Cookies et suivi de session

Cookies

Introduction
L'API javax.servlet.http.Cookie
Utilisation de cookies

Introduction (1/2)

Un cookie est une information envoyée par un serveur web à un navigateur et sauvegardé par celui-ci sur le disque de sa machine

Le navigateur retourne cette information (inchangée) lorsqu'il visite à nouveau le même site.

Ayant lu cette information, le serveur peut identifier le client et lui fournir un certain nombre de facilités :

- achat en ligne
- authentification
- personnalisation de portail
- diriger les recherches d'un moteur

Pas de menace sécuritaire

Limité à 4Ko. Un navigateur accepte 20 cookies/site et 300 au total

Introduction (2/2)

Cette technique est utilisée pour le suivi de session. Il mémorise l'identifiant de session (sessionid) pour le navigateur

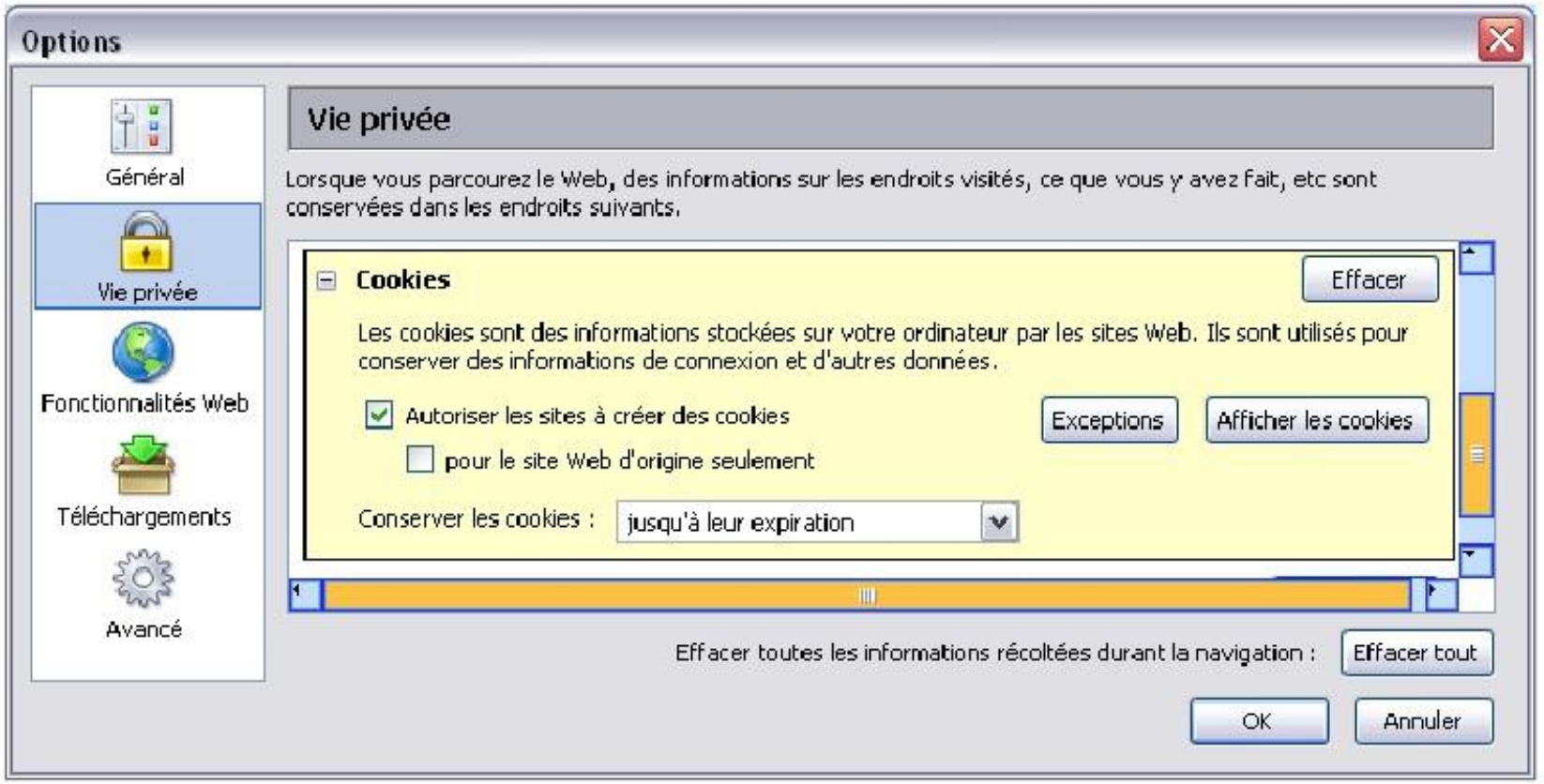
Pour stocker un cookie sur un navigateur, la servlet doit ajouter le cookie dans l'en-tête de sa réponse

