Débutez avec le framework Django

12 heures Moyenne

Mis à jour le 26/10/2022

Tirez le maximum de ce cours

01:54

Faites connaissance avec vos professeurs

Patrick Wampé

Patrick Wampé est développeur full-stack et data scientist freelance. Il est spécialisé dans les langages Python et JavaScript, bien qu'il ait travaillé sur de nombreux autres langages. Depuis plusieurs années, il donne des cours d'informatique dans différentes écoles d'ingénieurs, et a écrit un livre sur le Machine Learning et le Deep Learning.

Patrick Heneghan

Patrick Heneghan, originaire du Royaume-Uni, est ingénieur logiciel depuis 2007 et travaille avec Python depuis 2013. Il utilise Django et d'autres frameworks MVC/MVT dans les systèmes qu'il construit.

Rafiq Hilali

Rafiq Hilali est ingénieur logiciel senior travaillant pour Lambert Labs. Spécialisé dans le langage Python, il a conçu et livré plusieurs applications Django de niveau production. Originaire du Royaume-Uni, Rafiq profite de sa carrière de développeur pour voyager et travailler dans le monde entier, de l'Indonésie à la Nouvelle-Zélande en passant par les Philippines et le Canada.

Apprenez avec un scénario professionnel

Dans ce cours, vous créerez une application web complète du début à la fin. Voici le contexte de cette application web :

On vous a demandé de créer une démonstration de concept pour une nouvelle application web : la bourse aux marchandises Merch Exchange. Il s'agira d'une application où les gens pourront répertorier divers articles de musique rares et de collection, tels que des disques, des affiches et des billets de concert. Les utilisateurs doivent pouvoir publier des listes des articles qu'ils souhaitent vendre.

Utilisation de Git

Tout au long de votre travail sur ce projet, je vous recommande fortement de suivre votre code sur Git.

> Si vous n'avez pas encore installé Git, téléchargez le dernier installateur pour votre système d'exploitation à l'adresse suivante : **https://git-scm.com/downloads**.

Dans les premiers chapitres du cours, j'aborderai l'initialisation de votre repository et vous proposerai des suggestions pour votre fichier .gitignore, mais je vous laisserai le soin de faire le reste.

Vous pouvez également utiliser le dépôt GitHub du cours comme référence.

Lecture des exemples de code

Vous remarquerez que certains des exemples de code comprennent des ellipses (`...`) ; celles-ci indiquent des passages où j'ai omis une partie du code pour que les exemples restent courts. Nous serons souvent amenés à modifier des fichiers contenant de nombreuses lignes de code, et il ne serait pas judicieux d'afficher le fichier dans son intégralité à chaque fois.

> Il est courant dans les projets informatiques d'écrire son code en utilisant des termes **anglais** (**la langue véhiculaire internationale de l'informatique**). Donc ne soyez pas surpris de voir des mots anglais dans les exemples de code. Le niveau d'anglais demandé n'est pas élevé, et vous n'aurez aucune peine à comprendre le code.

Utilisation du terminal (alias le shell ou l'invite de commande)

Il existe de nombreux terminaux disponibles sur différents systèmes d'exploitation, et ils affichent tous les sorties et les entrées de différentes manières. Mais les principes de base restent toujours les mêmes :

- vous tapez des commandes dans le terminal (entrée) ;
- vous obtenez des lignes de texte en retour (sortie).

Lorsqu'un terminal est prêt et attend que vous tapiez quelque chose, vous voyez une invite. Une invite peut ressembler à ceci :

- `$`
- `→`
- `C:\>`

Pour simplifier les choses, les exemples de code de terminal de ce cours utiliseront le symbole `→` pour désigner l'invite.

Partout où vous voyez le symbole `→` dans les exemples de code du terminal, par exemple :

text

```
1 → ./manage.py runserver
```

... vous devez taper tout le texte qui vient après (sans le signe `→`), puis taper Entrée pour exécuter le code.

Les invites dans les terminaux seront souvent précédées du chemin du répertoire actuel pour vous rappeler où vous vous trouvez. Nous utiliserons un format similaire dans les exemples de code, afin que vous sachiez toujours où vous devez être avant d'exécuter une commande :

text

```
1 ~/projects/django-web-app/merchex
2 → ./manage.py runserver
```

Si vous n'êtes pas sûr du répertoire dans lequel vous vous trouvez, vous pouvez lancer la commande `pwd` (« print working directory ») à tout moment pour vous rappeler dans quel répertoire vous vous situez, par exemple :

text

```
→ pwd
~/projects
```

Les autres lignes qui ne commencent pas par `→` représentent la sortie de la commande que nous venons d'exécuter. Vous n'avez pas besoin de taper ce texte, il est juste présent pour que vous puissiez suivre. Et n'oubliez pas que ce que vous voyez dans votre terminal peut être légèrement différent de ce qui figure dans les exemples de code.

> Sous Windows, les slashs dans les chemins seront bien sûr `\` au lieu de `/` . De plus, Windows n'utilise pas le symbole `~` pour désigner le répertoire de travail personnel, mais vous pouvez remplacer ce symbole par `C:\Users\Votrenom` .

Dans le shell Django, l'invite ressemble à ceci :

text

```
>>>
```

À moins que vous n'ayez déjà installé IPython, auquel cas il ressemble à ceci :

text

```
In[1]:
```

Les exemples de code de ce cours sont écrits avec le premier de ces 2 symboles. Vous verrez donc des exemples de code comme celui-ci :

text

```
>>> from bands.models import Band
```

Encore une fois, vous devez taper tout ce qui suit (sans inclure) le `>>>` , et appuyer sur Entrée pour exécuter le code.

Vous pouvez vous référer à cette liste de commandes Linux avec leurs équivalents Windows. Par exemple, la commande Linux `ls` , qui permet d'afficher la liste des fichiers du répertoire dans lequel on se trouve, correspond à la commande `dir` dans un système Windows.

Retour au projet après un redémarrage

Vous devez toujours avoir votre environnement virtuel activé pendant le développement. Si vous redémarrez votre machine et revenez ensuite au projet, vous pouvez activer votre environnement virtuel en vous assurant d'abord que vous êtes dans le répertoire qui contient votre répertoire `env` et en exécutant :

text

```
1 → source env/bin/activate
```

Si votre site a cessé de fonctionner

De temps en temps, vous pouvez remarquer que le serveur de développement plante pendant que vous codez.

Ne paniquez pas, cela se produira souvent pendant le développement. Le serveur de développement recharge automatiquement votre code dès qu'il détecte que vous avez enregistré quelque chose de nouveau dans l'un de vos fichiers. C'est souvent très pratique, mais cela signifie également que si votre code est incomplet, le serveur rechargera automatiquement votre code inachevé et risque alors de planter.

N'oubliez pas que certains environnements de développement intégré (dont PyCharm) utilisent une fonction de sauvegarde automatique, qui peut déclencher un rechargement du serveur de développement sans sauvegarde manuelle !

Souvent, le processus du serveur reste en cours d'exécution et attend le prochain changement, après quoi il redémarre.

Il arrive cependant que le serveur ne puisse pas se remettre de cette erreur. Dans ce cas, vous devrez peut-être finir le code que vous êtes en train d'écrire, puis passer au terminal et :

- Retourner à une invite dans le terminal (si vous n'en voyez pas déjà une) avec Ctrl-C.
- Vous assurer que vous êtes dans le bon répertoire : si vous lancez la commande `ls` , vous devriez pouvoir voir `manage.py` . Si vous ne le voyez pas, naviguez vers le répertoire qui contient ce fichier, par exemple avec `cd bandub` .
- Redémarrer le serveur de développement avec `./manage.py runserver` .

  `manage.py not found`

  Assurez-vous que vous êtes dans le répertoire qui contient le script manage.py avant de l'exécuter ! Lancez `ls` pour lister le contenu du répertoire courant. Si vous ne voyez pas `manage.py` dans la liste, alors lancez une ou plusieurs commandes `cd` pour naviguer dans ce répertoire.

Maintenant que vous savez comment suivre ce cours, vous êtes prêt à installer Django !

#

J'ai terminé ce chapitre et je passe au suivant

Et si vous obteniez un diplôme OpenClassrooms ?

- Formations jusqu'à 100 % financées
- Date de début flexible
- Projets professionnalisants
- Mentorat individuel

Trouvez la formation et le financement faits pour vous

Être orienté

Comparer nos types de formations

Débutez avec le framework Django

Installez Django avec pip

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.

Français

Télécharger dans l'App Store