

Servlet:

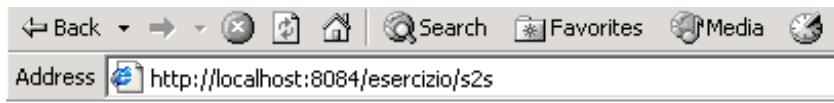
Comunicazioni tra Servlet

Intro Web Programming

AA 2013-2014 Gino Perna e Maurizio Marchese

Comunicazione tra servlet

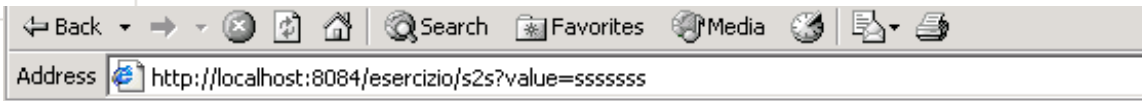
- **Perche' una servlet deve comunicare con un'altra?**
 - ❑ Spezzettare il codice
 - ❑ Comunicare solo le informazioni che servono
 - ❑ Riutilizzo di parti comuni



A form from Servlet #1

Enter a value to send to Servlet #2.

Send to Servlet #2



Servlet #2

Servlet #1 passed a String object via request scope. The value of the String is: **sssssss**.

Comunicazione tra servlet

- La servlet 1 chiama la servlet 2 passando tutti i parametri (o settandone altri)
- La comunicazione avviene direttamente sul server
- Avviene attraverso *getRequestDispatcher* e *forward*

Comunicazione tra servlet: Servlet s1

```
public class s1 extends HttpServlet {
    ...
    response.setContentType("text/html");
    String param = request.getParameter("value");

    if(param != null && !param.equals("")) {
        request.setAttribute("value", param);

        RequestDispatcher rd = request.getRequestDispatcher("/s2s2");
        rd.forward(request, response);

        return;
    }

    PrintWriter out = response.getWriter();
    out.println("<html><head><title>Servlet #1</title>");out.println("</head>");
    out.println("<body>"); out.println("<h1>A form from Servlet #1</h1>");
    out.println("<form>"); out.println("Enter a value to send to Servlet #2.");
    out.println("<input name=\"value\"><br>");
    out.print("<input type=\"submit\" ");
    out.println("value=\"Send to Servlet #2\">");
    out.println("</form></body></html>");
```

Comunicazione tra servlet: Servlet s2

```
public class s2 extends HttpServlet {  
  
    ...  
  
    response.setContentType("text/html");  
  
    PrintWriter out = response.getWriter();  
    out.println("<html><head><title>Servlet #2</title>");  
    out.println("</head>");  
    out.println("<body>");  
    out.println("<h1>Servlet #2</h1>");  
    String value = (String)request.getAttribute("value");  
    if(value != null && !value.equals("")) {  
        out.print("Servlet #1 passed a String object via ");  
        out.print("request scope. The value of the String is: ");  
        out.println("<b>" + value + "</b>.");  
    }  
    else {  
        out.println("No value passed!");  
    }  
    out.println("</body>");  
    out.println("</html>");  
    out.close();  
}
```

