

# Introduction

Orientation Réseaux et Services, Module EIP, Unité PDA

Olivier Liechti

Haute Ecole d'Ingénierie et de Gestion du Canton de Vaud (HEIG-Vd)

Yverdon-les-Bains

# Contenu du cours

## > Première partie

- Protocoles applicatifs: HTTP et LDAP
- Introduction aux Web Services: SOAP et REST
- Cette partie est enseignée par Olivier Liechti

## > Deuxième partie

- Voix sur IP
- Cette partie est enseignée par Juergen Ehrensberger

# Planning

- > **Semaine 1 (21/09)**
  - HTTP: protocole (1)
- > **Semaine 2 (28/09)**
  - HTTP: protocole (2)
- > **Semaine 3 (5/10)**
  - HTTP: infrastructure (1)
- > **Semaine 4 (12/10)**
  - **Travail écrit (HTTP)**
- > **Semaine 5 (19/10)**
  - LDAP: protocole + infrastructure
- > **Semaine 6 (2/11)**
  - Web Services (SOAP)
- > **Semaine 7 (9/11)**
  - Web Services (REST)
- > **Semaine 8 (16/11)**
  - **Travail écrit (LDAP + WS)**

- > **Semaine 2**
  - Laboratoire HTTP 1

**délai** : jeudi 4 octobre, 8h00

- > **Semaine 4**
  - Laboratoire HTTP 2

**délai** : jeudi 18 octobre, 8h00

- > **Semaine 6**
  - Laboratoire LDAP

**délai** : jeudi 8 novembre, 8h00

- > **Semaine 8**
  - Laboratoire Web Services

# Site du cours

## > Utilisation de Subversion et de Trac

- Subversion (SVN) est un outil de gestion de fichiers sources (gestion des versions)
- Trac est un outil de gestion de projet (y.c. un front-end web pour accéder à SVN)
- Les transparents sont dans le SVN
- Les exemples de code sont dans le SVN
- Les données des labos sont dans le wiki

## > Accès à l'infrastructure

- <http://193.134.218.14/trac/CoursOLI/wiki/HeigVdPda>
- configuration svn dans NetBeans (pas d'authentification requise)

# HTTP - le protocole

Orientation Réseaux et Services, Module EIP, Unité PDA

Olivier Liechti

Haute Ecole d'Ingénierie et de Gestion du Canton de Vaud (HEIG-Vd)

Yverdon-les-Bains

# AGENDA

## > Introduction

- Historique
- Vue d'ensemble et notions de base

## > Spécifications

- Spécifications: RFC 1945 (HTTP/1.0) & RFC 2616 (HTTP/1.1)
- Notions de "ressource" et de "représentation de ressource"

## > Aspects particuliers

- Gestion des connections
- Gestion de l'état (state management)

## > Préparation du laboratoire

- Implémenter HTTP en Java
- Objectifs du laboratoire

# *Historique et notions de base*

# Historique

- > **Vannevar Bush, "As We May Think" (1945)**
  - Memex (microfilms)
- > **Douglas Engelbart (60's)**
  - oNLine System (NLS)
- > **Ted Nelson (1965)**
  - "Hyperext", Xanadu
- > **Tim Bernes Lee (1980 - 1990)**
  - Internet + Hypertext = World Wide Web
- > **Marc Andreessen (1993)**
  - Mosaic Web Browser, Netscape
- > **James Gosling (90's - 1995)**
  - Java
- > **World Wide Web Consortium (W3C): 1994**

Web for  
Information Sharing

Web for  
eCommerce

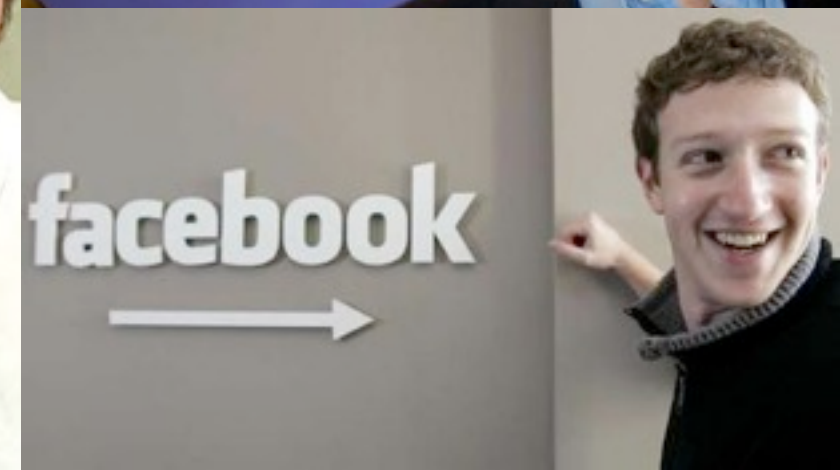
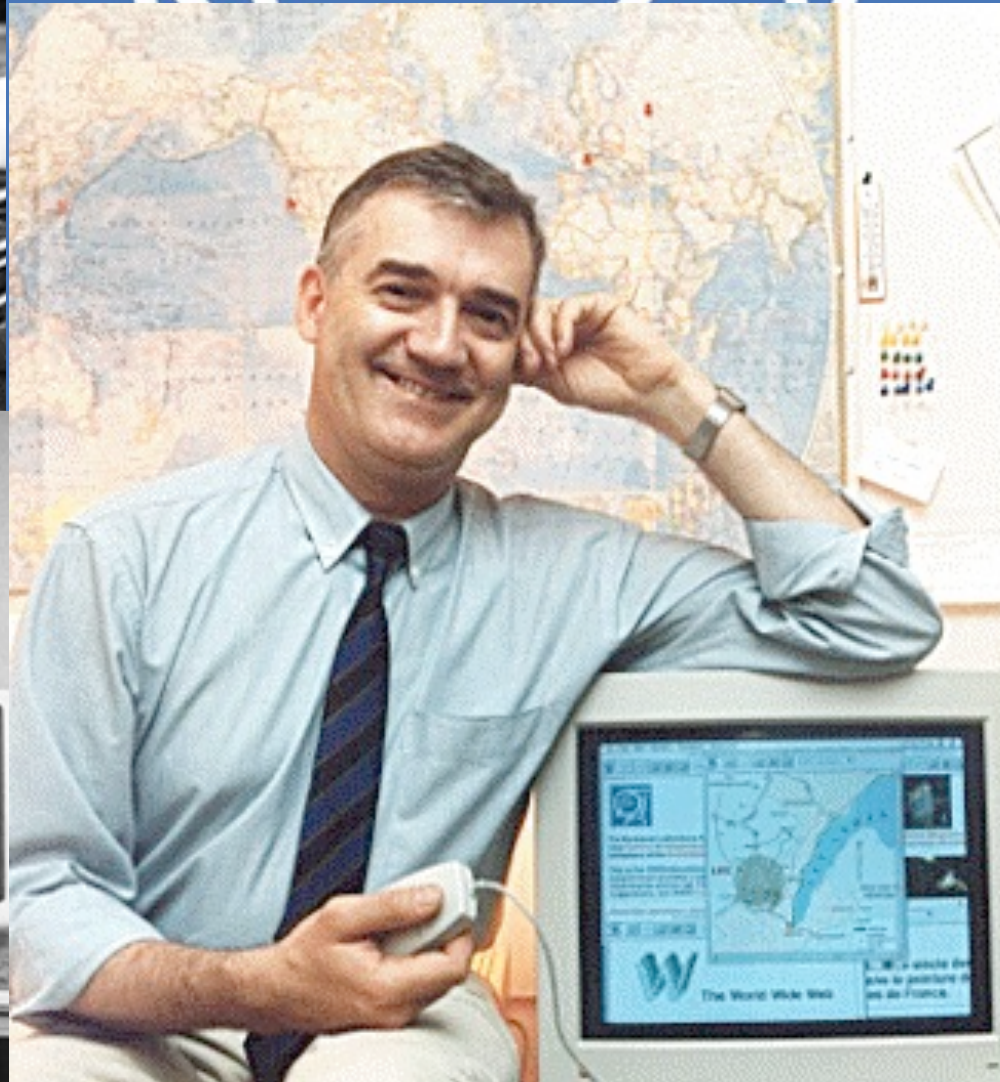
Web 2.0

