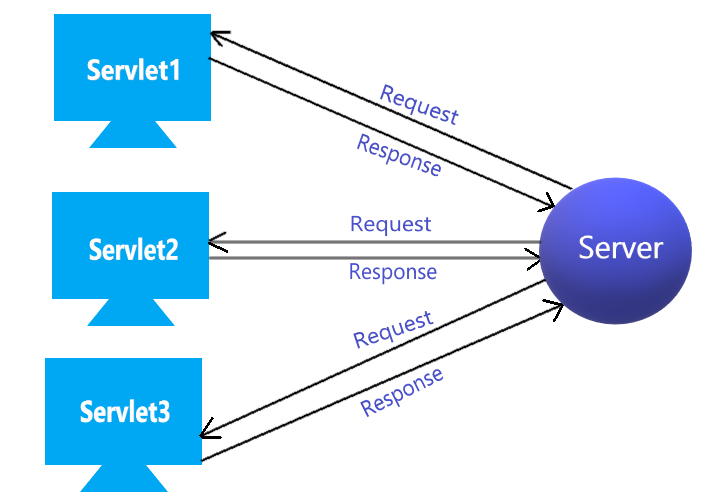
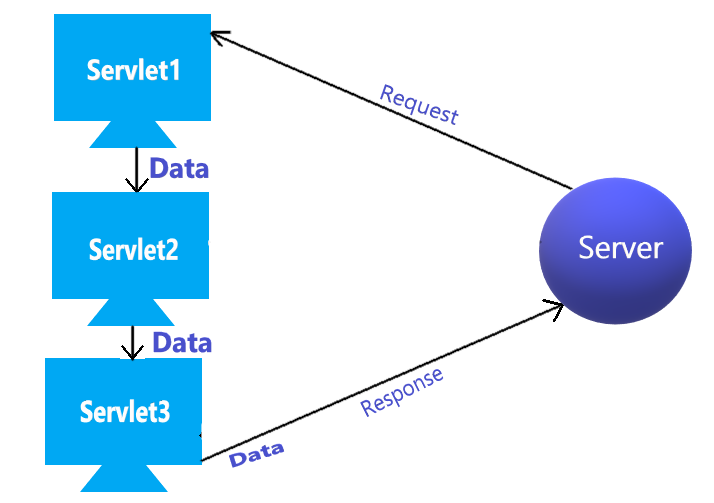
**Servlet dispatcher in java**

**Introduction**

Dispatcher is provision where request data can travel from one servlet to other servlet. Request dispatcher occur within server or container, while in case of send redirect for new request it come backs to network.



Consider situation where we have three servlet named as servlet1,servlet2 and Servlet 3 within server .In case if we don’t use dispatcher, whenever we request for servlet1, server pass control to servlet1 and after that if we request for servlet2 then control come backs from servlet 1 to server and from server passed to servlet2. In this case if server is located in India and servlet is requested from America then for second request it must come back to server(India) and go back to servlet(America). This option is not good if we have heavy traffic in between request and response. Solution of this problem is dispatcher.



In same case if we use dispatcher within server(container) then control is passed from servlet1 to servlet2 without coming back to server and without involving network. This concept is also known as servlet chaining. It is known as servlet chaining because we are creating chain of servlet request, from servlet1 to servlet2, Servlet2 to Servlet3 and server will get data from servlet3.

In servlet chaining not only control is passed but also data travel from one servlet to other servlet which is major advantage compare to send redirect. In send redirect every request is new request, every time you get new data.

Consider that servlet1 have some request parameter which should be executed by servlet3 in that case data can travel from servlet1 to servlet2 and after that from servlet2 to 3, so here we are preserving request from one servlet to other servlet.

Life of request is very small, as soon as we get response, request is over but here life of request can be preserved from one servlet to other. With help of this we can divide task in many servlet.

