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

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"

Plan

URL

Uniform Resource Locator

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

- 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 ;
- ...

accès à : http ://www.u-bordeaux1.fr

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

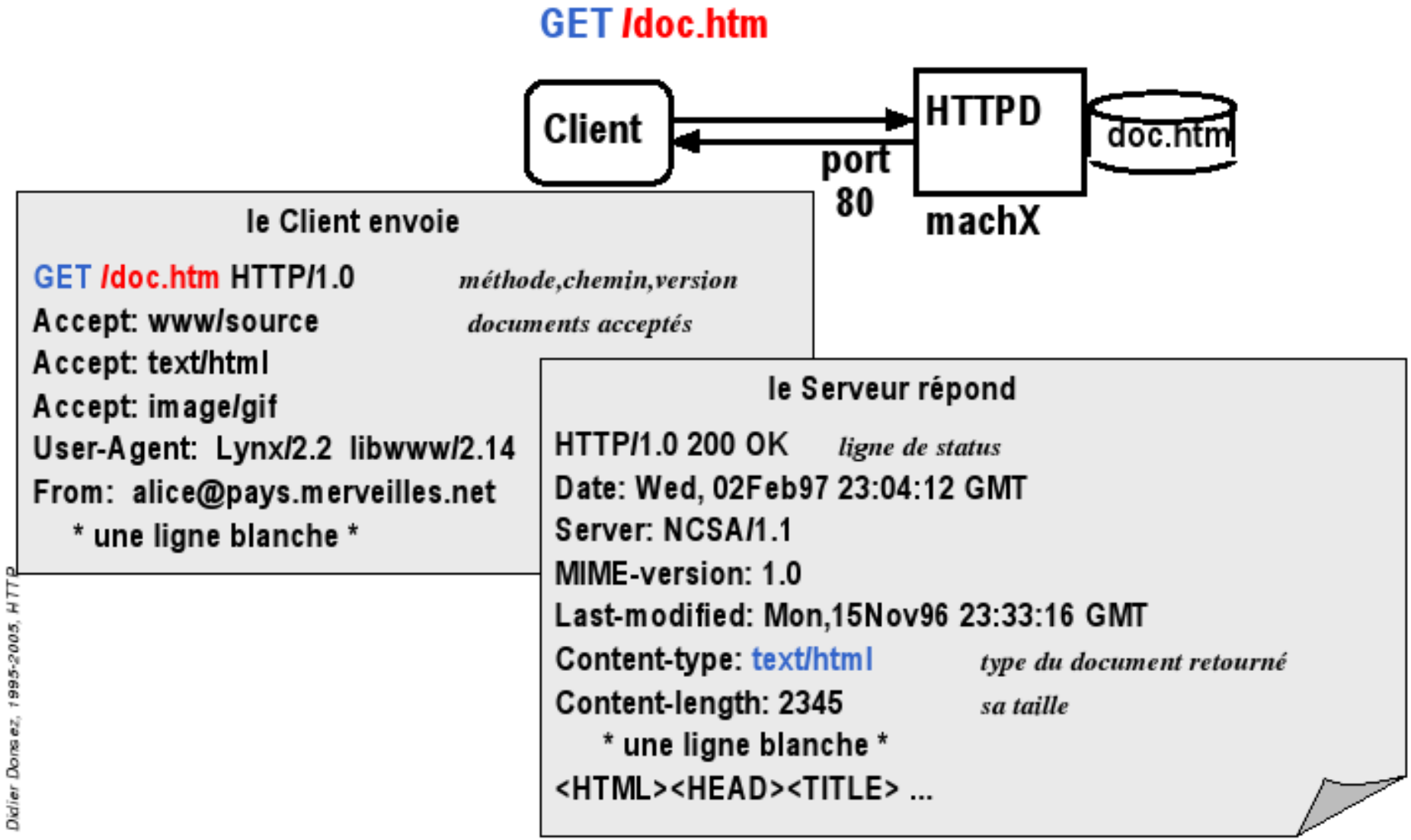
1. le navigateur interroge un serveur DNS pour connaitre l'adresse IP du serveur www.u-bordeaux1.fr ;
2. le serveur DNS répond x.x.x.x ;
3. le navigateur initie une connexion TCP sur le port HTTP (80) de la machine x.x.x.x ;
4. il envoie sur la connexion une requête via le protocole HTTP demandant le fichier par défaut : index.html ;
5. le serveur www.u-bordeaux1.fr 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.

Plan

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



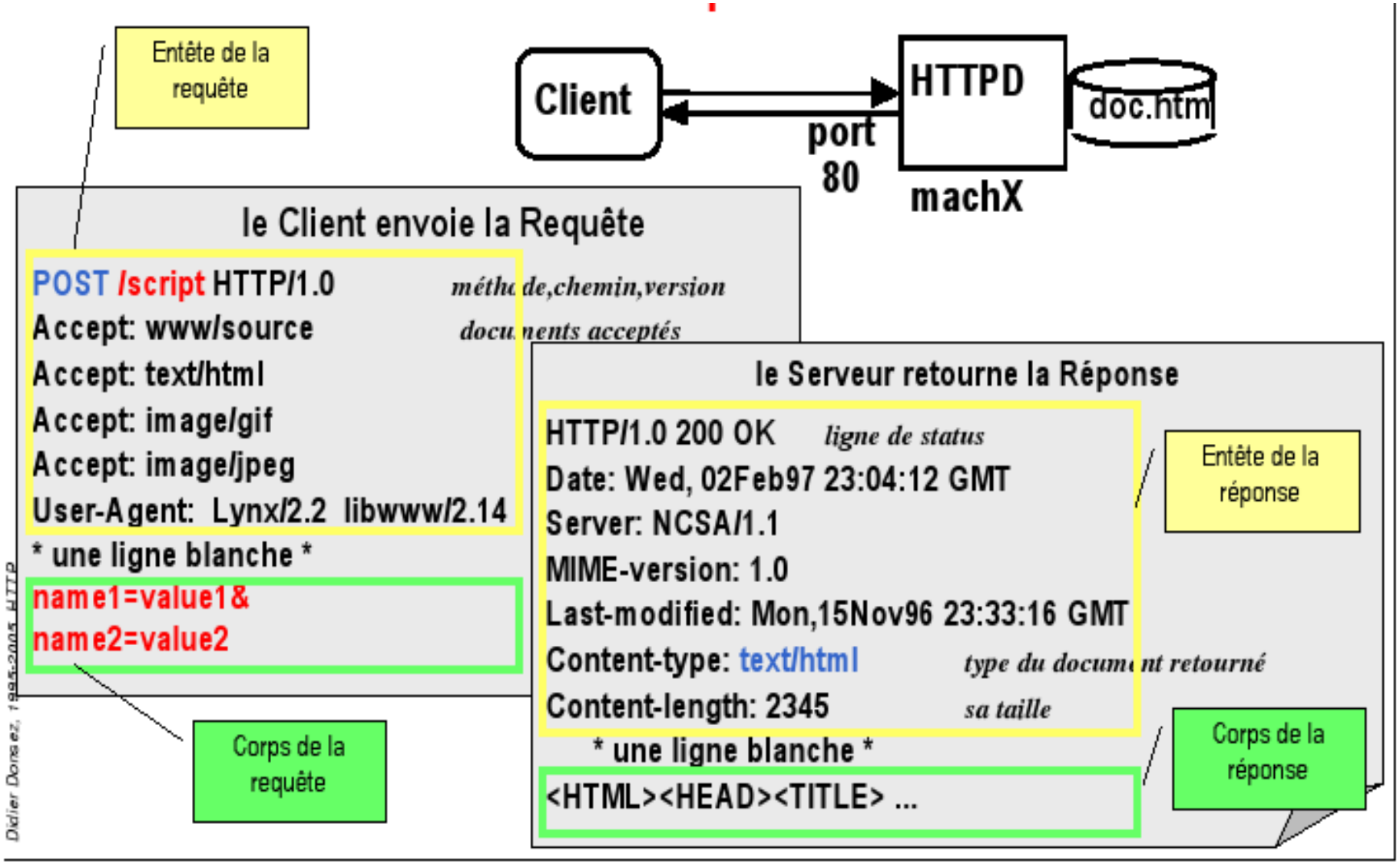
Dialogue HTTP

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.

