POD Translation
by *pod2pdf*

ajf@afco.demon.co.uk

# *MARC::XML.pm*

# Table of Contents MARC::XML.pm

## NAME

MARC::XML - A subclass of MARC.pm to provide XML support.

## SYNOPSIS

```
use MARC::XML;

#read in some MARC and output some XML
$myobject = MARC::XML->new("marc.mrc","usmarc");
$myobject->output({file=>">marc.xml",format=>"xml"});

#read in some XML and output some MARC
$myobject = MARC::XML->new("marc.xml","xml");
$myobject->output({file=>">marc.mrc","usmarc");
```

## DESCRIPTION

MARC::XML is a subclass of MARC.pm which provides methods for round-trip conversions between MARC and XML. MARC::XML requires that you have the CPAN modules MARC.pm and XML::Parser installed in your Perl library. As a subclass of MARC.pm a MARC::XML object will by default have the full functionality of a MARC.pm object. See the MARC.pm documentation for details.

The XML file that is read and generated by MARC::XML is not associated with a Document Type Definition (DTD). This means that your files need to be well-formed, but they will not be validated. When performing XML-MARC conversion it is important that the XML file is structured in a particular way. Fortunately, this is the same format that is generated by the MARC-XML conversion, so you should be able to be able to move your data easily between the two formats.

### Downloading and Intalling

Download

First make sure that you have **MARC.pm** and **XML::Parser** installed. Both Perl extensions are available from the CPAN http://www.cpan.org/modules/by-module, and they must be available in your Perl library for MARC::XML to work properly.

MARC::XML is provided in standard CPAN distribution format. Download the latest version from http://www.cpan.org/modules/by-module/MARC/XML. It will extract into a directory

MARC-XML-version with any necessary subdirectories. Once you have extracted the archive Change into the MARC-XML top directory and execute the following command depending on your platform.

Unix

```
perl Makefile.PL
make
make test
make install
```

Win9x/WinNT/Win2000

```
perl Makefile.PL
perl test.pl
perl install.pl
```

Test

Once you have installed, you can check if Perl can find it. Change to some other directory and execute from the command line:

```
perl -e "use MARC::XML"
```

If you **do not** get any response that means everything is OK! If you get an error like *Can‘t locate method "use" via package MARC::XML*. then Perl is not able to find MARC::XML—double check that the file copied it into the right place during the install.

## Todo

- Checking for field and record lengths to make sure that data read in from an XML file does not exceed the limited space available in a MARC record.
- Support for MARC <-> Unicode character conversions.
- MARC <-> EAD (Encoded Archival Description) conversion?
- Support for MARC <-> DC/RDF (Dublin Core Metadata encoded in the Resource Description Framework)?
- Support for MARC <-> FGDC Metadata (Federal Geographic Data Committee) conversion?

## Web Interface

A web interface to MARC.pm and MARC::XML is available at http://libstaff.lib.odu.edu/cgi-bin/marc.cgi where you can upload records and observe the results. If you‘d like to check out the cgi script take a look at http://libstaff.lib.odu.edu/depts/systems/iii/scripts/MARCpm/marc-cgi.txt However, to get the full functionality you will want to install MARC.pm and MARC::XML on your server or PC.

## Sample XML file

Below is an example of the flavor of XML that MARC::XML will generate and read. There are only four elements: the <*marc*> pair that serves as the root for the file; the <*record*> pair that encloses each record; the <*field*> pair which encloses each field; and the <*subfield*> pair which encloses each subfield. In addition the <*field*> and <*subfield*> tags have three possible attributes: *type* which defines the specific tag or subfield ; as well as *i1* and *i2* which allow you to define the indicators for a specific tag.

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>

<marc>

<record>
<field type="000">00901cam  2200241Ia 45e0</field>
<field type="001">ocm01047729 </field>
<field type="003">OCoLC</field>
<field type="005">19990808143752.0</field>
<field type="008">741021s1884    enkaf         000 1 eng d</field>
<field type="040" i1=" " i2=" ">
   <subfield type="a">KSU</subfield>
   <subfield type="c">KSU</subfield>
   <subfield type="d">GZM</subfield>
</field>
<field type="090" i1=" " i2=" ">
   <subfield type="a">PS1305</subfield>
   <subfield type="b">.A1 1884</subfield>
</field>
<field type="049" i1=" " i2=" ">
   <subfield type="a">VODN</subfield>
</field>
<field type="100" i1="1" i2=" ">
   <subfield type="a">Twain, Mark,</subfield>
   <subfield type="d">1835-1910.</subfield>
</field>
<field type="245" i1="1" i2="4">
   <subfield type="a">The adventures of Huckleberry Finn :</subfield>
   <subfield type="b">(Tom Sawyer’s comrade) : scene, the Mississippi Valley :
   <subfield type="c">by Mark Twain (Samuel Clemens) ; with 174 illustrations.<
</field>
<field type="260" i1=" " i2=" ">
   <subfield type="a">London :</subfield>
   <subfield type="b">Chatto &amp; Windus,</subfield>
   <subfield type="c">1884.</subfield>
</field>
```

