Welcome to my tiny Rss parser...I'm trying to create a small rss reader in liberty basic...I don't have any scientific background about programming, but little bit of little bit of that, i'm enjoying my time to write small applications in Liberty Basic (or other basics, but i find liberty basic really basic about GUI applications, is it only me?). Well as i'm writing the code, i'm trying to purify it as much as i can...The code i provide here is able strip most rss tags and print to mainwin...I'll change it by time, to get adapted for a GUI, so i need to place parsed elements to some string variable with index...I hope this will work for all...

---

See the first part for the URL,receive code from Alycesrestaurant.com

```
GLOBAL rss$
url$="http://rss.cnn.com/rss/cnn_topstories.rss"
localfile$="rss.xml"


a=geturl(url$,localfile$)
a= StripTags(rss$)



Function geturl(url$,localfile$)
uniUrl$ = MultiByteToWideChar$(url$)

if uniUrl$ = "" then
    print "Unable to convert URL to unicode string."
    end
end if

'if result = 0, URL is valid
result = ValidURL(uniUrl$)

if result = 0 then
    'download html from url
    Cursor Hourglass
    downloadresult = DownloadToFile(url$,localfile$)
    if downloadresult <> 0 then print "Error downloading ";url$
    Cursor Normal
else
    print "Invalid URL:"; url$
end if

'Now opening the rss file, and binding to local var...
'Will be array for future multiple rss...

open  localfile$ for input as #f
```

```
    rss$=input$ (#f, lof(#f))
close #f
End Function



function StripTags(raw$)
'print only the tags starting with < and ending with >
'According to RSS 2.0 standart we have the following channel data..
'First, start with rss or xml declaration, version...Then
'Channel
'       |-Title       Required
'       |-Link        Required
'       |-Description Required
'
'       |-Language         Optional
'       |-Copyright        Optional
'       |-managingEditor   Optional
'       |-webMaster        Optional
'       |-pubDate          Optional
'       |-lastBuildDate    Optional
'       |-category         Optional
'       |-generator        Optional
'       |-docs             Optional
'       |-cloud            Optional
'       |-ttl              Optional
'       |-image            Optional-----------------------------------
----------
'       |-rating           Optional
          |><url></url>
'       |-textInput        Optional
          |<title></title>
'       |-skipHours        Optional
          |<link></link>
'       |-skipDays         Optional
          |<width></width>
'
          |<height></height>
'
          |<description></description>
'
'
'
'
'
'
```

```
'
' Elements of items...Items are sub-elements of Channels btw...
'So
'Channel----
'           |---<item>
'           |---<item>
'           |---<item>
'           |---<item>
'                    |-Title
'                    |-Link
'                    |-Description
'                    |-Author
'                    |-Category
'                    |-Comments
'                    |-enclosure
'                    |-guid
'                    |-pubDate
'                    |-source
'But the point is that most rss feeds doesn't follow this sequence of
tags...So, first, we'll remove the tags
'with the information attached to them, and we'll have TAG=Content typ
e of arrays...

channeltags$=
"title,link,description,language,copyright,managingeditor,";_
"webmaster,pubdate,lastbuilddate,category,generator,docs,cloud,ttl,ima
ge,";_
"rating,textinput,skiphours,skipdays,"

itemtags$="title,link,description,author,category,";_
"comments,enclosure,guid,pubdate,source,"

imgtags$="url,title,link,width,height,description,"


while tagend<len(raw$)

tagstart=instr(raw$,"<",tagend)
tagend=instr(raw$,">",tagstart)
taglen=tagend-tagstart+1
'print mid$(raw$,leftbrack,taglen), rightbrack

    if instr(mid$(raw$,tagstart,taglen),"<channel>",0)>0 then
        chcount=1
        print "We found a channel! "
```

```
            while word$(channeltags$,chcount,",")<>""



'if instr(raw$,"<"+word$(channeltags$,chcount,",")+">")<>0 then

                    tagstart=instr(raw$,"<"+word$(
channeltags$,chcount,",")+">")+len("<"+word$(channeltags$,
chcount,",")+">")
                    tagend=instr(raw$,"</"+word$(
channeltags$,chcount,",")+">",tagstart)
                    taglen=tagend-tagstart

                        if word$(channeltags$,chcount,",")=
"image" then 'Parsing image...

