INTERACTIONS CLIENT-SERVEUR SUR LE WEB

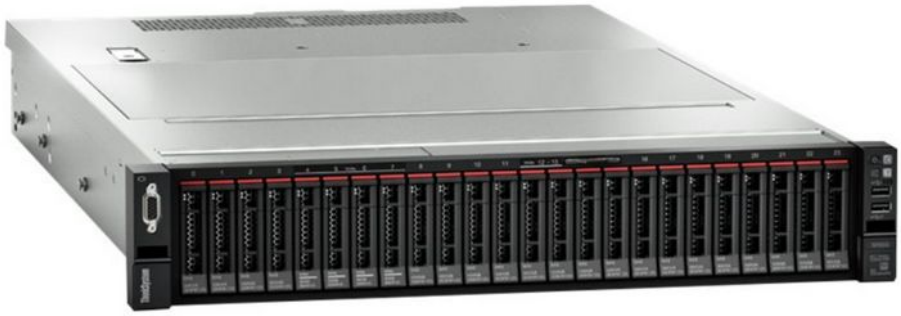
Après avoir étudié comment écrire des pages web, intéressons nous maintenant aux mécanismes qui permettent à un internaute de les consulter.

I) Le modèle client-serveur

1) Introduction

Sur internet, la plupart des échanges de données fonctionnent sur le modèle **client-serveur**.
Un **client** (par exemple l'ordinateur d'un internaute) fait une requête à un **serveur** (souvent situé dans un data center), puis ce dernier lui répond en lui envoyant les données demandées (par exemple un mail, une page web, une vidéo,...).

Selon le contexte, les termes **client** et **serveur** désignent soit des ordinateurs, soit les logiciels tournant sur ces ordinateurs et permettant la communication.



Exemple de serveur utilisé dans un data center

2) Dans le cas du web :

	Ordinateur	Logiciels	Langages
Client	Ordinateurs des internautes	Navigateurs des internautes : Chrome, Firefox, Edge, Safari, …	HTML, CSS, Javascript
Serveur	Ordinateurs spécialisés allumés 24h/24	Serveurs web : Apache, Nginx, IIS,…	Php, Python, Java,...

3) Pages web « statiques » et pages web « dynamiques »

On a coutume de distinguer les pages web **statiques** (qui sont enregistrées « telles quelles » sur les disques durs des serveurs Web) et les pages web **dynamiques** (dont le code HTML-CSS-JavaScript est généré par le serveur au moment où l'internaute veut les consulter).
Par exemple, ce site Web est statique alors que le site de Météo France est dynamique car la météo change tout le temps et n'est pas la même selon les lieux !

Dans le cas d'une page web dynamique, le serveur va « calculer » le code HTML-CSS-JavaScript à envoyer au client à l'aide de scripts écrits dans langages de programmation comme le Php, Python ou Java. Ces langages serveurs interagissent souvent avec une base de donnée.

Remarque : Un code Javascript étant exécuté sur le navigateur du client, il ne peut agir que sur la page web en cours, alors qu'un code Php est exécuté sur le serveur et peut modifier ce qui est stocké sur le serveur.

II) Le protocole HTTP

Un navigateur et un serveur web communiquent en utilisant le protocole HTTP (ou HTTPS) qui est un protocole de la couche [ ].

1) Requêtes HTTP

Supposons que l'on veuille consulter le site http://tfontanet.free.fr.

1) L'internaute tape l'adresse du site dans la barre d'adresse du navigateur ou clique sur un lien.
2) Le navigateur rédige une **requête HTTP** pour demander la page d'accueil du site au serveur qui héberge le domaine tfontanet.free.fr.
3) Le serveur reçoit la requête et y répond en envoyant le source html de la page web demandée.
4) Le navigateur lit le source html et identifie les ressources supplémentaires dont il va avoir besoin pour afficher la page (images, feuilles css,…), puis il envoie au serveur des **requêtes** supplémentaires pour obtenir ces ressources.
5) Le navigateur reçoit ces ressources au fur et à mesure et affiche la page web.

