Introduction to ServletRequest class

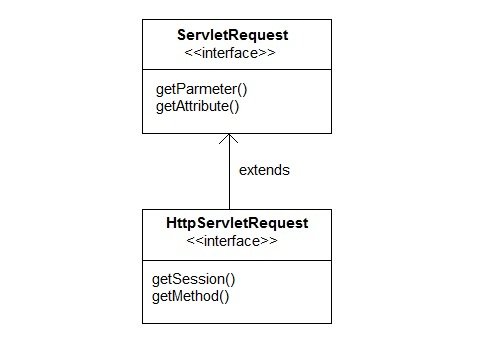
True job of a Servlet is to handle client request. Servlet API provides two important interfaces **javax.servlet.ServletRequest** and **javax.servlet.http.HttpServletRequest** to encapsulate client request. Implementation of these interfaces provide important information about client request to a servlet.

Some Important Methods of ServletRequest

|  |  |
| --- | --- |
| **Methods** | **Description** |
| Object getAttribute(String name) | return attribute set on request object by name |
| Enumeration getAttributeName() | return an Enumeration containing the names of the attributes available inthis request |
| int getContentLength() | return size of request body |
| int getContentType() | return media type of request content |
| ServletInputStream getInputStream() | returns a input stream for reading binary data |
| String getParameter(String name) | returns value of parameter by name |
| String getLocalAddr() | returns the Internet Protocol(IP) address of the interface on which the request was received |
| Enumeration getParameterNames() | returns an enumeration of all parameter names |
| String[] getParameterValues(String name) | returns an array of String objects containing all of the values the given request parameter has, or null if the parameter does not exist |
| ServletContext getServletContext() | return the servlet context of current request. |
| String getServerName() | returns the host name of the server to which the request was sent |
| int getServerPort() | returns the port number to which the request was sent |
| boolean isSecure() | returns a boolean indicating whether this request was made using a secure channel, such as HTTPS. |
| void removeAttribute(String name) | removes an attribute from this request |
| void setAttribute(String name, Object o) | stores an attribute in this request. |

HttpServletRequest interface

**HttpServletRequest** interface adds the methods that relates to the **HTTP** protocol.



Some important methods of HttpServletRequest

|  |  |
| --- | --- |
| **Methods** | **Description** |
| String getContextPath() | returns the portion of the request URI that indicates the context of the request |
| Cookies getCookies() | returns an array containing all of the Cookie objects the client sent with this request |
| String getQueryString() | returns the query string that is contained in the request URL after the path |
| HttpSession getSession() | returns the current HttpSession associated with this request or, if there is no current session and create is true, returns a new session |
| String getMethod() | Returns the name of the HTTP method with which this request was made, for example, GET, POST, or PUT. |
| Part getPart(String name) | gets the Part with the given name |
| String getPathInfo() | returns any extra path information associated with the URL the client sent when it made this request. |
| String getServletPath() | returns the part of this request's URL that calls the servlet |

Example demonstrating Servlet Request

In this example, we will show how a parameter is passed to a Servlet in a request object from HTML page.

**index.html**

<form method="post" action="check">

Name <input type="text" name="user" >

<input type="submit" value="submit">

</form>

Copy

**web.xml**

<servlet>

<servlet-name>check</servlet-name>

<servlet-class>MyServlet</servlet-class>

</servlet>

<servlet-mapping>

<servlet-name>check</servlet-name>

<url-pattern>/check</url-pattern>

</servlet-mapping>

Copy

**MyServlet.java**

import java.io.\*;

import javax.servlet.\*;

import javax.servlet.http.\*;

public class MyServlet 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 user = request.getParameter("user");

out.println("<h2> Welcome "+user+"</h2>");

} finally {

out.close();

