D’accord, je comprends que vous travaillez sur un TP qui concerne l’utilisation de Flask, un framework Python pour le développement web. Vous avez déjà installé Python et Flask sur votre machine et vous avez testé un serveur de démonstration. Maintenant, vous voulez créer votre propre application avec deux routes.

Pour la première partie de votre TP, voici un exemple de code pour un serveur simple avec deux routes :

**Python**

from flask import Flask, render\_template

app = Flask(\_\_name\_\_)

@app.route('/')

def hello\_world():

return render\_template('hello.html')

@app.route('/appli')

def welcome():

return render\_template('welcome.html')

if \_\_name\_\_ == '\_\_main\_\_':

app.run(debug=True)

Dans cet exemple, vous devez créer deux fichiers HTML hello.html et welcome.html dans le répertoire /templates. Ces fichiers seront rendus lorsque vous accédez aux routes correspondantes.

Pour la deuxième partie de votre TP, voici un exemple de code pour un serveur qui renvoie un nombre aléatoire et qui peut récupérer une valeur saisie dans un formulaire :

**Python**

import random

from flask import Flask, request, render\_template

app = Flask(\_\_name\_\_)

random\_number = random.randint(1, 100)

@app.route('/')

def home():

return str(random\_number)

@app.route('/form', methods=['GET', 'POST'])

def form():

if request.method == 'POST':

value = request.form.get('value')

return f'Vous avez saisi : {value}'

return render\_template('form.html')

if \_\_name\_\_ == '\_\_main\_\_':

app.run(debug=True)

Dans cet exemple, vous devez créer un fichier HTML form.html dans le répertoire /templates qui contient un formulaire pour saisir une valeur. Cette valeur sera affichée lorsque vous soumettez le formulaire.

N’oubliez pas de remplacer les noms des fichiers HTML par ceux que vous avez créés. J’espère que cela vous aidera dans votre TP. Si vous avez d’autres questions ou besoin d’aide supplémentaire, n’hésitez pas à me le faire savoir ! 😊

Vous

je voudrais que tu m'explique de la manière la plus détaillé et compréhensible comme réaliser cette partie : " Écrire un deuxième endpoint capable de récupérer une valeur saisie dans un formulaire, et qui l’affiche" en m'indiquant le code à faire pour créer le formulaire, et le code python

Copilot

D’accord, je vais vous expliquer comment créer un formulaire HTML et comment récupérer la valeur saisie dans ce formulaire en utilisant Flask.

Tout d’abord, vous devez créer un fichier HTML pour votre formulaire. Vous pouvez le nommer form.html et le placer dans le répertoire /templates de votre application Flask. Voici un exemple simple de formulaire HTML :

**HTML**

<!DOCTYPE html>

<html>

<body>

<h2>Mon formulaire</h2>

<form action="/form" method="post">

<label for="fname">Prénom:</label><br>

<input type="text" id="fname" name="fname"><br>

<input type="submit" value="Soumettre">

</form>

</body>

</html>

Dans cet exemple, le formulaire contient un champ de texte où l’utilisateur peut saisir son prénom. Lorsque l’utilisateur soumet le formulaire, les données du formulaire sont envoyées à l’URL /form en utilisant la méthode POST.

Ensuite, vous devez créer une route dans votre application Flask pour gérer les requêtes POST à l’URL /form. Voici comment vous pouvez le faire :

**Python**

from flask import Flask, request, render\_template

app = Flask(\_\_name\_\_)

@app.route('/form', methods=['GET', 'POST'])

def form():

if request.method == 'POST':

prenom = request.form.get('fname')

return f'Vous avez saisi : {prenom}'

return render\_template('form.html')

if \_\_name\_\_ == '\_\_main\_\_':

app.run(debug=True)

Dans cet exemple, lorsque l’utilisateur soumet le formulaire, la route /form récupère la valeur du prénom à partir des données du formulaire en utilisant request.form.get('fname'). Ensuite, elle renvoie une réponse qui affiche la valeur saisie.

J’espère que cela vous aide à comprendre comment créer un formulaire HTML et comment récupérer la valeur saisie dans ce formulaire en utilisant Flask. Si vous avez d’autres questions, n’hésitez pas à me les poser ! 😊

Vous

Écrire un programme qui tire un nombre entre 1 et 100, puis l’utilisateur doit le deviner (il propose une valeur ,et votre programme lui répond « c’est plus » ou « c’est moins » (ou « c’est gagné », bien sur!!)

