



# WordPress Plugin DEVELOPMENT

WordPress is by far the best content management system, with a fantastic framework for customisation. The use of Plugins, themes and multi-site options has made it very attractive. In this article, we will walk through creating a Plugin for WordPress. We will learn how to implement a single sign-on between WordPress and another Web application using cookies.

Plugins are used to add new features or services to the WordPress system. They allow easy customisation and the enhancement of the WordPress system. Instead of changing the core programming of WordPress, you can add functionality with WordPress plugins.

Before developing a new plugin, the first thing to do is to search the plugins repository (at <http://wordpress.org/extend/plugins/>) to find if someone has already created a plugin for this purpose.

## Creating a plugin

To create a well-structured WordPress plugin, you should follow certain steps, as given below.

### Folder

The plugin should be stored in the wp-content/plugins/ directory, relative to the WordPress installation.

### Plugin name

You should choose a unique name for your plugin. If you are planning to contribute it back to the community, you should verify that it does not conflict with any existing plugins.

### Plugin files

The plugin may comprise one or more PHP scripts. The PHP script names should be unique, and should not conflict with the names of scripts used by other plugins. You may also

logically split functions across multiple PHP scripts.

## README file

The more you document, the more it benefits your users. You should include a README file with explanations about the Plugin.

**Tip:** You can log in as a site admin in the UI, and edit the Plugin's source code. You will be able to edit all files that are part of the Plugin. You may not need shell or FTP access to modify and upload your Plugin.

## File headers

The Plugin details should be specified in the header section, in a pre-defined format. The WordPress system extracts the Plugin information from this section, and displays it to the users. For example:

```
<?php
/*
Plugin Name: Name Of The Plugin
Plugin URI: http://URI_Of_Page_Describing_Plugin_and_Updates
Description: A brief description of the Plugin.
Version: The Plugin's Version Number, e.g.: 1.0
Author: Name Of The Plugin Author
Author URI: http://URI_Of_The_Plugin_Author
License: A "Slug" license name e.g. GPL2
*/
?>
```

## Plugin development

There are two ways to do Plugin development:

- Using hooks
- Using templates

In this article, we will look at how to create Plugins using hooks. There are two types of hooks:

- Actions
- Filters

**Actions:** The core WordPress system launches the Action hook at specific points during execution, or during specific events. The Plugin can specify one or more PHP functions to execute, using the Action hook.

**Filters:** The core WordPress system launches the Filter hook to modify the text of various types before adding it to the database, or before sending it to the browser. The Plugin can specify one or more PHP functions to modify specific types of text, using the Filter hook.

**Options:** The Plugin data can be stored in a database, using the Options interface. The developer can design the Plugin to capture configuration data from the administrator, and store the values in the database.

## Single Sign-On Plugin

The use-case for creating this Plugin is to facilitate Single Sign-On (SSO) between a WordPress system and another

Web application, which should store the user credentials in cookies, perhaps in an encrypted format. The user credentials could be the username, password and email address. The Web application should set the cookies when the user logs in to the system.

The WordPress Plugin may use an authenticated filter hook to authenticate the user. This filter hook would read the encrypted user credentials from the cookie, decrypt the values, and authenticate the user using the WordPress API. The Plugin can be designed to create the account, automatically, if it does not already exist. The functions you could employ to authenticate the user, with the Authenticate filter hook, are as follows:

1. wp\_create\_user()
2. get\_userdatabylogin()
3. WP\_User class

The cookie information, like the cookie name, domain, encryption algorithm and mode, can be configured using the Options interface. This information is stored in the database. The Options interface is accessible to site administrators.




**Tip:** The Web application and the WordPress system should share a common domain name. For example, if the Web application is installed at app.foo.com, the WordPress system should be installed at blog.foo.com. The cookies should be stored using the domain name, foo.com. Otherwise, cookies are not accessible between the two systems.

Once the SSO Plugin is installed, users will not see the WordPress login screen. If they access any page that requires authentication, they will be automatically logged in, based on the values set in the cookies. If they are not logged in to the Web application, they will be redirected to the login page in the Web application. Once logged in there, they will be redirected to the WordPress system, where they will be logged in automatically. If the account does not exist in the WordPress system, it will be created automatically.



**Tip:** To allow users to log in to the WordPress site, you should create a separate login page. This URL can be used to bypass the Single Sign-On system.

To know more about WordPress Plugin development, you can refer to the Plugin Development section in the developer documentation, at [http://codex.wordpress.org/Developer\\_Documentation](http://codex.wordpress.org/Developer_Documentation). 

By: Bhuvaneshwaran A

The author is currently employed in CollabNet, managing the Release and Production Engineering group. To know more about him, check out his website: <http://www.livecipher.com>