Extending Forrest with Plugins

Table of contents

1 Overview....2
1.1 What plugins are available?....2
2 How is a Plugin Installed?....2
2.1 List of Plugins Needed by the Project....2
2.2 What Version of Plugins will be used ?....2
2.3 Where does Forrest look for Plugins sources?....3
3 Editing plugins sources to enhance functionality....4
4 Upgrading from a Version of Forrest Without Plugins....4
5 Avoiding Plugin Conflicts....5
6 Further Reading....5

1 Overview

Forrest provides the core functionality for generating documentation in various output formats from a range of input formats. However, it does not end there. Forrest can be extended through the addition of plugins. This document serves as an introduction to the Forrest plugin mechanism.

1.1 What plugins are available?

You can run the command `forrest available-plugins` to get a list of the known plugins for Forrest.

If you would like to have your own plugin added to this list then contact the developer mailing list.

2 How is a Plugin Installed?

2.1 List of Plugins Needed by the Project

If a site requires one or more plugins then the site designer will have to list them in the `project.required.plugins` property in the projects `forrest.properties` file. When Forrest builds the site it will automatically discover the plugins and install them. In otherwords, the user needs do nothing.

For example,

```
project.required.plugins=org.apache.forrest.plugin.output.pdf,
  org.apache.forrest.plugin.input.simplifiedDocbook
```

(note that that is all in one line with no spaces) will cause Forrest to load the plugins called "org.apache.forrest.plugin.output.pdf" and "org.apache.forrest.plugin.input.simplifiedDocbook".

> **Note:**
> By default a new forrest project has that property configured to include some plugins. Currently there is only one to generate PDF output from your source documents.

2.2 What Version of Plugins will be used ?

In the absence of a version number for the plugin (as is the case in the example above) the most recent version that is applicabe to your release of Forrest will be used. This may result in unexpected behaviour if a new version of the plugin has been released that is incompatible with your current site. To force Forrest into using a specific version of a plugin you should add "-VERSION_NUMBER" to the end of the plugin name. For example, to force forrest to use the version 0.2 of the "output.pdf" plugin you would use `org.apache.forrest.plugin.output.pdf-0.2`. If you define a version of the plugin that does not exist then it will fall back to using the most recent version available. This feature is useful when developing a new site as you can quickly force a plugin upgrade by deleting all installed plugins (use the command 'ant cleanPlugins'). However, this might result in the installation of an in-development plugin, therefore in a production environment you should always specify a known working version.

2.3 Where does Forrest look for Plugins sources?

2.3.1 Overview

This is fairly complex, so here is a simple overview. If you want to know the details you need to read the rest of this section. For most people this overview will be sufficient.

Forrest will try and retrieve a plugin from local source files first, if that fails it will look on a remote distribution server. Once it finds a copy of the plugin it will deploy it to the local Forrest instance and use it from there.

The local directories that will be searched are defined in the `project.required.plugins.src` property, which is defined in the `forrest.properties` file. By default this is set to `${forrest.home}/plugins,${forrest.home}/whiteboard/plugins`, which means that these two directories will be searched for plugins.

If you have a collection of local plugins the directory in which they are located must be added this property. For example, if your local plugins are in `/export/forrest_plugins` this property should be something like: `project.required.plugins.src=${forrest.home}/plugins,${forrest.home}/whiteboard/plugins,/export/forrest_plugins`

You can add this line to your projects `forrest.properties` file. However, it may be more convenient to add it to a `forrest.properties` file in your users home directory.

Forrest uses a fall back mecanism to find the plugins to install for a project.

For an unversionned plugin, Forrest tries to get it from :

1. different local sources directories (`project.required.plugins.src` property)
2. if not found : the remote site in the forrest version directory
3. if not found : the remote site (with no forrest version directory)

For a versionned plugin, Forrest tries to get :

1. the versionned plugin from different local sources directories (`project.required.plugins.src` property)
2. if not found : the versionned plugin from the remote site in the forrest version directory
3. if not found : the unversionned plugin in different local sources directory (`project.required.plugins.src` property again)
4. if not found : the unversionned plugin from the remote site in the forrest version directory
5. if not found : the remote site (with no forrest version directory)

By default, forrest looks into the two following directories to find plugins sources : `${forrest.home}/plugins` and `${forrest.home}/whiteboard/plugins`. It is possible to add other sources locations by specifying the `project.required.plugins.src` property in the projects `forrest.properties` file.

