The Four Steps

That Make Our WordPress Websites

Run Faster

An Obson Cheat Sheet

Why Performance Matters and What To Do about it

No one likes waiting for a website to load! After more than a few seconds a visitor will start to get impatient and may try to find another site. A slow website gives a poor impression, but, worse, it will be penalised by search engines, making it less likely to be discovered in the first place.

For websites that need to deliver their content, whether to make sales or to simply to get their message across, this can be crucial. In short a sluggish website is a less effective website.

First Things First: How Fast is my Website?

Most websites are slower than they need to be, but before we assume the worst we should do a bit of objective testing and find how fast, or slow, your site really is. We will use a publicly accessible tool called WebPageTest, which is at www.webpagetest.org/ (Illustration 1).

Enter the URL for your home page in the 'Enter a website URL' space, and click on 'Advanced Settings.. In Advanced Settings set 'Number of Tests to Run' to 9, for 'Repeat View' check 'First View and Repeat View', and make sure 'Capture Video' is checked.

Now click the yellow 'START TEST' button and wait.

By the way, do look to see if there are a lot of tests already waiting. If there's a message in red reading something like '33 Pending Tests' just select a different test location.

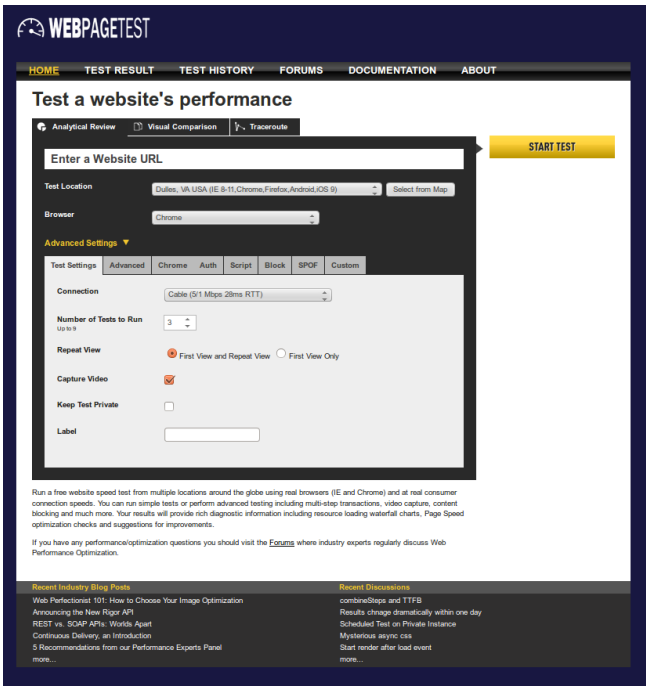

Illustration 1: WebPageTest Home Page with Advanced Settings selected

Time to First Byte (TTFB)

When this test finishes (it can take a few minutes) it will give us a lot of information, but the main thing we are interested in right now is how long it takes to *start* downloading.

The reason why we asked WebPageTest to try nine runs is that there can be a huge amount of variation from one run to the next. The results summary will give the median values, so ironing out these variations to some extent.

When the test has finished make a note of the Load Time and the Time To First Byte, both for first view and repeat view. You'll probably find that the load times are in excess of 4 seconds (first view) and 3 seconds (repeat view) – most sites are. Hopefully we'll get them down to less than 3 seconds and less than 2 seconds.

We carried out exactly the same process as we desscribe on a re-install of the OBSON.net site, and you can see the results in Table 1. This also illustrates some of the anomalies you might encounter – for example the baseline figure for repeat view load time is surprisingly low. We will discuss possible reasons for this as we go on.

Speeding Things up

To speed things up we need to use a number of techniques – image compression, page caching,

browser caching, and more. Some of these could be implemented manually, but it would be very time-consumimg, and quite difficult. The beauty of WordPress is that in most situations 'there's a plugin for it', and we found one plugin (W3 Total Cache, aka W3TC) that appeared to cover all the bases.

As a general rule we tend to think that the fewer plugins you use, the better, as they sometimes conflict and either fail to work properly or just slow things down, so we tried W3TC as it claimed to do practically everything to do with site optimisation. There was only one aspect that we couldn't get to work properly and that wasn't a show-stopper. Otherwise it was easy to use and effective.

