

Technologies et langages pour le Web

Coté serveur

Yannick Loiseau

Université Blaise Pascal

Licence Informatique 2^{me} année

Qu'est-ce que le Web ?

Web \neq Net

~~site internet~~

Internet

Réseau de réseaux \Rightarrow interconnexion de machines

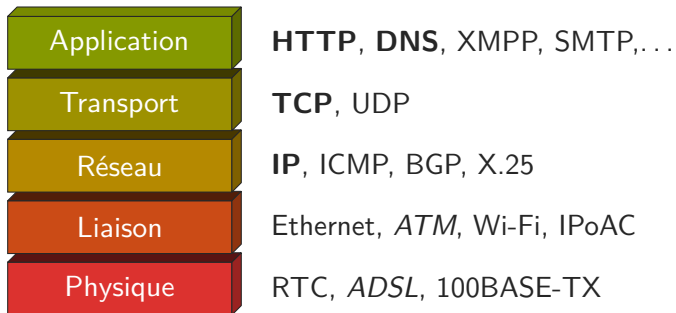
Buts

- tolérance aux pannes
- abstraction de l'infrastructure physique
- « intelligence » à la périphérie (neutralité)

⇒ protocoles en couches

- isolation
- indépendance des niveaux → évolution
- ⇒ connaissance limitée du système global

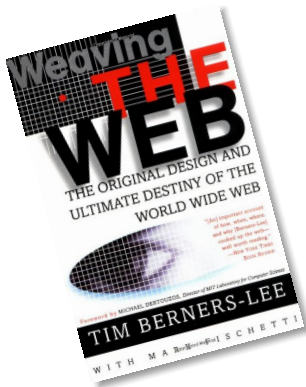
ARP



Suppose all information stored on computers everywhere were linked. Suppose I could program my computer to create a space in which anything could be linked to anything. There would be a single, global information space.

— *Tim Berner-Lee (1980)*

Weaving the Web (2000)



Tim Berners-Lee



Le Web

Interconnexion de ressources \Rightarrow hypermédia

\Rightarrow couche applicative

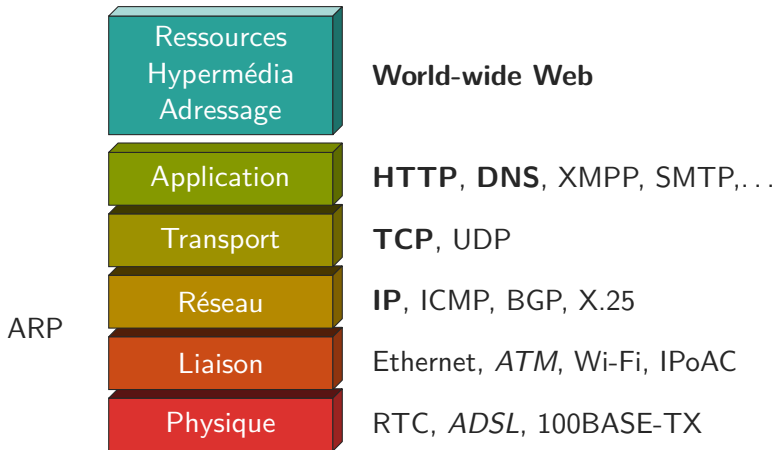
Buts

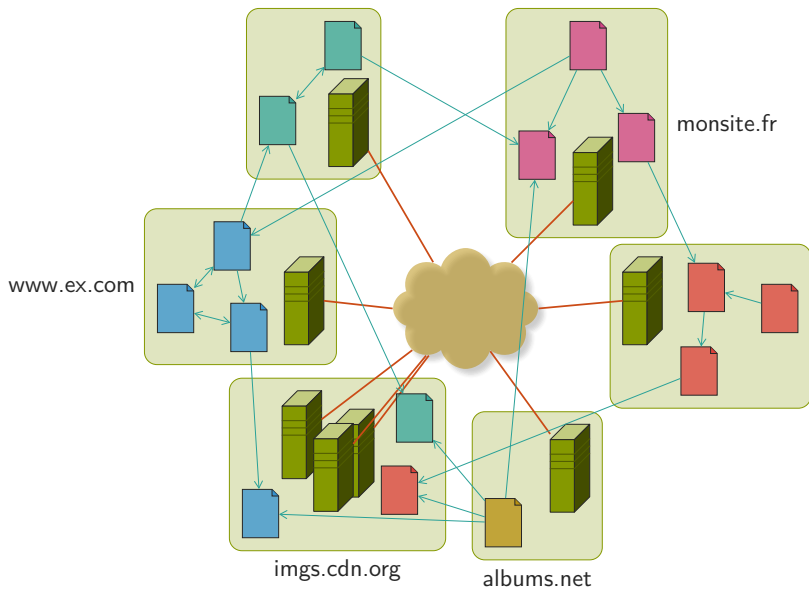
Système d'informations

- non/semi structuré
- global
- à grande échelle
- décentralisé
- non supervisé
- tolérant aux pannes
- extensible (évolutif)

⇒ contraintes architecturales FIELDING 2000

- client/serveur
- faible couplage : client générique, hypermédia
- ressources
- représentation homogènes (HTML)
- protocole : interface uniforme (HTTP)
- adressage uniforme : indépendance (URL)





Site web ?

- ≠ machine
- ≠ domaine
- apparence ?
- ⇒ ensemble de ressources « cohérentes »

