Excel + VBA

Das RegExp-Objekt

Autor & Copyright: Dipl.-Ing. Harald Nahrstedt

Version: 2016 / 2019 / 2021 / 365

Erstellungsdatum: 22.02.2024

Überarbeitung:

Beschreibung:

RegExp steht für regulärer Ausdruck (Regular Exüpression) und stammt ursprünglich aus Programmiersprachen wie Perl und PHP. Ein regulärer Ausdruck steht inzwischen zahlreichen Programmiersprachen zur Verfügung. Die Objektbibliothek findet sich unter VBA in Anwendungen wie Excel, Word etc. Hier wird der Einsatz in Word gezeigt.

Anwendungs-Datei: AW-003_RegExpObject.docm

1 Reguläre Ausdrücke

RegExp steht für regulärer Ausdruck (Regular Exüpression) und stammt ursprünglich aus Programmiersprachen wie Perl und PHP. Ein regulärer Ausdruck steht inzwischen zahlreichen Programmiersprachen zur Verfügung.

Reguläre Ausdrücke ermöglichen die Untersuchug von Zeichenfolgen anhand von Mustern, die weit über das hinausgehen, was mit integrierten Zeichenfolgefunktionen in VBA möglich ist.

2 Das RegExp-Objekt in VBA

Das RegExp-Objekt befindet sich in der Objektbibliothek Microsoft VBAScript Regular Expression (Bild 1).

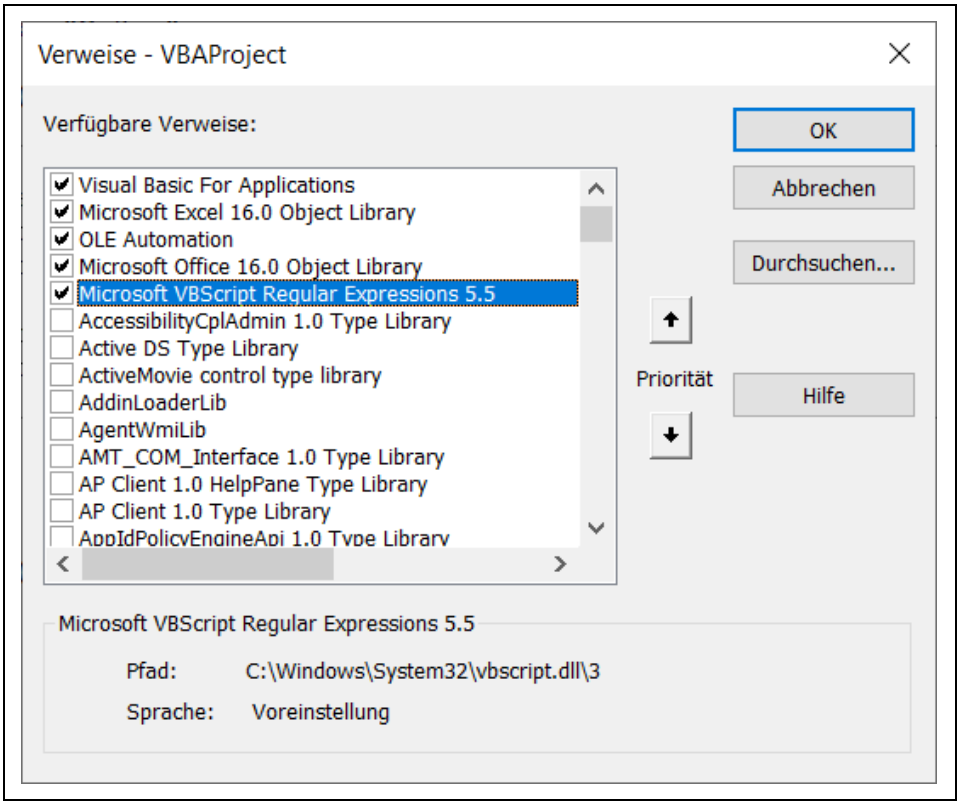

Bild 1. Verweis auf die Objektbibliothek Regular Expression

Auch das *RegExp*-Objekt verfügt über Eigenschaften (Tabelle 1) und Methoden (Tabelle 2).