Copilot

D’accord, voici un exemple de programme Python qui fait exactement cela. Ce programme génère un nombre aléatoire entre 1 et 100, puis demande à l’utilisateur de le deviner. En fonction de la réponse de l’utilisateur, le programme indique si le nombre à deviner est plus grand, plus petit ou si l’utilisateur a gagné.

**Python**

import random

def jeu\_devine\_le\_nombre():

nombre\_a\_deviner = random.randint(1, 100)

proposition = int(input("Devinez le nombre entre 1 et 100: "))

while proposition != nombre\_a\_deviner:

if proposition < nombre\_a\_deviner:

proposition = int(input("C'est plus. Essayez encore: "))

else:

proposition = int(input("C'est moins. Essayez encore: "))

print("C'est gagné ! Le nombre était bien ", nombre\_a\_deviner)

jeu\_devine\_le\_nombre()

Dans ce programme, la fonction jeu\_devine\_le\_nombre génère un nombre aléatoire entre 1 et 100. Ensuite, elle entre dans une boucle où elle demande à l’utilisateur de deviner le nombre. Si la proposition de l’utilisateur est inférieure au nombre à deviner, le programme affiche “C’est plus”. Si la proposition est supérieure, il affiche “C’est moins”. La boucle continue jusqu’à ce que l’utilisateur devine correctement le nombre, auquel cas le programme affiche “C’est gagné !” et termine.

Veuillez noter que ce programme est destiné à être exécuté dans un terminal ou un environnement similaire qui prend en charge l’entrée de l’utilisateur via la fonction input. Si vous souhaitez utiliser ce programme dans un environnement différent, comme une application web Flask, vous devrez adapter le code pour qu’il fonctionne avec votre environnement spécifique. Si vous avez besoin d’aide pour cela, n’hésitez pas à me le faire savoir ! 😊

Vous

TP 2 : Flask Application complete Un rendu par template Réalisez un nouveau serveur qui affiche une page d’accueil. Celle ci pointera vers un formulaire. Ce formulaire vous permettra de saisir la fiche pour un jeu vidéo : nom, plateforme, prix, description. Réalisez ce formulaire grâce à un template « Saisie » Créez un template « Liste » capable d’afficher en présentant de façon élégante les données saisies dans le formulaire. Il devra aussi permettre de revenir à la page d’accueil. Dans votre programme en Python, nous voudrions maintenant stocker les informations saisies. En attendant mieux, vous pouvez les mettre dans une liste Python dans le back. Cette liste contiendra tous les jeux. Chaque jeu est un ensemble d’information (nom, prix, description). Il faudra donc les représenter (le mieux est d’utiliser un dictionnaire) : (voir l’exemple films dans le support de cours) Ajouter maintenant un tableau dans votre template « Liste » : Ce tableau contiendra la description de plusieurs jeux vidéos. Nous voulons que le formulaire permette de remplir le tableau (donc il doit avoir les mêmes colonnes que le formulaire : nom, plateforme, prix, descriptions). Votre template doit maintenant afficher la liste de tous les jeux connus par le système. Il doit permettre de revenir au menu, ou de lancer le formulaire de saisie. Vérifiez que vous pouvez bien ajouter des jeux, et que ceux-ci apparaissent bien dans votre liste Ajoutez maintenant un bouton permettant de supprimer un jeu Ajoutez ensuite un bouton qui permet d’éditer chaque jeu, afin de pouvoir modifier son prix, son nom, sa description.

Copilot

D’accord, je vais vous expliquer comment réaliser cette application Flask étape par étape.

1. **Création du serveur Flask et de la page d’accueil**

Tout d’abord, vous devez créer un nouveau serveur Flask. Vous pouvez le faire en créant un nouveau fichier Python (par exemple app.py) et en y ajoutant le code suivant :

**Python**

from flask import Flask, render\_template

app = Flask(\_\_name\_\_)

@app.route('/')

def home():

return render\_template('home.html')

if \_\_name\_\_ == '\_\_main\_\_':

app.run(debug=True)

Ce code crée un nouveau serveur Flask avec une seule route (/) qui renvoie une page d’accueil. La page d’accueil est générée à partir d’un template appelé home.html que vous devez créer dans le répertoire /templates de votre application.

