CSI 3540

Structures, techniques et normes du Web

Programmation côté serveur : Servlets

Objectif:

- Introduction à la programmation côté serveur
- Introduction aux **Servlets**

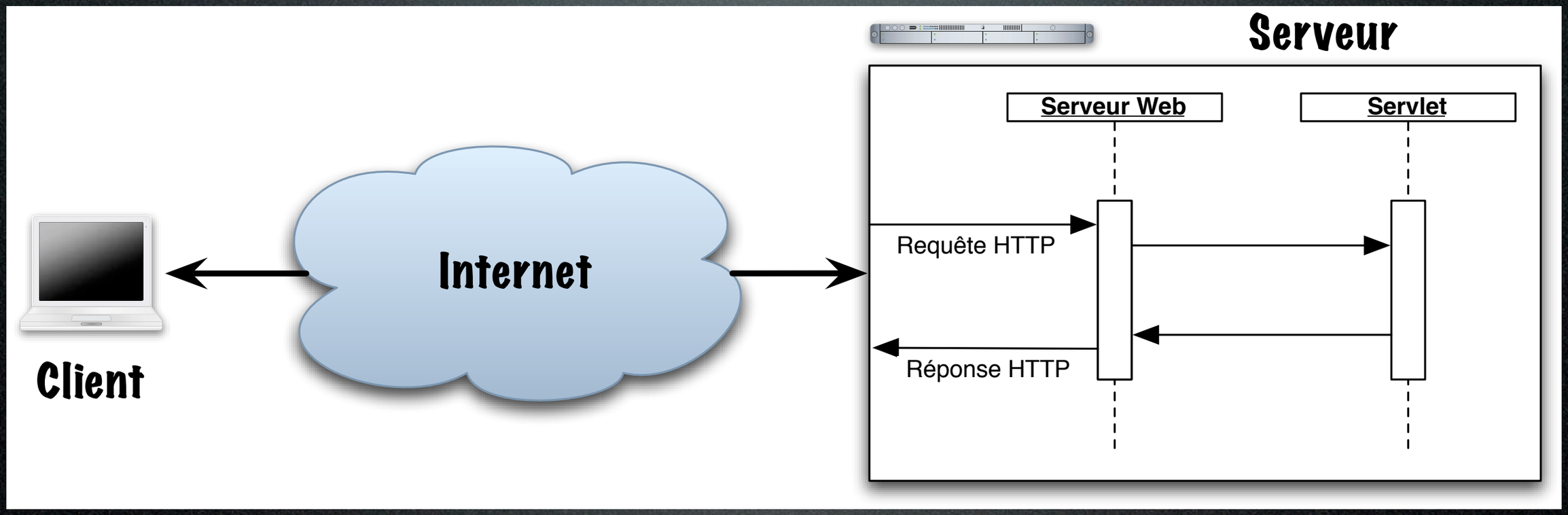
Lectures:

- Web Technologies (2007) § 6 Pages 323-350

Plan

1. Session
2. Témoins de connexion (cookies)
3. Réécritures d'URL

Servlet


Client
Internet
Serveur
Serveur Web
Servlet
Requête HTTP
Réponse HTTP

Résumé

- Un Servlet peut être vu comme une extension d'un serveur HTTP
- Un **Servlet** est une sous-classe de **HttpServlet**
  - Cycle de vie : **init**(), **service**(), **destroy**()
  - **doGet()** et **doPost()**, **HttpServletRequest** et **HttpServletResponse**
- Technologie sous-jacente des pages **JSP**

init()

```
public void init() throws UnavailableException {
    String filename = "/data/words.csv";
    ServletContext context = getServletContext();
    InputStream is = context.getResourceAsStream( filename );
    if ( is != null ) {
        try {
            InputStreamReader isr = new InputStreamReader( is );
            BufferedReader reader = new BufferedReader( isr );
            StringBuffer buffer = new StringBuffer();
            String line;
            while ( ( line = reader.readLine() ) != null ) {
                buffer.append( line );
            }
            is.close();
            dict = buffer.toString();
        } catch ( IOException e ) {
            throw new UnavailableException( "error while reading dictionary" );
        }
    }
}
```

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

public class GetTime extends HttpServlet {

    public void doGet( HttpServletRequest requete, HttpServletResponse reponse )
        throws ServletException, IOException {

        reponse.setContentType( "text/html; charset=\"UTF-8\"" );
        PrintWriter doc = reponse.getWriter();

        doc.println( "<!DOCTYPE html" );
          // ...
        doc.println( "</html>" );