Tabelle 1. Eigenschaften des RegExp-Objekts

Eigenschaft	Beschreibung
Pattern	Das Muster, das für den Abgleich verwendet werden soll
IgnoreCase	Mit True werden Groß- und Kleinschreibung ignoriert
Global	Mit True werden alle Übereinstimmungen des Musters in der Zeichenfolge gefunden
MultiLine	Mit True erfolgt der Musterabgleich über Zeilenumbrüche hinweg

Tabelle 2. Methoden des RegExp-Objekts

Methode	Beschreibung
Test	Sucht nach dem Muster in der Zeichenfolge und meldet True bei Übereinstimmung
Replace	Ersetzt Vorkommen des Musters in Zeichenfolge durch Ersatzzeichenfolge
Execute	Gibt Übimmung des Musters mit Zeichenfolge

3 Erste Anwendung

In VBA kann mit den Platzhalterzeichen des *Like*-Operators in einem Text nach einem Muster gesucht werden.

```
Sub Test1()
   MsgBox "Mayer" Like "Ma?er"
End Sub
```

Außerdem erlauben einige integrierte Zeichenfolgenfunktionen das Suchen und Verändern von Zeichenfolgen. Damit sind die Möglichkeiten aber bereits erschöpft. Anders ist es mit dem *RegExp*-Objekt.

```
Sub Test2()
   MsgBox Vergleiche("Mayer", "Ma.er")
End Sub

Public Function Vergleiche _
   (sText As String, sMuster As String) As Boolean
   Dim regEx As New RegExp

   regEx.Pattern = sMuster
   Vergleiche = regEx.Test(sText)
End Function
```

In der Funktion *Vergleiche* wird ein Objekt vom Typ *RegExp* angelegt. Das Vergleichsmuster wird der Objekteigenschaft *Pattern* übergeben. Für Vergleichsmuster gibt es eine *Syntax* (Tabelle 3).

Tabelle 3. Syntax für RegExp-Vergleichsmuster

Muster-Syntax	Beschreibung	Beispiel	Gefundene Übereinstimmungen
.	Entspricht jedem einzelnen Zeichen außer *vbNewLine*	f.n	fan, fon, f@n, fwn
[characters]	Entspricht jedem beliebigen einzelnen Zeichen zwischen eckigen Klammern[]	[fn]	Würde nur mit „f“ oder „n“ in Fan übereinstimmen
[^characters]	Entspricht jedem beliebigen einzelnen Zeichen, das nicht zwischen eckigen Klammern steht[]	[^fn]	Würde also mit „j“ in „fjn“ übereinstimmen
[start-end]	Entspricht jedem Zeichen, das Teil des Bereichs in Klammern ist[]	[1-5]	Würde mit „4“ und „5“ in „45“ übereinstimmen
\w	Entspricht alphanumerischen Zeichen und dem Unterstrich, aber nicht dem Leerzeichen	\w	Würde mit „c“ in „c“ übereinstimmen
\W	Entspricht allen nicht alphanumerischen Zeichen und dem Unterstrich	\W	Würde mit „@“ in „bb@bb“ übereinstimmen
\s	Entspricht allen „Weißzeichen“ wie Leerzeichen und Tabulatoren	\s	Würde mit “ “ in „Dies ist“ übereinstimmen
\S	Entspricht jedem Zeichen, das kein Leerzeichen ist	\S	Würde mit „T“ und „h“ in „T h“ übereinstimmen
\d	Entspricht jeder einzelnen Dezimalziffer	\d	Würde mit „7“ in „a7h“ übereinstimmen
\D	Entspricht jedeR einzelnen nicht-dezimalen Ziffer	\D	Würde mit j in „47j“ übereinstimmen
\	Umgeht Sonderzeichen und ermöglicht damit die Suche nach ihnen	\.	Würde mit „.“ in „59.pQ“ übereinstimmen
\t	Tabulator	\t	Würde mit einem Tabulatorzeichen übereinstimmen
\r	Zeilenumbruch	\r	Würde mit einem Zeilenumbruch (vbCr) übereinstimmen
\n	vbNewLine(vbTab)	\n	Würde mit einer neuen Zeile übereinstimmen

Zusätzlich können *Qualifizierer* verwendet werden, um anzugeben, wie oft das Muster mit der Zeichenfolge übereinstimmen soll (Tabelle 4).