1. **Création du formulaire**

Ensuite, vous devez créer un formulaire pour saisir les informations d’un jeu vidéo. Vous pouvez le faire en créant un nouveau template (par exemple saisie.html) dans le répertoire /templates de votre application. Voici un exemple de formulaire HTML :

**HTML**

<!DOCTYPE html>

<html>

<body>

<h2>Saisie d'un jeu vidéo</h2>

<form action="/ajouter\_jeu" method="post">

<label for="nom">Nom:</label><br>

<input type="text" id="nom" name="nom"><br>

<label for="plateforme">Plateforme:</label><br>

<input type="text" id="plateforme" name="plateforme"><br>

<label for="prix">Prix:</label><br>

<input type="text" id="prix" name="prix"><br>

<label for="description">Description:</label><br>

<input type="text" id="description" name="description"><br>

<input type="submit" value="Soumettre">

</form>

</body>

</html>

Ce formulaire contient quatre champs de texte pour saisir le nom, la plateforme, le prix et la description d’un jeu vidéo. Lorsque l’utilisateur soumet le formulaire, les données du formulaire sont envoyées à l’URL /ajouter\_jeu en utilisant la méthode POST.

1. **Gestion de la soumission du formulaire**

Vous devez ensuite ajouter une nouvelle route à votre serveur Flask pour gérer la soumission du formulaire. Vous pouvez le faire en ajoutant le code suivant à votre fichier app.py :

**Python**

from flask import request

jeux = []

@app.route('/ajouter\_jeu', methods=['POST'])

def ajouter\_jeu():

nom = request.form.get('nom')

plateforme = request.form.get('plateforme')

prix = request.form.get('prix')

description = request.form.get('description')

jeux.append({'nom': nom, 'plateforme': plateforme, 'prix': prix, 'description': description})

return render\_template('liste.html', jeux=jeux)

Ce code ajoute une nouvelle route (/ajouter\_jeu) qui gère les requêtes POST. Lorsqu’une requête POST est reçue, le code récupère les données du formulaire, crée un nouveau dictionnaire avec ces données, ajoute le dictionnaire à la liste jeux, puis renvoie une page générée à partir du template liste.html.

1. **Affichage de la liste des jeux**

Enfin, vous devez créer un template pour afficher la liste des jeux. Vous pouvez le faire en créant un nouveau template (par exemple liste.html) dans le répertoire /templates de votre application. Voici un exemple de template qui affiche une liste de jeux :

**HTML**

<!DOCTYPE html>

<html>

<body>

<h2>Liste des jeux</h2>

<table>

<tr>

<th>Nom</th>

<th>Plateforme</th>

<th>Prix</th>

<th>Description</th>

</tr>

{% for jeu in jeux %}

<tr>

<td>{{ jeu.nom }}</td>

<td>{{ jeu.plateforme }}</td>

<td>{{ jeu.prix }}</td>

<td>{{ jeu.description }}</td>

</tr>

{% endfor %}

</table>

<a href="/">Retour à la page d'accueil</a>

</body>

</html>

Ce template crée un tableau HTML pour afficher la liste des jeux. Chaque ligne du tableau correspond à un jeu et contient le nom, la plateforme, le prix et la description du jeu. Le template utilise la syntaxe de templating de Jinja pour insérer les données des jeux dans le tableau.

J’espère que cela vous aide à comprendre comment réaliser cette application Flask. Si vous avez d’autres questions, n’hésitez pas à me les poser ! 😊

**En savoir plus**

[1github.com](https://github.com/paladium/computer-science-course/tree/c1699ba2221a5d09476ece334c676985a50e85a1/09-Server%2F2-Flask-Templates.md)

Vous

à quoi sert le .append ?

Copilot

La méthode .append() en Python est utilisée pour ajouter un élément à la fin d’une liste. Par exemple, si vous avez une liste ma\_liste = [1, 2, 3] et que vous exécutez ma\_liste.append(4), la liste deviendra [1, 2, 3, 4].

Voici un exemple de code qui illustre cela :

**Python**

ma\_liste = [1, 2, 3]

print(ma\_liste) # Affiche : [1, 2, 3]

ma\_liste.append(4)

print(ma\_liste) # Affiche : [1, 2, 3, 4]

Dans le contexte de votre TP, la méthode .append() est utilisée pour ajouter un nouveau jeu à la liste des jeux. Chaque jeu est représenté par un dictionnaire contenant ses informations (nom, plateforme, prix, description), et la méthode .append() ajoute ce dictionnaire à la fin de la liste des jeux. Cela permet de stocker toutes les informations des jeux dans une seule structure de données. 😊