Laboratorio di Programmazione di Rete

Lezione del 22 Marzo 2010

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

Esercitazione del 16 Marzo 2010:

- Scrivere una servlet (e tutti i file necessari al corretto funzionamento dell'applicazione) che produca dinamicamente una pagina html di risposta contenente un messaggio di saluto.

```
1
2  // A simple servlet to process get requests.
3
4  import javax.servlet.*;
5  import javax.servlet.http.*;
6  import java.io.*;
7
8  public class ServletDiSaluto extends HttpServlet {
9
10    // process "get" requests from clients
11    protected void doGet( HttpServletRequest request,
12      HttpServletResponse response )
13        throws ServletException, IOException
14     {
15      response.setContentType( "text/html" );
16      PrintWriter out = response.getWriter();
17
18      // send XHTML page to client
19
20      // start XHTML document
21      out.println( "<?xml version = \"1.0\"?>" );
22
23      out.println( "<!DOCTYPE html PUBLIC \"-//W3C//DTD " +
24        "XHTML 1.0 Strict//EN\" \"http://www.w3.org" +
25        "/TR/xhtml1/DTD/xhtml1-strict.dtd\">" );
```

Importa i package javax.servlet e javax.servlet.http.

Estende HttpServlet in modo che gestisca le richieste HTTP get e post.

Sovrascrive il metodo doGet per fornire funzionalità personalizzate.

Utilizza l'oggetto response e il relativo metodo setContentType per specificare il tipo di contenuto dei dati che devono essere spediti in risposta al client.

Utilizza il metodo getWriter dell'oggetto response per ottenere un riferimento all'oggetto PrintWriter che abilita la servlet a spedire dati al client.

Crea il documento XHTML scrivendo stringhe attraverso il metodo println dell'oggetto out.

ServletDiSaluto.java

```
26
27      out.println(        "<html xmlns =    \"http://www.w3.org/1999/xhtml  \">" );
28
29      // head section of document
30      out.println(        "<head>" );
31      out.println(        "<title>A Simple Servlet Example</title>"          );
32         out.println(     "</head>" );
33
34      // body section of document
35      out.println(        "<body>" );
36      out.println(        "<h1>Welcome to Servlets!</h1>"    );
37      out.println(        "</body>" );
38
39      // end XHTML docu      ment
40      out.println(        "</html>" );
41         out.close();        // close stream to complete the page
42   }
43 }
```

Chiude l'output stream, esegue il flush del buffer di output e spedisce le informazioni al client.

ServletDiSaluto.java
(2 of 2)

File index.html

```
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  <!—File index.html -->
6
7  <html xmlns = "http://www.w3.org/1999/xhtml" >
8  <head>
9    <title>Handling an HTTP Get Request</title>
10  </head>
11
12  <body>
13    <form action = "/DirectoryDiSaluto/AliasServletDiSaluto" method = "get" >
14
15      <p><label>Click the button to invoke the servlet
16        <input type = "submit" value = "Get HTML Document" />
17      </label></p>
18
19    </form>
20  </body>
21  </html>
```

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

```
1  <!DOCTYPE web-app PUBLIC
2    "-//Sun Microsystems, Inc.//DTD Web Application 2.2//EN"
3    "http://java.sun.com/j2ee/dtds/web-app_2_2.dtd">
4
5  <web-app>
6
7    <!-- General description of your Web application
8    <display-name>
9      NomeDisplayDellaWebApplication
10       Viene visualizzato dall'admin
11   </display-name>
12
13   <description>
14     Si tratta di un'applcazione dove facciamo
15     Esempi di servlet.
16   </description>
17
18   <!-- Servlet definitions -->
19     <servlet>
20       <servlet-name>Paperino</servlet-name>
21
22       <description>
23         Una servlet che gestisce le richieste di GET
24       </description>
25
26       <servlet-class>
27         ServletDiSaluto
28       </servlet-class>
29   </servlet>
30
```

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

```
31    <!-- Servlet mappings -->
32    <servlet-mapping>
33      <servlet-name>Paperino</servlet-name>
34      <url-pattern>/AliasServletDiSaluto</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

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" >
```

