[supDeCode]

®

Mai 2021

contact@supdecode.fr

Objectifs

- Découverte rapide du framework.

Pré-requis

- Programmation Python
- Conception DB et langage SQL
- MySQL + phpMyAdmin
- Design pattern MVC :
  - Routage
  - Entités
  - Contrôleurs
  - Vues
- ORM

Contenu (1/2)

- Historique
- Installation
- Créer une application
- Définir la table de routage
- Configurer une DB
- Créer des entités
- Migrer les entités
- Créer un contrôleur
- Créer une vue

Contenu (2/2)

- Gérer les fichiers statiques
- Utiliser le langage de template
- Créer une classe de formulaire
- Créer un vue de formulaire
- Créer un contrôleur d'édition d'une entité

Outils

- Interpréteur Python et ses accessoires.
  //www.python.org
- Django
  //www.djangoproject.com
- VS Code avec l'extension Python de Microsoft.
  //code.visualstudio.com

Historique

- Créé en 2010 par Armin Ronacher (UK).
- En 2018, élu *Framework web le plus populaire* par le *Python Developers Survey*.
- Actuellement en version 1.1.

Caractéristiques

- Micro-framework open-source, gratuit et extensible.
- Serveur SWGI (*Web Server Gateway Interface*) intégré.
- Pas d'ORM (mais extension *Flask-SQLAlchemy*).
- Pas de gestion des formulaires (mais extension *WTForms*).
- Langage de template *Jinja*.

Domaines d'utilisation

- Petites applications web.

Documentation

- Site officiel
  //flask.palletsprojects.com/en/1.1.x
- SQLAlchemy
  //flask-sqlalchemy.palletsprojects.com
- WTForms
  //wtforms.readthedocs.io

Installation sous Windows

- En shell via **pip** (*Package Installer for Python*, installé avec Python).
- Afficher la version

  `> py -m pip --version`

  `-m` (*me*) pour l'interpréteur associé à l'utilisateur courant.
- Upgrader (indispensable)

  `> py -m pip install --upgrade pip`
- Installer/Désinstaller un package

  `> py -m pip install package`

  `> py -m pip uninstall package`
- Upgrader un package

  `> py -m pip install --upgrade package`

Travailler avec un environnement virtuel

- Evite les conflits de versions de packages entre projets.
  - Définir un répertoire **`.env`** des environnements virtuels et s'y positionner.
  - Créer un environnement **`acme-env`** pour le projet ACME :

    ```
    > py -m venv acme-env
    ```

  - Activer l'environnement :

    ```
    > acme-env\Scripts\activate
    ```

  - Désactiver si besoin :

    ```
    > acme-env\Scripts\deactivate
    ```

Installer Flask dans l'environnement

- D'abord, upgrader pip :

```
(acme-env)> py -m pip install --upgrade pip
```

- Installer Flask :

```
(acme-env)> py -m pip install flask
```

- Lister les packages installés :

```
(acme-env)> py -m pip list
```

- Afficher la version de Flask :

```
(acme-env)> py -m flask --version
```

Installer les extensions

- Installez :
  - flask-sqlalchemy
  - mysqlclient
  - wtforms

L'application ACME

- Vous allez créer l'application ACME qui sera un catalogue de produits. Elle contiendra :
  - Une vue d'ensemble des produits ventilés par catégories.
  - Une vue de détail d'un produit.
  - Une vue d'erreur (produit inexistant).
  - Un formulaire d'édition de produit (ajout / modification).

Créer l'application

- Créez un répertoire **acme** et y placer le fichier **acme.py** :

```
from flask import Flask
app = Flask(__name__)

@app.route('/')
def test():
    return "Test"
```

- Vous venez de créer le répertoire de l'application et une première route.

Démarrer le serveur web

- Créer 2 variables shell globales :

```
(acme-env)> set FLASK_APP=acme.py
(acme-env)> set FLASK_ENV=development
```
(debugger actif)

- Démarrer le serveur :

```
(acme-env)> flask run
```

- Vérifier le résultat dans votre navigateur sur le port 5000 :

  //localhost:5000

Squelette de l'application

```
acme                    Racine de l'application
  +-- forms             Classes de description des formulaires
  +-- static            Assets
        +-- css
        +-- img
        +-- uploads
  +-- templates         Vues
  +-- acme.py           Configuration, contrôleurs et table de routage
```

- Vous pouvez déjà créer les répertoires manquants.

TD

Créer la table de routage (1/2)

- Sur le modèle de la route *test* et à l'aide de la documentation :

  //flask.palletsprojects.com/en/1.1.x/quickstart/#routing

  créez dans `acme.py` la table de routage de la diapo suivante...
- **Pour l'instant, les contôleurs retournent simplement un texte brut.**
- Testez dans votre navigateur.

