TUTORIALS

Extending The WordPress Theme Customizer Boilerplate

LAST UPDATED ON: DECEMBER 27, 2017

Part 3 of the Theme Customizer series introduced you to the Theme Customizer Boilerplate which allows you to simplify code that handles your theme options. All you need to do is pass an array of option fields and the boilerplate will take care of registering Theme Customizer sections, settings and controls for you behind the scenes.

Until now, boilerplate allowed you to use text fields, checkboxes, radio buttons and select fields in Theme Customizer, this articles shows you how you can

extend it.

Note: Before proceeding, please download the latest version of the WordPress Theme Customizer Boilerplate from its Github repository. I've made some improvements to it since last tutorial, and it's important that your code is up to date. Take a look at previous post for more notes on changes, but in a nutshell, once you copy the boilerplate to your theme folder, you don't need to edit its files at all — all the editing is done using filter and action hooks.

Hooking into Theme Customizer Boilerplate

There are several action and filter hooks in the WordPress Theme Customizer Boilerplate. You can hook into any one of them from your theme's functions.php file by using add_action and add_filter functions:

- **'thsp_cbp_directory_uri'** – Filter hook defined in helpers.php, allows you to change location of Customizer Boilerplate in your theme folder. By default, boilerplate path looks like this – **get_template_directory_uri() . '/customizer-boilerplate'** – but if you'd rather move it to a custom location, this is the hook that can help you.
- **'thsp_cbp_menu_link_text'** – Filter hook defined in helpers.php, lets you change Menu text link. Boilerplate adds a link under Appearance in WordPress dashboard, allowing users easy access to Theme Customizer. By default, that link will say "Theme Customizer" and you can change the text using 'thsp_cbp_menu_link_text' filter hook.
- **'thsp_cbp_capability'** – Filter hook defined in helpers.php. Allows you to change default required capability used in $wp_customize->add_setting method.
- **'thsp_cbp_option'** – Filter hook defined in helpers.php. If you're using 'option' in your settings arguments, use this hook to change name of the entry your theme settings values will be stored under in wp_options table. Default value is 'thsp_cbp_theme_options', make sure you hook into this one and change that to something that has your theme name in it.
- **'thsp_cbp_options_array'** – Filter hook defined in options.php, you MUST hook into it and replace the default options array (containing sample options) with options that are used in your theme. I'll repeat that, bold it and underline it: **You MUST hook into it and replace the default options array with options that are used in your theme.**

- **'thsp_cbp_custom_controls'** – Action hook defined in custom-controls.php, by hooking into it you can create your own custom controls, keep reading to see an example of how to do it.
- **'tshp_cbp_remove_sections'**, **'tshp_cbp_remove_controls'** and **'tshp_cbp_remove_settings'** – Filter hooks defined in customizer.php. You can pass them arrays of built-in section IDs (or controls IDs, or settings IDs) to remove some of the built-in sections, controls or settings.

Note: While we're at extensibility and creating your own hooks so other developers can use them to extend your code, it's impossible to overstate how important this is. After all, that's how WordPress (core) works. And I couldn't thank [Pippin](#) and [his](#) [articles](#) enough for getting this idea in to my head.

Custom controls

The updated version of Theme Customizer (which you just checked out, right?) has a few more controls you can use — textarea field, HTML5 number field and images field, which is basically a fancy version of radio buttons.

These custom controls are defined in custom-controls.php, I won't go through all of them here, but let's take a look at one (HTML5 number field) to see how it all works:

