Introduction au Web

Cours d'introduction aux principes généraux du fonctionnement du Web

- Qu'est ce que le web ?
- Le Protocole HTTP
- Les clients du Web
- Les services Web
- Les Technologies du Web

Qu'est ce que le web ?

Le Web est un système d'information permettant la mise à disposition de ressources sur un réseau. Le web est l'application d'Internet la plus utilisées.

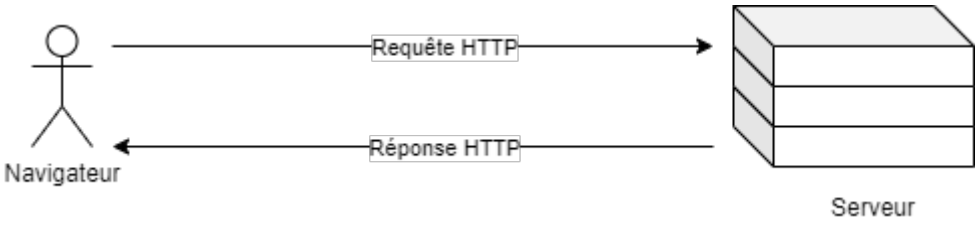
Il repose sur un protocole, HTTP (Hyper Text Transfer Protocol) pour transferer les ressources. Une ressource web est identifié par une URL (Uniform Resource Locator). Une ressources est mise à disposition par un serveur web.

Le web est un système d'information distribué, qui mets en jeu un serveur et des clients :

- Un serveur Web est une application exécutée sur une machine mettant à disposition des ressources Web
- Un client Web est une application exécutée sur une machine capable d'envoyer des requêtes HTTP à un serveur web afin de consommer des ressources HTTP.

Le Protocole HTTP

Le World Wide Web utilise principalement, le protocole HTTP (Hyper Text Transfer Protocole). Les protocole HTTP est un protocole client serveur. En effet, un client, votre navigateur, va émettre une requête à l'intention d'un serveur, qui va en retour lui fournir une réponse.

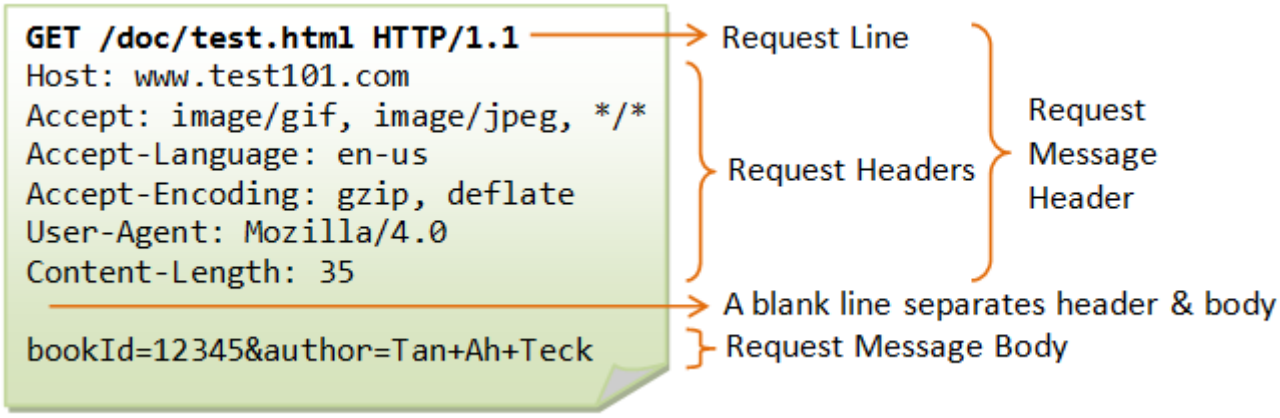


La connexion s'effectue via le protocole réseau TCP. Les échanges se font au format texte et sont très codifiés.

Etudions plus en détail la requête et la réponse HTTP.

