python™

serveur web

Samuel SORIN
contact@samuelsorin.fr

Serveur web

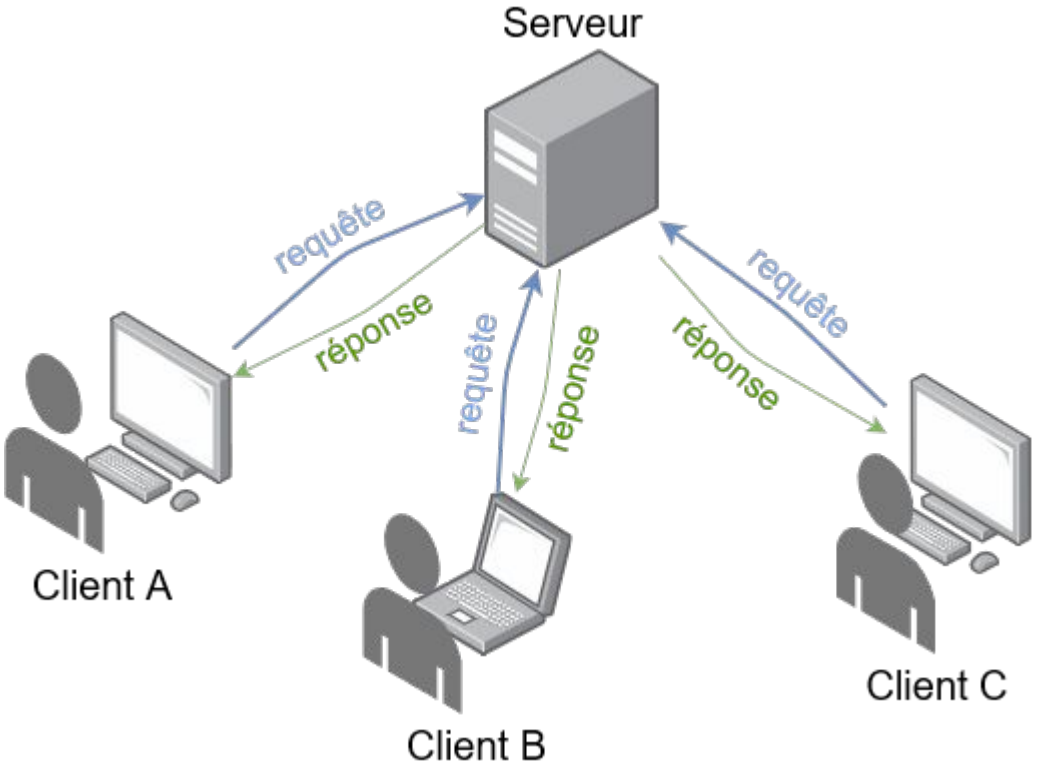
Serveur web - un peu de théorie : client / serveur

Les langages de programmation web sont séparés en 2 catégories :

- les technologies **côté client** : elles sont interprétées/exécutées par la machine cliente (html, css, js ...)

- les technologies **côté serveur** : elles sont interprétées/exécutées par le serveur (machine distante)
(Python, Java, PHP, ...)

Serveur web - un peu de théorie : client / serveur



Serveur web - un peu de théorie : client / serveur

Coté client

- le contenu
- la mise en forme
- les animations
- les interactions
- On récupère des infos sur l'utilisateur (mouvement du pointeur, géoposition, etc.)

Serveur web - un peu de théorie : client / serveur

Coté client

Avantage :

- fonctionne sur tous les navigateurs. Donc disponible partout, par tout le monde
- pas de configuration
- pas besoin de serveur

Inconvénient :

- aucune persistance des données
- ne surtout pas y mettre de la donnée sensible (c'est complètement ouvert)
- c'est du code qui est interprété par tous les clients connectés au site
- par sécurité, les navigateurs bloquent pas mal de choses (CORS, etc.)

