

# INF157 - Utilisation des Réseaux

## Licence 3 Informatique

Arnaud Pecher (repris par Damien Magoni)

Bureau 322, Bâtiment A30, LaBRI  
Université de Bordeaux

Licence 3 Informatique - Bordeaux

# Cours 3

## Le protocole de navigation hypertexte : HTTP

- 1 Principes
- 2 Le protocole HTTP
  - Syntaxe
  - Echange de données
  - Les cookies
  - Le suivi de session
- 3 Les extensions d'HTTP

### *Bibliographie :*

- M. Baron, "Java pour le développement d'applications Web' : J2EE",  
<http://baron.mick.free.fr/>
- D. Donsez, <http://www-adele.imag.fr/~donsez/cours/>
- A. Tanenbaum, "Réseaux"

- 1 Principes
- 2 Le protocole HTTP
  - Syntaxe
  - Echange de données
  - Les cookies
  - Le suivi de session
- 3 Les extensions d'HTTP

Une URL sert à désigner de manière universelle une page. Elle est formée de trois parties :

- le protocole de transport ;
- le nom DNS de la machine hébergeant la page ;
- la page et sa localisation (chemin relatif) sur le serveur.

Exemple :

<https://miage.emi.u-bordeaux1.fr/espaceEtudiants/index.php>

- protocole [https](#) : protocole [http](#) sécurisé ;
- serveur [miage.emi.u-bordeaux1.fr](#) ;
- page [index.php](#) située dans le répertoire [espaceEtudiants](#).

- **HTTP** : utilisé pour la navigation web ;
- **HTTPS** : version sécurisée ;
- **FTP** : File Transfer Protocol - transfert de fichiers ;
- **file** : fichier local ;
- **news** : newsgroups ;
- **mailto** : envoi de courrier électronique ;
- **webdav** : accès en lecture/écriture sur un espace d'un serveur web ;
- ...

Lorsque l'on ouvre la page <http://www.u-bordeaux1.fr> dans un navigateur web :

- 1 le navigateur interroge un serveur DNS pour connaître l'adresse IP du serveur [www.u-bordeaux1.fr](http://www.u-bordeaux1.fr) ;
- 2 le serveur DNS répond x.x.x.x ;
- 3 le navigateur initie une connexion TCP sur le port **HTTP** (80) de la machine x.x.x.x ;
- 4 il envoie sur la connexion une requête via le protocole **HTTP** demandant le fichier par défaut : [index.html](#) ;
- 5 le serveur [www.u-bordeaux1.fr](http://www.u-bordeaux1.fr) retourne le fichier demandé via le protocole **HTTP** (ex. [/var/www/html/index.html](#)) ;
- 6 la connexion TCP est coupée ;
- 7 le navigateur affiche tout le texte contenu dans la page ;
- 8 le navigateur récupère et affiche toutes les images.

## 1 Principes

## 2 Le protocole HTTP

- Syntaxe
- Echange de données
- Les cookies
- Le suivi de session

## 3 Les extensions d'HTTP

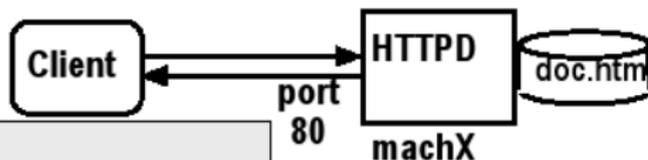
HTTP a été conçu pour être employé sur le Web, mais de façon suffisamment générique pour être employé dans un cadre moins restrictif.

L'utilisation de TCP comme protocole de transport sous-jacent garantit la fiabilité de la connexion (réémission des trames perdues, élimination des doublons, gestion des acquittements etc ...)

- avec **HTTP 1.0**, une seule requête/réception par connexion ;
- avec **HTTP 1.1**, gestion des connexions persistantes : plusieurs enchaînements requête/réponse possibles, ainsi que envois de requêtes en rafale, sans attendre les réponses.

# Exemple de requête HTTP

GET /doc.htm



le Client envoie

```
GET /doc.htm HTTP/1.0      méthode, chemin, version
Accept: www/source        documents acceptés
Accept: text/html
Accept: image/gif
User-Agent: Lynx/2.2 libwww/2.14
From: alice@pays.merveilles.net
* une ligne blanche *
```

le Serveur répond

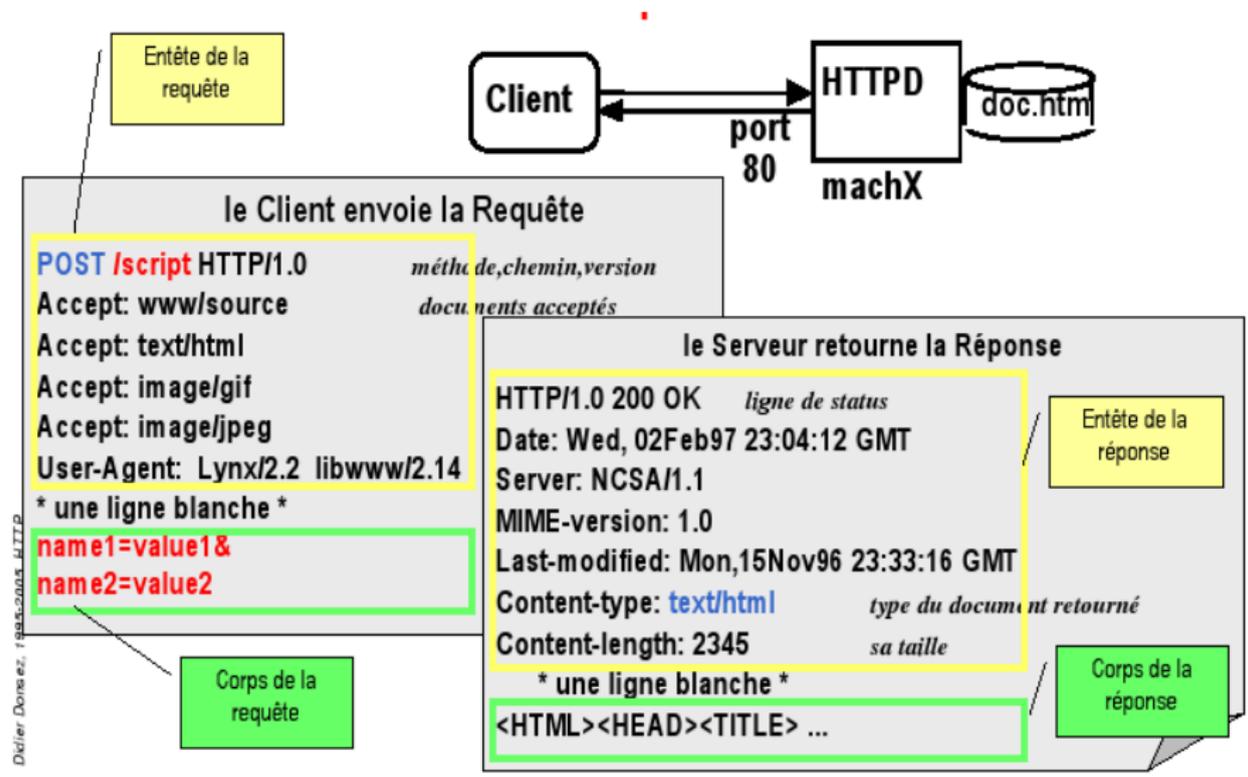
```
HTTP/1.0 200 OK          ligne de status
Date: Wed, 02Feb97 23:04:12 GMT
Server: NCSA/1.1
MIME-version: 1.0
Last-modified: Mon, 15Nov96 23:33:16 GMT
Content-type: text/html  type du document retourné
Content-length: 2345     sa taille
* une ligne blanche *
<HTML><HEAD><TITLE> ...
```

Protocole en mode de lignes de caractère

- à la main : `telnet miage.u-bordeaux.fr 80` ;
- Types de dialogue
  - récupération d'un document : méthode `GET` ;
  - soumission d'un formulaire : méthode `GET` ou `POST` ;
  - envoi de document et gestion de site : méthodes `PUT`, `DELETE`, `LINK`, `UNLINK` ;
  - gestion de proxy/cache : méthode `HEAD` (récupération des informations sur le document)

La compréhension du protocole est fondamentale pour la programmation web...