Requête HTTP

Une requête HTTP est constituée de plusieurs éléments



Méthode de requête (Http method)

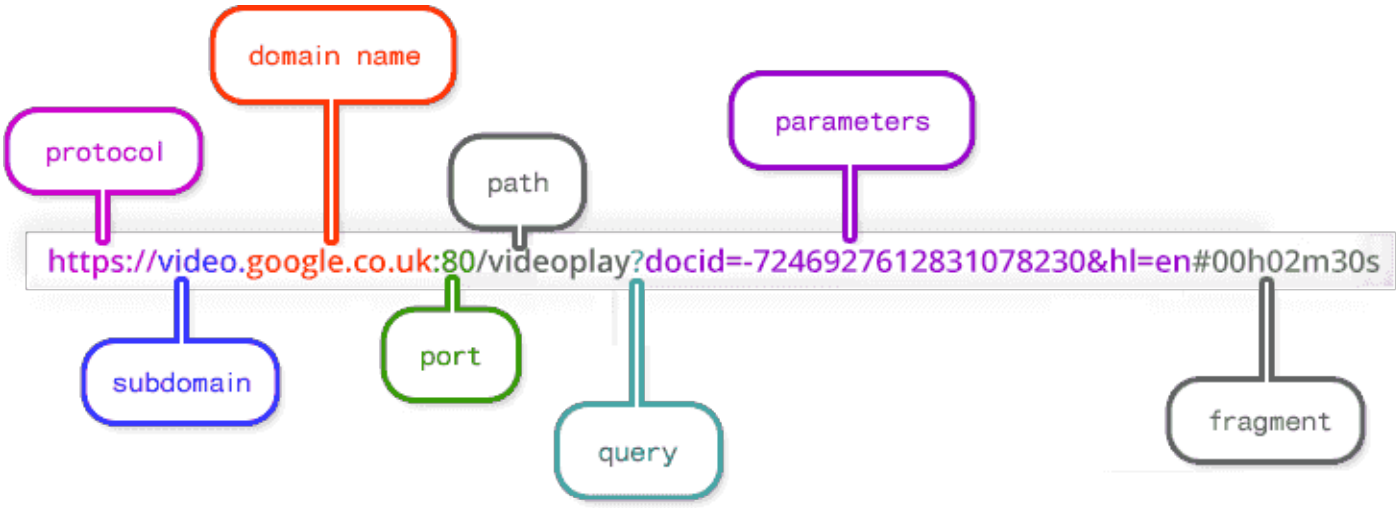
Le premier mot de la première ligne correspond à la méthode de requête (ex : GET, POST, PUT, DELETE ...), permet de décrire ce que va faire la requête, détaillé sur la page suivante.

URL

Sur la première ligne, on retrouve également l'URL de la ressource intérogée, c'est le chemin de la ressource sur le serveur (par exemple */index.html*).

Cependant on peut ajouter des paramètres via l'URL en ajoutant un `?` ç la fin de l'URL. Les paramètre sont ensuite écrit dans l'URL à la suite sous la forme `nomParam=valeurParam` séparés par des `&`.

Exemple :
`https://example.com/posts?page=1&order=asc` → deux paramètres, `page` avec la valeur `1` et `order` avec la valeur `asc`



Notion d'Endpoint

Un endpoint est la combinaison de la route d'une ressource et d'une méthode. On dit qu'une application possède un Endpoint pour une route et une méthode donnée. Par exemple :

GET `/api/posts` est un Endpoint

Version HTTP

A la fin de la première ligne est affichée la version du protocole HTTP utilisée par cette requête (ex : HTTP/1, HTTP/1.1, HTTP/2).

En tête (Headers)

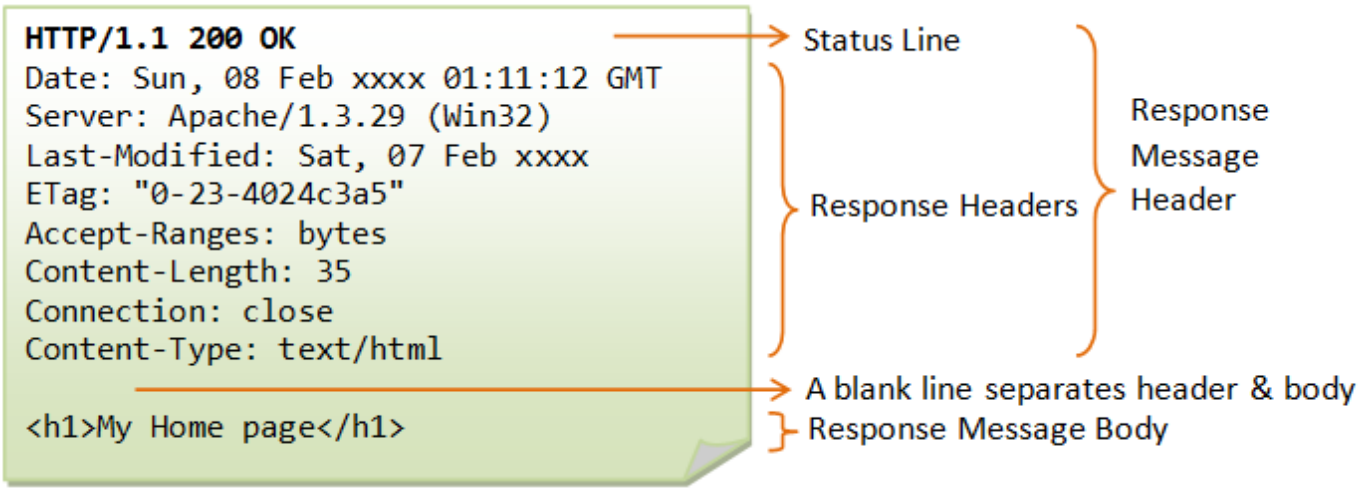
Ensuite, sur chaque ligne suivantes, on a les entêtes de la requête se présentent sous la forme d'un dictionnaire clé valeur qui renferme des méta-données sur la requête. Notamment utilisé pour l'authentification.

