How to create plugin from scratch

In this Tutorial we are going to learn How to create plugin from scratch?.

What is WordPress?

WordPress is an open source CMS (Content Management System) coded in PHP and MySQL. WordPress is completely free and it allows us to easily customize the look of our website using its themes and plugins architecture.

What is a WordPress plugin?

A WordPress Plugin is actually a single file or group of files which extends or enhances the functionality of a WordPress site. Without the use of plugins, WordPress is a little bit more like a blogging platform. For example, with the use of an e-commerce plugin, we can build e-commerce sites within WordPress. Using a membership plugin, a membership-based site can be built upon WordPress.

What are the prerequisites?

You will have to work with technologies like PHP, JavaScript, and MySql. You need to download wordpress zip file from www.wordpress.org .You may have basic knowledge of those technologies.

How do WordPress plugins work?

An important concept to understand in WordPress plugins are WordPress *hooks*, e.g. *actions* and *filters*. Hooks allows our plugins to run specific functionality at specific times within WordPress functions.

Actions

Actions are hooks triggered when WordPress performs events like publishing a post, initializing header and footer areas, creating a new user or deleting pages, etc. For example, the following is an action which runs specific functionality when admin menus are loaded in the administration panel:

```
add_action( ‘admin_menu’, ‘add_admin_menu’ );

function add_admin_menu() {

// add a group of custom menus

}
```

admin_menu is an action hook provided by WordPress which allows us to run a function, add_admin_menu, for our needs right when WordPress is going to load administration menus.

Filters

In general, a filter function takes the input as unmodified data and returns the modified data. Filters sit between the database and browsers and perform certain operations on data, like when WordPress is generating pages or adding new posts in the database. For example, take a look at the following filter:

```
add_filter( 'comment_text', 'filter_words' );

function filter_words( $data ) {

// filter comment words and return the modified comment

}
```

In the above filter, comment_text is the filter name and filter_words is a callback function which will take the comments as input and filter the words, returning the modified comments to the output.

3. Getting Started

STEP 1 : Storing your plugin

The first step to create your WordPress plugin is making a folder to store all your files. Plugins are saved in the following folder: /**wp-content/plugins/**. The folder you create needs a unique and descriptive name to ensure it doesn't clash with any other plugin. From the main WordPress directory, navigate to **wp-content**, then to **plugins**. Inside the **plugins folder,** create a new folder named **my-first-plugin**.

To make things easier for yourself in the future, it is best to separate the various files that make up your WordPress plugin and give them their own sub folders, rather than having all your plugin's files in the main folder. For example, If your plugin uses some custom CSS, you create a **CSS** folder and save all CSS files in there. If your plugin uses custom JavaScript, you create a **JavaScript** folder.

STEP 2 : Creating the first file

The first file in your plugin is an important one. It contains all the information WordPress needs to display your plugin in the plugin list, which allows you to actually activate the plugin.

In your **myfirstplugin** folder, create a new PHP file named **index.php.**

This file will primarily hold 'header comments' with various pieces of information that will be read/displayed by WordPress. In header comment, you must start by adding your plugin name with the description of the plugin, your own name, a link to your website, the current version of your plugin etc. Your file will look like this:

```
<?php
```

