WordPress

WordPress (WordPress.org) is a free and open-source content management system (CMS) written in PHP and paired with a MySQL or MariaDB database. WordPress was originally created as a blog-publishing system but has evolved to support other types of web content including more traditional mailing lists and forums, media galleries, membership sites, learning management systems (LMS) and online stores.

- Examples
  - Minimal Cron
  - Run WP CLI On All Sites Within A Network
  - Hooks for assets
  - Updating post data on status transition
  - Filter one-liners
  - WP-CLI create post with custom tax
  - Hiding blocks from the block inserter
- Troubleshooting
  - Sync the database schema
  - Troubleshooting local environment problems
  - Resolving composer.lock merge conflicts
  - Package type "library" is not supported
- WordPress Importer Plugin
- Knowledge
  - Nonces
  - Move a site to a new domain
  - Change root site
  - Blocks in Simple English
  - Filtering Block based content
  - Auto linking project toolchains
  - Performance Strategy

Examples

n. One that is representative of a group as a whole.
n. One serving as a pattern of a specific kind.
n. A similar case that constitutes a model or precedent.

Examples

Minimal Cron

Change the following:

- Prefix: svd_

```
function svd_deactivate() {
	wp_clear_scheduled_hook( 'svd_cron' );
}

add_action('init', function() {
	add_filter( 'cron_schedules', 'svd_schedule_cron' );
	add_action( 'svd_cron', 'svd_run_cron' );
	register_deactivation_hook( __FILE__, 'svd_deactivate' );

	if (! wp_next_scheduled ( 'svd_cron' )) {
		wp_schedule_event( time(), 'minute', 'svd_cron' );
	}
});

function svd_schedule_cron( $schedules ) {
	$schedules[ 'minute' ] = array( 'interval' => 1 * MINUTE_IN_SECONDS, 'display' => __( 'Every 1 minute.', 'svd' )
);
	return $schedules;
}

function svd_run_cron( ) {
	// Do your stuff!
}
```

Examples

Run WP CLI On All Sites Within A Network

```
# update an option.
wp site list --field=url | xargs -I % wp --url=% option update [option name] [option value]
```

Hooks for assets

Quick reference for where to hook in styles and scripts:

Where	*Action*
Admin	admin_enqueue_scripts
Frontend	wp_enqueue_scripts
Block editor (admin)	enqueue_block_editor_assets
Blocks (front and admin)	enqueue_block_assets

Don't register or enqueue scripts and styles in init, this will break WordPress updates and unexpected things might happen.

Examples

Updating post data on status transition

You might be tempted to set the WP_Post properties (but this is an action), or use `wp_update_post()` but there is some hard-coded behaviour in `wp_insert_post()` that might affect your data.

So it's better to emulate `wp_publish_post()`.

```
function action_on_future_to_publish( $post ) {
	global $wpdb;

	if ( ! wp_is_post_revision( $post ) ) {
		// Update via the database to bypass wp_insert_post resetting the dates.
		// @see wp_publish_post()
		$data = [
			'post_modified'     => $post->post_date,
			'post_modified_gmt' => $post->post_date_gmt,
		];
		$wpdb->update( $wpdb->posts, $data, [ 'ID' => $post->ID ] );

		// After bypassing the WP caches, refresh the cache.
		clean_post_cache( $post->ID );
	}
}
add_action( 'future_to_publish', 'action_on_future_to_publish', 10, 1 );
```

Examples

Filter one-liners

Render links in content as HTML A elements:

```
add_filter( 'the_content', 'make_clickable' );
```

Examples

WP-CLI create post with custom tax

`wp mycpt create --tags='test1,test2'`

```
$tags = null;
if ( isset( $assoc_args['tags'] ) ) {
	$tags = wp_parse_list( $assoc_args['tags'] );
	unset( $assoc_args['tags'] );
}

...

$command = "post create --porcelain " . assoc_args_to_str( $assoc_args );
$id = WP_CLI::runcommand( $command, [ 'return' => true ] );
if ( ! (bool) $id ) {
	WP_CLI::error( 'Post not saved!' );
}
wp_set_object_terms( $id, $tags, TAX_MYCPT_TAG );
WP_CLI::success( sprintf( 'Created post %s.', $id ) );
```

