

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.

Methods of RequestDispatcher

RequestDispatcher interface provides two important methods

Methods	Description
void <code>forward(ServletRequest request, ServletResponse response)</code>	forwards a request from a servlet to another resource (servlet, JSP file, or HTML file) on the server
void <code>include(ServletRequest request, ServletResponse response)</code>	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);
```

ServletRequest object

resource name

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

forward the request and response to "hello.html" page

OR

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

The diagram shows the following code with annotations:

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

Annotations:

- ServletRequest object**: An arrow points from this text to the `request` parameter in the `getRequestDispatcher` method call.
- Resource name**: An arrow points from this text to the string `"first.html"` in the `getRequestDispatcher` method call.
- include the response of "first.html" page in current servlet response**: An arrow points from this text to the `include` method call.

Example demonstrating usage of RequestDispatcher

In this example, we will show you 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, **ValidateServlet** will validate the password entered by the user, if the user has entered "studytonight" as password, then he will be forwarded to **Welcome Servlet** else the user will stay on the index.html page and an error message will be displayed.

Files to be created :

- **index.html** will have form fields to get user information.
- **Validate.java** will validate the data entered by the user.
- **Welcome.java** will be the welcome page.
- **web.xml** , the deployment descriptor.

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("studytonight"))

            {

                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>

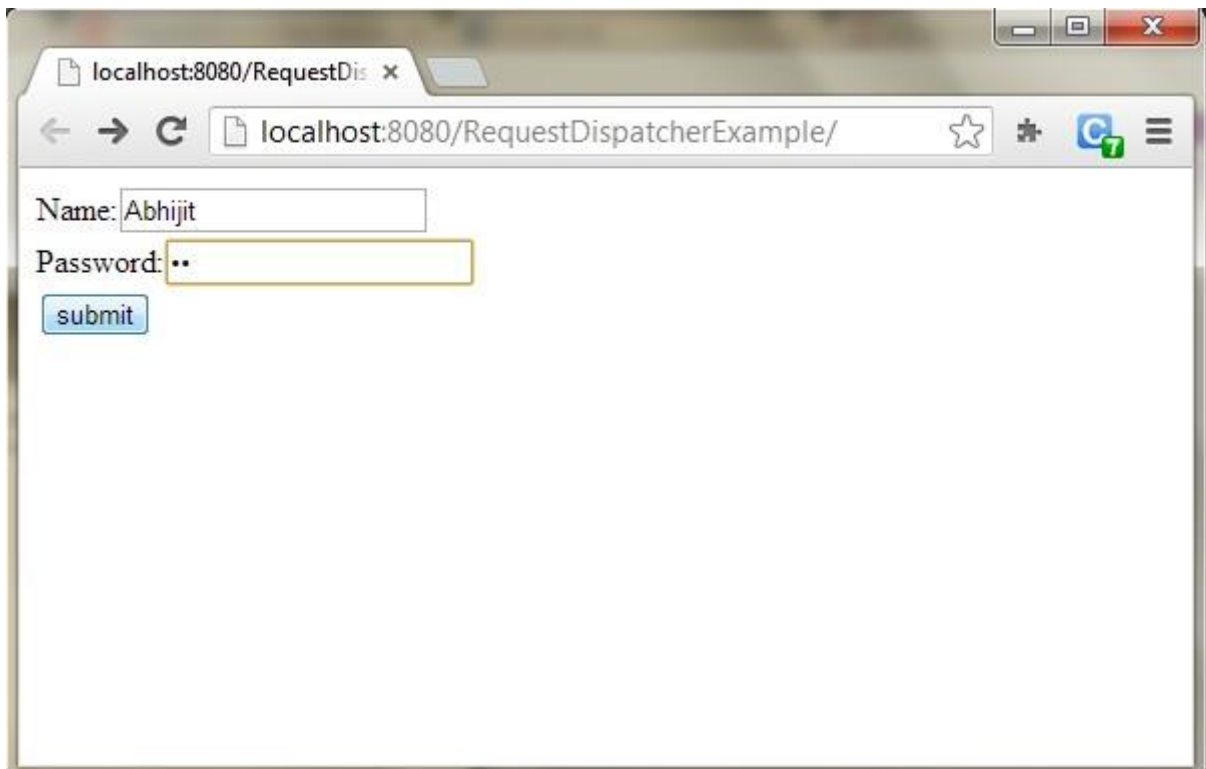
        <url-pattern>/Validate</url-pattern>

    </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>
```

