Servleti - gestionarea cookie-urilor și a sesiunilor

Obiective

- Ce sunt cookie-urile
- Tipuri de cookie-uri: sesiune și persistente
- Rescrierea URL-ului („URL rewrite”)
- Câmpuri ascunse („hidden fields”)
- Monitorizarea sesiunilor pe server – HttpSession
- Stocarea valorilor simple și a listelor de valori

1. Ce sunt cookie-urile.

Protocolul HTTP este unul de tip cerere răspuns, în care noțiunea de stare nu este definita explicit. Astfel, comunicarea este una fără stare deși sunt foarte multe situații în care conceptul de sesiune ar fi util. Sesiunile sunt necesare pentru a stoca informații referitoare la cererile anterioare ale browserului.

Exemple de situații in care este necesara memorarea starii in comunicarea cerere-raspuns:

- Coșul de cumparaturi pentru un site on-line de e-commerce

-Păstrarea local, pe client, a informațiilor de autentificare, evitand astfel introducerea de fiecare data a acestora

- Personalizarea unui site conform preferintelor diverșilor utilizatori (“CRM – Custom Relationship Management”)
- Publicitate focalizata pe preferințele utilizatorilor

Sesiunea poate fi vazuta ca o memorie comuna intre client si server. Astfel, se asigneaza un identificator unic fiecărui client. Pentru fiecare identificator de pe client corespunde un obiect de tip sesiune pe server.

Exista mai multe metode de monitorizarea sesiunilor:

- Cookies
- Rescrierea URL (“URL-Rewriting”)
- Campurile ascunse în formulare HTML (“Hidden Form Fields”)

Cookie-urile sunt fișiere salvate pe client care sunt trimise pe server la fiecare conexiune. Sunt incluse în headerele cererilor HTTP.

Cookie-urile funcționează după următorul mecanism:

- Servletul trimite un nume și o valoare la browserul client
- Browserul trimite același nume și aceeași valoare cand se conecteaza la același site sau domeniu (depinde de setările din cookie)

Exista probleme legate de confidentialitate (“privacy”) la folosirea cookie-urilor:

- Serverul poate stoca toate acțiunile anterioare
- Dacă se introduc informații personale acestea pot fi legate de acțiunile anterioare
- Aceste informații pot fi partajate cu alte parti

Pentru a crea un cookie pe un client:

```
Cookie myCookie = new Cookie("nume", "MyCookie");
```

Se poate seta durata de viata a unui cookie:

```
myCookie.setMaxAge(60); //one minute
```

Se plasează cookie-ul pe răspunsul HTTP:

```
response.addCookie(myCookie);
```

Pentru a recupera cookie-urile trimise de un client se folosește:

```
Cookie[] cookies = request.getCookies();
```

Afișarea cookie-urilor trimise pe cerere („request)":

```
Cookie[] cookies = request.getCookies();
if (cookies == null) {
      out.println("<TR><TH COLSPAN=2>No cookies");
 } else {
      for(Cookie cookie: cookies) {
        out.println
          ("<TR>\n" +
           "  <TD>" + cookie.getName() + "\n" +
           "  <TD>" + cookie.getValue());
        }
       }
```

Afișarea mesajelor de întâmpinare în funcție de accesarea unei resurse prima data și ulterior:

```
boolean newbie = true;
Cookie[] cookies = request.getCookies();
if (cookies != null) {
      for (Cookie c : cookies) {
            if ((c.getName().equals("repeatVisitor")) &&
            // Could omit test and treat cookie name as a flag
                  (c.getValue().equals("yes"))) {
                        newbie = false;
                        break;
                  }
            }
      }

      String title;
      if (newbie) {
            Cookie returnVisitorCookie = new Cookie("repeatVisitor", "yes");
            returnVisitorCookie.setMaxAge(60 * 5); // 5 minutes
            response.addCookie(returnVisitorCookie);
            title = "Bine ai venit prima data!";
      } else {
            title = "Bine ai revenit!";
      }
```

2. Atributele unui cookie și tipurile de cookie-uri.

Atributele unui cookie se pot folosi cu următoarele metode:

- *getName:* se obține numele unui cookie, nume setat prin constructor la crearea acestuia
- *getValue/setValue:* citirea/scrierea valorii asociate cu un cookie; de obicei valoarea unui cookie se seteaza prin constructor și nu prin *setValue;*
- *getDomain/setDomain:* se poate specifica domeniul pe care se aplica cookie-ul; mașina host trebuie să aparțină domeniului specificat;
- *getPath/setPath:* citește/seteaza calea la care se aplica cookie-ul; dacă nu se specifica se aplica URL-ului curent sau în cadrul directorului ce contine pagina curentă;
- *getSecure/setSecure:* citește/seteaza flagul care indica dacă cookie-ul se aplica doar conexiunilor SSL sau tuturor conexiunilor;
- *getMaxAge/setMaxAge:* citirea/scrierea timpului de expirare în secunde a unui cookie

În mod predefinit un cookie este persistent doar în sesiunea curentă, cât timp browserul este activ. Acest tip de cookie se numește ***cookie sesiune.*** Acest cookie are atributul *MaxAge* setat la valoarea -1. Dacă valoarea acestui atribut este una are (cateva ore, zile, saptamani etc), atunci acesta se numește ***cookie persistent***. El exista și după inchiderea browserului sau a calculatorului.

Pentru a șterge un cookie se setează valoarea *MaxAge* pe 0.

După schimbarea valorii unui atribut al cookie-ului, acesta trebuie reatasat răspunsului pentru a fi valid:

response.addCookie(myCookie);

Exemplu de metoda utilitara care cauta un cookie dupa nume:

```
public static String getCookieValue(HttpServletRequest request,
                String cookieName, String defaultValue) {
        Cookie[] cookies = request.getCookies();
        if (cookies != null) {
               for (Cookie cookie : cookies) {
                      if (cookieName.equals(cookie.getName())) {
                             return (cookie.getValue());
                      }
               }
        }
        return (defaultValue);
  }
```

Exemplu de clasa utilitara care furnizeaza o instanță de cookie persistent:

```
public class LongLivedCookie extends Cookie {
  public static final int SECONDS_PER_YEAR = 60*60*24*365;

  public LongLivedCookie(String name, String value) {
    super(name, value);
    setMaxAge(SECONDS_PER_YEAR);
  }
}
```

Exemplu de servlet care contorizează accesul utilizatorilor:

```
  public void doGet(HttpServletRequest request, HttpServletResponse response)
      throws ServletException, IOException
{
    String countString = CookieUtilities.getCookieValue(request, "accessCount", "1");
    int count = 1;
    try {
      count = Integer.parseInt(countString);
    } catch(NumberFormatException nfe) { }

    LongLivedCookie c =
      new LongLivedCookie("accessCount",
                          String.valueOf(count+1));
    response.addCookie(c);
}
```

3. Metodele de tip „URL rewriting” și „hidden fields”.

Protocolul HTTP este unul de tip cerere răspuns, în care noțiunea de stare nu este definita explicit. Astfel, comunicarea este una fără stare deși sunt foarte multe situații în care conceptul de sesiune ar fi util. Sesiunile sunt necesare pentru a stoca informații referitoare la cererile anterioare ale browserului.

Exista mai multe metode de monitorizarea sesiunilor:

- Cookies
- Rescrierea URL (“URL-Rewriting”)
- Campurile ascunse în formulare HTML (“Hidden Form Fields”)

Metoda de rescrierea URL-ului consta în modificarea acestuia în mod dinamic, prin adaugarea unor parametri, sub forma:

```
url?name1=value1&name2=value2
```

Avantajele folosirii metodei de rescrierea URL-ului (“URL rewriting”):

- Metoda funcționează chiar dacă browserele au dezactivate cookie-urile
- Nu este necesar alt apel suplimentar în paginile respective

Dezavantaje:

- Metoda funcționează numai prin referințe (“links”)
- Se pot trimite doar informații de tip text

Exemplu de cod care folosește metoda URL rewriting:

```
@WebServlet("/first-urlwrite")
public class FirstServlet extends HttpServlet {

public void doGet(HttpServletRequest request, HttpServletResponse response){
          try{

          response.setContentType("text/html");
          PrintWriter out = response.getWriter();
```

```
            String n=request.getParameter("userName");
            out.print("Welcome "+n);

            //appending the username in the query string
            out.print("<a href='second-urlwrite?uname="+n+"'>visit</a>");
            out.close();
          }catch(Exception e){System.out.println(e);}
    }
}

@WebServlet("/second-urlwrite")
public class SecondServlet extends HttpServlet {

public void doGet(HttpServletRequest request, HttpServletResponse response){
          try{

          response.setContentType("text/html");
          PrintWriter out = response.getWriter();

          //getting value from the query string
          String n=request.getParameter("uname");
          out.print("Hello "+n);

          out.close();

         }catch(Exception e){System.out.println(e);}
    }
}
```

Alta metoda de a monitoriza sesiunile o reprezinta campurile ascunse sau invizibile (“hidden form field”). În acest caz, informațiile sunt stocate în aceste câmpuri invizibile de unde pot fi transmise la alt servlet.

