Lezione 8 Cookie e JSP

Cookies

I cookies sono delle informazioni che il server invia al client nella sezione header della risposta http. Il client a seconda del tipo di cookie memorizza sul disco o meno questa informazione. Ad ogni richiesta fatta verso lo stesso server, il client nello header della richiesta ripropone i cookie che precedentemente il server gli ha inviato.
Ci sono dei limiti alla dimensione dei cookie e al numero di cookie che un server può inviare.
I cookie vengono usati dal motore delle servlet per gestire le sessioni e quindi risolvere tutti quei problemi legati alla sessione. Altre applicazione dei cokies sono:

- Personalizzazione di un sito in base alle pagine precedentemente visitate;
- Pubblicità mirata alle pagine precedentemente visitate;
- Eliminazione di username e password per l'accesso al sito;

Altro vantaggio offerto dai cookie è la possibilità di gestire informazioni senza bisogno di memorizzarle sul server stesso, ma in maniera distribuita sui client. Nei cookie si memorizzano informazioni che possono essere ricavate in altri modi, sono di solito una scorciatoia.
I cookie sono rappresentati da un nome e un valore:
Il nome deve rispettare le regole dettate in RFC 2109. Questo significa che esso può contenere solo caratteri ASCII alfanumerici esclusi virgole, punti e virgola e spazi, inoltre non possono iniziare col simbolo '$'.

```
...
Set-Cookie: CFID=145112387;domain=.unive.it;expires=Sun, 22-Feb-2043 08:39:00
GMT;path=/
Set-Cookie: CFTOKEN=38040093;domain=.unive.it;expires=Sun, 22-Feb-2043 08:39:00
GMT;path=/
Set-Cookie: JSESSIONID=703011da1cd525641e60;path=/
Set-Cookie: OP_NEWVISIT=0;path=/
Set-Cookie: PUBLIC_NEWVISIT=0;path=/
...
Cookie: JSESSIONID=703011da1cd525641e60; name2=value2
```

Nelle API Java, il nome del cookie non può essere cambiato dopo la sua creazione. Il valore del cookie non ha vicoli sul contenuto.
I metodi principali degli oggetti della classe javax.servlet.http.Cookie sono:

```
public void setComment(String s);
public String getComment();
```

Impostano e leggono il valore del commento al cookie. Il commento viene visualizzato dal browser nel caso chieda conferma se accettare o meno il cookie;

```
public void setVersion(int c);
public int getVersion ();
```

Impostano e leggono il valore della versione come sotto:
c== 0: Netscape standard
c== 1: RFC 2109

```
public void setMaxAge(int c);
public int getMaxAge();
```

Impostano e leggono il tempo di vita del cookie. Quando il cookie arriverà al browser verrà intrepresa una delle seguenti:
c >0: lo terrà in vita per c secondi;

c==0: elimiare il cookie;
c<0: lo terrà in vita finché dura la sessione del browser.

```
public void setDomain(String d);
public String getDomain();
```
Impostano e leggono il dominio per il quale i browser deve presentare il cookie quando effettua una richiesta. Di default il cookie viene presentato solo al dominio dal quale lo ha ricevuto.

```
public void setPath(int c);
public int getPath();
```
Impostano e leggono il path nel dominio per il quale i browser deve presentare il cookie quando effettua una richiesta.

Codice Esempio:

```
Cookie coo = new Cookie("nome","Alessandro");
userCookie.setMaxAge(60*60*24*365);
response.addCookie(userCookie);
```

Esempio impostazione cookie

Il seguente esempio imposta due cookie: uno che dura per un giorno e uno che dura finché il browser rimarrà aperto.