⇒ ressources / liens

Tim Berner-Lee



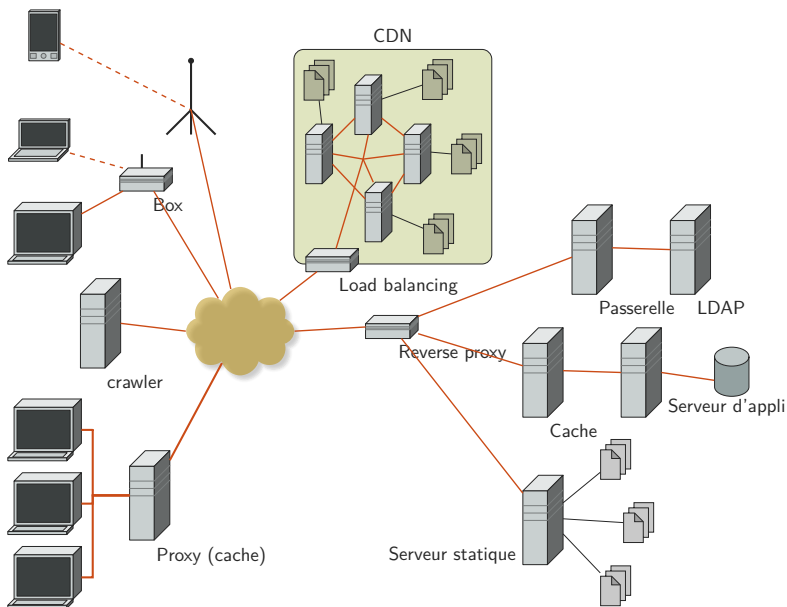
- Hypermédia : 1962-65 (D. Engelbart, T. Nelson)
- T. Berner-Lee (CERN) : ENQUIRE (1980) ; proposition (1989)
- HTTP/0.9, HTML, httpd, navigateur (CERN 1990-91)

Roy Fielding (UCI)



- Le CERN « libère » le Web en 1993 ; W3C (1994)
- HTTP (1996-99), URI (1998), Apache httpd (1995), libwww(-perl)
- REST (PhD Th. 2000)

Acteurs et composants du Web



Définition (Agent)

- client
- effectue les requêtes
- manipule les ressources (traitement, affichage)
- maintient l'état

Exemple

- navigateur (firefox, chrome, opera, IE)
- agrégateur (RSS, Podcasts)
- robots (indexation, extraction)
- téléchargement (wget, miroirs, ...)
- applications spécifiques (clients Twitter, Facebook, etc.)

Définition (Serveur)

- répond aux requêtes du client
- gère les ressources
 - état, création, destruction
 - représentations (génération)
 - adressage
- serveur statique / serveur d'application

Exemple

- Apache, Nginx, IIS, lighttpd, ...
- mod_php, node.js, tomcat, glassfish, gunicorn, ...

Définition (Passerelle)

- conversion de protocoles/formats
- à la fois serveur et client
- générique

≠ serveur d'appli. → logique métier *ad hoc*

Exemple

`mod_proxy_ftp`, `mod_proxy_ajp`

Définition (Cache)

- mémorisation des réponses
- performances
- tolérances pannes
- local \Rightarrow privé (navigateur)
- intermédiaire \Rightarrow partagé (proxy)
- \Rightarrow mutualisation

Exemple

Varnish, mod_cache

Définition (Proxy)

- intermédiaire
- explicite / transparent
- partage
- filtrage et contrôle d'accès (parental, pubs, ...)
- conversion (optim. mobile)
- cache
- masquage de source

Exemple

Squid, polipo, mod_proxy_http,

Définition (Reverse Proxy)

- intermédiaire coté serveur
- répartition de charge
- portail / intégration
- relais

Exemple

HAProxy, Apache, nginx, Pound, mod_proxy, mod_proxy_html, mod_proxy_balancer

Définition (CDN)

- *content delivey network*
- ensemble de serveurs
- miroirs
- répartis géographiquement
- public / privé
- \Rightarrow échelle

Exemple

Amazon, Google, Azure, OVH, ...

Hypermédia

Définition (Hypermédia)

- information unitaire : ressource
- information structurée : liens
- non linéaire (graphe)
- « navigation » non séquentielle
- inclusion (*transclusion*)
- \Rightarrow référence

Définition (Ressource)

- élément d'information
- abstrait $\Rightarrow \neq$ représentations
- adressable \Rightarrow URI

URI

Identifiant global unique pour une ressource

- URN : Uniform Resource Name
- URL : Uniform Resource Locator

URN

Exemple

- ▶ `urn :ietf :rfc :2141`

URL

(RFC3986)

Définition (URL)

Identifiant de ressource

- unique : espace de nom par le DNS
- définie par le serveur : sans gestion centrale
- déréléférençable : suffisant pour retrouver le document
- opaque : compréhension uniquement pas le serveur

comment (protocole) où (serveur) quoi (ressource locale au serveur)

http ://www.example.com :80 /foo/bar ?a=1&b=1 #ici

► scheme ●

► hostname ●

► port ●

► path ●

► query string ●

► fragment ●

Cool URIs don't change

URIs don't change : people change them

`http://www.w3.org/Provider/Style/URI`

- uri opaque
- sémantique serveur
- \Rightarrow contrôle total
- \Rightarrow découplage client/serveur

`http ://www.example.net/index.php?type=article&id=1234`

- ▶ `http ://www.example.net/articles/1234`
- ▶ `http ://www.example.net/article-1234`
- ▶ `http ://www.example.net/a531ae42-84c1-455e-86a3`

Conf. Apache :

```
RewriteEngine On  
RewriteRule ^([a-z]+)/([0-9]+)$ index.php?type=$1&id=$2
```

http://httpd.apache.org/docs/current/mod/mod_rewrite.html

`http ://www.example.net/monimage.jpg`

`http ://www.example.net/monimage`

Conf. Apache :

Options +Multiviews

http://httpd.apache.org/docs/current/mod/mod_negotiation.html

Principe : Design d'URL

- réflexion a priori
- espace d'adressage abstrait \Leftrightarrow représentation physique
- indépendante des info. variables

chemin, nom, domaine

- ▶ propriétaire/auteur (~john/article)
- ▶ techno (.php, /cgi-bin/,...)
- ▶ format (.html, .jpg,...)
- ▶ titre, catégorie, structure de l'organisation
⇒ redirection URL canonique
- ▶ accès (public, private)
- ▶ status (draft, etc.)
 - ✓ latest, today → uri canonique

URI \equiv API

- courte
- facile à retenir
- facile à écrire, dicter,
- « bidouillable »

DNS



Niveau réseau : identification des machines par l'adresse IP

Exemple

93.184.216.34

- IP dépendante de l'architecture réseau
- difficile à retenir

DNS

- niveau d'indirection
- résolution IP \Leftrightarrow nom de domaine
- indépendance du réseau
- système hiérarchique
 - serveurs
 - structure

Exemple

`www.example.net` \Leftrightarrow `93.184.216.119`

```
$ host www.twitter.com  
www.twitter.com is an alias for twitter.com.  
twitter.com has address 104.244.42.129  
twitter.com has address 104.244.42.193
```

```
$ host www.stackoverflow.com  
www.stackoverflow.com is an alias for stackoverflow.com.  
stackoverflow.com has address 151.101.65.69  
stackoverflow.com has address 151.101.193.69  
stackoverflow.com has address 151.101.1.69  
stackoverflow.com has address 151.101.129.69
```

- répartition de charge : 1 nom \leftrightarrow n IP
- virtual host : n nom \leftrightarrow 1 IP

Représentations

Définition (Représentation)

- données
- méta-données
- \Rightarrow état de la ressource

- ressource : multiples représentation
- \neq format : *media type*
- même URL
- \Rightarrow négociation

type \Leftrightarrow utilisation

- traitement automatique
- affichage

Exemple (Blog)

affichage (HTML) ou agrégation (RSS)

Exemple (Tableau de données)

- tableur (CSV)
- affichage (HTML)
- visualisation (SVG, jpeg, ...)
- ...

