Name of the School: School of Computer Science and Engineering

Course Code: R1UC602C Course Name: Web Technology

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Subject Name: Web Technology

Day: 28

Topics Covered: Java Servlet- Navigation Management

Prerequisites, Objectives and Outcomes

Prerequisite of topic: Basic concepts related to web programming

Objective: To make students aware about the Navigation Management using Servlet.

Outcome : 1. Students will be able to use Nevigation Management techniques.

2. Students will be able to apply to shift within and out of the application

3. Students will be able to implement in practical applications.

Page navigation in Servlet

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.

Types

- 1. RequestDispature
- 2. SendRedirect

Introduction to Request Dispatcher

- **RequestDispatcher** is an interface, implementation of which defines an object which can dispatch request to any resources(such as HTML, Image, JSP, Servlet) on the server.

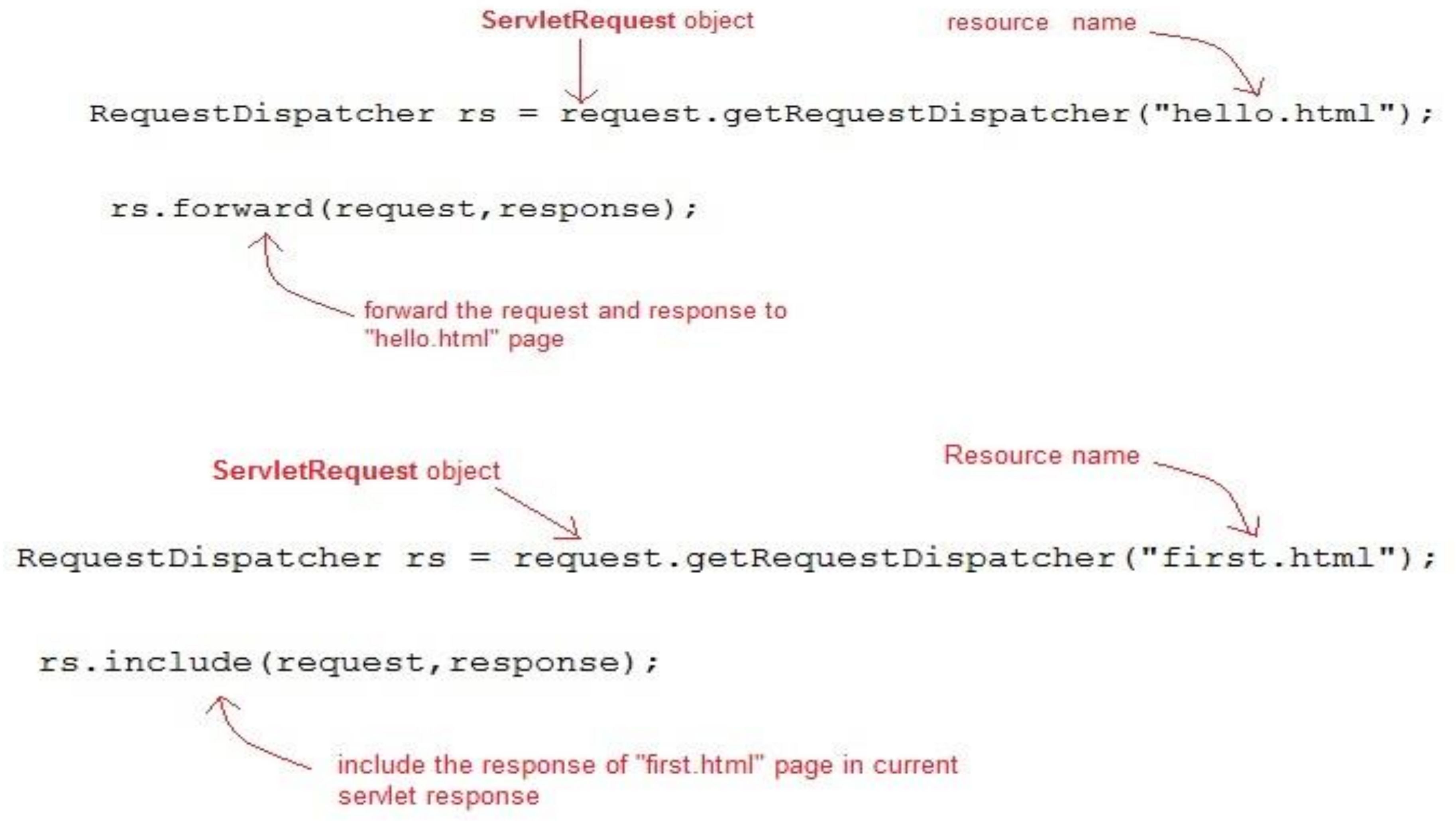
- This interface can also be used to include the content of another resource also. It is one of the way of servlet collaboration.