Servlet: Conservare lo stato della connessione Utilizzo delle Sessioni

Intro Web Programming

AA 2013-2014 Gino Perna e Maurizio Marchese

Sessioni

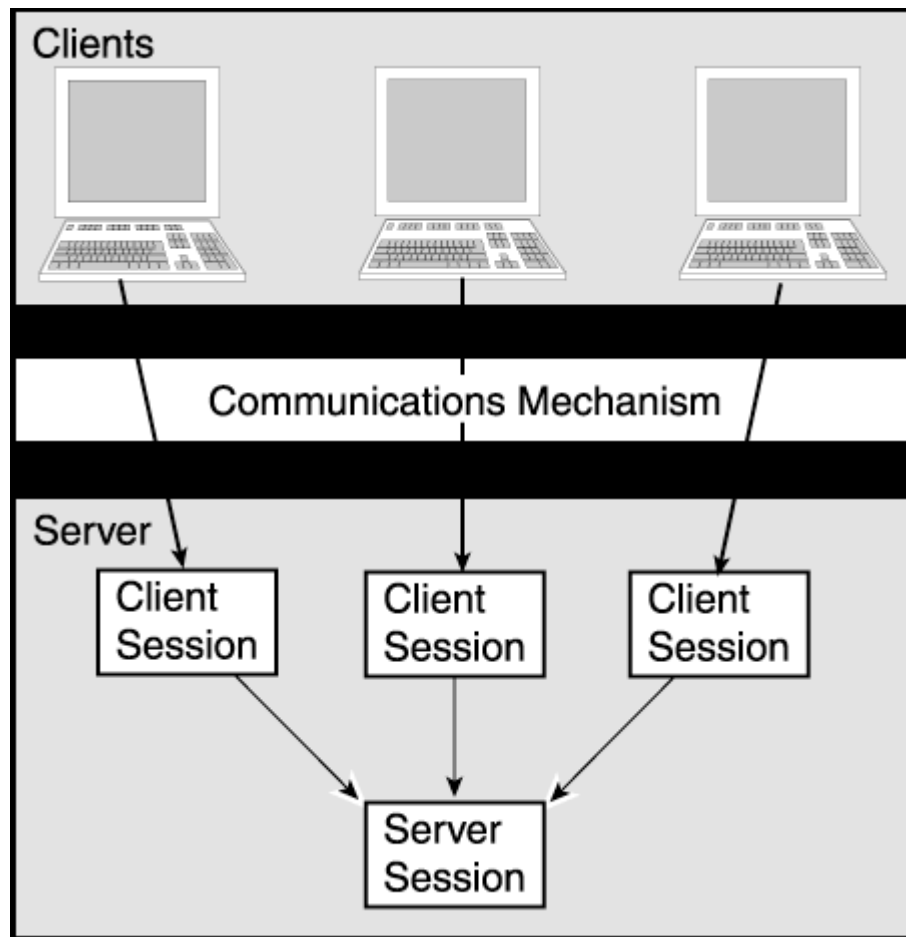
Mantenere lo stato della connessione di ciascun client connesso al sito attraverso una serie di coppie chiave, valore.

- Queste sono univocamente identificate per ciascun client.

Sessioni

- Un modo per tenere traccia dei client lato server (ambiente multi-thread):
 - ❑ A ciascun client viene attribuito un numero univoco, che viene continuamente passato tra client e server attraverso **cookie** o **URL rewriting**
 - ❑ Questo numero univoco identifica la sessione ed eventuali variabili di stato lato server che rimangono validi fino a che:
 - Il client invalida la sessione con un comando ed i dati svaniscono (lato server)

Sessioni



Sessioni: perche'?

- Cliente su un sito di e-commerce aggiunge articoli al carrello: come fa il server a memorizzarli?
- Il cliente decide di comperare, come fa il server a sapere quale e' il suo carrello?

Come si gestiscono le sessioni

Tre modalita' fondamentali

1. Cookies

2. URL Rewriting / Hidden Form Fields

3. Java Session: the HttpSession API

Come gestire sessioni: cookies

- Idea: associare un cookie con i dati sul server

```
String sessionID = makeUniqueString();  
HashMap sessionInfo = new HashMap();  
HashMap globalTable = findTableStoringSessions();  
globalTable.put(sessionID, sessionInfo);  
Cookie sessionCookie =  
    new Cookie("JSESSIONID", sessionID);  
sessionCookie.setPath("/");  
response.addCookie(sessionCookie);
```

to do !

- Da fare a mano:
 - ❑ Estrazione dei cookies che memorizzano la sessione
 - ❑ Corretta expiration date per il cookie
 - ❑ Associare ciascuna richiesta con la hash
 - ❑ Generare l'identificatore della sessione

Session Tracking: url rewriting

- **Idea:**

- Il Client appende delle informazioni extra alla fine di ciascun URL, in modo da identificare la sessione

- For example, the following link in a Web page:

``

is rewritten as:

``

- Il server associa l'identificatore ai dati memorizzati riguardanti quella sessione
- E.g., `http://host/path/file.html?jsessionid=1234`

