Java Servlets 3.0

Lesson 05:The Response Object

Lesson Objectives

- In this lesson, we will learn:
  - Structure of a Response
  - Working with response headers
  - Logging - Reporting Error to Client
  - Setting up Log4j

Lesson Objectives:

This lesson introduces Servlet's Response object. The lesson contents are as follows:

Lesson 05: The response object

       5.1: Structure of a Response

       5.2: Working with Response Headers

       5.3: Logging – Reporting Messages / Error to Client

       5.4: Setting up Log4j

5.1: Structure of a Response
Sending HTML Information

- The Structure of a Response:
  - A server can return three kinds of things to the client
    Status code, any number of HTTP response headers, and a response body.
    - A status code is an integer value that describes the status of the response.
    - Response Header provides HTTP-specific functionality in sending a response. For example, it has methods to access HTTP headers and cookies
    - The response body is main content of the response. A response body can be of any type and of any length. The client knows what to expect by reading and interpreting the HTTP headers in the response.

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

Copyright © Capgemini 2015. All Rights Reserved   3

Structure of a Response:

As part of Servlet lifecycle, we saw how response object is obtained. We saw examples of servlets where we used response object to set response type (response.setContentType()) and obtain the PrintWriter object to send response to browser (response.getWriter()). We shall see the HttpServletResponse object in more detail.

When a web server responds to a request, the response typically consists of a status line, some response headers and the response body itself.

> The status line consists of HTTP version, a status code, and a very short textual message corresponding to the status code.
> The response headers are optional except for "content-type" which specifies the MIME type of the document being sent.
> The response body contains the actual response that the browser lays out.

Servlets can perform a number of important tasks by manipulating the status line and the response headers. For example, Servlets can redirect the user to other sites, indicate the document type (example: image, PDF) to browser and tell the user that authentication is required to access any pages, and so on.

More details on Respo......ext subsequent slides - notes page.

	Response message
Start line	HTTP/1.0 200 OK
Headers	Content-type: text/plain Content-length: 19
Body	Hi! I'm a message!

5.1: Structure of a Response
Concept of Status Code

- public void HttpServletResponse.setStatus(int sc)
  - It sets response status code
  - If a servlet sets a status code that indicates an error during the handling of the request, then it can sendError() instead of setStatus()
  - public void HttpServletResponse.sendError(int sc)
  - public void HttpServletResponse.sendError(int sc, String message)

- Demo: ServletResponseCode.java

Concept of Status Codes:
As mentioned on the previous page, the status line consists of HTTP version, a status code, and a short textual message corresponding to the status code. The HTTP version is set by server and message is directly associated with status code. Therefore a servlet needs to only set the status code.
HTTP status codes are returned by the server to the client to determine the outcome of a request. The status code can indicate success or failure, or it can tell the client software to take further action to finish the request. A status code works behind the scenes and is interpreted by the browser software. If a servlet does not specifically set the status code, the server steps in and sets its value to the default 200 "OK" status.
Status codes are three-digit integer values, the first digit of which specifies one of five classes of response. So we have 1xx: Informational, 2xx: Success, 3xx: Redirection, 4xx: Client Error, and 5xx: Server Error.
Status codes are defined as mnemonic constants in the HttpServletResponse interface. For example: SC_OK for status code 200 and SC_NOT_FOUND for status code 404! For a full list, refer the HttpServletResponse class in the Servlet 3.0 documentation.
A Servlet can use the setStatus() method to set response status code. Remember, the setStatus( ) method should be called before the servlet returns any of its response body. A servlet can use sendError(int sc) (sc indicates the status code to indicate error during handling of request. In this case, the server generates and sends a server-specific page describing the error.)

5.2: Working with Response Headers
Setting an HTTP Header

- The HTTP response headers are as follows:
  - Cache-Control
  - Pragma
  - Refresh
  - Connection
  - Retry-After
  - Expires
  - Location
  - WWW-Authenticate
  - Content-Encoding

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

Copyright © Capgemini 2015. All Rights Reserved   5

Setting Response Headers:
In Lesson-4, we saw how client browser sends HTTP information in the form of request headers. Similarly, a servlet can set response headers to provide more information about its response. HTTP version 1.0 and 1.1 define a large number of request and response headers, description of which is beyond the scope of this course. Some of the most commonly used HTTP headers are listed below:

