Word-Dokument mit Benutzerformular bereinigen

Dr. Volker Thormählen, 29.April 2020

Inhalt

Abbildungen

Listings

1 Aufgabenstellung

In diesem Beitrag wird ein Benutzerformular detailliert beschrieben, das zur Bereinigung eines ausgewählten Word-Dokuments dient. Die Aufgabe des Benutzerformulars besteht also darin, ein interaktiv ausgewähltes Zieldokument mit vom Benutzer ausgewählten Prozeduren zu säubern.

Zunächst wird die Gestaltung und der Inhalt des Benutzerformulars beschrieben. Danach werden die vom Benutzer ausgewählten Aufgaben bzw. Prozeduren zur Bereinigung des jeweils ausgewählten Zieldokuments ausführlich dargestellt. Der VBA-Quellcode dieser Prozeduren wird ebenfalls gezeigt.

2 Aufbau des Benutzerformulars[1]

Das Benutzerformular (s. Abbildung 1) enthält rund 20 Steuerelemente:

- 1 Bezeichnungsfeld
- 5 Befehlsschaltflächen
- 4 Rahmen
- 11 Kontrollkästchen

Im Bezeichnungsfeld (s. o.) wird der vollständige Dateiname des vom Benutzer ausgewählten Word-Dokuments angezeigt. Die Auswahl erfolgt interaktiv.

Die Befehlsschaltflächen dienen zur Steuerung des Benutzerformulars. Die Schaltfläche "Ok" ist solange deaktiviert bis ein Word-Dokument vom Benutzer ausgewählt und im Bezeichnungsfeld mit dem Namen '*lblFileNm*' ausgewiesen wird.

Mit den 4 Rahmen werden die jeweiligen Kontrollkästchen sinnfällig gruppiert.

Jedes angehakte Kontrollkästchen bewirkt den Aufruf der jeweils zugeordneten Prozedur, nachdem die Schaltfläche '*OK*' angeklickt wird.

Alle Kontrollkästchen können mit der Schaltfläche "*Alles auswählen*" bequem angehakt werden. Die Schaltfläche "*Alles zurücksetzen*" bewirkt das Gegenteil. Außerdem ist die individuelle Aus- oder Abwahl der Kontrollkästchen möglich.

[1] Aufbau in Anlehnung an: The Editorium, FileCleaner for Microsoft Word, URL: http://www.editorium.com/14845.htm, veröffentlicht: 2016, gefunden am: 28.April 2020

