Biblio::CiteParser 1.10 Documentation

Mike Jewell

September 1, 2004

Contents

Chapter 1

Introduction

1.1 What is ParaTools?

ParaTools, short for ParaCite Toolkit, is a collection of Perl modules for reference parsing that is designed to be easily expanded and yet simple to use. The parsing modules make up the core of the package, but there are also useful modules to assist with OpenURL creation and the extraction of references from documents. The toolkit is released under the GNU Public License, so can be used freely as long as the source code is provided (see the COPYING file in the root directory of the distribution for more information).

The toolkit came about as a result of the ParaCite resource, a reference search engine located at http://paracite.eprints.org, which uses a template-based reference parser to extract metadata from provided references and then provides search results based on this metadata. The ParaCite parser is provided directly as the Biblio::CiteParser::Standard module, with a separate Templates module that can be replaced as new reference templates are located.

As well as providing examples for the provided parsing modules, ParaTools also includes examples for using the ParaCite web service. This is an alternate interface which provides access to ParaCite's search and parsing functionality for any language that supports the Web Services Description Language (WSDL).

1.2 Who should use ParaTools?

The ParaTools package has many applications, including:

- Converting reference lists into valid OpenURLs
- Converting existing metadata into valid OpenURLs
- Collecting metadata from references to carry out internal searches
- Extracting reference lists from documents

- Carrying out searches using ParaCite

The modularity of ParaTools means that it is very easy to add new techniques (and we would be very pleased to hear of new ones!).

1.3 What will it run on?

ParaTools should work on any platform that supports Perl 5.6.0 or higher, although testing was primarily carried out using Red Hat Linux 7.3 with Perl 5.6. Where possible platform-agnostic modules have been used for file functionality, so temporary files should be placed in the correct place for the operating system. Memory requirements for ParaTools are minimal, although the template parser and document parser will require more memory as the number of templates and sizes of documents increase.

1.4 This Documentation

This documentation is written in perl POD format and converted into Postscript (which is 2 pages to a sheet for printing), ASCII, PDF, and HTML.

The latest version of this documentation can be obtained from http://paracite.eprints.org/files/docs/

Chapter 2

Required Software

2.1 What software does Biblio::CiteParser need?

Perl Modules

URI

URI is required for the OpenURL encoding functions in Biblio::CiteParser::Utils.

Text::Unidecode

Used by Biblio::CiteParser::Citebase to allow for matching on unicode strings.

URI::OpenURL (Optional)

If you wish to create valid OpenURLs, URI::OpenURL provides a set of functions for this purpose. The metadata produced by Biblio::CiteParser can be used with this module.

SOAP::Lite (Optional)

This module is required if you wish to use the ParaCite web services, but optional otherwise. This requires several other modules, which are available in the soap subdirectory of http://paracite.eprints.org/files/perlmods/.

There are also some dependencies for the above modules, including MIME::Base64, HTML::TagSet, and Digest::MD5. The latest versions of these can be obtained from http://www.cpan.org/

Installing Perl Modules

This describes the way to install a simple perl module, some require a bit more effort. We will use the non-existent FOO module as an example.

Unpack the archive:

```
% tar xfvz FOO-5.2.34.tar.gz
```

Enter the directory this creates:

```
% cd FOO-5.2.34
```

Run the following commands:

```
% perl ./Build.PL
% ./Build
% ./Build test
% ./Build install
```

Chapter 3

How to Install Biblio::CiteParser

3.1 Installation

First unpack the Biblio::CiteParser archive:

```
% tar xfvz <packagename>.tar.gz
```

Move into the unpacked folder, and then do the following:

```
% perl Build.PL
% ./Build
```

You can optionally run

```
% ./Build test
```

which will carry out a few checks to ensure everything is working correctly. Finally, become root and do:

```
% ./Build install
```

This will install the modules and man pages into the correct locations.

3.2 Examples

The examples directory contains two categories of examples - parsing examples and web service examples. Note that the web service examples require the SOAP::Lite module (see Required Software for more information). To try out these samples after installation, simply cd into the directory and execute the example. More information about the examples is in the README file inside the examples directory.

Chapter 4

Reference Parsing

4.1 Parsing References

Biblio::CiteParser is designed for parsing citations, and this can be done very simply:

```
use Biblio::CiteParser::Standard;
my $parser = new Biblio::CiteParser::Standard();
my $metadata = $parser->parse("Jewell, M (2002) Parsing Examples");
```

The `$metadata` variable is a hash containing the information extracted from the reference.

If you'd prefer to use another parser, simply substitute the 'Standard' for the appropriate module. Biblio::CiteParser is distributed with the Jiao module, which is a slightly modified version of a module created by Zhuoan Jiao. To use this instead of the Standard module, you would do the following:

```
use Biblio::CiteParser::Jiao;
my $parser = new Biblio::CiteParser::Jiao();
my $metadata = $parser->parse("Jewell, M (2002) Parsing Examples");
```

The Standard module provides slightly richer metadata than the Jiao module, but it does rely on templates (see Biblio::CiteParser::Templates) so requires updating as new citation formats are found.

4.2 Creating an OpenURL

Once you have the metadata from the reference, it is easy to create an OpenURL from it:

```
use Biblio::CiteParser::Standard;
use Biblio::CiteParser::Utils;
```

```
my $parser = new Biblio::CiteParser::Standard();
my $metadata = $parser->parse("Jewell, M (2002) Parsing Examples");
my $openurl = create_openurl($metadata);
```

The OpenURLs created by Biblio::CiteParser do not have a Base URL prefixed, so this should be carried out before they are used (the ParaCite base URL is http://paracite.eprints.org/cgi-bin/openurl.cgi).

If you would like to try to extract more information from the metadata, you can use the `decompose_openurl` function:

```
my ($enriched_metadata, @errors) = decompose_openurl($metadata);
```

This tries to extract information from SICIs, page ranges, etc, and also checks the fields for validity (the `@errors` array contains any mistakes).

Note that the create_openurl has been superceded by URI::OpenURL, but the metadata returned by `trim_openurl` is in the correct format to be passed to this module.

4.3 Metadata Structure

Biblio::CiteParser supports all of the fields specified in Table 1 of the OpenURL specification (http://www.sfxit.com/openurl/openurl.html). Specific parsers can add their own fields, but these are not exported when OpenURLs are created. Biblio::CiteParser::Standard provides the following extra fields:

marked

A marked-up version of the reference. e.g. <author>Jewell, M</author> (<year>2002</year>) <title>A title</title>.

match

The template matched by Biblio::CiteParser::Standard

ref

The original reference

Chapter 5

ParaCite Web Service

5.1 The ParaCite Web Service

The Biblio::CiteParser package includes several examples that demonstrate the ParaCite web service, as well as the WSDL definition file. This section explains the web service, and gives an introduction to using it.

As ParaCite is written entirely in Perl, there are obvious issues if you wish to use Java, PHP, or another language. The ParaCite web services provides an interface into the reference parsing features of ParaCite, while remaining language agnostic.

5.2 Using the Web Service from Perl

To access the web service from Perl requires the SOAP::Lite module (see Required Software). Once this is present, this is all that is required to connect to the web service:

```
my $service = SOAP::Lite
       -> service("http://paracite.eprints.org/paracite.wsdl");
```

Three functions are now available from the `$service` variable:

doOpenURLConstruct($reference, $baseurl)

This returns an OpenURL, prefixed by the base URL if one is provided.

doReferenceParse($reference, $baseurl)

This returns a hash containing the metadata in the reference, and an OpenURL formed using the metadata and the base URL.

doParaciteSearch($reference, $baseurl)

This returns an hash containing 'resultElements' (an array of search results), and 'metadata' (a hash of metadata).

5.3 Web Service Examples

The following code parses a reference, and stores the metadata in `$metadata` and the OpenURL in `$openurl`:

```
use SOAP::Lite;
my $service = SOAP::Lite
       -> service("http://paracite.eprints.org/paracite.wsdl");
my $base_url = "http://paracite.eprints.org/cgi-bin/openurl.cgi?";
my $result = $service
       -> doReferenceParse("Jewell, M (2002) Example", $base_url);
my $metadata = $result->{metadata};
my $openurl = $result->{openURL};
```

If you do not want the metadata, and just want a link to an OpenURL resolver, the following will do that:

```
use SOAP::Lite;
my $service = SOAP::Lite
       -> service("http://paracite.eprints.org/paracite.wsdl");
my $base_url = "http://paracite.eprints.org/cgi-bin/openurl.cgi?";
my $open_url = $service
       -> doOpenURLConstruct("Jewell, M (2002) Example", $base_url);
```

Finally, this example uses the doParaciteSearch method to get the first match on a reference:

```
use SOAP::Lite;
my $service = SOAP::Lite
       -> service("http://paracite.eprints.org/paracite.wsdl");
my $base_url = "http://paracite.eprints.org/cgi-bin/openurl.cgi?";
my $query = "Harnad, Stevan (1995) The PostGutenberg Galaxy.";
my $result = $service
       -> doParaciteSearch($query, $base_url);
my $first_result = $result->{resultElements}->[0];
print "First result is: ".$first_result->{URL}."\n";
```

The web service automatically adds Google, Scirus, and Vivissimo as resources to the search request, so if no resources match the publication or subject these will be used as fall-backs.

5.4 Web Service Structures

Most of the Paracite structures have been modelled very closely on the Google web service structures to allow some degree of standardisation. Some additions have been made, and some fields are not yet used, but these may change in future versions.

ParaciteSearchResult

resultElements

This is an array of resources, along with the search URLs associated with them. See the ResultElement description later in this section.

estimatedTotalResultsCount

This returns the number of items in the resultElements array.

estimateIsExact

This currently always returns 1.

searchQuery

This contains the original reference.

openURL

This contains the OpenURL represented by the reference metadata (prefixed by base URL if one is supplied).

metadata

This is a Metadata object (see later in this section).

ResultElement

URL

This is a URL that searches the current resource for the reference.

template

This contains the template of the matching resource interface that was used to generate the search URL.

name

The name of the resource (e.g. Google).

description

Some more information about the resource.

tollfree

A boolean value that is true if the results can be viewed without cost.

fulltext

A boolean value that is true if a resulting article from this resource will have the full text available.

stratum

An integer representing the stratum in which this resource lies. A complete list of the strata is available at http://paracite.eprints.org/cgi-bin/views/viewstrata.cgi

Metadata

All of the fields in Metadata are valid fields in OpenURL metadata. See Table 1 at http://www.sfxit.com/openurl/openurl.html for a complete list.

Chapter 6

Troubleshooting

6.1 Troubleshooting

If you cannot find a solution to your problem here, make sure you are using the latest version of the toolkit and ask on the ParaTools mailing list (see http://paracite.eprints.org/developers/).

6.2 Reference Parsing

Reference does not parse correctly

If you are using the Standard parsing module, make sure that a template for the reference exists in the package. See the HOWTO for more information on how to do this. If you are using a contributed module, please email your query to the author of the module.

Chapter 7

How-To Guides

7.1 HOW TO: Modify Templates in Biblio::CiteParser::Standard

Adding new templates to the Standard parser is relatively easy:

- Locate where your Templates.pm file has been installed.

  On Linux systems this should just involve doing 'locate Templates.pm', otherwise 'find / -name Templates.pm' should work. Alternatively, you can edit the Templates.pm in the Biblio/CiteParser/ directory of an unpacked distribution, and install it once you have finished.

- Add the template to the list.

