Le Web

Jonathan Lejeune

Sorbonne Université/LIP6-INRIA

SRCS – Master 1 SAR 2019/2020

sources :

- IETF
- Cours de Lionel Médini (Université Lyon 1)

Le web

Qu'est-ce que c'est ?

Un système d'information distribué permettant le partage de ressources interconnectées par des références hypertextes.

Qu'est-ce que ça n'est pas ?

Internet : désigne le réseau physique d'interconnexion mondial

Autour du web

- Dénominations alternatives : World Wide Web, www, W3
- Un organisme de standardisation : le W3C
- Protocoles : HTTP, HTTPs
- Ressources URL, URI, URN
- Langages de présentation : XML, HTML, XHTML, JSON
- Langages de traitement : PHP, ASP, JavaScript, Java, Python, Ruby

Historique du web

Naissance

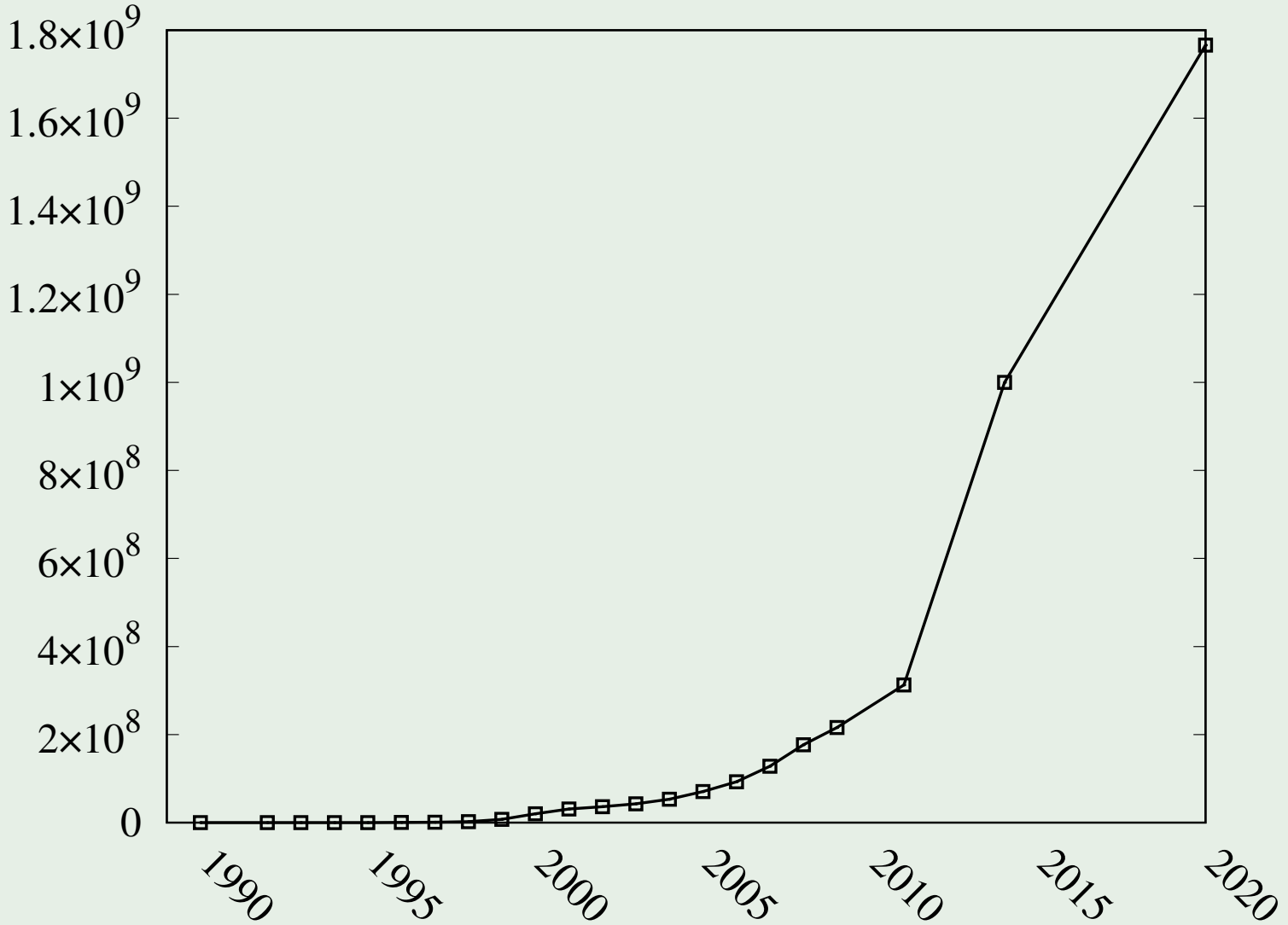
- **1989 (CERN, Genève)** : Tim Berners-Lee développe un système hypertexte pour améliorer la diffusion des informations internes
- **1990** : Robert Cailliau rejoint le projet. Apparition du 1er serveur Web et du 1er navigateur appelé *WorldWideWeb*

Apparition des grands acteurs du Web

- **1993** : Le CERN renonce à ses droits sur le Web qui passe dans le domaine public
- **1994** : Fondation du W3c, de Netscape Navigator
- **1995** : MSN, serveur HTTP Apache, Java, Javascript
- **1998** : Google
- **2001** : Wikipédia
- **2004** : Facebook, Mozilla Firefox

Une utilisation exponentielle

Évolution du nombre de sites web au cours du temps



Une omniprésence du Web

Pourquoi ?