Uso dei percorsi relativi in una web application (4)

WelcomeServlet Web application directory and file structure

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

Se nel file web.xml l'url pattern indicato è invece
<url-pattern>/A/B/topolino</url-pattern>

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

```
<form action = "A/B/topolino" method = "get" >
```

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

```
<form action = "../A/B/topolino " method = "get" >
```

Gestione di una richiesta HTTP `get` Contenente Dati

- Servlet `WelcomeServlet2`
  - Rispondere ad una richiesta di tipo `get` contenente dati

```
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
6
7  <html xmlns = "http://www.w3.org/1999/xhtml">
8  <head>
9    <title>Processing get requests with data</title>
10 </head>
11
12 <body>
13   <form action = "/DirectoryDiSaluto/welcome2" method = "get">
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     </p></label>
20
21   </form>
22 </body>
23 </html>
```

Get the first name from the user.

```
2  // Processing HTTP get requests containing data.
3
4
5  import javax.servlet.*;
6  import javax.servlet.http.*;
7  import java.io.*;
8
9  public class WelcomeServlet2 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 firstName = request.getParameter( "firstname" );
17
18      response.setContentType( "text/html" );
19      PrintWriter out = response.getWriter();
20
21      // send XHTML document 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
```

Il metodo **getParameter** dell'oggetto **request** riceve come argomento il nome del parametro e restituisce il corrispondente vaolre sotto forma di una stringa (tipo **String**)

```
33      // head section of document
34      out.println( "<head>" );
35      out.println(
36        "<title>Processing get 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  }
```

La stringa ottenuta alla linea 16 viene usata per costruire la risposta al client.

Modifica del file web.xml per includere la nuova servlet

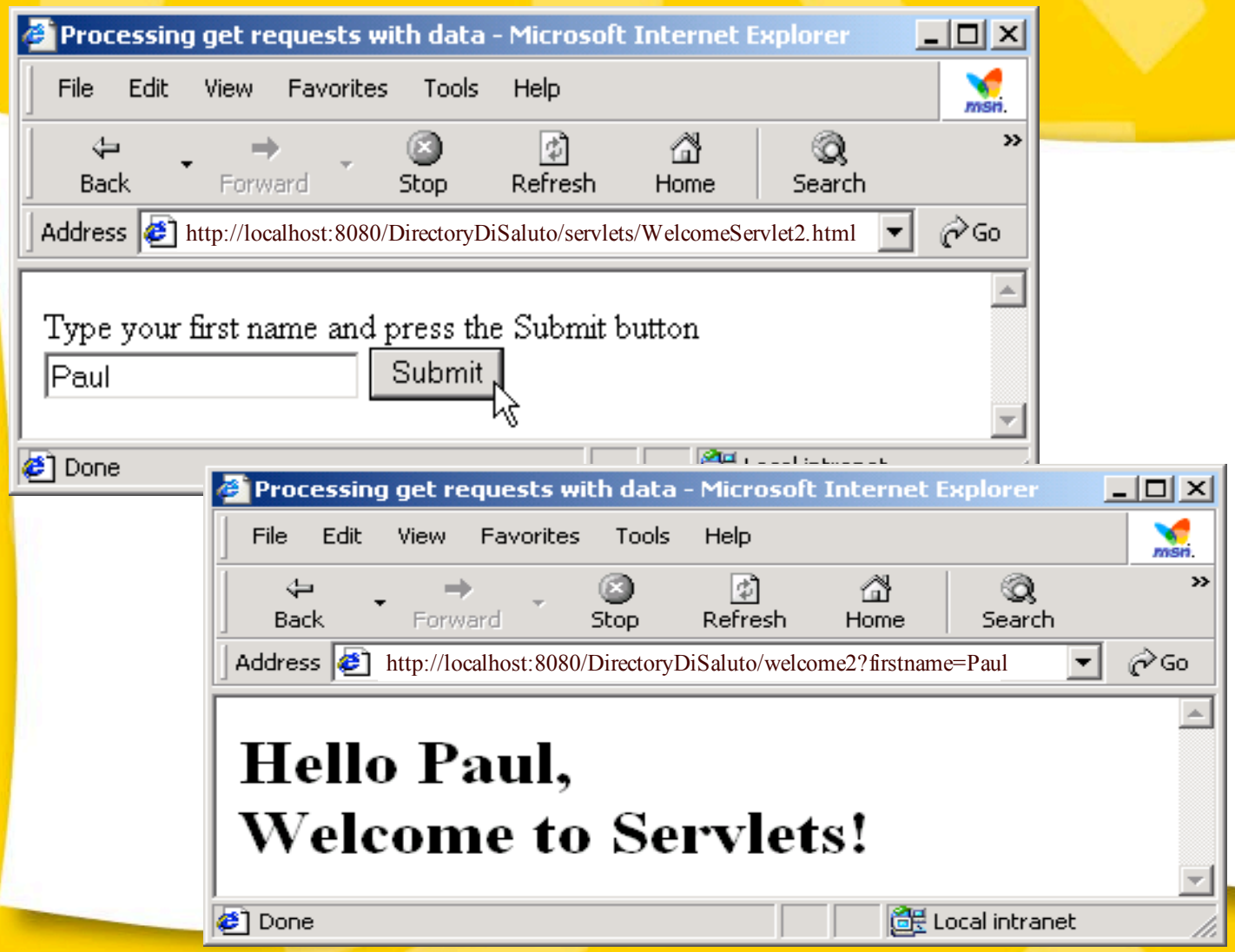
Descriptor element	Value
servlet *element*	
servlet-name	**Welcome2**
description	**Handling HTTP get requests with data.**
servlet-class	**WelcomeServlet2**
servlet-mapping *element*	
servlet-name	**Welcome2**
url-pattern	**/Welcome2**

