MarkupLanguageParserProposal

Markup Language Parser Proposal

Jérôme Charron <jerome.charron@gmail.com>, Chris A. Mattmann <chris.mattmann@jpl.nasa.gov>, François Martelet <fmartelet@yahoo.fr>, Sébastien Le Callonnec <slc_ie@yahoo.ie>

Wednesday, November 17th, 2005

DRAFT

Summary of Issue

Currently, Nutch provides some specific markup language parsing plugins: one for handling HTML, another one for RSS, but no generic XML parsing plugin. This is extremely cumbersome as adding support for a new markup language implies that you have to develop the whole XML parsing code from scratch. This methodology causes: (1) code duplication, with little or no reuse of common pieces of XML parsing code, and (2) dependency library duplication, where many XML parsing plugins may rely on similar xml parsing libraries, such as jaxen, or jdom, or dom4j, etc., but each parsing plugin keeps its own local copy of these libraries. It is also very difficult to identify precisely the type of XML content encountered during a parse. That difficult issue is outside the scope of this proposal, and will be identified in a future proposal.

Suggested Remedy

We propose to add a generic XML parser that allows other markup-related parsers to be easily implemented. The approach would entail simply:

1. Extending the generic XML parser 2. Providing (if needed) a markup pre-processor in charge of producing a valid XML document from the input markup file (e.g., allow transformation of a generic xhtml using an xpath pre-processor that extracts fields and generates an xml document of those fields, prior to index). 3. Providing an XSL file that extracts contents from the valid XML input document. The fields would include title, text, and any other domain specific metadata the user wants to index. The output of the XSL would be a parse-out xml file (the format of which is described below).

The parse-xml plugin

The parse-xml plugin we propose to construct will be standard parse plugin (an extension to the org.apache.nutch.parse.Parser extension-point) that defines a new extension point (org.apache.nutch.parse.MarkupParser) for more specific markup language parsers. The extension point interface would look something like below:

```
public interface MarkupParser {

/** The name of the extension point. */
final static String X_POINT_ID = MarkupParser.class.getName();

/** Performs "tidy" like process on the provided content */
 Content toWellFormedXml(Content c);

 /** Returns the transformer to apply on the specified Content */
 Transformer getTransformer(Content c);
}
```

We believe that the parse-xml plugin must be a fully operational XML parser: i.e. it would not only be a framework for some more specific markup-related parsers, but it would be able to parse any well-formed XML documents (so that it can be used as the default parser for xml documents if no specific parser is found).

The parse-out xml file

The parse-out xml file is the output of the parse-xml plugin that we propose to construct. Its responsibility is to clearly provide an externalized xml structure that can be used as an easy means of accessing the appropriate metadata fields that must be extracted from a particular piece of markup content. We want to use a flexible approach. We will support both a standard set of fields, based on the Dublin Core basic elements, and, at the same time, allow a user to specify domain specific fields to be indexed. A typical parse-out XML file would have the following format:

```
<?xml version="1.0"?>
 <nutch:parse xmlns:nutch="http://nutch.org/1.0/parse" status='success'>
 <nutch:title>Sample title</nutch:title>
 <nutch:creator>Jérôme Charron</nutch:creator>
 <nutch:content-type>text/xml</nutch:creator>
 <nutch:link>http://www.nutch.org/sample.xml</nutch:url>
 <nutch:language>fr</nutch:language>
 <nutch:text>Here is the textual content to index</nutch:text>
 <nutch:outlink url="http://www.nutch.org/outlink1">anchor of the outlink 1</nutch:outlink>
 <nutch:outlink url="http://www.nutch.org/outlink2">anchor of the outlink 2</nutch:outlink>
 ....
 <nutch:meta name="name.meta.1">meta 1 value</nutch:meta>
 <nutch:meta name="name.meta.2">meta 2 value</nutch:meta>
 ....
 </nutch:parse>
```

It is important to note that the parse-out file includes precisely what is needed by Nutch to index a particular piece of content, that is:

1. A status of the parse (SUCCESS, etc.), shown as nutch:parse status="" 2. The content title, shown as nutch:title 3. A set of outlinks, shown as a set of nutch:outlink tags 4. Domain-specific metadata, shown as a set of nutch:meta tags 5. The actual textual content to index, shown as nutch: text

The parse-out also include several pieces of metadata that are also useful, the language of the content, the creator of the content, etc., as stated above, these common metadata fields would be based on the Dublin Core set of elements to describe any electronic resource.

As part of our proposal we will develop a full XML schema for the parse-out file. We will base our final schema on the following initial parse-out XML schema that we have created below:

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
     <xs:element name='parse'>
     <xs:complexType>
          <xs:sequence>
           <xs:element name='title' type='xs:string'/>
           <xs:element name='creator' type='xs:string'/>
         <xs:element name='content-type' type='xs:string'/>   <!-- TODO: Define a more specific type for
content-type -->
           <xs:element name='link' type='xs:string' minOccurs='1'/>   <!-- TODO: Define a more specific type
for link -->
            <xs:element name='language' type='xs:string'/>    <!-- TODO: Define a more specific type for
language -->
           <xs:element name='text' type='xs:string'/>
           <xs:element name='outlink' type='xs:string' minOccurs='0' maxOccurs='unbounded'>
         <xs:complexType>
      <xs:attribute name='url' type='xs:string'/>    <!-- TODO: Define a more specific type for url -->
        </xs:complexType>
      </xs:element>
      <xs:element name='meta' type='xs:string' minOccurs='0' maxOccurs='unbounded'>
        <xs:complexType>
     <xs:attribute name='name' type='xs:string'/>   <!-- TODO: Define a more specific type for a field name -->
    </xs:complexType>
    </xs:sequence>
 </xs:element>
</xs:schema>
```

Impact on current releases of Nutch

- A new parse-xml plugin
- Rewrite the parse-html and parse-rss plugins based on the new parse-xml framework
- Update the parse-plugin.xml file (so that parse-xml is the default XML parser)

Issues

Create performance benchmarks and ensure that the new implementation gives at least the same performances as the parse-html plugin (the most used parse plugin in a whole web crawling)

Timeframe

- Nutch Release 0.8