        doc.close();
    }
}
```

Session

Témoins

Amazon.ca Checkout: Select Address

https://www.amazon.ca/gp/flex/checkout/sign-in/select.html

Google

amazon.ca

WELCOME ADDRESS ITEMS WRAP SHIP PAY PLACE ORDER

Choose a shipping address

Is the address you'd like to use displayed below? If so, click the corresponding "Ship to this address" button.

Or you can enter a new shipping address: Domestic (within Canada) | International (outside of Canada)

Address Book

Ship to this address

Marcel Turcotte
University of Ottawa
800 King Edward (SITE)
Ottawa, Ontario K1N 6N5
Canada

Edit

Sessions

Plusieurs requêtes peuvent provenir d'une même station de travail

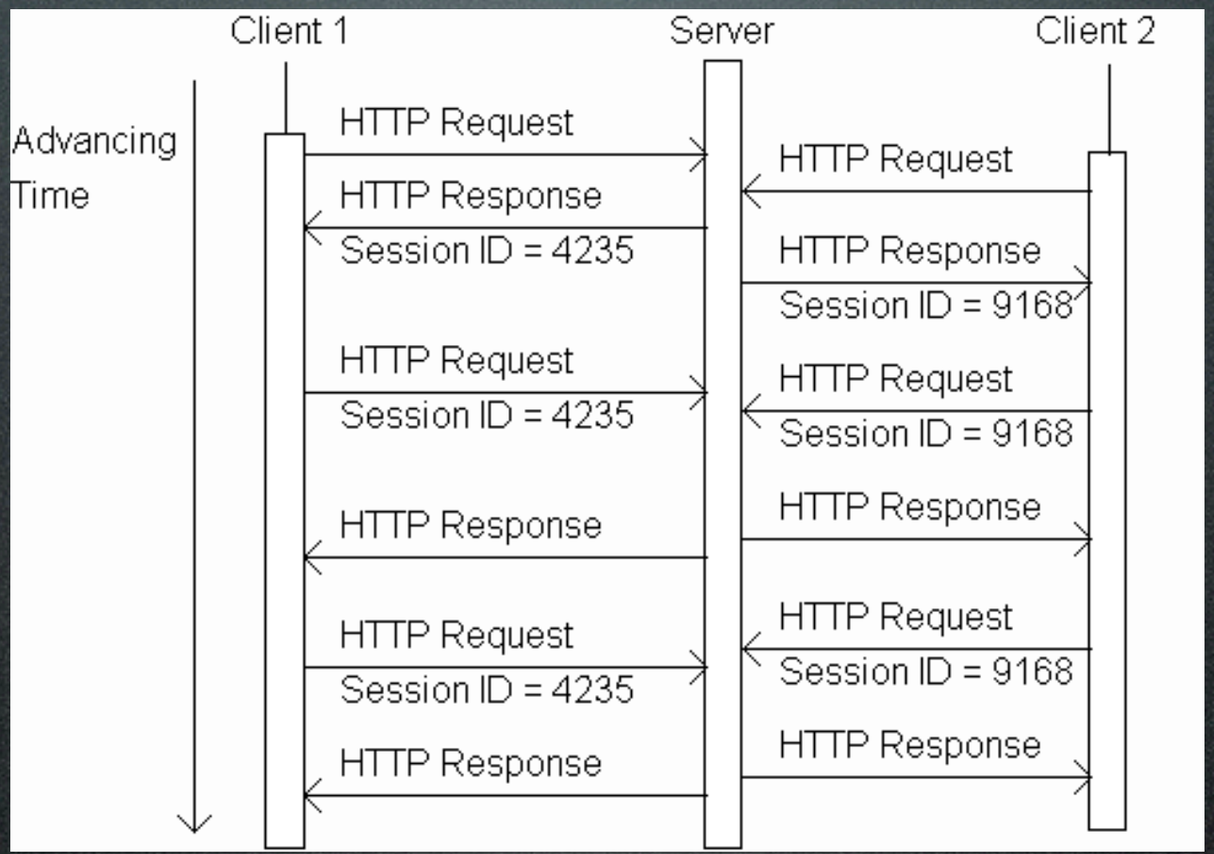
Le même usager pourrait même avoir deux sessions concourantes

- Fréquemment, les applications recueillent les informations à l'aide d'une suite de pages et donc de **plusieurs requêtes HTTP** (pensez au panier d'achats virtuel)
- **Problème :** comment établir des relations entre les différentes requêtes qui forment une même «session» ?

Sessions (haut niveau)

- Un identificateur est ajouté aux requêtes HTTP : Session ID
- Le **serveur** assignera un identificateur de session à toute requête qui n'en a pas
- L'identificateur sera retourné dans les messages réponse HTTP
- L'identificateur fera partie des requêtes subséquentes (caché et transmis par le **client**)

Sessions


Client 1
Server
Client 2
Advancing
Time
HTTP Request
HTTP Response
Session ID = 4235
HTTP Request
Session ID = 4235
HTTP Response
HTTP Request
Session ID = 4235
HTTP Response
HTTP Request
HTTP Response
Session ID = 9168
HTTP Request
Session ID = 9168
HTTP Response
HTTP Request
Session ID = 9168
HTTP Response

Session et Servlets

- Avec les Servlets, un objet réalisant l'interface **Session** modélise une session

```
HttpSession session = requete.getSession();
```

Application GetCount revisitée

```
HttpSession session = requete.getSession();