# Entête/Corps d'une Requête/Réponse



©D. Donsez

# Syntaxe d'une requête

## ■ Envoyé par le client au serveur

<Méthode> <URI> HTTP/<Version>

[<Champ d'entête>: <Valeur>]

[<tab><Suite Valeur si >1024>]

*ligne blanche*

[corps de la requête pour la méthode POST]

```
GET /docu2.html HTTP/1.0
```

```
Accept: www/source
```

```
Accept: text/html
```

```
Accept: image/gif
```

```
User-Agent: Lynx/2.2 libwww/2.14
```

```
From: alice@pays.merveilles.net
```

```
* une ligne blanche *
```

```
POST /script HTTP/1.0
```

```
Accept: www/source
```

```
Accept: text/html
```

```
Accept: image/gif
```

```
User-Agent: Lynx/2.2 libwww/2.14
```

```
From: alice@pays.merveilles.net
```

```
Content-Length: 24
```

```
* une ligne blanche *
```

```
name1=value1&
```

```
name2=value2
```

# Syntaxe d'une réponse

## ■ Réponse envoyé par le serveur au client

HTTP/<Version> <Status> <Commentaire Status>

Content-Type: <Type MIME du contenu>

[< Champ d 'entête >: <Valeur>]

[<tab><Suite Valeur si >1024>]

*Ligne blanche*

Document

```
HTTP/1.0 200 OK
Date: Wed, 02Feb97 23:04:12 GMT
Server: NCSA/1.1
MIME-version: 1.0
Last-modified: Mon,15Nov96 23:33:16 GMT
Content-type: text/html
Content-length: 2345
  * une ligne blanche *
<HTML><HEAD><TITLE> ...
</BODY></HTML>
```

# Connexion HTTP à la main

## Telnet

HTTP étant un protocole texte en ASCII, on peut communiquer de manière "brute", via le programme **telnet** :

```
telnet miage.u-bordeaux.fr 80 > log
GET /index.htm HTTP/1.1
Host: miage.u-bordeaux.fr
```

```
close
```

- on initie une connexion TCP sur le port 80 de **miage.u-bordeaux.fr** ;
- demande le fichier **index.html** avec la version 1.1 de HTTP ;
- **Host** spécifie l'hôte de destination (si plusieurs par adresse IP) ;
- ligne vide permet d'informer le serveur de la fin des entêtes ;
- **close** permet de mettre fin à la connexion.

# Connexion HTTP à la main

Examen du fichier généré

## Entête

---

```
Trying 147.210.36.199...
Connected to w3bdx1.drimm.u-bordeaux1.fr.
Escape character is '^]'.
GET /index.htm HTTP/1.1
Host: miage.u-bordeaux.fr

HTTP/1.1 200 OK
Date: Tue, 25 Oct 2005 19:57:48 GMT
Server: Apache/1.3.26 (Unix) Debian GNU/Linux mod_fastcgi/2.2.10 PHP
      /4.1.2 mod_ssl/2.8.9 OpenSSL/0.9.6c mod_perl/1.26 DAV/1.0.3
Last-Modified: Thu, 15 Jul 2004 13:00:58 GMT
ETag: "9b003b-263-40f6800a"
Accept-Ranges: bytes
Content-Length: 611
Content-Type: text/html
```

# Connexion HTTP à la main

## Examen du fichier généré

Données (code HTML de la page)

```
<html>
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Frameset//EN" "http://www.
    w3.org/TR/html4/frameset.dtd">
<head>
<title> .oO MIAGe de Bordeaux Oo. </title>
</head>

<frameset rows="99,*" cols="*" frameborder="NO" border="0" framespacing
    ="0">
    <frame src="titre.htm" name="topFrame" scrolling="NO" noresize />
    <frameset rows="*" cols="122,*" framespacing="0" frameborder="NO"
        border="0">
        <frame src="menu.htm" name="leftFrame" noresize/>
        <frame src="http://miage.labri.fr/FRONT/accueil.htm" name="
            mainFrame"/>
    </frameset>
</frameset>
<noframes><body>

</body></noframes>
</html>
```

## MonServeurWeb

Ecrire une classe **MonServeurWeb** qui retourne systématiquement une même page HTML fixée.

# Solution

```
import java.net.*; import java.io.*;

public class MonServeurWeb{

    public static void main( String[] args ) throws Exception {
        ServerSocket ss = new ServerSocket( 80 );
        System.out.println("mon serveur web en attente");
        Socket s = ss.accept();

        InputStream is = s.getInputStream(); // mise en place du canal en lecture
        LineNumberReader lnr = new LineNumberReader( new InputStreamReader(is) );

        String requete=""; String ligne = " ";
        while ((ligne.compareTo(""))!=0){ // lecture ligne par ligne de la requete, arret sur la ligne vide
            ligne = lnr.readLine(); requete+=ligne;
        }

        PrintWriter pw = new PrintWriter(s.getOutputStream());
        String page = "<html>\n<title>Ma super page web</title>\n</head>\n<body>\n<h1>Wow il est rapide ce serveur web!!</h1>\n</body>\n</html>";
        String reponse = "<heHTTP/1.1 200 OKDate: Thu, 10 Jun 2004 12:19:17 GMT\nServer: MonServeurWeb\nLast-Modified: Tue, 27 Aug 2002 16:21:3 6 GMT\nAccept-Ranges: bytes\nContent-Length: "+page.length()+"\nKeep-Alive: timeout =15, max=100\nConnection: Keep-Alive\nContent-Type: text/html; charset=UTF-8\n\n"+page;

        pw.print(reponse);
        pw.flush();
        s.close();
    }
}
```

### Paramètres d'une requête

- **User-Agent**, informations sur le navigateur et sa plateforme ;
- **Accept**, le type de pages que le client peut traiter ;
- **Accept-Charset**, les jeux de caractères acceptés par le client ;
- **Accept-Encoding**, les codage de page acceptés par le client ;
- **Host**, le nom DNS du serveur ;
- **Authorization**, informations sur l'identité du client ;
- **Cookie**, retourne un cookie préalablement placé par le serveur.

### Exercice

Ecrire une fonction qui affiche le contenu des différents paramètres de la requête.

### Paramètres requête & réponse

- **Date**, horodatage du message ;
- **Upgrade**, le protocole sur lequel l'émetteur veut basculer ;

### Paramètres d'une réponse

- **Content-Encoding**, méthode de codage du contenu ;
- **Content-Length**, la longueur de la page en octets ;
- **Last-Modified**, heure et date dernière modification de la page ;
- **Location**, demande au client d'envoyer sa requête ailleurs ;
- **Set-Cookie**, demande au client d'enregistrer un cookie.

### Exercice

Ecrire des fonctions retournant des paramètres de réponse adaptés.

Chaque requête est suivie d'une réponse, contenant un code d'état à 3 chiffres :

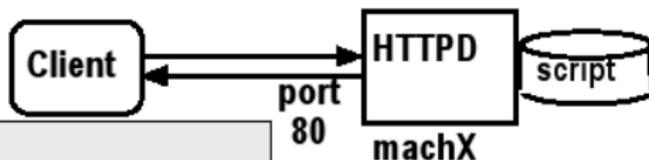
- **1xx** : Information ;
- **2xx** : Succès - exemple : 200 = la requête a réussi ;
- **3xx** : Redirection - exemples : 301 = page déplacée, 304 = page en cache toujours valide ;
- **4xx** : Erreur client - exemples : 403 = page interdite, 404 = page inexistante ;
- **5xx** : Erreur serveur - exemples : 500 = erreur interne, 503 = recommencer plus tard.

### Exercice

En utilisation le paramètre **Location**, modifier notre serveur web pour qu'il redirige systématiquement vers la page d'accueil du LaBRI.

