Tener traccia del client

- Raramente un'applicazione web è costituita da una singola pagina (risorsa).
- E' utile quindi tener traccia dei client che si collegano per rendere più semplice lo sviluppo dell'applicazione.
- Tuttavia il protocollo HTTP è "stateless":
  - il client fa una richiesta;
  - il server soddisfa (se sono verificate certe condizioni) la richiesta;
  - l'interazione client/server viene "dimenticata" e si passa a considerare la richiesta successiva.

Tener traccia del client

- Le soluzioni utilizzabili per tener traccia dello stato del client nelle varie pagine di un'applicazione sono:
  - inserire informazioni inerenti allo stato direttamente negli URL (soluzione complessa e intrinsecamente insicura);
  - inserire informazioni sullo stato in appositi campi hidden nei form (soluzione insicura);
  - cookie;
  - utilizzare il meccanismo delle sessioni (server-side).

I cookie

- I cookie consentono ad un'applicazione web di memorizzare piccole quantità di dati sui client.
- L'utilizzo tipico dei cookie è quello di consentire di identificare e tracciare il client.
- L'informazione scambiata fra server e client tramite i cookie viene inglobata negli header delle richieste/risposte definite dal protocollo HTTP.

Supporto per i cookie nelle servlet

- L'API delle servlet fornisce la classe **`javax.servlet.http.Cookie`** che rende facile al programmatore la manipolazione dei cookie.
- **`HttpServletResponse`** mette a disposizione il metodo **`addCookie()`** per creare un nuovo cookie.
- **`HttpServletRequest`** fornisce il metodo **`getCookies()`** per leggere dall'header della richiesta HTTP i cookie.

Scrivere un cookie

- Siccome i cookie vengono inviati nell'header della risposta HTTP, possono essere creati soltanto prima che qualsiasi altro contenuto sia inviato al client.
- Un browser è tenuto ad accettare al massimo 20 cookie per sito e 300 in totale per ogni utente; inoltre può limitare la lunghezza dei cookie a 4.096 byte.
- Esempio (**`createUserID`** è una funzione definita dall'utente):

```
String userid = createUserID();
Cookie c = new Cookie("userid", userid);
res.addCookie(c);
```

Leggere un cookie

- Tramite il metodo **`getCookies()`** si accede all'intero insieme dei cookie.
- Per leggere il valore di uno specifico cookie, bisogna quindi scorrere tutti i cookie:

```
Cookie[] cookies;
cookies = req.getCookies();
String userid = null;
for (int i = 0; i < cookies.length; i++)
    if (cookies[i].getName().equals("userid"))
                userid = cookies[i].getValue();
```

Proprietà dei cookie

- I cookie possono essere inizializzati in modo da risultare validi soltanto in un certo dominio, in un certo percorso e per un certo tempo:

```
String userid = createUserID();
Cookie c = new Cookie("userid",userid);
c.setDomain(".company.com");
c.setPath("/");
c.setMaxAge(900); // in secondi
resp.addCookie(c);
```

Esempio: scrittura di un cookie

```
import javax.servlet.*;
import javax.servlet.http.*;
import java.io.*;
public class ScriviCookie extends HttpServlet {
  public void doGet(HttpServletRequest req, HttpServletResponse resp)
              throws ServletException, IOException {
    Cookie c = new Cookie("cookie_test",
                          "cookie impostato dalla servlet ScriviCookie");
    resp.addCookie(c);
    PrintWriter out = resp.getWriter();
    resp.setContentType("text/html");
    out.println("<HTML><HEAD><TITLE>Scrittura di un " +
                "cookie</TITLE></HEAD>");
    out.println("<BODY>Questa servlet ti ha appena inviato un cookie");
    out.println("</BODY></HTML>");
  }
}
```

Modifiche da apportare al file `web.xml`

```
<?xml version="1.0" encoding="ISO-8859-1"?>

<!DOCTYPE web-app
    PUBLIC "-//Sun Microsystems, Inc.//DTD Web Application 2.3//EN"
    "http://java.sun.com/dtd/web-app_2_3.dtd">

<web-app>
   ...
   <servlet>
        <servlet-name>ScriviCookie</servlet-name>
        <servlet-class>ScriviCookie</servlet-class>
    </servlet>
    <servlet-mapping>
        <servlet-name>ScriviCookie</servlet-name>
        <url-pattern>/servlet/ScriviCookie</url-pattern>
    </servlet-mapping>
</web-app>
```

