How to Build a Plugin

0.1

This How-To describes the steps necessary to build a plugin for Forrest. Forrest uses plugins to add new input formats, output formats and to change its default behaviour. Since plugins are downloaded when needed and can be hosted at any location plugin code can be developed independantly of Apache Forrest. This how-to describes each of the major steps in creating a plugin and then works through some examples of plugin creation in order to illustrate the materials.

Table of contents

0.1

1. Overview

This How-To describes the steps necessary to build a plugin for Forrest. Forrest uses plugins to add new input formats, output formats and to change its default behaviour. Since plugins are downloaded when needed and can be hosted at any location plugin code can be developed independantly of Apache Forrest. This how-to describes each of the major steps in creating a plugin and then works through some examples of plugin creation in order to illustrate the materials.

1. Intended Audience

Users needing to add additional input formats or output formats or to change the way Forrest internals work.

Warning:

The Plugin Infrastructure is still at an early stage of design and implementation, consequently this How-To *may* be out of date and/or incomplete. If you are having problems with any of the steps described, please ask for help on the developers mailing list (and then provide patches for this document).

Warning:

Please make sure that you are using forrest 0.7-dev if you want use plugins. Forrest 0.6 will not work!!!

2. Purpose

This How-To will illustrate how to build a plugin, publish a plugin and configure a Forrest project to use their plugin.

3. Prerequisites

Plugin developers should have:

- a basic knowledge of XML, XSLT and Cocoon pipelines
- a clear use-case for extending Forrest
- verified with the Apache Forrest developer community that the requried functionality does not already exist

4. Steps

Here is how to proceed.

4.1. Type of Plugin

There are three types of plugin, each with a clear purpose, you must first decide which type of plugin you need to build.

4.2. Seed a New Plugin

Regardless of the type of plugin you are building, the directory structure is almost identical, as are most of the requried configuration files. In this How-To we will assume that you are creating a plugin in the Forrest source tree. All plugins are developed in the `forrest/plugins` directory.

Run the following set of commands:

```
cd [path_to_forrest]/plugins
ant seedPlugin
```

The above ant target will ask you the name of the plugin and will build a minimal plugin directory structure and configuration. You will need to customise these files to build your plugin.

Note:
Although you can name your project anything you like we do have some naming coventions that we recomend you follow.

Note:
If you plan on building your plugin elsewhere you can copy the `build.xml` build file to your own plugin work directory and use it there.

See What Does a Forrest Plugin Look Like? for more information on the plugin directory structure and configuration files.

4.2.1. Edit the Plugin Template

You now have a skeleton plugin project. However, it doesn't do anything useful yet. Now is a good time to edit some of the files provided. For example:

4.2.1.1. status.xml

This file is used to track changes to the plugin project and to manage lists of things that still need to be done. At this stage you should correct the `person` entry near the top of the file. It is also a good idea to add a few key milestones in the task list towards the bottom of the file.

As you work on the plugin you should record all major changes in this file so that it can then be used as a changelog for your plugin.

4.2.1.2. forrest.properties

This file defines many configuration parameters for Forrest. It does not need to be customised in most cases. However, see for more details.

4.2.1.3. src/documentation/skinconf.xml

This configures the skin for your plugins documentation. There are some items that need to be configured in here, for example, the copyright information. The file is heavily commented so probably best to read through it, changing what you need to.

4.2.1.4. Documentation

It is also a good idea to start writing the documentation at this stage. The above process created a very simple plugin documentation site for you. All you have to do is add the content.

4.3. Edit the Plugin XMap file(s)

The plugin `xmap` file is a Cocoon sitemap that is mounted at a strategic place in the Forrest pipeline. It is in this file that you will instruct Forrest how to operate. An input plugin must provide a `input.xmap` file, an output plugin must provide a `output.xmap` file, whilst an internal plugin

provides a `internal.xmap` file. In addition, an input plugin may provide a `resources.xmap` file to allow the plugin to handle items such as JavaScript files.