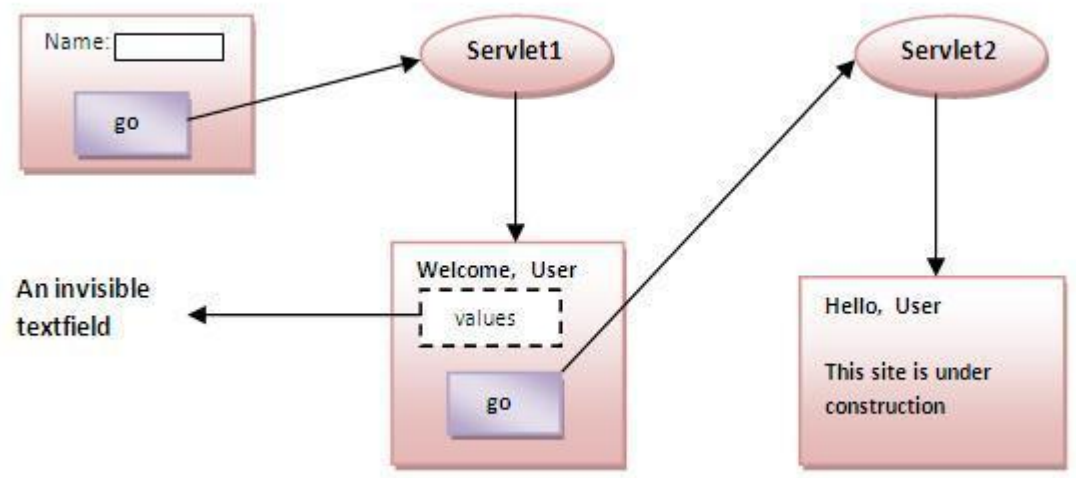
Avantajele folosirii metodei de monitorizarea sesiunilor prin campuri ascunse:

- Metoda funcționează chiar dacă browserele au dezactivate cookie-urile

Dezavantaje:

- Este gestionata la nivel de server
- Sunt necesare apeluri suplimentare în fiecare pagina
- Se pot trimite doar informații in format text

Exemplu de monitorizare a informațiilor de sesiune prin intermediul metodei de câmpuri ascunse:



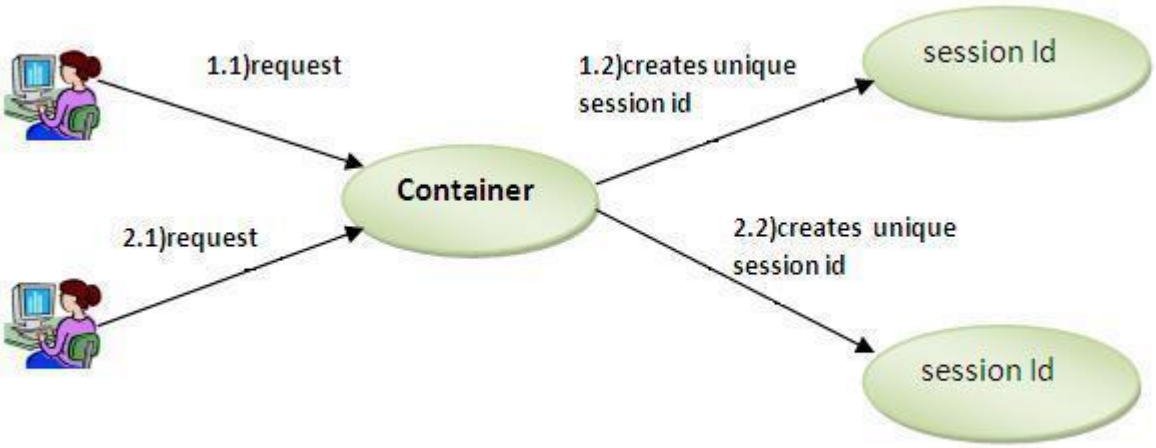
Exemplu de cod prin metoda campurilor ascunse:

```
@WebServlet("/first-hidden")
public class FirstServlet extends HttpServlet {
public void doGet(HttpServletRequest request, HttpServletResponse response){
            try{
            response.setContentType("text/html");
            PrintWriter out = response.getWriter();
            String n=request.getParameter("uname");
            out.print("Welcome "+n);

            //creating form that have invisible textfield
            out.print("<form action='welcome'>");
            out.print("<input type='hidden' name='uname' value='"+n+"'>");
            out.print("<input type='submit' value='go'>");
            out.print("</form>");

            out.close();
              }catch(Exception e){System.out.println(e);}
      }
}

@WebServlet("/second-hidden")
public class SecondServlet extends HttpServlet {
public void doGet(HttpServletRequest request, HttpServletResponse response){
            try{

            response.setContentType("text/html");
            PrintWriter out = response.getWriter();

            //Getting the value from the hidden field
            String n=request.getParameter("uname");
            out.print("Hello "+n);

            out.close();
            }catch(Exception e){System.out.println(e);}
      }
}
```