- RFC 2046
- IANA
- <http://www.iana.org/assignments/media-types/>

type/soustype

- ▶ application
 - ▶ xhtml+xml
 - ▶ json
 - ▶ pdf
 - ▶ gzip
 - ▶ octet-stream
- ▶ audio
 - ▶ mpeg
 - ▶ webm
- ▶ example
- ▶ image
 - ▶ png
 - ▶ jpeg
 - ▶ svg+xml
- ▶ message
 - ▶ http
 - ▶ rfc822
- ▶ model
- ▶ multipart
- ▶ text
 - ▶ plain
 - ▶ csv
 - ▶ html
- ▶ video
 - ▶ mpeg
 - ▶ webm

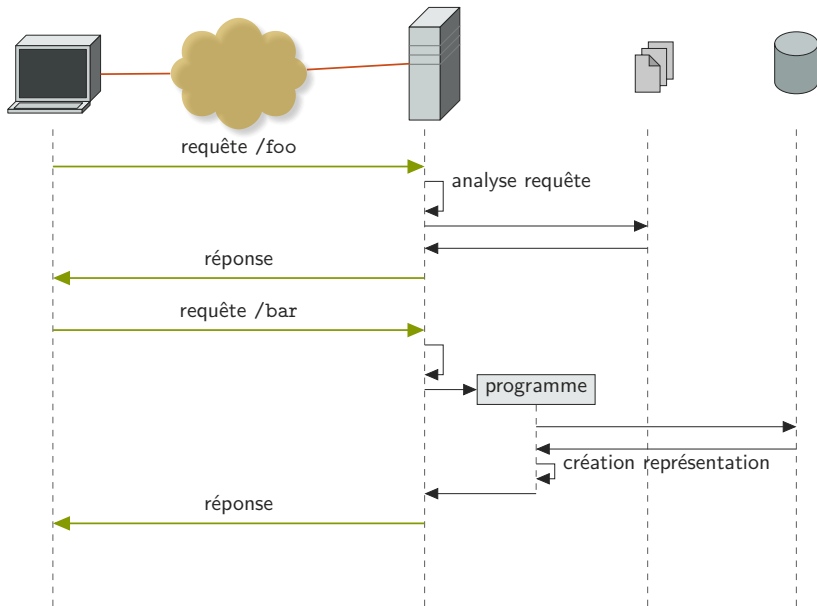
- \Rightarrow hiérarchique
- valeur par défaut (`text/*`)
- gestion par le client \Rightarrow générique/extensible
- pondération \rightarrow négociation
- requête / réponse

Protocole HTTP

Hypertext Transfert Protocol : (RFC2616)
RFC7230, RFC7231, RFC7232, RFC7233, RFC7234, RFC7235

Manipulation des représentations

Principe



Client/Serveur

- séparation des responsabilités
- Données – (Traitements) – Présentation
- portabilité de l'interface
- passage à l'échelle du serveur
- évolution indépendante

Sans état

- requête suffisante pour répondre
- pas de contexte serveur
- \Rightarrow intermédiaires
- état sur le client

Sans état

- ▶ visibilité : monitoring → requête (intermédiaires)
- ▶ fiabilité : reprise sur panne plus simple
- ▶ évolutivité (échelle) :
 - ▶ pas d'état \Rightarrow moins de ressource
 - ▶ implém. plus simple
- ▶ possibilité d'intermédiaires (couches)

Sans état

- ▶ performances : répétition d'informations
- ▶ consistance : pas de contrôle serveur → implém. / comportement clients

Caches

- réponse cachable
- sémantique claire
- marqueur

Caches

↘ interactions \Rightarrow ↗ performances

- efficacité
- échelle
- latence
- robustesse

↘ fiabilité

Interface uniforme

- découplage service ↔ implémentation
- évolution indépendante

Contraintes (REST)

- identification des ressources
- manipulation via représentations
- messages auto-descriptifs
- état conduit par hypermédia (HATEOAS)

Couches

- système en couches hiérarchiques
- \Rightarrow intermédiaires
- visibilité limitée
- indépendance
- passage à l'échelle

interface uniforme + couches \Rightarrow *pipe and filter*

composition des composants

flot de données

Messages HTTP

start line

en-têtes

ligne vide
corps

version, type de message,...
Nom : valeur Nom : valeur ...
...

Requêtes

Requête

start line : verbe identifiant version

Exemple

```
GET /foo HTTP/1.1
```

Méthodes de type CRUD : RFC7231

- Create : POST
- Read : GET (HEAD)
- Update : PUT – PATCH (RFC5789)
- Delete : DELETE

Propriétés des méthodes

- idempotence
- sans effet de bord

Définition (Idempotence)

n requêtes \equiv 1 requête

Définition (sans effet de bord)

pas de modification de l'état de la ressource

- indépendance des intermédiaires
- « cachabilité »
- reprise sur erreur
- automatisation / responsabilité

GET /articles/1234?action=delete

Méthode	Idempotente	Sans effet de bord
GET	✓	✓
HEAD	✓	✓
PUT	✓	✗
DELETE	✓	✗
POST	✗	✗
PATCH	✗	✗

À propos du POST

Pas de sémantique bien définie

- POSTa : append (commentaire, message de forum)
- POSTp : process (traitement quelconque)

RFC2310 : en-tête **Safe** : (yes|no)

- ▶ **OPTIONS** : méta-informations
- ▶ **CONNECT** : tunnel
- ▶ **TRACE** : echo

start line

en-têtes

ligne vide
corps

version, type de message,...
Nom : valeur Nom : valeur ...
...

