Servlet Collaboration

Servlet Collaboration

The exchange of information among servlets of a particular Java web application is known as **Servlet Collaboration**. This enables passing/sharing information from one servlet to the other through method invocations.

There are principle ways provided by Java to achieve Servlet Collaboration.

1. javax.servlet.RequestDispatcher
2. javax.servlet.http.HttpServletResponse

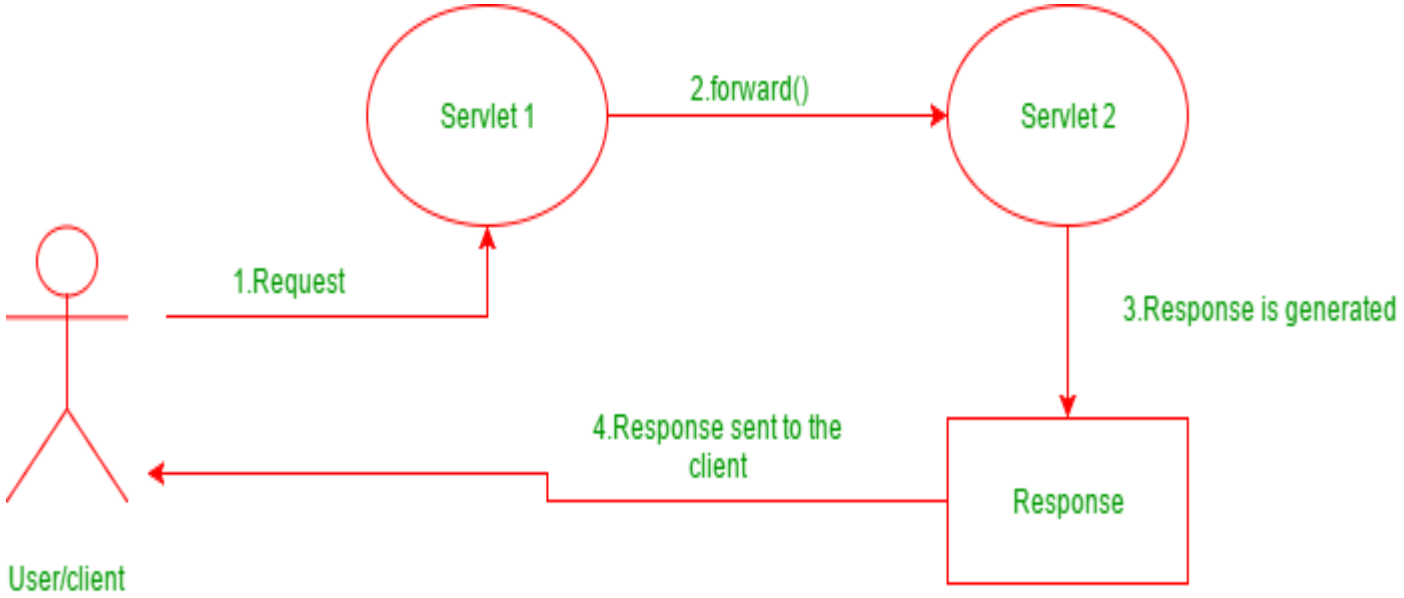
These two interfaces include the methods responsible for achieving the objective of sharing information between servlets.

Using RequestDispatcher Interface

The RequestDispatcher interface provides the option of dispatching the client's request to another web resource, which could be an HTML page, another servlet, JSP etc. It provides the following two methods:

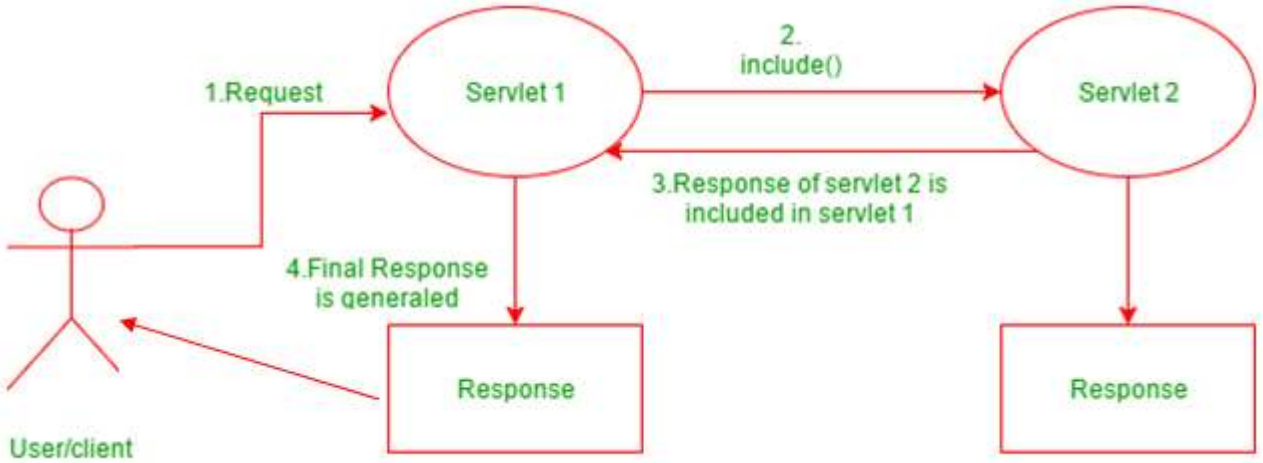
- **public void forward(ServletRequest request, ServletResponse response)throws ServletException, IOException**

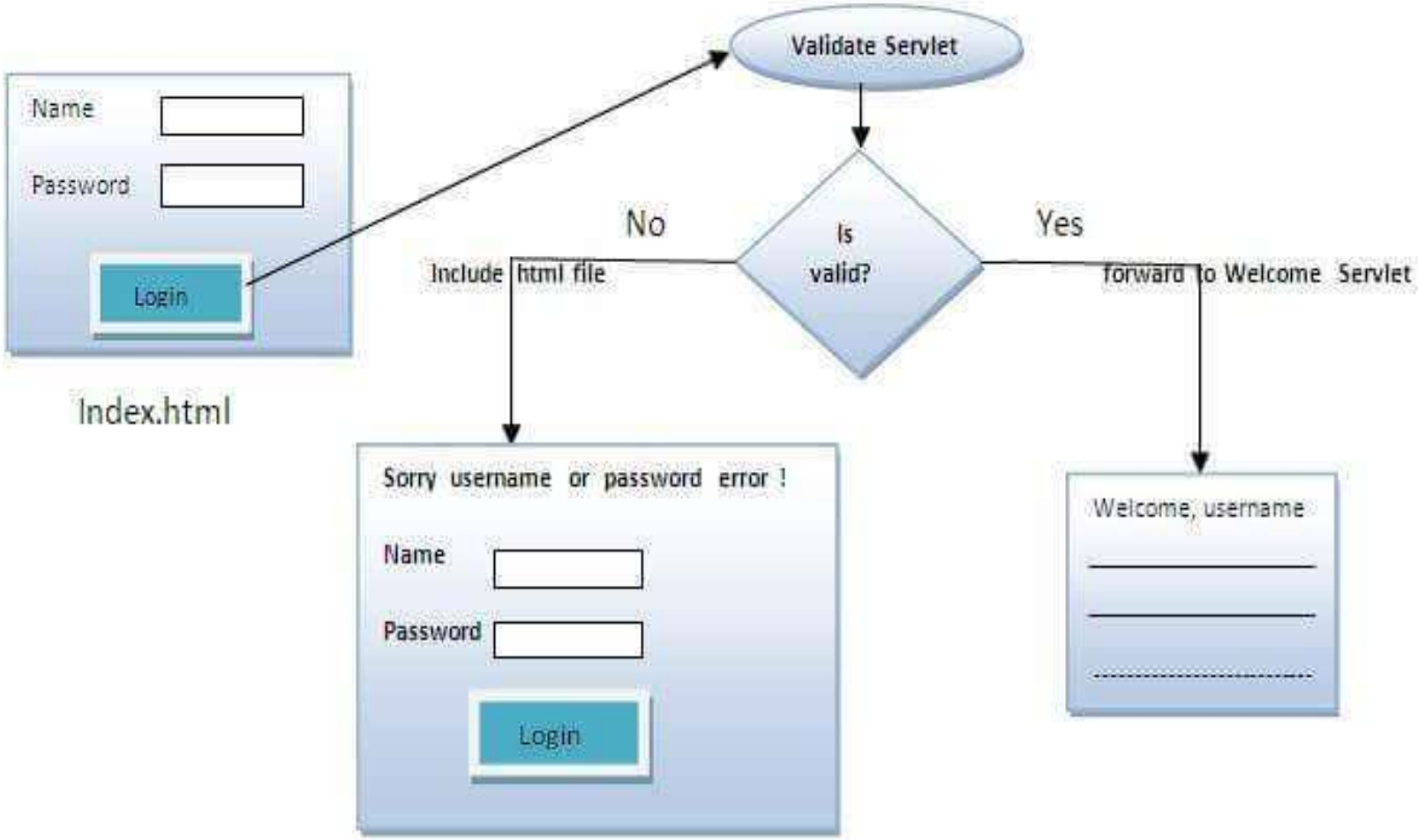
The forward() method is used to transfer the client request to another resource (HTML file, servlet, jsp etc). When this method is called, the control is transferred to the next resource called.



