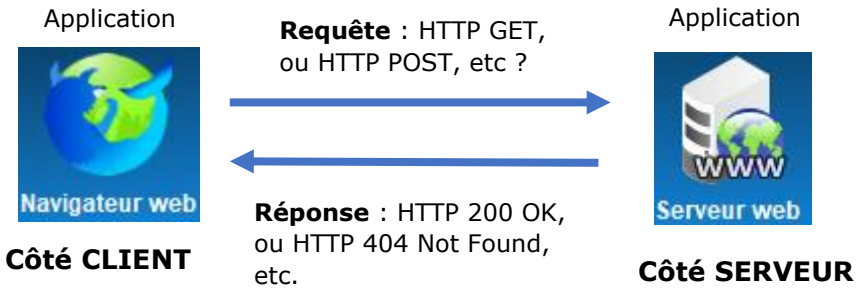
Chapitre 3
Dialogue client-serveur sur le Web

I. HTTP, le protocole du Web

Le protocole HTTP (HyperText Transfer Protocol = Transfert de protocole hypertexte) est un protocole de type client-serveur qui définit les messages envoyés entre le navigateur (client) et le serveur Web (serveur). C'est donc un protocole de la couche « Application » du modèle TCP/IP.
Les messages envoyés par le client sont appelés des *requêtes*, ceux envoyés par le serveur sont appelés des *réponses*.



A. L'URL d'une ressource

La première chose que l'on fait pour accéder à un site Web est de taper l'adresse du site dans la barre d'adresse du navigateur. Cette adresse s'appelle l'URL (Uniform Resource Locator) du site Web.

La syntaxe simplifiée d'une URL est la suivante :

`protocole://nom-ou-adresse/document`

Le protocole est `http` ou `https`. La partie `nom-ou-adresse` peut être le nom de domaine ou bien l'adresse IP du serveur contenant le site. Le document permet de localiser une ressource (souvent un fichier) stocké sur le serveur. Ici, le document correspond au chemin de la ressource sur le serveur.

B. Dialogue client-serveur

En tapant dans la barre d'adresse d'un navigateur l'URL

http://www.info-mounier.fr/essai.html

les actions suivantes se produisent :

1. Le navigateur Web isole le nom de machine www.info-mounier.fr.
2. Le navigateur Web effectue une requête DNS pour obtenir l'adresse IP du serveur Web hébergeant le site (`2001:8d8:100f:f000::20e` dans notre cas, IPv6).
3. Le navigateur Web se connecte à la machine dont l'adresse IP est `2001:8d8:100f:f000::20e`, en utilisant le protocole TCP sur le port 80.
4. Une fois la connexion établie, le navigateur Web envoie un certain nombre de messages en se conformant au protocole HTTP, pour demander la ressource `essai.html`. Il s'agit de la **requête HTTP**.

5. Le serveur Web se trouvant à l'adresse `2001:8d8:100f:f000::20e` envoie en réponse le contenu du fichier à notre navigateur Web. Il s'agit de la **réponse HTTP**.
6. Le navigateur Web peut ensuite parcourir le fichier, et afficher la page correspondante.

Nous allons maintenant nous intéresser plus en détails, aux requêtes et réponses HTTP.

II. Requêtes HTTP et réponses du serveur

A. Requêtes HTTP

Syntaxe d'une requête HTTP

La syntaxe d'une requête HTTP (client vers serveur) est :

```
Ligne de commande (Méthode, URL de la ressource, Version du protocole)
En-tête de requête (nom du serveur, client utilisé, type de document demandé, …)
<ligne vide>
Corps de la requête
```

Par exemple, en naviguant vers l'URL « `http://www.info-mounier.fr/essai.html` », le navigateur envoie la requête HTTP suivante :

Ligne de commande
```
GET /essai.html HTTP/1.1
```
En-tête de requête
```
Host: www.info-mounier.fr
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:71.0)
Gecko/20100101 Firefox/71.0
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: fr,fr-FR;q=0.8,en-US;q=0.5,en;q=0.3
Accept-Encoding: gzip, deflate
Connection: keep-alive
Upgrade-Insecure-Requests: 1
```

Explications :

- La *ligne de commande* indique que le navigateur souhaite communiquer avec le serveur avec la version 1.1 du protocole HTTP et qu'il souhaite obtenir la ressource `essai.html`. La méthode de requête GET est utilisée ici.
- L'*en-tête de requête* qui suit contient diverses informations, notamment le nom du serveur (`www.info-mounier.fr`), le client utilisé (Mozilla) et le type de document accepté (HTML, XHTML+XML, XML).
- Le *corps de la requête* est vide dans cet exemple (il n'y a donc pas de saut de ligne).

