Packaging

Overview

It is possible to install a plugin manually by copying the files using an SFTP client and modifying the database tables. It is more efficient to create a package file in the form on a composer.json document that will allow the Installer to do this for you. This package file resides in the top-level of your plugin's directory and contains a variety of information:

- basic descriptive details about your plugin (i.e. name), and optionally, a description, copyright and license information.
- the extension type (component, module, plugin, template)
- a destined install directory

Composer Manifest

This composer.json file just outlines basic information about the plugin such as the owner, version, etc. for identification by the installer and then tells the installer which files should be copied and installed.

A typical component manifest:

```
{
  "name": "myorg/plg_system_example",
  "description": "Example system plugin",
  "license": "MIT",
  "type": "hubzero-plugin",
  "extra": {
    "install-directory": "/plugins/system/example/"
  }
}
```

The hub includes some extra code that tells Composer where/how to install extensions, so it's important to use the designated types. Available types are: hubzero-component, hubzero-module, hubzero-plugin, hubzero-template. Unlike other extensions, the "install-directory" parameter found under "extra" is required for plugins.

Structure

Packaging a plugin for distribution is relatively easy. The file and directory structure is exactly as it would be after installation. Here's what a typical package will look like:

```
/plg_{type}_{name}
  {name}.php
  {name}.xml
  composer.json
```

XML Manifest (deprecated)

All plugins should include a manifest in the form of an XML document named the same as the plugin. So, a plugin named test.php would have an accompanying test.xml manifest.

```
<?xml version="1.0" encoding="utf-8"?>
<extension version="1.7" type="plugin" group="system">
 <name>System - Test</name>
 <author>Author</author>
 <creationDate>Month 2008</creationDate>
 <copyright>Copyright (C) 2008 Holder. All rights reserved.</copyright
>
 <license>GNU General Public License</license>
 <authorEmail>email</authorEmail>
 <authorUrl>url</authorUrl>
 <version>1.0.1</version>
 <description>A test system plugin</description>
 <files>
  <filename plugin="example">example.php</filename>
 </files>
 <config>
  <fieldset>
     <field name="example"
      type="text"
      default=""
      label="Example"
      description="An example text parameter" />
  </fieldset>
 </config>
</extension>
```

Let's go through some of the most important tags:

INSTALL/EXTENSION

This tag has several key attributes. The type must be "plugin" and you must specify the group. The group attribute is required and is the name of the directory you saved your files in (for example, system, content, etc). We use the method="upgrade" attribute to allow us to install the extension without uninstalling. In other words, if you are sharing this plugin with other, they can just install the new version over the top of the old one.

NAME

We usually start the name with the type of plugin this is. Our example is a system plugin and it has some some nebulous test purpose. So we have named the plugin "System - Test". You can name the plugins in any way, but this is a common format.

FILES

The files tag includes all of the files that will will be installed with the plugin. Plugins can also support be installed with subdirectories. To specify these just all a FOLDER tag, <folder>test</folder>. It is common practice to have only one subdirectory and name it the same as the plugin PHP file (without the extension of course).

PARAMS/CONFIG

Any number of parameters can be specified for a plugin. Please note there is no "advanced" group for plugins as there is in modules and components.