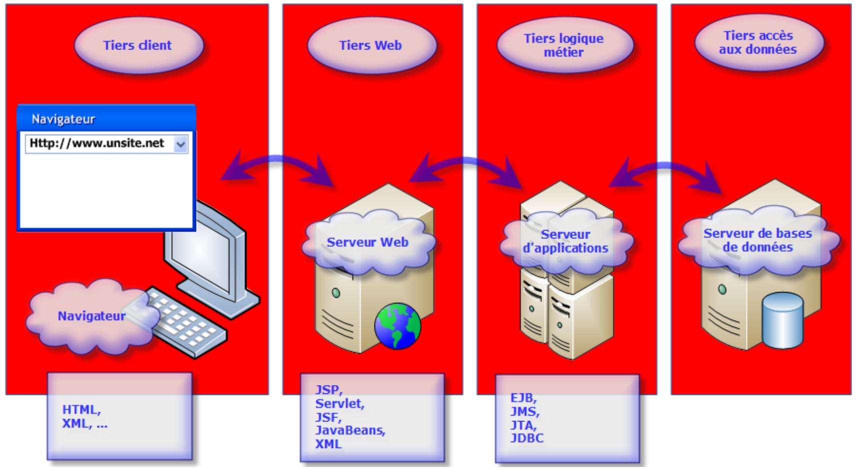
Servlets

Outline

1 Introduction



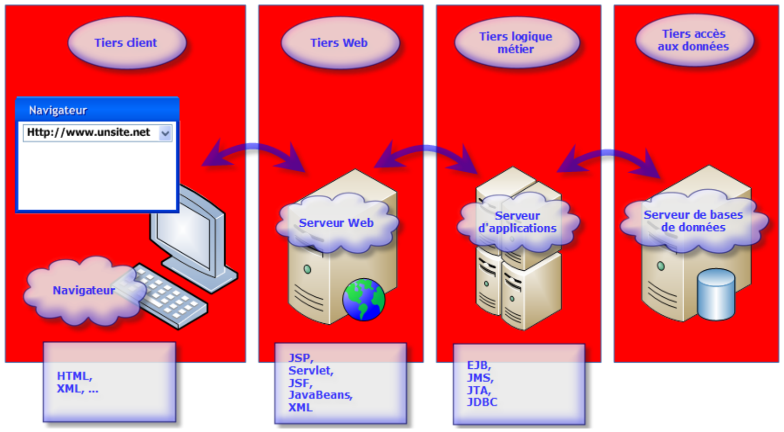
- Java networking capabilities
  - Socket-based and packet-based communications
    - Package **`java.net`**
  - Remote Method Invocation (RMI)
    - Package **`java.rmi`**
  - Servlets and Java Server Pages (JSP)
    - Request-response model
    - Packages **`javax.servlet`**
      `javax.servlet.http`
      `javax.servlet.jsp`
      `javax.servlet.tagext`
    - Form the Web tier of J2EE

Qu’est ce qu’une servlet ?

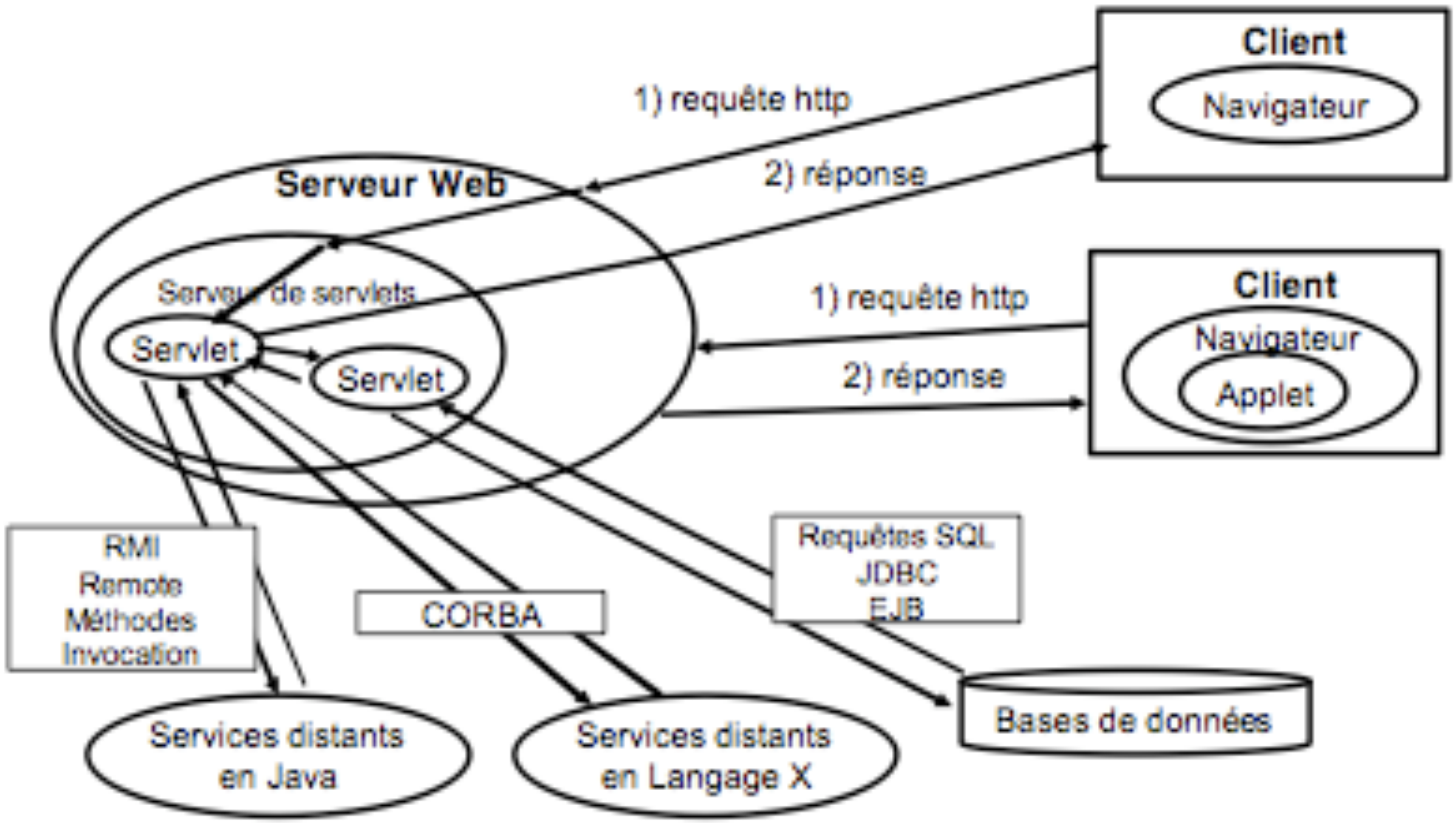
- Servlets
  - Thin clients
  - Request/response mechanism
  - redirection
- Les servlets sont la base de la programmation Web Java EE
- Une servlet est un programme Java coté serveur
- L’appellation d’une servlet passe par un URL liée à la servlet

2 Servlet Overview and Architecture

- Web servers (HTTP Server)
  - Récupérer requêtes HTTP, Redirection
  - Microsoft (IIS), Apache
- Servlet container (web container)
  - Server that executes a servlet
  - Tomcat (Jakarta project): official reference implementation of the JSP and servlet standards
- Application servers (EJB container)
  - Oracle Glassfish
  - RedHat's JBOSS
  - BEA's WebLogic Application Server
  - IBM's WebSphere Application Server
  - Many of them include Tomcat & Apache



Servlets


Client
Navigateur
1) requête http
2) réponse
Serveur Web
Serveur de servlets
Servlet
Servlet
1) requête http
2) réponse
Client
Navigateur
Applet
RMI
Remote
Méthodes
Invocation
CORBA
Requêtes SQL
JDBC
EJB
Services distants
en Java
Services distants
en Langage X
Bases de données

2.1 Interface `Servlet`

- Interface **`Servlet`**
  - All servlets must implement this interface
  - All methods of interface **`Servlet`** are invoked by servlet container
- Servlet implementation
  - **`GenericServlet`**
  - **`HttpServlet`**

2.1 Interface Servlet

Method	Description
`void init( ServletConfig config )`	
	The servlet container calls this method once during a servlet's execution 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 servlet's configuration information such as servlet initialization parameters and the servlet's `ServletContext`, 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 servlet's author 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 "cleanup" method 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.

Fig. 24.1 Methods of interface `Servlet` (package `javax.servlet`).

2.2 HttpServlet Class

- Overrides method **`service`**
- Two most common HTTP request types
  - **`get`** requests
  - **`post`** requests
- Method **`doGet`** responds to **`get`** requests
- Method **`doPost`** responds to **`post`** requests
- **`HttpServletRequest`** and **`HttpServletResponse`** objects

2.2 `HttpServlet` Class (Cont.)

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).

Fig. 24.2 Other methods of class HttpServlet.

2.3 `HttpServletRequest` Interface

- Web server
  - creates an **`HttpServletRequest`** object
  - passes it to the servlet's **`service`** method
- **`HttpServletRequest`** object contains the request from the client

2.3 `HttpServletRequest` Interface (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 client's 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 `Cookie`s for uniquely identifying clients.

Fig. 24.3 Some methods of interface `HttpServletRequest`.

2.4 `HttpServletResponse` Interface

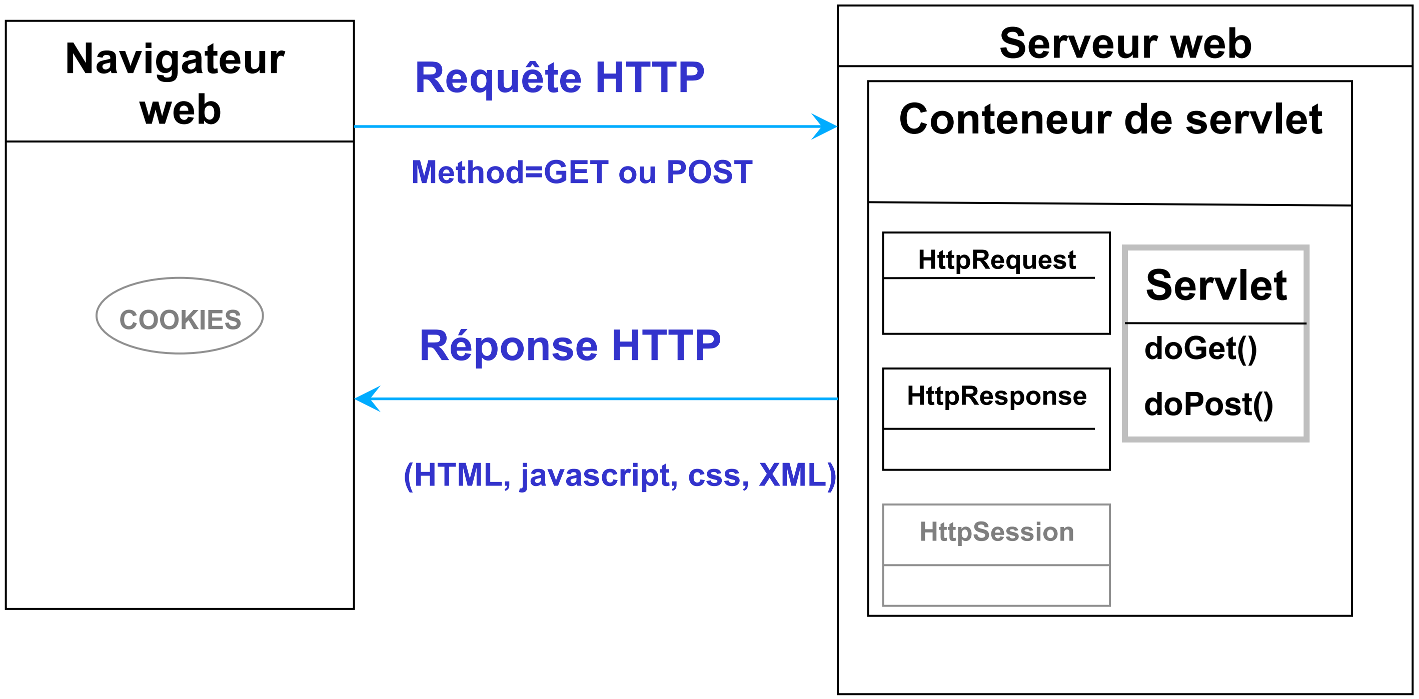
- Web server
  - creates an **`HttpServletResponse`** object
  - passes it to the servlet’s **`service`** method

2.4 `HttpServletResponse` Interface (Cont.)

Method	Description
`void addCookie( Cookie cookie )`	
	Used to add a `Cookie` to the header of the response to the client. The `Cookie`'s maximum age and whether `Cookie`s are enabled on the client determine if `Cookie`s 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.

Fig. 24.4 Some methods of interface `HttpServletResponse`.

HttpServlets



3 Handling HTTP `get` Requests

- **`get`** request
  - Retrieve the content of a URL
- Example: **`WelcomeServlet`**
  - a servlet handles HTTP **`get`** requests

```
1   // Fig. 5: WelcomeServlet.java
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 WelcomeServlet 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\">" );
26
```

Import the `javax.servlet` and `javax.servlet.http` packages.

Extends `HttpServlet` to handle HTTP `get` requests

Override method `doGet` to provide custom get request processing.

Uses the `response` object's

Uses the `response` object's `getWriter` method to obtain a reference to the `PrintWriter` object that enables the servlet to send content to the client.

Create the XHTML document by writing strings with the `out` object's `println` method.

WelcomeServlet

Lines 4-5

-42

```
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 document
40          out.println( "</html>" );
41          out.close();  // close stream to complete the page
42       }
43    }
```

`WelcomeServlet`

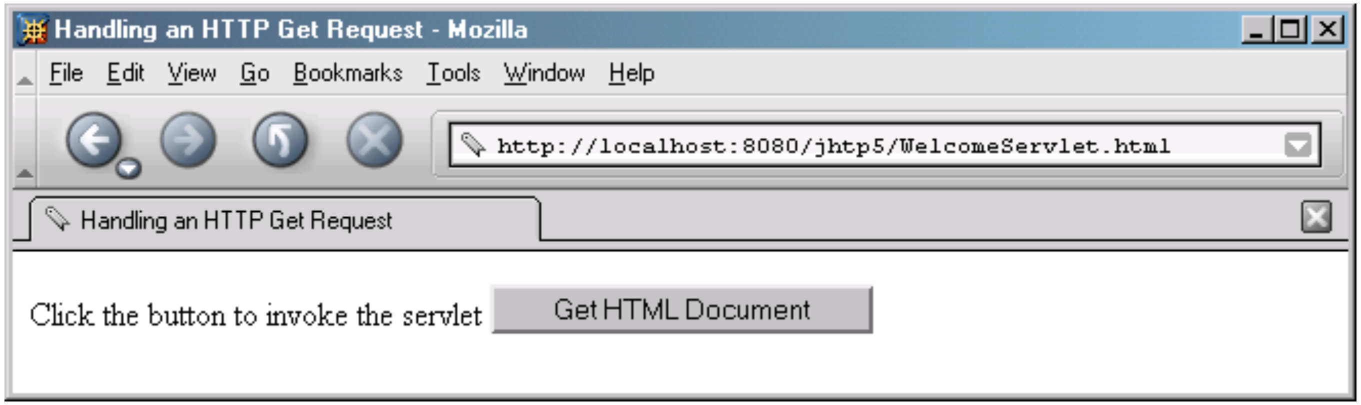
`Line 41`

Closes the output stream, flushes the output buffer and sends the information to the client.