Pour récupérer l'information à partir d'un cookie, la servlet doit extraire le cookie de l'en-tête de la requête

Ceci est réalisé et rendu transparent par l'API **Cookie** et ses méthodes d'accès : `addCookie()` et `getCookies()`

Cookies

Les navigateurs n'acceptent pas toujours les cookies (WAP, ...). Il est possible de les inhiber



API `javax.servlet.http.Cookie`

Constructeur :

public `Cookie(String name,String value)`

Envoi d'un cookie à un client par une servlet :

public `void HttpServletResponse.addCookie(Cookie cookie)`

Récupération des cookies. Il est impossible de récupérer un cookie connaissant son nom. On ne peut que récupérer un tableau de tous les cookies envoyés par le navigateur :

public `Cookie[] HttpServletRequest.getCookies()`

Modification de la valeur d'un cookie :

public `void setValue(String newValue)`

Utilisation de cookies

- Création d'un cookie (`res` est l'objet réponse)

```
Cookie unCookie = new Cookie("nom","martin");
res.addCookie(unCookie);
```

- Récupération d'un cookie (`req` est l'objet requête)

```
Cookie[] cookies = req.getCookies();
if (cookies != null)
   for ( int i=0;i<cookies.length;i++ ){
      if (cookies[i].getName().equals("nom")){
            String valeur = cookies[i].getValue();
          break;
      }
   }
}
```

Attributs des cookies (1/2)

Un cookie est défini par son nom auquel est associée une valeur.
Il est possible d'y ajouter certains attributs

`void setDomain(String pattern)`

- précise le domaine pour lequel le cookie est valide
- par défaut, le cookie est retourné seulement au serveur qui l'a sauvé

`void setMaxAge(int expiry)`

- spécifie l'age maximum du cookie en secondes
- `expiry<0` => le cookie expire avec le navigateur
- `expiry=0` => le cookie est détruit immédiatement

`void setPath(String uri)`

- définit le chemin dans le domaine pour lequel le cookie est valide
- par défaut, le cookie est valide pour la page qui le définit et toutes les pages du répertoire du dessous
- si `uri="/",` le cookie est valide pour toutes les pages du serveur

Attributs des cookies (2/2)

`void setSecure(boolean flag)`

- `flag=true` **=>** le cookie doit être envoyé via un canal sécurisé (SSL)

`void setComment(String comment)`

- définit un commentaire qui décrit le rôle du cookie

Exemple (1/10) - l'interface

L'exemple présente une page qui permet de saisir un mot.

Le mot saisi est conservé dans un cookie.

Exemple (2/10) - l'interface

Ainsi chaque fois que la page est demandée (après rechargement), le dernier mot saisi est affiché.