Corps de la requête (Request Body)

Certaines méthodes HTTP permettent de passer des données au serveur dans la requête, ces données se retrouvent dans le Body de la requête. La longueur du body est définie dans le Header Content-Length.

Réponse HTTP

La réponse HTTP est similaire à la requête, mais elle ne contient ni URL ni méthode, seulement des en-tête et un corps.



Status HTTP

A la fin de la première ligne figure le status HTTP, qui informe sur le résultat de la requpête. Il est constitué d'un code (200, 404...), et d'un texte (OK, NNOT FOUND...)

En tête (Headers)

Ensuite, sur chaque ligne suivantes, on a les entêtes de la requête se présentent sous la forme d'un dictionnaire clé valeur qui renferme des méta-données sur la réponse.

Corps de la réponse (Response Body)

Souvent, le serveur renvoie des données en réponse, ces données se retrouvent dans le Body de la réponse. La longueur du body est définie dans le Header Content-Length.

Le corps de la réponse peut être en texte (HTML, JSON, XML, ...) ou en flux binaire.

Principales méthodes HTTP.

GET

Les requêtes avec la méthode GET permettent de récupérer des données du serveur, lire une ressource. Cette méthode d'autorise pas de *Request Body*. La ressource est retournée dans le body de la réponse. Les requêtes GET sont dites *sans effet de bord*, elle ne doivent rien changer à l'état du serveur.

POST

Les requêtes avec la méthode POST permettent de créer une ressource sur le serveur. Le client envoie l'état désiré de la ressource dans le *Request Body*, l'état de la ressource après création est retournée dans le *Response Body*.

PUT

Les requêtes avec la méthode PUT permettent de remplacer des données sur le serveur, le client envoie le nouvel état de la ressource dans le *Request Body* de la requête. L'état modifié de la ressource doit être retrourné dans le *Response Body*.

DELETE

Les requêtes avec la méthode DELETE permettent de supprimer des données du serveur. Cette méthode n'autorise pas de *Request Body* ni de *Response Body*. Si la *Response* est en succès, cela veut dire que la ressource n'existe plus.

Codes de réponse HTTP

Le protocole HTTP présente différents types de codes :

- 1XX : Informations
- 2XX : Succès
- 3XX : Redirection

- 4XX : Erreurs clients (requête invalide)
- 5XX : Erreurs serveurs (échec de traitement)

Quelques exemples de succès

- 200 OK : la requête a été traitée avec succès
- 201 Created : la requête a été traitée avec succès et a créé une ressource (la réponse possède un Header "Location" qui contient l'adresse de la ressource créée
- 202 Accepted : la requête a été acceptée et son traitement est en cours
- 204 No Content : la requête a été traitée avec succès mais la réponse ne contient pas de Body.

Quelques exemples de redirection

Les Response avec un code redirection possèdent un Header "Location" contenant le nouvel emplacement de la ressource.