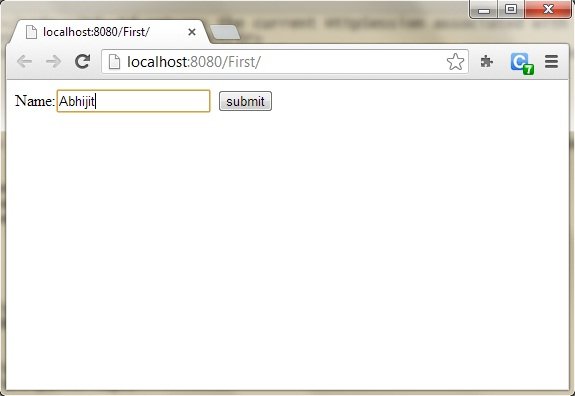
}

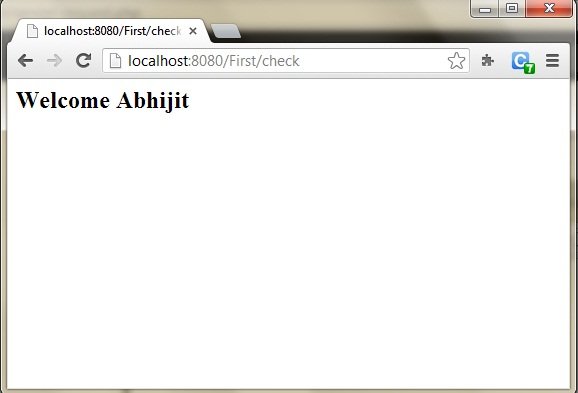
}

}

Copy

**Output :**





# Introduction to ServletResponse

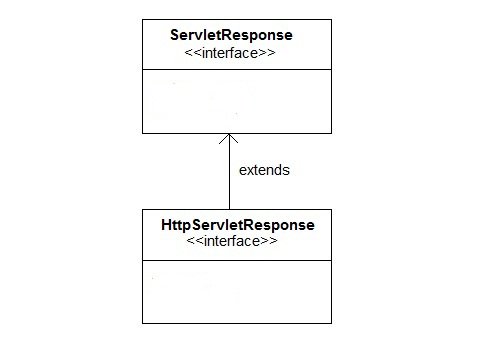
Servlet API provides two important interfaces **ServletResponse** and **HttpServletResponse** to assist in sending response to client.

## Some Important Methods of ServletResponse

|  |  |
| --- | --- |
| **Methods** | **Description** |
| PrintWriter getWriter() | returns a PrintWriter object that can send character text to the client. |
| void setBufferSize(int size) | Sets the preferred buffer size for the body of the response |
| void setContentLength(int len) | Sets the length of the content body in the response In HTTP servlets, this method sets the HTTP Content-Length header |
| void setContentType(String type) | sets the content type of the response being sent to the client before sending the respond. |
| void setBufferSize(int size) | sets the preferred buffer size for the body of the response. |
| boolean isCommitted() | returns a boolean indicating if the response has been committed |
| void setLocale(Locale loc) | sets the locale of the response, if the response has not been committed yet. |

## HttpServletResponse Interface

**HttpServletResponse** interface adds the methods that relate to the **HTTP** response. It extends the ServletResponse interface. The object of HttpServletResponse is created in servlet container.



### Some Important Methods of HttpServletResponse

|  |  |
| --- | --- |
| **Methods** | **Description** |
| void addCookie(Cookie cookie) | adds the specified cookie to the response. |
| void sendRedirect(String location) | Sends a temporary redirect response to the client using the specified redirect location URL and clears the buffer |
| int getStatus() | gets the current status code of this response |
| String getHeader(String name) | gets the value of the response header with the given name. |
| void setHeader(String name, String value) | sets a response header with the given name and value |
| void setStatus(int sc) | sets the status code for this response |
| void sendError(int sc, String msg) | sends an error response to the client using the specified status and clears the buffer |

### Example of HttpServletResponse interface on Eclipse

#### For creating a HttpServletResponse interface below is the directory structure of the program:

### Following are the steps for creating the program.

Step 1: Create a dynamic project on eclipse by clicking on **File => New => Dynamic Web Project**

**Step 2: Now create an HTML file.**

Right-click on the project and then click on HTML file. Give the name of the file and then click on the finish button.

#### And write the below code.

<!DOCTYPE html>

<html>

<head>

<meta charset="ISO-8859-1">

<title>test1</title>

</head>

<body>

<form align="center" action="display" method="get">

<h3>test1</h3>

<hr>

Enter User name: <input type="text" name="val1"><br><br>

Enter Password: &nbsp;&nbsp;<input type="password" name="val2" ><br><br>

<input type="submit" value="login">

</body>

</html>

Copy

**Step 3: now add the below code in web.xml file.**

web.xml file is a deployment descripter. Here we have all the configurations.

<?xml version="1.0" encoding="UTF-8"?>

<web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns="http://xmlns.jcp.org/xml/ns/javaee" xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/javaee http://xmlns.jcp.org/xml/ns/javaee/web-app\_4\_0.xsd" id="WebApp\_ID" version="4.0">

<display-name>ServletResponse</display-name>

<servlet>

<servlet-name>abc3</servlet-name>

<servlet-class>demo4</servlet-class>

</servlet>

<servlet-mapping>

<servlet-name>abc3</servlet-name>

<url-pattern>/display</url-pattern>

</servlet-mapping>

</web-app>

Copy

**Step 4: Now next create a servlet. For this create a class. Give the package name and the class name.**

Add the below code in the class file.

#### Demo4.java

import javax.servlet.http.\*;

import javax.servlet.\*;

import java.io.\*;

public class demo4 extends HttpServlet{

public void doGet(HttpServletRequest req,HttpServletResponse res)

throws ServletException,IOException

{

res.setContentType("text/html");

PrintWriter pwriter=res.getWriter();

String uname=req.getParameter("val1");

String pw=req.getParameter("val2");

pwriter.println("User Details Page:");

pwriter.println("Hello "+uname);

pwriter.println("Your Password is \*\*"+pw+"\*\*");

pwriter.close();

}

}

Copy

#### Now, Run the code

To run the code, right-click on the project and select Run As => Run on Server.

# Request Dispatcher in Servlet

In Java, the RequestDispatcher Interface is used for dispatching the request to a resource i.e Html, servlet or JSP. The Contents of another resource can be included in this interface. There are two methods of RequestDispatcher. They are as following:

## Servlet: Methods of RequestDispatcher

**RequestDispatcher** interface provides two important methods

|  |  |
| --- | --- |
| **Methods** | **Description** |
| public void forward(ServletRequest request,ServletResponse response)throws ServletException,java.io.IOException | It is used for forwarding the request from one servlet to another servlet on a server. |
| public void include(ServletRequest request,ServletResponse response)throws ServletException,java.io.IOException | It is used for including the content of the resource in the response. |

### forward() method:

### include() method:

### Example of forward() and include() method on Eclipse

For creating a program using forward() and include() method below is the directory structure of the program:

### Following are the steps for creating the program.

Step 1: Create a dynamic project on eclipse by clicking on **File => New => Dynamic Web Project**

Step 2: Now create an HTML file.

Right-click on the project and then click on HTML file. Give the name of the file and then click on the finish button.

And write the below code.

**Index.html**

<!DOCTYPE html>

<html>

<head>

<meta charset="ISO-8859-1">

<title>studytonight</title>

</head>

<body>

<form align="center" action="display" method="post">

<h3>test1</h3>

<hr>

Name: <input type="text" name="val1"><br> <br>

User Id: <input type="text" name="val2"><br> <br>

Password: <input type="password" name="val3"><br> <br>

<input type="submit" value="login">

</form>

</body>

</html>

Copy

Step 3: Now add the below code in web.xml file.

web.xml file is a deployment descripter. Here we have all the configurations.

<?xml version="1.0" encoding="UTF-8"?>

<web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns="http://xmlns.jcp.org/xml/ns/javaee" xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/javaee http://xmlns.jcp.org/xml/ns/javaee/web-app\_4\_0.xsd" id="WebApp\_ID" version="4.0">

<display-name>forward</display-name>

<servlet>

<servlet-name>demo5</servlet-name>

<servlet-class>forward.demo5</servlet-class>

</servlet>

<servlet>

<servlet-name>demo5i</servlet-name>

<servlet-class>forward.demo5i</servlet-class>

</servlet>

<servlet-mapping>

<servlet-name>demo5</servlet-name>

<url-pattern>/display</url-pattern>

</servlet-mapping>

<servlet-mapping>

<servlet-name>demo5i</servlet-name>

<url-pattern>/display1</url-pattern>

</servlet-mapping>

<welcome-file-list>

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

</welcome-file-list>

</web-app>

Copy

Step 4: Now next create a servlet. For this create a class. Give the package name and the class name.

For this example we need two servlet classes.

Now add the below code in the class file.

**demo5.java**

package forward;

import java.io.\*;

import javax.servlet.\*;

import javax.servlet.http.\*;

public class demo5 extends HttpServlet {

public void doPost(HttpServletRequest request, HttpServletResponse response)

throws ServletException, IOException {

response.setContentType("text/html");

PrintWriter out = response.getWriter();

String n=request.getParameter("val1");

String u=request.getParameter("val2");

String p=request.getParameter("val3");

if(p.equals("studytonight"))

{

RequestDispatcher rd=request.getRequestDispatcher("display1");

rd.forward(request, response);

}

else{

out.print("Incorrect UserId or Password");

RequestDispatcher rd=request.getRequestDispatcher("/index.html");

rd.include(request, response);

}

}

}

Copy

**demo5i.java**

package forward;

import java.io.\*;

import javax.servlet.\*;

import javax.servlet.http.\*;

public class demo5i extends HttpServlet {

public void doPost(HttpServletRequest request, HttpServletResponse response)

throws ServletException, IOException {

response.setContentType("text/html");

PrintWriter out = response.getWriter();

String n=request.getParameter("val2");

out.print("Welcome "+n);

}

}

Copy

Now, Run the code.

To run the code, right-click on the project and select Run As => Run on Server.

Below is the index.html page. Fill all the fields and click on the login button for landing in the servlet page.

If your password is correct then it will land on the servlet page demo5.java

If your password is incorrect then it will land on demo5i.java page.

### 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);

Copy



**OR**

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

rs.include(request,response);

Copy



### 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, **Validate** Servlet 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>