```
<field type="300" i1=" " i2=" ">
   <subfield type="a">xvi, 438 p., [1] leaf of plates :</subfield>
   <subfield type="b">ill. ;</subfield>
   <subfield type="c">20 cm.</subfield>
</field>
<field type="500" i1=" " i2=" ">
   <subfield type="a">First English ed.</subfield>
</field>
<field type="500" i1=" " i2=" ">
   <subfield type="a">State B; gatherings saddle-stitched with wire staples.</s
</field>
<field type="500" i1=" " i2=" ">
   <subfield type="a">Advertisements on p. [1]-32 at end.</subfield>
</field>
<field type="500" i1=" " i2=" ">
   <subfield type="a">Bound in red S cloth; stamped in black and gold.</subfiel
</field>
<field type="510" i1="4" i2=" ">
   <subfield type="a">BAL</subfield>
   <subfield type="c">3414.</subfield>
</field>
<field type="740" i1="0" i2="1">
   <subfield type="a">Huckleberry Finn.</subfield>
</field>
<field type="994" i1=" " i2=" ">
   <subfield type="a">E0</subfield>
   <subfield type="b">VOD</subfield>
</field>
</record>

</marc>
```

## METHODS

Here is a list of methods available to you in MARC::XML.

### new()

MARC::XML overides MARC.pm‘s new() method to create a MARC::XML object. Similar to MARC.pm‘s new() it can take two arguments: a file name, and the format of the file to read in. However MARC::XML‘s new() gives you an extra format choice "XML" (which is also the default). Internally, the XML source is converted to a series of **addfield()** and **createrecord()** calls. The order of MARC tags is preserved by default. But if an optional third argument is passed to new(), it is used as the *ordered* option for the **addfield()** calls. Due to the nature of XML::Parser, it is not possible to read only part of an XML input file. Some examples:

```
   #read in an XML file called myxmlfile.xml
use MARC::XML;
$x = MARC::XML->new("myxmlfile.xml","xml");
$x = MARC::XML->new("needsort.xml","xml","y");
```

Since the full funtionality of MARC.pm is also available you can read in other types of files as well. Although new() with no arguments will create an object with no records, just like MARC.pm, XML format not supported by openmarc() and nextmarc() for input. But you can output XML from a different format source.

```
   #read in a MARC file called mymarcfile.mrc
use MARC::XML;
$x = MARC::XML->new("mymarcfile.mrc","usmarc");
$x = MARC::XML->new();
```

### output()

MARC::XML‘s output() method allows you to output the MARC object as an XML file. It takes four arguments: *file*, *format*, *lineterm*, and *records*.

```
use MARC::XML;
$x = MARC::XML->new("mymarcfile.mrc","usmarc");
$x->output({file=>">myxmlfile.xml",format=>"xml"});
```

Or if you only want to output the first record:

```
$x->output({file=>">myxmlfile.xml",format=>"xml",records=>[1]});
```

If you like you can also output portions of the XML file using the *format* options: *xml_header*, *xml_body*, and *xml_footer*. Remember to prefix your file name with a to append though. This example will output record 1 twice.

```
use MARC::XML;
$x = MARC::XML->new("mymarcfile.mrc","usmarc");
$x->output({file=>">myxmlfile.xml",format=>"xml_header"});
$x->output({file=>">>myxmlfile.xml",format=>"xml_body",records=>[1]});
$x->output({file=>">>myxmlfile.xml",format=>"xml_body",records=>[1]});
$x->output({file=>">>myxmlfile.xml",foramt=>"xml_footer"});
```

Instead of outputting to a file, you can also capture the output in a variable if you wish.

```
use MARC::XML;
$x = MARC::XML->new("mymarcfile.mrc","usmarc");
$myxml = $x->output({format=>"xml"});
```

As with new() the full functionality of MARC.pm‘s output() method are available to you as well. So you could read in an XML file and then output it as ascii text:

```
use MARC::XML;
$x = MARC::XML->new("myxmlfile.xml","xml");
$x->output({file=>">mytextfile.txt","ascii");
```

## EXAMPLES

The **eg** subdirectory contains a few complete examples to get you started.

## AUTHORS

Chuck Bearden cbearden@rice.edu
Bill Birthisel wcbirthisel@alum.mit.edu
Derek Lane dereklane@pobox.com
Charles McFadden chuck@vims.edu
Ed Summers esummers@odu.edu

## SEE ALSO

perl(1), MARC.pm, MARC http://lcweb.loc.gov/marc , XML http://www.w3.org/xml .

## COPYRIGHT