System Administration: LibCB Extract Ejournals Script

Created By: Jeff Suszczynski
Created on: 5/15/2019

[This article was copied from the Voyager Wiki.]

This Perl script, with modifications as needed by your site, will extract your Ejournal holdings information and output to a pipe-delimited, flat-text file. To be used as part of the University of Rochester's LibCB (open source course resources and management system).

```
######################################################################

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

#

# author: jeff suszczynski

# email: jeffs@library.rochester.edu

# last mod: 02/27/03

# requires DBI and DBD:Oracle, MARC.pm

#

# Generates a flat-text file containing all Ejournal

# information.  This file will be put into a directory on our Voyager

# server that can accept a CFHTTP command, where a scheduled task on our Cold

# Fusion server will grab it and populate an SQL database with the data.

# This is a little clunky, but will have to do the trick until I port my Perl

# scripts into Cold Fusion - which will require installation of Perl and

# MARC.pm on the Cold Fusion server...

#

# If you don't currently extract your Ejournal information for something

# like an Ejournal Weblist, it is strongly recommended that you contact

# us before doing using this script.
```

```
#
# This Perl script should be run as a cron job.  We run ours nightly at 10pm.
# We have > 3 million holdings, and the script takes ~ 45 minutes to run.
#########################################################################

#############################################################
# Setting up local variables
#############################################################

$ENV{'ORACLE_HOME'}="/m1/oracle/app/oracle/product/8.0.5/";

$dbuser = 'xxx@LIBR';  # use login@INSTANCE syntax
$dbpassw = 'xxx';  #   (use a read-only login)

# this next value should be your database name
# which if your installation were called 123 it would likely
# look something like 123DB

$dbname = 'xxxdb';    # also known as Tablespace Name
$dbowner = 'xxxdb';   # usually the same as $dbname

# the following will be a path to a directory that is available
# for a CFHTTP.  For Voyager libraries, this may be something like
# /m1/voyager/$dbname/webvoyage/html/cfstuff

$cfhttp_dir = '';

################/ end of local variables /######################
```

```
# best practice is to use strict.  this script does not, however.
# use strict;

# this script requires DBI/DBD as well as MARC.pm
# in the near future, I will finally be switching to
# MARC::Record, the more current version of MARC.pm

use MARC;
use DBI;

#########################################################
# Connect to database, creating a database handle
#########################################################

$dbh = DBI->connect('dbi:Oracle:', $dbuser, $dbpassw, {PrintError=>1})
        || die "ERROR:connect: $DBI::errstr\n";

#################################################################
# First SQL statement, needed to pull records that contain
# 'Journal xyz Internet' in the BLOB.  The UR catalogs all electronic
# journals with 'Journal xyz Internet' in the 999 $a, where 'xyz'
# is an institution name.
# There is a better way to do this initial query using MARC.pm, but
# it would take quite a long time to run - so this is my workaround,
# looking for certain strings in the bibliographic BLOB.
#################################################################
```

```
$sqlstmt=

"SELECT DISTINCT

        bt.title,

        bm.mfhd_id,

        bi.bib_id,

        bt.issn,

        blob.record_segment


FROM

        $dbname.bib_text bt,

        $dbname.bib_mfhd bm,

        $dbname.bib_index bi,

        $dbname.bib_data blob,

        $dbname.bib_master mast


WHERE

        (blob.record_segment LIKE '%RCL Internet%'

        OR blob.record_segment LIKE '%9990025%')

        AND bi.bib_id = blob.bib_id

        AND bi.bib_id = bt.bib_id

        AND bt.bib_format = 'as'

        AND bi.bib_id = bm.bib_id

        AND bi.bib_id = mast.bib_id

        AND mast.suppress_in_opac <> 'Y'

ORDER BY bt.title ASC";


#############################################################

# Prepare and execute the first SQL statement
```

```
############################################################

$sth = $dbh->prepare($sqlstmt)
        || die "ERROR:prepare: $DBI::err: $DBI::errstr\n";
$rc = $sth->execute
        || die "ERROR:execute: $DBI::err: $DBI::errstr\n";

#########################################################
# Opens two files, called main_web.unsorted and nourl.unsorted
#########################################################

open(DATAOUT, ">cfej.unsorted") || die "File I/O error try again";
open(NOURLOUT, ">cfej.nourl") || die "NoUrl File I/O error try again";

#########################################################
# While loop to retrieve query data
#########################################################

while (($Jtitle, $mfhd, $bib, $issn, $marc_record) = $sth->fetchrow_array)

