Server-side
Web Programming

Lecture 15:
The Request and Response Objects

Http Requests and Responses

- **request** object
  - Properties of browser
  - IP address and host name of referring machine
    - `request.getRemoteAddr()`
    - `request.getHost()`
    - Not particularly useful for identification (too easy to fake)

- **response** object
  - Can be used to tell browser more than just html page to display
  - Format to display response page, etc.
Http Requests and Responses

An HTTP request
GET http://www.murach.com/email/join_email_list.html HTTP/1.1
connection: Keep-Alive
user-agent: Mozilla/4.61 [en] (Win98; I)
host: www.murach.com
accept: image/gif, image/x-xbitmap, image/jpeg, image/pjpeg, image/png, */*
accept-encoding: gzip
accept-language: en
cookie: emailCookie=jsmith%40hotmail.com; userID=39210

An HTTP response
HTTP/1.1 200 OK
date: Sat, 17 Aug 2002 10:32:54 GMT
server: Apache/1.3.6 (Unix) PHP/3.0.7
content-type: text/html
content-length: 201
last-modified: Fri, 16 Aug 2002 12:52:09 GMT

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
<head>
<title>Chapter 4 - Email List application</title>
</head>
<body>
<h1>Join our email list</h1>
</body>
</html>

Requests

- Contains information about browser that submitted request
- Main components:
  - **Referrer**: Page from which request was submitted
  - **Accept**: Preferred order of MIME types accepted by browser
  - **Accept-Encoding**: Types of compression understood by browser
    - gzip, etc.
  - **Accept-Language**: Language codes for accepted languages
    - “en”, “en-us”, etc.
  - **User-Agent**: Browser type
    - Long string containing identifiers specific to browser
      - “MSIE”, etc.
MIME Types

- **Multipurpose Internet Mail Extensions:**
  - Formats for transmitting data via email / internet
    - Text formats
    - Image formats
    - Application formats (programs browser can run to display page)
    - Audio and video multimedia formats

- Can use */* to indicate that accept anything (usually last resort)

<table>
<thead>
<tr>
<th>Type/Subtype</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>text/plain</td>
<td>Plain text document</td>
</tr>
<tr>
<td>text/html</td>
<td>HTML document</td>
</tr>
<tr>
<td>text/css</td>
<td>HTML cascading style sheet</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>image/x-xbitmap</td>
<td>Windows bitmap image</td>
</tr>
<tr>
<td>application/msword</td>
<td>Microsoft Word document</td>
</tr>
<tr>
<td>application/vnd.ms-excel</td>
<td>Microsoft Excel spreadsheet</td>
</tr>
<tr>
<td>application/pdf</td>
<td>Adobe Acrobat file</td>
</tr>
<tr>
<td>application/postscript</td>
<td>PostScript file</td>
</tr>
<tr>
<td>application/zip</td>
<td>Zip file</td>
</tr>
<tr>
<td>application/x-java-archive</td>
<td>Jar file</td>
</tr>
<tr>
<td>application/x-gzip</td>
<td>Gzip file</td>
</tr>
<tr>
<td>application/octet-stream</td>
<td>Binary data</td>
</tr>
<tr>
<td>audio/basic</td>
<td>A sound file (usually in the *.au or *.snd format)</td>
</tr>
<tr>
<td>video/mpeg</td>
<td>MPEG video clip</td>
</tr>
</tbody>
</table>

Accessing Request Properties

- Can get these properties using `request.getHeader(headername)`

- Example:
  ```java
  String browser =
    request.getHeader("Accept-Encoding");
  ```

  might return “gzip, deflate” for example

- Main use: Customizing response to abilities of browser
  - Only send information over if form browser can handle!

