

[tutoriel](#)

GCstar : Plugins internet

Pré-requis

Première étape

Autres étapes

version fonctionnelle pour le site de la Fnac

[GCFnac.pm](#)[/usr/share/gcstar/lib/GCPlugins/GCbooks/GCFnac.pm](#)

```
package GCPlugins::GCbooks::GCFnac;

#####
#
# Copyright 2005-2006 Tian
#
# This file is part of GCstar.
#
# GCstar is free software; you can redistribute it and/or modify
# it under the terms of the GNU General Public License as
# published by
# the Free Software Foundation; either version 2 of the License,
# or
# (at your option) any later version.
#
# GCstar is distributed in the hope that it will be useful,
# but WITHOUT ANY WARRANTY; without even the implied warranty of
# MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
# GNU General Public License for more details.
#
# You should have received a copy of the GNU General Public
# License
# along with GCstar; if not, write to the Free Software
# Foundation, Inc., 51 Franklin St, Fifth Floor, Boston, MA
# 02110-1301, USA
#
#####
```

```
use strict;
use utf8;

use GCPlugins::GCbooks::GCbooksCommon;

{
    package GCPlugins::GCbooks::GCPluginFnac;

    use base qw(GCPlugins::GCbooks::GCbooksPluginsBase);
    use URI::Escape;

    sub start
    {
        my ($self, $tagname, $attr, $attrseq, $origtext) = @_;

        $self->{inside}->{$tagname}++;

        if ($self->{parsingList})
        {

            # Détection début d'un nouvel ouvrage de la liste
            if ($attr->{class} eq 'js-minifa-title')
            {
                # Le prochain bloc de texte est le titre
                $self->{isTitle} = 1 ;
                # Créer la nouvelle entrée
                $self->{itemIdx}++;
                # Récupération de la page concernant l'ouvrage seul
                $self->{itemsList}[$self->{itemIdx}]->{url} =
$attr->{href};
                return;
            }

            # Détection éditeur + date
            elsif ($tagname eq 'vark')
            {
                # Le bloc de texte après le suivant contient l'éditeur et
la date
                $self->{isPublisher} = 1 ;
            }

        }
        else
        {
            # Détection éditeur
            if ($tagname eq 'varkeditor')
            {
                $self->{isPublisher} = 3 ;
            }
            # Détection ISBN
        }
    }
}
```

```
elseif ($tagname eq 'varkISBN')
{
$self->{isISBN} = 3 ;
}
# Détection pages
elseif ($tagname eq 'varkpages')
{
$self->{isPage} = 3 ;
}
# Détection date
elseif ($tagname eq 'varkdate')
{
$self->{isPublication} = 3 ;
}
# Détection auteurs
elseif ($tagname eq 'varkauthors')
{
$self->{isAuthor} = 3 ;
}
# Détection format
elseif ($tagname eq 'varkformat')
{
$self->{isFormat} = 3 ;
}
# Détection traducteur
elseif ($tagname eq 'varktranslator')
{
$self->{isTranslator} = 3 ;
}
# Détection titre
elseif ($tagname eq 'varkttitle')
{
$self->{isTitle} = 3 ;
}
# Capture image
elseif ($tagname eq 'varkimage')
{
$self->{curInfo}->{cover} = $attr->{src};
}
}

sub end
{
my ($self, $tagname) = @_ ;

$self->{inside}->{$tagname}--;

# Arrêt de l'ajout d'auteurs
if (($self->{isAuthor} == 3) && ($tagname eq 'li'))
{
```

```
        $self->{isAuthor} = 0;
    }
}

sub text
{
    my ($self, $origtext) = @_ ;

    if ($self->{parsingList})
    {
        # Capture du titre
        if ($self->{isTitle} == 1)
        {
            # Enleve les blancs en debut de chaine
            $origtext =~ s/^\s+//;
            # Enleve les blancs en fin de chaine
            $origtext =~ s/\s+$//g;
            $self->{itemsList}[$self->{itemIdx}]->{title} =
$origtext;
            $self->{isTitle} = 0 ;
            # Le texte suivant contient l'auteur
            $self->{isAuthor} = 1 ;
            return;
        }
        # Capture auteur
        elsif ($self->{isAuthor} == 1)
        {
            # Enleve les blancs en debut de chaine
            $origtext =~ s/^\s+//;
            # Enleve les blancs en fin de chaine
            $origtext =~ s/\s+$//g;
            if ($origtext ne '')
            {
                $self->{itemsList}[$self->{itemIdx}]->{authors} =
$origtext;
                $self->{isAuthor} = 0 ;
            }
        }
        elsif ($self->{isPublisher} == 1)
        {
            # Passe le texte contenant le type d'ouvrage; le texte
            # suivant contient éditeur et date
            $self->{isPublisher} = 2 ;
            return ;
        }
        # Capture éditeur et date
        elsif ($self->{isPublisher} == 2)
        {
            my @array = split(/-/, $origtext);
            $array[2] =~ s/^\s+//;
        }
    }
}
```

```
$array[2] =~ s/\s+$/g;
$array[3] =~ s/^\s+//;
$array[3] =~ s/\s+$/g;
    $self->{itemsList}[$self->{itemIdx}]->{edition} =
$array[2];
$self->{itemsList}[$self->{itemIdx}]->{publication} = $array[3];
    $self->{isPublisher} = 0 ;
}
}
else
{
    # Enleve les blancs en debut de chaine
    $origtext =~ s/^\s+//;
    # Enleve les blancs en fin de chaine
    $origtext =~ s/\s+$/g;

    # Capture titre
    if ($self->{isTitle} == 3)
    {
        $self->{curInfo}->{title} = $origtext;
        $self->{isTitle} = 0 ;
    }
    # Capture auteurs
    elsif (($self->{isAuthor} == 3) && ($origtext ne ','))
    {
        if ($self->{curInfo}->{authors} eq '')
        {
            $self->{curInfo}->{authors} = $origtext;
        }
    }
    else
    {
        $self->{curInfo}->{authors} .= ", " . $origtext;
    }
}
# Capture ISBN
elsif ($self->{isISBN} == 3)
{
    if ($origtext ne '')
    {
        $self->{curInfo}->{isbn} = $origtext;
        $self->{isISBN} = 0 ;
    }
}
# Capture éditeur
elsif ($self->{isPublisher} == 3)
{
    if ($origtext ne '')
    {
        $self->{curInfo}->{publisher} = $origtext;
        $self->{isPublisher} = 0 ;
    }
}
```

```
    }
    # Capture format
    elseif ($self->{isFormat} == 3)
    {
        if ($origtext ne '')
        {
            $self->{curInfo}->{format} = $origtext;
            $self->{isFormat} = 0 ;
        }
    }
    # Capture date
    elseif ($self->{isPublication} == 3)
    {
        if ($origtext ne '')
        {
            $self->{curInfo}->{publication} = $origtext;
            $self->{isPublication} = 0 ;
        }
    }
    # Capture pages
    elseif ($self->{isPage} == 3)
    {
        if ($origtext ne '')
        {
            $self->{curInfo}->{pages} = $origtext;
            $self->{isPage} = 0 ;
        }
    }
    # Capture traducteur
    elseif ($self->{isTranslator} == 3)
    {
        if ($origtext ne '')
        {
            $self->{curInfo}->{translator} = $origtext;
            $self->{isTranslator} = 0 ;
        }
    }
    # Capture description
    elseif (($self->{isDescription} == 4) && ($origtext ne
''))
    {
        $self->{curInfo}->{description} = $origtext;
        $self->{isDescription} = 0;
    }
    # Détection description (on saute une zone de texte)
    elseif (($self->{isDescription} == 3) && ($origtext ne
''))
    {
        $self->{isDescription} = 4;
    }
}
```

```

        # Détection description (elle est située deux zones de
        texte plus loin)
        elsif ($origtext eq 'Le mot de l\'éditeur')
        {
            $self->{isDescription} = 3;
        }
    }
}

sub new
{
    my $proto = shift;
    my $class = ref($proto) || $proto;
    my $self = $class->SUPER::new();
    bless ($self, $class);

    $self->{hasField} = {
        title => 1,
        authors => 1,
        publication => 1,
        format => 0,
        edition => 1,
        serie => 0,
    };

    $self->{isUrl} = 0;
    $self->{isTitle} = 0;
    $self->{isAuthor} = 0;
    $self->{isPublisher} = 0;
    $self->{isISBN} = 0;
    $self->{isPublication} = 0;
    $self->{isFormat} = 0;
    $self->{isSerie} = 0;
    $self->{isPage} = 0;
    $self->{isDescription} = 0;
    $self->{isCover} = 0;
    $self->{isTranslator} = 0;

    return $self;
}

sub preProcess
{
    my ($self, $html) = @_;

    if ($self->{parsingList})
    {
        # Mise en forme pour détecter facilement les éditeur et
        date
        $html =~ s|<div
        class='editorialInfo'><strong>|<vark>|gmi;

```

```
    }
    else
    {

        $html =~ s|<span class="Feature-
label"><span>Editeur</span></span>|<varkeditor>|omi;
        $html =~ s|<span class="Feature-label"><span>Date de
parution</span></span>|<varkdate>|omi;
        $html =~ s|<span class="Feature-
label"><span>EAN</span></span>|<varkISBN>|omi;
        $html =~ s|<span class="Feature-label"><span>Nombre de
pages</span></span>|<varkpages>|omi;
        $html =~ s|<span class="Feature-
label"><span>Auteur</span></span>|<varkauthors>|omi;
        $html =~ s|<span class="Feature-
label"><span>Format</span></span>|<varkformat>|omi;
        $html =~ s|<span class="Feature-
label"><span>Traduction</span></span>|<varktranslator>|omi;
        $html =~ s|h2 class="FAstrate-title"><span
class="FAstrate-title-color js-ProductSticky-
title">Caractéristiques détaillées</span><span class="FAstrate-
subtitle">|<varkttitle>|omi;
        $html =~ s|img class="js-ProductVisuals-
imagePreview"|varkimage|omi;

        $html =~ s|<li>|\n*|gi;
        $html =~ s|<br>|\n|gi;
        $html =~ s|<br />|\n|gi;
        $html =~ s|<b>|gi;
        $html =~ s|</b>|gi;
        $html =~ s|<i>|gi;
        $html =~ s|</i>|gi;
        $html =~ s|<p>|\n|gi;
        $html =~ s|</p>|gi;
        $html =~ s|</h4>|gi;
        $html =~ s|\x{92}|'|gi;
        $html =~ s|&#146;|'|gi;
        $html =~ s|&#149;|*|gi;
        $html =~ s|&#133;|\...|gi;
        $html =~ s|\x{85}|\...|gi;
        $html =~ s|\x{8C}|OE|gi;
        $html =~ s|\x{9C}|oe|gi;

    }

    return $html;
}

sub getSearchUrl
{
```



```
        my ($self, $word) = @_;  
  
        return  
        "http://www.fnac.com/search/quick.do?filter=-3&text=". $word  
        . "&category=book";  
    }  
  
    sub getItemUrl  
    {  
        my ($self, $url) = @_;  
  
        return $url if $url;  
        return 'http://www.fnac.com/';  
    }  
  
    sub getName  
    {  
        return "Fnac (FR)";  
    }  
  
    sub getCharset  
    {  
        my $self = shift;  
        return "ISO-8859-15";  
    }  
  
    sub getAuthor  
    {  
        return 'Varkolak';  
    }  
  
    sub getLang  
    {  
        return 'FR';  
    }  
  
    sub getSearchFieldsArray  
    {  
        return ['isbn', 'title', 'author'];  
    }  
}  
  
1;
```

[GCFnac.pm](#)

[GCFnac.pm](#)

```
package GCPlugins::GCbooks::GCFnac;
```

```
#####  
#  
# Copyright 2005-2006 Tian  
# Copyright 2015-2016 Kerenoc01 on Google Mail  
#  
# This file is part of GCstar.  
#  
# GCstar is free software; you can redistribute it and/or modify  
# it under the terms of the GNU General Public License as  
# published by  
# the Free Software Foundation; either version 2 of the License,  
# or  
# (at your option) any later version.  
#  
# GCstar is distributed in the hope that it will be useful,  
# but WITHOUT ANY WARRANTY; without even the implied warranty of  
# MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the  
# GNU General Public License for more details.  
#  
# You should have received a copy of the GNU General Public  
# License  
# along with GCstar; if not, write to the Free Software  
# Foundation, Inc., 51 Franklin St, Fifth Floor, Boston, MA  
# 02110-1301, USA  
#  
#####  
  
use strict;  
use utf8;  
  
use GCPlugins::GCbooks::GCbooksCommon;  
  
{  
    package GCPlugins::GCbooks::GCPluginFnac;  
  
    use base qw(GCPlugins::GCbooks::GCbooksPluginsBase);  
    use URI::Escape;  
  
    sub start  
    {  
        my ($self, $tagname, $attr, $attrseq, $origtext) = @_;  
  
        $self->{inside}->{$tagname}++;  
  
        if ($self->{parsingList})  
        {  
            if (($tagname eq 'div') && ($attr->{class} eq  
"Article-itemInfo"))  
            {  

```

```

        $self->{isTitle} = 1;
        $self->{isPublisher} = 0;
    }
    elsif (($self->{isTitle} eq 1) && ($tagname eq 'a') &&
($attr->{class} eq " js-minifa-title"))
    {
        $self->{itemIdx}++;
        $self->{itemsList}[$self->{itemIdx}]->{url} =
$attr->{href};
        $self->{isTitle} = 2;
    }
    elsif (($tagname eq 'p') && ($attr->{class} eq
"Article-descSub"))
    {
        $self->{isAuthor} = 1;
    }
    elsif (($self->{isAuthor} eq 1) && ($tagname eq 'a'))
    {
        $self->{isAuthor} = 2;
    }
    elsif ($self->{isAuthor} && $tagname eq 'div')
    {
        $self->{isAuthor} = 0;
    }
    elsif (($tagname eq 'div') && ($attr->{class} eq
"editorialInfo"))
    {
        $self->{isAnalyse} = 1;
    }
    }
    else
    {
        if (($tagname eq 'h1') && ($attr->{class} eq
'ProductSummary-title'))
        {
            $self->{isTitle} = 1 ;
        }
        elsif (($self->{isTitle} eq 1) && ($tagname eq 'span')
&& ($attr->{itemprop} eq 'name'))
        {
            $self->{isTitle} = 2 ;
        }
        elsif (($tagname eq 'div') && ($attr->{class} eq
'ProductSummary-subTitle'))
        {
            $self->{isAuthor} = 1 ;
        }
        elsif (($self->{isAuthor} eq 1) && ($tagname eq 'a'))
        {
            $self->{isAuthor} = 2 ;
        }
    }

```

```
        elseif (($tagname eq 'a') && ($attr->{class} eq
'expanding') && ($self->{bigPics}))
        {
            $self->{curInfo}->{cover} = $attr->{href} ;
        }
        elseif (($tagname eq 'img') && ($attr->{class} eq 'js-
ProductVisuals-imagePreview') && (!($self->{bigPics}) ||
($self->{curInfo}->{cover} eq '')))
        {
            $self->{curInfo}->{cover} = $attr->{src} ;
        }
        elseif ($tagname eq 'section' && $attr->{id} eq
'ficheResume')
        {
            $self->{isDescription} = 1 ;
        }
        elseif ($tagname eq 'div' && $attr->{class} eq
'productStrateTop')
        {
            $self->{isDescription} = 1 ;
        }
        elseif ($self->{isDescription} eq 1 && $tagname eq
'div' && $attr->{class} eq 'whiteContent')
        {
            $self->{isDescription} = 2 ;
        }
        elseif (($tagname eq 'ul') && ($attr->{class} =~
m/Feature-list/))
        {
            $self->{isAnalyse} = 1 ;
            $self->{isDescription} = 0;
        }
        elseif (($tagname eq 'span') && ($attr->{class} =~
m/Feature-label/))
        {
            $self->{isAnalyse} = 2 ;
        }
        elseif (($self->{isAnalyse} eq 1) && ($attr->{class} =~
m/Feature-desc/))
        {
            $self->{isPublisher} = 2 if ($self->{isPublisher});
            $self->{isPublication} = 2 if ($self->{isPublication});
            $self->{isSerie} = 2 if $self->{isSerie};
            $self->{isISBN} = 2 if $self->{isISBN};
            $self->{isPage} = 2 if $self->{isPage};
        }
    }
}

sub end
```

```

{
    my ($self, $tagname) = @_;

    $self->{inside}->{$tagname}--;

    if ($self->{isAnalyse} && $tagname eq 'div')
    {
        $self->{isAnalyse} = 0;
    }
    elsif ($self->{isAuthor} && $tagname eq 'div')
    {
        $self->{isAuthor} = 0;
    }
    elsif ($self->{isAnalyse} eq 2 && $tagname eq 'strong')
    {
        $self->{isAnalyse} = 1;
    }
    elsif ($self->{isDescription} eq 2 && $tagname eq 'div')
    {
        # parfois des descriptions en double : resume + le mot de
l'editeur
        # meme contenu avec une orthographe et une mise en page
différente!
        $self->{isDescription} = 0;
    }
}

sub text
{
    my ($self, $origtext) = @_;

    if ($self->{parsingList})
    {
        if ($self->{isTitle} eq 2)
        {
            # Enleve les blancs en debut de chaine
            $origtext =~ s/^\s+//;
            # Enleve les blancs en fin de chaine
            $origtext =~ s/\s+$//g;

            if (($self->{itemsList}[$self->{itemIdx}]->{title}
eq '' ) && ($origtext ne '' ))
            {
                $self->{itemsList}[$self->{itemIdx}]->{title} =
$origtext;
            }
            elsif ($origtext ne '')
            {
                $self->{itemsList}[$self->{itemIdx}]->{title}
.= ' - ';
                $self->{itemsList}[$self->{itemIdx}]->{title}

```

```
.= $origtext;
    }
    $self->{isTitle} = 0 ;
}
elseif ($self->{isAnalyse} > 0)
{
my @listInfo = split(/\n/, $origtext);
my $nbInfos = scalar @listInfo ;
if ($nbInfos eq 1)
{
    return;
}
else
{
    my $publication = $listInfo[$nbInfos-1];
    $publication =~ s/^\[-\s]+//;
    $publication =~ s/\s+$//g;
    $self->{itemsList}[$self->{itemIdx}]->{publication} =
$publication;
    my $edition = $listInfo[$nbInfos-2];
    $edition =~ s/^\[-\s]+//;
    $edition =~ s/\s+$//g;
    $self->{itemsList}[$self->{itemIdx}]->{edition} =
$edition;
}
}

elseif ($self->{isAuthor} eq 2)
{
    $origtext =~ s/^\s+//;
    $origtext =~ s/\s+$//g;

    if
(($self->{itemsList}[$self->{itemIdx}]->{authors} eq '') &&
($origtext ne ''))
    {
        $self->{itemsList}[$self->{itemIdx}]->{authors}
= $origtext;
    }
    elseif ($origtext ne '')
    {
        $self->{itemsList}[$self->{itemIdx}]->{authors}
.= ', ' . $origtext;
    }
    $self->{itemsList}[$self->{itemIdx}]->{authors}
.= $origtext;
}
    $self->{isAuthor} = 1;
}
}
else
{
```

```

# Enleve les blancs en debut de chaine
$origtext =~ s/^\s+//;
# Enleve les blancs en fin de chaine
$origtext =~ s/\s+$//g;
if ($self->{isTitle} eq '2')
{
    $self->{curInfo}->{title} = $origtext;
    $self->{isTitle} = 0 ;
}
elsif ($self->{isAnalyse} eq 2)
{
    $self->{isISBN} = 1 if ($origtext =~ m/ISBN/i);
    $self->{isPublisher} = 1 if ($origtext =~
m/Editeur/i);
    $self->{isFormat} = 1 if ($origtext =~
m/Format/i);
    $self->{isSerie} = 1 if ($origtext =~
m/Collection/i);
    $self->{isPublication} = 1 if ($origtext =~ m/Date
de parution/i);
    $self->{isPage} = 1 if ($origtext =~ m/pages/i);
    $self->{isTranslator} = 1 if ($origtext =~
m/Traduction/i);

    $self->{isAnalyse} = 1 ;
}
elsif ($self->{isAuthor} eq 2)
{
    # Enleve les virgules
    $origtext =~ s/,//;
    if ($origtext ne '')
    {
        $self->{author} = $origtext;
    }
    $self->{isAuthor} = 1;
}
elsif ($self->{isAuthor} eq 1)
{
    if ($origtext =~ m/\(Traduct/)
    {
        $self->{curInfo}->{translator} = $origtext;
    }
    elsif ($origtext =~ m/^\(/)
    {
        $self->{curInfo}->{authors} .= $self->{author};
        $self->{curInfo}->{authors} .= ", ";
    }
}
elsif ($self->{isISBN} eq 2)
{
    $self->{curInfo}->{isbn} = $origtext;
}

```

```
        $self->{isISBN} = 0 ;
    }
    elseif ($self->{isPublisher} eq 2)
    {
        if ($origtext ne '')
        {
            $self->{curInfo}->{publisher} = $origtext;
            $self->{isPublisher} = 0 ;
        }
    }
    elseif ($self->{isFormat} eq 2)
    {
        if ($origtext ne '')
        {
            $self->{curInfo}->{format} = $origtext;
            $self->{isFormat} = 0 ;
        }
    }
    elseif ($self->{isSerie} eq 2)
    {
        if ($origtext ne '')
        {
            $self->{curInfo}->{serie} = $origtext;
            $self->{isSerie} = 0 ;
        }
    }
    elseif ($self->{isPublication} eq 2)
    {
        $self->{curInfo}->{publication} =
        $self->decodeDate($origtext)
        if (!$self->{curInfo}->{publication});
        $self->{isPublication} = 0 ;
    }
    elseif (($self->{isPage} eq 2))
    {
        if ($origtext ne '')
        {
            $self->{curInfo}->{pages} = $origtext;
            $self->{isPage} = 0 ;
        }
    }
    elseif ($self->{isTranslator})
    {
        if ($origtext ne '')
        {
            $self->{curInfo}->{translator} = $origtext;
            $self->{isTranslator} = 0 ;
        }
    }
    elseif ($self->{isDescription} eq 2)
```



```

        {
            $origtext .= "\n";
            $self->{curInfo}->{description} .= $origtext;
        }
    }
}

sub new
{
    my $proto = shift;
    my $class = ref($proto) || $proto;
    my $self = $class->SUPER::new();
    bless ($self, $class);

    $self->{hasField} = {
        title => 1,
        authors => 1,
        publication => 1,
        format => 0,
        edition => 1,
        serie => 0,
    };

    $self->{isTitle} = 0;
    $self->{isAuthor} = 0;
    $self->{isPublisher} = 0;
    $self->{isISBN} = 0;
    $self->{isPublication} = 0;
    $self->{isFormat} = 0;
    $self->{isSerie} = 0;
    $self->{isPage} = 0;
    $self->{isDescription} = 0;
    $self->{isTranslator} = 0;

    return $self;
}

sub preProcess
{
    my ($self, $html) = @_;

    if ($self->{parsingList})
    {
        $html =~ s|</a><br>|</a><tpfpublicationtpf>|gmi;
    }
    else
    {
        # Le descriptif pouvant contenir des balises html je
        le repere maintenant
        my $found = index($html, "<strong>Mot de l'");
        if ( $found >= 0 )

