Bienvenue !

Pierre-Julien VILLOUD

pjvilloud@protonmail.com
https://github.com/pjvilloud
https://linkedin.com/pjvilloud
http://pjvilloud.github.io

Introduction

Les applications web sont aujourd'hui incontournables. Nous discuterons dans ce cours des éléments nécessaires à la mise en place d'une application Web dynamique.

Le navigateur

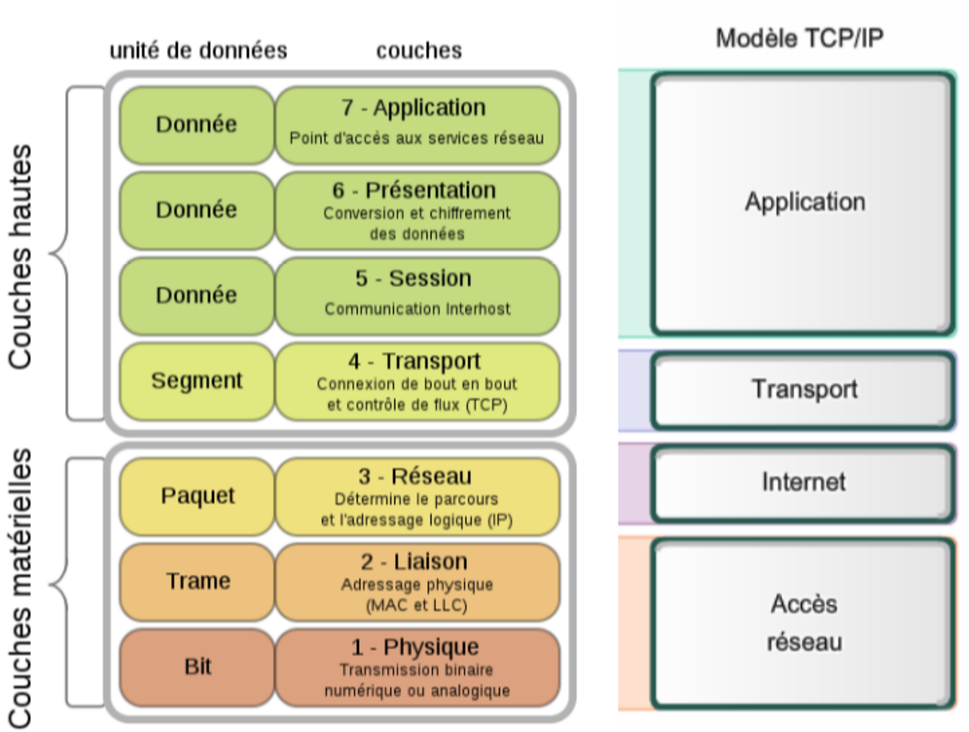
Le navigateur est un logiciel permettant d'accéder et d'afficher les sites Internet présents sur le *World Wide Web*.

Composants de communication basés sur les standards réseaux
Moteurs de rendu basés sur les standards web
Interface utilisateur
Plugins

Le premier navigateur a été conçu par Tim Berners-Lee en 1990

Les standards

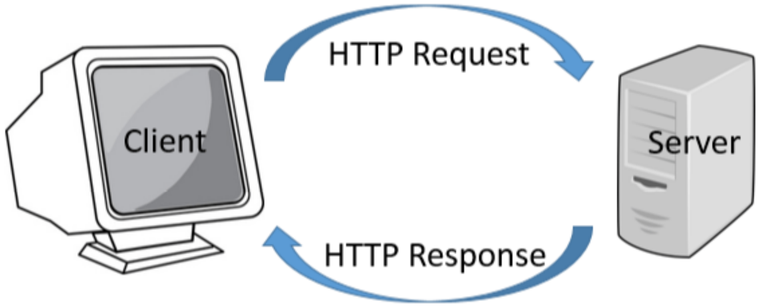
ℹ Avant de rentrer dans le détail des standards réseaux, voyons les différentes couches du *modèle OSI* (Open Systems Interconnection) qui standardise les échanges réseaux entre systèmes informatiques, ainsi que le modèle *TCP/IP* utilisé plus communément pour représenter le *modèle Internet*.