Copy

**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();

}

}

}

Copy

**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();

}

}

}

Copy

**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>

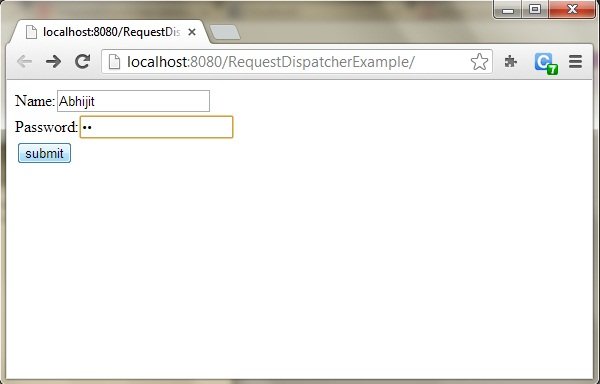
<welcome-file-list>

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

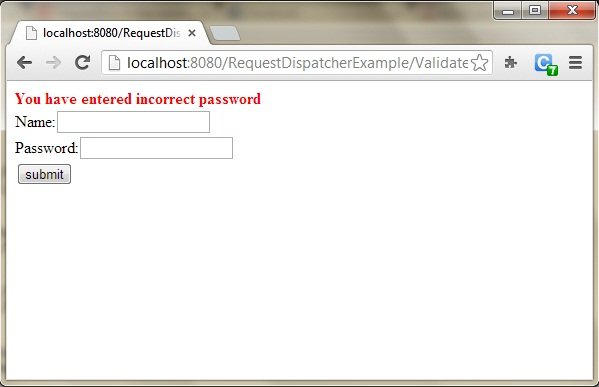
</welcome-file-list>

</web-app>

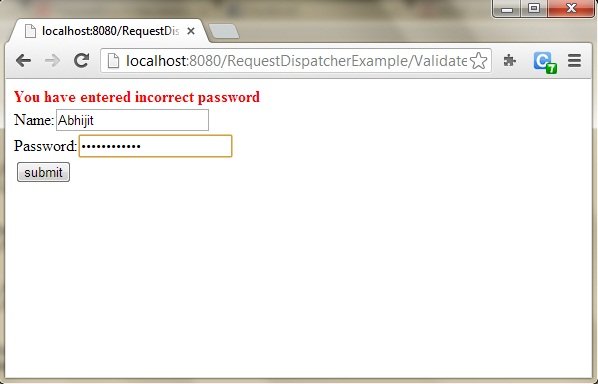
This will be the first screen. You can enter your Username and Password here.



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() Method in Servlet

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.

## Servlet: 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.

### Example demonstrating usage of sendRedirect()

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("https://www.test1");

}

finally {

out.close();

}

}

}

Copy

### Difference between forward() and sendRedirect()

|  |  |
| --- | --- |
| **forward()** | **sendRedirect()** |
| It works at server side. | It works at client side. |
| It always sends the same request and response object. | It always sends new request for the object. |
| It only works within the server. | It works both inside and outside the server. |
| In this method, all the processing is handled by web container internally. | In this method, all the processing is handled by another servlet. |
| It is faster. | It is slower. |
| Using forward() method address can be seen in address bar. | Using forward() method address can not be seen in address bar. |
| RequestDispatcher interface is used for declaration. | HttpServletResponse is used for declaration. |
| It is very useful in MVC design pattern to hide direct access. | It is not useful in MVC design pattern to hide direct access. |
| It reuses the object. | It does not reuse the object. |
| Example: request.getRequestDispacher("servlet\_1").forward(request response); | Example: response.sendRedirect("Servlet\_1"); |

Introduction to ServletConfig interface

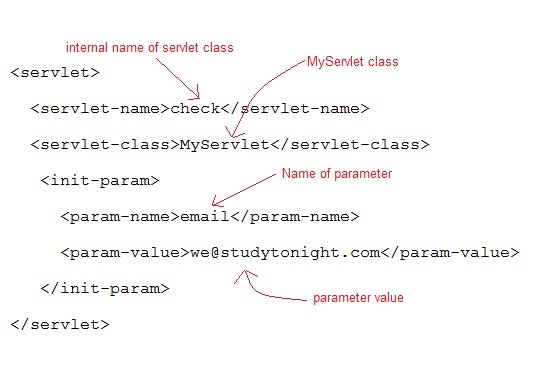
When the **Web Container** initializes a servlet, it creates a **ServletConfig** object for the servlet. ServletConfig object is used to pass information to a servlet during initialization by getting configuration information from **web.xml**(Deployment Descriptor).

Methods of ServletConfig

* String getInitParameter(String name): returns a String value initialized parameter, or NULL if the parameter does not exist.
* Enumeration getInitParameterNames(): returns the names of the servlet's initialization parameters as an Enumeration of String objects, or an empty Enumeration if the servlet has no initialization parameters.
* ServletContext getServletContext(): returns a reference to the ServletContext
* String getServletName(): returns the name of the servlet instance

How to Initialize a Servlet inside web.xml

**In the Deployment Descriptor(web.xml) file,**



**Or, Inside the Servlet class, using following code,**

ServletConfig sc = getServletConfig();

out.println(sc.getInitParameter("email"));

Copy

Example demonstrating usage of ServletConfig

**web.xml**

<web-app...>

<servlet>

<servlet-name>check</servlet-name>

<servlet-class>MyServlet</servlet-class>

<init-param>

<param-name>email</param-name>

<param-value>we@test1</param-value>

</init-param>

</servlet>

<servlet-mapping>

<servlet-name>check</servlet-name>

<url-pattern>/check</url-pattern>

</servlet-mapping>

<welcome-file-list>

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

</welcome-file-list>

</web-app>

Copy

**MyServlet class :**

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();

ServletConfig sc = getServletConfig();

out.println(sc.getInitParameter("email"));

}

}