Réponse

start line : version status message

Exemple

HTTP/1.1 200 OK

5 catégories :

- ▶ 1xx : Informations (100 Continue, 101 Switching Protocols)
- ▶ 2xx : Succès (200 OK, 201 Created, 204 No Content)
- ▶ 3xx : Redirection (301 Moved Permanently, 304 Not Modified)
- ▶ 4xx : Erreur du client (404 Not Found, 401 Unauthorized, 418 I'm a teapot¹)
- ▶ 5xx : Erreur du serveur (500 Internal Server Error, 504 Gateway Timeout)

1. HTCPCP : RFC2324

2xx : Succès

```
GET /example HTTP/1.1
Host : www.example.com
User-Agent : Mozilla/5.0
[...]
```

```
HTTP/1.1 200 OK
Date : Sat, 22 Dec 2012 00 :00 :00 GMT
Server : Apache
Content-Type : text/html
Content-Length : 1270
[...]
```

POST /articles/ HTTP/1.1

Host : www.example.com

Content-Type : text/plain

Content-Length : 13

Hello World !

HTTP/1.1 201 Created

Date : Sat, 22 Dec 2012 00 :00 :00 GMT

Location : /articles/42

GET /articles/42 HTTP/1.1

Host : www.example.com

HTTP/1.1 200 OK

Date : Sat, 22 Dec 2012 00 :00 :05 GMT

Content-Type : text/plain

Content-Length : 13

Hello World !

POST /articles/42 HTTP/1.1

Host : www.example.com

Content-Type : text/plain

Content-Length : 6

Salut

HTTP/1.1 200 OK

Date : Sat, 22 Dec 2012 00 :00 :20 GMT

Content-Type : text/plain

Content-Length : 19

Hello World !

Salut

POST /calculator HTTP/1.1

Host : www.example.com

[donnees de calcul]

HTTP/1.1 202 Accepted

Date : Sat, 22 Dec 2012 00 :00 :00 GMT

Location : /calculator/status/42

POST /search HTTP/1.1

Host : www.example.com

[requete complexe]

HTTP/1.1 303 See Other

Date : Sat, 22 Dec 2012 00 :00 :00 GMT

Location : /articles/42

GET /articles/42 HTTP/1.1

HTTP/1.1 200 OK

Date : Sat, 22 Dec 2012 00 :00 :02 GMT

Content-Type : text/plain

Content-Length : 19

Hello World !

Salut

DELETE /articles/42 HTTP/1.1

Host : www.example.com

HTTP/1.1 204 No Content

Date : Sat, 22 Dec 2012 00 :01 :00 GMT

GET /articles/42 HTTP/1.1

Host : www.example.com

HTTP/1.1 410 Gone

Date : Sat, 22 Dec 2012 00 :01 :02 GMT

4xx : Erreur du client

PATCH /articles/42 HTTP/1.1

Host : www.example.com

[...]

HTTP/1.1 405 Method Not Allowed

Date : Sat, 22 Dec 2012 00 :01 :00 GMT

Allow : OPTIONS, GET, HEAD, POST, DELETE

Content-Length : 0

```
PUT /articles/42 HTTP/1.1
Host : www.example.com
Content-Type : application/vnd.ms-excel
[...]
```

```
HTTP/1.1 415 Unsupported Media Type
Date : Sat, 22 Dec 2012 00 :01 :00 GMT
Content-Length : 0
```

```
PUT /articles/42 HTTP/1.1
Host : www.example.com
Content-Type : text/plain
[...]
```

```
HTTP/1.1 428 Precondition Required
Date : Sat, 22 Dec 2012 00 :01 :00 GMT
Content-Length : 0
```

Négociation de contenu

- choix de la représentation
- dialogue client ↔ serveur
- RFC2295

En-têtes

- **Accept** : type mime
- **Accept-Encoding** : compression (**gzip**, ...)
- **Accept-Language** : langue (**fr**, **en**, ...)
- **Accept-Charset** : jeux de caractères (**utf-8**, ...)
- **Accept-Features** : fonctionnalités
- **Negotiate** : **trans**, **vlist**, *

Accept-* : *valeur* ;q=*poid*, *valeur*...

Exemple

Accept : text/html, application/xhtml+xml;q=0.8,
application/pdf;q=0.6, text/*;q=0.4, */*;q=0.1

- **Content-Type** : type mime
- **Content-Encoding** : compression
- **Content-Language** : langue
- **Content-Location** : uri de la représentation choisie

- ▶ Alternates
- ▶ TCN : list, choice, adhoc
- ▶ Vary : negotiate, accept-language, accept-encoding

Codes de réponse

- 300 [Multiple Choices](#)
- 406 [Not Acceptable](#)

Exemples

Conf. Apache

Options +Multiviews

- `articles/a1234.en.html`,
- `articles/a1234.fr.html`,
- `articles/a1234.en.html.gz`,
- `articles/a1234.en.pdf`,
- ...

```
HEAD /articles/a1234 HTTP/1.1
```

```
Accept : text/html
```

```
Accept-Encoding : gzip
```

```
HTTP/1.1 200 OK
```

```
Content-Type : text/html ; charset=iso-8859-1
```

```
Content-Language : en
```

```
Content-Encoding : gzip
```

```
Content-Location : /articles/a1234.en.html
```

```
Vary : negotiate,accept-language
```

```
[...]
```

```
HEAD /articles/a1234 HTTP/1.1
```

```
Accept : application/pdf
```

```
Accept-Language : fr
```

```
HTTP/1.1 200 OK
```

```
Content-Type : application/pdf
```

```
Content-Language : fr
```

```
Content-Location : /articles/a1234.fr.pdf
```

```
Vary : negotiate,accept-language
```

```
[...]
```

```
GET /articles/a1234 HTTP/1.1
Accept : application/json
```

```
HTTP/1.1 406 Not Acceptable
Alternates : {"/articles/a1234.en.html" 1 {type text/html} {charset utf-8}
              {language en} {length ...}},
              {"/articles/a1234.fr.pdf" 0.9 {type application/pdf} {charset utf-8}
              {language fr} {length ...}}
...
Vary : negotiate, accept, accept-language
TCN : list
[...]
```

```
GET /articles/a1234 HTTP/1.1
```

```
Accept : */*
```

```
Negotiate : trans
```

```
HTTP/1.1 300 Multiple Choices
```

```
Alternates : {"/articles/a1234.en.html" 1 {type text/html} {charset utf-8}
```

```
  {language en} {length ...}},
```

```
    {"/articles/a1234.fr.pdf" 0.9 {type application/pdf} {charset utf-8}
```

```
  {language fr} {length ...}}
```

```
  ...
```

```
Vary : negotiate, accept, accept-language
```

```
TCN : list
```

```
[...]
```

Requêtes conditionnelles

RFC7232

Identifiant

- ETag
- If-Match
- If-None-Match

Date de modif.