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

<!-- Fig. 6: WelcomeServlet.html -->

<html xmlns = "http://www.w3.org/1999/xhtml">
<head>
   <title>Handling an HTTP Get Request</title>
</head>

<body>
   <form action = "/jhtp5/welcome1" method = "get">

      <p><label>Click the button to invoke the servlet
         <input type = "submit" value = "Get HTML Document" />
      </label></p>

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

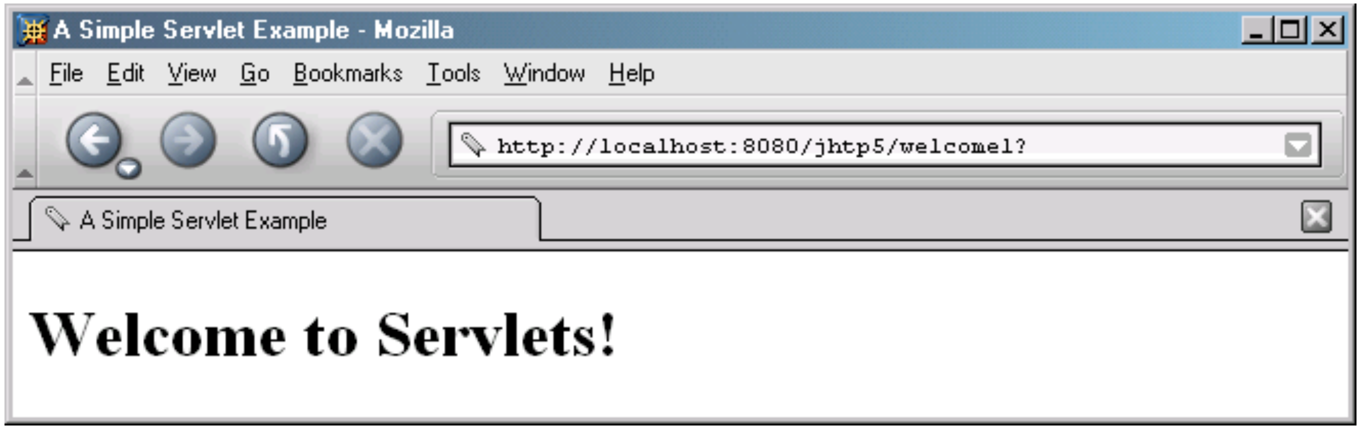
WelcomeServlet.html

Program output


Handling an HTTP Get Request - Mozilla
File Edit View Go Bookmarks Tools Window Help
Handling an HTTP Get Request

```
http://localhost:8080/jhtp5/WelcomeServlet.html
```

Click the button to invoke the servlet [Get HTML Document]

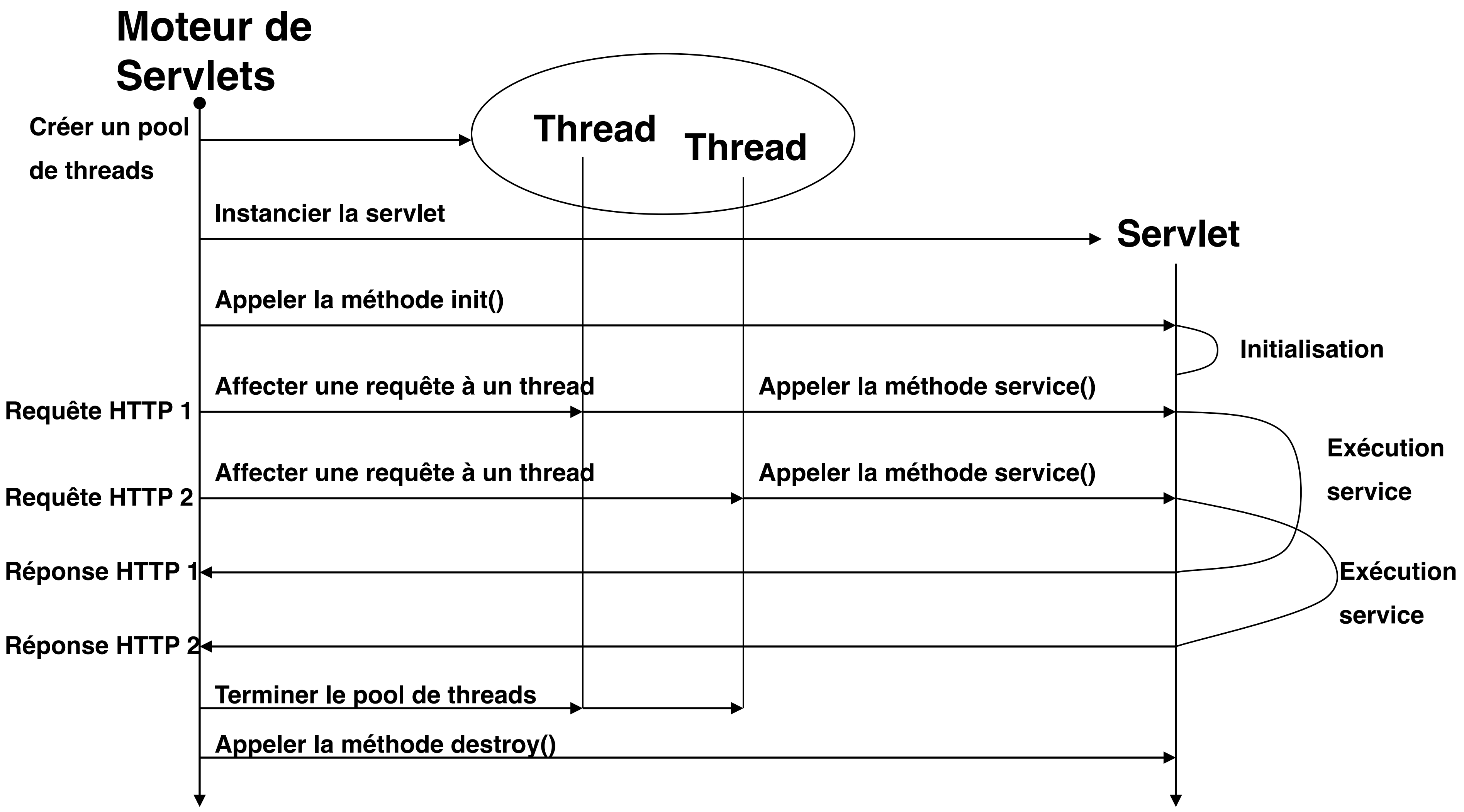

A Simple Servlet Example - Mozilla
File Edit View Go Bookmarks Tools Window Help
A Simple Servlet Example

```
http://localhost:8080/jhtp5/welcome1?
```

Welcome to Servlets!

DEMO

Gestion des servlets


Moteur de Servlets
Créer un pool de threads
Thread
Thread
Instancier la servlet
Servlet
Appeler la méthode init()
Initialisation
Affecter une requête à un thread
Appeler la méthode service()
Requête HTTP 1
Exécution service
Affecter une requête à un thread
Appeler la méthode service()
Requête HTTP 2
Réponse HTTP 1
Exécution service
Réponse HTTP 2
Terminer le pool de threads
Appeler la méthode destroy()

Fonctionnalités offertes par les servlets

- Génère une page WEB HTML dynamique
- Cette page WEB HTML dynamique peut être générée en fonction
  - des paramètres de la requête
  - de la nature de la requête
  - du résultat d’une requête à une base de données
  - de la connaissance de données sur le client
- Peut gérer concurremment la connexion avec plusieurs clients en partageant des données communes
- Contrôle les sessions avec un client particulier en sauvegardant ses données
- Reconnaît le contexte d'un client et peut accéder aux cookies
- Accède au Bases de données
- Traite et/ou stocke des données recueillies via un formulaire HTML

4 Handling HTTP `get` Requests Containing Data

- Servlet **`WelcomeServlet2`**
  - Responds to a **`get`** request that contains data

```
1   // Fig. 11: WelcomeServlet2.java
2   // Processing HTTP get requests containing data.
3
4   import javax.servlet.*;
5   import javax.servlet.http.*;
6   import java.io.*;
7
8   public class WelcomeServlet2 extends HttpServlet {
9
10     // process "get" request from client
11     protected void doGet( HttpServletRequest request,
12        HttpServletResponse response )
13           throws ServletException, IOException
14     {
15        String firstName = request.getParameter( "firstname" );
16
17        response.setContentType( "text/html" );
18        PrintWriter out = response.getWriter();
19
20        // send XHTML document to client
21
22        // start XHTML document
23        out.println( "<?xml version = \"1.0\"?>" );
24
```

WelcomeServlet2 responds to a get request that contains data.

Line 15

The `request` object's `getParameter` method receives the parameter name and returns the corresponding `String` value.

```
25          out.println( "<!DOCTYPE html PUBLIC \"-//W3C//DTD " +
26             "XHTML 1.0 Strict//EN\" \"http://www.w3.org" +
27             "/TR/xhtml1/DTD/xhtml1-strict.dtd\">" );
28
29          out.println( "<html xmlns = \"http://www.w3.org/1999/xhtml\">" );
30
31          // head section of document
32          out.println( "<head>" );
33          out.println(
34             "<title>Processing get requests with data</title>" );
35          out.println( "</head>" );
36
37          // body section of document
38          out.println( "<body>" );
39          out.println( "<h1>Hello " + firstName + ",<br />" );
40          out.println( "Welcome to Servlets!</h1>" );
41          out.println( "</body>" );
42
43          // end XHTML document
44          out.println( "</html>" );
45          out.close();  // close stream to complete the page
46       }
47    }
```

WelcomeServlet2 responds to a get request that contains data.

Line 39

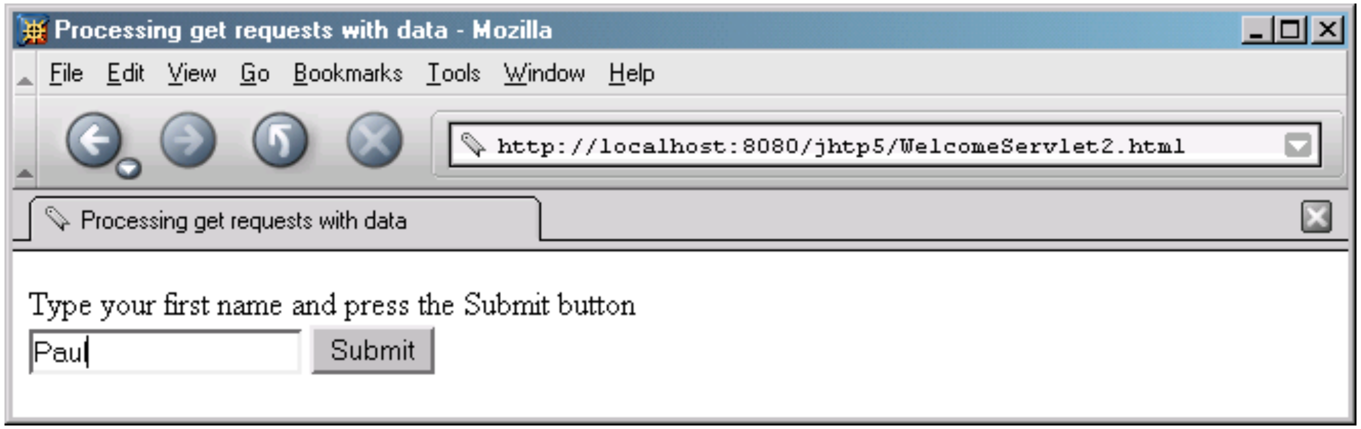
Uses the result of line 16 as part of the response to the client.

```
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   <!-- Fig. 12: WelcomeServlet2.html -->
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 = "/jhtp5/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.

HTML document in which the form's action invokes WelcomeServlet2 through the alias welcome2 specified in web.xml.

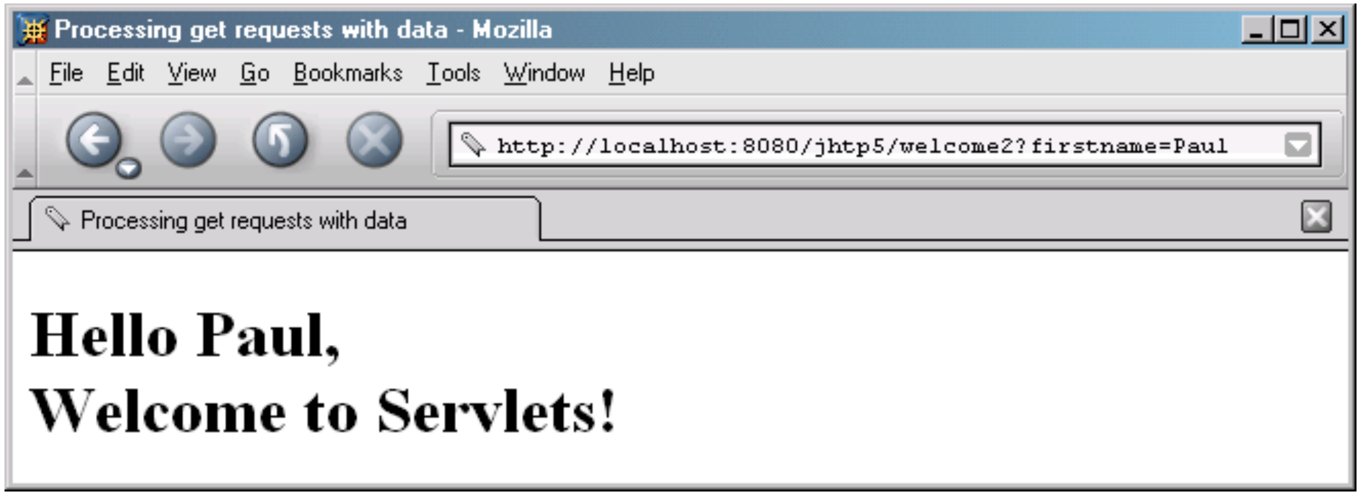
Line 17


Processing get requests with data - Mozilla
File Edit View Go Bookmarks Tools Window Help
Processing get requests with data
Submit

```
http://localhost:8080/jhtp5/WelcomeServlet2.html
```

Type your first name and press the Submit button

Paul


Processing get requests with data - Mozilla
File Edit View Go Bookmarks Tools Window Help
Processing get requests with data

```
http://localhost:8080/jhtp5/welcome2?firstname=Paul
```

```
Hello Paul,
Welcome to Servlets!
```

HTML document in which the form's action invokes WelcomeServlet2 through the alias welcome2 specified in web.xml.

Program output

4 Handling HTTP `get` Requests Containing Data (Cont.)

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`

Fig. 24.13 Deployment descriptor information for servlet `WelcomeServlet2`.

DEMO

5 Handling HTTP post Requests

- HTTP post request
  - Post data from an HTML form to a server-side form handler
  - Browsers cache Web pages
- Servlet WelcomeServlet3
  - Responds to a **`post`** request that contains data

Exercice

- Reprenez l’exemple précédent pour répondre à une requête à l’aide de la méthode « Post ».

```java
// Fig. 14: WelcomeServlet3.java
// Processing post requests containing data.