Les méthodes de requêtes HTTP

Il existe plusieurs méthodes pour une requête HTTP. Les plus importantes sont :

- GET : c'est la plus courante pour demander une ressource au serveur. Cette requête ne modifie pas la ressource.
- HEAD : cette méthode ne demande que des informations sur la ressource, sans demander la ressource elle-même.
- POST : Cette méthode est utilisée pour soumettre des données en vue d'un traitement (côté serveur). Typiquement c'est la méthode employée lorsque l'on envoie au serveur les données issues d'un formulaire.

B. Réponses HTTP

Syntaxe d'une réponse HTTP

La syntaxe d'une réponse HTTP (serveur vers client) est :

```
Ligne de statut (Version, Code-réponse, Texte-réponse)
En-tête de réponse (Type de document envoyé, type de serveur, date de création de la ressource, cookie, …)
<ligne vide>
Corps de réponse
```

Par exemple, en réponse à la requête HTTP du paragraphe précédent, le serveur renvoie la réponse HTTP suivante :

Ligne de statut

```
HTTP/1.1 200 OK
```

En-tête de réponse

```
Content-Type: text/html
Transfer-Encoding: chunked
Connection: keep-alive
Keep-Alive: timeout=15
Date: Mon, 30 Dec 2019 17:19:02 GMT
Server: Apache
Last-Modified: Thu, 26 Dec 2019 10:17:45 GMT
ETag: W/"159-59a98b02e9950"
Content-Encoding: gzip
```

Corps de réponse

```
<!DOCTYPE html>
<html lang="fr">
  <head>
    <meta charset="UTF-8" />
    <title>Informatique au lycée Mounier ANGERS</title>
  </head>
  <body>
    <h1>Protocole HTTP</h1>
    <p>Ceci est une page pour comprendre le protocole <b>HTTP</b></p>
  </body>
</html>
```

Explications :

- La *ligne de statut* indique que le serveur communique avec le client avec la version 1.1 du protocole HTTP, le code réponse « 200 » indique que le serveur a trouvé la ressource demandée par le client, le texte réponse « OK » est celui associé au code réponse 200.
- Dans l'*en-tête de réponse* on peut trouver le type de contenu du corps de réponse (ici un fichier texte au format HTML), le type de serveur Web (Apache), éventuellement la date de modification (sur le serveur) du fichier demandé, etc.
- Le *corps de réponse* est le contenu du fichier `essai.html` qui avait été demandé.

Les différents code-réponses HTTP

Il existe plusieurs code-réponses, voici les plus importants :

- 200 : Lorsque la ressource est disponible, la requête est donc un *succès*. Le texte-réponse associé est « OK ».
- 404 : Lorsque la *ressource est indisponible*. Le texte-réponse associé est « NOT FOUND ».
- 403 : Lorsque la *permission d'accéder à la ressource est refusée*. Le texte-réponse associé est « FORBIDDEN ».
- 500 : Lorsque le serveur rencontre une erreur interne. Le texte réponse associé est « INTERNAL ERROR ».

A faire **Activité 1 : Le protocole HTTP**

C. Échanges chiffrés avec HTTPS

Le protocole HTTP a un problème majeur : les données sont échangées « en clair ». Cela signifie que, n'importe qui ayant des droits suffisants sur une des machines par lesquelles transitent les requêtes et les réponses HTTP, peut intercepter les paquets échangés et connaître leurs contenus. Or, certaines données échangées sont très sensibles, comme un mot de passe, des coordonnées bancaires…

Il est important que les messages échangés contenant des données sensibles soient chiffrées. Dans ce cas, le client et le serveur utilisent le protocole **HTTPS** (pour *HyperText Transfer Protocole Secure*) pour communiquer. Ainsi, si un pirate informatique intercepte les paquets TCP, il ne pourra observer qu'une suite d'octets d'apparence aléatoire car seuls le client et le serveur sont capables de déchiffrer les paquets.