if ( session.isNew() ) {
   count++;
}
```

S'il s'agit d'une nouvelle session, l'appel à getSession() créera un nouvel objet
Sinon, retourne l'ancien
Ainsi, les appels succéssifs n'auront aucun effet

Terminer une session

- Dans bien des cas, on souhaite limiter la durée d'une session (système bancaire en ligne)

- session.setMaxInactiveInterval(int);
  le nombre de secondes

- session.invalidate();
  l'usager volontairement termine la session

Sauvegarder des données dans l'objet Session

- **Problème :** concevoir une application ou l'usager doit entrer un nom d'usager lors de sa première visite, ou sinon, lors des visites subséquentes, afficher le nom d'usager
- **Solution :** utiliser le concept de «session» afin d'identifier de façon unique un usager et de sauvegarder le nom d'usager dans l'objet Session

Ouverture de session

http://localhost:8080/servlet/Login

Google

Entrez votre nom d'usager :

Soumettre

LoginPage

doGet()

Succès

http://localhost:8080/servlet/Login

Google

Session établie pour l'usager **bobino**

Retour à la page principale

SuccesPage

doPost()

Continuation d'une session

http://localhost:8080/servlet/Login

Google

Bonjour **bobino!**

Terminer la session

MainPage

doGet()

Terminer

http://localhost:8080/servlet/Logout

Google

La session de l'usager **bobino** est terminée.

Retour à la page principale

Session

- Les 3 premières pages ont été générées par le même Servlet (**Login**)
- La dernière page a été générée par le Servlet **Logout**

Ouverture de session

http://localhost:8080/servlet/Login

Google

Entrez votre nom d'usager :

Soumettre

Succès

http://localhost:8080/servlet/Login

Google

Session établie pour l'usager **bobino**

Retour à la page principale

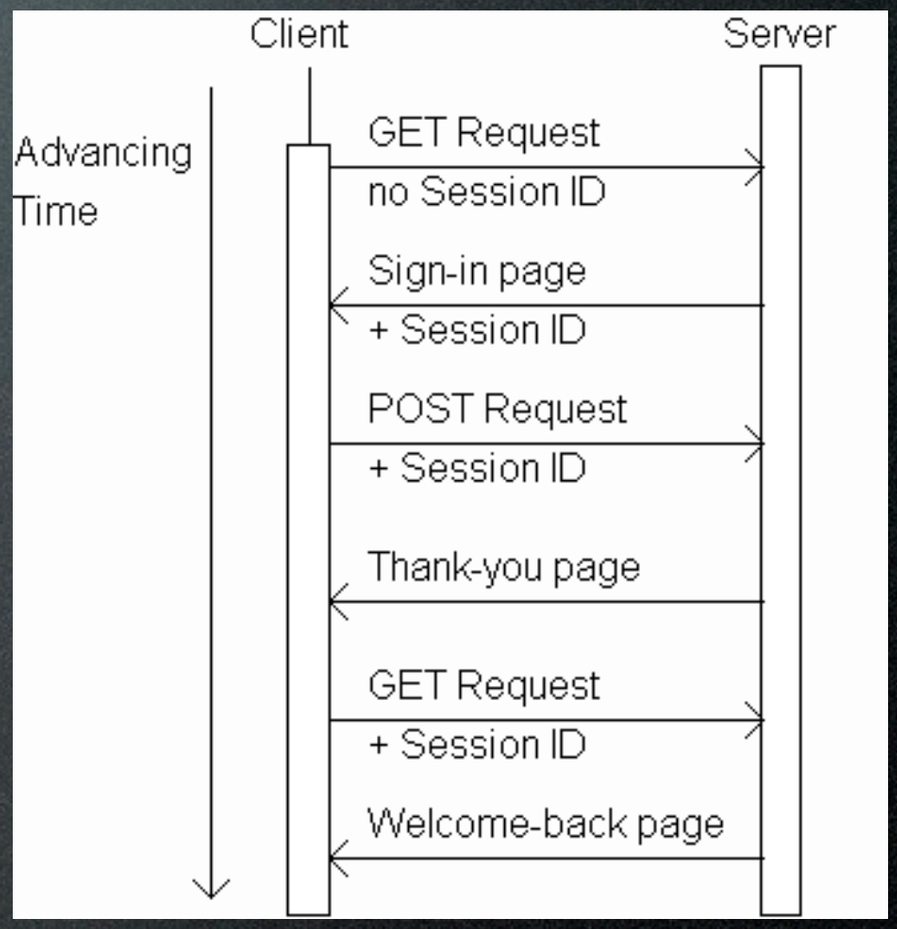
Continuation d'une session

http://localhost:8080/servlet/Login

Google

Bonjour **bobino!**

Terminer la session

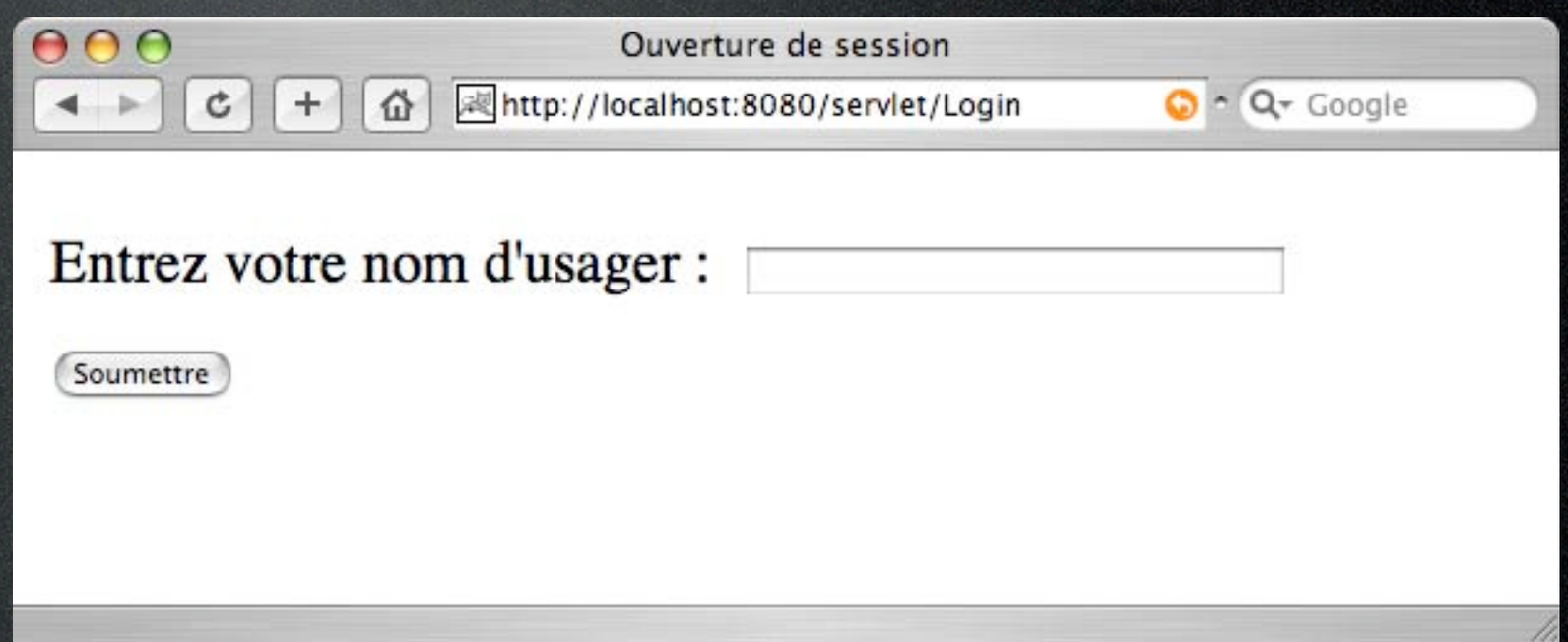


doGet()

```
reponse.setContentType( "text/html; charset=UTF-8" );
PrintWriter doc = reponse.getWriter();

HttpSession session = requete.getSession();
String usager = (String) session.getAttribute( "usager" );

if ( session.isNew() || usager == null ) {
    printLoginPage( doc );
} else {
    printMainPage( doc, usager );
}

doc.close();
```

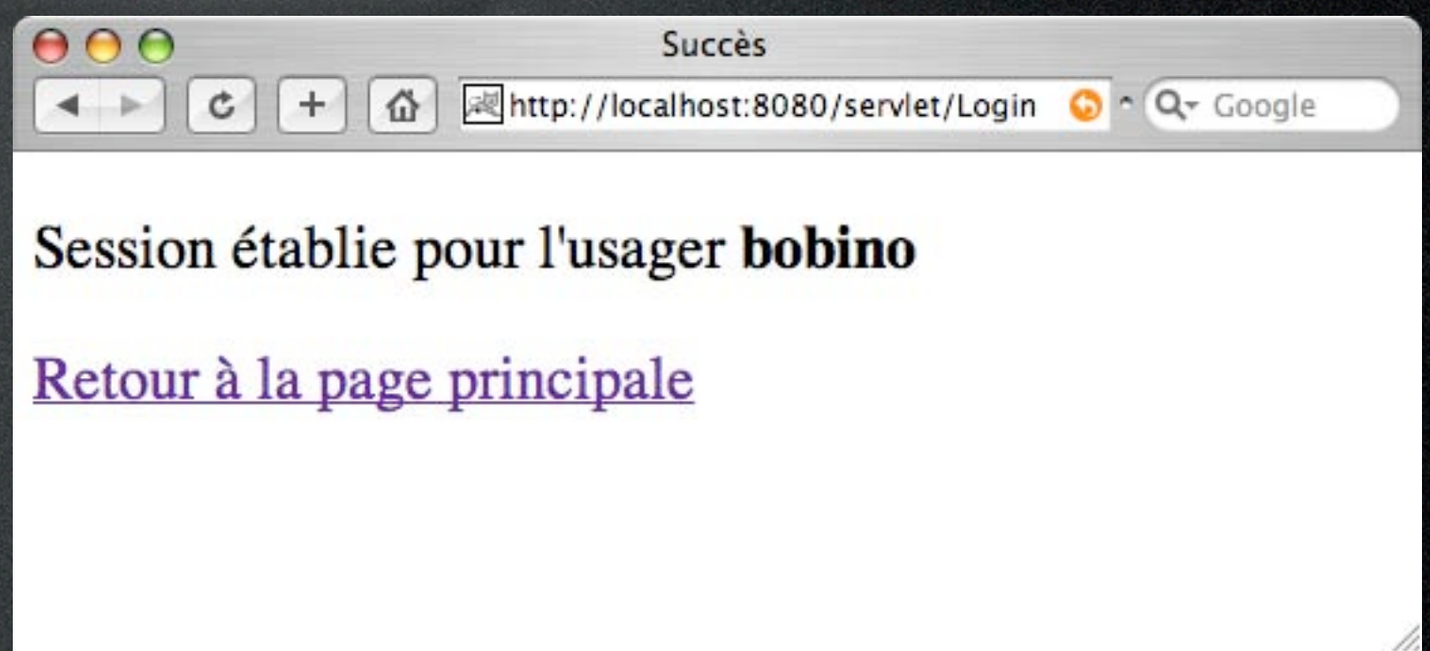
printLoginPage()