Session Tracking: url rewriting

- **Vantaggi**

- Funziona anche se i cookies sono disabilitati o non funzionano

- **Svantaggi**

- Encode di tutti gli URLs che si riferiscono al proprio sito
- Tutte le pagine devono essere generate dinamicamente
- Non funziona per bookmarks e links da altri siti

Session Tracking: **Hidden Form Fields**

- **Idea:**

```
<INPUT TYPE="HIDDEN" NAME="session"  
      VALUE="...">
```

- **Vantaggi**

- ❑ Funziona anche in caso di cookies disabled or unsupported

- **Svantaggi**

- ❑ Una miriade di if e processamenti;
- ❑ **Tutte le pagine sono dei form!**

Java Session Tracking API

- **Session objects che esistono sul server**
- **Sessioni associate automaticamente al client (via cookies o URL-rewriting)**

- Uso **request.getSession** per avere la session corrente

Behind the scenes, the system looks at cookie or URL extra info and sees if it matches the key to some previously stored session object. If so, it returns that object. If not, it creates a new one, assigns a cookie or URL info as its key, and returns that new session object.

- **Meccanismo tipo Hashtable: permette di immagazzinare oggetti arbitrary nelle sessioni**
 - **setAttribute** stores values
 - **getAttribute** retrieves values

Session Tracking: Basics

- **Access the session object**

- Call `request.getSession` to get HttpSession object
 - Questa e' una hashtable associata al client

- **Look up information associated with a session.**

- Call `getAttribute` on the HttpSession object, cast the return value to the appropriate type, and check whether the result is null.

- **Store information in a session.**

- Use `setAttribute` with a key and a value.

- **Discard session data.**

- Call `removeAttribute` discards a specific value.
- Call `invalidate` to discard an entire session.
- Call `logout` to log the client out of the Web/app server

Session Tracking: Esempio

```
HttpSession session = request.getSession();
SomeClass value =
    (SomeClass)session.getAttribute("someID");
if (value == null) {
    value = new SomeClass(...);
    session.setAttribute("someID", value);
}
doSomethingWith(value);
```

Default: l'ID di sessione viene spedito al client con un cookie specifico, i.e. tipicamente JSESSIONID

Session Tracking: url-rewrite

- **Session tracking code:**

```
HttpSession session = request.getSession();
```

```
...
```

- Nessun cambiamento rispetto a prima

- **Code that generates hypertext links back to same site:**

- Passare l' URL attraverso **response.encodeURL**.

- Se il server usa i cookies, ritorna l'URL non modificato;
- Se il server usa URL rewriting, appende il session info all'URL

```
String url = "http://path/order-page.html";
```

```
url = response.encodeURL(url);
```

- **Code that does sendRedirect to own site:**

- Passare l'URL attraverso **response.encodeRedirectURL**

Session Tracking: url-rewrite

- Rewrite URLs to return to the browser

Suppose you currently have this statement:

```
out.println("<a href=\"/store/catalog\">catalog<a>");
```

Change the servlet to call the `encodeURL` method before sending the URL to the output stream:

```
out.println("<a href=\"");  
out.println(response.encodeURL ("/store/catalog"));  
out.println("\>catalog</a>");
```

HttpSession Metodi principali

- **getAttribute**

- ▣ Extracts a previously stored value from a session object. Returns null if no value is associated with given name.

- **setAttribute**

- ▣ Associates a value with a name. Monitor changes: values implement HttpSessionBindingListener.

- **removeAttribute**

- ▣ Removes values associated with name.

- **getAttributeNames**

- ▣ Returns names of all attributes in the session.

- **getId**

- ▣ Returns the unique identifier.

HttpSession Metodi principali (cont)

- **isNew**

- ▣ Determines if session is new to *client* (not to *page*)

- **getCreationTime**

- ▣ Returns time at which session was first created

- **getLastAccessedTime**

- ▣ Returns time at which session was last sent from client

- **getMaxInactiveInterval, setMaxInactiveInterval**

- ▣ Gets or sets the amount of time session should go without access before being invalidated

- **invalidate**

- ▣ Invalidates current session

- **logout**

- ▣ Invalidates *all* sessions associated with user

Come cancellare dati da una sessione

When you are done with a user's session data, you have several options:

1. **Remove a particular attribute:** You can call public **void removeAttribute(String name)** method to delete the value associated with a particular key.
2. **Delete the whole session:** You can call public **void invalidate()** method to discard an entire session.
3. **Setting Session timeout:** You can call public **void setMaxInactiveInterval(int interval)** method to set the timeout for a session individually.

