

4.4

Requête HTTP

NSI 1ÈRE - JB DUTHOIT

4.4.1 Côté client, côté serveur

L'objectif est ici de voir comment circulent les informations sur le réseau internet.

Imaginons un client (un ordinateur personnel par exemple) qui demande avec son navigateur une page Web. Il s'agit d'une **requête**. Un serveur (qui est un ordinateur aussi) répond aux demandes lorsqu'il est interrogé.

La demande et la réponse se font grâce à un protocole de communication, le protocole **HTTP**.

Nous allons ici étudier cette communication.

On peut ainsi distinguer :

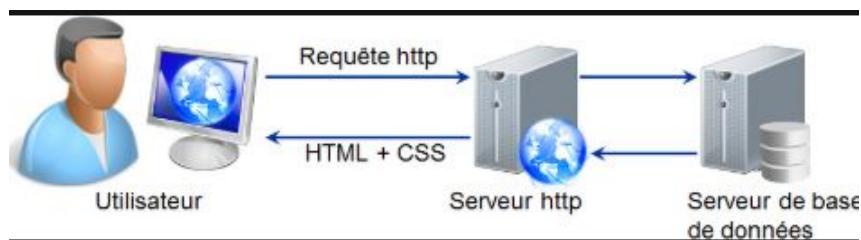
- Le côté **client** ou front-end :

- Il est centré sur le navigateur, et il est constitué par de pages HTML, CSS et du code JS.
- Les métiers concernés sont par exemple web designer, intégrateur CSS, ou développeur front-end.
- certaines vérifications de formulaires sont faites dans le navigateur via des attributs HTML spécialisés, comme pattern (ou avec du code JS).

- Le côté **serveur** ou back-end :

- Il assure la réponse à une demande faite lors de la soumission du formulaire par validation (submit ou avec un appel ajax en JS.)
- On utilise pour cela plusieurs outils et langage : PHP, Pythonet les bases de données relationnelles (programme de terminale).
- Ces programmes s'exécutent le plus souvent sur des serveurs distants, sur internet.
- Des vérifications de données sont aussi réalisées sur le serveur, notamment pour des questions de sécurité.

Protocole HTTP



C'est donc le rôle de protocole HTTP que d'assurer le dialogue entre serveur et client. **HTTP** signifie **HyperText Transfer Protocol**.

Remarque

Pour davantage de sécurité, on ajoute une couche **SSL** (Secure Socket Layer) à HTTP, lequel devient alors **HTTPS**. Cela assure le chiffrement des données et donc la confidentialité des échanges.

On parle d'encapsulation. La réponse contient les mêmes informations avec en plus à la fin l'adresse IP demandé (4 octets)

4.4.2 Les requêtes clients HHTTP

La syntaxe d'une requête client est toujours la même :

Méthode

En-tête de la requête

Corps de la requête

Par exemple, on peut avoir la requête client suivante

```
GET/fichier1 HTTP/1.1  
Host : www.monsite.fr  
Connection : Close
```

Il existe plusieurs méthodes pour les requêtes : la méthode **POST**, la méthode **GET** et la méthode **HEAD**.

4.4.3 Les réponses du serveur

En réponse, le serveur peut envoyer :

HTTP/1.1 200 OK

suivi par tout le code HTML de la page.

Le code 200 signifie que tout fonctionne correctement, et le code 404 correspond à l'erreur "page non trouvée".

Le navigateur interprète ensuite les fichiers et affiche la page Web.

A la lecture du code HTML, il peut effectuer d'autres requêtes, comme par exemple la demande de lire un fichier CSS : GET/styles/index.css HTTP/1.1, avec en réponse HTTP/1.1 200 OK (text.css) et le contenu du fichier CSS.

Il peut s'agir aussi d'images : GET/img/png/menu.png HTTP/1.1, avec une réponse HTTP/1.1 200 OK (PNG).

Le navigateur affiche un à un les éléments en fonctions des réponses reçues.

Un site peut être statique, et alors c'est la même page qui est envoyé à tous les clients qui le demandent.

Un site peut être dynamique , et dans ce cas, une page différente peut être conçue pour chaque client.