```

```
        {
            my $html2 = substr($html, $found
+length('<strong>Mot de l\'''),length($html)- $found -
length('<strong>Mot de l\'''));
            my $found2 = index($html2,"<h4 ");
            my $html3 = $html2;
            if ( $found2 >= 0 )
            {
                $html3 = substr($html2, $found2 +length('<h4
''),length($html2)- $found2 -length('<h4 '));
                $html2 = substr($html2, 0, $found2);
            }

            $found2 = index($html2,"</strong>");
            if ( $found2 >= 0 )
            {
                $html2 = substr($html2, $found2
+length('</strong>'),length($html2)- $found2 -
length('</strong>'));
            }

            $html2 =~ s|<li>|\n* |gi;
            $html2 =~ s|<br>|\n|gi;
            $html2 =~ s|<br />|\n|gi;
            $html2 =~ s|<b>||gi;
            $html2 =~ s|</b>||gi;
            $html2 =~ s|<i>||gi;
            $html2 =~ s|</i>||gi;
            $html2 =~ s|<p>|\n|gi;
            $html2 =~ s|</p>||gi;
            $html2 =~ s|</h4>||gi;
            $html2 =~ s|\x{92}|\'|g;
            $html2 =~ s|&#146;|\'|gi;
            $html2 =~ s|&#149;|*|gi;
            $html2 =~ s|&#133;|\...|gi;
            $html2 =~ s|\x{85}|\...|gi;
            $html2 =~ s|\x{8C}|OE|gi;
            $html2 =~ s|\x{9C}|oe|gi;

        }

    }

    return $html;
}

sub getSearchUrl
{
    my ($self, $word) = @_;
```

```
        return
        "http://www3.fnac.com/search/quick.do?filter=-3&text=". $word
        "&category=book";
    }

    sub getItemUrl
    {
        my ($self, $url) = @_;

        return $url if $url;
        return 'http://www.fnac.com/';
    }

    sub getName
    {
        return "Fnac (FR)";
    }

    sub getCharset
    {
        my $self = shift;
        return "ISO-8859-15";
    }

    sub getAuthor
    {
        return 'TPF - Kerenoc';
    }

    sub getLang
    {
        return 'FR';
    }

    sub getSearchFieldsArray
    {
        return ['isbn', 'title'];
    }

    sub decodeDate
    {
        my ($self, $date) = @_;

        # date déjà dans le bon format
        return $date if ($date =~ m|/|);

        # date à convertir au format jour/mois/année

        my @dateItems = split(/\s/, $date);
        my @listeMois =
        ("janvier", "février", "mars", "avril", "mai", "juin",
```

```
"juillet", "août", "septembre", "octobre", "novembre", "décembre");  
my $mois = 0;  
  
while ($mois < (scalar @listeMois) && $dateItems[(scalar  
@dateItems)-2] ne $listeMois[$mois])  
{  
    $mois++;  
}  
return "01/".sprintf("%02d", $mois)."/".$dateItems[1];  
}  
  
1;
```

Amazon

[GCAmazon.pm](#)

[/usr/share/gcstar/lib/GCPlugins/GCbooks/GCAmazon.pm](#)

```
package GCPlugins::GCbooks::GCAmazon;  
  
#####  
#  
# Copyright 2005-2009 Tian  
#  
# This file is part of GCstar.  
#  
# GCstar is free software; you can redistribute it and/or modify  
# it under the terms of the GNU General Public License as  
# published by  
# the Free Software Foundation; either version 2 of the License,  
# or  
# (at your option) any later version.  
#  
# GCstar is distributed in the hope that it will be useful,  
# but WITHOUT ANY WARRANTY; without even the implied warranty of  
# MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the  
# GNU General Public License for more details.  
#  
# You should have received a copy of the GNU General Public  
# License  
# along with GCstar; if not, write to the Free Software  
# Foundation, Inc., 51 Franklin St, Fifth Floor, Boston, MA  
# 02110-1301, USA  
#  
#####
```

```
use strict;
use utf8;

use GCPlugins::GCbooks::GCbooksCommon;

{
    package GCPlugins::GCbooks::GCPluginAmazon;

    use base qw(GCPlugins::GCbooks::GCbooksPluginsBase);
    use XML::Simple;
    use LWP::Simple qw($ua);
    use Encode;
    use HTML::Entities;
    use GCUtils;

    sub start
    {
        my ($self, $tagname, $attr, $attrseq, $origtext) = @_;

        $self->{inside}->{$tagname}++;

        if ($self->{parsingList})
        {
            # Identify beginning of comments
            if (($self->{isComment} == 0) && ($tagname eq
'varkcomment'))
            {
                $self->{isComment} = 1 ;
            }

            # Capture URL of book
            if (($self->{isComment} == 0) && ($self->{isUrl} == 1) &&
($tagname eq 'a'))
            {
                $self->{itemsList}[$self->{itemIdx}]->{url} =
$attr->{href};
                $self->{isUrl} = 0 ;
                $self->{isTitle} = 1 ;
                return;
            }

            # Identify beginning of new book (next text is title)
            if (($self->{isComment} == 0) && ($tagname eq 'li') &&
($attr->{id} =~ /result_[0-9]+/ ))
            {
                # Create new entry
                $self->{itemIdx}++;
                $self->{isUrl} = 1 ;
                $self->{isAuthor} = 0 ;
                return ;
            }
        }
    }
}
```

```
    }

    # Identify end of authors list
    if (($self->{isComment} == 0) && ($tagname eq
'varkendauthors') && ($self->{isAuthor} != 0))
    {
        $self->{isAuthor} = 0 ;
        return ;
    }
}
else
{
    # Detection of book themes
    if (($self->{isTheme} == 0) && ($tagname eq 'varkgenre'))
    {
        $self->{isTheme} = 1 ;
        return ;
    }

    # Detection of book page count
    if (($self->{isPage} == 0) && ($tagname eq 'varkdata'))
    {
        $self->{isPage} = 1 ;
        return ;
    }

    # Detection of authors
    if ($tagname eq 'varkauthor')
    {
        $self->{isAuthor} = 1;
        return ;
    }

    # Capture of image
    if (($tagname eq 'img') && ($attr->{class} eq 'a-dynamic-
image image-stretch-vertical frontImage'))
    {
        $attr->{src} =~ /http.+\.jpg/ ;
        $self->{curInfo}->{cover} = $attr->{src};
        $self->{isImage} = 0 ;
        return ;
    }

    # Detection of book description
    if (($self->{isDescription} == 0) && ($tagname eq
'varkdescription'))
    {
        $self->{isDescription} = 1 ;
        return ;
    }
}
```

```
    if (($self->{isDescription} == 1) && ($tagname eq 'div'))
    {
        $self->{isDescription} = 2 ;
        return ;
    }

    # Detection title
    if (($self->{isTitle} == 0) && ($tagname eq 'varktitle'))
    {
        $self->{isTitle} = 2 ;
        return ;
    }
}

sub end
{
    my ($self, $tagname) = @_ ;

    $self->{inside}->{$tagname}--;

    if ($self->{parsingList})
    {
        # Identify end of comments
        if (($self->{isComment} == 1) && ($tagname eq
'varkcomment'))
        {
            $self->{isComment} = 0 ;
        }
    }

    else
    {
        # Finishing themes analysis
        if (($self->{isTheme} != 0) && ($tagname eq 'li'))
        {
            $self->{isTheme} = 0 ;
            return ;
        }

        # Finishing description analysis
        if (($self->{isDescription} != 0) && ($tagname eq 'div'))
        {
            $self->{isDescription} = 0 ;
            return ;
        }
    }
}

sub text
{
```

```
my ($self, $origtext) = @_;  
  
if ($self->{parsingList})  
{  
    # Remove blanks before and after string  
    $origtext =~ s/^\s+//;  
    $origtext =~ s/\s+$//g;  
  
    # Capture of book title  
    if (($self->{isComment} == 0) && ($self->{isTitle} == 1)  
&& ($origtext ne ''))  
    {  
        $self->{itemsList}[$self->{itemIdx}]->{title} =  
$origtext;  
        $self->{isTitle} = 0 ;  
        $self->{isPublication} = 1 ;  
        return ;  
    }  
  
    # Capture of book publication date  
    if (($self->{isComment} == 0) && ($self->{isPublication}  
== 1) && ($origtext ne ''))  
    {  
$self->{itemsList}[$self->{itemIdx}]->{publication} = $origtext;  
        $self->{isAuthor} = 1 ;  
        $self->{isPublication} = 0 ;  
        return ;  
    }  
  
    # Avoid a text area before the first author  
    if (($self->{isComment} == 0) && ($self->{isAuthor} == 1)  
&& ($origtext ne ''))  
    {  
        $self->{isAuthor} = 2 ;  
        return ;  
    }  
  
    # Capture of authors  
    if (($self->{isComment} == 0) && ($self->{isAuthor} == 2)  
&& ($origtext ne ''))  
    {  
        if  
($self->{itemsList}[$self->{itemIdx}]->{authors} eq '')  
        {  
$self->{itemsList}[$self->{itemIdx}]->{authors} = $origtext;  
        }  
        else  
        {  
            $self->{itemsList}[$self->{itemIdx}]->{authors} .= " "  

```



```
. $origtext;
}
return;
}

else
{
# Remove blanks before and after string
$origtext =~ s/^\s+//;
$origtext =~ s/\s+$//g;

# Capture of title
if (($self->{isTitle} == 2) && ($origtext ne ''))
{
    $self->{isTitle} = 0 ;
    $self->{curInfo}->{title} = $origtext;
    return ;
}

# Capture of page number
if (($self->{isPage} == 1) && ($origtext =~ /^[0-9]+/))
{
    $self->{curInfo}->{pages} = $origtext;
    $self->{isPage} = 0 ;
    return ;
}

# Capture of editor and publication date
if (($self->{isEditor} == 0) && ($origtext eq
$self->getTranslation(1)))
{
    $self->{isEditor} = 1 ;
    return ;
}
if (($self->{isEditor} == 1) && ($origtext ne ''))
{
    my @array = split('\(', $origtext);
    $array[1] =~ s/\)//g;
    $array[0] =~ s/^\s+//;
    $array[0] =~ s/\s+$//g;
    $array[1] =~ s/^\s+//;
    $array[1] =~ s/\s+$//g;
    $self->{curInfo}->{publisher} = $array[0];
    $self->{curInfo}->{publication} = $array[1];
    $self->{isEditor} = 0 ;
    return ;
}

# Capture of language
```

```
        if (($self->{isLanguage} == 0) && ($origtext eq
$self->getTranslation(2)))
        {
            $self->{isLanguage} = 1 ;
            return ;
        }
        if (($self->{isLanguage} == 1) && ($origtext ne ''))
        {
            $self->{curInfo}->{language} = $origtext;
            $self->{isLanguage} = 0 ;
            return ;
        }

        # Capture of ISBN
        if (($self->{isISBN} == 0) && ($origtext eq
$self->getTranslation(3)))
        {
            $self->{isISBN} = 1 ;
            return ;
        }
        if (($self->{isISBN} == 1) && ($origtext ne ''))
        {
            $origtext =~ s|-||gi;
            $self->{curInfo}->{isbn} = $origtext;
            $self->{isISBN} = 0 ;
            return ;
        }

        # Capture of book dimensions
        if (($self->{isSize} == 0) && ($origtext eq
$self->getTranslation(4)))
        {
            $self->{isSize} = 1 ;
            return ;
        }
        if (($self->{isSize} == 1) && ($origtext ne ''))
        {
            $self->{curInfo}->{format} = $origtext;
            $self->{isSize} = 0 ;
            return ;
        }

        # Detection of themes
        if (($origtext eq '>') && ($self->{isTheme} == 1))
        {
            $self->{isTheme} = 2 ;
            return ;
        }
    }
```

```
# Capture of themes
if (($self->{isTheme} == 2) && ($origtext ne ''))
{
    if ($self->{curInfo}->{genre} eq '')
    {
        $self->{curInfo}->{genre} = $origtext;
    }
    else
    {
        $self->{curInfo}->{genre} .= ", " . $origtext;
    }
    $self->{isTheme} = 1 ;
    return;
}

# Capture of authors
if (($self->{isAuthor} == 1) && ($origtext ne '') &&
($origtext =~ /^(?:(!Ajax).)*$/))
{
    if ($self->{curInfo}->{authors} eq '')
    {
        $self->{curInfo}->{authors} = $origtext;
    }
    else
    {
        $self->{curInfo}->{authors} .= ", " . $origtext;
    }
    $self->{isAuthor} = 0 ;
    return;
}

# Capture of description
if (($self->{isDescription} == 2) && ($origtext ne ''))
{
    if ($self->{curInfo}->{description} eq '')
    {
        $self->{curInfo}->{description} = $origtext;
    }
    else
    {
        $self->{curInfo}->{description} .= $origtext;
    }
    return ;
}
}

sub new
{
```

```
my $proto = shift;
my $class = ref($proto) || $proto;
my $self = $class->SUPER::new();
bless ($self, $class);

$self->{hasField} = {
    title => 1,
    authors => 1,
    publication => 1,
    format => 0,
    edition => 0,
};

$self->{isComment} = 0;
$self->{isUrl} = 0;
$self->{isTitle} = 0;
$self->{isPublication} = 0;
$self->{isAuthor} = 0;
    $self->{isPage} = 0;
    $self->{isEditor} = 0;
    $self->{isISBN} = 0;
    $self->{isDescription} = 0;
$self->{isLanguage} = 0 ;
$self->{isTheme} = 0 ;

    return $self;
}

sub getItemUrl
{
my ($self, $url) = @_;
    return $url;
}

sub preProcess
{
    my ($self, $html) = @_;

if ($self->{parsingList})
{
    # Analysis of results must be disabled during comments
    $html =~ s|<!--|<varkcomment>|gi;
    $html =~ s|-->|</varkcomment>|gi;
    # Remove other commercial offers
    $html =~ s|END SPONSORED LINKS SCRIPT.*||s;
    # End of authors listing detection
    $html =~ s|<h3 class="a-size-small a-color-null s-inline
a-text-normal">|<varkendauthors>|gi;
    $html =~ s|<div class="a-row a-spacing-
mini">|<varkendauthors>|gi;
```

```

    }
    else
    {
        # Beginning of book data : pages, editor, publication
        date, ISBN, dimensions
        $html =~ s|<td class="bucket">|<varkdata>|gi;
        # Beginning of book image
        $html =~ s|<div class="a-column a-span3 a-spacing-micro
imageThumb thumb">|<varkimage>|;
        # Beginning and end of book description
        $html =~ s|<script id="bookDesc_override_CSS"
type="text/undefined">|<varkdescription>|;
        #$html =~ s|<div id="bookDesc_outer_postBodyPS"
style="overflow: hidden; z-index: 1; height: 0px; display:
block;">|</varkdescription>|;
        # Beginning of book title
        $html =~ s|<div id="booksTitle" class="feature" data-
feature-name="booksTitle">|<varkttitle>|gi;
        # Beginning of book themes
        $html =~ s|<ul class="zg_hrsr">|<varkgenre>|gi;
        # Beginning of authors
        $html =~ s|<span class="author notFaded" data-
width="">|<varkauthor>|gi;

        $html =~ s|<BR>| |gi;
        $html =~ s|<I>| |gi;
        $html =~ s|</I>| |gi;
        $html =~ s|\x{8C}|OE|gi;
        $html =~ s|\x{9C}|oe|gi;
        $html =~ s|&#146;|'| |gi;

    }

    return $html;
}

sub getSearchUrl
{
    my ($self, $word) = @_;
    return 'http://' . $self->baseWWamazonUrl .
'/s/ref=nb_sb_noss_1?url=search-alias=stripbooks&field-keywords='
. "$word";
}

sub baseWWamazonUrl
{
    return "www.amazon.com";
}

sub getName
{
    return "Amazon (US)";
}

```

```
}

sub getAuthor
{
    return 'Varkolak';
}

sub getLang
{
    return 'EN';
}

sub getCharset
{
    my $self = shift;
    return "ISO-8859-15";
}

sub getSearchFieldsArray
{
    return ['title', 'authors', 'isbn'];
}

# Used to get the local translation of editor, language, ISBN,
product dimension, series
sub getTranslation
{
    my $param = $_[1];

    if ($param == 1)
    {
        return 'Publisher:';
    }
    elsif ($param == 2)
    {
        return 'Language:';
    }
    elsif ($param == 3)
    {
        return 'ISBN-13:';
    }
    elsif ($param == 4)
    {
        return 'Product Dimensions:';
    }
    elsif ($param == 5)
    {
        return 'Series:';
    }
}
```

```
}
1;
```

GCAmazonFR

```
package GCPlugins::GCbooks::GCAmazonFR;

#####
#
# Copyright 2005-2009 Tian
#
# This file is part of GCstar.
#
# GCstar is free software; you can redistribute it and/or modify
# it under the terms of the GNU General Public License as published by
# the Free Software Foundation; either version 2 of the License, or
# (at your option) any later version.
#
# GCstar is distributed in the hope that it will be useful,
# but WITHOUT ANY WARRANTY; without even the implied warranty of
# MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
# GNU General Public License for more details.
#
# You should have received a copy of the GNU General Public License
# along with GCstar; if not, write to the Free Software
# Foundation, Inc., 51 Franklin St, Fifth Floor, Boston, MA 02110-1301, USA
#
#####

use strict;
use utf8;

use GCPlugins::GCbooks::GCAmazon;

{
    package GCPlugins::GCbooks::GCPluginAmazonFR;

    use base qw(GCPlugins::GCbooks::GCPluginAmazon);

    sub baseWWWamazonUrl
    {
        return "www.amazon.fr";
    }

    sub getName
    {
        return "Amazon (FR)";
    }
}
```

```
sub getLang
{
    return 'FR';
}

sub getTranslation
{
    my $param = $_[1];

    if ($param == 1)
    {
        return 'Editeur :';
    }
    elsif ($param == 2)
    {
        return 'Langue :';
    }
    elsif ($param == 3)
    {
        return 'ISBN-13: ';
    }
    elsif ($param == 4)
    {
        return 'Dimensions du produit: ';
    }
    elsif ($param == 5)
    {
        return 'Collection :';
    }
}

1;
```

GCAmazonFR.pm

GCAmazonFR.pm

```
package GCPlugins::GCfilms::GCAmazonFR;

#####
#
# Copyright 2005-2010 Christian Jodar
# Copyright 2015-2016 K  renoc (kerenoc01 on Google mail)
#
# This file is part of GCstar.
```



```
#
# GCstar is free software; you can redistribute it and/or modify
# it under the terms of the GNU General Public License as
# published by
# the Free Software Foundation; either version 2 of the License,
# or
# (at your option) any later version.
#
# GCstar is distributed in the hope that it will be useful,
# but WITHOUT ANY WARRANTY; without even the implied warranty of
# MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
# GNU General Public License for more details.
#
# You should have received a copy of the GNU General Public
# License
# along with GCstar; if not, write to the Free Software
# Foundation, Inc., 51 Franklin St, Fifth Floor, Boston, MA
# 02110-1301, USA
#
#####

use strict;

use GCPlugins::GCfilms::GCfilmsAmazonCommon;

{
    package GCPlugins::GCfilms::GCPluginAmazonFR;

    use base qw(GCPlugins::GCfilms::GCfilmsAmazonPluginsBase);

    sub start
    {
        my ($self, $tagname, $attr, $attrseq, $origtext) = @_;

        $self->{inside}->{$tagname}++;

        if ($self->{parsingEnded})
        {
            if ($self->{itemIdx} < 0)
            {
                $self->{itemIdx} = 0;
                $self->{itemsList}[0]->{url} = $self->{loadedUrl};
            }
            return;
        }

        if ($self->{parsingList})
        {
            if ($tagname eq 'input')
            {
                $self->{beginParsing} = 1
            }
        }
    }
}
```

```
        if $attr->{src} =~ /go-button-search/;
    }
    return if ! $self->{beginParsing};
    if ($tagname eq 'div' && $attr->{class} eq "s-item-
container")
    {
        $self->{isTitle} = 1;
    }
    elsif ($tagname eq 'publication')
    {
        $self->{isPublication} = 1;
    }
    elsif ($tagname eq 'actors')
    {
        $self->{isActors} = 1;
    }
    if ($tagname eq 'a' && $self->{isTitle})
    {
        my $urlId;
        if ($urlId = $self->isItemUrl($attr->{href}))
        {
            $self->{isTitle} = 2 if $self->{isTitle} eq
'1';

            return if $self->{alreadyRetrieved}->{$urlId};
            $self->{alreadyRetrieved}->{$urlId} = 1;
            $self->{currentRetrieved} = $urlId;
            my $url = $attr->{href};
            $self->{itemIdx}++;
            $self->{itemsList}[$self->{itemIdx}]->{url} =
$url;
        }
    }
}
else
{
    if (($tagname eq "span") && ($attr->{id} eq
"productTitle"))
    {
        $self->{isTitle} = 1;
    }
    elsif (($tagname eq "img") &&
(!$self->{curInfo}->{image}))
    {
        $self->{curInfo}->{image} =
$self->extractImage($attr);
    }
    elsif (($tagname eq 'div') && ($attr->{class} eq
'content'))
    {
        $self->{insideContent} = 1;
    }
}
```

```

    }
    elsif (($tagname eq 'h3'))
    {
        $self->{insideSynopsis} = 1
        if (!$self->{curInfo}->{synopsis});
    }
    elsif (($tagname eq "span") && ($self->{insideAge}) &&
($attr->{class} =~ /medSprite/))
    {
        $attr->{class} =~ s/\s*$//;
        $self->{curInfo}->{age} = 2 if ($attr->{class} =~
m/G$/);
        $self->{curInfo}->{age} = 5 if ($attr->{class} =~
m/PG$/);
        $self->{curInfo}->{age} = 13 if ($attr->{class} =~
m/PG13$/);
        $self->{curInfo}->{age} = 18 if ($attr->{class} =~
m/R$/);
        $self->{insideAge} = 0;
    }
    elsif ($tagname eq "span")
    {
        $self->{insideNameAndDate} = 1 if $attr->{id} eq
"btAsinTitle";
    }
}

sub end
{
    my ($self, $tagname) = @_;

    $self->{inside}->{$tagname}--;
    if ($tagname eq "li")
    {
        $self->{insideActors} = 0;
        $self->{insideDirector} = 0;
    }
}

sub text
{
    my ($self, $origtext) = @_;

    return if length($origtext) < 2;

    if ($self->{parsingList})
    {
        return if ! $self->{beginParsing};
        if (($self->{inside}->{title})
&& ($origtext !~ /^Amazon.fr/))

