AboutPlugins

Nutch's plugin system is based on the one used in Eclipse 2.x. Plugins are central to how Nutch works. All of the parsing, indexing and searching that Nutch does is actually accomplished by various plugins.

In writing a plugin, you're actually providing one or more *extensions* of the existing *extension-points* . The core Nutch *extension-points* are themselves defined in a plugin, the NutchExtensionPoints plugin (they are listed in the NutchExtensionPoints plugin.xml file). Each *extension-point* defines an interface that must be implemented by the *extension*. The core extension points are:

- IndexWriter – Writes crawled data to a specific indexing backends (Solr, ElasticSearch, a CVS file, etc.).
- IndexingFilter – Permits one to add metadata to the indexed fields. All plugins found which implement this extension point are run sequentially on the parse (from javadoc).
- Exchange – Allows to route a document during indexing to a specific index writer if there are multiple index writers.
- Parser – Parser implementations read through fetched documents in order to extract data to be indexed. This is what you need to implement if you want Nutch to be able to parse a new type of content, or extract more data from currently parseable content.
- HtmlParseFilter – Permits one to add additional metadata to HTML parses (from javadoc).
- Protocol – Protocol implementations allow Nutch to use different protocols (ftp, http, etc.) to fetch documents, see ProtocolImplementations.
- URLFilter – URLFilter implementations limit the URLs that Nutch attempts to fetch. The RegexURLFilter distributed with Nutch provides a great deal of control over what URLs Nutch crawls, however if you have very complicated rules about what URLs you want to crawl, you can write your own implementation.
- URLNormalizer – Interface used to convert URLs to normal form and optionally perform substitutions.
- ScoringFilter – A contract defining behavior of scoring plugins. A scoring filter will manipulate scoring variables in CrawlDatum and in resulting search indexes. Filters can be chained in a specific order, to provide multi-stage scoring adjustments.
- SegmentMergeFilter – Interface used to filter segments during segment merge. It allows filtering on more sophisticated criteria than just URLs. In particular it allows filtering based on metadata collected while parsing page.

Updated to Nutch apidocs version 1.18

Source Files

You'll find the following inside of a plugin source directory:

- A plugin.xml file that tells Nutch about the plugin.
- A build.xml file that tells ant how to build the plugin.
- A ivy.xml file that describes any dependencies required by the plugin. These are subsequently managed by Apache Ivy.
- The source code of the plugin.

Getting Nutch to Use a Plugin

In order to get Nutch to use a given plugin, you need to edit your conf/nutch-site.xml file and add the name of the plugin to the list of plugin.includes. Additionally we are required to add the various build configurations to build.xml in the plugin directory.

Using a Plugin From The Command Line

Nutch ships with a number of plugins that include a main() method, and sample code to illustrate their use. These plugins can be used from the command line - a good way to start exploring the internal workings of each plugin.

To do so, you need to use the bin/nutch script from the $NUTCH_HOME directory,

$ bin/nutch plugin Usage: [PluginRepository] pluginId className [arg1 arg2 ...]

As an example, if you wanted to execute the parse-html plugin,

$ bin/nutch plugin parse-html org.apache.nutch.parse.html.HtmlParser filename.html

The PluginRepository is the name of the plugin itself, and the pluginId is the fully qualified name of the plugin class.

<<< See also: WritingPluginExample

<<< See also: HowToContribute

<<< PluginCentral