Programmazione WEB

Lezione del 19 Marzo 2020

Docente: Novella Bartolini
Ricevimento: Mercoledi` 12.30-13.30
Via Salaria 113, terzo piano, stanza 309
Email: bartolini@di.uniroma1.it

Gestire la sessione con oggetti persistenti sul server (`HttpSession`)

- Interfaccia **`HttpSession`**
- Trovate informazioni all'indirizzo:

  http://127.0.0.1:8080/tomcat-docs/servletapi/index.html

- Un oggetto che implementi l'interfaccia HttpSession identifica un utente/browser attraverso diverse richieste e permette di memorizzare informazioni acquisite durante la sessione di navigazione.
- Tale oggetto rappresenta la sessione:
  - Persiste per uno specifico periodo di tempo.
  - Corrisponde ad un solo utente, che può visitare il sito anche più volte.

Perché usare HttpSession

- Non potremmo memorizzare le informazioni che ci servono direttamente attraverso gli attributi della servlet?

- Più client possono accedere alla stessa servlet in parallelo!

Metodi da conoscere per usare le sessioni (1/3)

- Metodi dell'oggetto HttpServletRequest
  - HttpSession **getSession** (boolean **create**)
    Restituisce l'oggetto HttpSession associato con la richiesta. Se non esiste ancora nessun oggetto HttpSession, ne viene creato uno nel caso **create** valga **true**.
- Metodi dell'oggetto HttpSession
  - public void **setAttribute** (java.lang.String name, **java.lang.Object value**)
    - Associa un oggetto alla presente sessione, utilizzando il nome specificato. Se esiste un collegamento con un oggetto con lo stesso nome, questo viene rimpiazzato.
  - java.lang.Object **getAttribute**(java.lang.String name)
    - Restituisce l'oggetto associato con il nome dato in input, oppure null se non esiste nessun oggetto con quel nome

Metodi da conoscere per usare le sessioni (2/3)

- Metodi dell'oggetto HttpSession
  - java.lang.String **getId**()
    Restituisce una stringa contenente l'identificativo unico della sessione
  - boolean **isNew**()
    Restituisce true se la sessione con il client è stata creata in seguito alla richiesta corrente
  - long **getCreationTime**()
    Restituisce l'istante di creazione della sessione misurato in millisecondi a partire dalle 00:00 del 1 Gennaio 1970, GMT
  - long **getLastAccessedTime**()
    Restituisce l'istante dell'ultima richiesta associata alla sessione misurato in millisecondi a partire dalle 00:00 del 1 Gennaio 1970, GMT

Metodi da conoscere per usare le sessioni (3/3)

- … metodi dell'oggetto HttpSession
  - int **getMaxInactiveInterval**()
    Restituisce il massimo intervallo di tempo, in secondi, in cui il container può mantenere la sessione aperta, tra due accessi consecutivi da parte del client.
  - java.util.Enumeration **getAttributeNames**()
    Restituisce un oggetto Enumeration (un elenco) di stringhe contenenti i nomi di tutti gli oggetti associati alla sessione

Esercizio: uso di HttpSession per memorizzare le scelte dell'utente

- Memorizziamo in un oggetto che implementa l'interfaccia **`HttpSession`** le informazioni che nell'esercizio precedente mettevamo nei cookie.
  - Possiamo memorizzare coppie nome – valore

    dove il valore può essere un oggetto (non solo una stringa come nei cookie)
- Servlet **`SessionServlet`**
  - Usa l'oggetto **`HttpSession`**
  - Gestisce sia richieste di **`GET`** che richieste di **`POST`**

E’ utile sapere per svolgere questo esercizio… (cont.)

- Interface `Enumeration`
  - Consente l’elencazione di una serie di elementi, uno alla volta
  - Due metodi:
    - java.lang.Object nextElement()
      restituisce il prossimo elemento dell’elenco
    - boolean hasMoreElements()
      restituisce true se l’elenco contiene ancora elementi

```
<?xml version = "1.0"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">

<!-- SessionSelectLanguage.html -->

<html xmlns = "http://www.w3.org/1999/xhtml">
<head>
  <title>Using Sessions</title>
 </head>

 <body>
   <form action = "/DDS/sessions" method = "post">

     <p>Select a programming language:</p>
     <p>
       <input type = "radio" name = "language"
         value = "C" />C <br />

       <input type = "radio" name = "language"
         value = "C++" />C++ <br />

       <!-- this radio button checked by default -->
       <input type = "radio" name = "language"
         value = "Java" checked = "checked" />Java<br />

       <input type = "radio" name = "language"
         value = "VB6" />VB 6
     </p>

     <p><input type = "submit" value = "Submit" /></p>

   </form>
 </body>
 </html>
```

Stesso form dell'esercizio sui cookie

```
1  // SessionServlet.java
2  // Using HttpSession to maintain client state information.
3
4
5  import javax.servlet.*;
6  import javax.servlet.http.*;
7  import java.io.*;
8  import java.util.*;
9
10 public class SessionServlet extends HttpServlet {
11    private final Map books = new HashMap();
12
13    // initialize Map books
14    public void init()
15    {
16       books.put( "C", "0130895725" );
17       books.put( "C++", "0130895717" );
18       books.put( "Java", "0130125075" );
19       books.put( "VB6", "0134569555" );
20    }
21
22    // receive language selection and create HttpSession object
23    // containing recommended book for the client
24    protected void doPost( HttpServletRequest request,
25       HttpServletResponse response )
26          throws ServletException, IOException
27    {
28       String language = request.getParameter( "language" );
29
30       // Get the user's session object.
31       // Create a session (true) if one does not exist.
32       HttpSession session = request.getSession( true );
33
34       // add a value for user's choice to session
35       session.setAttribute( language, books.get( language ) );
```

Chiave

Valore

Usa il metodo **getSession** dell'interfaccia **HttpServletRequest** per ottenere l'oggetto **HttpSession** o crearlo (true)

Usa **setAttribute** per inserire il nome del linguaggio **language** e il corrispondente numero ISBN nell'oggetto **HttpSession** object.

```
36
37      response.setContentType( "text/html" );
38      PrintWriter out = response.getWriter();
39
40      // send XHTML page to client
41
42      // start XHTML document
43      out.println( "<?xml version = \"1.0\"?>" );
44
45      out.println( "<!DOCTYPE html PUBLIC \"-//W3C//DTD " +
46        "XHTML 1.0 Strict//EN\" \"http://www.w3.org" +
47        "/TR/xhtml1/DTD/xhtml1-strict.dtd\">" );
48
49      out.println(
50        "<html xmlns = \"http://www.w3.org/1999/xhtml\">" );
51
52      // head section of document
53      out.println( "<head>" );
54      out.println( "<title>Welcome to Sessions</title>" );
55      out.println( "</head>" );
56
57      // body section of document
58      out.println( "<body>" );
59      out.println( "<p>Welcome to Sessions! You selected " +
60        language + ".</p>" );
61
62      // display information about the session
63      out.println( "<p>Your unique session ID is: " +
64        session.getId() + "<br />" );
65
66      out.println(
67        "This " + ( session.isNew() ? "is" : "is not" ) +
68        " a new session<br />" );
69
```

Usa il metodo **getID** di **HttpSession** per conoscere l'ID di sessione.