```
...
doc.println( "    <form action=\"http://localhost:8080/servlet/Login\" method=\"post\">" );
doc.println( "      <table border=\"0\" cellpadding=\"5\">" );
doc.println( "        <tr>" );
doc.println( "          <td>Entrez votre nom d'usager :</td>" );
doc.println( "          <td><input type=\"text\" size=\"30\" name=\"usager\" /></td>" );
doc.println( "        </tr>" );
doc.println( "        <tr>" );
doc.println( "          <td><input type=\"submit\" value=\"Soumettre\" /></td> " );
doc.println( "          <td></td>" );
doc.println( "        </tr>" );
doc.println( "      </table>" );
doc.println( "    </form>" );
...
```



doPost()

```
reponse.setContentType( "text/html; charset=UTF-8" );
PrintWriter doc = reponse.getWriter();

HttpSession session = requete.getSession();
String usager = requete.getParameter( "usager" );

if ( usager != null ) {
    printSuccessPage( doc, usager );
    session.setAttribute( "usager", usager );
} else {
    printLoginPage( doc );
}

doc.close();
```



printSuccessPage()

escapeAll(usager)

```
...
doc.println( "    <p>" );
doc.println( "    Session &eacute;tablie pour l'usager <b>" + usager + "</b>" );
doc.println( "    </p>" );
doc.println( "    <p>" );
doc.println( "    <a href=\"http://localhost:8080/servlet/Login\">Retour ...</a>" );
doc.println( "    </p>" );
...
```

L'utilisation d'une URL relative est plus flexible

```
doc.println( "    <a href=\"Login\">Retour ...</a>" );
```

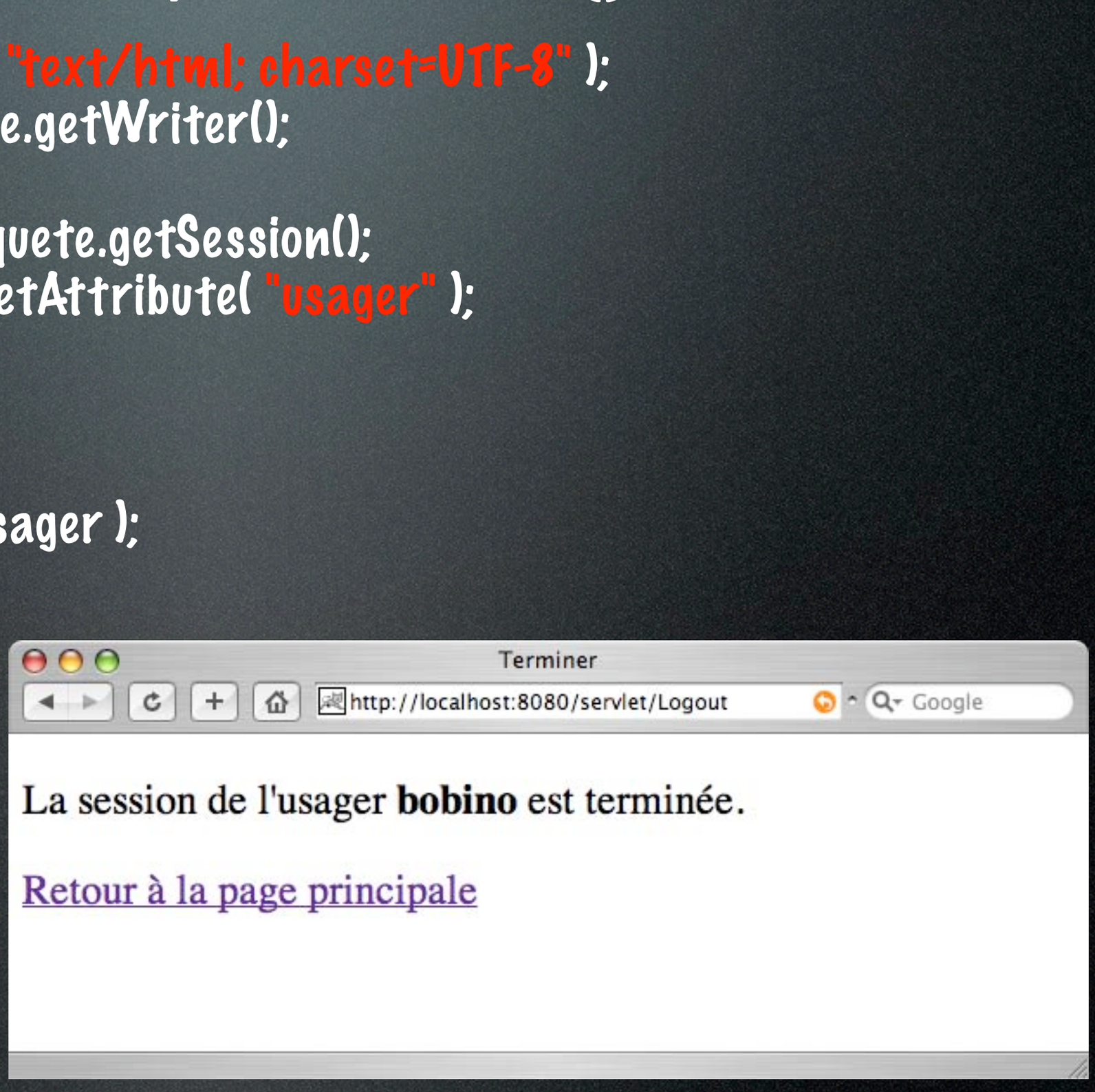
printMainPage()

escapeAll(usager)