Exemple de requête envoyée par le navigateur :

```
GET / HTTP/1.1
Accept: */*
Accept-Encoding: gzip, deflate
Connection: keep-alive
Host: tfontanet.free.fr
User-Agent: Chrome/114.0.0.0 (X11; Linux x86_64)
```

Cette requête commence par : `GET / HTTP/1.1`

- `GET` désigne la « méthode HTTP » à utiliser (voir ci-dessous).
- `/` précise le nom du fichier demandé au serveur (Ici, le slash désigne la page d'accueil du site).
- `HTTP/1.1` correspond à la version du protocole HTTP utilisée.

A cette requête, le serveur va répondre par un message en deux parties avec entête puis données :

```
HTTP/1.1 200 OK
Accept-Ranges: bytes
Connection: close
Content-Length: 2695
Content-Type: text/html
Date: Thu, 27 Jul 2023 14:25:43 GMT
ETag: "84c176-a87-64c0b5ec"
Last-Modified: Wed, 26 Jul 2023 05:58:04 GMT
Server: Apache/ProXad [Jan 23 2019 20:05:46]

<!doctype html>
<html lang="fr">
  <head>
    <title>Cours, devoirs et activités mathématiques à télécharger</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <link rel="stylesheet" type="text/css" href="style.css">
  </head>
  <body>
    <div id="marges">
    ……..……...
```

2) HTTP ou HTTPS ?

Avec le protocole HTTP, les échanges de données se font « en clair ». Si quelqu'un place un équipement pour intercepter le flux de données entre un internaute et un serveur, il aura accès à toutes les données échangées.

Avec le protocole HTTPS, les échanges de données sont chiffrés. Une personne malintentionnée pourra toujours intercepter le flux de données entre un internaute et un serveur mais ce flux sera inexploitable pour elle car elle ne dispose pas de la clé de déchiffrement.

Tant qu'un internaute consulte des sites publics sur lesquels il ne transmet aucune information personnelle (identifiants,…), le protocole HTTP est suffisant. En revanche, s'il doit par exemple s'identifier sur un site, il est prudent de vérifier que la connexion est bien en HTTPS pour éviter que le mot de passe ne transite en clair sur le réseau.

3) Code de réponse du serveur

Dans la première ligne de la réponse du serveur, il y a toujours un code HTTP.
Citons ici quelques codes à connaître :

Code	Signification
200	
304	
404	

III) Échanges de données : Le cas des formulaires HTML

1) Introduction

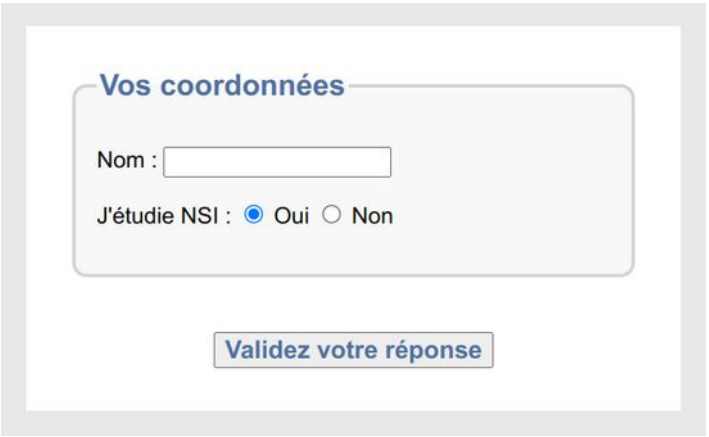
Le client et le serveur sont amenés à échanger régulièrement des données.
La plupart du temps, l'internaute se contente de consulter des pages web. Dans ce cas, les données (= source HTML, CSS ou Javascript, images, vidéos,…) vont uniquement du serveur au client.
Toutefois, il arrive aussi que le client ait à communiquer des informations au serveur (= des identifiants pour se connecter à un site, sa position GPS, un fichier à mettre sur le cloud,…). Dans ce cas, l'échange client vers serveur se fait habituellement via un **formulaire HTML**.