ℹ Nous allons laisser les couches *Transport*, *Internet* et *Accès réseau* pour se concentrer sur la couche *Application*

Le protocole HTTP

HTTP ou **H**yper**T**ext **T**ransfer **P**rotocol est le protocole de communication basé sur le principe de client serveur utilisé sur le *World Wide Web*. Il existe plusieurs *méthodes* ou *commandes* que le client peut envoyer au serveur.




```
Ligne de commande (Commande, URL, Version de protocole)
En-tête(s) de requête
[Ligne vide]
Corps de requête
```

```
GET /page.html HTTP/1.0
Host: example.com
Referer: http://example.com/
User-Agent: CERN-LineMode/2.15 libwww/2.17b3
```

```
Ligne de statut (Version, Code-réponse, Texte-réponse)
En-tête(s) de réponse
[Ligne vide]
Corps de réponse
```

```
HTTP/1.0 200 OK
Date: Fri, 31 Dec 1999 23:59:59 GMT
Server: Apache/0.8.4
Content-Type: text/html
Content-Length: 59
Expires: Sat, 01 Jan 2000 00:59:59 GMT
Last-modified: Fri, 09 Aug 1996 14:21:40 GMT

<TITLE>Exemple</TITLE>
<P>Ceci est une page d'exemple.</P>
```

Nous allons voir maintenant les différentes méthodes (ou commandes) ainsi que les codes réponses.

Méthodes HTTP

Voici les méthodes possibles avec le standard HTTP

GET : Méthode la plus courante, utilise pour demander une ressource sans modifier cette dernière.
POST : Méthode utilisée pour transmettre des données impliquant une création ou une modification sur une ressource
PUT : Méthode utilisée pour remplacer totalement ou ajouter une ressource
PATCH : Méthode utilisée pour remplacer partiellement une ressource
DELETE : Méthode utilisée pour supprimer une ressource
HEAD, OPTIONS, CONNECT, TRACE...

Les méthodes POST, PUT et PATCH requièrent un corps de requête contenant les données à envoyer au serveur.

Format des données échangées

ℹ Les données échangées (envoyées dans une requête HTTP, ou reçues dans une réponse HTTP) peuvent prendre plusieurs formats.

Consultation d'une page web	`image/png`
	`text/html`
	`text/javascript`
	`text/css`
		`<!DOCTYPE html>`
		`<html>`
		` <head><title>Titre</title></head>`
		` <body>`
		`  <p>Ceci est un paragraphe.</p>`
		` </body>`
		`</html>`

Formulaire	`application/x-www-form-urlencoded`	marque=Peugeot&modele=208

Web Service	`application/json` ou `application/xml`	`{`
		`  marque: "Peugeot",`
		`  modele: "208"`
		`}`
		
		`<vehicule>`
		`  <marque>Peugeot</marque>`
		`  <modele>208</modele>`
		`</vehicule>`

ℹ Lorsqu'il y a échange de données, le *client* ou le *serveur* doit spécifier dans l'en-tête *Content-Type* le type des données qu'il envoie.

Codes erreurs HTTP

Voici les différents codes erreurs que peut retourner une requête HTTP

2xx : Requête OK !	200 OK, 204 NO Content
3xx : Redirections	301 Moved Permanently, 302 Moved Temporarily
4xx : Erreurs client	400 Bad Request, 401 Unauthorized, 403 Forbidden, 404 Not Found, 409 Conflict
5xx : Erreurs serveur	500 Internal Server Error

Il y en a beaucoup d'autres...

Le serveur web

Le serveur web est la machine que contacte le client pour obtenir les ressources (fichiers HTML, JS, CSS ou autre).

Ce type de serveur peut embarqué des modules permettant l'exécution de scripts afin de dynamiser le contenu, comme par exemple PHP ou PERL, ou encore Python.