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

public class WelcomeServlet3 extends HttpServlet {

   // process "post" request from client
   protected void doPost( HttpServletRequest request,
      HttpServletResponse response )
         throws ServletException, IOException
   {
      String firstName = request.getParameter( "firstname" );

      response.setContentType( "text/html" );
      PrintWriter out = response.getWriter();

      // send XHTML page to client

      // start XHTML document
      out.println( "<?xml version = \"1.0\"?>" );

```

WelcomeServlet3 responds to a post request that contains data.

Declare a `doPost` method to responds to post requests.

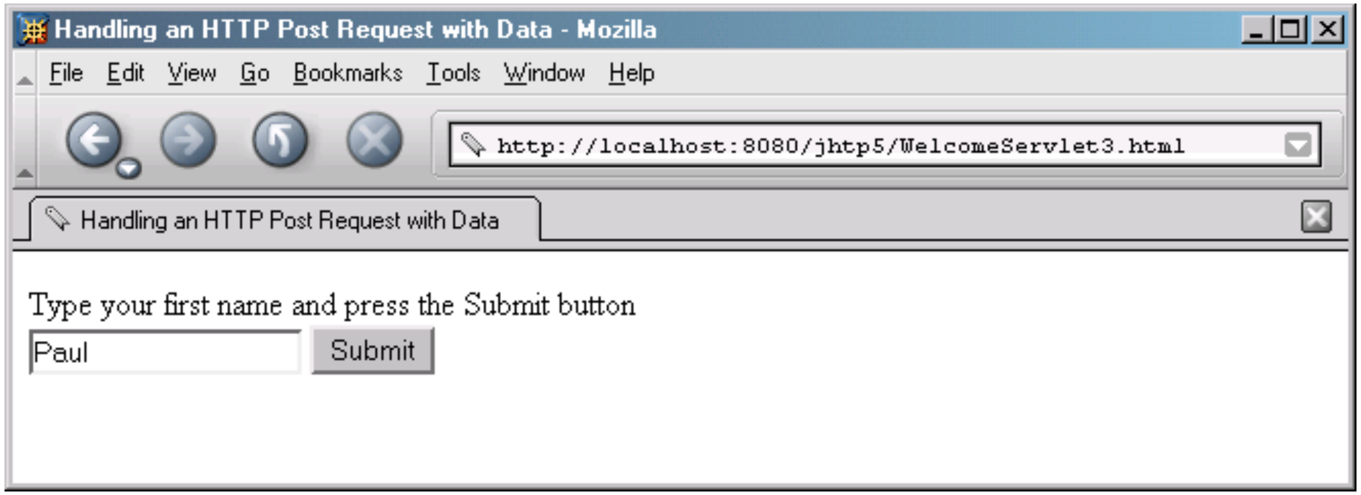
```
         out.println( "<!DOCTYPE html PUBLIC \"-//W3C//DTD " +
            "XHTML 1.0 Strict//EN\" \"http://www.w3.org" +
            "/TR/xhtml1/DTD/xhtml1-strict.dtd\">" );

         out.println( "<html xmlns = \"http://www.w3.org/1999/xhtml\">" );

         // head section of document
         out.println( "<head>" );
         out.println(
            "<title>Processing post requests with data</title>" );
         out.println( "</head>" );

         // body section of document
         out.println( "<body>" );
         out.println( "<h1>Hello " + firstName + ",<br />" );
         out.println( "Welcome to Servlets!</h1>" );
         out.println( "</body>" );

         // end XHTML document
         out.println( "</html>" );
         out.close();  // close stream to complete the page
      }
   }
```

WelcomeServlet3.java

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

<!-- Fig. 15: WelcomeServlet3.html -->

<html xmlns = "http://www.w3.org/1999/xhtml">
<head>
   <title>Handling an HTTP Post Request with Data</title>
</head>

<body>
   <form action = "/jhtp5/welcome3" method = "post">

      <p><label>
         Type your first name and press the Submit button
         <br /><input type = "text" name = "firstname" />
         <input type = "submit" value = "Submit" />
      </label></p>

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

HTML document in which the form's action invokes WelcomeServlet3 through the alias welcome3 specified in

Provide a `form` in which the user can input a name in the `text` input element `firstname`, then click the `Submit` button to invoke `WelcomeServlet3`.

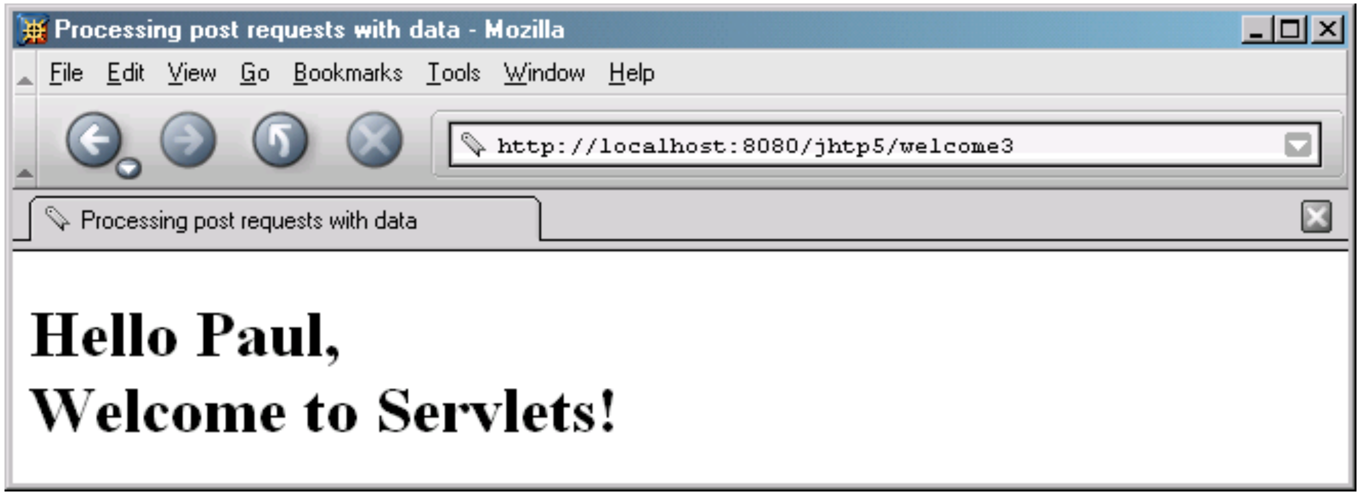

Handling an HTTP Post Request with Data - Mozilla
File Edit View Go Bookmarks Tools Window Help
http://localhost:8080/jhtp5/WelcomeServlet3.html
Handling an HTTP Post Request with Data

Type your first name and press the Submit button

Paul

Submit

HTML document in which the form's action invokes WelcomeServlet3 through the alias welcome3 specified in web.xml.


Processing post requests with data - Mozilla
File Edit View Go Bookmarks Tools Window Help
http://localhost:8080/jhtp5/welcome3
Processing post requests with data

```
Hello Paul,
Welcome to Servlets!
```

Program output

5 Handling HTTP `post` Requests (Cont.)

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`

Fig. 24.16 Deployment descriptor information for servlet `WelcomeServlet3`.

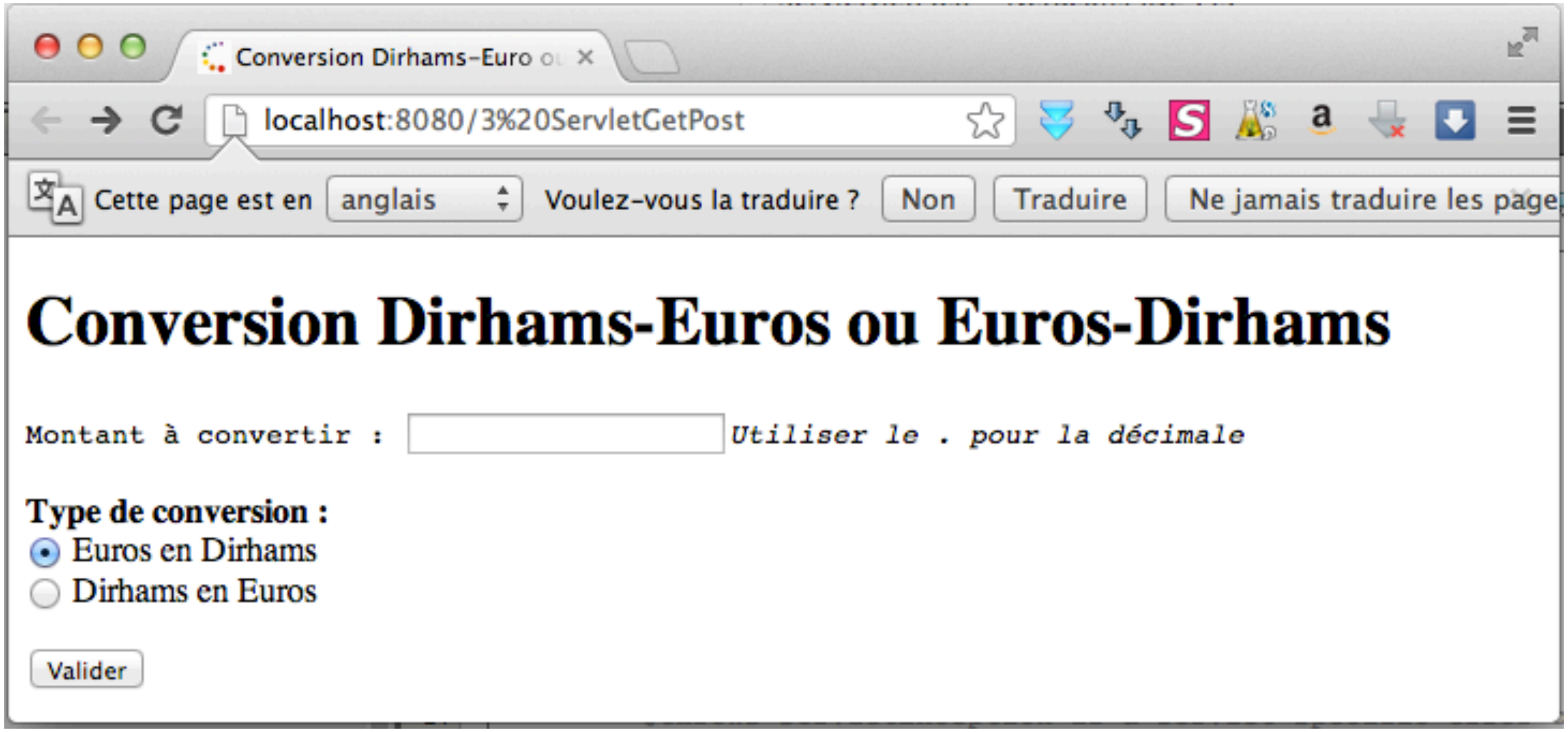
3ème exemple

```
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;
public class Conv extends HttpServlet {

          protected void doGet(HttpServletRequest req, HttpServletResponse res)
                    throws ServletException, IOException {

 response.setContentType("text/html;charset=UTF-8");
          PrintWriter out = response.getWriter();
        try {
        out.println("<!DOCTYPE html>");
          out.println("<html>");
          out.println("<head><title>Conversion Franc-Euro ou Euro-Franc</title></head>");
          out.println("<FORM METHOD=POST ACTION=http://localhost:8080/ServletGetPost/>");
          out.println("<h1><B>Conversion Francs-Euros ou Euros-Francs</B></h1>");
          out.print("<pre>Montant à convertir : ");
          out.print("<input type=text name=montant size=25>");
          out.println("<I>Utiliser le . pour la décimale</I></pre>");
          out.println("<P><B>Type de conversion :</B><BR>");
          out.println("<input type=radio name=choix value=\"EF\" checked> Euros en
Francs <BR>");
          out.println("<input type=radio name=choix value=\"FE\">Francs en Euros <BR></P>");
          out.println("<P><input type=submit value=\"Valider\"></P>");
          out.println("</FORM>");
          out.println("</html>");
         } finally {
            out.close();
        }
}
```

3ème exemple (suite)

```
protected void doPost(HttpServletRequest req, HttpServletResponse res)
                    throws ServletException, IOException {

          double montantOrigine;
          double montantConverti;

          montantOrigine=Double.parseDouble(request.getParameter("montant"));
          if (request.getParameter("choix").equals("FE")) {
                    montantConverti=montantOrigine / 6.55957;
          }
          else montantConverti = montantOrigine * 6.55957;

        response.setContentType("text/html;charset=UTF-8");
          PrintWriter out = response.getWriter();
        try {
        out.println("<!DOCTYPE html>");
          out.println("<html>");
          out.println("<head><title>Résultat de la conversion</title></head>");
          out.print("<P><B> Le Montant d'origine est : </B>");
          out.print(montantOrigine);
          out.println("</P>");
          out.print("<P><B> Montant converti  : </B>");
          out.print(montantConverti);
          out.println("</P>");
          out.println("</html>");
          out.close();
         } finally {
            out.close();
        }
}
```

Exécuter la servlet


Conversion Dirhams-Euro ou
localhost:8080/3%20ServletGetPost
Cette page est en anglais
Voulez-vous la traduire ?
Non
Traduire
Ne jamais traduire les page

Conversion Dirhams-Euros ou Euros-Dirhams

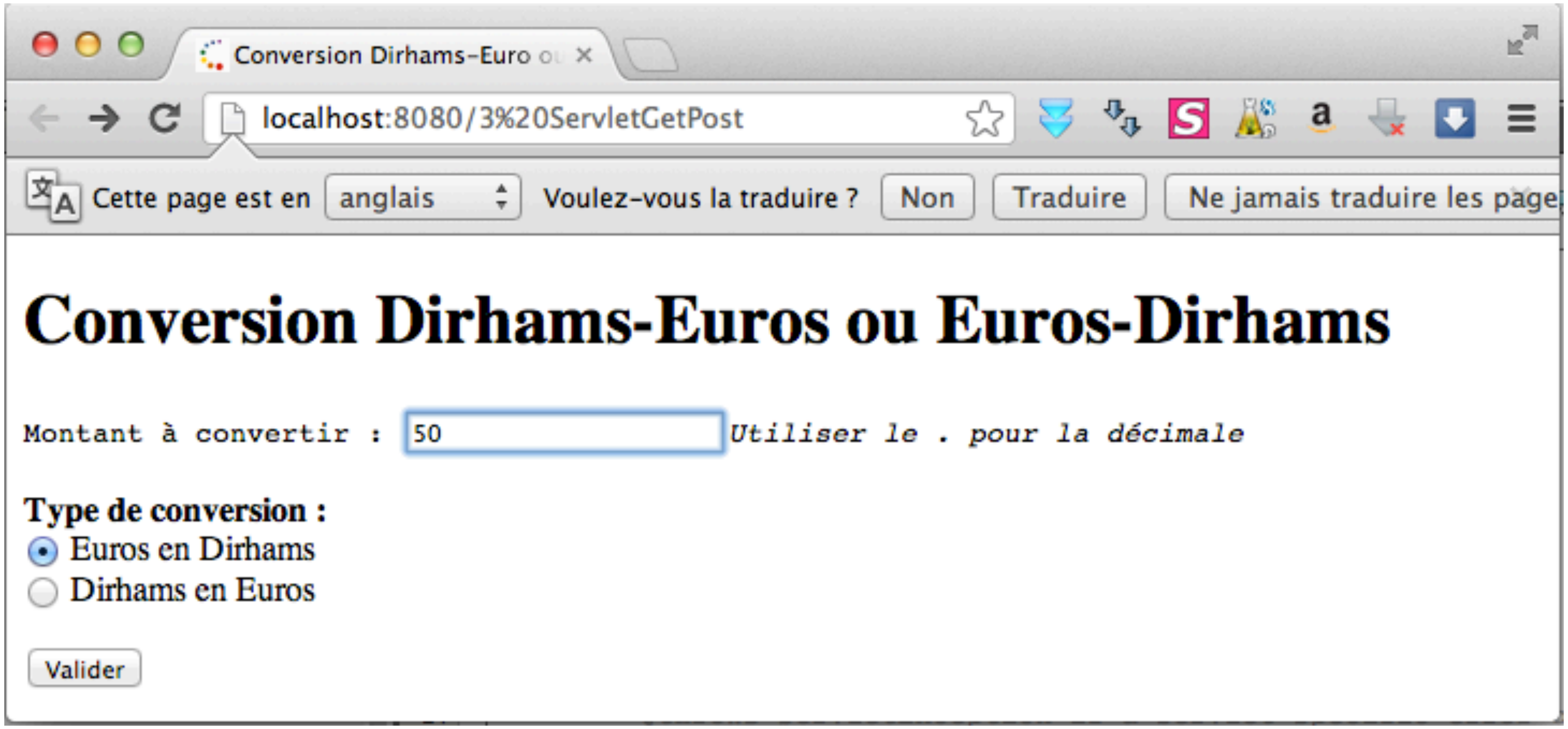
Montant à convertir : ____ *Utiliser le . pour la décimale*

Type de conversion :

[✓ Euros en Dirhams]
Dirhams en Euros

Valider

Exemple d'exécution


Conversion Dirhams-Euro ou
localhost:8080/3%20ServletGetPost
Cette page est en anglais Voulez-vous la traduire ? Non Traduire Ne jamais traduire les page

Conversion Dirhams-Euros ou Euros-Dirhams

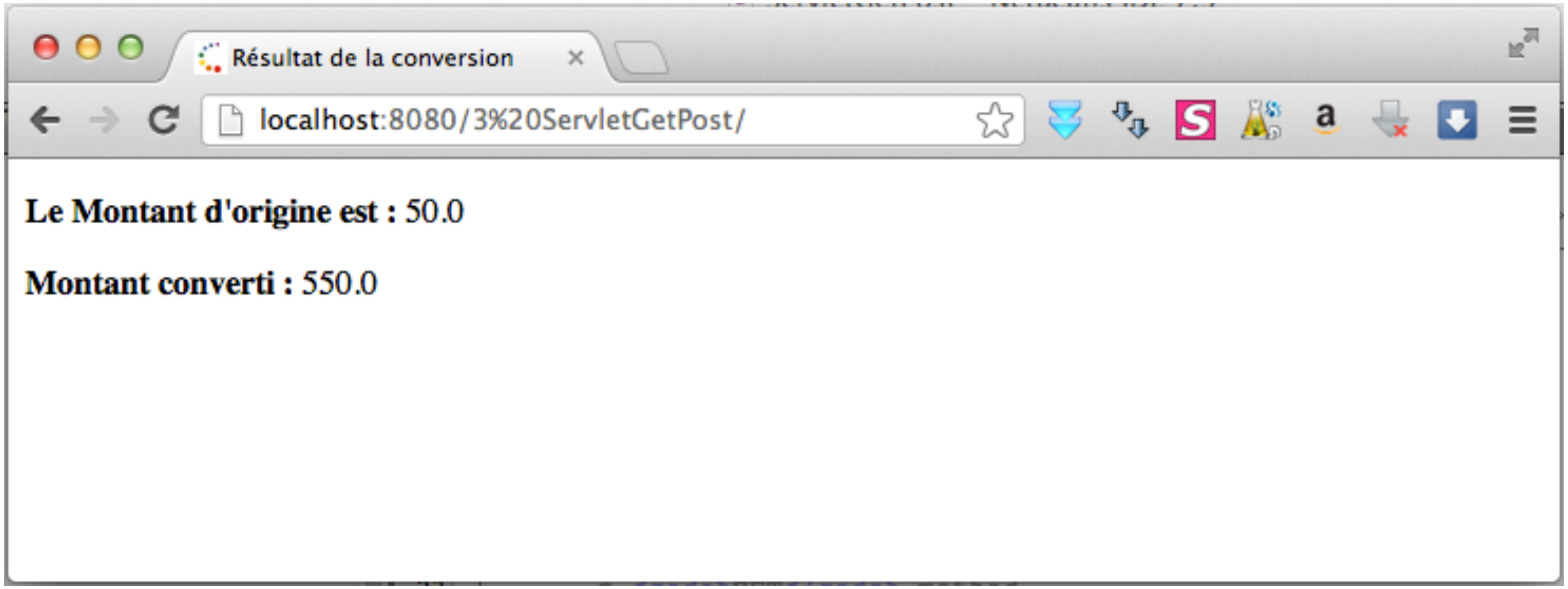
Montant à convertir : 50 *Utiliser le . pour la décimale*

Type de conversion :

[✓ Euros en Dirhams]
Dirhams en Euros

Valider

Résultat d'une bonne exécution



Le Montant d'origine est : 50.0

Montant converti : 550.0

DEMO

6 Redirecting Requests to Other Resources

- Servlet **`RedirectServlet`**
  - Redirects the request to a different resource
- Utilisation d'un `RequestDispatcher` obtenu via un objet request