The W3 Total Cache Plugin

Install W3TC from the WordPress dashboard by going to *Plugins* > *Add New*, searching for 'W3 Total Cache', selecting it and clicking *Install now*. When it's installed, activate it.

If you go the the W3TC General Settings and scroll down you will see that a lot of options are selected by default. These settings are close to what we want, and we'll use them as our starting-point. I've highlighted the ones I want you to check and, if necessary, change.

	Baseline	*+ W3TC*	
First view TTFB	1.452	0.167	88%
Repeat view TTBF	0.195	0.114	42%
First View Load	4.331	2.053	52%
Repeat View Load	1.591	1.500	6%

Table 1: Performance Record (Location: Dulles VA USA, Browser: Chrome)

Step 1: Page Caching

The data that browsers display is sent to them as HTML code. So whenever a browser asks a WordPress site for a page the first thing the site has to do is create that HTML. But generating each page every time it is accessed takes time and is wasteful of resources. Instead we want to store a copy of each generated page, so that it can be re-used rather than re-generating it every time. This technique is called 'page caching'

To turn page caching on, **select *Performance* > *General Settings* from the dashboard menu. In the Page Cache section check 'Enable', make sure 'Disk (enhanced)' is selected as caching method and click 'Save All Settings'.**

Step 2: Minification

The space occupied by HTML, Javascript and CSS files (and therefore the time taken to transfer them) can be reduced by a process called minification. It's more accurate to think of minification as a set of processes, as it is significantly different depending on the type of file.

Scroll down to *Minify*, check 'Enable' and click 'Save all settings'.

Step 3: Database Caching

Database caching refers to a similar technique but applied to the results of database 'queries'. The effects of this can be marginal as many of the database accesses will be rendered unnecessary by page caching anyway. It will be most effective on a highly dynamic site with a lot of users and

where the database is hosted on a different server. It's worth a try. If it makes no noticeable difference you can always turn it off again.

Scroll down to *Database Cache*, check 'Enable' and click 'Save all settings'.

Step 4: Browser Caching

Browser caching simply means sending instructions to the browser to ensure that it will cache 'static data' (things like images, style sheets, Javascript libraries that only change infrequently). Most browsers will do this anyway, but W3TC will ensure that they get the information they need to do it something like optimally.

Scroll down to *Browser Cache,* check 'Enable', and click 'Save all settings'.

Tweaking the Settings

You can probably get away with leaving the rest of the settings as they are, but I recommend that you go through the following list and make sure the settings are as described. At the very least, set the options I have highlighted! Some of these are needed to ensure that the WordPress dashboard functions correctly!

Tweaking the Page Cache settings

Go to *Performance* > *Page Cache*.

Under 'General' check:

- Cache posts page
- **Don't cache pages for logged in users (important!)**

Under 'Cache Preload' check:

- Automatically prime the page cache
- Preload the post cache upon publish events.

Click 'Save all settings'.

Tweaking the Minification settings

Go to *Performance* > *Minify*.

Under 'General' check 'Rewrite URL structure'

Under 'XML & HTML' enable 'HTML minify settings'

Under 'JS' check 'Enable'. In 'Operations in Areas' for 'Before </head>' check 'Minify', and (important!) select 'Non-blocking using async' as 'Embed type'.

Under 'CSS' check 'Enable'.

Click 'Save all settings'.

Tweaking the Browser Cache Settings

Go to *Performance* > *Browser Cache*.

Under 'General' the following boxes should be checked:

- Set Last-Modified header

- Set expires header
- Set cache-control header
- Set entity tag (eTag)
- Set W3 Total Cache header
- **Enable HTTP (gzip) compression**
- **Prevent caching of objects after settings change (important!)**

Under ‘CSS & JS’ all boxes should be checked except ‘Disable cookies for static files’. Set expires header lifetime to 31536000 (a year). Cache-control policy should be ‘Cache with max-age (“public, max-age=EXPIRES-SECONDS”)’.