Mobile & Ubiquitous  
Web

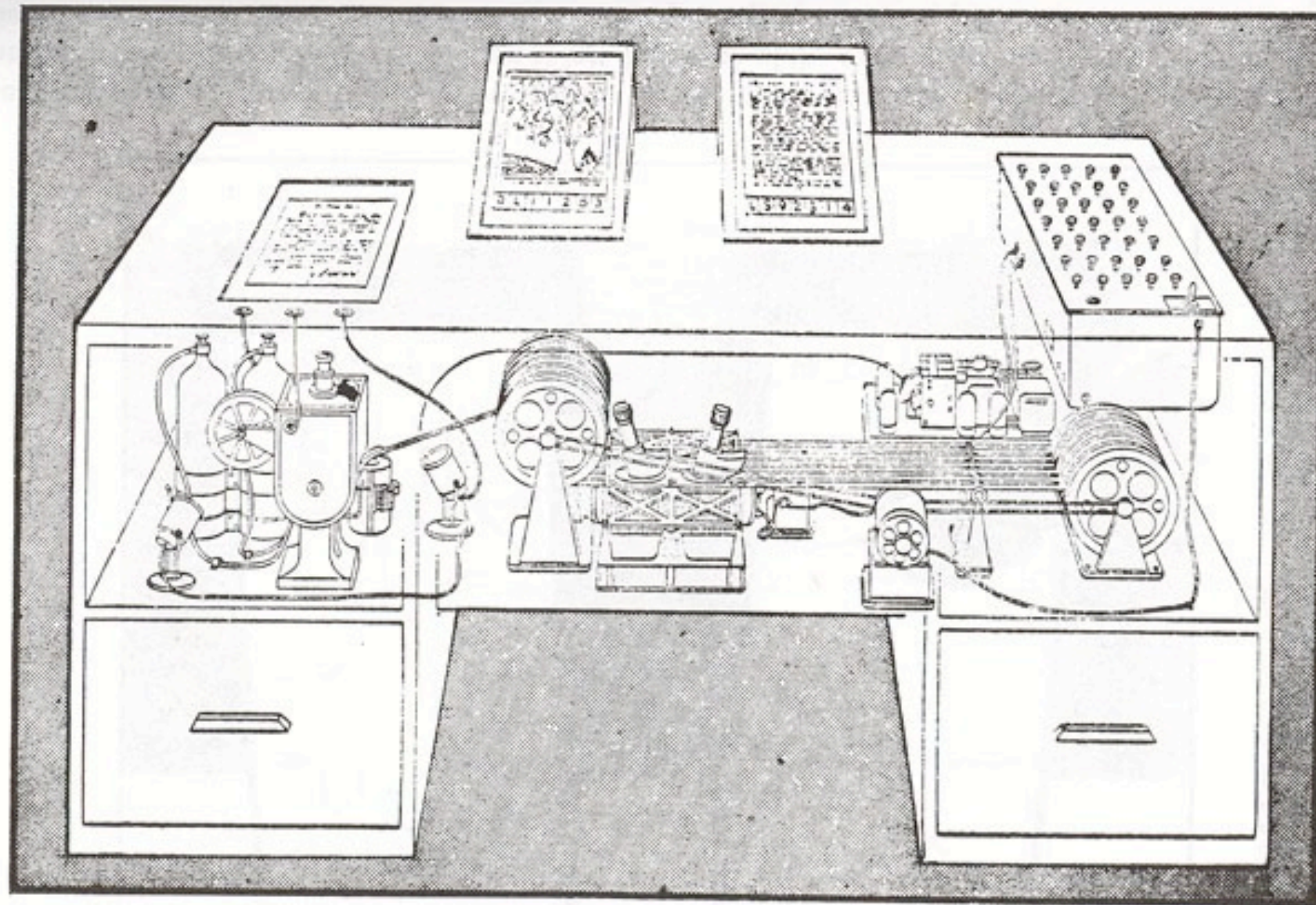
*<http://www.w3.org/History.html>*

*<http://www.w3.org/2004/Talks/w3c10-HowItAllStarted/>*









Memex in the form of a desk would instantly bring files and material on any subject to the operator's fingertips. Slanting translucent viewing screens magnify supermicrofilm filed by code numbers. At left is a mechanism which automatically photographs longhand notes, pictures and letters, then files them in the desk for future reference (*LIFE* 19(11), p. 123).