Methods of RequestDispatcher

RequestDispatcher interface provides two important methods

Methods	**Description**
void forward(ServletRequest request, ServletResponse response)	forwards a request from a servlet to another resource (servlet, JSP file, or HTML file) on the server
void include(ServletRequest request, ServletResponse response)	includes the content of a resource (servlet, JSP page, HTML file) in the response

How to get an Object of RequestDispatcher


ServletRequest object
resource name
forward the request and response to "hello.html" page
ServletRequest object
Resource name
include the response of "first.html" page in current servlet response

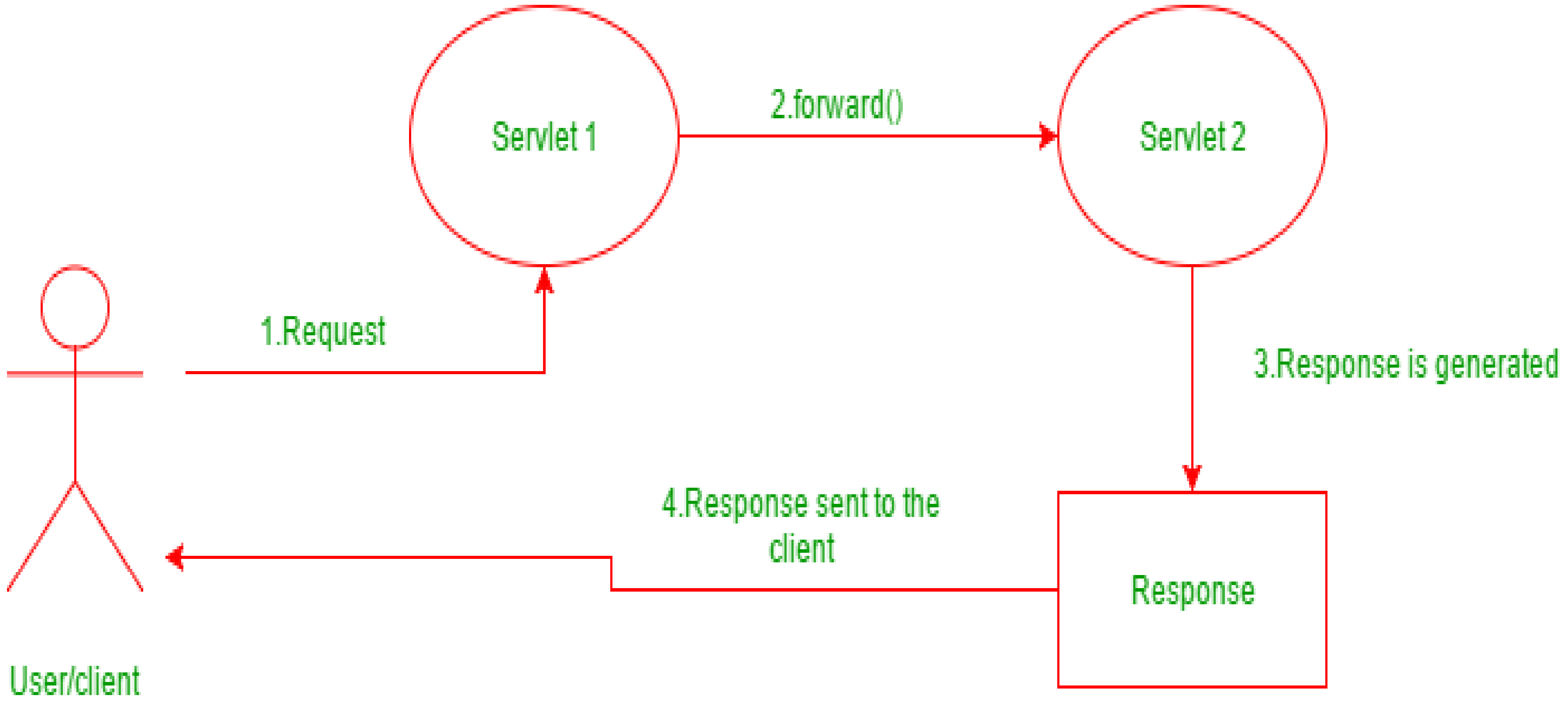
```
RequestDispatcher rs = request.getRequestDispatcher("hello.html");

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

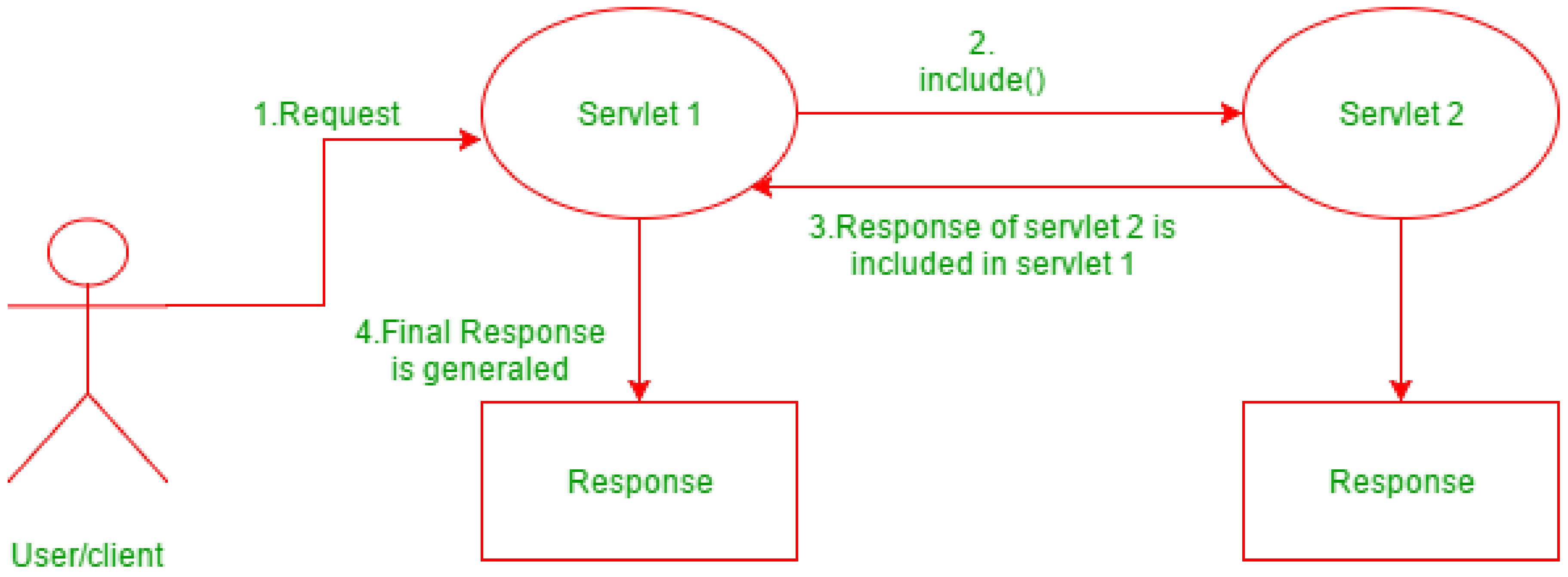
```
RequestDispatcher rs = request.getRequestDispatcher("first.html");