4. Monitorizarea sesiunilor pe server – interfața HttpSession

O alta metoda de a monitoriza informațiile de tip sesiune o reprezinta asocierea unui id pentru fiecare utilizator. Astfel, containerul folosește acest id pentru datele utilizatorilor, prin intermediul unor obiecte de tipul HttpSession. Un obiect de tipul HttpSession poate executa următoarele task-uri:

- Legarea obiectelor
- Vizualizarea și gestionarea informațiilor despre sesiune, cum ar fi identificatorul de sesiune, data creării, data ultimului acces.



Principala metoda prin care se returnează sesiunea asociată cu cererea curentă, iar dacă nu exista vreo sesiune se creaza una: *getSession()*.

Alte metode importante ale interfeței HttpSession sunt:

- *getAttribute:* extragerea valorii stocate anterior într-un obiect de tip sesiune; se returnează null dacă nu a fost asociat nici o valoare pentru numele dat;
- *setAttribute:* se asociază o valoare cu un nume pentru obiectul de tip sesiune;
- *removeAttribute:* șterge valoarea asociată cu un nume;
- *getAttributeNames:* returnează numele atributelor stocate în sesiune;
- *getId:* returnează identificatorul unic;
- *getCreationTime:* returnează data creării sesiunii;
- *getLastAccessedTime:* ultima data cand informații despre sesiune au fost trimise de pe client;
- *invalidate:* invalidează sesiunea curentă

Exemplu de folosire *HttpSession:*

```
@WebServlet("/first-session")
public class FirstServlet extends HttpServlet {

public void doGet(HttpServletRequest request, HttpServletResponse response){
            try{
            response.setContentType("text/html");
            PrintWriter out = response.getWriter();
```

```
                String n=request.getParameter("userName");
                out.print("Welcome "+n);
                HttpSession session=request.getSession();
                session.setAttribute("uname",n);
                out.print("<a href='second-session'>visit</a>");
                out.close();

          }catch(Exception e){System.out.println(e);}
        }
    }

    @WebServlet("/second-session")
    public class SecondServlet extends HttpServlet {

    public void doGet(HttpServletRequest request, HttpServletResponse response){
          try{

          response.setContentType("text/html");
          PrintWriter out = response.getWriter();

          HttpSession session=request.getSession(false);
          String n=(String)session.getAttribute("uname");
          out.print("Hello "+n);
          out.close();

          }catch(Exception e){System.out.println(e);}
    }
    }
```

5. Exemple de utilizare a sesiunilor.

Afișarea numărului de acces a unei pagini de un utilizator:

```
@WebServlet("/show-session")
public class ShowSession extends HttpServlet {
  @Override
  public void doGet(HttpServletRequest request, HttpServletResponse response)
      throws ServletException, IOException {
    response.setContentType("text/html");
    HttpSession session = request.getSession();
    synchronized(session) {
      String heading;
      Integer accessCount =
        (Integer)session.getAttribute("accessCount");
      if (accessCount == null) {
        accessCount = 0;
        heading = "Bine ai venit prima data!";
      } else {
        heading = "Bine ai revenit!";
        accessCount = accessCount + 1;
      }

    session.setAttribute("accessCount", accessCount);
  ... }
```