- Last-Modified
- If-Modified-Since
- If-Unmodified-Since

temps serveur \neq temps client

Requête partielle

RFC7233

- Range
- If-Range
- 206 Partial Content
- 416 Range Not Satisfiable

- ▶ 304 Not Modified
- ▶ 412 Precondition Failed
- ▶ 428 Precondition Required

```
HEAD / HTTP/1.1
```

```
Host : www.example.com
```

```
HTTP/1.0 200 OK
```

```
Last-Modified : Fri, 30 Jul 2010 15 :30 :18 GMT
```

```
ETag : "573c1-254-48c9c87349680"
```

```
[...]
```

HEAD / HTTP/1.1

Host : www.example.com

If-Match : "573c1-254-48c9c87349680"

HTTP/1.0 200 OK

Last-Modified : Fri, 30 Jul 2010 15 :30 :18 GMT

ETag : "573c1-254-48c9c87349680"

[...]

HEAD / HTTP/1.1

Host : www.example.com

If-Match : "foobar"

HTTP/1.0 412 Precondition Failed

Last-Modified : [Fri, 30 Jul 2010 15 :30 :18 GMT](#)

ETag : "573c1-254-48c9c87349680"

[...]

```
HEAD / HTTP/1.1
```

```
Host : www.example.com
```

```
If-None-Match : "foobar"
```

```
HTTP/1.0 200 OK
```

```
Last-Modified : Fri, 30 Jul 2010 15 :30 :18 GMT
```

```
ETag : "573c1-254-48c9c87349680"
```

```
[...]
```



```
HEAD / HTTP/1.1
Host : www.example.com
If-Modified-Since : Wed, 28 Jul 2010 00 :00 :00 GMT

HTTP/1.0 200 OK
Last-Modified : Fri, 30 Jul 2010 15 :30 :18 GMT
ETag : "573c1-254-48c9c87349680"
[...]
```

HEAD / HTTP/1.1

Host : www.example.com

If-Unmodified-Since : [Wed, 28 Jul 2010 00 :00 :00 GMT](#)

HTTP/1.0 412 [Precondition Failed](#)

Last-Modified : [Fri, 30 Jul 2010 15 :30 :18 GMT](#)

ETag : "573c1-254-48c9c87349680"

[...]

```
HEAD / HTTP/1.1
Host : www.example.com
If-Modified-Since : Sat, 31 Jul 2010 00 :00 :00 GMT

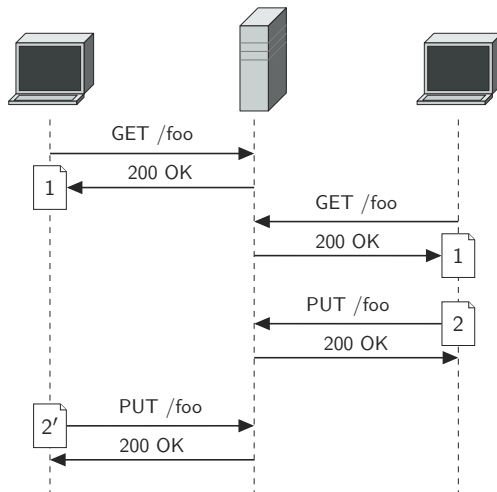
HTTP/1.0 304 Not Modified
Last-Modified : Fri, 30 Jul 2010 15 :30 :18 GMT
ETag : "573c1-254-48c9c87349680"
[...]
```

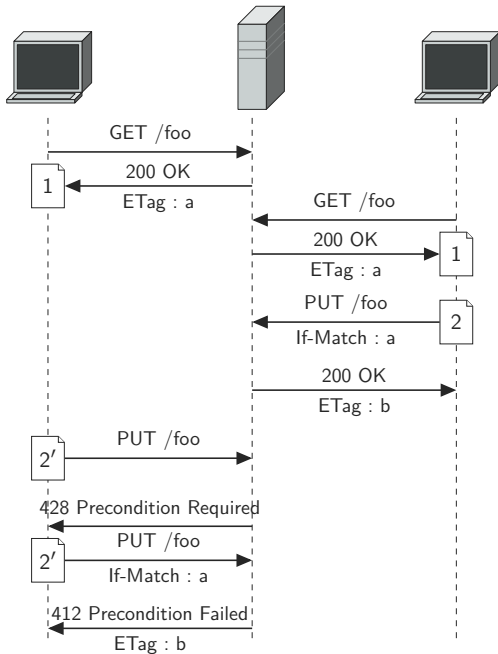
```
HEAD / HTTP/1.1
Host : www.example.com
If-Unmodified-Since : Sat, 31 Jul 2010 00 :00 :00 GMT

HTTP/1.0 200 OK
Last-Modified : Fri, 30 Jul 2010 15 :30 :18 GMT
ETag : "573c1-254-48c9c87349680"
[...]
```

Utilisation

- rafraichissement du cache
- modification concurrente





Cache HTTP

Cache-Control

Cache-Control

Quoi ?

- `public` (rep)
- `private` (rep)
- `no-cache` (req/rep)
- `no-store` (req/rep)

Cache-Control

Comment ?

- `max-age` (req/rep)
- `min-fresh` (req)
- `max-stale` (req)
- `s-maxage` (rep)
- `only-if-cached` (req) → 504 Gateway Timeout
- `must-revalidate` (rep)
- `proxy-revalidate` (rep)

Cache-Control

no-transform

- Safe
- Via
- Vary
- Age
- ETag
- Last-Modified
- Date
- Expires
- Warning

- ▶ 203 Non-Authoritative Information
- ▶ 304 Not Modified
- ▶ 504 Gateway Timeout

Principes de base

au delà des fichiers statiques

- passerelle
- calcul
- manipulation de ressources
- portail
- ...