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>
```

Exercice : un super serveur web

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 caractètes 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 :

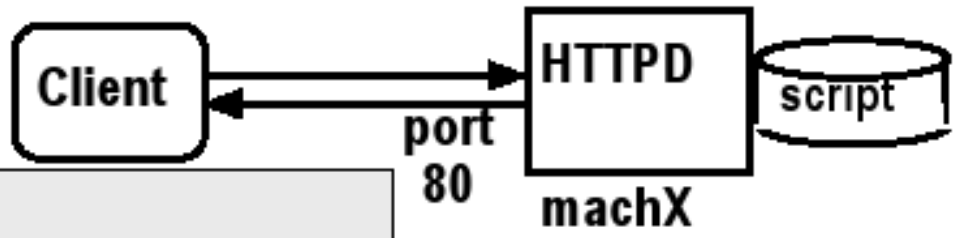
- 1xx : Information ;
- 2xx : Succès - exemple : 200 = la requête a réussi ;
- 3xx : Redirection - exemples : 301 = page déplacée, 304 = page en cache toujours valide ;
- 4xx : Erreur client - exemples : 403 = page interdite, 404 = page inexistante ;
- 5xx : Erreur serveur - exemples : 500 = erreur interne,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

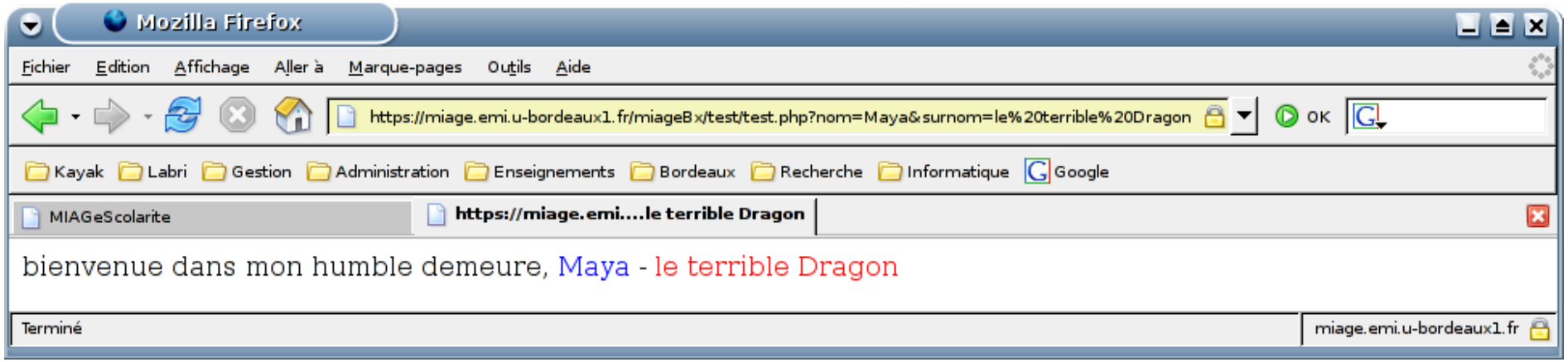
le Serveur répond

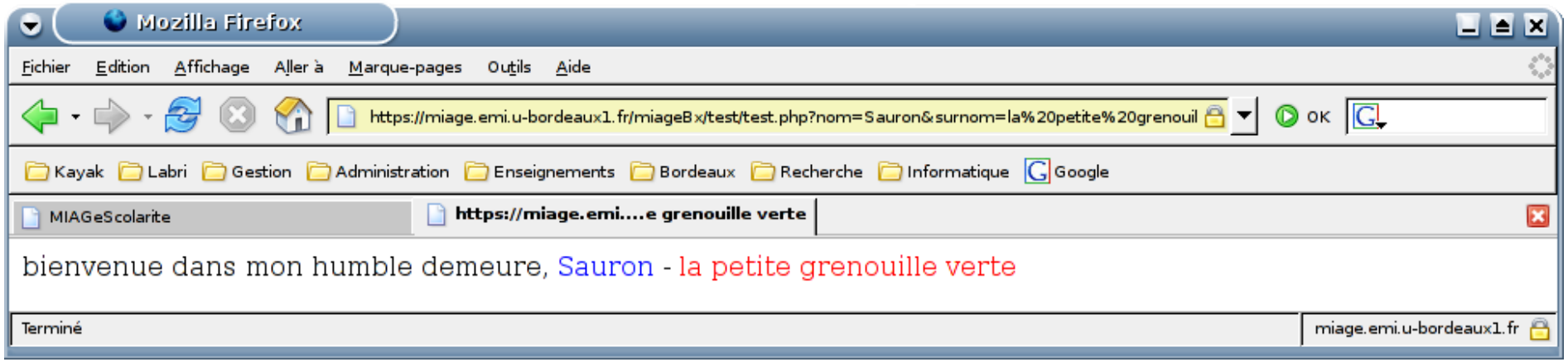
HTTP/1.0 200 OK *ligne de status*
Date: Wed, 02Feb97 23:04:12 GMT
Server: NCSA/1.1
MIME-version: 1.0
Last-modified: Mon,15Nov96 23:33:16 GMT
Content-type: text/html *type du document retourné*
Content-length: 2345 *sa taille*
* une ligne blanche *
<HTML><HEAD><TITLE> ...

Didier Donsez, 1995-2005, HTTP

©D. Donsez

GET : passage de valeurs dans l'URL

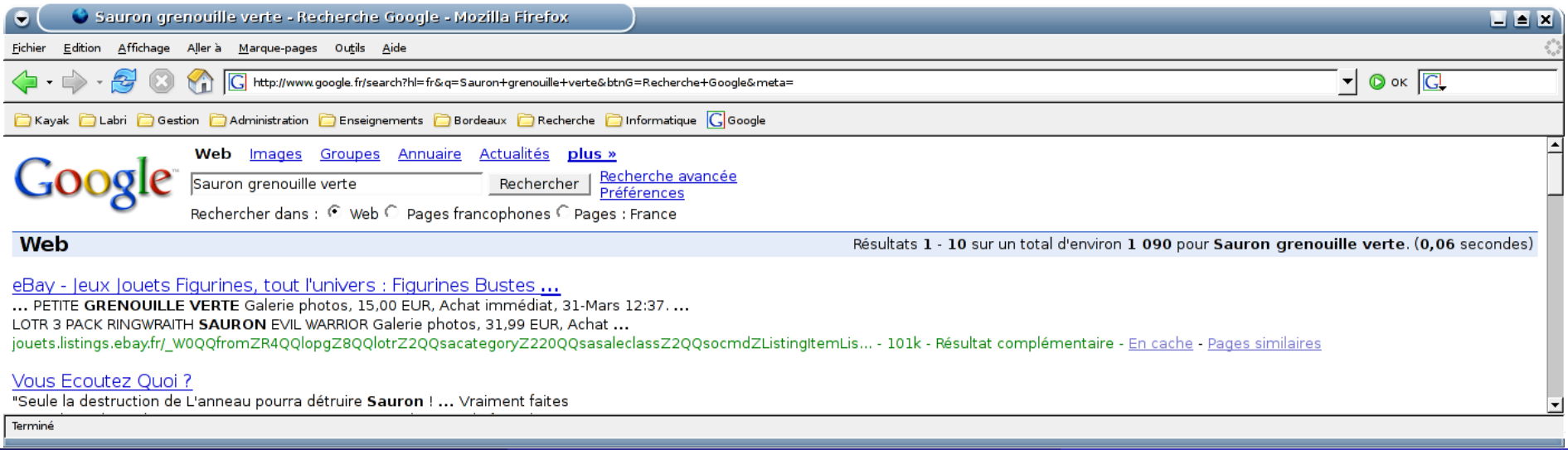




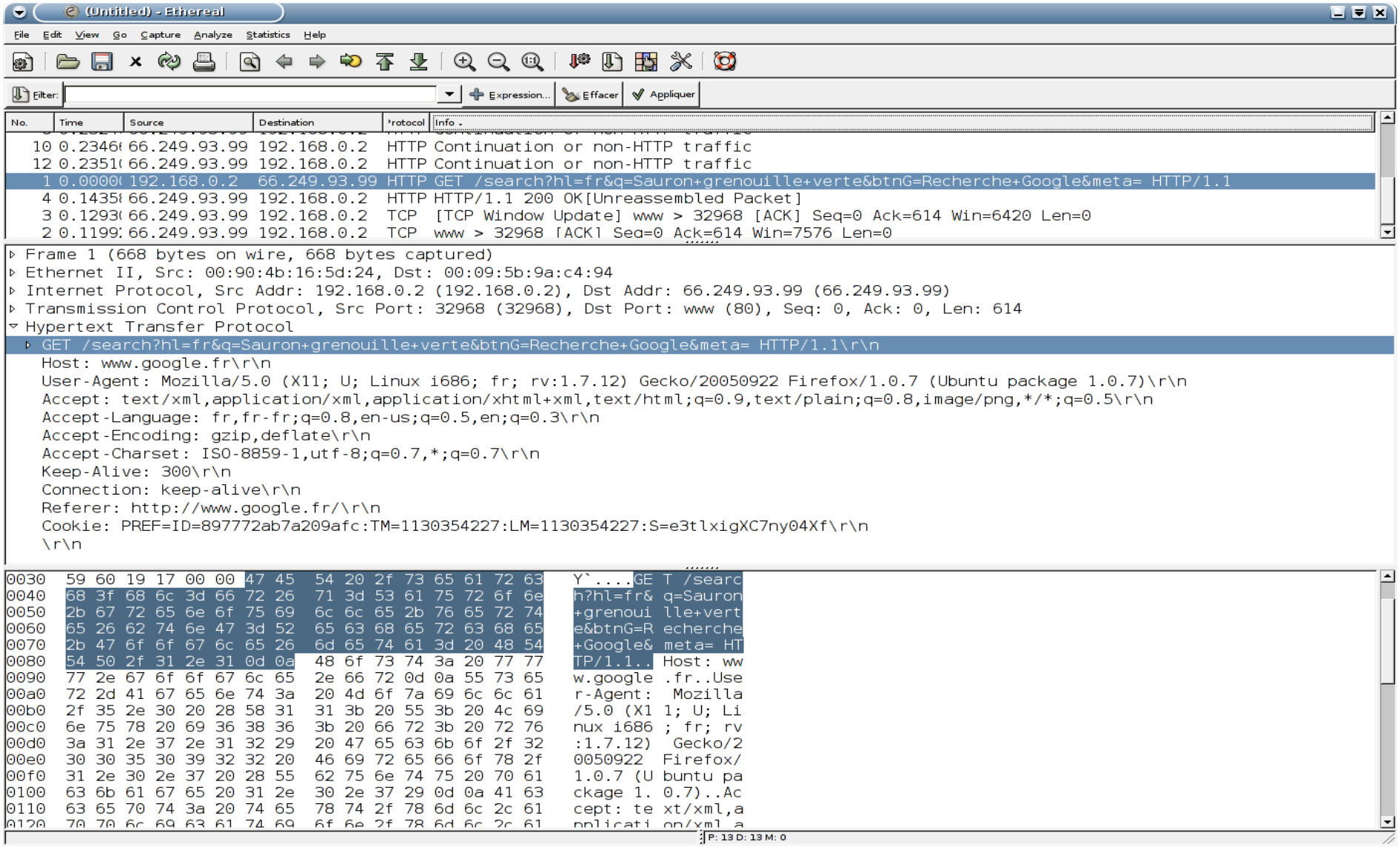
Formulaire envoyé avec GET

Un petit coup de google :

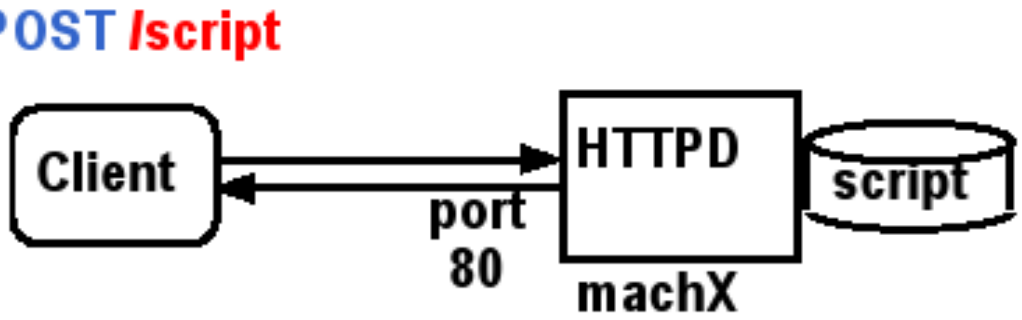
Dans le résultat, noter l'URL



Les valeurs ont transité en clair



Envoi de données - POST



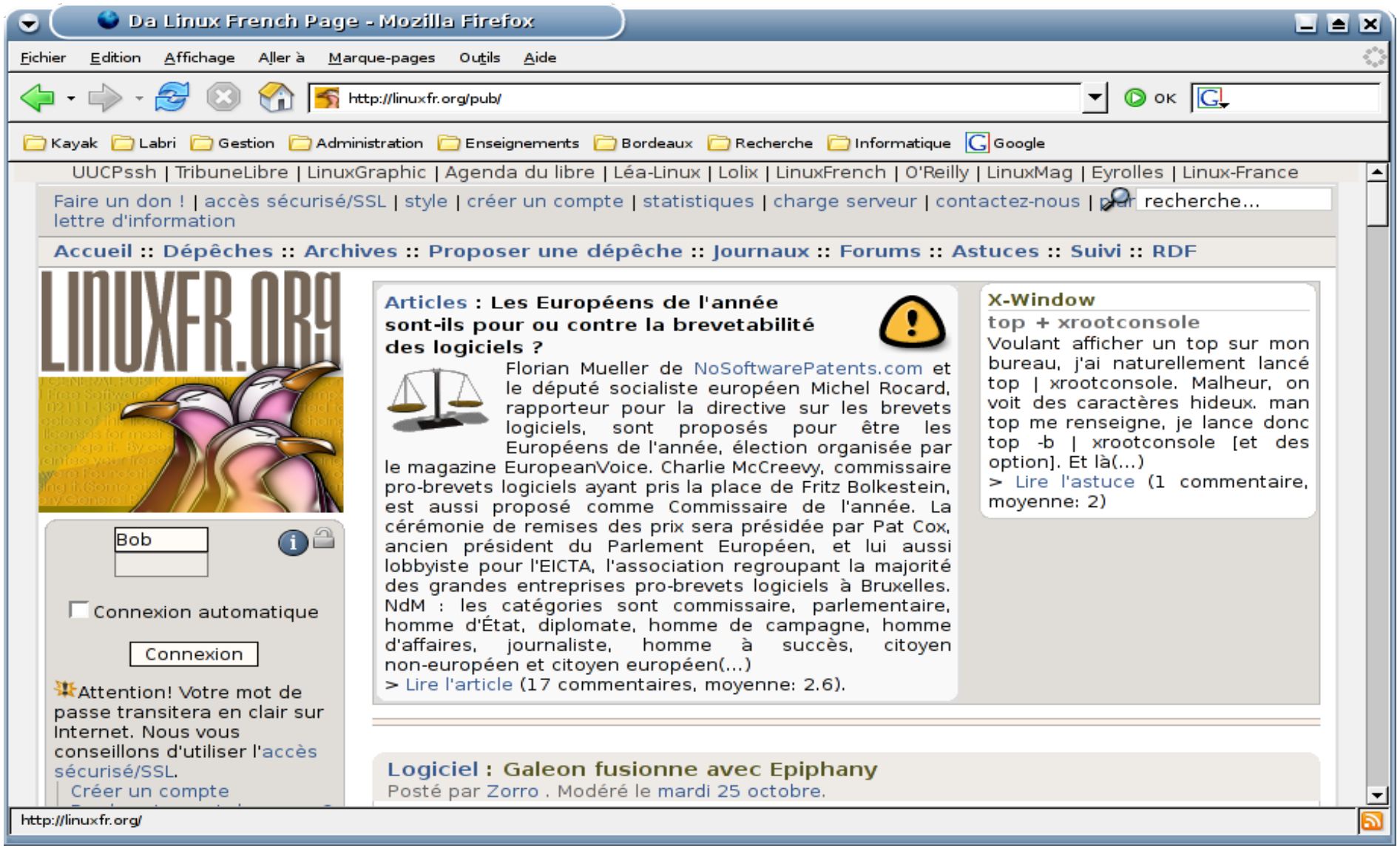
le Client envoie