Le protocole HTTPS est la combinaison du protocole HTTP et d'un protocole de sécurisation des données échangées sur Internet (comme SSL ou TLS). Les pages chiffrées sont reconnaissables dans la barre d'adresse car leur URL commence toujours par :

```
https://...
```

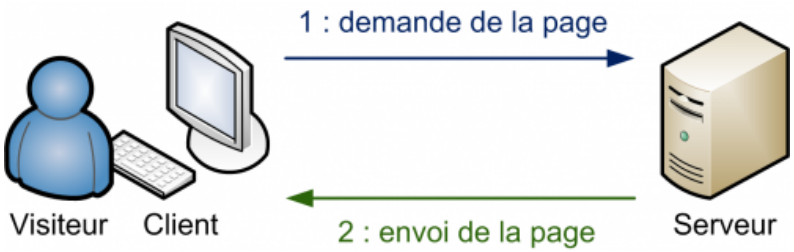
III. Passage de paramètres à un site

A. Site « statique » vs « dynamique »

Sur le Web, cohabitent des sites *statiques* et des sites *dynamiques*. Un site est dit statique lorsque le contenu de ses pages Web ne varie pas en fonction de la demande des utilisateurs : tous les internautes demandant la page (par une requête HTTP) reçoivent le même contenu (réponse HTTP). Un site est dit dynamique lorsque le contenu de ses pages Web est généré de manière dynamique à la demande de l'utilisateur : les internautes demandant la page ne reçoivent pas tous le même contenu.

Site « statique »

Dans le cas d'un site statique, le dialogue client-serveur consiste en un simple échange : une requête HTTP du client puis la réponse HTTP du serveur.



Dialogue client-serveur avec un site statique

Site « dynamique »

De plus en plus de sites Web sont dits dynamiques. Par exemple, sur le site de l'encyclopédie Wikipédia, on peut taper un mot dans le champ de recherche en haut à droite. Ce mot est transmis au serveur qui renvoie l'article correspondant. Autre

exemple, pour se connecter sur e-lyco ou sur un réseau social, on doit remplir des champs de saisies (un *formulaire*) pour s'identifier. Une fois les informations saisies, elles sont transmises au serveur qui génère (« fabrique ») la page d'accueil correspondant au profil de l'utilisateur.

Dans le cas d'un site dynamique, le dialogue client-serveur suit le principe suivant :

1. Le client (navigateur Web) envoie une requête HTTP vers un serveur Web ;
2. En fonction de la requête reçue, le serveur « fabrique » une page HTML grâce à l'exécution d'un programme écrit dans un langage serveur (PHP, Python, Ruby…) ;
3. Le serveur Web envoie la page nouvellement créée au client ;
4. Une fois reçue, la page HTML est affichée dans le navigateur.

Dialogue client-serveur avec un site dynamique

Pour interagir avec un site Web dynamique, **le client peut passer des paramètres au serveur lors de la requête HTTP** (étape 1 ci-dessus). Le passage de paramètres peut se faire :

- soit directement via l'URL (cf. Activité 2 de ce chapitre)
- soit en utilisant des éléments graphiques (*formulaire*) permettant à l'utilisateur de saisir des valeurs et de les envoyer au serveur (cf. Chapitre 4).

Les deux méthodes sont complémentaires et chacun a ses avantages et inconvénients. On utilise l'une ou l'autre selon l'action souhaitée. Nous reviendrons sur cela dans le chapitre 4.

B. Passage de paramètres via l'URL d'un site

Syntaxe plus complète d'une URL

La syntaxe donnée précédemment était incomplète. Voici une syntaxe plus complète :

```
protocole://nom-ou-adresse:port/document?n1=v1&n2=v2&…&nk=vk#id
```

Intéressons-nous uniquement à la partie située après `document`.