2) Exemple de formulaire HTML :

Code HTML

```
<form action="1nsi-client-serveur-reponse.php" method="get">
  <fieldset>
    <legend>Votre nom</legend>
    <p>Nom : <input type="text" name="nom" value=""></p>
    <p>
      J'étudie NSI :
      <input type="radio" name="nsi" value="Yes" checked> Oui
      &amp;nbsp;&amp;nbsp;&amp;nbsp;
      <input type="radio" name="nsi" value="No"> Non
    </p>
  </fieldset>
  <br>
  <div class="center">
    <input type="submit" value="Validez votre réponse">
  </div>
</form>
```

Rendu sur le navigateur



Lien vers le formulaire

Quand l'internaute va cliquer sur le bouton « Validez votre réponse », cela va déclencher une série d'actions :

1) Le client va envoyer au serveur les réponses de l'internaute appelées « paramètres ». Dans le formulaire ci-dessus, il y a 2 paramètres qui s'appellent **nom** et **nsi**.
2) Le serveur va exécuter le code php de la page « 1nsi-client-serveur-reponse.php » en utilisant les valeurs des paramètres pour produire une nouvelle page HTML.
3) Le serveur va envoyer cette page HTML au client.

Remarques :

- Un formulaire est encadré par les balises <form> et </form> et utilise autant de balises <input type="????" name="????" value="????"> qu'il y a de paramètres à transmettre.
- Les attributs **name** et **value** permettront au langage serveur de récupérer les valeurs entrées par l'internaute.
- Ce mécanisme est écrit en pur HTML. Il n'est pas utile d'ajouter ici du JavaScript.

3) Méthodes GET et POST

Une requête HTTP commence par un mot clé qui précise le type d'action attendu. On parle de « méthode HTTP ».

Lors de l'envoi d'un formulaire, il faut choisir entre **GET** et **POST :**

- La méthode GET va insérer les valeurs des paramètres dans l'URL (peu discret, mais pratique si on veut que le navigateur mémorise nos réponses).

  Ex avec le formulaire précédent :

```
GET /1nsi/1nsi-client-serveur-reponse.php?nom=Thibaud&nsi=Yes HTTP/1.1
Host: tfontanet.free.fr
```

- La méthode POST va insérer les valeurs des paramètres dans le corps de la requête (plus discret, et même indispensable s'il y a un volume important de données à transmettre).

  Même requête que ci-dessus mais avec la méthode POST :

```
POST /1nsi/1nsi-client-serveur-reponse.php HTTP/1.1
Host: tfontanet.free.fr
Content-Type: application/x-www-form-urlencoded
Content-Length: 16

nom=Thibaud&nsi=Yes
```

4) Exemple de code Php exécuté côté serveur

Voici le code de la page « 1nsi-client-serveur-reponse.php » :

```
<?php
  $methode = "Inconnue";
  $nom = "Anonyme";
  $nsi = "et vous auriez mieux fait de choisir NSI ;-(";
  if (isset($_GET['nom'])) {
    $methode = "GET";
    $nom = $_GET['nom'];
    if ($_GET['nsi'] == "Yes") {$nsi = "et vous avez bien fait de choisir NSI ;-)";}
  } elseif (isset($_POST['nom'])) {
    $methode = "POST";
    $nom = $_POST['nom'];
    if ($_POST['nsi'] == "Yes") {$nsi = "et vous avez bien fait de choisir NSI ;-)";}
  }
?>
<!doctype html>
<html lang="fr">
    <meta charset="utf-8">
    <style>
      body {background-color: #e8e8e8; font-family: sans-serif}
      div {margin: auto; max-width: 400px; background-color: #ffffff; padding: 30px}
      fieldset {background-color: #f8f8f8; padding: 15px; border: 3px solid #d4d4d4; border-radius: 10px}
    </style>
  </head>
  <body>
    <br><br>
    <div>
      <fieldset>
        <legend>Bonjour <?php echo $nom ?></legend>
        <p>Vous avez utilisé la méthode : <?php echo $methode ?></p>
        <p><?php echo "Il est ".date("H:i:s").", votre IP est ".$_SERVER['REMOTE_ADDR']; ?></p>
        <p><?php echo $nsi ?></p>
      </fieldset>
    </div><br><br>
  </body>
</html>
```