```
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
```

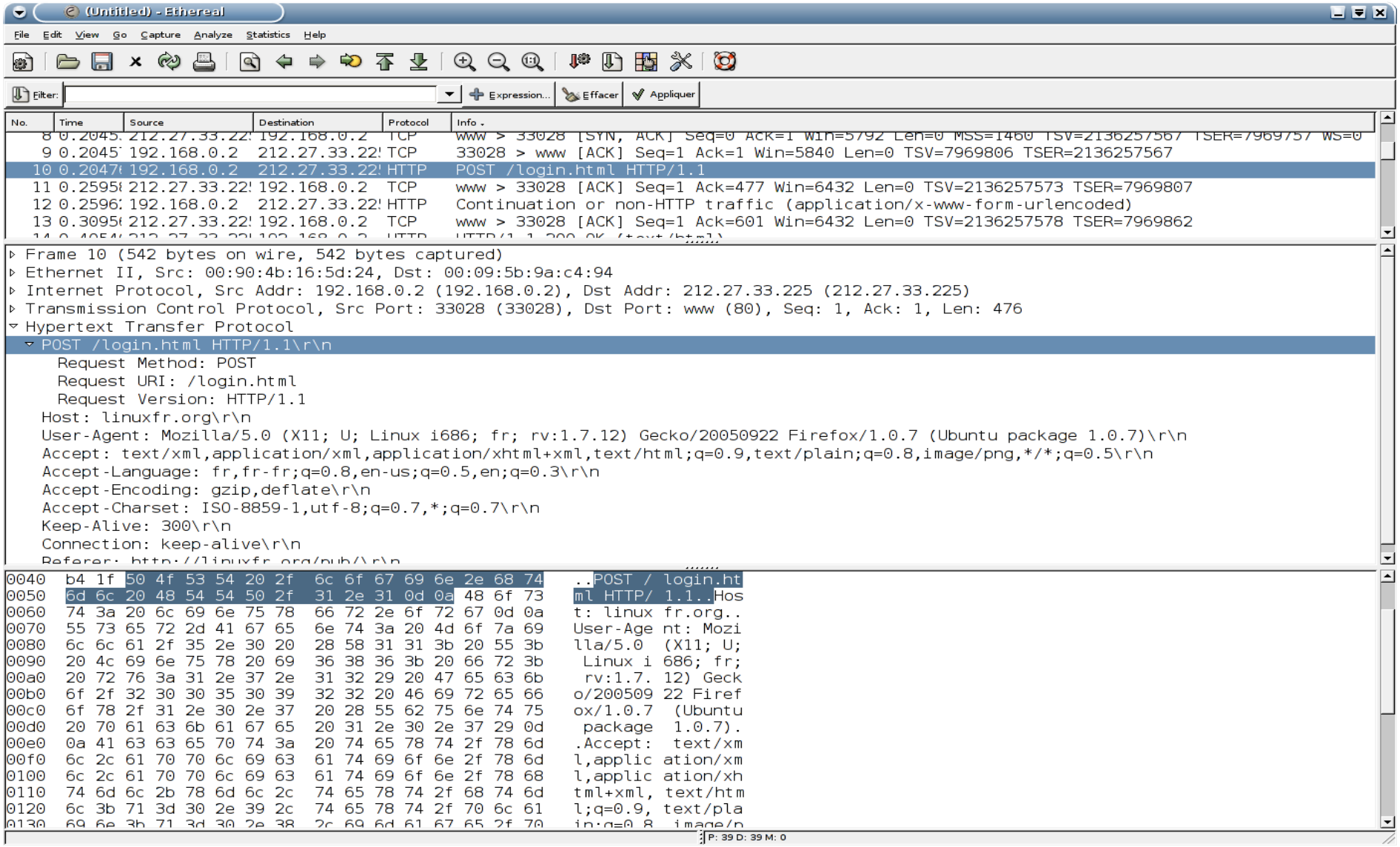
le Serveur répond

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

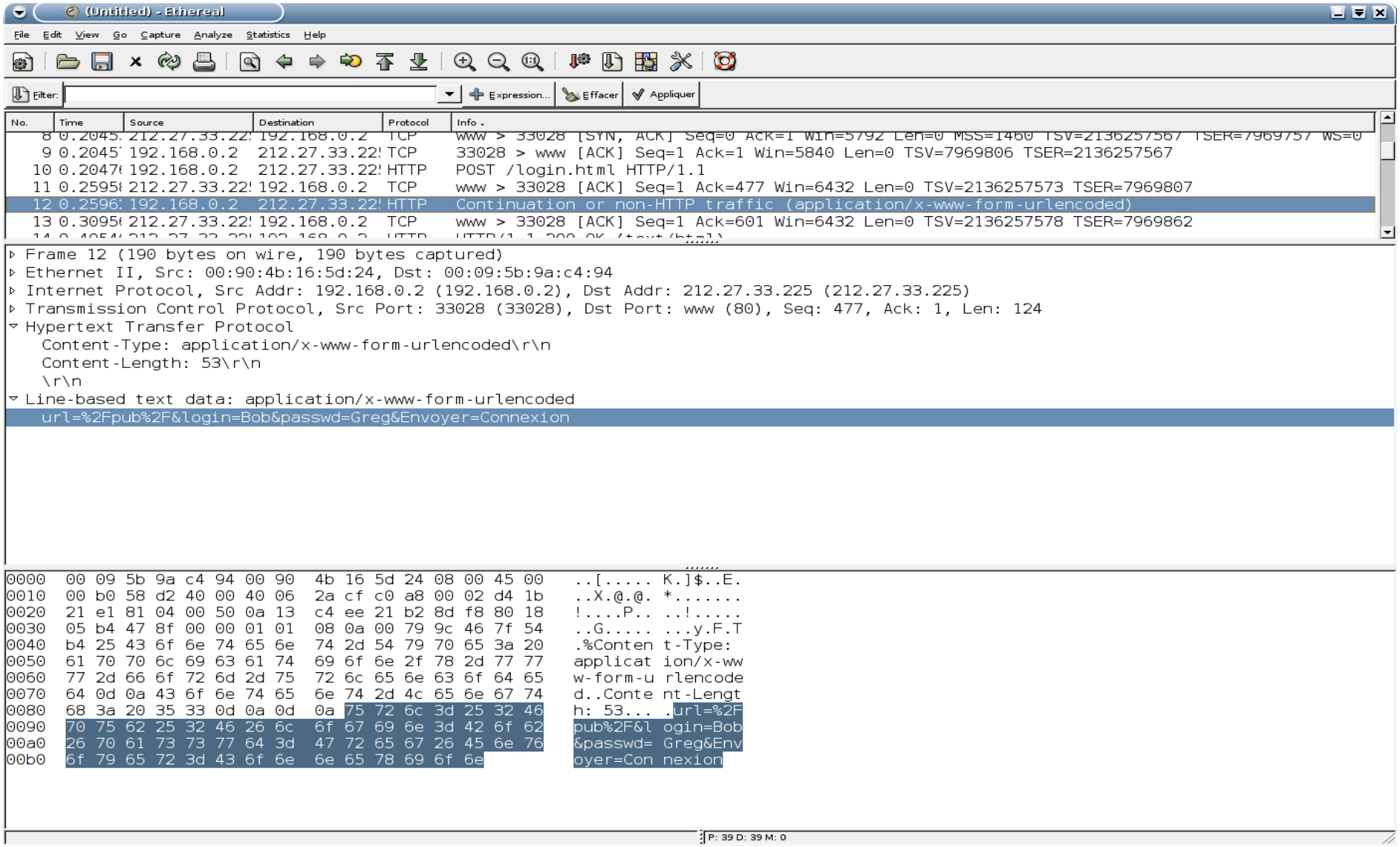
Formulaire envoyé avec POST



Vérification : trame



Le mot de passe a transité en clair...



Exercices : récupération des valeurs

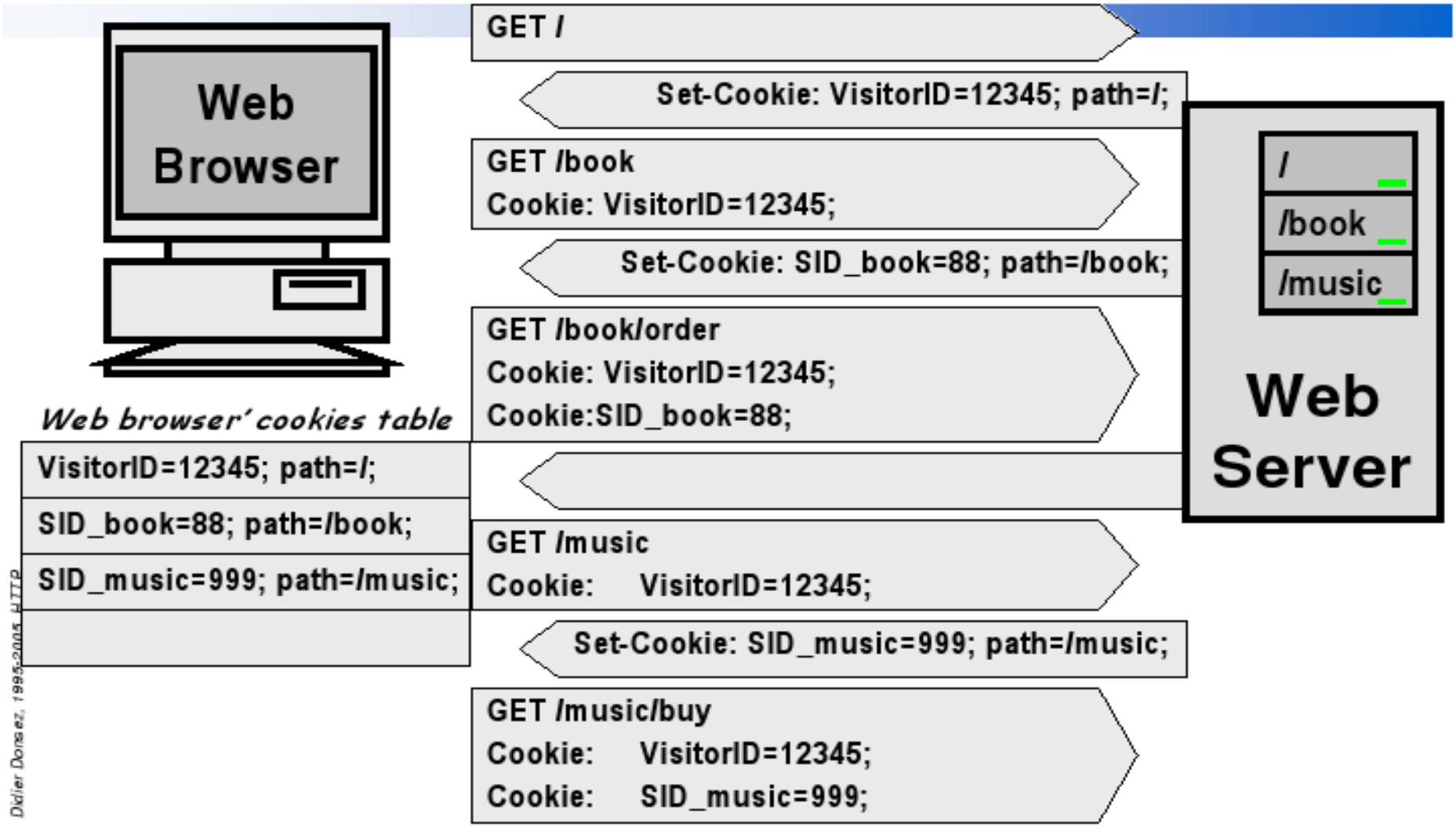
Exercice

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

Exercice

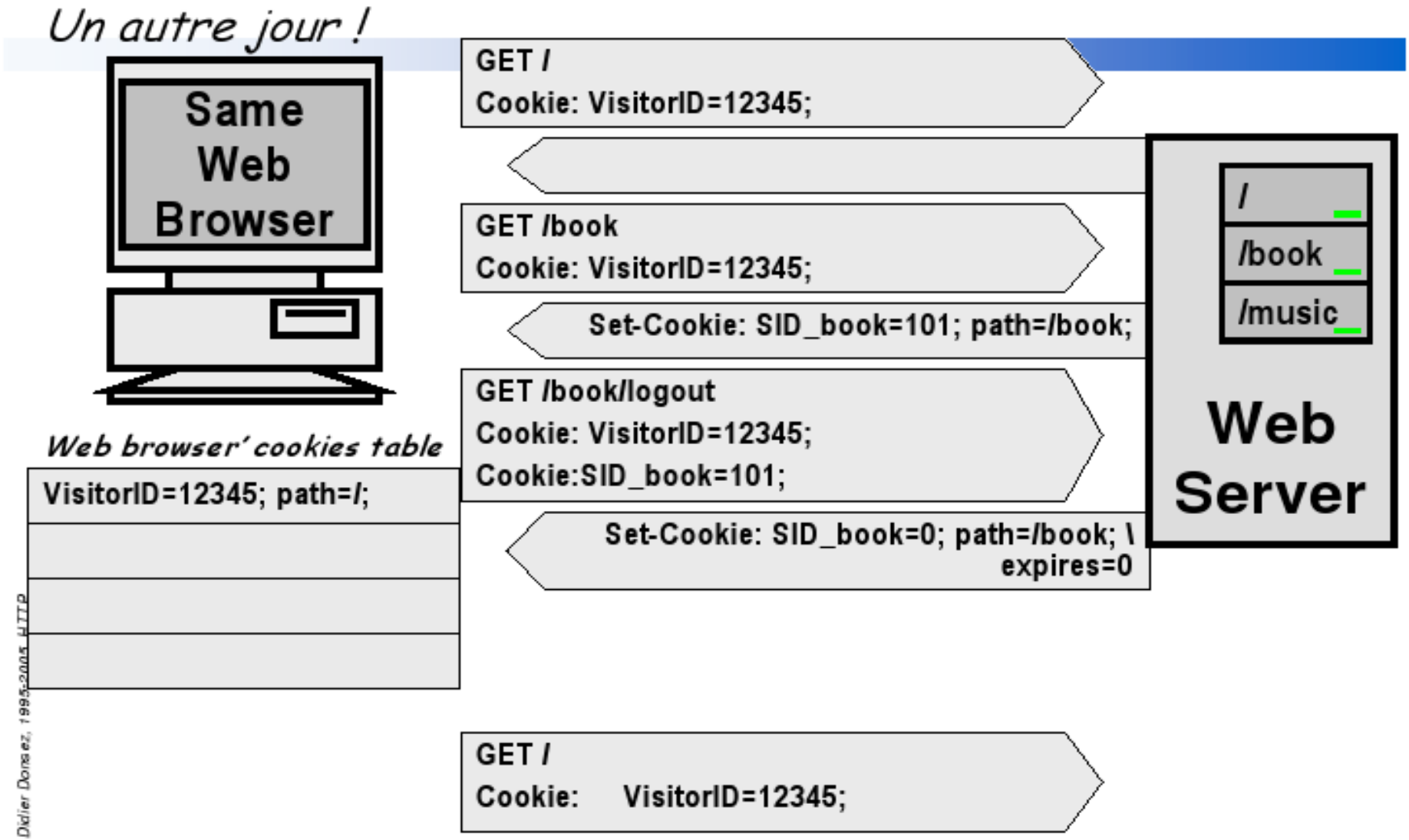
Modifier notre serveur web pour qu'il puisse récupérer les valeurs passées par le client avec la méthode POST.

Cookie (1)



©D. Donsez

Cookie (2)



©D. Donsez

Suivi de session & HTTP

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 ?

Principales méthodes