- Accès aux ressources facilités
- Intuitif pour les utilisateurs
- HTTP est devenu un standard
  $\Rightarrow$ Interopérabilité entre les applications

Ressources Web

Définition

Entité susceptible d'être identifiée, nommée, manipulée à travers ses représentations, par quelque moyen que ce soit sur le Web ou n'importe quel SI utilisant les technologies Web.

Exemples

- Contenu d'information multimédia
  - Page web : HTML
  - Image : png, JPEG, Gif
  - Vidéo
- Service
- Concepts abstraits : opérateurs/opérandes, valeurs numériques, objets

Identification de ressources

Uniform **R**esource **I**dentifier (RFC 3986)

- Chaîne de caractères normée identifiant une ressource abstraite ou physique
- Il existe deux types d'URI :
  - Les **U**niform **R**esource **L**ocator
  - Les **U**niform **R**esource **N**ame

URL

URI qui fournit un moyen de localiser la ressource et la manière en précisant le protocole pour y accéder

URN

URI qui identifie une ressource durablement (pendant et même après son existence), indépendamment de sa localisation ou de son accessibilité.

Identification de ressources

Syntaxe des URI

```
scheme://authority/path?query#fragment
```

Exemples d'URI

```
ftp://ftp.is.co.za/rfc/rfc1808.txt
git+ssh://jlejeune@scm.gforge.inria.fr/gitroot/mycloud/mycloud.git
http://www.ietf.org/rfc/rfc2396.txt
https://fr.wikipedia.org/wiki/Paris#Climat
ldap://[2001:db8::7]/c=GB?objectClass?one
mailto:John.Doe@example.com
news:comp.infosystems.www.servers.unix
tel:+1-816-555-1212
telnet://192.0.2.16:80/
urn:oasis:names:specification:docbook:dtd:xml:4.1.2
```

Identification de ressources

Le scheme (schéma)

- Indique le protocole utilisé.
- Toujours suffixé par " : "
- Obligatoire pour un référencement absolu
- Valeurs enregistrées et validées par l'**I**nternet **A**ssigned **N**umbers **A**uthority (**IANA**)
- Exemples :

about : informations de navigation
data : inclusion de données
feed : flux RSS
file : fichiers
ftp : File Transfer Protocol
git : gestion de versions avec Git
http : Hypertext Transfer Protocol
https :Hypertext Transfer Protocol Secure
imap :Internet Message Access Protocol

mailto : adresses email
news : Usenet Newsgroups
pop : POP3
rsync : Synchronisation des fichiers
sftp : SSH File Transfer Protocol
ssh : Secure Shell
tel : Numéros de téléphone
urn : Uniform Resource Names

Identification de ressources

L'authority (autorité)

- Identifie le domaine/l'hôte de la ressource
- Toujours précédé de "//"
- Champ facultatif de l'URI et caractérisé par
  - un nom d'utilisateur (facultatif)
  - une machine hôte (obligatoire) désignée par un nom DNS ou une IP
  - un port d'écoute (facultatif)

`user@host:port`

Le path (chemin)

- Indique le chemin d'accès à la ressource sur l'hôte
- **Obligatoire dans tous les cas** (omission = chemin de la racine)

Identification de ressources

La query (requête)

- Représente une action de requête.
- Élément facultatif de l'URI
- Toujours précédé d'un "?"

Le fragment

- Désigne un aspect partiel d'une ressource.
- Élément facultatif de l'URI
- Exemple : Ancre d'une page Web

Le protocole HTTP

Définition

- HTTP = **H**yper **T**ext **T**ransfer **P**rotocol
- Protocole applicatif de communication client-serveur pour le Web
- RFCs 1945, 2068, 2616, 7230 à 7237
- Port standard : 80, TCP (communication fiable et FIFO)
- Une version sécurisée : https (port tcp 443 par défaut)
- Sérialisation textuelle

Un protocole sans état

- Protocole non connecté
- Gestion légère des transactions
  - $\Rightarrow$ aucune information conservée entre 2 connexions
  - $\Rightarrow$ permet au serveur HTTP de servir plus de clients
- Nécessite un mécanisme de gestion de sessions :

  cookie, Id dans l'URL, champ caché de formulaire

Les différentes versions de HTTP

v0.9 - L'origine (1990)

- Une seule commande : `GET`
- Pas d'en-tête
- Une requête HTTP = une connexion TCP

v1.0 - Améliorations (1996 - RFC 1945)

- Introduction d'en-têtes (échange de "méta" info)
- Utilisation de caches
- Méthodes d'authentification

Les différentes versions de HTTP

v1.1 - Améliorations (1999 - RFC 2068, 2014 - RFC 7230 à 7237)

