Speed up your web site

seidengroup.com
Seiden Group and Club Seiden

Alan is a leader and expert in PHP on IBM i; leader, Zend’s PHP Toolkit for IBM i; and “Performance guru of PHP on IBM i”

Seiden Group is a team of experts available for mentoring/troubleshooting/project advice/development.

seidengroup.com, alan@seidengroup.com
Contact

Alan Seiden
alan@seidengroup.com
201-447-2437

www.SeidenGroup.com
twitter: @alanseiden
Contact information

Alan Seiden
alan@alanseiden.com
201-447-2437
alanseiden.com
twitter: @alanseiden
Agenda for fast user experience

- Fast user experience
  - Beyond speed of PHP and server
- Performance big picture
- Tools that show issues visually
- Tips and configurations
Why I started to focus on web front end

• Clients called me in for performance help
• Assumed drag was on the server, PHP/DB2
• BUT many of the problems were in the front end (HTML, JS, CSS)

• Today’s complex web and mobile applications require attention to front-end performance

Let’s start with the basics
HTTP (web) flow
Request-response protocol

- Client (browser) requests a file; server responds
- One file at a time (at most 2–6 in parallel)
- Browser requests HTML file, then as it parses HTML, finds other file names to request (images, css, js...)

![Diagram of request-response protocol](image_url)
Each request passes through several layers

client network

PC or mobile device

browser
user asks for example.com

HTML

image

CSS

js

request
response
request
response
request
response
request
response

server network

firewall
reverse proxy srv. (optional)

Web server
(IBM i or other)

Databases
DB2, MySQL...

IFS (stream files)
.php .js .gif .jpg .ico .css .html ......

CGI pgms
You might guess one top strategy

Each HTTP request travels through several layers

A common-sense performance strategy suggests itself

Reduce the number and size of HTTP requests
Perceived speed
When users say app is slow

- Watch them using the application
- Is slow page response the major problem?
- Or does the application not match their workflow?

- Can you help users get their job done with fewer clicks?
Tips for perceived speed

• Users want to feel successful in achieving their tasks. You can:
  ‣ Provide feedback and status information
  ‣ Give users a fast path through common tasks
  ‣ Reduce users’ anxiety by clearly labeling page elements, buttons, links, etc., using their own terminology
  ‣ Run slow tasks asynchronously so users can cancel if desired

• Old but interesting study: http://www.uie.com/events/roadshow/articles/download_time/
  ‣ “…when people accomplish what they set out to do on a site, they perceive that site to be fast.”

• Let users know that “something is happening”
  ‣ The spinning “waiting” graphic still works
Reduce HTTP requests
HTTP requests are costly

- Each request makes a round trip to server
- Each HTTP request consumes bandwidth and CPU
- In-network tests do not measure end-user performance outside the network
  - Users could have unpredictable DSL or mobile connections
  - Firewalls and proxy servers may sit between the web server and end user
    - I’ve seen convoluted network configurations
Can caching help?

- Browsers can cache most files
- Files won’t have to be downloaded again till server has updated versions

- BUT browser must check for updates to each file

- Possible successful status codes:
  - HTTP 200: Server delivered new, fresh version of file
  - HTTP 304: Server said “not modified.” Use cached copy.
    - Faster, but still requires that request to check the file’s timestamp

- More about blocking and caching on next slide
Requests cause “blocking” in browser

- Browsers typically limit themselves to 2–6 parallel requests to a given server
- File requests stack up, blocked by prev. requests
- Above, even “304 not modified” files caused blocking
- Solution: reduce number of images or improve caching via “Expires” headers
  - [http://httpd.apache.org/docs/2.0/mod/mod_expires.html](http://httpd.apache.org/docs/2.0/mod/mod_expires.html)
Example: “Expires” headers (caching)

- For aggressive caching, place these directives in Apache config file
- Can specify file types

```apache
ExpiresActive On
# A2592000 means expire after a month in the client's cache
ExpiresByType text/css A2592000
ExpiresByType application/x-javascript A2592000
ExpiresByType application/javascript A2592000
ExpiresByType text/html A2592000
ExpiresByType image/png A2592000
ExpiresByType image/gif A2592000
ExpiresByType image/jpeg A2592000
```

- Many options: [http://httpd.apache.org/docs/2.0/mod/mod_expires.html](http://httpd.apache.org/docs/2.0/mod/mod_expires.html)
More ways to reduce “blocking”

- If many .js or .css files are used:
  - Combine them into fewer files
  - Move contents of smaller .js or .css files inline to your pages, eliminating those external files
  - Page Speed tool will help you decide
Create a favicon for your site

