The Greenstone plugin architecture

Ian H. Witten
Computer Science Dept
University of Waikato
New Zealand
+64 7838-4346
ihw@cs.waikato.ac.nz

David Bainbridge
Computer Science Dept
University of Waikato
New Zealand
+64 7838-4407
davidb@cs.waikato.ac.nz

Gordon Paynter
University of California
Riverside
California, U.S.
+1 909 787-2279
gordon.paynter@ucr.edu

Stefan Boddie
Computer Science Dept
University of Waikato
New Zealand
+64 7838-6038
sjboddie@cs.waikato.ac.nz

ABSTRACT

This note describes how the Greenstone digital library system uses "plugins" to import documents and metadata in different formats, and associate metadata with the appropriate documents. Plugins that import documents can perform their own format conversion internally, or take advantage of existing conversion programs. Metadata can be read from the input documents, or from separate metadata files, or are computed from the documents themselves. New plugins can be written for novel situations.

Categories and Subject Descriptors:
H 3.7 Digital Libraries: systems issues.

General Terms: Algorithms, Design.

Keywords: DL architecture, metadata, Greenstone software.

1. INTRODUCTION

Flexible digital library systems need to be able to accept documents and metadata in many different forms. Documents may be available in web-oriented formats such as HTML and XML, or in word-processor formats such as Microsoft Word or RTF, or expressed in page description languages such as PostScript or PDF. Metadata may also be available in many different forms: embedded in the documents, in metadata files, in spreadsheets, or even encoded into file naming conventions; or it may be extracted from the documents themselves. Digital library designers must either insist that users adopt a prescribed scheme for document and metadata specification, or implement flexible, extensible, ways of allowing different formats to be accommodated.

The Greenstone digital library software uses an extensible architecture [1]. Input documents and metadata specifications are imported using software modules called "plugins." These convert documents and metadata into a uniform XML representation (which includes the possibility of referencing external resources) that is then used throughout the system. There are plugins for standard document formats and standard metadata formats. Plugins are even used to traverse the directory hierarchy in which source documents are stored—this admits the possibility of cross-referencing metadata with source documents based on the filename path. New plugins can be written to accommodate new document and metadata types.

2. THE PLUGIN PIPELINE

Greenstone collections typically involve documents and metadata in several different formats, and for each format there must be a plugin that can process it. Plugins fit together into a cascaded pipeline that provides a highly configurable workflow system. The order of plugins in the pipeline is determined by the order in which they are listed in the collection's configuration file. Conceptually, there are three types of plugin: *structural*, *markup*, and *extraction*; we give examples below.

The import process is initiated by feeding the name of the top-level directory that contains the source documents into the pipeline. This name is passed from plugin to plugin until one of them signals that it can process the file. Under normal operating conditions, any filename that names a directory is processed by a "structural" plugin called RecPlug (for "recursive plugin") that lists all files in the named directory and feeds their names, one by one, into the pipeline. In general this list includes further subdirectories, which will be processed in the same way. RecPlug is the normal way of traversing the directory structure associated with a collection. However, it can easily be adapted for particular situations, such as when certain files or directories need special treatment, or the names of directories are significant for metadata assignment purposes.

Microsoft Word documents, to take just one example of a document type, are processed into Greenstone's standard XML form by a "markup" plugin called WORDPlug. The text and metadata that WORDPlug identifies are passed into a secondary pipeline of "extraction" plugins, again determined by the collection's configuration file. These plugins extract "metadata" (we use the term in an extended sense) from the file, such as acronyms, dates and e-mail addresses.

If an item reaches the end of the pipeline without being processed by any plugin, the system generates a warning message and moves on to the next item. The process stops when the queue is empty.

3. PLUGINS

Plugins are written in a standard object-oriented fashion and utilize an inheritance structure to minimize code duplication. They all derive from a basic object called *BasPlug*, which performs universally-required operations like creating a new Greenstone document to work with, assigning an object identifier, and handling the sections in a document. The implementation does not enforce a rigid distinction between the three types of plugins because in practice there is considerable overlap in functionality. This allows, for example, structural plugins to be enhanced with markup capabilities and *vice versa*—a useful ability when metadata and documents are represented separately in the file system. As another example, it simplifies the triggering of the cascading pipeline, when a particular markup plugin needs to pass a document on to an extraction plugin for further processing. Finally, it increases efficiency of operation because there is no need to physically pass large volumes of textual data from one extraction plugin to the next.