Cache-Control: It tells all caching mechanisms from server to client whether the browser may cache the object
For example: Cache-Control: no-cache
Pragma: The HTTP 1.0 equivalent of Cache-Control, with no-cache as its only possible value
Connection: It is used to indicate whether the server is willing to maintain an open connection to the client. If so, its value is set to keep-alive.
Retry-After: If an entity is temporarily unavailable, this instructs the client to try again after a specified period of time (in seconds)
For example: Retry-After: 120
Expires: It gives the date/time after which the response is considered stale. For example: Expires: Thu, 01 Dec 2009 16:00:00 GMT
Location: It is used in redirection, or when a new resource has been created. Its value must be a fully qualified URL
WWW-Authenticate: It indicates the authentication scheme that should be used to access the requested entity. WWW-Authenticate: Basic
Content-Encoding: The type of encoding used on the data.
For example: Content-Encoding: gzip
Refresh: It is used in redirection, or when a new resource has been created

5.2: Working with Response Headers
Setting an HTTP Header

- Methods for setting response headers:
  - void HttpServletResponse.setHeader(String name, String value)
  - void ServletResponse.setContentType(String type)
  - void ServletResponse.setContentLength(int len)
  - void HttpServletResponse.addCookie(Cookie cookie)
  - void HttpServletResponse.sendRedirect(String location)

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

Setting Response Headers:
HTTP response headers can be used to give forwarding location, specify cookies, supply the page modification date, instruct the browser to reload the page after a designated interval, designate the type of document being generated, and so on.
Methods for setting arbitrary response headers are as follows:

> The setHeader(String name, String value) method sets the value of the named header as a string. For example:
> res.setHeader("Location", new_location);
> Use the setDateHeader(String name, long date) method, if there is a need to specify a timestamp for a header. It sets the value of the named header with the given name and date-value
> Use the setIntHeader(String name, int value) method to specify an integer value for a header
> The containsHeader() method returns a boolean indicating whether the named response header has already been set

Other than these methods, there are some additional methods to set response headers:

> setContentType: Sets the Content- Type header
> setContentLength: Sets the Content- Length header
> addCookie: Adds a value to the Set- Cookie header
> sendRedirect: Changes status code to redirection i.e. 300 and a specifies the client to request another page
> (Refer to Slide 6.2 – Client-side dispatch for further details on sendRedirect)

5.2: Working with Response Headers
Setting an HTTP Header

- Request can be redirected by either:
  - response.setStatus(res.SC_MOVED_TEMPORARILY);
    response.setHeader("Location", site);
  - response.sendRedirect(site);
- sendRedirect() syntax:
  - public void HttpServletResponse.sendRedirect(String location)
    - This method creates a roundtrip to the client and hence a new request for server.

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING
Copyright © Capgemini 2015. All Rights Reserved   7

Redirecting a Request:
One of the useful things a servlet can do using status codes and a header is redirecting a request. This is done by sending instructions to the client, to use another URL in the response. Redirection is generally used when a document moves (to send the client to the new location), for load balancing (so one URL can distribute the load to several different machine), or for simple randomization (choosing a destination at random).
The actual redirection happens in two lines:
        res.setStatus(res.SC_MOVED_TEMPORARILY);
        res.setHeader("Location", site);
The first line sets the status code to indicate a redirection is to take place, while the second line gives the new location.
These two lines can be simplified to one using the sendRedirect( ) convenience method:
        public void HttpServletResponse.sendRedirect(String location)
        throws IOException.
This method redirects the response to the specified location, automatically setting the status code and location headers. For our example, the two lines simply become:
        res.sendRedirect(site);

Demo

- Execute the SiteSelector servlet that performs a random redirect, sending a client to a random site selected from its site list.

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

Copyright © Capgemini 2015. All Rights Reserved   8

Note:
Refer to com.igate.ch4.SiteSelector.java class. A partial listing is shown below:
Invoke this servlet as:
        http://localhost:9090/RequestResponseDemoApp/SiteSelector
The servlet randomly picking one site from the vector list and displays.

5.2: Working with Response Headers
Using the "Refresh" Response Header

- ClientPull:
  - response.setHeader("Refresh", "3") .This method tells the client(browser) to reload same servlet regularly for three seconds
  - respose.setHeader("Refresh", "3;URL=url"): This method tells client to load page at URL url after three seconds.

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

ClientPull:

Client pull is similar to redirection, with one major difference – the browser actually displays the content from the first page and waits for some specified amount of time before retrieving and displaying the content from the next page. It is called client pull because the client is responsible for pulling the content from the next page.

There are two reasons for using client pull.

First, to communicate to the client that requested page has been moved before the next page is automatically loaded

Second, pages can be retrieved in sequence, making it possible to present a slow-motion page animation.

