RequestDispatcher in Servlet

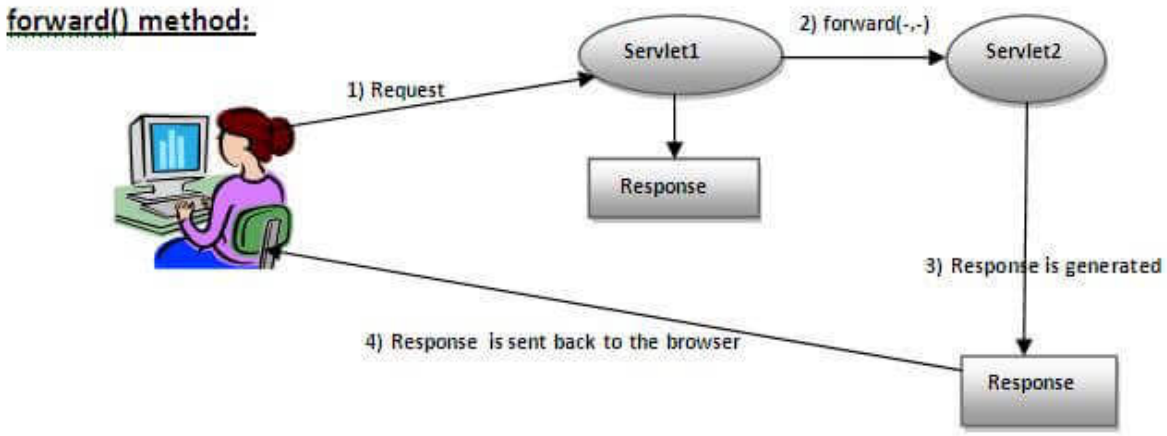
The RequestDispatcher interface provides the facility of dispatching the request to another resource it may be html, servlet or jsp. This interface can also be used to include the content of another resource also. It is one of the way of servlet collaboration.

There are two methods defined in the RequestDispatcher interface.

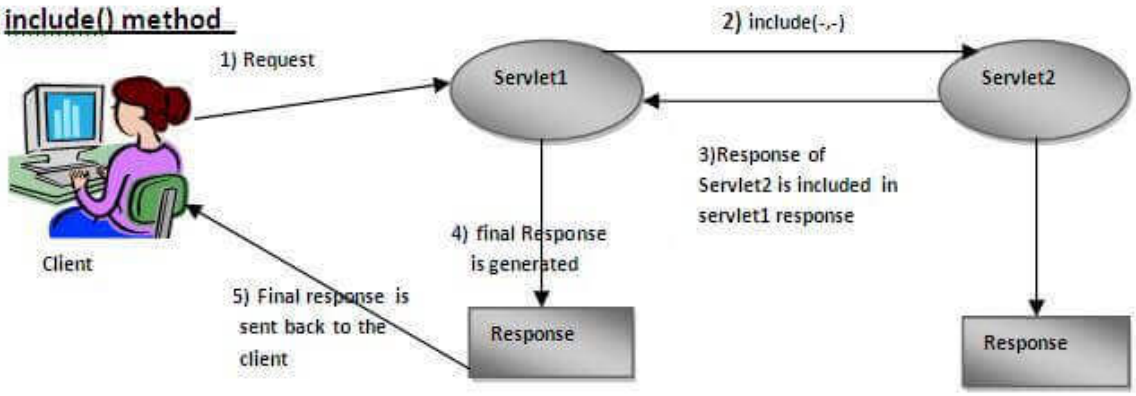
Methods of RequestDispatcher interface

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.

