

Re: macro for parsing text

Source:

<http://www.tech-archive.net/Archive/Excel/microsoft.public.excel.programming/2006-04/msg03070.html>

- *From:* Ron Rosenfeld <ronrosenfeld@xxxxxxxxxx>
 - *Date:* Thu, 13 Apr 2006 17:06:18 -0400
-

On 13 Apr 2006 11:16:10 -0700, markoium@xxxxxxxxxx wrote:

Hello. I have a very large .csv file (90 columns, 1396 rows) that was converted from running a web report. The .csv file contains all the HTML tags from the web report. I have tried using find/replace using < * > (a wildcard between opening and closing tags) and replacing all with a blank but I get a "Formula is too long" error. I want to write a macro to accomplish this and hopefully get past the error message. Many of the cells have numerous HTML tags in them. I am relatively new to macros and any help, even a push in the right direction, would be appreciated. Thanks in advance for any guidance provided. -Mark

Regular Expressions should be readily adaptable to this task.

Not sure how you want to process the file, but this may get you started. This SUB will remove from any cell selected the pattern <*> as you described (i.e. HTML tags).

Before you run this sub you MUST SET A REFERENCE to Microsoft VBScript Regular Expressions 5.5. You'll see this listed under the Tools/References drop-down selection.

Hope this gets you started.

You may want to also delete empty rows, but I'm not sure what your data looks like.

```
=====
Option Explicit
Sub CleanHTML()
Dim c As Range
Const Pattern As String = "<.*>"
Dim i As Long
Dim Temp As String
```

Re: macro for parsing text

```
For Each c In Selection
Temp = c.Text
For i = 1 To RECount(c.Text, Pattern)
Temp = Replace(Temp, REMid(c.Text, Pattern), "")
Next i
c.Value = Temp
Next c
End Sub

'-----
Function REMid(str As String, Pattern As String, _
Optional Index As Variant = 1, _
Optional CaseSensitive As Boolean = True) _
As Variant 'Variant as value may be string or array

Dim objRegExp As RegExp
Dim objMatch As Match
Dim colMatches As MatchCollection

Dim i As Long 'counter
Dim t() As String 'container for array results

' Create a regular expression object.
Set objRegExp = New RegExp

'Set the pattern by using the Pattern property.
objRegExp.Pattern = Pattern

' Set Case Insensitivity.
objRegExp.IgnoreCase = Not CaseSensitive

'Set global applicability.
objRegExp.Global = True

'Test whether the String can be compared.
If (objRegExp.Test(str) = True) Then

'Get the matches.
Set colMatches = objRegExp.Execute(str) ' Execute search.

On Error Resume Next 'return null string if a colmatch index is non-existent
If IsArray(Index) Then
ReDim t(1 To UBound(Index))
For i = 1 To UBound(Index)
t(i) = colMatches(Index(i) - 1)
Next i
REMid = t()
Else
REMid = CStr(colMatches(Index - 1))
If IsEmpty(REMid) Then REMid = ""
End If
On Error GoTo 0 'reset error handler
```

Re: macro for parsing text

```
Else  
REMid = ""  
End If  
End Function
```

```
Function RECount(str As String, Pattern As String, _  
Optional CaseSensitive As Boolean = True) As Long
```

```
Dim objRegExp As RegExp  
Dim objMatch As Match  
Dim colMatches As MatchCollection
```

```
' Create a regular expression object.  
Set objRegExp = New RegExp
```

```
'Set the pattern by using the Pattern property.  
objRegExp.Pattern = Pattern
```

```
' Set Case Insensitivity.  
objRegExp.IgnoreCase = Not CaseSensitive
```

```
'Set global applicability.  
objRegExp.Global = True
```

```
'Test whether the String can be compared.  
If (objRegExp.Test(str) = True) Then
```

```
'Get the matches.  
Set colMatches = objRegExp.Execute(str) ' Execute search.  
RECount = colMatches.Count  
Else  
RECount = 0  
End If  
End Function
```

```
Function REFind(str As String, Pattern As String, _  
Optional Index As Variant = 1, _  
Optional CaseSensitive As Boolean = True) _  
As Variant 'Variant as value may be string or array
```

```
Dim objRegExp As RegExp  
Dim objMatch As Match  
Dim colMatches As MatchCollection
```

```
Dim i As Long 'counter  
Dim t() As Long 'container for array results
```

```
' Create a regular expression object.  
Set objRegExp = New RegExp
```

```
'Set the pattern by using the Pattern property.  
objRegExp.Pattern = Pattern
```

Re: macro for parsing text

Re: macro for parsing text

```
' Set Case Insensitivity.
objRegExp.IgnoreCase = Not CaseSensitive

' Set global applicability.
objRegExp.Global = True

' Test whether the String can be compared.
If (objRegExp.Test(str) = True) Then

' Get the matches.
Set colMatches = objRegExp.Execute(str) ' Execute search.

On Error Resume Next ' return 0 if a colmatch index is non-existent
If IsArray(Index) Then
  ReDim t(1 To UBound(Index))
  For i = 1 To UBound(Index)
    t(i) = colMatches(Index(i) - 1).FirstIndex + 1
  Next i
  REFind = t()
Else
  REFind = colMatches(Index - 1).FirstIndex + 1

End If
On Error GoTo 0 ' reset error handler
End If
End Function
=====
--ron
.
```