

JSP - Server Response

In this chapter, we will discuss the Server Response in JSP. When a Web server responds to a HTTP request, the response typically consists of a status line, some response headers, a blank line, and the document. A typical response looks like this –

```
HTTP/1.1 200 OK
Content-Type: text/html
Header2: ...
...
HeaderN: ...
(Blank Line)
<!doctype ...>

<html>
  <head>...</head>
  <body>
    ...
  </body>
</html>
```

The status line consists of the HTTP version (**HTTP/1.1 in the example**), a status code (**200 in the example**), and a very short message corresponding to the status code (**OK in the example**).

Following is a summary of the most useful HTTP 1.1 response headers which go back to the browser from the web server. These headers are frequently used in web programming –

S.No.	Header & Description
1	Allow This header specifies the request methods (GET , POST , etc.) that the server supports.
2	Cache-Control This header specifies the circumstances in which the response document can safely be cached. It can have values public , private or no-cache etc. Public means document is cacheable, Private means document is for a single user and can only be stored in private (nonshared) caches and no-cache means document should never be cached.
3	Connection This header instructs the browser whether to use persistent HTTP connections or not. A value of close instructs the browser not to use persistent HTTP connections and keep-alive means using persistent connections.
4	Content-Disposition This header lets you request that the browser ask the user to save the response to disk in a file of the given name.
5	Content-Encoding This header specifies the way in which the page was encoded during transmission.
6	Content-Language

	This header signifies the language in which the document is written. For example, en , en-us , ru , etc.
7	Content-Length This header indicates the number of bytes in the response. This information is needed only if the browser is using a persistent (keep-alive) HTTP connection.
8	Content-Type This header gives the MIME (Multipurpose Internet Mail Extension) type of the response document.
9	Expires This header specifies the time at which the content should be considered out-of-date and thus no longer be cached.
10	Last-Modified This header indicates when the document was last changed. The client can then cache the document and supply a date by an If-Modified-Since request header in later requests.
11	Location This header should be included with all responses that have a status code in the 300s. This notifies the browser of the document address. The browser automatically reconnects to this location and retrieves the new document.
12	Refresh This header specifies how soon the browser should ask for an updated page. You can specify time in number of seconds after which a page would be refreshed.
13	Retry-After This header can be used in conjunction with a 503 (Service Unavailable) response to tell the client how soon it can repeat its request.
14	Set-Cookie This header specifies a cookie associated with the page.

The HttpServletResponse Object

The response object is an instance of a **javax.servlet.http.HttpServletResponse object**. Just as the server creates the request object, it also creates an object to represent the response to the client.

The response object also defines the interfaces that deal with creating new HTTP headers. Through this object, the JSP programmer can add new cookies or date stamps, HTTP status codes etc.

The following methods can be used to set HTTP response header in your servlet program. These methods are available with the *HttpServletResponse* object. This object represents the server response.

S.No.	Method & Description
1	String encodeRedirectURL(String url) Encodes the specified URL for use in the sendRedirect method or, if encoding is not needed, returns the URL unchanged.
2	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.
3	boolean containsHeader(String name) Returns a boolean indicating whether the named response header has already been set.
4	boolean isCommitted() Returns a boolean indicating if the response has been committed.
5	void addCookie(Cookie cookie) Adds the specified cookie to the response.
6	void addDateHeader(String name, long date) Adds a response header with the given name and date-value.
7	void addHeader(String name, String value) Adds a response header with the given name and value.
8	void addIntHeader(String name, int value) Adds a response header with the given name and integer value.
9	void flushBuffer() Forces any content in the buffer to be written to the client.
10	void reset() Clears any data that exists in the buffer as well as the status code and headers.
11	void resetBuffer() Clears the content of the underlying buffer in the response without clearing headers or status code.
12	void sendError(int sc) Sends an error response to the client using the specified status code and clearing the buffer.
13	void sendError(int sc, String msg) Sends an error response to the client using the specified status.
14	void sendRedirect(String location) Sends a temporary redirect response to the client using the specified redirect location URL.
15	void setBufferSize(int size) Sets the preferred buffer size for the body of the response.
16	void setCharacterEncoding(String charset) Sets the character encoding (MIME charset) of the response being sent to the client, for example, to UTF-8.
17	void setContentLength(int len) Sets the length of the content body in the response. In HTTP servlets; this method also sets the HTTP Content-Length header.
18	void setContentType(String type) Sets the content type of the response being sent to the client, if the response has not been committed yet.
19	void setDateHeader(String name, long date) Sets a response header with the given name and date-value.
20	void setHeader(String name, String value) Sets a response header with the given name and value.
21	void setIntHeader(String name, int value) Sets a response header with the given name and integer value.

22	void setLocale(Locale loc) Sets the locale of the response, if the response has not been committed yet.
23	void setStatus(int sc) Sets the status code for this response.

HTTP Header Response Example

Following example would use **setIntHeader()** method to set **Refresh** header to simulate a digital clock

```
<% @ page import = "java.io.*,java.util.*" %>

<html>

<head>
<title>Auto Refresh Header Example</title>
</head>

<body>
<center>
<h2>Auto Refresh Header Example</h2>
<%
// Set refresh, autoloading time as 5 seconds
response.setIntHeader("Refresh", 5);

// Get current time
Calendar calendar = new GregorianCalendar();

String am_pm;
int hour = calendar.get(Calendar.HOUR);
int minute = calendar.get(Calendar.MINUTE);
int second = calendar.get(Calendar.SECOND);

if(calendar.get(Calendar.AM_PM) == 0)
    am_pm = "AM";
else
    am_pm = "PM";
String CT = hour+":"+minute+":"+second+" "+am_pm;
out.println("Current Time is: " + CT + "\n");
%>
</center>

</body>
</html>
```

Now put the above code in **main.jsp** and try to access it. This will display the current system time after every 5 seconds as follows. Run the JSP. You will receive the following output: –

Auto Refresh Header Example

Current Time is: 9:44:50 PM

You can try working out on the other methods in a similar way.