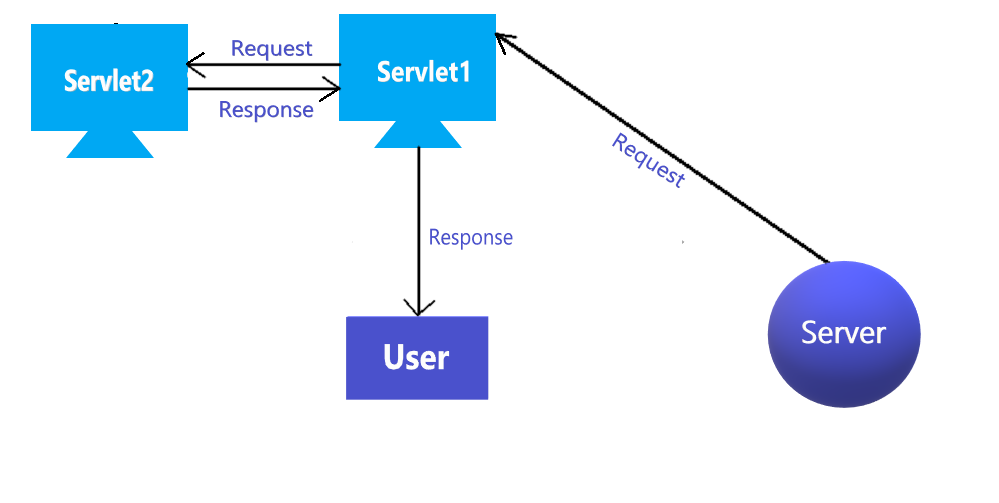
Most of the time dispatcher is efficient but in case of large data or if we don’t need data at all or in case of low trafficking send redirect work efficiently.

**Type of dispatcher**

1. Include
2. Forward

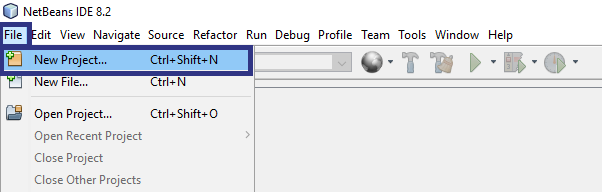
1) Include

In include calling servlet include data of called servlet. It is like method call where calling method get data of called method. In case of two servlet, servlet1 will include response of servlet2 and servlet1 is reverted back to client. Servlet2 will be called only to get response data.