Decide se si tratta di una nuova sessione usando il metodo **isNew**.

```
70      out.println( "The session was created at: " +
71        new Date( session.getCreationTime() ) + "<br />" );
72
73      out.println( "You last accessed the session at: " +
74        new Date( session.getLastAccessedTime() ) + "<br />" );
75
76      out.println( "The maximum inactive interval is: " +
77        session.getMaxInactiveInterval() + " seconds</p>" );
78
79      out.println( "<p><a href = " +
80        "\"servlets/SessionSelectLanguage.html\">" +
81        "Click here to choose another language</a></p>" );
82
83      out.println( "<p><a href = \"sessions\">" +
84        "Click here to get book recommendations</a></p>" );
85      out.println( "</body>" );
86
87      // end XHTML document
88      out.println( "</html>" );
89      out.close();    // close stream
90   }
91
92   // read session attributes and create XHTML document
93   // containing recommended books
94   protected void doGet( HttpServletRequest request,
95      HttpServletResponse response )
96         throws ServletException, IOException
97   {
98      // Get the user's session object.
99      // Do not create a session (false) if one does not exist.
100     HttpSession session = request.getSession( false );
101
102     // get names of session object's values
103     Enumeration valueNames;
104
```

Usa il metodo **getMaxInactiveInterval** per conoscere il massimo intervallo di tempo in cui la sessione può restare inattiva prima che il container la scarti.

ATTENZIONE all'uso dei percorsi relativi!!!

Il percorso di partenza è quello definito nell'url pattern e usato per invocare la servlet.

Provare a cambiare l'url pattern, dovrete ridefinire anche questi link

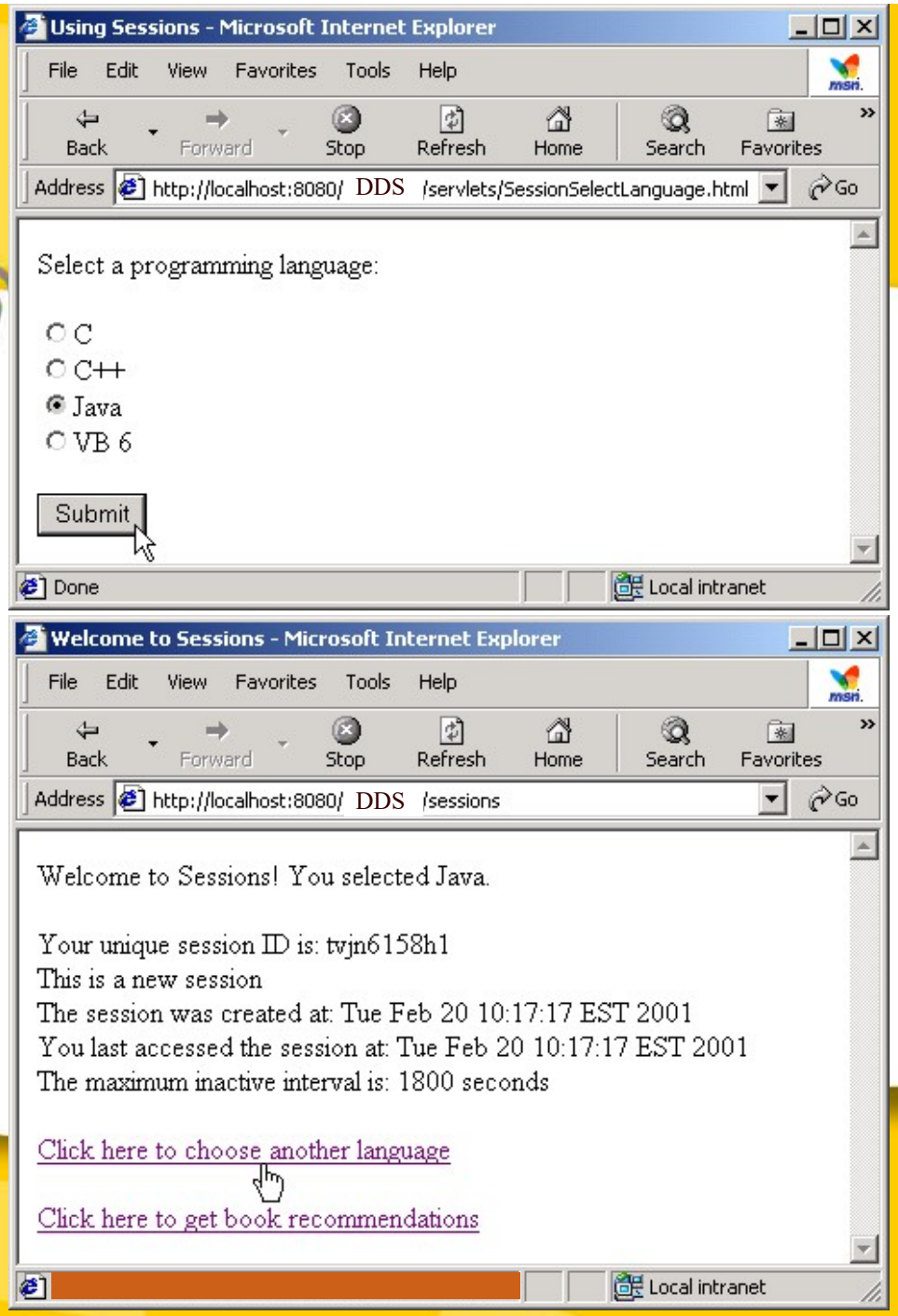
Ottiene l'oggetto **HttpSession** correlato alla sessione corrente, attraverso il metodo **getSession**.

