Static Content Optimization

Web Engineering ITEC-518
BS(IT) P-III
21st March 2016

Instructor:
M Bilal Shaikh
IICT, University of Sindh

Content

- Dynamic - Changes based on inputs from users and from external sources (e.g., databases)
- Static - Fixed or pre-generated. Examples: PDFs, Flash, fonts, audio, video, JavaScript, CSS
- Generating dynamic content is more expensive than serving static content
- The application and database tiers are only utilized when generating dynamic content...
- ...thus, application and database optimizations only increase performance for dynamic content
- However, the vast majority of content in a web application, both in terms of number of objects and in terms of size, is static content!
- Delivering static content to the browser > 85% of page load time!

Basic Optimization Methods

- Reduce the size of content to deliver
  - Compress content
  - What to compress: HTML, JS, RSS feeds, fonts, XML, CSS, images
  - Minify: remove all comments and as much whitespace (alas, make it less human readable)
- Reduce the number of requests needed to deliver content
  - Browsers spend most of their time *waiting* for content, not downloading content!
  - One method: combine JS and CSS files. Sending one 100 KB file is faster than ten 10 KB files. More data = more redundancy = better compression
- Organize the content so browsers can render it as fast as possible
  - Put CSS at the top of pages. Why? Allows the page to render progressively. Also effective for usability purposes (visual cues)
  - Put JavaScript at the *bottom* of pages
  - Make JavaScript and CSS external. The rule: *Using external files in the real world generally produces faster pages because the JavaScript and CSS files are cached by the browser. JavaScript and CSS that are inlined in HTML documents get downloaded every time the HTML document is requested.*

JavaScript and CSS Optimization Opportunities

- Combine JS and CSS files
  - <link href="http://.../?sitewide_20090710.css&newcars_20090630.css" type="text/css" rel="stylesheet" media="all" />
  - <script src="http://.../ext/yahoo-dom-event_2.7.0.js&ext/yahoo-animation_2.7.0.js&sitewide_20090710.js" type="text/javacript"></script>
  - Note: your server must be configured to understand combined requests!
- Why place JavaScript at the bottom of pages? Because the rendering of a page pauses while a JavaScript file loads (presumably because the JavaScript being loaded could alter the DOM already in the process of being created). In other words, loading them block parallel downloads. Alas, large files or network latency can cause significant delays as a page loads.
- Minification:
  - Tools: JSMin, Dojo Toolkit, Google Closure Tools, YUI Compressor
  - Problems: maintenance and obfuscation
- Many frameworks can combine automatically combine your JavaScript and CSS

Expiry

- Avoids unnecessary HTTP requests on subsequent page views and static content
- Very useful for Ajax applications: control caching for Ajax applications
- Informs the browser of the date after which the result is to be considered stale
- Example in PHP: header("Expires: Fri, 17 Jul 2010 16:00:00 GMT"); This is when the cached result will expire
- For dynamic content, use an appropriate Cache-Control header to help the browser with conditional requests

Content Delivery Network (CDN)

- The idea: disperse your static content
- Remember, the user's proximity to your web server has an impact on response times
- What you don't want to do: re-engineer or re-architect your web application to work in a distributed architecture
- Deploying your content across multiple, geographically dispersed servers will make your pages load faster from the user's perspective
- Example CDNs: Akamai, Amazon CloudFront

Caching

- Preserving and managing a collection of data that replicates original data computed earlier or stored in another location
- The idea: eliminate retrieving the original data repeatedly
- Caching opportunities: CSS and JavaScript (which may not change for a long time). A simple way to ensure the browser knows when to fetch a new version of a file is to give each file a version ID.

HTML5 Application Cache

- The browser cache: temporary storage for content (e.g., HTML, CSS, JavaScript, images)
- Problems with traditional browser cache: can be easily corrupted (e.g., by downloading big file)
- The HTML5 application cache
  - Independent of the traditional browser cache; stored in a separate location on disk
  - Can integrate with local storage or Web SQL (think data for an offline game)
  - APIs available to control downloading of cache files, and what not to cache
  - You can specify explicitly what to content to cache and what content requires network access!
  - The idea: have a cache-manifest file and specify it in the <html> tag
  - Reference and tutorial: http://www.html5rocks.com/en/tutorials/appcache/beginner/
- Example: Ms. Pac-Man
- Benefits: reduces server load, faster loading, conserves bandwidth
- Really powerful result: offline web applications