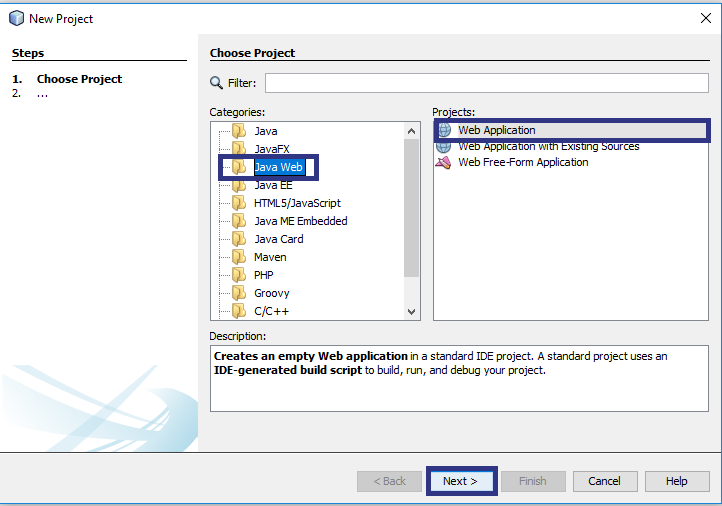


Let us create small example of include dispatcher

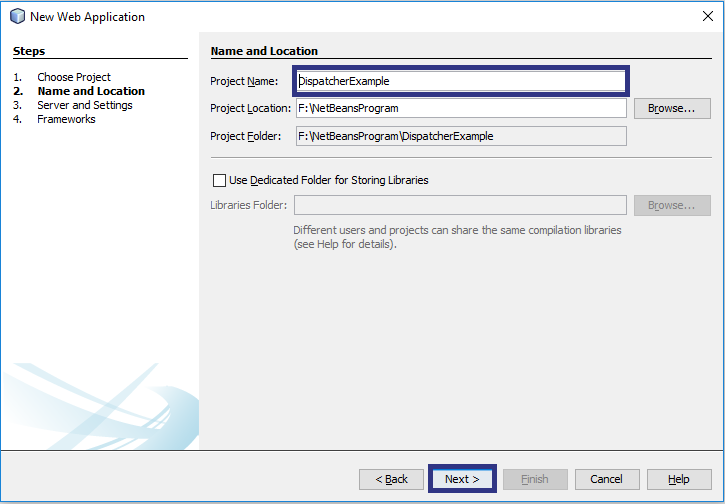
**Step: 1**



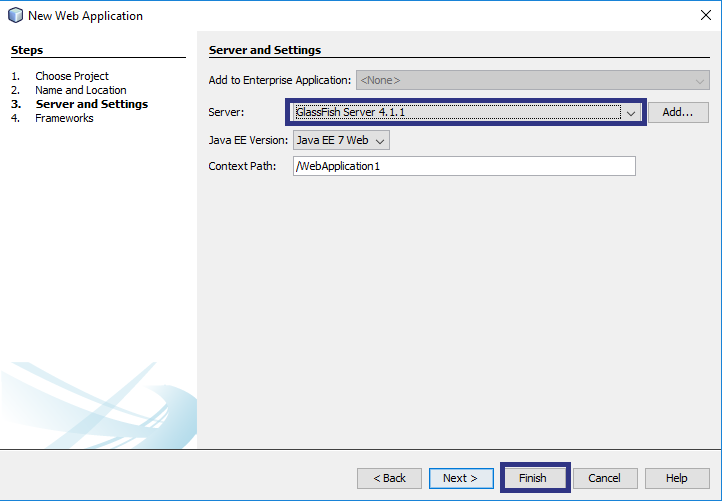
**Step: 2**



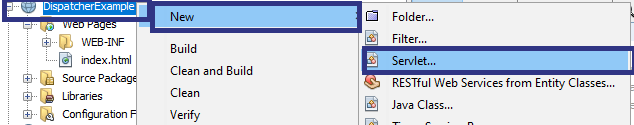
**Step: 3**

****

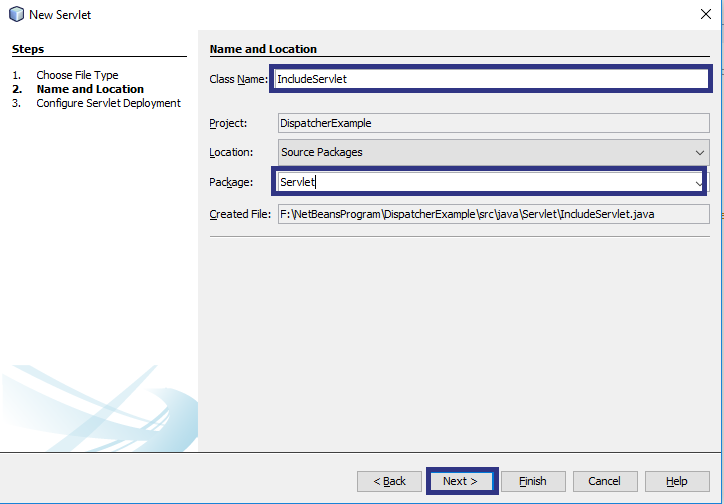
**Step: 4**

****

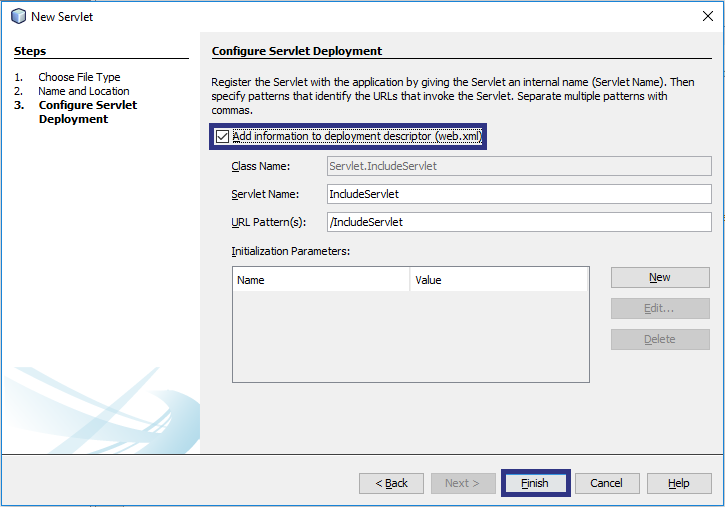
**Step: 5**



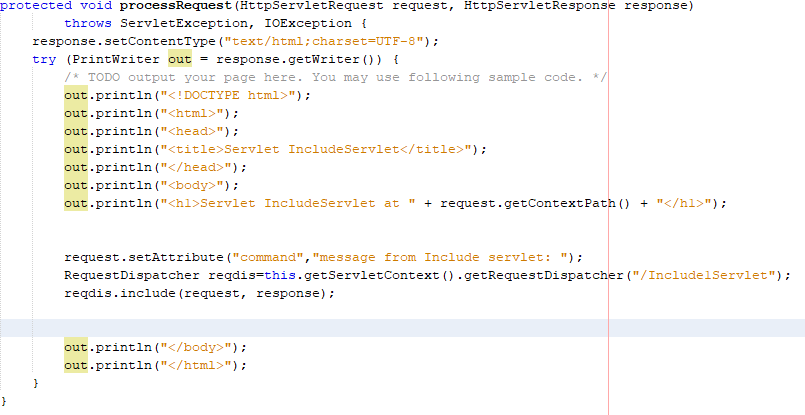
**Step: 6**



**Step: 7**



Add following three line code between body in IncludeServlet.



**Explanation**

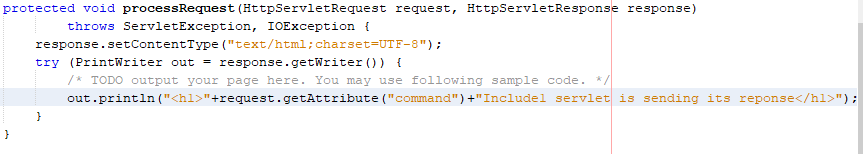