```
RequestDispatcher rd = request.getRequestDispatcher( "servlet1" );
```

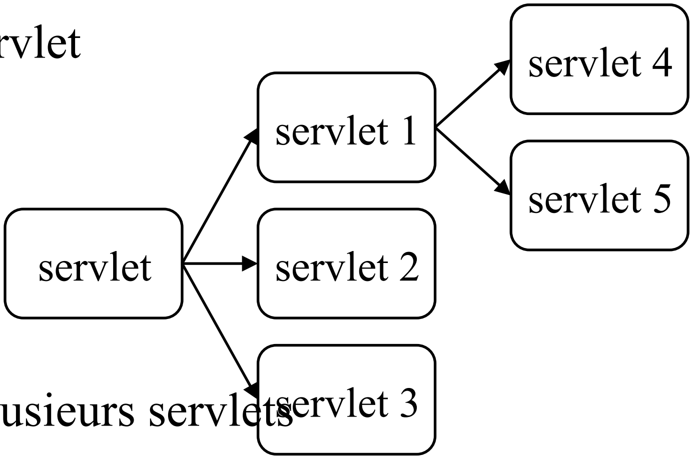
- Délégation du traitement à une autre servlet

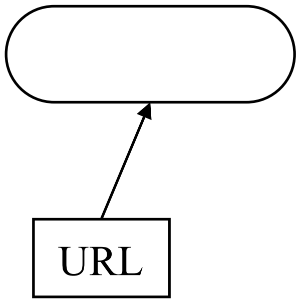
```
rd.forward(request, response);
```

- Inclusion du résultat d'une autre servlet

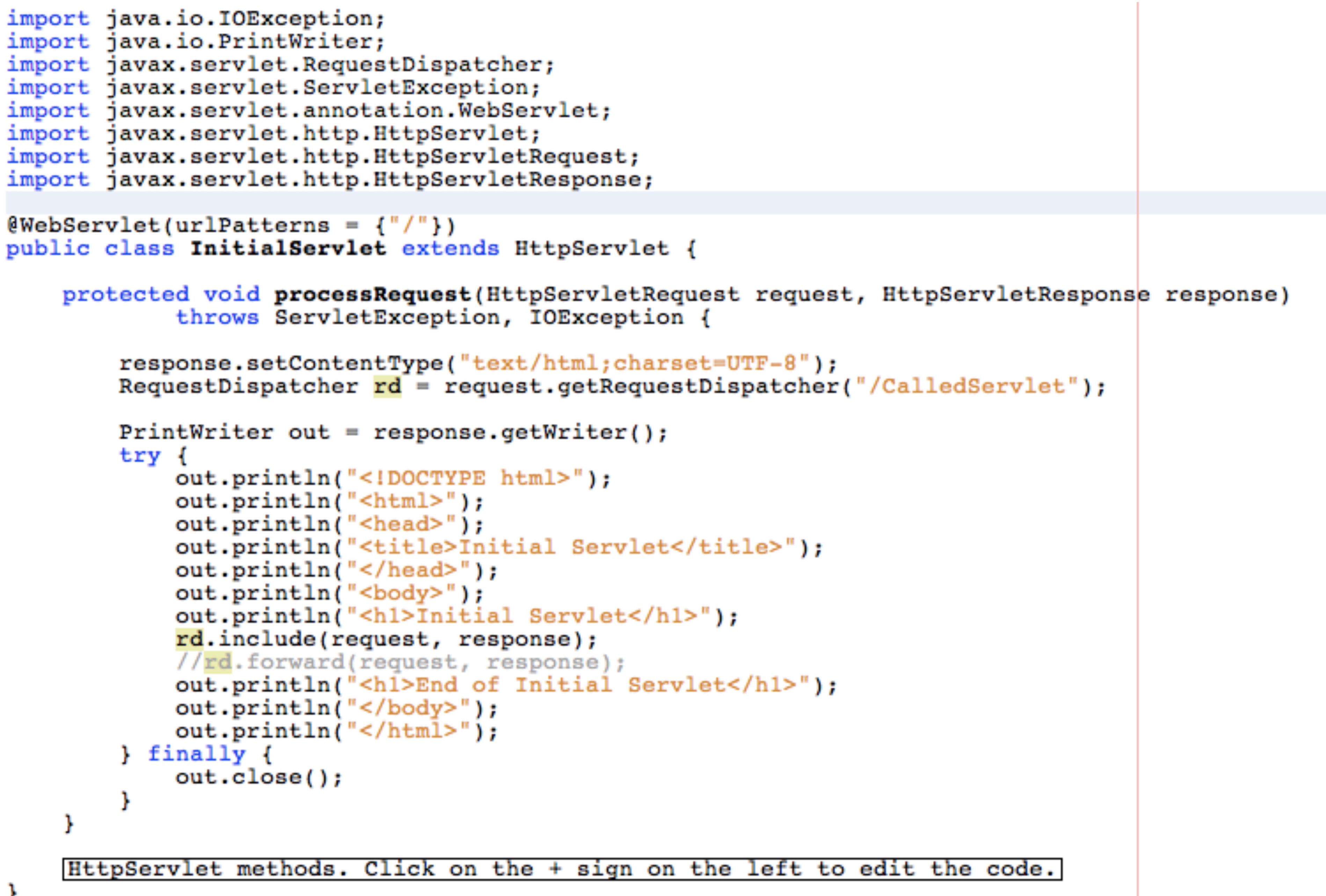
```
rd.include(request, response);
```

- Aggrégation des résultats fournis par plusieurs servlets
  - meilleure modularité
  - meilleure réutilisation



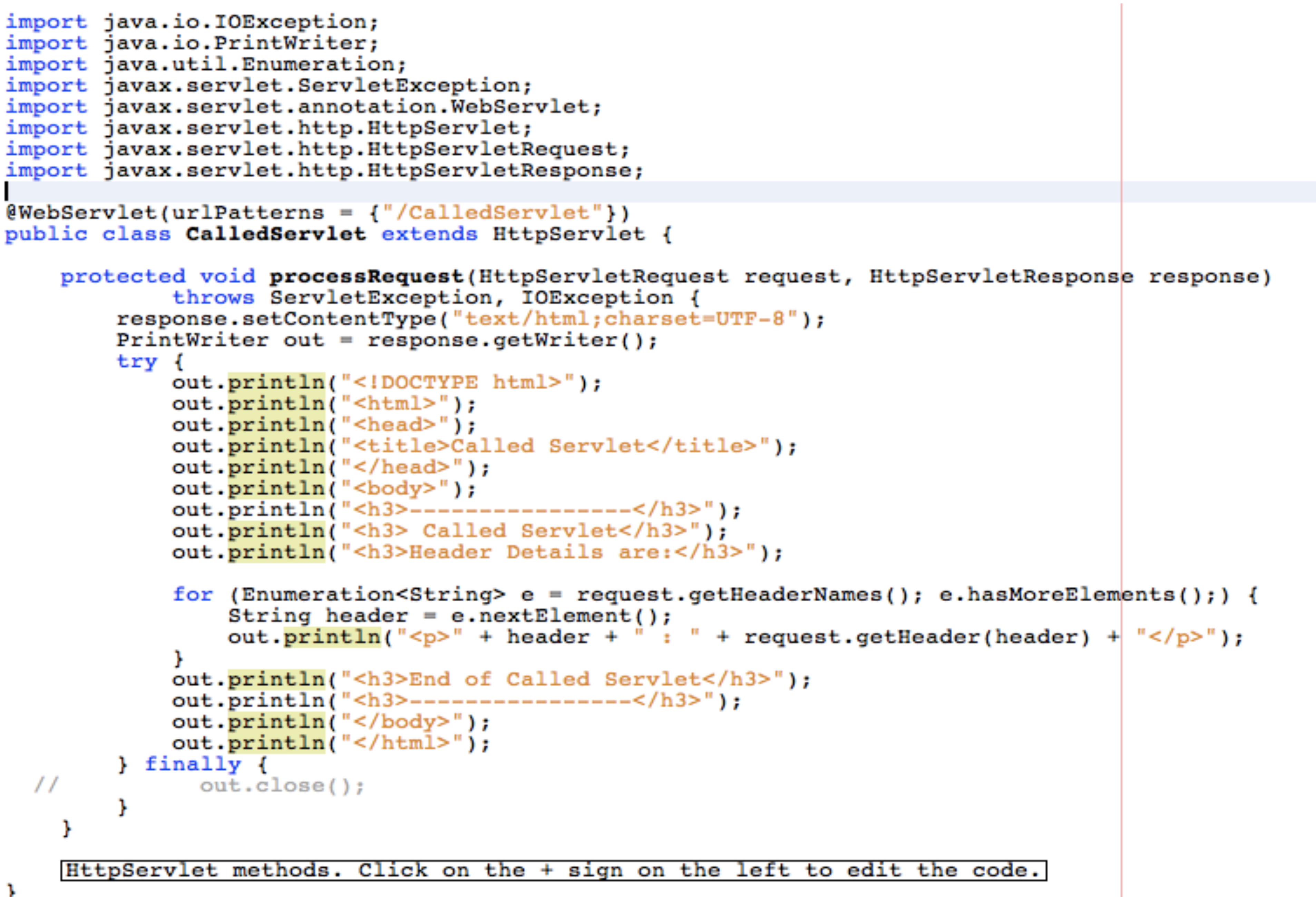


6 - Redirecting Requests to Other Resources (Initial Servlet)



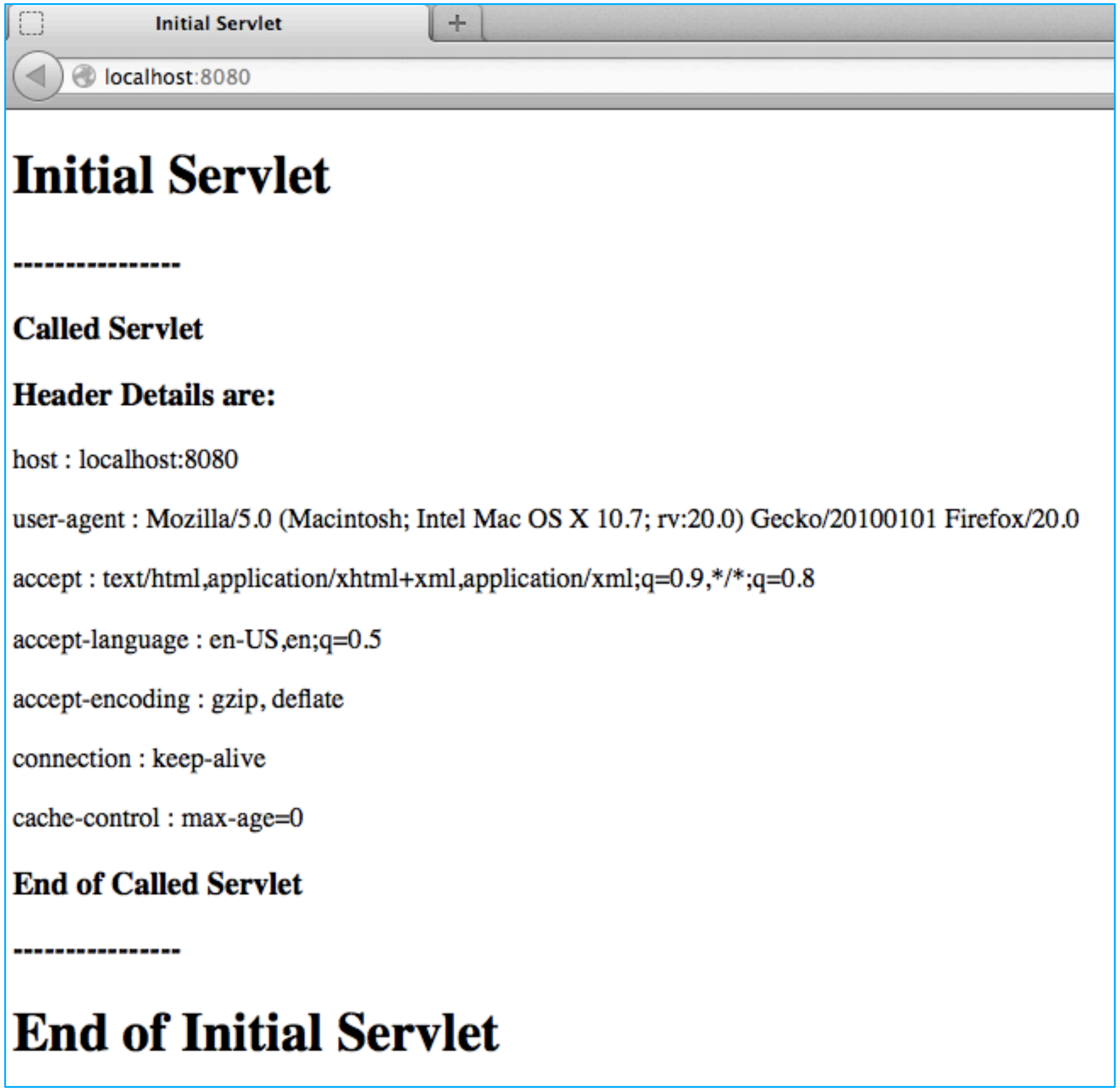
```
import java.io.IOException;
import java.io.PrintWriter;
import javax.servlet.RequestDispatcher;
import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

@WebServlet(urlPatterns = {"/"})
public class InitialServlet extends HttpServlet {

    protected void processRequest(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {

        response.setContentType("text/html;charset=UTF-8");
        RequestDispatcher rd = request.getRequestDispatcher("/CalledServlet");

        PrintWriter out = response.getWriter();
        try {
            out.println("<!DOCTYPE html>");
            out.println("<html>");
            out.println("<head>");
            out.println("<title>Initial Servlet</title>");
            out.println("</head>");
            out.println("<body>");
            out.println("<h1>Initial Servlet</h1>");
            rd.include(request, response);
            //rd.forward(request, response);
            out.println("<h1>End of Initial Servlet</h1>");
            out.println("</body>");
            out.println("</html>");
        } finally {
            out.close();
        }
    }

