Chapitre 5

Gestion des sessions dans les servlets avec HttpSession

5.1 Introduction

Nous avons déjà vu qu'avec la programmation web basée Java, la notion d'utiliser les servlets est très importante, le type de servlet utilisé pour la programmation web c'est bien le httpServlet.

HttpServlet se base sur le protocole HTTP, qui est un protocole de la couche 7 du modèle OSI. Pour rappel, http est un protocole sans état, qui se traduit par l'Impossibilité de garder des informations d'une requête à l'autre (identifier un client d'un autre). C'est vraiment un grand problème si nous voulons faire garder les informations de nos clients en cours d'utilisation de notre projet web.

5.2 Le suivie de session par solutions traditionnelles

Dans la littérature, nous trouvons une multitude de solutions proposées pour palier à ce type de problème. A savoir :

- Une solution basée sur l'authentification de l'utilisateur. En effet, chaque utilisateur doit s'authentifier pour utiliser notre projet web, avec cette opération d'authentification, nous gardon une trace de chaque utilisateur.
  Le point fort de cette solution c'est d'avoir une liste bien claire des utilisateurs qui peuvent accéder à notre projet web. Par contre, la limite de cette solution est que nous obligeons les utilisateurs à s'authentifier pour pouvoir travailler avec notre projet web.
- Une autre solution basée champs de formulaire caché. En effet, pour assurer une communication entre le client et le serveur, avec le protocole http nous faisons appel aux objets *request* et *response*, l'objet *request* englobe les paramètres saisis par le client dans un formulaire, pour les faire passer au serveur. La solution envisagée propose d'incarner des champs <input> de type *hidden* (cachée), dans ces champs nous passons des paramètres au serveur d'une façon totalement cachée et transparente à l'utilisateur. Qu'est-ce que nous devons passer comme paramètre caché ? nous pouvons par exemple créer une variable qui s'incrémente à chaque passage d'un nouvel utilisateur, cette variable est passée comme paramètre caché pour identifier l'utilisateur en cours.
  L'avantage de cette solution c'est la transparence de passage des paramètres cachées. L'utilisateurs reste à l'égard, et ses paramètres saisie vont passer avec les paramètres cachés dans l'objet *request*. Les inconvenants de cette solution sont divers, nous citons par exemple les lignes de code en supplément dans le formulaire du client pour pouvoir assurer cette fonctionnalité. Aussi cette solution doit utiliser un formulaire (clic sur un bouton submit) pour passer d'une page à une autre,
- Une autre solution c'est d'utiliser la réécriture d'URL pour conserver l'état des sessions. Cette solution envisage de faire passer, comme paramètre dans l'URL une paire clé/valeur. Ainsi, le passage de ce paramètre (identificateur) permet de relier un utilisateur donné à ses données stockées sur le serveur.
  Une URL doit ressembler à ceci :
  Lien hypertexte
  Avec cette solution nous arrivons à bien utiliser la notion de session, qui sépare chaque utilisateur des autres par un identifiant. Par contre, les pages stockés dans le serveur doivent être dynamiques vu que l'ID de session a été ajouté dynamiquement, ce qui signifie que toutes les pages du site doivent être dynamiques. Aussi, vu que les sessions

expirent dans le temps, nous pouvons envisager le cas où un utilisateur tiers accède aux données de l'utilisateur à l'aide de l'URL.

- Les navigateurs proposent une autre solution pour le suivie de session, c'est bien l'utilisation de la notion des cookies. En effet, un cookie est une petite information stockée au niveau du client, chaque requête client vers le serveur doit être accompagnée par le cookie de l'utilisateur, donc c'est le serveur qui créer ces cookies qui seront stockés dans le navigateur, et le serveur ne fait que consulter le cookie pour faire associer au client ses propres données (stockés sur le serveur). C'est comme une sorte de chèque de banque, à chaque passage d'un client au guichet de sa banque, il doit présenter son chéquier où s'est mentionné sont numéro de compte, avec ce numéro l'agent de banque pourra lui fournir les services appropriés à son compte.
  Cette solution est la plus intéressante par rapport aux autres pour stocker de petites informations. Toutefois l'utilisation de cookies pose quelques problèmes, nous citons par exemple le cas de la désactivation de l'utilisation des cookies par l'utilisateur par crainte, ou si l'utilisateur utilise un ancien navigateur qui ne gèrent pas les cookies !!!

