Mastering the Options API in WordPress

Key Takeaways:

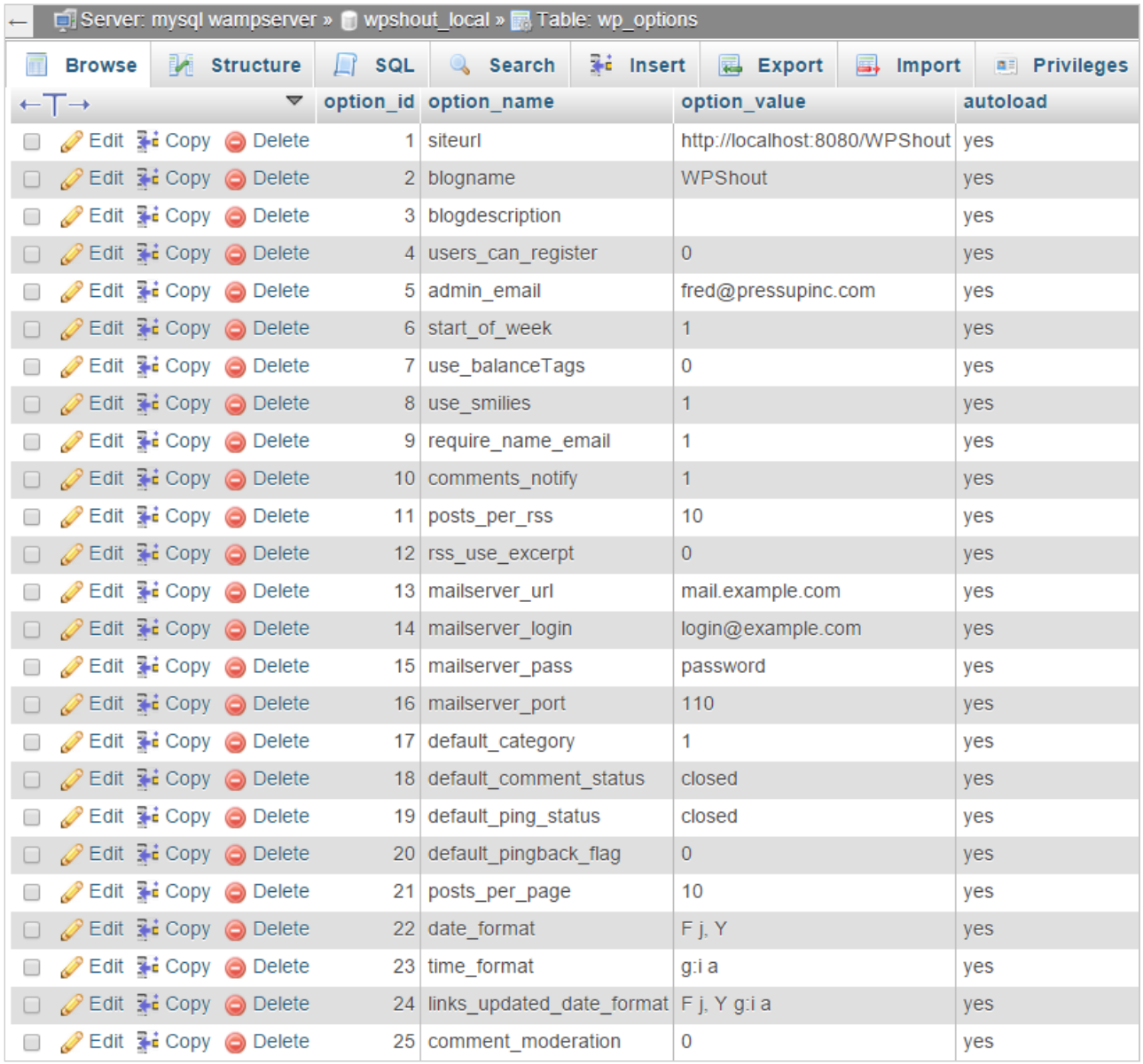
- Each WordPress site contains many sitewide options—from the site's time zone setting to a user-controlled footer background color. Each of these *site options* is stored as a simple key-value pair inside the `wp_options` table (note that the `wp_` prefix can change) of the WordPress database.
- WordPress's Options API is a simple interface for accessing, updating, and deleting sitewide options.
- `get_option()`, `update_option()`, and `delete_option()` are the three functions needed to work with site options. `update_option()` covers both adding new options and updating existing options.