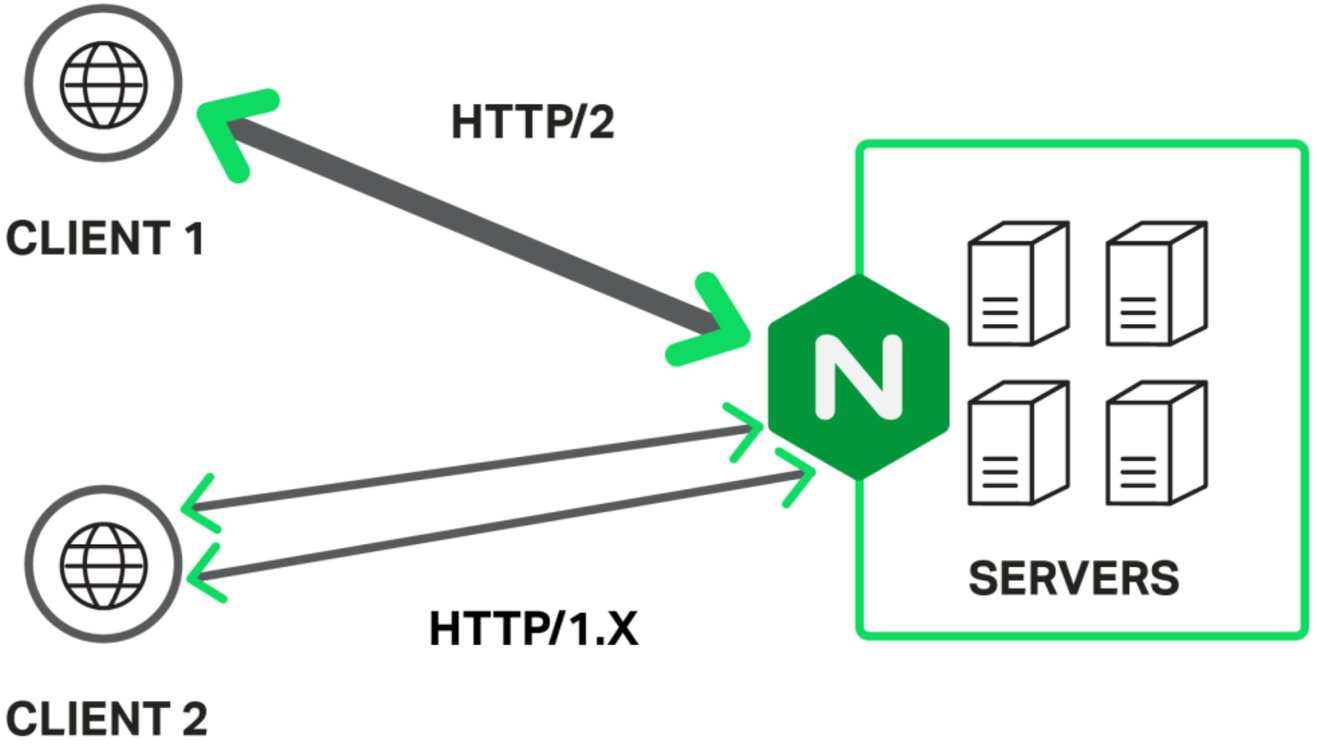
Apache

Le serveur Apache est un des serveurs web les plus populaires, du fait notamment de sa license libre. Il possède de nombreux modules permettant l'interprétation de langages de script comme PHP, Perl, Python ou Ruby.