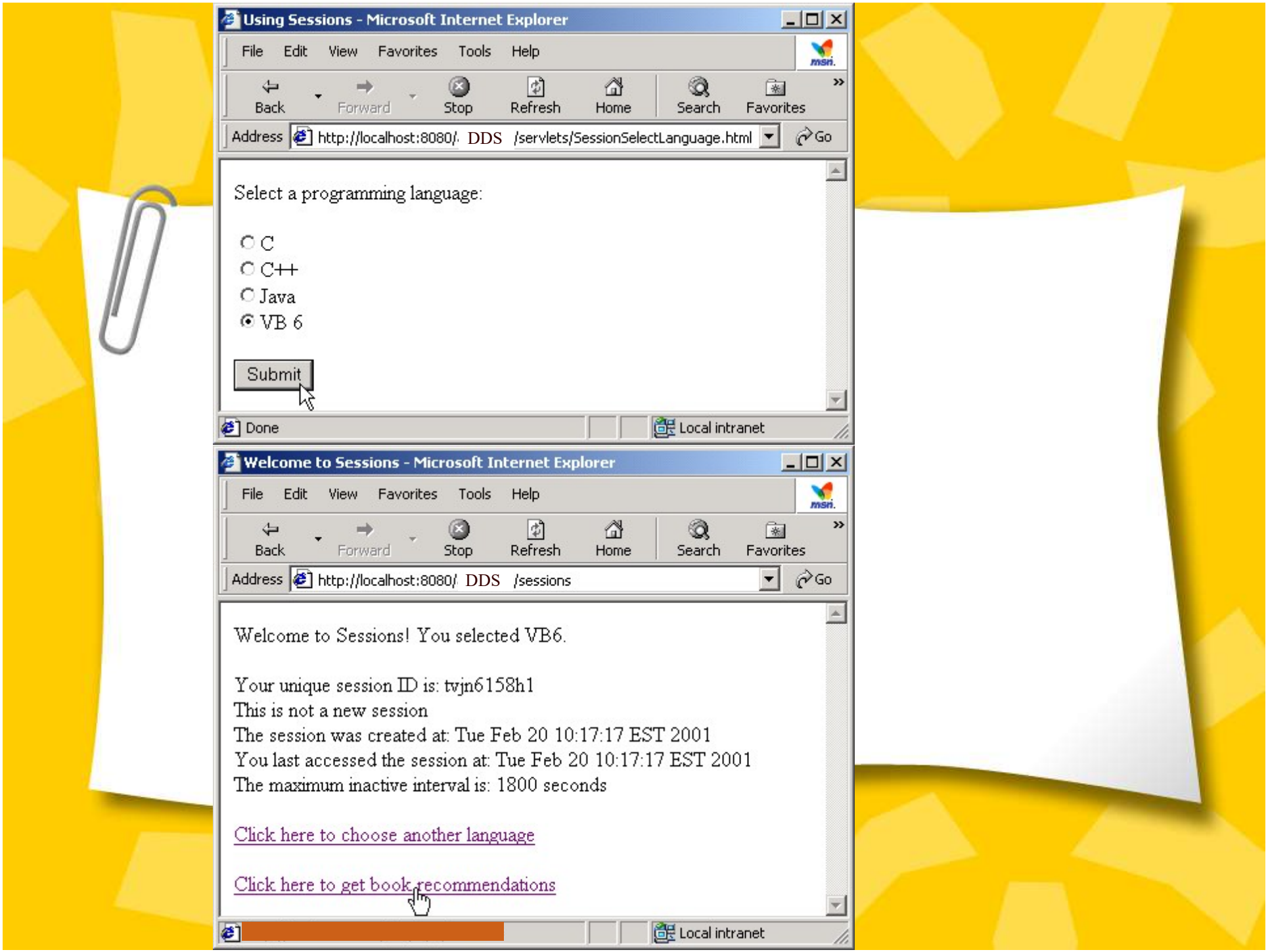
```
105     if ( session != null )
106        valueNames = session.getAttributeNames();
107     else
108        valueNames = null;
109
110     PrintWriter out = response.getWriter();
111     response.setContentType( "text/html" );
112
113     // start XHTML document
114     out.println( "<?xml version = \"1.0\"?>" );
115
116     out.println( "<!DOCTYPE html PUBLIC \"-//W3C//DTD " +
117        "XHTML 1.0 Strict//EN\" \"http://www.w3.org" +
118        "/TR/xhtml1/DTD/xhtml1-strict.dtd\">" );
119
120     out.println(
121        "<html xmlns = \"http://www.w3.org/1999/xhtml\">" );
122
123     // head section of document
124     out.println( "<head>" );
125     out.println( "<title>Recommendations</title>" );
126     out.println( "</head>" );
127
128     // body section of document
129     out.println( "<body>" );
130
131     if ( valueNames != null &&
132          valueNames.hasMoreElements() ) {
133        out.println( "<h1>Recommendations</h1>" );
134        out.println( "<p>" );
135
136        String name, value;
137
```

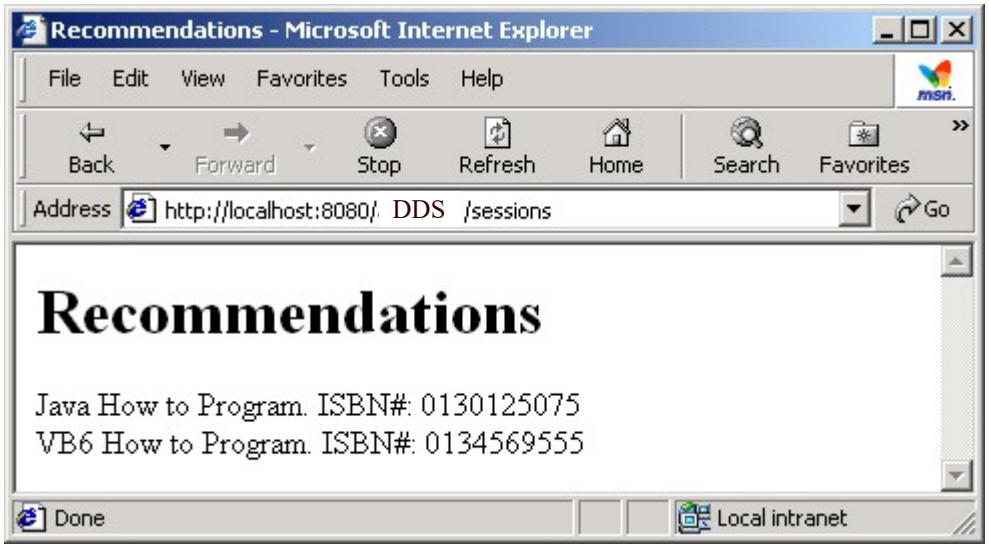
Uses **HttpSession** method **getAttributeNames** to retrieve an **Enumeration** of the attribute names.

```
138         // get value for each name in valueNames
139         while ( valueNames.hasMoreElements() ) {
140            name = valueNames.nextElement().toString();
141            value = session.getAttribute( name ).toString();
142
143            out.println( name + " How to Program. " +
144               "ISBN#: " + value + "<br />" );
145         }
146
147         out.println( "</p>" );
148      }
149      else {
150         out.println( "<h1>No Recommendations</h1>" );
151         out.println( "<p>You did not select a language.</p>" );
152      }
153
154      out.println( "</body>" );
155
156      // end XHTML document
157      out.println( "</html>" );
158      out.close();    // close stream
159   }
160 }
```

Invokes method **getAttribute** of **HttpSession** to retrieve the ISBN of a book from the **HttpSession** object.


