

How To Write A Plugin

How To Write A Plugin

*Please note that plugins are only compatible with v2.0+ of Ushahidi. Also, please refer to [Actions](#) and [Filters](#).

Introduction

Plugins reside inside the 'plugin' folder that's located in the root of the ushahidi website and adhere to [Kohana's cascading filesystem](#). The general structure of a plugin looks like this:

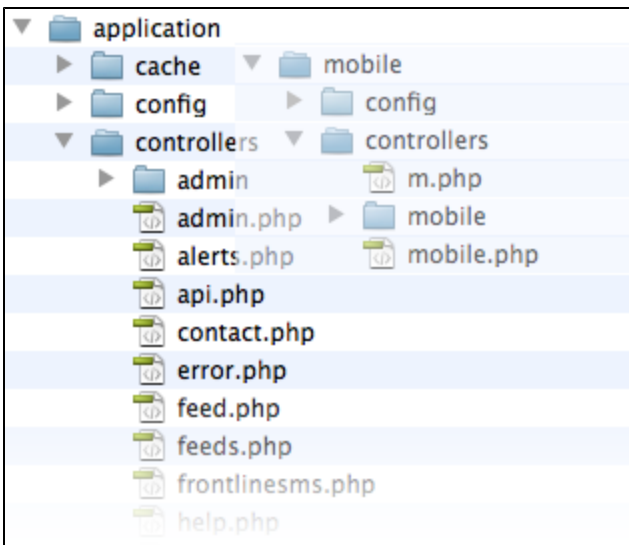
```
- [plugin folder]
| - [my_plugin]
|   + readme.txt
|   +- config
|   +- controllers
|   +- helpers
|   +- hooks
|   +- i18n
|   +- libraries
|   +- models
|   +- views
```

All these folders are not necessary for a single plugin - it just depends on what your needs are i.e. Does your plugin integrate views and/or models and controllers? It's all up to you. This gives you access to the Kohana MVC system in its magical beauty and splendor.

Best Practices

Naming Files

The Kohana cascading filesystem can be a little confusing. Think of it this way: The folders in the main application folder are mirrored into the plugin folder. Essentially, you could drop the plugin into the application folder in the same order and have it work the same way.



Therefore it is probably a good idea to either:

1. Give your files (view, controllers, hooks etc) unique names. You can achieve this by prefixing your files with the plugin name e.g. [myplugin_main].php
2. Place your files in folders with the same name as the plugin. Example:

```
- [plugin folder]
| - [my_plugin]
|   +- readme.txt
|   +- config
|   - controllers
|     - my_controller.php
|   +- helpers
|   +- hooks
|   +- i18n
|   +- libraries
|   +- models
|   +- views
```

The Readme File

If you don't have a file called `readme.txt` in the top directory (e.g. `my_plugin`), it doesn't matter how much code you write - you won't see your plugin in the plugins list on your Ushahidi site. Save yourself some frustration, and start by creating the `readme.txt` file. It should look something like this:

=== About ===

name: HelloWorld

website: <http://www.ushahidi.com>

description: Say hello to the world

version: 0.1

requires: 3.0

tested up to: 3.0

author: Joe Bloggs

author website: <http://www.joebloggsite.com>

== Description ==

Say hello to the world

== Installation ==

1. Copy the entire `/helloworld/` directory into your `/plugins/` directory.
2. Activate the plugin.

- Change the information in this file to match your plugin name, your name, your website etc.

The Hook

When a plugin is loaded, Ushahidi will automatically scan the plugin for any [hooks](#) and run them immediately. Methods within hooks are then available. Hooks are not required to create a plugin but are extremely useful for creating simple plugins.

```
- [plugin folder]
| - [my_plugin]
|   - hooks
|     my_hook.php
```

An Example Hook File

```

class hello {

    public function __construct()
    {
        // Hook into routing
        Event::add('system.pre_controller', array($this, 'add'));
    }

    public function add()
    {
        // Hook into main_sidebar event and call the hello method
        Event::add('ushahidi_action.main_sidebar', array($this, 'hello'));
    }

    public function hello()
    {
        // Print the words 'Hello World' in the front page side bar
        echo "Hello World!!!";
    }
}

//instatiation of hook
new hello;

```

Using Views

Okay, so now you want to apply your mad design skills and render a lot more HTML. 'printing' or 'echoing' everything out in your hook function can be a lot of work. No worries we have you covered.

