Kynatro.com
The personal website of Dave Shepard

Finally a practical use for WordPress' secret SHORTINIT constant!

Ever write a plugin that needs to utilize AJAX functionality for a rapid response, such as checking if a user exists in the database on key-up, but going through the usual admin-ajax route is too slow? You can write your own AJAX response addressing a PHP file directly with the secret SHORTINIT constant to safely load the WPDB Class and be able to access the WordPress database Object!
Create a PHP file in your theme or plugin that starts out with this code and write the rest of your work below it and you'll have a quick responding AJAX retrievable PHP script:

```
<?php
// Force a short-init since we just need core WP, not the entire framework stack
define( 'SHORTINIT', true );

// Build the wp-load.php path from a plugin/theme
$wp_root_path = dirname( dirname( dirname( __FILE__ ) ) );
// Require the wp-load.php file (which loads wp-config.php and bootstraps WordPress)
require( $wp_root_path . '/wp-load.php' );

// Include the now instantiated global $wpdb Class for use
global $wpdb;

// Do your PHP code for rapid AJAX calls with WP!

// Example: Retrieve and display the number of users.
$user_count = $wpdb->get_var( $wpdb->prepare( "SELECT COUNT(*) FROM $wpdb->users;" ) );
echo "<p>User count is {$user_count}</p>";

```

 kynatro / December 11, 2012 / Development, WordPress / ajax, shortinit, wordpress, wp-config, wpdb

21 thoughts on “Finally a practical use for WordPress’ secret SHORTINIT constant!”

Ajay

December 24, 2012 at 12:43 am

Dave, thanks for the code. I’ve been struggling to find a good example for this for my plugin.

Trevor Greenleaf

December 31, 2012 at 3:27 pm

This is super awesome! I have been trying to speed up my ajax calls for awhile.

kynatro

January 8, 2013 at 10:07 am

Just noticed a place where this script could potentially fail. If your wp-config.php file exists one level up from your website root folder, you’ll also need to make sure your ABSPATH constant is defined properly or WordPress will fail to load wp-settings.php. Note the added ABSPATH definition in the fallback conditional added to the code sample.

Ajay

January 8, 2013 at 1:18 pm

Thanks for the update. I think that’s what has been breaking the code of one of my users

Ajay

January 8, 2013 at 6:37 pm

kynatro, do you think it makes more sense to load wp-load.php instead of wp-config.php since wp-load.php does the necessary checking for wp-config.php

---

kynatro

January 10, 2013 at 10:20 am

Thats actually a good point. The wp-load.php file would certainly make the whole wp-config.php file check and ABSPATH definition moot.

Ajay

January 14, 2013 at 6:36 am

I need to still test this out by making my test install as non-standard. I'll let you know what my results for the whole thing are.

Unfortunately SHORTINIT fails when you need any plugin files.

kynatro

January 14, 2013 at 9:02 am

That is certainly a limitation of it, but that does not mean that you can't load the necessary files to make plugins work. If you browse through the wp-settings.php file, you can follow the code down to where the SHORTINIT constant is checked (line 95). The break that happens there is just before all the support includes are run that would enable plugin support. If you really need access to one plugin's capability in the AJAX call, you could consider loading just those files that would be necessary to make those secondary plugins work. If you're talking about integrating this sort

of AJAX end-point though in your own plugin and making it hookable, you should probably just stick to admin-ajax.php related calls.

hendrik

May 10, 2013 at 3:35 pm

thanks!!!
this was what i needed for my postviews!

Vladimir Miloslawsky

May 24, 2013 at 1:41 am

Thanks!!!
This speedup to 20-x times my AJAX requests!

Celso Bessa

June 11, 2013 at 3:43 pm

IT seems like really improved the performance – but something curious is happening when i benchmark the performance using webpagetest.org

Before this hack, it used to give me F grade First Byte responde time (around 1 and 1.5 secs). Now it give me better grades most of time, but almost everytime now happens a longer DNS lookup that wasn't happening before.

kynatro

June 12, 2013 at 9:05 pm

Odd. Not sure why your DNS response would go up, that doesn't really have anything to do with the PHP used here.

Martin Oxby

July 17, 2013 at 7:02 am

May I take a few minutes just to thank you for this – after an hour or so of trying to find the right answer to calling the dratted $wpdb via AJAX, your code snippet above works a treat

I had to amend this line:

// Require the wp-load.php file (which loads wp-config.php and bootstraps WordPress)
require('../../../wp-load.php');

But it's now working fine – thanks for sharing, much appreciated.

kynatro

July 17, 2013 at 8:02 am

Yeah you may have to adjust that path depending on where your AJAX processing file exists in your plugin or theme.

Andrew

August 26, 2013 at 7:41 pm

Great tip, thanks.

I wrote a little function to help me load wp regardless of depth of path. It still won't cater for installs where the plugin files are not stored hierarchically underneath the wp core, but it means I can use this in my plugins and not have to worry.

```
function find_require($file,$folder=null) {
if ($folder === null) $folder = dirname(__FILE__);
$path = $folder . '/' . $file;



clearstatcache();
if (file_exists($path)) {
require($path);
} else {
find_require($file,dirname($folder));
}

}

find_require('wp-load.php');



 //example to show that it works.
global $wpdb;
$user_count = $wpdb->get_var( $wpdb->prepare( "SELECT COUNT(*) FROM $wpdb->users;"
) );
echo "User count is {$user_count}";
```

David

March 24, 2014 at 5:02 pm

Thanks for the idea,

How can be this be applied to Woocomerce? Admin-ajax is killing my site, the only two calls to admin-ajax need more time than the rest of the site¡¡

Thanks

kynatro

March 25, 2014 at 9:47 am

I don't really have any experience with WooCommerce, so I'm not sure how this could help there specifically. This sort of technique may be useful for making rapid response AJAX calls to various WooCommerce API methods. You'd need to make sure you load the WooCommerce plugin manually of course and any other support libraries that it relies on. The SHORTINIT method terminates initialization of the WordPress framework before plugins and a majority of the common dependencies are loaded (which is where its speed advantage comes from). You may be better off using the wp_ajax_* hooks built into the WordPress plugin API (http://codex.wordpress.org/AJAX_in_Plugins) if you plan on interacting with a plugin.

Don

April 2, 2014 at 5:32 pm

Why don't you include it using document root?

```
define( 'SHORTINIT', true );
require_once( $_SERVER['DOCUMENT_ROOT'] . '/wp-load.php' );
```

kynatro

April 3, 2014 at 9:10 pm

That's also a good idea. I don't know about its reliability in all server and WordPress configurations (i.e. Network setup, etc.) however, so be sure to test it out when trying it in a new environment.

Alice Wonder

June 6, 2014 at 6:33 am

I came upon almost the same solution after pulling most of my hair out, but I didn't use the SHORTINIT,

Interestingly, when I added that, it broke my AJAX. I'm guessing that since what I am doing is a little more complex than just a few database queries, there is something I need that SHORTINIT doesn't provide. If I had to guess, it's my use of WP_PLUGIN_URL within the my plugin class that I call with the php, but I didn't test. It behaved like that was what was wrong though (images that needed that path didn't load, for example)

AJAX is suppose to be simple and WordPress has most definitely over-engineered it. Taking AJAX out of WordPress control lets the coding be KISS like it should be.

kynatro

June 7, 2014 at 7:44 am

Yeah, using an AJAX method like this is still a bit limited if you're looking at interacting with the plugin API or other plugins – at which point you're better off using the much simpler to implement admin-ajax method that the WordPress plugin API provides.