How to Build a Plugin

0.3.0

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 independently 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

1. Intended Audience

Users needing to add additional input formats or output formats or to change the operation of the Forrest internals.

Warning:
Please make sure that you are using forrest 0.8 or above 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
- read Plugin Infrastructure
- verified with the Apache Forrest developer community that the required 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. Make ant available on the command-line

The following instructions rely heavily on Apache Ant to automate some steps in the process. Since ant is distributed as part of Forrest, all you need to do is add Forrest's 'ant' executable directory to your system path. The name of this directory is `tools/ant/bin` in your Forrest program directory. Alternatively you can prefix all calls to ant in the following instructions with the full path of the ant binary directory, i.e. $FORREST_HOME/tools/ant/bin/ant

If instead you really want to use your own version of Ant, then you will need to copy forrest/lib/core/xml-commons-resolver.jar to $ANT_HOME/lib directory, otherwise the building of your plugins will go across the network to get the DTDs on every xml parse. Be aware that Forrest might be relying on some Ant features in its version.

4.3. Seed a New Plugin

Regardless of the type of plugin you are building, the directory structure is almost identical, as are most of the required 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 or the `forrest/whiteboard/plugins` directory.

Run the following set of commands:

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

The above ant target will ask you the name of the plugin and some additional information such as a brief description 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 conventions that we recommend you follow. Plugins intended to be held at forrest.apache.org must follow the naming convention.

You can also build your plugins from a location outside of the Forrest directory structure, for example from within your own project. If you don't already have one, create a plugins directory, for example:

```
cd $PROJECT_HOME
mkdir plugins
```

Then copy $FORREST_HOME/whiteboard/plugins/build.xml to $PROJECT_HOME/plugins. There are a couple of changes you now need to make to the newly copied build.xml file. Open up 'build.xml' for editing. You can change the project name value to something more suitable.Find the property name for forrest.plugins.dir and change the location to read

```
location="."
```

So, revised commands for $PROJECT_HOME/plugins:

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

See Plugin Infrastructure for more information about the plugin directory structure and configuration files.

4.3.1. 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.

Here are some general notes:

4.3.1.1. 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.

4.3.1.2. 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.

4.3.1.3. 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.

4.3.1.4. 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.

4.3.1.5. Style notes for plugins hosted at forrest.apache.org

After seeding a new plugin, copy the configuration from an existing plugin (e.g. org.apache.forrest.plugin.input.projectInfo). Copy src/documentation/skinconf.xml (and edit to suit) and src/documentation/content/xdocs/images/project-logo.gif

4.4. Edit the Plugin sitemap 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.

Note:
All input plugins should allow the original source to be retrieved by requesting the document with a `*.source.xml` extension. So you should ensure that you provide such a match.

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 good resource (check the archives before posting, please).

4.4.1. 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 xmap for this plugin 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 need to use `{forrest:forrest.plugins}/PLUGIN_NAME` instead of `{realpath:/}`.

See the examples below for more details.

4.5. Create the Necessary Resource Files

FIXME (open):
Discuss the XSL files and other such resources

4.5.1. Entity catalog for DTDs and other resources

If the plugin uses non-core DTDs and other entities, then add them to the `resources/schema` directory and configure a catalog.xcat file. The best way to do this is to copy an example from another plugin (e.g. "listLocations" has a simple example; "glossary" has a more complex example) and edit it to suit.

4.6. 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 functionality 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` configuration files.

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

4.7. 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 documentation 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 browser 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 recommend 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 recommend 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.

Ensure that your sitemaps are robust and handle matches for files in sub-directories, as well as those at the root level.

4.7.1. Testing During Development

In the current plugin system plugins are not used from their src directories, they must first be deployed locally. To do this run the command `$FORREST_HOME/tools/ant/bin/ant local-deploy`

Note:
The "test" target will do this deployment automatically for you. You need only run it manually if you wish to test the plugin whilst editing content in a live Forrest instance.

When you make changes to the plugin while doing its development, then you need to do the local-deploy again for those changes to have effect.

In most cases you can locally deploy a plugin without having to restart Forrest. However, if your plugin changes any configuration files in the `conf` directory you will, most likely, have to restart Forrest to see these changes.

4.8. Releasing a Plugin

4.8.1. Register the Plugin with Apache Forrest

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

4.8.2. 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 commit 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 ask the Forrest committers to publish the new document.

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

4.8.3. Managing the plugins descriptors

The files plugins/plugins.xml and whiteboard/plugins/plugins.xml are the "Plugins Descriptor" files. Each plugin is described with its name, purpose, location, and version information. These descriptors are deployed to the forrest website.