# Envoi de données - GET

**GET** /script?name1=value1&name2=value2



le Client envoie

**GET** /script?name1=value1&name2=value2 HTTP/1.0

Accept: www/source

*documents acceptés*

Accept: text/html

Accept: image/gif

User-Agent: Lynx/2.2 libwww/2.14

From: alice@pays.merveilles.net

\* une ligne blanche \*

le Serveur répond

HTTP/1.0 200 OK *ligne de status*

Date: Wed, 02Feb97 23:04:12 GMT

Server: NCSA/1.1

MIME-version: 1.0

Last-modified: Mon,15Nov96 23:33:16 GMT

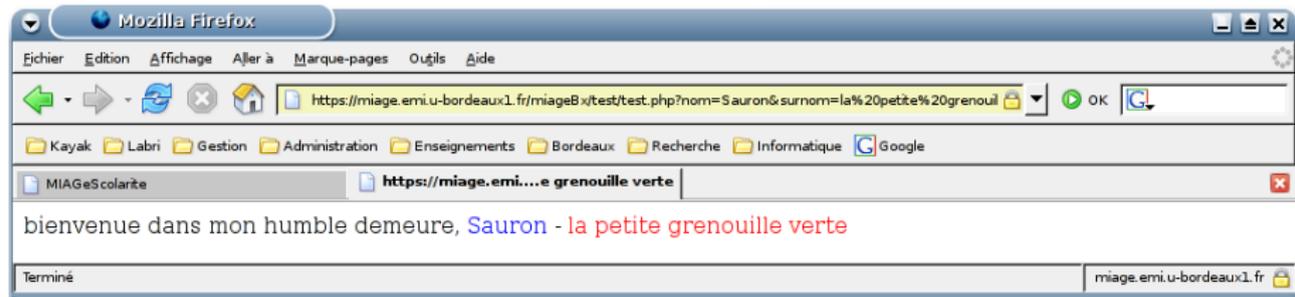
Content-type: **text/html** *type du document retourné*

Content-length: 2345 *sa taille*

\* une ligne blanche \*

<HTML><HEAD><TITLE> ...

# GET : passage de valeurs dans l'URL





# Les valeurs ont transité en clair

The screenshot shows the Wireshark network protocol analyzer interface. The main pane displays a list of captured packets, with packet 1 selected. The packet list shows:

No.	Time	Source	Destination	Protocol	Info
10	0.2346	66.249.93.99	192.168.0.2	HTTP	Continuation of non-HTTP traffic
12	0.2351	66.249.93.99	192.168.0.2	HTTP	Continuation of non-HTTP traffic
1	0.0000	192.168.0.2	66.249.93.99	HTTP	GET /search?hl=fr&q=Sauron+grenouille+verte&btnG=Recherche+Google&meta= HTTP/1.1
4	0.1435	66.249.93.99	192.168.0.2	HTTP	HTTP/1.1 200 OK[Unreassembled Packet]
3	0.1293	66.249.93.99	192.168.0.2	TCP	[TCP Window Update] www > 32968 [ACK] Seq=0 Ack=614 Win=6420 Len=0
2	0.1199	66.249.93.99	192.168.0.2	TCP	www > 32968 [ACK] Seq=0 Ack=614 Win=7576 Len=0

The packet details pane for the selected packet (No. 1) shows the following structure:

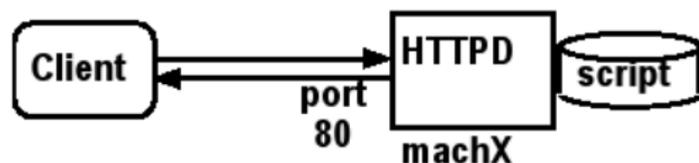
- Frame 1 (668 bytes on wire, 668 bytes captured)
- Ethernet II, Src: 00:90:4b:16:5d:24, Dst: 00:09:5b:9a:c4:94
- Internet Protocol, Src Addr: 192.168.0.2 (192.168.0.2), Dst Addr: 66.249.93.99 (66.249.93.99)
- Transmission Control Protocol, Src Port: 32968 (32968), Dst Port: www (80), Seq: 0, Ack: 0, Len: 614
- Hypertext Transfer Protocol
  - GET /search?hl=fr&q=Sauron+grenouille+verte&btnG=Recherche+Google&meta= HTTP/1.1\r\n
  - Host: ww.google.fr\r\n
  - User-Agent: Mozilla/5.0 (X11; U; Linux i686; fr; rv:1.7.12) Gecko/20050922 Firefox/1.0.7 (Ubuntu package 1.0.7)\r\n
  - Accept: text/xml,application/xml,application/xhtml+xml,text/html;q=0.9,text/plain;q=0.8,image/png; /\*; q=0.5\r\n
  - Accept-Language: fr,fr-fr;q=0.8,en-us;q=0.5,en;q=0.3\r\n
  - Accept-Encoding: gzip,deflate\r\n
  - Accept-Charset: ISO-8859-1,utf-8;q=0.7,\*;q=0.7\r\n
  - Keep-Alive: 300\r\n
  - Connection: keep-alive\r\n
  - Referer: http://www.google.fr/\r\n
  - Cookie: PREF=ID=89772ab7a209afc:TM=1130354227:LM=1130354227:S=e3t1xigXC7ny04Xf\r\n\r\n

The packet bytes pane shows the raw data of the captured packet, with the first few lines highlighted in blue:

```
0030 59 60 19 17 00 00 47 45 54 20 2f 73 65 61 72 63 Y*...GET/search
0040 58 3f 68 6c 3d 66 72 26 71 3d 53 61 75 72 6f 6e h?hl=fr&q=Sauron
0050 2b 67 72 65 6e 6f 75 69 6c 6c 65 2b 76 65 72 74 +grenouille+verte
0060 55 26 62 74 6e 47 3d 52 65 63 68 65 72 63 68 65 e&btnG=Recherche
0070 2b 47 6f 6f 67 6c 65 26 6d 65 74 61 3d 20 48 54 +Google&meta=HI
0080 54 50 2f 31 2e 31 0d 0d 48 6f 73 74 3a 20 77 77 IP/1.1\r\n Host: ww
0090 77 2e 67 6f 6f 67 6c 65 2e 66 72 0d 0a 55 73 65 w.google.fr..Use
00a0 72 2d 41 67 65 6e 74 3a 20 4d 6f 7a 69 6c 6c 61 r-Agent: Mozilla
00b0 2f 35 2e 30 20 28 58 31 31 3b 20 55 3b 20 4c 69 /5.0 (X11; U; Li
00c0 6e 75 78 20 69 36 38 36 3b 20 66 72 3b 20 72 76 nux i686 ; fr; rv
00d0 3a 31 2e 37 2e 31 32 29 20 47 65 63 6b 6f 2f 32 :1.7.12) Gecko/2
00e0 30 30 35 30 39 32 32 20 46 69 72 65 66 6f 78 2f 0050922 Firefox/
00f0 31 2e 30 2e 37 20 28 55 62 75 6e 74 75 20 70 61 1.0.7 (U buntu pa
0100 63 6b 61 67 65 20 31 2e 30 2e 37 29 0d 0a 41 63 ckage 1.0.7)..Ac
0110 63 65 70 74 3a 20 74 65 78 74 2f 78 6d 6c 2c 61 cept: te xt/xml,a
0120 70 70 6c 60 63 61 74 60 6f 6e 2f 78 6d 6c 2c 61 pplication/xml;a
```

# Envoi de données - POST

## POST /script



le Client envoie

```
POST /script HTTP/1.0
Accept: www/source
Accept: text/html
Accept: image/gif
User-Agent: Lynx/2.2 libwww/2.14
From: alice@pays.merveilles.net
  * une ligne blanche *
name1=value1&
name2=value2
```

le Serveur répond

```
HTTP/1.0 200 OK
...
Content-length: 2345
  * une ligne blanche *
<HTML><HEAD><TITLE> ...
```

# Formulaire envoyé avec POST

Da Linux French Page - Mozilla Firefox

Fichier Edition Affichage Aller à Marque-pages Outils Aide

http://linuxfr.org/pub/

Kayak Labri Gestion Administration Enseignements Bordeaux Recherche Informatique Google

UUCPssh | TribuneLibre | LinuxGraphic | Agenda du libre | Léa-Linux | Lolix | LinuxFrench | O'Reilly | LinuxMag | Eyrolles | Linux-France

Faire un don ! | accès sécurisé/SSL | style | créer un compte | statistiques | charge serveur | contactez-nous | [par recherche...](#) lettre d'information

Accueil :: Dépêches :: Archives :: Proposer une dépêche :: Journaux :: Forums :: Astuces :: Suivi :: RDF

**LINUXFR.ORG**

Bob

Connexion automatique

Connexion

Attention! Votre mot de passe transitera en clair sur Internet. Nous vous conseillons d'utiliser l'accès sécurisé/SSL.  
[Créer un compte](#)

**Articles : Les Européens de l'année sont-ils pour ou contre la brevetabilité des logiciels ?**

 Florian Mueller de [NoSoftwarePatents.com](#) et le député socialiste européen Michel Rocard, rapporteur pour la directive sur les brevets logiciels, sont proposés pour être les Européens de l'année, élection organisée par le magazine [EuropeanVoice](#). Charlie McCreedy, commissaire pro-brevets logiciels ayant pris la place de Fritz Bolkestein, est aussi proposé comme Commissaire de l'année. La cérémonie de remises des prix sera présidée par Pat Cox, ancien président du Parlement Européen, et lui aussi lobbyiste pour l'EICTA, l'association regroupant la majorité des grandes entreprises pro-brevets logiciels à Bruxelles. NDM : les catégories sont commissaire, parlementaire, homme d'État, diplomate, homme de campagne, homme d'affaires, journaliste, homme à succès, citoyen non-européen et citoyen européen(...)  
> [Lire l'article](#) (17 commentaires, moyenne: 2.6).