- La première partie marquée par un « `?` » est la **liste des paramètres de requête**. Les **paramètres** sont des paires $n_i = v_i$ dans lesquelles `n1`, `n2`,…, `nk` sont les *noms* des paramètres et `v1`, `v2`,…, `vk` sont les *valeurs* respectives des paramètres. Ces paires sont séparées par le symbole « `&` ». Elles ne font pas partie du nom de la ressource que l'on essaie de récupérer mais sont une façon pour le client de passer des valeurs au serveur. Les paramètres seront récupérés par la ressource `document`.
- La portion finale **`#id`** est appelé un **signet** ou **ancre** et permet de cibler un élément en particulier dans la ressource demandée. En pratique, il s'agit de l'identifiant d'un élément HTML de la page demandée.

A faire **Activité 2 : Passage de paramètres à un site**

IV. Le Web côté serveur

A. Utilisation des cookies

Dans le cas d'un site Web dynamique, nous avons vu que le serveur exécutait un programme pour chaque requête HTTP afin de produire une page Web qui est la réponse à la requête. Pour fabriquer une page Web personnalisée pour chaque client, le serveur est capable :

- D'identifier et différencier les différents clients qui se connectent à un site Web.
- De se souvenir de l'état dans lequel était un client lors de sa dernière requête HTTP : cela permet pour un client de ne pas avoir à s'authentifier à chaque requête puisque le serveur se souvient qu'il s'est authentifié un peu plus tôt et qu'il a accès aux pages restreintes.

Ces deux phénomènes sont gérés par l'utilisation de *cookies*.

Les cookies

Un cookie est une petite quantité de données composée d'un *nom*, d'une *valeur* et optionnellement d'une *date d'expiration*. Le nom, la valeur et la durée de vie sont choisis par le serveur. Lorsqu'un serveur veut déposer un cookie particulier chez un client, il l'envoie comme un en-tête de sa réponse HTTP. Par exemple, si un client demande une page `index.php`, le serveur peut répondre

```
HTTP/1.1 200 OK
Date: Mon, 30 Dec 2019 17:19:02 GMT
Server: Apache
Content-Type: text/html
Set-Cookie : test=42 ; Expires=Mon, 31 Dec 2019 18:19:02 GMT;
…
```

En recevant cette réponse du serveur, le navigateur voit que le serveur lui envoie un cookie dont le nom est *test*, la valeur 42 et la date d'expiration est le 31 décembre 2019 à 18h19, 2 secondes. Le navigateur stocke le cookie en l'associant au nom de domaine du site. Si le navigateur se reconnecte au serveur (possiblement longtemps après), il **renvoie le cookie au serveur dans sa requête**, si la date d'expiration de celui-ci n'est pas dépassée. Si une date d'expiration d'un cookie est dépassée, le navigateur le supprime.

Le serveur peut donc identifier et différencier chaque client grâce au cookie envoyé. En effet, si le client lui envoie un cookie, le serveur sait que ce client s'est déjà connecté, qu'il s'agit du client identifié par le numéro de 42, et peut ainsi personnaliser ses réponses. Par exemple, le serveur peut stocker dans un cookie la dernière langue utilisée par un client, et afficher automatiquement le site dans cette langue.

Les sessions HTTP

Supposons qu'un utilisateur lance la page d'accueil d'un site (une requête et une réponse HTTP), puis qu'il s'authentifie avec son identifiant et son mot de passe (une requête et une réponse HTTP), puis qu'il navigue sur les pages à accès restreints du site (plusieurs requêtes et réponses HTTP) et enfin qu'il se déconnecte (une requête et une réponse HTTP). L'ensemble de ces échanges entre le client et le serveur s'appelle une *session*

HTTP. Une session commence par une phase de création de session, puis un ensemble de requêtes-réponses et enfin une phase de destruction de session.

L'intérêt est qu'une fois la session créée, le serveur se souvient des requêtes de la session, et ce pour chaque client. Cela permet de ne pas avoir à entrer son mot de passe à chaque requête HTTP, ce qui rendrait la navigation très compliquée.

En pratique, les sessions HTTP sont gérées par des cookies. Voici le principe :

- Lorsqu'un client se connecte à un serveur, ce dernier regarde si le client lui envoie un cookie particulier (appelons-le `ID`). Si ce n'est pas le cas, alors le serveur crée une valeur unique appelé *identifiant de session* (par exemple `12345`).
- Le serveur fait alors deux choses :
  - Il garde en mémoire dans un *dictionnaire global* une entrée associant l'identifiant de session (ici `12345`) à un dictionnaire de valeurs.
  - Il envoie aussi le cookie `ID=12345` au client.
- Lorsque le client se reconnecte, il renvoie son identifiant de session au serveur.
- Le serveur peut alors récupérer les informations associées à cet identifiant qu'il a conservées et les passer au programme qui va générer la page Web. C'est ainsi que des données sont *enregistrées côté serveur pour chaque client*.
- Lorsque le client se déconnecte, le serveur retire juste l'entrée associée à cet identifiant et efface les valeurs conservées.

Cela explique le fait que si les cookies sont désactivés dans un navigateur, il est impossible de se connecter à une session sur un site car le serveur ne peut alors pas envoyer un identifiant de session (dans un cookie) au client. Chaque navigateur gère ses propres cookies : ainsi, si vous vous connectez à une session avec un navigateur et que vous utilisez un autre navigateur pour accéder au même site, il faudra vous authentifier à nouveau avec cet autre navigateur. Dans le cas d'une navigation privée, les cookies sont tous supprimés à la fermeture de la fenêtre, même ceux n'ayant pas expiré.

A faire Activité 3 : Les cookies et les sessions

B. Survol du langage PHP

Le langage PHP a été créé en 1995 par Rasmus Lerdorf. C'est un des langages les plus populaires pour le développement d'applications Web, on parle de « langage serveur » puisque c'est un langage exécuté par les serveurs. Comme Python, PHP est un langage interprété : l'interprète PHP lit les fichiers PHP et exécute le code qui s'y trouve.

Le langage PHP, exécuté côté serveur, a pour objectif de construire une page HTML, qui peut être personnalisée selon chaque client. Ce langage permet de récupérer les paramètres passés au serveur, de récupérer et interpréter le contenu des cookies, de faire des calculs, …

Le contenu d'un fichier PHP ressemble beaucoup à une page HTML, à ceci prêt qu'il contient également des balises `<?php … ?>` entre lesquelles sera écrit le code PHP.

Voici le fonctionnement côté serveur :

1. A réception d'un requête HTTP d'un client, le serveur passe le fichier PHP à l'interprète PHP.
2. Celui-ci copie tous les caractères du fichier sur sa sortie standard et exécute le code contenu entre les balises `<?php … ?>`.
3. Le serveur récupère le fichier de sortie et l'envoie au client.

A faire Activité 4 : Découverte du langage PHP

A retenir

Le protocole HTTP est plus qu'un protocole d'échange de fichiers car il permet à des pages Web d'**envoyer des paramètres de requêtes** (via l'URL ou via un formulaire). Ces paramètres permettent au serveur de **calculer** le contenu de la page à fabriquer et à renvoyer au client : on parle alors de site dynamique car le contenu est calculé à chaque requête côté serveur grâce à un langage serveur comme le PHP.
L'utilisation de **sessions HTTP** permettent de rendre encore plus riches les interactions entre client et serveur puisque ce dernier a la possibilité de se souvenir de calculs intermédiaires entre deux requêtes. L'implémentation des sessions est rendue possible par l'utilisation de **cookies** qui sont des petits paquets d'information créés par le serveur et stockés sur le client.

Ressources :

- *Numérique et Sciences Informatiques*, T. BALABONSKI, S. CONCHON, J.-C. FILLIATRE, K. NGUYEN, Ellipses.
- *Réalisez votre site web avec HTML 5 et CSS 3*, Mathieu Nebra, OpenClassrooms, EYROLLES (cours OpenClassrooms : https://bit.ly/38V9pRh)