Modifiche da apportare al file `index.html`

```
<HTML>
<HEAD>
<TITLE>Servlet di prova</TITLE>
</HEAD>

<BODY>
  <TABLE>
    ...
    <TR>
      <TD>
        <A HREF="servlet/ScriviCookie">Scrittura di un cookie</A>
      </TD>
    </TR>
  </TABLE>
</BODY>
</HTML>
```

Lettura dei cookie (I)

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

public class LeggiCookie extends HttpServlet {

  public void doGet(HttpServletRequest req, HttpServletResponse resp)
              throws ServletException, IOException {
    Cookie[] cookies;
    cookies = req.getCookies();
    PrintWriter out = resp.getWriter();
    resp.setContentType("text/html");
    out.println("<HTML><HEAD><TITLE>Lettura dei " +
                " cookie</TITLE></HEAD>");
    out.println("<BODY>");
```

Lettura dei cookie (II)

```
    if(cookies!=null) {
      out.println("I tuoi cookie sono i seguenti:<BR>"+
                  "<TABLE BORDER=\"1\">");
      out.println("<TR><TD>Nome del cookie</TD>"+
                  "<TD>Valore del cookie</TD></TR>");

      for (int i = 0; i < cookies.length; i++)
        out.println("<TR><TD>"+cookies[i].getName()+"</TD><TD>"+
                    cookies[i].getValue()+"</TD></TR>");

        out.println("</TABLE>");
    }
    else
      out.println("Non hai nessun cookie impostato"+
                  " da questa applicazione.");
    out.println("</BODY></HTML>");
  }
}
```

Modifiche da apportare al file `web.xml`

```
<?xml version="1.0" encoding="ISO-8859-1"?>

<!DOCTYPE web-app
    PUBLIC "-//Sun Microsystems, Inc.//DTD Web Application 2.3//EN"
    "http://java.sun.com/dtd/web-app_2_3.dtd">

<web-app>
   ...
   <servlet>
        <servlet-name>LeggiCookie</servlet-name>
        <servlet-class>LeggiCookie</servlet-class>
    </servlet>
    <servlet-mapping>
        <servlet-name>LeggiCookie</servlet-name>
        <url-pattern>/servlet/LeggiCookie</url-pattern>
    </servlet-mapping>
</web-app>
```

Modifiche da apportare al file `index.html`

```
<HTML>
<HEAD>
<TITLE>Servlet di prova</TITLE>
</HEAD>

<BODY>
  <TABLE>
    ...
    <TR>
      <TD>
        <A HREF="servlet/LeggiCookie">Lettura dei cookie</A>
      </TD>
    </TR>
  </TABLE>
</BODY>
</HTML>
```

Sessioni

- E' possibile usare le funzionalità di "session tracking" offerte dall'API delle servlet per delegare al server la gestione esplicita del tracciamento del client (utente).
- La prima volta che un utente si collega ad una servlet che supporta le sessioni viene creato un oggetto di tipo **`javax.servlet.http.HttpSession`** in cui è possibile memorizzare le informazioni di interesse.
- Il metodo **`getSession()`** di **`HttpServletRequest`** serve a recuperare la sessione corrente.

Il metodo `getSession()`

- **`getSession()`** prende un argomento di tipo **`boolean`**:
  - se viene passato **`true`** come argomento e non esiste una sessione corrente, viene creato un nuovo oggetto di tipo **`HttpSession`**;
  - se viene passato **`false`** come argomento e non esiste una sessione corrente, viene restituito **`null`**.

Session ID

- Quando viene creato un nuovo oggetto di tipo **`HttpSession`**, il server genera un Session ID univoco per quel particolare client.
- Dato che i client possono differire nelle funzionalità offerte, esistono varie possibilità per associare il Session ID al client. Tra queste le più comuni sono:
  - utilizzo dei cookie,
  - codifica diretta del Session ID nei link.

Codifica del Session ID nei link

- Se vogliamo codificare il Session ID nei link per rendere fruibile la nostra applicazione anche con browser che non supportano i cookie, usiamo il metodo encodeURL.

  Quindi un codice come il seguente:

```
out.println("<A HREF=\"LeggiCookie\">Check Out</A>");
```

  va rimpiazzato con il seguente:

```
out.print("<A HREF=\"");
out.print(res.encodeUrl("LeggiCookie");
out.println("\">Check Out</A>");
```

  Il server poi genererà il seguente codice HTML:

```
<A
HREF="LeggiCookie;jsessionid=FD8AD991FD4779CF45B2A70C7E2ACB38">Che
ckOut</A>
```

Metodi utili di `HttpSession`

- **`getID()`**: restituisce l'ID della sessione;
- **`setAttribute()`**: associa un oggetto alla sessione:

```
session.setAttribute("myservlet.hitcount",
                                    new Integer(34));
```

- **`getAttribute()`**: recupera un oggetto dalla sessione:

```
Integer hits =
(Integer)session.getAttribute("myservlet.hi
tcount")
```

- Per evitare conflitti si segue la convenzione di dare un nome agli oggetti secondo il pattern seguente:

  <nome-applicazione>.<nome-oggetto>

Esempio: contare il numero di visite con le sessioni (I)

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

public class ContatoreSessione extends HttpServlet {
  public void doGet(HttpServletRequest req,
                    HttpServletResponse resp)
              throws ServletException, IOException {
    PrintWriter out = resp.getWriter();
    resp.setContentType("text/html");
    HttpSession utente = req.getSession(true);
    Integer visite; // contatore delle visite
```

Recupera la sessione corrente oppure crea una nuova sessione

Esempio: contare il numero di visite con le sessioni (II)

```
if(!utente.isNew()) {
  // Non considera le sessioni appena create
  visite =(Integer)utente.
              getAttribute("ContatoreSessione.visite");

  if(visite == null)
    visite = new Integer(1);
  else
    visite = new Integer(visite.intValue() + 1);
}
else
  visite = new Integer(1);
```

Restituisce true
se la sessione
è nuova (i.e., il session ID non
è ancora arrivato sul client
e tornato indietro sul server)

Esempio: contare il numero di visite con le sessioni (III)

```
    utente.setAttribute("ContatoreSessione.visite", visite);
    out.println("<HTML><HEAD><TITLE>Contatore con " +
                " sessione</TITLE></HEAD>");
    out.println("<BODY>Hai visitato questa pagina " +
                visite + " volta(e)");
    out.println("dal momento in cui la tua ultima " +
                "sessione &egrave; scaduta.");
    out.println("<BR>Il tuo Session ID &egrave; " +
                utente.getId());
    out.println("</BODY></HTML>");
  }
}
```

Recupera il Session ID

Modifiche da apportare al file **web.xml**

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE web-app
    PUBLIC "-//Sun Microsystems, Inc.//DTD Web Application
   2.3//EN"
    "http://java.sun.com/dtd/web-app_2_3.dtd">
<web-app>
 ...
 <servlet>
   <servlet-name>ContatoreSessione</servlet-name>
   <servlet-class>ContatoreSessione</servlet-class>
 </servlet>
 <servlet-mapping>
   <servlet-name>ContatoreSessione</servlet-name>
   <url-pattern>/servlet/ContatoreSessione</url-pattern>
 </servlet-mapping>
</web-app>
```

Modifiche da apportare al file `index.html`

