B.Sc. In Software Development. Year 4. Enterprise Development. Semester I. Working With HTTP Requests and Responses

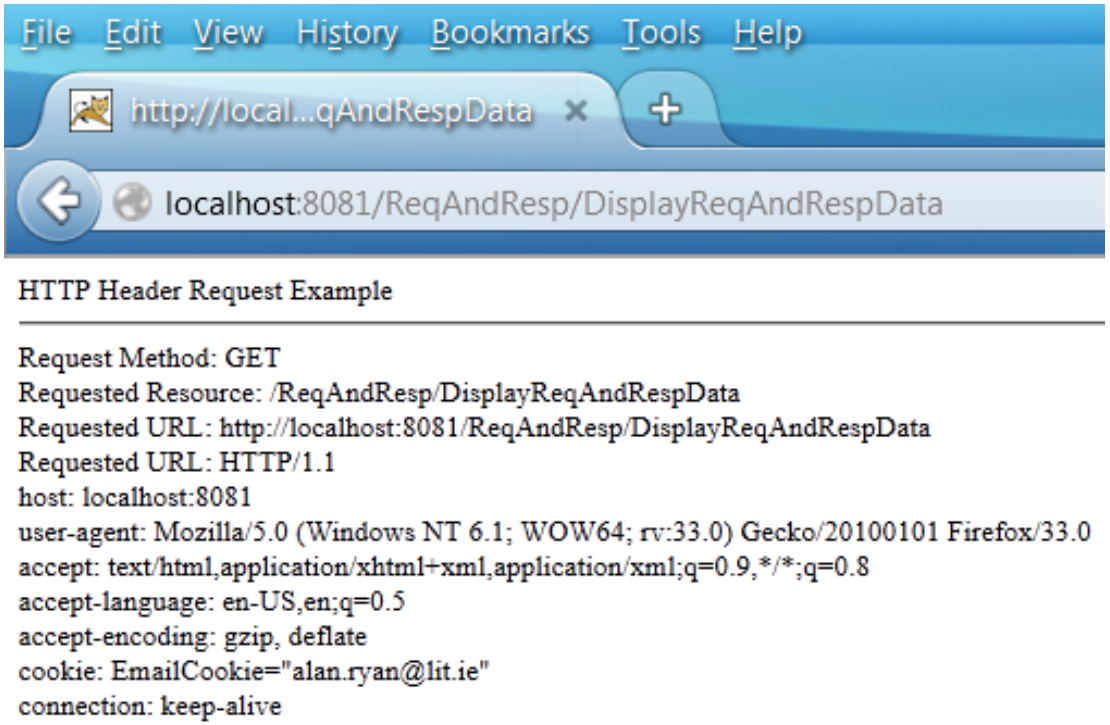
LIMERICK INSTITUTE OF TECHNOLOGY
SCHOOL OF SCIENCE, ENGINEERING & I.T.
Department of Information Technology

Introduction

- When you write Servlets and JSP's, the classes and methods of the servlet API shelter you from having to work directly with HTTP.
- Sometimes you need to know more about HTTP requests & responses.
  - You need to use the methods of the servlet API to work with them.

Introduction

- The following is an example of a typical HTTP request that has been made to a servlet using Firefox.



Introduction

- After the response headers, a HTTP response contains a blank line, followed by the response entity or response body.
- In this example, the response entity is a HTML document but it could be an XML document, plain text, tab-delimited text, an image, a PDF file, a sound file, a video file etc….

