**Servlet Collaboration In Java Using RequestDispatcher and HttpServletResponse**

**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.

**What are the principle ways provided by Java to achieve Servlet Collaboration?**  
The servlet api provides two interfaces namely:

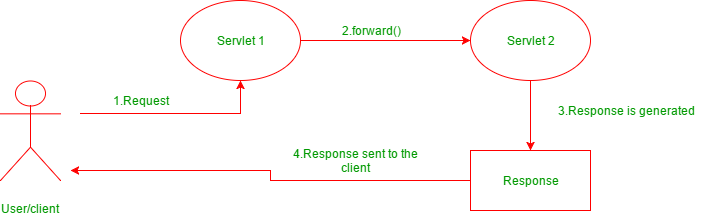
1. javax.servlet.RequestDispatcher
2. javax.servlet.http.HttpServletResponse

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

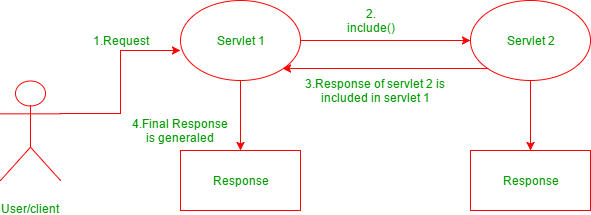
**Using RequestDispatcher Interface**

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, java.io.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. On the other hand, the include() method is used to include the content of the calling file into the called file. After calling this method, the control remains with the calling resource, but the processed output is included into the called resource.  
  The following diagram explains the way it works:



* **public void include(ServletRequest request, ServletResponse response)throws ServletException, java.io.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.  
  The following diagram explains how it works:



* **Example of using RequestDispatcher for Servlet Collaboration**  
  The following example explains how to use RequestDispatcher interface to achieve Servlet Collaboration:  
  **index.html**
* html

|  |
| --- |
| <**html**>  <**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**
* 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 = response.getWriter();            // fetches username          String n = request.getParameter("userName");            // fetches password          String p = request.getParameter("userPass");  **if**(p.equals("Thanos"){              RequestDispatcher rd = request.getRequestDispatcher("servlet2");              // Getting RequestDispatcher object              // for collaborating with servlet2                // forwarding the request to servlet2              rd.forward(request, response);      }  **else**{              out.print("Password mismatch");              RequestDispatcher rd = request.getRequestDispatcher("/index.html");            rd.include(request, response);            }      }    } |

* **Welcome.java**
* Java

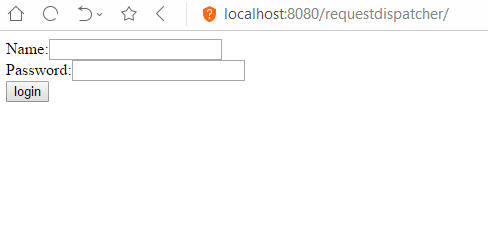
|  |
| --- |
| // Called servlet in case password matches  **import** java.io.\*;  **import** javax.servlet.\*;  **import** javax.servlet.http.\*;    **public** **class** Welcome **extends** HttpServlet {    **public** **void** doPost(HttpServletRequest request,                         HttpServletResponse 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**
* html

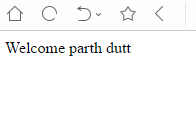
|  |
| --- |
| <**web-app**>   <**servlet**>      <**servlet-name**>Login</**servlet-name**>      <**servlet-class**>Login</**servlet-class**>    </**servlet**>    <**servlet**>      <**servlet-name**>WelcomeServlet</**servlet-name**>      <**servlet-class**>Welcome</**servlet-class**>    </**servlet**>        <**servlet-mapping**>      <**servlet-name**>Login</**servlet-name**>      <**url-pattern**>/servlet1</**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**> |

**Output**:

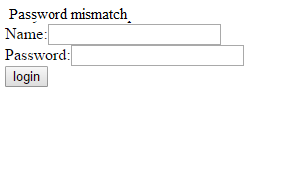
* **index.html**



* **If password matches:**



* **If password doesn’t match:**



**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.

**Example of using sendRedirect() for redirection**

* The following example of a web application created using servlet takes the text written in the text field in the webpage, and directs it to the servlet. The servlet then redirects it to google, which then produces search results based on the text written.  
  **index.html**
* html

|  |
| --- |
| <**html**>  <**head**>  <**body**>  <**form** action="search" method="GET">  <**input** type="text" name="name">  <**input** type="submit" value="search">  </**form**>  </**body**>  </**html**> |

