adresse, la retrouver dans un signet ou cliquer sur un lien à partir d'une autre page ou d'un

autre site, mais le résultat est le même : vous devez entrer une URL dans votre barre d'adresse avant que quoi que ce soit d'autre puisse se produire. Par exemple :

https://www.merchex.xyz/abous-us/

Sous le capot, votre navigateur commence à réfléchir à ce qu'il doit faire ensuite. Tout d'abord, il doit connaître l'adresse IP du serveur sur lequel ce site est hébergé. Pour ce faire, il effectue une recherche DNS sur le **nom de domaine** ou le **nom d'hôte**, c'est-à-dire tout ce qui se trouve entre les « :// » et le premier « / » : www.merchex.xyz

> Essayez maintenant de faire votre propre recherche DNS sur l'adresse **https://www.dynu.com/fr-FR/NetworkTools/DNSLookup** : tapez « google.com » ou « www.google.com » dans le champ Hôte, choisissez le "Serveur de nom public" Google et voyez quelle adresse IP est renvoyée.

Maintenant que le navigateur sait où se trouve le serveur, il prépare une **requête HTTP** pour y aller. La demande contient :

- une **méthode** : dans notre cas, `GET` . On peut considérer que cela signifie « Obtenez-moi cette page, s'il vous plaît » ;
- le **chemin de** l'URL : c'est-à-dire tout ce qui se trouve à partir du premier slash (`/`).

Lorsque la demande arrive sur notre serveur, c'est maintenant au tour de notre application de déterminer ce que veut la requête. Elle le fait en examinant le chemin.

Voici quelques chemins URL que l'on peut s'attendre à voir dans les requêtes HTTP adressées à notre application web :

- /about-us/
- /contact-us/
- /help/

Django a besoin d'un moyen de distinguer ces chemins, afin de pouvoir répondre avec le bon contenu.

Et c'est là que les modèles d'URL entrent en jeu. Voyons comment ils fonctionnent.

Faites correspondre une URL avec un modèle d'URL

Ouvrez le fichier `merchex/urls.py` . Dans ce fichier, nous définissons une liste de modèles d'URL. Chaque fois que Django reçoit une requête HTTP, il parcourt ces modèles dans l'ordre, un par un, et essaie de trouver une correspondance.

python

```python
# ~/projects/django-web-app/merchex/merchex/urls.py

urlpatterns = [
    path('admin/', admin.site.urls),
    path('hello/', views.hello),
```

```
6 ]
```

Ici, nous pouvons voir le modèle d'URL que nous avons créé à l'étape précédente, correspondant au chemin `hello/` .

> Lorsque vous ajoutez un chemin à un modèle d'URL, ne mettez jamais le slash de tête. C'est pourquoi, dans notre modèle d'URL, nous avons tapé `hello/` et non `/hello/` .

Si vous incluez un slash, votre modèle d'URL ne sera pas reconnu et votre page ne se chargera pas. Django vous affichera utilement un avertissement dans le terminal si vous faites cela.

> Nous pouvons également voir un autre modèle d'URL, correspondant à la chaîne « admin/ ». Il a été généré par l'interface en ligne de commande (CLI) de Django lorsque nous avons configuré notre projet. Nous verrons à quoi sert ce modèle d'URL dans [Effectuer des opérations CRUD via l'administration de Django] (lien vers P3C2).

Ajoutons un troisième modèle d'URL à cette liste pour une nouvelle page « about us ». Ajoutez une nouvelle ligne de code à la liste `urlpatterns` :

python

```python
# ~/projects/django-web-app/merchex/merchex/urls.py

urlpatterns = [
    path('admin/', admin.site.urls),
    path('hello/', views.hello),
    path('about-us/', views.about),  # ajoutez cette ligne
]
```

> À quoi servent les arguments que nous avons passés à la fonction `path` ?

Voyez si vous pouvez deviner l'objectif de chaque argument avant de poursuivre votre lecture.

- Le premier argument est une chaîne de caractères. Il s'agit du chemin URL que nous allons faire correspondre : « about-us/ ».
- Le second argument doit être une fonction de vue que nous avons définie dans views.py. La demande sera transmise à cette vue, et la vue générera une page.

> Est-ce que vous voyez le problème avec la vue que nous référençons dans notre nouveau modèle d'URL ? Elle n'existe pas encore ! Il n'y a pas de fonction dans « views.py » qui se nomme `about` . Cela fera planter le serveur de développement s'il est en cours d'exécution, mais ne paniquez pas. Tout sera de nouveau opérationnel dans la prochaine section.

Lorsqu'une URL correspond à un modèle d'URL, le déroulement passe à l'étape suivante : la demande HTTP est transmise à la vue spécifiée.

Renvoyez une réponse HTTP avec une vue

C'est maintenant au tour de notre vue de générer une réponse HTTP qui peut être envoyée au navigateur.