    HttpServlet methods. Click on the + sign on the left to edit the code.
}
```

6 - Redirecting Requests to Other Resources (Called Servlet)



```
import java.io.IOException;
import java.io.PrintWriter;
import java.util.Enumeration;
import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

@WebServlet(urlPatterns = {"/CalledServlet"})
public class CalledServlet extends HttpServlet {

    protected void processRequest(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {
        response.setContentType("text/html;charset=UTF-8");
        PrintWriter out = response.getWriter();
        try {
            out.println("<!DOCTYPE html>");
            out.println("<html>");
            out.println("<head>");
            out.println("<title>Called Servlet</title>");
            out.println("</head>");
            out.println("<body>");
            out.println("<h3>----------------</h3>");
            out.println("<h3> Called Servlet</h3>");
            out.println("<h3>Header Details are:</h3>");

            for (Enumeration<String> e = request.getHeaderNames(); e.hasMoreElements();) {
                String header = e.nextElement();
                out.println("<p>" + header + " : " + request.getHeader(header) + "</p>");
            }
            out.println("<h3>End of Called Servlet</h3>");
            out.println("<h3>----------------</h3>");
            out.println("</body>");
            out.println("</html>");
        } finally {
  //            out.close();
        }
    }

    HttpServlet methods. Click on the + sign on the left to edit the code.
}
```

```
rd.include(request, response); // in Initial Servlet
```

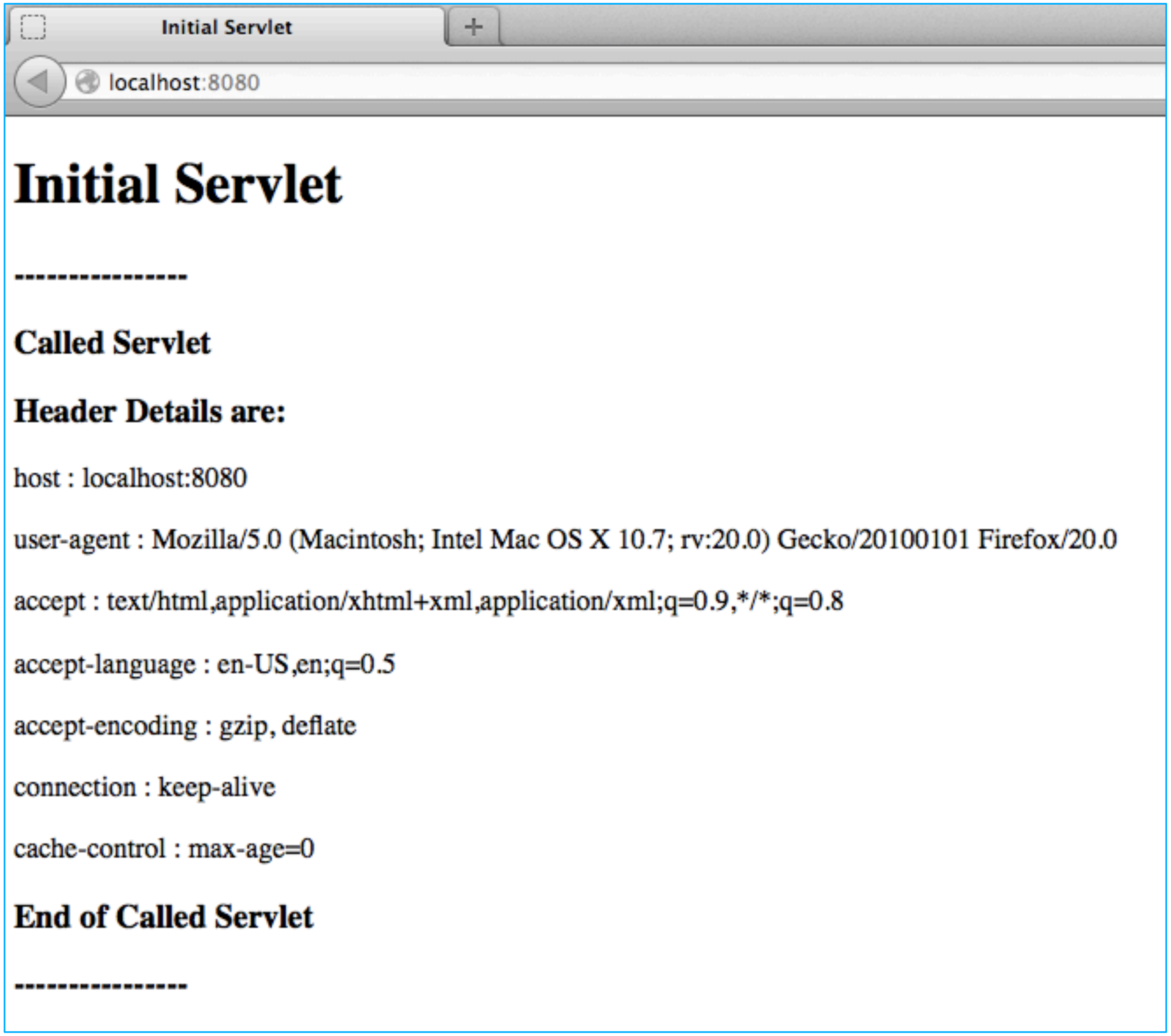


Initial Servlet

localhost:8080

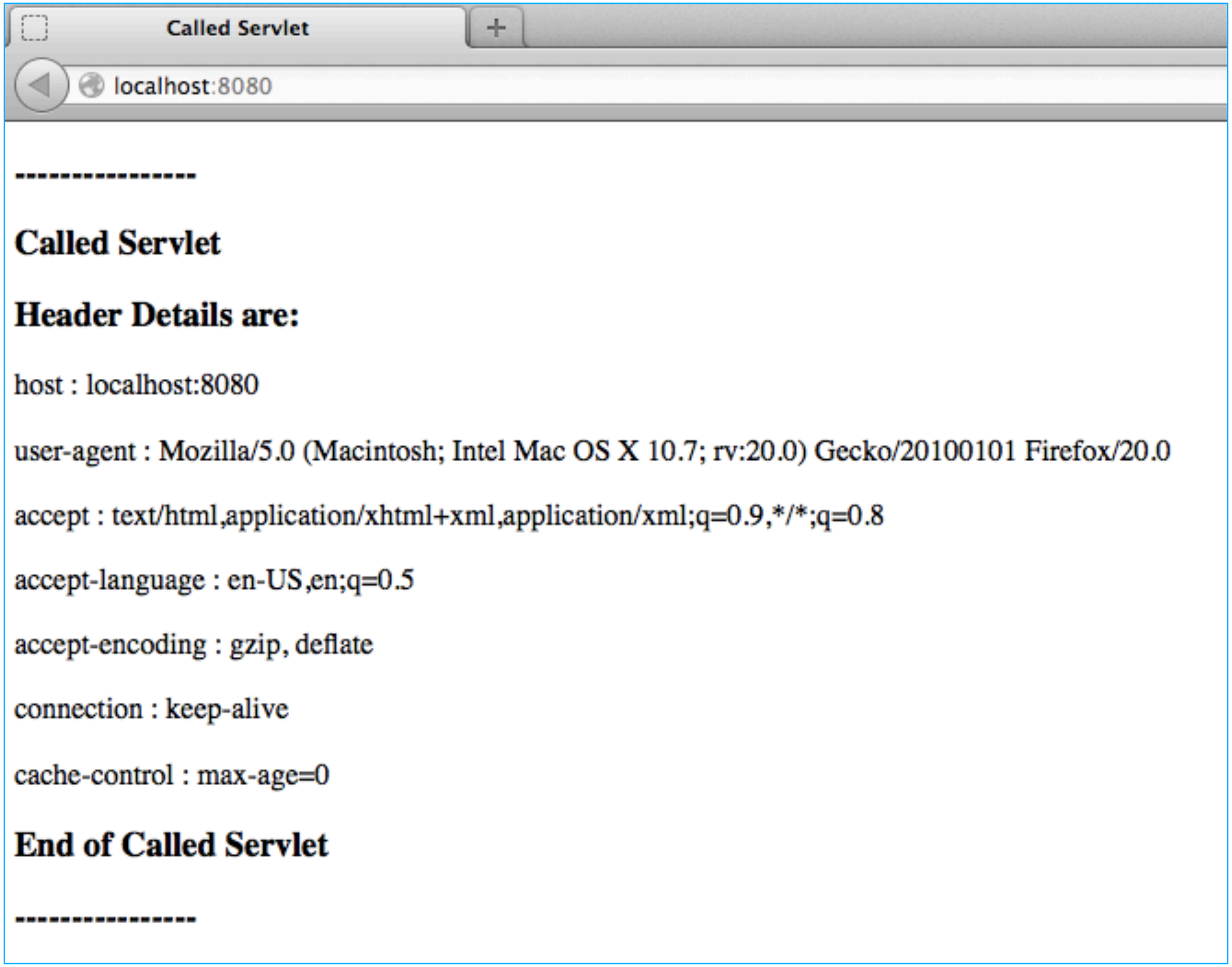
```
Initial Servlet

----------------

Called Servlet

Header Details are:

host : localhost:8080

user-agent : Mozilla/5.0 (Macintosh; Intel Mac OS X 10.7; rv:20.0) Gecko/20100101 Firefox/20.0

accept : text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8

accept-language : en-US,en;q=0.5

accept-encoding : gzip, deflate

connection : keep-alive

cache-control : max-age=0

End of Called Servlet

----------------

End of Initial Servlet
```

`out.close();` // in Called Servlet



```
Initial Servlet

----------------

Called Servlet

Header Details are:

host : localhost:8080

user-agent : Mozilla/5.0 (Macintosh; Intel Mac OS X 10.7; rv:20.0) Gecko/20100101 Firefox/20.0

accept : text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8

accept-language : en-US,en;q=0.5

accept-encoding : gzip, deflate

connection : keep-alive

cache-control : max-age=0

End of Called Servlet

----------------
```

`rd.forward(request, response);` // in Initial Servlet



```
----------------

Called Servlet

Header Details are:

host : localhost:8080

user-agent : Mozilla/5.0 (Macintosh; Intel Mac OS X 10.7; rv:20.0) Gecko/20100101 Firefox/20.0

accept : text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8

accept-language : en-US,en;q=0.5

accept-encoding : gzip, deflate

connection : keep-alive

cache-control : max-age=0

End of Called Servlet

----------------
```

DEMO

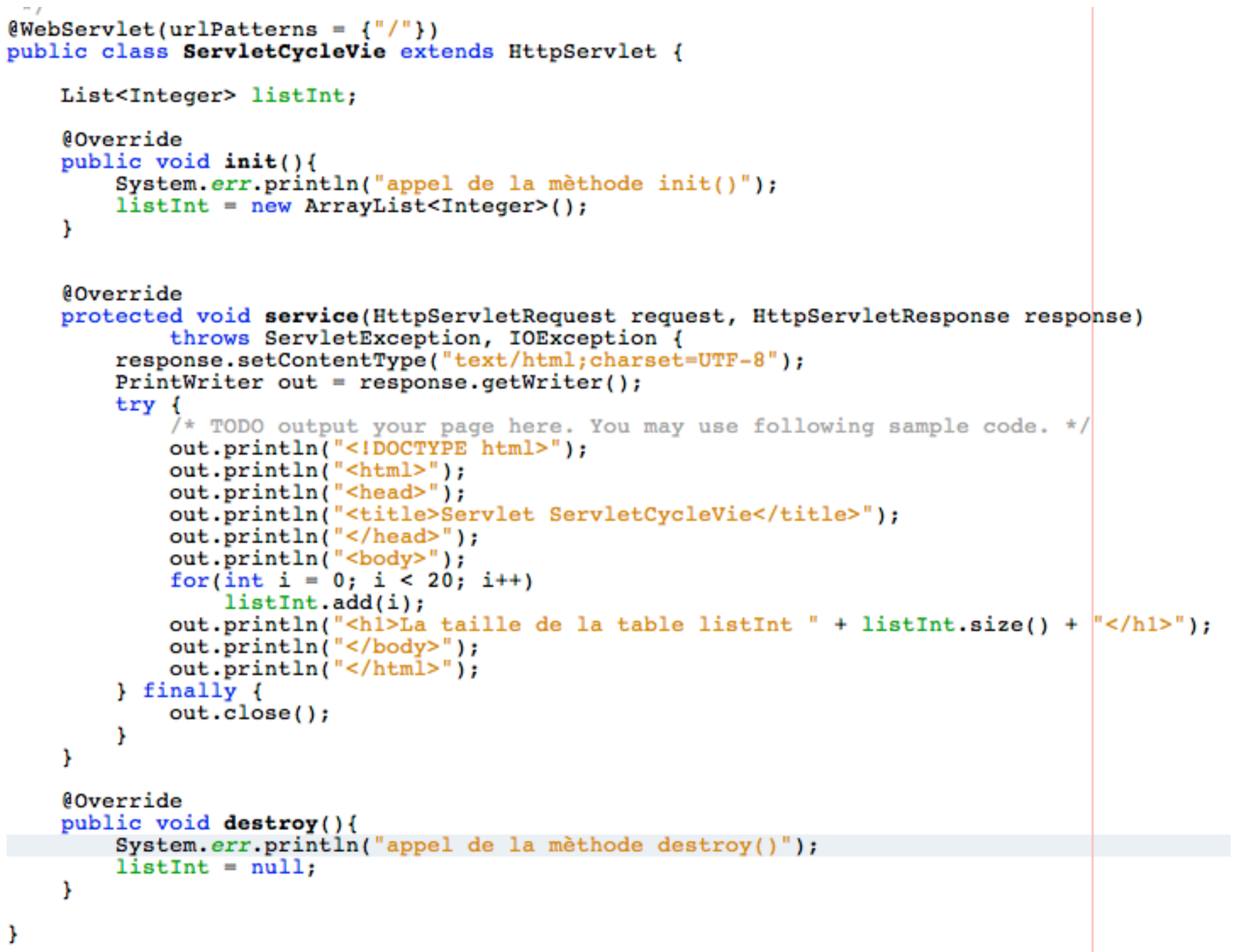
Le cycle de vie

1. la *servlet* est créée puis initialisée (**`init()`**)
   - cette méthode n’est appelée par le serveur <u>qu’une seule fois</u> lors du chargement en mémoire par le moteur de servlet

2. le service du client est implémenté (**`service()`**)
   - cette méthode est appelée automatiquement par le serveur à chaque requête de client

3. la *servlet* est détruite (**`destroy()`** **)**
   - cette méthode n’est appelée par le serveur <u>qu’une seule fois</u> à la fin
   - permet de libérer des ressources (allouées par `init()`)

Illustration du cycle de vie d’une servlet

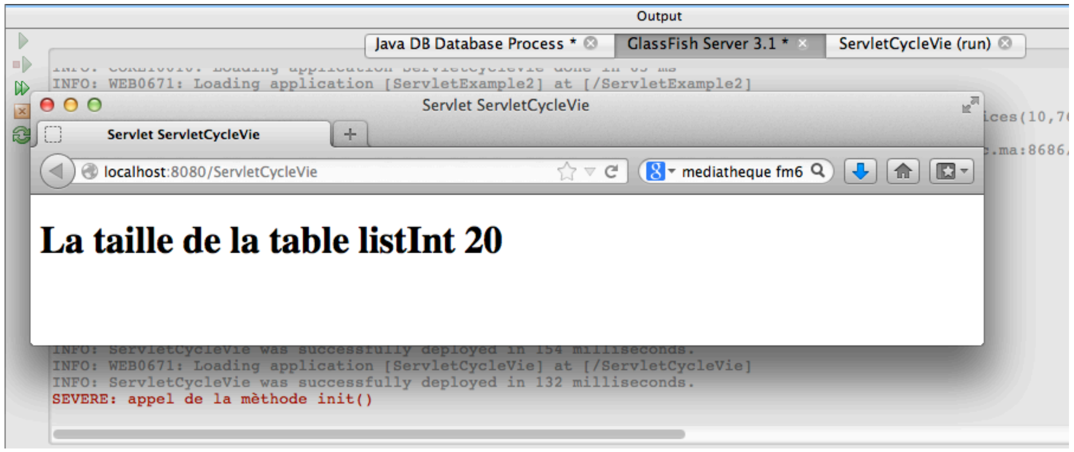


