BàO 1 : extraction de texte dans une arborescence de fils RSS

Sommaire

Objectifs

- Extraire le texte avec des méthodes « rustiques » : les expressions régulières (cf solution 1)
- Extraire le texte avec des outils adaptés : la bibliothèque XML ::RSS (cf solution 2)
- Extraire le texte avec des outils adaptés : la bibliothèque XML ::XPath (cf solution 3)
- Intégrer ces traitements dans le programme de parcours d'une arborescence de fils RSS
- Préparer au moins 2 types de sortie : texte brut (sortie pour étiquetage avec Cordial), texte structuré en XML

1. Problèmes à résoudre

- Problèmes d'encodage : les fils en entrée ne sont pas tous encodés de la même manière (utf-8, iso-8859-1…)
- Certains fils contiennent des scories d'encodage…
- Les fils RSS ont une structure XML commune mais pour certains le fichier est écrit sur une seule ligne, pour d'autres sur plusieurs lignes…
- Une des sorties doit être en XML : s'assurer de construire du xml bien formé (on pourra aussi construire une feuille de style pour afficher ou extraire certains contenus…)
- Etc.

2. Exemple de fils RSS disponibles

2.1 Exemple n°1 : (extrait)

```
<?xml version='1.0' encoding='UTF-8'?><?xml-stylesheet type='text/xsl'
href='http://rss.feedsportal.com/xsl/fr/rss.xsl'?>

<rss xmlns:itunes="http://www.itunes.com/dtds/podcast-1.0.dtd"
xmlns:dc="http://purl.org/dc/elements/1.1/"
xmlns:taxo="http://purl.org/rss/1.0/modules/taxonomy/"
xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
version="2.0"><channel><title>Le Monde.fr : à la
Une</title><link>http://www.lemonde.fr</link><description>Toute l'actualité au
moment de la connexion</description><language>en</language><copyright>Copyright
Le Monde.fr</copyright><pubDate>Wed, 28 Jan 2009 17:39:52
GMT</pubDate><lastBuildDate>Wed, 28 Jan 2009 17:39:52
GMT</lastBuildDate><ttl>30</ttl><image><title
/><url>http://medias.lemonde.fr/mmpub/img/lgo/lemondefr_rss.gif</url><link>http:
//www.lemonde.fr</link></image><item><title>Benoît XVI tente de calmer la
polémique sur l'évêque négationniste
réintégré</title><link>http://www.lemonde.fr/europe/article/2009/01/28/benoit-
xvi-tente-de-calmer-la-polemique-sur-l-eveque-negationniste-
reintegre_1147816_3214.html#xtor=RSS-3208</link><description>Le pape a exprimé
sa "solidarité totale et incontestable" avec le peuple juif lors de son audience
générale hebdomadaire, mercredi. Quelques jours après sa décision de lever
l'excommunication de quatre évêques intégristes, parmi lesquels Mgr Williamson,
auteur de propos négationnistes.&lt;img width='1' height='1'
src='http://rss.feedsportal.com/c/205/f/3050/s/2e7d602/mf.gif'
border='0'/&gt;&lt;br/&gt;&lt;br/&gt…
```

Encodage -> UTF-8

Le fichier est écrit sur 2 lignes

Le contenu de la balise description contient des balises de lien, d'image…

2.2 Exemple n°2 : (extrait)

```
<?xml version="1.0" encoding="iso-8859-1"?><rss version="2.0"
xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#">

<channel>

<title>Examens 2010 - Le Monde.fr</title>

<link>http://www.lemonde.fr/web/sequence/0,2-3404,1-0,0.html</link>

<description>Toute l'actualité au moment de la connexion</description>

<copyright>Copyright Le Monde.fr</copyright>

<image><url>http://medias.lemonde.fr/mmpub/img/lgo/lemondefr_rss.gif</url><title>Le
Monde.fr</title><link>http://www.lemonde.fr</link></image>

<pubDate>Wed, 21 Apr 2010 16:43:13 GMT</pubDate>    <item>

<title> Grande-Bretagne : Gordon Brown passe avec difficulté l'épreuve du débat télévisé
</title>

<link>http://www.lemonde.fr/europe/article/2010/04/15/grande-bretagne-gordon-brown-passe-
avec-difficulte-l-epreuve-du-debat-televise_1334435_3214.html#xtor=RSS-3404</link>

<description>Pour la première fois dans l&#38;#39;histoire politique britannique, un débat
télévisé opposait les trois principaux candidats aux élections législatives du 6
mai.</description>

<pubDate>Thu, 15 Apr 2010 21:40:56 GMT</pubDate>

<guid isPermaLink="false">http://www.lemonde.fr/europe/article/2010/04/15/grande-bretagne-
gordon-brown-passe-avec-difficulte-l-epreuve-du-debat-televise_1334435_3214.html#xtor=RSS-
3404</guid>

<enclosure
url="http://medias.lemonde.fr/mmpub/edt/ill/2010/04/21/h_1_ill_1334434_29cc_cameron.jpg"
type="image/jpeg" length="2439"></enclosure>

</item>
```

