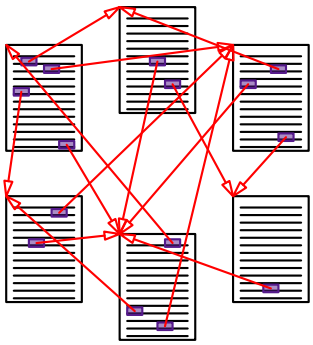
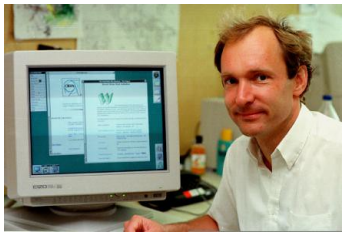
Le Web

World Wide Web



Même si on confond souvent Internet et le Web, ils ne correspondent pas à la même chose. Internet, c'est le réseau mondial. Le Web, c'est une de ses applications, tout comme les e-mails, FTP, le pair-à-pair… Le World Wide Web a été inventé en 1989 par Tim Berners-Lee. Il reprend l'idée de Ted Nelson de 1965 de lien **hypertexte** premettant de passer d'un document à un autre en cliquant sur un lien. Sauf que pour le Web, les documents ne sont pas forcément stockés sur la même machines mais peuvent se trouver sur des machines distantes reliées par Internet.



En 1991, le même Berners-Lee invente le langage HTML, le protocole HTTP et le concept de navigateur web. Les navigateurs sont les logiciels qui permettent d'accéder au Web. Les plus utilisés sont Firefox, Chrome, Edge et Opera. Les navigateurs affichent les pages HTML qu'ils ont obtenues en utilisant le protocole HTTP.

La première page web est toujours accessible à l'adresse :

http://info.cern.ch/hypertext/WWW/TheProject.html

HyperText Transfert Protocol

HTTP est un protocole de transmission. Il sert à obtenir des données auprès d'un serveur web. On parle de modèle **client-serveur**. Le **client** (le navigateur Internet) demande les données et le **serveur** (celui qui héberge le site web) envoie ces données. Il est également possible que le serveur demande des informations au client (mot de passe, formulaire…), mais les données vont majoritairement du serveur au client.
Lorsqu'on veut accéder à une page web, le navigateur envoie une suite de requêtes :

1) Tout d'abord il fait une requête DNS pour trouver l'adresse IP du site.
2) Ensuite, il établit une connexion TCP avec le serveur.
3) Le navigateur fait une requête au serveur pour obtenir la page.
4) Le serveur génère cette page et l'envoie.
5) Le navigateur va ensuite demander tous les éléments nécessaires à l'affichage de la page : feuille de style, images, sons…
6) Une fois tous ces éléments reçus, la page est affichée.

Généralement, la connexion entre le serveur et le client est alors fermée.
Les **requêtes** HTTP sont des messages dont la forme est présentée ci-contre.

```
Méthode
En-tête de requête
<saut de ligne>
Corps de la requête
```

Il y a plusieurs Méthodes possibles pour les requêtes, mais les principales sont GET, POST et HEAD. GET et POST sont assez similaires et servent à recevoir une ressource depuis le serveur. Par rapport à GET, POST permet également d'envoyer des données, comme les réponses à un formulaire. La méthode HEAD permet surtout d'avoir des informations sur la ressources, sans la télécharger.

L'en-tête contient un ensemble de données nécessaires au serveur comme l'adresse de la ressource demandée comme le navigateur utilisé, le langage accepté (anglais, français...) ou s'il y a déjà un cookie pour ce site.
Voici un exemple de requête complète :

```
GET / HTTP/1.1
Host: nsi.janviercommelemois.fr
User-Agent: Mozilla/5.0 (X11; Fedora; Linux x86_64; rv:73.0)
            Gecko/20100101 Firefox/73.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,
        */*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
DNT: 1
Connection: keep-alive
Upgrade-Insecure-Requests: 1
If-Modified-Since: Sat, 21 Mar 2020 13:12:35 GMT
Cache-Control: max-age=0
```