- **public void include(ServletRequest request, ServletResponse response)throws ServletException, IOException**

  The include() method is used to include the contents of the calling resource into the called one. When this method is called, the control still remains with the calling resource. It simply includes the processed output of the calling resource into the called one.




Validate Servlet
Name
Password
Login
Index.html
No
Is
valid?
Yes
Include html file
forward to Welcome Servlet
Sorry username or password error !
Name
Password
Login
Welcome, username

Index.html

```
<!DOCTYPE html>
<html>
<head>
<title>Login</title>
</head>
<body>
<form action="login" method="post">
Name:<input type="text" name="userName"/><br/>
Password:<input type="password" name="userPass"/><br/>
<input type="submit" value="login"/>
</form>
</body>
</html>
```

Login.java:

```
// First java servlet that calls another resource
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;

public class Login extends HttpServlet {

   public void doPost(HttpServletRequest req,
                      HttpServletResponse res)
throws ServletException, IOException
   {
      // The method to receive client requests
      // which are sent using 'post'

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

      // fetches username
      String n = req.getParameter("userName");

      // fetches password
      String p = req.getParameter("userPass");

      if(p.equals("Vasuda"))
         {
            RequestDispatcher rd =
        req.getRequestDispatcher("servlet2");
            // Getting RequestDispatcher object
            // for collaborating with servlet2


            // forwarding the request to
        servlet2
            rd.forward(req, res);
      }
      else{
            out.print("Password mismatch");
            RequestDispatcher rd =
        req.getRequestDispatcher("/index.htm
        l");
            rd.include(req, res);


         }
      }

}
```

Welcome.java:

```
// Called servlet in case password matches
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;

public class Welcome extends HttpServlet {

   public void doPost(HttpServletRequest request,
                      HttpServletResponse response)
      throws ServletException, IOException
   {

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

      // fetches username
      String n = request.getParameter("userName");

      // prints the message
      out.print("Welcome " + n);
   }
}
```

Web.xml:

```
<display-name>ServletCollaboration</display-name>
 <servlet>
  <servlet-name>Login</servlet-name>
  <servlet-class>Login</servlet-class>
 </servlet>
 <servlet>
  <servlet-name>WelcomeServlet</servlet-name>
  <servlet-class>Welcome</servlet-class>
 </servlet>

 <servlet-mapping>
  <servlet-name>Login</servlet-name>
  <url-pattern>/login</url-pattern>
 </servlet-mapping>
 <servlet-mapping>
  <servlet-name>WelcomeServlet</servlet-name>
  <url-pattern>/servlet2</url-pattern>
 </servlet-mapping>

 <welcome-file-list>
  <welcome-file>index.html</welcome-file>
  </welcome-file-list>
</web-app>
```

