Programmazione WEB

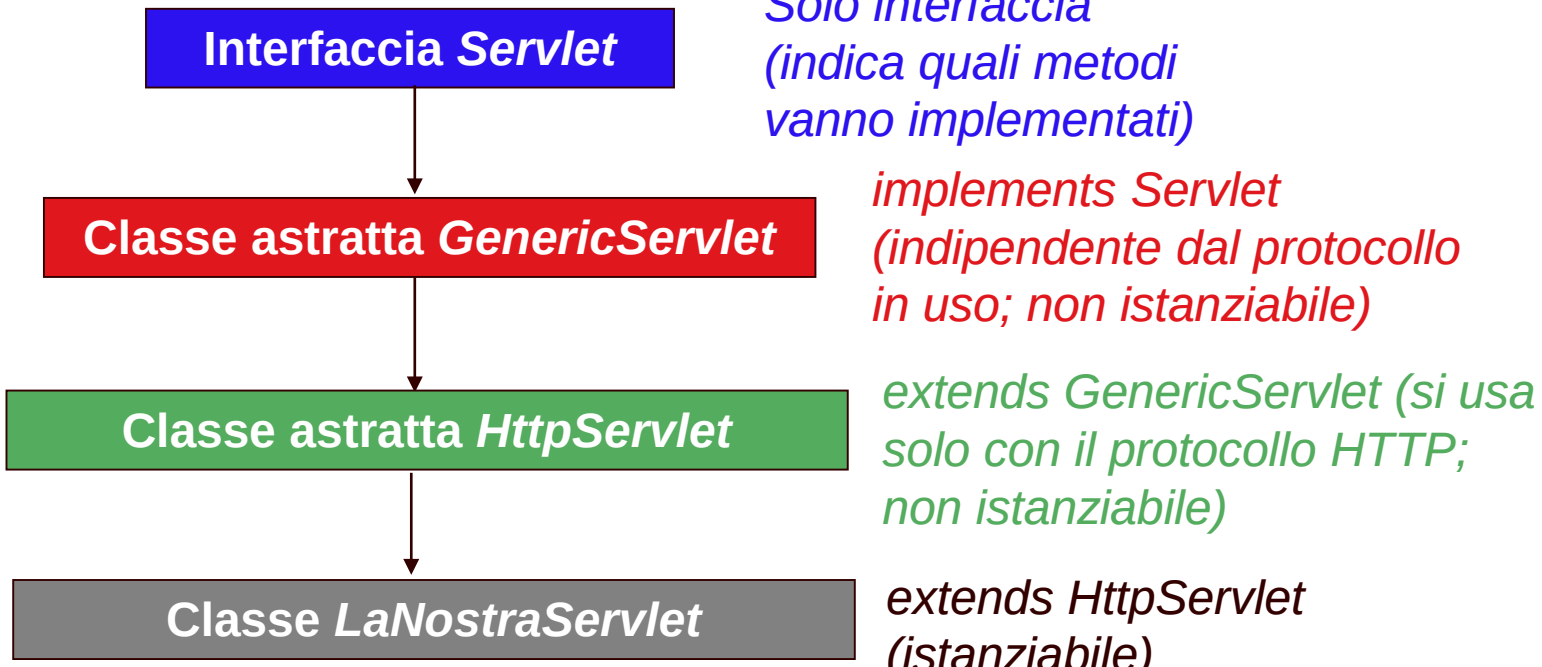
Lezione del 12 Marzo 2020

Docente: Novella Bartolini
Ricevimento: Mercoledì ore 12:30-14:00
Via Salaria 113, terzo piano, stanza 309
Email: bartolini@di.uniroma1.it

Servlet

- Programma applicativo che viene eseguito da un server
  - Accoglie ed elabora richieste provenienti dal client (attraverso comandi http: POST o GET, ma anche attraverso altri protocolli)
  - Produce una risposta HTTP contenente codice HTML generato dinamicamente
- Molto diffuse per applicazioni che fanno uso di database
  - Thin clients (client molto semplici che richiedono supporto minimo)
  - Meccanismo request/response

Servlet API



Solo interfaccia (indica quali metodi vanno implementati)

implements Servlet (indipendente dal protocollo in uso; non istanziabile)

extends GenericServlet (si usa solo con il protocollo HTTP; non istanziabile)

extends HttpServlet (istanziabile)

Servlet

- Le servlet possono essere utilizzate qualunque sia il servizio espletato dal server, ovvero qualunque sia il protocollo di interazione client/server: es HTTP, FTP
- Nella sua forma più generale, una servlet è un'estensione di una classe javax.servlet.GenericServlet che implementa l'interfaccia javax.servlet.Servlet
- Le servlet usate nel web sono estensioni della classe javax.servlet.http.HttpServlet che implementa l'interfaccia javax.servlet.Servlet

Interfaccia `Servlet`

- Interfaccia `Servlet` (package *`javax.servlet`*)
  - Tutte le servlet devono implementare questa interfaccia
  - Tutti i metodi dell'interfaccia `Servlet` vengono invocati dal servlet container secondo un prefissato *ciclo di vita*
    - (vi ricordate come funziona con le applet? Init-start-paint-stop-destroy vengono sempre chiamati in sequenza dal browser per eseguire una applet)

Ciclo di vita di una servlet

- Caricamento della servlet in memoria
- Il container delle servlet invoca il metodo `init`
  - Solo in questo momento la servlet è in grado di rispondere alla prima richiesta
  - Inizializzazione delle variabili globali
  - Invocato una sola volta
- Il metodo `service` gestisce le richieste
  - Riceve la richiesta
  - Elabora la richiesta
  - Confeziona un oggetto risposta
  - Invocato ad ogni richiesta del client
- Il metodo `destroy` rilascia le risorse allocate dalla servlet quando il container la termina

Classi astratte per definire le Servlet

- Esistono due classi astratte che implementano l'interfaccia *Servlet*
  - **`GenericServlet` (package `javax.servlet`)**
  - **`HttpServlet` (package `javax.servlet.http`)**
    (quest'ultima implementa l'interfaccia servlet indirettamente, estendendo GenericServlet)
- Queste classi forniscono l'implementazione di default di tutti i metodi dell'interfaccia *Servlet*
- Il metodo chiave è *service*:

  riceve gli oggetti ServletRequest e ServletResponse che forniscono accesso agli stream di i/o permettendo la ricezione e l'invio di informazioni al client

