Annexe 1 – Le protocole HTTP Hyper Text Transfert Protocol

Sommaire :

I - Introduction 1
II - Fonctionnement 1
III - Requête HTTP 2
IV - Réponse HTTP 3
V - HTTP/2 ou HTTP 2.0 4
VI - HTTP et PHP 4
VI.1. Introduction 4
VI.2. Fonctionnement 4
VII - HTTPS (HTTP Secured) 4

I - Introduction

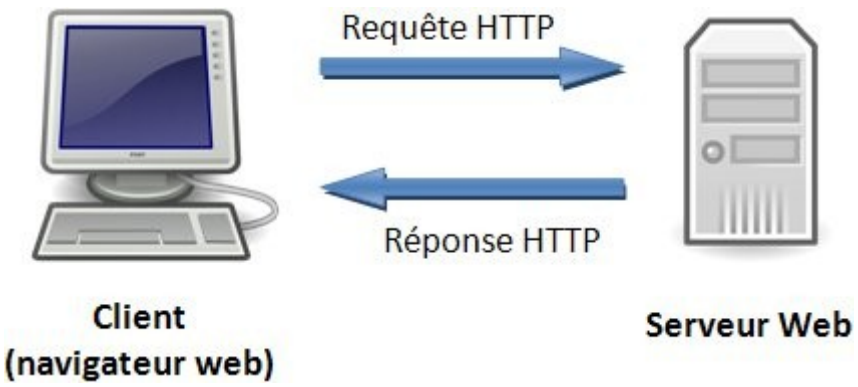
HTTP signifie **H**yper**T**ext **T**ransfer **P**rotocol. Il s'agit d'un **protocole applicatif** (**couche 7 OSI**) qui permet la transmission de documents distribués et multimédia. Le protocole **HTTP 1.1** est décrit dans la RFC 2616.

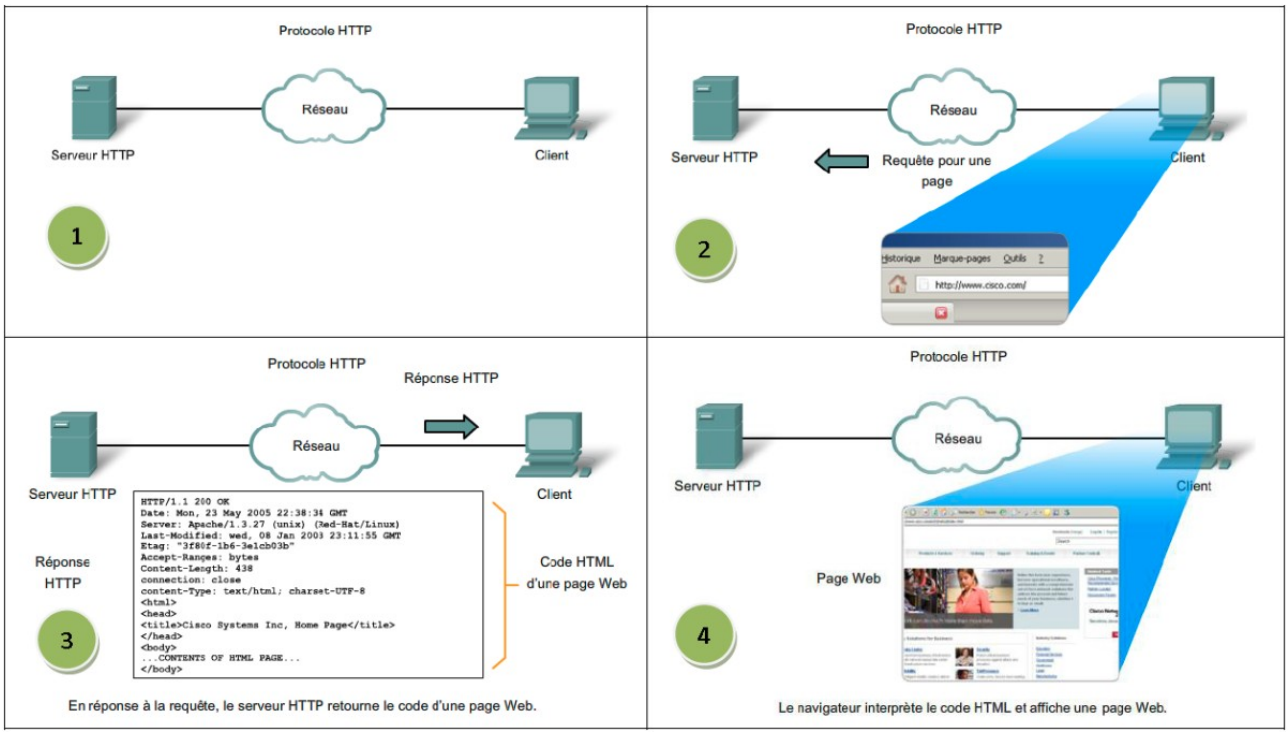
Ce protocole fixe le **principe de communication** entre un logiciel de navigation Web (communément appelé "**navigateur**" ou « **browser** » ou « **client Web** ») et un **serveur Web** (souvent appelé **httpd** , **d** pour daemon sous Unix).