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=To1128mC3371855752157707 5At
?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

Installation sous linux Ubuntu :

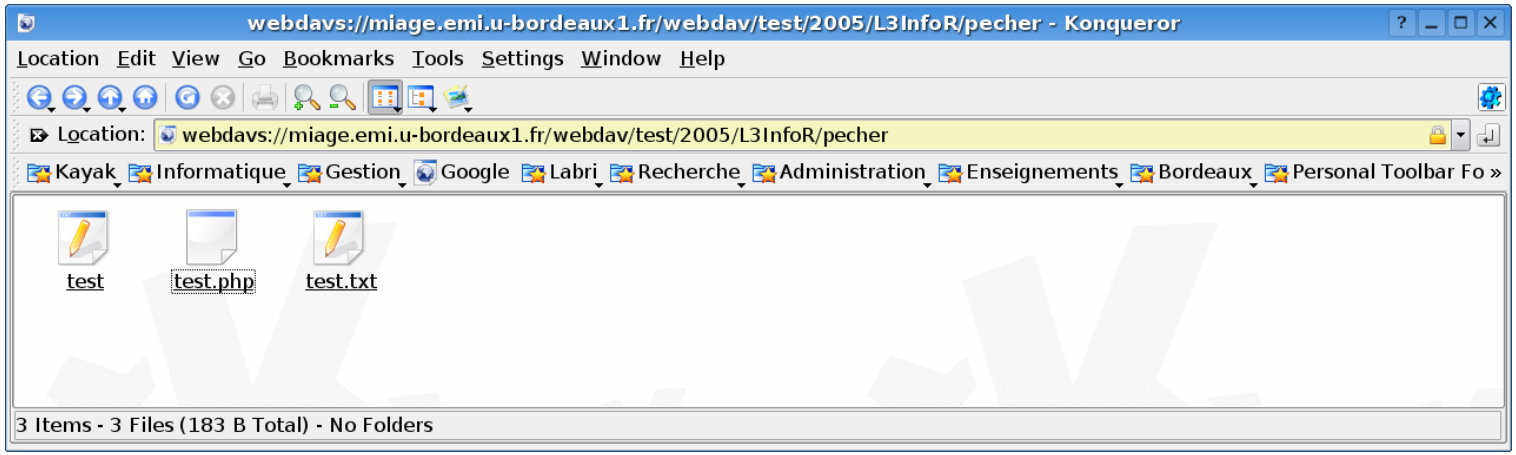
Plan

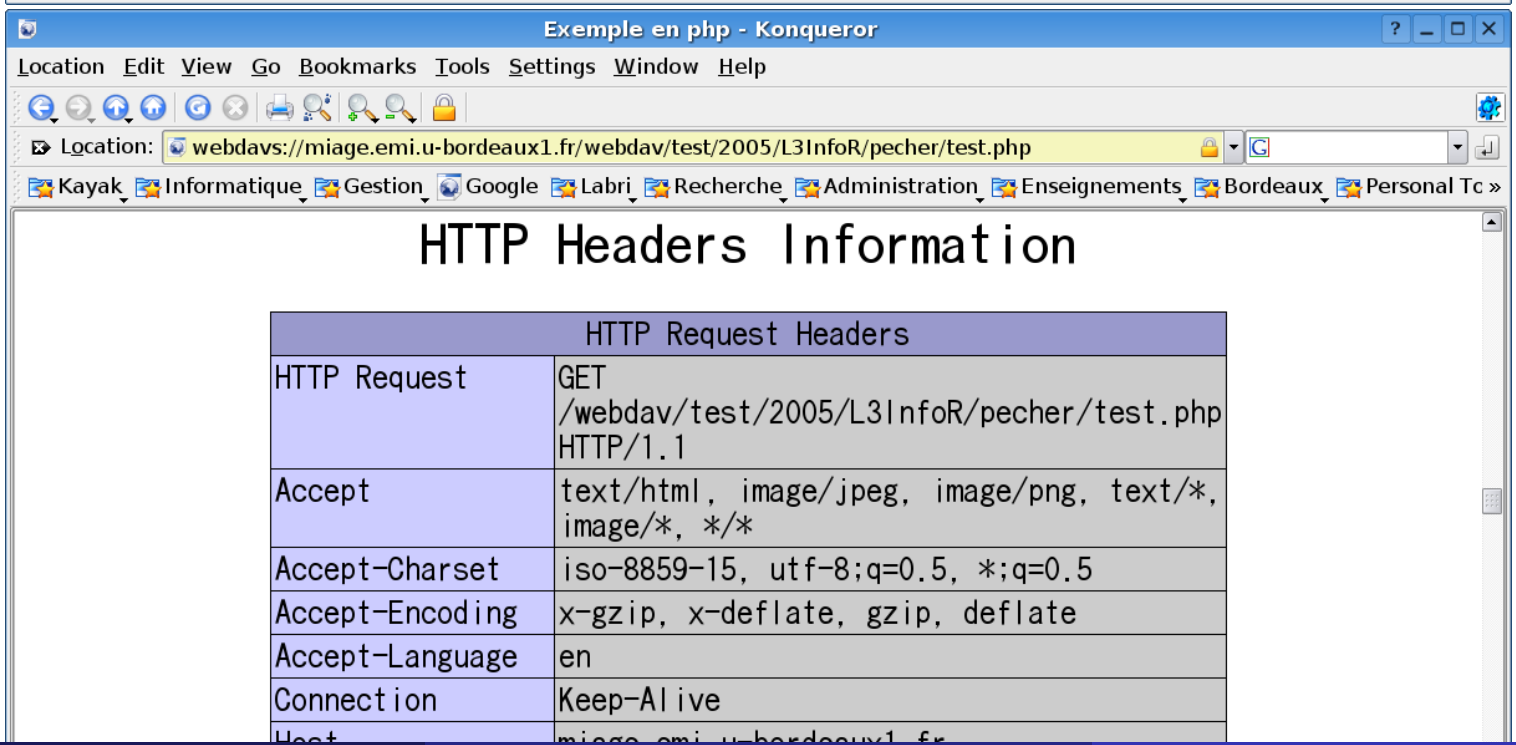
Extension de HTTP pour la mise à jour de site :

- 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 déverouille un
- serveurs WebDAV : MS IIS5, mod_dav pour Apache, support dans Tomcat 4...
- clients WebDAV : MS Office, Cadaver, konqueror...

Webdav

webdavs ://miage.emi.u-bordeaux1.fr/prod/webdav/test/2006/M2MIAGE/





Motivation

- Sécuriser (authentification, confidentialité) l'accès à un service Web
- SSL ;
  - Authentification du serveur et/ou du client par PKI ;
  - Chiffrage 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), iPlanet, MS IIS, OracleAS, IBM WebSphere ;