5.3 Le suivie de session par l'API servlet

L'API servlet nous fournit principalement deux méthodes pour pouvoir gérer les sessions.

5.3.1 Le suivie de session par cookies

La classe javax.servlet.http.Cookie fournie par l'API servlet pour travailler avec les cookies, et oui, nous pouvons travailler avec la notion des cookies même avec les servlets. Le principe c'est le même avec la solution traditionnelle des cookies, ici c'est les servlets qui procèdent à la création de tels cookies.

Pour pouvoir travailler avec cette classe, des méthodes ont étaient proposés par l'API servlet, à savoir :

- Cookie(String name, String value) : qui permet de construire un cookie, donc c'est une methode utilisé par le serveur
- String getName() : qui retourne le nom du cookie
- String getValue() : qui retourne la valeur du cookie
- setValue(String new_value) : qui donne une nouvelle valeur au cookie, c'est dans le cas où nous voulons changer la valeur du cookie en cours
- setMaxAge(int expiry) : qui spécifie l'âge maximum du cookie. En effet, nous pouvons mentionner la durée de vie du cookie si nous voulons une durée de vie inférieur à la durée de vie du navigateur (avant de le quitter). Sinon =-1.

La façon de créer un cookie par l'API servlet est très simple, il suffit que la servlet ajoute à la réponse (HttpServletResponse) ce qui suit :

```
addCookie(Cookie cook1)
```

cette méthode permet d'ajouter à la réponse un cookie.

Et pour que la servlet récupère la valeur des cookies du client, elle exploitant la requête (HttpServletRequest)

```
Cookie[] getCookies()
```

Qui récupère l'ensemble des cookies.

Exemples

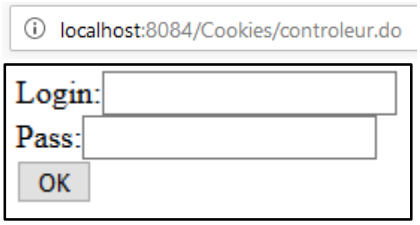
Pour créer un cookie et l'ajouter au client :

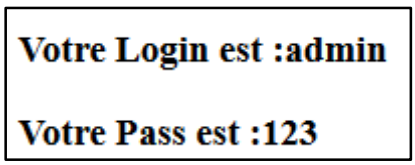
```
Cookie cookies = new Cookie("pass", "123");
cookies.setMaxAge(3 * 24 * 60 * 60); // durée de vie = 3 jours
response.addCookie(cookies);
```

Pour récupérer les cookies :

```
Cookie[] cookies = request.getCookies();
If (cookies != null){
For (int i=0, i<cookies.length; i++){
String name = cookies [i].getName();
String name = cookies [i].get Value();
 }
}
```

Example de fonctionnement





Login.html

```
<html>
  <head>
    <title>Authentification</title>
  </head>
  <body>
    <form action="controleur.do" method="post">
      Login:<input type="text" name="login"><br/>
      Pass:<input type="password" name="pass"><br/>
      <input type="submit" value="OK">
    </form>
  </body>
</html>
```

Servlet

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

public class FirstServlet extends HttpServlet {
```

```
    @Override
    protected void doGet(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {
            System.out.println("aa");

        Cookie[] cookies = request.getCookies();
        String login = null, pass = null;
        for (Cookie c : cookies) {
            if (c.getName().equals("login")) {
                login = c.getValue();
            }
            if (c.getName().equals("pass")) {
                pass = c.getValue();
            }
        }
            System.out.print(pass);
        if ((login != null) && (pass != null)) {
            request.setAttribute("login", login);
            request.setAttribute("pass", pass);
            request.getRequestDispatcher("Menu.jsp").forward(request, response);
        } else {
            System.out.println("b");
            request.getRequestDispatcher("Login.html").forward(request, response);
        }
    }