For example,

```
project.required.plugins.src=${forrest.home}/plugins,${forrest.home}/whiteboard/plugins,/export/
forrest_plugins
```

will add the project specific directory `${project.home}/plugins` to the list of directories to search in.

If sources are not found, forrest will try to get them from the Web. Forrest *knows* the plugins description with plugins descriptors files in which plugins are described as follows :

```

<plugin name="org.apache.forrest.plugin.output.pdf"
```

```
  type="output"
  author="Apache Forrest Project"
  website="http://forrest.apache.org/pluginDocs/plugins_0_80/org.apache.forrest.plugin.output.pdf"
  url="http://forrest.apache.org/plugins/"
  version="0.2">
  <description>
    Enable Forrest documents to be output in PDF format.
  </description>
  <forrestVersion>0.8</forrestVersion>
</plugin>
```

The url to download the different plugins is indicated in this file.

By default, forrest gets the two following plugins descriptors files : `http://forrest.apache.org/plugins/plugins.xml` and `http://forrest.apache.org/plugins/whiteboard-plugins.xml`. It is possible to add other plugins descriptors files by specifying the `forrest.plugins.descriptors` property in the projects `forrest.properties` file.

For example,

```
forrest.plugins.descriptors=http://forrest.apache.org/plugins/plugins.xml,http://forrest.apache.org/
plugins/whiteboard-plugins.xml,file:///${project.home}/plugins/plugins.xml
```

will add the project specific plugins descriptors file `file:///${project.home}/plugins/plugins.xml` to the list of descriptors.

3 Editing plugins sources to enhance functionality

> **Note:**
> Of course, developers would need the sources. They are available via SVN. Alternatively, see your $FORREST_HOME/build/plugins/ directory. Copy them somewhere for your development and set the project.required.plugins.src property as described above. The SVN method is preferred.

> **Note:**
> Until issue FOR-388 is fixed to enable the use of plugins in-place, any change to sources requires you to either restart forrest or to manually deploy the plugin locally with "ant local-deploy". See Further reading for "How to build a Plugin". It is worth noting that if your changes are to Java files you will always have to restart Forrest to ensure the class loader loads your new classes.

If you need to add specific behaviour to an existing plugin, you should first consider whether your changes will be of use to all users of the plugin or not. If they are of general use then you can edit the plugin source files in their original location (i.e. not in the build directory). Once you have completed your changes please prepare a patch and submit it for inclusion in code base.

If your changes are specific to your own use of the plugin you can create a local copy of the plugin for this. However, we strongly advise against this since you will need to manually update your plugin each time a new release of the original is made. In the vast majority of cases local enhancements to a plugin wil be of use to the wider communtiy. Please donate back to the project and help keep it vibrant and useful. See the Further Reading section at the end of this document for links to documents with more information.

4 Upgrading from a Version of Forrest Without Plugins

The plugin functionality was introduced in version 0.7 of Forrest. At this time some of the functionality previously in Forrest was extracted into a plugin. However, we have not broken

backward compatability with earlier versions. In the absence of a `project.required.plugins` property in the projects `forrest.properties` file all plugins that contain functionality previously part of Forrest itself will be loaded automatically. Unless you intend to use new functionality provided by a plugin you will not need to make any changes to your project.

If you do require additional plugin functionality, be sure to include all required plugins in the `project.required.plugins` property in the project's `forrest.properties`. You can view `main/webapp/default-forrest.properties` to see the names of plugins that provide previously core functionality.

It is also worth noting that there is a small performance improvement if you remove plugins that are not in use. Therefore, if you do not use one or more of the plugins named in the `project.required.plugins` property of `main/webapp/default-forrest.properties` it is recomended that you override this value in your project's `forrest.properties` file.

5 Avoiding Plugin Conflicts

Clashes between plugins can occur. For example, the simplified-docbook and full docbook plugins may try and process the same files. In this instance the one that is mounted first will take precedence. Plugins are mounted in the order they appear in the `project.required.plugins` property, therefore the mounting order and therefore processing precedence is under user control.

6 Further Reading

- Plugin Infrastructure
- How to build a Plugin