{
# first let's clean up the Journal title a bit.

        $Jtitle =~ s/\.\s$//;
        $Jtitle =~ s/\.$//;
        $Jtitle =~ s/\s$//;
        $Jtitle =~ s/^(A|An|The)\s//;
        $Jtitle=~ s/\[computer file\].*//;
```

```
        $Jtitle=~ s/\[electronic resource\].*//;


############################################################
# This next line calls a subroutine, written by Michael Doran, which will
# run each character in $Jtitle through a subroutine which strips out diacritics.
############################################################


        $Jtitle =~ s/([^\w\s\,\<\.\>\/\?\;\:\'\"\[\{\]\}\`\~\!\@\#\$\%\&\(\)\-
\=\+])/&RemoveDiacritics/ge;


##### Initialize values for $mfhd_data and $segcnt #####


        $mfhd_data = '';
        $segcnt = 0;


############################################################
# Second SQL statement, used to pull the MFHD (holdings) information out
# After we get MFHD info, we can pull specific fields from the MFHD
# using MARC.pm
############################################################


        $sql2=
        ("SELECT
                $dbname.mfhd_data.record_segment
        FROM
                $dbname.mfhd_data
        WHERE
                $dbname.mfhd_data.mfhd_id = $mfhd
```

```
        ORDER BY seqnum") || die $DBI::errstr;

################################################
# Prepare and execute the second SQL query
################################################

        $sth2 = $dbh->prepare($sql2)
        || die "ERROR:prepare: $DBI::err: $DBI::errstr\n";
        $rc2 = $sth2->execute
        || die "ERROR:execute: $DBI::err: $DBI::errstr\n";

################################################
# while loop that will fetch the results of the
# second query (the MFHD record) and act on them with marc.pm
################################################

        while (($record_segment) = $sth2->fetchrow_array)
               {
        $segcnt = $segcnt + 1;
        $mfhd_data = $mfhd_data . $record_segment;
        }

################################################
# Open new filehandle, put info into mfhd.unsorted
################################################

        open(MFHDOUT, ">mfhd.unsorted") || die "File I/O MFHD error try again";
        print MFHDOUT "$mfhd_data";
```

```
        close(MFHDOUT);


############################################################
# This is a function of the MARC.pm module.  Will open the file of MARC
# record and parse through it.  Looking for specific fields,
# 852, 856 and 866 for URL, source and coverage info.
############################################################

        $x= new MARC;
        $x->openmarc({file=>"mfhd.unsorted", 'format'=>"usmarc",increment=>"-1"});

                @Source=$x->getvalue({record=>'1',field=>'852',subfield=>"z"});

                @URL=$x->getvalue({record=>'1',field=>'856',subfield=>'u'});

                @Note=$x->getvalue({record=>'1',field=>'856',subfield=>'z'});

                @Coverage=$x->getvalue({record=>'1',field=>'866',subfield=>'a'});

        $x->closemarc();


############################################################
# join the arrays for coverage, note and url
# so that they will be a continuous string in
# the SQL database when the file is used to populate
# said database.
# also perform some cleanup on the URL.
############################################################
```

```
        $URL = join($",@URL);
        $URL=~ s/onClick.*//;
        $Coverage= join($",@Coverage);
        $Note= join("+",@Note);

#####################################################
# Call to subroutine which looks for regular expressions
# in the URL to add Source info which is not coded
# in the Holdings record
#####################################################

        &source;

#####################################################
# This tells the program to put MFHDs without a URL
# into a separate file, NOURLOUT (not really useful to this project)
#####################################################

        if ($URL eq '')
                { print NOURLOUT "$bib|$mfhd|$Source|$Jtitle|$issn|$Coverage|$Note\n";}
    else
        { print DATAOUT "$bib|$mfhd|$Source|$Jtitle|$issn|$URL|$Coverage|$Note\n";}
}

#############################################################
# Finish with the statement handles, disconnect from database
#############################################################
```

```
$sth->finish;

$sth2->finish;


# logout from the database


$dbh->disconnect || warn $DBI::errstr;


############################################################

# Now to sort the file and put newly sorted data into another

# file, using the system command

############################################################


system("sort -o cfejournals.txt -t '|' -f -u -d +3 +1 +0 cfej.unsorted");


#########################################################

# This will now copy the newly created file into a directory

