19.6.6 Suchen und Ersetzen von Zeichenketten in einem Text

Für das Suchen und Ersetzen von Zeichenketten in einem vorgegebenen Text wurden in einem Projekt einerseits *reguläre Ausdrücke* und andererseits *Methoden zur Manipulation von Zeichenketten* kombiniert eingesetzt. Sie können im Text nach unterschiedlichen – fest vorgegebenen – Zeichenketten suchen, die eine spezielle Bedeutung haben oder Ihre Muster selbst eintragen:

- MAC-Adresse mit einem Doppelpunkt als Trennzeichen
- Uhrzeit – 23:45 Uhr
- Dezimalzahl mit einem Komma als Dezimaltrennzeichen
- Text in der Form "Text"
- Datum
- IP-Adresse
- URL mit http://www.werist.da oder www.werist.da oder als Kombination
- Postleitzahl für Deutschland
- Geldwert mit Währung €
- ISBN mit Trennstrich nach ISO 2108
- Telefonnummer (Vorwahl) Nummer
- EMail-Adresse
- Farbwert mit &HC3DDFF oder &C3DDFF

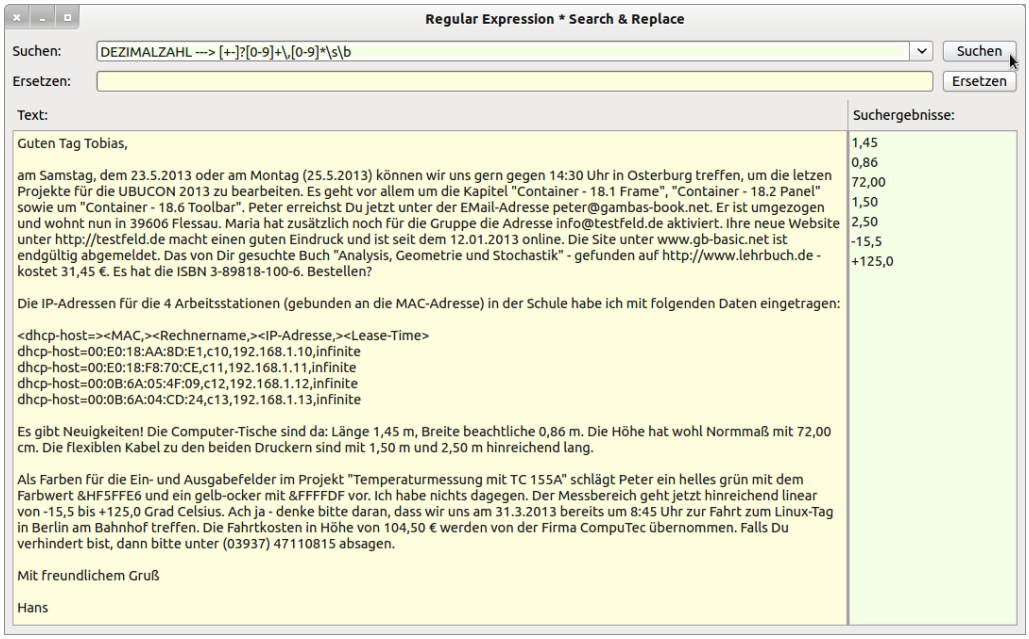

Abbildung 19.6.6.1: Suchen und Ersetzen von Zeichenketten in einem Text

Es hat sich herausgestellt, das für komplexe Texte für jede Klasse von Zeichenketten ein Muster so konstruiert werden muss, das es mindestens ein Alleinstellungsmerkmal hat, das es von anderen Klassen unterscheidet. Sonst könnten Sie zum Beispiel den Geldwert 12,44 € nicht von 12,44 als normaler Dezimalzahl unterscheiden. Gegenwärtig können Sie nur nach ISBN (10) suchen, die nach ISO 2108 mit Trennstrichen formatiert sind.