```
...
doc.println( "    <p>" );
doc.println( "    Bonjour <b>" + usager + "!</b>" );
doc.println( "    </p>" );
doc.println( "    <p>" );
doc.println( "    <a href=\"Logout\">Terminer la session</a>" );
doc.println( "    </p>" );

...
```

Logout/doGet()

```
reponse.setContentType( "text/html; charset=UTF-8" );
PrintWriter doc = reponse.getWriter();

HttpSession session = requete.getSession();
String usager = session.getAttribute( "usager" );

if ( usager != null ) {
    session.invalidate();
    printMainPage( doc, usager );
} else {
    printErrorPage( doc );
}

doc.close();
```



- L'objet **Session** sauvegarde des associations **nom**/**valeur** :
  - **setAttribute(String, Object)**
  - **getAttribute(String)**: retourne l'objet associé au **nom** passé en paramètre ou **null**
  - **removeAttribute(String)**
  - **getAttributeNames()** retourne un objet **Enumeration** contenant la liste des attributs

- Une session a une durée de vie limitée
  - Soit la valeur par défaut déterminée par le serveur
  - Soit la valeur spécifiée par la méthode **setMaxInactiveInterval(sec)**

- On peut aussi terminer la session à l'aide de **invalidate()**
- **getMaxInactiveInterval(), getLastAccessedTime(), getCreationTime()**

Valeur négative signifie sans limite de temps

- L'objet **Session** est géré par le conteneur (Tomcat/GlassFish, par exemple)
- En général, l'objet n'est pas persistant (certains serveurs ont des objets **Session** persistants)

Session : implémentation

- Avec la technologie des Servlets, la gestion des sessions est quasiment transparente
- Il y a deux techniques d'implémentation :
  - À l'aide des témoins de connexion (cookies)
  - À l'aide de la réécriture d'URL

- Un **témoin** est une paire <nom,valeur>
- Un témoin est **créé** et sa valeur **mise à jour** dans une en-tête Set-Cookie d'un message **réponse HTTP** (donc par le serveur)
- L'agent utilisateur sauvegarde les témoins dans un fichier
- Lorsqu'un client communique avec un serveur, il lui envoit tous les témoins reçus de ce serveur, dans l'en-tête de la **requête HTTP** (lecture)

$ telnet www.amazon.ca 80
Trying 207.171.166.50...
Connected to www.amazon.ca.
Escape character is '^]'.
GET / HTTP/1.1
HOST: www.amazon.ca
Connection: close

HTTP/1.1 200 OK
Date: Fri, 05 Feb 2010 16:26:45 GMT
Content-Type: text/html; charset=ISO-8859-15
Set-cookie: skin=noskin; path=/; domain=.amazon.ca; expires=Fri, 05-Feb-2010 16:26:45 GMT
Set-cookie: session-id-time=1265961600l; path=/; domain=.amazon.ca; expires=Fri Feb 12 ...
Set-cookie: session-id=180-2449260-4625443; path=/; domain=.amazon.ca; expires=Fri ...

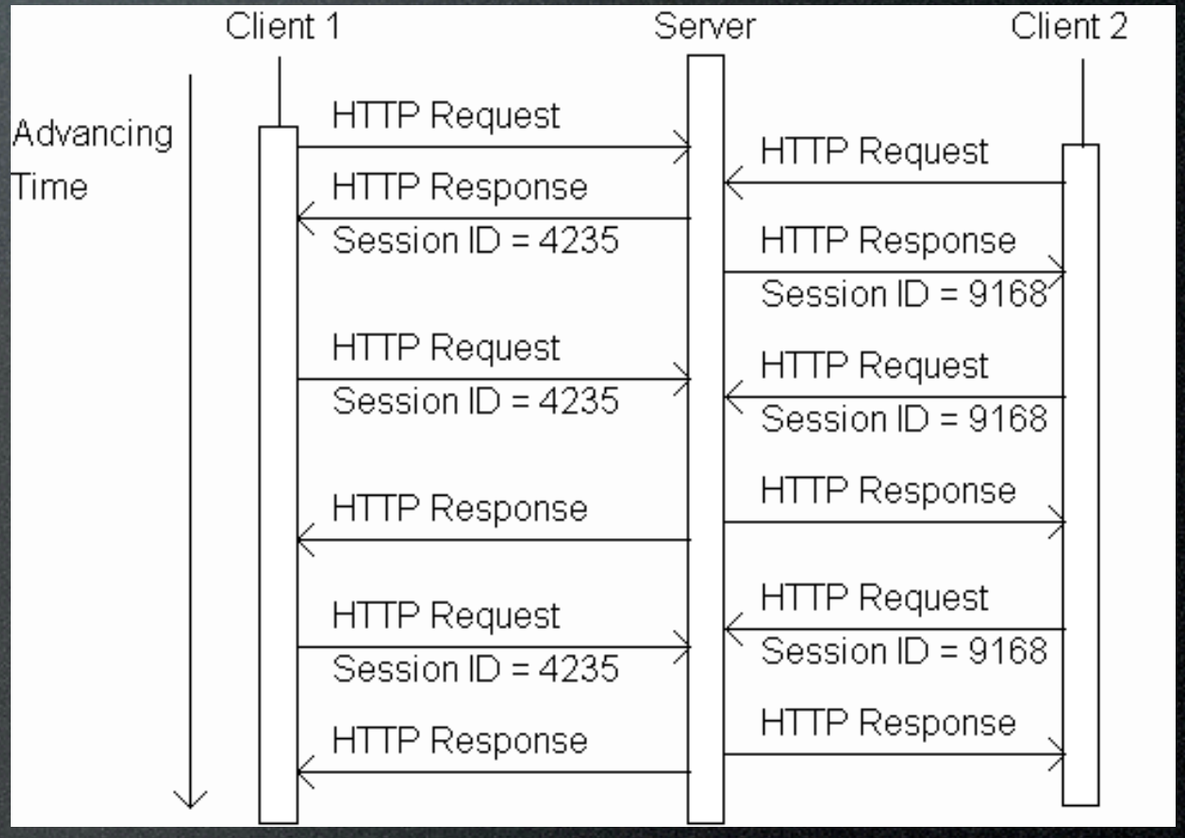
...

$ telnet www.amazon.ca 80
Trying 207.171.166.50...
Connected to www.amazon.ca.
Escape character is '^]'.
GET /gp/cart/view.html/ref=gno_cart HTTP/1.1
HOST: www.amazon.ca
Cookie: session-id=180-2449260-4625443
Connection: close

- Tomcat et GlassFish sauvegardent l'identificateur de session dans un témoin nommé JSESSIONID

Set-Cookie: JSESSIONID=699348562f8df0f5a64641 18457b; path=/Login

Cookie: JSESSIONID=699348562f8df0f5a64641 18457b



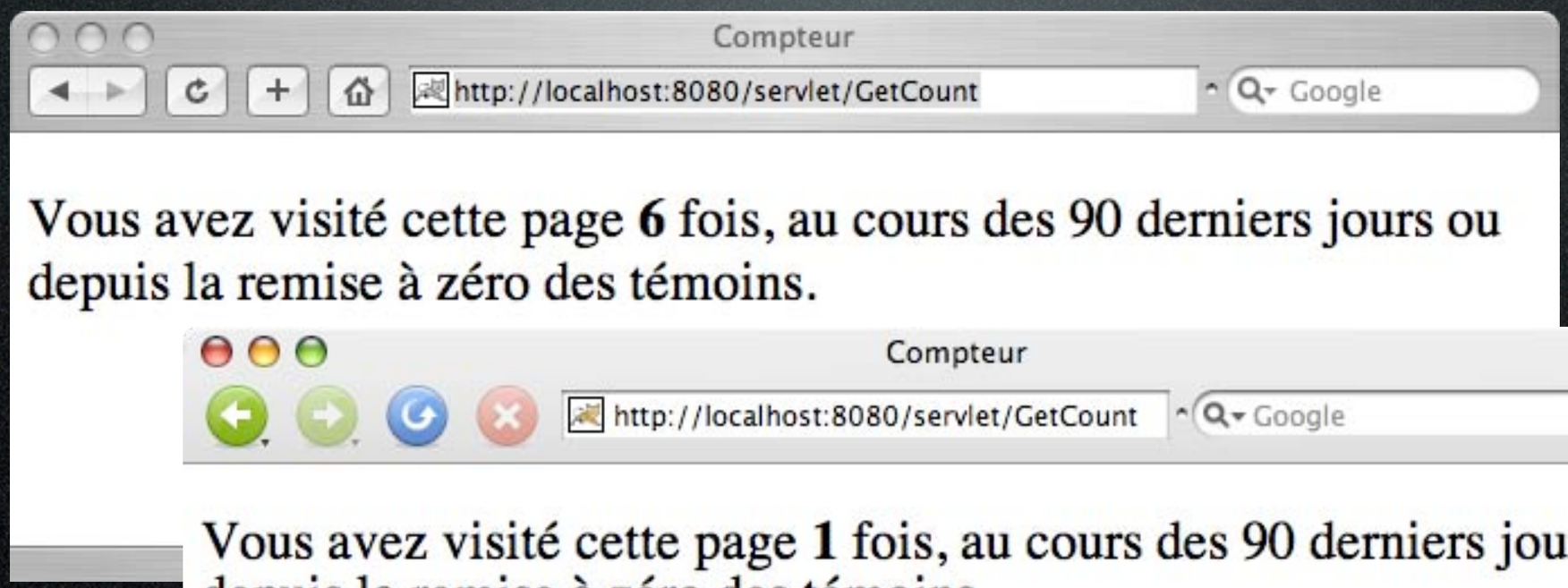
- Les Servlets peuvent manipuler les témoins (**cookies**) explicitement
  - **requete.getCookies()**
  - **reponse.addCookie(Cookie)** : retourne un tableau de témoins

- **Cookie(String nom, String valeur)**
- **String getName()**
- **String getValue()**
- **void setMaxAge(int seconds)**

Valeur négative signifie sans limite de temps

GetCount

- **Problème :** concevoir une application Web afin de compter de nombre de visites par usager