- Mode connexions persistantes par défaut
  - Plusieurs transactions HTTP (ressources) pour une connexion TCP
  - La connexion est maintenue tant que le serveur ou le client ne décident pas de la fermer
- Introduction des serveurs virtuels

v2.0 - Améliorations (2015 - RFC 7540)

- Fondé sur le protocole SPDY (Google) et rétrocompatible avec 1.1
- Le serveur peut anticiper l'envoi de plusieurs ressources avant la demande du client
- Segmentation des données améliorée
- Multiplexage des requêtes
- Compression des headers

Les rôles définis dans le protocole HTTP

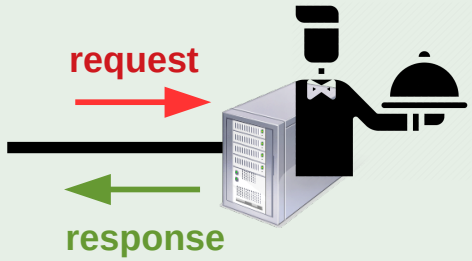
Client

Un programme applicatif qui établit des connexions pour envoyer des requêtes

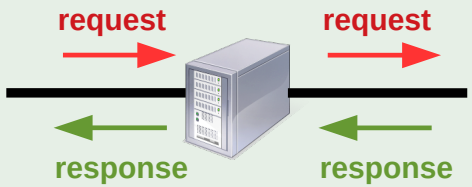
Serveur

Un programme applicatif qui accepte des connexions pour traiter des requêtes et y répondre.

serveurs origine : le serveur hébergeant la ressource faisant l'objet d'une requête



serveur intermédiaire : serveur qui a un rôle de retransmission de requête

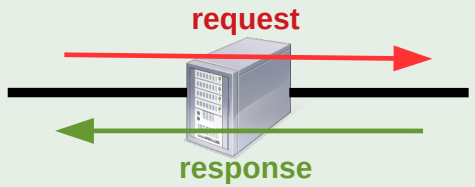


Les types de serveurs intermédiaires définis dans HTTP

Tunnel

Programme intermédiaire faisant un relai entre deux connexions :

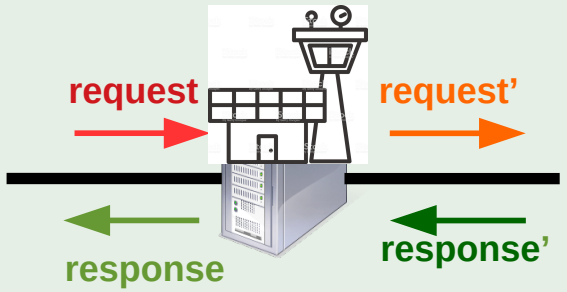
- Invisible d'un point de vue application HTTP
- Peut être initié par une requête HTTP
- N'existe plus dès qu'une des 2 connexions se ferme