- Can use `request.getHeaderNames()` to get list of all property names sent over from browser
Accessing Request Properties

```html
<h4>Headers:</h4>
<table border="1">
  <tr>
    <th>Name</th>
    <th>Value</th>
  </tr>
  <tr>
    <td>accept</td>
    <td>image/gif, image/x-xbitmap, image/jpeg, image/png, application/vnd.ms-powerpoint, application/vnd.ms-excel, application/vnd.ms-word, application/x-shockwave-flash, */*</td>
  </tr>
  <tr>
    <td>referer</td>
    <td>http://localhost:8080/Request/</td>
  </tr>
  <tr>
    <td>accept-language</td>
    <td>en-us</td>
  </tr>
  <tr>
    <td>accept-encoding</td>
    <td>gzip, deflate</td>
  </tr>
  <tr>
    <td>user-agent</td>
    <td>Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1; SV1; .NET CLR 1.1.4322; .NET CLR 1.0.3705; .NET CLR 2.0.50727)</td>
  </tr>
  <tr>
    <td>host</td>
    <td>localhost:8080</td>
  </tr>
  <tr>
    <td>connection</td>
    <td>Keep-Alive</td>
  </tr>
  <tr>
    <td>cookie</td>
    <td>SESSIONID=61FC47B8356B328A57FDC5735148D91C</td>
  </tr>
  <tr>
    <td>x-nanocat</td>
    <td>v1.2</td>
  </tr>
</table>
```

Accessing Request Properties

![Image of request headers and requester information]
Using Request Properties

- Example: Sending custom image types
  - Send `.png` image if supported
  - Send `.jpg` image otherwise

```java
String imagetypes = request.getHeader("Accept");
boolean acceptsPng = imagetypes.contains("PNG");
if (acceptsPng) {
    // insert link to .png image
} else {
    // insert link to .jpg image
}
```

- Example: Customizing response to browser type
  - Will contain the string "MSIE" if Internet Explorer used

```java
String browser = request.getHeader("User-Agent");
boolean isIE = browser.contains("MSIE");
if (isIE) {
    // forward to IE specific page
} else {
    // forward to general response page
}
```
Response Properties

• Can set properties of response

```plaintext
HTTP/1.1 200 OK
date: Sat, 17 Aug 2002 10:32:54 GMT
server: Apache/1.3.6 (Unix) PHP/3.0.7
content-type: text/html
content-length: 201
last-modified: Fri, 16 Aug 2002 12:52:09 GMT
```

• Useful type to set: Content type
  – Form in which browser should display information sent
  – Default: text/html (standard html format)
  – Should first examine request to make sure that form is supported!

Setting Content Type

• Syntax: `response.setContentType("MIME type")`;

• Example: forcing browser to display response as Excel spreadsheet
  – `response.setContentType("application/vnd.ms-excel")`;
  – Send response back in simple format:
    • Cells in same row separated by tab `\t`
    • Move to next row with return `\n`
  – Write that string to `response` object using `PrintWriter` (like old style response page)
  – Much more efficient than sending an entire spreadsheet as file!
Setting Content Type

```java
protected void processRequest(HttpServletRequest request, HttpServletResponse response)
    throws ServletException, IOException {
    String quantity = request.getParameter("quantity");
    String name = request.getParameter("name");
    String email = request.getParameter("email");

    // Create a string that will be converted to the Excel spreadsheet.
    // Cells on the same row are separated by '\t' while printing a '\n'
    // moves to the next row.

    String result = "Name\t"+name+"\nEmail\t"+email+"\nQuantity\t"+quantity;

    // Set the content type to force the browser to display as an Excel spreadsheet.
    response.setContentType("application/vnd.ms-excel");

    // Create a PrintWriter to send the string to the response object.
    PrintWriter out = response.getWriter();
    out.println(result);
}
```

Controlling Caching

- For efficiency, most browsers cache pages received from server
  - Stored in local memory
- Next time user requests page, check to see whether in cache before downloading again
- Problem for pages that change regularly
  - Stock price pages, etc.

- Can force browser to remove page after certain interval of time
  - Browser will then download current version of page
- Syntax:
  `response.setHeader("cache-control", "no-cache");`
Forcing Page Refresh

• Can force browser to refresh page after certain interval of time
  – Gamecasts, etc.

• Syntax:
  `response.setIntHeader("refresh", time in seconds);`

• Example:
  `response.setIntHeader("refresh", 60);`