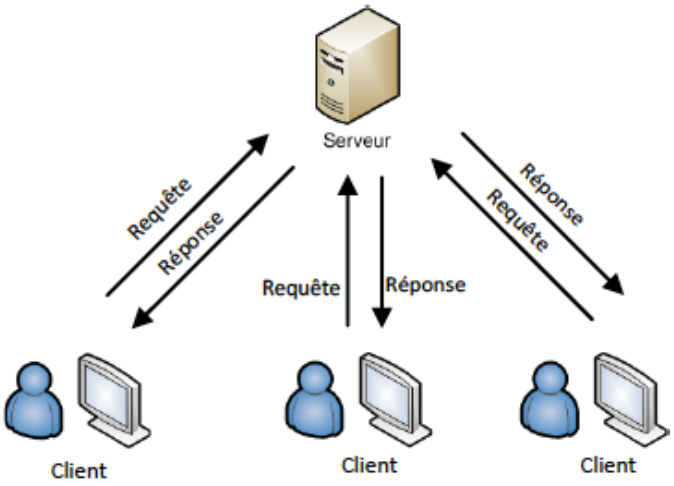
1. Introduction

Le web a été conçu au départ pour permettre à de multiples utilisateurs de pouvoir consulter le même document au même moment. Cela a été rendu possible grâce à l'architecture client / serveur. Dans le monde du web, ce mode de communication se fait entre deux programmes, le client est représenté par le navigateur web. Le protocole de communication utilisé est le HTTP (*HyperText Transfer Protocol*).



2. Front-end (coté client)

On désigne par front-end ce qui se passe principalement dans le navigateur. L'ensemble des langages qui lui sont associés sont composés du trio HTML + CSS + JavaScript. C'est le domaine de prédilection des web designer ou des développeurs front-end.

Comme vous l'avez vu dans la fiche JavaScript, certaines vérifications ou interactions sont faites du coté de cette partie cliente.

3. Back-end (coté serveur)

La partie serveur est là pour répondre aux demandent des clients, comme quand un formulaire est validé coté client (*submit*). Plutôt que de simplement transférer le contenu d'un fichier (ex : une page html, ou une image), à la réception d'une requête type formulaire, le serveur va appeler un programme ou un script. Les technologies utilisées sont très variées, citons les plus connues comme **PHP**, **Python**, **.NET** ou **java**.

Bien souvent, ces programmes ou scripts sont amenés à interroger une base de données pour construire la réponse qui va être apportée au client. C'est alors qu'on peut parler de pages web dynamiques, car construites en fonction de la requête du client.

Au-dessus du système d'exploitation (*linux*, *windows* …) faisant fonctionner la machine serveur, le cœur du back-end s'appuie sur un logiciel serveur HTTP, vous trouverez les plus utilisés ci-dessous :

4. HTTP et HTTPS : Le dialogue entre client et serveur :

a. http

Le protocole **HTTP** (*HyperText Transfer Protocol*) est l'artisan de la communication client-serveur. C'est un protocole de haut niveau, il se trouve au-dessus des couches formées par les protocoles **TCP** et **IP**. Ce protocole permet les échanges entre un navigateur et un serveur connecté à Internet. En plus des requêtes permettant d'atteindre une ressource (*URL*), il assure aussi la transmission des informations saisies dans un formulaire.

b. https

En cas de besoin d'échanges de données sécurisés, une couche est ajoutée au protocole http, il s'agit de la couche SSL (Secure Socket Layer). Celle-ci va assurer le chiffrement de bout en bout des échanges afin de garantir leur confidentialité.

Depuis les révélations faites par Edward Snowden sur la NSA et son espionnage à grande échelle, l'utilisation du protocole en version https a tendance à se systématiser sur les sites web partout dans le monde. Cette action a pour but d'aider les utilisateurs à garder leur vie privée.

5. Les méthodes GET et POST

Il existe 2 méthodes différentes pour transmettre les données d'un formulaire : les méthodes **GET** et **POST**. Le choix de la méthode dépend de la façon dont les données sont reçues, de la taille et la nature des données.

La méthode GET ajoute les données à l'URL :

Dans un formulaire, elle est spécifiée ainsi:

```
<form method="get" action="page.html">
</form>
```

Avec cette méthode, les données du formulaire seront encodées dans une URL et sera récupérable par le serveur dans la variable d'environnement QUERY_STRING. Celle-ci est composée du nom de la page ou du script à charger avec les données de formulaire empaquetée dans une chaîne.

Les données sont séparées de l'adresse de la page pas le caractère **?** et entre elles par le caractère **&**.

Ainsi si on envoie à page.html les valeurs "couleur bleu" et "forme rectangle", l'URL construite par le navigateur sera :

```
https://www.mapage.fr/page.html?couleur=bleu&forme=carre
```

- GET est utilisé quand la requête ne cause pas de changement dans les données stockées coté serveur. Donc on va la privilégier quand on opère une simple lecture d'informations sur le serveur.
- La taille d'une URL est limitée à par le serveur, souvent un peu plus de 2000 caractères.s
- Attention, lorsqu'on utilise le bouton retour du navigateur, les requêtes GET sont exécutées à nouveau.
- Les données de formulaire doivent être uniquement des codes ASCII.

