Importing MARC data into DSpace

Author: *Steve Thomas, Senior Systems Analyst,*
University of Adelaide Library

Last Update: *7/08/2006 9:36 am*

Abstract: *Describes a methodology and Perl scripts used to import data into DSpace derived from a file of MARC records.*

Problem

We have two collections of documents stored locally on web servers, with a record describing each document in our Catalogue. Each catalogue record includes a URL linking to the document file. We wanted to migrate the documents into our new DSpace repository, and so needed to convert the Catalogue records into Dublin Core to provide the metadata for each document.

Importing items into DSpace requires that they be organised into a specific structure: import is from a directory containing one sub-directory for each item, with each subdirectory containing the document file(s), a contents file listing the document files, and a dublin_core.xml file containing the metadata. This is all detailed in the DSpace system documentation, which provides the following succinct diagram:

```
archive_directory/
    item_000/
        dublin_core.xml -- qualified Dublin Core metadata
        contents        -- text file containing one line per filename
        file_1.doc      -- files to be added as bitstreams to the item
        file_2.pdf
    item_001/
        dublin_core.xml
        contents
        file_1.png
        ...
```

So the task was to generate the required directory structure and the dublin_core.xml file from the MARC catalogue record.

Conversion of MARC to Dublin Core

The first task was to convert our catalogue MARC records into Dublin Core, in the format required by DSpace.

DSpace uses a reasonably simple version of qualified Dublin Core , thus:

```
<dublin_core>
    <dcvalue element="title" qualifier="none">A Tale of Two Cities</dcvalue>
    <dcvalue element="date" qualifier="issued">1990</dcvalue>
    <dcvalue element="title" qualifier="alternate" language="fr" ">J'aime
les Printemps</dcvalue>
</dublin_core>
```

To convert to this format, I developed a simple Perl script. While I could have written code to pull apart the MARC record, there are well-developed modules available from CPAN to do this, so I used two of these: MARC::File::USMARC which provides basic operations to convert MARC into an internal data structure, and MARC::Crosswalk::DublinCore which uses that structure to create a qualified Dublin Core data structure.

This gave me a basic MARC->DC conversion script, which looks like this:

```
#!/usr/local/bin/perl -w

use MARC::Crosswalk::DublinCore;
use MARC::File::USMARC;

$/ = chr(29); # MARC record separator

print qq|<collection>\n|;

while (my $blob = <>) { # suck in one MARC record at a time

        print qq|<dublin_core>\n|;

        # convert the MARC to DC
        my $marc = MARC::Record->new_from_usmarc( $blob );
        my $crosswalk = MARC::Crosswalk::DublinCore->new( qualified => 1 );
        my $dc        = $crosswalk->as_dublincore( $marc );

        # output the DC as XML
        for( $dc->elements ) {

                my $element     = $_->name;
                my $qualifier   = $_->qualifier;
                my $scheme      = $_->scheme;
                my $content     = $_->content;

                printf qq|  <dcvalue element="%s"|, $element;
                printf qq| qualifier="%s"|, $qualifier if $qualifier;
                printf qq| scheme="%s"|, $scheme if $scheme;
                printf qq| language="en">%s</dcvalue>\n|, $content;
        }

        print qq|</dublin_core>\n|;
}

print qq|</collection>\n|;

exit;
```

So far so good. This script will take a file of MARC records and output an XML file of Dublin Core records, which look like this:

```
<dublin_core>
  <dcvalue element="Title" language="en">Potassium fertiliser use in South
Australia [electronic resource] / P.M. Barrow</dcvalue>
  <dcvalue element="Creator" language="en">Barrow, P. M.</dcvalue>
  <dcvalue element="Creator" language="en">South Australia. Agronomy
Branch</dcvalue>
  <dcvalue element="Subject" scheme="LCSH" language="en">Potassium
fertilizers South Australia.</dcvalue>
  <dcvalue element="Description" language="en">"Paper presented at a
Symposium on 'Potassium in South-Eastern Australia', held at Monash
University, August 21-22, 1967."</dcvalue>
  <dcvalue element="Description" language="en">Electronic reproduction.
Adelaide, S. Aust. : University of Adelaide, Barr Smith Library, 2006. Title
from t.p. on PDF file (viewed 23 May 2006). Electronic text, download as a
PDF file. Available via the World Wide Web.</dcvalue>
  <dcvalue element="Description" language="en">System requirements: Adobe
Acrobat Reader required to view/print PDF files.</dcvalue>
  <dcvalue element="Publisher" language="en">[Adelaide, S. Aust.] : Dept. of
Agriculture, South Australia,</dcvalue>
  <dcvalue element="Date" qualifier="Created" language="en">1968</dcvalue>
  <dcvalue element="Date" qualifier="Created" language="en">2006.</dcvalue>
  <dcvalue element="Date" qualifier="Issued" language="en">1968</dcvalue>
  <dcvalue element="Date" qualifier="Issued" language="en">1968</dcvalue>
  <dcvalue element="Type" scheme="DCMI Type Vocabulary"
language="en">Text</dcvalue>
  <dcvalue element="Format" qualifier="Extent" language="en">11 leaves
;</dcvalue>
  <dcvalue element="Language" scheme="ISO 639-2" language="en">eng</dcvalue>
  <dcvalue element="Relation" qualifier="Requires" language="en">System
requirements: Adobe Acrobat Reader required to view/print PDF
files.</dcvalue>
  <dcvalue element="Relation" qualifier="isPartOf" language="en">Agronomy
Branch report ; no. 1</dcvalue>
  <dcvalue element="Relation" qualifier="isFormatOf" language="en">Also
available in a print form.</dcvalue>
  <dcvalue element="Relation" qualifier="hasFormat" language="en">Also
available in a print form.</dcvalue>
  <dcvalue element="Coverage" qualifier="Spacial" language="en">South
Australia.</dcvalue>
</dublin_core>
```