Client pull information is sent to the client using the Refresh HTTP header. This header's value specifies the number of seconds to display the page before pulling the next one, and it optionally includes a URL string that specifies the URL from which to pull. If no URL is given the same URL is used. Some examples follow:

A call to setHeader() that tells the client to reload this same servlet after showing its current content for three seconds:
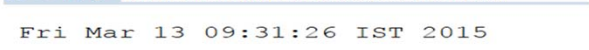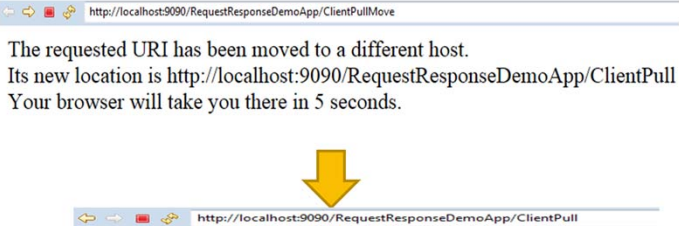
setHeader("Refresh", "3");

setHeader("Refresh", "3;URL=https://ispace.igate.com");

method call tells client to display IGATE's home page after three seconds.

Demo

- Execute the ClientPull servlet that uses client pull to display the current time, updated every 10 seconds.
- Use the ClientPullMove servlet to redirect request to another URL.

Demo

http://localhost:9090/RequestResponseDemoApp/ClientPullMove

The requested URI has been moved to a different host.
Its new location is http://localhost:9090/RequestResponseDemoApp/ClientPull
Your browser will take you there in 5 seconds.

http://localhost:9090/RequestResponseDemoApp/ClientPull

Fri Mar 13 09:31:26 IST 2015

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

Note:
Refer the following classes:
        com.igate.ch4.ClientPull.java
        com.igate.ch4.ClientPullMove.java classes.
A partial listing of both these servlets is shown below:

Invoke the ClientPullMove servlet as:
        http://localhost:9090/RequestResponseDemoApp/ ClientPullMove
See the servlet ClientPullMove redirecting to ClientPull servlet after 5 seconds. The ClientPull servlet displays updated time after every 10 seconds.

5.3: Logging - Reporting Messages / Error to Client
Concept of Logging

- Servlets have the ability to write their actions and their errors to a log file using the log() method:
  - public void ServletContext.log(String msg)
  - public void ServletContext.log(String msg , Throwable t)

```
try {
       // code to read the file contents
    }
   catch (IOException e)  {
     log(" could not find file: "+ e.getMessage());
     res.sendError(res.SC_NOT_FOUND);    }
}
```

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING
Copyright © Capgemini 2015. All Rights Reserved   11

Logging:
Logging of messages is a common requirement in all applications. In a server environment, it is a critical feature due to the distributed multi-user interaction that is characteristic of a server. Many users interact simultaneously with an application server and some degree of logging of the interactions is essential for support.
The log() method aids debugging by providing a way to track a servlet's actions. It also offers a way to save a complete description of any errors encountered by the servlet. The description can be the same as the one given to the client, or it can be more exhaustive and detailed.
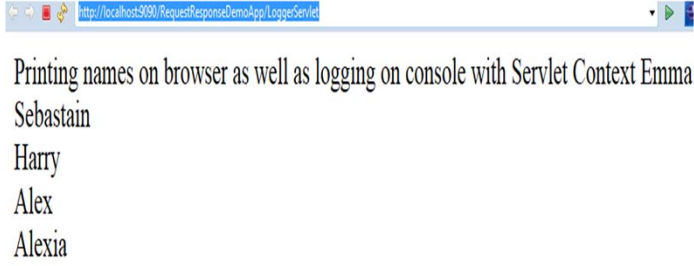The single-argument method writes the given message to a servlet log, which is usually event log file. The two-argument version writes an explanatory message and a stack trace for a given Throwable exception to the servlet log file. The name and type of the servlet log file is specific to the servlet container, usually an event log.
Refer to the code snippet in the above slide.
> In the catch block, log() is used to record on the server, when requested files do not exist, while returning a simple "404 Not Found" page to the client.