Using Sessions - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Forward Stop Refresh Home Search Favorites
Address http://localhost:8080/ DDS /servlets/SessionSelectLanguage.html
Go
Select a programming language:
C
C++
Java
VB 6
Submit
Done
Local intranet
Welcome to Sessions - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Forward Stop Refresh Home Search Favorites
Address http://localhost:8080/ DDS /sessions
Go
Welcome to Sessions! You selected Java.
Your unique session ID is: tvjn6158h1
This is a new session
The session was created at: Tue Feb 20 10:17:17 EST 2001
You last accessed the session at: Tue Feb 20 10:17:17 EST 2001
The maximum inactive interval is: 1800 seconds
Click here to choose another language
Click here to get book recommendations
Local intranet


Using Sessions - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Forward Stop Refresh Home Search Favorites
Address http://localhost:8080/ DDS /servlets/SessionSelectLanguage.html
Go
Select a programming language:
C
C++
Java
VB 6
Submit
Done
Local intranet
Welcome to Sessions - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Forward Stop Refresh Home Search Favorites
Address http://localhost:8080/ DDS /sessions
Go
Welcome to Sessions! You selected VB6.
Your unique session ID is: tvjn6158h1
This is not a new session
The session was created at: Tue Feb 20 10:17:17 EST 2001
You last accessed the session at: Tue Feb 20 10:17:17 EST 2001
The maximum inactive interval is: 1800 seconds
Click here to choose another language
Click here to get book recommendations
Local intranet


Recommendations - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Forward Stop Refresh Home Search Favorites
Address http://localhost:8080/ DDS /sessions
Go
Recommendations
Java How to Program. ISBN#: 0130125075
VB6 How to Program. ISBN#: 0134569555
Done
Local intranet

Session Tracking with `HttpSession` (Cont.)

Descriptor element	Value
servlet *element*	
servlet-name	**sessions**
description	Using sessions to maintain state information.
servlet-class	**SessionServlet**
servlet-mapping *element*	
servlet-name	**sessions**
url-pattern	**/sessions**

Chiusura della sessione di navigazione

- Mentre per i cookie non è previsto un metodo esplicito per la cancellazione dei cookie,
  - Prelevare il cookie
  - Configurare il suo tempo di vita a zero (metodo setMaxAge(int) della classe Cookie)
  - Inviare di nuovo il cookie al client

  la cancellazione della sessione deve fare uso del metodo invalidate() dell'interfaccia HttpSession

Riflessioni…

- Q: Come può il server riconoscere le richieste di una stessa sessione e associarle correttamente all'oggetto persistente con il contesto di visibilità voluto?

Riflessioni…

- Q: Come può il server riconoscere le richieste di una stessa sessione e associarle correttamente all'oggetto persistente con il contesto di visibilità voluto?
- R: Viene usato un cookie!!!

Un COOKIEEEEE ????

- Q1: che senso ha usare la sessione se poi questa dipende dal cookie?
- R1: i dati sensibili restano sul server, il cookie creato non è trasferibile da un computer ad un altro in quanto contiene un identificativo generato dinamicamente.

Riflessioni…

- Q2: se il browser ha i cookie disabilitati come faccio a gestire la sessione sul server?
- R2: Si deve fare in modo che tutti i percorsi utilizzati dal browser appendano all'url l'identificativo di sessione (sarà sempre il server a scrivere dinamicamente questi url con una tecnica detta di URL rewriting). Questo si ottiene con il metodo dell'interfaccia HttpServletResponse:

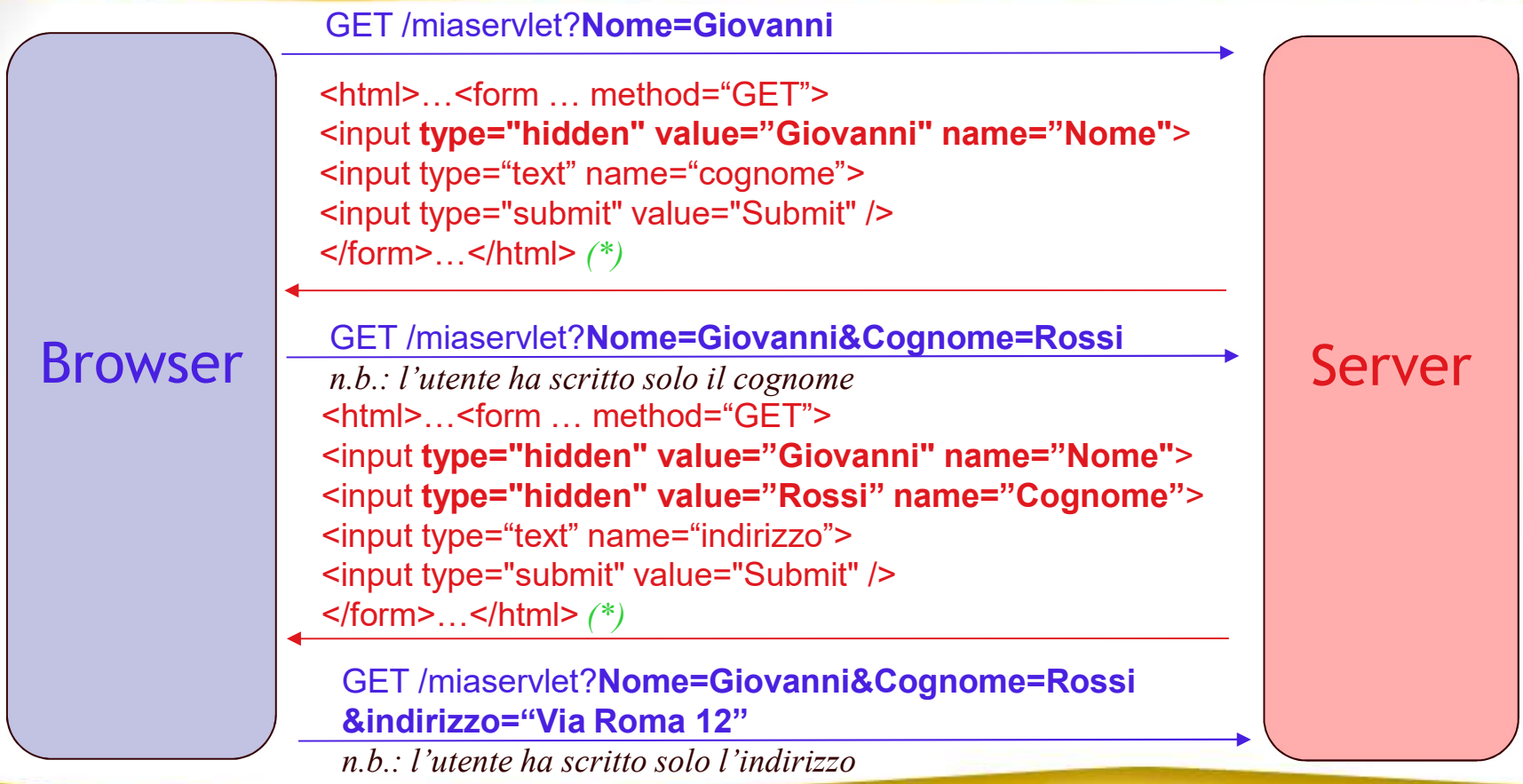
java.lang.String **encodeURL**(java.lang.String url)