Come cancellare dati da una sessione

4. Log the user out: The servers that support servlets 2.4, you can call **logout** to log the client out of the Web server and invalidate all sessions belonging to all the users.

5. web.xml Configuration: If you are using Tomcat, apart from the above mentioned methods, you can configure session time out in web.xml file as follows.

```
<session-config>
  <session-timeout>15</session-timeout>
</session-config>
```

- The timeout is expressed as minutes, and overrides the default timeout which is 30 minutes in Tomcat.
- The `getMaxInactiveInterval()` method in a servlet returns the timeout period for that session in seconds. So if your session is configured in web.xml for 15 minutes, `getMaxInactiveInterval()` returns 900.

Servlet: Conta gli accessi per Client

```
public class ShowSession extends HttpServlet {  
    public void doGet(HttpServletRequest request, HttpServletResponse  
        response)  
        throws ServletException, IOException {  
        response.setContentType("text/html");  
        HttpSession session = request.getSession();  
        String heading;  
        Integer accessCount =(Integer)session.getAttribute("accessCount");  
        if (accessCount == null) {  
            accessCount = new Integer(0);  
            heading = "Welcome, Newcomer";  
        } else {  
            heading = "Welcome Back";  
            accessCount =new Integer(accessCount.intValue() + 1);  
        }  
        session.setAttribute("accessCount", accessCount);  
    }  
}
```

Servlet: Conta gli accessi per Client (cont)

```
out.println("<HTML><TITLE>"+title+"</TITLE>" +  
  
    "<BODY BGCOLOR=\"#FDF5E6\">\n" +  
    "<H1 ALIGN=\"CENTER\">" + heading + "</H1>\n" +  
    "<H2>Information on Your Session:</H2>\n" +  
    "<TABLE BORDER=1 ALIGN=\"CENTER\">\n" +  
    "<TR BGCOLOR=\"#FFAD00\">\n" +  
    " <TH>Info Type<TH>Value\n" +  
    "<TR>\n" +    " <TD>ID\n" +  
    " <TD>" + session.getId() + "\n" +    "<TR>\n" +  
    " <TD>Creation Time\n" +  
    " <TD>" + new Date(session.getCreationTime()) + "\n" +  
    "<TR>\n" +    " <TD>Time of Last Access\n" +  
    " <TD>" + new Date(session.getLastAccessedTime()) + "\n" +  
  
    "<TR>\n" +  
    " <TD>Number of Previous Accesses\n" +  
    " <TD>" + accessCount + "\n" +  
    "</TABLE>\n" +    "</BODY></HTML>");
```

Conta Accessi: primo accesso

Welcome, Newcomer

Information on Your Session:

Info Type	Value
ID	4C8291E68C640DB70E4E47A0EF3C3101
Creation Time	Thu Nov 25 08:08:03 CET 2004
Time of Last Access	Thu Nov 25 08:08:03 CET 2004
Number of Previous Accesses	0