WordPress developers commonly need to record and change small pieces of sitewide data. These pieces of data are known as *site options* (or, sometimes, "site settings"). Examples could include:

1. Retrieve the URL of the site's custom header logo
2. Set a custom background color across the site
3. Programmatically update the email address of the site's main administrator

Beyond these few examples, you'll probably find browsing your `wp_options` table fascinating. (As a note, the `wp_` prefix can change based on site configuration, but the `options` part won't.)

So many things are recorded as site options:



	option_id	option_name	option_value	autoload
Edit Copy Delete	1	siteurl	http://localhost:8080/WPShout	yes
Edit Copy Delete	2	blogname	WPShout	yes
Edit Copy Delete	3	blogdescription		yes
Edit Copy Delete	4	users_can_register	0	yes
Edit Copy Delete	5	admin_email	fred@pressupinc.com	yes
Edit Copy Delete	6	start_of_week	1	yes
Edit Copy Delete	7	use_balanceTags	0	yes
Edit Copy Delete	8	use_smilies	1	yes
Edit Copy Delete	9	require_name_email	1	yes
Edit Copy Delete	10	comments_notify	1	yes
Edit Copy Delete	11	posts_per_rss	10	yes
Edit Copy Delete	12	rss_use_excerpt	0	yes
Edit Copy Delete	13	mailserver_url	mail.example.com	yes
Edit Copy Delete	14	mailserver_login	login@example.com	yes
Edit Copy Delete	15	mailserver_pass	password	yes
Edit Copy Delete	16	mailserver_port	110	yes
Edit Copy Delete	17	default_category	1	yes
Edit Copy Delete	18	default_comment_status	closed	yes
Edit Copy Delete	19	default_ping_status	closed	yes
Edit Copy Delete	20	default_pingback_flag	0	yes
Edit Copy Delete	21	posts_per_page	10	yes
Edit Copy Delete	22	date_format	F j, Y	yes
Edit Copy Delete	23	time_format	g:i a	yes
Edit Copy Delete	24	links_updated_date_format	F j, Y g:i a	yes
Edit Copy Delete	25	comment_moderation	0	yes

This page captures many of the options you set on a new WordPress site (and can change in Settings > General in the WordPress admin menu). For example, `blogname | WPShout` gives the overall title for the site itself, and `start_of_week | 1` means that the site's week starts on Monday.

In this chapter, we'll be walking through how to access, add, change, and remove site options.

How to Work with Site Options

> The Options API is lovably simple.

The Options API bears a strong resemblance to WordPress custom fields, which we covered in *How to Create and Use WordPress Custom Fields*. Custom fields let you store discrete *post-level* data, whereas the Options API lets you store little bits of *sitewide* data. Other than that, they operate pretty similarly!

All site options are essentially name/value pairs: the name of the option, and the value that the option should be assigned. You can access, add or change, and delete these options with three easy-to-use functions:

1. `get_option()`
2. `update_option()`
3. `delete_option()`

We'll examine each of these functions, and get into a couple of examples.

`get_option()`

The Options API is lovably simple, and nowhere more so than `get_option()`. Here's how to access our `blogname` option—the formal title we've set for the site:

```
<?php get_option( 'blogname' ); ?>
```

That's it! Only a couple of footnotes to add to this very simple picture.

If the Option Doesn't Exist

By default, if you ask for an option that hasn't been set, `get_option()` will `return` `false`. If you want something other than `false`, you can pass in a second value to `get_option()`, specifying what you want back if the option doesn't exist. We don't use this option most of the time, so we're not dwelling on it.