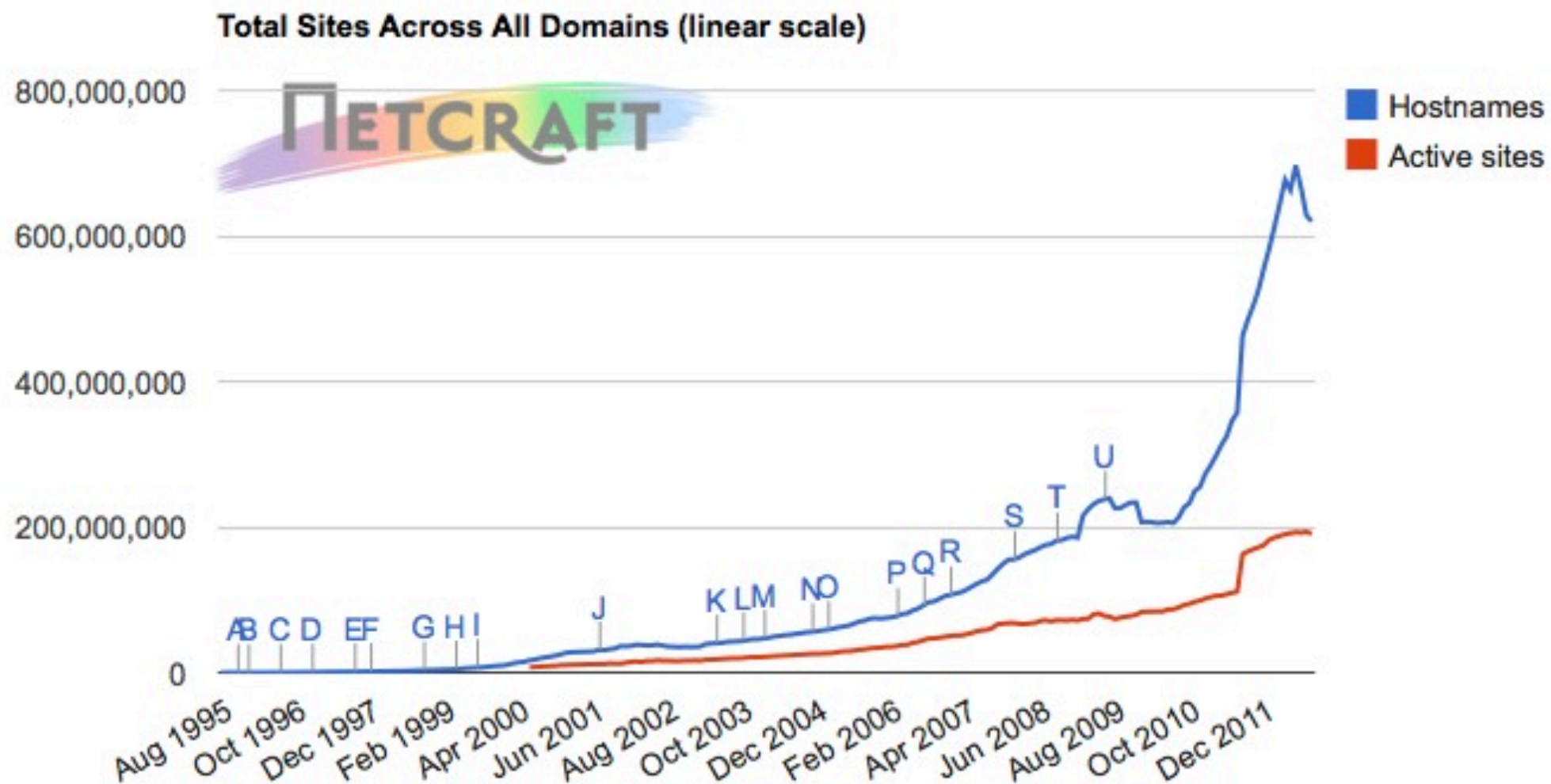




[http://en.wikipedia.org/wiki/Image:First\\_Web\\_Server.jpg](http://en.wikipedia.org/wiki/Image:First_Web_Server.jpg)



# L'explosion du nombre de sites web



<http://news.netcraft.com/archives/category/web-server-survey/>

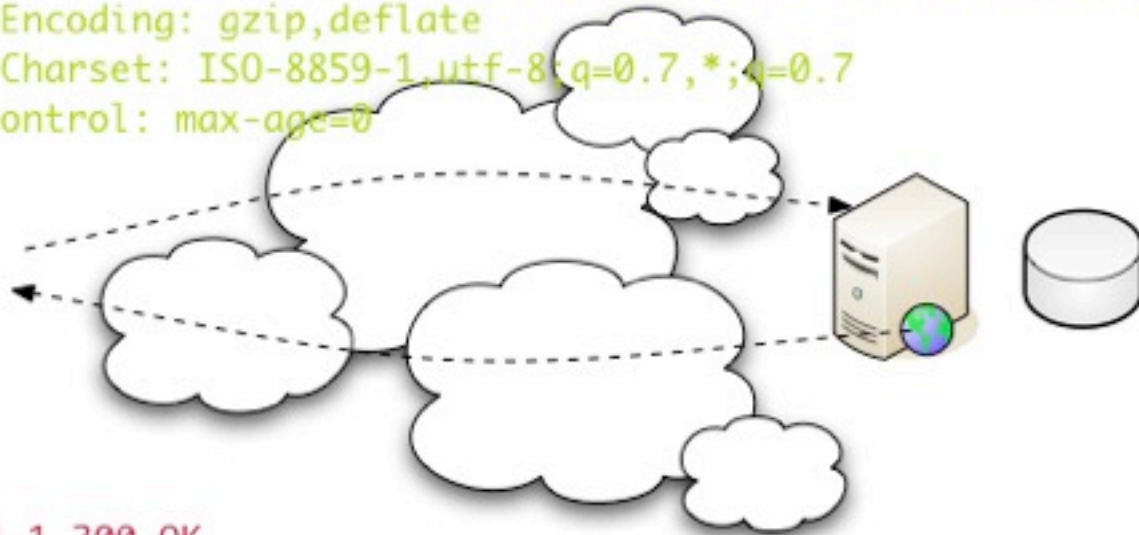
# Qu'est-ce qu'un système hypertexte?

- > *"Un système hypertexte est un système contenant des documents liés entre eux par des hyperliens permettant de passer automatiquement (en pratique grâce à l'informatique) du document consulté à un autre document lié.*
- > *Un document hypertexte est donc un document qui contient des hyperliens.*
- > *Lorsque les documents ne sont pas uniquement textuels, mais aussi audiovisuels, on peut parler de système et de documents hypermédias."*