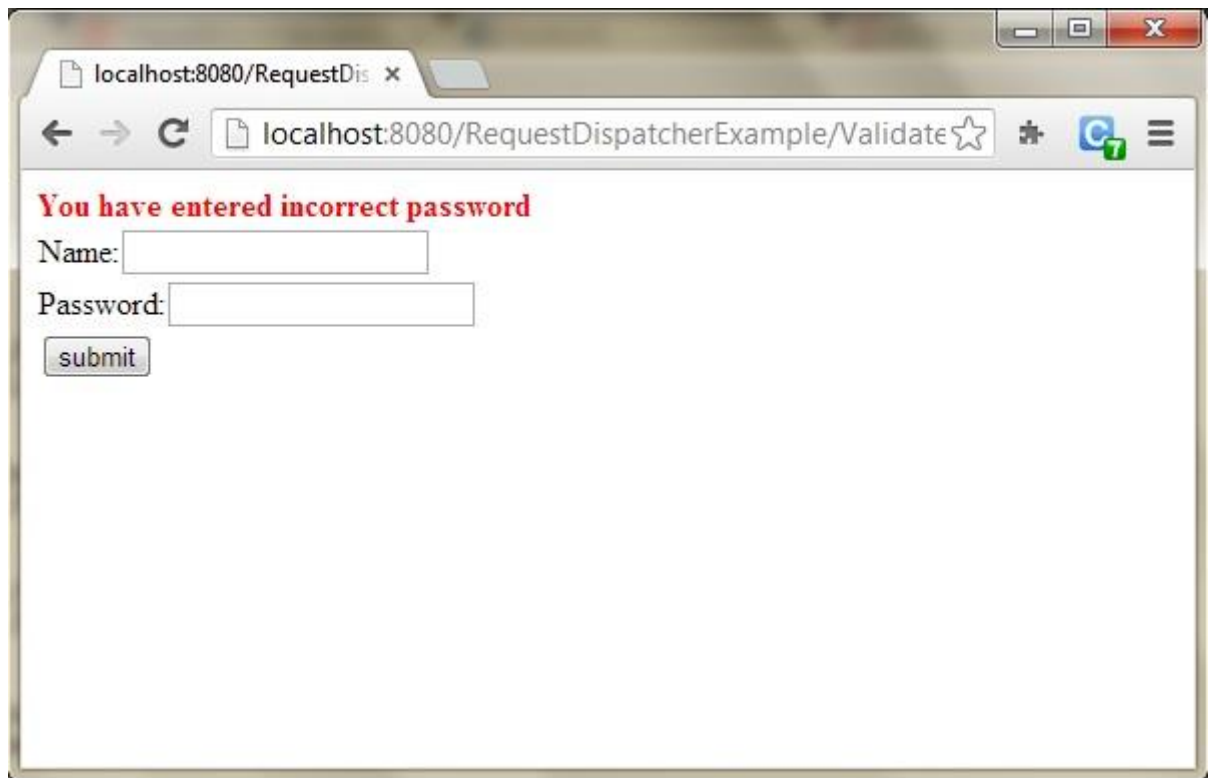
This will be the first screen. You can enter your Username and Password here.