When the Option is a Complex Data Type

If our option was a PHP object or array when set, that's what we get back from `get_option()`: the full-fledged PHP object or array. So `get_option()` isn't confined to strings, and you don't have to worry about any manual `unserialize()`-type work that you might be considering if you come from a general PHP background.

update_option()

Both *creating* and *updating* an option uses `update_option()`, as follows:

```
<?php update_option( 'blogname', 'Up and Running' ); ?>
```

`update_option()` updates an option's value. If the option doesn't already exist, it creates the option with the specified value. It `returns` `true` if the option was changed, and `false` if the option was not changed from its prior value, *or* if the attempt to update the option failed.

update_option()Function Arguments

`update_option()` takes two arguments:

1. The name of the option
2. The value the option is to take

As with custom fields, there is an `add_` function, `add_option()`, that we recommend you ignore. `update_option()` behaves more predictably in both adding and updating options.

Since WordPress version 4.2.0, the function also takes a third argument, `$autoload`. We'd recommend you ignore this option as well unless you have good reason to use it.

`delete_option()`

Deleting an option is as follows:

```
<?php delete_option( 'blogname' ); ?>
```

This option's only argument is the name of the option to be deleted. It returns `true` if the option was deleted, or `false` if deleting the option failed (or the option did not exist in the first place).

Example: Change the Site Title on April Fool's

This plugin changes the site title to "WPSnort"—only when it's April Fool's (April 1). Here's the code:

```php
<?php
/*
Plugin Name: WPShout April Fool's Title
*/

function wpshout_april_fools_title() {
    $joke_title = 'WPSnort';
    $site_title = get_option( 'blogname' );

    // Save "normal title" if not currently joke title
    if( $site_title !== $joke_title ) {
        update_option( 'site_normal_title', $site_title );
    }

    // On April 1, set site title to joke title
    $day = date( 'F j' );
    if( $day === 'April 1' ) {
        update_option( 'blogname', $joke_title );
        return;
    }

    // If normal_title exists and the site title's the joke title, change it back
    $normal_title = get_option( 'site_normal_title' );
    if ( $site_title === $joke_title && $normal_title ) {
        update_option( 'blogname', $normal_title );
    }
}
add_action( 'init', 'wpshout_april_fools_title' );
```

We chose this an an example of both `get_option()` and `update_option()`. The main thing to notice is that we're registering a new option, `'site_normal_title'`, which *stores* the site's "regular" title so it's not lost when the April Fool's title overwrites it. This "regular title" gets saved back to the site's title anytime it's not April 1.

By the way, if you wanted to actually implement this functionality, simply filtering the site title on April 1 would be quite a bit more elegant. The complexity of the function above should perhaps be a giveaway: functions that have lots of `if`-statements and "placeholder" variables can often (although not always) be done a simpler way.

Sidenote: Viewing All Site Options at `/wp-admin/options.php`

Before we move on, we just want to share a cool trick we picked up a while back. If you want to view *all* your site's options—on a single page, with no need to browse your database—simply go to `http://yoursite.com/wp-admin/options.php`. It's all there!

That's the Basics of Options

As we hope we've made clear, WordPress options are pretty darn simple. And once you understand them, you'll have a lot of power over the global configuration of your WordPress sites.

Summary Limerick

`update_option()`'s the best option for
Bits of site data you need to store.
Does `get_option()` retrieve it?
You'd better believe it.
Then `delete_option()` and it's no more.

Quiz Time!

1. A "site option" is *not*:
    A. Designed to expire at a set date
    B. A key-value pair
    C. Stored in the database's "options" table (`wp_options` by default)

2. The result of calling `get_option()` on a nonexistent option is:
    A. The option is created in the database with value `''`, and the function call `return`s this value
    B. The function call `returns` `false`
    C. A PHP "Data object not found" error

3. One difference between `update_option()` and both `get_option()` and `delete_option()` is that `update_option()`:
    A. *Changes* database values
    B. Does not `return` a value
    C. Requires two arguments

Answers and Explanations

1. `A`. The Transients API is used for options that are designed to expire.

2. `B`. Both parts of `A` are false—the requested option is not created if it does not already exist. `C` is a made-up error.

3. `C`. `update_option()` requires both the option's key and desired value. `delete_option()` does change values, and all three functions `return` values.