**X-Window**

```
top + xrootconsole
Voulant afficher un top sur mon bureau, j'ai naturellement lancé top | xrootconsole. Malheur, on voit des caractères hideux mon top me renseigne, je lance donc top -b | xrootconsole [et des option]. Et là(...)  
> Lire l'astuce (1 commentaire, moyenne: 2)
```

**Logiciel : Galeon fusionne avec Epiphany**

Posté par Zorro . Modéré le mardi 25 octobre.

http://linuxfr.org

# Vérification : trame

The image shows a Wireshark interface with a packet capture of an HTTP POST request. The packet list pane shows several packets, with packet 10 selected. The packet details pane shows the structure of the selected packet, including Ethernet II, Internet Protocol, Transmission Control Protocol, and Hypertext Transfer Protocol. The Hypertext Transfer Protocol section is expanded to show the request details.

No.	Time	Source	Destination	Protocol	Info.
8	0.2045	212.27.33.22	192.168.0.2	TCP	www > 33028 [SYN, ACK] Seq=0 Ack=1 Win=5792 Len=0 MSS=1460 TSV=2136257567 TSER=7969757 WS=0
9	0.2045	192.168.0.2	212.27.33.22	TCP	33028 > www [ACK] Seq=1 Ack=1 Win=5840 Len=0 TSV=7969806 TSER=2136257567
10	0.2047	192.168.0.2	212.27.33.22	HTTP	POST /login.html HTTP/1.1
11	0.2595	212.27.33.22	192.168.0.2	TCP	www > 33028 [ACK] Seq=1 Ack=477 Win=6432 Len=0 TSV=2136257573 TSER=7969807
12	0.2596	192.168.0.2	212.27.33.22	HTTP	Continuation or non-HTTP traffic (application/x-www-form-urlencoded)
13	0.3095	212.27.33.22	192.168.0.2	TCP	www > 33028 [ACK] Seq=1 Ack=601 Win=6432 Len=0 TSV=2136257578 TSER=7969862
14	0.4054	212.27.33.22	192.168.0.2	HTTP	HTTP/1.1 200 OK (text/html)

Frame 10 (542 bytes on wire, 542 bytes captured)  
Ethernet II, Src: 00:90:4b:16:5d:24, Dst: 00:09:5b:9a:c4:94  
Internet Protocol, Src Addr: 192.168.0.2 (192.168.0.2), Dst Addr: 212.27.33.225 (212.27.33.225)  
Transmission Control Protocol, Src Port: 33028 (33028), Dst Port: www (80), Seq: 1, Ack: 1, Len: 476  
Hypertext Transfer Protocol  
POST /login.html HTTP/1.1\r\nRequest Method: POST  
Request URI: /login.html  
Request Version: HTTP/1.1  
Host: linuxfr.org\r\nUser-Agent: Mozilla/5.0 (X11; U; Linux i686; fr; rv:1.7.12) Gecko/20050922 Firefox/1.0.7 (Ubuntu package 1.0.7)\r\nAccept: text/xml,application/xml,application/xhtml+xml,text/html;q=0.9,text/plain;q=0.8,image/png,\*/\*;q=0.5\r\nAccept-Language: fr,fr-fr;q=0.8,en-us;q=0.5,en;q=0.3\r\nAccept-Encoding: gzip,deflate\r\nAccept-Charset: ISO-8859-1,utf-8;q=0.7,\*;q=0.7\r\nKeep-Alive: 300\r\nConnection: keep-alive\r\nReferer: http://linuxfr.org/pub/

Offset	Hex	ASCII
0040	b4 1f 50 4f 53 54 20 2f	..POST / login.ht
0050	6c 0c 20 48 54 54 50 2f	ml HTTP/ 1.1..Hos
0060	74 3a 20 6c 69 6e 75 78	t: linux fr.org..
0070	55 73 65 72 2d 41 67 65	User-Age nt: Mozi
0080	6c 6c 61 2f 35 2e 30 20	lla/5.0 (X11; U;
0090	20 4c 69 6e 75 78 20 69	Linux i 686; fr;
00a0	20 72 76 3a 31 2e 37 2e	rv:1.7. 12) Geck
00b0	6f 2f 32 30 30 35 30 39	o/200509 22 Firef
00c0	6f 78 2f 31 2e 30 2e 37	ox/1.0.7 (Ubuntu
00d0	20 70 61 63 6b 61 67 65	package 1.0.7).
00e0	0a 41 63 63 65 70 74 3a	.Accept: text/xm
00f0	6c 2c 61 70 70 6c 69 63	l,applic ation/xm
0100	6c 2c 61 70 70 6c 69 63	l,applic ation/xh
0110	74 6d 6c 2b 78 6d 6c 2c	tml+xml, text/htm
0120	6c 3b 71 3d 30 2e 39 2c	l;q=0.9, text/pla
0130	69 6e 3b 71 3d 30 2e 38	in;q=0.8 image/pn

# Le mot de passe a transité en clair...

The image shows a Wireshark capture of an HTTP form submission. The packet list pane shows a SYN-ACK, followed by a POST request, and then an ACK. The packet details pane for the POST request (Frame 12) shows the following structure:

- Ethernet II, Src: 00:90:4b:16:5d:24, Dst: 00:09:5b:9a:c4:94
- Internet Protocol, Src Addr: 192.168.0.2 (192.168.0.2), Dst Addr: 212.27.33.225 (212.27.33.225)
- Transmission Control Protocol, Src Port: 33028 (33028), Dst Port: ww (80), Seq: 477, Ack: 1, Len: 124
- Hypertext Transfer Protocol
  - Content-Type: application/x-www-form-urlencoded\r\n
  - Content-Length: 53\r\n
  - \r\n
  - Line-based text data: application/x-www-form-urlencoded
    - url=%2Fpub%2F6&login=Bob&passwd=Greg&Envoyer=Connexion

The packet bytes pane shows the raw data of the POST request, with the password 'Greg' clearly visible in the 'passwd' field of the form data.

No.	Time	Source	Destination	Protocol	Info.
8	0.2045	212.27.33.22	192.168.0.2	TCP	www > 33028 [SYN, ACK] Seq=0 Ack=1 Win=5792 Len=0 MSS=1460 TSV=2136257567 TSER=7969757 WS=0
9	0.2045	192.168.0.2	212.27.33.22	TCP	33028 > www [ACK] Seq=1 Ack=1 Win=5840 Len=0 TSV=7969806 TSER=2136257567
10	0.2047	192.168.0.2	212.27.33.22	HTTP	POST /login.html HTTP/1.1
11	0.2595	212.27.33.22	192.168.0.2	TCP	www > 33028 [ACK] Seq=1 Ack=477 Win=6432 Len=0 TSV=2136257573 TSER=7969807
12	0.2596	192.168.0.2	212.27.33.22	HTTP	Continuation of non-HTTP traffic (application/x-www-form-urlencoded)
13	0.3095	212.27.33.22	192.168.0.2	TCP	www > 33028 [ACK] Seq=1 Ack=601 Win=6432 Len=0 TSV=2136257578 TSER=7969862

Frame 12 (190 bytes on wire, 190 bytes captured)

Ethernet II, Src: 00:90:4b:16:5d:24, Dst: 00:09:5b:9a:c4:94

Internet Protocol, Src Addr: 192.168.0.2 (192.168.0.2), Dst Addr: 212.27.33.225 (212.27.33.225)

Transmission Control Protocol, Src Port: 33028 (33028), Dst Port: ww (80), Seq: 477, Ack: 1, Len: 124

Hypertext Transfer Protocol

- Content-Type: application/x-www-form-urlencoded\r\n
- Content-Length: 53\r\n
- \r\n
- Line-based text data: application/x-www-form-urlencoded
  - url=%2Fpub%2F6&login=Bob&passwd=Greg&Envoyer=Connexion

## Exercice

Modifier notre serveur web pour qu'il puisse récupérer les valeurs passées par le client avec la méthode GET.