/*

Plugin name: My first plugin

Description: This is my first plugin

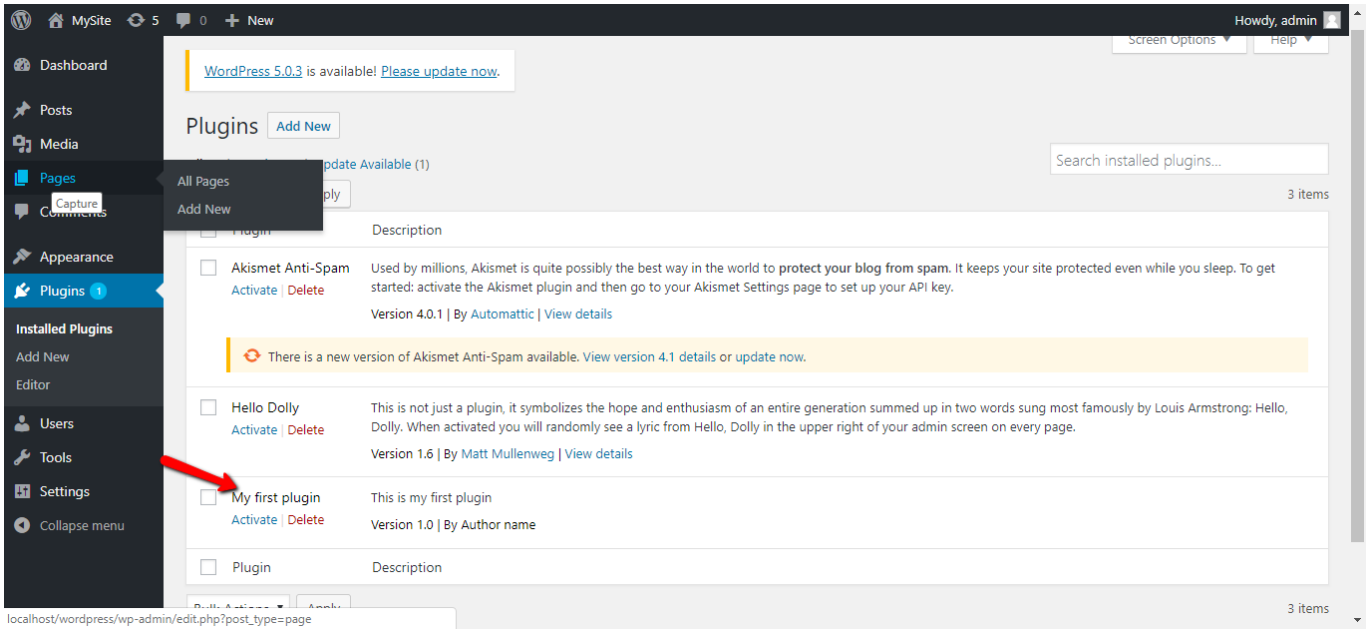
Author: Author name

Version: 1.0

*/

?>

Now when you go to plugins section in wordpress you will see your plugin in the list. See below screenshot:



STEP 3 : Writing Plugin's functions

To write plugin's functions, it is better practice to place plugin's code/functions in separate file which we called by our code. It is recommended that give all your files, functions, and variables a unique name to avoid any conflicts with other plugins. Now create new php file and save it as **functions.php**. This new file is where all your plugin's functions will be stored. We need to make it include the **functions.php** file so we can actually use the new functions. Since this is the main plugin file, including **functions.php** here makes the functions available to any other file in your plugin. Use **require_once/include** to ensure the plugin only works if the functions file is available.

Edit **index.php** as shown below then save and upload it once again, overwriting the previous version

when asked.

```
<?php

/*

Plugin name: My first plugin

Description: This is my first plugin

Author: Author name

Version: 1.0

*/

include('functions.php');
?>
```

It's a great idea to group similar functions together and write a multi-line comment above each group describing the group, followed by a short single-line comment above each function briefly describing it. That way you don't have to read through the entire code to find a function and figure out what it does. We'll name the function **add_menus.**The function will add a new top-level link to the Admin Control Panel's navigation menu.

To recap – writing a new function involves the following steps:

- Write a comment describing the function
- Name the function
- Write the function

Inside our function, we need to use the built-in WordPress function **add_menu_page()** to give our menu a name, a title and dictate who is allowed to see it. Then we tell it what to display when you go to the page. You can also give the menu link an icon and set its position in the admin control panel navigation menu – these are both optional, so we will leave them out for this tutorial. The default cog icon will be shown on the link to our page. Our link will appear at the bottom of the admin control panel navigation menu. All this information is entered as parameters of **add_menu_page()**.

In functions.php, write the following:

```
<?php
// add menu for plugin

function add_menus{
//my code goes here
}
```

?>

The five required parameters of **add_menu_page()** all appear on their own line to improve readability, in this order:

1. The title of the page you see after clicking the link (displayed in the tab in your browser)
2. Text to show as the menu link (displayed in the admin control panel navigation list), this should be the name of your plugin
3. User capability requirement to view the menu, in this example only users with the 'manage_options' capability can access the page (don't worry about this for now)
4. The fourth parameter can simply be a string of text that is displayed in the url after '**wp-admin/admin.php?page='**. If you enter 'my-plugin-page', the URL becomes **'wp-admin/admin.php?page=my-plugin-page'**.
5. The function name to display when clicking the link. It is known as a 'slug'.

Edit **functions.php,** remove **// My code goes here,** replace it with **add_menu_page()** and give it parameters as shown below:

```
<?php
// add menu for plugin

function add_menus{
add_menu_page(
'My First Page', // Title of the page
'My First Plugin', // Text to show on the menu link
'manage_options', // Capability requirement to see the link
'myfirstplugin',// text to be displayed in URL
'plug_function' // The 'slug' – function to display when clicking the link
);
}
?>
```

To make this function actually run, we need to use the WordPress function named **add_action()** with two parameters, as described in the 'Adding Functions To An Action Hook' section of this tutorial. You may want to read over that section again before continuing.

- The first parameter is the **action hook** you want to target. In our case the action hook is **admin_menu** – this means our function is run when the Admin Menu is being generated.
- The second parameter is just the name of the function to run. The function we wrote is named **add_menus**. Note that the parentheses are NOT used here. Remember that PHP evaluates the entire script before running it, allowing you to use add_action() before defining the function named in parameter 2.