<http://fr.wikipedia.org/wiki/Hypertexte>

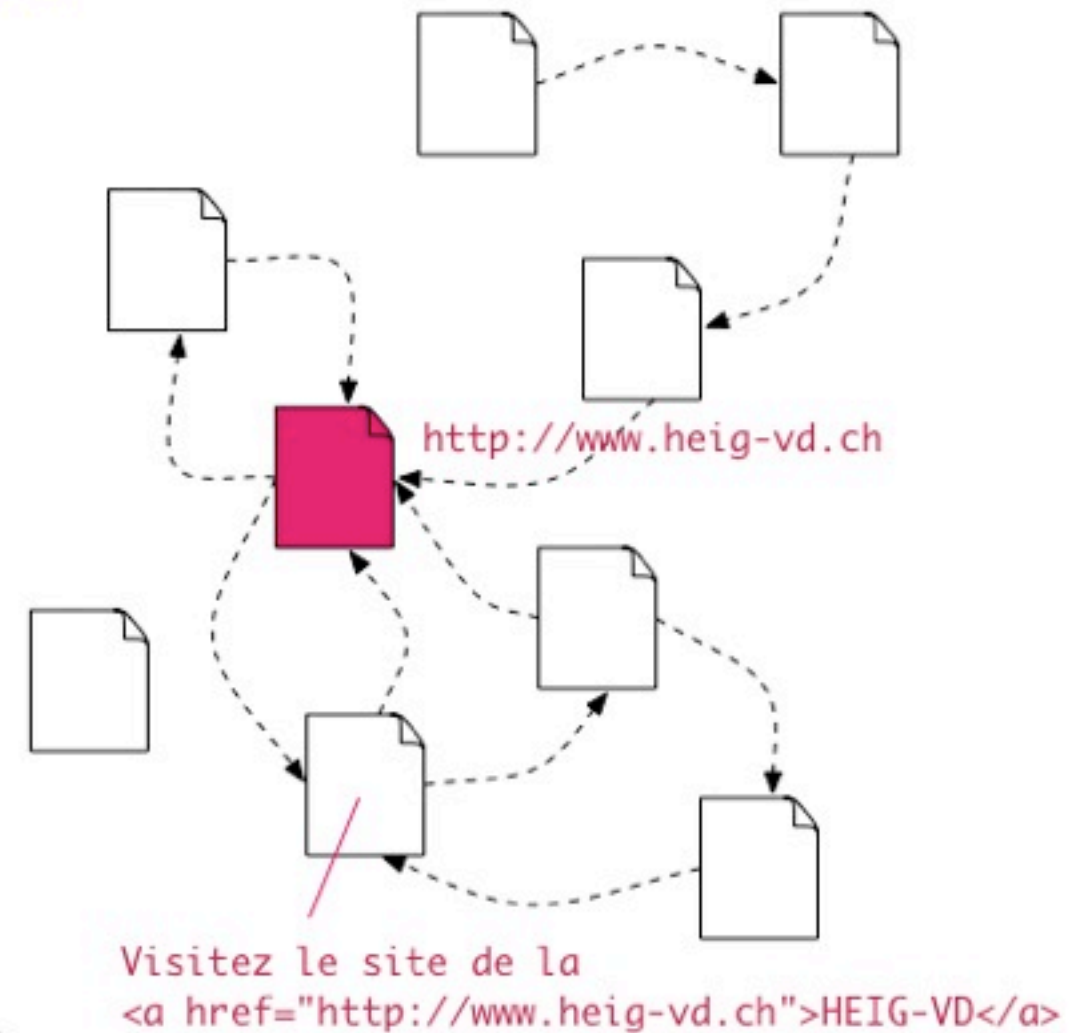
# Le World Wide Web

```
GET http://www.heig-vd.ch/ HTTP/1.1
User-Agent: Mozilla/5.0 (Macintosh; U; Intel Mac OS X 10.4; en-US; rv:1.9.0.1)
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Encoding: gzip,deflate
Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.7
Cache-Control: max-age=0
```



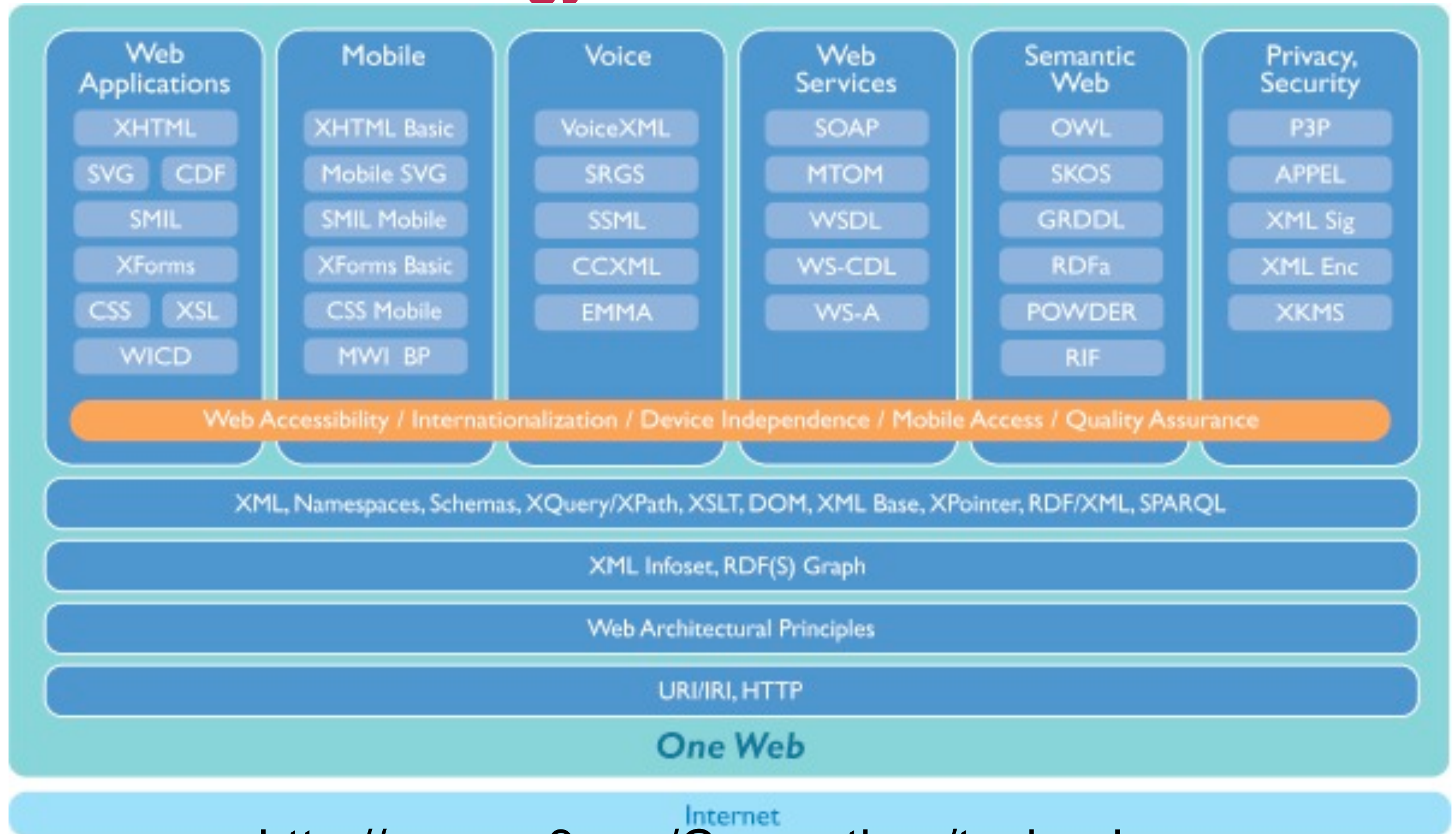
```
HTTP/1.1 200 OK
Content-Length: 2859
Content-Type: text/html
Content-Location: http://193.134.217.17/index.html
Last-Modified: Wed, 30 Jan 2008 14:10:51 GMT
Accept-Ranges: bytes
Connection: close
```

