Gmail **Jeffrey Kao <jeffrey.kao.taipei@gmail.com>**

[WordPress Plugin Directory] Request: clone-and-translate-post

1 message

WordPress.org Plugin Directory <plugins@wordpress.org> Wed, Dec 18, 2019 at 9:34 AM
To: Jj449 <jeffrey.kao.taipei@gmail.com>

There are issues with your plugin code.

Please read this email in its entirety, address all listed issues, and reply to this email with your corrected code attached (or linked).

Remember in addition to code quality, security and functionality, we require all plugins adhere to our guidelines. If you have not yet, please read them:

- https://developer.wordpress.org/plugins/wordpress-org/detailed-plugin-guidelines/

You will not be able to submit another plugin while this one is being reviewed, so please read the email carefully. We know it can be long, but you must follow the directions at the end as not doing so will result in your review being delayed. It is required for you to read and reply to these emails, and failure to do so will result in significant delays with your plugin being accepted.

Including Libraries Already In Core

Your plugin has included a copy of a library that WordPress already includes.

WordPress includes a number of useful libraries, such as jQuery, Atom Lib, SimplePie, PHPMailer, PHPass, and more. For security and stability reasons, plugins may not include those libraries in their own code, but instead must use the versions of those libraries packaged with WordPress.

While we do not YET have a decent public facing page to list all these libraries, we do have a list here:
https://meta.trac.wordpress.org/browser/sites/trunk/api.wordpress.org/public_html/core/credits/wp-53.php#L443

It's fine to locally include add-ons not in core, but please ONLY add those additional files. For example, you do not need the entire jquery UI library for one file. If your code doesn't work with the built-in versions of jQuery, it's most likely a noConflict issue.

Example(s) from your plugin:

clone-and-translate-post/public/js/jquery-1.10.2.min.js
clone-and-translate-post/jquery-1.10.2.min.js

Using file_get_contents on remote files

Many hosts block the use of file_get_contents on remote content. This is a security measure that we fully endorse.

Thankfully, WordPress comes with an extensive HTTP API that can be used instead. It's fast and more extensive than most home-grown alternatives. It'll fall back to curl if it has to, but it'll use a lot of WordPress' native functionality first.

https://developer.wordpress.org/plugins/http-api/

Example(s) from your plugin:

clone-and-translate-post/clone-and-translate-post.php:156: $google_trans_lan_list_json = file_get_contents(plugin_dir_url(__FILE__) . 'google_lan_code_list.json') ;
clone-and-translate-post/clone-and-translate-post.php:161: $response = file_get_contents($google_trans_lan_list_url);

Please use wp_enqueue commands

Your plugin is not correctly including JS and/or CSS. You should be using the built in functions for this:

- https://developer.wordpress.org/reference/functions/wp_enqueue_script/

- https://developer.wordpress.org/reference/functions/wp_enqueue_style/

And remember you can use this function to add inline javascript:

- https://developer.wordpress.org/reference/functions/wp_add_inline_script/

If you're trying to enqueue on the admin pages you'll want to use the admin enqueues

- https://developer.wordpress.org/reference/hooks/admin_enqueue_scripts/
- https://developer.wordpress.org/reference/hooks/admin_print_scripts/
- https://developer.wordpress.org/reference/hooks/admin_print_styles/

Example(s) from your plugin:

clone-and-translate-post/clone-and-translate-post.php:271:<link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/bootstrap/4.4.1/css/bootstrap.min.css" integrity="sha384-Vkoo8x4CGsO3+Hhxv8T/Q5PaXtkKtu6ug5TOeNV6gBiFeWPGFN9MuhOf23Q9Ifjh" crossorigin="anonymous">
clone-and-translate-post/clone-and-translate-post.php:274:<script src="https://stackpath.bootstrapcdn.com/bootstrap/4.4.1/js/bootstrap.min.js" integrity="sha384-wfSDF2E50Y2D1uUdj0O3uMBJnjuUD4Ih7YwaYd1iqfktj0Uod8GCExl3Og8ifwB6" crossorigin="anonymous"></script>

Calling files remotely

Offloading images, js, css, and other scripts to your servers or any remote service (like Google, MaxCDN, jQuery.com etc) is disallowed. When you call remote data you introduce an unnecessary dependency on another site. If the file you're calling isn't a part of WordPress Core, then you should include it -locally- in your plugin, not remotely. If the file IS included in WordPress core, please call that instead.