How to get the object of RequestDispatcher

The getRequestDispatcher() method of ServletRequest interface returns the object of RequestDispatcher. Syntax:

Syntax of getRequestDispatcher method

```
1. public RequestDispatcher getRequestDispatcher(String resource);
```

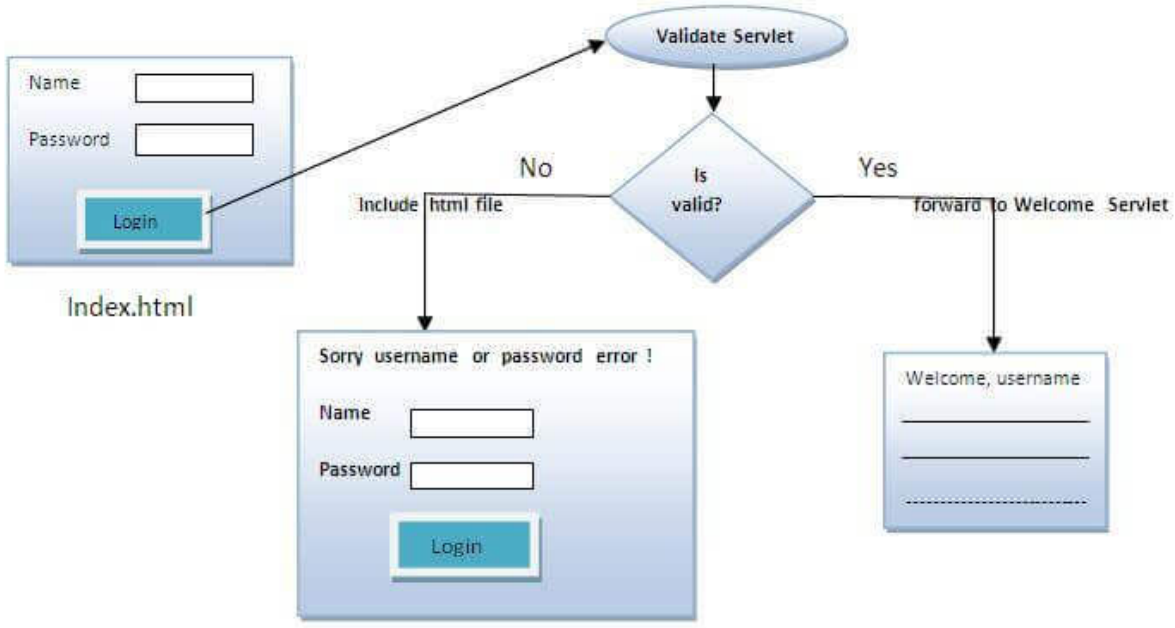
Example of using getRequestDispatcher method

```
1. RequestDispatcher rd=request.getRequestDispatcher("servlet2");
2. //servlet2 is the url-pattern of the second servlet
3.   rd.forward(request, response);//method may be include or forward
```

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.



Snippet of Login.java (First Servlet)

String u=request.getParameter(“user”);

String p=request.getParameter(“pass”);

```
String x=”ADMIN”,y=”123”;

if(u.equals(x) && p.equals(y))

{

        out.println(“Password matched”);

        RequestDispatcher rd=request.getRequestDispatcher(“Home”);

        rd.include(request,response);

}
```

_____

Home.java

Out.println(“Welcome”);

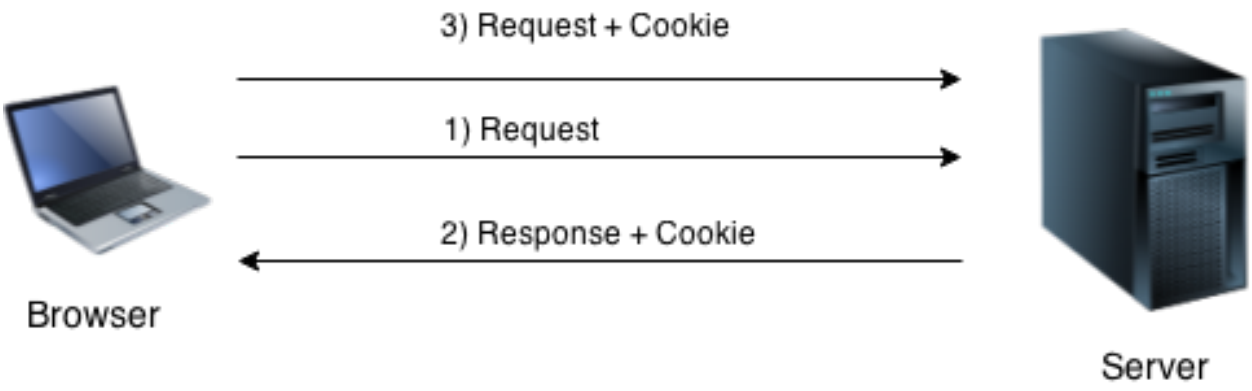
Cookies in Servlet

A **cookie** is a small piece of information that is persisted between the multiple client requests.