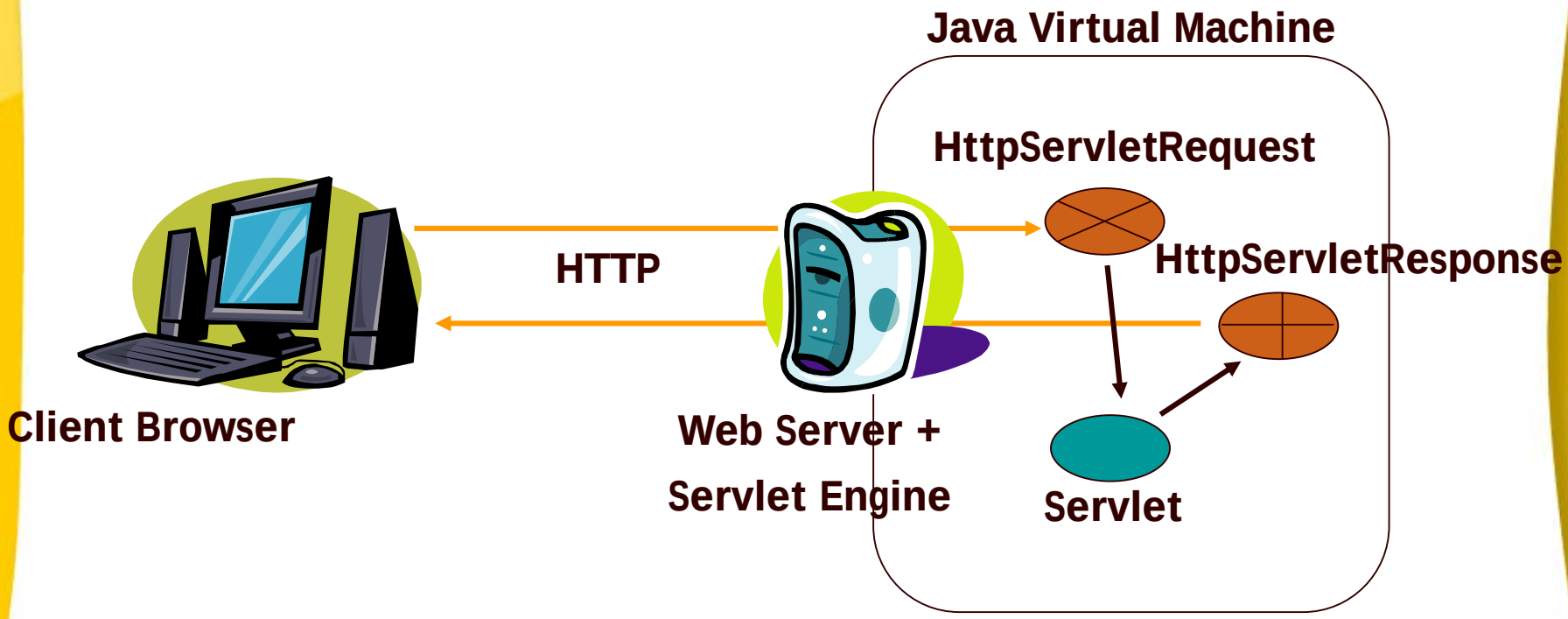
Metodi dell'interfaccia `Servlet`

Method	Description
void init(ServletConfig config)	
	The servlet container calls this method once during a servletÕsexecution cycle to initialize the servlet. The ServletConfig argument is supplied by the servlet container that executes the servlet.
ServletConfig getServletConfig()	
	This method returns a reference to an object that implements interface ServletConfig. This object provides access to the servletÕsconfiguration information such as servlet initialization parameters and the servletÕsServletContext, which provides the servlet with access to its environment (i.e., the servlet container in which the servlet executes).
String getServletInfo()	
	This method is defined by a servlet programmer to return a string containing servlet information such as the servletÕsauthor and version.
void service(ServletRequest request, ServletResponse response)	
	The servlet container calls this method to respond to a client request to the servlet.
void destroy()	
	This ÑcleanupÓmethod is called when a servlet is terminated by its servlet container. Resources used by the servlet, such as an open file or an open database connection, should be deallocated here.

Funzionamento delle servlet HTTP

- Il server web riceve dal browser del client una richiesta HTTP GET o POST
- Il server web direziona la richiesta HTTP al motore servlet o servlet engine (pr. èngine) o servlet container
- Se la servlet non è ancora in memoria viene caricata e inizializzata dal servlet engine (esecuzione del metodo init)
- Il servlet engine incapsula la richiesta HTTP in una classe **HttpServletRequest** e la passa al metodo doPost o doGet della servlet
- La servlet risponde scrivendo il codice HTML nella **HttpServletResponse** che viene rimandata al web server e poi riconsegnata al client via HTTP

Servlet e Web Server



La classe `HttpServlet`

- Sovrascrive il metodo `service` della classe `GenericServlet`
- Prevede i metodi per rispondere alle richieste HTTP
  - `GET`
  - `POST (ma anche HEAD, PUT, OPTIONS ecc.)`
- Il metodo `service()` invoca i metodi `doGet` e `doPost`
  - Il metodo `doGet()` risponde alle richieste `GET`
  - Il metodo `doPost()` risponde alle richieste `POST`
    - Ricevono come parametri gli oggetti `HttpServletRequest` e `HttpServletResponse`
- `Non implementare il metodo service se si usa questo tipo di servlet`

Altri metodi della classe `HttpServlet`

Method	Description
`doDelete`	Called in response to an HTTP *`delete`* request. Such a request is normally used to delete a file from a server. This may not be available on some servers, because of its inherent security risks (e.g., the client could delete a file that is critical to the execution of the server or an application).
`doHead`	Called in response to an HTTP *`head`* request. Such a request is normally used when the client only wants the headers of a response, such as the content type and content length of the response.
`doOptions`	Called in response to an HTTP *`options`* request. This returns information to the client indicating the HTTP options supported by the server, such as the version of HTTP (1.0 or 1.1) and the request methods the server supports.
`doPut`	Called in response to an HTTP *`put`* request. Such a request is normally used to store a file on the server. This may not be available on some servers, because of its inherent security risks (e.g., the client could place an executable application on the server, which, if executed, could damage the server—perhaps by deleting critical files or occupying resources).
`doTrace`	Called in response to an HTTP *`trace`* request. Such a request is normally used for debugging. The implementation of this method automatically returns an HTML document to the client containing the request header information (data sent by the browser as part of the request).

Interfaccia `HttpServletRequest`

- Il servlet engine che esegue la servlet
  - Crea un oggetto **`HttpServletRequest`**
  - Lo passa al metodo **`service`** della servlet http
- L'oggetto **`HttpServletRequest`** contiene la richiesta del client
- Esistono molti metodi per estrarre informazioni dalla richiesta del client. Ne vediamo alcuni →

Interfaccia `HttpServletRequest` (Cont.)

Method	Description
String getParameter(String name)	
	Obtains the value of a parameter sent to the servlet as part of a get or post request. The name argument represents the parameter name.
Enumeration getParameterNames()	
	Returns the names of all the parameters sent to the servlet as part of a post request.
String[] getParameterValues(String name)	
	For a parameter with multiple values, this method returns an array of strings containing the values for a specified servlet parameter.
Cookie[] getCookies()	
	Returns an array of Cookie objects stored on the client by the server. Cookie objects can be used to uniquely identify clients to the servlet.
HttpSession getSession(boolean create)	
	Returns an HttpSession object associated with the clientÕs current browsing session. This method can create an HttpSession object (true argument) if one does not already exist for the client. HttpSession objects are used in similar ways to Cookies for uniquely identifying clients.

Interfaccia `HttpServletResponse`

- Il servlet engine
  - Crea un oggetto `HttpServletResponse`
  - Lo passa al metodo `service` della servlet (attraverso i metodi doGet e doPost)
- Ogni chiamata ai metodi `doGet` e `doPost` riceve un oggetto che implementa l'interfaccia `HttpServletResponse`
- Esistono molti metodi che consentono la scrittura della risposta da inviare al client. Ne vediamo alcuni →

Interfaccia `HttpServletResponse`

Method	Description
void addCookie(Cookie cookie)	
	Used to add a Cookie to the header of the response to the client. The CookieÕs maximum age and whether Cookies are enabled on the client determine if Cookies are stored on the client.
ServletOutputStream getOutputStream()	
	Obtains a byte-based output stream for sending binary data to the client.
PrintWriter getWriter()	
	Obtains a character-based output stream for sending text data to the client.
void setContentType(String type)	
	Specifies the MIME type of the response to the browser. The MIME type helps the browser determine how to display the data (or possibly what other application to execute to process the data). For example, MIME type "text/html" indicates that the response is an HTML document, so the browser displays the HTML page.

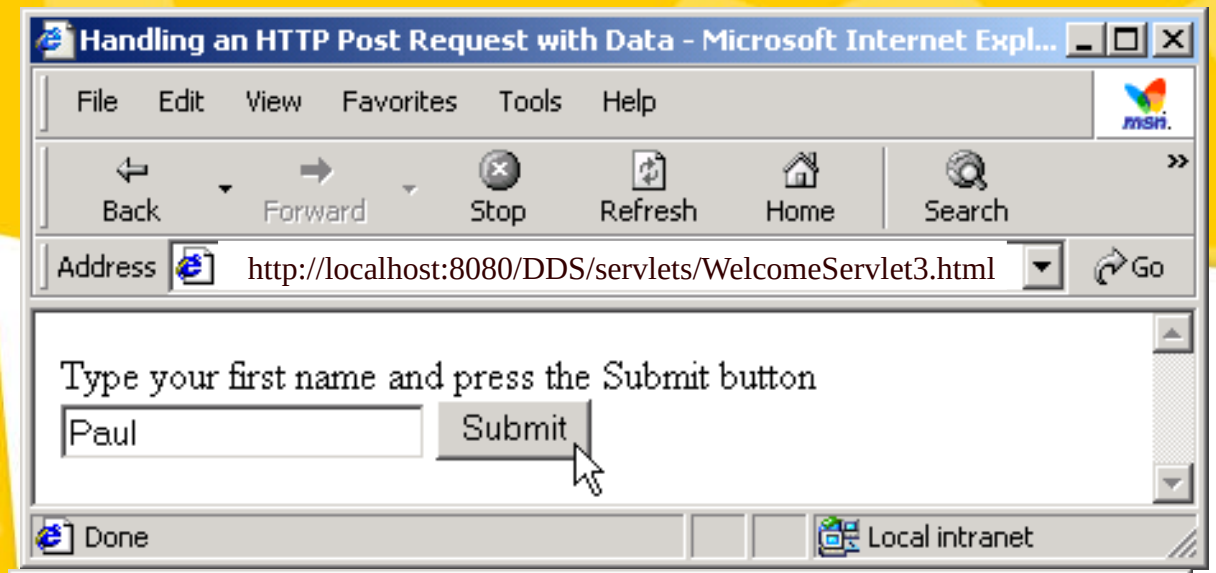
```
1  <?xml version = "1.0"?>
2  <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
3    "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
4
5  <!-- WelcomeServlet3.html -->
6
7  <html xmlns = "http://www.w3.org/1999/xhtml">
8  <head>
9    <title>Handling an HTTP Post Request with Data</title>
10 </head>
11
12 <body>
13   <form action = "/DirectoryDiSaluto/welcome3" method = "post">
14
15     <p><label>
16       Type your first name and press the Submit button
17       <br /><input type = "text" name = "firstname" />
18       <input type = "submit" value = "Submit" />
19     </label></p>
20
21   </form>
22 </body>
23 </html>
```