Modifica del file web.xml per includere la nuova servlet

- Si aggiungono le seguenti direttive:

```
<servlet>
  <servlet-name>Welcome2</servlet-name>
  <display-name>
    NomeDisplayDiSalutoPersonalizzato</display-name>
  <description>Facciamo un test con personalizzazione</description>
  <servlet-class>WelcomeServlet2</servlet-class>
 </servlet>

<servlet-mapping>
  <servlet-name>Welcome2</servlet-name>
  <url-pattern>/Welcome2</url-pattern>
 </servlet-mapping>
```


Processing get requests with data - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Forward Stop Refresh Home Search
Address http://localhost:8080/DirectoryDiSaluto/servlets/WelcomeServlet2.html
Go
Type your first name and press the Submit button
Paul
Submit
Done
Processing get requests with data - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Forward Stop Refresh Home Search
Address http://localhost:8080/DirectoryDiSaluto/welcome2?firstname=Paul
Go
Hello Paul,
Welcome to Servlets!
Done
Local intranet

Servlet che gestisce richieste di POST

- HTTP post request
  - Spedire dati da un form HTML ad un handler di form localizzato sul server
  - Ricordare che i browser non fanno caching delle risposte a richieste di tipo POST
- Servlet WelcomeServlet3
  - Risponde ad una richiesta di tipo POST contenente dati

```
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 }
```

Notare che il parametro firstname non viene passato nell'url come nel caso dell'esempio di richiesta di GET

Modifiche al file web.xml

Descriptor element	Value
servlet *element*	
servlet-name	**welcome3**
description	**Handling HTTP post requests with data.**
servlet-class	**WelcomeServlet3**
servlet-mapping *element*	
servlet-name	**welcome3**
url-pattern	**/welcome3**

Non dimenticate di ricaricare l'applicazione dopo avere ultimato e salvato le modifiche al file web.xml

Osservazione

Una stessa servlet può trattare in modo completamente diverso richieste di tipo GET e POST (basta scrivere in modi diverso i metodi doGet(…) e doPost(…).

