



Diese Seite wurde noch nicht vollständig übersetzt. Bitte helfen Sie bei der Übersetzung.

(diesen Absatz entfernen, wenn die Übersetzung abgeschlossen wurde)

Basics for creating a plugin

```
$a = array(1, 2, 3, 17);

foreach ($a as $v) {
    echo "Aktueller Wert von \$a: $v.\n";
}
```

If you understand how this code works, you can develop a LoxBerry plugin!

Developing LoxBerry plugins is not too difficult for anyone who has ever scripted or programmed anything. Conveniently, there is now the Internet where you can find examples or use them directly.

Some LoxBerry plugins, for example, have emerged from already existing scripts or programs from the Internet, which primarily required a web interface for the configuration of these scripts.

If you are planning a plugin for the first time, break down the development into the following parts:

- The program that executes the actual function of your plugin.
- A configuration file from which your program reads the parameters it needs.

This gives you the true functionality. Now all the details are missing, so the next steps are:

- A web interface to read and write the configuration file
- If necessary, a cronjob to execute your program at intervals

Step 0: Update your LoxBerry to the latest pre-release

This will give you all the bug fixes and new functions on your Raspberry.

Step 1: Create plugin directory

Start by downloading (not installing!) the sample plugin and unpacking it on your PC. It contains Perl or PHP code, but any language can be used for the implementation. Perl and PHP have the advantage that LoxBerry offers an SDK for this, which makes many tasks easier (e.g. creating the web interface or reading the LoxBerry configuration and creating log files).

Step 2: customize plugin.cfg

In your copy of the sample plugin, adjust the name of the plugin, author, version number, etc. By

changing this data, it is now **your** plugin.

Step 3: Install your plugin

Zip the directory and install it on the LoxBerry. Ignore any installation errors - these come from the sample data of the sample plugin and can be corrected later. You will then find your plugin in the plugin administration. LoxBerry has created directories corresponding to the name of your plugin on the Raspberry.

Folder	Function
/opt/loxberry/webfrontend/htmlauth/plugins/ <i>deinplugin</i> /	Directory for your web interface (authentication required)
/opt/loxberry/webfrontend/html/plugins/ <i>deinplugin</i> /	Directory for Websites without authentication
/opt/loxberry/templates/plugins/ <i>deinplugin</i> /	Especially when using Perl and HTML::Template, you store HTML templates here. The lang subdirectory contains the Language files.
/opt/loxberry/config/plugins/ <i>deinplugin</i> /	Directory for your configuration files
/opt/loxberry/bin/plugins/ <i>deinplugin</i> /	Directory for Executable files that should not be accessible from the web
/opt/loxberry/log/plugins/ <i>deinplugin</i> /	Directory for your log files
/opt/loxberry/data/plugins/ <i>deinplugin</i> /	Directory for any other data that your plugin requires or generates

Step 4: Implement function

Decide whether the function of your plugin should be triggered time-controlled or via a web call.

- Time-controlled: Develop your program in your bin directory
- Web call: Develop your program in your webfrontend/htmlauth directory

Program your functionality and use a config file in your config directory for everything that should later be customizable by the user on the web.

For Perl and PHP, include the LoxBerry library LoxBerry::System or loxberry_system.php - use its global variables for your directories! For other programming languages, you have to write your own routines to find out your plugin directory. The name of your plugin directory is variable, so you must not hardcode it.

Look through the SDK for your language - For example, if you need the miniserver configuration from LoxBerry, there is the SDK function get_miniservers. To create log files there is LogBerry::Log or loxberry_log.php.

Step 5: Implementing the web interface

When your functions are running, develop your web interface for them.

Use the Perl-Lib LoxBerry::Web or for PHP loxberry_web.php. **Look through the functions of the SDK** and use code from the sample plugin.

Use readlanguage (Perl, PHP) to offer alternative languages for your plugin.

Step 6: Customize installation

If your things work reasonably well on the installed LoxBerry, update your plugin package:

- Copy the new program files into the directory from which you initially created the ZIP with the modified plugin.cfg.
- If special steps are required during the installation or during the plugin update, update the install scripts in this directory (e.g. save the config during the update)
- If special Debian packages are required for your plugin, update the apt file in the apt folder and add the package names there.

Step 7: Test and continue

Bevor zu deinen Quellordner neu zippst und installierst, stelle sicher, dass wirklich alle Dateien vom LoxBerry in deiner Quelle am PC sind - während des Updates werden alle Ordner am LoxBerry gelöscht, um diese bei der Installation wieder anzulegen.

Danach kannst du die Installation testen. Meist kommt man dabei auf etwaige Fehler in den Installationsscripten drauf. Du kannst danach alles so durchtesten, wie es sich dann auch bei anderen Benutzern verhält.

Step 8: Get help

Manche Sachen sind beim ersten Mal ungewöhnlich, deswegen nicht verzweifeln sondern Rat einholen.

Alle LoxBerry-Plugin-Entwickler schauen regelmäßig im LoxForum in den Entwickler-Bereich für LoxBerry: <https://www.loxforum.com/forum/projektforen/loxberry/entwickler>

Außerdem betreiben wir eine WhatsApp-Gruppe für LoxBerry-Entwickler, wo das Troubleshooting viel schneller geht. Bitte dafür mit [Christian Fenzl](#) Kontakt aufnehmen (inkl. Name und Telefonnummer).

Step 9: Public test

Schlussendlich würden wir vorerst deine Alpha/Beta-Version im LoxForum anbieten zum Testen (mit entsprechendem Disclaimer, dass es noch Alpha-Stand ist). Wenn die größeren Probleme ausgeräumt sind, solltest du im LoxWiki dein neues Plugin anlegen und dokumentieren. Machs so, wie es die anderen Plugin-Autoren machen.

Recommendations

Code management with GitHub

With the Windows software GitHub Desktop, you can host your plugin source directory as a public repository on Github. The advantages are: You have real code management with change tracking and the ability to undo changes. In addition, other developers can support you more easily with problems or errors if they have direct insight into your code.

LoxBerry's built-in update function for plugins is optimized for GitHub.

From:

<https://wiki.loxberry.de/> - **LoxBerry Wiki - BEYOND THE LIMITS**

Permanent link:

https://wiki.loxberry.de/en/entwickler/grundlagen_zur_erstellung_eines_plugins?rev=1737657090

Last update: **2025/01/23 19:31**