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="fr" lang="fr">
<head>
<title>Accueil-LPTherm</title>
...
```





# The W3C Technology Stack



<http://www.w3.org/Consortium/technology>

# *Les spécifications*

*RFC 2616 (1.1), RFC 1945 (1.0)*

<http://www.w3.org/Protocols/>



## RFC 2616: Abstract

*The **Hypertext Transfer Protocol (HTTP)** is an application-level protocol for **distributed**, collaborative, hypermedia information systems.*

*It is a **generic, stateless**, protocol which can be used for many tasks beyond its use for hypertext, such as name servers and distributed object management systems, through **extension** of its **request methods**, **error codes** and **headers** [47].*

*A feature of HTTP is the **typing** and **negotiation** of **data representation**, allowing systems to be built **independently** of the data being transferred.*

# Contenu de la spécification

## > Introduction et définitions

- Sections 1 à 3 (pp 1 à 30)

## > Format des messages

- Sections 4 à 7 (pp 31 à 43)

## > Gestion des connections

- Section 8 (pp 44 à 50)

## > Définition des méthodes

- Section 9 (pp 51 à 57)

## > Définition des codes de statut

- Section 10 (pp 57 à 71)

## > Authentification

- Section 11 (p 71) (RFC 2617)

## > Négotiation

- Section 12 (pp 71 à 74)

## > Caching

- Section 13 (pp 74 à 99)

## > Définition des entêtes

- Section 14 (pp 100 à 150)

## > Sécurité

- Section 15 (pp 150 à 156)

## > Conclusion et annexes

- Sections 16 à 19 (pp 156 à 176)

*HTTP est utilisé pour  
interagir avec des  
"ressources"*

# Qu'est-ce qu'une "ressource"?

- > **A première vue, on pourrait penser qu'une "ressource" est un "fichier" stocké sur un serveur web:**
  - un document HTML, un document XML, une image PNG, etc.
- > **Cette définition est appropriée pour le web "statique":**
  - le client envoie une requête HTTP au serveur;
  - la requête spécifie une URL qui identifie un document (e.g. /dir1/menu.html)
  - le serveur renvoie le contenu du document au client.
- > **Mais le web est bien plus qu'une simple "bibliothèque" de documents:**
  - au travers du web, nous interagissons avec des **services**; le contenu des réponses envoyées au client est alors généré **dynamiquement**;
  - au travers du web, nous interagissons avec le monde "**physique**".
- > **Nous avons donc besoin d'une définition plus large pour le terme de "ressource"...**

# Qu'est-ce qu'une "ressource"?

- > **Une ressource est "quelque chose" que l'on peut nommer et identifier de manière unique:**
  - Exemple 1: un article particulier publié dans le journal "24 heures"
  - Exemple 2: la collection des articles publiés dans une rubrique du journal
  - Exemple 3: le CV d'une personne
  - Exemple 4: le cours actuel de l'action Nestlé
  - Exemple 5: le distributeur de boissons dans le hall de l'école
  - Exemple 6: la liste des notes obtenues par l'étudiant Jean Dupont
- > **Une URL (Uniform Resource Locator) est une chaîne de caractères qui permet d'identifier et de localiser une ressource:**
  - Exemple 1: <http://www.24heures.ch/vaud/vaud/2008/08/04/trente-etudiants-partent-rencontre-patrons>
  - Exemple 2: <http://www.24heures.ch/articles/vaud/actu>
  - Exemple 5: <http://www.smart-machines.ch/customers/heig/machines/8272>

# Ressource vs. représentation

- > Une "ressource" peut donc être quelque chose d'abstrait (de l'information, un concept, un état) ou quelque chose qui a une existence tangible dans le monde physique (un objet, une personne).
- > Le protocole HTTP permet le transfert d'information entre un client et un serveur.
- > L'information qui est transférée lors d'un échange HTTP n'est pas une ressource. C'est la représentation d'une ressource!
- > La représentation d'une ressource est un ensemble de données (bytes) accompagné d'un ensemble de méta-données décrivant les données:
  - Un document HTML, une image, un son sont des exemples de représentations de ressources.
  - Exemples de méta-données: définition du type, longueur, encodage, etc.
- > Une ressource peut avoir plusieurs représentation:
  - Exemple 1: la ressource identifiée par l'URL "http://www.abc.com/monCV" peut avoir une représentation HTML, une représentation PDF, etc.

# Comment peut-on interagir avec une ressource?

- > HTTP définit un ensemble de "méthodes"
- > Ces méthodes sont des "verbes" qu'on peut appliquer à une resource:
  - **GET**: retrieve whatever information is identified by the Request-URI
  - **POST**: used to request that the origin server accept the entity enclosed in the request as a new subordinate of the ressource identified by the Request-URI in the Request-Line
  - **PUT**: requests that the enclosed entity be stored under the supplied Request-URI.
  - **DELETE**: requests that the origin server delete the ressource identified by the Request-URI.
  - **HEAD**: identical to GET except that the server MUST NOT return a message-body in the response
  - **TRACE**: used for debugging (echo)
  - **CONNECT**: reserved for tunneling purposes

# Comment peut-on interagir avec une ressource?

- > **Certains protocoles ont proposé des extensions et défini d'autres méthodes. C'est par exemple le cas de WebDAV:**
  - RFC 4918
  - DAV = Distributed Authoring and Versioning
  - Le protocole permet entre autres la gestion de propriétés associées à des ressources, la gestion de collections de ressources, l'utilisation de verrous (locking).
- > **Ce protocole est notamment utilisé:**
  - Par des systèmes de gestion de contenu, de gestion de documents
  - Par des outils de gestion de code (e.g. Subversion / SVN)
  - Pour communiquer avec des services de messagerie électronique
- > **Nouvelles méthodes définies par WebDAV:**
  - PROPFIND, PROPPATCH, MKCOL, COPY, MOVE, LOCK, UNLOCK
- > **Attention: certains proxy ne laissent pas passer ces méthodes!**



## URI & URL (Paragraph 3.2)

- > **As far as HTTP is concerned, Uniform Resource Identifiers are simply formatted strings which identify--via name, location, or any other characteristic--a resource.**

`http_URL = "http:" "://" host [ ":" port ] [ abs_path [ "?" query ]]`

`http://www.google.com/`

`http://192.168.0.2/`

`http://development.intranet.company.com:8080/`

`http://www.site.com/level1/level2/level3/resource.html`

`http://www.application.com/callFunction?param1=hello&param2=world`

*HTTP est un protocole  
client-serveur*

# Principes de base du protocole

- > **Le serveur accepte des demandes de connexion TCP:**
  - le port standard est 80
  - les ports suivants sont souvent utilisés sur des serveurs de test: 81, 8080
- > **Le client fait une demande de connexion.**
- > **Quand elle est établie, il envoie une requête HTTP:**
  - une première ligne, qui identifie la ressource cible
  - plusieurs entêtes
  - une ligne vide
  - éventuellement des données (lors de l'envoi de formulaires, par exemple)
- > **Le serveur traite la requête et envoie la réponse HTTP:**
  - une première ligne, avec le code de statut
  - plusieurs entêtes
  - une ligne vide
  - les données qui décrivent la ressource cible

# Format des requêtes