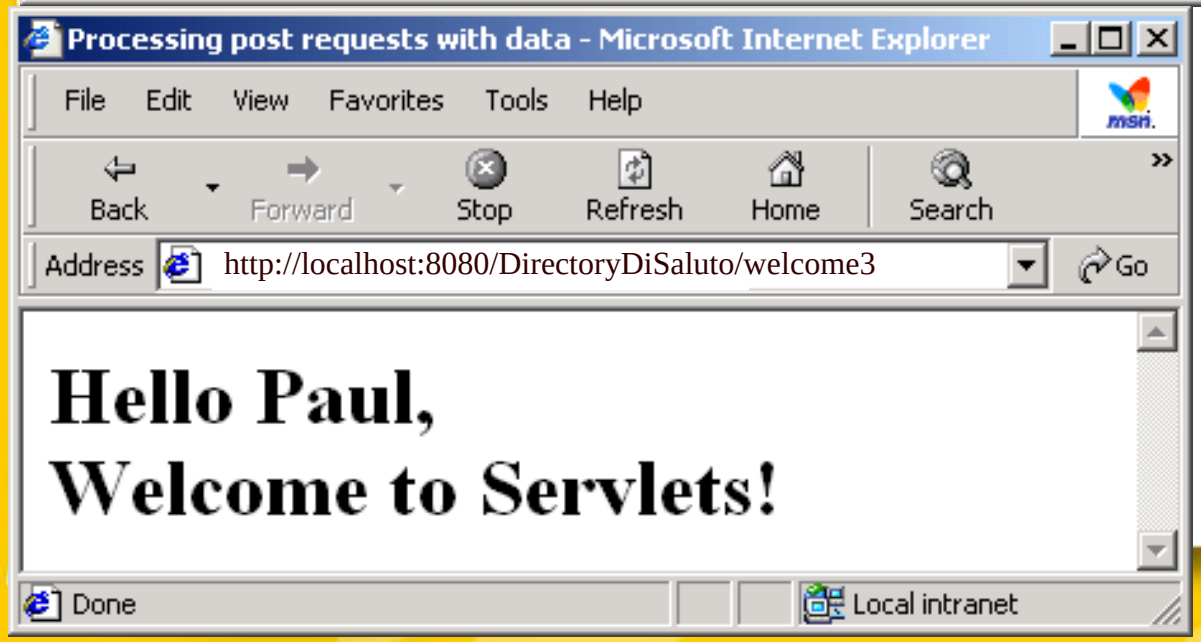
Viene fornito un **form** in cui l'utente può scrivere il proprio nome nell'elemento di input **firstname** di tipo **text**. Quando viene cliccato il bottone **Submit** viene invocata **WelcomeServlet3**.

```
2  // Processing post requests containing data.
3
4
5  import javax.servlet.*;
6  import javax.servlet.http.*;
7  import java.io.*;
8
9  public class WelcomeServlet3 extends HttpServlet {
10
11    // process "post" request from client
12    protected void doPost( HttpServletRequest request,
13      HttpServletResponse response )
14        throws ServletException, IOException
15    {
16      String firstName = request.getParameter( "firstname" );
17
18      response.setContentType( "text/html" );
19      PrintWriter out = response.getWriter();
20
21      // send XHTML page to client
22
23      // start XHTML document
24      out.println( "<?xml version = \"1.0\"?>" );
25
26      out.println( "<!DOCTYPE html PUBLIC \"-//W3C//DTD " +
27        "XHTML 1.0 Strict//EN\" \"http://www.w3.org" +
28        "/TR/xhtml1/DTD/xhtml1-strict.dtd\">" );
29
30      out.println(
31        "<html xmlns = \"http://www.w3.org/1999/xhtml\">" );
32
```

Define a **doPost** method to responds to post requests.

```
33      // head section of document
34      out.println( "<head>" );
35      out.println(
36        "<title>Processing post requests with data</title>" );
37      out.println( "</head>" );
38
39      // body section of document
40      out.println( "<body>" );
41      out.println( "<h1>Hello " + firstName + ",<br />" );
42      out.println( "Welcome to Servlets!</h1>" );
43      out.println( "</body>" );
44
45      // end XHTML document
46      out.println( "</html>" );
47      out.close();  // close stream to complete the page
48   }
49 }
```


Handling an HTTP Post Request with Data - Microsoft Internet Expl...
File Edit View Favorites Tools Help
Back Forward Stop Refresh Home Search
Address http://localhost:8080/DDS/servlets/WelcomeServlet3.html
Go
Type your first name and press the Submit button
Paul
Submit
Done
Local intranet


Processing post requests with data - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Forward Stop Refresh Home Search
Address http://localhost:8080/DirectoryDiSaluto/welcome3
Go
Hello Paul,
Welcome to Servlets!
Done
Local intranet

Struttura della directory di una web application

- Context root – top level directory di una applicazione web

  Si crea una sottodirectory nella directory webapps di Tomcat

Struttura della directory di una web application

Directory	Description
context root (nome che volete)	E' la directory radice della vostra applicazione web. Il nome è scelto dallo sviluppatore dell'applicazione (voi). Contiene tutti i file dell'applicazione, eventualmente organizzati in sottocartelle (file JSP, pagine HTML, immagini incorporate nelle pagine, possono risiedere dove volete, classi delle servlet e altre classi di supporto devono essere organizzate in cartelle e package. Si possono inserire sottocartelle nella context root per organizzarne meglio i contenuti. Negli esempi che seguono la context root si chiama **`directorydisaluto`**.
`WEB-INF`	Questa cartella contiene un file detto *deployment descriptor* (***`web.xml`***).
`WEB-INF/classes`	Questa cartella contiene le classi delle servlet e altre classi di supporto usate dalla web application. Se le classi fanno parte di un package, la struttura completa del package deve cominciare da questa cartella.
`WEB-INF/lib`	Questa cartella contiene archivi (JAR). I file JAR files possono contenere I file delle classi servlet e altre classi di supporto che sono utilizzate in una web application.

Deployment descriptor (web.xml)

- Dopo avere configurato l'albero delle directory, dobbiamo configurare l'applicazione web in modo che possa gestire le richieste

→ Si usa un file web.xml che chiamiamo

deployment descriptor

Specifica diversi parametri come:

- il nome della classe che definisce la servlet,
- i percorsi che causano l'invocazione della servlet da parte del container

```
<web-app>

    <!-- General description of your W
    <display-name>
      NomeDisplayDellaWebApplication
      Viene visualizzato dall'admin
    </display-name>

   <description>
      Si tratta di un'applcazione dove facciamo
      Esempi di servlet.
    </description>


<!-- Servlet definitions -->
    <servlet>

      <servlet-name>Paperino</servlet-name>

      <description>
        Una servlet che gestisce le richieste di GET
      </description>

      <servlet-class>
       WelcomeServlet3
      </servlet-class>
    </servlet>
```

L'elemento <**web-app**> definisce la configurazione di tutte le servlet della applicazione web

L'elemento <**display-name**> specifica un nome che viene utilizzato dall'amministratore del server (e visualizzato dal pannello del manager)

L'elemento <**description**> specifica una descrizione della applicazione che viene visualizzata nel pannello del manager

L'elemento <**servlet**> descrive una specifica servlet: se ci sono più servlet ci saranno più elementi <servlet>

L'elemento <**servlet-name**> definisce un nome per la servlet all'interno di questo file

L'elemento <**description**> fornisce una descrizione di questa servlet specifica

L'elemento <**servlet-class**> specifica il nome completo della classe servlet compilata

```
    <!-- Servlet mappings -->
     <servlet-mapping>
        <servlet-name>Paperino</servlet-name>
        <url-pattern>/AliasServletDiSaluto</url-pattern>
     </servlet-mapping>

</web-app>
```

L'elemento <**servlet-mapping**> specifica l'associazione tra il nome dato alla servlet nell'interno del file web.xml (<**servlet-name**>) e l'<**url-pattern**> con cui la servlet potrà essere invocata

n.b.: non si specifica la context root nell'url-pattern!

Ricordiamoci che all'interno del server la tecnologia è java, mentre il colloquio con il client avviene nel protocollo HTTP.

L'associazione tra l'URL usata nelle richieste HTTP e la classe java che deve essere eseguita dal server è lo scopo principale del deployment descriptor.

Tale associazione è creata nel file web.xml attraverso l'elemento <servlet-name> che viene associato prima con il nome della classe (<servlet-class>) e poi con l'URL (<url-pattern>).

n.b. se la servlet fosse stata parte di un package, avremmo dovuto includere nella directory **classes** l'intera struttura del package

```
31    <!-- Servlet mappings -->
32    <servlet-mapping>
33       <servlet-name>Paperino</servlet-name>
34       <url-pattern>/A/B/topolino</url-pattern>
35    </servlet-mapping>
36
37 </web-app>
```

L'elemento <**servlet-mapping**> specifica l'associazione tra il nome dato alla servlet nell'interno del file web.xml (<**servlet-name**>) e l'<**url-pattern**> con cui la servlet potrà essere invocata

Notare che /A e /B non corrispondono a nessuna cartella sul server!

