Generating the Server Response: HTTP Response Headers
Agenda

• Format of the HTTP response
• Setting response headers
• Understanding what response headers are good for
• Building Excel spread sheets
• Generating JPEG images dynamically
• Sending incremental updates to the browser
HTTP Request/Response

• Request

GET /servlet/SomeName HTTP/1.1
Host: ...
Header2: ...
...
HeaderN: (Blank Line)

• Response

HTTP/1.1 200 OK
Content-Type: text/html
Header2: ...
...
HeaderN: ...
  (Blank Line)
<!DOCTYPE ...
<html>
<head>...</head>
<body>
  ...
</body></html>
Setting Arbitrary Response Headers

- The most general way to specify headers is to use the `setHeader` method of the `HttpServletResponse` class
  - `public void setHeader(String headerName, String headerValue)`
- Two specialized methods set headers with dates and integers
  - `public void setDateHeader(String name, long millisecs)`
    - Converts milliseconds since 1970 to a date string in GMT format
  - `public void setIntHeader(String name, int headerValue)`
    - Prevents need to convert int to String before calling `setHeader`
- `addHeader`, `addDateHeader`, `addIntHeader`
  - Adds new occurrence of header instead of replacing
Setting Common Response Headers

- **setContentType (String mimeType)**
  - Sets the Content-Type header (MIME types)
- **setContentLength (int length)**
  - Sets the Content-Length header (number of bytes in the response), which is useful if the browser supports persistent HTTP connections.
- **addCookie (Cookie c)**
  - Adds a value to the Set-Cookie header.
- **sendRedirect (String address)**
  - Sets the Location header (plus changes status code).
Common HTTP 1.1 Response Headers

- **Cache-Control (1.1) and Pragma (1.0)**
  - A no-cache value prevents browsers from caching page.

- **Content-Disposition**
  - Lets you request that the browser ask the user to save the response to disk in a file of the given name
    
    ```
    Content-Disposition: attachment;
    filename=file-name
    ```

- **Content-Encoding**
  - The way document is encoded

- **Content-Length**
  - The number of bytes in the response.
  - Use `ByteArrayOutputStream` to buffer document before sending it, so that you can determine size.
Common HTTP 1.1 Response Headers (Continued)

- **Content-Type**
  - The MIME type of the document being returned.

- **Expires**
  - The time at which document should be considered out-of-date and thus should no longer be cached.
  - Use setDateHeader to set this header.

- **Last-Modified**
  - The time document was last changed.
  - Use the getLastModified method instead
Common HTTP 1.1 Response Headers (Continued)

• **Location**
  - The URL to which browser should reconnect.
  - Use sendRedirect instead of setting this directly.

• **Refresh**
  - The number of seconds until browser should reload page.
    Can also include URL to connect to.

• **Set-Cookie**
  - The cookies that browser should remember. Don’t set this header directly; use addCookie instead.
# Common MIME Types

