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/*deinplugin*/|Directory for your web interface (authentication required) | |/opt/loxberry/webfrontend/html/plugins/*deinplugin*/ |Directory for Websites without authentication | |/opt/loxberry/templates/plugins/*deinplugin*/ |Especially when using Perl and HTML::Template, you store HTML templates here. The lang subdirectory contains the Sprachdateien.| |/opt/loxberry/config/plugins/*deinplugin*/ |Directory for your configuration files | |/opt/loxberry/bin/plugins/*deinplugin*/ |Directory for Executable files that should not be accessible from the web | |/opt/loxberry/log/plugins/*deinplugin*/ |Directory for your log files | |/opt/loxberry/data/plugins/*deinplugin*/ |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.

Schritt 6: Installation anpassen

Wenn am installierten LoxBerry deine Sachen halbwegs funktionieren, aktualisiere dein Plugin-Paket:

- Kopiere die neuen Programmdateien in das Verzeichnis, wovon du anfangs das ZIP mit der geänderten plugin.cfg erstellt hast.
- Wenn während der Installation, oder während des Plugin-Updates spezielle Schritte nötig sind, aktualisiere in diesem Verzeichnis die Install-Scripts (z.B. Sichern der Config während des Updates)
- Wenn spezielle Debian-Pakete für dein Plugin nötig sind, aktualisiere im Ordner apt das apt-File und füge dort die Paketnamen ein.

Schritt 7: Testen und weitermachen

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.

Schritt 8: Hilfe holen

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 FenzlKontakt aufnehmen (inkl. Name und Telefonnummer).

Schritt 9: Öffentlicher 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 gröberen Probleme ausgeräumt sind, solltest du im LoxWiki dein neues Plugin anlegen und dokumentieren. Machs so, wie es die anderen Plugin-Autoren machen.

Empfehlungen

Codeverwaltung mit GitHub

Mit der Windows-Software GitHub Desktop kannst du dein Plugin-Quellverzeichnis als öffentliches Repository bei Github hosten. Vorteile sind: Du hast eine echte Code-Verwaltung mit Änderungsverfolgung und der Möglichkeit, Änderungen rückgängig zu machen. Außerdem können dich andere Entwickler bei Problemen oder Fehlern leichter unterstützen, wenn sie direkt Einblick in deinen Code bekommen.

Die in LoxBerry eingebaute Update-Funktion für Plugins ist für GitHub optimiert.