Encodage -> iso8859

Le fichier est écrit sur plusieurs lignes

Le contenu de la balise description contient des entités…

3. Scriptage en Perl

3.1 Solution n°1

```
use Unicode::String qw(utf8);
my $encodagesortie="utf-8";
my $encodage=`file -i $ARGV[0] | cut -d= -f2`;
print "ENCODAGE : $encodage \n";
chomp($encodage);
open(FILE,"<:encoding($encodage)", $ARGV[0]);
open(OUT1,">:encoding($encodagesortie)","sortie-textebrut.txt");
open(OUT2,">:encoding($encodagesortie)","sortie-textexml.xml");
print OUT2 "<?xml version=\"1.0\" encoding=\"$encodagesortie\" ?>\n";
print OUT2 "<file>\n";
print OUT2 "<name>$ARGV[0]</name>\n";
my $texte="";
while (my $ligne=<FILE>) {
       #print $ligne;
       $ligne =~ s/\n//g;
       $texte .= $ligne;
}
$texte =~ s/> *</></g;
$texte=~/<pubDate>([^<]+)<\/pubDate>/;
my $date=$1;
if (uc($encodage) ne "UTF-8") {utf8($date);}
print OUT2 "<date>".$date."</date>\n";
print OUT2 "<items>\n";
while ($texte =~
/<item><title>(.+?)<\/title>.+?<description>(.+?)<\/description>/g) {
    my $titre=$1;
    my $resume=$2;
    if (uc($encodage) ne "UTF-8") {utf8($titre);utf8($resume);}
    $titre = &nettoietexte($titre);
    $resume = &nettoietexte($resume);
    print OUT1 "Titre : $titre \n";
    print OUT1 "Resume : $resume \n";;
    print OUT2 "<item><title>$titre</title><abstract>$resume</abstract></item>\n";
}
print OUT2 "</items>\n</file>\n";
close(OUT1);
close(OUT2);
close(FILE);
exit;
#-----------------------------------------------------
sub nettoietexte {
    my $texte=shift;
    $texte =~ s/&lt;/</g;
    $texte =~ s/&gt;/>/g;
    $texte =~ s/<a href[^>]+>//g;
    $texte =~ s/<img[^>]+>//g;
    $texte =~ s/<\/a>//g;
    $texte =~ s/&#38;#39;/'/g;
    $texte =~ s/&#38;#34;/"/g;
    $texte =~ s/<[^>]+>//g;
    return $texte;
}
```

