

Configuration de l'application

Une application Flask est une instance de la classe **Flask**. Tout ce qui concerne l'application, comme la configuration et les URLs, sera enregistré avec cette classe.

La façon la plus simple de créer une application Flask est de créer une instance globale **Flask** directement en haut de votre code, comme dans l'exemple « Hello, World ! » de la page précédente. Bien que cette méthode soit simple et utile dans certains cas, elle peut poser des problèmes délicats lorsque le projet se développe.

Au lieu de créer une instance **Flask** globalement, vous la créerez dans une fonction. Cette fonction est connue sous le nom de *fabrique d'applications*. Toute configuration, tout enregistrement et tout autre réglage dont l'application a besoin se fera dans la fonction, puis l'application sera retournée.

Fabrique d'applications

C'est le moment de commencer à coder ! Créez le répertoire `flaskr` et ajoutez le fichier `__init__.py`. Le fichier `__init__.py` a une double fonction : il contiendra la fabrique de l'application, et il indique à Python que le répertoire `flaskr` doit être traité comme un paquet.

```
$ mkdir flaskr
```

```
flaskr/__init__.py
```

```
import os
```

```
from flask import Flask
```

```
def create_app(test_config=None):
    # create and configure the app
    app = Flask(__name__, instance_relative_config=True)
    app.config.from_mapping(
        SECRET_KEY='dev',
        DATABASE=os.path.join(app.instance_path, 'flaskr.sqlite'),
    )

    if test_config is None:
        # load the instance config, if it exists, when not testing
        app.config.from_pyfile('config.py', silent=True)
    else:
        # load the test config if passed in
        app.config.from_mapping(test_config)

    # ensure the instance folder exists
    try:
```

 v: latest ▼

```


    os.makedirs(app.instance_path)
except OSError:
    pass

# a simple page that says hello
@app.route('/hello')
def hello():
    return 'Hello, World!'

return app

```

`create_app` est la fonction de fabrique de l'application. Vous la complétez plus tard dans le tutoriel, mais elle fait déjà beaucoup.

1. `app = Flask(__name__, instance_relative_config=True)` crée l'instance Flask.
 - `__name__` est le nom du module Python actuel. L'application a besoin de savoir où il se trouve pour configurer certains chemins, et `__name__` est un moyen pratique de le lui dire.
 - `instance_relative_config=True` indique à l'application que les fichiers de configuration sont relatifs au dossier instance. Le dossier d'instance est situé en dehors du paquet `flaskr` et peut contenir des données locales qui ne doivent pas être soumises au contrôle de version, comme les secrets de configuration et le fichier de base de données.
2. `app.config.from_mapping()` définit une certaine configuration par défaut que l'application utilisera :
 - SECRET_KEY est utilisé par Flask et les extensions pour garder les données en sécurité. Elle est définie à 'dev' pour fournir une valeur pratique pendant le développement, mais elle doit être remplacée par une valeur aléatoire lors du déploiement.
 - DATABASE est le chemin où le fichier de la base de données SQLite sera enregistré. Il se trouve sous `app.instance_path`, qui est le chemin que Flask a choisi pour le dossier de l'instance. Vous en apprendrez plus sur la base de données dans la section suivante.
3. `app.config.from_pyfile()` remplace la configuration par défaut par des valeurs prises dans le fichier `config.py` du dossier de l'instance s'il existe. Par exemple, lors du déploiement, cela peut être utilisé pour définir une véritable SECRET_KEY.
 - `test_config` peut aussi être passé à la fabrique, et sera utilisé à la place de la configuration de l'instance. Ceci afin que les tests que vous écrirez plus tard dans le tutoriel puissent être configurés indépendamment des valeurs de développement que vous avez configurées.
4. `os.makedirs()` assure que `app.instance_path` existe. Flask ne crée pas le dossier d'instance automatiquement, mais il doit être créé car votre projet y créera le fichier  `v: latest` ▼ base de données SQLite.

5. `@app.route()` crée une route simple pour que vous puissiez voir l'application fonctionner avant d'aborder le reste du tutoriel. Elle crée une connexion entre l'URL `/hello` et une fonction qui renvoie une réponse, la chaîne de caractères `'Hello, World!'` dans ce cas.

Démarrer l'application

Maintenant vous pouvez démarrer votre application en utilisant la commande `flask`. Depuis le terminal, dites à Flask où trouver votre application, puis lancez-la en mode développement. Rappelez-vous, vous devriez toujours être dans le répertoire de haut niveau `flask-tutorial`, pas dans le paquet `Flaskr`.

Le mode développement affiche un débogueur interactif chaque fois qu'une page lève une exception, et redémarre le serveur chaque fois que vous apportez des modifications au code. Vous pouvez le laisser fonctionner et recharger simplement la page du navigateur pendant que vous suivez le tutoriel.

Bash

CMD

Powershell

```
$ export FLASK_APP=flaskr
$ export FLASK_ENV=development
$ flask run
```

Vous verrez un résultat similaire à celui-ci :

```
* Serving Flask app "flaskr"
* Environment: development
* Debug mode: on
* Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
* Restarting with stat
* Debugger is active!
* Debugger PIN: 855-212-761
```

Visitez <http://127.0.0.1:5000/hello> dans un navigateur et vous devriez voir le message « Hello, World ! ». Félicitations, vous avez démarré votre première application web Flask !

Passez à [Définir et accéder à la base de données.](#)