La méthode POST n'a pas de taille limite

Dans un formulaire, elle est spécifiée ainsi :

```
<form method="post" action="page.php">
</form>
```

Avec la méthode POST, les paramètres sont transmis au programme externe dans son entrée standard et sont invisibles dans l'URL.

- Elle est indispensable pour des codes non ASCII ou des données de taille importante
- Elle est recommandée pour modifier les données sur le serveur

⇨ A vous de jouer !

En python, créer 2 scripts, un client et un serveur à l'aide des 2 programmes ci-après, exécuter les 2 scripts simultanément, que constatez vous ?

Partie Serveur :

```
1  from socket import socket, AF_INET, SOCK_STREAM
2
3  s = socket(AF_INET, SOCK_STREAM)
4  s.bind(("",13450)) #le serveur est à l'écoute sur le port 13450
5  s.listen(1)
6  test = True
7  while test:
8      connexion, adresse = s.accept()
9      print ("connexion de", adresse)
10     req = connexion.recv(1024).decode()
11     if req != "":
12         print(req)
13         connexion.send(b"OK")
14         if req == "Fin":
15             test = False
16 connexion.close()
17 s.close()
```

Partie Client :

```python
from socket import socket, AF_INET, SOCK_STREAM

hote = "localhost"
port = 13450

s = socket(AF_INET, SOCK_STREAM)
s.connect((hote,port))
print("Connexion sur le port", port)
s.send(b"Bonjour, je m'appelle Toto !")
print(s.recv(1024).decode())
s.close()

s = socket(AF_INET, SOCK_STREAM)
s.connect((hote,port))
print("Connexion sur le port", port)
s.send(b"Fin")
print(s.recv(1024).decode())
print("Fin")
s.close()
```

COMMENT CELA FONCTIONNE-T-IL ? :

Les programmes ci-dessus font appelle à un **socket**. Un socket est une interface de connexion gérée par le système d'exploitation et permettant de se connecter à un réseau TCP par exemple. Les sockets sont implémentés de base dans certains langages comme **Java** ou le **langage C**, cependant en **Python**, il faut importer le module socket pour pouvoir les exploiter.

Le protocole TCP utilise des adresses IP. Pour chaque adresse IP, un numéro de port est associé à chaque extrémité de la connexion TCP, coté client et coté serveur. L'entité adresse IP + port constitue un identifiant unique permettant d'accéder à un service spécifique sur Internet.

⇨ **A vous de jouer 1 :**

A l'aide du programme ci-dessous créez votre serveur web !!

Essayez de vous connectez avec plusieurs navigateurs en tapant dans la barre d'adresse localhost :13450, qu'observez-vous ?

```
from socket import socket, AF_INET, SOCK_STREAM

def html():
    rep = b"""HTTP/1.1 200 OK
    host : mon site local
    Content-Type: text/html\n
    <!DOCTYPE html>
    <html>
    <head>
    <title> Ma page</title>
    </head>
    <body>
    <h1>Bonjour</h1>
    <h2>Le test est concluant !</h2>
    </body>
    </html>"""
    return rep

s = socket(AF_INET, SOCK_STREAM)
s.bind(("",13450))

s.listen(5)
while True:
    connexion, adresse = s.accept()
    print ("connexion de", adresse)
    req = connexion.recv(1024).decode()
    if req != "":
        print(req)
        connexion.send(html())
    connexion.close()

print("Fin")
s.close()
```

⇨ **A vous de jouer 2 : Ajoutons une image**

Un peu d'aide, pour ajouter une image il y a plusieurs modifications à faire :

- Ajouter le code d'insertion d'image dans le html (à chercher sur internet)
- Recharger la page, que constatez-vous ?

Pour que cela fonctionne nous devons :

- Préparer l'envoi de l'image en mode binaire, pour cela nous allons créer une fonction qui va permettre d'ouvrir le fichier image et de le lire en binaire puis qui va retourner ses données. (NB : pour ouvrir un fichier en binaire on utilise le paramètre « rb » de la fonction open())

- Gérer la demande de l'image en modifiant la boucle de surveillance serveur. On rajoute donc les lignes suivantes :

```
if req != "":
    print(req)
    if 'BTS' in req:
        connexion.send('HTTP/1.1 200 OK\r\n'.encode())
        connexion.send("Content-Type: image/jpeg\r\n".encode())
        connexion.send("Accept-Ranges: bytes\r\n\r\n".encode())
        connexion.send(image())
    else:
        connexion.send(html())
```

La fonction image étant celle qui permet de récupérer le contenu de l'image.

Faites les modifications jusqu'à réussir à afficher l'image dans le navigateur. (Pensez à ajouter l'image dans votre dossier de travail !)

⇨ **A vous de jouer 3 : ajoutons du style !**

Le css permet de styliser les pages web

- A partir d'un tuto d'internet, créer un fichier css simple pour styliser la page web
- En refaisant les même étapes que pour l'image, faites en sorte que le css soit transmis au navigateur et s'affiche.