Note:
Refer the com.igate.ch4.LoggerServlet.java class. A partial listing is shown below:
Invoke this servlet as:
        http://localhost:9090/RequestResponseDemoApp/LoggerServlet
The servlet checks for the filename to be read and prints out the contents. For WildFly, this is the server.log file found in the wildfly-8.1.0.Final\standalone\log folder. The output for this servlet is as seen in the figure above.
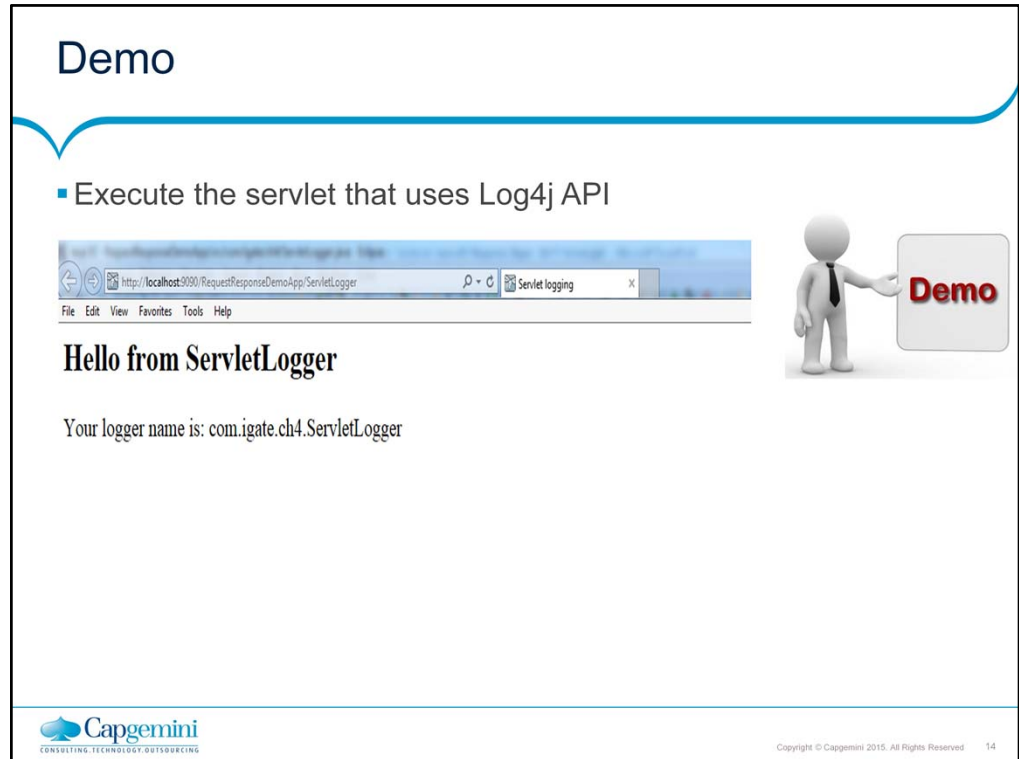
The partial contents of the log file are: This shows servlet has logged the usernames in server.log file.

2015-03-16 05:23:00,940 INFO [io.undertow.servlet] (default task-1) LoggerServlet: Servlet context logging the users>>>Emma
2015-03-16 05:23:00,940 INFO [io.undertow.servlet] (default task-1) LoggerServlet: Servlet context logging the users>>>Sabastian
2015-03-16 05:23:00,950 INFO [io.undertow.servlet] (default task-1) LoggerServlet: Servlet context logging the users>>>Harry
2015-03-16 05:23:00,950 INFO [io.undertow.servlet] (default task-1) LoggerServlet: Servlet context logging the users>>>Alex
2015-03-16 05:23:00,950 INFO [io.undertow.servlet] (default task-1) LoggerServlet: Servlet context logging the users>>>Alexia

5.4: Setting up Log4j
Log4j with Servlet

- Log4j can be setup with Servlets for standard logging facility
- Need to refer to Log4j package under Apache Software License
- Need to place the jar file (log4j.jar) under WEB-INF directory of the web application created
- Create a Log4j properties file and place it in the web application's WEB-INF directory
  - This is typically a text file with name / value pairs for configuring log4j elements like loggers, appenders and layouts
- In servlets logger could be used

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

Refer the com.igate.ch4.ServletLogger.java class. A partial listing is shown below:

Invoke this servlet as follows:
    http://localhost:9090/RequestResponseDemoApp/ServletLogger
The servlet first executes the init() method and logs the message to both – console and log.out file (under D:\usr\home\log4j). It then logs an info and a debug level message. The log.out file will contain both, but console will only show the info-level message. The output for this servlet is as seen in the figure above.

Summary

- In this lesson, we have learnt:
  - Structure of a Response
  - Setting response headers
  - Logging - Reporting message / Error to Client
  - Configuring server to report errors into a log file (Using log4j tool)

Summary

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

Review Questions

- Question 1: The sendRedirect method defined in the HttpServlet class is equivalent to invoking the setStatus method with the following parameter and a Location header in the URL.

  - Option 1: SC_OK
  - Option 2: SC_MOVED_TEMPORARILY
  - Option 3: SC_NOT_FOUND
  - Option 4: SC_INTERNAL_SERVER_ERROR
  - Option 5: SC_BAD_REQUEST

Review Questions

- Question 2: Browsers that support content encoding feature, indicate that they do so by setting the ___ request header.
  - Option 1: Cache-Control
  - Option 2: Encoding-true
  - Option 3: Accept-Encoding
  - Option 4: Content-Encoding

- Question 3: The ___ method aids debugging by providing a way to track a servlet's actions.
  - Option 1: debug()
  - Option 2: log()
  - Option 3: logFile()

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING