Plan du projet

Créez un répertoire de projet et entrez-y :

```
$ mkdir flask-tutorial
$ cd flask-tutorial
```

Suivez ensuite les instructions d'installation pour configurer un environnement virtuel Python et installer Flask pour votre projet.

Le tutoriel supposera que vous travaillez dans le répertoire `flask-tutorial` à partir de maintenant. Les noms de fichiers en haut de chaque bloc de code sont relatifs à ce répertoire.

Une application Flask peut être aussi simple qu'un simple fichier.

`hello.py`

```
from flask import Flask

app = Flask(__name__)


@app.route('/')
def hello():
    return 'Hello, World!'
```

Cependant, lorsqu'un projet prend de l'ampleur, il devient difficile de conserver tout le code dans un seul fichier. Les projets Python utilisent des *paquets* pour organiser le code en plusieurs modules qui peuvent être importés là où c'est nécessaire, et le tutoriel fera de même.

Le répertoire du projet contiendra :

- `flaskr/`, un paquet Python contenant le code et les fichiers de votre application.
- `tests/`, un répertoire contenant les modules de test.
- `venv/`, un environnement virtuel Python où Flask et d'autres dépendances sont installées.
- Les fichiers d'installation indiquant à Python comment installer votre projet.
- La configuration du contrôle de version, tel que git. Vous devriez prendre l'habitude d'utiliser un certain type de contrôle de version pour tous vos projets, quelle que soit leur taille.
- Tout autre fichier du projet que vous pourriez ajouter à l'avenir.

À la fin, la mise en page de votre projet ressemblera à ceci :

```
/home/user/Projects/flask-tutorial
├── flaskr/
│   ├── __init__.py
│   ├── db.py
```

v: latest

```
    │   ├── schema.sql
    │   ├── auth.py
    │   ├── blog.py
    │   ├── templates/
    │   │   ├── base.html
    │   │   ├── auth/
    │   │   │   ├── login.html
    │   │   │   └── register.html
    │   │   └── blog/
    │   │       ├── create.html
    │   │       ├── index.html
    │   │       └── update.html
    │   └── static/
    │       └── style.css
    ├── tests/
    │   ├── conftest.py
    │   ├── data.sql
    │   ├── test_factory.py
    │   ├── test_db.py
    │   ├── test_auth.py
    │   └── test_blog.py
    ├── venv/
    ├── setup.py
    └── MANIFEST.in
```

Si vous utilisez le contrôle de version, les fichiers suivants, générés lors de l'exécution de votre projet, doivent être ignorés. Il peut y avoir d'autres fichiers en fonction de l'éditeur que vous utilisez. En général, ignorez les fichiers que vous n'avez pas écrits. Par exemple, avec git :

```
.gitignore

venv/

*.pyc
__pycache__/

instance/

.pytest_cache/
.coverage
htmlcov/

dist/
build/
*.egg-info/
```

Continuer vers Configuration de l'application.

v: latest