Lickity Split (/blog)

Zoompf's Web Performance Blog

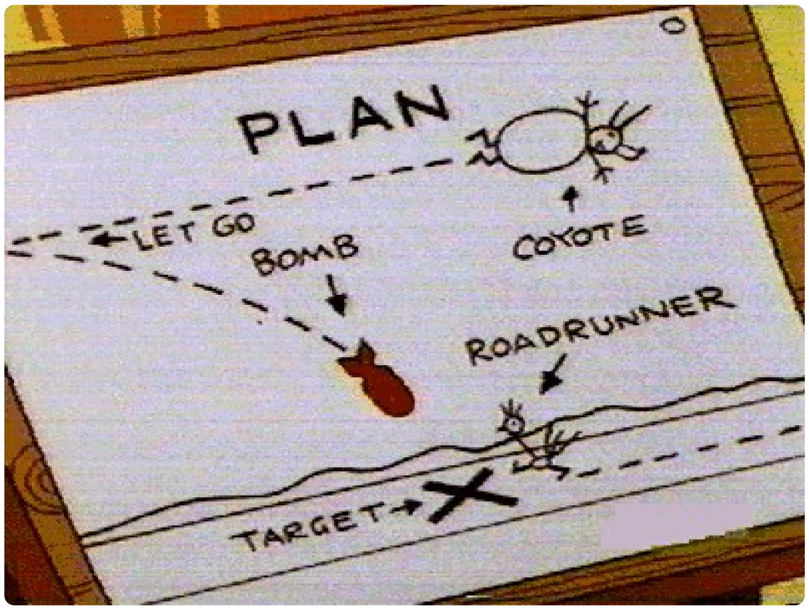
Note: Archived Content

This is the archived version of the Zoompf blog. Since our acquisition by Rigor (/blog/2015/10/zoompf-gets-acquired), all our new research and posts on web performance are being published on The Rigor Blog (http://rigor.com/blog)

Performance Optimization is a Process Not an Event

Billy Hoffman (/contact) on March 16, 2015. Category: random (/blog/category/random)

If someone asks you, *"is your website optimized?"* your answer should always be *"no"*, because, like software, it's never done. Optimizing your website's performance is not a one time event, but rather a seamless, repeatable, and non-disruptive process that the web team implements for applying performance optimizations. The *concept* of website performance optimization is simple, improves user experience (UX), and decreases operational costs. Much like how the Coyote's plans to catch the Road Runner (http://en.wikipedia.org/wiki/Wile_E._Coyote_and_The_Road_Runner) are simple:



There are numerous tools available for detecting and remediating slow pages and plenty of publicly available information about how to optimize website performance. However, it's the the **implementation** of a plan where most teams get into trouble.

Because web sites that require optimization are, by their very nature, complex, the implementation of an optimization process can be messy. Any change to an enterprise website involves multiple departments (developers, designers, DBAs, IT Admins), requires a high level of communication, and has a direct impact on the Software Development Life Cycle (http://en.wikipedia.org/wiki/Software_development_process) (SDLC). Website performance optimizations affect development, bug fixes, change control, site structure, source code, and design.

If a proper and agreed-upon process is not implemented, optimizations will have the opposite of the intended effect, and result in little nightmares like:

- Image file names being changed after optimizing/crunching.
- Caching being implemented without change control.

- Removal of color profile information resulting in broken colors.
- Minifying the only copy of JS/CSS.
- Creating CSS Sprites and deleting the original images.

So while the intent is to optimize website performance, without a clear process, the performance of any website can be degraded or even broken rather than improved.

By leveraging the publishing process, source code management, change control, and common libraries, your web team can create an optimization process that is seamless, repeatable, and non-disruptive. Implementing such a process should be done in three phases:

- **Phase 1**: Stop the bleeding
- **Phase 2**: Streamline existing design
- **Phase 3**: Design for speed

Phase 1: Stop the bleeding

In Phase 1, focus on optimization that can be done with minimal coordination on the production site. Steps towards optimization include:

- HTTP Compression (/blog/2012/02/lose-the-wait-http-compression) (robots.txt & favicon.ico)
- Lossless image optimizations
- Minification of all text resources (JavaScript, CSS, static HTML, Feeds, XML, etc.)

Next, work towards automating these optimization as part of the publishing process. The tools you use can be as simple or as complex as you like, including tools like `xargs`, perl, ant (http://en.wikipedia.org/wiki/Apache_Ant), or make (http://en.wikipedia.org/wiki/Make_(software)). You should make sure you do optimizations before copying to server, run them on a local machine, NOT on the server, and don't do this at runtime!

While this approach handles most resources, be sure to consider whether content is being added outside of publish process via CMS systems, blogs, etc. A nightly scheduled task or cron (http://en.wikipedia.org/wiki/Cron) job can be used to apply these optimizations on the directories containing those regular content updates.

Phase 2: Streaming existing design

In Phase 2, you are addressing your existing design with the goal of improvement, but without altering your basic design. The benefits of optimizing your existing design actually give you larger performance gains than in Phase 1, but doing so requires clear coordination between the dev team and the IT teams.

The optimizations to make in phase 2 include far-future caching, domain sharding, combining resources, and removing cruft or bloat on both the server and the web site.

You can make these phase 2 optimization by modifying your website were possible to avoid hard coding a path to a resource. For example, in your code to generate HTML, instead of hardcoding a link to an image (`<img src=...`), use function to instead to write out the path. In PHP, that might look like `<? =insertImage("/images/logo.png");?>`. By moving resource loading into code, you can control what URL is written out which allows you to implement phase 2 optimizations. You can now use far-future caching, since your `insertImage()` function can automatically add file versioning. You can also implement code that allows for easier combining, minifying, `data:` embedding, or domain sharding.

Phase 3: Designing for speed

Finally, in stage 3, begin to integrate speed into the core design of your web application. Designing for speed requires a fully coordinated design-development-IT team, but maximizes performance improvements.

For example, by get your designers involved in the speed process early, they will be designing beautiful images that load fast and match your website's design. Too often I see mobile sites loading desktop sized images because the designers were not involved to create appropriate or responsive assets. Consider lossy optimizations following our guidelines for image types and sizes (/blog/2013/05/achieving-better-image-optimization-with-lossy-techniques).

Create production scripts for handling image maps, adjacent images, and for sprites. Plan out which pages are using which assets or functionality, so you can implement lazy loading. Consider every design/development decision through the lens of performance. This opens up all kinds of possible optimizations, like last-modified/caching logic (http://www.fastly.com/blog/caching-dynamic-content/) for dynamic content and in-lining content or 3rd party resources (/blog/2014/01/step-by-step-guide-to-optimizing-your-apache-site-with-mod_pagespeed).

Conclusions

Those are the three stages to the website optimization process: stop the bleeding, streamline existing design, and design for speed. How do you work through these phases? Have a plan, start small, and measure everything. In order to succeed, the performance optimization process must be seamless, repeatable, and non-disruptive, or it will not work.

Are you trying to implement a performance process at your company? Then you will love Zoompf and our free tools to help you detect and correct front-end performance issues on your website: check out our free report (/free) to analyze your website for common performance problems, and if you like the results consider signing up for our free alerts (/alerts) to get notified when changes to your site slow down your performance.

Next Post (/blog/2015/03/treat-web-performance-issues-like-software-bugs)

Earlier Post (/blog/2015/03/two-new-zoompf-features)

Comments

Have some thoughts, a comment, or some feedback? Talk to us on Twitter @zoompf (https://twitter.com/zoompf) or use our contact us form (/contact).