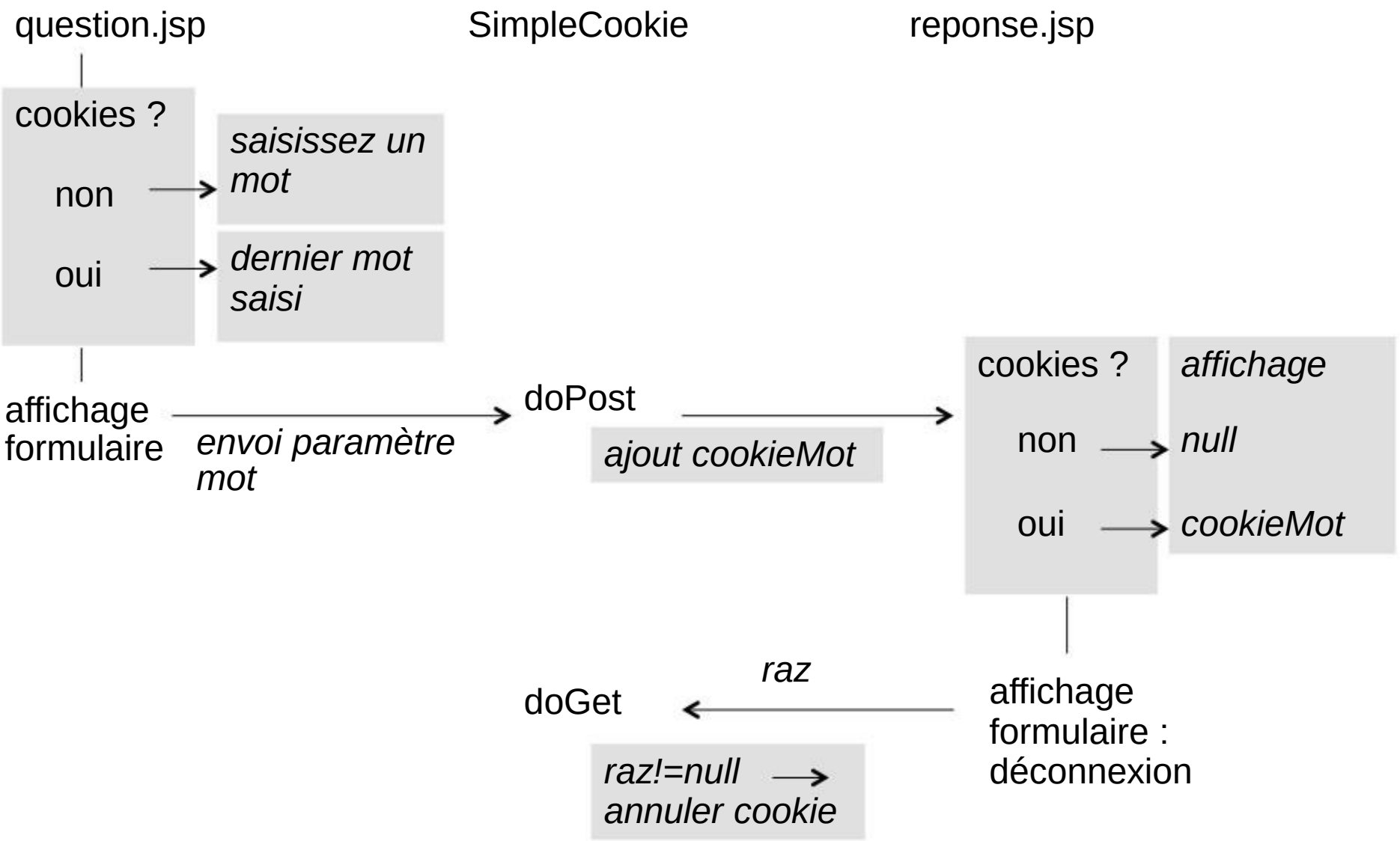
De même si une requête à cette page est réalisée à travers une autre instance du navigateur

Après déconnexion, la valeur du cookie est réinitialisée

Exemple (1/10) - web.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE web-app SYSTEM "http://java.sun.com/dtd/web-app_2_3.dtd">
<web-app>
 <servlet>
  <servlet-name>simpleCookie</servlet-name>
  <servlet-class>servlets.SimpleCookie</servlet-class>
  <init-param>
    <param-name>JSPQuestion</param-name>
    <param-value>/question.jsp</param-value>
   </init-param>
   <init-param>
    <param-name>JSPReponse</param-name>
    <param-value>/reponse.jsp</param-value>
   </init-param>
 </servlet>
 <servlet-mapping>
   <servlet-name>simpleCookie</servlet-name>
   <url-pattern>/simpleCookie</url-pattern>
 </servlet-mapping>
</web-app>
```

Fonctionnement



Exemple (2/10) - question.jsp

fichier
question.jsp