Examples

Hiding blocks from the block inserter

Hiding blocks allow the blocks to be used for existing content, but not for new content.

```
function get_plugin_settings() {
	$disabled_blocks = [];
    $disabled_blocks[] = 'my/blockname';

	return [
		'disabledBlocks' => $disabled_blocks,
	];
}

# enqueue scripts hook:
	wp_localize_script(
		'myblocks-editor',
		'myBlocksSettings',
		get_plugin_settings()
	);
```

Javascript:

```
import { addFilter } from '@wordpress/hooks';

// Define disabled blocks.
// If you need to rely on logic available only in PHP,
// pass this data using a global variable instead.
const DISABLED_BLOCKS = window.myBlocksSettings.disabledBlocks;

/**
 * Conditionally enable/disable insertion of blocks.
 *
 * Ensure sure this function runs BEFORE you register your blocks.
 *
```

```
 * @param {object} settings block type definition
 * @param {string} name name of the block type
 * @returns {object} block type settings
 */
function filterBlockRegistration( settings, name ) {
	if ( ! DISABLED_BLOCKS.includes( name ) ) {
		return settings;
	}

	// Ensure there is a supports section
	if ( undefined === settings.supports ) {
		settings.supports = {};
	}

	// Disable the UI to add this block .
	// If the block is added in another way,
	// e.g. legacy, programmatically, copy-paste, it will still work.
	settings.supports.inserter = false;

	return settings;
}

addFilter(
	'blocks.registerBlockType',
	'myNamespace',
	filterBlockRegistration
);
```

Troubleshooting

Troubleshooting

Sync the database schema

In rare situations, the WordPress database schema might not match the schema expected by the applications or plugin. For example switching to multisite, and the wp_users table is missing the spam and deleted colums

To fix this issue locally, **create a file in the *mu-plugins*** folder such as `mu-plugins/db-upgrade.php` with the following contents, and load the site:

```
<?php
require_once ABSPATH . 'wp-admin/includes/schema.php';
require_once ABSPATH . 'wp-admin/includes/upgrade.php';
dbDelta( wp_get_db_schema( 'global' ) );
```

After the page is loaded the file can be deleted.

Troubleshooting local environment problems

A collection of troubleshooting tips to diagnose issues with a local environment that's not working right.

WP CLI Issues

1. Try `wp cli info`, if this fails then the issue is limited to WP CLI or PHP as this doesn't touch the codebase / DB.

Troubleshooting

Resolving composer.lock merge conflicts

`composer update --lock` fixes lock file when merging branches with .lock conflicts.

Troubleshooting

Package type "library" is not supported

You might see the following message after running composer:

```
Reading /Users/svandragt/.composer/auth.json
Reading /Users/svandragt/dev/_hm/siemens/milestones.local/vendor/composer/installed.json
Reading /Users/svandragt/.composer/vendor/composer/installed.json
Loading plugin OomphInc\ComposerInstallersExtender\Plugin_composer_tmp1
Package type "library" is not supported
```

This is a bug in oomphinc/composer-installers-extender >1.1.2 <= 2.0 and can be fixed by downgrading to 1.1.2

```
# Pin version to 1.1.2
composer require oomphinc/composer-installers-extender 1.1.2 --no-update --no-plugins
# Update the plugin
composer update oomphinc/composer-installers-extender --no-plugins
# Update dependencies
composer update oomphinc/composer-installers-extender
```

WordPress Importer Plugin

- It has support for: deduplication (see below), mapping image urls
- All imported posts haven `import_id` meta which is the ID they had in the import WXR.

Post processing:

Example

```
/**
 * Usage: IMPORT_POST_TYPES=blogpost,attachment wp import ...
 */
WP_CLI::add_hook( 'before_invoke:import', function () {
	add_filter( 'wp_import_posts', function ( $posts ) {
		if ( getenv( 'IMPORT_POST_TYPES' ) ) {
			$post_types = explode( ',', getenv( 'IMPORT_POST_TYPES' ) );
			$post_types = array_map( 'trim', $post_types );
			$posts = array_filter( $posts, function ( $post ) use ( $post_types ) {
				return in_array( $post['post_type'], $post_types, true );
			} );

			return $posts;
		}

		return $posts;
	} );
} );
```

