WhatsTheProblemWithPluginsAndClass-loading

This text describes how class-loading works in the Nutch plugin architecture. To understand this text you should be familiar with the Java class-loading mechanism.

What's the problem with class-loading and plugins

One aspect of any plugin system is that as a plugin developer you need to take care what other plugin developers do. Since you are free to use all third party jar libraries you wish to use, it may happen that there may be two plugins that use different versions of the same library. Therefore in some cases it is necessary to hide classes. Publishing classes for sharing issues may be necessary as well, since there could be plugins that provide extension points. As you can see, we are in a dilemma and it must be possible to configure whether a library should be hidden or public to other plugins.

How does the Nutch plugin system solve the class-loading problem

To solve the problem of class-loading in plugins, the Nutch plugin system uses a very easy solution. Each plugin has it own url class-loader that is initiated upon plugin start up.

The jar libraries a plugin uses must be defined in the plugin manifest file ("plugin.xml"). It is possible to define a jar library as public, shared, or hidden to other plugins.

The class-loader of a plugin gets all jar libraries assigned until initialisation that are defined in the manifest file. Beside these 'local' libraries, the dependency chain of a plugin is analysed, and all jar libraries defined as public are assigned to the class-loader as well.

If at runtime when a class tries to load another class, first we try to load the class from the plugin's class-loader. In case loading a class from the plugin's class-loader fails, we forward the class load request to the parent of the plugin class-loader.

The parent class-loader of a plugin class-loader can be a servlet class-loader, a jmx class-loader or maybe the normal Java runtime system class-loader; it depends in which environment you are running Nutch.

The plugin class-loader is a url class-loader so theoretically it is possible to load jar libraries from a central server. This may interesting for example in case you run a server farm and wish to have a centralised library management for maintenance issues.

Pitfalls

The Nutch plugin system can avoid dependency version conflicts between plugins and to some extend between a plugin and Nutch core dependencies. There are some situations where things go wrong, e.g., if an object is passed from core to a plugin but the plugin has a different class instance (and implementation) in its class loader.