Uso di percorsi relativi in una Web Application (1)

- Invocazione della servlet attraverso un **percorso relativo al server**:

  "/DirectoryDiSaluto/AliasServletDiSaluto"

  - **`/DirectoryDiSaluto`** specifica la context root
  - **`/AliasServletDiSaluto`** specifica l'URL pattern
  - Si tratta di un percorso relativo al server riferito alla radice del server (dir. webapps), comincia con il carattere "/"
  - Questo percorso va bene qualunque sia la posizione del file chiamante all'interno del server (anche al di fuori della context root, da un'altra applicazione purchè all'interno dello stesso server)

```
<form action = "/DirectoryDiSaluto/AliasServletDiSaluto" method = "get">
```

Uso di percorsi relativi in una web application (2)

- Invocazione della servlet attraverso un **percorso relativo al file chiamante**:

  "percorso_x/AliasServletDiSaluto"

  - percorso_x specifica il percorso dalla cartella contenente il file chiamante alla context root dell'applicazione
  - Si tratta di un percorso relativo al file chiamante riferito alla posizione del file in cui è specificato l'url della servlet che si vuole invocare.
  - Il percorso relativo comincia senza il carattere "/"
  - Questo percorso va bene solo per la posizione del file chiamante e non può essere utilizzato in altre applicazioni, nemmeno se girano sullo stesso server

Uso dei percorsi relativi in una web application (3)

WelcomeServlet Web application directory and file structure

```
+ DirectoryDiSaluto
    - index.html
    + htmls
        - HtmlDiSaluto.html
    + WEB-INF
        - web.xml
        + classes
          -  ServletDiSaluto.class
```

Nel file web.xml l'url pattern indicato è
<url-pattern>/AliasServletDiSaluto</url-pattern>

Nel file index.html la servlet viene invocata nel seguente modo:

```
<form action = "AliasServletDiSaluto" method = "get">
```

Nel file HtmlDiSaluto.html la servlet viene invocata nel seguente modo:

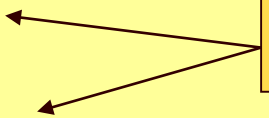
```
<form action = "../AliasServletDiSaluto" method = "get">
```

REDIREZIONE DI RICHIESTE

Inoltro delle richieste ad altre risorse

- Servlet di esempio **RedirectServlet**
  - Inoltra la richiesta ad una risorsa diversa
  - Fa uso del metodo sendRedirect() dell'oggetto HttpServletResponse
  - Il parametro di ingresso è una stringa: il percorso di destinazione della ridirezione
    - Il metodo accetta percorsi assoluti e relativi.
    - Un percorso relativo senza "/" iniziale viene interpretato rispetto alla URL corrente
    - Un percorso relativo con la "/" iniziale viene interpretato rispetto alla dir. radice del container

```
1  <?xml version = "1.0"?>
2  <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
3    "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
4
5  <!--  RedirectServlet.html -->
6
7  <html xmlns = "http://www.w3.org/1999/xhtml">
8  <head>
9    <title>Redirecting a Request to Another Site</title>
10 </head>
11
12 <body>
13   <p>Click a link to be redirected to the appropriate page</p>
14   <p>
15   <a href = "/DirectoryDiSaluto/redirect?page=deitel">
16     www.deitel.com</a><br />
17   <a href = "/DirectoryDiSaluto/redirect?page=welcome1">
18     Welcome servlet</a>
19   </p>
20 </body>
21 </html>
```



Fornisce hyperlink che invocano **RedirectServlet**

```
1
2  // Redirecting a user to a different Web page.
3
4
5  import javax.servlet.*;
6  import javax.servlet.http.*;
7  import java.io.*;
8
9  public class RedirectServlet extends HttpServlet {
10
11    // process "get" request from client
12    protected void doGet( HttpServletRequest request,
13      HttpServletResponse response )
14        throws ServletException, IOException
15    {
16      String location = request.getParameter( "page" );
17
18      if ( location != null )
19
20        if ( location.equals( "deitel" ) )
21          response.sendRedirect( "http://www.deitel.com" );
22        else
23          if ( location.equals( "welcome1" ) )
24            response.sendRedirect( "welcome1" );
25
26        // codice che viene eseguito solo se questa servlet non riesce a redirigere
27        // la richiesta come voluto
28
29      response.setContentType( "text/html" );
30      PrintWriter out = response.getWriter();
31
```

Attenzione all'uso di percorsi relativi all'interno di una servlet:

per non sbagliare pensate sempre attentamente a come viene formulato l'url per intero una volta che viene passato al browser

Il percorso relativo welcome1 viene aggiunto al percorso della servlet chiamante il metodo sendRedirect

Viene ottenuto il parametro **page** dalla richiesta.

Si valuta se il valore di **page** sia "deitel" o "welcome1"

Si inoltra la richiesta all'url: www.deitel.com.

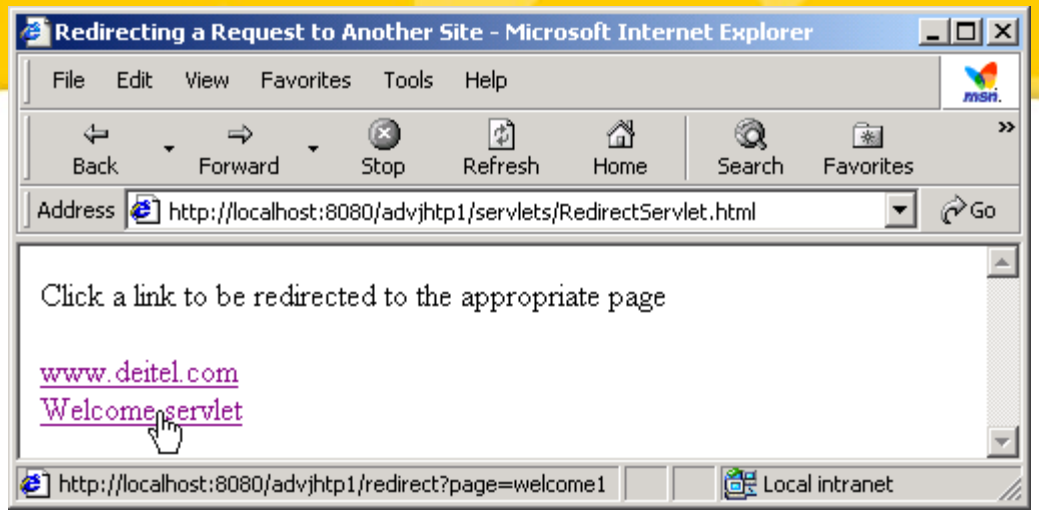
Si inoltra la richiesta al servlet **WelcomeServlet1**.

Pagina web di output che viene visualizzata nel caso in cui si sia ricevuta una richieta non valida (non viene invocato il metodo **sendRedirect**).

```
32      // start XHTML document
33      out.println( "<?xml version = \"1.0\"?>" );
34
35      out.println( "<!DOCTYPE html PUBLIC \"-//W3C//DTD " +
36        "XHTML 1.0 Strict//EN\" \"http://www.w3.org" +
37        "/TR/xhtml1/DTD/xhtml1-strict.dtd\">" );
38
39      out.println(
40        "<html xmlns = \"http://www.w3.org/1999/xhtml\">" );
41
42      // head section of document
43      out.println( "<head>" );
44      out.println( "<title>Invalid page</title>" );
45      out.println( "</head>" );
46
47      // body section of document
48      out.println( "<body>" );
49      out.println( "<h1>Invalid page requested</h1>" );
50      out.println( "<p><a href = " +
51        "\"servlets/RedirectServlet.html\">" );
52      out.println( "Click here to choose again</a></p>" );
53      out.println( "</body>" );
54
55      // end XHTML document
56      out.println( "</html>" );
57      out.close();  // close stream to complete the page
58    }
59  }
```

Si noti come l'invocazione di altre risorse facenti parte della stessa web application non richieda di specificare esplicitamente la context root.

Si assume che la servlet invocata si trovi nella stessa context root a meno che non venga specificata una URL completa.


Redirecting a Request to Another Site - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Forward Stop Refresh Home Search Favorites
Address http://localhost:8080/advjhtp1/servlets/RedirectServlet.html
Go
Click a link to be redirected to the appropriate page
www.deitel.com
Welcome servlet
http://localhost:8080/advjhtp1/redirect?page=welcome1
Local intranet


A Simple Servlet Example - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Forward Stop Refresh Home Search Favorites
Address http://localhost:8080/advjhtp1/welcome1
Go
Welcome to Servlets!
Done
Local intranet

Inoltro delle richieste ad altre risorse: modifiche al file web.xml

Descriptor element	Value
servlet *element*	
servlet-name	**redirect**
description	**Redirecting to static Web pages and other servlets.**
servlet-class	**RedirectServlet**
servlet-mapping *element*	
servlet-name	**redirect**
url-pattern	**/redirect**

??????

