WhichTechnicalConceptsAreBehindTheNutchPluginSystem

This text explains a set of concepts involved in the nutch plugin system as a kind of vocabulary definition.

As usual in a vocabulary definition, we differ between "should have", "must have" and "can have".

Extension point

A extension point is a plug that can be extended by third party functionality. Seen as publisher - listener pattern the extension point is a publisher. A extension point must have a definition of the counterpart male connector in form of a java interface. As java doc comments the interface should document what kind of data the two parts will exchange. The interface name should start with a capital **I** e.g *IMyExtensionPoint* . It must be documented if the extension point need 1 or 0...n corresponding enhancements. The extension point implementation must implement a handling of exceptions that can be thrown by the third party counterpart male connectors.

Extension

An extension is the corresponding match of a extension point. It is the third-party functionality enhancement of an extension point. The extension must implement the extension point interface and should return data in the expected format.

Plugin

A plugin is a collection of one or more extension implementations. The plugin bundles all required Java classes and libraries that are necessary to drive the extension points. Furthermore, a plugin must have an XML plugin manifest file as a kind of deployment descriptor that contains relevant metadata. A plugin can have a plugin class that handles the life cycle of a plugin. A set of extension points can be bundled within a plugin as well.

Plugin class

A plugin can have a plugin class specified in the plugin manifest file. A plugin class extends the *org.apache.nutch.plugin.Plugin* class and can override the startUp and shutDown methods. Lifecycle-relevant interactions - like database connections - should be handled in these methods. Until Nutch runtime, only one instance of such a plugin class is alive in the Java virtual machine.

Plugin manifest

Each plugin must have a manifest (plugin.xml) file. The plugin manifest file is an XML file that contains a set of relevant metadata that describe the content of a plugin in machine-readable form. Besides the information on which extension corresponds to which extension points, it contains information about an optional plugin class and jar libraries which are required. Furthermore, it describes dependencies on other plugins.

Plugin Dependency Management with Ivy

Each plugin must also maintain it's own ivy.xml file including all third party libraries required to run the plugin when it is loaded into the plugin system.

Plugin repository

The plugin repository is a kind of registry data base for Nutch plugins until runtime, and the heart of the plugin system. After a plugin dependency validation, extension points and corresponding extensions are registered in the Repository. At runtime, an extension point must query the repository for installed extensions to invoke them. Further more the plugin repository handles the plugin life cycle by invoking the startUp and shutDown methods of a plugin class and manages the class-loaders of the plugins.

<<<PluginCentral