Documentation d'Apache

Nginx

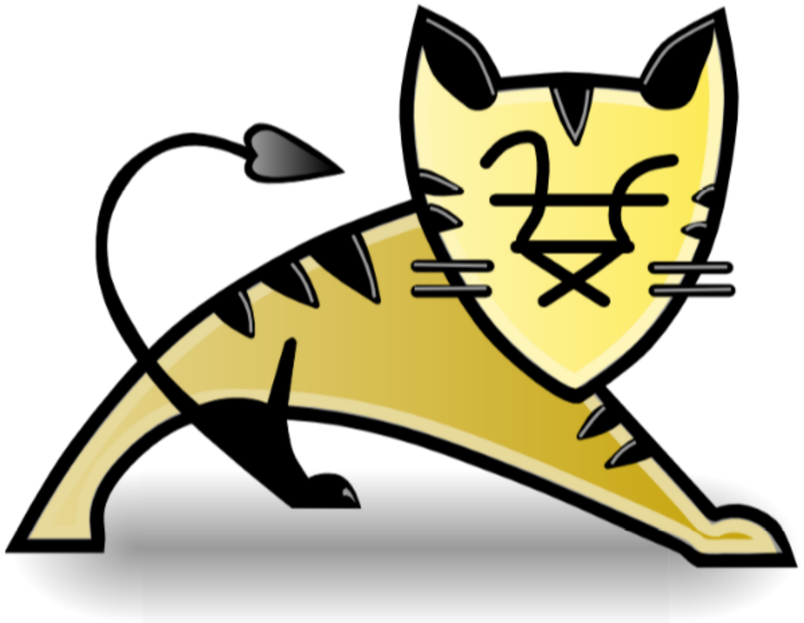
Nginx est un autre serveur web reconnu pour ces hautes performances et sa faible consommation mémoire. Il est souvent utilisé aussi pour sa fonctionnalité de proxy inverse.



Documentation Nginx

Le serveur d'application

Le serveur d'application propose un *contexte d'exécution* permettant de développer des composants applicatifs avec des langages tels que Java ou C# nécessitant chacun en environnement spécifique (JVM ou plateforme .NET).

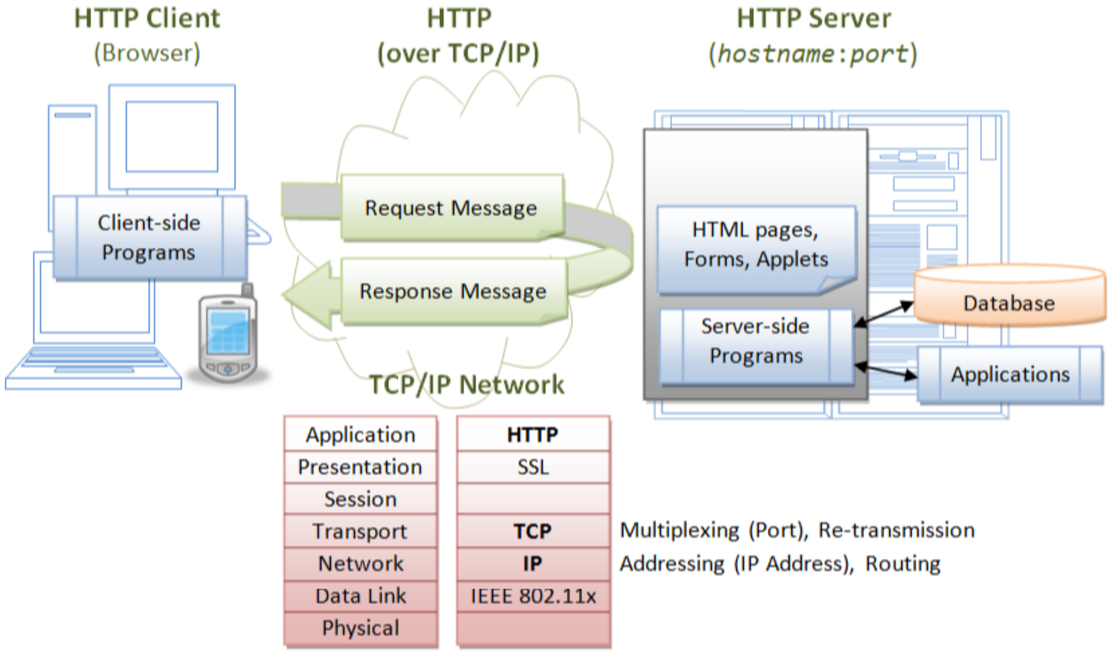


Documentation de Tomcat

Ces serveurs permettent le développement d'applications complexes et multiples.