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

forward


Servlet 1
2.forward()
Servlet 2
1.Request
3.Response is generated
4.Response sent to the client
Response
User/client

include


1.Request
Servlet 1
2.
include()
Servlet 2
3.Response of servlet 2 is
included in servlet 1
4.Final Response
is generaled
Response
Response
User/client

How to get the object of RequestDispatcher

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

- Syntax of getRequestDispatcher method

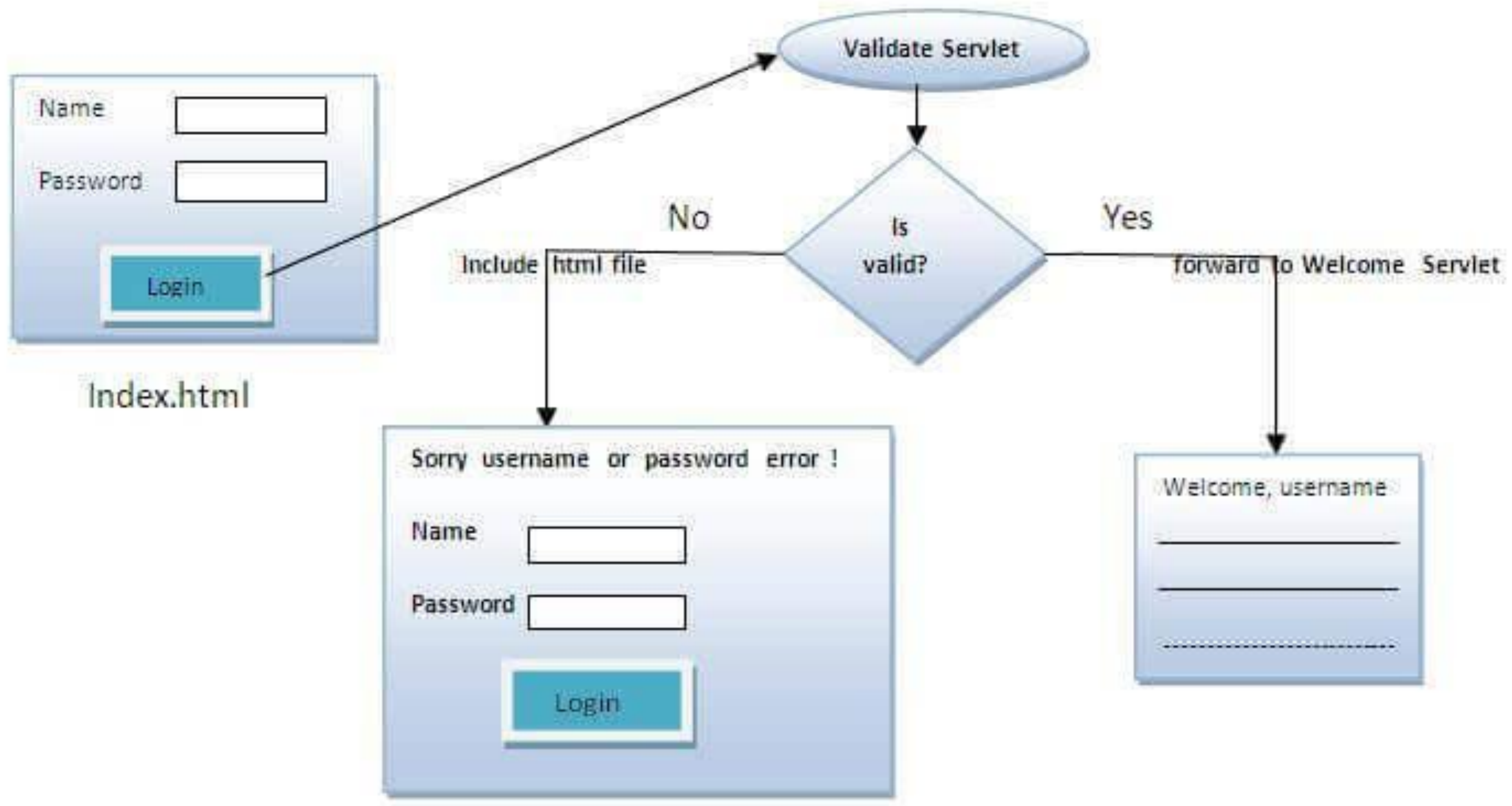
  public RequestDispatcher getRequestDispatcher(String resource);

