Plugin Topics: Adding Plugin Admin Pages

If we want to allow users to configure our plugin, we can add menus to the admin pages either as a top-level menu or a submenu of an existing menu.

When used with the settings and options APIs we get control of how users configure the plugin from the admin area, assuming that they have permission to do so.

Top Level Menus

Top-Level Menus are what people see when they first view the admin area. This means that our admin page has more visibility but, depending on the plugin, it may be better to use a submenu

The first step is to create a function that outputs the HTML. In this function, we will perform the necessary security checks and render the options we've registered using the Settings API.

> **Note:**
>
> Wrap your HTML using a `<div>` with a class of `wrap`.

```
<?php
function rivendellweb_options_page_html() {
?>
  <div class="wrap">
    <h1><?php echo esc_html( get_admin_page_title() ); ?></h1>
    <form action="options.php" method="post">
      <?php
      // output security fields for the registered setting "rivendellweb_
      settings_fields( 'rivendellweb_options' );
      'rivendellweb_options'// output setting sections and their fields
```

```
      // (sections are registered for "rivendellweb", each field is regist
      // output save setting'rivendellweb'// output save settings buttonin
      ?>', '?>
    </form>
  </div>
  <?php
  }
```

The second step is to register the Rivendellweb menu. Make sure you attach your function as a callback to the admin_menu hook.

```
<?php
function rivendellweb_options_page() {
  add_menu_page(
      'Rivendellweb',
      'Rivendellweb Options',
      'manage_options',
      'rivendellweb',
      'rivendellweb_options_page_html',
      plugin_dir_url(__FILE__) . 'images/icon_rivendellweb.png',
      20
  );
}

add_action( 'admin_menu', 'rivendellweb_options_page' );
```

For a list of parameters and what each, see the add_menu_page() in the reference.

Sub Menus

Sub-Menus place the admin page as a submenu of an existing page. For most plugins, this makes more sense than creating a top-level menu.

Just like with top-level menus, adding a submenu is a two-step process.

The first step creates a function that will output the HTML. In this function, we will perform the necessary security checks and render the options we've registered

using the Settings API.

```
<?php
function rivendellweb_options_page_html() {
    // check user capabilities
    if ( ! current_user_can( 'manage_options' ) ) {
        return;
    }
    'manage_options'?>
    <div class="wrap">
        <h1><?php echo esc_html( get_admin_page_title() ); ?></h1>
        <form action="options.php" method="post">
            <?php
            // output security fields for the registered setting "rivende
            settings_fields( 'rivendellweb_options' );
            'rivendellweb_options'// output setting sections and their fi
            // (sections are registered for "rivendellweb", each field is
            // output save setting'rivendellweb'// output save settings bu
            ?>', '?>
        </form>
    </div>
    <?php
}
```

The second step is to register the Sub-menu. Attach the registration to the `admin_menu` action hook.

```
<?php
function rivendellweb_options_page()
{
    add_submenu_page(
        'tools.php',
        'Rivendellweb Options',
        'Rivendellweb Options',
        'manage_options',
        'rivendellweb',
        'rivendellweb_options_page_html'
```

```
    );
}
add_action('admin_menu', 'rivendellweb_options_page');
```

For a list of parameters and what each do see the add_submenu_page() in the reference.

Settings API

You must define a new setting using register_setting(), it will create an entry in the `{$wpdb->prefix}_options` table.

> You can add new sections on existing pages using add_settings_section().
>
> You can add new fields to existing sections using add_settings_field().

Alert:

`register_setting()` as well `add_settings_*()` functions should be added to the `admin_init` action hook.

This is a full example of settings:

```
<?php
function rivendellweb_settings_init() {
    register_setting('read'reading'vendellweb_setting_name');

    add_settings_section(
        'rivendellweb_settings_section',
        ');

    add_settings_section(
        'llweb_settings_section_callback',
        'reading'
    );
```

```
    add_settings_field(
        'rivendellweb_settings_field',
        'Rivendellweb Setting', 'rivendellweb_settings_field_callback',
        'reading',
        'rivendellweb_settings_section'
  ',
        '/**
 * register rivendellweb_settings_init to the admin_init action hook
 */d_action('admin_init', 'rivendellweb_settings_init');

/**'admin_init'/**
 * callback functions
 */endellweb_settings_section_callback() {
    echo '<p>Rivendellweb Section Introduction.</p>';
}

// field conten'<p>Rivendellweb Section Introduction.</p>'// field content
    ?>'rivendellweb_setting_name'?>
    <input type="text" name="rivendellweb_setting_name" value="<?php echo
    <?php
}
```

Registering Settings

See register_setting() in the Function Reference for an explanaton about the used parameters.

Addings Settings Sections

Sections are the groups of settings you see on WordPress settings pages with a shared heading. In your plugin, you can add new sections to existing settings pages rather than creating a whole new page. This makes your plugin simpler to maintain and creates fewer new pages for users to learn.

> Please add_settings_section() for full explanation about the used parameters.

Adding Fields

Refer to the Function Reference about add_settings_field() for information about the used parameters

Getting Settings

Getting settings is accomplished with the get_option() function.

The function accepts two parameters: the name of the option and an optional default value for that option.

```
<?php
$setting = get_option('rivendellweb_setting_name');
```

Options API

The Options API allows developers to create, read, update and delete WordPress options. ***As far as I understand it, the Settings API lays out the settings page and the Options API handles storing and retrieving the options from the database.***

Where Options are Stored?

Options are stored in the `{$wpdb->prefix}_options` table. `$wpdb->prefix` is defined by the `$table_prefix` variable set in the wp-config.php file.

How Options are Stored?

Options may be stored in the WordPress database in one of two ways: as a single value or as an array of values.

Single Value

When saved as a single value, the option name refers to a single value.

```
<?php
```

```
// add a new option
add_option('rivendellweb_custom_opti'rivendellweb_custom_option'// get an
$option = get_option('rivendellweb_custom_option');
```

Array of Values

When saved as an array of values, the option name refers to an array, which itself may be comprised key/value pairs.

If you have a large number of options to set, using arrays of values is the better solution since adding an array reduced the number of computationally expensive transactions required to save the options.

```
<?php
// array of options
$data_r = array('title' => 'hello world!''title'lse );
// add a new option
add_option('rivendellweb_custom_option', $data_r);
'rivendellweb_custom_option'// get an optionvendellweb_custom_option');
// output the title
echo esc_html($options_r['title']);
');
// output the title
echo esc_html($options_r['
```

Available functions

The following functions are available as part of the Options API. The difference between the `*_site_option` functions and their plain counterparts is that the `site` functions are meant for WordPress multi site networks where the other ones are meant for individual sites.

- Add Option
  - add_option()
  - add_site_option
- Get Option
  - get_ottion()
  - get_site_option()

- Update Option
    - update_option()
    - update_site_option()
- Delete Option
    - delete_option()
    - delete_site_option()