Tabelle 4. Quantifizierer für RegExp-Vergleichsmuster

Quantifizierer	Beschreibung	Beispiel	Gefundene Übereinstimmungen
*	Findet null oder mehr Vorkommen	fn*a	fna, fa, fnna, fnnna, fnfnnna
+	Entspricht einem oder mehreren Vorkommen	fn+a	fna, fnna, fnfnna
?	Trifft auf null oder eins zu	fn?a	fa, fna
{n}	Trifft „n“-mehrmals zu	d\W{4}	Würde mit „d….“ in „d….&5hi“ übereinstimmen
{n,}	Stimmt mindestens „n“-mal überein	d\W{4,}	Würde mit „d….&“ in „d….&5hi“ übereinstimmen

Gruppierungen ermöglichen die Verwendung eines Musters, um einen Teil der Zeichenfolge zu erfassen und zu extrahieren. Dadurch wird nicht nur das Muster abgeglichen, sondern auch der Teil der Zeichenkette bestimmt, der mit dem Muster übereinstimmt (Tabelle 5).

Tabelle 5. Gruppierung für RegExp-Vergleichsmuster

Muster	Beschreibung	Beispiel	Gefundene Übereinstimmungen
(Ausdruck)	Gruppiert und erfasst das Muster in Klammern	(\W{4})	Würde „@@@@“ aus „1@@@@1jlmba“ gruppieren und erfassen

4 Test auf Übereinstimmung

Mit der Funktion *=rand(10,10)* erzeugen wir uns eine Spielwiese, auf der wir ein Muster auf Übereinstimmung mit einer Zeichenfolge testen können. Nachdem ein Textteil im Dokument markiert wurde, kann die folgende Prozedur aufgerufen werden.

```
Sub Test3()
   Dim sText As String
   Dim sMuster As String
   Dim rexTemp As RegExp

   Set rexTemp = New RegExp
   sMuster = InputBox("Muster: ")
   rexTemp.Pattern = sMuster
   sText = Selection.Text
   MsgBox rexTemp.Test(sText)
   Set rexTemp = Nothing
End Sub
```

Zuerst wird das Muster abgefragt, mit dem danach getestet wird.

5 In einer Zeichenfolge ein Muster ersetzen

Um ein Muster in einer Zeichenfolge zu ersetzen, wird die *Replace*-Methode verwendet. Wenn die *Global*-Eigenschaft auf False gesetzt wird, dann wird nur die erste Instanz ersetzt.

```
Sub Test4()
   Dim sText As String
   Dim sAlt As String
   Dim sNeu As String
   Dim rexTemp As RegExp

   Set rexTemp = New RegExp
   rexTemp.Global = False
   sAlt = InputBox("Ersetze: ")
   rexTemp.Pattern = sAlt
   sNeu = InputBox("Durch", , sAlt)
   sText = Selection.Text
   MsgBox rexTemp.Replace(sText, sNeu)
   Set rexTemp = Nothing
End Sub
```

Wir markieren einen Satz in unserem Beispieltext (Bild 2).

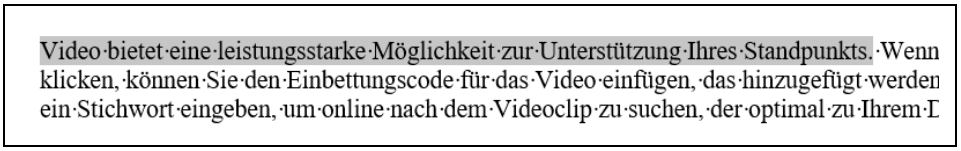

Bild 2. Ausgewählter Satz in unserem Beispieltext

Nach dem Aufruf der Prozedur wird mit der *InputBox*-Methode nach dem zu ersetzenden Textteil gefragt (Bild 3).