```
<HTML>
<HEAD>
<TITLE>Servlet di prova</TITLE>
</HEAD>
<BODY>
  <TABLE>
    ...
    <TR>
      <TD>
        <A HREF="servlet/ContatoreSessione">
          Un contatore di visite con sessione</A>
      </TD>
    </TR>
  </TABLE>
</BODY>
</HTML>
```

Log

- Tomcat mantiene un log delle operazioni e degli eventi in file di testo memorizzati nella directory **`$TOMCAT_HOME/logs/`**.
- Oltre alle informazioni registrate automaticamente da Tomcat, è possibile per il programmatore registrarne altre esplicitamente, durante l'esecuzione di una servlet.
- Il metodo utilizzabile a tal fine è **`log(String)`** definito in **`javax.servlet.GenericServlet`** che viene quindi ereditato anche nelle servlet HTTP.

Esempio (I)

Scriviamo una servlet che, tramite un form, consenta di inviare dei messaggi da registrare nel file di log:

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

public class RegistraLog extends HttpServlet {
   private String messaggio;

   public void init() {
        messaggio = "";
   }
```

Esempio (II)

```
public void doGet(HttpServletRequest req, HttpServletResponse res)
                  throws ServletException, IOException {
      res.setContentType("text/html");
      PrintWriter out = res.getWriter();
      out.println("<HTML><BODY>");
      out.println("<H2>Registrazione nel file di log</H2>");
      out.println("Ultimo messaggio registrato: <B>" +
                  (messaggio.length()==0 ? "" : messaggio) +
                  "</B><BR>");
      out.println("<FORM METHOD=\"post\" ACTION=\"RegistraLog\">");
      out.println("Messaggio: <INPUT TYPE=\"text\" NAME=\"Messaggio\""+
                  " SIZE=\"30\"><BR>");
      out.println("<INPUT TYPE=\"submit\" NAME=\"Invia\""+
                  " VALUE=\"Invia &gt;&gt;\">");
      out.println("</FORM>");
      out.println("</BODY></HTML>");
}
```

Esempio (III)

```
   public void doPost(HttpServletRequest req,
                      HttpServletResponse res)
               throws ServletException, IOException {
        messaggio = req.getParameter("Messaggio");

        if(messaggio.trim().length()>0)
                log("RegistraLog servlet: "+messaggio);

        doGet(req, res); // ripresenta il form
   }

}
```

Modifiche da apportare al file `web.xml`

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE web-app
    PUBLIC "-//Sun Microsystems, Inc.//DTD Web Application
   2.3//EN"
    "http://java.sun.com/dtd/web-app_2_3.dtd">
<web-app>
 ...
 <servlet>
   <servlet-name>LogServlet</servlet-name>
   <servlet-class>RegistraLog</servlet-class>
 </servlet>
 <servlet-mapping>
   <servlet-name>LogServlet</servlet-name>
   <url-pattern>/servlet/RegistraLog</url-pattern>
 </servlet-mapping>
</web-app>
```

Modifiche da apportare al file `index.html`

```
<HTML>
<HEAD>
<TITLE>Servlet di prova</TITLE>
</HEAD>
<BODY>
  <TABLE>
    ...
    <TR>
      <TD>
        <A HREF="servlet/RegistraLog">Registrare
           messaggi nel file di log</A>
      </TD>
    </TR>
  </TABLE>
</BODY>
</HTML>
```

Come appare il messaggio nel file di log `localhost.2006-10-19.log`

```
...
19-ott-2006 10.09.49
   org.apache.catalina.core.StandardWrapperValve invoke
INFO: Servlet LeggiCookie is currently unavailable
19-ott-2006 10.09.52
   org.apache.catalina.core.StandardWrapperValve invoke
INFO: Servlet LeggiCookie is currently unavailable
19-ott-2006 11.02.34
   org.apache.catalina.core.ApplicationContext log
INFO: LogServlet: RegistraLog servlet: Messaggio di prova
...
```

Note

- Utilizzare il metodo **`log(String)`** per registrare dei messaggi nei file di log può essere utile per il debugging delle servlet.
- Infatti, utilizzando la funzione **`log(String)`** all'interno dei blocchi **`catch`** è possibile registrare dei dettagliati messaggi d'errore che possono essere esaminati off-line dal programmatore.

Esercizio

- Modificare la servlet Bilancio (dopo aver implementato le modifiche degli esercizi della lezione precedente) in modo da:
  - tener traccia in un cookie della data dell'ultimo collegamento;
  - usare il meccanismo di sessione per evitare di inserire ogni volta username e password (in pratica, tramite la sessione, la servlet dovrà ricordarsi se un utente ha già effettuato il login).