    @Override
    protected void doPost(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {
        String log = request.getParameter("login");
        String pass = request.getParameter("pass");
        Cookie cLog = new Cookie("login", log);
        cLog.setMaxAge(-1);
        /*le cookie sera valide jusqu'a la fermeutre du navigateur*/
        Cookie cPass = new Cookie("pass", pass);
        cPass.setMaxAge(2 * 24 * 60 * 60);
        response.addCookie(cLog);
        response.addCookie(cPass);
        request.setAttribute("login", log);
        request.setAttribute("pass", pass);
        request.getRequestDispatcher("Menu.jsp").forward(request, response);
    }
}
```

Menu.jsp

```
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
        <title>JSP Page</title>
    </head>
    <body>
```

```
        <h3>Votre Login est :<%=request.getAttribute("login")%></h3>
        <h3>Votre Pass est :<%=request.getAttribute("pass")%></h3>
    </body>
</html>
```

5.3.2 Le suivie de session par HttpSession

Comme nous l'avons déjà mentionné, l'utilisation des cookies se limite par les navigateurs qui ne les acceptent pas toujours. Aussi un simple clique d'un utilisateur pourra suspendre l'utilisation des cookies dans son navigateur. Et même si les deux cas de figures précédemment cités sont valides, Les navigateurs n'acceptent que 20 cookies par site, 300 par utilisateur et la taille d'un cookie peut être limitée à 4096 octets.

D'où la nécessité de penser autrement avec les servlets, pour ne pas arriver à une situation d'handicape. En effet, l'API servlet a pensé d'inclure une nouvelle façon de suivre les sessions, c'est bien avec HttpSession.

L'objet HttpSession sert à sauvegarder les paramètres saisis par l'utilisateur dans une sorte de table de correspondance entre chaque id de session avec l'ensemble des informations relatives à l'utilisateur.

Création ou récupération d'une session (voir volet pratique)

Pour créer une session, les méthodes utilisées liées à l'objet *request* (HttpServletRequest) sont les suivantes :

– HttpSession getSession() : retourne la session associée à l'utilisateur, si aucune session n'est associée, création d'une nouvelle session

– HttpSession getSession(boolean p) : création selon la valeur de p

Gestion des données de l'utilisateur dans une session (voir volet pratique)

Pour stocker les données dans la session il suffit d'utiliser la méthode setAttribute() en lui passant comme attributs la clé et la valeur associée.

```
Object setAttribute("cle","valeur")
```

Pour récupérer une valeur sauvegardée dans l'objet HttpSession, il suffit d'utiliser la méthode :

```
Object getAttribute("cle")
```

Qui retourne un objet, il faut donc effectuer un surtypage pour obtenir un type particulier de données.

Nous pouvons aussi supprimer un attribut déjà lié à une session par la méthode :

```
removeAttribut(String na)
```

qui supprime l'attribut associé à na

Destruction d'une session (voir volet pratique)

Dans certains cas, il peut être utile de supprimer manuellement une session. Pour supprimer une session, il suffit de faire appel à la méthode invalidate() de l'objet HttpSession

Exemple explicatif pour l'utilisation de session

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

public class MyServlet2 extends HttpServlet {

   public void doGet(HttpServletRequest request,
HttpServletResponse response) {
      try {
         response.setContentType("text/html");
         PrintWriter pwriter = response.getWriter();
         HttpSession session = request.getSession(false);
         Integer count = (Integer) session.getAttribute("count");
         if (count == null) {
            count = new Integer(1);
         } else {
            count = new Integer(count.intValue() + 1);
         }
         session.setAttribute("count", count);
         pwriter.print("Vous avez visité cette page " + count + " fois.");
         pwriter.close();
      } catch (Exception exp) {
         System.out.println(exp);
      }
   }
}
```