**What is 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.

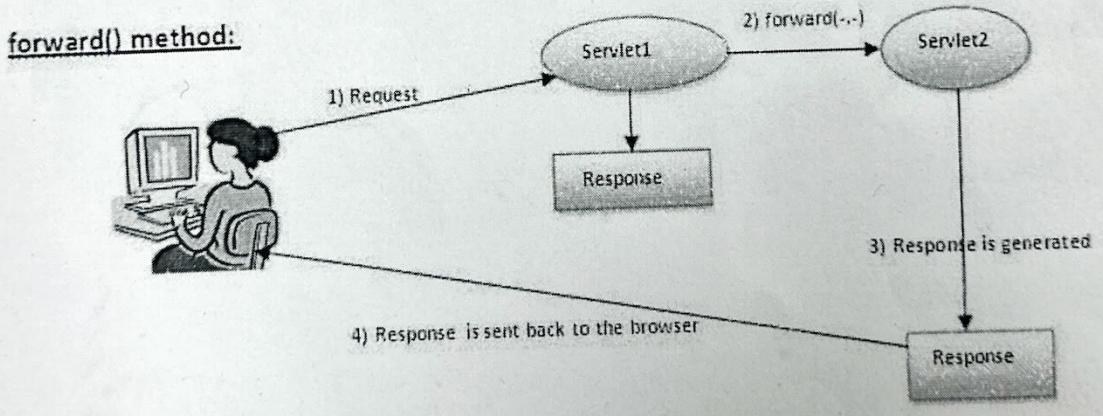
The RequestDispatcher interface provides the facility of dispatching the request to another resource it may be html, servlet or jsp. This interface can also be used to include the content of another resource also. It is one of the way of servlet collaboration.

There are two methods defined in the RequestDispatcher interface.

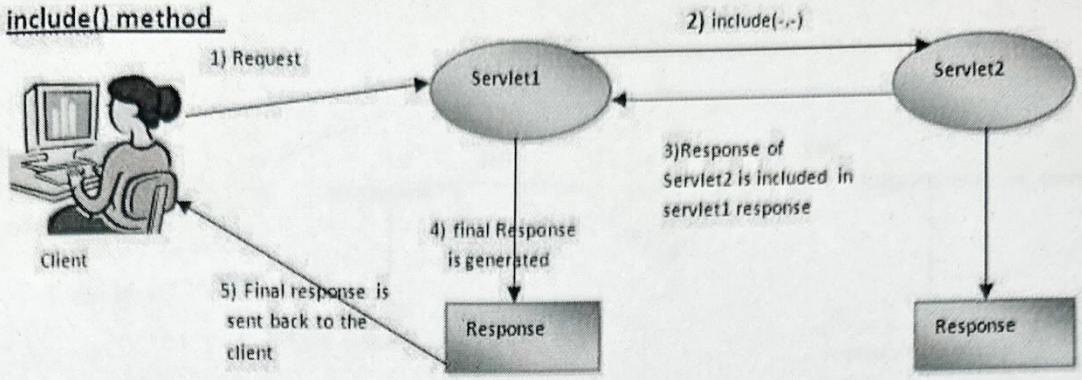
## Methods of RequestDispatcher interface

The RequestDispatcher interface provides two methods. They are:

1. **public void forward(ServletRequest request,ServletResponse response)throws ServletException,java.io.IOException:**Forwards a request from a servlet to another resource (servlet, JSP file, or HTML file) on the server.
2. **public void include(ServletRequest request,ServletResponse response)throws ServletException,java.io.IOException:**Includes the content of a resource (servlet, JSP page, or HTML file) in the response.




As you see in the above figure, response of second servlet is sent to the client. Response of the first servlet is not displayed to the user.

