Packaging

Overview

It is possible to install a component 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 component's directory and contains a variety of information:

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

Composer Manifest

This composer.json file just outlines basic information about the component 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/com_example",
 "description": "Example component",
 "license": "MIT",
 "type": "hubzero-component"
}
```

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.

Structure

Packaging a component 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:

```
/com_{componentname}
```

```
{componentname}.xml
composer.json
/site
  {componentname}.php
  controller.php
  /views
    /{viewname}
      /tmpl
        default.php
  /models
    {modelname}.php
  /controllers
    {controllername}.php
/admin
  {componentname}.php
  controller.php
  /views
    /{viewname}
      /tmpl
        default.php
  /models
    {modelname}.php
  /controllers
    {controllername}.php
```