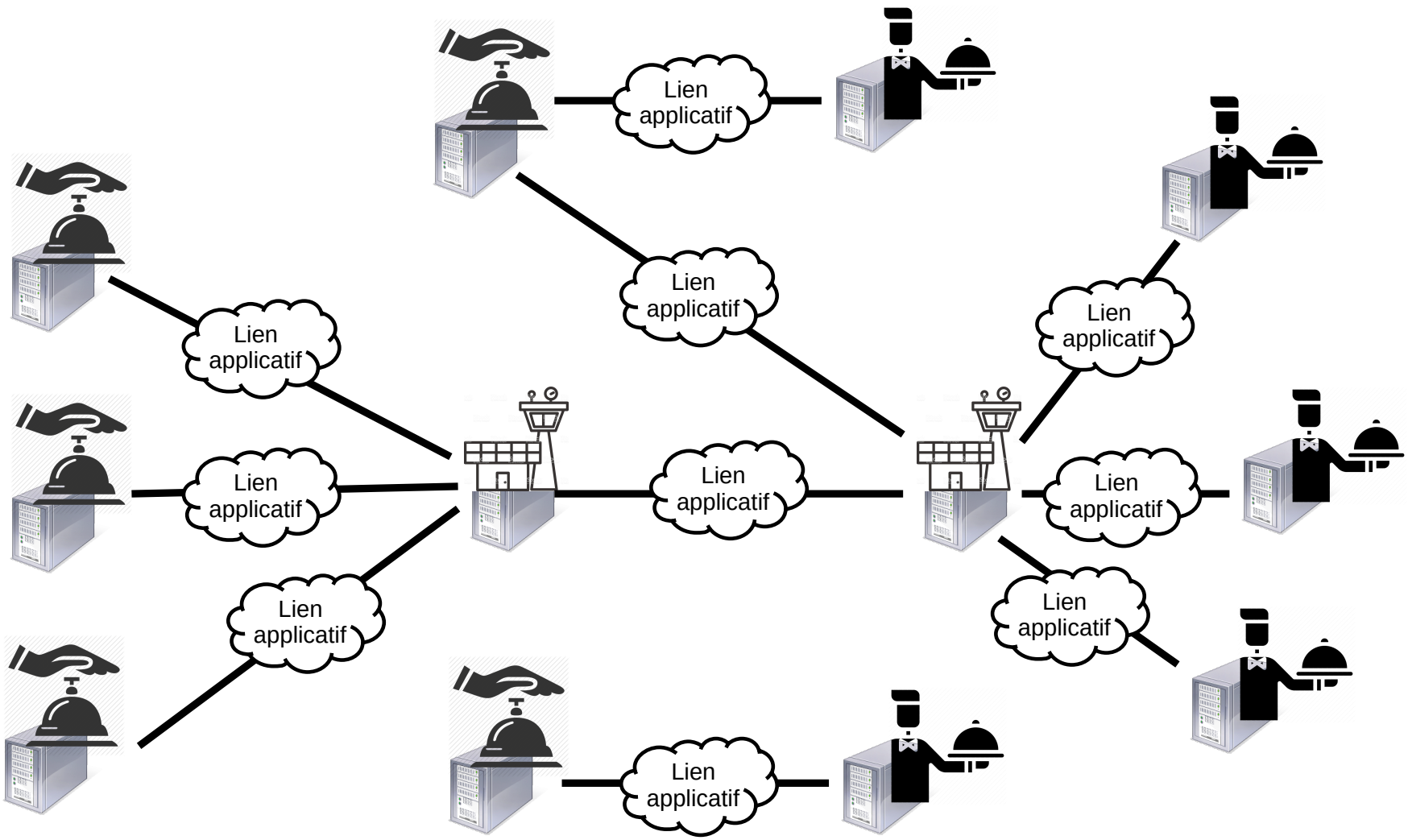
Proxy/Gateway

Programme intermédiaire se comportant à la fois comme un client et un serveur HTTP :

- Contrôle et filtre les échanges (Firewall)
- Peut modifier la requête et la réponse
- Peut servir d'équilibrage de charge pour un ensemble de serveurs origine (reverse-proxy)
- Peut servir de serveur de cache



Réseau applicatif HTTP



Le protocole HTTP fournit un mécanisme de routage applicatif entre un client et un serveur

Format d'une requête HTTP (Client → Serveur)

```
method ressource versionHTTP
en-tête1: valeur_en-tête1
en-tête2: valeur_en-tête2
...
en-têten: valeur_en-têten
(ligne vide)
Contenu du message
```

Première ligne

- Méthodes : `GET`, `POST`, `HEAD`, `PUT`, `DELETE`, `TRACE`, `OPTIONS`, `CONNECT`
- Ressource : identifiée par une URL relative au serveur
- Version HTTP : `HTTP/1.0`, `HTTP/1.1` ou `HTTP/2.0`

Remarques

- Toute donnée est transmise en clair
- L'URL à une taille limitée à 4ko

Méthodes HTTP

La méthode `GET`

- Méthode standard pour requêter une ressource en lecture
  - récupérer un fichier, une image, ...
  - activer un programme en lui transmettant des données
- Le corps (contenu) du message est toujours vide
- Ajout de paramètres dans le champ query de l'URL de la ressource (champs séparés par des &)

```
GET /cgi-bin/f.php?email=toto@site.fr&pass=toto&s=login HTTP/1.1
```

La méthode `HEAD`

- Identique à `GET` mais la réponse ne contiendra que des en-têtes
- Utile pour récupérer des informations sur le serveur

  date de modif, taille de la ressource, type de ressource acceptée, ...
- Le serveur peut ne pas implanter cette fonctionnalité

Méthodes HTTP

La méthode `POST`

Transmission des données dans le corps de la requête :

- champs de formulaire HTML
- poster un message (mailing list, blog, ...)
- créer une nouvelle ressource
- Ajouter des données à une ressource existante

La méthode `PUT`

Permet de créer ou remplacer une ressource par le contenu du message

La méthode `DELETE`

Permet d'effacer une ressource existante

Méthodes HTTP

Autres méthodes

- `CONNECT` : Créer un tunnel avec la cible via un proxy HTTP
- `OPTIONS` : Demander des informations sur les options disponibles sur la ressource cible
- `TRACE` : Réalise un message de test aller/retour en suivant le chemin de la ressource visée.

Classification des méthodes

- **Méthodes safe** : ne changent pas l'état de la ressource
  $\Rightarrow$ `GET`, `HEAD`, `OPTIONS` et `TRACE`
- **Méthodes idempotent** : plusieurs applications ont le même effet qu'une seule application $\Rightarrow$ `PUT`, `DELETE`, `GET`, `HEAD`
- **Méthodes cacheable** : La réponse peut être stockée pour une éventuelle réutilisation $\Rightarrow$ `GET`, `HEAD` (et `POST` éventuellement)

Les en-têtes de requête HTTP

En-têtes de contrôle

- `Host` : domaine + port tcp du serveur (obligatoire pour 1.1)
- `Expect` : Le code de retour attendu par le client
- `Cache-Control` : Indication sur la politique de cache (ex : temps)
- `Max-Forwards` : Nombre maximal de transferts entre proxies
- `Range` : spécifie un intervalle d'octets pour obtenir une partie de la ressource

En-têtes sur le message

- `Content-Length` : Taille du message (utile pour les barres de progression)
- `Content-Encoding` : Type de compression du message
- `Content-Type` : Type MIME de contenu

Les en-têtes de requête HTTP

En-têtes conditionnelles

- `If-Match/If-None-Match` : test de correspondance entre la ressource demandée et la ressource du serveur
- `If-Modified-Since/If-Unmodified-Since` : test de date de modification

Si le test est faux , le serveur répond par une erreur (code 412).