# that can be reached with a CFHTTP, then delete the originals.

# No need for us to ftp because our Voyager web server resides on/in

# the same server that holds our Voyager database.  Your situation

# may be different.

#########################################################


system("cp cfejournals.txt $cfhttp_dir/ejournals.txt");

system("rm cfejournals.txt cfej.unsorted");


# close filehandles
```

```
close (NOURLOUT);

close (DATAOUT);


exit;


#########################################################

# Subroutine to find sources for Ejournals that don't

# have the information in the 852 $z.  Basically just looks

# to match regular expressions in the URL.

# This list is obviously incomplete, and only assigns

# a Source for those vendors who have a decent amount

# of Ejournals.  Standalone Ejournals will most likely

# not have anything in Source.

#########################################################


sub source

        {


        if ($URL=~ /.*proquest.*/)

               {$Source = "ProQuest";}

        elsif ($URL=~ /.*jstor.*/)

               {$Source = "JSTOR";}

        elsif ($URL=~ /.*muse\.jhu.*/)

               {$Source = "Project Muse";}

        elsif ($URL=~ /.*\.gov.*/)

               {$Source = "Government Publication";}

        elsif ($URL=~ /.*link\.springer.*/)

               {$Source = "Springer-Verlag";}
```

```
elsif ($URL=~ /.*interscience\.wiley\.com.*/)
       {$Source = "Wiley Interscience";}
elsif ($URL=~ /.*idealibrary.*/)
       {$Source = "IDEAL";}
elsif ($URL=~ /.*catchword.*/)
       {$Source = "Catchword";}
elsif ($URL=~ /.*sciencedirect\.com.*/)
       {$Source = "ScienceDirect";}
elsif ($URL=~ /.*pubs\.acs\.org.*/)
       {$Source = "American Chemical Society";}
elsif ($URL=~ /.*ojps\.aip\.org.*/)
       {$Source = "American Institute of Physics";}
elsif ($URL=~ /.*journals\.cup\.org.*/)
       {$Source = "Cambridge University Press";}
elsif ($URL=~ /.*press\.umich.*/)
       {$Source = "University of Michigan Press";}
elsif ($URL=~ /.*journals\.uchicago.*/)
       {$Source = "University of Chicago Press";}
elsif ($URL=~ /.*oup\.co\.uk.*/)
       {$Source = "Oxford University Press";}
elsif ($URL=~ /.*ingenta.*/)
       {$Source = "Ingenta";}
elsif ($URL=~ /.*\.rsc\.org.*/)
       {$Source = "Royal Society of Chemistry";}
elsif ($URL=~ /.*\.mrs\.org.*/)
       {$Source = "Materials Research Society";}
elsif ($URL=~ /.*\.iop\.org.*/)
```

```
        {$Source = "Institute of Physics";}
elsif ($URL=~ /.*opticsinfobase.*/)
        {$Source = "Optics InfoBase";}
elsif ($URL=~ /.*bloomberg\.com.*/)
        {$Source = "Bloomberg";}
elsif ($URL=~ /.*ama\.org.*/)
        {$Source = "American Marketing Association";}
elsif ($URL=~ /.*ams\.org.*/)
        {$Source = "American Mathematical Society";}
elsif ($URL=~ /.*frbchi.*/)
        {$Source = "Federal Reserve Bank of Chicago";}
elsif ($URL=~ /.*oupjournals\.org.*/)
        {$Source = "Oxford University Press";}
elsif ($URL=~ /.*bmjjournals\.com.*/)
        {$Source = "BMJ";}
elsif ($URL=~ /.*harvard\.edu.*/)
        {$Source = "Harvard University";}
elsif ($URL=~ /.*blackwellpublishers.*/)
        {$Source = "Blackwell Publishers";}
elsif ($URL=~ /.*dallasfed\.org.*/)
        {$Source = "Federal Reserve Bank of Dallas";}
elsif ($URL=~ /.*frbsf.*/)
        {$Source = "Federal Reserve Bank of San Francisco";}
elsif ($URL=~ /.*clev\.frb.*/)
        {$Source = "Federal Reserve Bank of Cleveland";}
elsif ($URL=~ /.*endojournals\.org.*/)
        {$Source = "Endocrine Society";}
```

```
        elsif ($URL=~ /.*cepr\.org.*/)

               {$Source = "Center for Economic Policy Research";}

        elsif ($URL=~ /.*\.asm\.org.*/)

               {$Source = "American Society for Microbiology";}

        elsif ($URL=~ /.*emerald-library\.com.*/)

               {$Source = "MCB University Press";}

        elsif ($URL=~ /.*biologists\.com.*/)

               {$Source = "The Company of Biologists, Ltd.";}

        elsif ($URL=~ /.*sgmjournals\.org.*/)