  If you are editing an already installed Templates.pm file you will probably have to be root to do this. If you are editing the Templates.pm inside an unpacked distribution, you will have to reinstall the modules once you are finished (see the Installation section).

The Templates.pm file should contain a structure similar to this:

```
$Biblio::CiteParser::Templates::templates = [
       '_AUTHORS_, _PUBLICATION_, _YEAR_, _ISSUE_, _SPAGE_-_EPAGE_',
...
       ];
```

Each template is a string containing a set of placeholders. For example, '_AUTHORS_ (_YEAR_) _TITLE_' can match 'Jewell, M (2002) Title'. The following are valid field names:

ANY

Matches anything.

AUFIRST

Matches the first name of an author.

AULAST

Matches the last name of an author.

AUTHORS

Matches a list of authors.

CAPPUBLICATION

Matches a capitalised publication title (e.g. "Journal of Lemurs").

CAPTITLE

Matches a capitalised title.

CHAPTER

Matches a chapter number.

DATE

Matches a date in nn/nn/nn form.

EDITOR

Matches an editor's name.

EPAGE

Matches the last page in a page range.

ISBN

Matches an ISBN number.

ISSN

Matches an ISSN number.

ISSUE

Matches an issue number.

PAGES

Matches a page range in nn-nn form.

PUBLICATION

Matches a publication name.

PUBLISHER

Matches a publisher name.

PUBLOC

Matches the location of a publisher.

SPAGE

Matches the start page.

SUBTITLE

Matches a subtitle.

TITLE

Matches an article title.

UCPUBLICATION

Matches a publication in entirely upper-case (e.g. JOURNAL OF LEMURS).

UCTITLE

Matches a title in entirely upper-case.

URL

Matches a URL.

VOLUME

Matches a volume.

YEAR

Matches a year (4 digits).

7.2 HOW TO: Integrate ParaTools with EPrints 2

EPrints already contains ParaCite support, but using a specially built version of the module before it was part of ParaTools. To alter your cgi/paracite script to use ParaTools, you need to do the following:

First replace

```
use Citation::Parser::Simple;
```

with

```
use Biblio::CiteParser::Standard;
```

Next, replace this line:

```
my $parser = new Citation::Parser::Simple();
```

with this line:

```
my $parser = new Biblio::CiteParser::Standard();
```

This should work fine, although you can obviously integrate ParaCite more if you wish.

7.3 HOW TO: Create a New Parser

Creating a Citation Parser

All new citation parsers should be named Biblio::CiteParser::SomeName, where SomeName is replaced with a unique name (ideally the author's surname). The parser should extend the Biblio::CiteParser module like so:

```
package Biblio::CiteParser::SomeName;
require Exporter;
@ISA = ("Exporter", "Biblio::CiteParser");
our @EXPORT_OK = ( 'new', 'parse' );
```

You should then override the 'new' and 'parse' methods:
e.g.

```
sub new
{
        my($class) = @_;
        my $self = {};
        return bless($self, $class);
}

sub parse
{
        my($self, $ref) = @_;
        my $hashout = $self->extract_metadata($ref);
        return $hashout;
}
```

This makes it easy for users to swap out one reference parser for another.

Chapter 8

Problems, Questions and Feedback

8.1 Bug Report Policy

There is currently no online bug tracking system. Known bugs are listed in the BUGLIST file in the distribution and a list will be kept on the http://paracite.eprints.org/developers/ site.

If you identify a bug or "issue" (issues are not bugs, but are things which could be clearer or better), and it's not already listed on the site, please let us know at paracite@ecs.soton.ac.uk - include all the information you can: what version of Biblio::CiteParser (see VERSION if you're not sure), what operating system etc.

8.2 Where to go with Questions and Suggestions

There is a mailing list for ParaTools (encompassing Biblio::CiteParser) which may be the right place to ask general questions and start discussions on broad design issues.

To subscribe send an email to majordomo@ecs.soton.ac.uk containing the text

```
subscribe paratools
```