```
 */
@WebServlet(urlPatterns = {"/"})
public class ServletCycleVie extends HttpServlet {

    List<Integer> listInt;

    @Override
    public void init(){
        System.err.println("appel de la mèthode init()");
        listInt = new ArrayList<Integer>();
    }


    @Override
    protected void service(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {
        response.setContentType("text/html;charset=UTF-8");
        PrintWriter out = response.getWriter();
        try {
            /* TODO output your page here. You may use following sample code. */
            out.println("<!DOCTYPE html>");
            out.println("<html>");
            out.println("<head>");
            out.println("<title>Servlet ServletCycleVie</title>");
            out.println("</head>");
            out.println("<body>");
            for(int i = 0; i < 20; i++)
                listInt.add(i);
            out.println("<h1>La taille de la table listInt " + listInt.size() + "</h1>");
            out.println("</body>");
            out.println("</html>");
        } finally {
            out.close();
        }
    }

    @Override
    public void destroy(){
        System.err.println("appel de la mèthode destroy()");
        listInt = null;
    }

}
```

Deux exécutions successives de l'exemple

Chaque servlet n'est instanciée **1 seule fois** → persistance de ces données entre 2 invocations

1ère invocation

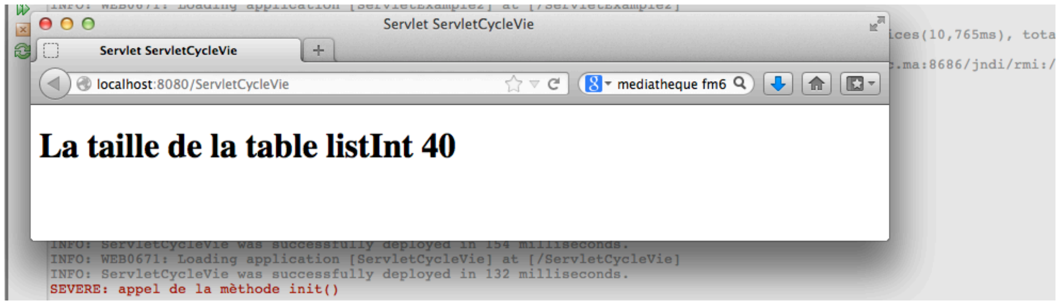


```
INFO: WEB0671: Loading application [ServletExample2] at [/ServletExample2]
```

La taille de la table listInt 20

```
INFO: ServletCycleVie was successfully deployed in 154 milliseconds.
INFO: WEB0671: Loading application [ServletCycleVie] at [/ServletCycleVie]
INFO: ServletCycleVie was successfully deployed in 132 milliseconds.
SEVERE: appel de la mèthode init()
```

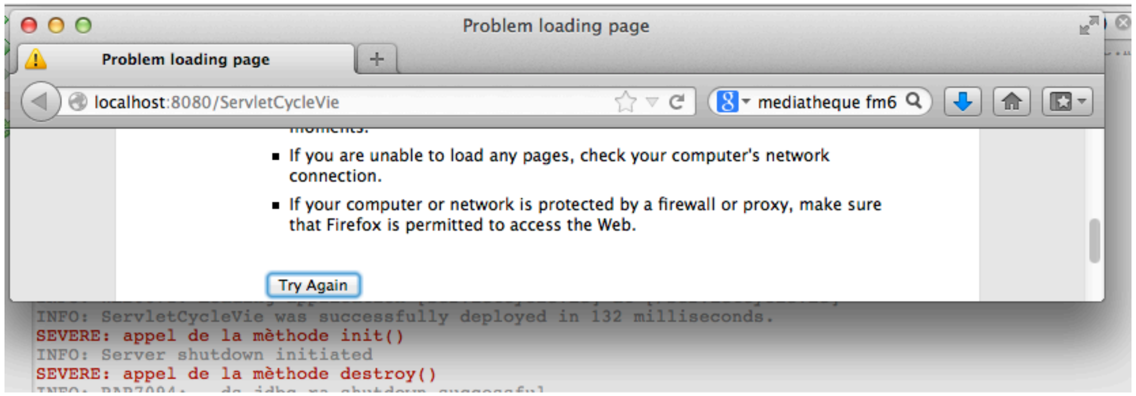
2ème invocation



La taille de la table listInt 40

```
INFO: ServletCycleVie was successfully deployed in 154 milliseconds.
INFO: WEB0671: Loading application [ServletCycleVie] at [/ServletCycleVie]
INFO: ServletCycleVie was successfully deployed in 132 milliseconds.
SEVERE: appel de la mèthode init()
```

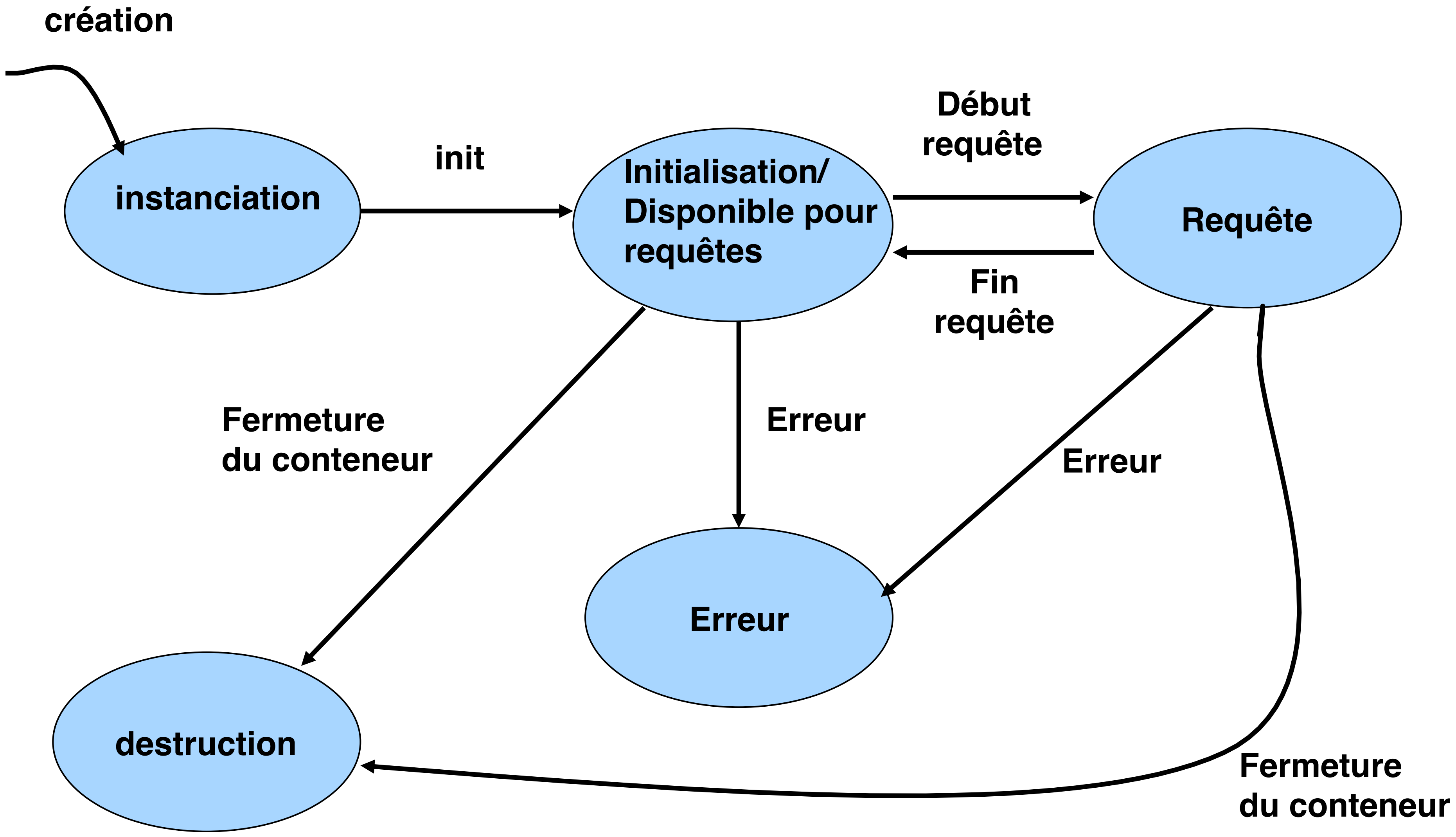
Arrêt



- If you are unable to load any pages, check your computer's network connection.
- If your computer or network is protected by a firewall or proxy, make sure that Firefox is permitted to access the Web.

```
INFO: ServletCycleVie was successfully deployed in 132 milliseconds.
SEVERE: appel de la mèthode init()
INFO: Server shutdown initiated
SEVERE: appel de la mèthode destroy()
```

Cycle de vie


création
instanciation
init
Initialisation/ Disponible pour requêtes
Début requête
Requête
Fin requête
Fermeture du conteneur
Erreur
Erreur
Erreur
destruction
Fermeture du conteneur

DEMO

Problématique

- Protocole HTTP = protocole Internet <u>déconnecté</u>
  - différent de Telnet, Ftp, …
  - traite les requêtes et les réponses comme transactions simples et isolées

- Certaines applications Web (e-commerce : caddie) ont besoin de maintenir une "mémoire" entre deux requêtes
  - ie. maintenir une connexion de l'utilisateur sur le serveur
  - pour se faire : concept de "suivi de sessions"

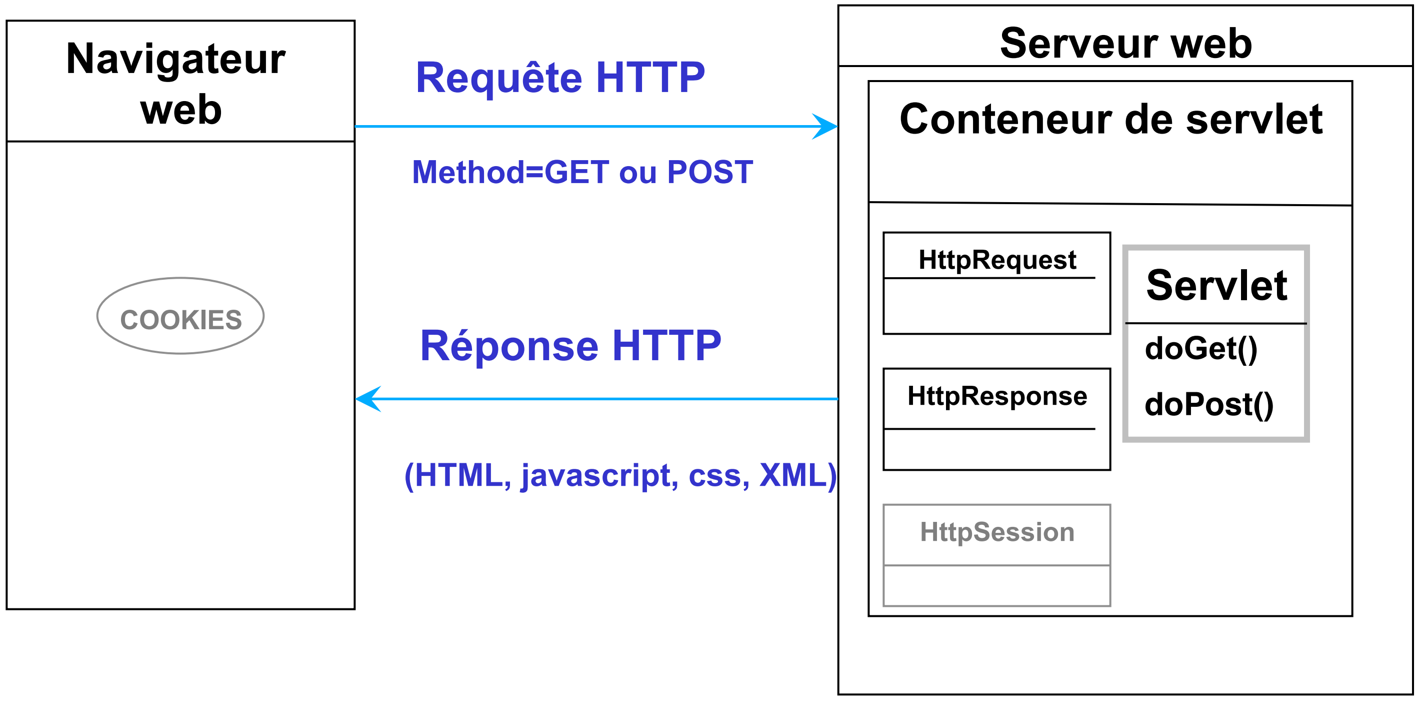
Suivi de session : qu'est-ce que c'est ?

- Mémoire de ce que fait l'utilisateur d'une page à l'autre
  - consiste au transfert de données générées par une requête vers les requêtes suivantes

- 4 méthodes avec les servlets Java
  1. utilisation des cookies
  2. utilisation du JSDK (HttpSession API)
  3. réécriture d'URL : passage de paramètres
  4. utilisation des champs de formulaire "hidden"

Cookies

- Données textuelles envoyées par le serveur au client
- Stockées chez le client
- Renvoyées vers le serveur lors de toute requête vers le serveur
- Durée de vie réglable
- Permet la persistance



A quoi ça sert ?

- Identification des utilisateurs (e-commerce)
- Eviter la saisie d’informations à répétition
  - login, password, adresse, téléphone…
- Gérer des « préférences utilisateur »
  - sites portails…
- …

Cookie et sécurité

- Jamais interprété ou exécuté : pas de virus
- Un cookie est limité à 4KB et les navigateurs se limitent à 300 cookies (20 par site) : pas de surcharge de disque
- Bien pour rendre privées des données non sensibles
  - nom, adresse, … mais pas N° CB !
- … mais ne constitue pas un traitement sérieux de la sécurité

Manipuler les cookies

- Utiliser les fonctions de l'API des servlets…
  - créer un cookie : utiliser la classe **`Cookie`**
  - écrire/lire un cookie : **`addCookie(cookie), getCookies()`**
  - positionner des attributs d'un cookie : **`cookie.setXxx`**(…)

- Exemple d'envoi d'un cookie :

```
...
String nom = request.getParameter("nom");
Cookie unCookie = new Cookie("nom", nom);
...ici positionner des attributs si on le désire
response.addCookie(unCookie);
...
```

Création d'un cookie

- **`Cookie unCookie = new Cookie(name, value);`**
  - 2 arguments de type **`java.lang.String`** :
    - **`name`** et **`value`**
  - caractères non autorisés :
    - espace blanc
    - [ ] ( ) = , " / ? @ : ;

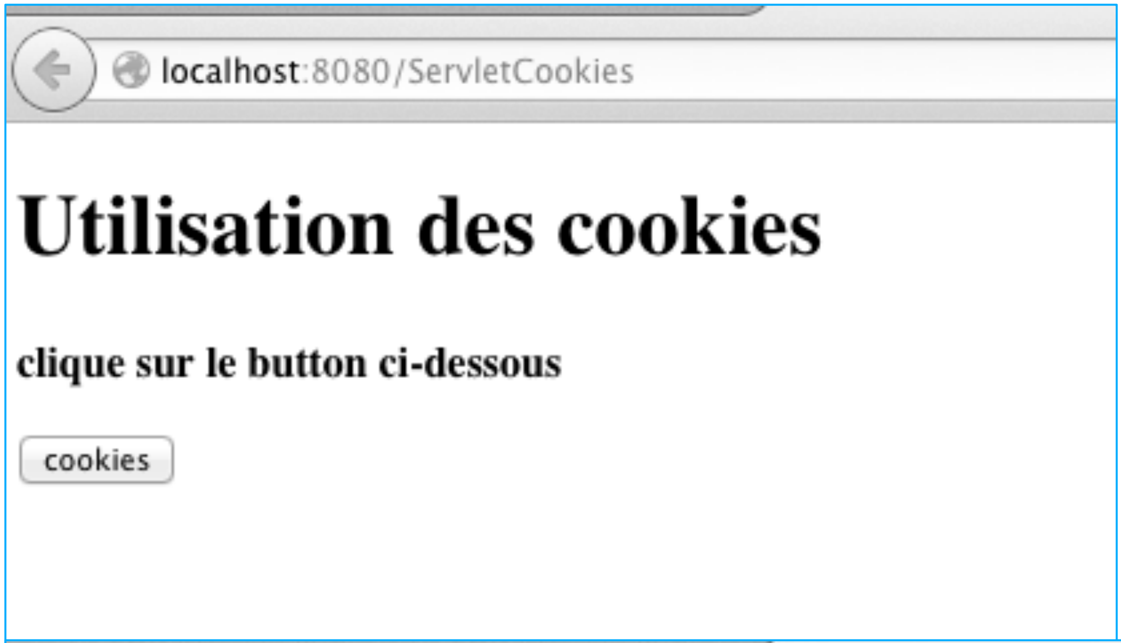
Récupération des cookies

- Exemple de récupération des cookies