* Java

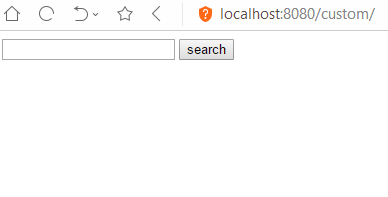
|  |
| --- |
| // Servlet class to redirect the text keyword  // in the 'name' field to google.com  // using sendRedirect()  **import** java.io.IOException;  **import** javax.servlet.ServletException;  **import** javax.servlet.http.HttpServlet;  **import** javax.servlet.http.HttpServletRequest;  **import** javax.servlet.http.HttpServletResponse;    **public** **class** MySearcher **extends** HttpServlet {  **protected** **void** doGet(HttpServletRequest                               request,                           HttpServletResponse response)  **throws** ServletException, IOException      {            String name = request.getParameter("name");          response.sendRedirect("<https://www.google.co.in/#q=>" + name);          // response redirected to google.com      }  } |

* **web.xml**
* html

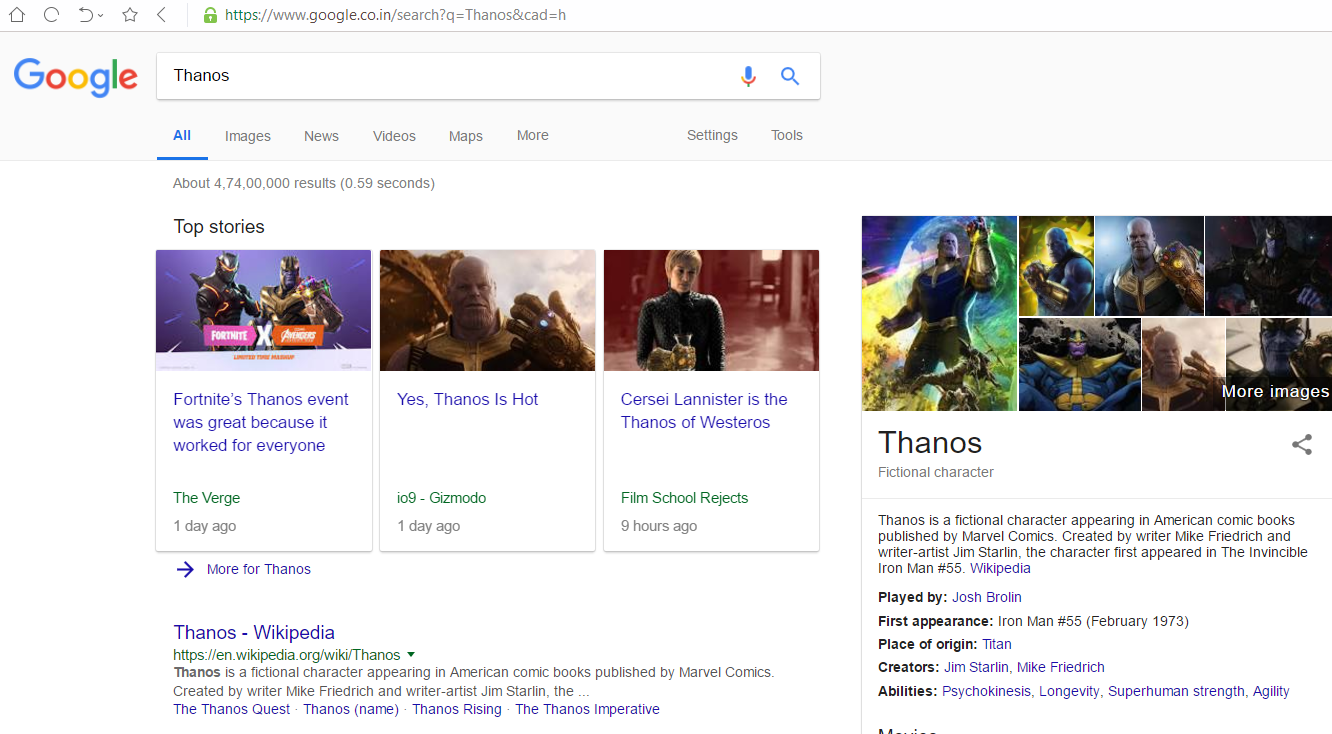
|  |
| --- |
| <**web-app**>  <**servlet**>  <**servlet-name**>MySearcher</**servlet-name**>  <**servlet-class**>MySearcher</**servlet-class**>  </**servlet**>    <**servlet-mapping**>  <**servlet-name**>MySearcher</**servlet-name**>  <**url-pattern**>/search</**url-pattern**>  </**servlet-mapping**>    <**welcome-file-list**>     <**welcome-file**>index.html</**welcome-file**>    </**welcome-file-list**>  </**web-app**> |

**Output:**

* **index.html**



* **Search result**



**What is the difference between forward() method of RequestDiispatcher and sendRedirect() of HttpServletResponse?**

* Although the two methods appear to do the same thing, there are still differences between the two, which are as follows:

|  |  |
| --- | --- |
| **forward()** | **sendRedirect()** |
| It works on the server side | It works on the client side |
| It sends the same request and response objects to another resource. | It always send a new request |
| It works only within the server. | It can be used within and outside the server. |