Methods in Generic Servlet

public void init(ServletConfig config) is used to initialize the servlet.

public abstract void service(ServletRequest request, ServletResponse response) provides service for the incoming request. It is invoked at each time when user requests for a servlet.

public void destroy() is invoked only once throughout the life cycle and indicates that servlet is being destroyed.

public ServletConfig getServletConfig() returns the object of ServletConfig.

public String getServletInfo() returns information about servlet such as writer, copyright, version etc.

public void init() it is a convenient method for the servlet programmers, now there is no need to call super.init(config)

Methods in Generic Servlet

public ServletContext getServletContext() returns the object of ServletContext.

public String getInitParameter(String name) returns the parameter value for the given parameter name.

public Enumeration getInitParameterNames() returns all the parameters defined in the web.xml file.

public String getServletName() returns the name of the servlet object.

public void log(String msg) writes the given message in the servlet log file.

public void log(String msg,Throwable t) writes the explanatory message in the servlet log file and a stack trace.

Methods in HTTP Servlet

public void service(ServletRequest req,ServletResponse res) dispatches the request to the protected service method by converting the request and response object into http type.

protected void service(HttpServletRequest req, HttpServletResponse res) receives the request from the service method, and dispatches the request to the doXXX() method depending on the incoming http request type.

protected void doGet(HttpServletRequest req, HttpServletResponse res) handles the GET request. It is invoked by the web container.

protected void doPost(HttpServletRequest req, HttpServletResponse res) handles the POST request. It is invoked by the web container.

protected void doHead(HttpServletRequest req, HttpServletResponse res) handles the HEAD request. It is invoked by the web container.

Methods in HTTP Servlet

protected void doOptions(HttpServletRequest req, HttpServletResponse res) handles the OPTIONS request. It is invoked by the web container.

protected void doPut(HttpServletRequest req, HttpServletResponse res) handles the PUT request. It is invoked by the web container.

protected void doTrace(HttpServletRequest req, HttpServletResponse res) handles the TRACE request. It is invoked by the web container.

protected void doDelete(HttpServletRequest req, HttpServletResponse res) handles the DELETE request. It is invoked by the web container.

protected long getLastModified(HttpServletRequest req) returns the time when HttpServletRequest was last modified since midnight January 1, 1970 GMT.

doGet vs doPost

GET	POST
Query string or form data is simply appended to the URL as name-value pairs.	Form name-value pairs are sent in the body of the request, not in the URL itself.
Query length is limited (~1KB).	Query length is unlimited.
Users can see data in address bar.	Data hidden from users.
http://mycompany.com/support?Name=John+Smith&Product=go+kart&Complaint=the+engine+is+sputtering+oil.	http://mycompany.com/support <values of Name, Product, and Complaint are in request body>.
doGet().	doPost().
For getting (retrieving) data only.	For causing a change on the server (store data in DB).
ASCII.	ASCII + Binary.
Easy to bookmark.	Hard to bookmark.

Methods in HttpServletResponse

i. public void addCookie(Cookie cookie)

Adds the specified cookie to the response. This method can be called multiple times to set more than one cookie.

ii. public void addDateHeader (String name, long date)

Adds a response header with the given name and date-value.

iii. public void addHeader(String name,String value)

Adds a response header with the given name and value.

iv public void addIntHeader(String name,int value)

Adds a response header with the given name and integer value.

v public boolean containsHeader(String name)

Returns a boolean indicating whether the named response header has already been set.

vi. public String encodeRedirectURL(String url)

Encodes the specified URL for use in the sendRedirect method or, if encoding is not needed, returns the URL unchanged.

vii. public String encodeURL(String url)

Encodes the specified URL by including the session ID in it, or, if encoding is not needed, returns the URL unchanged.

Methods in HttpServletResponse

viii. public void sendError(int *sc)* throws IOException

Sends an error response to the client using the specified status code and clearing the buffer.

ix. public void sendError(int Be, String mag) throws IOException

Sends an error response to the client using the specified status clearing the buffer.

x. public void sendRedirect(string location) throws IOException

Sends a temporary redirect response to the client using the specified redirect location URL.

xi. public void setDateHeader(String name,long date)

Sets a response header with the given name and date-value.

xii. public void setHeader(String name, String value)

Sets a response header with the given name and value.

xiii. public void setlntHeader (String name,int value)

Sets a response header with the given name and integer value.

xiv public void setStatus(int sc)

Sets the status code for this response.

HTTP Response

HTTP Response sent by a server to the client. The response is used to provide the client with the resource it requested. It is also used to inform the client that the action requested has been carried out. It can also inform the client that an error occurred in processing its request.