```
<%
Cookie[] cookies = request.getCookies();
String mot=null;
if (cookies != null) {
   for (int i=0;i<cookies.length;i++){
      if(cookies[i].getName().equals("cookieMot"))
         mot=cookies[i].getValue();
      }
   }
%>
<%@page pageEncoding="iso-8859-1"%>
<%@page session="false"%>
<html>
   <head><title>Question</title></head>
   <body bgcolor=yellow text=blue>
      <center>
```

Exemple (3/10) - question.jsp

```
<% if(mot==null || mot.equals("  ")) {%>
  <h2> Saisissez un mot !</h2>
<%}else{%>
  <h2>Dernier mot saisi  <%=mot %></h2>
<%}%>
<form action='simpleCookie' method="post" >
<hr>
<table><tr>
  <td>Votre mot</td>
  <td><input name="Mot" value="<%= mot %>"
                                    type="text" size="20">
</tr></table>
<table><tr>
<td><input type="submit" value="Envoyer"></td>
<td><input type="reset" value="Rétablir"></td>
</tr></table>
</form>
</center>
</body>
</html>
```

fichier `question.jsp` (suite)

Exemple (4/10) - la servlet

```
package servlets;
import javax.servlet.http.*;
import javax.servlet.ServletConfig;
import javax.servlet.ServletContext;
import javax.servlet.ServletException;
import java.io.*;

public class SimpleCookie extends HttpServlet      {
    private String JSPQuestion =null;
    private String JSPReponse =null;


//init
    public void init(ServletConfig config)
                        throws ServletException {
      JSPQuestion = config.getInitParameter("JSPQuestion");
      JSPReponse = config.getInitParameter("JSPReponse");
}
```

Servlet
SimpleCookie
.java

méthode
init()

Exemple (5/10) - la servlet - doPost

méthode
`doPost(...)`

```
public void doPost
(HttpServletRequest request,HttpServletResponse response)
            throws IOException, ServletException {

  String mot = request.getParameter("Mot");
  Cookie cookieMot = new Cookie("cookieMot",mot);
  cookieMot.setMaxAge(600);
  response.addCookie(cookieMot);
  request.getRequestDispatcher(JSPReponse).
                                forward(request,response);
 }

}
```

Exemple (6/10) - reponse.jsp

fichier
reponse.jsp

```
<%
Cookie[] cookies = request.getCookies();
String mot=null;
if (cookies != null) {
   for (int i=0;i<cookies.length;i++){
   if(cookies[i].getName().equals("cookieMot"))
   mot=cookies[i].getValue();
   }
}
%>
<%@page pageEncoding="iso-8859-1"%>
<%@page session="false"%>
```

Exemple(7/10) - reponse.jsp

fichier `reponse.jsp` (suite)

```
<html>
 <head><title>Bienvenue</title></head>
 <body bgcolor=green text=white>
 <center>
    <h2> Vous avez saisi :  <%= mot%> </h2>
    <FORM action='simpleCookie' method='get'>
    <INPUT type="submit" value="déconnexion">
    <INPUT type="hidden" name='raz' value='1'>
    </FORM>
  </center>
</body>
</html>
```

Exemple (8/10) - la servlet - doGet

méthode
`doGet(...)`

GET

```
public void doGet
 (HttpServletRequest request, HttpServletResponse response)
                    throws IOException, ServletException {

 //Récupération du paramètre de déconnexion
  String raz=request.getParameter("raz");

  //Traitement du paramètre de déconnexion
  if(raz!=null) {
    Cookie[] cookies = request.getCookies();
    if (cookies!=null) {
      for (int i=0;i<cookies.length;i++){
        if(cookies[i].getName().equals("cookieMot")) {
           cookies[i].setValue("  ");
           response.addCookie(cookies[i]);
  } } } }
request.getRequestDispatcher(JSPQuestion).forward(request,response);

}
```

Autre solution

Cette solution au même problème n'utilise que 2 servlets

`Question.java`
`Reponse.java`

Exemple (1/5) - web.xml