Come mai per redirigere una richiesta faccio uso di un metodo dell'oggetto HttpServletResponse response**?**

Alcune precisazioni...

1. L'oggetto della classe **`HttpServletResponse`** su cui viene invocato il metodo **`sendRedirect(String location)`** viene comunque utilizzato dal web server per costruire la risposta HTTP.
2. La risposta HTTP contiene un header di redirezione verso la nuova locazione
   <meta http-equiv="refresh" content="0" url=http://example.com/" />
3. L'header di redirezione viene interpretato dal browser del client
4. Il client spedisce automaticamente una nuova richiesta verso la nuova locazione.

Alcune precisazioni (continua)

- La locazione può anche essere esterna alla applicazione web da cui viene invocato il metodo (il metodo sendRedirect accetta sia percorsi relativi che assoluti)
- La redirezione coinvolge il client (che può decidere di non accettare redirezioni)
- Può essere utilizzata esclusivamente per richieste di GET (la specifica HTTP richiede che tutte le richieste di redirezione debbano essere inoltrate attraverso richieste di GET) – non si può usare sendRedirect per inviare richieste di POST
- I parametri della richiesta sono visibili al client (appesi all'URL nella richiesta di GET)

Documentazione dell'interfaccia HttpServletResponse, metodo sendRedirect

- **sendRedirect**
- public void **sendRedirect**(java.lang.String location) throws java.io.IOException
  - Sends a temporary redirect response to the client using the specified redirect location URL. This method can accept relative URLs; the servlet container must convert the relative URL to an absolute URL before sending the response to the client. If the location is relative without a leading '/' the container interprets it as relative to the current request URI. If the location is relative with a leading '/' the container interprets it as relative to the servlet container root. If the response has already been committed, this method throws an IllegalStateException. After using this method, the response should be considered to be committed and should not be written to.
  - **Parameters:**
    - location - the redirect location URL
  - **Throws:**
    - java.io.IOException - If an input or output exception occurs
    - IllegalStateException - If the response was committed or if a partial URL is given and cannot be converted into a valid URL

Session Tracking

- Personalizzazione delle informazioni presenti nella pagina web
- Problema della tutela delle informazioni private
- HTTP – è un protocollo stateless
  - Non supporta la persistenza delle informazioni
- Per tracciare i client individualmente:
  - Cookie
  - Session tracking
  - **`hidden`** type **`input`**
  - URL rewriting (parametri get)

I cookie

- Vengono memorizzati sul computer dell'utente (client) per utilizzi successivi
- Dati di tipo testo, spediti da servlet
- Età massima di un cookie
- Cancellati automaticamente quando scadono

- Esempio: servlet **CookieServlet**
  - Gestisce sia richieste di **get** che di **post**

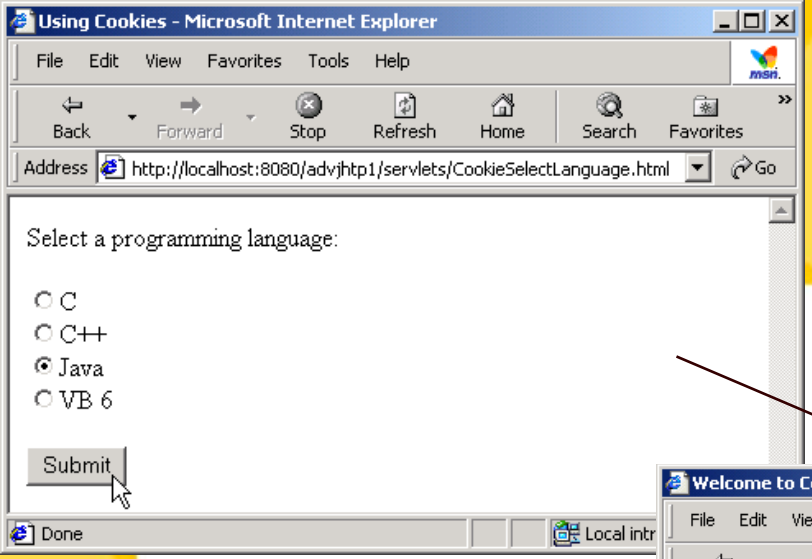
Esempio

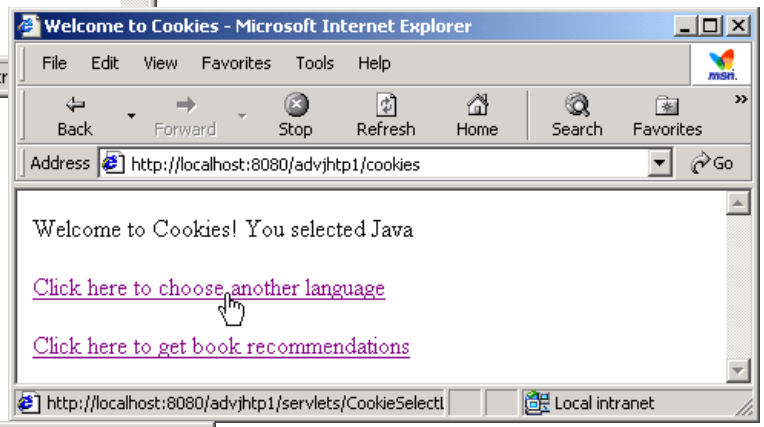
- Vogliamo scrivere una servlet che gestisca scelte che l'utente opera attraverso successive interazioni con l'applicativo.
  - L'utente sceglie un linguaggio di programmazione alla volta
  - Dopo una serie di scelte da parte dell'utente la servlet propone una serie di libri sugli argomenti selezionati.
  - N.B.: La servlet "ricorda" le scelte dell'utente nonostante l'interazione avvenga attraverso un protocollo stateless (http)

  - gestione della sessione di navigazione attraverso il mantenimento di strutture dati persistenti (dal lato del client, ovvero cookie)

L'uso del radio button obbliga l'utente ad effettuare una sola scelta alla volta!