```

Full-Request  = Request-Line           ; Section 5.1
                *( General-Header      ; Section 4.3
                  | Request-Header     ; Section 5.2
                  | Entity-Header )    ; Section 7.1
                CRLF
                [ Entity-Body ]        ; Section 7.2

```

```

Request-Line = Method SP Request-URI SP HTTP-Version CRLF

```

```

Request-Header = Authorization          ; Section 10.2
                | From                  ; Section 10.8
                | If-Modified-Since     ; Section 10.9
                | Referer               ; Section 10.13
                | User-Agent            ; Section 10.15

```

```

Entity-Header = Allow                  ; Section 10.1
                | Content-Encoding      ; Section 10.3
                | Content-Length        ; Section 10.4
                | Content-Type          ; Section 10.5
                | Expires                ; Section 10.7
                | Last-Modified         ; Section 10.10
                | extension-header

```

# Format des réponses

```

Full-Response  = Status-Line           ; Section 6.1
                  *( General-Header     ; Section 4.3
                    | Response-Header   ; Section 6.2
                    | Entity-Header )   ; Section 7.1
                  CRLF
                  [ Entity-Body ]       ; Section 7.2

```

```

Status-Line = HTTP-Version SP Status-Code SP Reason-Phrase CRLF

```

```

Response-Header = Location              ; Section 10.11
                  | Server               ; Section 10.14
                  | WWW-Authenticate    ; Section 10.16

```

```

Entity-Header  = Allow                  ; Section 10.1
                  | Content-Encoding    ; Section 10.3
                  | Content-Length      ; Section 10.4
                  | Content-Type        ; Section 10.5
                  | Expires             ; Section 10.7
                  | Last-Modified       ; Section 10.10
                  | extension-header

```

# Codes de statut

The first digit of the Status-Code defines the class of response. The last two digits do not have any categorization role. There are 5 values for the first digit:

- o 1xx: Informational - Not used, but reserved for future use
- o 2xx: Success - The action was successfully received, understood, and accepted.
- o 3xx: Redirection - Further action must be taken in order to complete the request
- o 4xx: Client Error - The request contains bad syntax or cannot be fulfilled
- o 5xx: Server Error - The server failed to fulfill an apparently valid request

# Codes de statut

```
Status-Code      = "200"      ; OK
                   | "201"      ; Created
                   | "202"      ; Accepted
                   | "204"      ; No Content
                   | "301"      ; Moved Permanently
                   | "302"      ; Moved Temporarily
                   | "304"      ; Not Modified
                   | "400"      ; Bad Request
                   | "401"      ; Unauthorized
                   | "403"      ; Forbidden
                   | "404"      ; Not Found
                   | "500"      ; Internal Server Error
                   | "501"      ; Not Implemented
                   | "502"      ; Bad Gateway
                   | "503"      ; Service Unavailable
                   | extension-code
```

extension-code = 3DIGIT

Reason-Phrase = \*<TEXT, excluding CR, LF>

# Quelques entêtes intéressantes

- > Host (requête)
  - Spécifie le nom du système hôte (www.heig-vd.ch) et le numéro de port cible
  - Obligatoire depuis HTTP 1.1, pour permettre les "virtual hosts" partageant la même adresse IP
- > Content-Location (réponse)
  - Indique l'emplacement de la ressource au moment où la réponse a été générée. Cet emplacement peut être différent du chemin indiqué dans l'URL.
  - Important pour les liens relatifs!
- > Location (réponse)
  - Utilisé lors des redirections, indique l'emplacement où atteindre la ressource
- > User-Agent (requête)
  - Identifie le type de navigateur envoyant la requête.
- > Server (réponse)
  - Identifie le type de serveur envoyant la réponse.
- > <http://www.cs.tut.fi/~jkorpela/http.html>



# Comment tester ou analyser un échange HTTP?

- > **Utiliser un navigateur standard + un analyseur comme wireshark:**
  - Analyse des messages HTTP (requêtes et réponses)
  - Analyse des sessions TCP
- > **Utiliser un client telnet au lieu d'un navigateur**

```
olivier-liechtis-computer:~ oliechti$ telnet www.apple.ch 80
Trying 17.149.160.31...
Connected to euro-red.apple.com.
Escape character is '^]'.
GET / HTTP/1.0
```

```
HTTP/1.0 301 Found
Server: Apache/1.3.26 (Darwin)
Date: Tue Jun  1 12:48:03 PDT 1999 PDT
Location: http://www.apple.com/
Content-type: text/html
Content-length: 265
```