```
<web-app>
  <servlet>
    <servlet-name>question</servlet-name>
    <servlet-class>servlets.Question</servlet-class>
  </servlet>
  <servlet>
    <servlet-name>reponse</servlet-name>
    <servlet-class>servlets.Reponse</servlet-class>
  </servlet>

  <servlet-mapping>
    <servlet-name>question</servlet-name>
    <url-pattern>/question</url-pattern>
  </servlet-mapping>

  <servlet-mapping>
    <servlet-name>reponse</servlet-name>
    <url-pattern>/reponse</url-pattern>
  </servlet-mapping>
</web-app>
```

fichier web.xml

Exemple (2/5) - Question.java

fichier
Question.java

```
package servlets;
import java.io.IOException;
import java.io.PrintWriter;
import javax.servlet.*;
import javax.servlet.http.*;
public class Question extends HttpServlet {
  public void doGet
   (HttpServletRequest request,HttpServletResponse response)
                              throws IOException, ServletException {
    response.setContentType("text/html");
    PrintWriter out = response.getWriter();
    String raz=request.getParameter("raz");
    if(raz!=null) {
       Cookie[] cookies = request.getCookies();
       if (cookies != null) {
          for (int i=0;i<cookies.length;i++){

             if(cookies[i].getName().equals("cookieMot")) {
               cookies[i].setValue("  ");
                    response.addCookie(cookies[i]);}
          }    }
     }String mot=null;
```

Exemple (3/5) - Question.java

```
out.println("<html><head><title>Question</title></head>"+
    " <body bgcolor=yellow text=blue>"+
    "<center>");
    if(mot==null || mot.equals("  ")) {
       out.println("<h2> Saisissez un mot !</h2>");
       mot="   ";
    }else{
       out.println("<h2>Dernier mot saisi"+mot +"</h2>");
    }
    out.println("<form action='reponse' method='post' >"+
    "<hr><table><tr>"+
    "<td>Votre mot</td>"+
    "<td><input name='Mot' value="+mot+" \"");
    out.println("type='text' size='20'></tr></table><table>"+
    "<tr>"+
    "<td><input type='submit'value='Envoyer'</td>"+
    "<td><input type='reset' value='Rétablir'></td>"+
    "</tr></table>"+
    "</form>"+
    "</center>"+"</body>"+"</html>");
    }
}
```

fichier Question.java (suite)

Exemple (4/5) - Reponse.java

fichier
Reponse.java

```
package servlets;
import java.io.IOException;
import java.io.PrintWriter;
import javax.servlet.ServletException;
import javax.servlet.http.*;
public class Reponse extends HttpServlet {
   public void doPost
     (HttpServletRequest request,HttpServletResponse response)
                    throws IOException, ServletException {
       response.setContentType("text/html");
       PrintWriter out = response.getWriter();
       String mot = request.getParameter("Mot");
       Cookie cookieMot = new Cookie("cookieMot",mot);
       cookieMot.setMaxAge(600);
       response.addCookie(cookieMot);
       Cookie[] cookies = request.getCookies();
       if (cookies != null) {
          for (int i=0;i<cookies.length;i++){
              if(cookies[i].getName().equals("cookieMot"))
                 mot=cookies[i].getValue();
          }
       }
```

Exemple (5/5) - Reponse.java

fichier `Reponse.java` (suite)

```
out.println(
"<html>"+
"<head>"+
"<title>Bienvenue</title>"+
"</head>"+
"<body bgcolor=green text=white>"+
"<center>"+
"<h2> Vous avez saisi :"+  mot+"</h2>"+
"<FORM action='question' method='get'>"+
"<INPUT type='submit' value='déconnexion'>"+
"<INPUT type='hidden' name='raz' value='1'>"+
"</FORM>"+
"</center>"+
"</body>"+
"</html>");
}