Now, for the purpose of import into DSpace, there are a number of problems with this, due to the requirements of DSpace:

- the element and qualifier must be lower-case;
- the content may contain reserved characters (&<>) that need escaping;
- the author must use element=”contributor” and qualifier=”author”;
- the qualifier “isPartOf” needs to be “ispartofseries”;
- some elements provide a scheme rather than a qualifier; DSpace does not recognise the scheme attribute;
- the type element is technically correct but uninformative, and should be replaced with a DSpace type;
- the format element is used internally by DSpace to describe files, so should probably be avoided;
- date elements have been repeated, because MARC::Crosswalk::DublinCore is helpfully extracting data from as many fields and subfields as possible;

- some MARC data is irrelevant in the DSpace context, because it relates to print editions, or reflects excessive zeal on the part of the cataloguer.

Many of these problems can be solved with a few additions to the script:

```
        for( $dc->elements ) {

                # lowercase all attributes
                my $element     = lc $_->name;
                my $qualifier   = lc $_->qualifier;
                my $scheme      = lc $_->scheme;
                my $content     = $_->content;

                # escape reserved characters
                $content =~ s/&/&amp;/gs;
                $content =~ s/</&lt;/gs;
                $content =~ s/>/&gt;/gs;

                # munge attributes for DSpace compatibility
                if ($element eq 'creator') {
                        $element = 'contributor';
                        $qualifier = 'author';
                }
                if ($element eq 'format') {
                        $element = 'description';
                        $qualifier = '';
                }
                if ($element eq 'language') {
                    if ($scheme eq 'iso 639-2') {
                        $qualifier = 'iso';
                        $scheme = '';
                    } else {
                        $element = 'description';
                        $qualifier = '';
                    }
                }
                if ($qualifier eq 'ispartof') {
                        $qualifier = 'ispartofseries';
                }

                printf qq|  <dcvalue element="%s"|, $element;
                printf qq| qualifier="%s"|, $qualifier if $qualifier;
                # output scheme as qualifier
                printf qq| qualifier="%s"|, $scheme if $scheme;
                printf qq| language="en">%s</dcvalue>\n|, $content;
        }
```

It is also possible to prove a filter function to MARC::Record->new that will filter out unwanted MARC tags. For example, one might ignore the 008 in order not to duplicate date elements.

Other issues are perhaps better dealt with by post-conversion editing of the xml file.

The script is then run as follows:

```
> ./marc2dc.pl marc.bib > collection.xml
```

Building the import directory structure

Having converted our metadata into Dublin Core, I then needed to use this to build the required directory structure. This is accomplished with a second Perl script, build.pl, which needs to be customised for each collection to be imported. The basic idea is:

1. extract each Dublin Core record from the XML file;
2. create a sub-directory for the record;
3. extract the document file name from the identifier;
4. create the dublin_core.xml file;
5. create the contents file;
6. create a symbolic link to the document file.

Note that this assumes that the files in the identifier field actually reside on the same machine as DSpace, either because the we server is also running there, or because they've been copied from the web server. The key part of the script is the regular expression used to extract the file and path information; this will be different for every import, and will require customisation for each import.

The script is as follows:

```
#!/usr/local/bin/perl -w

$/ = "</dublin_core>\n"; # record separator

$what = 100001; # dummy id for when there's no file

while (<>) {

    # discard the top and bottom tags
    s/<collection>\n//;
    s/<\/collection>\n//;

    # extract the file path from the identifier
    # use the file name as an id
    # note that identifier element is discarded!
    if (s!<dcvalue element="identifier" qualifier="uri"
language="en">http://.*/theses/(.*?)/([^/]+).pdf<\/dcvalue>\n!!s) {
        $path = $1;
        $id = $2;
    } else {
        $path = '';
        $id = $what++;
    }

    # let the operator know where we're up to
    print "$path/$id\n";

    # create the item directory
    mkdir "import/$id", 0755;

    # create the dublin_core.xml file
    open DC, ">import/$id/dublin_core.xml"
      or die "Cannot open dublin core for $id, $!\n";
    print DC $_;
    close DC;

    # assuming we have a file ...
    if ($path) {

        # ... create the contents file ...
        open OUT, ">import/$id/contents"
          or die "Cannot open contents for $id, $!\n";
        print OUT "$id.pdf";
        close OUT;

        # ... and create a symbolic link to the actual file
        symlink "/scratch/dspace/import/theses/$path/$id.pdf",
          "import/$id/$id.pdf";

    }

}

__END__
```

The script is then run against the xml file produced earlier:

```
> mkdir import
> ./build.pl collection.xml
```

After running the script, we should have an import directory structure we can use to import into a DSpace collection in the usual way.

References

DSpace System Documentation
http://www.dspace.org/technology/system-docs/

MARC::File::USMARC
http://search.cpan.org/~petdance/MARC-Record-1.38/lib/MARC/File/USMARC.pm

MARC::Crosswalk::DublinCore
http://search.cpan.org/~bricas/MARC-Crosswalk-DublinCore-0.02/lib/MARC/Crosswalk/DublinCore.pm