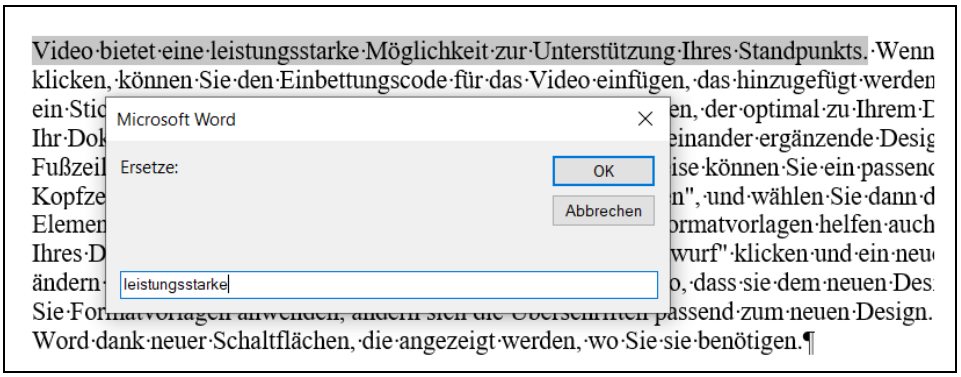

Bild 3. Eingabe des zu ersetzenden Textteils

Mit Bestätigung durch die *OK*-Schaltfläche wird durch eine weitere *InputBox*-Methode nach dem neuenTextteil gefragt (Bild 4). Dabei wird der zu ersetzenden Textteil invers dagestellt und kann überschrieben werden.

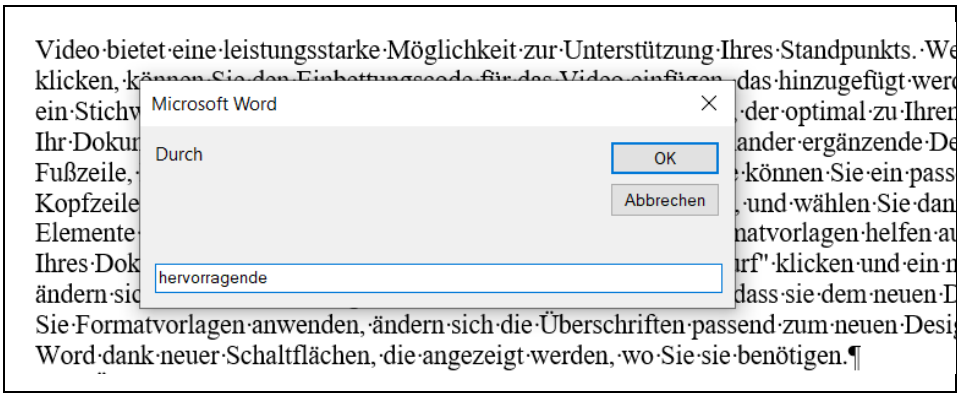

Bild 4. Eingabe des neuen Textteils

Auch hier muss die Bestätigung durch die *OK*-Schaltfläche erfolgen. Die Replace-Methode des RegExp.Objekts liefert dann den neuen Text (Bild 5).