An HTTP response contains the following things:

Status Line

Response Header Fields or a series of HTTP headers

Message Body

In the request message, each HTTP header is followed by a carriage returns line feed (CRLF). After the last of the HTTP headers, an additional CRLF is used and then begins the message body

Status Line

In the response message, the status line is the first line. The status line contains three items:

a) HTTP Version Number

b) Status Code

c) Reason Phrase

a) HTTP Version Number

HTTP-Version = HTTP/1.1

Status Code

It is a three-digit number that indicates the result of the request. The first digit defines the class of the response. The last two digits do not have any categorization role. There are five values for the first digit, which are as follows:

Code and Description

1xx: Information

It shows that the request was received and continuing the process.

2xx: Success

It shows that the action was received successfully, understood, and accepted.

3xx: Redirection

It shows that further action must be taken to complete the request.

4xx: Client Error

It shows that the request contains incorrect syntax, or it cannot be fulfilled.

5xx: Server Error

It shows that the server failed to fulfil a valid request.

Reason Phrase

It is also known as the status text. It is a human-readable text that summarizes the meaning of the status code.

An example of the response line is as follows:

HTTP/1.1 200 OK

Here,

HTTP/1.1 is the HTTP version.

200 is the status code.

OK is the reason phrase.

Response Header Fields

The HTTP Headers for the response of the server contain the information that a client can use to find out more about the response, and about the server that sent it. This information is used to assist the client with displaying the response to a user, with storing the response for the use of future, and with making further requests to the server now or in the future.

response-header = Accept-Ranges

| Age

| Location

| Proxy-Authenticate

| Retry-After

| Server

| Vary

| WWW-Authenticate

Message Body

The response's message body may be referred to for convenience as a response body.

The body of the message is used for most responses. The exceptions are where a server is using certain status codes and where the server is responding to a client request, which asks for the headers but not the response body.

For a response to a successful request, the body of the message contains either some information about the status of the action which is requested by the client or the resource which is requested by the client. For the response to an unsuccessful request, the body of the message might provide further information about some action the client needs to take to complete the request successfully or about the reason for the error.

Sharing Information between servlet

The exchange of information among servlets of a particular Java web application is known as **Servlet Collaboration**. This enables passing/sharing information from one servlet to the other through method invocations.

The servlet api provides two interfaces namely:

javax.servlet.RequestDispatcher

javax.servlet.http.HttpServletResponse

These two interfaces include the methods responsible for achieving the objective of sharing information between servlets.

Using RequestDispatcher Interface

Using HttpServletResponse Interface

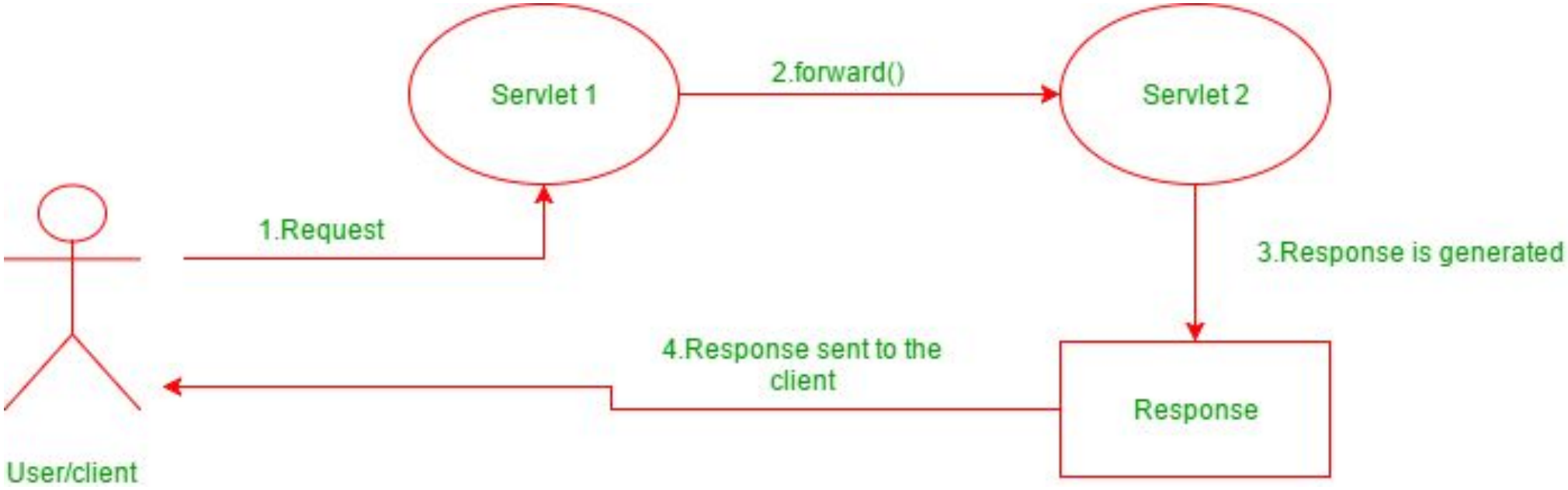
Using RequestDispatcher Interface

The RequestDispatcher interface provides the option of dispatching the client's request to another web resource, which could be an HTML page, another servlet, JSP etc. It provides the following two methods:

public void forward(ServletRequest request, ServletResponse response)throws ServletException, java.io.IOException:

The forward() method is used to transfer the client request to another resource (HTML file, servlet, jsp etc). When this method is called, the control is transferred to the next resource called. On the other hand, the include() method is used to include the content of the calling file into the called file. After calling this method, the control remains with the calling resource, but the processed output is included into the called resource.
The following diagram explains the way it works:

Forward() method

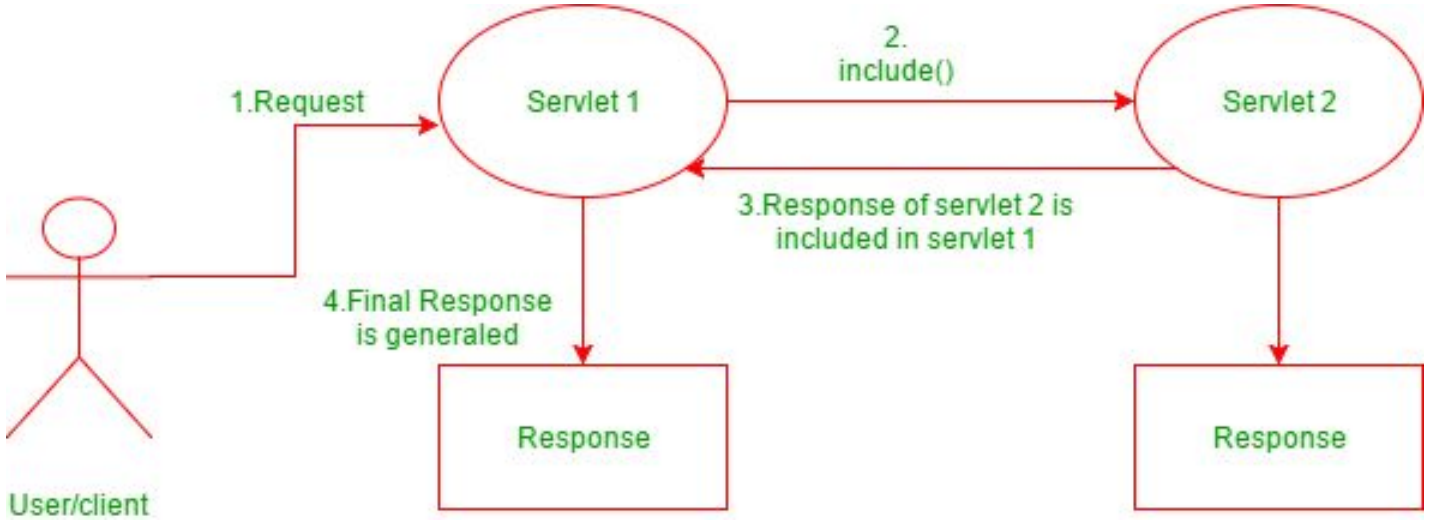


Include() method

public void include(ServletRequest request, ServletResponse response)throws ServletException, java.io.IOException:

The include() method is used to include the contents of the calling resource into the called one. When this method is called, the control still remains with the calling resource. It simply includes the processed output of the calling resource into the called one.

The following diagram explains how it works:



Using HttpServletResponse Interface

The HttpServletResponse interface is entrusted with managing Http responses. To achieve servlet collaboration, it uses the following method:

public void sendRedirect(String URL)throws IOException;

This method is used redirect response to another resource, which may be a servlet, jsp or an html file. The argument accepted by it, is a URL which can be both, absolute and relative. It works on the client side and uses the browser's URL bar to make a request.

forward() method of RequestDiispatcher Vs sendRedirect() of HttpServletResponse

Although the two methods appear to do the same thing, there are still differences between the two, which are as follows:

forward()

It works on the server side

It sends the same request and response objects to another resource.

It works only within the server.

sendRedirect()

It works on the client side

It always send a new request

It can be used within and outside the server.