Example of manipulating HTTP Response

- What if you wanted to refresh a page every five seconds?
- Can be done by manipulating a HTTP Response.

```
protected void processRequest(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException {

    // Set refresh, autoload time as 5 seconds
    response.setIntHeader("Refresh", 5);

    // Set response content type
    response.setContentType("text/html");

    Date d = new Date();

    PrintWriter out = response.getWriter();
    out.println("<h3>Auto Refresh Page Using Header Setting</h3><br>");
    out.println(d);

}
```

Common MIME Types

- MIME (Multi-purpose Internet Mail Extensions) types form a standard way of classifying file types on the Internet.
- Web servers and browsers all have a list of MIME types, so that they can transfer files of the same type in the same way, no matter what operating system they are working in.
- A MIME type has two parts: a type and a subtype. They are separated by a slash (/).
    - The MIME type for Microsoft Word files is application and the subtype is msword. Together, the complete MIME type is application/msword.

Common MIME Types

- Can be used in the accept header of a request or the content-type header of a response.

Type/Subtype	Type/Subtype
text/plain	application/msword
text/html	application/ms-excel
text/css	application/pdf
text/xml	application/zip
image/gif	application/x-java-archive
image/jpeg	audio/mpeg
image/png	audio/wav
image/x-bitmap	video/mpeg

Common HTTP **Request** Headers

- The following are the most common HTTP response headers
- The vast majority of the time a browser automatically sets these headers when it makes a request.
- When the server receives the request it can examine them to learn more about the browser.
- Servlets can also set these too.

Common HTTP **Request** Headers

Name	Description
accept	The preferred order of MIME types that the browser can accept.
accept-language	The language (code) that the browser prefers.
cookie	Specifies any cookies that were sent by the server
host	The host and port of the machine that sent the request
referer	The URL of the originating page
user-agent	Indicates the type of browser the client is using.

How to Work With the **Request**

The following methods can be used to get any of the headers in an HTTP request.

`int`	`getIntHeader(String name)` Returns the value of the specified request header as an `int`.
`String`	`getHeader(String name)` Returns the value of the specified request header as a `String`.
`long`	`getDateHeader(String name)` Returns the value of the specified request header as a `long` value that represents a `Date` object.
`Enumeration<String>`	`getHeaders(String name)` Returns all the values of the specified request header as an `Enumeration` of `String` objects.
`Enumeration<String>`	`getHeaderNames()` Returns an enumeration of all the header names this request contains.

How to Work With the **Request**

```
139     //check the mime type accepted by the browser
140     String mimeTypes = request.getHeader("Accept");
141     if (mimeTypes.indexOf("image/png") > -1)
142         returnPNG();
143     else
144         returnGIF();
145
146     //check the browser type
147     String browser = request.getHeader("User-Agent");
148     if (browser.indexOf("MSIE") > -1)
149         doIECode();
150     else
151         doFirefoxCode();
```

http://www.useragentstring.com/pages/useragentstring.php

How to Work With the **Request**

Useful methods for working with the request.

`Cookie[]`	`getCookies()` Returns an array containing all of the `Cookie` objects the client sent with this request.
`String`	`getAuthType()` Returns the name of the authentication scheme used to protect the servlet.
`String`	`getContentType()` Returns the MIME type of the body of the request, or `null` if the type is not known.

Common HTTP Status Codes

- The most common status codes:
  - For successful requests, the server typically returns a 200 (OK) status code.
  - If the server can’t find the requested file it typically returns (the dreaded) 404 - Not Found - status code.
  - If the server encounters an error trying to retrieve the file it may return a 500 - Internal Server Error - status code.

HTTP Categories of Status Codes

Number	Type	Description
100-199	Informational	The req was received and is being processed
200-299	Success	The req was made successfully
300-399	Redirection	Further action I required to fulfil this request.
400-499	Client error	The req contains an error
500-599	Server error	The server has encountered an error

Common HTTP Status Codes

Number	Type
200	OK
301	Moved Permanently
302	Found
400	Bad Request
404	Not found
500	Internal Server Error

https://www.steveschoger.com/status-code-poster/img/status-code.png

Common HTTP **Response** Headers

- Most of the time the server automatically sets these response headers when it returns the response.
- There may be time when you wan to modify these headers yourself (see the example earlier in these notes on refreshing a web page).
- You could also use these headers to control how your web server caches a response (an example of doing this will follow later).