HTTPR

R pour Reliable (Fiable)

- 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.

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/`

Plan

Servlet

Definition

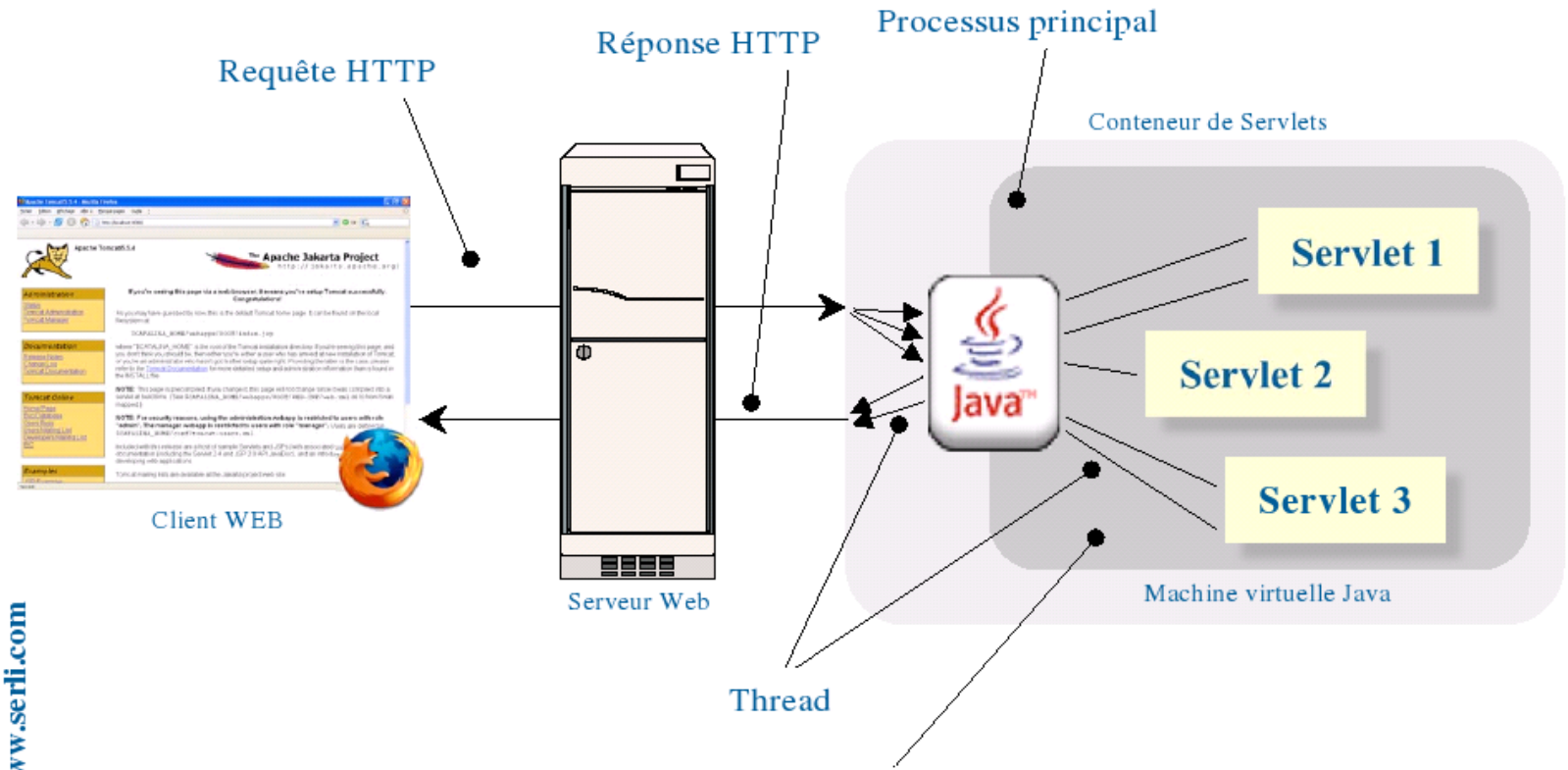
Servlet Composant logiciel écrit en Java fonctionnant côté serveur.

Concurrents directs :

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

Une Servlet s'exécute dans un conteneur de Servlets.

Architecture



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

www.serli.com

©M. Baron (2006)

Atouts

- 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.

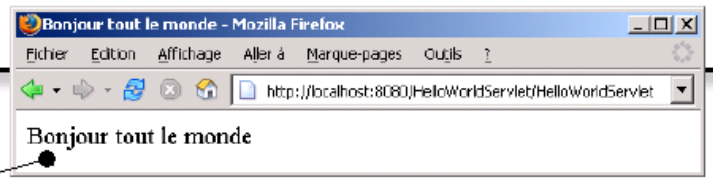
Une première Servlet

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/hrml");
        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>");
    }
}
```