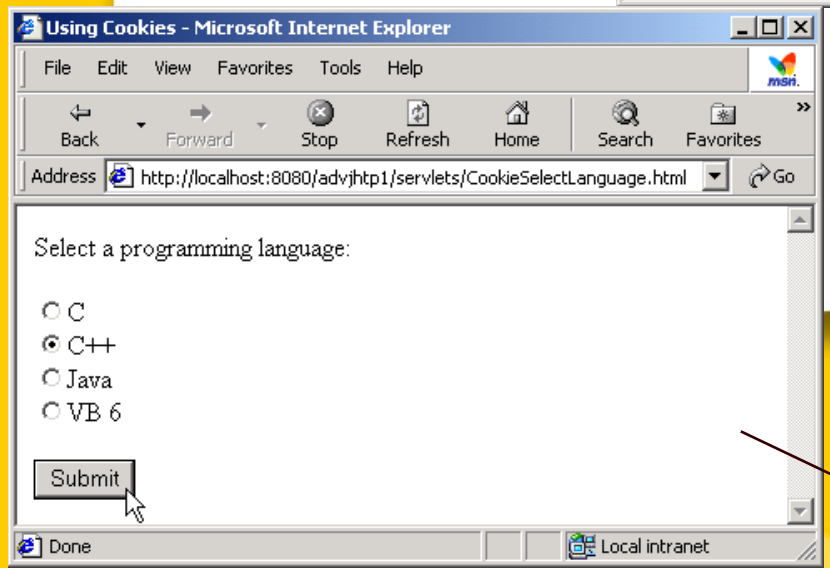
POST



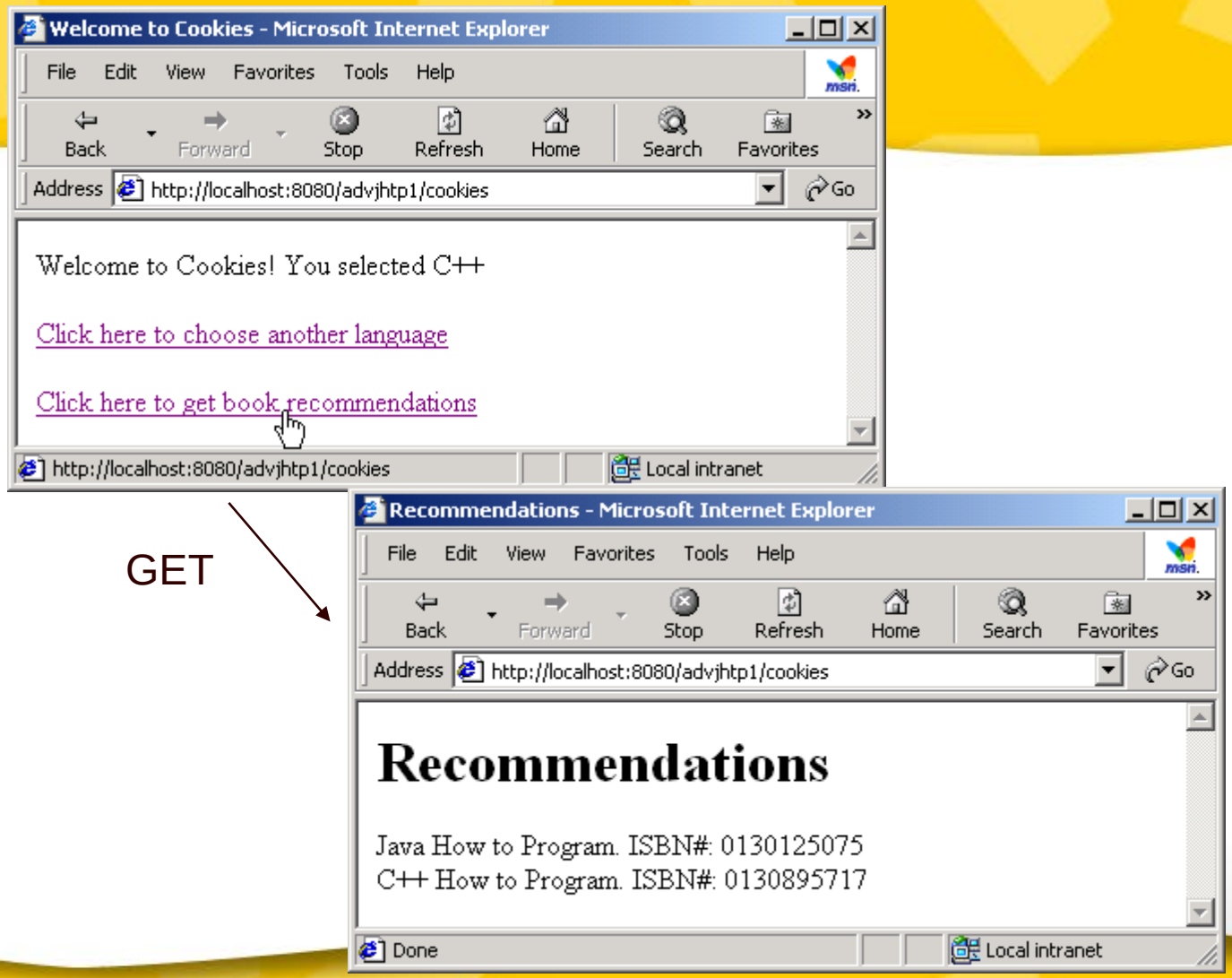


Cosa vogliamo ottenere?

POST



Cosa vogliamo ottenere?



Inviamo le selezioni con una richiesta di POST e le preleviamo con richieste di GET

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

<body>
  <form action = "/DirectoryDiSaluto/cookies" 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>
```

```
1  CookieServlet.java
2  // Using cookies to store data on the client computer.
3
4
5  import javax.servlet.*;
6  import javax.servlet.http.*;
7  import java.io.*;
8  import java.util.*;
9
10 public class CookieServlet 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 send cookie containing
23   // recommended book to the client
24   protected void doPost( HttpServletRequest request,
25     HttpServletResponse response )
26       throws ServletException, IOException
27   {
28     String language = request.getParameter( "language" );
29     String isbn = books.get( language ).toString();
30     Cookie cookie = new Cookie( language, isbn );
31
32     response.addCookie( cookie );  // must precede getWriter
33     response.setContentType( "text/html" );
34     PrintWriter out = response.getWriter();
35
```

Chiave

Valore

Il metodo **init** popola la collezione books con 4 coppie chiave/valore.

Uso il metodo **getParameter** per ottenere il valore selezionato dall'utente

Ricerco nella collezione books l'elemento corrispondente al linguaggio scelto, per ottenere l'ISBN

Creo un nuovo oggetto **Cookie**, usando le due stringhe di testo **language** e **isbn** come nome e valore rispettivamente

Invio il cookie al client attraverso l'oggetto **response** con il metodo **addCookie**