Our final file looks like this:

```
<?php
```

```
// add menu for plugin

// Hook the ‘admin_menu’ action hook, run the function named ‘mfp_Add_My_Admin_Link()’
add_action( ‘admin_menu’, ‘add_menus’ );

//Add a new top level menu link
function add_menus{
add_menu_page(
‘My First Page’, // Title of the page
‘My First Plugin’, // Text to show on the menu link
‘manage_options’, // Capability requirement to see the link
‘myfirstplugin’,// text to be displayed in URL
‘plug_function’ // The ‘slug’ – function to display when clicking the link
);
}
?>
```

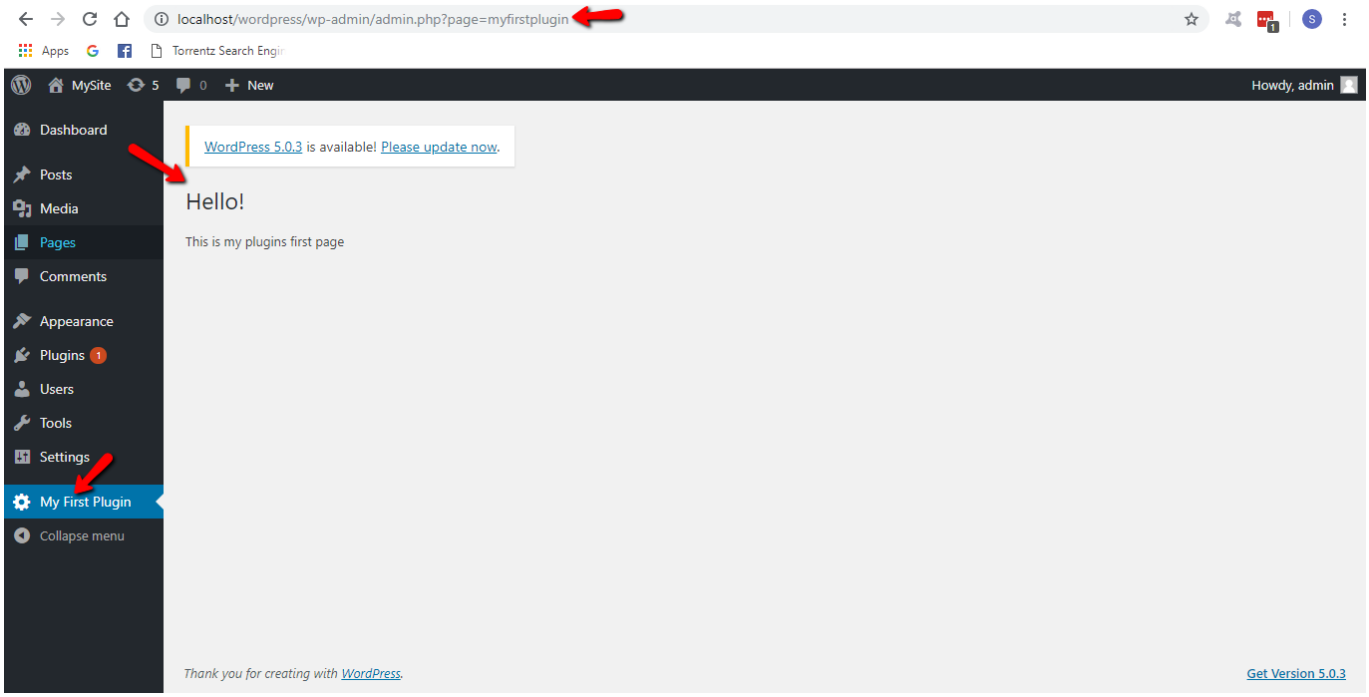
STEP 4 : Create function to display page

Now we can create the page to be displayed when you click on your admin control panel link. Go to **functions.php,** create the function **plug_function** and write your code here. Our code will look like below:

```
<?php
// add menu for plugin

// Hook the ‘admin_menu’ action hook, run the function named ‘mfp_Add_My_Admin_Link()’
add_action( ‘admin_menu’, ‘add_menus’ );

//Add a new top level menu link
function add_menus{
add_menu_page(
‘My First Page’, // Title of the page
‘My First Plugin’, // Text to show on the menu link
‘manage_options’, // Capability requirement to see the link
‘myfirstplugin’,// text to be displayed in URL
‘plug_function’ // The ‘slug’ – function to display when clicking the link
);
}

function plug_function(){
echo ‘<div class=”wrap”>
<h1>Hello!</h1>
<p>This is my plugins first page</p>
</div>’ ;
}
```

?>

Go back to your plugin list in the WordPress Admin Control Panel and activate the plugin. Once the page loads, look at the bottom of the admin control panel navigation menu. There is the brand-new link named ‘My First Plugin’. Click it, and you have your very own admin control panel page!



9 Conclusion

Let’s summarize what were the steps taken in developing this WordPress plugin from scratch:

- Create a files and folder structure for your plugin to store css, js.
- Create a plugin folder and write the Plugin header information in main plugin file.
- Add your plugin menus into administration menus.
- Create custom page which displays in admin panel.

What can go wrong

- **Adding action to one hook and calling other hook functions**
- **Same in case of filter.**