En-têtes de négociation de contenu

- `Accept` : types MIME de la réponse acceptés par le client
- `Accept-Charset` : encodages de caractère acceptés (ex : utf8)
- `Accept-Encoding` : formats de compression du message acceptés
- `Accept-Language` : langues acceptées du message de réponse (ex : fr-FR)

Les en-têtes de requête HTTP

En-têtes d'informations sur la requête

- `From` : l'adresse email du client
- `Referer` : URL d'où l'on vient
- `User-Agent` : le nom du logiciel client (Mozilla, curl, ...)

En-têtes de session

- `Authorization` : Identifiants pour l'authentification HTTP
- `Proxy-Authorization` : Identifiants pour se connecter à un proxy HTTP
- `Cookie` : Un cookie HTTP envoyé précédemment par le serveur

Format d'une réponse HTTP (Serveur → Client)

```
versionHTTP codeReponse significationCodeReponse
en-tête1: valeur_en-tête1
en-tête2: valeur_en-tête2
...
en-têten: valeur_en-têten
(ligne vide)
Contenu du message
```

Première ligne

- Version HTTP : `HTTP/1.0`, `HTTP/1.1` ou `HTTP/2.0`
- Code de Réponse (sur 3 chiffres) :
  - Indique le résultat de la requête : succès ou échec
  - En cas d'échec, le contenu de la réponse en décrit la raison (ex : fichier non présent, problème de droit)
- Signification : une valeur définie par code de réponse

Codes de réponses

Classifications des codes

- De 100 à 199 : information
- De 200 à 299 : succès
- De 300 à 399 : redirection
- De 400 à 499 : erreur à cause du client
- De 500 à 599 : erreur à cause du serveur

Exemples

```
100 Continue
200 OK
201 Created
202 Accepted
301 Moved Permanently
400 Bad Request
401 Unauthorized
402 Payment Required
403 Forbidden
404 Not Found
500 Internal Server Error
501 Not Implemented
502 Bad Gateway
503 Service Unavailable
```

Les en-têtes de réponses HTTP

En-têtes similaires aux requêtes

`Cache-Control`, `Content-Encoding`, `Content-Language`, `Content-Length`, `Content-Range`, `Content-Type`

En-têtes les plus communs

- Last-Modified : date de dernière modification
- Allow : méthodes autorisées pour ce document
- Expires : date d'expiration de la ressource
- Date : date de la requête
- Server : type du serveur

Exemple d'interaction client-serveur HTTP

```
GET /Jonathan.Lejeune/index.html HTTP/1.1
Host: pages.lip6.fr
User-Agent: curl/7.58.0
Accept: */*
```

pages.lip6.fr

```
HTTP/1.1 200 OK
Date: Sun, 26 Apr 2020 12:15:05 GMT
Server: Apache/2.4.38 (Debian)
Vary: Accept-Encoding
Content-Length: 3538
Content-Type: text/html; charset=UTF-8

<!doctype html>
<html lang="fr">
...
```

Cookies

Définition

- Chaîne de caractères url-encodée de 4ko max stockée sur le disque dur du client
- Informations associées à un ensemble d'URL, utilisées lors de toute requête vers l'une de ces URL

À quoi ça sert ?

- propager un code d'accès : évite une authentification lors de chaque requête
- identification dans une base de données
- fournir des éléments statistiques au serveur : compteurs de pages visitées...
- Maintenir une session

Cookies

Directive coté serveur

En-tête `Set-Cookie` dans la réponse HTTP lors de la 1ère connexion

- String url-encodée de 4ko max stockée sur le disque dur du client
- Informations associées à un ensemble d'URL, utilisées lors de toute requête vers l'une de ces URL

Utilisation d'un cookie par le client

- Avant chaque requête, le client vérifie dans sa liste de cookies s'il y en a un qui est associé à cette requête
- Si c'est le cas, le client utilise la directive Cookie dans l'en-tête de la requête HTTP

Cookies

Format envoyé par le serveur

```
Set-Cookie: nom=valeur; expires=date; path=chemin_acces;
            domain=nom_domaine; secure
```

- `nom=valeur` : contenu du cookie (champ obligatoire)
- `expires` : devient invalide après la date d'expiration
- `path=/pub` : valable pour toutes les requêtes dont l'URL contient `/pub`
- `domain` : nom de domaine du serveur pour lequel le cookie est valable
- `secure` : tag indiquant que le cookie n'est valable que sur une connexion sécurisée (HTTPs)

Format envoyé par le client

```
Cookie: nom1=valeur1; nom2=valeur2; ...
```

Encodage des ressources

Problématique

- Un serveur Web peut servir différents types de ressources :

  texte, pages Web, images, documents, fichiers exécutables, ...
- Chaque type de ressource est codé de façon différente
- Un client doit connaître le type de ressource pour pouvoir la traiter.

  visualisation dans un navigateur, utilisation d'un plugin, application externe

Solution : le type MIME

Le standard **M**ulti-purpose **I**nternet **M**ail **E**xtensions qui définit un identifiant normalisé pour chaque format de données.