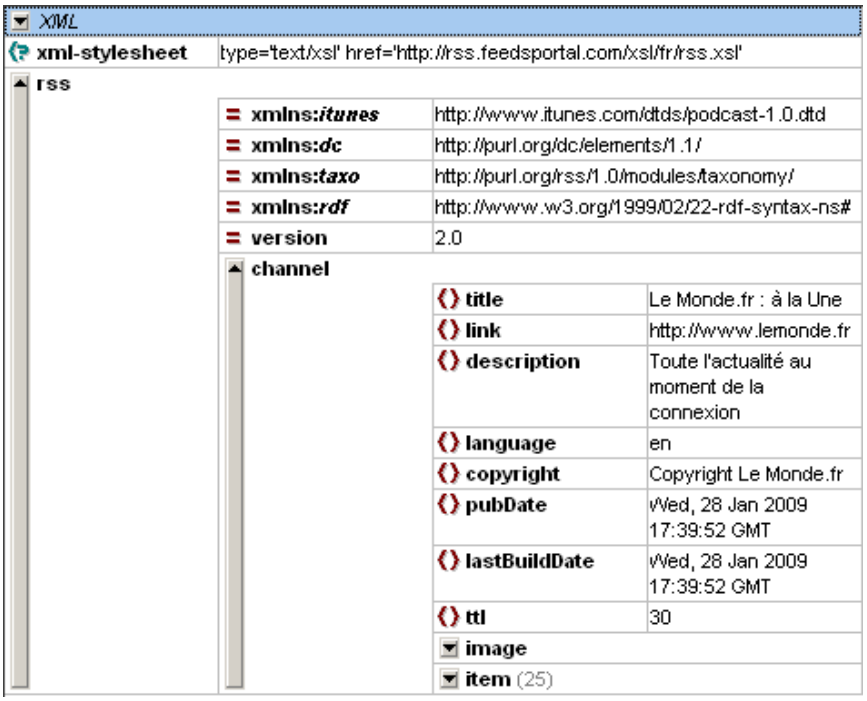
3.2 Solution n°2

```
#!/usr/bin/perl
use XML::RSS;
use Unicode::String qw(utf8);
#-------------------------------------------------------------
my $encodagesortie="utf-8";
my $encodage=`file -i $ARGV[0] | cut -d= -f2`;
open(OUT1,">:encoding($encodagesortie)","sortie-textebrut-avec-xmlrss.txt");
open(OUT2,">:encoding($encodagesortie)","sortie-textexml-avec-xmlrss.xml");
print OUT2 "<?xml version=\"1.0\" encoding=\"$encodagesortie\" ?>\n";
print OUT2 "<file>\n";
print OUT2 "<name>$ARGV[0]</name>\n";

#-------------------------------------------------------------
my $file="$ARGV[0]";
my $rss=new XML::RSS;
#-------------------------------------------------------------
eval {$rss->parsefile($file); };
if( $@ ) {
    $@ =~ s/at \/.*?$//s;               # remove module line number
    print STDERR "\nERROR in '$file':\n$@\n";
}
else {
    my $date=$rss->{'channel'}->{'pubDate'};
    print OUT2 "<date>$date</date>\n";
    print OUT2 "<items>\n";
    foreach my $item (@{$rss->{'items'}}) {
       my $titre=$item->{'title'};
       my $resume=$item->{'description'};
       $titre=&nettoietexte($titre);
       $resume=&nettoietexte($resume);
       if (uc($encodage) ne "UTF-8") {utf8($titre);utf8($resume);}
       print OUT1 "Titre : $titre \n";
       print OUT1 "Resume : $resume \n";;
       print OUT2
"<item><title>$titre</title><abstract>$resume</abstract></item>\n";
    }
}
#-------------------------------------------------------------
print OUT2 "</items>\n</file>\n";
close(OUT1);
close(OUT2);
close(FILE);
exit;
#-------------------------------------------------------------
#-------------------------------------------------------------
sub nettoietexte {
    my $texte=shift;
    $texte=~s/&#39;/'/g;
    $texte=~s/&#34;/"/g;
    $texte =~ s/&lt;/</g;
    $texte =~ s/&gt;/>/g;
    $texte =~ s/<a href[^>]+>//g;
    $texte =~ s/<img[^>]+>//g;
    $texte =~ s/<\/a>//g;
    $texte =~ s/&#38;#39;/'/g;
    $texte =~ s/&#38;#34;/"/g;
    $texte =~ s/<[^>]+>//g;
    return $texte;
}
```