```

```
        {
            $self->{parsingEnded} = 1;
        }
        if ($origtext =~ m/Distribution:/)
        {
            $self->{isActors} = 1;
        }
        elsif ($self->{isTitle})
        {
            $self->{itemsList}[$self->{itemIdx}]->{title} =
$origtext;
            $self->{isTitle} = 0;
            $self->{isPublication} = 1;
            return;
        }
        elsif ($self->{isPublication})
        {
            $origtext =~ m/([0-9]{4})/;
            $self->{itemsList}[$self->{itemIdx}]->{date} = $1;
            $self->{isPublication} = 0;
            return;
        }
        elsif ($self->{isActors})
        {
            $origtext =~ s/^\s*//;
            $origtext =~ s/\s*$//;
            $self->{itemsList}[$self->{itemIdx}]->{actors} =
$origtext
                if !
$self->{itemsList}[$self->{itemIdx}]->{actors};
            $self->{isActors} = 0;
            return;
        }
    }
    else
    {
        $origtext =~ s/\s{2,}//g;

        if ($self->{isTitle})
        {
            $origtext =~ s/\[.*\]//;
            $self->{curInfo}->{title} = $origtext;
            $self->{isTitle} = 0;
        }
        elsif (($self->{insideActors}) && ($origtext !~ /^,/))
        {
            $origtext =~ s/^\s*//;
            $origtext =~ s/\s+,/,/;
            if ($self->{actorsCounter} <
$GCPlugins::GCfilms::GCfilmsCommon::MAX_ACTORS)
```

```

        {
            push @{$self->{curInfo}->{actors}},
[$origtext];
            $self->{actorsCounter}++;
        }
    }
    elsif (($self->{insideDirector}) && ($origtext !~
/^,/))
    {
        $origtext =~ s/^\s//;
        $origtext =~ s/,.\$//;
        $self->{curInfo}->{director} .= ", "
            if $self->{curInfo}->{director};
        $self->{curInfo}->{director} .= $origtext;
    }
    elsif ($self->{insideTime})
    {
        $origtext =~ s/^\s//;
        $origtext =~ s/\n//g;
        $origtext =~ s/minutes//;
        $self->{curInfo}->{time} = $origtext;
        $self->{insideTime} = 0;
    }
    elsif ($self->{insideDate})
    {
        $origtext =~ s/^\s//;
        $origtext =~ s/\n//g;
        $origtext =~ s/\-$//;
        $self->{curInfo}->{date} =
$self->decodeDate($origtext);
        $self->{insideDate} = 0;
    }
    elsif (($self->{insideSynopsis} eq 1) && ($origtext eq
'Amazon.fr'))
    {
        $self->{insideSynopsis} = 2;
    }
    elsif ($self->{insideSynopsis} eq 2)
    {
        $self->{curInfo}->{synopsis} .= $origtext;
        $self->{insideSynopsis} = 0;
    }
    elsif ($self->{insideAudio})
    {
        $origtext =~ s/^\s*//;
        $self->{curInfo}->{audio} = $origtext;
        $self->{insideAudio} = 0;
    }
    elsif ($self->{insideSubTitle})
    {

```

```
        $origtext =~ s/^\s*//;
        $self->{curInfo}->{subt} = $origtext;
        $self->{insideSubTitle} = 0;
    }
    elsif ($self->{inside}->{b})
    {
        $self->{insideActors} = 1 if $origtext =~
/Acteurs\s*/;
        $self->{insideDirector} = 1 if $origtext =~
/R.alisateurs?\s*/;
        $self->{insideDate} = 1 if $origtext =~ /Date de
sortie/;
        $self->{insideTime} = 1 if $origtext =~
/Dur.e\s*/;
        $self->{insideAudio} = 1 if $origtext =~
/Audio\s*/;
        $self->{insideSubTitle} = 1 if $origtext =~ /Sous-
titres\s*/;
    }
}

sub new
{
    my $proto = shift;
    my $class = ref($proto) || $proto;
    my $self = $class->SUPER::new();
    bless ($self, $class);

    $self->{hasField} = {
        title => 1,
        date => 1,
        director => 0,
        actors => 1,
    };

    $self->{suffix} = 'fr';

    return $self;
}

sub preProcess
{
    my ($self, $html) = @_ ;

    $html = $self->SUPER::preProcess($html);
    if ($self->{parsingList})
    {
        $self->{isItem} = 0;
    }
}
```

```

        $html =~ s|~(.*)<span class="bindingBlock">\(<span
class="binding">(.*)</span>(<
.*?[0-9]{4})?\</span>|<actors>$1</actors><format>$2</format><publ
ication>$3</publication>|gsm;

    }
    else
    {
        $html =~ s/(<i>|<\i>)//gim;
        $html =~ s/<p>/\n/gim;
        $html =~ s|</p>|\n|gim;
        $html =~ s/(<ul>|<\ul>)/\n/gim;
        $html =~ s/<li>([^\<])/- $1/gim;
        $html =~ s|([^\>])</li>|$1\n|gim;
        $html =~ s|<br ?/?>|\n|gi;
        $html =~ s|<a href="/gp/imdb/[^"]*">(.*)</a>|$1|gm;
        $html =~ s/<a href="\exec\obidos\search-handle-
url\index=dvd-fr&field-
(?:actor|director|keywords)=[^\>]*">([^\<]*)<\a>/$1/gm;
    }

    $self->{parsingEnded} = 0;
    $self->{alreadyRetrieved} = {};
    $self->{beginParsing} = 1;

    return $html;
}

sub getName
{
    return "Amazon (FR)";
}

sub getLang
{
    return 'FR';
}

sub getAuthor
{
    return 'Tian - Kerenoc';
}

sub decodeDate
{
    my ($self, $date) = @_;

    # date déjà dans le bon format
    return $date if ($date =~ m|/|);

    # date à convertir au format jour/mois/année

```

```
my @dateItems = split(/\s/, $date);
my @listeMois =
("janvier","f.*vrier","mars","avril","mai","juin",
"juillet","ao.*t","septembre","octobre","novembre","décembre");
my $mois = 0;
my $nbDates = (scalar @dateItems);

while ($mois < (scalar @listeMois) && !($dateItems[$nbDates-2]
=~ m/$listeMois[$mois]/))
{
    $mois++;
}
$mois++;
return
sprintf("%02d/%02d",$dateItems[0],$mois)."/". $dateItems[$nbDates-1
] if ($nbDates > 2);
# si pas de jour, on prend le premier du mois
return sprintf("01/%02d",$mois)."/". $dateItems[1] if ($nbDates
eq 2);
return "";
}
}

1;
```

Voici donc la nouvelle version du plugin Amazon (qui devrait toujours fonctionner de la même manière avec les plugins de traduction), à copier sous le nom GCAmazon.pm dans /usr/share/gcstar/lib/GCPlugins/GCbooks/ :

[GCAmazon.pm](#)

[/usr/share/gcstar/lib/GCPlugins/GCbooks/GCAmazon.pm](#)

```
package GCPlugins::GCbooks::GCAmazon;

#####
#
# Copyright 2005-2009 Tian
#
# This file is part of GCstar.
#
# GCstar is free software; you can redistribute it and/or modify
# it under the terms of the GNU General Public License as
# published by
# the Free Software Foundation; either version 2 of the License,
# or
# (at your option) any later version.
#
# GCstar is distributed in the hope that it will be useful,
```



```

# but WITHOUT ANY WARRANTY; without even the implied warranty of
# MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
# GNU General Public License for more details.
#
# You should have received a copy of the GNU General Public
License
# along with GCstar; if not, write to the Free Software
# Foundation, Inc., 51 Franklin St, Fifth Floor, Boston, MA
02110-1301, USA
#
#####

use strict;
use utf8;

use GCPlugins::GCbooks::GCbooksCommon;

{
    package GCPlugins::GCbooks::GCPluginAmazon;

    use base qw(GCPlugins::GCbooks::GCbooksPluginsBase);
    use XML::Simple;
    use LWP::Simple qw($ua);
    use Encode;
    use HTML::Entities;
    use GCUtils;

    sub start
    {
        my ($self, $tagname, $attr, $attrseq, $origtext) = @_;

        $self->{inside}->{$tagname}++;

        if ($self->{parsingList})
        {
            # Identify beginning of comments
            if (($self->{isComment} == 0) && ($tagname eq
'varkcomment'))
            {
                $self->{isComment} = 1 ;
            }

            # Capture URL of book
            if (($self->{isComment} == 0) && ($self->{isUrl} == 1) &&
($tagname eq 'a'))
            {
                $self->{itemsList}[$self->{itemIdx}]->{url} =
$attr->{href};
                $self->{isUrl} = 0 ;
                $self->{isTitle} = 1 ;
                return;
            }
        }
    }
}

```

```
    }

    # Identify beginning of new book (next text is title)
    if (($self->{isComment} == 0) && ($tagname eq 'li') &&
($attr->{id} =~ /result_[0-9]+/ ))
    {
        # Create new entry
        $self->{itemIdx}++;
        $self->{isUrl} = 1 ;
        $self->{isAuthor} = 0 ;
        return ;
    }

    # Identify end of authors list
    if (($self->{isComment} == 0) && ($tagname eq
'varkendauthors') && ($self->{isAuthor} != 0))
    {
        $self->{isAuthor} = 0 ;
        return ;
    }
}
else
{
    # Detection of book themes
    if (($self->{isTheme} == 0) && ($tagname eq 'varkgenre'))
    {
        $self->{isTheme} = 1 ;
        return ;
    }

    # Detection of book page count
    if (($self->{isPage} == 0) && ($tagname eq 'varkdata'))
    {
        $self->{isPage} = 1 ;
        return ;
    }

    # Detection of authors
    if ($tagname eq 'varkauthor')
    {
        $self->{isAuthor} = 1;
        return ;
    }

    # Capture of image
    if ($tagname eq 'varkimage')
    {
        $attr->{adress} =~ /http.*?\./jpg/;
        $attr->{adress} =~ s|https://images-na.ssl-images-
amazon.com/images/I/|http://z2-ec2.images-amazon.com/images/I/|;
```

```
$self->{curInfo}->{cover} = $attr->{adress};
    return ;
}

# Detection of book description
if (($self->{isDescription} == 0) && ($tagname eq
'varkdescription'))
{
    $self->{isDescription} = 1 ;
    return ;
}
if (($self->{isDescription} == 1) && ($tagname eq 'div'))
{
    $self->{isDescription} = 2 ;
    return ;
}

# Detection title
if (($self->{isTitle} == 0) && ($tagname eq 'varkttitle'))
{
    $self->{isTitle} = 2 ;
    return ;
}
}

sub end
{
    my ($self, $tagname) = @_ ;

    $self->{inside}->{$tagname}--;

    if ($self->{parsingList})
    {
        # Identify end of comments
        if (($self->{isComment} == 1) && ($tagname eq
'varkcomment'))
        {
            $self->{isComment} = 0 ;
        }
    }

    else
    {
        # Finishing themes analysis
        if (($self->{isTheme} != 0) && ($tagname eq 'li'))
        {
            $self->{isTheme} = 0 ;
            return ;
        }
    }
}
```

```
# Finishing description analysis
if (($self->{isDescription} != 0) && ($tagname eq 'div'))
{
    $self->{isDescription} = 0 ;
    return ;
}

}

}

}

sub text
{
    my ($self, $origtext) = @_ ;

    if ($self->{parsingList})
    {
        # Remove blanks before and after string
        $origtext =~ s/^\s+//;
        $origtext =~ s/\s+$//g;

        # Capture of book title
        if (($self->{isComment} == 0) && ($self->{isTitle} == 1)
        && ($origtext ne ''))
        {
            $self->{itemsList}[$self->{itemIdx}]->{title} =
$origtext;
            $self->{isTitle} = 0 ;
            $self->{isPublication} = 1 ;
            return ;
        }

        # Capture of book publication date
        if (($self->{isComment} == 0) && ($self->{isPublication}
== 1) && ($origtext ne ''))
        {
            $self->{itemsList}[$self->{itemIdx}]->{publication} = $origtext;
            $self->{isAuthor} = 1 ;
            $self->{isPublication} = 0 ;
            return ;
        }

        # Avoid a text area before the first author
        if (($self->{isComment} == 0) && ($self->{isAuthor} == 1)
        && ($origtext ne ''))
        {
            $self->{isAuthor} = 2 ;
            return ;
        }

        # Capture of authors
```

```

        if (($self->{isComment} == 0) && ($self->{isAuthor} == 2)
        && ($origtext ne ''))
        {
            if
            ($self->{itemsList}[$self->{itemIdx}]->{authors} eq '')
            {
                $self->{itemsList}[$self->{itemIdx}]->{authors} = $origtext;
            }
            else
            {
                $self->{itemsList}[$self->{itemIdx}]->{authors} .= " "
                . $origtext;
            }
            return;
        }

    }
    else
    {
        # Remove blanks before and after string
        $origtext =~ s/^\s+//;
        $origtext =~ s/\s+$//g;

        # Capture of title
        if (($self->{isTitle} == 2) && ($origtext ne ''))
        {
            $self->{isTitle} = 0 ;
            $self->{curInfo}->{title} = $origtext;
            return ;
        }

        # Capture of page number
        if (($self->{isPage} == 1) && ($origtext =~ /^[0-9]+/))
        {
            $self->{curInfo}->{pages} = $origtext;
            $self->{isPage} = 0 ;
            return ;
        }

        # Capture of editor and publication date
        if (($self->{isEditor} == 0) && ($origtext eq
        $self->getTranslation(1)))
        {
            $self->{isEditor} = 1 ;
            return ;
        }
        if (($self->{isEditor} == 1) && ($origtext ne ''))
        {
            my @array = split('\(', $origtext);
            $array[1] =~ s/\)//g;

```

```
$array[0] =~ s/^\s+//;
$array[0] =~ s/\s+$//g;
$array[1] =~ s/^\s+//;
$array[1] =~ s/\s+$//g;
    $self->{curInfo}->{publisher} = $array[0];
$self->{curInfo}->{publication} = $array[1];
    $self->{isEditor} = 0 ;
    return ;
}

# Capture of language
if (($self->{isLanguage} == 0) && ($origtext eq
$self->getTranslation(2)))
{
    $self->{isLanguage} = 1 ;
    return ;
}
if (($self->{isLanguage} == 1) && ($origtext ne ''))
{
    $self->{curInfo}->{language} = $origtext;
    $self->{isLanguage} = 0 ;
    return ;
}

# Capture of ISBN
if (($self->{isISBN} == 0) && ($origtext eq
$self->getTranslation(3)))
{
    $self->{isISBN} = 1 ;
    return ;
}
if (($self->{isISBN} == 1) && ($origtext ne ''))
{
    $origtext =~ s|-||gi;
    $self->{curInfo}->{isbn} = $origtext;
    $self->{isISBN} = 0 ;
    return ;
}

# Capture of book dimensions
if (($self->{isSize} == 0) && ($origtext eq
$self->getTranslation(4)))
{
    $self->{isSize} = 1 ;
    return ;
}
if (($self->{isSize} == 1) && ($origtext ne ''))
{
    $self->{curInfo}->{format} = $origtext;
    $self->{isSize} = 0 ;
```

```
        return ;
    }

    # Detection of themes
    if (($origtext eq '>') && ($self->{isTheme} == 1))
    {
        $self->{isTheme} = 2 ;
        return ;
    }

    # Capture of themes
    if (($self->{isTheme} == 2) && ($origtext ne ''))
    {
        if ($self->{curInfo}->{genre} eq '')
        {
            $self->{curInfo}->{genre} = $origtext;
        }
        else
        {
            $self->{curInfo}->{genre} .= ", " . $origtext;
        }
        $self->{isTheme} = 1 ;
        return;
    }

    # Capture of authors
    if (($self->{isAuthor} == 1) && ($origtext ne '') &&
($origtext =~ /^(?:(!Ajax).)*$/))
    {
        if ($self->{curInfo}->{authors} eq '')
        {
            $self->{curInfo}->{authors} = $origtext;
        }
        else
        {
            $self->{curInfo}->{authors} .= ", " . $origtext;
        }
        $self->{isAuthor} = 0 ;
        return;
    }

    # Capture of description
    if (($self->{isDescription} == 2) && ($origtext ne ''))
    {
        if ($self->{curInfo}->{description} eq '')
        {
            $self->{curInfo}->{description} = $origtext;
        }
        else
        {

```

```
        $self->{curInfo}->{description} .= $origtext;
    }

    return ;
}
}
```

sub new

```
{
    my $proto = shift;
    my $class = ref($proto) || $proto;
    my $self = $class->SUPER::new();
    bless ($self, $class);
```

```
    $self->{hasField} = {
        title => 1,
        authors => 1,
        publication => 1,
        format => 0,
        edition => 0,
    };

```

```
    $self->{isComment} = 0;
    $self->{isUrl} = 0;
    $self->{isTitle} = 0;
    $self->{isPublication} = 0;
    $self->{isAuthor} = 0;
    $self->{isPage} = 0;
    $self->{isEditor} = 0;
    $self->{isISBN} = 0;
    $self->{isDescription} = 0;
    $self->{isLanguage} = 0 ;
    $self->{isTheme} = 0 ;

```

```
    return $self;
}
```

sub getItemUrl

```
{
    my ($self, $url) = @_;
    return $url;
}
```

sub preProcess

```
{
    my ($self, $html) = @_;

    if ($self->{parsingList})
```



```

    {
        # Analysis of results must be disabled during comments
        $html =~ s|<!--|<varkcomment>|gi;
        $html =~ s|-->|</varkcomment>|gi;
        # Remove other commercial offers
        $html =~ s|END SPONSORED LINKS SCRIPT.*||s;
        # End of authors listing detection
        $html =~ s|<h3 class="a-size-small a-color-null s-inline
a-text-normal">|<varkendauthors>|gi;
        $html =~ s|<div class="a-row a-spacing-
mini">|<varkendauthors>|gi;
    }
    else
    {
        # Beginning of book data : pages, editor, publication
date, ISBN, dimensions
        $html =~ s|<td class="bucket">|<varkdata>|gi;
        # Beginning and end of book description
        $html =~ s|<script id="bookDesc_override_CSS"
type="text/undefined">|<varkdescription>|;
        #$html =~ s|<div id="bookDesc_outer_postBodyPS"
style="overflow: hidden; z-index: 1; height: 0px; display:
block;">|</varkdescription>|;
        # Beginning of book title
        $html =~ s|<div id="booksTitle" class="feature" data-
feature-name="booksTitle">|<varkttitle>|gi;
        # Beginning of book themes
        $html =~ s|<ul class="zg_hrsr">|<varkgenre>|gi;
        # Beginning of authors
        $html =~ s|<span class="author notFaded" data-
width="">|<varkauthor>|gi;
        # Beginning of image
        $html =~ s|class="a-dynamic-image image-stretch-
vertical frontImage" id="imgBlkFront" data-a-dynamic-
image="{&quot;|><varkimage adress="|;

        $html =~ s|<BR>||gi;
        $html =~ s|<I>||gi;
        $html =~ s|</I>||gi;
        $html =~ s|\x{8C}|0E|gi;
        $html =~ s|\x{9C}|oe|gi;
        $html =~ s|&#146;|'|gi;

    }

    return $html;
}

sub getSearchUrl
{
    my ($self, $word) = @_;
    return 'http://' . $self->baseWWWamazonUrl .