⇒ code coté serveur

procédure

- entrée : requête
- sortie : réponse

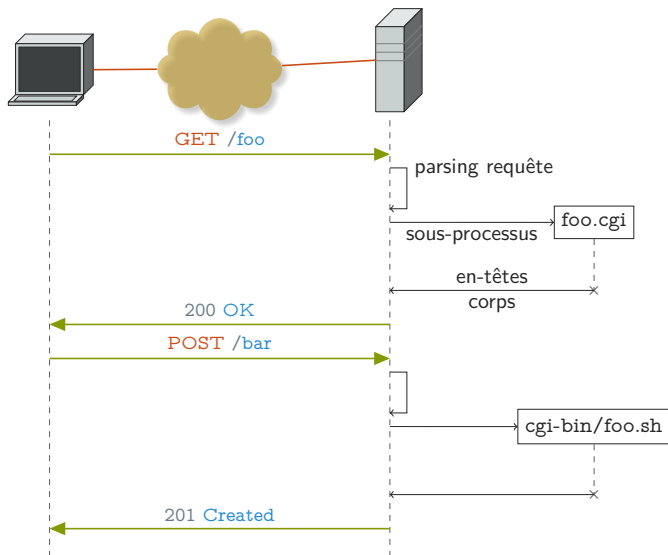
architecture n-tiers

- V1.0 1993 NCSA
- V1.1 1995 → 2004 RFC3875

Principe de fonctionnement

programme exécutable

- autonome
- \forall langage (script)



configuration serveur \Rightarrow correspondance uri \leftrightarrow programme

programme

- test méthode
- test chemin
- test en-têtes
- lecture et analyse du corps
- génération des en-têtes
- génération du corps

Communication

Variables d'environnement

- `GATEWAY_INTERFACE` : CGI/version
- `SERVER_NAME`
- `SERVER_PROTOCOL`
- `SERVER_PORT`
- `REQUEST_METHOD`
- `PATH_INFO` : chemin relatif au script CGI
- `SCRIPT_NAME` :
- `QUERY_STRING` : application/x-www-form-urlencoded brut
- `REMOTE_ADDR` : IP du client
- `AUTH_TYPE`
- `REMOTE_USER`
- `CONTENT_TYPE` : type média du contenu de la requête
- `CONTENT_LENGTH`
- `HTTP_ACCEPT` : type média demandés par le client
- `HTTP_*` : tous les en-têtes HTTP)

GET /cgi-bin/foo.sh/bar/baz?a=1&b=2 HTTP/1.1

SERVER_PROTOCOL : HTTP/1.1

REQUEST_METHOD : GET

PATH_INFO : /bar/baz

SCRIPT_NAME : /cgi-script/foo.sh

QUERY_STRING : a=1&b=2

CONTENT_LENGTH : 0

Corps → stdin

Réponse → stdout

- en-têtes + ligne vide + corps
- en-tête spécial **Status**

```
#!/bin/sh

if [ $REQUEST_METHOD != "GET" ] ; then
    echo "Content-Type : text/plain"
    echo "Status : 405 NotAllowed"
    echo "Allow : GET"
    echo ""
    echo "Method not Allowed"
    exit 0
fi

echo "Content-Type : plain/text"
echo ""
echo "Hello world!"
```

bibliothèques

- accès env.
- parsing *query string* et formulaire (`application/x-www-form-urlencoded`)
- lecture stdin
- codes erreurs
- gestion en-têtes
- échapement (html)

Avantage



- executable « normal » $\Rightarrow \forall$ techno. serveur
- autonome
- tests
- déploiement
- gestion droits (suexec)

Problèmes

A red 'X' icon, likely indicating a problem or a negative example.

- 1 req. \Rightarrow 1 sous-processus
- pas d'état partagé : connexion BD, cache interne, ...

CGI \rightarrow script \Rightarrow interprété

interpréteur → serveur

Exemple

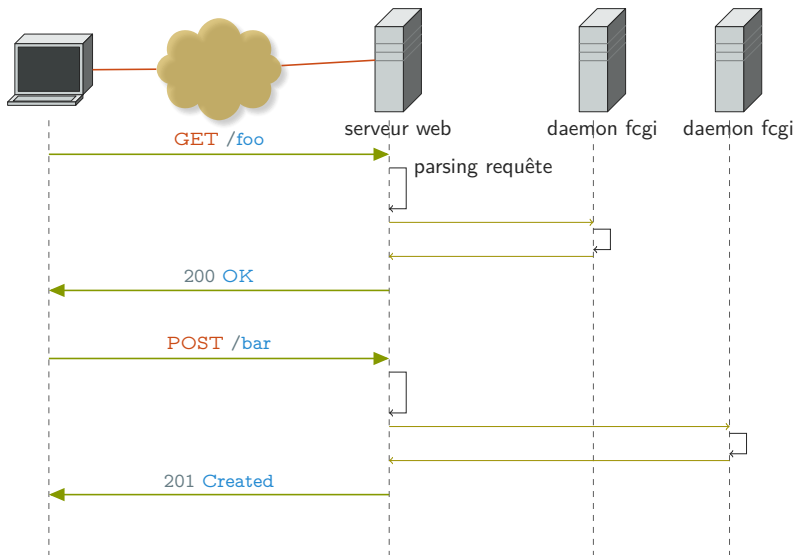
- ▶ `mod_perl`
- ▶ `mod_php`
- ▶ `mod_python`, `mod_wsgi`
- ▶ `mod_lua`
- ▶ `mod_mono`
- ▶ `mod_ruby`
- ▶ `mod_tcl`
- ▶ ...