- Example of using getRequestDispatcher method
- RequestDispatcher rd=request.getRequestDispatcher("servlet2");

- //servlet2 is the url-pattern of the second servlet
-

- rd.forward(request, response);//method may be include or forward

Flow: RequestDispatcher interface



```
•  <web-app>
•   <servlet>
•      <servlet-name>Login</servlet-name>
•      <servlet-class>Login</servlet-class>
•    </servlet>
•    <servlet>
•      <servlet-name>WelcomeServlet</servlet-name>
•      <servlet-class>WelcomeServlet</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>
```

Related 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.

RequestDispatcher program

- Index.html
- <form method="post" action="Validate">
- Name:<input type="text" name="user" /><br/>
- Password:<input type="password" name="pass" ><br/>
- <input type="submit" value="submit">
- </form>

Validate.java

```
• import java.io.*; import javax.servlet.*;import javax.servlet.http.*;
• public class Validate extends HttpServlet {
•    protected void doPost(HttpServletRequest request, HttpServletResponse response)
•     throws ServletException, IOException {
•       response.setContentType("text/html;charset=UTF-8");
•      PrintWriter out = response.getWriter();
•      try {
•          String name = request.getParameter("user");
•         String password = request.getParameter("pass");
•         if(password.equals(“abesit"))                  {
•            RequestDispatcher rd = request.getRequestDispatcher("Welcome");
•            rd.forward(request, response);
•         }          else            {
•          out.println("<font color='red'><b>You have entered incorrect password</b></font>");
•                 RequestDispatcher rd = request.getRequestDispatcher("index.html");
•                 rd.include(request, response);            }
•         }finally {
•             out.close();      }                              }          }
```

Welcome.java

```
•   import java.io.*;
•   import javax.servlet.*;
•   import javax.servlet.http.*;
•   public class Welcome extends HttpServlet {
•     protected void doPost(HttpServletRequest request, HttpServletResponse
    response)
•             throws ServletException, IOException {
•         response.setContentType("text/html;charset=UTF-8");
•         PrintWriter out = response.getWriter();
•         try {
•      out.println("<h2>Welcome user</h2>");
•         } finally {
•             out.close();
•         }   }
•   }
```

Introduction to sendRedirect() Method

- sendRedirect() method redirects the response to another resource. This method actually makes the client(browser) to create a new request to get to the resource. The client can see the new url in the browser.
- **sendRedirect()** accepts relative **URL**, so it can go for resources inside or outside the server.

sendRedirect() and Request Dispatcher

- The main difference between a **redirection** and a **request dispatching** is that, redirection makes the client(browser) create a new request to get to the resource, the user can see the new URL while request dispatch get the resource in same request and URL does not changes.
- Also, another very important difference is that, sendRedirect() works on **response** object while request dispatch work on **request** object.

- import java.io.*; import javax.servlet.*; import javax.servlet.http.*;
- public class MyServlet extends HttpServlet {
-   protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
-         response.setContentType("text/html;charset=UTF-8");
-         PrintWriter out = response.getWriter();
-         try {
-         response.sendRedirect("http:/ /www.abc.com");
-         }finally {
-             out.close();   }
-     } }

Difference between forward() vs include() method

- To understand the difference between these two methods, lets take an example: Suppose you have two pages X and Y. In page X you have an include tag, this means that the control will be in the page X till it encounters include tag, after that the control will be transferred to page Y. At the end of the processing of page Y, the control will return back to the page X starting just after the include tag and remain in X till the end.

  In this case the final response to the client will be send by page X.

- Now, we are taking the same example with forward. We have same pages X and Y. In page X, we have forward tag. In this case the control will be in page X till it encounters forward, after this the control will be transferred to page Y. The main difference here is that the control will not return back to X, it will be in page Y till the end of it.
  In this case the final response to the client will be send by page Y.