```

```
'/s/ref=nb_sb_noss_1?url=search-alias=stripbooks&field-keywords='
. "$word";
}

sub baseWWWamazonUrl
{
    return "www.amazon.com";
}

sub getName
{
    return "Amazon (US)";
}

sub getAuthor
{
    return 'Varkolak';
}

sub getLang
{
    return 'EN';
}

sub getCharset
{
    my $self = shift;
    return "ISO-8859-15";
}

sub getSearchFieldsArray
{
    return ['title', 'authors', 'isbn'];
}

# Used to get the local translation of editor, language, ISBN,
product dimension, series
sub getTranslation
{
    my $param = $_[1];

    if ($param == 1)
    {
        return 'Publisher:';
    }
    elsif ($param == 2)
    {
        return 'Language:';
    }
    elsif ($param == 3)
```

```

    {
        return 'ISBN-13: ';
    }
    elseif ($param == 4)
    {
        return 'Product Dimensions: ';
    }
    elseif ($param == 5)
    {
        return 'Series: ';
    }
}
1;

```

voici une version améliorée du plugin GCAmazon.pm pour les livres, qui corrige deux petits détails gênants:

- la présence d'un point-virgule apparaissant parfois après l'éditeur - le fait que certains noms d'auteurs sur le site sont rédigés en majuscules

[GCAmazon.pm](#)

[GCAmazon.pm](#)

```

package GCPlugins::GCbooks::GCAmazon;

#####
#
# Copyright 2005-2009 Tian
#
# This file is part of GCstar.
#
# GCstar is free software; you can redistribute it and/or modify
# it under the terms of the GNU General Public License as
# published by
# the Free Software Foundation; either version 2 of the License,
# or
# (at your option) any later version.
#
# GCstar is distributed in the hope that it will be useful,
# but WITHOUT ANY WARRANTY; without even the implied warranty of
# MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
# GNU General Public License for more details.
#
# You should have received a copy of the GNU General Public
# License
# along with GCstar; if not, write to the Free Software

```

```
# Foundation, Inc., 51 Franklin St, Fifth Floor, Boston, MA
02110-1301, USA
#
#####

use strict;
use utf8;

use GCPlugins::GCbooks::GCbooksCommon;

{
    package GCPlugins::GCbooks::GCPluginAmazon;

    use base qw(GCPlugins::GCbooks::GCbooksPluginsBase);
    use XML::Simple;
    use LWP::Simple qw($ua);
    use Encode;
    use HTML::Entities;
    use GCUtils;

    sub start
    {
        my ($self, $tagname, $attr, $attrseq, $origtext) = @_;

        $self->{inside}->{$tagname}++;

        if ($self->{parsingList})
        {
            # Identify beginning of comments
            if (($self->{isComment} == 0) && ($tagname eq
'varkcomment'))
            {
                $self->{isComment} = 1 ;
            }

            # Capture URL of book
            if (($self->{isComment} == 0) && ($self->{isUrl} == 1) &&
($tagname eq 'a'))
            {
                $self->{itemsList}[$self->{itemIdx}]->{url} =
$attr->{href};
                $self->{isUrl} = 0 ;
                $self->{isTitle} = 1 ;
                return;
            }

            # Identify beginning of new book (next text is title)
            if (($self->{isComment} == 0) && ($tagname eq 'li') &&
($attr->{id} =~ /result_[0-9]+/ ))
            {

```

```
# Create new entry
    $self->{itemIdx}++;
    $self->{isUrl} = 1 ;
    $self->{isAuthor} = 0 ;
    return ;
}

# Identify end of authors list
if (($self->{isComment} == 0) && ($tagname eq
'varkendauthors') && ($self->{isAuthor} != 0))
{
    $self->{isAuthor} = 0 ;
    return ;
}
}
else
{
    # Detection of book themes
    if (($self->{isTheme} == 0) && ($tagname eq 'varkgenre'))
    {
        $self->{isTheme} = 1 ;
        return ;
    }

    # Detection of book page count
    if (($self->{isPage} == 0) && ($tagname eq 'varkdata'))
    {
        $self->{isPage} = 1 ;
        return ;
    }

    # Detection of authors
    if ($tagname eq 'varkauthor')
    {
        $self->{isAuthor} = 1;
        return ;
    }

    # Capture of image
    if ($tagname eq 'varkimage')
    {
        $attr->{adress} =~ /http.*?\./jpg/;
        $attr->{adress} =~ s|https://images-na.ssl-images-
amazon.com/images/I/|http://z2-ec2.images-amazon.com/images/I/|;
        $self->{curInfo}->{cover} = $attr->{adress};
        return ;
    }

    # Detection of book description
    if (($self->{isDescription} == 0) && ($tagname eq
'varkdescription'))
```

```
{
    $self->{isDescription} = 1 ;
    return ;
}
if (($self->{isDescription} == 1) && ($tagname eq 'div'))
{
    $self->{isDescription} = 2 ;
    return ;
}

# Detection title
if (($self->{isTitle} == 0) && ($tagname eq 'varktitle'))
{
    $self->{isTitle} = 2 ;
    return ;
}
}

sub end
{
    my ($self, $tagname) = @_ ;

    $self->{inside}->{$tagname}--;

    if ($self->{parsingList})
    {
        # Identify end of comments
        if (($self->{isComment} == 1) && ($tagname eq
'varkcomment'))
        {
            $self->{isComment} = 0 ;
        }
    }

    else
    {
        # Finishing themes analysis
        if (($self->{isTheme} != 0) && ($tagname eq 'li'))
        {
            $self->{isTheme} = 0 ;
            return ;
        }

        # Finishing description analysis
        if (($self->{isDescription} != 0) && ($tagname eq 'div'))
        {
            $self->{isDescription} = 0 ;
            return ;
        }
    }
}
```

```
    }  
}  
  
sub text  
{  
    my ($self, $origtext) = @_;  
  
    if ($self->{parsingList})  
    {  
        # Remove blanks before and after string  
        $origtext =~ s/^\s+//;  
        $origtext =~ s/\s+$//g;  
  
        # Capture of book title  
        if (($self->{isComment} == 0) && ($self->{isTitle} == 1)  
&& ($origtext ne ''))  
        {  
            $self->{itemsList}[$self->{itemIdx}]->{title} =  
$origtext;  
            $self->{isTitle} = 0 ;  
            $self->{isPublication} = 1 ;  
            return ;  
        }  
  
        # Capture of book publication date  
        if (($self->{isComment} == 0) && ($self->{isPublication}  
== 1) && ($origtext ne ''))  
        {  
            $self->{itemsList}[$self->{itemIdx}]->{publication} = $origtext;  
            $self->{isAuthor} = 1 ;  
            $self->{isPublication} = 0 ;  
            return ;  
        }  
  
        # Avoid a text area before the first author  
        if (($self->{isComment} == 0) && ($self->{isAuthor} == 1)  
&& ($origtext ne ''))  
        {  
            $self->{isAuthor} = 2 ;  
            return ;  
        }  
  
        # Capture of authors  
        if (($self->{isComment} == 0) && ($self->{isAuthor} == 2)  
&& ($origtext ne ''))  
        {  
            if  
($self->{itemsList}[$self->{itemIdx}]->{authors} eq '')  
            {  
                $self->{itemsList}[$self->{itemIdx}]->{authors} = $origtext;  
            }  
        }  
    }  
}
```

```
    }
    else
    {
        $self->{itemsList}[$self->{itemIdx}]->{authors} .= " "
. $origtext;
    }
    return;
}

}
else
{
    # Remove blanks before and after string
    $origtext =~ s/^\s+//;
    $origtext =~ s/\s+$//g;

    # Capture of title
    if (($self->{isTitle} == 2) && ($origtext ne ''))
    {
        $self->{isTitle} = 0 ;
        $self->{curInfo}->{title} = $origtext;
        return ;
    }

    # Capture of page number
    if (($self->{isPage} == 1) && ($origtext =~ /^[0-9]+/))
    {
        $self->{curInfo}->{pages} = $origtext;
        $self->{isPage} = 0 ;
        return ;
    }

    # Capture of editor and publication date
    if (($self->{isEditor} == 0) && ($origtext eq
$self->getTranslation(1)))
    {
        $self->{isEditor} = 1 ;
        return ;
    }
    if (($self->{isEditor} == 1) && ($origtext ne ''))
    {
        my @array = split('\(', $origtext);
        $array[1] =~ s/\)//g;
        $array[0] =~ s/^\s+//;
        $array[0] =~ s/\s+$//g;
        $array[0] =~ s/\\;/g;
        $array[1] =~ s/^\s+//;
        $array[1] =~ s/\s+$//g;
        $self->{curInfo}->{publisher} = $array[0];
    }
}
```



```
$self->{curInfo}->{publication} = $array[1];
    $self->{isEditor} = 0 ;
    return ;
}

# Capture of language
if (($self->{isLanguage} == 0) && ($origtext eq
$self->getTranslation(2)))
{
    $self->{isLanguage} = 1 ;
    return ;
}
if (($self->{isLanguage} == 1) && ($origtext ne ''))
{
    $self->{curInfo}->{language} = $origtext;
    $self->{isLanguage} = 0 ;
    return ;
}

# Capture of ISBN
if (($self->{isISBN} == 0) && ($origtext eq
$self->getTranslation(3)))
{
    $self->{isISBN} = 1 ;
    return ;
}
if (($self->{isISBN} == 1) && ($origtext ne ''))
{
    $origtext =~ s|-||gi;
    $self->{curInfo}->{isbn} = $origtext;
    $self->{isISBN} = 0 ;
    return ;
}

# Capture of book dimensions
if (($self->{isSize} == 0) && ($origtext eq
$self->getTranslation(4)))
{
    $self->{isSize} = 1 ;
    return ;
}
if (($self->{isSize} == 1) && ($origtext ne ''))
{
    $self->{curInfo}->{format} = $origtext;
    $self->{isSize} = 0 ;
    return ;
}

# Detection of themes
if (($origtext eq '>') && ($self->{isTheme} == 1))
```

```
{
    $self->{isTheme} = 2 ;
    return ;
}

# Capture of themes
if (($self->{isTheme} == 2) && ($origtext ne ''))
{
    if ($self->{curInfo}->{genre} eq '')
    {
        $self->{curInfo}->{genre} = $origtext;
    }
else
{
    $self->{curInfo}->{genre} .= ", " . $origtext;
}
$self->{isTheme} = 1 ;
return;
}

# Capture of authors
if (($self->{isAuthor} == 1) && ($origtext ne '') &&
($origtext =~ /^(?:(!Ajax).)*$/))
{
    # Lower case for author names, except for first letters
    $origtext =~ s/([[:alpha:]]+)/ucfirst(lc $1)/egi;
    if ($self->{curInfo}->{authors} eq '')
    {
        $self->{curInfo}->{authors} = $origtext;
    }
else
{
    $self->{curInfo}->{authors} .= ", " . $origtext;
}
$self->{isAuthor} = 0 ;
return;
}

# Capture of description
if (($self->{isDescription} == 2) && ($origtext ne ''))
{
    if ($self->{curInfo}->{description} eq '')
    {
        $self->{curInfo}->{description} = $origtext;
    }
else
{
    $self->{curInfo}->{description} .= $origtext;
}

    return ;
}
```

```
    }
    }
}

sub new
{
    my $proto = shift;
    my $class = ref($proto) || $proto;
    my $self = $class->SUPER::new();
    bless ($self, $class);

    $self->{hasField} = {
        title => 1,
        authors => 1,
        publication => 1,
        format => 0,
        edition => 0,
    };

    $self->{isComment} = 0;
    $self->{isUrl} = 0;
    $self->{isTitle} = 0;
    $self->{isPublication} = 0;
    $self->{isAuthor} = 0;
    $self->{isPage} = 0;
    $self->{isEditor} = 0;
    $self->{isISBN} = 0;
    $self->{isDescription} = 0;
    $self->{isLanguage} = 0 ;
    $self->{isTheme} = 0 ;

    return $self;
}

sub getItemUrl
{
    my ($self, $url) = @_;
    return $url;
}

sub preProcess
{
    my ($self, $html) = @_;

    if ($self->{parsingList})
    {
        # Analysis of results must be disabled during comments
        $html =~ s|<!--|<varkcomment>|gi;
        $html =~ s|-->|</varkcomment>|gi;
    }
}
```

```
# Remove other commercial offers
$html =~ s|END SPONSORED LINKS SCRIPT.*||s;
# End of authors listing detection
$html =~ s|<h3 class="a-size-small a-color-null s-inline
a-text-normal">|<varkendauthors>|gi;
$html =~ s|<div class="a-row a-spacing-
mini">|<varkendauthors>|gi;
}
else
{
# Beginning of book data : pages, editor, publication
date, ISBN, dimensions
$html =~ s|<td class="bucket">|<varkdata>|gi;
# Beginning and end of book description
$html =~ s|<script id="bookDesc_override_CSS"
type="text/undefined">|<varkdescription>|;
#$html =~ s|<div id="bookDesc_outer_postBodyPS"
style="overflow: hidden; z-index: 1; height: 0px; display:
block;">|</varkdescription>|;
# Beginning of book title
$html =~ s|<div id="booksTitle" class="feature" data-
feature-name="booksTitle">|<varktitle>|gi;
# Beginning of book themes
$html =~ s|<ul class="zg_hrsr">|<varkgenre>|gi;
# Beginning of authors
$html =~ s|<span class="author notFaded" data-
width="">|<varkauthor>|gi;
# Beginning of image
$html =~ s|class="a-dynamic-image image-stretch-
vertical frontImage" id="imgBlkFront" data-a-dynamic-
image="{&quot;|><varkimage adress="|;

$html =~ s|<BR>||gi;
$html =~ s|<I>||gi;
$html =~ s|</I>||gi;
$html =~ s|\x{8C}|OE|gi;
$html =~ s|\x{9C}|oe|gi;
$html =~ s|&#146;|'|gi;

}

return $html;
}

sub getSearchUrl
{
my ($self, $word) = @_;
return 'http://' . $self->baseWWWamazonUrl .
'/s/ref=nb_sb_noss_1?url=search-alias=stripbooks&field-keywords='
. "$word";
}
```

```
sub baseWwamazonUrl
{
    return "www.amazon.com";
}

sub getName
{
    return "Amazon (US)";
}

sub getAuthor
{
    return 'Varkolak';
}

sub getLang
{
    return 'EN';
}

sub getCharset
{
    my $self = shift;
    return "ISO-8859-15";
}

sub getSearchFieldsArray
{
    return ['title', 'authors', 'isbn'];
}

# Used to get the local translation of editor, language, ISBN,
product dimension, series
sub getTranslation
{
    my $param = $_[1];

    if ($param == 1)
    {
        return 'Publisher: ';
    }
    elsif ($param == 2)
    {
        return 'Language: ';
    }
    elsif ($param == 3)
    {
        return 'ISBN-13: ';
    }
    elsif ($param == 4)
```

```
{
    return 'Product Dimensions: ';
}
elseif ($param == 5)
{
    return 'Series: ';
}
}
1;
```

encore une petite modification du plugin GCAmazon.pm en réponse à une altération du site d'Amazon, qui devrait rendre la recherche par auteur ou titre plus propre :

[GCAmazon.pm](#)

[GCAmazon.pm](#)

```
package GCPlugins::GCbooks::GCAmazon;

#####
#
# Copyright 2005-2009 Tian
#
# This file is part of GCstar.
#
# GCstar is free software; you can redistribute it and/or modify
# it under the terms of the GNU General Public License as
# published by
# the Free Software Foundation; either version 2 of the License,
# or
# (at your option) any later version.
#
# GCstar is distributed in the hope that it will be useful,
# but WITHOUT ANY WARRANTY; without even the implied warranty of
# MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
# GNU General Public License for more details.
#
# You should have received a copy of the GNU General Public
# License
# along with GCstar; if not, write to the Free Software
# Foundation, Inc., 51 Franklin St, Fifth Floor, Boston, MA
# 02110-1301, USA
#
#####

use strict;
use utf8;
```

```
use GCPlugins::GCbooks::GCbooksCommon;

{
    package GCPlugins::GCbooks::GCPluginAmazon;

    use base qw(GCPlugins::GCbooks::GCbooksPluginsBase);
    use XML::Simple;
    use LWP::Simple qw($ua);
    use Encode;
    use HTML::Entities;
    use GCUtils;

    sub start
    {
        my ($self, $tagname, $attr, $attrseq, $origtext) = @_;

        $self->{inside}->{$tagname}++;

        if ($self->{parsingList})
        {
            # Identify beginning of comments
            if (($self->{isComment} == 0) && ($tagname eq
'varkcomment'))
            {
                $self->{isComment} = 1 ;
            }

            # Capture URL of book
            if (($self->{isComment} == 0) && ($self->{isUrl} == 1) &&
($tagname eq 'a'))
            {
                $self->{itemsList}[$self->{itemIdx}]->{url} =
$attr->{href};
                $self->{isUrl} = 0 ;
                $self->{isTitle} = 1 ;
                return;
            }

            # Identify beginning of new book (next text is title)
            if (($self->{isComment} == 0) && ($tagname eq 'li') &&
($attr->{id} =~ /result_[0-9]+/ ))
            {
                # Create new entry
                $self->{itemIdx}++;
                $self->{isUrl} = 1 ;
                $self->{isAuthor} = 0 ;
                return ;
            }

            # Identify end of authors list

```

```
if (($self->{isComment} == 0) && ($tagname eq
'varkendauthors') && ($self->{isAuthor} != 0))
{
    $self->{isAuthor} = 0 ;
    return ;
}
else
{
    # Detection of book themes
    if (($self->{isTheme} == 0) && ($tagname eq 'varkgenre'))
    {
        $self->{isTheme} = 1 ;
        return ;
    }

    # Detection of book page count
    if (($self->{isPage} == 0) && ($tagname eq 'varkdata'))
    {
        $self->{isPage} = 1 ;
        return ;
    }

    # Detection of authors
    if ($tagname eq 'varkauthor')
    {
        $self->{isAuthor} = 1;
        return ;
    }

    # Capture of image
    if ($tagname eq 'varkimage')
    {
        $attr->{adress} =~ /http.*?\./jpg/;
        $attr->{adress} =~ s|https://images-na.ssl-images-
amazon.com/images/I/|http://z2-ec2.images-amazon.com/images/I/|;
        $self->{curInfo}->{cover} = $attr->{adress};
        return ;
    }

    # Detection of book description
    if (($self->{isDescription} == 0) && ($tagname eq
'varkdescription'))
    {
        $self->{isDescription} = 1 ;
        return ;
    }
    if (($self->{isDescription} == 1) && ($tagname eq 'div'))
    {
        $self->{isDescription} = 2 ;
    }
}
```



```
        return ;
    }

    # Detection title
    if (($self->{isTitle} == 0) && ($tagname eq 'varktitle'))
    {
        $self->{isTitle} = 2 ;
        return ;
    }
}

sub end
{
    my ($self, $tagname) = @_ ;

    $self->{inside}->{$tagname}-- ;

    if ($self->{parsingList})
    {
        # Identify end of comments
        if (($self->{isComment} == 1) && ($tagname eq
'varkcomment'))
        {
            $self->{isComment} = 0 ;
        }
    }

    else
    {
        # Finishing themes analysis
        if (($self->{isTheme} != 0) && ($tagname eq 'li'))
        {
            $self->{isTheme} = 0 ;
            return ;
        }

        # Finishing description analysis
        if (($self->{isDescription} != 0) && ($tagname eq 'div'))
        {
            $self->{isDescription} = 0 ;
            return ;
        }
    }
}

sub text
{
    my ($self, $origtext) = @_ ;
```

```
if ($self->{parsingList})
{
    # Remove blanks before and after string
    $origtext =~ s/^\s+//;
    $origtext =~ s/\s+$//g;

    # Capture of book title
    if (($self->{isComment} == 0) && ($self->{isTitle} == 1)
    && ($origtext ne ''))
    {
        $self->{itemsList}[$self->{itemIdx}]->{title} =
$origtext;

        $self->{isTitle} = 0 ;
        $self->{isPublication} = 1 ;
        return ;
    }

    # Capture of book publication date
    if (($self->{isComment} == 0) && ($self->{isPublication}
== 1) && ($origtext ne ''))
    {
        $self->{itemsList}[$self->{itemIdx}]->{publication} = $origtext;
        $self->{isAuthor} = 1 ;
        $self->{isPublication} = 0 ;
        return ;
    }

    # Avoid a text area before the first author
    if (($self->{isComment} == 0) && ($self->{isAuthor} == 1)
    && ($origtext ne ''))
    {
        $self->{isAuthor} = 2 ;
        return ;
    }

    # Capture of authors
    if (($self->{isComment} == 0) && ($self->{isAuthor} == 2)
    && ($origtext ne ''))
    {
        if
($self->{itemsList}[$self->{itemIdx}]->{authors} eq '')
        {
            $self->{itemsList}[$self->{itemIdx}]->{authors} = $origtext;
        }
        else
        {
            $self->{itemsList}[$self->{itemIdx}]->{authors} .= " "
. $origtext;
        }
        return;
    }
}
```

```

    }

    }
    else
    {
        # Remove blanks before and after string
        $origtext =~ s/^\s+//;
        $origtext =~ s/\s+$//g;

        # Capture of title
        if (($self->{isTitle} == 2) && ($origtext ne ''))
        {
            $self->{isTitle} = 0 ;
            $self->{curInfo}->{title} = $origtext;
            return ;
        }

        # Capture of page number
        if (($self->{isPage} == 1) && ($origtext =~ /^[0-9]+/))
        {
            $self->{curInfo}->{pages} = $origtext;
            $self->{isPage} = 0 ;
            return ;
        }

        # Capture of editor and publication date
        if (($self->{isEditor} == 0) && ($origtext eq
$self->getTranslation(1)))
        {
            $self->{isEditor} = 1 ;
            return ;
        }
        if (($self->{isEditor} == 1) && ($origtext ne ''))
        {
            my @array = split('\(', $origtext);
            $array[1] =~ s/\)//g;
            $array[0] =~ s/^\s+//;
            $array[0] =~ s/\s+$//g;
            $array[0] =~ s/\\;/g;
            $array[1] =~ s/^\s+//;
            $array[1] =~ s/\s+$//g;
            $self->{curInfo}->{publisher} = $array[0];
            $self->{curInfo}->{publication} = $array[1];
            $self->{isEditor} = 0 ;
            return ;
        }

        # Capture of language
        if (($self->{isLanguage} == 0) && ($origtext eq
$self->getTranslation(2)))

```

```
{
    $self->{isLanguage} = 1 ;
    return ;
}
if (($self->{isLanguage} == 1) && ($origtext ne ''))
{
    $self->{curInfo}->{language} = $origtext;
    $self->{isLanguage} = 0 ;
    return ;
}

# Capture of ISBN
if (($self->{isISBN} == 0) && ($origtext eq
$self->getTranslation(3)))
{
    $self->{isISBN} = 1 ;
    return ;
}
if (($self->{isISBN} == 1) && ($origtext ne ''))
{
    $origtext =~ s|-||gi;
    $self->{curInfo}->{isbn} = $origtext;
    $self->{isISBN} = 0 ;
    return ;
}

# Capture of book dimensions
if (($self->{isSize} == 0) && ($origtext eq
$self->getTranslation(4)))
{
    $self->{isSize} = 1 ;
    return ;
}
if (($self->{isSize} == 1) && ($origtext ne ''))
{
    $self->{curInfo}->{format} = $origtext;
    $self->{isSize} = 0 ;
    return ;
}

# Detection of themes
if (($origtext eq '>') && ($self->{isTheme} == 1))
{
    $self->{isTheme} = 2 ;
    return ;
}

# Capture of themes
if (($self->{isTheme} == 2) && ($origtext ne ''))
```

```

        {
            if ($self->{curInfo}->{genre} eq '')
            {
                $self->{curInfo}->{genre} = $origtext;
            }
        }
    else
    {
        $self->{curInfo}->{genre} .= ", " . $origtext;
    }
    $self->{isTheme} = 1 ;
    return;
}

# Capture of authors
if (($self->{isAuthor} == 1) && ($origtext ne '')) &&
($origtext =~ /^(?:(!Ajax).)*$/))
{
    # Lower case for author names, except for first letters
    $origtext =~ s/([[:alpha:]]+)/ucfirst(lc $1)/egi;
    if ($self->{curInfo}->{authors} eq '')
    {
        $self->{curInfo}->{authors} = $origtext;
    }
}
else
{
    $self->{curInfo}->{authors} .= ", " . $origtext;
}
$self->{isAuthor} = 0 ;
return;
}

# Capture of description
if (($self->{isDescription} == 2) && ($origtext ne ''))
{
    if ($self->{curInfo}->{description} eq '')
    {
        $self->{curInfo}->{description} = $origtext;
    }
}
else
{
    $self->{curInfo}->{description} .= $origtext;
}
return ;
}
}

sub new
{

