```perl
package C4::Coupon;

# Copyright 2014 Britsh Council
#
# The United Kingdom's international organisation for cultural relations and
# educational opportunities.
#
# A registered charity: 209131 (England and Wales) SC037733 (Scotland)
#
# This module provides functionality to apply promotional coupons to various
# patron transactions such as registration etc.

use strict;
use warnings;
use C4::Context;
use Data::Dumper;
use MIME::Base64;
use Business::ISBN;
use LWP::UserAgent ();
use XML::Simple ('XMLin');
use Date::Simple ('date', 'today');

use C4::SQLHelper qw(SearchInTable InsertInTable UpdateInTable
GetRecordCount);

use vars qw(%book $IMAGE_ROOT $STATIC_URL $IMAGE_EXT $NO_IMAGE_NAME $VERSION
@ISA @EXPORT);

BEGIN {
    $VERSION     = 1.00;
    @ISA         = qw(Exporter);
    @EXPORT      = qw(getItemDetails);
  }

$IMAGE_EXT      = '.jpeg';
$NO_IMAGE_NAME  = 'noimage.jpg';
$STATIC_URL     = C4::Context->get_custom_config("static_url");
$IMAGE_ROOT     = C4::Context->get_custom_config("image_save_location");
%book    = ('id'              => undef,
            'isbn10'          => undef,
            'isbn13'          => undef,
            'vendor_id'       => undef,
            'type'            => undef,
            'title'           => undef,
            'author'          => undef,
            'publisher'       => undef,
            'description'     => undef,
            'date_published'  => undef,
            'copyright_year'  => undef,
            'subject'         => undef,
            'thumbnail_url'   => undef,
            'image_url'       => undef,
            'edition'         => undef,
            'number_of_pages' => undef,
            'imageflag'       => undef,
            'timestamp'       => undef,
            );


&testIt();


sub testIt (){
    getItemDetails('9780194422703');

    getItemDetails('9781860498824');
    #getItemDetails('9780194422703');
    #getItemDetails('9781860498824');
}

sub getItemDetails($) {
    my ($isbn) = shift;
    croak($isbn." is not a valid ISBN.\n") unless (Business::ISBN->new($isbn)->is_valid());
    my $mybook = checkNielsenCache($isbn);
    if ($mybook) {
        return $mybook if ($mybook->{'title'});
        my $lastFetchDate = Date::Simple->new($mybook->{'timestamp'});
        my $today         = Date::Simple->today();
        if ($lastFetchDate < $today) {
            $mybook = fetchDataFromNielsen($isbn, $mybook->{'id'});
        }
    } else {
        $mybook = fetchDataFromNielsen($isbn, undef);
    }
    return $mybook;
}

sub checkNielsenCache($) {
    my ($isbn13, $result);
    $isbn13  = Business::ISBN->new(shift)->as_isbn13();
    $result =  SearchInTable('externaldata', {'isbn13' => $isbn13->isbn(), 'vendor_id'=>'NIELSEN'}, undef, undef, undef, undef, "exact")->[0];
    return $result unless ($result);
    if ($result->{'imageflag'} eq '1') {
        $result->{'thumbnail_url'}  = $STATIC_URL.$isbn13->isbn().$IMAGE_EXT;
        $result->{'image_url'}      = $STATIC_URL.$isbn13->isbn().$IMAGE_EXT;
    } elsif ($result->{'imageflag'} eq '2') {
        $result->{'thumbnail_url'}  = $STATIC_URL.$NO_IMAGE_NAME;
        $result->{'image_url'}      = $STATIC_URL.$NO_IMAGE_NAME;
    }
    return $result
}

sub cacheNielsenData {
    return InsertInTable ('externaldata',\%book, undef);
}

sub fetchDataFromNielsen() {
    my ($isbntext, $cacheId) = @_;
    my ($isbnobj, $isbn10, $isbn13, $resourceUrl, $bookinfo, $imageflag);
    my ($image, $image_file_name, $outfile, $flag);
    $isbnobj = Business::ISBN->new($isbntext);
    getNielsenInfo($isbnobj->isbn());
    $image = getNielsenImage($isbnobj->isbn()) if($book{'imageflag'} eq '1');
    if ($image) {
        $image_file_name  = $isbnobj->as_isbn13()->isbn().$IMAGE_EXT;
        open($outfile, '>', $IMAGE_ROOT.$image_file_name) or croak $!;
        binmode ($outfile);
        print {$outfile} decode_base64($image);
        $book{'imageflag'}  = '1';
    } else {
        $image_file_name  = $NO_IMAGE_NAME;
        $book{'imageflag'}  = '2';
    }
    if ($cacheId) {
        $book{'id'} = $cacheId;
        UpdateInTable('externaldata', \%book);
    } else {