```
36      // send XHTML page to client
37
38      // start XHTML document
39      out.println( "<?xml version = \"1.0\"?>" );
40
41      out.println( "<!DOCTYPE html PUBLIC \"-//W3C//DTD " +
42        "XHTML 1.0 Strict//EN\" \"http://www.w3.org" +
43        "/TR/xhtml1/DTD/xhtml1-strict.dtd\">" );
44
45      out.println(
46        "<html xmlns = \"http://www.w3.org/1999/xhtml\">" );
47
48      // head section of document
49      out.println( "<head>" );
50      out.println( "<title>Welcome to Cookies</title>" );
51      out.println( "</head>" );
52
53      // body section of document
54      out.println( "<body>" );
55      out.println( "<p>Welcome to Cookies! You selected " +
56        language + "</p>" );
57
58      out.println( "<p><a href = " +
59        "\"/DirectoryDiSaluto/CookieSelectLanguage.html\">" +
60        "Click here to choose another language</a></p>" );
61
62      out.println( "<p><a href = \"/DirectoryDiSaluto/cookies\">" +
63        "Click here to get book recommendations</a></p>" );
64      out.println( "</body>" );
65
66      // end XHTML document
67      out.println( "</html>" );
68      out.close();    // close stream
69   }
70
```

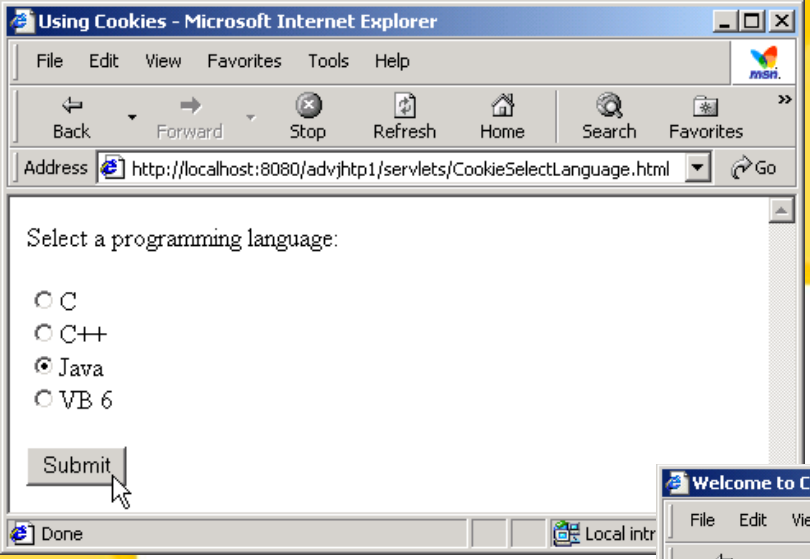
```
71  // read cookies from client and create XHTML document
72  // containing recommended books
73  protected void doGet( HttpServletRequest request,
74     HttpServletResponse response )
75        throws ServletException, IOException
76  {
77     Cookie cookies[] = request.getCookies();  // get cookies
78
79     response.setContentType( "text/html" );
80     PrintWriter out = response.getWriter();
81
82     // start XHTML document
83     out.println( "<?xml version = \"1.0\"?>" );
84
85     out.println( "<!DOCTYPE html PUBLIC \"-//W3C//DTD " +
86        "XHTML 1.0 Strict//EN\" \"http://www.w3.org" +
87        "/TR/xhtml1/DTD/xhtml1-strict.dtd\">" );
88
89     out.println(
90        "<html xmlns = \"http://www.w3.org/1999/xhtml\">" );
91
92     // head section of document
93     out.println( "<head>" );
94     out.println( "<title>Recommendations</title>" );
95     out.println( "</head>" );
96
97     // body section of document
98     out.println( "<body>" );
99
100    // if there are any cookies, recommend a book for each ISBN
101    if ( cookies != null && cookies.length != 0 ) {
102       out.println( "<h1>Recommendations</h1>" );
103       out.println( "<p>" );
104
```

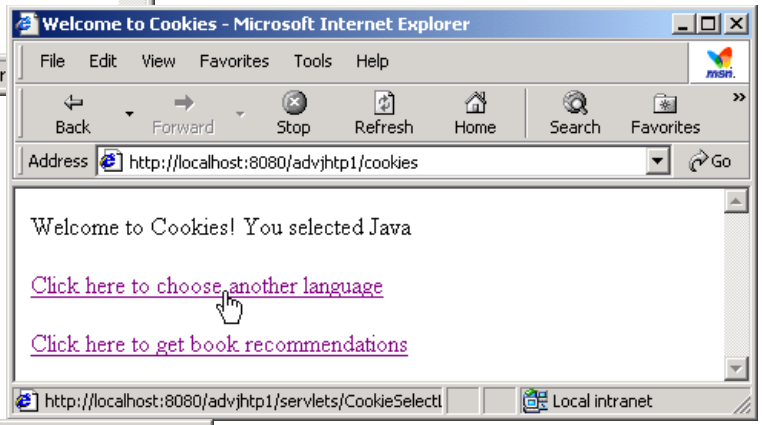
Legge i cookie presenti sul client utilizzando il metodo **getCookies** dell'oggetto **Request**, che restituisce un array di oggetti **Cookie**

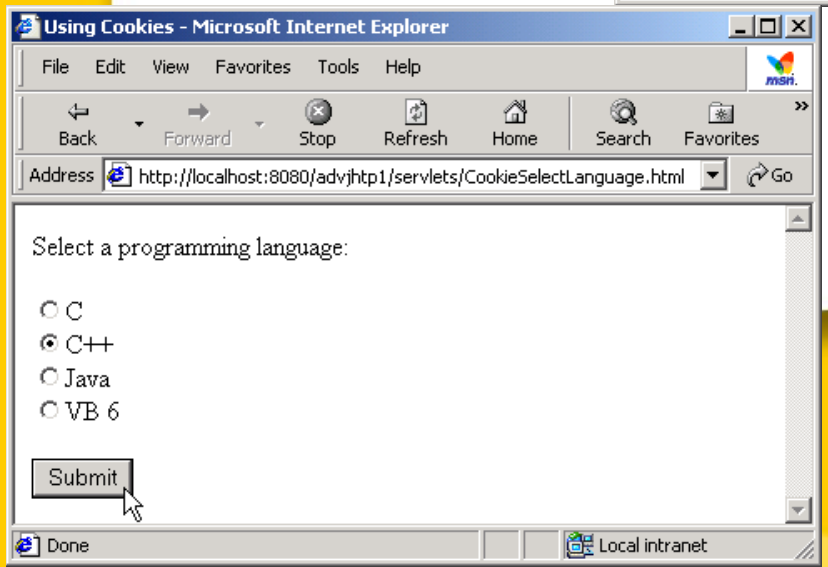
Nota:

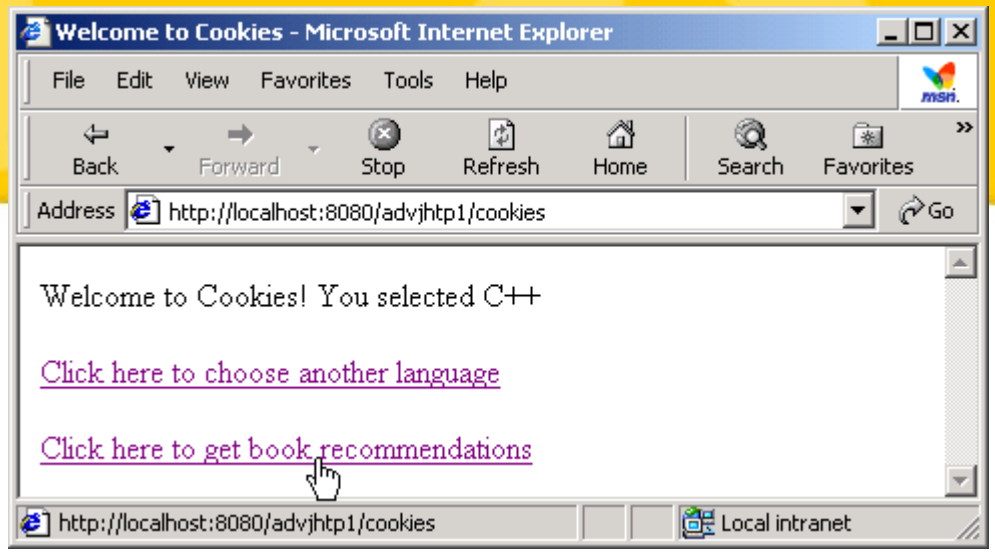
Aggiungiamo i cookie all'oggetto risposta e li preleviamo dall'oggetto richiesta

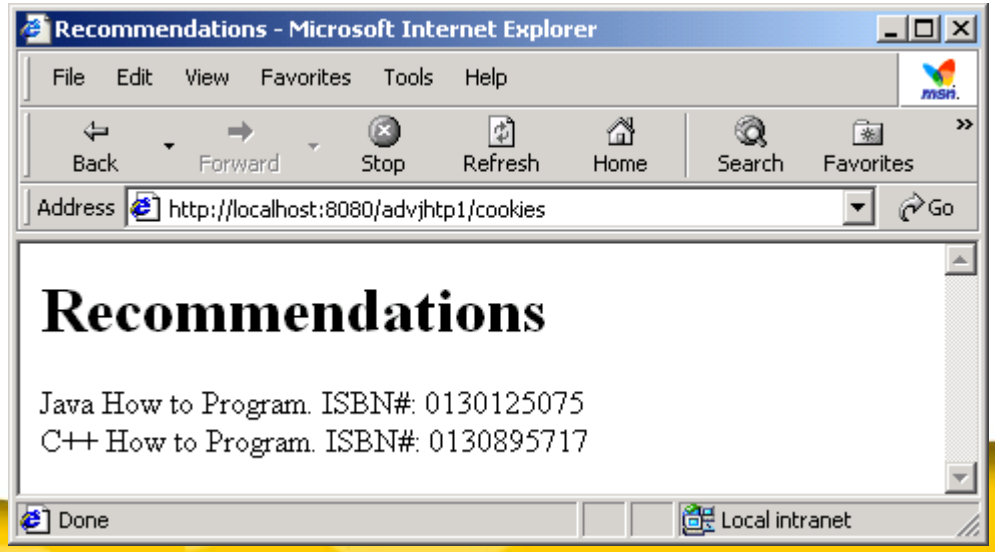
```
105         // get the name of each cookie
106         for ( int i = 0; i < cookies.length; i++ )
107            out.println( cookies[ i ].getName() +
108               " How to Program. ISBN#: " +
109               cookies[ i ].getValue() + "<br />" );
110
111         out.println( "</p>" );
112      }
113      else {   // there were no cookies
114         out.println( "<h1>No Recommendations</h1>" );
115         out.println( "<p>You did not select a language.</p>" );
116      }
117
118      out.println( "</body>" );
119
120      // end XHTML document
121      out.println( "</html>" );
122      out.close();    // close stream
123   }
124 }
```


Using Cookies - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Forward Stop Refresh Home Search Favorites
Address http://localhost:8080/advjhtp1/servlets/CookieSelectLanguage.html
Go
Select a programming language:
C
C++
Java
VB 6
Submit
Done
Local intr


Welcome to Cookies - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Forward Stop Refresh Home Search Favorites
Address http://localhost:8080/advjhtp1/cookies
Go
Welcome to Cookies! You selected Java
Click here to choose another language