```

```
my $proto = shift;
my $class = ref($proto) || $proto;
my $self = $class->SUPER::new();
bless ($self, $class);

$self->{hasField} = {
    title => 1,
    authors => 1,
    publication => 1,
    format => 0,
    edition => 0,
};

$self->{isComment} = 0;
$self->{isUrl} = 0;
$self->{isTitle} = 0;
$self->{isPublication} = 0;
$self->{isAuthor} = 0;
    $self->{isPage} = 0;
    $self->{isEditor} = 0;
    $self->{isISBN} = 0;
    $self->{isDescription} = 0;
$self->{isLanguage} = 0 ;
$self->{isTheme} = 0 ;

    return $self;
}

sub getItemUrl
{
my ($self, $url) = @_;
    return $url;
}

sub preProcess
{
    my ($self, $html) = @_;

if ($self->{parsingList})
{
    # Analysis of results must be disabled during comments
    $html =~ s|<!--|<varkcomment>|gi;
    $html =~ s|-->|</varkcomment>|gi;
    # Remove other commercial offers
    $html =~ s|END SPONSORED LINKS SCRIPT.*||s;
    # End of authors listing detection
    $html =~ s|</span></div></div><div class="a-row"><div
class="a-column a-span7"><div class="a-row a-spacing-
none">|<varkendauthors>|gi;
    $html =~ s|<h3 class="a-size-small a-color-null s-inline
```

```

a-text-normal">|<varkendauthors>|gi;
    $html =~ s|<div class="a-row a-spacing-
mini">|<varkendauthors>|gi;
    }
    else
    {
        # Beginning of book data : pages, editor, publication
date, ISBN, dimensions
        $html =~ s|<td class="bucket">|<varkdata>|gi;
        # Beginning and end of book description
        $html =~ s|<script id="bookDesc_override_CSS"
type="text/undefined">|<varkdescription>|;
        #$html =~ s|<div id="bookDesc_outer_postBodyPS"
style="overflow: hidden; z-index: 1; height: 0px; display:
block;">|</varkdescription>|;
        # Beginning of book title
        $html =~ s|<div id="booksTitle" class="feature" data-
feature-name="booksTitle">|<varkttitle>|gi;
        # Beginning of book themes
        $html =~ s|<ul class="zg_hrsr">|<varkgenre>|gi;
        # Beginning of authors
        $html =~ s|<span class="author notFaded" data-
width="">|<varkauthor>|gi;
        # Beginning of image
        $html =~ s|class="a-dynamic-image image-stretch-
vertical frontImage" id="imgBlkFront" data-a-dynamic-
image="{&quot;}>|<varkimage adress=""|;

        $html =~ s|<BR>| |gi;
        $html =~ s|<I>| |gi;
        $html =~ s|</I>| |gi;
        $html =~ s|\x{8C}|OE|gi;
        $html =~ s|\x{9C}|oe|gi;
        $html =~ s|&#146;|'| |gi;

    }

    return $html;
}

sub getSearchUrl
{
    my ($self, $word) = @_;
    return 'http://' . $self->baseWWWamazonUrl .
'/s/ref=nb_sb_noss_1?url=search-alias=stripbooks&field-keywords='
. "$word";
}

sub baseWWWamazonUrl
{
    return "www.amazon.com";
}

```

```
sub getName
{
    return "Amazon (US)";
}

sub getAuthor
{
    return 'Varkolak';
}

sub getLang
{
    return 'EN';
}

sub getCharset
{
    my $self = shift;
    return "ISO-8859-15";
}

sub getSearchFieldsArray
{
    return ['title', 'authors', 'isbn'];
}

# Used to get the local translation of editor, language, ISBN,
product dimension, series
sub getTranslation
{
    my $param = $_[1];

    if ($param == 1)
    {
        return 'Publisher:';
    }
    elsif ($param == 2)
    {
        return 'Language:';
    }
    elsif ($param == 3)
    {
        return 'ISBN-13:';
    }
    elsif ($param == 4)
    {
        return 'Product Dimensions:';
    }
    elsif ($param == 5)
```



```

    {
        return 'Series: ';
    }
}

}
1;

```

Chapitre.com

GCChapitre.pm

/usr/share/gcstar/lib/GCPlugins/GCbooks/GCChapitre.pm

```

package GCPlugins::GCbooks::GCChapitre;

#####
#
# Copyright 2005-2006 Tian
# Copyright 2015-2016 Kereno01 on Google Mail
#
# This file is part of GCstar.
#
# GCstar is free software; you can redistribute it and/or modify
# it under the terms of the GNU General Public License as
# published by
# the Free Software Foundation; either version 2 of the License,
# or
# (at your option) any later version.
#
# GCstar is distributed in the hope that it will be useful,
# but WITHOUT ANY WARRANTY; without even the implied warranty of
# MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
# GNU General Public License for more details.
#
# You should have received a copy of the GNU General Public
# License
# along with GCstar; if not, write to the Free Software
# Foundation, Inc., 51 Franklin St, Fifth Floor, Boston, MA
# 02110-1301, USA
#
#####

use strict;
use utf8;

use GCPlugins::GCbooks::GCbooksCommon;

```

```
{
    package GCPlugins::GCbooks::GCPluginChapitre;

    use base qw(GCPlugins::GCbooks::GCbooksPluginsBase);
    use URI::Escape;

    sub start
    {
        my ($self, $tagname, $attr, $attrseq, $origtext) = @_;

        $self->{inside}->{$tagname}++;

        if ($self->{parsingList})
        {
            if (($tagname eq 'a') && ( $attr->{id} =~
m/_searchProductDisplay_h1ProductTitle/))
            {
                $self->{itemIdx}++;
                $self->{itemsList}[$self->{itemIdx}]->{url} =
"http://www.chapitre.com" . $attr->{href};
                $self->{isTitle} = 1 ;
            }
            elsif ($tagname eq 'em')
            {
                $self->{isAuthor} = 1 ;
            }
            elsif (($tagname eq 'span') && ( $attr->{class} eq
'editeur'))
            {
                $self->{isPublisher} = 1;
            }
            elsif (($tagname eq 'span') && ( $attr->{class} eq
'dateParution'))
            {
                $self->{isPublication} = 1;
            }
        }
        else
        {
            if (($tagname eq 'div') && ($attr->{class} eq
'clear'))
            {
                $self->{isDescription} = 0 ;
            }
            elsif (($tagname eq 'h1') && ( $attr->{class} eq
'ProductSummary-title'))
            {
                $self->{isTitle} = 1 ;
            }
            elsif (($tagname eq 'h1') && ( $attr->{class} eq
```

```

'product-title'))
{
    $self->{isTitle} = 1 ;
}
elseif (($tagname eq 'div') && ( $attr->{id} eq
'ctl00_PHCenter_ProductFile1_ProductTitle1_pnlTranslator'))
{
    $self->{isTranslator} = 1 ;
}
elseif (($tagname eq 'tpftraducteurtpf') && (
$self->{isTranslator} eq 1))
{
    $self->{isTranslator} = 2 ;
}
elseif (($tagname eq 'img') && ( $attr->{itemprop} eq
'image') && (
index($attr->{src},"http://images.chapitre.com/indispo") eq -1 ))
{
    $self->{curInfo}->{cover} = $attr->{src} if
($self->{curInfo}->{cover} eq "");
}
elseif (($tagname eq 'div') && ($attr->{itemprop} eq
'description'))
{
    $self->{isDescription} = 1 ;
}
elseif (($tagname eq 'tpfdescriptiontpf') &&
($self->{isDescription} eq 1))
{
    $self->{isDescription} = 2 ;
}
elseif (($tagname eq 'span') && ($self->{isDescription}
eq 1))
{
    $self->{isDescription} = 2 ;
}
elseif (($tagname eq 'a') && ( $attr->{href} =~
m|/CHAPITRE/fr/search/Default.aspx?collection=|i))
{
    $self->{isCollection} = 1 ;
}
elseif (($tagname eq 'a') && ( $attr->{href} =~
m|/CHAPITRE/fr/search/Default.aspx?editeur=|i))
{
    $self->{isPublisher} = 1 ;
}
elseif (($tagname eq 'a') && ( $attr->{href} =~
m|/CHAPITRE/fr/t/|i))
{
    $self->{isGenre} = 1 ;
}
}

```

```
        elsif (0 eq 1 && ($tagname eq 'a') && ( $attr->{href}  
=~ m|/CHAPITRE/fr/p/|i) && ( $attr->{id} =~  
m|ctl00_PHCenter_productTop_|i))  
        {  
            $self->{isAuthor} = 1 ;  
        }  
        elsif (($tagname eq 'span') && ( $attr->{id} =~  
m|ctl00_PHCenter_productBottom_productDetail_rpDetails_|i))  
        {  
            $self->{isAnalyse} = 1 ;  
        }  
    }  
}  
  
sub end  
{  
    my ($self, $tagname) = @_ ;  
  
    $self->{inside}->{$tagname}-- ;  
}  
  
sub text  
{  
    my ($self, $origtext) = @_ ;  
  
    if ($self->{parsingList})  
    {  
        if ($self->{isTitle})  
        {  
            $self->{itemsList}[$self->{itemIdx}]->{title} =  
$origtext ;  
            $self->{isTitle} = 0 ;  
        }  
        elsif ($self->{isAuthor})  
        {  
            if ($origtext =~ m/Les outils de recherche/i)  
            {  
                # the search failed  
                $self->{isAuthor} = 0 ;  
                return ;  
            }  
  
            # Enleve les blancs en debut de chaine  
            $origtext =~ s/^\s+// ;  
            # Enleve les blancs en fin de chaine  
            $origtext =~ s/\s+$//g ;  
            if  
($self->{itemsList}[$self->{itemIdx}]->{authors} eq '')  
            {  
                $self->{itemsList}[$self->{itemIdx}]->{authors}
```

```

= $origtext;
    }
    else
    {
        $self->{itemsList}[$self->{itemIdx}]->{authors}
.= ' ', ' ';
        $self->{itemsList}[$self->{itemIdx}]->{authors}
.= $origtext;
    }
    $self->{isAuthor} = 0 ;
}
elsif ($self->{isAnalyse})
{
    $self->{isPublisher} = 1 if ($origtext =~
m/Editeur :/i);
    $self->{isSerie} = 1 if ($origtext =~ m/Collection
:/i);
    $self->{isPublication} = 1 if ($origtext =~ m/Date
:/i);
    $self->{isAnalyse} = 0 ;
}
elsif ($self->{isPublisher})
{
    $origtext =~ s/Editeur : //;
    $origtext =~ s/\.$//;
    my @array = split(/\n/, $origtext);
    $self->{itemsList}[$self->{itemIdx}]->{edition} =
$array[0];
    $self->{isPublisher} = 0 ;
}
elsif ($self->{isPublication})
{
    $origtext =~ s/Date de parution ://;
    $origtext =~ s/\.$//;
    my @array = split(/\n/, $origtext);
    $self->{itemsList}[$self->{itemIdx}]->{publication} = $array[0];
    $self->{isPublication} = 0 ;
}
elsif ($self->{isSerie})
{
    my @array = split(/\n/, $origtext);
    $self->{itemsList}[$self->{itemIdx}]->{serie} =
$array[0];
    $self->{isSerie} = 0 ;
}
}
else
{
    # Enleve les blancs en debut de chaine
    $origtext =~ s/^\s+//;
    # Enleve les blancs en fin de chaine

```

```
$origtext =~ s/\s+$/g;
if ($self->{isTitle})
{
    $self->{curInfo}->{title} = $origtext;
    $self->{isTitle} = 0 ;
}
elsif ($self->{isAuthor} eq 1)
{
    if ( $origtext ne '' )
    {
        my @array = split(/;/,$origtext);
        my $element;
        foreach $element (@array)
        {
            my @nom_prenom = split(/,/, $element);
            # Enleve les blancs en debut de chaine
            $nom_prenom[0] =~ s/^\s+//;
            $nom_prenom[1] =~ s/^\s+//;
            # Enleve les blancs en fin de chaine
            $nom_prenom[0] =~ s/\s+$//;
            $nom_prenom[1] =~ s/\s+$//;
            if ($self->{curInfo}->{authors} eq '')
            {
                if ($nom_prenom[1] ne '')
                {
                    $self->{curInfo}->{authors} =
$nom_prenom[1] ." " . $nom_prenom[0];
                }
                else
                {
                    $self->{curInfo}->{authors} =
$nom_prenom[0];
                }
            }
            else
            {
                if ($nom_prenom[1] ne '')
                {
                    $self->{curInfo}->{authors} .= ", " .
$nom_prenom[1] ." " . $nom_prenom[0];
                }
                else
                {
                    $self->{curInfo}->{authors} .= ", " .
$nom_prenom[0];
                }
            }
        }
        $self->{isAuthor} = 0 ;
    }
}
```

```
}
elseif ($self->{isTranslator} eq 2)
{
    $self->{curInfo}->{translator} = $origtext;
    $self->{isTranslator} = 0 ;
}
elseif ($self->{isPublisher} eq 1)
{
    $self->{curInfo}->{publisher} = $origtext;
    $self->{isPublisher} = 0 ;
}
elseif ($self->{isDescription} eq 2)
{
    $self->{curInfo}->{description} = $origtext;
    $self->{isDescription} = 0 ;
}
elseif ($self->{isPublication} && $self->{isAnalyse})
{
    $self->{curInfo}->{publication} = $origtext;
    $self->{isPublication} = 0 ;
}
elseif ($self->{isISBN} && $self->{isAnalyse})
{
    $self->{curInfo}->{isbn} = $origtext;
    $self->{isISBN} = 0 ;
}
elseif ($self->{isPage} && $self->{isAnalyse})
{
    if ($origtext ne '')
    {
        $self->{curInfo}->{pages} = $origtext;
        $self->{isPage} = 0 ;
    }
}
elseif ($self->{isCollection})
{
    $self->{curInfo}->{serie} = $origtext;
    $self->{isCollection} = 0 ;
}
elseif ($self->{isGenre})
{
    $origtext =~ s|/,|gi;
    $self->{curInfo}->{genre} = $origtext;
    $self->{isGenre} = 0 ;
}
elseif ($self->{isLanguage} && $self->{isAnalyse})
{
    $self->{curInfo}->{language} = $origtext;
    $self->{isLanguage} = 0 ;
    $self->{isAnalyse} = 0 ;
}
```

```
        elseif ($self->{isAnalyse})
        {
            $self->{isPublication} = 1 if ($origtext =~
m/parution/i);
            $self->{isISBN} = 1 if ($origtext =~ m/EAN13/i);
            $self->{isPublisher} = 1 if ($origtext =~
m/Editeur/i);
            $self->{isLanguage} = 1 if ($origtext =~
m/Langue/i);
            $self->{isPage} = 1 if ($origtext =~ m/Nombre de
page/i);
            $self->{isAuthor} = 1 if ($origtext =~
m/Auteur/i);

            $self->{isAnalyse} = 0 ;
        }
    }
}

sub new
{
    my $proto = shift;
    my $class = ref($proto) || $proto;
    my $self = $class->SUPER::new();
    bless ($self, $class);

    $self->{hasField} = {
        title => 1,
        authors => 1,
        publication => 1,
        format => 0,
        edition => 1,
        serie => 1,
    };

    $self->{isTitle} = 0;
    $self->{isAuthor} = 0;
    $self->{isPublisher} = 0;
    $self->{isSerie} = 0;
    $self->{isPublication} = 0;
    $self->{isAnalyse} = 0;
    $self->{isDescription} = 0;
    $self->{isISBN} = 0;
    $self->{isLanguage} = 0;
    $self->{isCollection} = 0;
    $self->{isTranslator} = 0;
    $self->{isGenre} = 0;

    return $self;
}
```



```

sub preProcess
{
    my ($self, $html) = @_;

    $self->{isTitle} = 0;
    $self->{isAuthor} = 0;
    $self->{isPublisher} = 0;
    $self->{isSerie} = 0;
    $self->{isPublication} = 0;
    $self->{isAnalyse} = 0;
    $self->{isDescription} = 0;
    $self->{isISBN} = 0;
    $self->{isLanguage} = 0;
    $self->{isCollection} = 0;
    $self->{isTranslator} = 0;
    $self->{isGenre} = 0;

    if ($self->{parsingList})
    {
        $html =~ s|<b>||gi;
        $html =~ s|</b>||gi;
        $html =~ s|</a>,|</a>,<tpfauthortpf>|gi;
        $html =~
s|(auteur)</em>|(auteur)</em><tpfauthortpf>|gi;
    }
    else
    {

        $html =~ s|</strong>|</strong><tpftraducteurtpf>|gi;
        #$html =~ s|</h3>|</h3><tpfdescriptiontpf>|gi;

        $html =~ s|<u>||gi;
        $html =~ s|<li>|\\n* |gi;
        $html =~ s|<br>|\\n|gi;
        $html =~ s|<br />|\\n|gi;
        $html =~ s|<b>||gi;
        $html =~ s|</b>||gi;
        $html =~ s|<i>||gi;
        $html =~ s|</i>||gi;
        $html =~ s|<p>|\\n|gi;
        $html =~ s|</p>||gi;
        $html =~ s|\\x{92}|'|g;
        $html =~ s|&#146;|'|gi;
        $html =~ s|&#149;|*|gi;
        $html =~ s|&#133;|...|gi;
        $html =~ s|\\x{85}|...|gi;
        $html =~ s|\\x{8C}|OE|gi;
        $html =~ s|\\x{9C}|oe|gi;
        $html =~ s|\\n+|\\n|gi;
    }
}

```

```
    }

    return $html;
}

sub getSearchUrl
{
my ($self, $word) = @_;

# utilisation d'une requête GET à la place d'un POST
#$word =~ s/\+/ /g;
#return
('http://www.chapitre.com/CHAPITRE/fr/search/Default.aspx?search=t
rue', ["quicksearch" => "$word"] );
#return
('http://www.chapitre.com/CHAPITRE/fr/search/Default.aspx?search=t
rue&quicksearch='.$word);
return
"http://www.chapitre.com/CHAPITRE/fr/search/Default.aspx?quicksear
ch=" . $word . "&optSearch=BOOKS";
}

sub getItemUrl
{
my ($self, $url) = @_;

return $url;
}

sub getName
{
return "Chapitre.com";
}

sub getCharset
{
my $self = shift;
return "ISO-8859-15";
}

sub getAuthor
{
return 'TPF - Kerenoc';
}

sub getLang
{
return 'FR';
}
```

```

    sub getSearchFieldsArray
    {
        return ['isbn', 'title'];
    }
}

1;

```

Films : GCAllocine

[GCAllocine.pm](#)

[GCAllocine.pm](#)

```

package GCPlugins::GCfilms::GCAllocine;

#####
#
# Copyright 2005-2010 Christian Jodar
# Copyright 2015-2016 Kérénoc (kerenoc01 à Google Mail)
#
# This file is part of GCstar.
#
# GCstar is free software; you can redistribute it and/or modify
# it under the terms of the GNU General Public License as
# published by
# the Free Software Foundation; either version 2 of the License,
# or
# (at your option) any later version.
#
# GCstar is distributed in the hope that it will be useful,
# but WITHOUT ANY WARRANTY; without even the implied warranty of
# MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
# GNU General Public License for more details.
#
# You should have received a copy of the GNU General Public
# License
# along with GCstar; if not, write to the Free Software
# Foundation, Inc., 51 Franklin St, Fifth Floor, Boston, MA
# 02110-1301, USA
#
#####

use strict;
use utf8;

use GCPlugins::GCfilms::GCfilmsCommon;

```

```
{

package GCPlugins::GCfilms::GCPluginAllocine;

use base qw(GCPlugins::GCfilms::GCfilmsPluginsBase);

sub start
{
    my ($self, $tagname, $attr, $attrseq, $origtext) = @_;
    $self->{inside}->{$tagname}++;

    if ($self->{parsingList})
    {
        if ($self->{insideResults} eq 1)
        {
            if ( ($tagname eq "a")
                && ($attr->{href} =~
/^\s*/film\s*/fichefilm_gen_cfilm=/)
                && ($self->{isMovie} eq 0))
            {
                my $url = $attr->{href};
                $self->{isMovie} = 1;
                $self->{isInfo} = 0;
                $self->{itemIdx}++;
                $self->{itemsList}[ $self->{itemIdx} ]->{url}
= $url;
            }
            elsif (($tagname eq "td") && ($self->{isMovie} eq
1))
            {
                $self->{isMovie} = 2;
            }
            elsif (($tagname eq "a") && ($self->{isMovie} eq
2))
            {
                $self->{isMovie} = 3;
            }
            elsif (($tagname eq "br") && ($self->{isMovie} eq
3))
            {
                $self->{itemsList}[ $self->{itemIdx}
]->{title} =~ s/^\s*//;
                $self->{itemsList}[ $self->{itemIdx}
]->{title} =~ s/\s*$//;
                $self->{itemsList}[ $self->{itemIdx}
]->{title} =~ s/\s+/ /g;
                $self->{isMovie} = 4;
            }
            elsif (($tagname eq "span")
                && ($attr->{class} eq "fs11"))
        }
    }
}
```

```

        && ($self->{isMovie} eq 4))
    {
        $self->{isInfo} = 1;
        $self->{isMovie} = 0;
    }
    elsif (($tagname eq "br") && ($self->{isInfo} eq
1))
    {
        $self->{isInfo} = 2;
    }
    elsif (($tagname eq "br") && ($self->{isInfo} eq
2))
    {
        $self->{isInfo} = 3;
    }
    }
    else
    {
        if (($tagname eq "span") && ($attr->{class} eq "thumbnail-
link"))
        {
            $self->{insidePicture} = 1;
        }
        elsif (($tagname eq "img") && ($self->{insidePicture} eq
1))
        {
            my $src = $attr->{src};
            if (!$self->{curInfo}->{image})
            {
                $self->{curInfo}->{image} = $src;
            }
            $self->{insidePicture} = 0;
        }
        elsif ($tagname eq "h1")
        {
            $self->{insideTitle} = 1;
        }
        elsif (($tagname eq "span") && ($self->{insideDate} eq 1))
        {
            $self->{insideDate} = 2;
        }
        elsif (($tagname eq "span") && ($attr->{itemprop} eq
"director"))
        {
            $self->{insideDirector} = 1;
        }
        elsif (($tagname eq "span") && ($attr->{itemprop} eq
"duration"))
        {
            $self->{insideTime} = 1;

```

```
        }
        elseif (($tagname eq "span") && ($self->{insideDirector} eq
1))
        {
            $self->{insideDirector} = 2;
        }
        elseif (($tagname eq "div") && ($attr->{itemprop} eq
"actor") && !$self->{curInfo}->{nextUrl})
        {
            # recuperation des acteurs uniquement dans la page du
casting : nextUrl = 0
            $self->{insideActor} = 1;
        }
        elseif (($tagname eq "span") && ($attr->{itemprop} eq
"name") && ($self->{insideActor} eq 1))
        {
            $self->{insideActor} = 2;
            # item where the actor name is followed by role : name
part
        }
        elseif (($tagname eq "span") && ($attr->{class} =~ m/col-
xs/) && ($self->{insideActor} eq 1))
        {
            $self->{insideActor} = 3;
            # item where the role is followed by actor name : role
part
        }
        elseif (($tagname eq "span") && ($self->{insideGenre} eq
1))
        {
            $self->{insideGenre} = 2;
        }
        elseif (($tagname eq "span") && ($self->{insideCountry} eq
1))
        {
            $self->{insideCountry} = 2;
        }
        elseif (($tagname eq "span") && ($attr->{class} eq
"stareval-note") && ($self->{insidePressRating} eq 1))
        {
            $self->{insidePressRating} = 2;
        }
        elseif (($tagname eq "div") && ($attr->{class} eq
"breaker"))
        {
            $self->{insidePressRating} = 0;
        }
        elseif (($tagname eq "div") && ($attr->{itemprop} eq
"description"))
        {