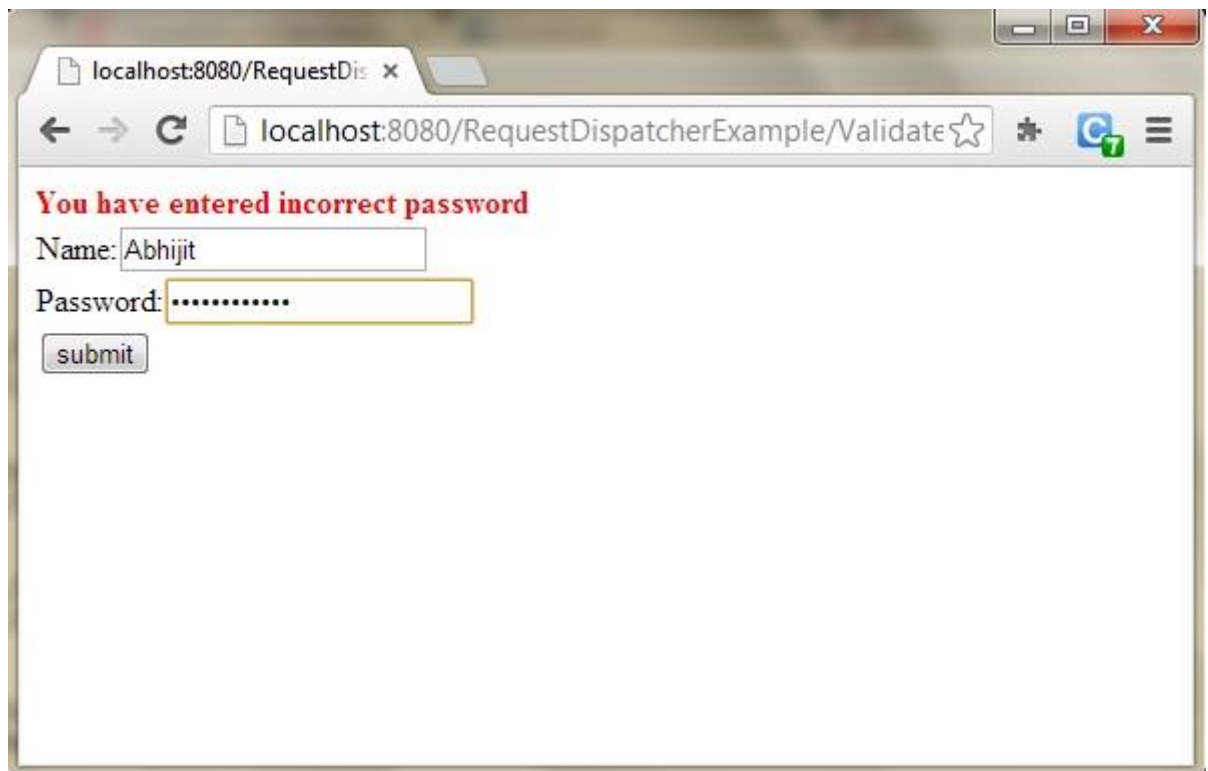
The screenshot shows a web browser window with the address bar displaying 'localhost:8080/RequestDispatcherExample/'. The page content includes a form with the following elements:

- A text input field labeled 'Name:' containing the text 'Abhijit'.
- A text input field labeled 'Password:' containing masked characters (dots).
- A blue button labeled 'submit'.

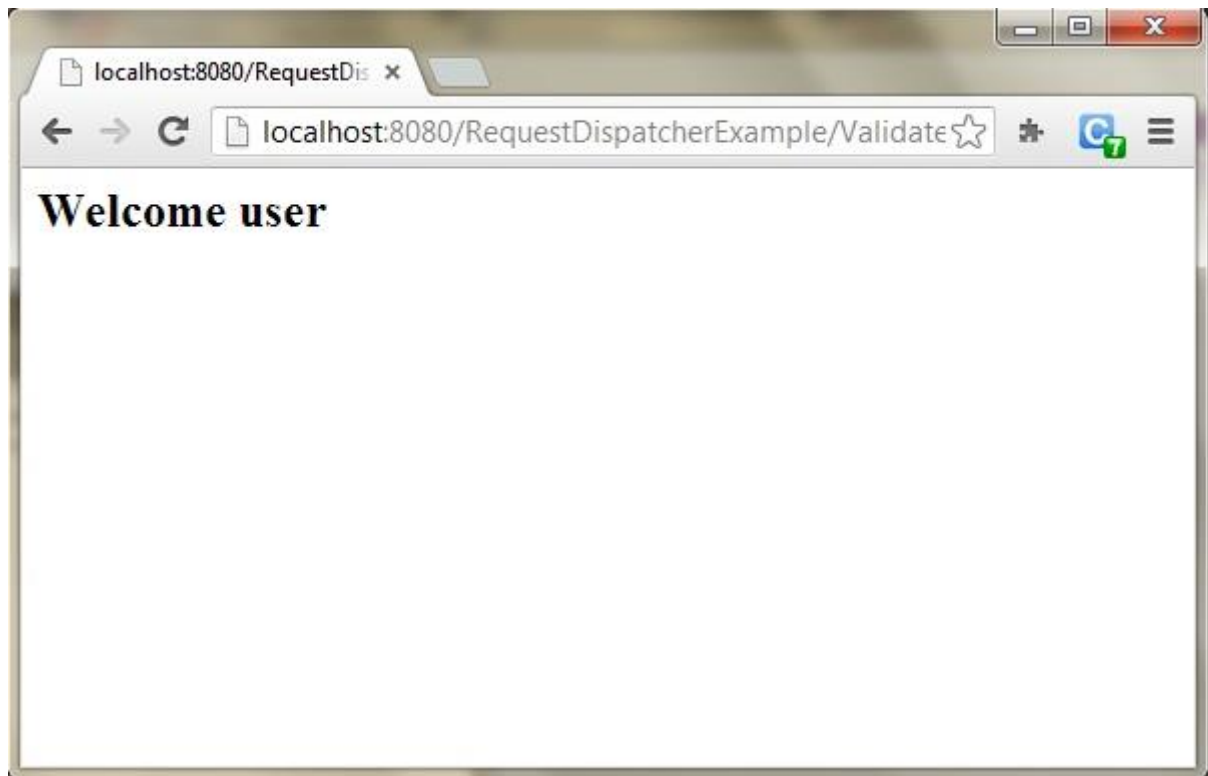
When you click on Submit, Password will be validated, if it is not 'studytonight', error message will be displayed.