Dalla documentazione:

Encodes the specified URL by including the session ID in it, or, if encoding is not needed, returns the URL unchanged.

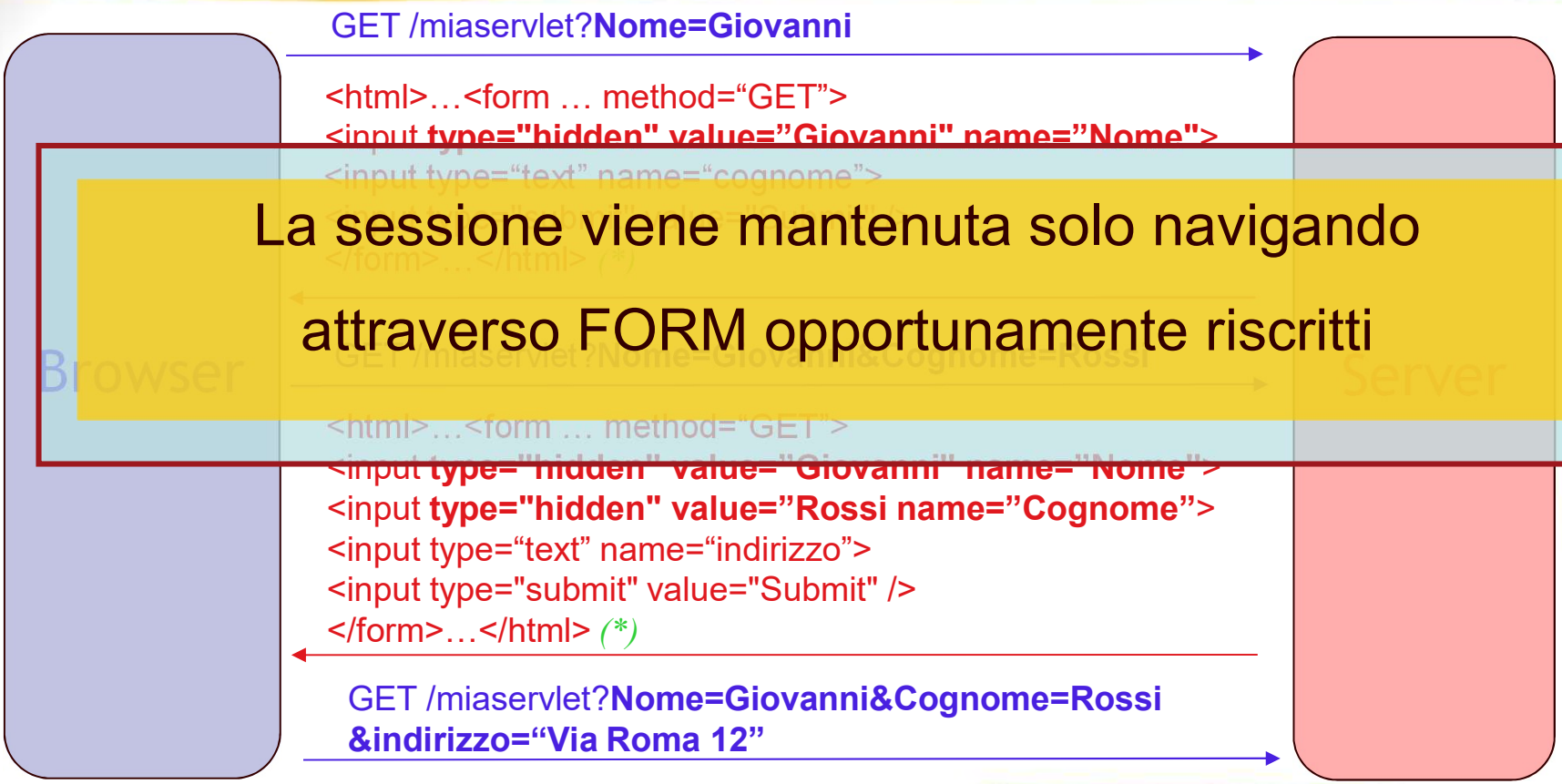
Riassunto: gestione della sessione tramite parametri hidden

Riassunto: gestione della sessione tramite parametri hidden



(*) *Il testo della pagina html viene generato dinamicamente dalla servlet in funzione dei parametri ricevuti nella richiesta, sia che fossero hidden sia che fossero visibili nel form*

