A Joomla! Extension Build Script

Introduction

This phing script is designed to automate two of the key steps of a workflow (shown in the diagram below) used in the development of Joomla! extensions.

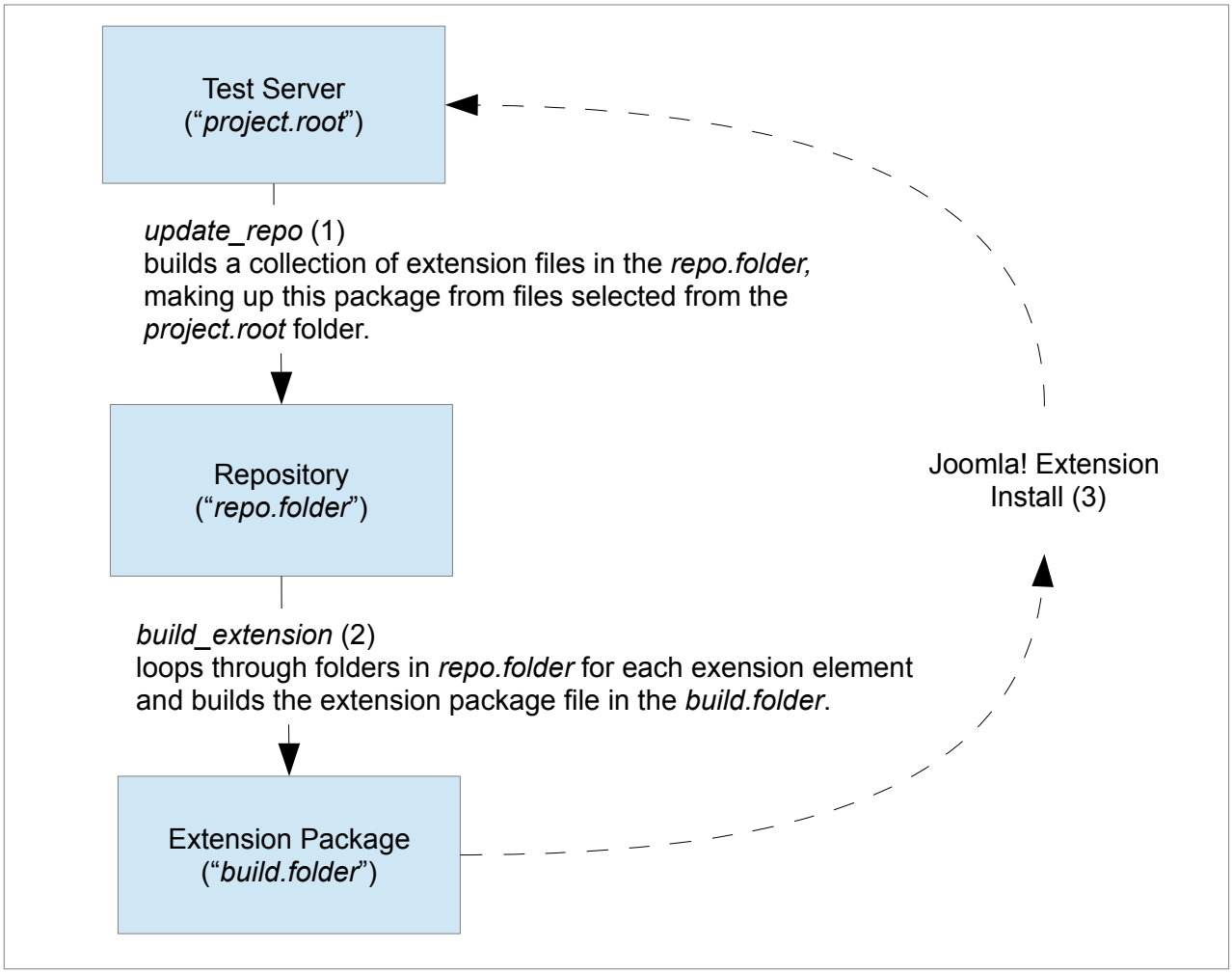
1. *update_repo* – moves updated or edited extension code files to a repository folder (or just a convenient holding folder),

2. *build_extension* – builds an installable package file (.ZIP) from the extension files in the repository.

3. the third step in the workflow is the test installation of the built extension package using normal Joomla! installer, prior to further code test and development, or as test of the viability of the package prior to distribution.

It also includes a number of automation features such as auto-update of version number and build date for PHP file headers and .XML package manifest files.

NB – both parts of the script assume that, for packages of two or more elements, there will be a suitable manifest file (e.g. *pkg_name.xml*) already present in the repository.



System Requirements

The script has been developed and tested in a Linux environment with phing V2.4.12

Properties files

The operating of the script is determined by two property files – *build.properties* and *project.properties*.

build.properties is a standard phing *build.xml* file. With this script it is used to define the locations of files and folders used in the workflow. These are :

1. the Joomla development/test server web root folder (*project.root*).
2. the project repository location – or it may be just a temporary folder (*repo_folder*)
3. the location to be used for the extension build. The finished package file is left in a sub-folder of this location (*build.folder*).
4. the folder that contains the *project.properties* (see below) file (*project.properties.folder*).

The *project.properties* file defines :

1. the project name (used to label the holding folder in the *repo.folder*. Is used to name the extension package file if this is a multi-element package.
2. a comma-separated list of elements (e.g. component(s), plugin(s), module(s)) making up the extension. (May be just one item if a simple extension) The list names must follow the naming conventions described below.
3. the version number of the code being processed.

(See the example project.properties file that accompanies this script.)

Naming conventions

The script expects the component(s) of the extension to be listed as the *element.list* property in the *project.properties* file in a specific format i.e. :

- component : "com_"
- module : "mod_"
- template : "tpl_"
- plugin: "plg_GROUP_" (GROUP = "system", "authentication", etc.)

Administrator modules and templates must use the following prefixes :

- module: "mod_admin_"
- template: "tpl_admin_"

Language files

This script includes the language files found in the extension folders and also searches globally for associated language files of the form *element.name.ini*.

Usage examples

1. Updating extension files in a repository with edited/updated files from a Joomla development/test server:

   ```
   $ phing -f /path/to/buildfile/build.xml update_repo
   ```

2. Creating an extension package from the extension files in a repository:

   ```
   $ phing -f /path/to/buildfile/build.xml build_extension
   ```

References

- phing online documenation at http://www.phing.info/docs/guide/stable/.

Acknowledgements

- Development of this script has drawn heavily on the code and ideas from here : http://www.twilight-zone.com/2012/04/building-joomla-extensions-with-phing/