Dans cet exemple, on utilise GET pour obtenir le document à la racine du site "/" en utilisant HTTP/1.1.
Les **réponses** du serveur sont de la forme ci-contre.

```
Ligne de status
En-tête de réponse
<saut de ligne>
Corps de réponse
```

La ligne de status contient la version de HTTP utilisée par le serveur et du code erreur renvoyé. Il y a de nombreux codes d'erreurs, mais les plus connus sont 200 pour dire que tout va bien et 404 pour dire que la ressource n'est pas trouvée. L'entête contient, entre autres, le type de ressource, la date de modification, le type de serveur et la taille de la resssource. Le contenu de la ressource est envoyé dans le corps de la réponse.

200	OK
301	Moved Permanently
304	Not Modified
400	Bad Request
404	Not Found
418	I'm a teapot
500	Internal Server Error
502	Bad Gateway or Proxy Error
503	Service Unavailable

Voici la réponse du serveur à la requête précédente.

```
HTTP/1.1 200 OK
Connection: close
Content-Type: text/html
Last-Modified: Sun, 22 Mar 2020 14:36:59 GMT
Accept-Ranges: bytes
Content-Encoding: gzip
Vary: Accept-Encoding,User-Agent
Content-Length: 4352
Date: Sun, 22 Mar 2020 14:37:10 GMT
Server: LiteSpeed
```

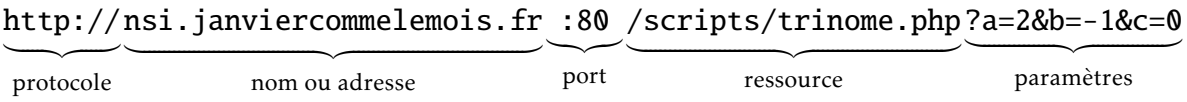
Le contenu est ensuite envoyé. Avec le protocole HTTP, il est envoyé en clair. C'est-à-dire que si une personne malvaillante a accès à un des ordinateurs par lesquels passent les données, il peut lire tout ce qui est envoyé. Pour plus de sécurité, il faut utiliser HTTPS qui chiffre les données échangées afin qu'une personne espionnant les paquets ne puisse pas obtenir d'information.

Exercice 1 : En considérant la requête et la réponse ci-dessus :

1) Quelle est l'adresse du site ?
2) Quel est le navigateur utilisé ?
3) Quelle est la langue utilisée dans le navigateur ?
4) Quand a été modifiée la page ?
5) À quoi peut servir la ligne `If-Modified-Since` ?
6) Quelle est la taille de la page (en octets) ?
7) Quel est le type de serveur utilisé ?

Adresse web

Lorsqu'on veut accéder à une ressource, on tape en général l'adresse ou on clique sur un lien. C'est le navigateur qui va transformer cette adresse, appelée URL, en requête.
Une adresse web est composée de plusieurs parties :



Ainsi, cette requête correspond au script `trinome.php` qui se trouve dans le dossier scripts du site `nsi.janviercommelemois.fr`, contacté sur le port 80 avec le protocole HTTP et les paramètres $a = 2$, $b = -1$ et $c = 0$.
Le port n'est pas obligatoire. Par défaut c'est le 80 qui est utilisé. De même, toutes les ressources ne nécessitent forcément pas de paramètres.
On peut également rajouter un signet, par exemple "`#conclusion`", qui permet d'aller directement à une partie particulière de la page.
Il peut y avoir plusieurs sites hébergés sur le même serveur. L'adresse du site permet de savoir quel site veut visiter l'utilisateur. Par exemple, le site `nsi.janviercommelemois.fr` se trouve dans le dossier `/home/janvierc/public_html/nsi/` sur le serveur. C'est la racine du site. Le fichier `trinome.php` se trouve donc dans `/home/janvierc/public_html/nsi/scripts/`. Le chemin donné est à partir de la racine du site.
Lorsque la ressource n'est pas spécifiée, par exemple avec `http://nsi.janviercommelemois.fr`, le serveur va renvoyer le fichier qui est configuré pour être à la base du site, en général `index.html`.