Réponse sous format HTML de la Servlet HelloWorld

Le résultat sur le client

www.serli.com

serli informatique

Plan

Conteneurs de 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) ;
- ...

Jakarta Tomcat

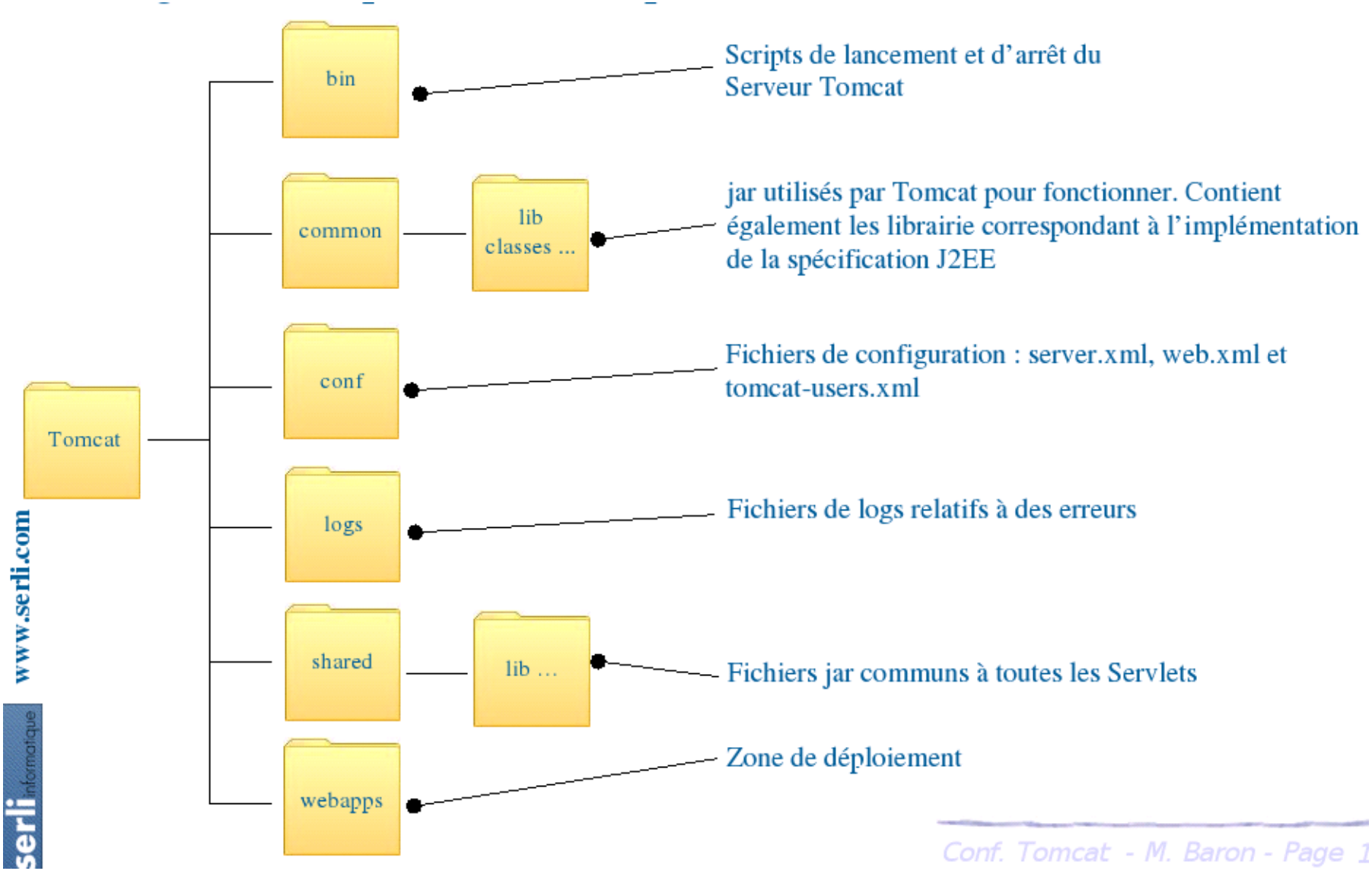
- Tomcat 5 respecte la spécification Servlet 2.4 et JSP 2.0 ;
- Ecrit entièrement en Java, il peut donc être utilisé sur n'importe quel système disposant d'une machine virtuelle ;
- Disponible gratuitement sous forme d'une licence Open Source ;
- Implémentation de référence de la spécification J2EE.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

Installation de tomcat sous linux

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 faà§on 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



Rôles

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

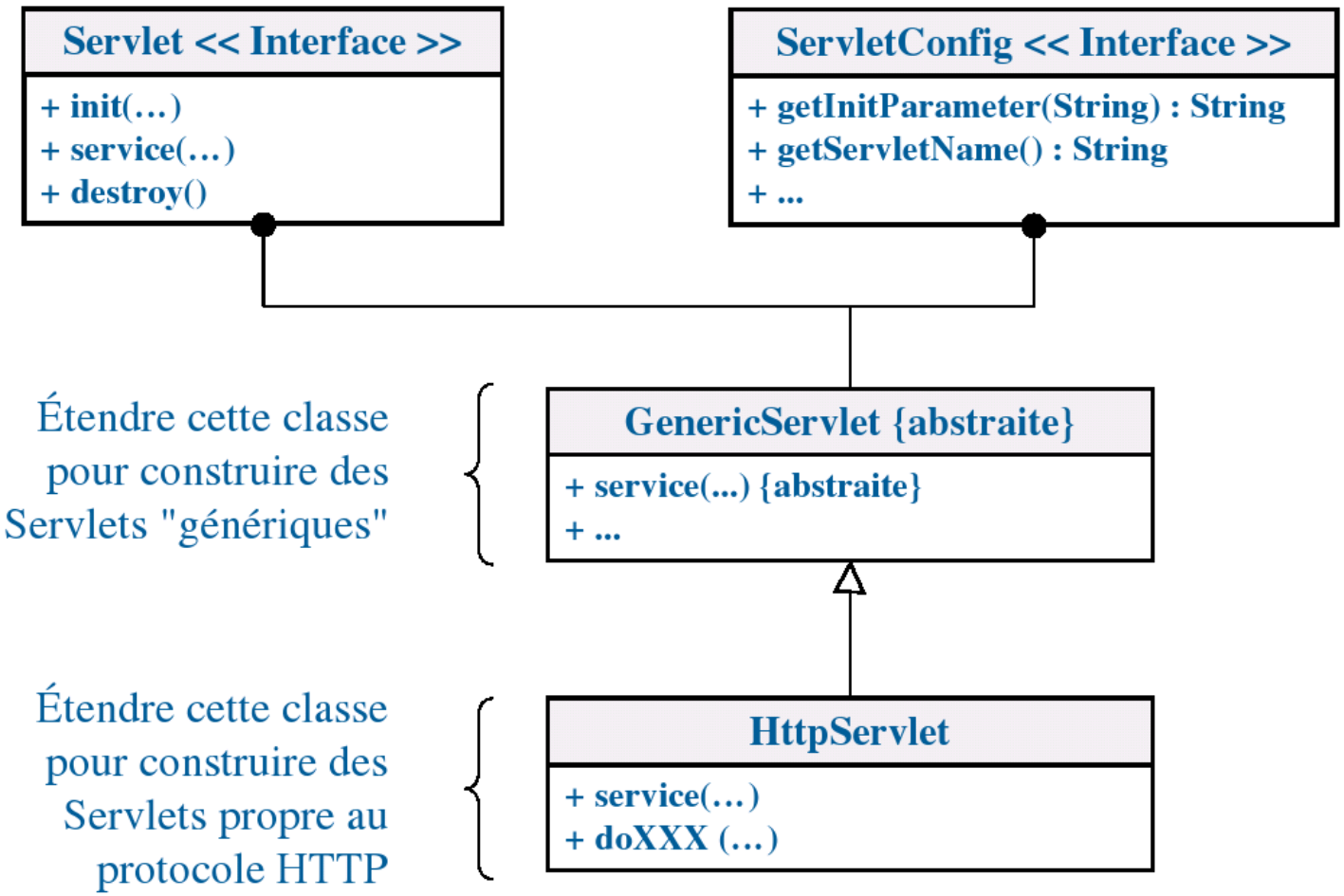
tomcat-users.xml

```
<?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`

Plan

API

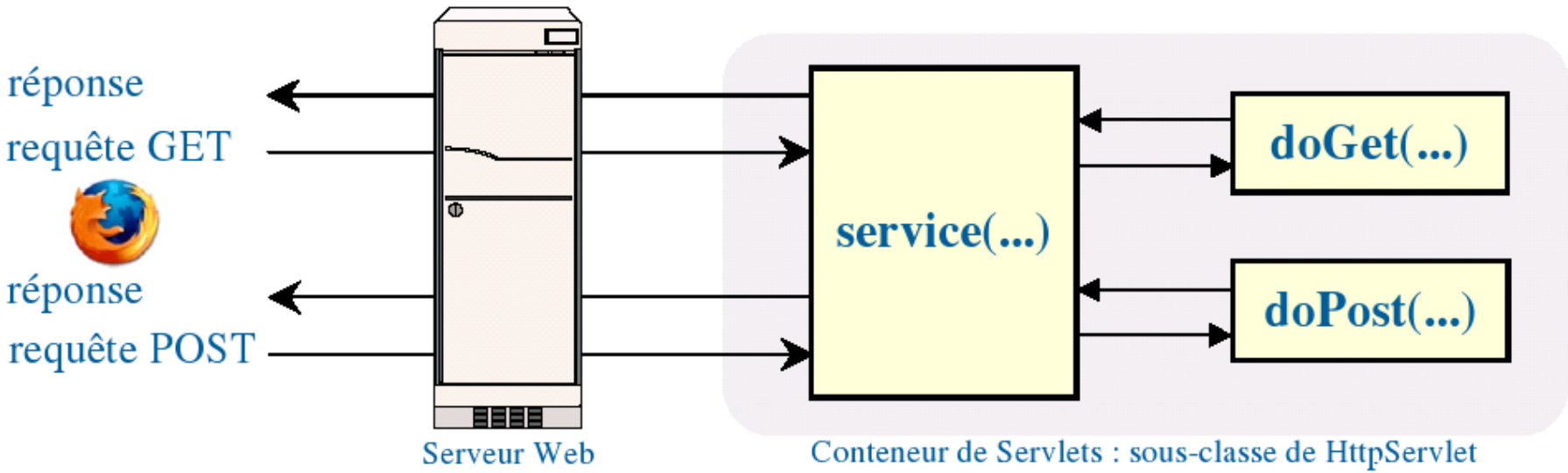


©M. Baron (2006)

HttpServlet

Definition

HttpServlet La classe `HttpServlet` redéfinit la méthide `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

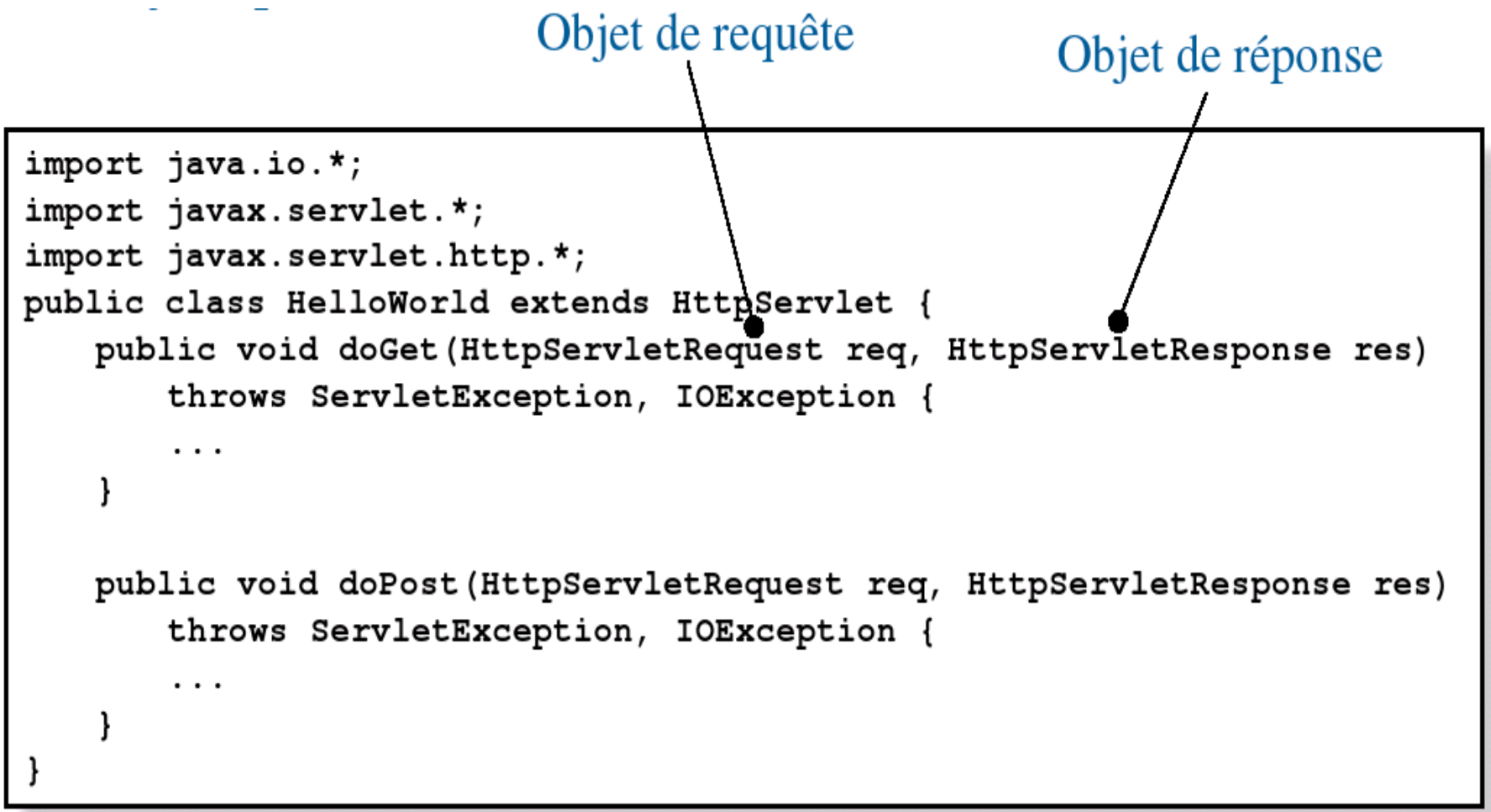
de la super-classe