```
Cookie [] cookies = request.getCookies();

String nom = getCookieValue(cookies, "nom", "non trouvé");

...

public static String getCookieValue(Cookie [] cookies,
              String cookieName, String defaultValue) {
   for(int i=0; i < cookies.length; i++) {
     Cookie cookie = cookies[i];
    if(cookieName.equals(cookie.getName())
   return(cookie.getValue());
   }
   return(defaultValue);
}
```

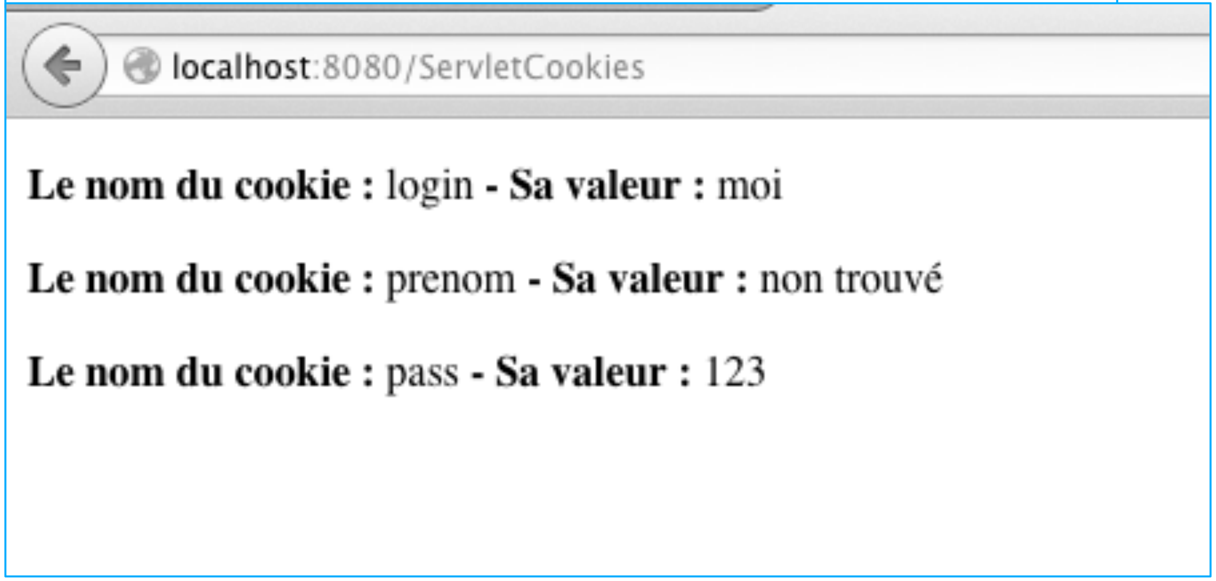
```java
@WebServlet(urlPatterns = {"/"})
public class Cookies extends HttpServlet {

    Cookie[] cookies;

    @Override
    protected void doGet(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {

        response.setContentType("text/html;charset=UTF-8");
        PrintWriter out = response.getWriter();
        try {
            out.println("<!DOCTYPE html>");
            out.println("<html>");
            out.println("<head><title>Cookies Get</title></head>");
            out.println("<FORM METHOD=POST ACTION=http://localhost:8080/ServletCookies>");
            out.println("<h1><B>Utilisation des cookies</B></h1>");
            out.println("<P><B>clique sur le button ci-dessous</B><BR>");
            out.println("<P><input type=submit value=\"cookies\"></P>");
            out.println("</FORM>");
            out.println("</html>");

            response.addCookie(new Cookie("login", "moi"));
            response.addCookie(new Cookie("pass", "123"));

        } finally {
            out.close();
        }
    }
```

```
@Override
protected void doPost(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException {

    String cookie;
    String valeur;
    cookies = request.getCookies();

    response.setContentType("text/html;charset=UTF-8");
    PrintWriter out = response.getWriter();
    try {
        out.println("<!DOCTYPE html>");
        out.println("<html>");
        out.println("<head><title>Cockies Post</title></head>");

        cookie = "login";
        valeur = getCookieValue(cookies, cookie, "non trouvé");
        out.print("<P><B> Le nom du cookie : </B>" + cookie );
        out.print("<B> - Sa valeur : </B>" + valeur + "</P>");

        cookie = "prenom";
        valeur = getCookieValue(cookies, cookie, "non trouvé");
        out.print("<P><B> Le nom du cookie : </B>" + cookie );
        out.print("<B> - Sa valeur : </B>" + valeur + "</P>");

        cookie = "pass";
        valeur = getCookieValue(cookies, cookie, "non trouvé");
        out.print("<P><B> Le nom du cookie : </B>" + cookie );
        out.print("<B> - Sa valeur : </B>" + valeur + "</P>");

        out.println("</html>");
        out.close();
    } finally {
        out.close();
    }
}

public static String getCookieValue(Cookie[] cookies, String cookieName, String defaultValue) {
```


localhost:8080/ServletCookies

Utilisation des cookies

clique sur le button ci-dessous


localhost:8080/ServletCookies

Le nom du cookie : login **- Sa valeur :** moi

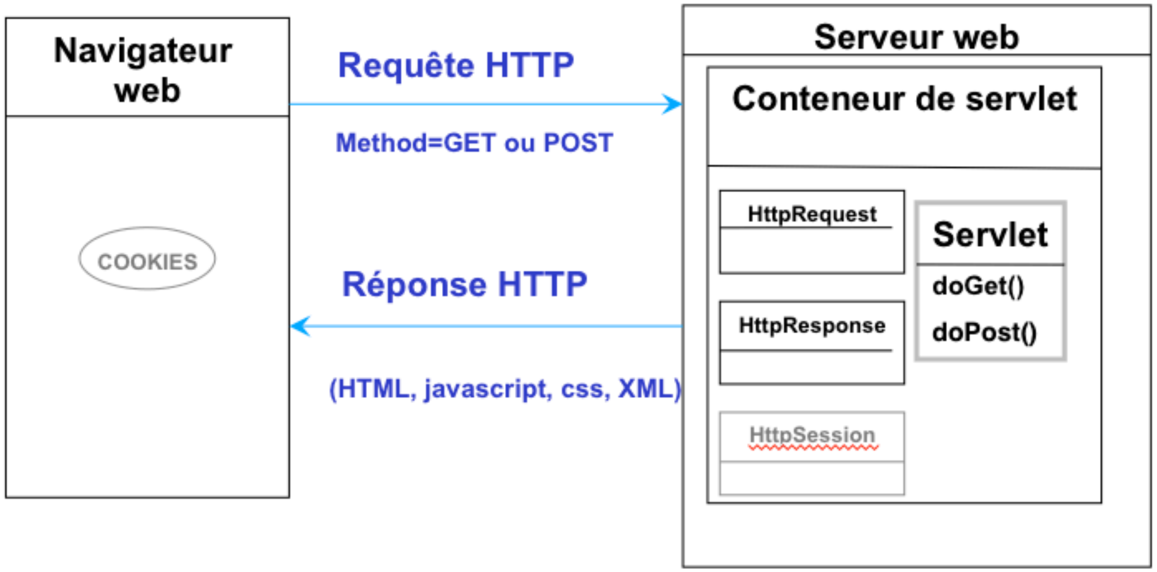
Le nom du cookie : prenom **- Sa valeur :** non trouvé

Le nom du cookie : pass **- Sa valeur :** 123

DEMO

L'objet session

- Très simple avec l'API des servlets (JSDK)
  - objet **`HttpSession`**
- Principe :
  - Un objet "session" peut être associé *à chaque requête*
  - Il va servir de "container" pour des informations persistantes
  - Durée de vie limitée et réglable



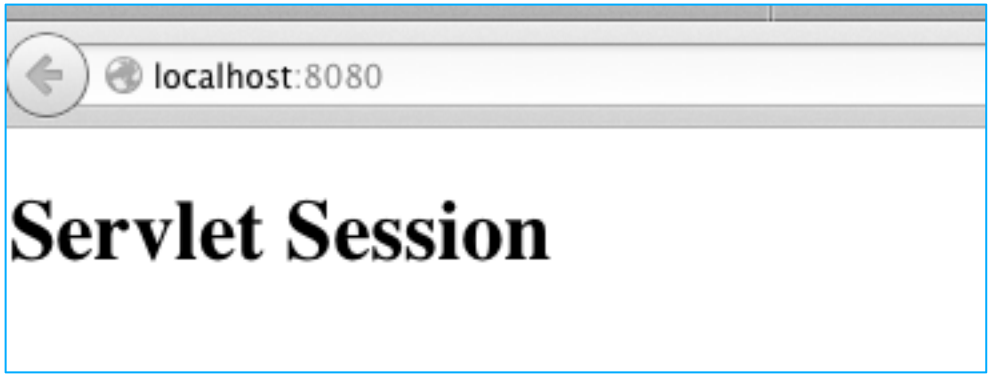
Servlet: HttpSession

- API de suivi de session HttpSession
- Méthodes de création liées à la requête (HttpServletRequest)
  - HttpSession getSession() : retourne la session associée à l’utilisateur
  - HttpSession getSession(boolean p) : création selon la valeur de p
- Gestion d’association (HttpSession)
  - Enumeration getAttributeNames() : retourne les noms de tous les attributs
  - Object getAttribute(String name) : retourne l’objet associé au nom
  - setAttribute(String na, Object va) : modifie na par la valeur va
  - removeAttribute(String na) : supprime l’attribut associé à na
- Destruction (HttpSession)
  - invalidate() : expire la session
  - logout() : termine la session

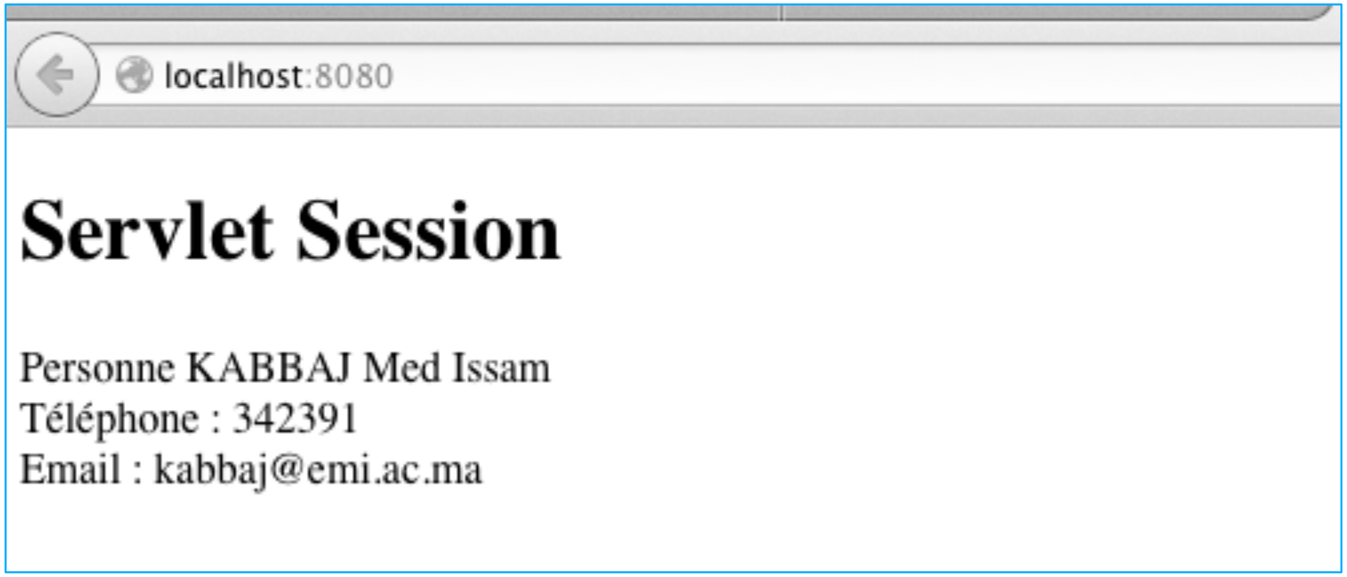
Modèle basique

```
HttpSession session = request.getSession(true);
Caddy caddy = (Caddy) session.getValue("caddy");

if(caddy != null) {
   // le caddy n'est pas vide !
  afficheLeContenuDuCaddy(caddy);
   …
   caddy.ajouterUnAchat(request.getParameter("NoArticle2"));
   session.putValue("caddy", caddy);
   …
} else {
   caddy = new Caddy();
   ...
   caddy.ajouterUnAchat(request.getParameter("NoArticle1"));
   session.putValue("caddy", caddy);
}....
```

```java
import java.io.IOException;
import java.io.PrintWriter;
import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import javax.servlet.http.HttpSession;

@WebServlet(urlPatterns = {"/"})
public class ServletSession extends HttpServlet {

    protected void processRequest(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {
        response.setContentType("text/html;charset=UTF-8");
        PrintWriter out = response.getWriter();

        HttpSession session = request.getSession(true);
        Personne personne = (Personne) session.getAttribute("personne");

        try {
            out.println("<!DOCTYPE html>");
            out.println("<html>");
            out.println("<head>");
            out.println("<title>Servlet Session</title>");
            out.println("</head>");
            out.println("<body>");
            out.println("<h1>Servlet Session</h1>");
            if (personne != null) {
                out.println("<p>Personne " + personne.getNom() + " " + personne.getPrenom() + "<br/>");
                out.println(" Téléphone : " + personne.getTéléphone() + "<br/>");
                out.println(" Email : " + personne.getEmail() + "<br/>");
            } else {
                personne = new Personne("KABBAJ", "Med Issam", 01234567, "kabbaj@emi.ac.ma");
                session.setAttribute("personne", personne);
            }
            out.println("</body>");
            out.println("</html>");
        } finally {
            out.close();
        }
    }