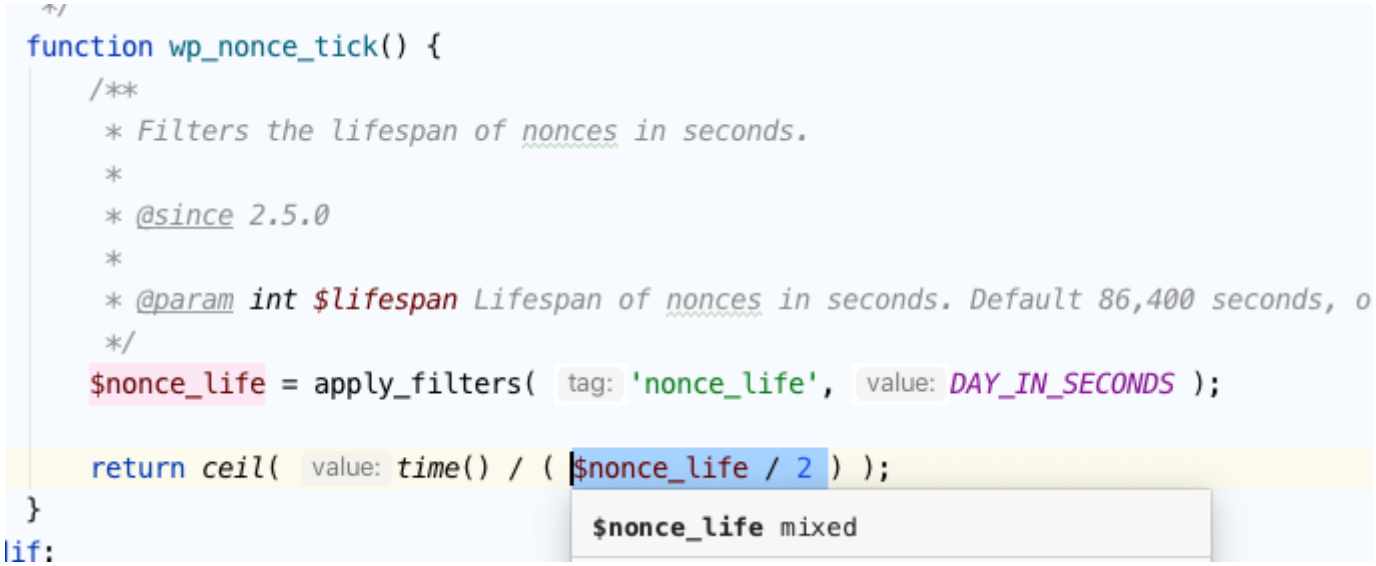
Knowledge

Knowledge is a familiarity, awareness, or understanding of someone or something, such as facts, information, descriptions, or skills, which is acquired through experience or education by perceiving, discovering, or learning.

Knowledge

Nonces

- WordPress nonces are not cryptographic nonces, as the latter are used only once, and the former are not:



> Nonces are regenerated every 12h, but are valid for 24h, hence that code. (12h = 1 tick, and they're valid for two ticks)

Knowledge

Move a site to a new domain

Move a site workflow:

1. Delete existing domain mappings for the site.
2. `wp search replace $old_url $new_url --all-tables; wp cache flush`, avoiding trailing slashes.
3. `wp rewrite flush --url=$new_url; wp rewrite flush;`
4. Login to the network site, edit the site, press Save Changes. (not sure why)
5. Verify site loads and login works.
6. re-add domain.
7. restart browsers because of cached redirects.

Knowledge

Change root site

To change the root site in a multisite network:

```
define( 'BLOG_ID_CURRENT_SITE', 1 );
```

> Plugins might not be compatbile with this change.

Knowledge

Blocks in Simple English

The following block types exist and this is what they do. I'm always getting confused by the terminology, as I'm not a native English speaker and the terminology does not cleanly map on other programming paradigms.