Serveur web - un peu de théorie : client / serveur

Coté serveur

- permet de traiter les données (trie, calcul, encodage …)
- Définir des modèles de données
- Stocker les données (bases de données, json, xml...) pour les rendre persistantes
- on peut contrôler comment les fichiers et les contenus sont envoyés au client
- logique métier des applications
- etc, etc

Serveur web - un peu de théorie : client / serveur

Coté serveur

avantages :

- le code interprété par le serveur n'est pas visible par le client
- vous pouvez exécuter de grosses opérations de manière transparente

Inconvénients :

- vous ne pouvez pas savoir ce que l'utilisateur fait (vous voyez juste les requêtes)
- Installation et mise en place d’un serveur et de l’interpréteur correspondant au langage utilisé

Serveur web - servir des page html

Dans un terminal, placez vous dans le dossier contenant les pages html à servir, puis :

sous linux :

```
python -m http.server 8088
```

sous windows :

```
py -m http.server 8088
```

Serveur web - Créer un serveur web

Commencez par créer un fichier http_server.py et copiez le code suivant

```
import http.server

PORT = 8888                                          # port sur lequel le serveur sera disponible
server_address = ("", PORT)                          # adresse sur laquelle le serveur sera disponible
                                                     # exemple : www.tonsite.com:8080


server = http.server.HTTPServer                      # instancie le serveur
handler = http.server.CGIHTTPRequestHandler          # ajoute la gestion du GET et HEAD au serveur
handler.cgi_directories = ["/"]                      # dossier de travail, où sont placés tous les scripts.py
                                                     # (il s'agit de l'adresse relative)
print("Serveur actif sur le port :", PORT)           # message pour dire que le serveur est actif

httpd = server(server_address, handler)              # démarrage du serveur
httpd.serve_forever()                                # le serveur "tourne" tant qu'on ne l'arrête pas
```

Serveur web - Créer une page web

Dans le même répertoire où se trouve le fichier http_server.py, créer un dossier www et placez-y vos script python. par exemple hello.py avec le code suivant :

```
#!/usr/bin/env python3                          # on précise l'interpréteur pour le serveur
# coding: utf-8                                 # codage du script - pour pouvoir utiliser les accents
import cgi                                      # interface cgi qui permet à Python et au client de "dialoguer"

print("Content-type: text/html; charset=utf-8\n")   # on déclare le type de contenu que l'on va généré.
                                                    # Cette information sera directement utilisé par le navigateur du client


html = """                                      # on génère le code HTML à envoyer au navigateur du client
  <!DOCTYPE html>                               # Cette information sera directement utilisé par le navigateur du client
  <head>
     <title>Mon programme</title>
  </head>
  <body>
     <h1>Hello World!</h1>
  </body>
  </html>
"""

print(html)                                     # on "envoie" le code généré au navigateur client
```

Serveur web - Créer une page web

À partir d'un terminal, lancer votre serveur http_serveur.py.

```
python3 http_serveur.py
ou
py http_serveur.py
```

Dans votre navigateur web, aller à l'adresse : http://127.0.0.1:8080/www/*monfichier*.py

Attention,
- pour lancer votre serveur, vous devez obligatoirement vous placez à la racine du projet avant de lancer la commande.
- Vos fichiers .py doivent avoir la permission d'être "exécuté comme un programme"

Serveur web - utiliser un formulaire

```
[...]
import cgi

form = cgi.FieldStorage()                    # on récupère le formulaire
name = form.getvalue("name")                 # on récupère la valeur “name” présente dans le formulaire

html = f"""
[...]
      <h1>Hello { name } !</h1>

      <form action="/form.py" method="post">
         <input type="text" name="name" value="" placeholder="Votre nom"/>
         <input type="submit" value="Envoyer information au serveur">
      </form>
[...]
"""

[...]
```

Serveur web - utiles

Afficher les erreurs Python sur votre page web (debug) :

```
import cgitb
cgitb.enable()
```

Afficher les variables d'environnement :

Affiche toutes les informations concernant votre serveur web / page en cours

```
cgi.test()
```

Serveur web - exercice 1

a - Créer un serveur web et afficher une page html

b - Créer un nouveau serveur web, acceptant les scripts python pour afficher “Hello World” dans une page web.

c - Ajouter des paramètres GET pour modifier le message “Hello World” par “Hello *[prenom] [nom]*”, où [prenom] [nom] sont des paramètres

d - Utiliser un formulaire html pour envoyer les paramètres GET

e - Transformer les paramètres GET en paramètres POST