A cookie has a name, a single value, and optional attributes such as a comment, path and domain qualifiers, a maximum age, and a version number.

How Cookie works

By default, each request is considered as a new request. In cookies technique, we add cookie with response from the servlet. So cookie is stored in the cache of the browser. After that if request is sent by the user, cookie is added with request by default. Thus, we recognize the user as the old user.



Types of Cookie

There are 2 types of cookies in servlets.

1. Non-persistent cookie
2. Persistent cookie

Non-persistent cookie

It is **valid for single session** only. It is removed each time when user closes the browser.

Persistent cookie

It is **valid for multiple session** . It is not removed each time when user closes the browser. It is removed only if user logout or signout.

Advantage of Cookies

1. Simplest technique of maintaining the state.
2. Cookies are maintained at client side.

Disadvantage of Cookies

1. It will not work if cookie is disabled from the browser.
2. Only textual information can be set in Cookie object.

Cookie class

javax.servlet.http.Cookie class provides the functionality of using cookies. It provides a lot of useful methods for cookies.

Constructor of Cookie class

Constructor	Description
Cookie()	constructs a cookie.
Cookie(String name, String value)	constructs a cookie with a specified name and value.

Useful Methods of Cookie class

There are given some commonly used methods of the Cookie class.

Method	Description
public void setMaxAge(int expiry)	Sets the maximum age of the cookie in seconds.
public String getName()	Returns the name of the cookie. The name cannot be changed after creation.
public String getValue()	Returns the value of the cookie.
public void setName(String name)	changes the name of the cookie.
public void setValue(String value)	changes the value of the cookie.

Other methods required for using Cookies

For adding cookie or getting the value from the cookie, we need some methods provided by other interfaces. They are:

1. **public void addCookie(Cookie ck):**method of HttpServletResponse interface is used to add cookie in response object.
2. **public Cookie[] getCookies():**method of HttpServletRequest interface is used to return all the cookies from the browser.

How to create Cookie?

Let's see the simple code to create cookie.

```
1. Cookie ck=new Cookie("user","sonoo jaiswal");//creating cookie object
2. response.addCookie(ck);//adding cookie in the response
```

How to delete Cookie?

Let's see the simple code to delete cookie. It is mainly used to logout or signout the user.

```
1. Cookie ck=new Cookie("user","");//deleting value of cookie
2. ck.setMaxAge(0);//changing the maximum age to 0 seconds
3. response.addCookie(ck);//adding cookie in the response
```

How to get Cookies?

Let's see the simple code to get all the cookies.

```
1. Cookie ck[]=request.getCookies();
2. for(int i=0;i<ck.length;i++){
3.  out.print("<br>"+ck[i].getName()+" "+ck[i].getValue());//printing name and value of cooki
   e
4. }
```

Simple example of Servlet Cookies

In this example, we are storing the name of the user in the cookie object and accessing it in another servlet. As we know well that session corresponds to the particular user. So if you access it from too many browsers with different values, you will get the different value.

//Servlet1.java

String s=request.getParameter("t1");

```
Cookie c1= new Cookie("key",s);
response.addCookie(c1);
response.sendRedirect("Servlet2");
```

```
//Servlet2.java

Cookie[] x=request.getCookies();
for(int i=0; i<x.length;i++)
{
        String name=x[i].getName();
        String value=x[i].getValue();
        out.println("Name: "+name);
        out.println("Value: "+value);
}
```

The HttpSession Object

Apart from the above mentioned three ways, servlet provides HttpSession Interface which provides a way to identify a user across more than one page request or visit to a Web site and to store information about that user.