                        itcount=1
                        imgraw$=mid$(raw$,tagstart,taglen)

                            while word$(imgtags$,itcount,",")<>""


  imgstart=instr(imgraw$,"<"+word$(imgtags$,itcount,",")+">")+len(
"<"+word$(imgtags$,itcount,",")+">")

imgend=instr(imgraw$,"</"+word$(imgtags$,itcount,",")+">",imgstart)
                                imglen=imgend-imgstart
                                print "Channel>Image>";word$(
imgtags$,itcount,",");">";mid$(imgraw$,imgstart,imglen)
                                itcount=itcount+1

                            wend
                            goto [skipimage]
                        end if


                    print upper$(word$(channeltags$,chcount,","));
">>>>";mid$(raw$,tagstart,taglen)

'here, if we have image, seperately we must parse it...
                [skipimage]
              'end if
                    chcount=chcount+1
            wend

'Upto here, everything is working fine...Missing parts are GUID and me
```

```
dia...image tag should be parsed independent...

            filepos=instr(raw$,"<item>")

            while instr(raw$,"<item>")<>0
'Loop until there are no items left...

'since channel is finished, cut the raw$ to only size of items...
            currentitem$=mid$(raw$,filepos-1,instr(raw$,"</item>"))
            raw$=right$(raw$,len(raw$)-instr(raw$,"/item>"))
            itemcount=1

             print "New item!":print:print

            while word$(itemtags$,itemcount,",")<>""

                        if instr(lower$(currentitem$),"<"+word$(
itemtags$,itemcount,",")+">")<>0 then
                           tagstart=instr(lower$(
currentitem$),"<"+word$(itemtags$,itemcount,",")+">")+len("<"+word$(
itemtags$,itemcount,",")+">")
                           tagend=instr(lower$(
currentitem$),"</"+word$(itemtags$,itemcount,",")+">",tagstart)
                           taglen=tagend-tagstart
                           print upper$(word$(itemtags$,
itemcount,","));">>>>";mid$(currentitem$,tagstart,taglen)
'                           print tagstart,tagend,taglen
                        end if
                    itemcount=itemcount+1
            filepos=1
            wend'single item

            wend'whole items...

    end if 'channel end if...



wend

'Upto here, everything is working fine...Missing parts are GUID and me
dia
End Function

Function Window(sizex,sizey,posx,posy)
```

```
End Function



Function lowercase (string$)
lowercase$=lower$(string$)
End Function



Function DownloadToFile(urlfile$, localfile$)
    open "URLmon" for dll as #url
    calldll #url, "URLDownloadToFileA",_
    0 as long,_          'null
    urlfile$ as ptr,_    'url to download
    localfile$ as ptr,_  'save file name
    0 as long,_          'reserved, must be 0
    0 as long,_          'callback address, can be 0
    DownloadToFile as ulong  '0=success
    close #url
end function

Function ValidURL(urlfile$)
    open "URLmon" for dll as #url
    calldll #url, "IsValidURL",_
    0 as long,_          'ignored, must be 0
    urlfile$ as ptr,_    'urlfile to check
    0 as ulong,_         'ignored, must be 0
    ValidURL as long
    close #url
end function


function MultiByteToWideChar$(String$)
    'converts any string into unicode
    CodePage = 0  :  dwFlags = 0
    cchMultiByte = -1
    lpMultiByteStr$ = String$
    cchWideChar = len(String$) * 3
    lpWideCharStr$ = space$(cchWideChar)

    calldll #kernel32, "MultiByteToWideChar", _
    CodePage as ulong, _    'CP_ACP=0, ansi code page
    dwFlags as ulong, _     'use 0, flags for character translation
```

```
    lpMultiByteStr$ as ptr,_'the ascii string to convert
    cchMultiByte as long, _ 'len of string, -1 for null-
terminated string
    lpWideCharStr$ as ptr, _'buffer for returned ansi string
    cchWideChar as long, _  'size in wide characters of string buffer
    result as long
'returns number of wide characters written to buffer

    if result = 0 then
        MultiByteToWideChar$ = ""
    else
        MultiByteToWideChar$ = left$(lpWideCharStr$, result * 2)
    end if
    end function


'Updated on 12/12/2009 - Minor Changes
```