```
int count = 0;
Cookie[] cookies = requete.getCookies();

if ( cookies != null ) {
    for ( int i=0; i<cookies.length; i++ ) {
        if ( cookies[ i ].getName().equals( "COMPTEUR" ) ) {
            count = Integer.parseInt( cookies[ i ].getValue() );
        }
    }
}

count++;

Cookie cookie = new Cookie( "COMPTEUR", Integer.toString( count ) );
cookie.setMaxAge( 90 * 24 * 60 * 60 );

reponse.addCookie( cookie );

reponse.setContentType( "text/html; charset=UTF-8" );
PrintWriter doc = reponse.getWriter();

printPage( doc, count );
```

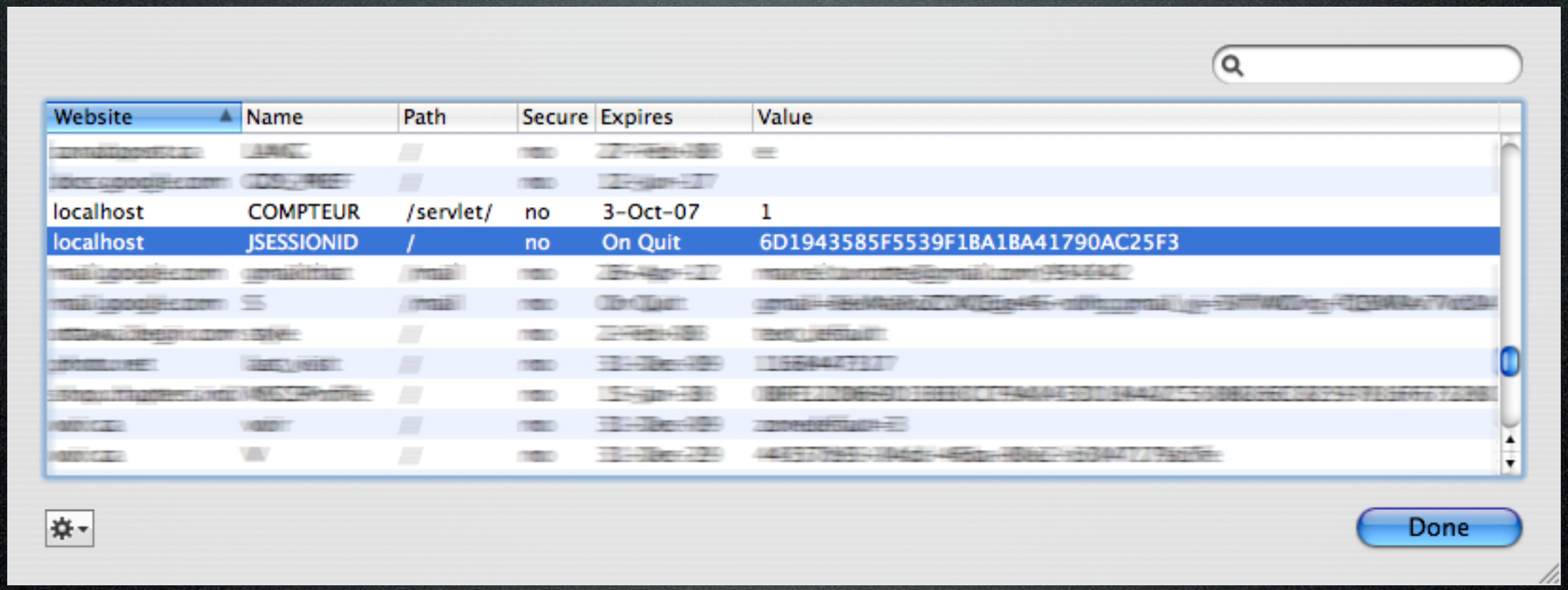
Témoins (cookies)

- Puisque les témoins sont mise à jour dans l'en-tête du message HTTP réponse, l'appel à la méthode **addCookie** doit se faire avant toute écriture dans le corps du message réponse!

Session et témoins

- Un conteneur de servlet (Tomcat) peut utiliser un témoin (**JSESSIONID**) afin d'implémenter les sessions
- Si le témoin **JSESSIONID** ne fait partie de la liste des témoins, c'est une nouvelle session
- Il crée alors un nouveau témoin, qu'il transmettra dans l'en-tête de la prochaine réponse (qui sera inclu dans toutes les requêtes suivantes du client)

Preferences | Privacy | Show Cookies ...



Témoins (cookies)

- Les informations persistent (suite au redémarrage du client et/ou du serveur)
- Certains fureteurs n'acceptent pas les témoins ou en accepte un nombre limité (20)
- Puisque certains fureteurs refusent les témoins, cette technique ne peut être la seule pour l'implémentation des sessions

- **Un témoin n’est transmis qu’au serveur duquel il a été reçu!**
- Comment peut-on alors établir un profil de l’usager à travers plusieurs sites?
- La stratégie **DoubleClick** (acheté par Google en 2007)
  - Les membres participants ajoutent une bannière publicitaire à leurs pages : élément IMG dont la source est une URL du domaine **doubleclick.com**