```

```
                $self->{insideSynopsis} = 1;
            }
            elsif (($tagname eq "span") && ($self->{insideOriginal} eq
1))
            {
                $self->{insideOriginal} = 2;
            }
        }
    }

    sub end
    {
        my ($self, $tagname) = @_;
        $self->{inside}->{$tagname}--;

        if ($tagname eq "li")
        {
            $self->{insideDirector} = 0;
            $self->{insideGenre} = 0;
        }
        elsif ($tagname eq "div")
        {
            $self->{insideCountry} = 0;
            $self->{insideSynopsis} = 0;
            $self->{insideGenre} = 0;
        }
        elsif ($tagname eq "th")
        {
            $self->{insideSynopsis} = 0;
        }
        elsif ($tagname eq "table")
        {
            $self->{insideResults} = 0;
        }
    }

    sub text
    {
        my ($self, $origtext) = @_;

        if ($self->{parsingList})
        {
            if (($origtext =~ m/(\d+) r..?sultats? trouv..?s? dans
les titres de films/) && ($1 > 0))
            {
                $self->{insideResults} = 1;
            }
            if ($self->{isMovie} eq 3)
            {
```

```
        $self->{itemsList}[ $self->{itemIdx} ]->{title} .=
$origtext;
    }
    if ($self->{isInfo} eq 1)
    {
        if ($origtext =~ /\s*([0-9]{4})/)
        {
            $self->{itemsList}[ $self->{itemIdx} ]->{date}
= $1;
        }
    }
    elseif ($self->{isInfo} eq 2)
    {
        if ($origtext =~ /^s*de (.*)/)
        {
            $self->{itemsList}[ $self->{itemIdx}
]->{director} = $1;
        }
    }
    elseif ($self->{isInfo} eq 3)
    {
        if ( ($origtext =~ m/^s*avec (.*)/)
            && (!$self->{itemsList}[ $self->{itemIdx}
]->{actors}))
        {
            $self->{itemsList}[ $self->{itemIdx}
]->{actors} = $1;
        }
        $self->{isInfo} = 0;
    }
}
else
{
    my ($self, $origtext) = @_;
    $origtext =~ s/[\r\n]//g;
    $origtext =~ s/^s*//;
    $origtext =~ s/s*$//;

    if ($self->{insideTitle} eq 1)
    {
        # two pass plugin : {title} is set in the first pass
        if (! $self->{curInfo}->{title})
        {
            # loading second web page for casting
            my $fileCasting =
$self->{curInfo}->{$self->{urlField}};
            $fileCasting =~ s/_gen_cfilm=/-/;
            $fileCasting =~ s/.html/\casting/;
            $self->{curInfo}->{nextUrl} = $fileCasting;
        }
    }
}
```



```

        $self->{curInfo}->{title} = $origtext if (!
$self->{curInfo}->{title});
        $self->{insideTitle} = 0;
    }
    elsif (($self->{insideDate} eq 2) && (length($origtext) >
1))
    {
        $self->{curInfo}->{date} = $self->decodeDate($origtext)
            if !($origtext =~ /inconnu/);
        $self->{insideDate} = 0;
        $self->{insideTime} = 1;
    }
    elsif ($self->{insideTime} eq 1)
    {
        $origtext =~ s/^\s+//;
        $origtext =~ s/\(//;
        $origtext =~ s/min\\)/g;
        my $hours = $origtext;
        $hours =~ s/h.*//;
        my $minutes = $origtext;
        $minutes =~ s/.*h *//;
        $self->{curInfo}->{time} = $hours * 60 + $minutes;
        $self->{insideTime} = 0;
    }
    elsif (($origtext =~ /^Date de sortie/)
        && (!$self->{curInfo}->{date}))
    {
        $self->{insideDate} = 1;
    }
    elsif (($origtext =~ /^Date de reprise/)
        && (!$self->{curInfo}->{date}))
    {
        $self->{insideDate} = 1;
    }
    elsif ($self->{insideTime} eq 1)
    {
        $origtext =~ /(\d+)h\s*(\d+)m/;
        my $time = ($1*60) + $2;
        $self->{curInfo}->{time} = $time." m.";
        $self->{insideTime} = 0;
    }
    elsif ($self->{insideDirector} eq 2)
    {
        if ($self->{curInfo}->{director})
        {
            $self->{curInfo}->{director} .= ", ".$origtext;
        }
        else
        {
            $self->{curInfo}->{director} .= $origtext;

```

```
}
$self->{insideDirector} = 0;
}
elseif ($self->{insideGenre} eq 2)
{
$origtext = "," if $origtext =~ m/^,/;
$self->{curInfo}->{genre} .= $origtext;
}
elseif ($origtext =~ /^[\s\n]*Genre/)
{
$self->{insideGenre} = 1;
}
elseif ($self->{insideCountry} eq 2)
{
$origtext = "," if $origtext =~ m/^,/;
$self->{curInfo}->{country} .= $origtext;
}
elseif ($self->{insideActor} > 1)
{
$origtext =~ s/\s*plus\s*//;
$origtext =~ s/\s*Rôle\s*:\s*//;

return if ($origtext eq "," || $origtext eq ' ');

if ($self->{insideActor} eq 2)
{
    $self->{actor} = $origtext;
    $self->{insideActor} = 3 if (!$self->{role});
}
elseif ($self->{insideActor} eq 3)
{
    $self->{role} = $origtext;
    $self->{insideActor} = 2 if (!$self->{actor});
}
if ($self->{actor} && $self->{role})
{
    push @{$self->{curInfo}->{actors}},
[$self->{actor}];
    push
@{$self->{curInfo}->{actors}->[$self->{actorsCounter}]},
$self->{role};
    $self->{actorsCounter}++;
    $self->{actor} = "";
    $self->{role} = "";
    $self->{insideActor} = 0;
}
}
elseif ($origtext =~ /Nationalité/)
{
$self->{insideCountry} = 1;
}
```

```
    }
    elsif ($origtext =~ /^Presse$/)
    {
        $self->{insidePressRating} = 1;
    }
    elsif ($self->{insidePressRating} eq 2)
    {
        $origtext =~ s/,/,./;
        $self->{curInfo}->{ratingpress} .= $origtext * 2;
        $self->{insidePressRating} = 0;
    }
    elsif ($origtext =~ m/^Interdit aux moins de (\d+) ans/)
    {
        $self->{curInfo}->{age} = $1;
    }
    elsif ($self->{insideSynopsis} eq 1)
    {
        $self->{curInfo}->{synopsis} .= $origtext;
    }
    elsif ($self->{insideOriginal} eq 2)
    {
        $self->{curInfo}->{original} = $origtext;
        $self->{insideOriginal} = 0;
    }
    elsif ($origtext =~ /^R..?alis..? par/)
    {
        $self->{insideDirector} = 1;
    }
    elsif ($origtext =~ m/Titre original/)
    {
        $self->{insideOriginal} = 1;
    }
}
}
```

sub new

```
{
    my $proto = shift;
    my $class = ref($proto) || $proto;
    my $self = $class->SUPER::new();

    $self->{hasField} = {
        title    => 1,
        date     => 1,
        director => 1,
        actors   => 1,
    };

    $self->{isInfo}      = 0;
    $self->{isMovie}     = 0;
    $self->{insideResults} = 0;
}
```

```
$self->{curName}      = undef;
$self->{curUrl}        = undef;
$self->{actorsCounter} = 0;

bless($self, $class);
return $self;
}

sub preProcess
{
    my ($self, $html) = @_;

    return $html;
}

sub getSearchUrl
{
    my ($self, $word) = @_;

    # f=3 ?
    # return
    "http://www.allocine.fr/recherche/?q=$word&f=3&rub=1";
    return "http://www.allocine.fr/recherche/1/?q=$word";
}

sub getSearchCharset
{
    my $self = shift;

    # Need urls to be double character encoded
    return "utf8";
}

sub getItemUrl
{
    my ($self, $url) = @_;

    return "http://www.allocine.fr" . $url;
}

sub getName
{
    return "Allocine.fr";
}

sub getAuthor
{
    return 'Tian - Kerenoc';
}
```

```

sub getLang
{
    return 'FR';
}

sub getCharset
{
    # return "UTF-8"; # For 1.5.0 Win32
    return "ISO-8859-1"; # For 1.5.0 Win32 with
/lib/gcstar/GCPlugins/ ver.1.5.9svn
}

sub decodeDate
{
    my ($self, $date) = @_;

    # date déjà dans le bon format
    return $date if ($date =~ m|/|);

    # date à convertir au format jour/mois/année

    my @dateItems = split(/\s/, $date);
    my @listeMois =
    ("janvier", "f.*vrier", "mars", "avril", "mai", "juin",
    "juillet", "ao.*t", "septembre", "octobre", "novembre", "décembre");
    my $mois = 0;
    my $nbDates = (scalar @dateItems);

    while ($mois < (scalar @listeMois) && !($dateItems[$nbDates-2]
    =~ m/$listeMois[$mois]/))
    {
        $mois++;
    }
    $mois++;
    return
    sprintf("%02d/%02d", $dateItems[0], $mois)."/".$dateItems[$nbDates-1]
    if ($nbDates > 2);
    return sprintf("01/%02d", $mois)."/".$dateItems[1] if ($nbDates
    eq 2);
    return "";
}

1;

```

Attention : pour pouvoir obtenir le casting complet il a fallu traiter la récupération de plusieurs pages web pour un film donné et donc modifier aussi le fichier GCPluginsBase.pm (chargement de plusieurs pages tant que l'attribut \$self->{curlInfo}->{nextUrl} est positionné). En plus de l'option 1 de debug qui sauve les fichiers web et 2 qui utilise les fichiers sauvés, j'ai ajouté une option 3 qui combine les

deux pour agir comme un cache et limiter la sollicitation des serveurs en phase de mise au point.

[GCPluginsBase.pm](#)

```
package GCPlugins::GCPluginsBase;

#####
#
# Copyright 2005-2010 Christian Jodar
# Copyright 2015-2016 K  renoc (kerenoc01 on Google mail)
#
# This file is part of GCstar.
#
# GCstar is free software; you can redistribute it and/or modify
# it under the terms of the GNU General Public License as published by
# the Free Software Foundation; either version 2 of the License, or
# (at your option) any later version.
#
# GCstar is distributed in the hope that it will be useful,
# but WITHOUT ANY WARRANTY; without even the implied warranty of
# MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
# GNU General Public License for more details.
#
# You should have received a copy of the GNU General Public License
# along with GCstar; if not, write to the Free Software
# Foundation, Inc., 51 Franklin St, Fifth Floor, Boston, MA 02110-1301, USA
#
#####

use strict;
use utf8;

{
    package GCPluginParser;
    use base qw(HTML::Parser);
    use LWP::Simple qw($ua);
    use HTTP::Cookies::Netscape;
    use URI::Escape;
    use HTML::Entities;
    use Encode;
    use File::Spec;

    sub new
    {
        my $proto = shift;
        my $class = ref($proto) || $proto;
        my $self = $class->SUPER::new();

        $ua->agent('Mozilla/5.0 (X11; U; Linux i686; en-US; rv:1.7.5)
```

```
Gecko/20041111 Firefox/1.0');
    $ua->default_header('Accept-Encoding' => 'x-gzip');
    $ua->default_header('Accept' => 'text/html');
    $self->{ua} = $ua;

    $self->{itemIdx} = -1;
    $self->{itemsList} = ();

    bless ($self, $class);
    return $self;
}

sub getItemsNumber
{
    my ($self) = @_;

    return $self->{itemIdx} + 1;
}

sub getItems
{
    my ($self) = @_;
    return @{$self->{itemsList}};
}

sub load
{
    my $self = shift;

    $self->checkProxy;
    $self->checkCookieJar;

    $self->{itemIdx} = -1;
    $self->{isInfo} = 0;
    $self->{itemsList} = ();

    #my $word = uri_escape_utf8($self->{title});
    my $title2 = encode($self->getSearchCharset, $self->{title});
    my $word = uri_escape($title2);
    $word =~ s/%20/+/g;

    my $post;
    my $html;

    # For multi-pass plugins, the plugin will have set the url to load
for
the
    # the next pass as nextUrl. If this doesn't exist, we're either on

    # first pass, or only using a one-pass plugin, so call getSearchUrl
    # to find the url to retrieve
    if ($self->{nextUrl})
```

```
{
    $html = $self->loadPage($self->{nextUrl});
}
else
{
    $html = $self->loadPage($self->getSearchUrl($word));
}

return if (length $html eq 0);
$self->{parsingList} = 1;

$html = $self->preProcess($html);

decode_entities($html)
    if $self->decodeEntitiesWanted;
$self->{inside} = undef;

$self->parse($html);

my @noConversion = @{$self->getNotConverted};
foreach my $item (@{$self->{itemsList}})
{
    foreach (keys %{$item})
    {
        next if $_ eq 'url';
        $item->{$_} = $self->convertCharset($item->{$_})
            if ! GCUtils::inArrayTest($_, @noConversion);
    }
}
}

}

sub loadPage
{
    my ($self, $url, $post, $noSave) = @_;
    my $debugPhase = $ENV{GCS_DEBUG_PLUGIN_PHASE};
    my $debugFile;

    $debugFile = File::Spec->tmpdir.'/'.GCUtils::getSafeFileName($url)
        if ($debugPhase > 0);
    $self->{loadedUrl} = $url if ! $noSave;
    my $response;
    my $result;
    if ($debugPhase < 2 || (!(-f $debugFile)))
    {
        if ($post)
        {
            $response = $ua->post($url, $post);
        }
    }
}
```



```
        else
        {
            $response = $ua->get($url);
        }

        #UnclePetros 03/07/2011:
        #code to handle correctly 302 response messages
        my $label1 = $response->code;
        if($response->code == '302'){
            my $location = $response->header("location");
            $response = $ua->get($location);
            $self->{loadedUrl} = $location;
        }

        eval {
            $result = $response->decoded_content;
        };
        if ($debugPhase == 1 || $debugPhase == 3)
        {
            open DEBUG_FILE, ">$debugFile";
            binmode(DEBUG_FILE, ":utf8");
            close DEBUG_FILE;
        }
    }
    else
    {
        local $/;
        open DEBUG_FILE, "$debugFile";
        $result = <DEBUG_FILE>;
        utf8::decode($result);
    }
    return $result || ($response && $response->content);
}

sub capWord
{
    my ($self, $msg) = @_;

    use locale;

    (my $newmsg = lc $msg) =~ s/(\s|,|^)(\w)(\w)(\w*?)/$1U$2\E$3$4/gi;
    return $newmsg;
}

sub getSearchFieldsArray
{
    return [''];
}

sub getSearchFields
{

```

```
my ($self, $model) = @_;

my $result = '';
$result .= $model->getDisplayedLabel($_).', ' foreach
(@{$self->getSearchFieldsArray});
$result =~ s/, $//;
return $result;
}

sub hasField
{
    my ($self, $field) = @_;

    return $self->{hasField}->{$field};
}

sub getExtra
{
    return '';
}

# Character set for web page text
sub getCharset
{
    my $self = shift;

    return "ISO-8859-1";
}

# Character set for encoding search term, can sometimes be different
# to the page encoding, but we default to the same as the page set
sub getSearchCharset
{
    my $self = shift;

    return getCharset;
}

# For some plugins, we need extra checks to determine if urls match
# the language the plugin is written for. This allows us to correctly
determine
# if a drag and dropped url is handled by a particular plugin. If these
# checks are necessary, return 1, and make sure plugin handles the
# the testURL function correctly
sub needsLanguageTest
{
    return 0;
}

# Used to test if a given url is handled by the plugin. Only required if
```

```
# needsLanguageTest is true.
sub testURL
{
    my ($self, $url) = @_;
    return 1
}

# Determines whether plugin should be the default plugins gcstar uses.
# Plugins with this attribute set will appear first in plugin list,
# and will be highlighted with a star icon. A returned value of 1
# means the plugin is preferred if it's language matches the user's
language,
# a returned value of 2 mean's it's preferred regardless of the
language.
sub isPreferred
{
    return 0;
}

sub getPreferred
{
    return isPreferred;
}

sub getNotConverted
{
    my $self = shift;
    return [];
}

sub decodeEntitiesWanted
{
    return 1;
}

sub getDefaultPictureSuffix
{
    return '';
}

sub convertCharset
{
    my ($self, $value) = @_;

    my $result = $value;
    if (ref($value) eq 'ARRAY')
    {
        foreach my $line(@{$value})
        {
            my $i = 0;
            eval {
```

```
        map {$_ = decode($self->getCharset, $_)} @{$line};
    }
}
else
{
    eval {
        $result = decode($self->getCharset, $result);
    };
}
return $result;
}

sub getItemInfo
{
    my $self = shift;

    eval {
        $self->init;
    };
    my $idx = $self->{wantedIdx};
    my $url = $self->getItemUrl($self->{itemsList}[$idx]->{url});
    $self->{curInfo} = {};
    $self->loadUrl($url);

    # multi-pass plugins that requires multiple web page to get all info on
a single collection item
    # for example : Allmovie (tabs to get casting), Allocine (idem)
    # the plugin can set {nextUrl} to fetch next web page, the information
is cumulative in {curInfo}
    while ($self->{curInfo}->{nextUrl})
    {
        my $nextUrl = $self->{curInfo}->{nextUrl};
        $self->{curInfo}->{nextUrl} = 0;
        $self->loadUrl($nextUrl);
    }

    return $self->{curInfo};
}

sub changeUrl
{
    my ($self, $url) = @_;

    return $url;
}

sub loadUrl
{
    my ($self, $url) = @_;
    $self->checkProxy;
```

```

$self->checkCookieJar;
my $realUrl = $self->changeUrl($url);
my $html = $self->loadPage($realUrl);
$self->{parsingList} = 0;
#$html = $self->convertCharset($html);

# $self->{curInfo} = {} if (!$self->{curInfo}->{title});
# once the urlField is set don't change it (plugins fetching multiple
pages for one item)
$self->{curInfo}->{$self->{urlField}} = $url if
(!$self->{curInfo}->{$self->{urlField}});

$html = $self->preProcess($html);
decode_entities($html)
    if $self->decodeEntitiesWanted;

$self->{inside} = undef;
$self->parse($html);

my @noConversion = @{$self->getNotConverted};

foreach (keys %{$self->{curInfo}})
{
    next if $_ eq $self->{urlField};
    $self->{curInfo}->{$_} =
$self->convertCharset($self->{curInfo}->{$_})
        if ! GCUtils::isArrayTest($_, @noConversion);
    if (ref($self->{curInfo}->{$_}) ne 'ARRAY')
    {
        $self->{curInfo}->{$_} =~ s/\\|/,/gm;
        $self->{curInfo}->{$_} =~ s/\\r/gm;
        $self->{curInfo}->{$_} =~ s/[ \t]*$/gm;
    }
}
$self->{curInfo}->{$self->{urlField}} .=
$GCModel::linkNameSeparator.$self->getName;
return $self->{curInfo};
}

sub setProxy
{
    my ($self, $proxy) = @_;

    $self->{proxy} = $proxy;
}

sub checkProxy
{
    my $self = shift;
    $ua->proxy(['http'], $self->{proxy});
    #$self->{ua}->proxy(['http'], $self->{proxy});
}

```

```
}

sub setCookieJar
{
    my ($self, $cookieJar) = @_;
    $self->{cookieJar} = $cookieJar;
}

sub checkCookieJar
{
    my $self = shift;
    $ua->cookie_jar(HTTP::Cookies::Netscape->new(
        'file' => "$self->{cookieJar}",
        'autosave' => 1,));
}

# Used to set the number of passes the plugin requires
sub getNumberPasses
{
    # Most plugins only need to search once, so default to one pass
    return 1;
}

# Returns undef if it doesn't support search using barcode scanner
sub getEanField
{
    return undef;
}

}

1;
```

Pour ceux qui veulent récupérer des infos sur des films en anglais, voici le plugin GCfilms/GCAllmovie.pm (attention c'est un plugin en 2 passes qui nécessite donc un GCPluginsBase.pm modifié).