Here's your new structure:

```

- [plugin folder]
| - [my_plugin]
|   - hooks
|     my_hook.php
|   - views
|     - [my_plugin]
|       my_view.php

```

Ehh... views > my_plugin > ... ? So why create an additional folder called my_plugin again within the views folder? Remember the Kohana system is cascading which in laymans terms means what is in the applications folder is identical to what is in the plugin folder, so if you have a view called 'header.php', the 'header.php' view file in the Ushahidi core app will be given priority. Creating a [plugin folder] > [my_plugin] > [views] > [my_plugin] > my_view.php is no different from creating [application] > [views] > [my_plugin] > my_view.php i.e. if you move the folder over it'll still work.

An Example implementation

We'll use the same HelloWorld example but this time with a view

```

| hooks/helloworld.php

```

```
// Hook into the main_sidebar event and call the method 'hello'
Event::add('ushahidi_action.main_sidebar', 'hello');
```

```
public function hello()
{
    // This time we'll get the content from our view
    View::factory('helloworld/my_html')->render(TRUE);
}
```

views/helloworld/my_html.php

```
<br />
<div class="fancy">
    <h5>Hello World!</h5>
    Hello From ushahidi!
</div>
```

Using Controllers

Now we're getting into the juicy stuff. You want to add even more functionality to your plugin - which you could actually do with your hook alone, but you want to add more structure to your 'mini-app'.

As you can see we're getting into a full blown app of a plugin:

```
- [plugin folder]
| - [my_plugin]
|   - controllers
|     hello.php
|   - hooks
|     my_hook.php
|   - views
|     - [my_plugin]
|       my_view.php
```

And the three of our files laid out:

hooks/helloworld.php

```
// Hook into the main_sidebar event and call the Hello controller
// and the method _say_hello within the Hello controller
Event::add('ushahidi_action.main_sidebar', array('Hello_Controller', '_say_hello'));
```

controllers/hello.php

```

class Hello_Controller extends Controller {
    public function _say_hello()
    {
        $view = View::factory('helloworld/my_html');
        $view->ip_address = $_SERVER['REMOTE_ADDR'];
        $view->render(TRUE);
    }
}

```

views/helloworld/my_html.php

```

<br />
<div class="fancy">
    <h5>Hello World!</h5>
    Hello From ushahidi! <br />
    Your IP Address is: <?php echo $ip_address; ?>
</div>

```

A Sample "Hello World" Plugin File

Attached is a "hello world" plugin file. To download and use the file, click on the tools menu (top right corner of this window), and then click on attachments - helloworld.zip. You just need to drop the helloworld folder into the /plugins folder of your ushahidi installation and then activate it on the back-end. The readme text file tells Ushahidi that this is a plugin that can be activated. When it has been activated, in the main page, you'll see "Hello World" in big letters.

Creating Tables With Plugins

If you've reached this far, you are without a doubt a plugin pro - so this step takes you to a whole other level. You've built your ushahidi mini-application but it needs to store some information that wouldn't fit into the available tables. Examples include settings and/or data. So here's how you do it:

1. Create a libraries folder within your plugin (if you don't already have one).
2. Within this folder, create a file called [my_plugin_name]_install.php. Please note that [my_plugin_name] is the same exact unique name of your plugin.
3. This library file is actually a PHP class that contains 2 methods. A constructor is optional. The required methods are run_install() and uninstall(). The plugin activator built into Ushahidi looks for these two methods when activating or deleting plugins.
4. Please note that the class name must begin with a capitalized letter.
5. Here's an example:

```

class My_plugin_name_Install {

    /**
     * Constructor to load the shared database library
     */
    public function __construct()
    {
        $this->db = new Database();
    }

    /**
     * Creates the required database tables for my_plugin_name
     */
    public function run_install()
    {
        // Create the database tables
        // Include the table_prefix
        $this->db->query("
            CREATE TABLE IF NOT EXISTS `".Kohana::config('database.default.table_prefix')."ELLO_table`
            (
                `id` INT UNSIGNED NOT NULL AUTO_INCREMENT,
                `incident_id` INT NOT NULL COMMENT 'incident_id of the new report that is created',
                `user_id` INT NOT NULL COMMENT 'user_id of the user that performed this assessment',
                PRIMARY KEY (id)
            );");
    }

    /**
     * Deletes the database tables for my_plugin_name
     */
    public function uninstall()
    {
        $this->db->query("
            DROP TABLE `".Kohana::config('database.default.table_prefix')."jasd_table;
            ");
    }
}

```

Adding a New Page

see [Adding a New Page](#)

Admin Settings

See [Adding Admin Settings to your Plugin](#)