Each plugin has a build.xml file which defines its version information. Please keep that synchronised with the plugins.xml files. Later FOR-533 will generate this from the various build.xml files.

The Apache Forrest committers manage these files in SVN and publish them as needed. Here are some notes.

When a plugin gains new functionality, then it will be dependent on a more recent version of Forrest. Deploy the plugin one final time before implementing the new work. For example, if current release is 0.7 then ...

- Review the docs and ensure any version numbers in text are "0.7"
- Edit the skinconf.xml to ensure that all version numbers are "0.7", e.g. the MOTD.
- Edit the plugin's descriptors to ensure that the "forrestVersion" is 0.7 and that the "version" is appropriate.
- Ensure that the "website" parameter includes "pluginDocs/plugins_0_70"
- Edit status.xml to set the release date. Ensure that the changes notes are complete.

Now the plugin gains functionality that binds it to 0.8-dev (e.g. converted to use locationmap) so ...

- Review the docs and ensure any version numbers in text are "0.8"
- Edit the skinconf.xml to ensure that all version numbers are "0.8-dev", e.g. the MOTD.
- Edit the plugin's descriptors to ensure that the "forrestVersion" is 0.8 and that the "version" is incremented.
- Ensure that the "website" parameter includes "pluginDocs/plugins_0_80"
- Edit status.xml to add a new section and set the release date. Start adding changes notes.

4.9. Experimental Functionality

Warning:

This section describes functionality that is considered experimental. This functionality may be defective and is not part of the official release at this time, use at your own risk. If you do choose to use this functionality then we recomend that you join the Forrest dev list in order to keep abreast of the changes as they occur.

Note:

For an example of each of these features in use see the `org.apache.forrest.internal.NoteTaking` plugin.

4.9.1. Locationmap

Plugins can use the Forrest locationmap to expose resources to your project and other plgins. To use this functionality add your `locationmap.xml` file to the root of the plugin directory.

We have an issue for the status of locationmap development.

4.9.2. Dispatcher

Dispatcher (previous codename Forrest Views) is the collective name for the various pieces of functionality that are intended to replace skins in the future. They allow for a much more configurable system of defining the contents and look and feel of a site.

Plugins can expose contracts, resources and tiles for use in structurer files used within Dispatcher-based sites. In order to do this you should develop your contracts as normal and place them in `PLUGIN_HOME/resources/themes`. However, this, by itself, is not sufficient to export your contracts. You will also need to add the following match to your plugin's `locationmap.xml` file:

```
    <match pattern="resolvePluginContract.*.**">
      <select type="exists">
        <location
src="{forrest:forrest.plugins}/PLUGIN_NAME/resources/themes/{properties:dispatcher.theme}/
```

```
/>
        <location
src="{forrest:forrest.plugins}/PLUGIN_NAME/resources/themes/{properties:dispatcher.fallbac
/>
      </select>
    </match>
    <match pattern="resolvePluginThemes.*.**">
      <select type="exists">
        <location
src="{forrest:forrest.plugins}/PLUGIN_NAME/resources/themes/{properties:dispatcher.theme}/
/>
        <location
src="{forrest:forrest.plugins}/PLUGIN_NAME/resources/themes/{properties:dispatcher.fallbac
/>
      </select>
    </match>
    <match pattern="resolvePluginTiles.**">
      <select type="exists">
        <location
src="{forrest:forrest.plugins}/PLUGIN_NAME/resources/themes/{properties:dispatcher.theme}/
/>
        <location
src="{forrest:forrest.plugins}/PLUGIN_NAME/resources/themes/{properties:dispatcher.fallbac
/>
      </select>
    </match>
```

Of course, you should replace `PLUGIN_NAME` with the name of your plugin.

Once Dispatcher becomes stable we will add this matches to the default locationmap which is generated when you seed a new plugin, but for now it must be done manually.

4.9.3. Plugin Properties

Plugins can define properties that each project can over-ride. For more information see the issue below.

We have an issue for the status of this new configuration system.

4.10. Examples

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

4.10.1. Input Plugin

FIXME (RDG):

Discuss OpenOffice.org plugin here

4.10.2. Output Plugin

FIXME (RDG):

Discuss s5 plugin here

4.10.3. Internal Plugin

FIXME (RDG):

Discuss IMSManifest plugin here

4.11. Further Reading

- Plugin Infrastructure Documentation for Developers
- Plugins Documentation for users

4.12. Summarise the Entire Process

FIXME (open):

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