An exception to this rule is if your plugin is performing a service. We will permit this on a case by case basis. Since this can be confusing we have some examples of what are **not** permitted:

- Offloading jquery CSS files to Google - You should include the CSS in your plugin.
- Inserting an iframe with a help doc - A link, or including the docs in your plugin is preferred.
- Calling images from your own domain - They should be included in your plugin.

Here are some examples of what we **would** permit:

- Calling font families from Google or their approved CDN (if GPL compatible)
- API calls back to your server to process possible spam comments (like Akismet)
- Offloading comments to your own servers (like Disqus)
- oEmbed calls to a service provider (like Twitter or YouTube)

Please remove external dependencies from your plugin and, if possible, include all files within the plugin (that is not called remotely). If instead you feel you **are** providing a service, please re-write your readme.txt in a manner that explains the service, the servers being called, and if any account is needed to connect.

Example(s) from your plugin:

clone-and-translate-post/clone-and-translate-post.php:273:<script src="https://cdn.jsdelivr.net/npm/popper.js@1.16.0/dist/umd/popper.min.js" integrity="sha384-Q6E9RHvbIyZFJoft+2mJbHaEWldlvI9IOYy5n3zV9zzTtmI3UksdQRVvoxMfooAo" crossorigin="anonymous"></script>
clone-and-translate-post/clone-and-translate-post.php:271:<link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/bootstrap/4.4.1/css/bootstrap.min.css" integrity="sha384-Vkoo8x4CGsO3+Hhxv8T/Q5PaXtkKtu6ug5TOeNV6gBiFeWPGFN9MuhOf23Q9Ifjh" crossorigin="anonymous">
clone-and-translate-post/clone-and-translate-post.php:274:<script src="https://stackpath.bootstrapcdn.com/bootstrap/4.4.1/js/bootstrap.min.js" integrity="sha384-wfSDF2E50Y2D1uUdj0O3uMBJnjuUD4Ih7YwaYd1iqfktj0Uod8GCExl3Og8ifwB6" crossorigin="anonymous"></script>

Calling core loading files directly

Including wp-config.php, wp-blog-header.php, wp-load.php directly via an include is not permitted.

These calls are prone to failure as not all WordPress installs have the exact same file structure. In addition it opens your plugin to security issues, as WordPress can be easily tricked into running code in an unauthenticated manner.

Your code should always exist in functions and be called by action hooks. This is true even if you need code to exist outside of WordPress. Code should only be accessible to people who are logged in and authorized, if it needs that kind of access. Your plugin's pages should be called via the dashboard like all the other settings panels, and in that way, they'll always have access to WordPress functions.

- https://developer.wordpress.org/plugins/hooks/

If you need to have a 'page' accessed directly by an external service, you should use query_vars and/or rewrite rules to create a virtual page which calls a function.

- https://developer.wordpress.org/reference/hooks/query_vars/
- https://codepen.io/the_ruther4d/post/custom-query-string-vars-in-wordpress

If you're trying to use AJAX, please read this:

- https://developer.wordpress.org/plugins/javascript/ajax/

Example(s) from your plugin:

clone-and-translate-post/access_count.php:9:require(dirname(__FILE__) . '../../../../wp-blog-header.php');
clone-and-translate-post/translate.php:32:require_once(dirname(__FILE__) . '../../../../wp-blog-header.php');

Data Must be Sanitized, Escaped, and Validated

When you include POST/GET/REQUEST/FILE calls in your plugin, it's important to sanitize, validate, and escape them. The goal here is to prevent a user from accidentally sending trash data through the system, as well as protecting them from potential security issues.

SANITIZE: Data that is input (either by a user or automatically) must be sanitized as soon as possible. This lessens the possibility of XSS vulnerabilities and MITM attacks where posted data is subverted.

VALIDATE: All data should be validated, no matter what. Even when you sanitize, remember that you don't want someone putting in 'dog' when the only valid values are numbers.

ESCAPE: Data that is output must be escaped properly when it is echo'd, so it can't hijack admin screens. There are many esc_*() functions you can use to make sure you don't show people the wrong data.