## Exercice

Modifier notre serveur web pour qu'il puisse récupérer les valeurs passées par le client avec la méthode POST.

# Cookie (1)



*Web browser's cookies table*

VisitorID=12345; path=/;

SID\_book=88; path=/book/;

SID\_music=999; path=/music/;

GET /

Set-Cookie: VisitorID=12345; path=/;

GET /book

Cookie: VisitorID=12345;

Set-Cookie: SID\_book=88; path=/book/;

GET /book/order

Cookie: VisitorID=12345;

Cookie: SID\_book=88;

GET /music

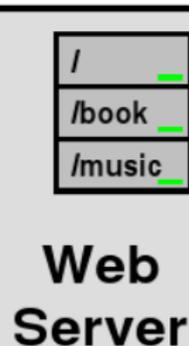
Cookie: VisitorID=12345;

Set-Cookie: SID\_music=999; path=/music/;

GET /music/buy

Cookie: VisitorID=12345;

Cookie: SID\_music=999;



Web Server

# Cookie (2)

*Un autre jour !*



*Web browser' cookies table*

VisitorID=12345; path=/;

GET /  
Cookie: VisitorID=12345;

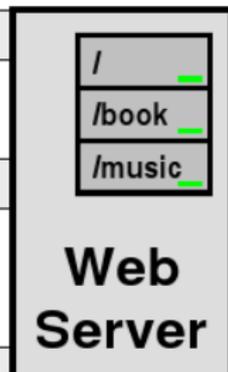
GET /book  
Cookie: VisitorID=12345;

Set-Cookie: SID\_book=101; path=/book;

GET /book/logout  
Cookie: VisitorID=12345;  
Cookie: SID\_book=101;

Set-Cookie: SID\_book=0; path=/book; \ expires=0

GET /  
Cookie: VisitorID=12345;



**Motivation** : la notion de session est importante dans de nombreuses applications (ex. commerce électronique - gestion d'un panier )

Cependant avec HTTP, le serveur ne maintient pas d'informations liées aux requêtes précédentes d'un même client : les enchainements Requête/Réponse sont indépendants les uns des autres.

**Comment implanter la notion de session sur plusieurs requêtes HTTP ?**

## Méthodes

- le serveur génère un identificateur de session et associe un état (et une date limite de validité) à une session ;
- le client renvoie l'identificateur de session à chaque requête HTTP vers le serveur ;

## Echange et Stockage de l'identificateur de session

- input HIDDEN dans les formulaires ;
- réécriture des URLs ;
- cookies (désactivable) ;
- identificateur de session SSL (Secure Socket Layer) ;

L'identifiant de session est encodé dans les URLs des documents HTML retournés par le serveur :

Exemple :

`http://www.bob.fr/servlet/cart?PROD_ID=383`

devient

`http://www.bob.fr/servlet/cart;jsessionid=To1128mC33718557521577075At?PROD_ID=383`

Limite : l'URL doit être générée par un script (programmation côté serveur)

Chaîne décrivant l'état d'une session :

- NAME=. ;
- expires=. ;
- path=. ;
- domain=. ;

stockée sur le client.

Communiquée dans les entêtes de requêtes (Cookie :) et dans les entêtes des réponses HTTP (Set- Cookie :)

**Limites** : 300 cookies simultanées par client, 20 cookies par serveur ou domaine, 4Ko par cookie.

**Un client peut refuser des cookies.**

# Le serveur Apache

## Statistiques

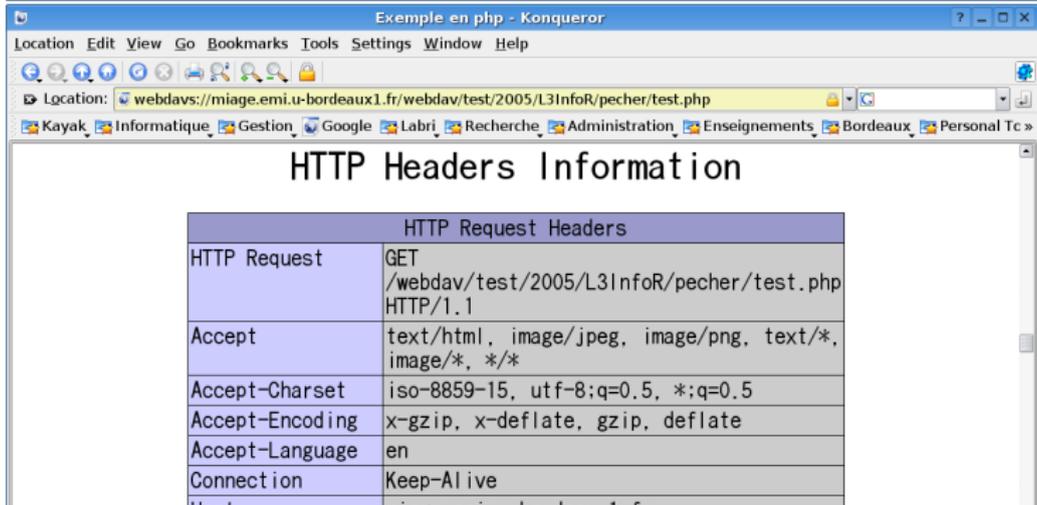
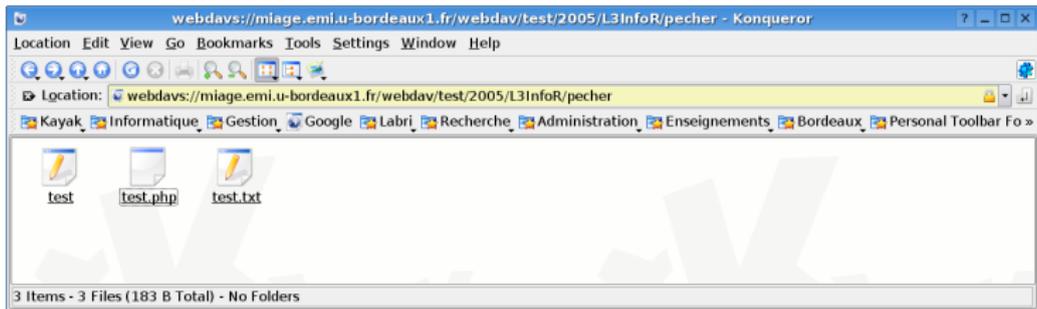
- <http://www.netcraft.com/survey>
  - Janvier 2003 (pour 35,424,956 sites)

Serveur	Nombre	Pourcentage
Apache	11178715	66.42
Microsoft	4172101	24.79
Zeus	261652	1.55
SunONE	233105	1.39

- 1 Principes
- 2 Le protocole HTTP
  - Syntaxe
  - Echange de données
  - Les cookies
  - Le suivi de session
- 3 Les extensions d'HTTP

### Extension de HTTP pour la mise à jour de site :

- ajout de nouvelles commandes HTTP :
  - PROPFIND : retourne les propriétés ;
  - PROPPATCH : modifie les propriétés ;
  - MKCOL : crée une nouvelle collection ;
  - COPY & MOVE : copie ou déplace une ressource au sein d'un espace de nommage ;
  - LOCK & UNLOCK : verrouille et déverrouille un
- serveurs WebDAV : MS IIS5, mod\_dav pour Apache, support dans Tomcat 4...
- clients WebDAV : MS Office, Cadaver, konqueror...



## Motivation

- Sécuriser (authentification, confidentialité) l'accès à un service Web
- SSL ;
  - Authentification du serveur et/ou du client par PKI ;
  - Chiffrement avec une clé (secrète) symétrique de session ;
  - Reprise après déconnexion
- HTTP over SSL : URL : `https ://host/document` (Port TCP par défaut : 443)
- HTTPS en java :
  - JSSE (`javax.net.ssl.*`) inclus dans Java 1.4 ;
  - classe `javax.net.HttpsURLConnection` ;
- Serveurs : Apache/SSL (ex. [miage.emi.u-bordeaux1.fr](https://miage.emi.u-bordeaux1.fr)), iPlanet, MS IIS, OracleAS, IBM WebSphere ;

- proposé par IBM au W3C (**en cours d'instruction**) ;
- HTTP est une couche de transport pour les messages SOAP dans la mise en oeuvre les Web Services (B2B, EAI) ;
- motivations :
  - fiabiliser l'échange de messages : échanger une et une seul fois même en cas d'interruption ;
  - regrouper plusieurs messages par commande.
- mécanismes :
  - journaliser les messages (requêtes et réponses) côté client et côté serveur ;
  - notion de transaction ;
  - Valider/Acquiter les échanges ;
  - ajout de nouvelles commandes.