Under XML & HTML all boxes should be checked. Set cache-control policy to ‘Cache with max-age (“public, max-age=EXPIRES-SECONDS”)’. I suggest keeping ‘Expires Header lifetime’ fairly short (3600, or an hour) as HTML (and XML) are not covered by the ‘Prevent caching of objects after settings change’ setting and we don’t want visitors to be stuck with copy that’s months out of date!.

Under ‘Media and Other Files’ all boxes should be checked except ‘Disable cookies for static files’. Set cache-control policy to ‘Cache with max-age (“public, max-age=EXPIRES-SECONDS”)’. Set expires header lifetime to 31536000 (a year).

Click ‘Save all settings’.

Measuring the Improvements

Now go back to WebPage Test, re-run the test and check the median results. Hopefully you will see a significant improvement. Exactly how much will depend on a number of factors: how many images you have (see the note below on resizing), how much Javascript you have and how much of it is needed to render the page, the complexity of your markup, and the location of your server, to name just a few.

It’s customary at this point to tell you how great the system is and to promise you riches beyond the dreams of avarice from following my advice. But I come from a technical, not a marketing background and I prefer to concentrate on the facts! Which are...

Yes, we seem to have achieved some quite impressive results (if yours are like mine). TTFB improved by 88% on first view and 42% on repeat views. That is undeniably good, but visitors to your site won’t care about TTFB – what they will notice is the load time.

Here we have a 52% improvement for first view, which is excellent. What about repeat views though? Only 6%? There are two possible explanations for this:

1. The 1.591 figure was an outlier. There is a huge amount of variation in the results from WebPageTest. Not surprising when you consider the complexity of the internet and how much is going on at any particular time. There can be a sudden burst of activity causing congestion on a route through the web, or a sudden relative dearth of activity. The only way to check this would be to set up a large-scale test and average out the results.

   Or, possibly,

2. The OBSON.net site was fairly efficient in the first place. This is my preferred explanation! There’s only one image, which comes from Thrive Themes and is probably well-optimised, and not much else on the page. There’s a surprising amount of rendering to do for such an apparently simple page, and this is what seems to account for most of the time it takes to load. But nothing you can do on the server-side is going to make much difference to that.

To check this I tried benchmarking the OBSON.net blog page (http://obson.net/blog/) and there the difference was much more pronounced.

Something else to remember...

When you make changes you will sometimes see a message at the top of the dashboard telling you the page cache or the minify cache needs clearing. This is because the changes you just made will have made some of the data in the cache out of date. It won't be served to the browser anyway as we set options up to ensure users would always get the latest content, but it does take up space and increase your site's resource reqirements.

Just click the buttons when requested and you will prevent this from happening.

I mentioned at the beginning that there was a feature in W3TC that we couldn't get to work. If you go to *Performance* > *General Settings* and scroll down to the bottom you will see a section called Import/Export. The idea is presumably that once you've got settings you like you can export them, try some changes, and if you don't like the changes re-import your original settings.

In our experience the export worked, but after making changes and re-importing the exported file the original settings were not reinstated. Perhaps you'll have better luck, but we think you'll probably be safer writing down the changes on a piece of paper and reinstating them by hand!

Other Things You Might Like To Try...

1. Check that your images are no bigger than they need to be (some themes, Thrive Themes for example, will resize them for you if necessary).

This can make a huge difference, particularly to the first time view, as the size of an image file is related to the *square* of its linear dimensions. Twice as big as it needs to be and you will increase the file size four times. Ten times the size means you will increase the file size by a factor of 100!

2. Add a CDN (content distribution network). These can be expensive, but if you have installed Jetpack you can use Photon, which is a free CDN. If you do this I recommend that you see my blog (http://obson.net/denial-of-service-via-xmlrpc/) about security issues around Jetpack and XMLRPC.

3. Use Google Pagespeed (https://developers.google.com/speed/pagespeed/insights/) to get a different take on your website's behaviour. Its answers aren't always the same as WebPageTest's, but it's more focused on how your site will appear to the Google search engine, so well worth looking at.

In Conclusion

I hope I've managed to demystify website optimisation for you. The procedure I've described works for us and for many others, and unless you've already implemented pretty tight site optimisation it should work for you too. If you'd like to let us know how you get on please go to our contact page (http://obson.net/contact/) and drop us a line.