Composition du type MIME

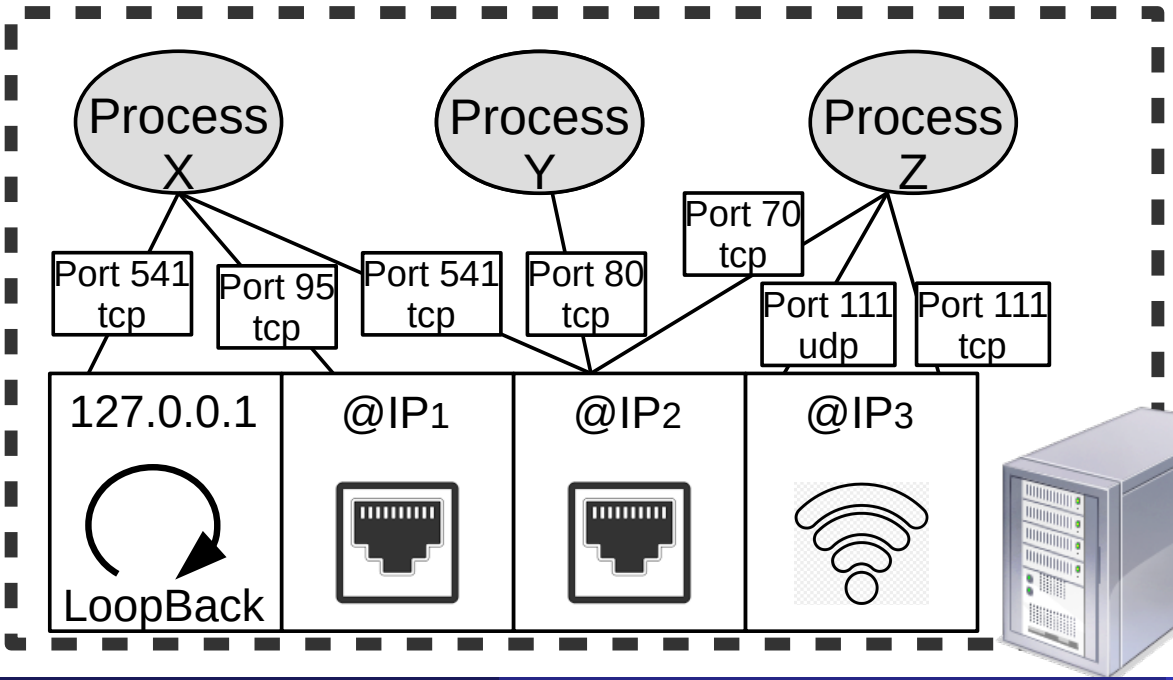
- Un type général : `text`, `image`, `audio`, `video`, `application`, `font`, `model`, ...
- Un sous-type qui dépend du type général : `image/gif`, `application/pdf`, `text/plain`, `text/html`, `text/xml`, `video/mpeg`, ...

Serveur (Rappel)

Qu'est-ce qu'un serveur ?

Processus rendant un service continu et accessible à distance via une socket caractérisée par :

- **une adresse IP** attribuée à la machine hôte
- **un numéro de port**
- **un protocole de transport** : TCP ou UDP



N.B. : l'adresse 0.0.0.0 désigne toutes les IP de la machine hôte

Machine hôte

Serveur HTTP

Objectif d'un Serveur HTTP

Interpréter une URI et une méthode pour produire une réponse HTTP.

Contenu type d'un fichier de configuration d'un serveur HTTP

- Les **adresses d'écoute** (IP+port) du serveur (en `http` ou en `https`)
  $\Rightarrow$ scheme et l'authority des URIs qui pourront être traitées
- Un **répertoire de travail** sur le système de fichiers de la machine hôte
  $\Rightarrow$ racine des path des URIs qui pourront être traitées

Notion de serveur virtuel

- Un processus serveur HTTP peut correspondre à plusieurs noms de domaines
- Un serveur virtuel = un nom de domaine + une adresse d'écoute + un répertoire de travail

Algorithme simplifié d'un serveur HTTP

```
Attendre requête HTTP sur l'ensemble des adresses d'écoute
Parser la requête
  => répondre erreur 400 si requête mal formée
Extraire première ligne (méthode HTTP + URI + version)
  => répondre erreur 505 si version non supportée
  => répondre erreur 414 si URI trop longue
Extraire les en-têtes
  => répondre erreur 431 si en-têtes trop volumineux
Extraire le message
  => répondre erreur 411 si content-length requis et absent
Vérifier que la méthode et l'URI sont valides
  => répondre erreur 404 si la ressource de l'URI n'existe pas
  => répondre erreur 403 si pas les droits d'accès à la ressource
  => répondre erreur 405 si méthode non définie sur la ressource
Traiter la requête
  => répondre succes ou erreur
```