Ces commandes sont portées par la commande HTTP POST :

- PUSH
  - Le client envoie des messages journalisés par le serveur ;
  - La séquence de messages est associée à un identifiant de transaction ;
  - Peut contenir des acquittements à des commandes précédentes.
- PULL
  - Demande à recevoir des messages en attente coté serveur ;
  - Peut contenir des acquittements.
- EXCHANGE : combine PUSH et PULL ;
- RESOLVE : déterminer les messages journalisés par le serveur ;
- REPORT : rapporter au serveur journalisés par le client.

4 Fondamentaux

5 Un conteneur : Tomcat

6 Servlets

Cette partie est basée fortement sur les transparents de cours de Mickaël Baron :

*Bibliographie :*

- M. Baron, "Java pour le développement d'applications Web' : J2EE" (2006),  
<http://baron.mick.free.fr/>

4 Fondamentaux

5 Un conteneur : Tomcat

6 Servlets

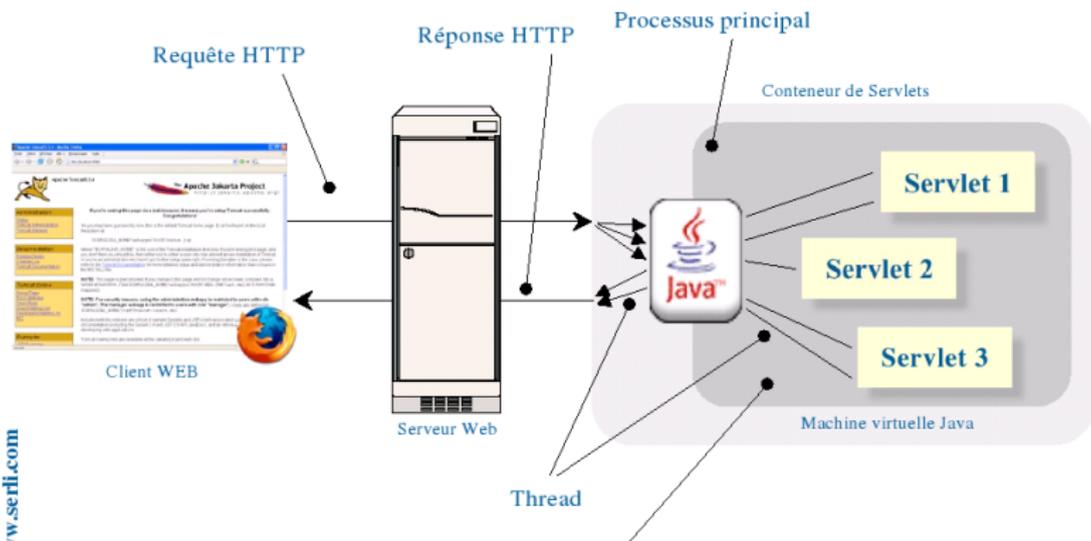
## Definition

Servlet Composant logiciel écrit en Java fonctionnant côté serveur.

### Concurrents directs :

- scripts CGI (Common Gateway Interface) ;
- langages de script PHP, ASP etc ...

Une Servlet s'exécute dans un conteneur de Servlets.



Les Servlets peuvent être toutes gérées par des threads séparés au sein d'un même processus de machine virtuelle

©M. Baron (2006)

- **Portabilité** :
  - Technologie indépendante de la plate-forme et du serveur ;
  - Un langage (Java) et plusieurs plate-forme (.NET plusieurs langages et une plate-forme) ;
- **Puissance** :
  - Disponibilité de l'API de Java ;
  - Manipulation d'images, connectivité aux bases de données (JDBC), etc.
- **Efficacité** :
  - Une Servlet est chargée une seule fois ;
  - Une Servlet conserve son état ;
- **Sécurité** :
  - Typage fort de Java ;
  - Gestion des erreurs par exception.

# Une première Servlet

Ne pas oublier d'importer la bibliothèque Java des Servlets

HelloWorld est un objet de type HttpServlet

Redéfinition de la méthode doGet (traitement d'une requête GET)

```
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;

public class HelloWorld extends HttpServlet {

    public void doGet(HttpServletRequest req, HttpServletResponse res)
        throws ServletException, IOException {
        res.setContentType("text/html");
        PrintWriter out = res.getWriter();

        out.println("<HTML>");
        out.println("<HEAD><TITLE>Bonjour tout le monde</TITLE></HEAD>");
        out.println("<BODY>");
        out.println("<BIG>Bonjour tout le monde</BIG>");
        out.println("</BODY></HTML>");
    }
}
```

www.servlet.com

Réponse sous format HTML de la Servlet HelloWorld



Le résultat sur le client

Servlets - M. Baron - Page 41

©M. Baron (2006)

4 Fondamentaux

5 Un conteneur : Tomcat

6 Servlets

## Definition

Conteneur Une Servlet s'exécute dans un **conteneur de Servlet** permettant d'établir le lien entre la Servlet et le serveur Web.

## Deux types de conteneurs :

- conteneurs de Servlets **autonomes** : serveur web intégrant le support des Servlets ;
- conteneurs de Servlets **additionnels** : fonctionnent comme un plug-in à un serveur web existant.

## Principaux conteneurs :

- Tomcat Server (Apache) ;
- JBoss ;
- WebSphere Application Server (IBM) ;
- Weblogic (BEA) ;
- ...

- Tomcat 5 respecte la spécification Servlet 2.4 et JSP 2.0 ;
- Ecrit entièrement en Java, il peut donc être utilisé sur n'importe quel système disposant d'une machine virtuelle ;
- Disponible gratuitement sous forme d'une licence Open Source ;
- Implémentation de référence de la spécification J2EE. Il fournit donc les librairies de façon à concevoir des Servlets (`javax.servlet.http.HttpServlet`)

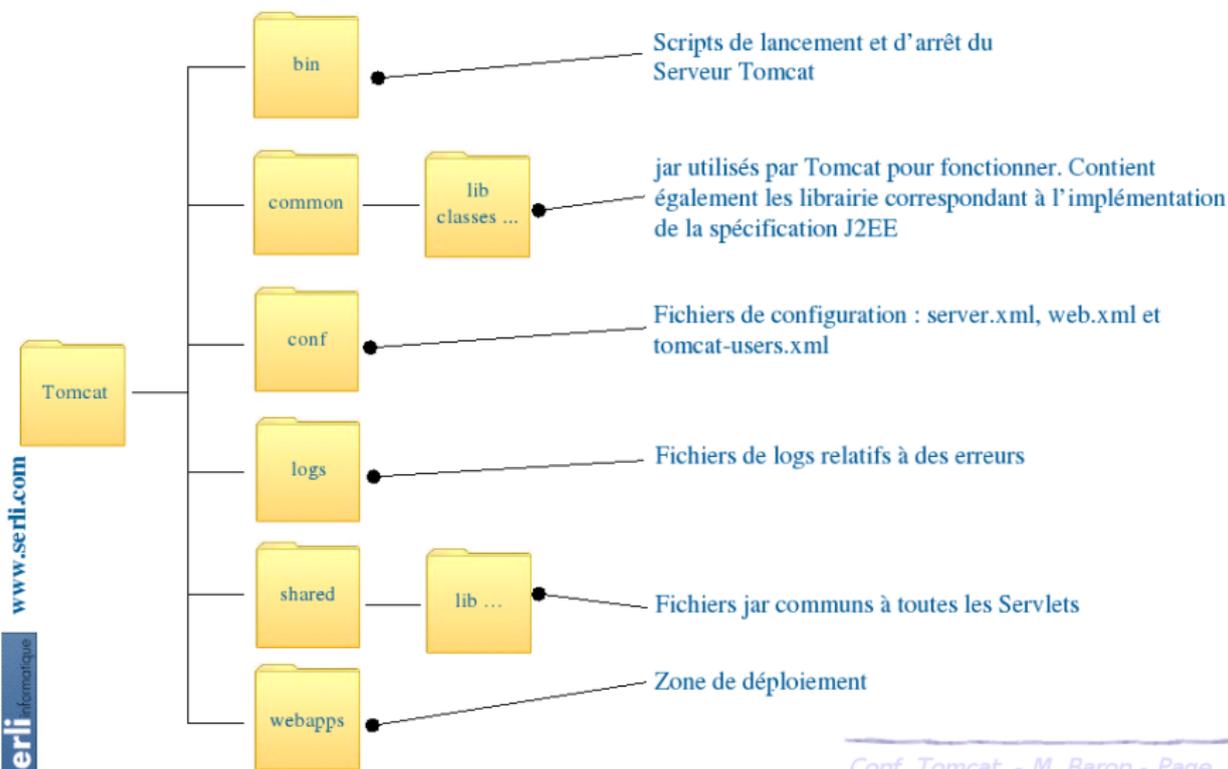
## 2 options :

- 1 avec le gestionnaire de paquets : `apt-get install tomcat5.5 tomcat5.5-admin tomcat5.5-webapps`

Ubuntu Feisty : cassé, plusieurs problèmes !