Common HTTP **Response** Headers

Number	Type
cache-Control	Controls when and how a browser caches a page
content-disposition	Used to specify that the response contains a binary file
content-length	The length of the response
expires	The time that the page should no longer be cached
last-modified	The time the document was last modified
refresh	The number of seconds the browser should wait before requesting a new copy

How to Work With the **Response**

- The following shows how to use the fields and methods of the response object to set the data that's contained in an HTTP response.

`void`	`setStatus(int sc)` Sets the status code for this response.

- The status codes map to fields of the response object.

  200 (ok) => SC_OK

  404 (not found) => SC_NOT_FOUND

  XXX (xxx xxxx) => SC_XXX_XXXX

  ```
  response.setStatus(404);
  response.setStatus(response.SC_NOT_FOUND);
  ```

How to Work With the **Response**

- Most of the time the web server automatically sets the status code for an HTTP response.
- If you need to set the status code you can use the setStatus method.
- To specify the value for this code, you can use either an integer value or one of the fields of the response object.

How to Work With the **Response**

Useful methods for working with the response.

void	`setDateHeader(String name, long date)` Sets a response header with the given name and date-value.
void	`setHeader(String name, String value)` Sets a response header with the given name and value.
void	`setIntHeader(String name, int value)` Sets a response header with the given name and integer value.
void	`setLocale(Locale loc)` Sets the locale of the response, if the response has not been committed yet.
void	`setContentType(String type)` Sets the content type of the response being sent to the client, if the response has not been committed yet.
void	`setContentType(String type)` Sets the content type of the response being sent to the client, if the response has not been committed yet.

```
161     response.setContentType("text/html");
162     response.setContentLength(403);
163     response.setIntHeader("refresh", 30);
164     response.setDateHeader("expires", currentDate.getTime + 60 * 60 + 1000);
```

Practical Skills: Example 1 – *Returning a tab delimited file as an Excel Spreadsheet*

- Most browsers can use Excel to read tab-delimited text.
- As a result, to display tab-delimited text as a spreadsheet, you can create some tab-delimited text, set the Content-Type to “application/vnd.ms-excel” and return the tab-delimited text to the browser.
- The browser will then endeavour to load the file in excel.
- A partial code listing for this example (a Servlet) appears on the next slide).

Practical Skills: Example 1 – *Returning a tab delimited file as an Excel Spreadsheet*

```
34          PrintWriter out = response.getWriter();
35          try {
36              User u1 = new User("Alan", "Ryan", "alan.ryan@lit.ie");
37              User u2 = new User("Brendan", "Watson", "brendan.watson@lit.ie");
38              User u3 = new User("Gerry", "Guinane","Gerry.Guinane@lit.ie" );
39              User u4 = new User("Mike", "Winterburn","Mike.Winterburn@lit.ie" );
40              User u5 = new User("Pat", "Donohue","Pat.Donohue@lit.ie" );
41
42              ArrayList<User> list = new ArrayList();
43
44              list.add(u1); list.add(u2); list.add(u3); list.add(u4); list.add(u5);
45
46              String report = "The user table\n\n" +
47                              "First Name\t" +
48                              "Last Name\t" +
49                              "Email Address\n";
50
51              for (User aUser : list) {
52                  report += aUser.getFirstName() + "\t" + aUser.getLastName() + "\t" + aUser.getEmailAddress() + "\n";
53              }
54
55              response.setContentType("application/vnd.ms-excel");
56              response.setHeader("cache-control", "no-cache");
57              out.println(report);
```

