lexicons will be read. Defaults to lexicon/. Values may be relative (to the application's directory) or absolute (starting with / at the web root). Each lexicon will have its own directory within lexiconPath.

The <globalfuseactions> Section

This section specifies fuseactions to be run in combination with whatever fuseaction is explicitly called. There are three subelements allowed in this section:

- appinit – New to Fusebox 5.1. Specifies fuseaction(s) to run when the application initializes.
- preprocess – Specifies fuseaction(s) to run before every requested fuseaction.
- postprocess – Specifies fuseaction(s) to run after every requested fuseaction.

In each case, any number of <fuseaction> elements may be specified inside the subelements above. For example, to run a header and footer fuseaction before an after each fuseaction, respectively, we could have a <globalfuseactions> section that looks like this:

```
<globalfuseactions>
  <preprocess>
    <fuseaction name="layout.showHeader" />
  </preprocess>
  <postprocess>
    <fuseaction name="layout.showFooter" />
  </postprocess>
</globalfuseactions>
```

So global fuseactions can be run in either the preprocess or postprocess mode, or both. We can also have appinit global fuseactions, which run only when the application initializes.

The <plugins> Section

Plugins are somewhat like global fuseactions, with one important exception: plugins are comprised of code that is not contained within a fuseaction in the application. So a plugin can be any chunk of CF code that we want to run at a specific point in the Fusebox process.

There are six defined plugin points in the process: preProcess, preFuseaction, postFuseaction, postProcess, processError, and fuseactionException. Their names indicate where in the Fusebox process each is fired—before or after the whole

process, and before or after the fuseaction code. The two that might be a bit confusing are processError and fuseactionException.

A fuseaction exception is just an exception that occurs within the logic of a fuse. That is, it's a problem with the application code. A plugin for the fuseactionException plugin point supersedes the default Fusebox error handling described later in this chapter.

A process error is a more serious condition, representing a problem that occurs within the Fusebox framework itself. It is differentiated from a fuseaction exception so that different responses can be specified for each condition.

Within each subsection, a plugin can be defined with the following syntax:

```
<plugin name="pluginName" template="pluginFile"/>
```

Each plugin needs to have a unique value in the name attribute. The template attribute specifies the name of the CFML template to use. Plugin templates are located in the plugins directory specified in the <parameters> section of fusebox.xml.

Plugins may also accept values passed to them through the use of the <parameter> tag. So a plugin declaration might also look like this:

```
<plugin name="pluginName" template="pluginFile">
  <parameter name="parameterName" value="parameterValue" />
</plugin>
```

Just as the application's configuration is controlled by fusebox.xml, each circuit's configuration is controlled by its circuit.xml file.

The Circuit Configuration File(s) (circuit.xml)

Each circuit in a Fusebox application carries its own configuration file, named circuit.xml. As with fusebox.xml, the circuit.xml files are typically seen with the .cfm extension appended to the file name to prevent direct browsing of the contents of the file.

Fundamentally, the circuit.xml file defines the fuseactions that exist within the circuit.