- 301 Moved Permanently : la ressource a été deplacée pour de bon
- 307 Temporary Redirect : la ressource a été deplacée temporairement
- 302 Found : pointe vers une ressource à GET

Quelques exemples d'erreur client

- 400 Bad Request : Mauvais format de requête
- 401 Unauthorized : L'authentification à échouée
- 403 Forbidden : Vous n'avez pas accès à cette ressource
- 404 Not Found : La ressource n'existe pas
- 406 Not Acceptable : Le format demandé n'est pas disponible
- 418 I'm A Teapot : Le serveur est un théière (poisson d'avril de la RFC en 1997)

Quelques exemples d'erreurs serveur

- 500 Internal error : Erreur de traitement coté serveur
- 501 Not Implemented : Cette ressource devrait marcher mais elle n'est pas finie
- 503 Service Unavailable : Le service est en maintenance ou crashé

Les clients du Web

Différents types de client intéragissent avec des serveur en utlisant le Web.

Le Navigateur web

Le navigateur web est une application donc l'objectif est de demander des pages webs à des serveurs web. Le standard technologique des navigateurs Web est le suivant :

- HTML pour définir la structure des pages
- CSS pour pour définir l'apparance des pages
- Javascript pour gérer les interactions de l'utilisateur avec la page

Le navigateur contient donc :

- Un client HTTP : récupérer les pages auprès des serveurs
- Un moteur de rendu HTML / CSS : pour afficher l'interface à partir des document HTML et des feuilles de styles CSS
- Un environnement d'exécution Javascript : pour exécuter les sources Javascript rattachées au pages

Il existe plusieurs types d'applications Web consultables depuis un navigateur

Sites statiques

Les sites statiques, comme par exemple les sites vitrines, sont des sites non dynamiques qui sont composés de fichiers HTML/CSS/JS qui servit naïvement par un serveur web à partir de son système de fichier.

Applications web clientes riches

Les applications dites "client riches" ou SPA (single page applications) sont des applications fortement interactive, dynamiques, rendues entièrement cotés clients. Servies par le serveur sous forme de "bundle", elles vont intéragir programmatiquement avec ce dernier par le biais de requête HTTP pour pour récupérer et envoyer des données, mais l'interface n'est pas demandé au serveur mais est rendue programmatiquement (en utilisant du Javascript) dans une seule page.

Applications "à pages"

Les applications dites "à pages" sont des applications dynamiques dont les interfaces sont rendus coté serveur. Comme pour un site statique, le navigateur demande des pages au serveur, mais va aussi lui en envoyer par le biai de la soumission de formulaire.

Les autres clients

Applications du bureau & mobiles

Les applications mobiles et de bureau faisant partie d'un système d'information utilisent également le Web pour interagir avec un serveur d'applications qui propose des services web.

Autres serveurs

Un serveur web peut également proposer des service web à d'autres serveur pour permettre à des systèmes d'information d'interopérer.

Les services Web

Representationnal State Transfer (ReST)

Le standard ReST est un standard sans état qui vise à facilité l'interopérabilité en donnant plus de controle au client. ReST est très proche des standards et de la sémantique du Web :

- Orienté ressource
- Les méthodes HTTP définissent la sémantique des appels

Une ressource est une donnée présente sur le serveur, sur laquel on va effectuer des opérations en utilisant la sémantique des méthodes HTTP. On peut interagir avec des collections ou des entités individuelles de la ressource.

Le transport des données en Rest se fait soit au format XML, soit au format JSON, mais JSON est beaucoup plus utilisé.

Exemple d'API ReST

SOAP

SOAP (Simple Object Access Protocol) est un standard de webservice qui expose des fonctions qui peuvent être appelées par le client. La spécification se fait à travers d'un fichier WSDL (Web Service Description Language) au format XML. Ce standard est un peu désuet car il est assez lourd à utiliser.

GRPC

gRPC est un système d'invocation de procédure distance (remote procedure call) très moderne développé par Google. Les données sont transportées en binaires pour augmenter les performances et la version 2 du protocole HTTP est utilisée. Les services sont décrits par un langage dédié (Protobuf), avec support de génération de code pour la plupart des langages. Il supporte nativement des fonctionnalités comme l'authentification et le streaming bidirectionnel. Le protocole gRPC est très utilisé dans les infrastructures cloud pour la communication entre les conteneurs applications.

Les Technologies du Web

Coté Serveur

Langage	Frameworks
Java	Jakarta EE, Spring
C#	ASP .NET Core
Javascript / Typescript	Express.js,Nest
Python	Django, Flask
PHP	Laravel, Symphony

Coté Client

Langages : Javascript ou Typescript

- Angular
- Vue
- React
- Svelte