- \equiv CGI
- pas de sous-processus
- accès direct aux variables
- fonctions util.
- interprété

- 1996 Open Market
- <http://www.fastcgi.com/devkit/doc/fcgi-spec.html>

Principe

- \equiv CGI
- daemon
- communication \rightarrow socket



Avantages

- pas de processus \Rightarrow performances
- isolation \Rightarrow sécurité
- réutilisation (cache, connexions BD)
- répartition charge
- \forall langage, serveur

- fcgiwrap
- spawn-fcgi

aussi SCGI, AJP

- conteneur
- \cong FastCGI
- \cong embarqué

langage spécifique \Rightarrow accès facilité aux paramètres

- entrée : objet requête
- sortie : objet réponse

communication HTTP

Exemple

- ▶ Java : Tomcat, GlassFish, Jetty, WebSphere, WildFly (JBoss)
- ▶ Javascript : Node.js
- ▶ .Net : IIS
- ▶ Python : Gunicorn, Paste, Tornado, uWSGI, Zope
- ▶ Ruby : Passenger, Mongrel, Unicorn

API spécifique

- Rack
- WSGI
- servlet, JSP, JAX-(R/W)S
- ...



lourd

- \cong serveur d'application
- léger

- serveur autonome
- embarqué
- reverse proxy
- répartition
- SOA

⇒ micro-services

Cookies

Définition (Cookie)

- information serveur : clé → valeur + méta
- stockée client
- renvoyée à chaque requête

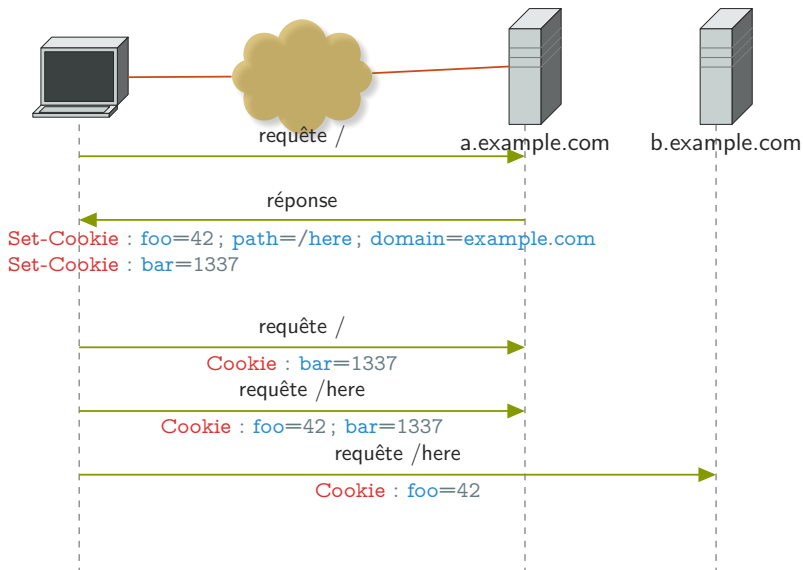
RFC 6265

en-têtes :

- Set-Cookie
- Cookie

Méta-données

- serveur : `domain`
- chemin : `path`
- durée : `expires`, `max-age`
- protocole : `secure`, `httponly`



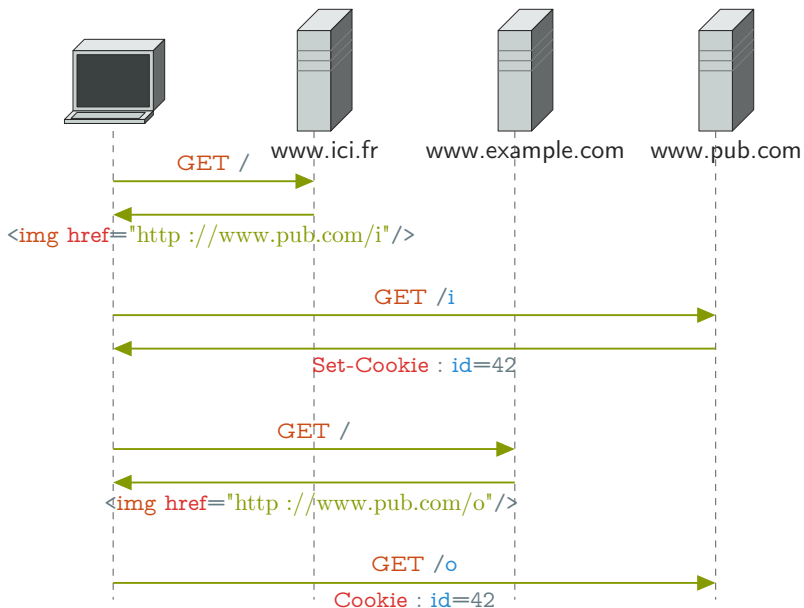
⇒ ajout d'état

- persistant entre sessions
- partagé entre serveurs

Cookies tiers

Définition (Cookie tier)

cookie dans la réponse d'un serveur tiers (pub, image)



Cookies tiers



vie privée \Rightarrow blocages, suppression

Sécurité

- autorité globale : cross-site request forgery (CSRF)
- données en clair : session hijacking, replay
- faible intégrité : \neq hosts

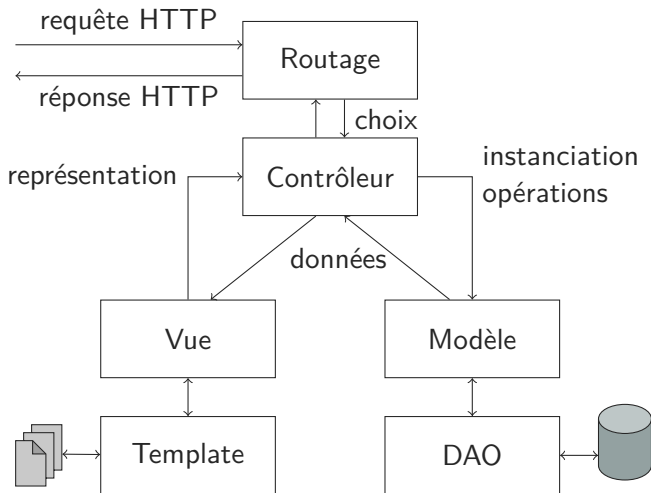
Le modèle MVC

Modèle – Vue – Contrôleur

Modèle : données et règles métier

Vue : présentation et formatage

Contrôleur : interactions et connexion M-V



- Hierarchical MVC
- Presentation – Abstraction – Control
- Model – View – Adapter
- Model – View – Presenter
- MVVM

Micro-frameworks

- Sinatra (Ruby)
- Flask (Python)
- Spark (Java)
- ...

orchestration

- url (en-têtes) → choix modèle instantiation
- verbe (en-têtes) → opération
- extraction information
- en-têtes (url) → choix vue
- génération réponse
- capture exceptions → code erreur HTTP