Le navigateur enregistre des copies pages consultées sur le disque dur. Cela permet un gain de temps si une page statique est demandée à nouveau.

4.4.4 Illustration avec un programme Python

Voici un programme Python nommé `serveur.py` qui écoute sur un port et renvoie une page HTML.

```
from socket import socket, AF_INET, SOCK_STREAM

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

s = socket(AF_INET, SOCK_STREAM)
s.bind(("" ,13450)) # le programme python \'ecoute sur le port
13450

s.listen(5) # 5 connexions maximales
while True:
    connexion, adresse = s.accept()
    print('Connexion de',adresse)
    req = connexion.recv(1024).decode()
    if req != "":
        print(req)
        connexion.send(reponse)
    connexion.close()
```

Il faut exécuter ce programme et le curseur de l'interpréteur python clignote (il est en attente). Sur un navigateur, et sur le même ordinateur, nous écrivons dans la barre d'adresse 127.0.0.1 :13450.

La page Web s'affiche alors !

4.4.5 Précisions sur la méthodes GET et POST

La méthode GET :

- Il s'agit de la méthode la plus fréquente.
- Une requête GET n'a aucun effet sur la ressource.
- Avec cette méthode, les données du formulaires seront encodées dans une URL construite par le navigateur du client et transmise au serveur.
- C'est ce que fait par exemple Google⁶ lors d'une recherche web.

- La longueur de l'URL est parfois limitée ; cette méthode fonctionne donc avec des formulaires de taille raisonnable.
- Avec cette méthode, les données sont visibles dans l'URL ; elle n'est donc pas adaptée pour des données sensibles.

La méthode POST

- Les paramètres sont transmis au programme externe et sont invisibles dans l'URL.
- C'est une méthode plus compliquée que la méthode GET

4.4.6 Exemple de formulaire HTML, avec la méthode GET ,et une action du type HTML.

Avant d'exécuter ce fichier HTML, il est nécessaire d'exécuter le fichier serveur.py .

Voici le code du petit formulaire :

```

1 <html>
2     <head>
3         <title> Formulaire </title>
4     </head>
5     <body>
6         <p> Voici un formulaire </p>
7         <form method="GET" action = "http://127.0.0.1:13450">
8             <p>
9                 <label for="nom">Nom</label>
10                <input type="text" name="utilisateur" id="nom" autofocus/>
11            </p>
12            <input type="submit" value = "Envoyer" />
13        </form>
14    </body>
15 </html>
```

Le formulaire HTML et le champ renseigné

Voici un formulaire

Nom

Le retour du "serveur" et le champ renseigné se retrouve en URL

127.0.0.1:13450/?utilisateur=JBaptiste+Duthoit

4.4.7 Exemple de formulaire avec la méthode POST ,et une action du type HTML.

Avant d'exécuter ce fichier HTML, il est nécessaire d'exécuter le fichier serveur.py .

Voici le code du formulaire :

```

1 <html>
2     <head>
3         <title> Formulaire </title>
4     </head>
```

```
5      <body>
6          <p> Voici un formulaire </p>
7          <form method="POST" action = "http://127.0.0.1:13450">
8              <p>
9                  <label for="nom">Nom</label>
10                 <input type="text" name="utilisateur" id="nom" autofocus/>
11             </p>
12             <input type="submit" value = "Envoyer" />
13         </form>
14     </body>
15 </html>
```

Pour le même champ saisi, on ne retrouve plus l'information dans l'URL



Les données sont insérées dans le corps de la requête

```
utilisateur=JBaptiste+Duthoit
Connexion de ('127.0.0.1', 61563)
GET /favicon.ico HTTP/1.1
Host: 127.0.0.1:13450
Connection: keep-alive
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537
.36 (KHTML, like Gecko) Chrome/80.0.3987.149 Safari/537.36
Sec-Fetch-Dest: image
Accept: image/webp,image/apng,image/*,*/*;q=0.8
Sec-Fetch-Site: same-origin
Sec-Fetch-Mode: no-cors
Referer: http://127.0.0.1:13450/
Accept-Encoding: gzip, deflate, br
Accept-Language: fr-FR,fr;q=0.9,en-US;q=0.8,en;q=0.7
```