Using HttpServletResponse Interface

- The HttpServletResponse interface is entrusted with managing Http responses. To achieve servlet collaboration, it uses the following method:

  public void sendRedirect(String URL)throws IOException;

  This method is used redirect response to another resource, which may be a servlet, jsp or an html file. The argument accepted by it, is a URL which can be both, absolute and relative. It works on the client side and uses the browser's URL bar to make a request.

- **Index.html:**

```
<!DOCTYPE html>
<html>
<head>
<meta charset="ISO-8859-1">
<title>Insert title here</title>
</head>
<body>
<form action="search" method="GET">
<input type="text" name="name">
<input type="submit" value="search">
</form>
</body>
</html>
```

- **MySeracher.java:**

```
import java.io.IOException;
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("/search")
public class MySearcher extends HttpServlet {
   protected void doGet(HttpServletRequest  req,
                    HttpServletResponse res) throws ServletException, IOException
    {

      String name = req.getParameter("name");
      res.sendRedirect("https://www.google.co.in"  + name);
      // response redirected to google.com
   }
}
```

forward() vs sendRedirect()

forward() method	sendRedirect() method
The forward() method works at server side.	The sendRedirect() method works at client side.
It sends the same request and response objects to another servlet.	It always sends a new request.
It can work within the server only.	It can be used within and outside the server.
Example: request.getRequestDispacher("servlet2").forward(request,response);	Example: response.sendRedirect("servlet2");

ServletRequest Interface

An object of ServletRequest is used to provide the client request information to a servlet such as content type, content length, parameter names and values, header informations, attributes etc.

Content type parameters:

- text/html
- text/plain
- application/msword
- images/jpeg
- images/png
- images/gif
- audio/mp3
- video/mp4

Methods of ServletRequest interface

Method	Description
public String getParameter(String name)	is used to obtain the value of a parameter by name.
public String[] getParameterValues(String name)	returns an array of String containing all values of given parameter name. It is mainly used to obtain values of a Multi select list box.
java.util.Enumeration getParameterNames()	returns an enumeration of all of the request parameter names.
public int getContentLength()	Returns the size of the request entity data, or -1 if not known.
public String getCharacterEncoding()	Returns the character set encoding for the input of this request.
public String getContentType()	Returns the Internet Media Type of the request entity data, or null if not known.
public ServletInputStream getInputStream() throws IOException	Returns an input stream for reading binary data in the request body.
public abstract String getServerName()	Returns the host name of the server that received the request.
public int getServerPort()	Returns the port number on which this request was received.

ServletResponse Interface

- The servlet container is connected to the web server that receives Http Requests from client on a certain port. When client sends a request to web server, the servlet container creates HttpServletRequest and HttpServletResponse objects and passes them as an argument to the servlet service() method.
- The response object allows us to format and send the response back to the client.

Method of ServletResponse interface

- **String getCharacterEncoding():** It returns the name of the MIME charset used in body of the response sent to the client.
- **String getContentType():** It returns the response content type. e.g. text, html etc.
- **ServletOutputStream getOutputStream():** Returns a ServletOutputStream suitable for writing binary data in the response.
- **java.io.PrintWriter getWriter():** Returns the PrintWriter object.
- **void setCharacterEncoding(java.lang.String charset):** Set the MIME charset (character encoding) of the response.
- **void setContentLength(int len):** It sets the length of the response body.
- **void setContentType(java.lang.String type):** Sets the type of the response data.
- **void setBufferSize(int size):** Sets the buffer size.
- **int getBufferSize():** Returns the buffer size.
- **void flushBuffer():** Forces any content in the buffer to be written to the client.
- **boolean isCommitted():** Returns a boolean indicating if the response has been committed.
- **void reset():** Clears the data of the buffer along with the headers and status code.