Serveur HTTP : Traitement d'une requête

Cas où le path de l'URI désigne un fichier

- Cas GET :
  - si fichier non interprétable par le serveur : réponse = contenu du fichier
  - si fichier interprétable par le serveur : exécuter l'interpréteur qui génère la réponse
- Cas POST et fichier interprétable : l'interpréteur agit en fonction du contenu du message
- Cas `DELETE` : effacer le fichier
- Cas `PUT` : Créer/écraser le fichier avec le contenu du message

Cas où le path de l'URI désigne un composant Web

Le traitement et la génération de la réponse sont délégués au composant.

Exemple type de composant : Servlet Java, service web

Fichier non interprétable vs. Fichier interprétable

Fichier non interprétable

Fichier contenant des données brutes non exploitables dans le traitement d'une requête $\Rightarrow$ texte brut, image, page html statique, ...

Fichier interprétable

Fichier contenant du code interprétable par le serveur.

- Page HTML dynamique :
  - **S**erver **S**ide **I**ncludes (SSI) : directives simples intégrées dans du HTML
- Fichier de script exécutable directement par le serveur :
  - PHP, ASP, JavaScript
- Fichier exécutable quelconque via **C**ommon **G**ateway **I**nterface (CGI)
  - Interface normalisée de communication entre le processus serveur et un processus externe qui traitera la requête
  - Indépendant du langage de programmation
    - Langages les plus courants : Perl, Bash, Python, C

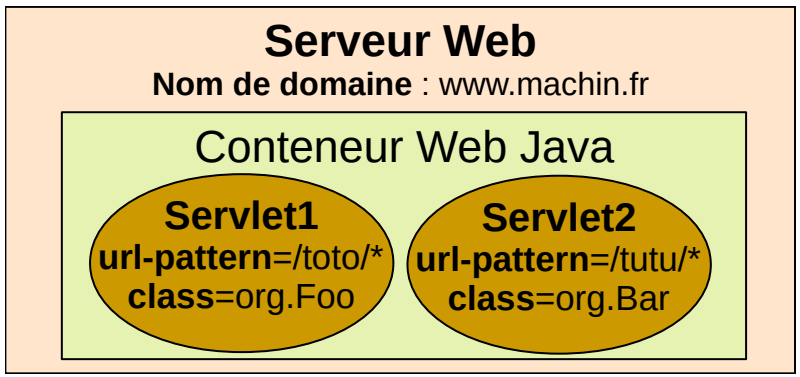
Servlet Java

Qu'est-ce que c'est ?

Composant Web définit par une classe Java et qui s'exécute dans un container Web JavaEE.

Avantages/Inconvénients

- ✓ Java = JVM + multi plateforme
- ✓ JavaEE = standard stable et reconnu
- ✓ Pas forcément limité au protocole HTTP
- ✓ Extensible à d'autres technologies Java (RMI, JDBC, JSP, ...)
- ✗ Complexe à prendre en main



Servlet Java en pratique

Une classe Java

```
package org.srcs;
public class FooServlet extends HttpServlet {
 public void doGet(HttpServletRequest req, HttpServletResponse res)
 {/*traitement get*/}
 // doPut, doPost, doHead, doDelete, doOptions, doTrace
 // sont également possibles
}
```

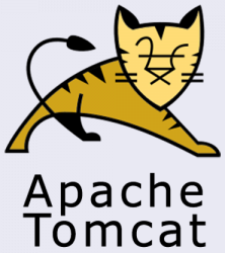
Un fichier de description XML (fichier web.xml)

```
<web-app version="3.1">
  <servlet>
    <servlet-name>foo</servlet-name>
    <servlet-class>org.srcs.FooServelet</servlet-class>
  </servlet>
  <servlet-mapping>
     <servlet-name>foo</servlet-name>
     <url-pattern>/*</url-pattern>
  </servlet-mapping>
</web-app>
```

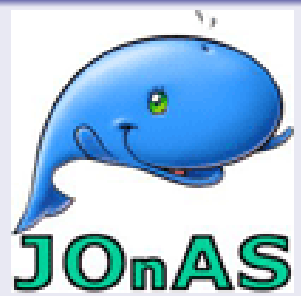
Serveur Web (des plus simples aux plus lourds)

Les serveurs Http classiques (et les plus populaires)