- 2 avec la version archive :
  - 1 Décompresser l'archive ;
  - 2 Modifier la variable PATH de façon à prendre en compte le chemin binaire de Tomcat ;
  - 3 Ajouter la variable CATALINA\_HOME pointant sur le chemin de Tomcat
  - 4 Ajouter la variable JAVA\_HOME pointant sur la machine virtuelle
  - 5 Compléter la variable CLASSPATH pointant sur les librairies J2EE (`CLASSPATH=$CLASSPATH :tomcat/common/lib`)
  - 6 Pour vérifier que votre serveur fonctionne, lancer `startup.sh`

# Hiérarchie des répertoires de Tomcat



## Definition

Rôle Permet d'ajouter des utilisateurs et de définir des droits sur les Servlets.

Ajout, suppression et modification des rôles :

- soit par la Servlet **Administration** ;
- soit en éditant directement le fichier **tomcat-users.xml**

**tomcat-users.xml**

---

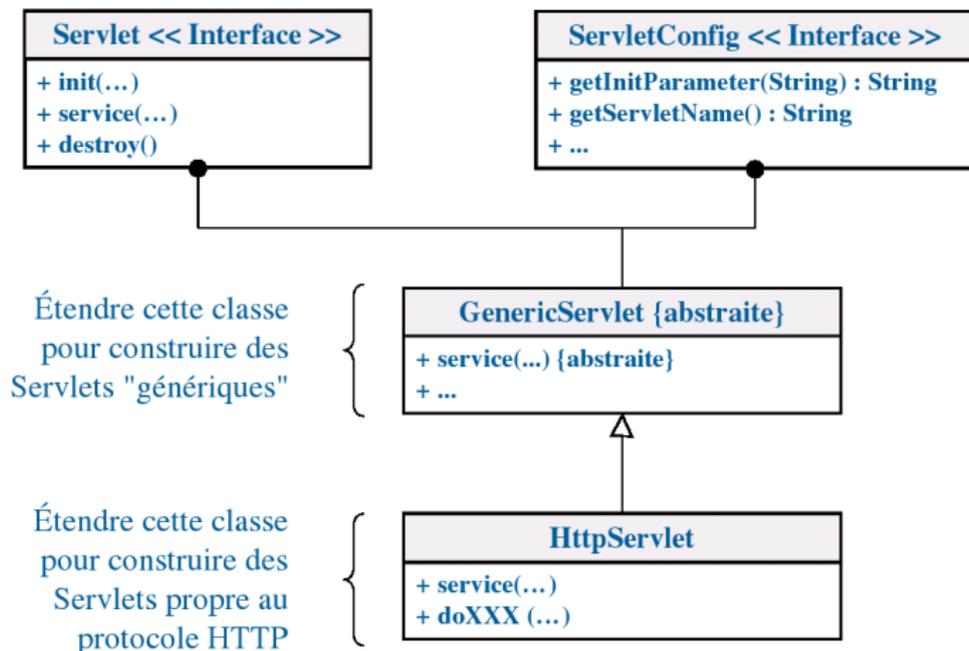
```
<?xml version='1.0' encoding='utf-8'?>
<tomcat-users>
<user username="admin" password="admin" roles="admin,manager" />
</tomcat-users>
```

A la première utilisation de tomcat, il faut modifier le fichier tomcat-users.xml

4 Fondamentaux

5 Un conteneur : Tomcat

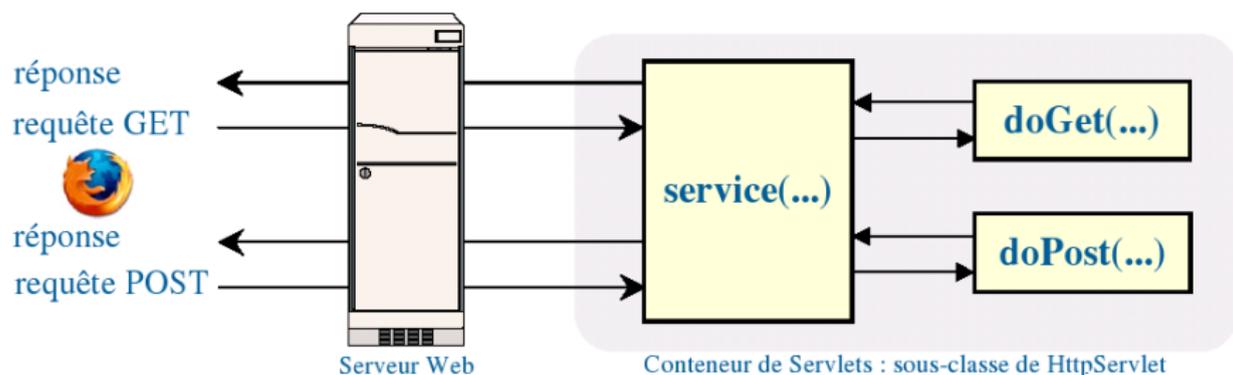
**6 Servlets**



©M. Baron (2006)

## Definition

HttpServlet La classe `HttpServlet` redéfinit la méthode `service(...)` qui interprète la requête HTTP et transmet celle-ci à la méthode appropriée (`doGet(...)`, `doPost(...)`, `doHead(...)`, ...).



©M. Baron (2006)

# Implémentation par défaut

L'implémentation par défaut retourne une erreur 405.

➤ Type de requête non supporté par l'URL

de la super-classe

```
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;
public class HelloWorld extends HttpServlet {
    public void doGet(HttpServletRequest req, HttpServletResponse res)
        throws ServletException, IOException {
        super.doGet(req, res);
    }
}
```

Apache Tomcat/5.5.4 - Rapport d'erreur - Mozilla Firefox

Etat HTTP 405 - La méthode HTTP GET n'est pas supportée par cette URL

Type: Rapport d'état

message: La méthode HTTP GET n'est pas supportée par cette URL.

description: La méthode HTTP spécifiée n'est pas autorisée pour la ressource demandée. (La méthode HTTP GET n'est pas supportée par cette URL.)

Apache Tomcat/5.5.4

Terminé

**Ne vous trompez pas de  
méthode à redéfinir selon le  
type de requête**



*Servlets - M. Baron - Page 47*

©M. Baron (2006)

# Requête et réponse HTTP

2 objets : `HttpServletRequest` et `HttpServletResponse`

Objet de requête

Objet de réponse

```
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;
public class HelloWorld extends HttpServlet {
    public void doGet(HttpServletRequest req, HttpServletResponse res)
        throws ServletException, IOException {
        ...
    }

    public void doPost(HttpServletRequest req, HttpServletResponse res)
        throws ServletException, IOException {
        ...
    }
}
```

©M. Baron (2006)

## Principales méthodes :

- **String getMethod()** : retourne le type de requête ;
- **String getServerName()** : retourne le nom du serveur ;
- **String getParameter(String name)** : retourne la valeur d'un paramètre ;
- **String[] getParameterNames()** : retourne le nom de tous les paramètres ;
- **String getRemoteHost()** : retourne l'IP du client ;
- **String getServerPort()** : retourne le port sur lequel le serveur écoute
- **String getQueryString()** : retourne la chaîne d'interrogation ;
- ...

## Principales méthodes :

- `void setStatus(int)` : définit le code de retour de la réponse ;
- `void setContentType(String)` : définit le type de contenu MIME ;
- `ServletOutputStream getOutputStream()` : flot pour envoyer des données binaires au client ;
- `void sendRedirect(String)` : redirige le navigateur vers l'URL
- ...