[GCAllmovie](#)

[GCfilms/GCAllmovie](#)

```
package GCPlugins::GCfilms::GCAllmovie;

#####
#
# Copyright 2005-2010 Christian Jodar
# Copyright 2015-2016 Kerenoc (kerenoc01 on Google mail)
#
# This file is part of GCstar.
```

```
#
# GCstar is free software; you can redistribute it and/or modify
# it under the terms of the GNU General Public License as
# published by
# the Free Software Foundation; either version 2 of the License,
# or
# (at your option) any later version.
#
# GCstar is distributed in the hope that it will be useful,
# but WITHOUT ANY WARRANTY; without even the implied warranty of
# MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
# GNU General Public License for more details.
#
# You should have received a copy of the GNU General Public
# License
# along with GCstar; if not, write to the Free Software
# Foundation, Inc., 51 Franklin St, Fifth Floor, Boston, MA
# 02110-1301, USA
#
#####

use strict;

use GCPlugins::GCfilms::GCfilmsCommon;

{

    package GCPlugins::GCfilms::GCPluginAllmovie;

    use base qw(GCPlugins::GCfilms::GCfilmsPluginsBase);

    sub start
    {
        my ($self, $tagname, $attr, $attrseq, $origtext) = @_;

        $self->{inside}->{$tagname}++;

        if ($self->{parsingList})
        {
            if ($tagname eq "div" && ($attr->{class} eq "title"))
            {
                $self->{isMovie} = 1;
            }
            elsif ($tagname eq "a" && $self->{isMovie} eq 1)
            {
                $self->{isMovie} = 2;
                $self->{isYear} = 1;

                $self->{itemIdx}++;
                $self->{itemsList}[$self->{itemIdx}]->{url} =
                $attr->{href};
            }
        }
    }
}
```

```
    }
    elseif ($tagname eq "div" && ($attr->{class} eq "artist"))
    {
        $self->{isDirector} = 1;
    }
    elseif (0 eq 1 && $tagname eq "div" && ($attr->{class} eq
"title"))
    {
        $self->{isYear} = 1;
    }
    elseif ($tagname eq "a" && $self->{isDirector} eq 1)
    {
        $self->{isDirector} = 2;
    }
    elseif ($tagname eq "div" && $attr->{ratingValue})
    {
        $self->{isRatingPress} = 1;
    }
    elseif ($tagname eq "div" && $self->{isMovie} eq 1)
    {
        $self->{isMovie} = 2;
        $self->{itemIdx}++;
        $self->{itemsList}[$self->{itemIdx}]->{url} =
$attr->{href};
    }
    elseif ($tagname eq "tr")
    {
        $self->{isFound} = 1;
    }
    elseif ($tagname eq "title")
    {
        $self->{insideHTMLtitle} = 1;
        # trying to be kind on server which sometimes returns 500
HTTP errors
        sleep 1;
    }
    else
    {
        if (($tagname eq "h2") && ($attr->{class} eq "movie-
title"))
        {
            $self->{insideTitle} = 1;
            # trying to be kind on server which sometimes returns 500
HTTP errors
            sleep 1;
        }
        elseif ($tagname eq "span" && $self->{insideCountry} eq 1)
        {
            $self->{insideCountry} = 2;
        }
    }
}
```



```

    }
    elsif ($tagname eq "span" && $self->{insideRating} eq 1)
    {
        $self->{insideRating} = 2;
    }
    elsif ($tagname eq "span" && $self->{insideTime} eq 1)
    {
        $self->{insideTime} = 2;
    }
    elsif ($tagname eq "span" && $self->{insideYearRuntime} eq
1)
    {
        $self->{insideYearRuntime} = 2;
    }
    elsif (($tagname eq "h3") && ($attr->{class} eq
"movie-director"))
    {
        $self->{insideDirector} = 1;
    }
    elsif (($tagname eq "a") && $self->{insideDirector} eq
1)
    {
        $self->{insideDirector} = 2;
    }
    elsif (($tagname eq "span") && ($attr->{class} eq
"header-movie-genres"))
    {
        $self->{insideGenre} = 1;
    }
    elsif (($tagname eq "a") && $self->{insideGenre} eq 1)
    {
        $self->{insideGenre} = 2;
    }
    elsif (($tagname eq "span") && ($attr->{class} eq
"release-year"))
    {
        $self->{insideYear} = 1;
    }
    elsif (($tagname eq "hgroup") && ($attr->{class} eq
"details"))
    {
        $self->{insideLeftSidebarTitle} = 1;
    }
    elsif (($tagname eq "div") && ($attr->{class} eq
"cast_name artist-name"))
    {
        $self->{insideActors} = 1;
    }
    elsif ($self->{insideActors} eq 1 && $tagname eq "a")
    {
        $self->{insideActors} = 2;
    }

```

```
    }
    elseif ($self->{insideActors} eq 2 && $tagname eq "div" &&
$attr->{class} eq "cast_role")
    {
        $self->{insideActors} = 3;
    }
    elseif (($tagname eq "div") && ($attr->{itemprop} eq
"description"))
    {
        $self->{insideSynopsis} = 1;
    }
    elseif (($tagname eq "a") && ($attr->{href} =~
m/\//cast-crew/ ))
    {
        if ($self->{firstPass} eq 1)
        {
            # trigger the load of web page with the list of actors
and roles
            $self->{curInfo}->{nextUrl} =
"http://www.allmovie.com".$attr->{href};
            $self->{firstPass} = 0;
        }
    }
    elseif (
        ($tagname eq "div")
        && ( ($attr->{id} eq "left-sidebar-title")
            || ($attr->{id} eq "left-sidebar-title-
small"))
    )
    {
        $self->{insideLeftSidebarTitle} = 1;
    }
    elseif ($tagname eq "a")
    {
        if ($self->{insideDirectorList})
        {
            $self->{insideDirector} = 1;
        }
        elseif ($self->{nextIsSeries})
        {
            $self->{insideSeries} = 1;
            $self->{nextIsSeries} = 0;
        }
    }
    elseif (($tagname eq "img") && ($attr->{itemprop} eq
"image"))
    {
        $self->{curInfo}->{image} = ($attr->{src});
    }
}
```

```

    }

    sub end
    {
        my ($self, $tagname) = @_;

        $self->{inside}->{$tagname}--;

        if ($tagname eq "div" && $self->{isYear})
        {
            $self->{isYear} = 0;
        }
        elsif ($tagname eq "div" && $self->{insideSynopsis})
        {
            $self->{insideSynopsis} = 0;
        }
    }

    sub text
    {
        my ($self, $origtext) = @_;
        return if ((length($origtext) == 0) || ($origtext eq "
"));

        $origtext =~ s/&#34;/"/g;
        $origtext =~ s/&#179;/3/g;
        $origtext =~ s/&#[0-9]*;//g;
        $origtext =~ s/\\n//g;

        if ($self->{parsingList})
        {
            if ($self->{isMovie} eq 2)
            {
                $self->{itemsList}[$self->{itemIdx}]->{title} =
$origtext;
                $self->{isMovie} = 0;
            }
            elsif ($self->{isYear})
            {
                $origtext =~ s/^\s+\\(*/;
                $origtext =~ s/\\)*\s+$//g;

                $self->{itemsList}[$self->{itemIdx}]->{date} =
$origtext
                # $self->{isYear} = 0;
            }
            elsif ($self->{isDirector} eq 2)
            {
                $self->{itemsList}[$self->{itemIdx}]->{director}
= $origtext;

```

```
        $self->{isDirector} = 0;
    }
}
else
{
    if ($self->{insideTitle})
    {
        # plugin with multiple passes : {curInfo}->{title} is set
        during the first pass
        if (! $self->{curInfo}->{title})
        {
            $self->{firstPass} = 1;
        }

        # Strip leading and tailing spaces
        $origtext =~ s/^\s+//;
        $origtext =~ s/\s+$//g;
        $self->{curInfo}->{title} = $origtext;
        $self->{insideTitle} = 0;
    }
    elsif ($self->{insideDirector} eq 2)
    {
        $origtext =~ s/^\s+//;
        $origtext =~ s/\s+$//g;
        $self->{curInfo}->{director} = $origtext;
        $self->{insideDirector}      = 0;
        $self->{insideDirectorList} = 0;
    }
    elsif ($self->{insideGenre} eq 2)
    {
        my $genre = $self->capWord($origtext);
        if (! ($self->{curInfo}->{genre} =~ m/$genre/))
        {
            $self->{curInfo}->{genre} .= $self->capWord($origtext)
            . ' , ' ;
        }

        $self->{insideGenre} = 0;
    }
    elsif ($self->{insideYear})
    {
        $origtext =~ s/^\s+//;
        $origtext =~ s/\s+$//g;
        $self->{curInfo}->{date} = $origtext;
        $self->{insideYear} = 0;
    }
    elsif ($self->{insideYearRuntime} eq 2)
    {
        $origtext =~ s/^\s+//;
        $origtext =~ s/\s+$//g;
        $self->{curInfo}->{date} = $origtext;
    }
}
```

```

$self->{insideYearRuntime} = 0;

}

elseif ($self->{insideActors} eq 2)
{
    $$self->{curInfo}->{actors} .= $origtext . ', '
    # if ($self->{actorsCounter} <
$GCPlugins::GCfilms::GCfilmsCommon::MAX_ACTORS);
    $$self->{actorsCounter}++;
    $$self->{insideActors} = 0;
    $self->{actor} = $origtext if (! $self->{actor});
}
elseif ($self->{insideActors} eq 3)
{
    $origtext =~ s/^\s*//;
    $origtext =~ s/\s*$//;
    $self->{role} = $origtext;
    push @{$self->{curInfo}->{actors}}, [$self->{actor}];
    push
@{$self->{curInfo}->{actors}->[$self->{actorsCounter}]},
$self->{role};
    $self->{actorsCounter}++;
    $self->{actor} = 0;
    $self->{role} = 0;
    $self->{insideActors} = 0;
}

elseif ($self->{insideSynopsis})
{
    $origtext =~ s/^\s+//;
    $self->{curInfo}->{synopsis} .= $origtext." ";
}
elseif ($self->{insideCountry} eq 2)
{
    $self->{curInfo}->{country} = $origtext;
    $self->{insideCountry} = 0;
}
elseif ($self->{insideTime} eq 2)
{
    $origtext =~ s/\s*min.*//;
    $self->{curInfo}->{time} = $origtext;
    $self->{insideTime} = 0;
}
elseif ($self->{insideRating} eq 2)
{
    $self->{curInfo}->{age} = 1
    if ($origtext eq 'Unrated') || ($origtext eq
'Open');
    $self->{curInfo}->{age} = 2
    if ($origtext eq 'G') || ($origtext eq
'Approved');

```

```
        $self->{curInfo}->{age} = 5
        if ($origtext eq 'PG') || ($origtext eq 'M') ||
($origtext eq 'GP');
        $self->{curInfo}->{age} = 13 if $origtext eq
'PG13';
        $self->{curInfo}->{age} = 17 if $origtext eq 'R';
        $self->{curInfo}->{age} = 18
        if ($origtext eq 'NC17') || ($origtext eq 'X');
        $self->{insideRating} = 0;
    }
elseif ($self->{isRatingPress})
{
    $origtext =~ s/\s//g;
    $self->{curinfo}->{ratingPress} = $origtext * 2;
}

# be careful to keep this test at the end
elseif ($self->{insideLeftSidebarTitle})
{
    if ($origtext eq "Genres")
    {
        $self->{insideGenreList} = 1;
    }
    elseif ($origtext =~ m/Release Date/)
    {
        $self->{insideYearRuntime} = 1;
    }
    elseif ($origtext =~ m/Countries/)
    {
        $self->{insideCountry} = 1;
    }
    elseif ($origtext =~ m/Run Time/)
    {
        $self->{insideTime} = 1;
    }
    elseif ($origtext =~ m/MPAA Rating/)
    {
        $self->{insideRating} = 1;
    }
}
elseif ($origtext =~ /Is part of the series:$/)
{
    $self->{nextIsSeries} = 1;
}

elseif ($self->{insideOtherTitles})
{
    $self->{tempOriginal} = $origtext;
    $self->{tempOriginal} =~ s/\s*$/;
    $self->{tempOriginal} =~ s/^\s*//;
```

```

        $self->{curInfo}->{original} .=
$self->{tempOriginal} . ', ' ;
        $self->{insideOtherTitles} = 0;
    }
    elsif ($self->{insideSeries})
    {
        $self->{curInfo}->{serie} = $origtext;
        $self->{curInfo}->{serie} =~ s/( \[.*\])//;
        $self->{insideSeries} = 0;
    }
}

sub new
{
    my $proto = shift;
    my $class = ref($proto) || $proto;
    my $self = $class->SUPER::new();
    bless($self, $class);

    $self->{hasField} = {
        title    => 1,
        date     => 1,
        director => 1,
        actors   => 0,
    };

    $self->{isInfo}   = 0;
    $self->{isMovie}  = 0;
    $self->{curName}   = undef;
    $self->{curUrl}    = undef;

    return $self;
}

sub preProcess
{
    my ($self, $html) = @_;

    $html =~ s/"&#34;/'"/g;
    $html =~ s/&#34;/'"/g;
    $html =~ s|</a></b><br>|</a><br>;

    return $html;
}

sub getSearchUrl
{
    my ($self, $word) = @_;

```

```
        my $wordFiltered = $word;

        # Allmovie doesn't return correct results if searching
        with a prefix like 'the'
        $wordFiltered =~ s/^(the|a)?[+\s]+[^\s-a-zA-Z0-9]*\s*//i;
        # return ('http://allmovie.com/search/all', ['q' =>
        $wordFiltered, 'submit' => 'SEARCH']);
        return ('http://allmovie.com/search/all/' .
        $wordFiltered);
    }

    sub getItemUrl
    {
        my ($self, $url) = @_;
        return $url if $url =~ /^http:/;
        return "http://allmovie.com" . $url;
    }

    sub getName
    {
        return "Allmovie";
    }

    sub getAuthor
    {
        return 'Zombiepig - Kerenoc';
    }

    sub getLang
    {
        return 'EN';
    }
}

1;
```

GCPlugins/GCPluginsBase.pm

GCPlugins/GCPluginsBase.pm

```
package GCPlugins::GCPluginsBase;

#####
#
# Copyright 2005-2010 Christian Jodar
```



```
#
# This file is part of GCstar.
#
# GCstar is free software; you can redistribute it and/or modify
# it under the terms of the GNU General Public License as
# published by
# the Free Software Foundation; either version 2 of the License,
# or
# (at your option) any later version.
#
# GCstar is distributed in the hope that it will be useful,
# but WITHOUT ANY WARRANTY; without even the implied warranty of
# MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
# GNU General Public License for more details.
#
# You should have received a copy of the GNU General Public
# License
# along with GCstar; if not, write to the Free Software
# Foundation, Inc., 51 Franklin St, Fifth Floor, Boston, MA
# 02110-1301, USA
#
#####

use strict;
use utf8;

{
    package GCPluginParser;
    use base qw(HTML::Parser);
    use LWP::Simple qw($ua);
    use HTTP::Cookies::Netscape;
    use URI::Escape;
    use HTML::Entities;
    use Encode;
    use File::Spec;

    sub new
    {
        my $proto = shift;
        my $class = ref($proto) || $proto;
        my $self = $class->SUPER::new();

        $ua->agent('Mozilla/5.0 (X11; U; Linux i686; en-US;
rv:1.7.5) Gecko/20041111 Firefox/1.0');
        $ua->default_header('Accept-Encoding' => 'x-gzip');
        $ua->default_header('Accept' => 'text/html');
        $self->{ua} = $ua;

        $self->{itemIdx} = -1;
        $self->{itemsList} = ();
    }
}
```

```
        bless ($self, $class);
        return $self;
    }

    sub getItemsNumber
    {
        my ($self) = @_;

        return $self->{itemIdx} + 1;
    }

    sub getItems
    {
        my ($self) = @_;
        return @{$self->{itemsList}};
    }

    sub load
    {
        my $self = shift;

        $self->checkProxy;
        $self->checkCookieJar;

        $self->{itemIdx} = -1;
        $self->{isInfo} = 0;
        $self->{itemsList} = ();

        #my $word = uri_escape_utf8($self->{title});
        my $title2 = encode($self->getSearchCharset,
$self->{title});
        my $word = uri_escape($title2);
        $word =~ s/%20/+/g;

        my $post;
        my $html;

        # For multi-pass plugins, the plugin will have set the url
to load for
        # the next pass as nextUrl. If this doesn't exist, we're
either on the
        # first pass, or only using a one-pass plugin, so call
getSearchUrl
        # to find the url to retrieve
        if ($self->{nextUrl})
        {
            $html = $self->loadPage($self->{nextUrl});
        }
        else
```

```
        {
            $html = $self->loadPage($self->getSearchUrl($word));
        }

return if (length $html eq 0);
$self->{parsingList} = 1;

$html = $self->preProcess($html);

decode_entities($html)
    if $self->decodeEntitiesWanted;
$self->{inside} = undef;

$self->parse($html);

my @noConversion = @{$self->getNotConverted};
foreach my $item (@{$self->{itemsList}})
{
    foreach (keys %{$item})
    {
        next if $_ eq 'url';
        $item->{$_} = $self->convertCharset($item->{$_})
            if ! GCUtils::isArrayTest($_, @noConversion);
    }
}

}

sub loadPage
{
    my ($self, $url, $post, $noSave) = @_;
    my $debugPhase = $ENV{GCS_DEBUG_PLUGIN_PHASE};
    my $debugFile;

    $debugFile =
File::Spec->tmpdir.'/'.GCUtils::getSafeFileName($url)
        if ($debugPhase > 0);
    $self->{loadedUrl} = $url if ! $noSave;
    my $response;
    my $result;
    if ($debugPhase < 2 || (!(-f $debugFile)))
    {
        if ($post)
        {
            $response = $ua->post($url, $post);
        }
        else
        {
            $response = $ua->get($url);
        }
    }
}
```

```
#UnclePetros 03/07/2011:
#code to handle correctly 302 response messages
my $label1 = $response->code;
if($response->code == '301' || $response->code ==
'302'){
    my $location = $response->header("location");
    $response = $ua->get($location);
    $self->{loadedUrl} = $location;
}
elsif ($response->code ne '200')
{
    return "";
}
eval {
    $result = $response->decoded_content;
};
if ($debugPhase == 1 || $debugPhase == 3)
{
    open DEBUG_FILE, ">$debugFile";
    binmode(DEBUG_FILE, ":utf8");
    print DEBUG_FILE ($result || $response->content);
    close DEBUG_FILE;
}
}
else
{
    local $/;
    open DEBUG_FILE, "$debugFile";
    $result = <DEBUG_FILE>;
    utf8::decode($result);
}
return $result || ($response && $response->content);
}

sub capWord
{
    my ($self, $msg) = @_;

    use locale;

    (my $newmsg = lc $msg) =~
s/(\s|,|^)(\w)(\w)(\w*?)/$1U$2\E$3$4/gi;
    return $newmsg;
}

sub getSearchFieldsArray
{
    return [''];
}
```

```
sub getSearchFields
{
    my ($self, $model) = @_;

    my $result = '';
    $result .= $model->getDisplayedLabel($_).', ' foreach
(@{$self->getSearchFieldsArray});
    $result =~ s/, $//;
    return $result;
}

sub hasField
{
    my ($self, $field) = @_;

    return $self->{hasField}->{$field};
}

sub getExtra
{
    return '';
}

# Character set for web page text
sub getCharset
{
    my $self = shift;

    return "ISO-8859-1";
}

# Character set for encoding search term, can sometimes be
different
# to the page encoding, but we default to the same as the page
set
sub getSearchCharset
{
    my $self = shift;

    return getCharset;
}

# For some plugins, we need extra checks to determine if urls
match
# the language the plugin is written for. This allows us to
correctly determine
# if a drag and dropped url is handled by a particular plugin.
If these
# checks are necessary, return 1, and make sure plugin handles
the
# the testURL function correctly
```

```
sub needsLanguageTest
{
    return 0;
}

# Used to test if a given url is handled by the plugin. Only
required if
# needsLanguageTest is true.
sub testURL
{
    my ($self, $url) = @_;
    return 1
}

# Determines whether plugin should be the default plugins
gcstar uses.
# Plugins with this attribute set will appear first in plugin
list,
# and will be highlighted with a star icon. A returned value
of 1
# means the plugin is preferred if it's language matches the
user's language,
# a returned value of 2 mean's it's preferred regardless of
the language.
sub isPreferred
{
    return 0;
}

sub getPreferred
{
    return isPreferred;
}

sub getNotConverted
{
    my $self = shift;
    return [];
}

sub decodeEntitiesWanted
{
    return 1;
}

sub getDefaultPictureSuffix
{
    return '';
```

```
sub convertCharset
{
    my ($self, $value) = @_;

    my $result = $value;
    if (ref($value) eq 'ARRAY')
    {
        foreach my $line(@{$value})
        {
            my $i = 0;
            eval {
                map {$_ = decode($self->getCharset, $_)} @{$line};
            }
        }
    }
    else
    {
        eval {
            $result = decode($self->getCharset, $result);
        };
    }
    return $result;
}

sub getItemInfo
{
    my $self = shift;

    eval {
        $self->init;
    };
    my $idx = $self->{wantedIdx};
    my $url =
$self->getItemUrl($self->{itemsList}[$idx]->{url});
    $self->{curInfo} = {};
    $self->loadUrl($url);

    # multi-pass plugins that requires multiple web page to get
    # all info on a single collection item
    # for example : Allmovie (tabs to get casting), Allocine
    # (idem)
    # the plugin can set {nextUrl} to fetch next web page, the
    # information is cumulative in {curInfo}
    while ($self->{curInfo}->{nextUrl})
    {
        my $nextUrl = $self->{curInfo}->{nextUrl};
        $self->{curInfo}->{nextUrl} = 0;
        $self->loadUrl($nextUrl);
    }
    return $self->{curInfo};
}
```

```
sub changeUrl
{
    my ($self, $url) = @_;

    return $url;
}

sub loadUrl
{
    my ($self, $url) = @_;
    $self->checkProxy;
    $self->checkCookieJar;
    my $realUrl = $self->changeUrl($url);
    my $html = $self->loadPage($realUrl);
    $self->{parsingList} = 0;
    #$html = $self->convertCharset($html);

    # $self->{curInfo} = {} if (!$self->{curInfo}->{title});
    # once the urlField is set don't change it (plugins fetching
    # multiple pages for one item)
    $self->{curInfo}->{$self->{urlField}} = $url if
    (!$self->{curInfo}->{$self->{urlField}});

    $html = $self->preProcess($html);
    decode_entities($html)
        if $self->decodeEntitiesWanted;

    $self->{inside} = undef;
    $self->parse($html);

    my @noConversion = @{$self->getNotConverted};

    foreach (keys %{$self->{curInfo}})
    {
        next if $_ eq $self->{urlField};
        $self->{curInfo}->{$_} =
        $self->convertCharset($self->{curInfo}->{$_})
            if ! GCUtils::isArrayTest($_, @noConversion);
        if (ref($self->{curInfo}->{$_}) ne 'ARRAY')
        {
            $self->{curInfo}->{$_} =~ s/\\|/,/gm;
            $self->{curInfo}->{$_} =~ s/\\r//gm;
            $self->{curInfo}->{$_} =~ s/[ \t]*$/gm;
        }
    }
    my $linkName = $GCMODEL::linkNameSeparator.$self->getName;
    $self->{curInfo}->{$self->{urlField}} .= $linkName
        if (!$self->{curInfo}->{$self->{urlField}} =~
        m/$linkName/));
```



```
        return $self->{curInfo};
    }

    sub setProxy
    {
        my ($self, $proxy) = @_;

        $self->{proxy} = $proxy;
    }

    sub checkProxy
    {
        my $self = shift;
        $ua->proxy(['http'], $self->{proxy});
        $$self->{ua}->proxy(['http'], $self->{proxy});
    }

    sub setCookieJar
    {
        my ($self, $cookieJar) = @_;
        $self->{cookieJar} = $cookieJar;
    }

    sub checkCookieJar
    {
        my $self = shift;
        $ua->cookie_jar(HTTP::Cookies::Netscape->new(
            'file' => "$self->{cookieJar}",
            'autosave' => 1,));
    }

    # Used to set the number of passes the plugin requires
    sub getNumberPasses
    {
        # Most plugins only need to search once, so default to one
        pass
        return 1;
    }

    # Returns undef if it doesn't support search using barcode
    scanner
    sub getEanField
    {
        return undef;
    }
}

1;
```

Le-Livre.com (ou .fr)