Afișarea unei liste de valori stocate pe sesiune:

```
@WebServlet("/show-items")
public class ShowItems extends HttpServlet {
  @Override
  public void doPost (HttpServletRequest request, HttpServletResponse response)
      throws ServletException, IOException {
    HttpSession session = request.getSession();
    synchronized(session) {
      @SuppressWarnings("unchecked")
      List<String> previousItems = (List<String>)session.getAttribute("previousItems");
      if (previousItems == null) {
        previousItems = new ArrayList<String>();
      }
      String newItem = request.getParameter("newItem");
      if ((newItem != null) && (!newItem.trim().equals(""))) {
        previousItems.add(newItem);
      }
      session.setAttribute("previousItems", previousItems);
      response.setContentType("text/html");
      PrintWriter out = response.getWriter();

      String title = "Items Purchased";
      String docType =
        "<!DOCTYPE HTML PUBLIC \"-//W3C//DTD HTML 4.0 " +
        "Transitional//EN\">\n";
      out.println(docType +
                  "<HTML>\n" +
                  "<HEAD><TITLE>" + title + "</TITLE></HEAD>\n" +
                  "<BODY BGCOLOR=\"#FDF5E6\">\n" +
                  "<H1>" + title + "</H1>");
      if (previousItems.size() == 0) {
        out.println("<I>No items</I>");
      } else {
        out.println("<UL>");
        for(String item: previousItems) {
          out.println("  <LI>" + item);
        }
        out.println("</UL>");
      }
      out.println("</BODY></HTML>");
    }
}
```

Serverele web au capabilități prin care sesiunile pot fi distribuite pe mai multe mașini și pot fi persistente pe hard-disk. Astfel, pentru serverele web distribuite, în funcție de încărcarea unui server, cererile pot fi redirecționate pe alta masina. In acest caz, sesiunea creata la cererile anterioare este recunoscută și pe noul server, printr-un mecanism de replicare internă.

Sesiunile pot fi persistate pe hard-disc care apoi sunt reincarcate la restartarea serverului. Pentru a implementa astfel de sesiuni persistente, clasele trebuie sa implementeze interfața *java.io.Serializable.*

6. Exerciții.

1) Creați un fișier *index.html*, care sa conțină referințe spre rezolvarile problemelor de mai jos. Creați un servlet care sa afiseze următoarele mesaje: la primul acces al servletului se va afișa mesajul `"Bine ai venit prima data!",`
la următoarele accesari ale servletului se va afișa mesajul `"Bine ai revenit!"`.

2) În fișierul *index.html* adaugati o referință către o pagina care afișează un tabel cu 4 cookie-uri, 2 la nivel de sesiune și 2 persistente (3 minute). Testați în browsere diferite în diverse situații (apeluri succesive, restartare browsere etc).

3) În fișierul *index.html* adaugati o referință către un servlet care afișează numărul de accesări ale unei pagini web .Testați în diverse scenarii servletul.

4) În fișierul *index.html* adaugati o referință către un servlet care sa afiseze mesajul „Pagina 1” și care sa conțină o referinta spre un servlet care afișează pagina doi cu mesajul „Pagina 2”. Pagina 2 nu poate fi apelată înainte de a se apela pagina 1. Folosiți cookie-uri.

5) Creați doi servleti pentru afișarea valorii introduse într-un formular prin metoda „hidden fields”.

6) Sa se afiseze tabelar și următoarele informații: id-ul sesiunii, data creării sesiunii, data ultimului acces, numărul cumulat de accese la pagina a utilizatorului.

7) În fișierul *index.html* adaugati o referință către un servlet prin care se introduc numele, prenumele și emailul unei persoane. Creați un servlet care sa afișeze numele parametrilor și a valorilor acestora. Dacă o valoare nu este introdusă și este primul acces al utilizatorului, atunci se va afișa o valoare predefinita (ex: email lipsa). Dacă utilizatorul a mai accesat pagina atunci se vor folosi valorile adaugate anterior. Folosiți cookie-uri, apoi sesiuni.

8) Creati un formular in care sa comandati un produse. Apelați din formular un servlet care sa tina o lista cu toate elementele (produsele) comandate și care sa le afișeze în browser. Lista poate fi resetata de la un buton. Se va folosi Cookie și apoi HttpSession.

9) În fișierul *index.html* adaugati o referință către un servlet care are un titlu și afișează un text. Predefinit, culoarea textului este roșie iar culoarea background-ului galben. Pagina contine și un link spre alta pagina prin care se pot seta cele doua culori de către utilizator. Folosiți Cookie și apoi HttpSession.