Les serveurs Http déployant des servlets Java



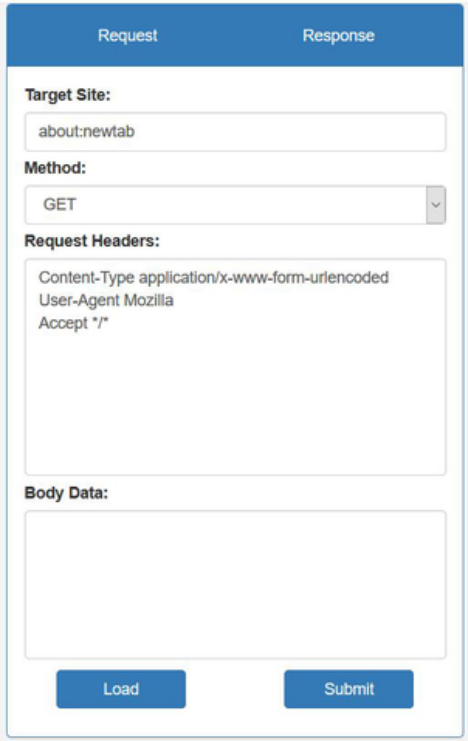
Les serveurs d'applications JavaEE

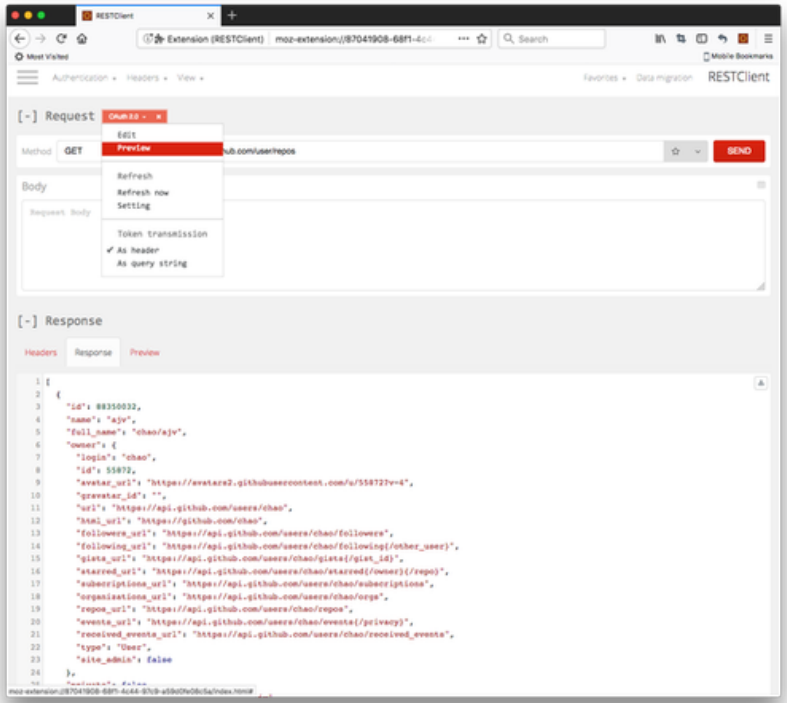


Mettre en œuvre un client HTTP

Via un navigateur

- Requête méthode GET de base dans la barre d'outils
- Autre type de méthode en installant un plug-in
  - HTTP request maker
  - RESTClient





Mettre en œuvre un client HTTP

Via la commande `curl` (client **URL**)

- Utilitaire pour envoyer une requête sur une URL en ligne de commande
- Affichage de la réponse sur la sortie standard
- Supporte la plupart des protocoles standards : HTTP, FTP, ...

```
curl [options] URL
```

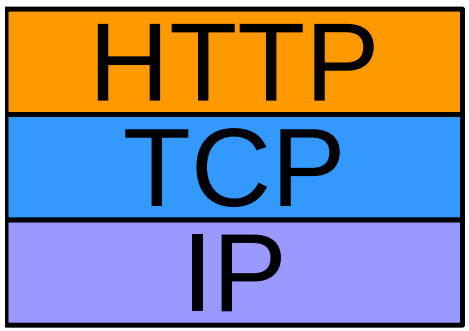
Options

- `-X <method>` : indique la méthode HTTP (par défaut `GET`)
- `-http1.1` ou `-http1.0` ou `-http2` : version de HTTP (par def. `1.1`)
- `-i` : inclure l'en-tête de réponse sur la sortie standard
- `-H "nameHeader: valHeader"` : ajouter un entête dans la requête
- `-d "contenu"` : ajouter un contenu de message
- `-v` : mode verbeux, affiche la requête envoyée

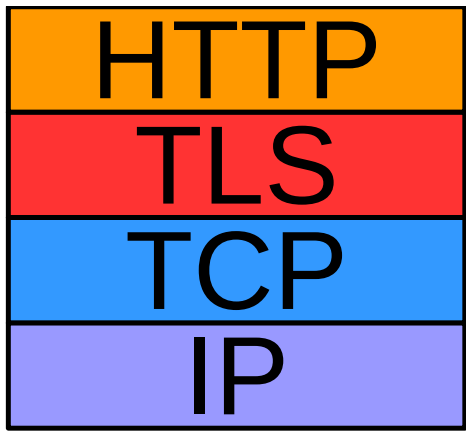
Un mot sur HTTPs (RFC 2818)

Principe

- Insertion d'un protocole de sécurisation entre TCP et HTTP
  - Historiquement : SSL (**S**ecure **S**ocket **L**ayer)
  - Aujourd'hui : TLS (**T**ransport **L**ayer **S**ecurity)
- Permet d'authentifier le serveur, chiffrer les échanges, assurer l'intégrité des données



Pile HTTP



Pile HTTPs