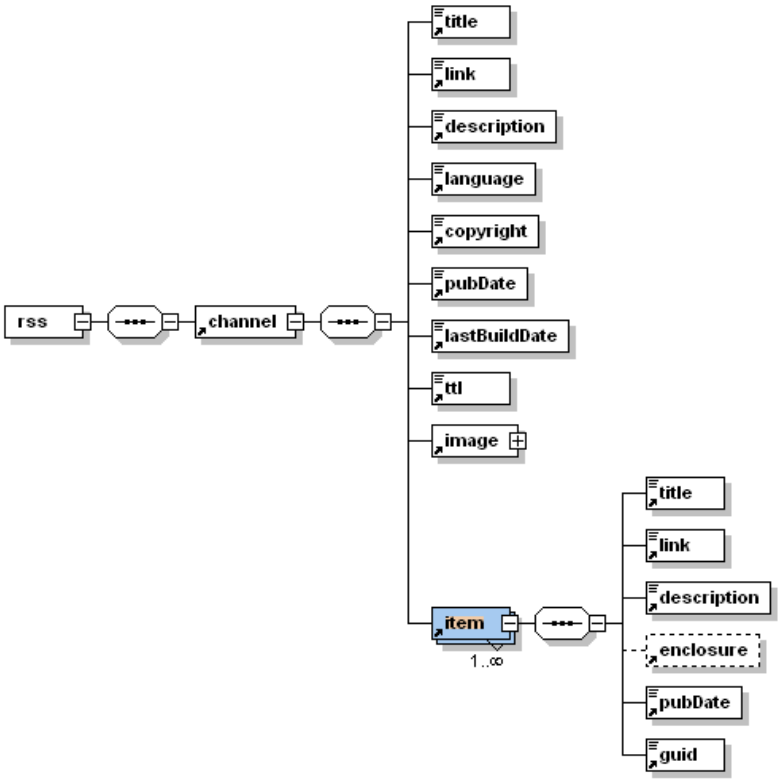
3.3 Solution n°3

Un fil RSS est un document structuré au format XML :



XML		
xml-stylesheet	type='text/xsl' href='http://rss.feedsportal.com/xsl/fr/rss.xsl'	
rss		
	xmlns:*itunes*	http://www.itunes.com/dtds/podcast-1.0.dtd
	xmlns:*dc*	http://purl.org/dc/elements/1.1/
	xmlns:*taxo*	http://purl.org/rss/1.0/modules/taxonomy/
	xmlns:*rdf*	http://www.w3.org/1999/02/22-rdf-syntax-ns#
	version	2.0
	channel	
		title: Le Monde.fr : à la Une
		link: http://www.lemonde.fr
		description: Toute l'actualité au moment de la connexion
		language: en
		copyright: Copyright Le Monde.fr
		pubDate: Wed, 28 Jan 2009 17:39:52 GMT
		lastBuildDate: Wed, 28 Jan 2009 17:39:52 GMT
		ttl: 30
		image
		item (25)

Ce document peut donc être associé à une représentation arborescente :



XPath (source wikipédia) http://fr.wikipedia.org/wiki/XPath

XPath est un langage (non XML) pour localiser une portion d'un document XML. Initialement créé pour fournir une syntaxe et une sémantique aux fonctions communes à XPointer et XSL, XPath a rapidement été adopté par les développeurs comme langage d'interrogation simple d'emploi.

Une expression XPath est un *chemin de localisation*, constitué de *pas de localisation*. Les pas de localisation sont séparés par le caractère « / ». Un chemin ressemble ainsi au chemin dans un système de fichiers.

Les pas de localisation ont chacun trois composants :

1. un axe ;
2. un test de nœud ;
3. des prédicats.

L'axe indique la direction dans laquelle se déplacer dans l'arbre XML, relativement au nœud courant ou depuis la racine. Par exemple, child:: sélectionnera les nœuds enfants du nœud courant. Dans XPath, quand l'axe n'est pas précisé, il s'agit implicitement de l'axe des enfants (child::). Un autre axe largement utilisé est celui des attributs, représenté avec le caractère arobase (@). Il existe en tout 13 axes qui permettent d'exprimer des relations généalogiques, ou qui considèrent l'ordre de lecture du document.

Le test de nœud permet de sélectionner ou non les nœuds en fonction de leur nom ou de leur type. Par exemple le test text() sélectionnera tous les nœuds de type texte (dans l'axe considéré).

Les prédicats sont des expressions plus complexes ; ils sont utilisés pour filtrer les nœuds sélectionnés par l'axe et le test de nœud. Les prédicats sont écrits entre crochets (« [», «] »). Si le prédicat est évalué à vrai, les nœuds correspondants seront sélectionnés.