En petit bonus, une mise à jour pour le site Le-Livre.com (ou .fr). Attention, leur système de classement est un peu archaïque, et fonctionne encore en ISBN-10.

```
package GCPlugins::GCbooks::GCLeLivre;

#####
#
# Copyright 2005-2006 Tian
# Copyright 2016 Varkolak
#
# This file is part of GCstar.
#
# GCstar is free software; you can redistribute it and/or modify
# it under the terms of the GNU General Public License as published by
# the Free Software Foundation; either version 2 of the License, or
# (at your option) any later version.
#
# GCstar is distributed in the hope that it will be useful,
# but WITHOUT ANY WARRANTY; without even the implied warranty of
# MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
# GNU General Public License for more details.
#
# You should have received a copy of the GNU General Public License
# along with GCstar; if not, write to the Free Software
# Foundation, Inc., 51 Franklin St, Fifth Floor, Boston, MA 02110-1301, USA
#
#####

use strict;
use utf8;

use GCPlugins::GCbooks::GCbooksCommon;

{
    package GCPlugins::GCbooks::GCPluginLeLivre;

    use base qw(GCPlugins::GCbooks::GCbooksPluginsBase);
    use URI::Escape;

    sub start
    {
        my ($self, $tagname, $attr, $attrseq, $origtext) = @_;

        $self->{inside}->{$tagname}++;

        if ($self->{parsingList})
```

```
{

    if ($tagname eq 'book')
    {
        $self->{itemIdx}++;
        $self->{isTitle} = 1;
    }
    elsif ($self->{isTitle})
    {
        $self->{itemsList}[$self->{itemIdx}]->{url} = $attr->{href};
        $self->{itemsList}[$self->{itemIdx}]->{title} =
$attr->{title};
        $self->{isTitle} = 0;
    }
    elsif ($tagname eq 'auth')
    {
        $self->{isAuthor} = 1;
    }
    elsif ($tagname eq 'edito')
    {
        $self->{isPublisher} = 1;
    }
}

else
{
    if ($tagname eq 'image')
    {
        $self->{isImage} = 1;
    }
    elsif (($tagname eq 'a') && ($self->{isImage}))
    {
        $self->{curInfo}->{cover} = 'http://www.le-livre.fr/' .
$attr->{href};
        $self->{isImage} = 0;
    }
    elsif ($tagname eq 'titre')
    {
        $self->{isTitle} = 1;
    }
    elsif ($tagname eq 'isbn')
    {
        $self->{isISBN} = 1;
    }
    elsif ($tagname eq 'commentaires')
    {
        $self->{isComm} = 1;
    }
    elsif ($tagname eq 'auteur')
    {
        $self->{isAuthor} = 1;
    }
}
```

```
    }
    elsif ($tagname eq 'divers')
    {
        $self->{isFormat} = 1;
    }
}

sub end
{
    my ($self, $tagname) = @_;

    $self->{inside}->{$tagname}--;
}

sub text
{
    my ($self, $origtext) = @_;

    $origtext =~ s/^\s+//;
    $origtext =~ s/\s+$//g;

    if ($self->{parsingList})
    {
        if (($self->{isAuthor}) && ($origtext ne ''))
        {
            $self->{itemsList}[$self->{itemIdx}]->{authors} = $origtext;
            $self->{isAuthor} = 0 ;
        }
        elsif (($self->{isPublisher}) && ($origtext ne ''))
        {
            my @array = split(/\./,$origtext);
            $self->{itemsList}[$self->{itemIdx}]->{edition} = $array[0];
            $self->{itemsList}[$self->{itemIdx}]->{publication} =
$array[1];

            $self->{itemsList}[$self->{itemIdx}]->{format} = $array[2];
            $self->{isPublisher} = 0 ;
        }
    }

    else
    {
        if ($self->{isTitle})
        {
            $self->{curInfo}->{title} = $origtext;
            $self->{isTitle} = 0;
        }
        elsif ($self->{isISBN})
        {
            $self->{curInfo}->{isbn} = $origtext;
        }
    }
}
```

```

        $self->{isISBN} = 0;
    }
    elsif (($self->{isComm}) && ($origtext ne ''))
    {
        $self->{curInfo}->{description} = $origtext;
        my @array = split(/\./,$origtext);
    for my $i (0 .. $#array)
    {
        if ($array[$i] =~ s/[0-9]-//i)
        {
            # $array[$i] =~ s/-//i;
            $array[$i] =~ s/^\s+//;
            $self->{curInfo}->{genre} = $array[$i];
        }
    }

        $self->{isComm} = 0 ;
    }
    elsif ($self->{isAuthor})
    {
        my @array = split(/-/,$origtext);
        $array[0] =~ s/([\w']+)/\u\L$1/g;
        $self->{curInfo}->{authors} = $array[0];
    for my $i (1 .. $#array)
    {
        $array[$i] =~ s/([\w']+)/\u\L$1/g;
        $self->{curInfo}->{authors} .= ", " . $array[$i];
    }

        $self->{isAuthor} = 0 ;
    }
    elsif (($self->{isFormat}) && ($origtext ne ''))
    {
        my @array = split(/\./,$origtext);
        $array[0] =~ s/([\w']+)/\u\L$1/g;
        $self->{curInfo}->{publisher} = $array[0];
        $array[1] =~ s/^\s+//;
        $self->{curInfo}->{publication} = $array[1];
        $array[2] =~ s/^\s+//;
        $self->{curInfo}->{format} = $array[2];
    for my $i (3 .. $#array)
    {
        if ($array[$i] =~ /pages/i)
        {
            $array[$i] =~ s/ pages.*//i;
            $array[$i] =~ s/^\s+//;
            $self->{curInfo}->{pages} = $array[$i];
        }
    }

        $self->{isFormat} = 0 ;
    }
}
}
}

```

sub new

```
{
    my $proto = shift;
    my $class = ref($proto) || $proto;
    my $self = $class->SUPER::new();
    bless ($self, $class);

    $self->{hasField} = {
        title => 1,
        authors => 1,
        publication => 1,
        format => 1,
        edition => 1,
        serie => 0,
    };

    $self->{isImage} = 0;
    $self->{isTitle} = 0;
    $self->{isAuthor} = 0;
    $self->{isPublisher} = 0;
    $self->{isISBN} = 0;
    $self->{isFormat} = 0;
    $self->{isTranslator} = 0;
    $self->{isComm} = 0;

    return $self;
}
```

sub preProcess

```
{
    my ($self, $html) = @_ ;

    if ($self->{parsingList})
    {
        $html =~ s|<td class="illustration" alt="">|<book>|gi;
        $html =~ s|<td class="auteur">|<auth>|gi;
        $html =~ s|"ref="|"|gi;
        $html =~ s|<td class="caracteristique">|<edito>|gi;
        $html =~ s|<b>|gi;
        $html =~ s|</b>|gi;
    }

    else
    {
        $html =~ s|<div class="watermark">|<image>|i;
        $html =~ s|<h1 class="FicheTitre" itemprop="name">|<titre>|i;
        $html =~ s|<span class="FicheDetailISBN" >|<isbn>|i;
        $html =~ s|<span class="FicheDetailAuteur" >|<auteur>|i;
        $html =~ s|<h2 class="TitreFicheDescription"> Description
```

```
</h2>|<infos>|i;
    $html =~ s|<h2 class="TitreFicheDescription"> Informations
Supplémentaires </h2>|<commentaires>|i;
    $html =~ s|<br /> <br />|<divers>|i;

    $html =~ s|\x{92}|'|g;
    $html =~ s|&#146;|'|gi;
    $html =~ s|&#149;|*|gi;
    $html =~ s|&#133;|...|gi;
    $html =~ s|\x{85}|...|gi;
    $html =~ s|\x{8C}|OE|gi;
    $html =~ s|\x{9C}|oe|gi;

}

return $html;
}

sub getSearchUrl
{
    my ($self, $word) = @_;

    return
"http://www.le-livre.fr/default.asp?Rech=1&Submit_Rech_Rapide=1&rechercherap
=" . $word;

}

sub getItemUrl
{
    my ($self, $url) = @_;

    return $url;
}

sub getName
{
    return "Le-Livre";
}

sub getCharset
{
    my $self = shift;
    return "ISO-8859-15";
}

sub getAuthor
{
    return 'Varkolak';
}
```

```
sub getLang
{
    return 'FR';
}

sub getSearchFieldsArray
{
    return ['ISBN', 'title', 'author', 'publication'];
}

1;
```

GCgames/GCJeuxVideoCom.pm

En quête de perfection smile je n'ai pu m'empêcher d'aller voir le plugin GCgames/GCJeuxVideoCom.pm. J'ai juste modifié la classe de 2 tags HTML et cela semble un peu mieux marcher (jeux multi-supports, titre, secrets). Je n'ai pas fait de beaucoup de tests car je ne gère pas avec GCstar ce type de collection.

```
package GCPlugins::GCgames::GCJeuxVideoCom;

#####
#
# Copyright 2005-2015 Tian
#
# This file is part of GCstar.
#
# GCstar is free software; you can redistribute it and/or modify
# it under the terms of the GNU General Public License as published by
# the Free Software Foundation; either version 2 of the License, or
# (at your option) any later version.
#
# GCstar is distributed in the hope that it will be useful,
# but WITHOUT ANY WARRANTY; without even the implied warranty of
# MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
# GNU General Public License for more details.
#
# You should have received a copy of the GNU General Public License
# along with GCstar; if not, write to the Free Software
# Foundation, Inc., 51 Franklin St, Fifth Floor, Boston, MA 02110-1301, USA
#
#####

use strict;
use utf8;
```



```
use GCPlugins::GCgames::GCgamesCommon;

{
    package GCPlugins::GCgames::GCPluginJeuxVideoCom;

    use base 'GCPlugins::GCgames::GCgamesPluginsBase';

    sub decryptUrl
    {
        my ($self, $src) = @_;
        my $n = '0A12B34C56D78E9F';
        my $res = 'http://www.jeuxvideo.com';
        my $s = reverse $src;
        my ($c, $l);
        while (length $s)
        {
            $l = index $n, chop $s;
            $c = index $n, chop $s;
            my $car = $l * 16 + $c;
            $res .= chr $car;
        }
        return $res;
    }

    sub loadMultipleResults
    {
        my ($self, $url) = @_;
        my $page = $self->loadPage($url);
        $page =~ /<div\s+class="game-top-version-dispo">(.*?)</div>/s;
        my $tabs = $1;
        $page =~ /<strong>Sortie\s+France\s+:\s+<\/strong>(.*?)i;
        my $released = $1;
        $page =~ /<h1\s+class="highlight">(.*?)<\/h1>/i;
        my $name = $1;
        $name =~ s/&#039;/'/g;
        my @lines = split /\n/, $tabs;
        foreach my $line (@lines)
        {
            if ($line =~ /href="([^\"]*)".*?>([0-9a-zA-Z_. ]*)<\/a>/)
            {
                my $url = $1;
                my $platform = $2;
                $self->{itemIdx}++;
                $self->{itemsList}[$self->{itemIdx}]->{url} =
                'http://www.jeuxvideo.com/' . $url;
                $self->{itemsList}[$self->{itemIdx}]->{name} = $name;
                $self->{itemsList}[$self->{itemIdx}]->{platform} =
                $platform;
                $self->{itemsList}[$self->{itemIdx}]->{released} =
                $released;
            }
        }
    }
}
```

```
    }
}

sub start
{
    my ($self, $tagname, $attr, $attrseq, $origtext) = @_;
    $self->{inside}->{$tagname}++;
    if ($self->{parsingList})
    {
        if ($tagname eq 'span')
        {
            if (($attr->{class} =~ /JvCare\s+([0-9A-F]*)\s+lien-jv/) &&
($attr->{title} ne ""))
            {
                my $url = $self->decryptUrl($1);
                if (! exists $self->{urls}->{$url})
                {
                    if ($url =~ /\$/)
                    {
                        #If it ends with a /, it means it's a multi-
platform game, and the link points to a common page
                        $self->loadMultipleResults($url);
                        $self->{urls}->{$url} = 1;
                    }
                    else
                    {
                        $self->{itemIdx}++;
                        $self->{itemsList}[$self->{itemIdx}]->{url} =
$url;

                        $self->{isGame} = 1;
                        # Note : some game's name contains '-' => not
                        $self->{isName} = 1;

                        my @array = split(/-/, $attr->{title});
                        if (scalar(@array) ge 3 )
                        {
                            if (!($array[$#array] =~ /date/i))
                            {
                                $self->{itemsList}[$self->{itemIdx}]->{released} = $array[$#array];
                            }
                        }

                        $self->{urls}->{$url} = 1;
                    }
                }
            }
            return if !$self->{isGame};
            if ($attr->{class} =~ /recherche-aphabetique-item-machine/)
            {
```

```

        $self->{isPlatform} = 1;
    }
}
}
elseif ($self->{parsingTips})
{
#    if ($attr->{class} eq 'rubrique-asl collapsed')
    if ($attr->{class} eq 'rubrique-asl')
    {
        $self->{isTip} = 1;
    }
    elseif (($tagname eq 'tpfdebuttpf') && ($self->{isTip} eq 2))
    {
        $self->{isTip} = 3;
    }
    elseif ( ($tagname eq 'p') || ($tagname eq 'h2') || ($tagname
eq 'h3')) && (($self->{isTip} eq 3) || ($self->{isTip} eq 4)) )
    {
        $self->{curInfo}->{secrets} .= "\n" if
$self->{curInfo}->{secrets};
    }
    elseif (($tagname eq 'tpffintpf') && ($self->{isTip} ne 0))
    {
        $self->{isTip} = 2;
    }
    elseif ($tagname eq 'head')
    {
        $self->{isTip} = 0;
        $self->{urlTips} = '';
    }
}
else
{
    if ($tagname eq 'span')
    {
        if ($attr->{class} =~ 'label-support active')
        {
            $self->{is} = 'platform';
        }
        elseif ($attr->{itemprop} eq 'description')
        {
            $self->{is} = 'description';
        }
        elseif ($attr->{itemprop} eq 'genre')
        {
            $self->{is} = 'genre';
        }
        elseif ($attr->{class} eq 'recto-jaquette actif')
        {
            $self->{is} = 'boxpic';
        }
    }
}
}

```

```
    }
    elseif ($attr->{class} eq 'verso-jaquette actif')
    {
        $self->{is} = 'backpic';
    }
    elseif (($attr->{'data-modal'} eq 'image') && $self->{is})
    {
        $self->{curInfo}->{$self->{is}} = 'http:'. $attr->{'data-
selector'};
        $self->{is} = '';
    }
}
elseif ($tagname eq 'div')
{
    if ($attr->{class} eq 'game-top-title')
    {
        $self->{is} = 'name';
    }
    elseif ($attr->{class} eq 'bloc-note-redac')
    {
        $self->{is} = 'ratingpress';
    }
    elseif ($attr->{class} eq 'bloc-img-fiche')
    {
        $self->{is} = 'screenshot1';
    }
    elseif ($attr->{class} eq 'bloc-all-support')
    {
        $self->{curInfo}->{exclusive} = 0;
    }
}
elseif ($tagname eq 'img')
{
    if ($self->{is} =~ /screenshot/)
    {
        (my $src = 'http:'. $attr->{src}) =~ s/images-sm/images/;
        $self->{curInfo}->{$self->{is}} = $src;
        if ($self->{is} eq 'screenshot1')
        {
            $self->{is} = 'screenshot2';
        }
        else
        {
            $self->{is} = '';
        }
    }
}
elseif (($tagname eq 'h2') && ($attr->{class} =~ /titre-bloc/))
{
    $self->{isTip} = 1;
}
```

```

    }
    elsif (($self->{isTip} eq 2) && ($attr->{href} =~ /wiki/i))
    {
        $self->{urlTips} = "http://www.jeuxvideo.com/" .
$attr->{href};
        $self->{isTip} = 0;
    }
}

sub end
{
    my ($self, $tagname) = @_;

    $self->{inside}->{$tagname}--;
}

sub text
{
    my ($self, $origtext) = @_;

    if ($self->{parsingList})
    {
        return if !$self->{isGame};
        if ($self->{isPlatform})
        {
            if ($self->{itemsList}[$self->{itemIdx}]->{platform} eq "" )
            {
                # Enleve le " - " présent en début de chaîne
                $origtext =~ s/- //;
                $self->{itemsList}[$self->{itemIdx}]->{platform} =
$origtext;
            }
            $self->{isPlatform} = 0;
        }
        elsif ($self->{isName})
        {
            # Enleve les blancs en debut de chaine
            $origtext =~ s/^\s+//;
            # Enleve les blancs en fin de chaine
            $origtext =~ s/\s+$//;
            $self->{itemsList}[$self->{itemIdx}]->{name} = $origtext;
            $self->{isName} = 0;
        }
    }
    elsif ($self->{parsingTips})
    {
        # Enleve les blancs en debut de chaine
        $origtext =~ s/^\s+//;
        # Enleve les blancs en fin de chaine
        $origtext =~ s/\s+$//;
    }
}

```

```
# There are problems with some texts if ended blanks are removed
if ($self->{isTip} eq 1)
{
    $origtext =~ s|playstation 3|ps3|gi;
    $origtext =~ s|playstation 4|ps4|gi;
    $origtext =~ s|playstation|ps1|gi;
    $origtext =~ s|wii u|wiiu|gi;
    $origtext =~ s|playstation portable|PSP|gi;
    $origtext =~ s|gameboy advance|GBA|gi;
    $origtext =~ s|Super Nintendo|SNES|gi;
    $origtext =~ s|n-gage|NGAGE|gi;
    $origtext =~ s|Nintendo 64|N64|gi;
    $origtext =~ s|Master system|MS|gi;
    $origtext =~ s|Game Gear|G.GEAR|gi;
    if ($origtext =~ /$self->{curInfo}->{platform}/i)
    {
        $self->{isTip} = 2;
    }
    else
    {
        $self->{isTip} = 0;
    }
}
elseif ($self->{isTip} eq 4)
{
    $self->{curInfo}->{secrets} .= $origtext;
}
elseif ($self->{isTip} eq 3)
{
    chomp($origtext);
    if ( ($self->{curInfo}->{secrets}) && ($origtext ne "") )
    {
        $self->{curInfo}->{secrets} .= "\n\n"
    }
    $self->{curInfo}->{secrets} .= $origtext;
    $self->{isTip} = 4;
}
}
else
{
    $origtext =~ s/^\s*//;
    if ($self->{is} && $origtext)
    {
        if ($self->{is} eq 'genre')
        {
            $self->{curInfo}->{$self->{is}} .= "$origtext,";
        }
        else
        {
            $self->{curInfo}->{$self->{is}} = $origtext;
        }
    }
}
```

```

    }
    $self->{curInfo}->{$self->{is}} =~ s/Non/1/i if $self->{is}
eq 'players';
    $self->{curInfo}->{$self->{is}} =
int($self->{curInfo}->{$self->{is}} / 2) if $self->{is} eq 'ratingpress';
    $self->{is} = '';
}
else
{
    if ($self->{isTip} eq 1)
    {
        if (($origtext =~ /wiki/i) || ($origtext =~ /etajv/i))
        {
            $self->{isTip} = 2;
        }
        else
        {
            $self->{isTip} = 0;
        }
    }
    elsif ($origtext eq 'Editeur(s) / D veloppeur(s) : ')
    {
        $self->{is} = 'editor';
    }
    elsif ($origtext =~ /^s*\|s*$/)
    {
        $self->{is} = 'developer' if !
$self->{curInfo}->{developer};
    }
    elsif ($origtext eq 'Sortie France : ')
    {
        $self->{is} = 'released';
    }
    elsif ($origtext eq 'Nombre maximum de joueurs : ')
    {
        $self->{is} = 'players';
    }
}
}
}

sub getTipsUrl
{
    my $self = shift;
    return $self->{urlTips};
}

sub new
{
    my $proto = shift;
    my $class = ref($proto) || $proto;

```

```
my $self = $class->SUPER::new();
bless ($self, $class);

$self->{hasField} = {
    name => 1,
    platform => 1,
    released => 1
};

$self->{isTip} = 0;
$self->{urlTips} = "";

return $self;
}

sub preProcess
{
    my ($self, $html) = @_;
    if ($self->{parsingList})
    {
        $self->{isGame} = 0;
        $self->{isName} = 0;
        $self->{isReleased} = 0;
        $self->{isPlatform} = 0;
        $self->{urls} = {};
        $html =~ s/<\/?b>//ge;
    }
    elsif ($self->{parsingTips})
    {
        $html =~ s|<a data-jvcode="HTMLBLOCK"
href="(.)">|$self->RecupTips("http://www.jeuxvideo.com/" . $1)|ge;
        $html =~ s|Chargement du lecteur vid(.)o...|<p>"Une video est
disponible"</p>|gi;
        $html =~ s||$2|gi;
    }
    else
    {
        $self->{is} = '';
        $self->{curInfo}->{exclusive} = 1;
    }
    return $html;
}

sub RecupTips
{
    my ($self, $url) = @_;

    my $html = $self->loadPage($url);
```



```
my $found = index($html,"<h2 class=\"titre-bloc\">");
if ( $found >= 0 )
{
    $html = substr($html, $found +length('<h2 class="titre-
bloc">'),length($html)- $found -length('<h2 class="titre-bloc">'));
    $found = index($html,"<div class=\"bloc-lien-revision\">");
    if ( $found >= 0 )
    {
        $html = substr($html, 0, $found);
    }
}

return "<tpfdebutpf>" . $html . "<tpffintpf>";
}

sub getSearchUrl
{
    my ($self, $word) = @_ ;
    $word =~ s/\+/ /g;
    return 'http://www.jeuxvideo.com/recherche.php?q=' . $word . '&m=9' ;
}

sub getItemUrl
{
    my ($self, $url) = @_ ;

    return $url if $url;
    return 'http://www.jeuxvideo.com/';
}

sub getName
{
    return 'jeuxvideo.com';
}

sub getAuthor
{
    return 'Tian & TPF';
}

sub getLang
{
    return 'FR';
}

sub isPreferred
{
    return 1;
}
}
```

1;

Conclusion

Le tuto existe déjà http://wiki.gcstar.org/fr/websites_plugins. De plus Varkolak a aussi publié quelques conseils plus tôt dans ce même fil de discussion.

Problèmes connus

Voir aussi

- (fr) <https://forum.ubuntu-fr.org/viewtopic.php?id=1919911>

Basé sur « [Article](#) » par Auteur.

From:

<http://doc.nfrappe.fr/> - **Documentation du Dr Nicolas Frappé**

Permanent link:

<http://doc.nfrappe.fr/doku.php?id=tutoriel:bureautique:gcstar:internet:start>

Last update: **2022/11/08 19:40**