Errori tipici

- Il compilatore java non trova il package javax.servlet.*
  - Avete scritto correttamente la variabile d'ambiente CLASSPATH?
- La classe viene compilata correttamente ma il tomcat manager non visualizza la vostra applicazione nella lista delle applicazioni attive
  - Avete rispettato la struttura delle cartelle di una web application? Il file web.xml è nella cartella WEB-INF?
  - Avete controllato la sintassi del file web.xml?
  - Avete messo la vostra classe compilata nella cartella WEB-INF/classes?
- L'applicazione è nell'elenco ma quando scrivete l'url pattern per invocare la servlet le cose non funzionano:
  - Sintassi del file web.xml: l'url pattern comincia con il carattere "/"?
  - L'associazione <servlet-class> all' <url-pattern> è descritta correttamente?
  - State invocando il server sulla porta 8080?
  - State usando il percorso corretto?

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 welcome2 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.

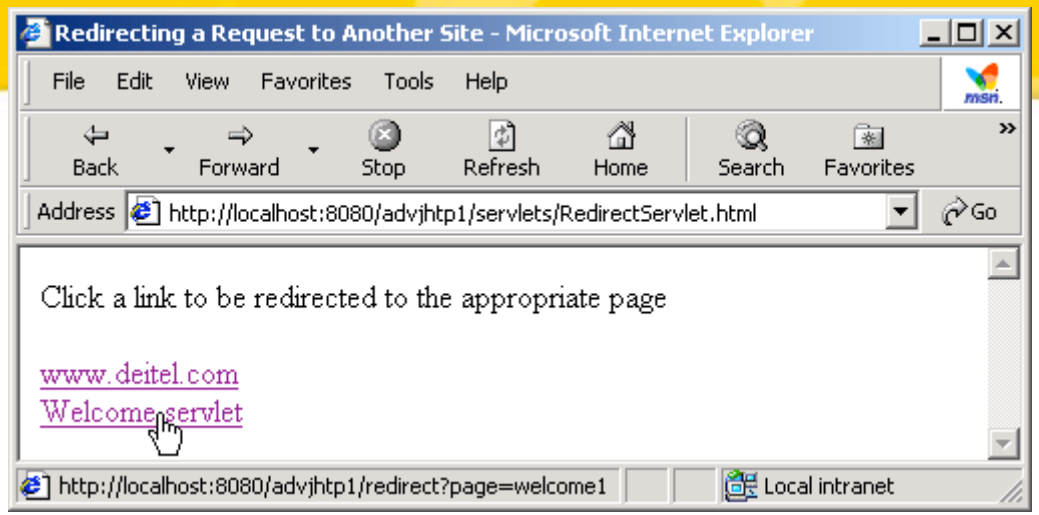
Si inoltra la richiesta al servlet **WelcomeServlet2**.

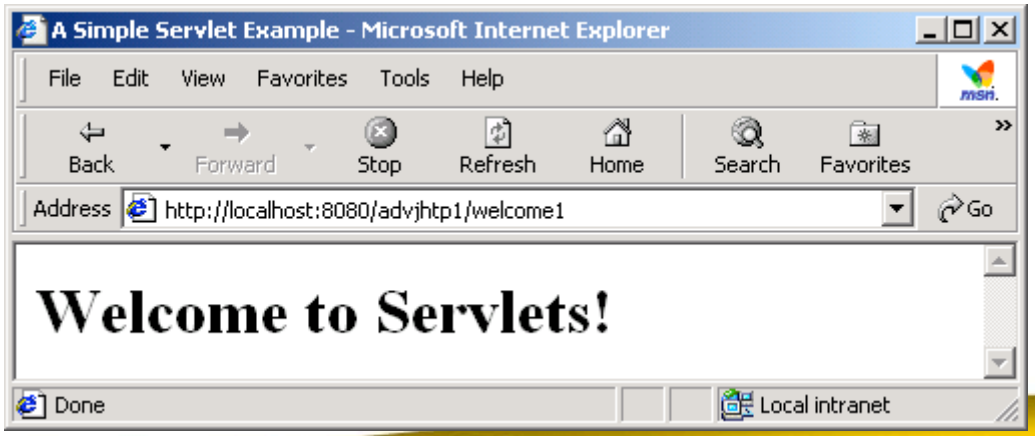
Pagina web di output che viene visualizzata nel caso in cui si sia ricevuta una richieta non valida (non viene invocato il metodo **sendRedirect**).

```
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
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