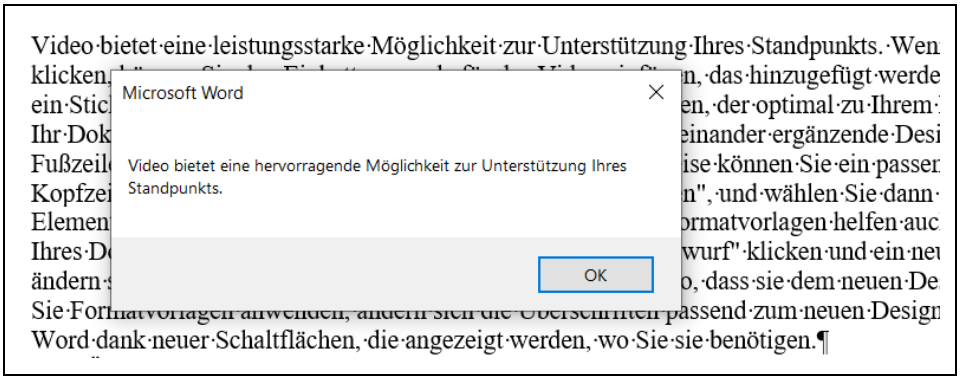

Bild 5. Ergebnis der Replace-Methode

6 Eine Zahl in einer Zeichenfolge ändern

Um eine Zahl in einer Zeichenfolge zu ändern, kann das nachfolgende Muster verwendet werden.

```
Sub Test5()
   Dim sText   As String
   Dim sMuster As String
   Dim rexTemp As RegExp

   Set rexTemp = New RegExp
   rexTemp.Global = False
   sMuster = "[^\D]+"
   rexTemp.Pattern = sMuster
   sText = "Die Postleitzahl lautet 59519"
   MsgBox rexTemp.Replace(sText, "59494")
   Set rexTemp = Nothing
End Sub
```

Ausgang ist die Zeichenfolge: *Die Postleitzahl lautet 59519*. Mit dem Muster und der *Replace*-Methode wird daraus: *Die Postleitzahl lautet 59494*.

7 Komplexere Muster

Um jede Instanz eines Musters in einer Zeichenfolge anzusprechen, muss die *Global*-Eigenschaft auf True gesetzt werden. Danach können komplexe Muster definiert werden.

```
Sub Test6()
   Dim sText   As String
   Dim sMuster As String
   Dim rexTemp As RegExp

   Set rexTemp = New RegExp
   rexTemp.Global = True
   sMuster = "\W\A\d+C\W"
   rexTemp.Pattern = sMuster
   sText = "ABC-A1234C-ABC-A1234C-ABC"
   MsgBox rexTemp.Replace(sText, "XYZ")
   Set rexTemp = Nothing
End Sub
```

Mithilfe der Prozedur wird aus der Ausgangszeichenfolge ABS-A1234C-ABC-A1234C-ABC die Zeichenfolge ABCXYZABCXYZABC.

8 Die Execute-Methode

Die *Execute*-Methode kann verwendet werden, um eine oder alle Instanzen (Global) eines Musters abzugleichen.

```
Sub Test7()
   Dim sText   As String
   Dim sMuster As String
   Dim rexTemp As RegExp
   Dim vFinde  As Variant
   Dim vMatch  As Variant

   Set rexTemp = New RegExp
   rexTemp.Global = True
   sMuster = "A.C"
   rexTemp.Pattern = sMuster
   rexTemp.IgnoreCase = True
   sText = "ABC-A1234C-AXC-A1234C-AYC"
   Set vFinde = rexTemp.Execute(sText)
   For Each vMatch In vFinde
      Debug.Print vMatch.Value
   Next
   Set vFinde = Nothing
   Set rexTemp = Nothing
End Sub
```

Die Prozedur liefert ein Array der gefundenen Zeichenfolgen ABC, AXC und AYC.

Soll aus diesen aber zum Beispiel nur AXC zum Abgleich gefunden werden, dann muss sich das Muster und die *Global*-Eigenschaft entsprechend ändern.

```
Sub Test8()
   Dim sText   As String
   Dim sMuster As String
   Dim rexTemp As RegExp
   Dim vFinde  As Variant
   Dim vMatch  As Variant

   Set rexTemp = New RegExp
   rexTemp.Global = False
   sMuster = "\-\A.C\-"
   rexTemp.Pattern = sMuster
   rexTemp.IgnoreCase = True
   sText = "ABC-A1234C-AXC-A1234C-AYC"
   Set vFinde = rexTemp.Execute(sText)
   For Each vMatch In vFinde
      Debug.Print vMatch.Value
   Next
   Set vFinde = Nothing
   Set rexTemp = Nothing
End Sub
```

Es dauert einige Zeit, um den Umgang mit dem RegExp-Objekt zu lernen. Doch es lohnt sich, denn es ist eine extrem leistungsfähige Möglichkeit zur Erkennung und Manupulation von Zeichenfolgen.