Tomcat

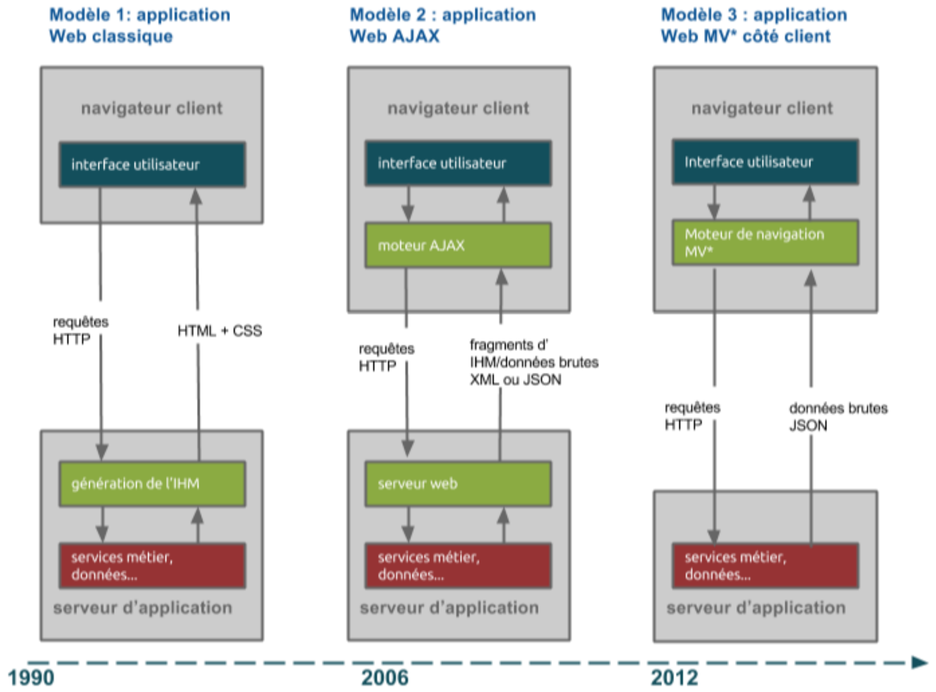
ℹ Le serveur d'application Tomcat peut agir comme un serveur web classique mais il donne la possibilité de gérer JSP et servlets. Nous verrons la notion de servlet dans le cours Java 320, et les JSP dans le cours 330.



ℹ Pour simplifier cela permet d'utiliser Java pour développer des applications web dynamiques.

Les différentes architectures

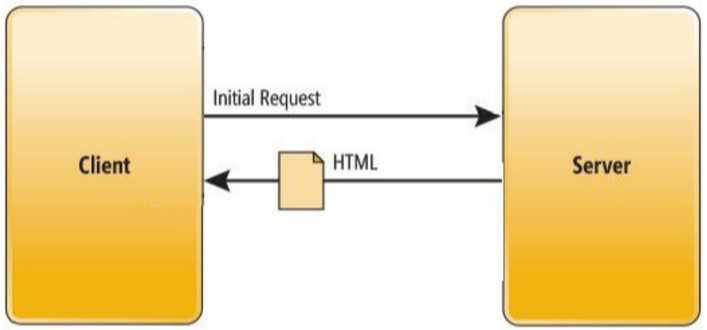
ℹ Il existe plusieurs modèles permettant la réalisation d'applications web aujourd'hui.



ℹ Voyons en détail chacun de ces modèles

Application web statique

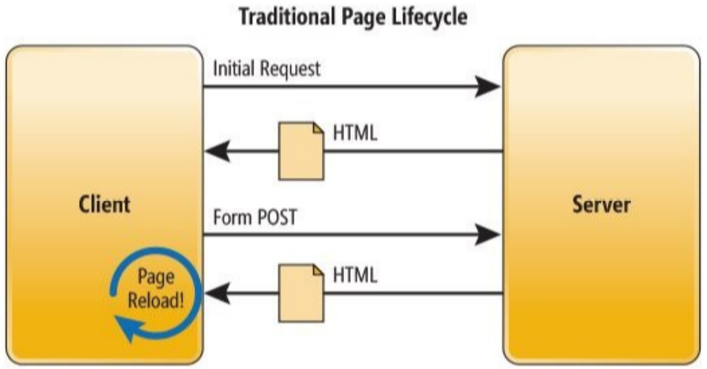
Une application web statique consiste à effectuer des `GET` sur des ressources (pages HTML, fichiers CSS, images, éventuellement fichiers JS) servies par le serveur.