Riassunto: gestione della sessione tramite parametri hidden



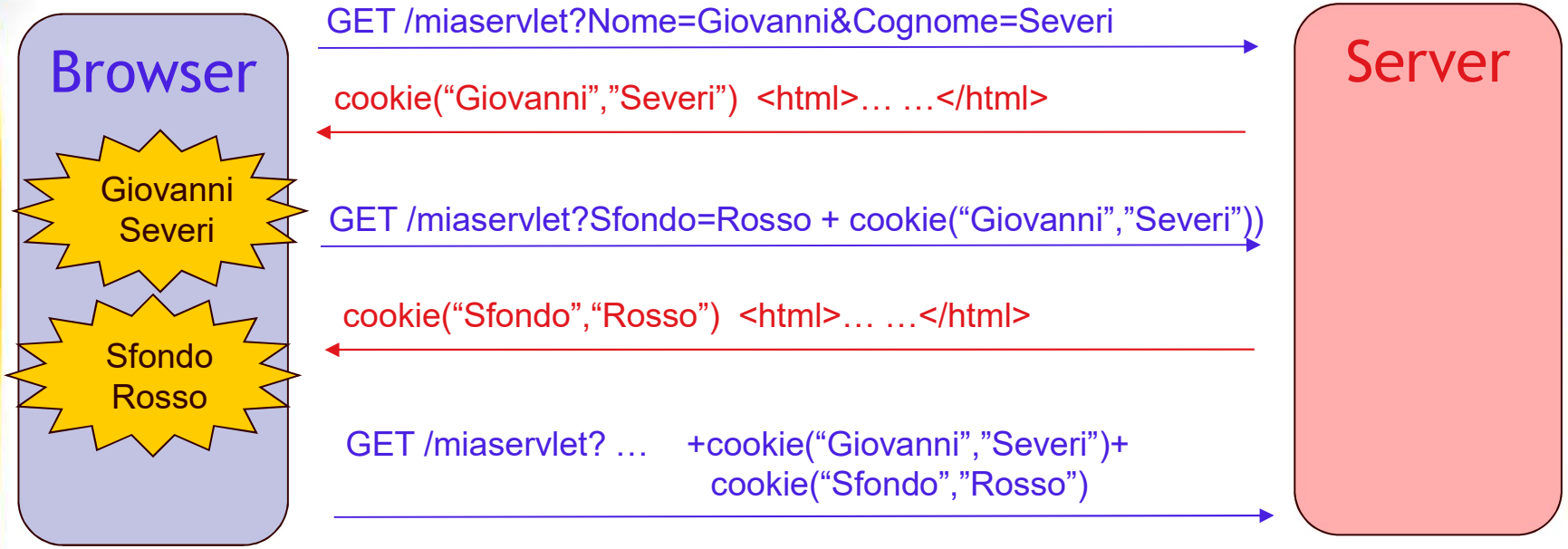
La sessione viene mantenuta solo navigando

attraverso FORM opportunamente riscritti

(*) Il testo della pagina html viene generato dinamicamente dalla servlet in funzione dei parametri ricevuti nella richiesta, sia che fossero hidden sia che fossero visibili nel form

Riassunto: gestione della sessione tramite cookie

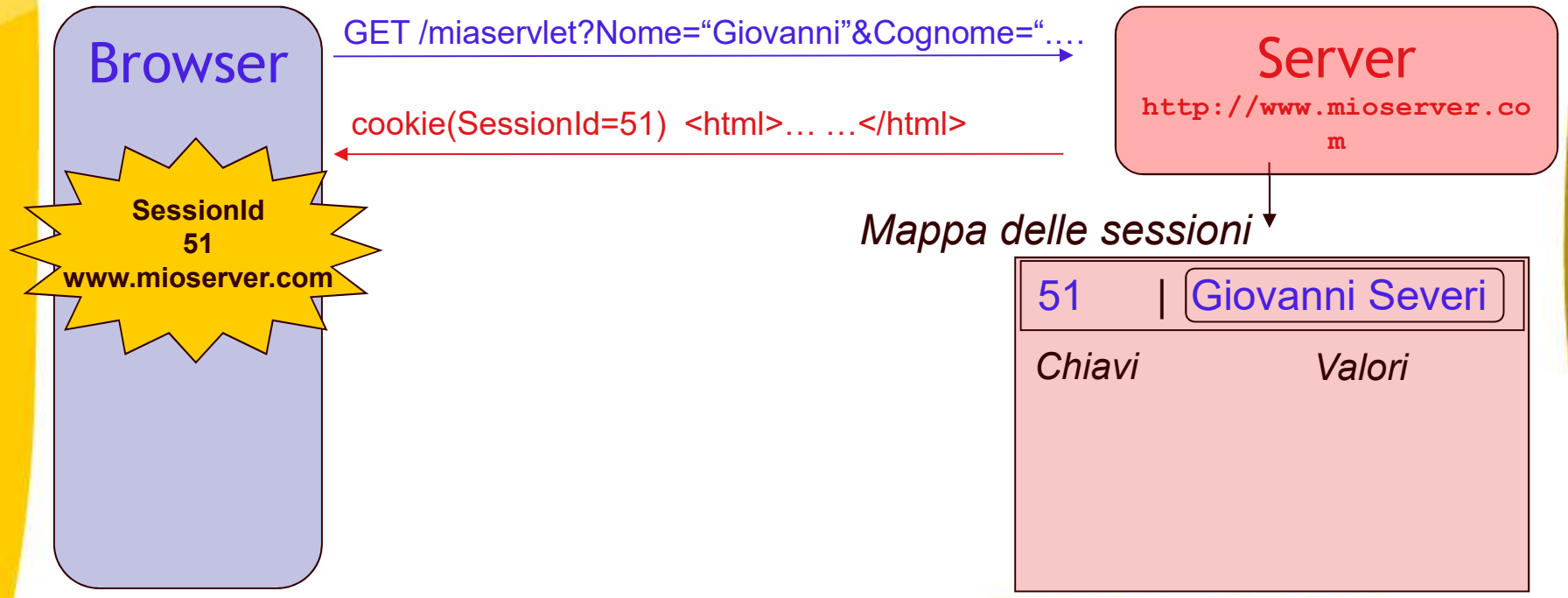
Riassunto: gestione della sessione tramite cookie



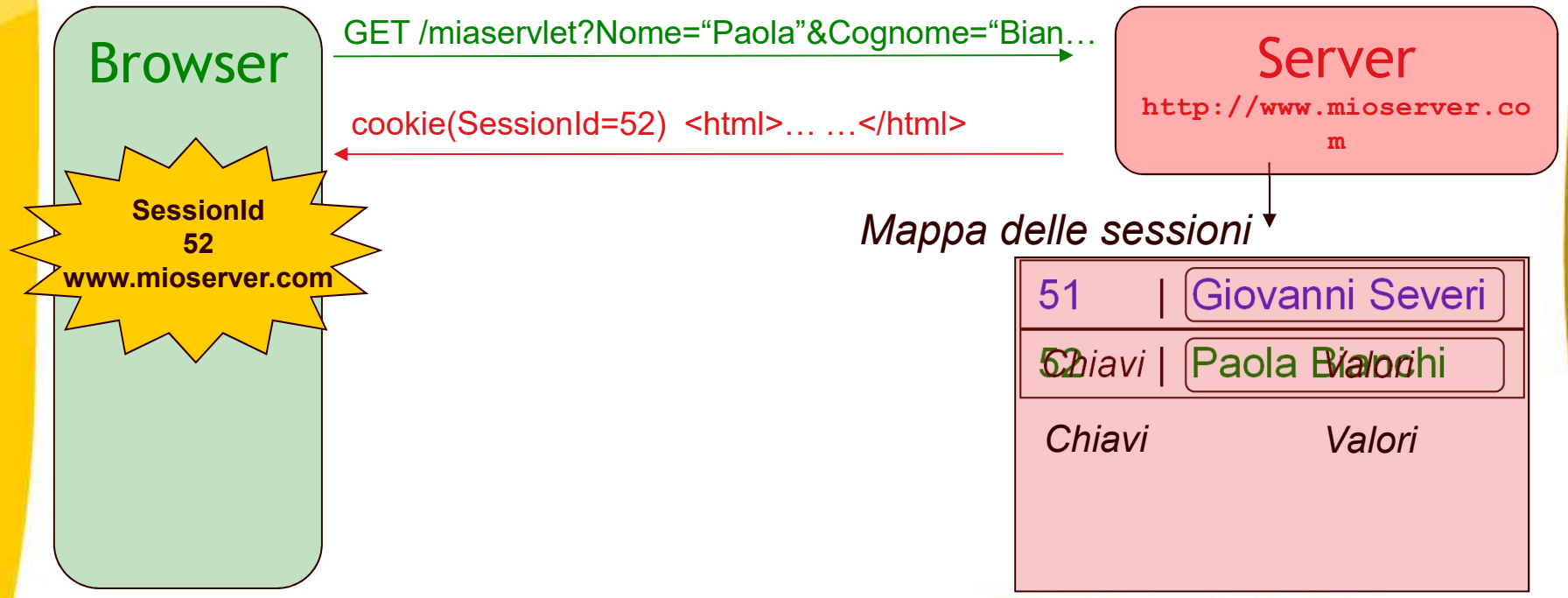
Gestione della sessione di navigazione tramite oggetti persistenti sul server