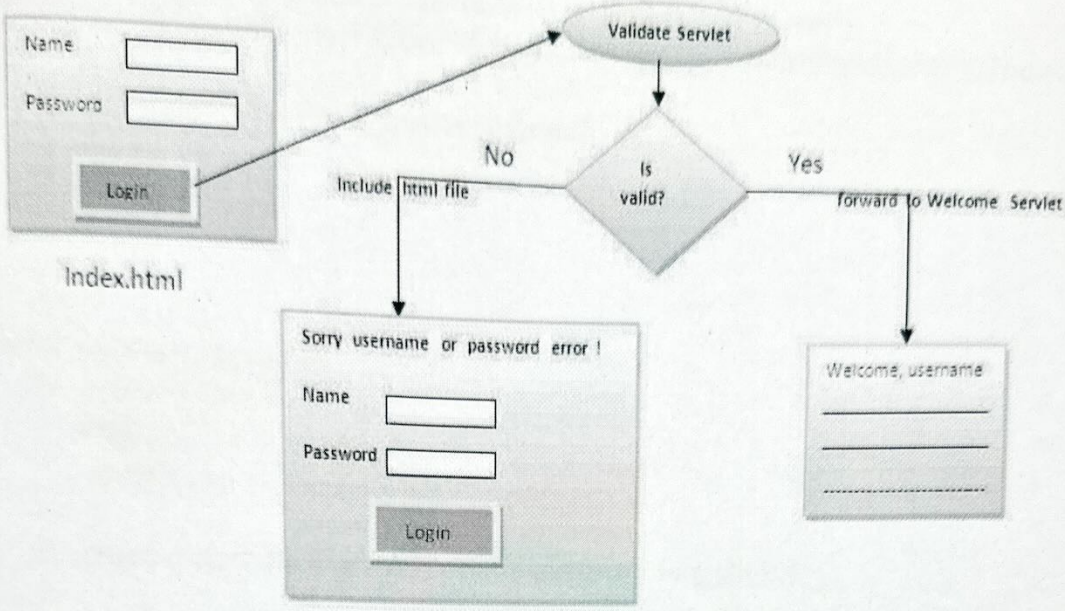



As you can see in the above figure, response of second servlet is included in the response of the first servlet that is being sent to the client.

## Example of RequestDispatcher interface

In this example, we are validating the password entered by the user. If password is servlet, it will forward the request to the WelcomeServlet, otherwise will show an error message: sorry username or password error!. In this program, we are cheking for hardcoded information. But you can check it to the database also that we will see in the development chapter. In this example, we have created following files:

- **index.html file:** for getting input from the user.
- **Login.java file:** a servlet class for processing the response. If password is servet, it will forward the request to the welcome servlet.
- **WelcomeServlet.java file:** a servlet class for displaying the welcome message.
- **web.xml file:** a deployment descriptor file that contains the information about the servlet.




**index.html**

```
1. <form action="servlet1" method="post">
2. Name:<input type="text" name="userName"/><br/>
3. Password:<input type="password" name="userPass"/><br/>
4. <input type="submit" value="login"/>
5. </form>
```