    HttpServlet methods. Click on the + sign on the left to edit the code.
}
```


localhost:8080

Servlet Session


localhost:8080

Servlet Session

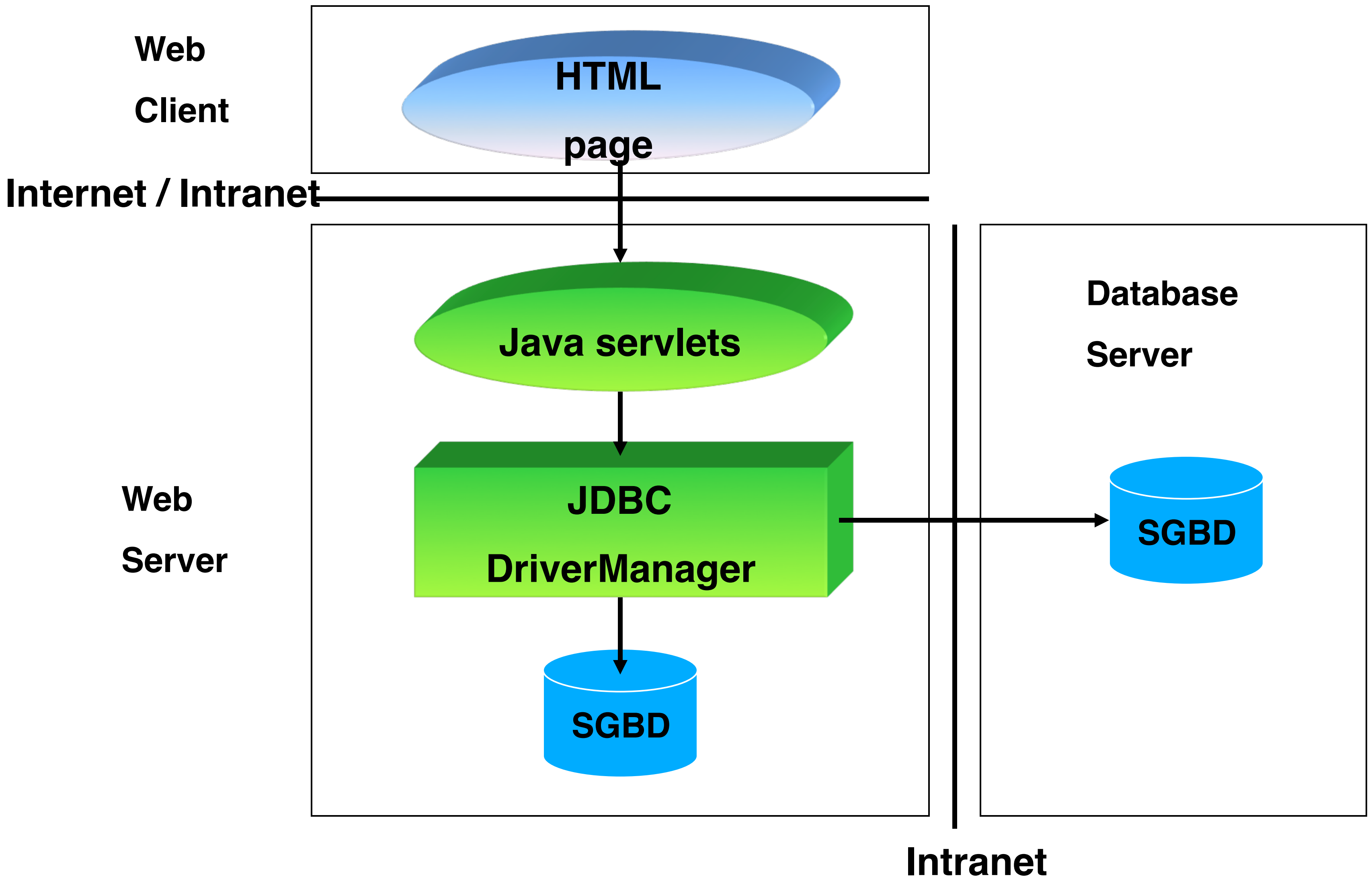
Personne KABBAJ Med Issam
Téléphone : 342391
Email : kabbaj@emi.ac.ma

DEMO

7 Multi-Tier Applications: Using JDBC from a Servlet

- Three-tier distributed applications
  - User interface
  - Business logic
  - Database access
- Web servers often represent the middle tier
- Three-tier distributed application example
  - Yours to do

Architecture



```
1   // Fig. 20: SurveyServlet.java
2   // A Web-based survey that uses JDBC from a servlet.
3   package com.deitel.jhtp5.servlets;
4
5   import java.io.*;
6   import java.text.*;
7   import java.sql.*;
8   import javax.servlet.*;
9   import javax.servlet.http.*;
10
11  public class SurveyServlet extends HttpServlet {
12     private Connection connection;
13     private Statement statement;
14
15     // set up database connection and create SQL statement
16     public void init( ServletConfig config ) throws ServletException
17     {
18        // attempt database connection and create Statements
19        try {
20           System.setProperty( "db2j.system.home",
21              config.getInitParameter( "databaseLocation" ) );
22
23           Class.forName( config.getInitParameter( "databaseDriver" ) );
24           connection = DriverManager.getConnection(
25              config.getInitParameter( "databaseName" ) );
```

Servlets are initialized by overriding method `init`.

Specify database location

Loads the database driver.

Attempt to c… o the `animalsurvey` database.

SurveyServlet.java
Multi-tier Web-based survey using XHTML, servlets and JDBC.

Lines 16-38

Lines 20-21

Line 23

I

```
26
27          // create Statement to query database
28          statement = connection.createStatement();
29       }
30
31       // for any exception throw an UnavailableException to
32       // indicate that the servlet is not currently available
33       catch ( Exception exception ) {
34          exception.printStackTrace();
35          throw new UnavailableException(exception.getMessage());
36       }
37
38    }  // end of init method
39
40    // process survey response
41    protected void doPost( HttpServletRequest request,
42       HttpServletResponse response )
43          throws ServletException, IOException
44    {
45       // set up response to client
46       response.setContentType( "text/html" );
47       PrintWriter out = response.getWriter();
48       DecimalFormat twoDigits = new DecimalFormat( "0.00" );
49
```

Create `Statement` to query database.

SurveyServlet.java
Multi-tier Web-based survey using XHTML, servlets and JDBC.

Line 28

```
50         // start XHTML document
51         out.println( "<?xml version = \"1.0\"?>" );
52
53         out.println( "<!DOCTYPE html PUBLIC \"-//W3C//DTD " +
54            "XHTML 1.0 Strict//EN\" \"http://www.w3.org" +
55            "/TR/xhtml1/DTD/xhtml1-strict.dtd\">" );
56
57         out.println(
58            "<html xmlns = \"http://www.w3.org/1999/xhtml\">" );
59
60         // head section of document
61         out.println( "<head>" );
62
63         // read current survey response
64         int value =
65            Integer.parseInt( request.getParameter( "animal" ) );
66         String query;
67
68         // attempt to process a vote and display current results
69         try {
70
71            // update total for current surevy response
72            query = "UPDATE surveyresults SET votes = votes + 1 " +
73                  "WHERE id = " + value;
74            statement.executeUpdate( query );
75
```

Obtain the survey response

Create query to update total

response

Execute query to update total for current survey response

SurveyServlet.java
Multi-tier Web-based survey using XHTML, servlets and JDBC.

Lines 64-65

2-73

Line 74

```
76           // get total of all survey responses
77           query = "SELECT sum( votes ) FROM surveyresults";
78           ResultSet totalRS = statement.executeQuery( query );
79           totalRS.next();
80           int total = totalRS.getInt( 1 );
81
82           // get results
83           query = "SELECT surveyoption, votes, id FROM surveyresults " +
84              "ORDER BY id";
85           ResultSet resultsRS = statement.executeQuery( query );
86           out.println( "<title>Thank you!</title>" );
87           out.println( "</head>" );
88
89           out.println( "<body>" );
90           out.println( "<p>Thank you for participating." );
91           out.println( "<br />Results:</p><pre>" );
92
93           // process results
94           int votes;
95
96           while ( resultsRS.next() ) {
97              out.print( resultsRS.getString( 1 ) );
98              out.print( ": " );
99              votes = resultsRS.getInt( 2 );
100            out.print( twoDigits.format(
101               ( double ) votes / total * 100 ) );
102            out.print( "%  responses: " );
103            out.println( votes );
104         }
```

Create query to get total of all su

Execute query to get total of all survey responses

Create query to get

Execute query to get survey results

SurveyServlet.java
Multi-tier Web-
JDBC.

Line 77

Line 78

Lines 83-84

Line 85

```java

         resultsRS.close();

         out.print( "Total responses: " );
         out.print( total );

         // end XHTML document
         out.println( "</pre></body></html>" );
         out.close();

      } // end try

      // if database exception occurs, return error page
      catch ( SQLException sqlException ) {
         sqlException.printStackTrace();
         out.println( "<title>Error</title>" );
         out.println( "</head>" );
         out.println( "<body><p>Database error occurred. " );
         out.println( "Try again later.</p></body></html>" );
         out.close();
      }

   }  // end of doPost method

```

SurveyServlet.java
Multi-tier Web-based survey using XHTML, servlets and JDBC.

```
129    // close SQL statements and data                         tes
130    public void destroy()
131    {
132       // attempt to close statement
133       try {
134          statement.close();
135          connection.close();
136       }
137
138       // handle database exceptions by returning error to client
139       catch ( SQLException sqlException ) {
140          sqlException.printStackTrace();
141       }
142    }
143
144 } // end class SurveyServlet
```

Method destroy closes `Statement` and database connection.

`SurveyServlet.java`
`Multi-tier Web-based survey using XHTML, servlets and JDBC.`

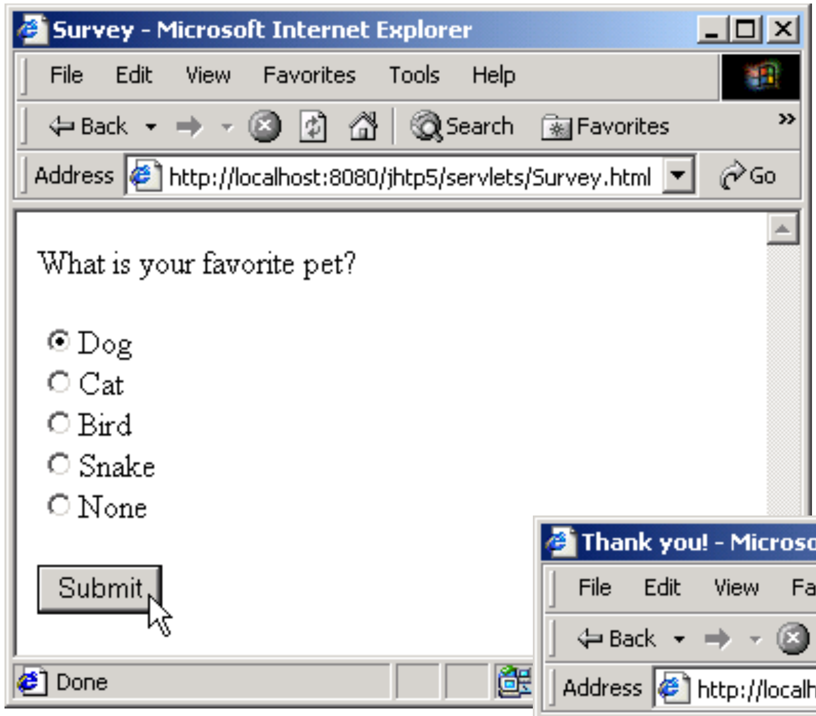
`Lines 130-136`

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

<!-- Fig. 21: Survey.html -->

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

<body>
<form method = "post" action = "/jhtp5/animalsurvey">

   <p>What is your favorite pet?</p>

```

Survey.html document that allows users to submit survey responses to SurveyServlet.

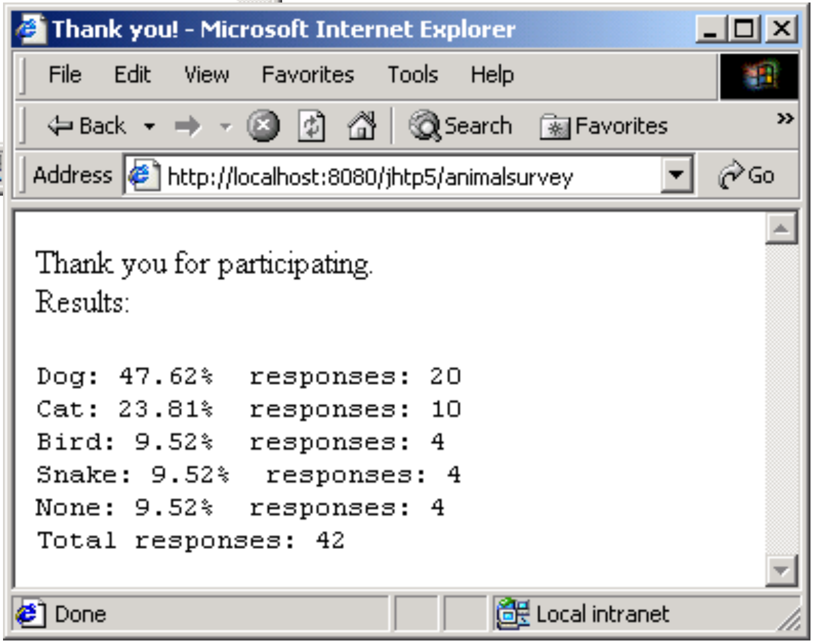
```
   <p>
      <input type = "radio" name = "animal"
         value = "1" />Dog<br />
      <input type = "radio" name = "animal"
         value = "2" />Cat<br />
      <input type = "radio" name = "animal"
         value = "3" />Bird<br />
      <input type = "radio" name = "animal"
         value = "4" />Snake<br />
      <input type = "radio" name = "animal"
         value = "5" checked = "checked" />None
   </p>

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

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

Survey.html document that allows users to submit survey responses to SurveyServlet.


Survey - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites
Address http://localhost:8080/jhtp5/servlets/Survey.html
Go

What is your favorite pet?

[✓ Dog]
Cat
Bird
Snake
None


Thank you! - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites
Address http://localhost:8080/jhtp5/animalsurvey
Go

Thank you for participating.
Results:

```
Dog: 47.62%  responses: 20
Cat: 23.81%  responses: 10
Bird: 9.52%  responses: 4
Snake: 9.52%  responses: 4
None: 9.52%  responses: 4
Total responses: 42
```

Survey.html document that allows users to submit survey responses to SurveyServlet.

Exercice

- Reprenez l'exemple de connexion JDBC avec une servlet au lieu de swing

Deploying a Web Application

- Web applications
  - JSPs, servlets and their supporting files
- Deploying a Web application
  - Directory structure
    - Context root
  - Web application archive file (WAR file)
  - Deployment descriptor
    - **`web.xml (JEE5)`** $\rightarrow$ **`annotations (JEE6)`**

Deploying a Web Application (Cont.)

Directory	Description
context root	This is the root directory for the Web application. All the JSPs, HTML documents, servlets and supporting files such as images and class files reside in this directory or its subdirectories. The name of this directory is specified by the Web application creator. To provide structure in a Web application, subdirectories can be placed in the context root. For example, if your application uses many images, you might place an images subdirectory in this directory. The examples of this chapter use `jhtp5` as the context root.
`WEB-INF`	This directory contains the Web application *deployment descriptor* (*`web.xml`*).
`WEB-INF/classes`	This directory contains the servlet class files and other supporting class files used in a Web application. If the classes are part of a package, the complete package directory structure would begin here.
`WEB-INF/lib`	This directory contains Java archive (JAR) files. The JAR files can contain servlet class files and other supporting class files used in a Web application.

Fig. 24.8 Web application standard directories.

```
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 descri
8     <display-name>
9        Java How to Program JSP
10       and Servlet Chapter Examples
11    </display-name>
12
13    <description>
14       This is the Web appli
15       demonstrate our JSP ar
16    </description>
17
18    <!-- Servlet definitions -->
19    <servlet>
20       <servlet-name>welcome1</servlet-name>
21
22       <description>
23          A simple servlet th                 quest.
24       </description>
25
```

Element `web-app` defines the configuration of each servlet in the Web application and the servlet mapping for each servlet.

Element `display-name` specifies a name that can be displayed to the administrator of the server on which the Web application is installed.

Element `description` specifies a description of the Web application that might be displayed to the administrator of the server.

Eleme t.

Element `servlet-name` is the name for the servlet.

Element `description` specifies a description for this particular servlet.

web.xml

Lines 5-37

Lines 8-11

Lines 13-16

Lines 19-29

Line 20

Lines 22-24

26-28

```
26         <servlet-class>
27            WelcomeServlet
28         </servlet-class>
29      </servlet>
30
31      <!-- Servlet mappings -->
32      <servlet-mapping>
33         <servlet-name>welcome1</servlet-name>
34         <url-pattern>/welcome1</url-pattern>
35      </servlet-mapping>
36
37   </web-app>
```

Element `servlet-class` specifies compiled servlet' s fully qualified class name.

Element `servlet-mapping` specifies `servlet-name` and `url-pattern` elements.

web.xml

es 26-28

es 32-35

Deploying a Web Application (Cont.)

- Invoke **`WelcomeServlet`** example
  - /jhtp5/welcome1
    - **`/jhtp5`** specifies the context root
    - **`/welcome1`** specifies the URL pattern
- URL pattern formats
  - Exact match
    - /jhtp5/welcome1
  - Path mappings
    - /jhtp5/example/*
  - Extension mappings
    - *.jsp
  - Default servlet
    - /

Deploying a Web Application (Cont.)

`WelcomeServlet` Web application directory and file structure
```
jhtp5
   servlets
      WelcomeServlet.html
   WEB-INF
      web.xml
      classes
         *WelcomeServlet.class*
```

Fig. 24.10 Web application directory and file structure for `WelcomeServlet`.

8 Internet and World Wide Web Resources

- Servlet resources
  - java.sun.com/products/servlet/index.html
  - www.servlets.com
  - www.servletsource.com
  - www.servletforum.com
  - www.coolservlets.com