It is beyond the scope of this How-To to give details about how to build your plugins XMap. See the Sitemap Reference for general information. See also Plugin Infrastructure for some hints and tips on creating plugin sitemaps. In addition, as with all development work on Forrest, you will find the developer mailing list a very doog resource (check the archives before posting, please).

4.3.1. Components, Actions and Resources

If your plugin uses any components (i.e. generators, transformers or serializers), actions or resources they must be defined in either the plugins xmap or one of its parents. The parents of an `input.xmap` are `sitemap.xmap` and `forrest.xmap`, whilst the parent of both `output.xmap` and `internal.xmap` are `sitemap.xmap`.

If you want to use the realpath where the sitemap.xmap of your plugin resides then you have to use `{forrest:plugins}/PLUGIN_NAME` instead of `{realpath:/}`.

See the examples below for more details.

4.4. Create the Necessary Resource Files

FIXME (rdg):

Discuss the XSL files and other such resources

4.5. Create Samples in the Documentation

Plugin documentation should provide (as a minimum) an index page that provides an overview and a set of samples that demonstrate the functionlaity of the plugin. Typically these samples will be provided in a `samples` subdirectory in the plugin documentation and will be referenced from both `site.xml` and `tabs.xml`.

Try to provide a sample for all the major functions of your plugin and document any configuration that is available.

4.6. Testing a Plugin

Since your documentation for the plugin illustrates all of its functionality you can use that site for testing the plugin. However, you must first deploy in your local install of Forrest. Each plugin contains a buildfile that includes a `test` target. This target, by default, builds the documtantion for your plugin.

Run the command `ant test` in the plugins directory.

Of course, the build should complete without errors.

Note:

You can also use `forrest run` to interactively examine your documentation (point your prowser at http://localhost:8888).

It is also a really good idea to build proper tests for your plugins using a suitable testing framework, for example, WebTest. We recomend that you extend the `test` target in your plugins build file because this target is also used when performing integration tests on Forrest. In addition, we recomend

that you use the samples in your documentation for your tests, this way you are documenting your plugin at the same time as writing your tests.

4.7. Releasing a Plugin

4.7.1. Register the Plugin with Apache Forrest

FIXME (rdg):
Describe the plugins.xml file

FIXME (rdg):
Describe making a request of Forrest devs for inclusion

4.7.2. Deploying the Plugin

To deploy the plugin so that others can use it, it must be made available as a zip from the URL indicated in the `plugins.xml` file. The plugins build file provides targets to assist with this task.

To deploy a plugin simply run the command `ant deploy` from within the plugin directory.

This command will, by default, deploy to the Apache Forrest web site. In order to do this you need write access to Forrest. If you want to deploy your plugin to a different location you can build the zip of your plugin with `ant dist` and then copy the zip file from `build/dist` to wherever you intend to host the plugin.

Note:
Running this command on any plugin will also deploy any changes to the `plugins.xml` file. If you are deploying to your own website you will have to request changes to the `plugins.xml` and have a Forrest committer publish the new document.

Warning:
Running the `deploy` or `dist` targets will always run the `test` target first. This is to ensure that your only deploy working plugins. This adds a little time to the deploy cycle, but we feel the peace of mind is worth it.

4.8. Examples

This section will provide some example plugins to help illustrate the steps discussed above.

4.8.1. Input Plugin

FIXME (RDG):
Discuss OpenOffice.org plugin here

4.8.2. Output Plugin

FIXME (RDG):
Discuss s5 plugin here

4.8.3. Internal Plugin

FIXME (RDG):

Discuss IMSManifest plugin here

4.9. Further Reading

- Plugin Infrastrucuture Documentation for Developers
- Plugins Documentation for users

4.10. Summarise the Entire Process

FIXME (rdg):

In a few sentences, remind the reader what they have just learned. This helps to reinforce the main points of your How-To.