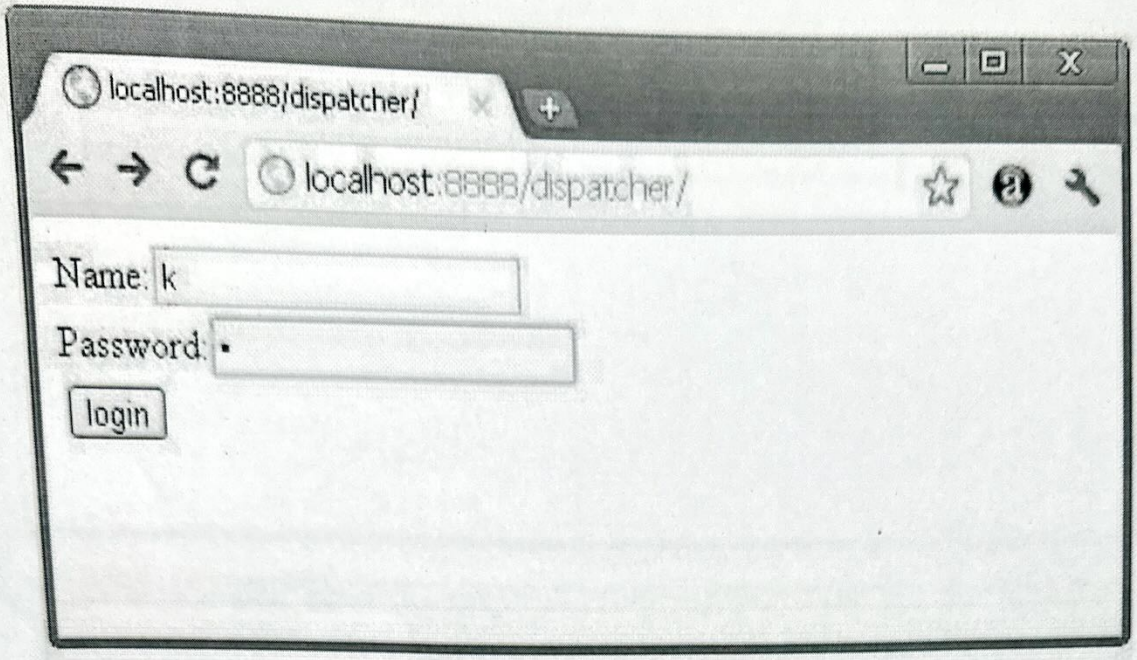
**Login.java**

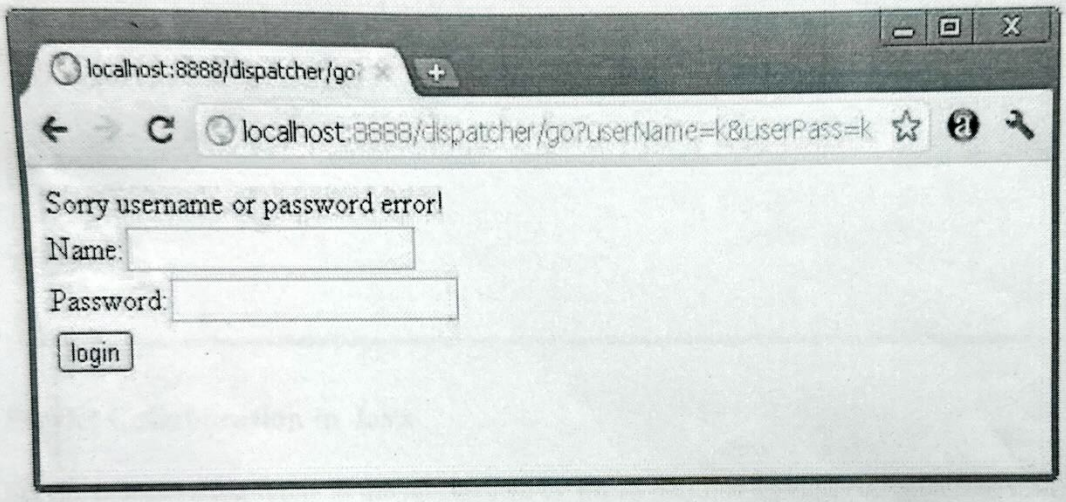
```
1. import java.io.*;
2. import javax.servlet.*;
3. import javax.servlet.http.*;
4.
5.
6. public class Login extends HttpServlet {
7.
8. public void doPost(HttpServletRequest request, HttpServletRespons
   e response)
9.       throws ServletException, IOException {
10.
11.    response.setContentType("text/html");
12.    PrintWriter out = response.getWriter();
13.
14.    String n=request.getParameter("userName");
15.    String p=request.getParameter("userPass");
16.
17.    if(p.equals("servlet"){
18.       RequestDispatcher rd=request.getRequestDispatcher("servlet2")
   ;
19.       rd.forward(request, response);
20.    }
21.    else{
```

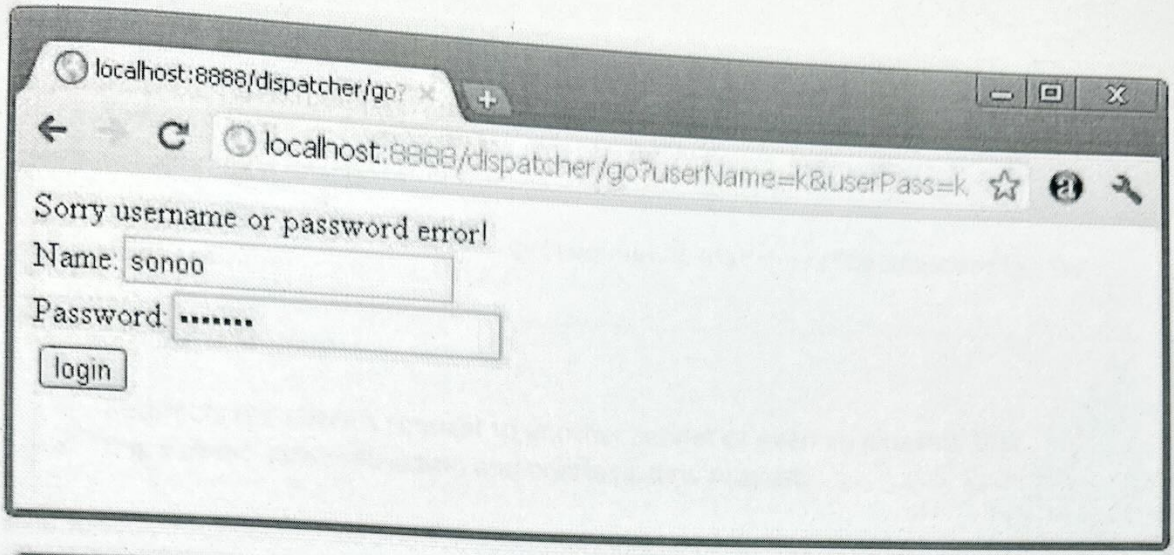
```
22.       out.print("Sorry UserName or Password Error!");
23.       RequestDispatcher rd=request.getRequestDispatcher("/index.ht
    ml");
24.       rd.include(request, response);
25.
26.       }
27.   }
28.
29.}
```