Exercice 2 : On considère l'adresse :

`http://www.mon-serveur.com/trucs/caches/acces.php?prenom=Pika&nom=Chu`

1) Quel est le nom du site ?
2) Quel est le protocole utilisé ?
3) Quel est le port utilisé ?
4) Quelle est la ressource demandée ?
5) Quel est le chemin pour aller à cette ressource depuis la racine du site ?
6) Quelle semble être l'identité de la personne ayant fait la requête ?

Pages statiques ou dynamiques

Lorsque nous avons fait des pages HTML, c'était des pages **statiques**. C'est-à-dire que quelque soit l'utilisateur, le contenu de la page était toujours le même. Mais lorsque vous vous connectez sur l'ENT ou sur votre webmail, vous voyez une page personnalisée. Elle contient

votre nom, votre identifiant, ou la liste de vos mails. Le serveur ne contient pas une page par utilisateur. Au contraire, elle contient un page **dynamique**. La page est générée à chaque nouvelle requête en fonction de l'utilisateur. Il n'est pas possible de faire cela en HTML. Le serveur doit donc utiliser un autre langage de programmation pour générer cette page. Il existe de multitude de langages spécifiques ou de bibliothèques pour des langages génériques (comme flask pour Python) permettant de générer ces pages.

Nous prendre comme exemple PHP qui est un langage spécifique qui s'insère dans le code HTML. La page est lue par le serveur et le code PHP est exécuté au fur et à mesure, comme pour un script Python. Le résultat est une page HTML qui est ensuite envoyée au client.

```
<html>
  <body>
    Bonjour <?php echo $nom ?>
  </body>
</html>
```

Considérons un exemple basique ci-dessus que nous appellerons `bonjour.php`. Lorsque qu'un client demande cette page, le serveur va l'évaluer et envoyer :

```
<html>
 <body>
  Bonjour Michelle
 </body>
</html>
```

```
<html>
 <body>
  Bonjour Yacine
 </body>
</html>
```

```
<html>
 <body>
  Bonjour Anonymous
 </body>
</html>
```

Mais comment obtenir la valeur de la variable ? C'est là que les deux méthodes GET et POST entrent en jeu. Ces méthodes proposent deux façons pour envoyer des valeurs au serveur.

Méthode GET

Avec la méthode GET, les paramètres sont transmis directement dans l'adresse de la page. Ainsi, la page `http://monsite.fr/bonjour.php?nom=NSI` permet d'afficher le message "Bonjour NSI". Il faut indiquer au script comment obtenir la valeur de cette variable.

```
<html>
  <body>
    <?php $nom=$_GET['nom'] ?>
    Bonjour <?php echo $nom ?>
  </body>
</html>
```

On indique au script qu'il doit aller chercher la valeur de la variable "nom" transmise par la méthode GET.

On peut remarquer que cette méthode donne la possibilité à l'utilisateur de changer à la main les valeurs des paramètres sans avoir à repasser par un formulaire dans une autre page. Bien entendu, GET n'est absolument pas adaptée pour transmettre des données sensibles, comme un mot de passe ou un numéro de carte bancaire.

Méthode POST

Avec la méthode POST, les paramètres ne sont transmis directement dans l'en-tête mais dans le contenu de la requête. Ce contenu n'est pas directement accessible à l'utilisateur et c'est, par exemple, un formulaire qui peut le générer.

```
<html>
  <body>
    <?php $nom=$_POST['nom'] ?>
    Bonjour <?php echo $nom ?>
  </body>
</html>
```

Même si avec la méthode POST, la valeur des paramètres n'est pas visible dans la barre d'adresse, cela ne veut pas dire qu'ils sont protégés. Il faut privilégier HTTPS pour des données sensibles.