}
```

Suivi de session

Introduction

Solutions

Support Session de l'API Servlet

Introduction

Le protocole HTTP est un protocole sans état =>

le serveur ignore qu'une séquence de requêtes provient d'un même client

En mode HTTP, pour le serveur, 2 requêtes successives d'un même client sont indépendantes

Le serveur HTTP voit les requêtes, pas les clients.

En effet une adresse IP n'est pas suffisante pour identifier un utilisateur

Exemple : un serveur web de commerce électronique gère un panier. Les articles achetés, ajoutés au panier, donnent lieu à différentes requêtes

Gestion de session

La notion de session n'est pas liée au protocole HTTP.

Une session est définie comme une collection de requêtes HTTP entre un client et un serveur web sur une période de temps

La notion de session permet d'associer un ensemble de requêtes et de les identifier comme appartenant à un même client.

La technique de la session est donc utilisée pour maintenir un lien entre plusieurs requêtes

L’API de suivi de session (1/2)

A chaque utilisateur est associé implicitement un objet utilisé par les servlets pour sauvegarder un ensemble d’objets (un panier par exemple)

Cet objet de type `HttpSession` permet de suivre l'activité d'un client sur plusieurs pages

les requêtes provenant d'un même utilisateur sont associées à ce même objet

Chaque `ServletContext` gère ses propres instances de `HttpSession`

L'API de suivi de session (2/2)

- `HttpSession HttpServletRequest.getSession()`
pour récupérer l'objet session courant

- `void HttpSession.setAttribute(String name, Object value)`
ajoute un couple `(name, value)` à cette session

- `Object HttpSession.getAttribute( String name )`
retourne l'objet associé à la clé `name` ou `null`

- `void HttpSession.removeAttribute( String name )`
enlève le couple de clé `name`

- `java.util.Enumeration HttpSession.getAttributeNames()`
retourne tous les noms d'attributs associés à la session

- `void HttpSession.setMaxIntervalTime( int seconds )`
spécifie la durée de vie maximum d'une session

Fonctionnement d'une session

A la première requête vers une application web :

- un objet `HttpSession` est créé
- ainsi qu'un identifiant unique pour cet objet

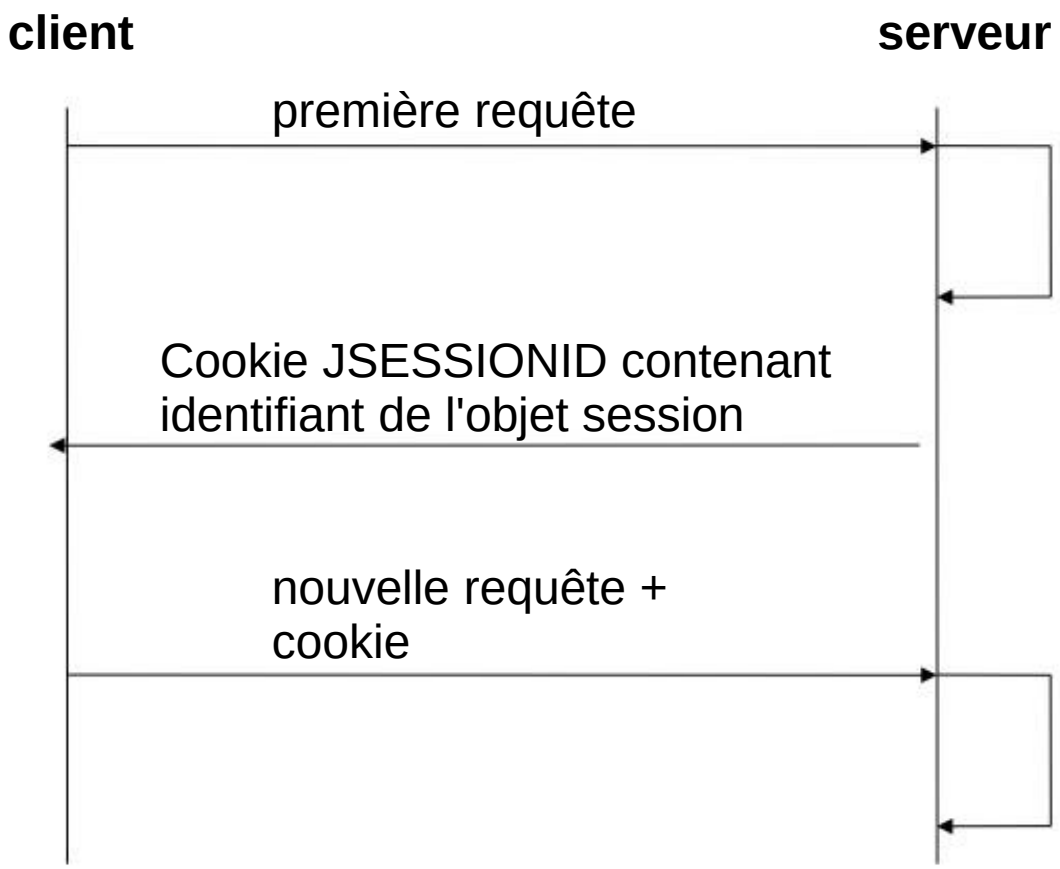
L'identifiant est en général sauvegardé par un cookie appelé JSESSIONID
=> seul l'identifiant de session est envoyé au client

Grâce à cet identifiant, le serveur détermine l'objet session correspondant à la requête courante

A toute nouvelle requête émise par l'utilisateur, le cookie est transmis via le serveur web et l'identifiant de session récupéré par la méthode :