Der Quelltext wird fast vollständig angegeben, wobei Ihnen vor allem das Zusammenspiel von *regulären Ausdrücken* und *Methoden zur Manipulation von Zeichenketten* gezeigt werden soll:

```
' Gambas class file

Private $rExpression As Regexp
Private $sPattern As String

Public Sub Form_Open()
  $rExpression = New Regexp
  ...
End ' Form_Open()
```

```
Public Sub btnSearch_Click()

  If Not cbxRegexpression.Text Then
    Message.Error("Es ist KEIN Suchtext vorhanden!")
    cbxRegexpression.SetFocus
    Return
  Endif ' Not cbxRegexpression.Text ?

  Try $rExpression.Compile(SetPattern(cbxRegexpression.Text)) ' Fehler im Muster abfangen!
  If Error Then
     Message.Error("FEHLER IM REGULÄREN AUSDRUCK!")
     Return
  Endif ' Error ?

  lbxSearch.List = Search(txaText.Text)

End ' btnSearch_Click()

Public Sub btnReplace_Click()

  If Not cbxRegexpression.Text Then
    Message.Error("Es ist KEIN Suchtext vorhanden!")
    Return
    cbxRegexpression.SetFocus
  Endif ' Not cbxRegexpression.Text ?

  If Not txtReplace.Text Then
     If Message.Error("KEIN Text vorhanden!", "Mit NICHTS ersetzen!", "Abbrechen!") = 1 Then
        Try $rExpression.Compile(SetPattern(cbxRegexpression.Text))
        If Error Then
           Message.Error("FEHLER IM REGULÄREN AUSDRUCK!")
           Return
        Endif ' Error ?
        txaText.Text = TextReplace(txaText.Text, txtReplace.Text)
     Else
        Return
        cbxRegexpression.SetFocus
     Endif
  Endif ' Not cbxRegexpression.Text ?
End ' btnReplace_Click()

Private Function Search(sText As String) As String[]
  Dim aSearchList As New String[]
  Dim iStart As Integer = 1

  Try $rExpression.Exec(sText)  ' ---> Fehler abfangen!
  If Error Then
     Message.Error("Es trat ein Fehler auf ...!")
     Return
  Endif ' Error ?

  While $rExpression.Offset <> -1
    aSearchList.Add($rExpression.Text)
    iStart += $rExpression.Offset + Len($rExpression.Text)
    If iStart > Len(sText) Then Break
    $rExpression.Exec(Mid$(sText, iStart))
  Wend

  Return aSearchList

End ' Function Search(..)

Private Function TextReplace(sText As String, sReplace As String) As String
  Dim iStart As Integer = 1

  Try $rExpression.Exec(sText) ' ---> Fehler abfangen!
  If Error Then
     Message.Error("Es trat ein Fehler auf ...!")
     Return
  Endif ' Error ?

  While $rExpression.Offset <> -1
    iStart += $rExpression.Offset
```

```
    sText = Mid$(sText,1,iStart - 1) & sReplace & Mid$(sText, iStart + Len($rExpression.Text))
    iStart += Len(sReplace)
    If iStart > Len(sText) Then Break
    $rExpression.Exec(Mid$(sText, iStart))
  Wend

  Return sText

End ' Function TextReplace(..)

'Die Funktion SetPattern() ist nur notwendig, um den Mustern einen Hinweistext voranzustellen.

Private Function SetPattern(sInput As String) As String
   Dim iPosition As Integer

   iPosition = InStr(sInput, "--->")
   If iPosition = 0 Then
      $sPattern = sInput
   Else
      $sPattern = Replace(sInput, Left(sInput, iPosition + 3), "")
      $sPattern = Trim($sPattern)
   Endif ' iPosition = 0 ?

   Return $sPattern

End '  Function SetPattern(..)
```

Als Besonderheit werden im Projekt die beiden Methoden *Regexp.Compile()* und *Regexp.Exec()* eingesetzt.

19.6.6.1 Methode Regexp.Compile (gb.pcre)

```
Syntax:     Sub Compile ( Pattern As String [ , CompileOptions As Integer ] )
```

Mit der Methode Compile() können Sie einen regulären Ausdruck (Pattern) für die spätere Ausführung durch die Exec-Methode vor-kompilieren. Dies ist nützlich, wenn Sie ein Muster oft für sehr viel Text einsetzen wollen.

19.6.6.2 Methode Regexp.Exec (gb.pcre)

```
Syntax:     Sub Exec ( Subject As String [ , ExecOptions As Integer ] )
```

Die Methode Exec() ermöglicht Ihnen den Einsatz eines zuvor kompilierten regulären Ausdruck Dies ist vor allem nützlich, wenn Sie viele verschiedene Texte prüfen wollen. Sie müssen einen regulären Ausdruck nur einmal kompilieren und können dann Exec(..) wiederholt für verwenden, was eine höhere Geschwindigkeit erwarten lässt.

Die Methoden Compile(..) und Exec(..) werden automatisch ausgeführt, wenn Sie ein Muster (Pattern) und einen Text (Subjekt) angeben und ein (neues) Regexp-Objekt aufrufen.

Eine Auswahl der folgenden Konstanten können Sie als Option in den beiden Methoden einsetzen:

```
Anchored  BadMagic  BadOption  BadUTF8  BadUTF8Offset  Callout  Caseless  DollarEndOnly  Do-
tAll  Extended  Extra  MatchLimit  MultiLine  NoAutoCapture  NoMatch  NoMemory  NoSubstring
NoUTF8Check  NotBOL  NotEOL  NotEmpty  Null  UTF8  Ungreedy  UnknownNode
```

Beispiel: Regexp.Anchored – Konstante Anchored As Integer = 16

Wenn diese Konstante als Option angegeben ist, wird das Muster auf die erste passende Stelle im Subjekt "verankert". Dieser Effekt kann auch durch geeignete Konstrukte im Muster selbst erreicht werden.

Im nächsten Quelltext-Ausschnitt aus einem Projekt zur Konvertierung von Gambas-Quelltext in HTML-Quelltext wird u.a. im Gambas-Quelltext nach einer URL *gesucht* und bei positivem Ergebnis durch einen passenden Hyperlink *ersetzt*:

```
Private Sub parseLinks(sURL As String) As String
  Dim sRegex As Regexp
  Dim sPattern, sReplace As String

  sPattern = "(https?://([-\\w\\.]+)+(:\\d+)?(/([\\w/_\\.-]*(\\?\\S+)?)?)?)|(@[\\w]*|#[\\w]*)"
  sRegex = New Regexp(sURL, sPattern)
  sReplace = "<a href=\"" & sRegex.Text & "\">" & sRegex.Text & "</a>"

  Return Replace(sURL, sRegex.Text, sReplace)

End ' parseLinks(..)

Public Sub Test_Click()
  Print parseLinks("http://www.gambas-buch.de")
End
```

Das Ergebnis kann sich sehen lassen:

```
<a href="http://www.gambas-buch.de">http://www.gambas-buch.de</a>
```