**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.

Method of RequestDispatcher

RequestDispatcher interface provides two important methods

* 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?**

getRequestDispatcher() method of ServletRequest returns the object of RequestDispatcher.

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

rs.forward(request,response);



Or

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

rs.include(request,response);



Example

In this example, we will show how RequestDispatcher is used to forward or include response of a resource in a servlet. Here we are using index.html to get username and password from the user, Validate servlet will validate the password entered by the user, if the user has entered "test" as password, then he will be forwarded to Welcome servlet else the index.html will be shown again to the user.

Files to be created

index.html will have form field to get user information

Validate.java will validate the data entered by the user

Welcome.java will be the welcome page

web.xml

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("test"))

{

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();

}

}

}

web.xml

<web-app>

<servlet>

<servlet-name>Validate</servlet-name>

<servlet-class>Validate</servlet-class>

</servlet>

<servlet>

<servlet-name>Welcome</servlet-name>

<servlet-class>Welcome</servlet-class>

</servlet>

<servlet-mapping>

<servlet-name>Validate</servlet-name>

<url-pattern>/Validate</url-pattern>

</servlet-mapping>

<servlet-mapping>

<servlet-name>Welcome</servlet-name>

<url-pattern>/Welcome</url-pattern>

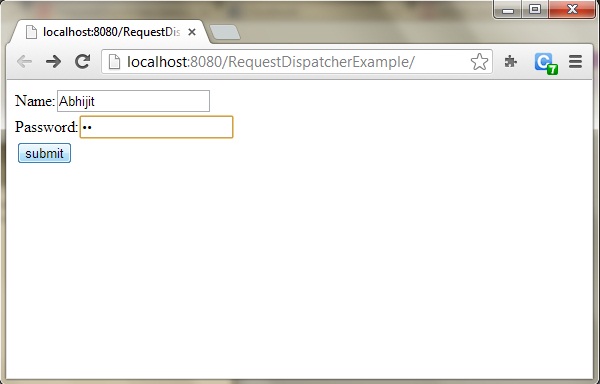
</servlet-mapping>

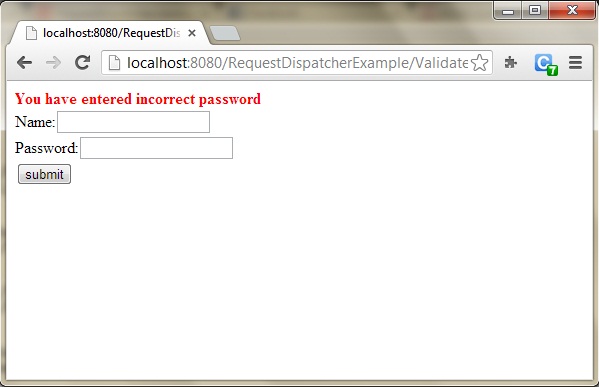
<welcome-file-list>

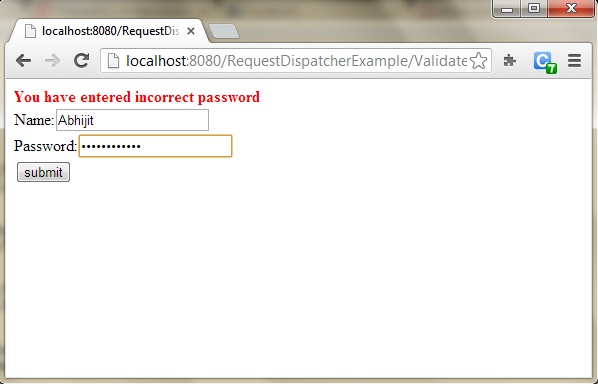
<welcome-file>index.html</welcome-file>

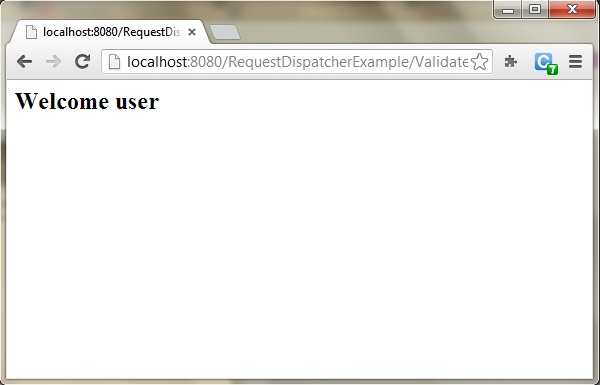
</welcome-file-list>

</web-app>









sendRedirect Method

sendRedirect() method redirect the response to another resource. sendRedirect() 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() accept relative URL, so it can go for resources inside or outside the server.

sendRedirect and request dispatcher

The main difference between a redirect and a request dispatch is that, redirect 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.

sendRedirect works on response object while request dispatch work on request object.

Example

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.studytonight.com");

}finally {

out.close();

}

}

}