```
public String HttpSession.getId()
```

Fonctionnement d’une session



Cycle de vie d'une session

A sa création, une période de temps est affectée à la session

Elle expire automatiquement à la fin de cette période (par défaut 30mns avec Tomcat)

Elle peut être invalidée explicitement par une servlet :
`HttpSession.invalidate()`

par une jsp :

```
<%@page session="false"%>
```

A l'expiration (invalidation), les données de l'objet session (`HttpSession`) sont retournées au moteur de servlets

Les sessions ne sont donc pas invalidées à la fermeture du navigateur

Codage du délai d'expiration (timeout)

Le timeout par défaut peut-être codé dans le descripteur de déploiement (fichier `web.xml`) en ajoutant les balises :

```
<session-config>
   <session-timeout>
     120
   </session-timeout>
</session-config>
```

Une valeur de timeout peut aussi être affectée à chaque session en informant l'objet session par la méthode :

```
public void HttpSession.setMaxInactiveInterval(int secs)
```

Exemple : suivi de session

Dans l'exemple qui suit, on souhaite mémoriser le nombre de fois que le mot "vu" a été saisi à travers les requêtes HTTP d'un même client.
Pour cela, on utilise un objet `HttpSession` dans lequel peuvent être stockées des ressources sous la forme (nom -`String`-, valeur - `Object`-).
Le compteur de saisies sera mémorisé par la session.

La servlet est **rechargée** :
- après arrêt et redémarrage du serveur web (le rechargement évite la recompilation de la classe)
- lorsque sa classe est modifiée

Le cookie est conservé

Notons que :
les variables statiques sont partagées par les instances multiples d'une même servlet
les propriétés de session et les classes utilitaires sont partagées par toutes les servlets d'un même moteur de servlets

Les sessions sont différenciées par leur id

SimpleSession.java (1/5)

saisie.jsp

```
<body> <h1>Saisissez un mot</h1>
    <% String mot = "";
       int vu = 0;
       if(session.isNew()) session.setAttribute("compteur",0);
       else  vu = (Integer)session.getAttribute("compteur");
    %>
    <FORM method='GET' action='SimpleSession'>
       <INPUT type='text' name='mot' value="<%=mot%>">
       <INPUT type='submit' value="valider">
    </FORM>
       <P><P><H3>le mot 'vu' a été saisi : <%=vu%>  fois </H3>
  </body>
</html>
```

SimpleSession.java (2/5)

```
public class SimpleSession extends HttpServlet {
    private static final String compteur = "compteur";
    private static final String mot = "vu";
    private static int compteVu = 0;

