Formation Python/Django

Une formation pratique sur Django, un framework Python très populaire.

Tableau des matieres

Installation de Python

Pour pouvoir utiliser Django, vous avez besoin d'installer Python. Pour l'installer, rendez-vous au site Python.org et suivre les instructions.

Installation de Visual Studio Code

Visual Studio ou VS Code, c'est un éditeur de code simple et facile à utiliser. Pour l'installer, visitez vscode

Mettre en place son projet

Une fois tout est bien installé, les choses pratiques démarrent.

Créer un dossier nommé, projet_multiservice puis entrez dessus.

```
mkdir projet_multiservice && cd projet_multiservice
```

Dans le dossier, nous allons créer notre premier projet. Ne vous inquiétez pas, le processus est simple, Django gère tous les aspects de création de projet.

Juste suivez moi.

Utiliser un gestionnaire de package

Un environnement virtuel te permet de mieux versionner vos applications.

Ici on utilise venv. Le choix est large(virtualenv, pipenv, poetry etc...)

```
python3 -m venv monenv
```

La commande précédente vous permet de créer un environnement virtuel, appelé **monenv**.

Activez le en tapant cette commande:

```
source monenv/bin/activate
```

Ou sous windows

```
monev\Scirpts\activate.bat
```

Installer Django

Une fois que l'environnement est actif, nous procédons à l'installation de Django.

```
pip install django
```

Générer le projet

Cette commande vous permet de générer un nouveau projet Django facilement.

```
django-admin startproject multiservice .
```

Tester votre projet

Lancez le serveur depuis votre terminal ou CMD

```
python manage.py runserver
```

Ouvrez votre navigateur et rendez-vous au http://localhost:8000

Créer votre première application

Sur Django, un projet peut contenir plusieurs applications et une application peut être mise sur différents projets.

Lancez cette commande pour générer une nouvelle application

```
python manage.py startapp product
```

Informez Django votre nouvelle application. Ouvrez le fichier `settings.py` et mettez `product` dans `INSTALLED_APPS`.

```
INSTALLED_APPS = [
   'django.contrib.admin',
   'django.contrib.auth',
   'django.contrib.contenttypes',
   'django.contrib.sessions',
   'django.contrib.messages',
   'django.contrib.staticfiles',

   'product', # nouveau
]
```

Ajoutez votre modèle de données

Ouvrez le fichier `models.py` qui se trouve dans le dossier `product`. Mettez ces informations:

```
# product/models.py
from django.db import models
from django.contrib.auth.models import User


class Product(models.Model):
   name = models.CharField(max_length=60)
   image = models.ImageField(upload_to="product/")
   quantity = models.PositiveIntegerField(default=1)
   price = models.DecimalField(default=0, max_digits=13, decimal_places=2)
   created_at = models.DateTimeField(auto_now=True)
   owner = models.ForeignKey(User, on_delete=models.CASCADE)

   def __str__(self):
       return self.name
```

Une fois le modèle de données est bon, nous pouvons ajouter notre **views**.

Une views en Django, c'est à peuprès un contrôleur dans les autres framework MVC.