               {$Source = "Society for General Microbiology";}

        elsif ($URL=~ /.*aspetjournals\.org.*/)

               {$Source = "American Society for Pharmacology and Experimental
Therapeutics";}

        elsif ($URL=~ /.*\.acm\.org.*/)

               {$Source = "Association for Computing Machinery";}

        elsif ($URL=~ /.*\.nsba\.org.*/)

               {$Source = "National School Boards Association";}

        elsif ($URL=~ /.*\.siam\.org.*/)

               {$Source = "Society for Industrial and Applied Mathematics";}

        elsif ($URL=~ /.*\.imf\.org.*/)

               {$Source = "International Monetary Fund";}

        elsif ($URL=~ /.*ahajournals\.org.*/)

               {$Source = "American Heart Association";}

        elsif ($URL=~ /.*vanderbilt\.edu.*/)

               {$Source = "Vanderbilt University";}

        elsif ($URL=~ /.*\.journals\.cambridge\.org.*/)

               {$Source = "Cambridge Journals Online";}

        elsif ($URL=~ /.*columbia\.edu.*/)
```

```
        {$Source = "Columbia University";}
elsif ($URL=~ /.*mpls\.frb\.fed\.\.us.*/)
        {$Source = "Federal Reserve Bank of Minneapolis";}
elsif ($URL=~ /.*epress\.com.*/)
        {$Source = "EPress";}
elsif ($URL=~ /.*ama-assn.*/)
        {$Source = "American Medical Association";}
elsif ($URL=~ /.*astm\.org.*/)
        {$Source = "ASTM";}
elsif ($URL=~ /.*ebsco\.com.*/)
        {$Source = "Ebsco Online";}
elsif ($URL=~ /.*wspc\.com\.sg.*/)
        {$Source = "WorldSciNet";}
elsif ($URL=~ /.*eg\.miner\.rochester\.edu:9000.*/)
        {$Source = "Miner Digital Library";}
elsif ($URL=~ /.*blackwell-synergy\.com.*/)
        {$Source = "Blackwell Science Synergy";}
elsif ($URL=~ /.*ada\.org.*/)
        {$Source = "American Dental Association";}
elsif ($URL=~ /.*nato\.int.*/)
        {$Source = "NATO";}
elsif ($URL=~ /.*wkap\.nl.*/)
        {$Source = "Kluwer";}
elsif ($URL=~ /.*nature\.com.*/)
        {$Source = "Nature Publishing Group";}
elsif ($URL=~ /.*bos\.frb\.org.*/)
        {$Source = "Federal Reserve Bank of Boston";}
```

```
        elsif ($URL=~ /.*wiley-vch\.de.*/)
                {$Source = "Wiley-VCH";}
        elsif ($URL=~ /.*intlpress\.com.*/)
                {$Source = "International Press";}

# clean up $Source a little

        else
                {
                $Source = join($",@Source);
       $Source =~ s/Available through\s//;
                $Source =~ s/Access restricted.*//;
       $Source =~ s/Access.*//;
                }
        }
#-------------------------------------------------
############################################################
#  RemoveDiacritics
#  This subroutine looks at the non-word, non-space
#  characters, supplied by the substitution function
#  that calls it.
#
#  If the character has a decimal value of 161 or greater
#  it is removed.  Otherwise it is left alone.  This has
#  to do with differences in character encoding in the
#  MARC and non-MARC worlds.
#
#  Characters with decimal values between 224 and 255
```

```
#  (hex E0-FF) are MARC "combining" (non-spacing)
#  graphical marks, chiefly diacritical marks. Ideally,
#  they would be combined with the adjacent spacing
#  graphic character to display the desired diacritic.
#  If these non-spacing marks are not removed, they
#  become unwanted (and unsightly & confusing) spacing
#  marks in our non-MARC world.
#
#  Characters with decimal values between 161 and 223
#  (hex A1-DF) are meant to be spacing graphical marks,
#  however they map to different characters in the MARC
#  world and the non-MARC world, so are removed.
#
# This subroutine will eventually be replaced by a subroutine
# written by Michael Doran which will render the diacritics.
##########################################################

sub RemoveDiacritics
        {
    my ($character) = @_;
    if (ord($character) >= 161)
                {
        return("")
        }
        else
                {
        return($character);
        }
```

```
        }
#-------------------------------------------------------
```

Report