Web dynamique et bases de données

Pour obtenir des pages dynamiques, on ne pourrait pas se contenter de demander des valeurs à l'utilisateur. Dans ce cas, il suffirait d'utiliser d'autres langages, comme JavaScript, qui sont exécutés directement par le navigateur. Si on exécute du code sur le serveur, c'est avant tout parce que le serveur contient des informations qui ne sont pas forcément disponibles pour l'utilisateur. Il peut, par exemple, contenir toutes les notes mises au cours de l'année au lycée. Il ne faut pas qu'un utilisateur ait accès à des notes qui ne le concernent pas. Le serveur va donc interroger la **base de données** où sont stockées les notes et n'envoyer que celles de l'élève ou du professeur.
C'est le **Système de Gestion de Bases de Données** qui s'occuper de l'accès à la base de données.

Les cookies

Un même serveur peut recevoir de nombreuses requêtes simultanément. Chacun des clients connecté va demander différentes ressources. Puisque les pages à générer dépendent des informations fournies par les clients, comme son login ou son mot de passe, comment faire pour que le serveur sache quelles données associer à quelle requête ? On pourrait regarder l'adresse IP du client, mais plusieurs ordinateurs dans le même réseau ont la même adresse IP publique. À contrario, un téléphone portable peut changer d'antenne et d'adresse IP en cours de communication avec le serveur. Il faut donc trouver un autre moyen d'identifier les clients.
C'est là qu'on utilise les **cookies**. Ce sont des fichiers textes qui sont transmis par le serveur au client. Ils contiennent un identifiant temporaire (une suite de lettres et de chiffres) qui permet d'identifier la session en cours. Lorsque le client envoie une requête au serveur, il regarde s'il a déjà un cookie pour ce serveur. Si c'est le cas, il le joint à sa requête. Le serveur conserve une liste des numéros de sessions et lorsqu'il reçoit un cookie avec un identifiant connu, il récupère les informations déjà transmises par le client et n'a pas forcément besoin de redemander le login et le mot de passe.
Le cookie contient également une date d'expiration. Cette date peut être très courte ou au contraire durer des mois ou des années. Lorsque la date est expirée, le navigateur efface le cookie. De même, lors d'une navigation privée, tous les cookies enregistrés sont effacés à la fermeture du navigateur. Ainsi, si un nouvel utilisateur réouvre le navigateur, il ne pourra pas retrouver la session du précédent.
Les cookies peuvent aussi contenir d'autres informations comme certains paramètres d'affichage (thème sombre, plein écran…), les paramètres de langue, l'historique de recherche sur le site… Ils peuvent contenir des informations permettant d'en apprendre plus sur les habitudes ou intérêts de l'utilisateur. C'est pourquoi ils sont utilisés par certains sites à des buts publicitaires. Un site ne peut pas accéder à tous les cookies enregistrés sur un ordinateur. Il n'a accès qu'aux cookies qu'il a lui même déposé sur l'ordinateur. Mais lorsqu'un site contient des publicités ou des boutons "partager" de réseaux sociaux, ce sont ces sites qui peuvent déposer des cookies, qui leur disent "l'utilisateur est allé voir telle page sur tel site". C'est pourquoi des réglementations comme la RGPD ont pour but d'encadrer l'utilisation des cookies. De même certains navigateurs, comme Firefox, proposent de limiter l'enregistrement de cookies provenant d'autres sites que celui visité.

Repères historiques

- 1877 : Commercialisation du téléphone aux États-Unis
- 1954 : Naissance du modem
- 1969 : Premier lien ARPANET entre l'université de Californie à Los Angeles et le Stanford Research Institute
- 1971 : Premier envoi de courriel
- 1974 : IP et TCP sont mis en œuvre
- 1980 : UDP mis en œuvre
- 1983 : TCP/IP et DNS sont adoptés
- 1990 : Création de HTTP
- 1991 : Création du World Wide Web
- 1992 : Ouverture d'Internet aux fournisseurs d'accès et aux sites commerciaux
- 1991 : Sortie de Mosaic, le premier navigateur web
- 1994 : Création de HTTPS
- 1994 : Premier FAI français
- 1994 : Création de Yahoo
- 1995 : Sortie de Internet Explorer
- 1995 : Création d'Amazon
- 1998 : Création de Google
- 2002 : Sortie de Phenix qui deviendra Firefox plus tard
- 2004 : Création de Facebook
- 2005 : Création de Youtube
- 2007 : Premier téléphone portable connecté à Internet