- Browsers always look for a file called favicon.ico in your document root
- Those little icons that appear in the browser
- Once found, will be “remembered” by browser
- If not found, will be requested every time
- How to create a favicon:
Keep connections open
Keep HTTP connections alive

- Enable “KeepAlive” setting in Apache

- The TCP connection will stay open, waiting for you
  - Good when downloading many images, css, js files
  - You’ll reduce the number of three-way “handshakes” that establish a connection
  - Even more important with longer SSL handshakes
KeepAlive details

- Configurable by number of seconds, number of files to be downloaded, before closing connection

- Recommended settings for average site
  - KeepAlive On
  - KeepAliveTimeout 15

- Details:
  - http://httpd.apache.org/docs/2.0/mod/core.html#keepalive

- Don’t overdo it—you are locking out other users from that HTTP job while it’s dedicated to you
Connecting takes time

- **Clues that Keepalive is off**
  - “Connection: close”, “Connecting”
- **Example bottom right: 3.6 seconds “Connecting” (longer than average but it really happened)**
What you see when Keep-alive is on

- Firebug’s “Net” tab shows “Connection: Keep-Alive”, and, here, timeout=300 seconds (5 minutes)
- Zero seconds to connect
- Keep-alive is working!
Use compression
Compression reduces file size

- Called gzip or mod_deflate, the same for our purposes

- Compresses, speeds up html, javascript, css, favicons, anything text-based
Netflix improved with gzip/deflate

- Saw 13-25% performance improvement
- Cut outbound traffic in half
  - That saves money for a busy site such as Netflix
- Details:
  - http://www.slideshare.net/billwscott/improving-netflix-performance-experience

- It really works!
My compression test

- http://your-server:10080/Samples/SQL_access/DB2_SQL_example.php

- Before compression: 31.0kb; loaded in 250ms
- After compression: 4.4kb; loaded in 109ms.
- That’s 14% of the size and 50% of the time!
Details of deflate/gzip compression

- **Apache directives (sample)**

  # Load IBM i's module that performs compression
  LoadModule deflate_module /QSYS.LIB/QHTTPSVR.LIB/QZSRCORE.SRVPGM

  # Specify content types to compress
  AddOutputFilterByType DEFLATE application/x-httpd-php application/json text/css application/x-javascript application/javascript text/html

- **Tutorial on my blog:**

- **Apache reference:**
  - [http://httpd.apache.org/docs/2.0/mod/mod_deflate.html](http://httpd.apache.org/docs/2.0/mod/mod_deflate.html)
Ajax: friend or foe?
AJAX=Asynchronous Javascript And XML

- AJAX updates parts of a page without reloading the whole page
- Not always XML. These days, JSON too
- Requests and file sizes are generally small
- Meant to bring SPEED to the web
- Potential problems if overused
AJAX mistake #1

• Too much of a good thing
  ▪ Requiring several AJAX requests to complete before the page itself can load fully
  ▪ Too many HTTP requests at once
  ▪ I’ve seen a situation where 4 AJAX requests were embedded in a page load
    • The AJAX doesn’t even start till the page loads
    • Causes “blocking” as the requests pile up, waiting for the previous ones to complete
    • Sessions may be shared by all AJAX calls, so locks can occur

• Solution: when page first loads, embed AJAX content in the page
  • Re-use logic on the server side when building page
  • Subsequent updates can be done with AJAX
AJAX mistake #2

- **Duplicate requests**
  - Might go unnoticed with javascript library tools (Dojo, jQuery...)
  - Happens more than you would expect! Common, actually
AJAX mistake #3

- Dynamically generating static content (don’t do that)
  - Especially JSON to feed dropdown widgets

- Solutions:
  - Change to static files
  - Cache URLs (e.g. with Zend Page Cache if using PHP, or Apache caching) See example below, before and after caching

- (Apologies for blurring: protecting confidentiality)
Blocking from JS/CSS
Javascript is expensive for speed

• Besides all the HTTP requests, JS must be parsed and run by your browser
  ‣ Even worse for mobile. Uses battery, CPU. Blocks UI

• JS libraries (Dojo, jQuery) include dozens of JS files that you may not need
  ‣ Take a look with the tools shown later in this presentation. You may see 100+ JS files
  ‣ Customize your JS library build to make distribution more compact

• CSS (style sheets) are another area to examine. Cut down/consolidate if you can
More tips for JS/CSS

• “Minify” if you can
  ‣ Strip out spaces/comments for production code
    • http://www.jsmini.com/
    • http://www.csscompressor.com/
    • Many other tools
  ‣ Saves bandwidth; browser parses JS/CSS more quickly

• Create a custom build of your JS library
  ‣ Tutorial to create custom build of jQuery
    • http://www.packtpub.com/article/building-a-custom-version-of-jquery
Live demos of front-end tools
Front-end tools demystify the web