<table>
<thead>
<tr>
<th>Type</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>application/msword</td>
<td>Microsoft Word document</td>
</tr>
<tr>
<td>application/octet-stream</td>
<td>Unrecognized or binary data</td>
</tr>
<tr>
<td>application/pdf</td>
<td>Acrobat (.pdf) file</td>
</tr>
<tr>
<td>application/postscript</td>
<td>PostScript file</td>
</tr>
<tr>
<td>application/vnd.ms-excel</td>
<td>Excel spreadsheet</td>
</tr>
<tr>
<td>application/vnd.ms-powerpoint</td>
<td>Powerpoint presentation</td>
</tr>
<tr>
<td>application/x-gzip</td>
<td>Gzip archive</td>
</tr>
<tr>
<td>application/x-java-archive</td>
<td>JAR file</td>
</tr>
<tr>
<td>application/x-java-vm</td>
<td>Java bytecode (.class) file</td>
</tr>
<tr>
<td>application/zip</td>
<td>Zip archive</td>
</tr>
<tr>
<td>audio/basic</td>
<td>Sound file in .au or .snd format</td>
</tr>
<tr>
<td>audio/x-aiff</td>
<td>AIFF sound file</td>
</tr>
<tr>
<td>audio/x-wav</td>
<td>Microsoft Windows sound file</td>
</tr>
<tr>
<td>audio/midi</td>
<td>MIDI sound file</td>
</tr>
<tr>
<td>text/css</td>
<td>HTML cascading style sheet</td>
</tr>
<tr>
<td>text/html</td>
<td>HTML document</td>
</tr>
<tr>
<td>text/plain</td>
<td>Plain text</td>
</tr>
<tr>
<td>text/xml</td>
<td>XML document</td>
</tr>
<tr>
<td>image/gif</td>
<td>GIF image</td>
</tr>
<tr>
<td>image/jpeg</td>
<td>JPEG image</td>
</tr>
<tr>
<td>image/png</td>
<td>PNG image</td>
</tr>
<tr>
<td>image/tiff</td>
<td>TIFF image</td>
</tr>
<tr>
<td>video/mpeg</td>
<td>MPEG video clip</td>
</tr>
<tr>
<td>video/quicktime</td>
<td>QuickTime video clip</td>
</tr>
</tbody>
</table>
Building Excel Spreadsheets

• Though servlets usually generate HTML output, other types of output are possible
• Microsoft Excel content can be generated so that the features of Excel can be exploited
• The key is to include the following code

```java
response.setContentType("application/vnd.ms-excel");
PrintWriter out = response.getWriter();
```

• Example will generate output in tab-separated format (include \t in the output strings)
Building Excel Spreadsheets

```java
public class ApplesAndOranges extends HttpServlet {
    public void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
        response.setContentType("application/vnd.ms-excel");
        PrintWriter out = response.getWriter();
        out.println("Q1\tQ2\tQ3\tQ4\tTotal");
        out.println("Apples\t78\t87\t92\t29\t=SUM(B2:E2)" );
        out.println("Oranges\t77\t86\t93\t30\t=SUM(B3:E3)" );
    }
}
```
Building Excel Spreadsheets

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
<th>F</th>
<th>G</th>
<th>H</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td></td>
<td>Q1</td>
<td>Q2</td>
<td>Q3</td>
<td>Q4</td>
<td>Total</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>Apples</td>
<td>78</td>
<td>87</td>
<td>92</td>
<td>29</td>
<td>286</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>Oranges</td>
<td>77</td>
<td>86</td>
<td>93</td>
<td>30</td>
<td>286</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Image: An Excel spreadsheet showing data for apples and oranges in different quarters.
Requirements for Handling Long-Running Servlets

• What to do if a calculation requires a long time to complete (20 seconds) or whose results change periodically

• Store data between requests.
  – For data that is not specific to any one client, store it in a field (instance variable) of the servlet.
  – For data that is specific to a user, store it in the HttpSession object
  – For data that needs to be available to other servlets or JSP pages (regardless of user), store it in the ServletContext

• Keep computations running after the response is sent to the user.
  – Start a Thread but set the thread priority to a low value so that it does not slow down the server.
Requirements for Handling Long-Running Servlets

• Send updated results to the browser when they are ready.
  – Browser does not maintain and open connection to the server. Use Refresh header to tell browser to ask for updates
    
    ```java
    if (!isLastResult) {
        response.setIntHeader("Refresh", 5);
    }
    ```

Using Servlets to Generate JPEG Images

1. Create a BufferedImage
2. Draw into the BufferedImage
3. Set the Content-Type response header
   ```java
   response.setContentType("image/jpeg");
   ```
4. Get an output stream
   ```java
   OutputStream out = response.getOutputStream
   ```
5. Send the BufferedImage in JPEG format to the output stream
   ```java
   try {
       ImageIO.write(image, "jpg", out);
   } catch (IOException ioe) {
       System.err.println("Error writing JPEG file: " + ioe);
   }
   ```