XPath offre ainsi une recherche séquentielle par nœuds. Le résultat de l'évaluation d'une expression XPath est une séquence contenant des nœuds et des valeurs atomiques (textes, booléens...).

En fonction de la nature (nombre, booléen, texte) des valeurs sélectionnées, XPath offre un certain nombre de fonctions. Ces fonctions sont limitées car elles sont plus destinées à être utilisées dans les prédicats que pour effectuer un traitement sur les données sélectionnées.

Les fonctions qui s'appliquent aux nombres les plus utilsées sont : sum(), count() et les opérateurs arithmétiques. Les fonctions qui s'appliquent aux chaines les plus utilisées sont : subtring(), string-length(), concat().

Considérons le document XML suivant :

```
<?xml version="1.0"?>
<racine>
 <encyclopedie nom="Wikipedia" site="http://fr.wikipedia.org/">
  <article nom="XPath"></article>
  <auteurs>
   <auteur>
    <nom>Dupont</nom>
   </auteur>
   <auteur>
    <nom>Dubois</nom>
   </auteur>
  </auteurs>
 </encyclopedie>
</racine>
```

alors les expressions XPath suivantes

Expression XPath	Résultat
/	sélectionne un nœud "fictif", dit *root element*, qui englobe tout le document sauf <?xml version="1.0"?>
/root	sélectionne le nœud vide, puisqu'il n'y a pas d'élément "root" (mais "racine")
//article	sélectionne tous les éléments "article" du document où qu'ils soient
/racine/encyclopedie	sélectionne l'unique élément "encyclopedie" puisqu'il est ici le seul fils de "racine" portant ce nom
//article[@nom='XPath']	sélectionne tous les éléments "article" du document où qu'ils soient, ayant un attribut "nom" dont la valeur est "XPath"

Toutes ces expressions XPath sont **absolues**, c'est-à-dire qu'elles donnent le même résultat quel que soit le contexte. Les expressions suivantes sont **relatives**. Si le contexte courant est l'unique élément "encyclopedie", elles donnent :

Expression XPath	Résultat
article	sélectionne l'élément "article"
racine	ne sélectionne rien, vu le contexte
article[1]/auteurs/auteur[2]	sélectionne le second auteur (Dubois) du premier article
article[count(article/auteurs/auteur) >1]	sélectionne les articles qui ont au moins 2 auteurs
../racine	sélectionne l'élément "racine", puisqu'il est parent de l'élément courant

Le résultat de ces sélections dépendra de la nature de la tâche :

- En affichage, ce sera la valeur textuelle, propre à chaque type d'élément, qui apparaîtra. Si plusieurs nœuds sont sélectionnés, comme pour //article, seul le premier est concerné.
- En sélection, il se comportera comme un pointeur sur lequel d'autres requêtes XPath pourront être effectuées.