- Certains usagers désactivent les témoins !

Serveur

Get

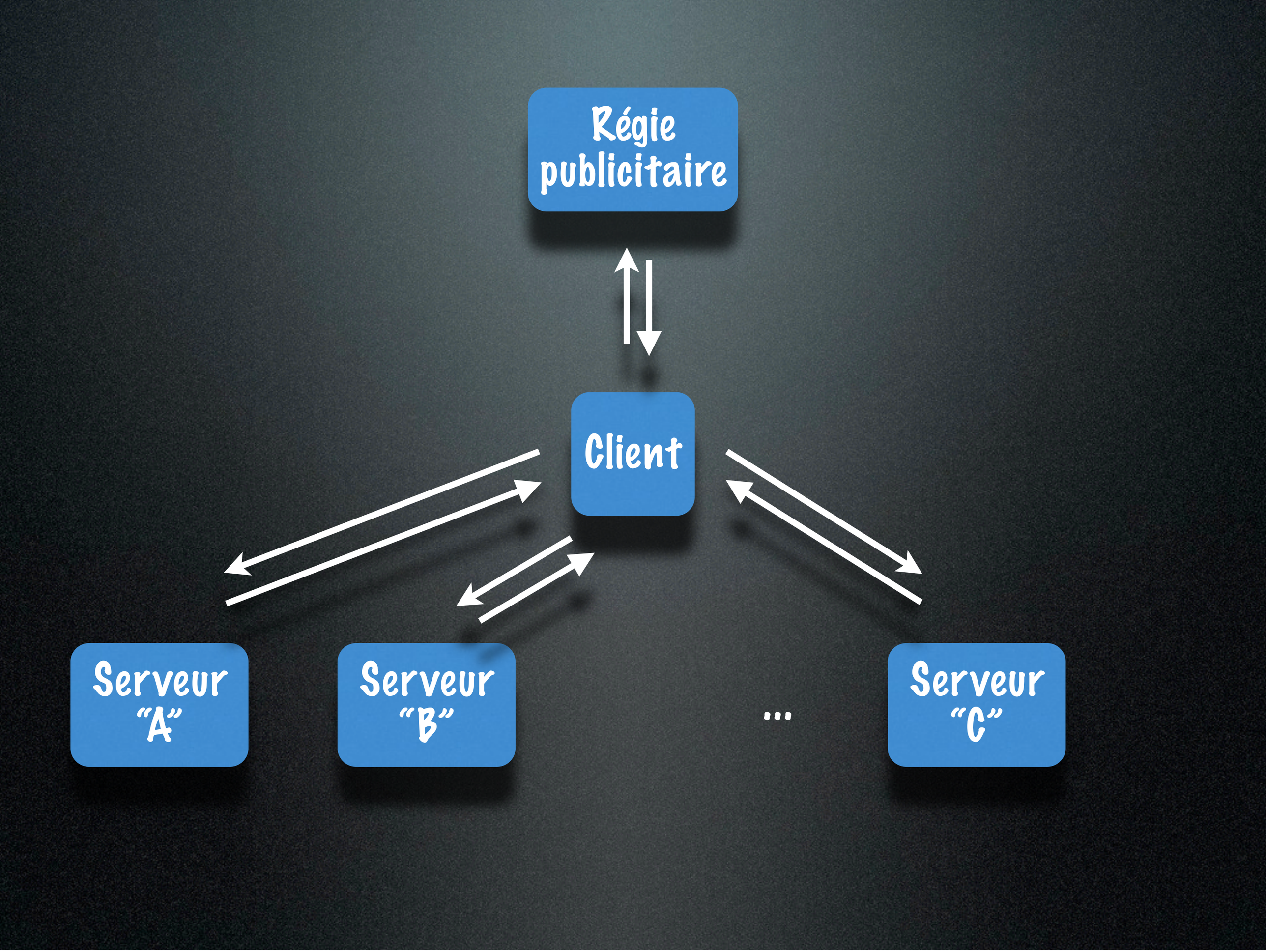
Page contenant

Client

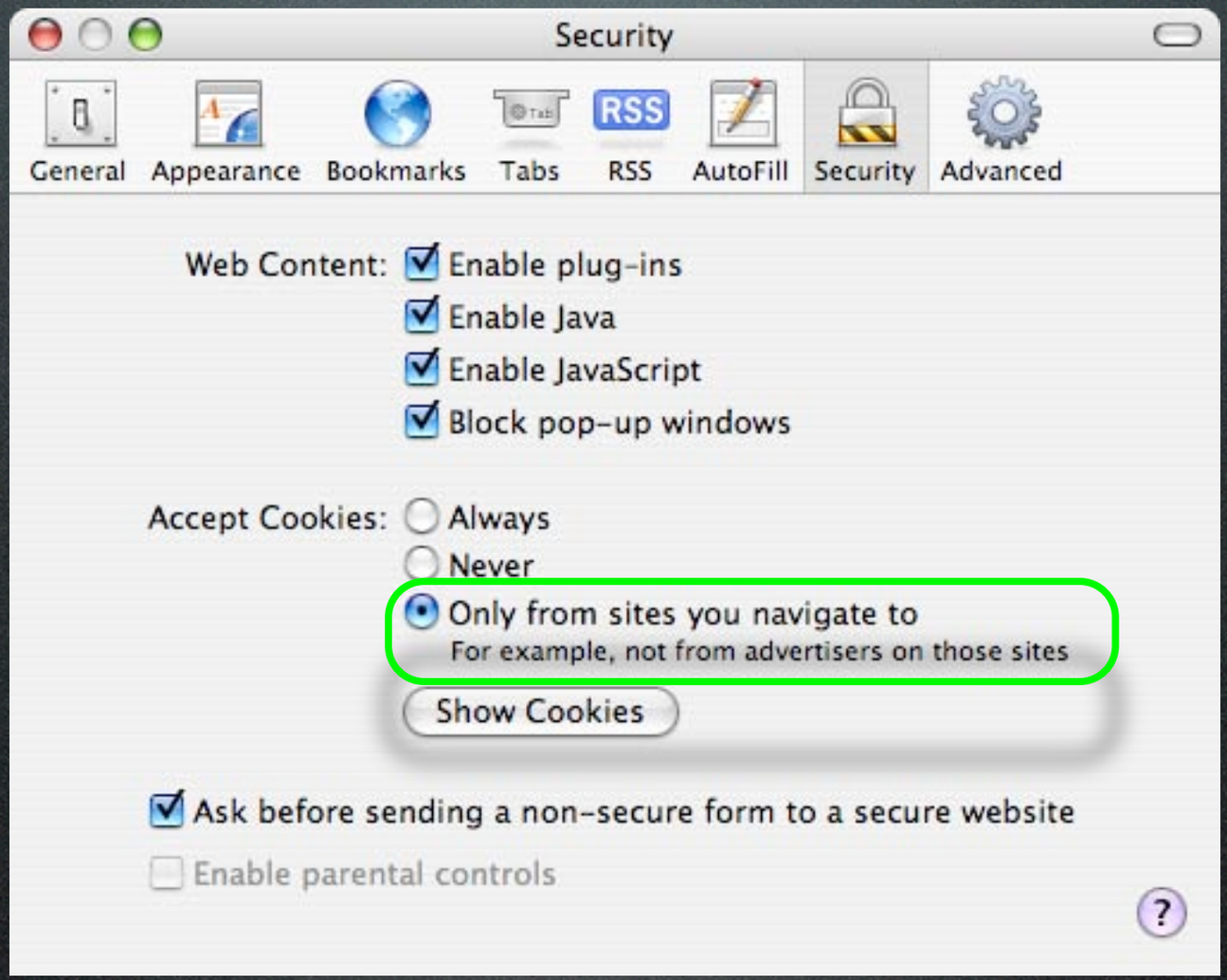
Get
+ témoins

Régie
publicitaire

banner.png
+
Set-Cookie: ...


Régie publicitaire
Client
Serveur "A"
Serveur "B"
...
Serveur "C"

Témoins et vie privée



Ouverture de session

http://localhost:8080/servlet/Login

Google

Entrez votre nom d'usager :

Soumettre

Succès

http://localhost:8080/servlet/Login

Google

Session établie pour l'usager **picolo**

Retour à la page principale

General Appearance B

Open "http://localho

Web Content: Enable plu

Enable Java

Enable Java

Block pop-

Accept Cookies: Always

Never

Only from

For example, not from advertisers on those sites

Show Cookies

Ask before sending a non-secure form to a secure website

Enable parental controls

?

Ouverture de session

http://localhost:8080/servlet/Login

Google

Entrez votre nom d'usager :

Soumettre

Réécriture d'URL

- Un segment de l'URL sert à passer des paramètres à l'application Web :

  http://localhost:8080/servlet/GetParam?name=bond&id=007

  getQueryString()

- De même, la recommandation RFC 2396 permet aussi l'ajout de paramètres (path parameters) à la suite du symbole «;»

  http://localhost:8080/servlet/Login;jsessionid=1234

  getQueryString()

- Lorsque l'application produit une page XHTML, tous les liens vers les autres pages d'une même session sont transformés à l'aide d'un appel à la méthode **reponse.encodeURL(url)**
- La méthode retourne la valeur de l'URL à laquelle elle aura ajouté un suffixe **;jsessionid=E54A06ADD16698C709CA50EF22CCF9D5**
- Lorsque le serveur reçoit une requête HTTP, il consulte d'abord les témoins,

Réécriture d'URL