- Uso di oggetti che implementano l'interfaccia HttpSession

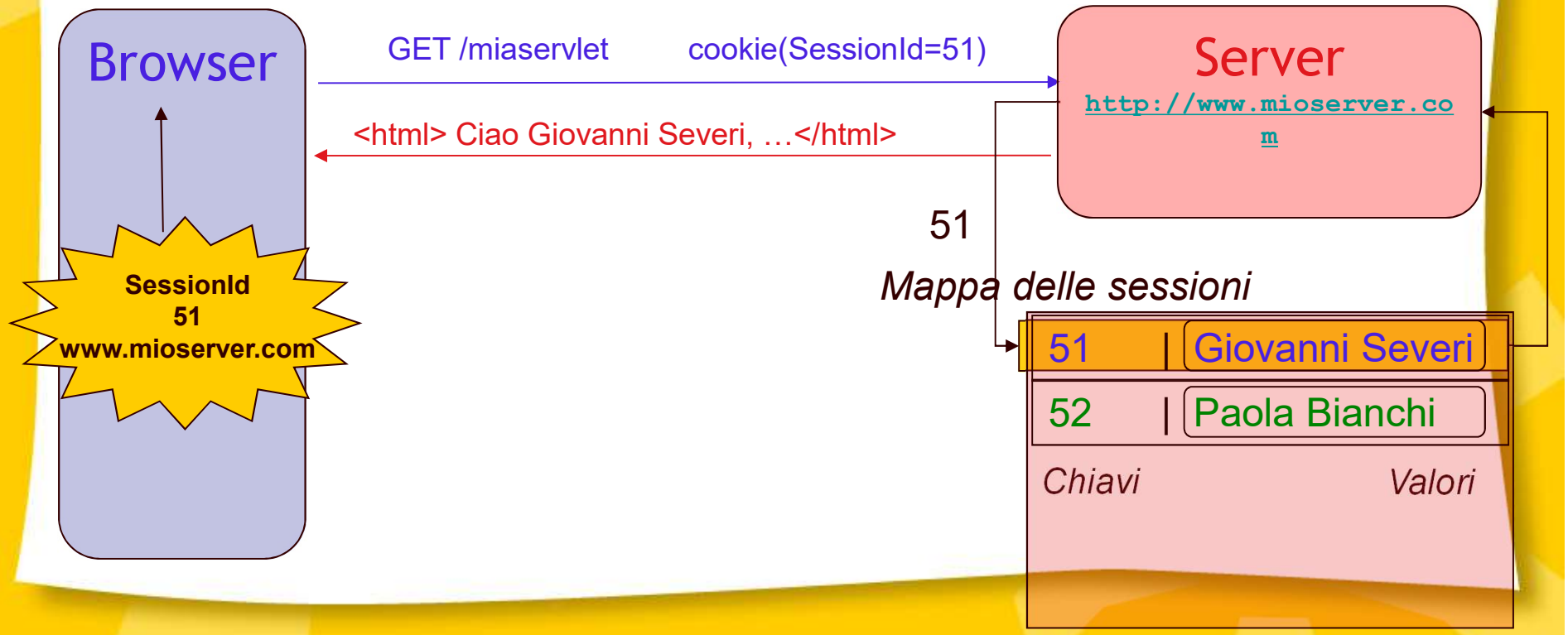
Gestione della sessione tramite oggetto persistente sul server (interfaccia **HttpSession**) (1/3)



Gestione della sessione tramite oggetto persistente sul server (interfaccia **HttpSession**) (2/3)



Gestione della sessione tramite oggetto persistente sul server (interfaccia **HttpSession**) (3/3)



Chiusura della sessione di navigazione

- Come detto, la cancellazione della sessione deve fare uso del metodo invalidate() dell'interfaccia HttpSession
- I cookie usati per la gestione della sessione sono cookie di sessione: scadono alla chiusura del browser

Uso dei cookie di sessione

- In pratica il server gestisce una **mappa**: ogni entry rappresenta una sessione di lavoro distinta
  - La chiave della mappa è un identificatore per la sessione
  - Il valore della mappa è un oggetto che contiene informazioni associate a quella specifica sessione
- In tutte le risposte al client il server aggiunge automaticamente e in maniera trasparente un cookie che contiene l'identificatore di sessione
- Attraverso questo identificatore, ad ogni richiesta il server è in grado di recuperare l'oggetto con le informazioni associate alla sessione

Supporto del container alla gestione della sessione di navigazione (1/2)

Tomcat (così come gli altri servlet container) mette a disposizione del programmatore una particolare infrastruttura e le API per una **gestione trasparente** di questo meccanismo:

- Nel processare una richiesta, il programmatore deve semplicemente chiedere all'oggetto `HttpRequest` l'oggetto associato alla sessione
- Il contenitore, in maniera trasparente, estrae dalla richiesta l'identificatore di sessione e lo usa per recuperare dalla mappa delle sessioni l'oggetto associato a quell'identificatore

Supporto del container alla gestione della sessione di navigazione (2/2)

- Se nella richiesta non viene trovato nessun identificatore allora, ove necessario (argomento true nel metodo getSession()), il container procede alla creazione della nuova sessione.
  - Vengono creati:
    - un nuovo identificatore,
    - un nuovo oggetto sessione,
    - una relativa entry nella mappa delle sessioni,
    - un cookie di sessione che viene agganciato alla risposta.
- Se il browser ha i cookie disabilitati come faccio a gestire la sessione sul server?