Ajouter votre views

Ouvrez le fichier `models.py` qui se trouve dans le dossier `product`. Mettez ces informations:

```
# product/views.py

from transaction.models import Transaction
from django.contrib import messages
from django.db.models.aggregates import Sum
from django.contrib.auth.decorators import login_required
from django.shortcuts import get_object_or_404, redirect, render

from product.forms import NewProductForm

from .models import Product


def about(request):
   return render(request, "product/about.html")
```

```
def contact(request):
    return render(request, "product/contact.html")


@login_required
def index(request):
    owner = request.user
    transaction_spents = Transaction.objects.filter(
        transaction_type=0, product__owner=owner)
    transaction_incomes = Transaction.objects.filter(
        transaction_type=1, product__owner=owner)
    product_count = Product.objects.count()
    expenses = Transaction.objects.filter(
        transaction_type=0, product__owner=owner).aggregate(Sum("price"))["price__sum"]
    incomes = Transaction.objects.filter(
        transaction_type=1, product__owner=owner).aggregate(Sum("price"))["price__sum"]
    sales_figures = incomes - expenses if incomes and expenses else 0
    return render(request, "product/index.html", {"transaction_spents": transaction_spents, "transaction_incomes": transaction_income


# Lister tous les produits/services

@login_required
def all_products(request):
    owner = request.user
    products = Product.objects.filter(owner=owner)
    return render(request, "product/all-products.html", {"products": products})


# Afficher le détail d'un service
@login_required
def product_detail(request, id):
    product = get_object_or_404(Product, pk=id)
    return render(request, "product/product-detail.html", {"product": product})

# Ajouter un nouveau produit/service


@login_required
def add_product(request):
    if request.method == "POST":
        form = NewProductForm(request.POST, request.FILES)
        if form.is_valid():
            product = form.save(commit=False)
            product.owner = request.user
            product.save()
            return redirect("all_products")
        else:
            print(form.errors)
            messages.error(request, f"Erreur : {form.errors}")
            return redirect("add_product")
    else:
        form = NewProductForm()
    return render(request, "product/add-prodcut.html", {"form": form})


# Modifier un produit/service
@login_required
def edit_product(request, id):
    product = get_object_or_404(Product, pk=id)
    if request.method == "POST":
        form = NewProductForm(request.POST, request.FILES, instance=product)
        if form.is_valid():
```

```
            product = form.save(commit=False)
            product.save()
            return redirect("all_products")
        else:
            messages.error(request, f"Erreur : {form.errors}")
            return redirect("edit_product")
    else:
        form = NewProductForm(instance=product)
    return render(request, "product/edit-prodcut.html", {"form": form})

# Supprimer un produit/service


def delete_product(request, id):
    product = get_object_or_404(Product, pk=id)
    if request.method == "POST":
        product.delete()
        return redirect("all_products")
    return render(request, "product/delete-product.html", {"product": product})
```

Nous avons ajouté différentes fonctions pour lister, ajouter, modifier et supprimer un produit.

Vous avez remarqué la présence de `NewProductForm`, c'est un formulaire Django.

Pour ajouter un formulaire, créez un fichier `forms.py` dans le dossier product.

Voici le contenu du fichier `forms.py`.

```
 # prouct/forms.py
from django import forms
from .models import Product


class NewProductForm(forms.ModelForm):
    name = forms.CharField(
        label="Nom du produit",
        widget=forms.TextInput({"class": "form-control"}))
    quantity = forms.CharField(
        label="Quantité en stock",
        widget=forms.NumberInput({"class": "form-control"}))
    logo = forms.ImageField(
        label="Image du produit",
        widget=forms.FileInput({"class": "form-control"}))
    price = forms.CharField(
        label="Prix",
        widget=forms.NumberInput({"class": "form-control"}))

    class Meta:
        model = Product
        fields = ('name', 'quantity', 'price', 'logo',)
```

Il nous reste deux choses: les `urls` et les `templates`.

Ajouter les urls

Comme vous le voyez, depuis le début du projet, nous avons mis toutes les `routes` sur un seul fichier `urls.py`.

Cette méthode est interdite pour les grands projets. La solution est de créer un fichier `urls.py` pour chaque application.

Ajouter un fichier `urls.py` dans le dossier `product`.

Voici le contenu de ce fichier:

```
 # product/urls.py
from django.urls.conf import include
from django.urls import path
from .views import edit_product, index, about, contact, all_products, add_product, product_detail, delete_product


urlpatterns = [
   path("", index, name="index"),
   path("about/", about, name="about"),
   path("contact/", contact, name="contact"),
   path("products/", all_products, name="all_products"),
   path("products/<int:id>/", product_detail, name="product_detail"),
   path("add-product/", add_product, name="add_product"),
   path("products/<int:id>/edit", edit_product, name="edit_product"),
   path("products/<int:id>/delete", delete_product, name="delete_product"),
]
```

Voici le code source du projet au complét, si vous avez des blocages.

Des points à retenir

- Vous avez besoin de lancer la migration pour chaque modification des modèles de données(le fichier `models.py`):

Les commandes: `python manage.py makemigrations` `python manage.py migrate`

- La documentation de Django est votre meilleur ami.
- Vous ne pouvez pas devenir développeur en suivant une formation seulement
- Formez-vous toujours constamment

Me contacter

Je reste disponible 24/7

Whatsapp: +221 76 377 22 60 Email: mstspr1155@gamil.com

Happy Pythonning -:)