    public void doGet(HttpServletRequest req,HttpServletResponse res)
            throws ServletException, IOException {
        res.setContentType("text/html");
        PrintWriter pw = res.getWriter();
        RequestDispatcher dispatcher;
        HttpSession laSession = req.getSession(true);
        String vu = req.getParameter("mot");
        if (mot.equals(vu)) {
            if (compteVu == 0) compteVu = 1;
            else compteVu = compteVu + 1;
        }
        laSession.setAttribute(compteur, compteVu);
        dispatcher=req.getRequestDispatcher("JSPSaisie");
        dispatcher.forward(req,res);
    }...
```

SimpleSession.java (3/5)

dans une première fenêtre firefox

SimpleSession.java (4/5)

Puis, on ouvre une seconde fenêtre firefox

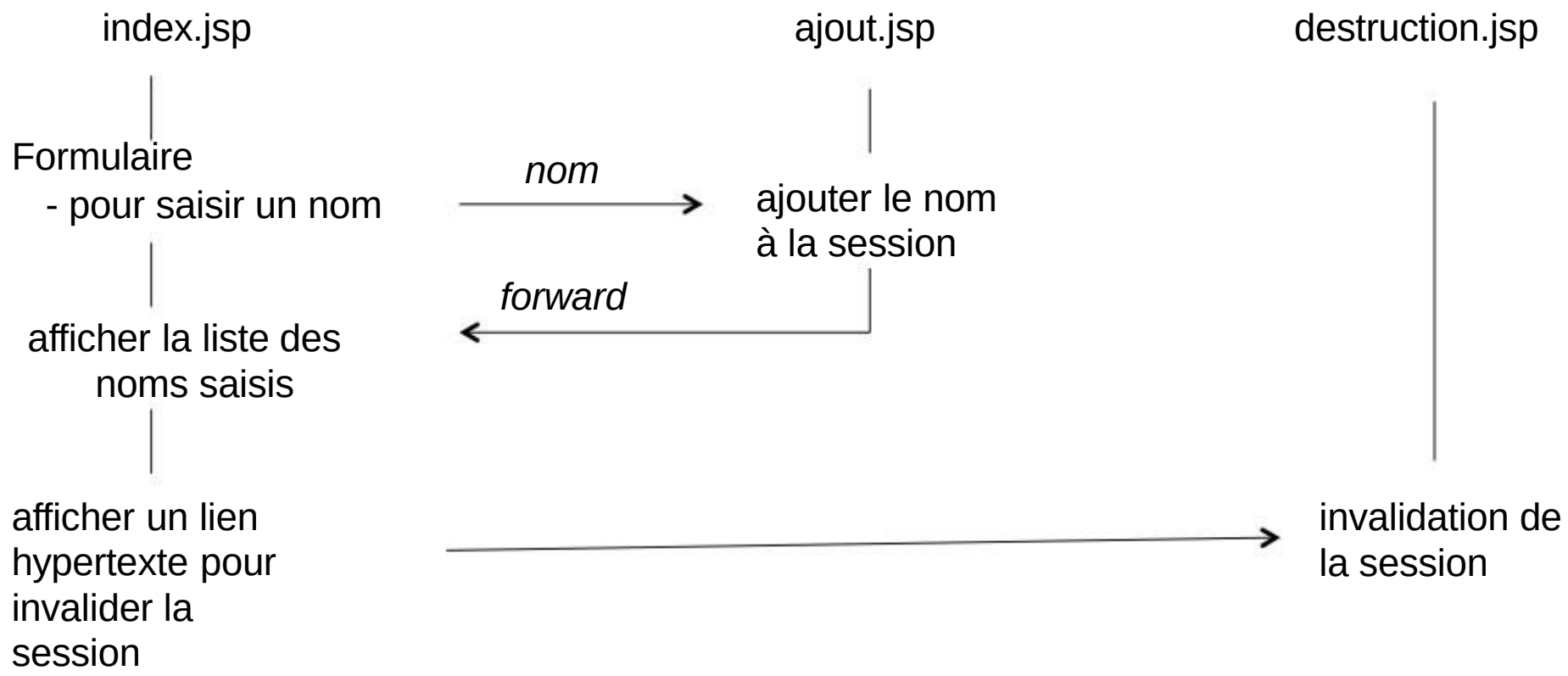
Conservation du cookie

SimpleSession.java (5/5)

On ouvre maintenant une fenêtre safari

Le cookie est perdu !

Exercice


index.jsp
ajout.jsp
destruction.jsp
Formulaire
- pour saisir un nom
nom
ajouter le nom
à la session
forward
afficher la liste des
noms saisis
afficher un lien
hypertexte pour
invalider la
session
invalidation de
la session