Block Type	**What They Say**	**What I say**
Synced patterns Enduser	*Previously Reusable Blocks* *... you will be able to arrange blocks in unlimited ways and save them as patterns for use throughout your site, directly within the editing experience. You can also specify whether to sync your patterns, so that one change applies to all parts of your site, or to keep them unsynced, so you can customize each instance.*	Patterns as symlinks: use one pattern in many places. Can be created by editors Can be converted to regular patterns.
Block Variations Developer	*Block Variations is the API that allows a block to have similar versions of it, but all these versions share some common functionality. Each block variation is differentiated from the others by setting some initial attributes or inner blocks. Then at the time when a block is inserted these attributes and/or inner blocks are applied.* *A great way to understand this API better is by using the embed block as an example.*	Blocks as a function: pass in parameters and the block acts differently.
Block Patterns Enduser	*Block Patterns are a collection of predefined blocks that you can insert into posts and pages and then customize with your own content. Using a Block Pattern can reduce the time required to create content on your site, as well as being a great way to learn how different blocks can be combined to produce interesting effects.*	Blocks group snapshot: insert a combination of blocks in one go. Good for templating layout sections, not content (use innerblocks instead)

Dynamic Blocks Developer	*Dynamic blocks are blocks that build their structure and content on the fly when the block is rendered on the front end.*	Live data blocks.
Block Styles Enduser	*Block Styles allow alternative styles to be applied to existing blocks. They work by adding a className to the block's wrapper. This className can be used to provide an alternative styling for the block if the block style is selected.*	Block Styles!

Knowledge

Filtering Block based content

Name	Type	Usage
`parse_blocks( string $content )`	Function	if you want to take a bunch of block attributes and store them in meta on save / generate stuff. More info
`pre_render_block`	Filter hook	More info
`render_block`	Filter hook	More info
`render_block_data`	Filter hook	More info

More resources

- A Crash Course in WordPress Block Filters

Knowledge

Auto linking project toolchains

If you use direnv you can automatically setup your project toolchain requirements. In the case of WordPress projects this typically includes composer, PHP, node, npm.

Simply add the required php and composer version to the project's `.envrc` file:

```
#example project requirements
use php 7.4
use composer 2

# Set node version
nvmrc=~/.nvm/nvm.sh
if [ -e $nvmrc ]; then
  source $nvmrc
  nvm use
fi
PATH_add node_modules/.bin
```

To make this happen, you must add support for the switching by adding the following to `~/.direnvrc` :

```
# Usage: use php <version>
#
# Loads the specified php version into the environent
#
use_php() {
  php --version | grep -q "PHP $1" || (brew unlink php@$1 &&  brew link php@$1 --force)
}

# Usage: use composer <version>
#
# Loads the specified composer version into the environent
#
use_composer() {
  composer --version | grep -q "version $1" || composer self-update --$1
```

```
}
```

Knowledge

Performance Strategy

- CDN for assets
- Full page cache such as Batcache
- Fragment caching for menus
- Longcache for lower traffic sites.

Knowledge

Performance

When migrating content, **suspend cache invalidation and flush the cache afterwards**.

With that in mind, the following optimisation reduces the number of database queries to 1:

```
-□□$post_data = [
-□□□'ID' => $post_id,
-□□□'post_content' => $content,
-□□];
-
-□□$updated = wp_update_post( $post_data, true );
+□□$updated = $wpdb->update(
+□□□$wpdb->posts,
+□□□[
+□□□□'post_content' => $content,
+□□□],
+□□□[
+□□□□'ID' => $post_id,
+□□□]
+□□);
```

Knowledge

WP-CLI

To download all attachment files from a remote site: from your local uploads directory:

```
wp post list --post_type=attachment --field=_wp_attached_file | xargs -I {} wget -x -nH --cut-dirs=2 "https://$DOMAIN/wp-content/uploads/{}"
```

Installing wp-env

wp-env is pretty nice and the .wp-env.json file makes me think it's most of the way there as a generic local wp setup. of course it doesn't have redis / composer / support for multiple php versions, caching plugins, and dev tools installed with it.

Requirements

Requires nvm and docker.

Installation

First install `docker-compose` (this script doesn't work with `docker compose`):

```
# On Debian or derivatives
$ sudo apt install docker-compose -y
```

Then install wp-env

```
# Install and use the latest LTS node
$ nvm install --lts
$ nvm use lts/*

# Install wp-env as a global node LTS package
$ npm install --global @wordpress/env
```

Usage

Now it's ready:

```
$ cd ~/path/to/myplugin

# Now you can use it anytime
$ wp-env start
WordPress development site started at http://localhost:8888/
```

```
WordPress test site started at http://localhost:8889/
MySQL is listening on port 32773
MySQL for automated testing is listening on port 32772

$ wp-env stop
✔ Stopped WordPress. (in 3s 165ms)`
```

The information is based on the official wp-env announcement