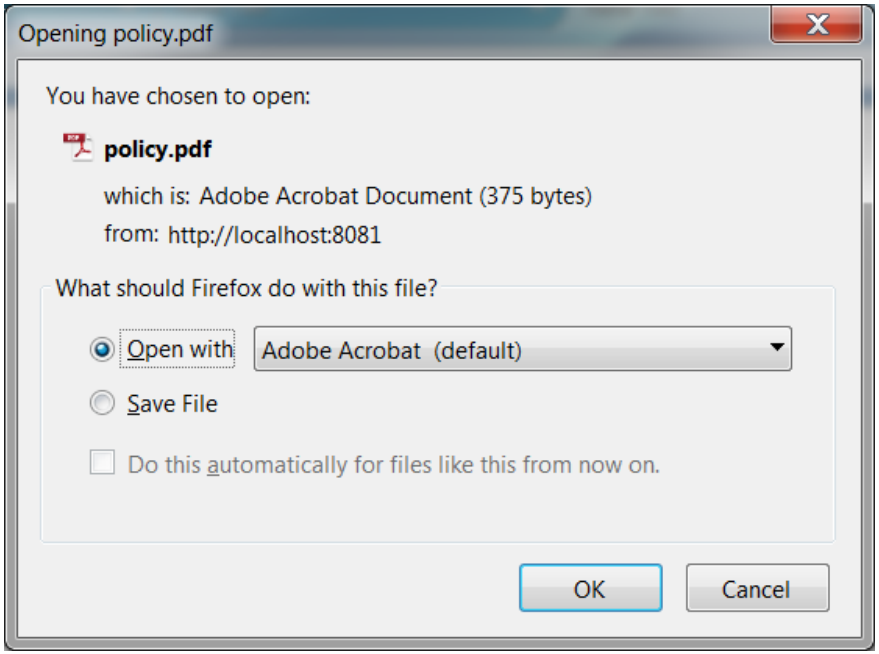
Practical Skills: Example 2 – *Allowing a user to download a file with a download dialog box.*

- Within a web application you can code a HTML link that points to a downloadable file.
- For example, you can code a link that points to a PDF file.
- When the user clicks on the link, most browsers automatically try to display the PDF within the Adobe Reader.
  - This is more than adequate for most applications.
  - However the reader doesn't indicate how long it will take to open the file.
  - This means the user can be left staring at a blank screen for several minutes while the file downloads.
- Another option is to display a File Download dialog box.
  - That way the user will have the choice of opening the PDF or saving it to disk.
  - The dialog box will automatically indicate the progress of the download.

Practical Skills: Example 2 – *Allowing a user to download a file with a download dialog box.*

View our policy → Opens the file in Adobe Reader

Download our policy here



Practical Skills: Example 2 – *Allowing a user to download a file with a download dialog box.*

Code for the HTML file that provides the two links to my policy document.

```
13      <body>
14
15          <a href="policy.pdf"> View our policy</a>
16          <br><br><br>
17          <a href="downloadFile?name=policy.pdf"> Download our policy here </a>
18
19      </body>
```

Practical Skills: Example 2 – *Allowing a user to download a file with a download dialog box.*

Code for the Servlet

```
31      protected void processRequest(HttpServletRequest request, HttpServletResponse response)
32              throws ServletException, IOException {
33
34          try {
35             ServletContext sc = getServletContext();
36             String path = sc.getRealPath("/");
37             String fileName = request.getParameter("name");
38
39             response.setContentType("application/octet-stream");
40             response.setHeader("content-disposition", "attachment; filename=" + fileName);
41
42             FileInputStream in = new FileInputStream(path + fileName);
43
44             PrintWriter out = response.getWriter();
45
46              int i = in.read();
47              while ( i != 1) {
48                  out.write(i);
49                  i = in.read();
50              }
51              in.close();
52              out.close();
53
54          }//end try
            catch(Exception ex) {
56              System.out.println(ex);
57          }
```

References

Murach, J., (2014) *MurachsJava Servlets JSP*, 3rd edn. Mike Murach and Associates, Inc.

https://developer.mozilla.org/en-US/docs/Web/HTTP/Basics_of_HTTP/MIME_types/Complete_list_of_MIME_types