II - Fonctionnement

Le service **HTTP** est construit sur un modèle **Client-Serveur**. Il utilise le protocole de transport **TCP** (**T**ransport **C**ontrol **P**rotocol). Le **serveur HTTP** utilise le **port 80**.

Le principe de communication est simple, la communication s'initie toujours à la **demande du client**. Celui ci s'adresse à un serveur caractérisé par une adresse IP ou un nom, sur un port connu, le plus souvent le port **80** :





III - Requête HTTP

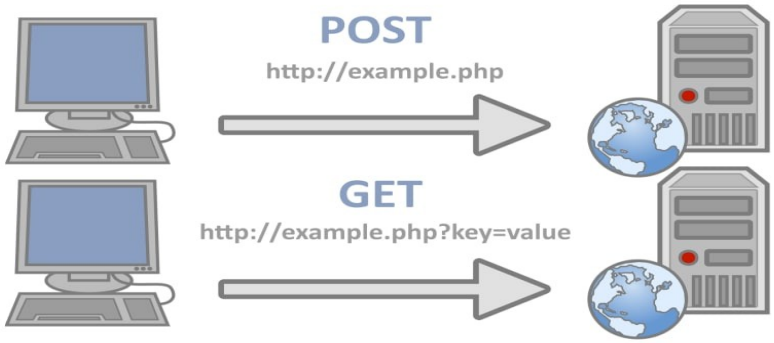
Le format d'une **requête HTTP** est le suivant :

Ligne de commande (Commande, URL et Version)
En-tête de la requête
[ligne vide]
Corps de la requête

La ***ligne de commande*** possède trois champs : ***Commande***, ***URL*** *et* ***Version***.

Le premier champ, ***Commande***, contient une des commandes définies dans le protocole **HTTP**. Les principales ***commandes*** sont les suivantes :

- **GET** : demande au serveur de renvoyer le contenu de l'information pointée par l'URL spécifiée dans la ligne de commande.
- **POST** : permet au client d'envoyer des données au serveur, comme par exemple le contenu d'un formulaire renseigné par l'utilisateur. Ces données constituent le *corps de la requête*.



Exemple de requête :

GET /index.html HTTP/1.1 **Host : localhost** **Accept : text/html** If-Modified-Since : Sunday, 11-May-2022 19:33:11 GMT User-Agent : Mozilla/4.0 (compatible; MSIE 5.0; WinXP)

Ici, on demande l'envoi de la page **index.html** du serveur sur lequel on est connecté à condition que cette page ait été modifiée depuis le **11 Mai 2022**. De plus le client transmet des informations concernant son navigateur.

IV - Réponse HTTP

Le format d'une **réponse HTTP** est le suivant :

Ligne de statut (Version, code-réponse, texte-réponse)
En-tête de la requête
[ligne vide]
Corps de la requête

La ***ligne de statut*** d'une réponse **HTTP** comprend trois champs : ***Version***, ***code-réponse***, ***texte-réponse***. Les principaux ***code-réponse*** sont les suivants :

	Code	**Méthode**	**Signification**
Succès	*200*	*GET, PUT, DELETE*	**[OK]** : Opération réussie.
Erreurs client	*404*	*GET, POST, PUT, DELETE*	**[NOT FOUND]** : Ressource non trouvée.
	405	*GET, POST, PUT, DELETE*	**[NOT ALLOWED]** : Méthode non autorisée sur la ressource
Erreur serveur	*500*	*GET, POST, PUT*	**[INTERNAL SERVER ERROR]** : Erreur interne au serveur.

Le ***corps*** ou ***body*** contient en fait le document demandé. Cela peut être un fichier HTML simple ou un fichier binaire quelconque, dont le type sera précisé dans l'en-tête par le champ **Content-Type**.

Exemple de réponse :

HTTP/1.1 200 OK Date : Sunday, 11-May-2022 19:33:14 GMT Server : Apache/1.1 Content-Type : text/html Content-Lenght : 65 Last-Modified : Sunday, 11-May-2022 10:54:42 GMT
<HTML> <body> Bienvenue sur notre site…… </body> /HTML>

Dans cet exemple, le serveur renvoie une page au format HTML, en précisant quelques informations comme la version du logiciel serveur et la date de dernière modification du fichier considéré.

V - HTTP/2 ou HTTP 2.0

HTTP/2 est une révision majeure de la spécification du protocole **HTTP**. Le but de **HTTP/2** est d'améliorer les performances **HTTP** en traitant les problèmes de latence qui existaient dans la version **HTTP 1.1** du protocole.

HTTP/2 conserve la majorité de la syntaxe de **HTTP 1.1**. Un élément a été modifié : la manière dont la donnée est segmentée et transportée entre le client et les serveurs ce qui n'a pas d'impact sur les applications existantes.

VI - HTTP et PHP

VI.1. Introduction

Le langage **HTML** ne permet pas à lui seul de créer des documents dynamiques ou interactifs capables par exemple d'indiquer la date du jour ou de donner le résultat d'une requête sur une base de données.

Le langage **PHP** va nous permettre de construire un document HTML **dynamique** correspondant à la demande spécifique du client.

VI.2. Fonctionnement

Le client indique le nom d'un **fichier** à l'aide d'une **URL** (**U**niform **R**essource **L**ocator), par exemple ***http://www.site.fr/index.php***, non pour **recevoir son contenu**, mais pour demander son **exécution** au serveur. Ce dernier exécute le programme indiqué et renvoie au client la sortie standard de ce programme (c'est à dire ce que l'on aurait obtenu à l'écran en lançant le programme par une ligne de commande).

Il est possible d'envoyer au serveur des données saisies par l'utilisateur. Ces données constituent des **arguments d'entrée** pour le programme à exécuter.

Elles sont codées selon le format suivant :

nom_champ1=valeur1&nom_champ2=valeur2…

où chaque couple (nom du champ, valeur de saisie) est séparé par un '**&**' et où nom du champ et valeur de saisie sont séparés par un '**=**'. Une fois la chaîne des données construite, elle est envoyée au serveur selon la méthode **GET** ou **POST** de **HTTP**.

Avec la méthode **GET**, la syntaxe d'une URL est la suivante :

http://www.site.fr/index.php?nom=Dupont&pr%E9nom=Ren%E9

Si on utilise la méthode **POST**, l'URL a la syntaxe suivante :

http://www.site.fr/index.php

VII - HTTPS (HTTP Secured)

HTTPS (avec **S** pour ***Secured***, soit « sécurisé ») est la variante de **HTTP** basée sur les protocoles **SSL** (**S**ecure **S**ocket **L**ayer) ou **TLS** (**T**ransport **L**ayer **S**ecurity).

Il permet au visiteur de vérifier l'identité du site auquel il accède grâce à un ***certificat d'authentification***. Il permet également de chiffrer la communication. Il est généralement utilisé pour les transactions financières en ligne : commerce électronique, banque en ligne, courtage en ligne, etc.

Le **HTTPS** utilise le port **443** par défaut. Le RFC 2818 définit le protocole **HTTPS**.