        InsertInTable ('externaldata',\%book, undef);
    }
    $book{'thumbnail_url'}  = $STATIC_URL.$image_file_name;
    $book{'image_url'}      = $STATIC_URL.$image_file_name;
    return \%book;
}

sub getNielsenInfo {
    my ($isbnText) = shift;
    my ($resourceUrl, $bookinfo);
    my $isbn     = Business::ISBN->new($isbnText);
    $resourceUrl = getNielsenAPIUrl($isbn->isbn(), 'info');
    $bookinfo    = parseNielsenXML(get($resourceUrl), 'info');
    $book{'isbn10'}             = $isbn->as_isbn10()->isbn();
    $book{'isbn13'}             = $isbn->as_isbn13()->isbn();
    $book{'vendor_id'}          = 'NIELSEN';
    $book{'type'}                 = 'BOOK';
    $book{'title'}                = $bookinfo->{'TL'};
    my @authors;
    my $author;
    for (my $count = 1; $count <=10; $count++) {
        $author = $bookinfo->{'CNF'."$count"};
        push(@authors, $author) if ($author);
    }
    $book{'author'}               = join(',', @authors);
    $book{'publisher'}            = $bookinfo->{'IMPN'};
    $book{'date_published'}       = Date::Simple->new($bookinfo->{'PUBPD'})-
>as_iso() if ($bookinfo->{'PUBPD'});
    $book{'subject'}              = $bookinfo->{'BIC2ST1'};
    $book{'number_of_pages'}      = $bookinfo->{'PAGNUM'};
    $book{'imageflag'}            = ($bookinfo->{'IMAGFLAG'} eq 'Y')? '1':'2';
    $book{'edition'}              = '';
    if (defined($bookinfo->{'NBDSD'}) && $bookinfo->{'NBDSD'} ne '') {
        unless($book{'description'} = $bookinfo->{'NBDSD'}) {
            $book{'description'} = 'No Description Available!';
        }
    } else {
        unless($book{'description'} = $bookinfo->{'CIS'}) {
            $book{'description'} = 'No Description Available!';
        }
    }
}

sub getNielsenImage {
    my ($isbnText) = shift;
    my ($resourceUrl, $image);
    my $isbn       = Business::ISBN->new($isbnText);
    $resourceUrl = getNielsenAPIUrl($isbn->isbn(), 'image');
    $image       = parseNielsenXML(get($resourceUrl), 'image');
    return $image;
}

sub getNielsenAPIUrl {
    my ($isbn, $dataType)  = @_;
    my ($host, $path, %queryParams, $queryString);
    $host ='http://ws.nielsenbookdataonline.com';
    $path ='/BDOLRest/RESTwebServices/BDOLrequest';
    %queryParams     = (
                        'clientId'   => 'BcouncildelhiBDWS01',
                        'password'   => 'kcl510dk4873',
                        'format'     => '7',
                        'from'       => '0',
                        'to'         => '10',
                        'resultView' => '2',
```

```perl
                            'value0'     => $isbn,
                            'logic0'     => '0',
                            'sortField0' => '0'
    );
    if (uc($dataType) eq 'INFO') {
        $queryParams{'field0'} = '0';
        $queryParams{'indexType'} = '0';
    }  elsif (uc($dataType) eq 'IMAGE') {
        $queryParams{'field0'} = '1';
        $queryParams{'indexType'} = '2';
    } else {
        warn("Requested URL can not be obtained.
              Argument should be either INFO or
              IMAGE to this routine");
              return undef;
    }
    foreach my $key ( keys %queryParams ) {
        $queryString .= $key.'='.$queryParams{$key}.'&';
    }
    return $host . $path . '?' . $queryString;
}

sub parseNielsenXML {
    my ($xml, $dataType) = @_;
    my $bookxml;
    $xml = '<data></data>' unless(defined($xml));
    $xml  =~ s/\<\?xml.*?\?\>//g; # Remove XML declaration if there is one
    if (uc($dataType) eq 'INFO') {
        $bookxml = XMLin($xml)->{'data'}->{'data'}->{'record'};
    }  elsif (uc($dataType) eq 'IMAGE') {
        $bookxml = XMLin($xml)->{'data'};
    } else {
        warn("Requested URL can not be obtained.
              Argument should be either INFO or
              IMAGE to this routine");
              return undef;
    }
    return $bookxml;
}

sub get ($) {
    my $ua = LWP::UserAgent->new;
    $ua->timeout(10);
    my $response = $ua->get(shift);
    return $response->decoded_content if $response->is_success;
    return undef;
}

```

```
END { }     # module clean-up code here (global destructor)

1;

__END__

=head1 AUTHORS

Techletsolutions Team

=cut
```