Click here to get book recommendations
http://localhost:8080/advjhtp1/servlets/CookieSelectl
Local intranet


Using Cookies - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Forward Stop Refresh Home Search Favorites
Address http://localhost:8080/advjhtp1/servlets/CookieSelectLanguage.html
Go
Select a programming language:
C
C++
Java
VB 6
Submit
Done
Local intranet


Welcome to Cookies - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Forward Stop Refresh Home Search Favorites
Address http://localhost:8080/advjhtp1/cookies
Go
Welcome to Cookies! You selected C++
Click here to choose another language
Click here to get book recommendations
http://localhost:8080/advjhtp1/cookies
Local intranet


Recommendations - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Forward Stop Refresh Home Search Favorites
Address http://localhost:8080/advjhtp1/cookies
Go
Recommendations
Java How to Program. ISBN#: 0130125075
C++ How to Program. ISBN#: 0130895717
Done
Local intranet

I cookie (modifica del file web.xml)

Descriptor element	Value
servlet *element*	
servlet-name	**cookies** (ma va bene anche **“Paperino”**)
description	Using cookies to maintain state information.
servlet-class	**CookieServlet**
servlet-mapping *element*	
servlet-name	**cookies** (ma va bene anche **“Paperino”**)
url-pattern	**/cookies**

I cookie (Cont.)

Method	Description
`getComment()`	Returns a **`String`** describing the purpose of the cookie (**`null`** if no comment has been set with **`setComment`**).
`getDomain()`	Returns a **`String`** containing the cookie's domain. This determines which servers can receive the cookie. By default, cookies are sent to the server that originally sent the cookie to the client.
`getMaxAge()`	Returns an **`int`** representing the maximum age of the cookie in seconds.
`getName()`	Returns a **`String`** containing the name of the cookie as set by the constructor.
`getPath()`	Returns a **`String`** containing the URL prefix for the cookie. Cookies can be "targeted" to specific URLs that include directories on the Web server. By default, a cookie is returned to services operating in the same directory as the service that sent the cookie or a subdirectory of that directory.
`getSecure()`	Returns a **`boolean`** value indicating if the cookie should be transmitted using a secure protocol (**`true`**).
`getValue()`	Returns a **`String`** containing the value of the cookie as set with **`setValue`** or the constructor.
`getVersion()`	Returns an **`int`** containing the version of the cookie protocol used to create the cookie. A value of 0 (the default) indicates the original cookie protocol as defined by Netscape. A value of 1 indicates the current version, which is based on *Request for Comments (RFC) 2109*.

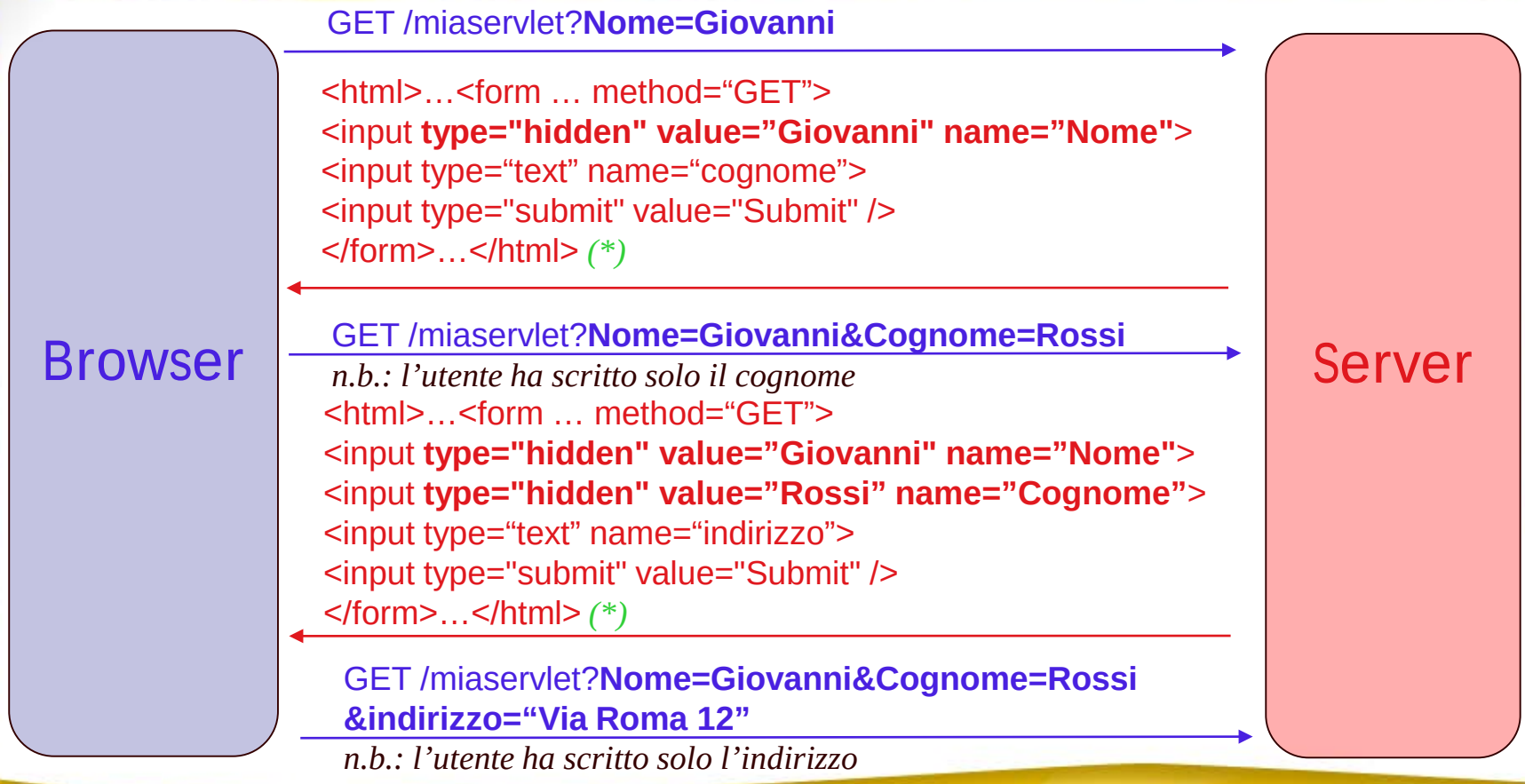
(Part 1 of 2)

I cookie (Cont.)

`setComment(String)`	The comment describing the purpose of the cookie that is presented by the browser to the user. (Some browsers allow the user to accept cookies on a per-cookie basis.)
`setDomain(String)`	This determines which servers can receive the cookie. By default, cookies are sent to the server that originally sent the cookie to the client. The domain is specified in the form **`".deitel.com"`**, indicating that all servers ending with **`.deitel.com`** can receive this cookie.
`setMaxAge(int)`	Sets the maximum age of the cookie in seconds.
`setPath(String)`	Sets the “target” URL prefix indicating the directories on the server that lead to the services that can receive this cookie.
`setSecure(boolean)`	A **`true`** value indicates that the cookie should only be sent using a secure protocol.
`setValue(String)`	Sets the value of a cookie.
`setVersion(int)`	Sets the cookie protocol for this cookie.

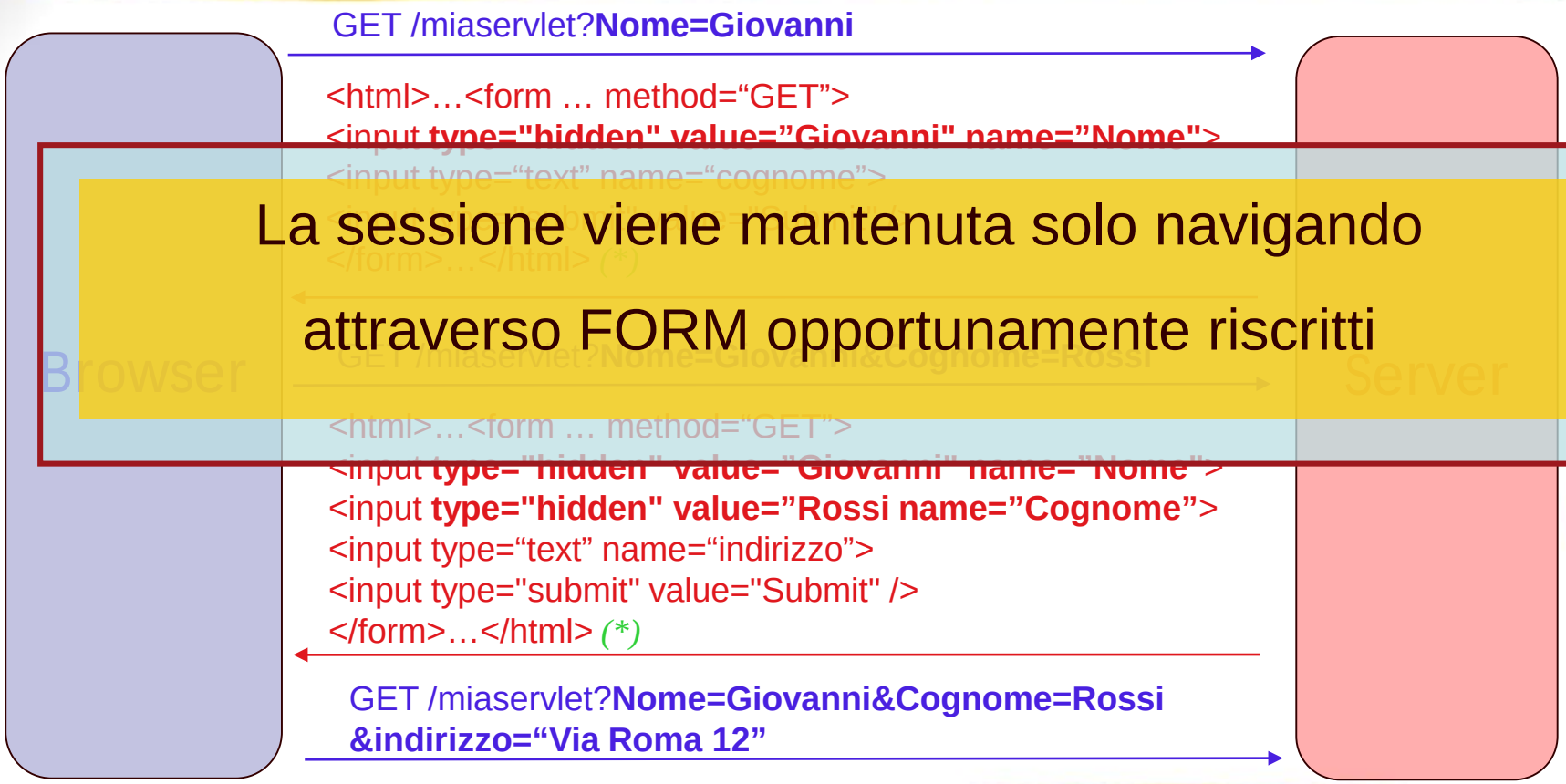
Riassunto: gestione della sessione tramite parametri hidden

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

Gestione della sessione tramite cookie