La bibliothèque Perl XML ::XPath

http://xml.sergeant.org/xpath/

XML::XPath - a set of modules for parsing and evaluating XPath statements

Synopsis

```
use XML::XPath;
use XML::XPath::XMLParser;

my $xp = XML::XPath->new(filename => 'test.xhtml');

my $nodeset = $xp->find('/html/body/p'); # find all paragraphs

foreach my $node ($nodeset->get_nodelist) {
       print "FOUND\n\n",
              XML::XPath::XMLParser::as_string($node),
              "\n\n";
}
```

```
#/usr/bin/perl
use XML::XPath;
# On vérifie le nombre d'arguments de l'appel au script ($0 : le nom du script)
if($#ARGV!=0){
	print "usage : perl $0 fichier_tag fichier_motif";
	exit;
}
#-------------------------------------------------------------------------------
-----------------------------------------------------------------
my $encodagesortie="utf-8";
open(OUT1,">:encoding($encodagesortie)","sortie-textebrut-avec-xmlxpath.txt");
open(OUT2,">:encoding($encodagesortie)","sortie-textexml-avec-xmlxpath.xml");
print OUT2 "<?xml version=\"1.0\" encoding=\"$encodagesortie\" ?>\n";
print OUT2 "<file>\n";
print OUT2 "<name>$ARGV[0]</name>\n";
my $input_file= shift @ARGV;
my $xp = XML::XPath->new( filename => $input_file ) or die "big trouble";
my $search_path.="//item";
# boucle sur les nœud s reconnus du chemin xpath
foreach my $noeud ( $xp->find($search_path)->get_nodelist ) {
	my $titre=$noeud->find('title')->string_value;
	my $resume=$noeud->find('description')->string_value;
	$titre=&nettoietexte($titre);
	$resume=&nettoietexte($resume);
	print OUT1 "Titre : $titre \n";
	print OUT1 "Resume : $resume \n";;
	print OUT2
"<item><title>$titre</title><abstract>$resume</abstract></item>\n";
}

#-----------------------------------------------------------
print OUT2 "</items>\n</file>\n";
close(OUT1);
close(OUT2);
close(FILE);
exit;
sub nettoietexte {
    my $texte=shift;
    $texte=~s/&#39;/'/g;
    $texte=~s/&#34;/"/g;
    $texte =~ s/&lt;/</g;
    $texte =~ s/&gt;/>/g;
    $texte =~ s/<a href[^>]+>//g;
    $texte =~ s/<img[^>]+>//g;
    $texte =~ s/<\/a>//g;
    $texte =~ s/&#38;#39;/'/g;
    $texte =~ s/&#38;#34;/"/g;
    $texte =~ s/<[^>]+>//g;
    return $texte;
}
```

La bibliothèque Perl XML ::LibXML

Conçue par *Matt Sergeant* et *Christian Glahn* et est activement maintenue par *Petr Pajas* . Il est rapide, complet et stable. Il peut être utilisé dans un contexte validant ou non validant et utilise DOM sous le support XPath. Sa méthode DOM et la gestion de la mémoire ont été écrites en C (il utilise libxml2 écrit en C) offrant ainsi des performances significatives. Il permet également de faire très facilement des transformations XSL en la combinant au module XML::LibXSLT (qui utilise la librairie écrite en c libxslt).

http://search.cpan.org/dist/XML-LibXML/

```
#/usr/bin/perl
use XML::LibXML;
# On vérifie le nombre d'arguments de l'appel au script ($0 : le nom du script)
if($#ARGV!=0){
	print "usage : perl $0 fichier_tag fichier_motif";
	exit;
}
#-------------------------------------------------------------------------------
-----------------------------------------------------------------
my $encodagesortie="utf-8";
open(OUT1,">:encoding($encodagesortie)","sortie-textebrut-avec-xmllibxml.txt");
open(OUT2,">:encoding($encodagesortie)","sortie-textexml-avec-xmllibxml.xml");
print OUT2 "<?xml version=\"1.0\" encoding=\"$encodagesortie\" ?>\n";
print OUT2 "<file>\n";
print OUT2 "<name>$ARGV[0]</name>\n";
my $input_file= shift @ARGV;

my $parser = XML::LibXML->new();
my $xp = $parser->parse_file($input_file);

# boucle sur les nœud s reconnus du chemin xpath
foreach my $noeud ( $xp->findnodes('//item')->get_nodelist ) {
	my $titre=$noeud->findnodes('title')->string_value;
	my $resume=$noeud->findnodes('description')->string_value;
	$titre=&nettoietexte($titre);
	$resume=&nettoietexte($resume);
	print OUT1 "Titre : $titre \n";
	print OUT1 "Resume : $resume \n";;
	print OUT2
"<item><title>$titre</title><abstract>$resume</abstract></item>\n";
}

#-----------------------------------------------------------
print OUT2 "</items>\n</file>\n";
close(OUT1);
close(OUT2);
close(FILE);
exit;
sub nettoietexte {
    my $texte=shift;
    $texte=~s/&#39;/'/g;
    $texte=~s/&#34;/"/g;
    $texte =~ s/&lt;/</g;
    $texte =~ s/&gt;/>/g;
    $texte =~ s/<a href[^>]+>//g;
    $texte =~ s/<img[^>]+>//g;
    $texte =~ s/<\/a>//g;
    $texte =~ s/&#38;#39;/'/g;
    $texte =~ s/&#38;#34;/"/g;
    $texte =~ s/<[^>]+>//g;
    return $texte;
}
```