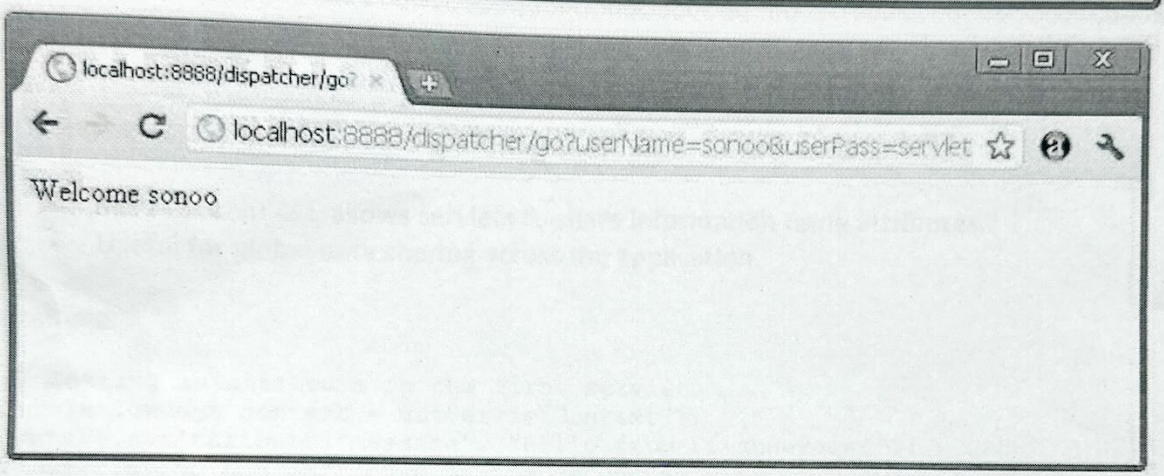
**WelcomeServlet.java**

```
1.  import java.io.*;
2.  import javax.servlet.*;
3.  import javax.servlet.http.*;
4.
5.  public class WelcomeServlet extends HttpServlet {
6.
7.    public void doPost(HttpServletRequest request, HttpServletRespo
    nse response)
8.        throws ServletException, IOException {
9.
10.   response.setContentType("text/html");
11.   PrintWriter out = response.getWriter();
12.
13.   String n=request.getParameter("userName");
14.   out.print("Welcome "+n);
15.   }
16.
17. }
```


localhost:8888/dispatcher/
localhost:8888/dispatcher/
Name: k
Password:
login


localhost:8888/dispatcher/go?
localhost:8888/dispatcher/go?userName=k&userPass=k
Sorry username or password error!
Name:
Password:
login




## Servlet Collaboration in Java

Servlet collaboration refers to the mechanism by which multiple servlets communicate with each other to process client requests. This is useful when a single servlet cannot handle the entire request and needs assistance from another servlet.

---

## Ways to Achieve Servlet Collaboration

Servlet collaboration can be done using the following techniques:

*1. RequestDispatcher Interface (Forward & Include)*

- `RequestDispatcher` allows one servlet to delegate the request to another servlet.
- This can be done in two ways:
  - **Forwarding a Request:** The control is transferred to another servlet, and the original servlet's execution stops.
  - **Including a Response:** The response of another servlet is included in the response of the calling servlet.

Example: Forwarding a Request

```
RequestDispatcher rd = request.getRequestDispatcher("SecondServlet");
rd.forward(request, response);
```

Example: Including a Response

```
RequestDispatcher rd = request.getRequestDispatcher("SecondServlet");
rd.include(request, response);
```

---

*2. sendRedirect() Method*

- **Redirects the client's request to another servlet or even an external URL.**
- **It is a client-side redirection and creates a new request.**

Example:

```
response.sendRedirect("SecondServlet");
```

---

*3. Using ServletContext*

- `ServletContext` **allows servlets to share information using attributes.**
- **Useful for global data sharing across the application.**

Example:

```
// Setting an attribute in the first servlet
ServletContext context = getServletContext();
context.setAttribute("message", "Hello from FirstServlet");
// Retrieving the attribute in the second servlet
ServletContext context = getServletContext();
String message = (String) context.getAttribute("message");
response.getWriter().println("Message: " + message);
```

---

*4. Using HttpSession*

- **Used when data needs to be shared across multiple requests from the same client.**
- **Suitable for storing user-related session data.**

Example:

```
// Storing data in session in the first servlet
HttpSession session = request.getSession();
session.setAttribute("username", "JohnDoe");
// Retrieving data from session in the second servlet
HttpSession session = request.getSession();
String user = (String) session.getAttribute("username");
response.getWriter().println("User: " + user);
```

---

**When to Use Each Approach?**