```
<HTML><HEAD><TITLE>Found</TITLE></HEAD><BODY>
<H1>Found</H1> This document has moved to a new <a href="http://www.apple.com/"> location</a>.
Please update your documents and hotlists accordingly. http://www.apple.com/
<!--httphost URI / qs env -->
</BODY></HTML>Connection closed by foreign host.
```

# Comment tester ou analyser un échange HTTP?

- > **Attention: si vous êtes derrière un proxy http:**
  - transparent: vous pouvez faire un telnet vers le serveur cible
  - non-transparent: vous devez faire un telnet vers le serveur proxy!
- > **Utiliser Firebug avec Firefox**

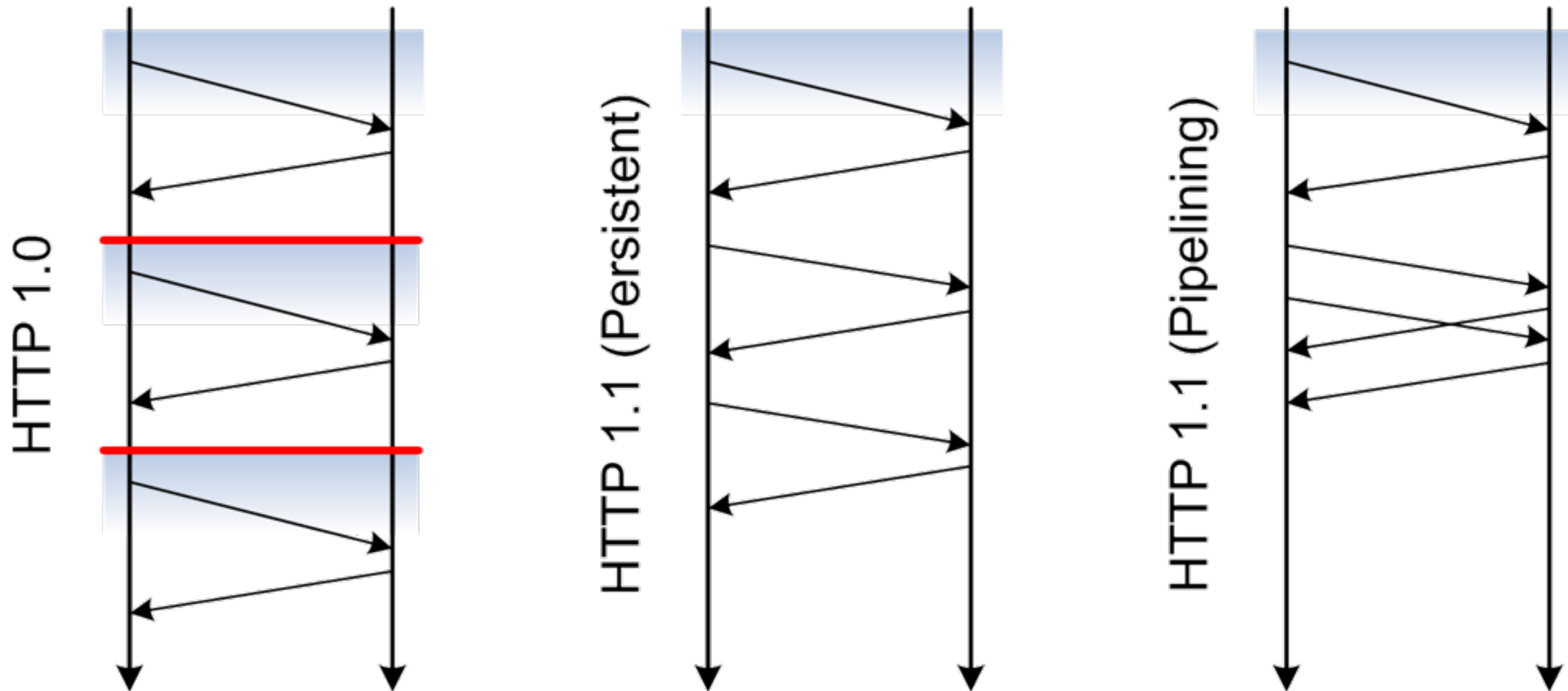


# *HTTP 1.1 optimise les connections TCP*

# Différences entre HTTP 1.0 et HTTP 1.1

- > **Avec HTTP 1.0, une session TCP est établie pour chaque échange:**
  - Or, la quantité d'information échangée est souvent faible (e.g. page HTML)
  - Les segments TCP utilisés pour l'ouverture et la clôture de la session représentent donc un "overhead" conséquent.
  - D'autre part, l'algorithme "slow-start" mis en oeuvre par TCP pour le contrôle de congestion fait qu'une taille de fenêtre sous-dimensionnée est utilisée pour la plupart des échanges HTTP.
  - Voir: <http://www.w3.org/Protocols/HTTP/1.0/HTTPPerformance.html>
- > **La plupart des pages HTML font référence à des ressources qui sont présentées à l'utilisateur en même temps que la page:**
  - `<IMG SRC="/images/logo.png">`
  - `<LINK rel="stylesheet" type="text/css" href="mystyle.css">`
  - `<script src="slidy.js" type="text/javascript"></script>`
- > **Il est donc possible d'optimiser le protocole en échangeant ces données dans la même session TCP.**

# Différences entre HTTP 1.0 et HTTP 1.1



[http://dret.net/lectures/web-fall07/foundations#\(20\)](http://dret.net/lectures/web-fall07/foundations#(20))  
<http://www.apacheweek.com/features/http11>

# Différence entre HTTP 1.0 et HTTP 1.1

- > Une des conséquences de cette modification est la nécessité, pour le serveur, d'identifier la
- > Dans HTTP 1.0, le serveur pouvait se contenter:
  - d'envoyer 1 ligne avec le code de statut
  - d'envoyer n lignes d'entête
  - d'envoyer 1 ligne vide
  - d'envoyer n bytes de données
  - de fermer la connexion TCP
- > Dans HTTP 1.1, le serveur doit:
  - envoyer 1 ligne avec le code de statut
  - envoyer n lignes d'entête, dont 1 ligne "Content-Length" qui indique la taille de l'entité en bytes
  - d'envoyer n bytes de données
  - traiter la requête suivante...

# Le cas du contenu généré dynamiquement

- > **Pour le contenu statique, il est facile pour le serveur de déterminer la taille du contenu avant d'envoyer la réponse:**
  - La valeur de l'entête "Content-Length" est donc facile à déterminer.
- > **Quand le contenu est généré dynamiquement, il n'est pas possible de connaître la taille *a priori*.**
  - Une première possibilité serait 1) de générer la réponse en mémoire, 2) de calculer la taille et 3) d'envoyer les headers et le contenu. Cette approche est acceptable quand la taille du contenu est petite.
  - Une second possibilité est d'utiliser le "chunked encoding". Dans ce cas, le serveur envoie le contenu en plusieurs parties (chunks). Avant d'envoyer chaque "chunk", il indique la taille de celui-ci.
- > **Exemple: envoyez une requête de recherche à Google (comparez le comportement HTTP/1.0 et HTTP/1.1)**

# Performances HTTP 1.1

- > <http://www.isi.edu/lam/publications/http-perf/>
- > <http://www.w3.org/Protocols/HTTP/Performance/>



*HTTP permet au client et au serveur de négocier les modalités de l'échange*


# Négociation

- > **Un objectif de la spécification HTTP était de réduire le couplage entre clients et serveurs.**
- > **Il était notamment souhaitable de permettre le développement de logiciels clients et serveurs par des organisations différentes, tout en garantissant une interopérabilité.**
- > **Même dans le cas où les mêmes logiciels sont utilisés (p.ex. Firefox et apache), leur configuration peut être différente (en fonction des choix de l'utilisateur, de l'administrateur).**
- > **HTTP permet au client et au serveur de "négocier" les modalités de leur conversation:**
  - Quels types de données sont-ils compris par le client?
  - Quels formats d'encodage de caractères sont-ils supportés?
  - Est-il possible de comprimer les données pendant l'échange?
- > **Des entêtes spéciales sont prévues à cet effet.**


# Quelques entêtes utilisées pour la négociation

- > **Déclaration de capacités (que suis-je capable de traiter? avec quelle préférence?)**
  - Accept
  - Accept-Charset
  - Accept-Encoding
  - Accept-Language
- > **Description du contenu (comment sont formatées les données dans ce message?)**
  - Content-Type
  - Content-Encoding
  - Content-Language
  - Content-Length

# Example



```
Host: www.apple.com
User-Agent: Mozilla/5.0 (Macintosh; U; Intel Mac OS X 10.4; en-US; rv:1.9.0.1) Gecko/2008070206
Firefox/3.0.1
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-us,en;q=0.5
Accept-Encoding: gzip,deflate
Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.7
Keep-Alive: 300
Connection: keep-alive
```



```
Age: 454
X-Cache-TTL: 146
Accept-Ranges: bytes
Date: Sun, 14 Sep 2008 15:06:23 GMT
Content-Type: text/html; charset=utf-8
Content-Encoding: gzip
Expires: Sun, 14 Sep 2008 15:16:23 GMT
Cache-Control: max-age=600
Server: Apache/2.2.8 (Unix)
```

```
X-Cached-Time: Sun, 14 Sep 2008 15:06:23 GMT
```

*HTTP est un protocole sans état (stateless)*

# Notion d'état dans un protocole

## > Définition

- Un protocole est sans état (stateless) si chaque échange de message entre le client et le serveur est indépendant du précédent.
- En d'autres termes, le serveur ne doit pas gérer une notion de “session” ou de “conversation”

## > Avantages

- Simplicité
- "Scalabilité" (capacité à monter en charge)
- Caching

## > Inconvénients

- Si l'application qui utilise le protocole a besoin d'une notion de “session”, alors le développeur doit mettre en oeuvre un mécanisme lui-même.
- Chaque requête doit inclure des informations supplémentaires, qui sont interprétées par le serveur pour associer la requête à une session applicative.

# A-t-on besoin d'un état sur le Web?

> **Dans beaucoup de cas, non!**

- <http://www.heig-vd.ch/Default.aspx?tabid=54>
- <http://www.letemps.ch/suisse.asp>

> **Oui, chaque fois que l'utilisateur établit une session (login/logout)**

- Magasin en ligne avec "caddie électronique"
- Application bancaire

# Gérer un état de session avec HTTP

## > Première solution:

- Transférer l'état avec chaque requête
- Client: "Bonjour, je voudrais accéder au formulaire A.HTML"
- Server: "Hum... j'ai besoin d'un mot de passe"
- Client: "Bonjour, je voudrais accéder au formulaire A.HTML; je suis Olivier et mon mot de passe est Sjks83@3"
- Server: "Voici le formulaire A.HTML"
- Client: "Voici les valeurs pour le formulaire A.HTML; je voudrais accéder au formulaire B.HTML; je suis Olivier et mon de passe est Sjks83@3"
- Server: "Voici le formulaire B.HTML"
- Client: "Voici les valeurs que j'avais donné pour le formulaire A.HTML; voicil les valeurs pour le formulaire B.HTML; je voudrais accéder au formulaire C.HTML; je suis Olivier et mon mot de passe est Sjks83@3"

## > Technique d'implémentation

- Champs cachés dans formulaires HTML (<INPUT TYPE="HIDDEN">)



# Gérer un état de session avec HTTP

## > Deuxième solution:

- Gérer une session du côté serveur; chaque session a un identifiant unique; l'état est stocké soit en mémoire, soit dans une base de données.
- Avec chaque requête, le client indique l'identifiant de session; le serveur peut alors récupérer l'état.

## > Technique d'implémentation

- Utiliser des "cookies" (Mécanisme développé par Netscape, standardisé ensuite dans le RFC 2965)
- Le serveur envoie un header "Set-Cookie" ou "Set-Cookie2", avec une petite quantité de données (typiquement l'identifiant de session)
- Le client renvoie ces données avec un header "Cookie" dans les requêtes ultérieures.
- Contraintes: un client ne renvoie les cookies que vers le serveur qui les a envoyés; un chemin d'accès peut également être spécifié.

Shop All Departments

Search Amazon.com

GO

Cart

Your Lists

Inspect Clear All HTML CSS JS XHR Images Flash

Q

Options

GET www.amazon.com

200 OK

amazon.com

36 KB

1.03s

Headers Response

## Response Headers

Date Tue, 16 Sep 2008 05:32:05 GMT  
Server Server  
Set-Cookie skin=noskin; path=/; domain=.amazon.com; expires=Tue, 16-Sep-2008 05:32:05 GMT  
x-amz-id-1 0AMYBGG562T0KFYEETBS  
x-amz-id-2 aEsWu3ydkyoz3qCgx96qVM5pNvKju5t  
Vary Accept-Encoding, User-Agent  
Content-Encoding gzip  
Content-Type text/html; charset=ISO-8859-1  
Transfer-Encoding chunked

## Request Headers

Host www.amazon.com  
User-Agent Mozilla/5.0 (Macintosh; U; Intel Mac OS X 10.4; en-US; rv:1.9.0.1) Gecko/2008070206 Firefox/3.0.1  
Accept text/html,application/xhtml+xml,application/xml;q=0.9,\*/\*;q=0.8  
Accept-Language en-us,en;q=0.5  
Accept-Encoding gzip,deflate  
Accept-Charset ISO-8859-1,utf-8;q=0.7,\*;q=0.7  
Keep-Alive 300  
Connection keep-alive  
Cookie apn-user-id=P1BAJ09E109IU2; session-id-time=12220668001; session-id=103-3847474-5676657; ubid-main=103-8889405-6584028; session-token=ulgAdE76IbYtzH7+RyShZ74S/xphNZCucNW/0fBRwQcOP00zQUG/vLSFEhjmHlexGyzCaVFR/JtxpNxb1Fo5sSD4KNTkmhiYAPcKWp+Xa3v1I6nrP7604RuJKqW5bKIMndilySJ51M5IXI0Ub/2z2YsFNo8zqdSDS61HBP561taUZR+MrP/XDe6xwygtbtPdrpYHLzXApY=