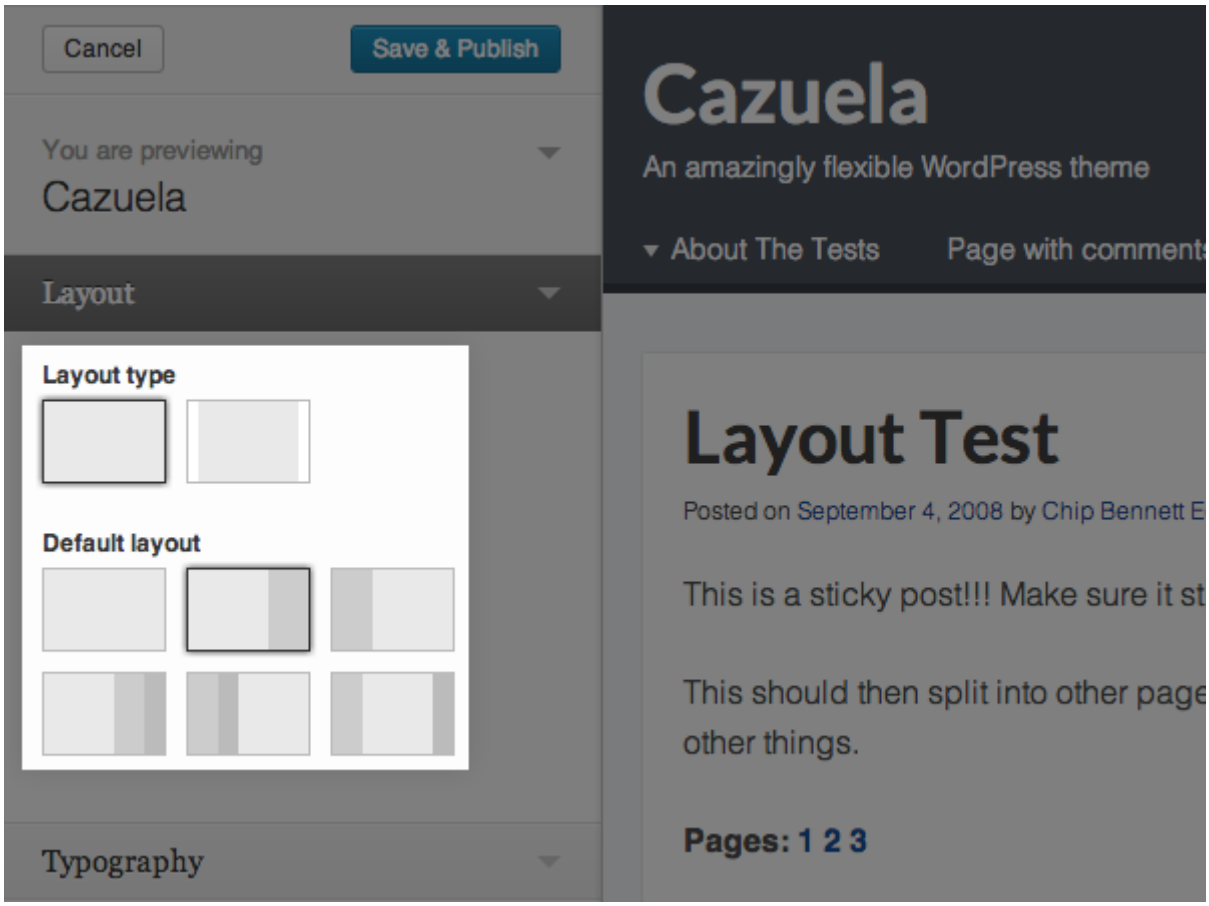
```
/**
 * Creates Customizer control for input[type=number] field
 *
 * @since   Theme_Customizer_Boilerplate 1.0
 */
class CBP_Customizer_Number_Control extends WP_Customize_Control {

    public $type = 'number';

    public function render_content() {
        echo '<label>
            <span class="customize-control-title">'.  esc_html( $this->la
            <input type="number" '. $this->link() .' value="'. intval( $
        </label>';
    }

}
```

As you can see, all you need to do is define new control $type and its render_content function which outputs the control in Theme Customizer screen.

Using Customizer Boilerplate's built-in custom controls

It's the same as simple fields covered in previous tutorial, the only only thing you need to be aware of are 'types' you need to use for each one:

- Number field – **'number'**
- Textarea field – **'textarea'**
- Images that act as radio buttons – **'images_radio'**, here's an example of this control in an upcoming free [Cazuela theme](#):



Knowing names of these new control types, adding one is easy. Here's how you can add a number field control to array that holds all your options:

```
/*
 * ============
 * ============
 * Number Field
 * ============
 * ============
```

```
 */
'new_number_field' => array(
    'setting_args' => array(
        'default' => '',
        'type' => 'option',
        'capability' => $thsp_cbp_capability,
        'transport' => 'refresh',
    ),
    'control_args' => array(
        'label' => __( 'Number', 'my_theme_textdomain' ),
        'type' => 'number', // Textarea control
        'priority' => 8
    )
)
```

Note: If you're not sure where to add this, check "Using Options Array to Add Customizer Sections, Settings and Controls" section of [Part 3 of this series](). Also, there's a sample for each one of custom controls in options.php file.

Adding your own custom controls

Let's get back to 'thsp_cbp_custom_controls' action hook I mentioned earlier:

```
/**
 * Action hook that allows you to create your own controls
 */
do_action( 'thsp_cbp_custom_controls' );
```

It's a simple WordPress action hook that allows you to add your own custom controls without modifying Theme Customizer Boilerplate files. Why would you do want to avoid editing them? Because if you're hooking into boilerplate instead, whenever someone updates it, you can just grab the latest version, drop it into your theme and not lose the changes you made. Think editing WordPress core files vs. writing a plugin, editing a theme vs. creating a child theme etc.

If you ever need to add your own custom controls, this is how you can do it:

```
function my_theme_add_customizer_boilerplate_control() {
    /**
     * Creates custom control to use with Theme Customizer Boilerplate
     * Use a unique class prefix!
```

```
     *
     * @since   Theme_Customizer_Boilerplate 1.0
     */
    class CBP_Customizer_My_Control extends WP_Customize_Control {

        public $type = 'my_type'; // Change this

        public function render_content() {
            // Control output goes here
        }

    }
}
add_action( 'thsp_cbp_custom_controls', 'my_theme_add_customizer_boilerp
```

Make sure you prefix your custom control class with something unique, so its name doesn't clash with another class. I used 'CBP_' (Customizer Boilerplate) — since you're using boilerplate in a theme, your theme's name makes a lot of sense and should work fine for you.

Theme Customizer: What's Next?

Now that the WordPress Theme Customizer Boilerplate is extensible through hooks, we'll take a look at to add "conditional theme options" – ones that will only appear if a certain plugin is active and help you keep Theme Customizer screen de-cluttered.

What are your thoughts on Customizer Boilerplate so far? Do you plan to use it in your themes? Any ideas on how it could be improved? Your feedback is welcome, always.

Article by Slobodan Manic
WPExplorer.com author