We have created command attribute which can travel between servlet. After that we have created instance of RequestDispatcher(right click on requestDispatcher and select fix imports which will add necessary namesapce) which specify servlet name which’s response we want to add in servlet. After that we have used reqdis.include(request,response) which will include response of Include1Servlet.

**Create Include1Servlet** and remove following code

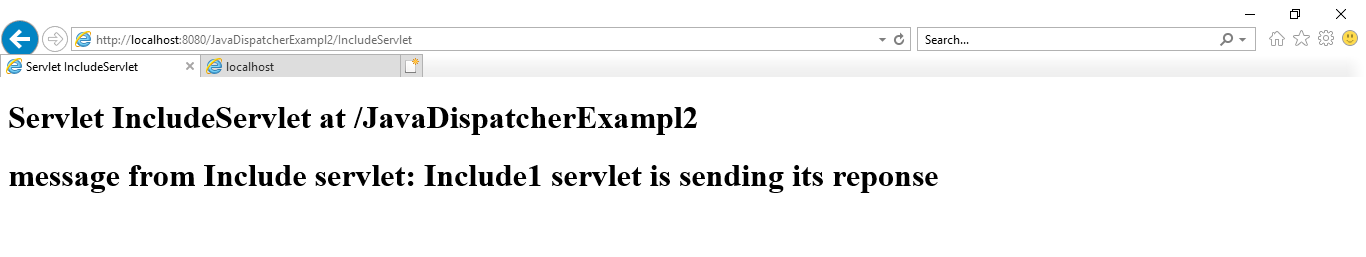
From out.println("<!DOCTYPE html>");

To out.println("</html>");

After that add following code.