To help you with this, WordPress comes with a number of sanitization and escaping functions. You can read about those here:

- https://developer.wordpress.org/plugins/security/securing-input/
- https://developer.wordpress.org/plugins/security/securing-output/

Remember: You must use the *most* appropriate functions for the context. If you're sanitizing email, use sanitize_email(), if you're outputting HTML, use esc_html(), and so on.

An easy mantra here is this:

Sanitize *early*
Escape *Late*
Always Validate

Clean everything, check everything, escape everything, and never trust the users to always have input sane data. After all, users come from all walks of life.

Example(s) from your plugin:

clone-and-translate-post/translate.php:29: $target = $_POST['new_lan'];
clone-and-translate-post/translate.php:26: $source = $_POST['old_lan'];

Including a copy of wp-config

Please remove clone-and-translate-post/wp-config.php

Including backup files

Please remove clone-and-translate-post/clone-and-translate-post_bk.php

Generic function (and/or define) names

All plugins must have unique function names, namespaces, defines, and class names. This prevents your plugin from conflicting with other plugins or themes. We need you to update your plugin to use more unique and distinct names.

A good way to do this is with a prefix. For example, if your plugin is called "Easy Custom Post Types" then you could use names like these:

- function ecpt_save_post()
- define('ECPT_LICENSE', true);
- class ECPT_Admin{}
- namespace EasyCustomPostTypes;

Don't try to use two letter prefixes anymore. We host nearly 100-thousand plugins on WordPress.org alone. There are tens of thousands more outside our servers. Believe us, you're *going* to run into conflicts.

You also need to avoid the use of __ (double underscores), wp_ , or _ (single underscore) as a prefix. Those are reserved for WordPress itself. You can use them *inside* your classes, but not as stand-alone function.

Related to this, using if (!function_exists('NAME ')) { around all your functions and classes sounds like a great idea until you realize the fatal flaw. If someone else has a function with the same name and their plugin or theme loads first, your plugin breaks. Using if-exists should be reserved for *shared* libraries only.

Remember: Good prefix names are unique and distinct to your plugin. This will help you and the next person in debugging, as well as prevent conflicts.

Example(s) from your plugin:

clone-and-translate-post/access_count.php:143:function so46492768_insert_post() {
clone-and-translate-post/option.php:30:function get_option($option, $default = false) {
clone-and-translate-post/option.php:165:function wp_protect_special_option($option) {
clone-and-translate-post/option.php:184:function form_option($option) {

Allowing Direct File Access to plugin files

Direct file access is when someone directly queries your file. This can be done by simply entering the complete path to the file in the URL bar of the browser but can also be done by doing a POST request directly to the file. For files that only contain a PHP class the risk of something funky happening when directly accessed is pretty small. For files that contain procedural code, functions and function calls, the chance of security risks is a lot bigger.

You can avoid this by putting this code at the top of all php files:

if (! defined('ABSPATH')) exit; // Exit if accessed directly

Incomplete Readme

Your readme is either missing or incomplete.

In some cases, such as for first plugins, ones with dependencies, or plugins that call external services, we require you to provide a complete readme.

Our goal with this is to make sure everyone knows what they're installing and what they need to do before they install it. No surprises. This is especially important if your plugin is making calls to other servers. As such, we need to make sure you're providing the users with all the information they need BEFORE they install your plugin.

Your readme MUST validate per http://wordpress.org/plugins/about/validator/ or we will reject it. Keep in mind, we don't want to see a readme.MD. Among other things, the formatting for markup is different, and the filetype isn't read by our system.

Please create your readme one based on this: https://wordpress.org/plugins/readme.txt

---- ---- ---- ----

Please make sure you've addressed all issues brought up in this email. There is no timeframe on this review, however if we have no response from this email address in 6 months, we will reject this submission. To keep your review active, all we ask is that you make corrections and reply.

When you've corrected your code, **reply** to this email with the updated code attached as a zip, or provide a link to the new code for us to review. If you use gmail, you won't be able to send a ZIP file if it contains any JS files (yes, we know it's stupid, blame Google).

If you have questions, concerns, or need clarification, please reply to this email and just ask us.

(While we have tried to make this review as exhaustive as possible we, like you, are humans and may have missed things. As such, we will re-review the **entire** plugin when you send it back to us. We appreciate your patience and understanding in this.)

--

WordPress Plugin Review Team | plugins@wordpress.org
https://make.wordpress.org/plugins/
https://developer.wordpress.org/plugins/wordpress-org/detailed-plugin-guidelines/