```
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);
    }
```

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

Requête et réponse HTTP

2 objets : `HttpServletRequest` et `HttpServletResponse`



©M. Baron (2006)

HttpServletRequest

Principales méthodes :

- String getMethod() : retourne le type de requête ;
- String getServeurName() : 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 ;
- ...

HttpServletResponse

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
- ...

Pratique

Exercice

Ecrire

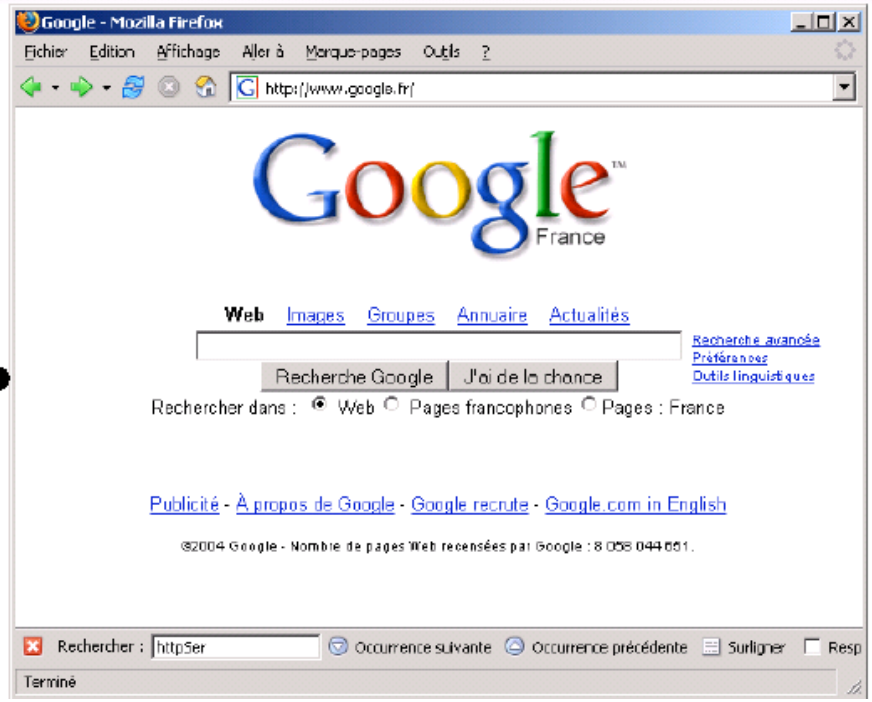
1. une Servlet HelloWord qui retourne un message de type TEXT ;
2. une Servlet Redirection qui effectue une redirection vers `miage.u-bordeaux.fr` ;
3. une Servlet InfosServlet qui retourne des informations extraites de la requête HTTP utilisée.

HelloWorld

```
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

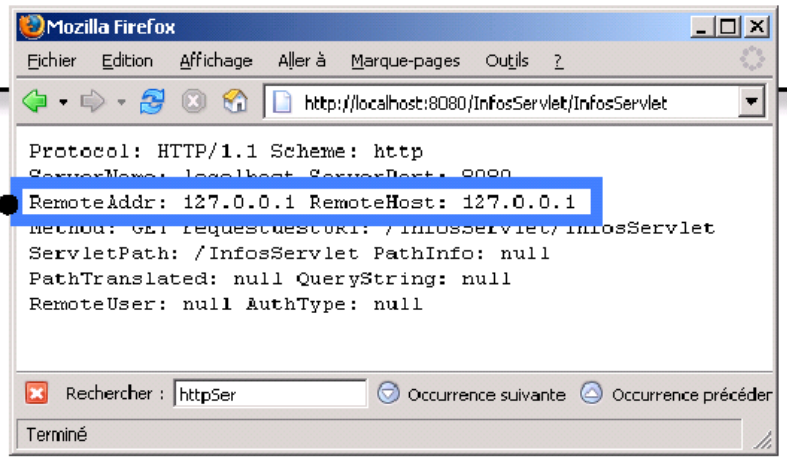
```
public class InfosServlet extends HttpServlet {
    public void doGet(HttpServletRequest req, HttpServletResponse res)
        throws ServletException, IOException {
        res.sendRedirect("http://www.google.fr");
}}
```



©M. Baron (2006)

InfosServlet

```
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());
        ...
}}
```

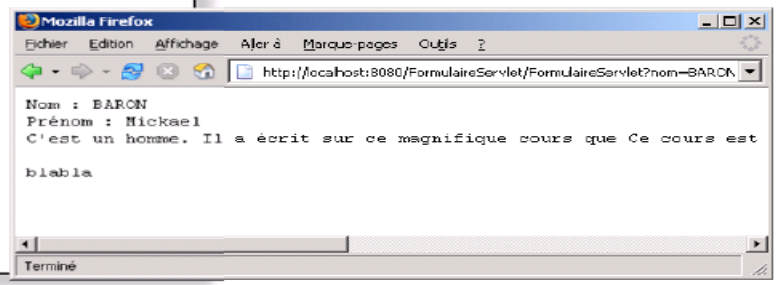


Récupération de valeurs passées par un formulaire

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

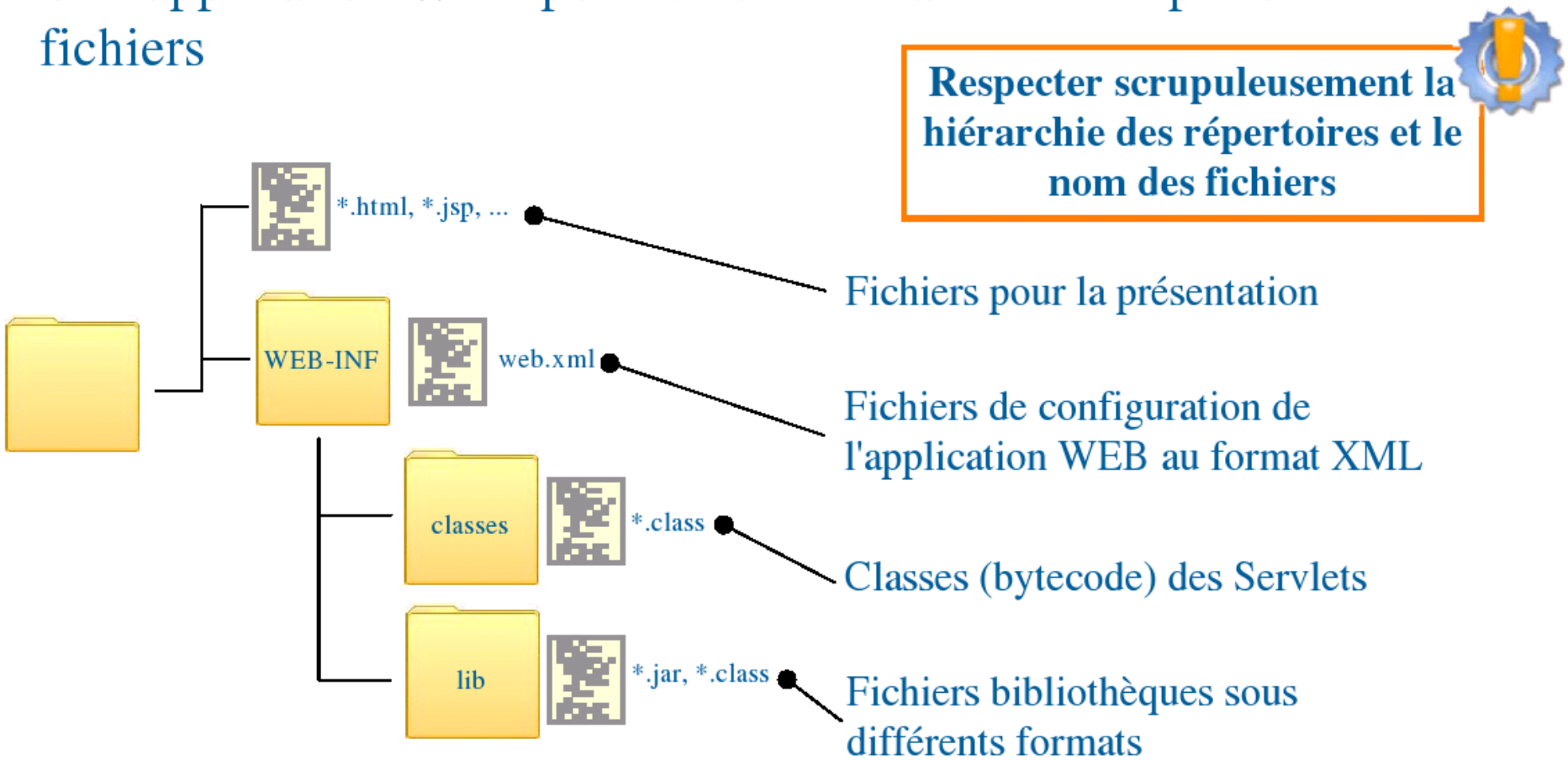
```
public class FormulaireServlet extends HttpServlet {
     protected void doGet(HttpServletRequest req, HttpServletResponse res)
             throws ServletException, IOException {
          res.setContentType("text/plain");
          PrintWriter out = res.getWriter();
          out.println("Nom :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



Architecture d'une application WEB

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



Déploiement d'une Servlet

Avec Eclipse :

1. télécharger/installer le plugin Tomcat
   `http ://www.eclipsetotale.com/tomcatPlugin.html`
2. manuel :
   `http ://www.eclipsetotale.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>
```

Pour aller plus loin

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 .