- Visualize HTTP requests
- Find ways to eliminate requests or shrink responses
- Test more easily
- Capture “before and after” results
  - For your own documentation
  - For a report to management
Favorite front-end performance tools

- **REDbot**
  - [http://redbot.org](http://redbot.org)

- **Firebug**
  - [https://addons.mozilla.org/firefox/addon/firebug/](https://addons.mozilla.org/firefox/addon/firebug/)
  - Even better with Page Speed add-on
    - [http://code.google.com/speed/page-speed/](http://code.google.com/speed/page-speed/)

- **Page Speed Insights from Google**
  - [https://developers.google.com/speed/pagespeed/insights](https://developers.google.com/speed/pagespeed/insights)

- **Web Page Test**
  - [http://webpagetest.org](http://webpagetest.org)
• **Firebug**
  - [https://addons.mozilla.org/firefox/addon/firebug/](https://addons.mozilla.org/firefox/addon/firebug/)
  - Along with Page Speed, empowers anyone for performance
Firebug “Net” tab example
REDbot shows HTTP headers, codes

- Visitors to alanseiden.com are redirected to www.alanseiden.com
- Although redirects can harm performance, this one (‘www’) helps search engines
Page Speed Insights by Google

https://developers.google.com/speed/pagespeed/insights
A tip from Page Speed Insights

Serve scaled images
Properly sizing images can save many bytes of data.

Suggestions for this page
The following images are resized in HTML or CSS. Serving scaled images could save 3.5 MiB (98% reduction):
- http://microweaver.com/v1/Promo_SIN_SPK_MikePavlak.jpg is resized in HTML or CSS from 2,561 x 3,588 to 85 x 112. Serving a scaled image could save 3.5 MiB (98% reduction).
- http://microweaver.com/v1/Pro_Tutorial_Data_Structures_Modern.jpg is resized in HTML or CSS from 198 x 284 to 77 x 103. Serving a scaled image could save 28.4 KiB (85% reduction).
- http://microweaver.com/v1/acs-mg-image.jpg is resized in HTML or CSS from 182 x 101 to 141 x 79. Serving a scaled image could save 7.1 KiB (39% reduction).
- http://microweaver.com/v1/acs_get_image.jpg is resized in HTML or CSS from 180 x 101 to 141 x 79. Serving a scaled image could save 6 KiB (39% reduction).
- http://microweaver.com/v1/Seiden_Alan_0407.jpg is resized in HTML or CSS from 87 x 100 to 85 x 92. Serving a scaled image could save 92.5 KiB (11% reduction).

An Introduction to PHP for the RPG Programmer
In this technical online training course, you’ll get a deep-dive into the basics of PHP with expert Mike Pavlak.

A large headshot was scaled to a small size. Better to use a smaller photo.
Web Page Test (webpagetest.org)
Webpagetest “Video/filmstrip” view
Advanced Settings of webpagetest

- Stop Test at Document Complete
- Disable Javascript
- Clear SSL Certificate Caches
- Ignore SSL Certificate Errors
- Disable Compatibility View (IE Only)
- Capture network packet trace (teardump)
- Save response bodies
- Preserve original User Agent string
- DOM Element
- Minimum test duration

Network packet trace (Advanced Settings)
Keep front-end performance in mind
Remember...

- To provide an speedy overall user experience, use front-end performance techniques, such as to:
  - Reduce or shrink file sizes when you can
  - Use gzip/deflate
  - Enable keepalive (in moderation)
  - Use a favicon
  - Keep an eye on AJAX performance

- Let Firebug, Web Page Test, and Page Speed Insights assist you

- Get help when you need it

- To keep learning, see “Resources” slide, coming right up
Resources
Resources for front-end performance

• “Avoid These 7 Web Performance Snags”
  ‣ Alan’s article from June 2013 (subscription to iProDeveloper required)
    • http://iprodeveloper.com/web-amp-mobile-development/avoid-these-7-web-performance-snags

• Performance Calendar (front-end performance articles)
  ‣ http://calendar.perfplanet.com/

• Meetup groups and conferences: live and remote
  ‣ http://web-performance.meetup.com/
  ‣ http://velocityconf.com/

• Steve Souders (formerly Yahoo!, now Google)
  ‣ http://stevesouders.com
  ‣ @souders
  ‣ Books: High Performance Web Sites, Even Faster Web Sites
Contact and tips

Alan Seiden
Seiden Group
Ho-Ho-Kus, NJ

Free newsletter:
http://seidengroup.com/tips

alan@seidengroup.com ● 201-447-2437 ● twitter: @alanseiden