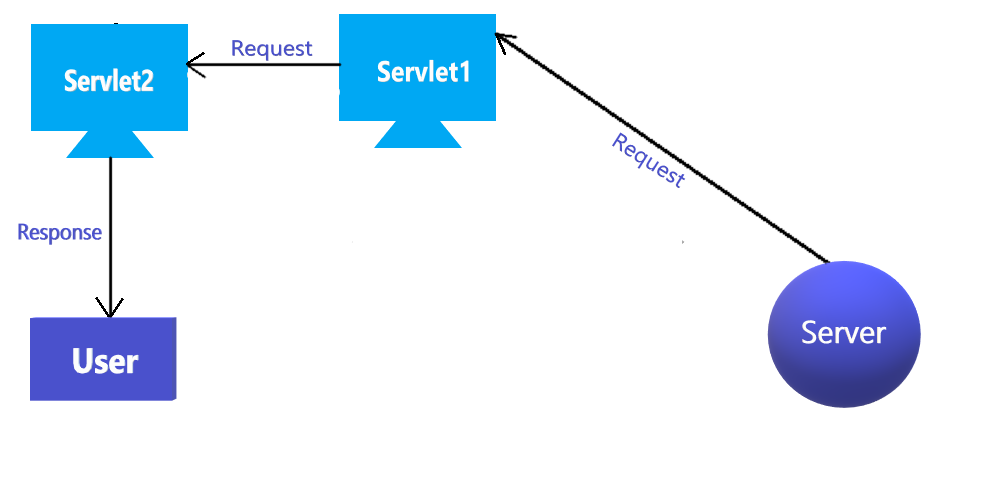


**Output:**



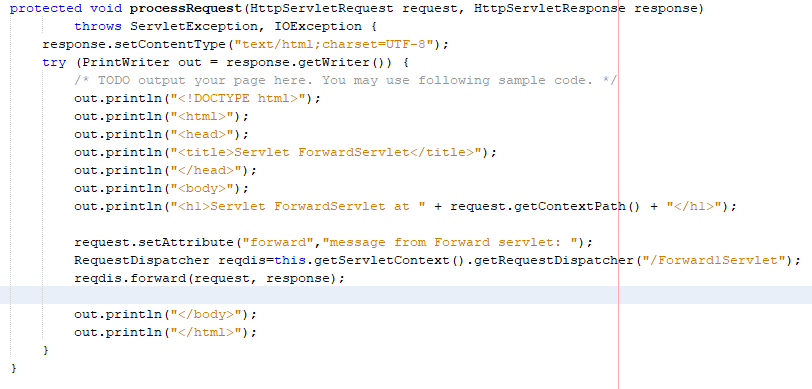
2) Forward

In forward, whatever data servlet1 have is forwarded to servlet2 and Servlet2 is reverted back to client.



**Create ForwardServlet**

Add following three line code between body in ForwardServlet.

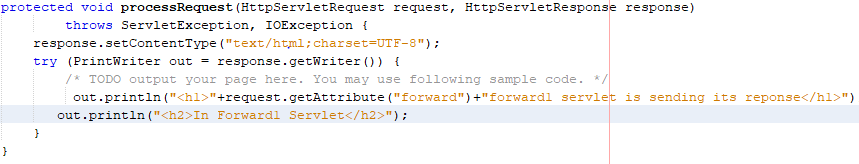


**Create Include1Servlet** and remove following code

From out.println("<!DOCTYPE html>");

To out.println("</html>");

After that add following code.



**Output:**