The inheritance hierarchy distinguishes two special categories of markup plugin: *conversion* plugins and *splitter* plugins, and appropriate high-level functionality is provided for each type. Conversion plugins are those that use existing external programs to convert a particular format to either HTML or text form and add them back into the pipeline. Splitter plugins process files (such as e-mail formats) that contain several documents per file, and provides functions for splitting them into separate documents for the processing pipeline. To be useful, BasPlug, ConvertToPlug and SplitPlug must each be further subclassed into fully-functioning plugins that are used in the actual pipeline.

Plugins are written in the Perl language, and new ones can be added that process document formats not handled by existing ones, format documents in some special way, assign metadata specified in new formats, or extract new kinds of metadata from the document text. Given the established body of plugins available, the best way to write a new plugin is usually to find an existing one that does something similar and modify it. The object-oriented structure encourages subclassing particular plugins to add new features.

4. DOCUMENT PROCESSING PLUGINS

Table 1 lists the plugins used for widely-used document formats. TEXTPlug interprets a plain text file as a simple document. HTMLPlug processes HTML files. It extracts *Title* metadata based on the HTML <*title*> tag; other metadata expressed using HTML's metatag syntax can be extracted too. It also parses and processes any links that the file contains.

TEXTPlug	Plain text.
HTMLPlug	HTML, replacing hyperlinks appropriately.
WordPlug	Microsoft Word documents.
PDFPlug	PDF documents.
PSPlug	PostScript documents.
EMAILPlug	E-mail messages, recognizing author, subject, date, etc.
BibTexPlug	Bibliography files in *BibTex* format.
ReferPlug	Bibliography files in *refer* format.
SRCPlug	Source code files.
ImagePlug	Image files for creating a library of images.
SplitPlug	Splits a document file into parts.
ZIPPlug	Uncompresses files.
BookPlug	Specially marked-up HTML.
GBPlug	Project Gutenberg E-text.
TCCPlug	E-mail documents from Computists' Weekly.
PrePlug	HTML output from the PRESCRIPT program.

Table 1 Document processing plugins

Links to other files in the collection are trapped and replaced by references to the corresponding documents within the digital library. Several options are available with this plugin, including a switch for when source documents are derived from a set of mirrored web sites.

WORDPlug imports Microsoft Word documents. There are many variants on this format—even Microsoft programs sometimes have conversion failures. PDFPlug imports documents in PDF, Adobe's "Portable Document Format." Both these plugins use independent programs to convert source files to HTML.

PSPlug imports documents in PostScript. It works best if a standard conversion program is already installed on your computer, but if not the system resorts to a simple text extraction algorithm. EMAILPlug imports files containing e-mail, and deals with common formats such as are used by the Netscape, Eudora, and Unix mail readers. Each source document is examined to see if it contains an e-mail, or several joined together in one file, and if so the contents are processed. The plugin extracts *Subject*, *To*, *From*, and *Date* metadata.

ZIPPlug handles compressed and/or archived input formats. It relies on standard utility programs being present. The final four entries in Table 1 are one-off plugins for particular collections.

5. PLUGIN OPTIONS

The collection configuration file can supply options to a plugin that modify its behavior. For example:

```
plugin  WordPlug -input_encoding iso_8859_1
```

fixes the character encoding for Word documents to be ISO 8859-1 rather than relying on the default. Aside from ASCII and Unicode, the *input_encoding* option has some 30 other possible values, including about 15 for languages such as Chinese, Cyrillic, Greek, Hebrew; 5 ISO standards; and 10 different Windows standards. The default value, *auto*, automatically determines the character encoding for each document individually using an automatic language identification module.

This option is implemented by BasPlug, and therefore by all plugins that inherit from it. There are several others. One specifies which files a plugin can process, in terms of a set of filename patterns. Each plugin has a default value; HTMLPlug's is an expression that includes filenames with the extension *.htm* or *.html* regardless of case. Another option specifies files that are to be blocked, and not passed further down the list of plugins. HTMLPlug blocks files with such extensions as *.gif*, *.png* and *.jpg* because they do not contain any text or metadata but are embedded in documents when they are viewed.

Plugins can define their own options as well as inheriting base-level ones. Thus they can augment the set of possible options with ones that are meaningful in particular contexts.

6. REFERENCES

[1] Witten, I.H., McNab, R.J., Boddie, S.J. and Bainbridge, D. (2000) "Greenstone: A comprehensive open-source digital library software system." Proc Digital Libraries, 113–121