Enter any Username, but enter 'studytonight' as password.



Password will be successfully validated and you will be directed to the Welcome Servlet.



SendRedirect in servlet

The **sendRedirect()** method of **HttpServletResponse** interface can be used to redirect response to another resource, it may be servlet, jsp or html file.

It accepts relative as well as absolute URL.

It works at client side because it uses the url bar of the browser to make another request. So, it can work inside and outside the server.

This method is used to redirect client request to some other location for further processing ,the new location is available on different server or different context. Our web container handle this and transfer the request using browser ,and this request is visible in browser as a new request. Sometime this is also called as client side redirect.

Difference between SendRedirect and Forward

Now let's see some difference between these two method of servlet API in tabular format.

Forward()	SendRedirect()
When we use forward method request is transfer to other resource within the same server for further processing.	In case of sendRedirect request is transfer to another resource to different domain or different server for further processing.
In case of forward Web container handle all process internally and client or browser is not involved.	When you use SendRedirect container transfers the request to client or browser so url given inside the sendRedirect method is visible as a new request to the client.
When forward is called on requestDispatcher object we pass request and response object so our old request object is present on new resource which is going to process our request	In case of SendRedirect call old request and response object is lost because it's treated as new request by the browser.
Visually we are not able to see the forwarded address, its is transparent	In address bar we are able to see the new redirected address it's not transparent.
Using forward () method is faster then send redirect.	SendRedirect is slower because one extra round trip is required because completely new request is created and old request object is lost. Two browser request required.
When we redirect using forward and we want to use same data in new resource we can use request.setAttribute () as we have request object available.	But in sendRedirect if we want to use we have to store the data in session or pass along with the URL.

Syntax of sendRedirect() method

1. **public void** sendRedirect(String URL)**throws** IOException;

Example of sendRedirect() method

```
1. response.sendRedirect("http://www.javatpoint.com");
```

Full example of sendRedirect method in servlet

In this example, we are redirecting the request to the google server. Notice that sendRedirect method works at client side, that is why we can our request to anywhere. We can send our request within and outside the server.

DemoServlet.java

```
1. import java.io.*;
2. import javax.servlet.*;
3. import javax.servlet.http.*;
4.
5. public class DemoServlet extends HttpServlet{
6. public void doGet(HttpServletRequest req,HttpServletResponse res)
7. throws ServletException,IOException
8. {
9. res.setContentType("text/html");
10. PrintWriter pw=res.getWriter();
11.
12. response.sendRedirect("http://www.google.com");
13.
14. pw.close();
15. }}
```

Creating custom google search using sendRedirect

In this example, we are using sendRedirect method to send request to google server with the request data.

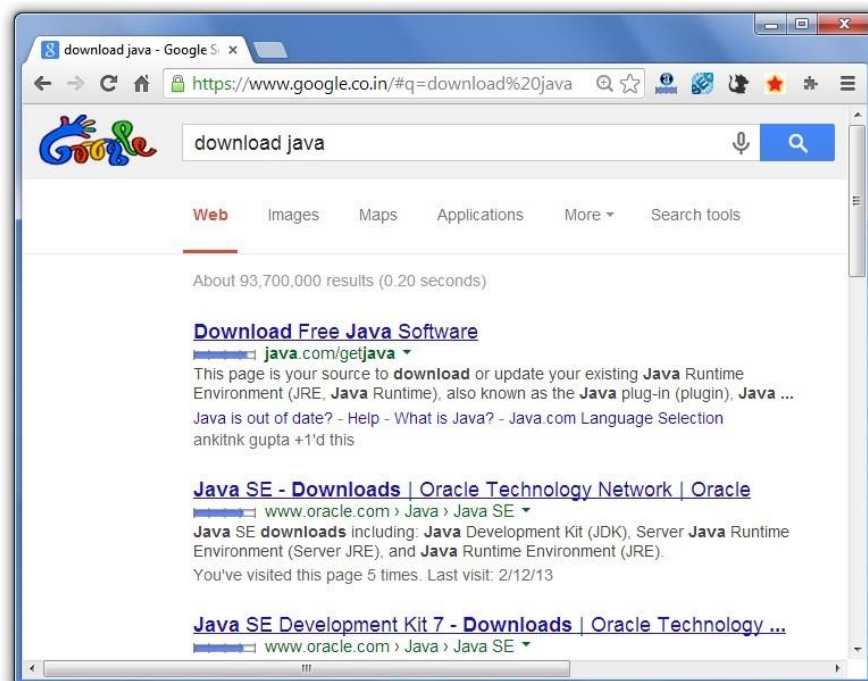
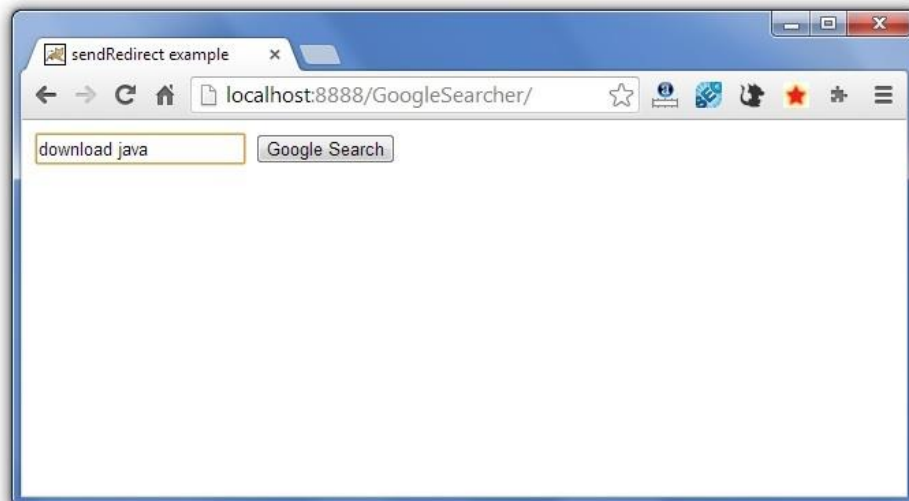
index.html

```
1. <!DOCTYPE html>
2. <html>
3. <head>
4. <meta charset="ISO-8859-1">
5. <title>sendRedirect example</title>
6. </head>
7. <body>
8.
9.
10. <form action="MySearcher">
11. <input type="text" name="name">
12. <input type="submit" value="Google Search">
13. </form>
14.
15. </body>
16. </html>
```

MySearcher.java

```
1. import java.io.IOException;
2. import javax.servlet.ServletException;
3. import javax.servlet.http.HttpServlet;
4. import javax.servlet.http.HttpServletRequest;
5. import javax.servlet.http.HttpServletResponse;
6.
7. public class MySearcher extends HttpServlet {
8.     protected void doGet(HttpServletRequest request, HttpServletResponse response)
9.         throws ServletException, IOException {
10.
11.         String name=request.getParameter("name");
12.         response.sendRedirect("https://www.google.co.in/#q="+name);
13.     }
14. }
```

Output



Example of forward and SendRedirect in JSP Servlet:

Any kind of online payment when we use merchant site will redirect us to net banking site which is completely new request it process our request and again redirect to merchant site?

In Banking Application when we do login normally we use forward method. In case of online banking we are asked for username and password if it's a correct some another servlet or resource will handle the request other wise request has been forwarded to error page.

Which one is good?

Its depends upon the scenario that which method is more useful.

If you want control is transfer to new server or context and it is treated as completely new task then we go for Send Redirect.

Normally forward should be used if the operation can be safely repeated upon a browser reload of the web page will not affect the result.

SendRedirect and forward method are still very useful while programming or working on any web application project using **servlet jsp**. This is still a popular interview questions so don't forget to revise **forward and sendRedirect** before appearing for any job interview.