⇒ framework

- code
- configuration
- convention

Jax-RS (Jersey)

```
@Path("hello/{name}")
public class HelloResource {
    @GET
    @Produces(MediaType.TEXT_PLAIN)
    public String sayHello(@PathParam("name") String name) {
        return String.format("Hello %s!", name);
    }
}
```

Spark

```
public class App {  
    public static void main(String[] args) {  
        get("hello/ :name",  
            (req, res) -> String.format("Hello %s!", req.params(" :name")));  
    }  
}
```

- règles métier
- accès aux données \Rightarrow persistance

pas de dépendance :

- contrôleur
- vue

Définition (Impedance Mismatch)

différence de concepts entre modèles objet et relationnel

- encapsulation, sous-typage, polymorphisme
- types scalaires vs. références
- (multi)set vs. graphe
- interface uniforme (CRUD) vs. interface spécifique
- déclaratif vs. impératif
- transactions
- identité vs. égalité

DAO Data Access Object

⇒ encapsule l'accès aux données

DTO Data Transfert Object
structure, modèle



```
PreparedStatement stm = con.prepareStatement("select name, age from user  
    where id = ?");  
stm.setInt(1, 42);  
ResultSet rs = stmt.executeQuery();  
rs.first();  
User u = new User();  
u.setName(rs.getString(1));  
u.setAge(rs.getInt(2));
```

ORM (*Object–Relational Mapper*)

```
String name = Personne.getById(1).getName();
```

magie noire

- configuration (xml)
- annotations, redéfinition
- convention


-  many to many \Rightarrow jointures
-  synchro

Définition (Template)

- génération
- patron → forme générale
- moteur → données

Illimité

langage complet (hôte)

- traitement quelconque
-  peut modifier le modèle

Limité

langage spécifique

- valeur en lecture
- pas d'effet de bord

pull : vue (template) \rightarrow données (modèle)
 \Rightarrow à la demande

push : contrôleur (vue) \rightarrow template
 \Rightarrow a priori

- pipeline
- callback
- reverse callback

Callback

- mélange code/template → changement de contexte
- → appel de code natif
- *pull*



Exemple

server page (ASP, JSP, PHP)

```
<?php
$data = DAO.getObject($GET["id"]);
?>
<h2><?=$data->getTitle() ?></h2>
<ul>
<?foreach ($data->getElements() as $elt) { ?>
    <li><a href='<?=$elt->url ?>'><?=$elt->name ?></a></li>
<?} ?>
</ul>
```

```
<?php
$data = DAO.getObject($GET["id"]);

echo "<h2>" . $data->getTitle() . "</h2>";
echo "<ul>";
foreach ($data->getElements() as $elt) {
    echo "<li><a href='" . $elt->url . "'>";
    echo $elt->name;
    echo "</a></li>";
}
echo "</ul>";
?>
```

- ✗ mélange logique et présentation
- ≈ affichage
- ✗ \neq représentations \Rightarrow duplication
- ≈  développeur
- ≈  designer

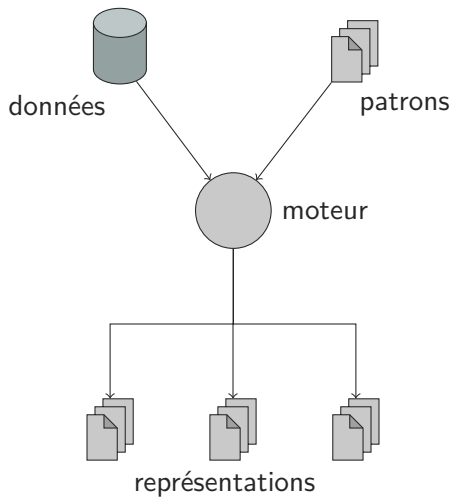
Pipeline

- patron externe
- compilé/interprété
- *push*
- \Rightarrow langage spécifique

Exemple

- FreeMarker (Java)
- Jinja2 (Python)
- Smarty (PHP)
- mustache (*)
- ...

https://en.wikipedia.org/wiki/Template_engine_%28web%29



```
<h2>{{title}}</h2>
{% if elements %}
<ul>
    {%- for name, url in elements %}
    <li><a href="{{url}}">{{name}}</a></li>
    {%- endfor %}
</ul>
{%-else%}
<p>Rien...</p>
{%endif%}
```

```
<table><caption>{{title}}</caption>
    <thead><tr><th>Name</th><th>URL</th></tr></thead>
    <tbody>
    {%- for name, url in elements %}
    <tr><td>{{name}}</td>
        <td><a href="{{url}}">{{url}}</a></td></tr>
    {%- endfor %}
    </tbody>
</table>
```

```
from jinja2 import Environment, FileSystemLoader
templates = './listings/templates/'
env = Environment(loader=FileSystemLoader(templates))
data = [
    {
        "title" : "Mon super titre",
        "elements" : [
            ("nom1", "http://url1.example.com/"),
            ("nom2", "http://url2.example.com/")
        ]
    },
    {
        "title" : "Autre titre",
        "elements" : []
    }
]

for t in ['list', 'table'] :
    with file(templates + t + 'out.html', 'w') as out :
        for d in data :
            out.write(env.get_template(t + ".html").render(d))
```

```
<h2>Mon super titre</h2>
```

```
<ul>
```

```
  <li><a href="http ://url1.example.com/">nom1</a></li>
```

```
  <li><a href="http ://url2.example.com/">nom2</a></li>
```

```
</ul>
```

```
<h2>Autre titre</h2>
```

```
<p>Rien...</p>
```

```
<table><caption>Mon super titre</caption>
```

```
  <thead><tr><th>Name</th><th>URL</th></tr></thead>
```

```
  <tbody>
```

```
    <tr><td>nom1</td>
```

```
      <td><a
```

```
        href="http ://url1.example.com/">http ://url1.example.com/</a></td></tr>
```

```
    <tr><td>nom2</td>
```

```
      <td><a
```

```
        href="http ://url2.example.com/">http ://url2.example.com/</a></td></tr>
```

```
  </tbody>
```

```
</table>
```

```
<table><caption>Autre titre</caption>
```

```
  <thead><tr><th>Name</th><th>URL</th></tr></thead>
```

```
  <thead>
```

- ✓ sépare logique et présentation
- ✓ affichage
- ≈ duplication de la logique → présentation
- ✓ 🙌 développeur
- ✓ 🙌 designer