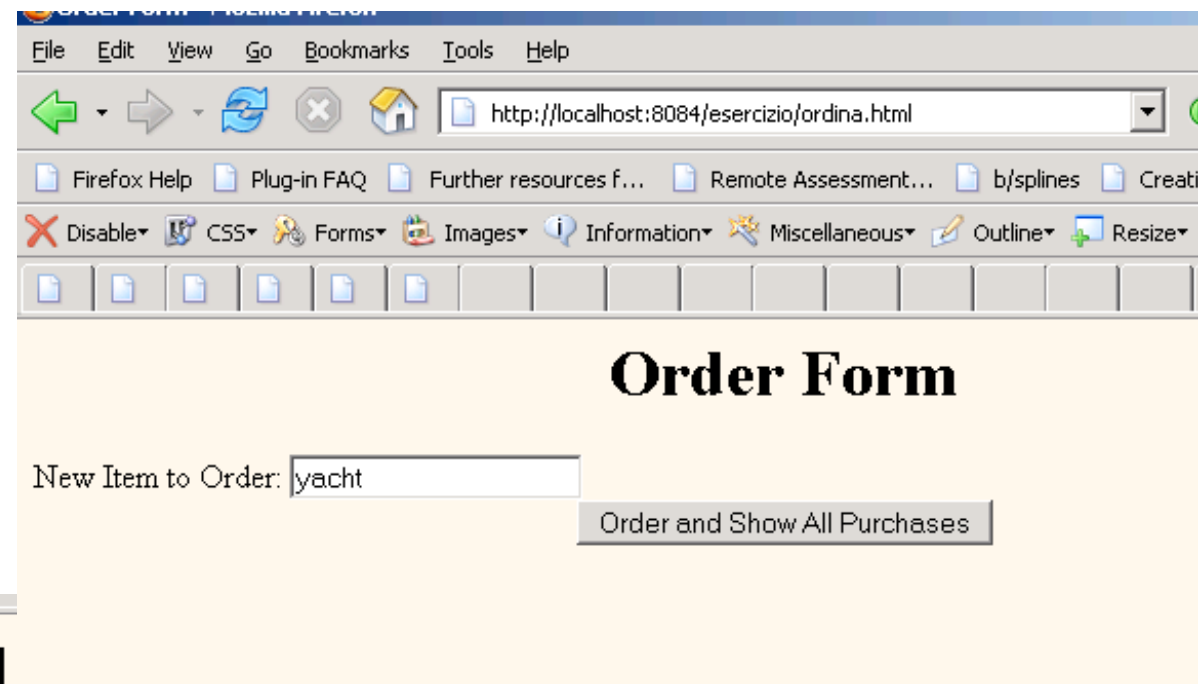
Conta Accessi: accessi successivi

Welcome Back

Information on Your Session:

Info Type	Value
ID	CE0ADF3AAF0C93C461435A5DBECD71B2
Creation Time	Wed Nov 24 22:34:56 CET 2004
Time of Last Access	Wed Nov 24 22:56:34 CET 2004
Number of Previous Accesses	7

Esercizio: lista articoli



Order Form

New Item to Order:

Items Purchased

- yacht
- yacht
- bara
- lamborghini

Per ordinare ancora, click [here](#)

Memorizzazione di dati

```
public class ShowItems extends HttpServlet {  
    public void doGet(HttpServletRequest request,  
        HttpServletResponse response)  
        throws ServletException, IOException {  
        HttpSession session = request.getSession();  
        ArrayList previousItems =  
            (ArrayList)session.getAttribute("previousItems");  
        if (previousItems == null) {  
            previousItems = new ArrayList();  
            session.setAttribute("previousItems",  
                previousItems);  
        }  
    }  
}
```

Memorizzazione di dati (cont)

```
String newItem = request.getParameter("newItem");
PrintWriter out = response.getWriter();
...
synchronized(previousItems) {
    if (newItem != null) {
        previousItems.add(newItem);
    }
    if (previousItems.size() == 0) {
        out.println("<I>No items</I>");
    } else {
        out.println("<UL>");
        for(int i=0; i<previousItems.size(); i++) {
            out.println("<LI>" + (String)previousItems.get(i));
        }
        out.println("</UL>");
    }
}
```

Attenzione: tornare al form!

```
out.println("Per ordinare ancora, click <A HREF=\"\" +  
    response.encodeURL("/esercizio/ordina.html") +  
    "\">here</A>");  
  
} out.println("</BODY></HTML>");}
```


Lista articoli

Items Purchased

- ♦ yacht
- ♦ yacht
- ♦ bara
- ♦ lamborghini

Per ordinare ancora, click [here](#)

Order Form

New Item to Order:

Sommario

- **Sessions do not travel across network**
 - ▣ Only unique identifier does
- **Get the session**
 - ▣ `request.getSession`
- **Estrazione dati dalle sessioni**
 - ▣ `session.getAttribute`
 - ▣ **Attenzione al typecast e check for null**
- **Memorizzazione dati nelle sessioni**
 - ▣ `session.setAttribute`

Bibliografia

- <http://www.tutorialspoint.com/servlets/servlets-session-tracking.htm>
-