## Exercice

### Ecrire

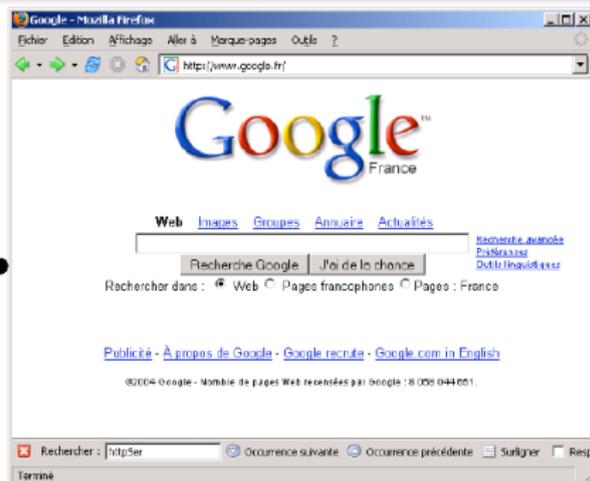
- 1 une Servlet **HelloWord** qui retourne un message de type TEXT ;
- 2 une Servlet **Redirection** qui effectue une redirection vers `miage.u-bordeaux.fr` ;
- 3 une Servlet **InfosServlet** qui retourne des informations extraites de la requête HTTP utilisée.

```
public class HelloWorld extends HttpServlet {  
  
    public void doGet(HttpServletRequest req, HttpServletResponse res)  
        throws ServletException, IOException {  
        res.setContentType("text/plain");  
  
        PrintWriter out = res.getWriter();  
  
        out.println("Premier Message");  
        out.println("Coucou voilà comment écrire un message");  
        out.println("Second Message");  
    }  
}
```



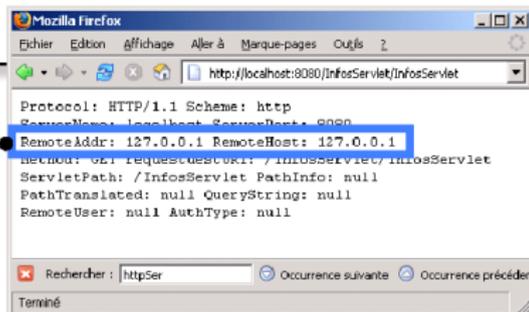
# Redirection

```
public class InfosServlet extends HttpServlet {  
    public void doGet(HttpServletRequest req, HttpServletResponse res)  
        throws ServletException, IOException {  
        res.sendRedirect("http://www.google.fr");  
    }  
}
```



©M. Baron (2006)

```
public class InfosServlet extends HttpServlet {  
    public void doGet(HttpServletRequest req, HttpServletResponse res)  
        throws ServletException, IOException {  
        response.setContentType("text/plain");  
        PrintWriter out= response.getWriter();  
        out.println("Protocol: " + request.getProtocol());  
        out.println("Scheme: " + request.getScheme());  
        out.println("ServerName: " + request.getServerName());  
        out.println("ServerPort: " + request.getServerPort());  
        out.println("RemoteAddr: " + request.getRemoteAddr());  
        out.println("RemoteHost: " + request.getRemoteHost());  
        out.println("Method: " + request.getMethod());  
        ...  
    }  
}
```



*Servlets - M. Baron - Page*

©M. Baron (2006)

# Récupération de valeurs passées par un formulaire

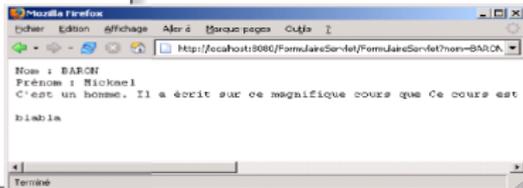
- **String** `getParameter(String p)` : retourne la valeur du paramètre `p` ;
- **String[]** `getParameterValues(String p)` : retourne les valeurs du paramètre `p`.

```
public class FormulaireServlet extends HttpServlet {
    protected void doGet(HttpServletRequest req, HttpServletResponse res)
        throws ServletException, IOException {
        res.setContentType("text/plain");
        PrintWriter out = res.getWriter();
        out.println("Nom : " + req.getParameter("nom"));
        out.println("Prénom : " + req.getParameter("prenom"));

        if (req.getParameterValues("radio1")[0].equals("mas"))
            out.print("C'est un homme. Il");
        else {
            out.print("C'est une femme. Elle");
        }

        out.print(" a écrit sur ce magnifique cours que ");
        out.println(req.getParameter("textarea"));
    }
}
```

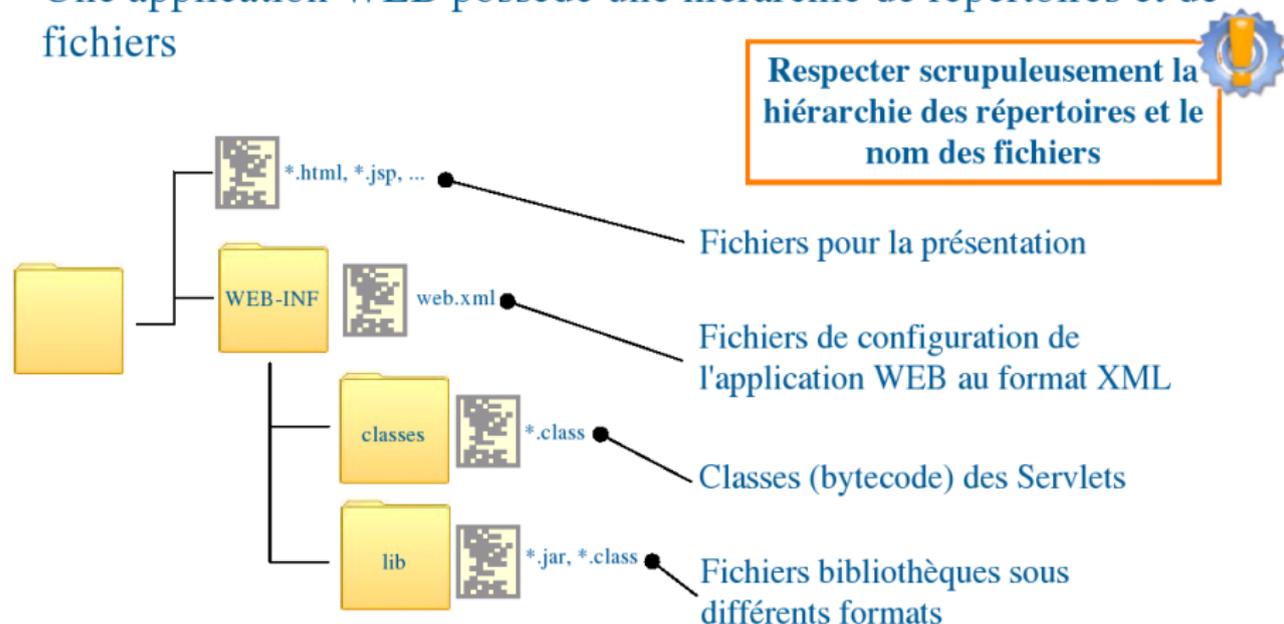
Cette méthode est utile lorsque dans un formulaire vous avez plusieurs composants qui portent le même nom



©M. Baron (2006)

# Architecture d'une application WEB

- Une application WEB possède une hiérarchie de répertoires et de fichiers



©M. Baron (2006)

# Déploiement d'une Servlet

## Avec Eclipse :

- 1 télécharger/installer le plugin Tomcat  
<http://www.eclipse totale.com/tomcatPlugin.html>
- 2 manuel :  
<http://www.eclipse totale.com/articles/tomcat/>;
- 3 **Attention : le fichier `web.xml` n'est pas généré par le plugin !**

Exemple de `web.xml` (dans `WEB-INF` du projet) \_\_\_\_\_

```
<?xml version="1.0" encoding="utf8"?>
<web-app xmlns="http://java.sun.com/xml/ns/javaee" xmlns:xsi="http://
  www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://java.sun.com/xml/ns/javaee http://java.sun.
    com/xml/ns/javaee/web-app_2_5.xsd"
  version="2.5">
  <display-name> Tests </display-name>
  <servlet><servlet-name>test</servlet-name>
  <servlet-class>MaServlet</servlet-class>
  </servlet>
  <servlet-mapping><servlet-name>test</servlet-name>
  <url-pattern>/*</url-pattern>
</servlet-mapping></web-app>
```

Le cours de Mickaël Baron aborde des notions plus subtiles :

- cycles de vie des Servlets ;
- sessions : cookies, HttpSession ;
- partage d'informations entre Servlets ;
- partager le contrôle d'une requête Http ;
- authentification ;
- etc .