TD

Créer la table de routage (2/2)

Verbe	Route	Méthode	Description
GET	`/products`	`listProducts`	Liste des produits
GET	`/products/{id}`	`showProduct`	Détail d'un produit
GET	`/products/noProduct`	`noProduct`	Produit inexistant
GET	`/products/edit (id=None)`	`editProduct`	Formulaire de saisie (ajout)
GET	`/products/edit/{id}`		Formulaire de saisie (modif.)
POST	`/products/edit (id=None)`		Ajout puis redirection
POST	`/products/edit/{id}`		Modification puis redirection

TD

Connecter la DB via *SQLAlchemy*

- Via *phpMyAdmin*, créez la DB vide : **`acmeflask`**
- A l'aide de la documentation :

  //flask-sqlalchemy.palletsprojects.com/en/2.x/config

  configurez et instanciez la connexion DB dans **`acme.py`**.
- Appelez **`db`** l'instance de connexion.

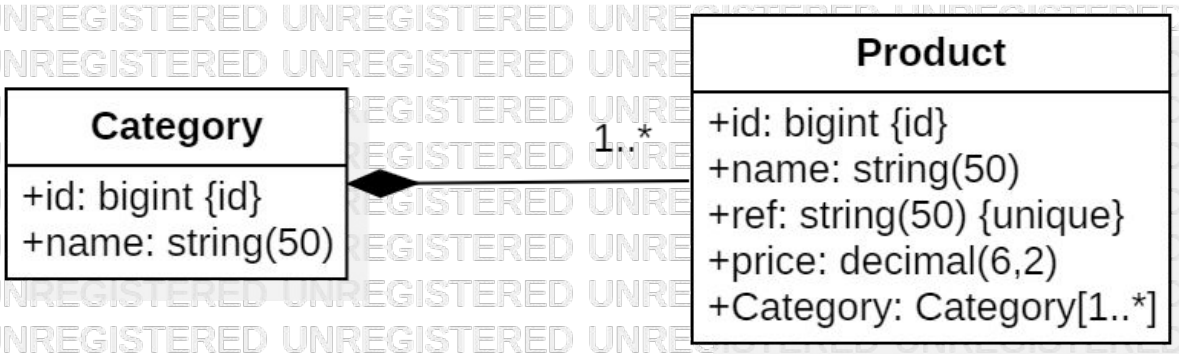
TD

Créez les entités

- A l'aide de la documentation :

  //flask-sqlalchemy.palletsprojects.com/en/2.x/models

  et du diagramme de classe :



  créez les entités **`Category`** et **`Product`** dans **`acme.py`** sans vous soucier des images.

TD

Migrer les entités

- A l'aide de la documentation :

  //flask-sqlalchemy.palletsprojects.com/en/2.x/quickstart

  migrez les entités vers la **`DB`** en utilisant la méthode d'instance **`create_all()`** de *SQLAlchemy* :

  - dans **`acme.py`** (puis commentez la ligne !),

    ou

  - à la console.

- Vérifiez le résultat dans la DB.

TD

Coder le contrôleur de la vue de détail (1/2)

- A l'aide de la documentation :

  //flask-sqlalchemy.palletsprojects.com/en/2.x/queries

  codez le contrôleur de la vue de détail.

- **Pour l'instant, ne vous préoccupez pas des éventuels *ID* inexistants.**

TD

Coder le contrôleur de la vue de détail (2/2)

- A l'aide de la documentation :

  //flask.palletsprojects.com/en/1.1.x/quickstart/#unique-urls-redirection-behavior

  redirigez vers la route *noProduct* en cas d'*ID* inexistant.

TD

Créer la vue de détail

- Placez le template fourni **`showProduct.html`** dans le répertoire **`templates`** et aidez vous de la documentation :

  `//flask.palletsprojects.com/en/1.1.x/tutorial/templates`

  pour que le contrôleur **`showProduct`** rende ce template au lieu d'afficher un simple texte.

- Vérifiez le résultat dans votre navigateur.
  Vous traiterez les CSS et l'image dans le TD suivant.

TD

Référencer les fichiers statiques

- Copiez le répertoire **`assets`** fourni à la racine du projet et renommez le **`static`**.
- Modifiez la vue de détail en conséquence.
- Vérifiez le résultat dans votre navigateur.

TD

Créer la vue pour les *ID* inexistants

- Placez le template fourni **`noProduct.html`** dans le répertoire **`templates`** et apportez les modifications nécessaires.

TD

Créer la vue d'ensemble

- Placez le template fourni **`listProducts.html`** dans le répertoire **`templates`** et apportez les modifications nécessaires pour afficher les produits ventilés par catégories.

TD

Créer la classe de description du formulaire d'édition

- A l'aide de la documentation :

  //flask.palletsprojects.com/en/1.1.x/patterns/wtforms

  créez dans le répertoire **`forms`** le fichier **`formProduct.py`** contenant la classe **`FormProduct`** de description du formulaire d'édition.

- **Ne vous souciez pas de l'image pour l'instant.**

TD

Créer le contrôleur du formulaire d'édition

- Créez le contrôleur pour les différentes routes.
- Testez au fur et à mesure.

TD

Créer la vue du formulaire d'édition

- Placez le template fourni **`editProducts.html`** dans le répertoire **`templates`** et apportez les modifications nécessaires.
- Testez dans votre navigateur.

TD

Utiliser l'héritage dans les vues

- A l'aide de la documentation :

  //flask.palletsprojects.com/en/1.1.x/patterns/templateinheritance

  utilisez l'héritage pour simplifier l'ensemble des vues.