Créons la vue que nous avons référencée dans notre modèle d'URL. Ouvrez `listings/views.py` . Ajoutez une autre fonction de vue sous celle que nous avons créée plus tôt dans ce chapitre :

python

```python
# ~/projects/django-web-app/merchex/listings/views.py

from django.http import HttpResponse
from django.shortcuts import render

def hello(request):
    return HttpResponse('<h1>Hello Django !</h1>')

def about(request):
    return HttpResponse('<h1>À propos</h1> <p>Nous adorons merch !</p>')
```

Les vues sont des fonctions Python qui doivent remplir les critères suivants :

- Le premier paramètre de la fonction est une variable contenant l'objet `HttpRequest` , et par convention nous le nommons toujours `request` . Cet objet contient certains attributs utiles liés à la demande. Nous les examinerons plus avant au chapitre Récupérez les données de l'utilisateur avec Django Forms.
- La valeur de retour de la fonction est toujours un objet `HttpResponse` . Ce qu'il contient dépend du type d'application : dans notre application, nous allons placer du HTML dans nos réponses.

Génial ! Notre fonction de vue a renvoyé une réponse HTTP. La dernière chose que fait Django est de renvoyer cette réponse au navigateur qui a fait la demande initiale. Django a fait son travail et c'est maintenant au navigateur d'afficher la page. On teste ?...

Vérifiez que votre serveur est en cours d'exécution (démarrez ou redémarrez-le si nécessaire), puis visitez http://127.0.0.1:8000/about-us.

La page « about-us ».

Et nous avons une nouvelle page sur notre site !

Si vous ne savez toujours pas comment les modèles d'URL et les vues fonctionnent ensemble, cette vidéo devrait vous aider à comprendre :

01:12

> Je comprends donc que les vues sont des fonctions et nous en avons écrit deux... mais je ne me souviens pas avoir ***appelé*** ces fonctions dans le code que nous avons écrit. Alors comment ça marche ?

C'est une excellente question. La réponse, c'est que le framework Django appelle ces fonctions de vue en votre nom. Tout ce que vous aviez à faire était de spécifier (dans urls.py) la fonction qui devait être utilisée pour chaque motif d'URL. Django s'occupe du reste.

Il s'agit d'un exemple de ce que nous appelons la « magie » des frameworks de programmation : les choses que le framework fait « sous le capot » et qui, en surface, semblent enfreindre les règles que nous connaissons en matière de programmation. Cela peut sembler peu intuitif au début, mais ça devient vite une seconde nature.

> Cela me semble étrange : nous avons visité **http://127.0.0.1:8000/about-us** dans le navigateur, mais il n'y a pas de fichier dans mon projet appelé « about-us ». Habituellement, il doit y avoir un fichier quelque part, peut-être appelé « about.html » ou « about.php », mais pas dans Django. Comment cela fonctionne en pratique ?

Si vous êtes habitué à accéder directement aux fichiers .html ou .php dans le navigateur, Django peut vous sembler étrange. Mais persévérez. Tout au long du cours, nous verrons comment Django propose

le contenu sans avoir besoin d'un fichier physique pour chaque page.

C'est à vous ! Ajoutez des pages à votre site avec des modèles d'URL et des vues

Je vous ai montré comment ajouter de nouvelles pages à votre site en utilisant des modèles d'URL et des vues. Maintenant, je veux que vous utilisiez ce que vous avez appris pour ajouter deux nouvelles pages à votre application. Ce sont des pages sur lesquelles nous allons nous appuyer dans les chapitres suivants.

Les deux pages seront une page « listings » (où nous afficherons éventuellement une liste des annonces pour les articles) et une page « contact us » (où nous créerons un formulaire de contact). Pour le moment, ces pages peuvent être très simples : juste une balise <h1> et une balise <p>. Nous construirons le contenu complet plus tard dans le cours.

Vous devez visiter ces pages dans le navigateur pour vérifier qu'elles s'affichent correctement.

> Si vous avez besoin de conseils :
>
> Pour chaque page, vous aurez besoin de :
>
> - une vue dans listings/views.py ;
> - un modèle d'URL dans merchex/urls.py. Chaque modèle d'URL nécessite 2 arguments :
>   - une chaîne qui correspond au chemin d'accès à l'URL de la page,
>   - une vue (n'oubliez pas de l'importer en haut du fichier !)
>
> Si vous ne voyez pas vos pages dans le navigateur, il se peut que le serveur de développement ait planté ! Voir [Tirez le maximum de ce cours] (lien vers P1C1) pour les étapes de débogage.

En résumé

- Lorsque Django reçoit une requête HTTP, il tente de trouver une correspondance pour le chemin de cette requête dans une liste de modèles d'URL, définis dans urls.py.

- Si le chemin correspond à un modèle d'URL, Django transmet la requête à la vue correspondante, que nous définissons dans views.py.
- La vue est une fonction qui accepte la demande en tant que paramètre et qui renvoie une `HttpResponse` comme valeur de retour. Cette réponse contient le HTML que le navigateur utilise pour afficher la page.

Maintenant que nous avons la vue, le « V » de notre architecture MVT, nous sommes prêts à afficher des données dans nos pages, avec un modèle.

Indiquer que ce chapitre n'est pas terminé

Configurez un nouveau projet avec l'utilitaire de ligne de commande de Django

Sauvegardez des données dans une base de données avec un modèle et une migration

Les professeurs

Patrick Wampé

Développeur full stack et Data Scientist. Formateur dans plusieurs écoles d'informatique, il a également écrit un livre sur l'IA.

Patrick Heneghan

Software engineer in the UK, coding mostly in Python on backend systems.

Rafiq Hilali

British Software Engineer and Django expert with Lambert Labs. Currently based in BC, Canada.

OPENCLASSROOMS

OPPORTUNITÉS

AIDE

POUR LES ENTREPRISES

EN PLUS

Français

Télécharger dans l'App Store