URL REWRITING nella gestione delle sessioni

- Se il browser ha i cookie disabilitati come faccio a gestire la sessione sul server?
- Si deve fare in modo che **tutti i percorsi utilizzati dal browser appendano all'url l'identificativo di sessione** (tecnica detta di **URL rewriting**).
  - N.B. il programmatore non conosce l'ID di sessione che verrà usato

Per una gestione semplice e trasparente delle operazioni di URL rewriting si ricorre al metodo dell'interfaccia HttpServletResponse:

java.lang.String **encodeURL**(java.lang.String url)

Dalla documentazione:

Encodes the specified URL by including the session ID in it, or, if encoding is not needed, returns the URL unchanged.

URL rewriting

Si deve poter rispondere a due questioni fondamentali:

1. Quali sono le URL che l'utente utilizzerà in futuro?
2. Come possiamo costringere l'utente ad appendere a queste URL l'identificativo di sessione?

Non possiamo sapere con certezza che richieste verranno effettuate dall'utente, ma possiamo controllare quelle effettuate attraverso gli hyperlink presenti nella pagina.

URL rewriting

- Se il client rifiuta i cookie, è possibile chiedere al contenitore di appendere l'identificatore della sessione agli URL degli hyperlink presenti nel codice HTML della risposta
  - Pagine di risposta generate da sessioni diverse conterranno URL diversi (con diversi identificativi di sessione appesi all'url)
  - Se la navigazione dell'utente procederà attraverso gli hyperlink la sessione verrà mantenuta: il client riproporrà al server l'identificatore della sessione nelle successive richieste HTTP
  - Identificazione trasparente della sessione

URL rewriting tramite il metodo encodeURL(…)

- Attenzione: il contenitore **riscrive gli URL solo se lo sviluppatore lo richiede esplicitamente**.
- Per realizzare questa operazione si usa il metodo **`String encodeURL(String url)`** di **`HttpServletResponse`**
  - il parametro **`url`** rappresenta l'URL non riscritto
  - il risultato del metodo è l'URL riscritto dal contenitore con appeso l'ID di sessione
- Se il programmatore richiede la riscrittura degli URL il comportamento del container è quello di commutare automaticamente tra le due modalità:
  - Se il browser del client accetta i cookie la sessione viene gestita solo con i cookie
  - Se i cookie vengono rifiutati, viene attivata la riscrittura degli URL

URL rewriting

- Poiché la riscrittura dell'URL avviene in modo trasparente da parte del contenitore
  - Lo sviluppatore deve solo usare **`encodeURL()`**
- Il contenitore si preoccupa di adottare la riscrittura degli url **solo** nel caso in cui i cookie vengono rifiutati
- E' quindi conveniente utilizzare sempre **`encodeURL()`** per rendere più robusta la gestione delle sessioni

Domande:

- Appurato che il meccanismo di gestione della sessione sul server basato su trasmissione di cookie potrebbe non funzionare, allora:
- 1. Perché non adottare sempre e solo l'url rewriting?
- 2. Perché non adottare entrambi i metodi contemporaneamente e non alternativamente?

Domanda 1: solo URL rewriting?

- Perché usare i cookie se a volte i browser li rifiutano? Non sarebbe stato meglio progettare container che utilizzassero sempre e solo l'URL rewriting?

  L'URL rewriting consente la gestione della sessione solo se l'utente naviga attraverso i link della pagina di risposta.

  La sessione viene persa se:

  - l'utente usa dei bookmark
  - l'utente usa il tasto backward raggiungendo una pagina richiesta precedentemente senza identificativo riscritto
  - l'utente scrive l'url sulla barra degli indirizzi del browser

Domanda 2: URL rewriting AND cookie?

- Perché il metodo encodeURL() non funziona semplicemente aggiungendo sempre l'id di sessione invece che farlo solo se il browser dell'utente non accetta i cookie?
- L'encoding dell'URL costituisce un carico sul server per riscrivere le stringhe
- Carico sulla rete perché ciascuna risposta conterrà diversi url riscritti (notare che nel caso di gestione tramite cookie il server invia il cookie di sessione solo al momento della sua creazione).

Logica di funzionamento del metodo encodeURL()

- L'interfaccia HttpServletResponse fornisce questo metodo:

  java.lang.String **encodeURL**(java.lang.String url)

Dalla documentazione:

Encodes the specified URL by including the session ID in it, or, if encoding is not needed, returns the URL unchanged

Logica di funzionamento del metodo encodeURL()

- Se il browser dell'utente accetta i cookie, il metodo lascia le URL inalterate
- Se il browser dell'utente NON accetta i cookie, il metodo effettua la riscrittura dell'URL passato come argomento
- COME FA QUESTO METODO A CAPIRE SE IL BROWSER DELL'UTENTE ACCETTA I COOKIE???

  Se la richiesta non contiene già dei cookie non ha modo di saperlo. Torniamo allora al primo esempio …

Logica di funzionamento del metodo encodeURL()

Prima che il server si accorga se il browser dell'utente accetta i cookie o no, le URL devono essere già state riscritte, altrimenti la sessione viene persa!

Logica di funzionamento del metodo encodeURL(): prima richiesta di una sessione

- All'atto della **creazione** dell'oggetto sessione, il **cookie** di sessione viene **sempre automaticamente aggiunto** all'oggetto rappresentativo della risposta
- La prima volta che viene utilizzato il metodo encodeURL nel corso di una sessione, il container può non sapere se il browser accetti i cookie o no.
- Al primo utilizzo del metodo encodeURL vengono inviati sia il cookie di sessione che le URL riscritte (solo quelle per cui lo sviluppatore avrà richiesto la riscrittura).

Logica di funzionamento del metodo encodeURL(): richieste successive alla prima di una sessione

- All'arrivo di richieste successive a quella che ha generato la sessione corrente, viene controllato se l'ID di sessione è stato ottenuto 1) anche (o solo) tramite un cookie o 2) non è stato ottenuto da un cookie.
  - Nel primo caso non viene effettuato encoding,
  - Nel secondo caso l'URL viene riscritta con appeso l'ID di sessione.

Supporto dell'interfaccia HttpServletRequest alle operazioni di encoding dell'URL

- L'interfaccia HttpSerlvetRequest fornisce i seguenti metodi:
  - boolean **isRequestedSessionIdFromCookie()**

    Checks whether the requested session ID came in as a cookie.

  - boolean **isRequestedSessionIdFromURL()**

    Checks whether the requested session ID came in as part of the request URL.