```
public void doGet( HttpServletRequest requete, HttpServletResponse reponse )
  ...

  printSuccessPage( doc, usager, reponse.encodeURL( "Login" ) );
  }
  ...
}

private void printSuccessPage( PrintWriter doc, String usager, String url ) {

  ...
  doc.println( "      <a href=\"" + url + "\">Retour &agrave; la page principale</a>" );
  ...
}
```

Ouverture de session

http://localhost:8080/servlet/Login

Google

Entrez votre nom d'usager :

Soumettre

Succès

http://localhost:8080/servlet/Login;jsessionid=E54A06ADD16698C709CA50EF22CCF9D5

Google

Session établie pour l'usager **bernard**

Retour à la page principale

Continuation d'une session

http://localhost:8080/servlet/Login;jsessionid=E54A06ADD16698C709CA50EF22CCF9D5

Google

Bonjour **bernard!**

Terminer la session

Terminer

http://localhost:8080/servlet/Logout;jsessionid=E54A06ADD16698C709CA50EF22CCF9D5

Google

La session de l'usager **bernard** est terminée.

Retour à la page principale

Open "http://localhost:8080/servlet/Login" in a new tab

Réécriture d'URL

- **Tous les liens doivent être encodés**, sinon la relation avec une session existante ne peut être établie
- On ne peut donc **pas** concevoir une application mixte, mélangeant des **pages dynamiques** et des **pages statiques**
- **Coté sécurité**, les URLs encodent explicitement l'identifiant de session

Ressources

- SR-000154 JavaTM Servlet 2.4 Specification [http://jcp.org/aboutJava/communityprocess/final/jsr154/index.html] 2007
- Java Servlet Technology [http://java.sun.com/products/servlet] 2008-02-26