Request Dispatchers

Table of Contents

[Initialization 3](#_Toc100530000)

[Example 4](#_Toc100530001)

[Redirects 6](#_Toc100530002)

So far, we have seen that a client makes a request that a servlet responds to. However, consider that we have a webpage that is being served by one servlet but some small section of the webpage comes from another servlet. Basically, we need to include the results of one servlet inside the results of another servlet. We can do this using a **request dispatcher**. The operation used to include the results of another servlet is called an **include operation**.

There is another use case for a request dispatcher. Suppose the servlet that the client requested is incapable of giving the required output itself. In this case, it can forward the request to another servlet, which can respond properly. This is called a **forward operation**. Note that for a forward operation, the first servlet does not give any output.

A request dispatcher is actually an **interface** which has the include() and forward() methods. These methods are called inside the do\_\_() methods of the first servlet. The inclusion or forwarding must be handled by the **same** do\_\_\_() method in the second servlet, e.g. if the first servlet is using the doGet() method, the second servlet must use the doGet() method as well.

For both the include() and the forward() methods, we must pass the request and response objects as parameters. This is because the second servlet will only be able to produce an output if it can access the request made by the client and can write to the response that will be sent back.

The include() method can also work with **static or dynamic content** instead of a servlet. This means we can include some resource, such as an HTML page.

## Initialization

To **initialize** a request dispatcher, we can use one of the three methods:

1. In the first method, we use the getServletContext() method. A **servlet context** contains information about all the servlets in our application. This could be from the web.xml file or from the annotations.

*RequestDispatcher* rd = getServletContext().getRequestDispatcher("/SecondServlet");

JAVA

Note that is this case, we have to use the **entire relative path** to the servlet.

1. In the second method, we can use the **request object** itself to get the request dispatcher. Again, we must use the **entire relative path** to the servlet.

*RequestDispatcher* rd = request.getRequestDispatcher("/SecondServlet");

JAVA

1. The last option we have is to again use the **servlet context**, but this time with a different method, one which takes just the **name of the servlet**, not the path.

*RequestDispatcher* rd = getServletContext().getNamedDispatcher("SecondServlet");

JAVA

Once we have the request dispatcher object, we can easily call the include() or forward() methods.

rd.include(request, response);  
rd.forward(request, response);

JAVA

If we have some **static or dynamic content** we want to include, we can initialize the request dispatcher like this:

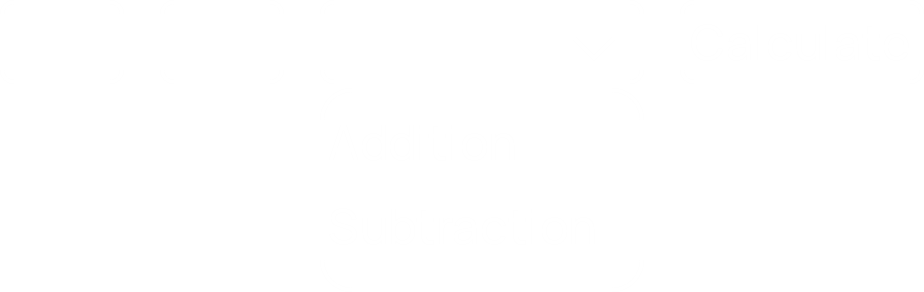
*RequestDispatcher* rd = request.getRequestDispatcher("staticpage.html");

JAVA

Again, this must be the **full relative path**.

### Example

Consider that we have a webpage where clients can input two numbers and then select one of the two operations, addition or subtraction, from a drop-down list.



Pressing the calculate button causes the request to go to a servlet FirstServlet. This servlet does not itself respond with anything. It simply checks which of the two operations is being requested and forwards the request to the appropriate servlet, either AdditionServletor SubtractionServlet. This is an example of a forward operation.

protected void doGet(*HttpServletRequest* request, *HttpServletResponse* response) throws ServletException, IOException {  
 String choice = request.getParameter("choice");  
 *RequestDispatcher* rd1 = getServletContext().getRequestDispatcher("/AdditionServlet");  
 *RequestDispatcher* rd2 = request.getRequestDispatcher("SubtractionServlet");

if (choice.equalsIgnoreCase("add")) rd1.forward(request, response);  
 else if (choice.equalsIgnoreCase("subtract")) rd2.forward(request, response);  
}

JAVA

Now suppose both the AdditionalServlet and the SubtractionServlet will include the same footer in their response. This footer is in a separate HTML page. This is an example of an include operation.

protected void doGet(*HttpServletRequest* request, *HttpServletResponse* response) throws ServletException, IOException {  
 double num1 = Double.*parseDouble*(request.getParameter("number1"));  
 double num2 = Double.*parseDouble*(request.getParameter("number2"));  
 double sum = num1 + num2;  
 PrintWriter out = response.getWriter();  
 out.println("<h2>Your result of addition is: " + sum + "</h2>");  
  
 *RequestDispatcher* rd = request.getRequestDispatcher("/footer.html");  
 rd.include(request, response);  
}

JAVA

## Redirects

There is one more method under the *RequestDispatcher* interface, sendRedirect(). This is used when the first servlet wishes to tell the client to send its request again, this time to a **different servlet**.

response.sendRedirect("loginPage.html");

JAVA

Although the sendRedirect() and forward() methods are similar, there are a few differences. The sendRedirect() method causes a round trip to the client. Additionally, the first request object is lost, so if the first servlet wishes to send any information to the second servlet, it must be appended to the URL as a **query string**.