The servlet container uses this interface to create a session between an HTTP client and an HTTP server. The session persists for a specified time period, across more than one connection or page request from the user.

You would get HttpSession object by calling the public method **getSession()** of HttpServletRequest, as below –

```
HttpSession session = request.getSession();
```

You need to call *request.getSession()* before you send any document content to the client. Here is a summary of the important methods available through HttpSession object –

Sr.No.	Method & Description
1	**public Object getAttribute(String name)** This method returns the object bound with the specified name in this session, or null if no object is bound under the name.
2	**public Enumeration getAttributeNames()** This method returns an Enumeration of String objects containing the names of all the objects bound to this session.
3	**public long getCreationTime()** This method returns the time when this session was created, measured in milliseconds since midnight January 1, 1970 GMT.
4	**public String getId()** This method returns a string containing the unique identifier assigned to this session.
5	**public long getLastAccessedTime()** This method returns the last accessed time of the session, in the format of milliseconds since midnight January 1, 1970 GMT
6	**public int getMaxInactiveInterval()** This method returns the maximum time interval (seconds), that the servlet container will keep the session open between client accesses.
7	**public void invalidate()**

	This method invalidates this session and unbinds any objects bound to it.
8	**public boolean isNew()** This method returns true if the client does not yet know about the session or if the client chooses not to join the session.
9	**public void removeAttribute(String name)** This method removes the object bound with the specified name from this session.
10	**public void setAttribute(String name, Object value)** This method binds an object to this session, using the name specified.
11	**public void setMaxInactiveInterval(int interval)** This method specifies the time, in seconds, between client requests before the servlet container will invalidate this session.

```
// Create a session object if it is already not  created.
    HttpSession session = request.getSession(true);

    // Get session creation time.
    Date createTime = new Date(session.getCreationTime());

    // Get last access time of this web page.
    Date lastAccessTime = new Date(session.getLastAccessedTime());

    String title = "Welcome Back to my website";
    Integer visitCount = new Integer(0);
    String visitCountKey = new String("visitCount");
    String userIDKey = new String("userID");
    String userID = new String("ABCD");

    // Check if this is new comer on your web page.
    if (session.isNew()) {
      title = "Welcome to my website";
      session.setAttribute(userIDKey, userID);
    } else {
```

```
      visitCount = (Integer)session.getAttribute(visitCountKey);
      visitCount = visitCount + 1;
      userID = (String)session.getAttribute(userIDKey);
    }
    session.setAttribute(visitCountKey,  visitCount);
// Set response content type
    response.setContentType("text/html");
    PrintWriter out = response.getWriter();

    String docType =
      "<!doctype html public \"-//w3c//dtd html 4.0 " +
      "transitional//en\">\n";

    out.println(docType +
      "<html>\n" +
        "<head><title>" + title + "</title></head>\n" +

        "<body bgcolor = \"#f0f0f0\">\n" +
          "<h1 align = \"center\">" + title + "</h1>\n" +
          "<h2 align = \"center\">Session Infomation</h2>\n" +
          "<table border = \"1\" align = \"center\">\n" +

            "<tr bgcolor = \"#949494\">\n" +
              "  <th>Session info</th><th>value</th>
            </tr>\n" +

            "<tr>\n" +
              "  <td>id</td>\n" +
              "  <td>" + session.getId() + "</td>
            </tr>\n" +

            "<tr>\n" +
              "  <td>Creation Time</td>\n" +
              "  <td>" + createTime + "  </td>
            </tr>\n" +

            "<tr>\n" +
              "  <td>Time of Last Access</td>\n" +
              "  <td>" + lastAccessTime + "  </td>
            </tr>\n" +

            "<tr>\n" +
              "  <td>User ID</td>\n" +
              "  <td>" + userID + "  </td>
            </tr>\n" +
```

```
            "<tr>\n" +
              "  <td>Number of visits</td>\n" +
              "  <td>" + visitCount + "</td>
            </tr>\n" +
          "</table>\n" +
        "</body>
      </html>"
    );
```