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

public class TestSet extends HttpServlet {
  public void doGet(HttpServletRequest request, HttpServletResponse
response)
              throws ServletException, IOException {


    Cookie cookie = new Cookie("Session-Cookie","Cookie-Value-S");
    response.addCookie(cookie);

    cookie = new Cookie("Persistent-Cookie","Cookie-Value-P");

    cookie.setMaxAge(60*60*24);
    response.addCookie(cookie);

    response.setContentType("text/html");
    PrintWriter out = response.getWriter();
    out.println (("<HTML><HEAD><TITLE>Impostazione
Cookie</TITLE></HEAD>" +
       "<BODY><H1 >Impostazione Cookie</H1>Prova impostazione 2
cookie</BODY></HTML>");
  }
}
```

Esempio lettura cookie

Il seguente esempio legge tutti i cookie presentati dalla richiesta.

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

```
public class ShowCookies extends HttpServlet {
  public void doGet(HttpServletRequest request, HttpServletResponse
response)
              throws ServletException, IOException {
    response.setContentType("text/html");
    PrintWriter out = response.getWriter();
    out.println(("<HTML><HEAD><TITLE>Lettura Cookie</TITLE></HEAD>" +
                "<BODY><H1>Lettura Cookie</H1>" +
                "<TABLE>\n" +
                "<TR>\n" +
                "  <TH>Nome</TH><TH>Valore</TH>");
    Cookie[] cookies = request.getCookies();
    for(int i=0; i<cookies.length; i++)
      out.println("<TR><TD>" + cookie[i].getName() + "</TD><TD>" +
cookie[i].getValue()+"</TD></TR>\n");
    out.println("</TABLE></BODY></HTML>");
  }
}
```

Esempio uso cookie per gesione sessioni

```
String sessionID = makeUniqueString();
Hashtable sessionInfo = new Hashtable();
Hashtable globalTable = getTableStoringSession();
globalTable.put(sessionID,sessionInfo);
Cookie sessionCookie=new Cookie(“SessionID”,sessionID);
sessionCookie.setPath("/");
response.addCookie(sessionCookie);
```

JSP

Le Java Server Page (JSP) sono una alternativa alle Servlet quando la parte di presentazione (codice HTML) è preponderante rispetto al codice JAVA necessario per la generazione della parte dinamica. In un primo momento vedremo le JSP come alternativa alle servlet, successivamente vedremo invece come sfruttare i benefici di entrambi all'interno della stessa richiesta.
Ad esempio la servlet che stampa la data odierna viene più convenientemente
Esempio:

```
<HTML>
<BODY>
<H1>
<%= new java.util.Date()
%>
</H1>
</BODY>
</HTML>
```

Le JSP sono implementate dal motore delle servlet generando automaticamente il codice Java di una servlet che esegue le operazioni necessarie a stampare il codice HTML presente nella pagina JSP. Inoltre include il codice java presente nella pagina JSP nella giusta posizione nella servlet.
Ad ogni richiesta di una pagina JSP il motore delle servlet verifica se la pagina è stata modificata e in questo caso crea e compila la servlet corrispondente.
Il codice Java presente in una pagina JSP si divide in:

- Direttive: `<%@ codice %>`;
- Dichiarazioni: `<%! codice %>`;
- Scriptlets: `<% codice %>`;

- Azioni Standard: `<jsp:azione />`;
- Espressioni: `<%= codice %>`;

Trasformazione da JSP a Servlet:

- Direttive => nel file della servlet;
- Dichiarazioni => attributi e metodi della servlet;
- Scriptlets e azioni standard => codice nel metodo `doGet`;
- Espressioni => argomenti di `out.print` nel metodo `doGet`.

Esempio (puramente indicativo):

JSP file prova.jsp	Servlet file prova.jsp$32342.java

```
<%@ page import="java.util.*"%>



<%! private String htmlencode(String s)
{
   int i;
   ...
}
    MyClass obj = new MyClass();%>




<html><body>
<%= request.getParameter("nome")%>

<%
if("store".equals(request.getParameter("op
")))
      obj.storeDb(request);
%>

</body></html>
```

```
import java.util.*;
public class prova.jsp$32342 extends
HttpServlet
{
   private String htmlencode(String s)
   {
      int i;
      ...
   }

    MyClass obj = new MyClass();
  public void doGet(HttpRequest request,
HttpResponse response)
{
... inizializzazione;
PrintWriter out = request.getWriter();
out.print("<html><body>\n");

out.print(request.getParameter("nome"));
out.print("\n\n");

if("store".equals(request.getParameter("op
")))
        obj.storeDb(request);

out.print("</body></html>\n");
   }
}
```