Ce modèle montre rapidement ses limites lorsque l'on veut afficher des éléments à partir de données dynamiques. Le serveur web se contente alors de fournir les ressources.

Application web dynamique serveur

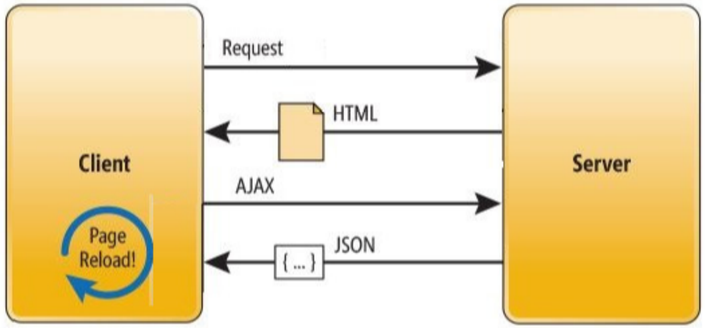
Une première possibilité pour dynamiser une application web consiste à demander au serveur de générer le HTML via un langage de script.



C'est ce qu'il se passe dans une application web PHP classique. Le serveur exécute dynamiquement les scripts contenu dans les pages avant de servir le résultat au client sous forme de HTML.

Application web dynamique AJAX

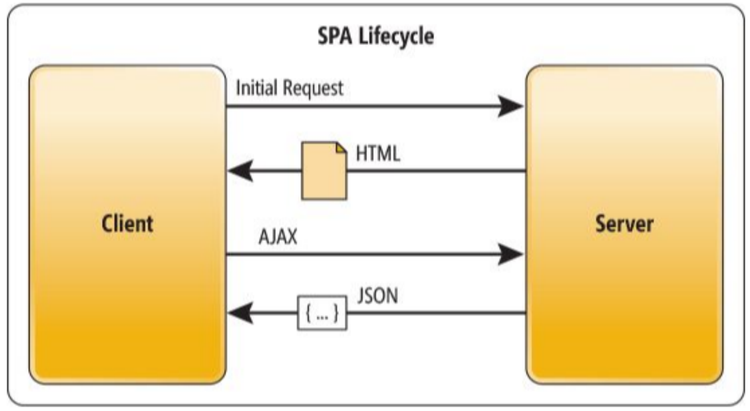
Une deuxième possibilité qui vient souvent en complément du modèle précédent consiste à effectuer une requête au serveur en récupérant des données ou des fragments HTML afin de mettre à jour seulement une partie de la page sans nécessité de recharger toute la page.



L'intérêt de cette technique est la réactivité de l'interface du fait de la réduction du nombres et de la taille des requêtes.

Application web dynamique MV* client

ℹ Un modèle émergeant depuis quelques années consiste à servir uniquement des ressources statiques (principalement du JS) à l'aide d'un serveur web classique. Le client s'occupe alors de générer les pages ainsi que l'arborescence du site. Lorsqu'il est nécessaire de récupérer des données dynamiquement, le client effectue des requêtes HTTP à un serveur d'application.



ANGULARJS / BACKBONE.JS / ember

ℹ Ainsi, un seul *gros* appel est effectué au serveur web afin de récupérer les ressources statiques gérées par l'application. Les appels suivants au serveur d'application ne véhiculent que les données manipulées par l'application.

Choix d'une architecture

Il est important de choisir une architecture pertinente en fonction des frameworks et technologies utilisés. Ce choix demande de l'expérience et des connaissances avancées en infrastructure. En entreprise, cela est généralement géré par des équipes dédiées.

Il est cependant important de comprendre les différentes parties de ces architectures afin de pouvoir être réactif en cas de problème (rôle des serveurs web, emplacement des logs) et également afin de savoir comment livrer votre application web (format de packaging, endroit de copie)