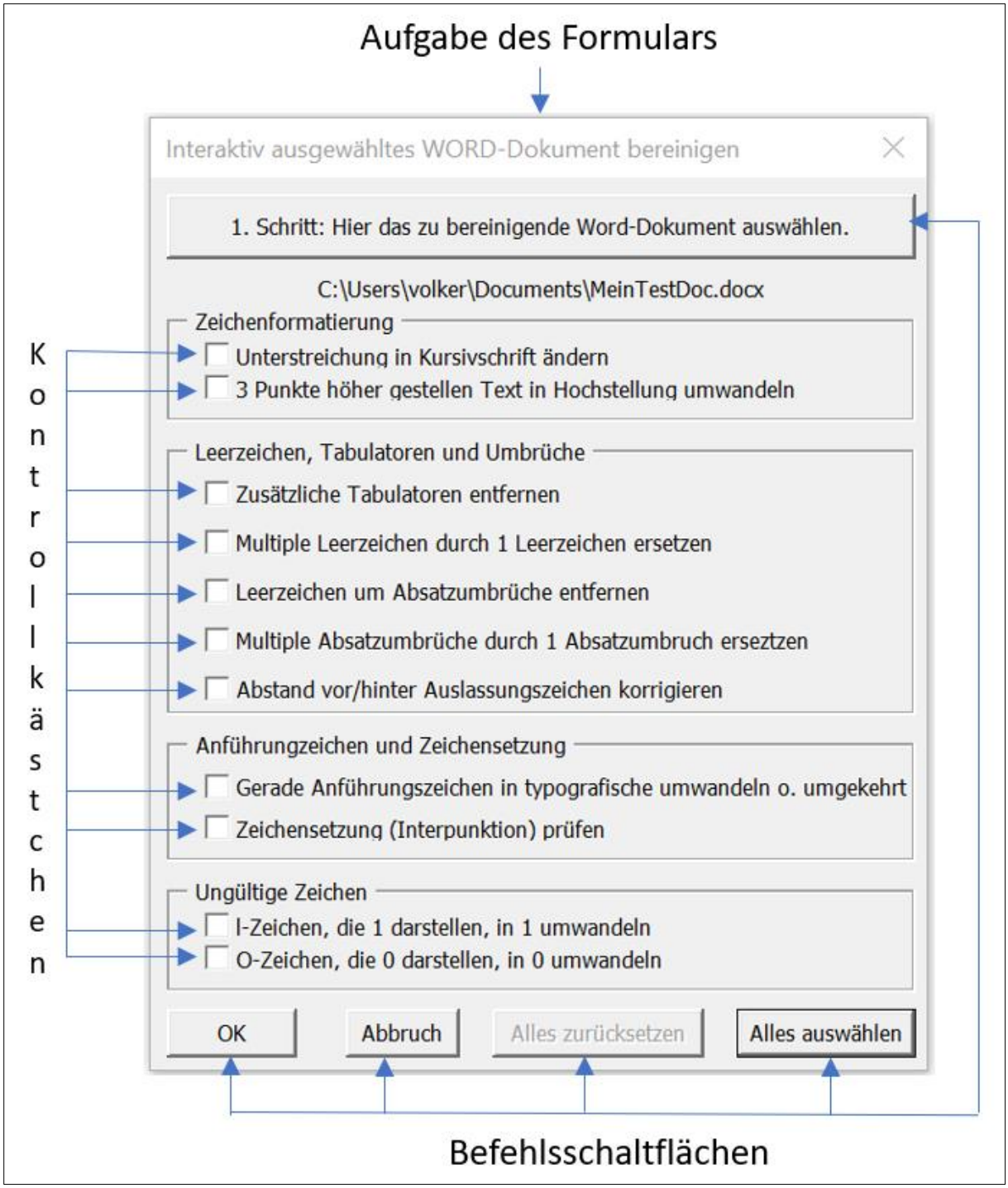

Abbildung 1: Benutzerformular zur Bereinigung von Word-Dokumenten

3 Aufruf des Benutzerformulars

Das Benutzerformular wird im Objekt '*ThisDocument*' (s. Abbildung 2) mit folgenden Codezeilen (s. Listing 1) automatisch aufgerufen:

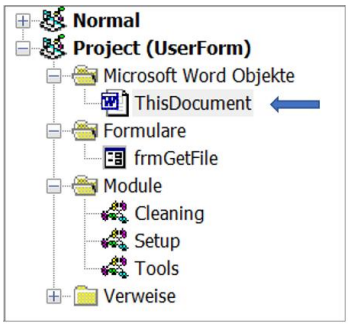

Abbildung 2: Projektexplorer mit dem Objekt 'ThisDocument'

```
Private Sub Document_Open()
   Load frmGetFile
   frmGetFile.Show vbModeless
End Sub
```

Listing 1: Automatischer Aufruf des Benutzerformulars

Das Benutzerformular mit dem Namen '*frmGetFile*' wird geladen und ungebunden (*vbModeless*) angezeigt (s. Listing 1).

4 Zieldokument auswählen

Mit der Schaltfläche *"1. Schritt: Hier das zu bereinigende Word-Dokument auswählen"* des Benutzerformulars (s. Abbildung 1) muss zuerst das Zieldokument vom Benutzer interaktiv ausgewählt werden. Die zugehörige Klick-Prozedur enthält den Befehl *"OpenWordFile"* (s. Listing 2). Damit wird die betreffende VBA-Prozedur im Modul "*Tools*" (s. Abbildung 2) aufgerufen:

```
Private Sub cmdGetFile_Click()
    ' Zieldokument auswählen und öffnen
    Call OpenWordFile
    With Me
        .cmdSelectAll.Enabled = True
        .cmdStart.Enabled = True
    End With
End Sub
```

Listing 2: Klick-Prozedur zur Auswahl des Zieldokuments

Der VBA-Code (s. Listing 3) der betreffenden Auswahlprozedur befindet sich – wie bereits erwähnt – im Modul "*Tools*" (s. Abbildung 2).

```
Sub OpenWordFile()
    ' Aufgabe: Word-Datei auswählen, öffnen und interaktiv bereinigen
    ' Dim docTgt     As Document ' Zieldokument
   Dim strFullNm  As String     ' vollständiger Dateiname des ausgewählten
Zieldokuments
   Dim lngRtnCode As Long       ' Rückgabewert von MsgBox
   On Error GoTo Err_Point      ' Fehlerroutine
   With Application.FileDialog(msoFileDialogOpen)
      .Title = "Dateiauswahl"
      .ButtonName = "Auswählen"
      .AllowMultiSelect = False
      .InitialFileName = ThisDocument.Path
      With .Filters
         .Clear
         .Add "Word-Dateien (*.docx)", "*.docx", 1
         .Add "Word-Dateien (*.docm)", "*.docm", 2
         .Add "Alle Dateien (*.*)", "*.*", 3
      End With
      .FilterIndex = 1
      If .Show = True Then
         strFullNm = CStr(.SelectedItems(1)) ' Pfadname + Dateiname
         If Len(strFullNm) > 0 And Dir(strFullNm) <> vbNullString Then
            frmGetFile.lblFileNm = strFullNm
            frmGetFile.cmdStart.Enabled = True
         Else
            MsgBox "Zieldokument kann nicht geöffnet werden", vbExclamation, "OpenWordFile"
         End If
      Else
         MsgBox "Es wurde nichts ausgewählt!", vbOKOnly, "Dateiauswahl"
      End If
   End With
Exit_Point:
   Exit Sub
Err_Point:
   MsgBox "Laufzeitfehler: " & Err.Description, vbCritical, "OpenWordFile"
   Resume Exit_Point
End Sub
```

Listing 3: VBA-Code der Funktionsprozedur "OpenWordFile"

5 Bereinigungsprozeduren starten

Nach erfolgreicher Auswahl des Zieldokuments wird die Schaltfläche des Benutzerformulars mit dem Namen "*cmdStart*" durch den Befehl *.cmdStart.Enabled = True* aktiviert (s. Listing 2).

Beim Klick auf die genannte Schaltfläche werden alle Bereinigungsprozeduren ausgeführt, deren betreffende Kontrollkästchen (CheckBox1 … CheckBox11) im Benutzerformular ein Häkchen enthalten (s. Listing 5).

6 Existenzprüfung der Bereinigungsprozeduren

Aus Listing 5 geht vorher, dass alle 11 Bereinigungsprozeduren vor ihrer jeweiligen Ausführung eine Existenzprüfung bestehen müssen. Die entsprechende Funktionsprozedur mit dem Namen "*ProcExists*" (s. Listing 4) befindet sich im Modul "*Tools*" (s. Abbildung 2) des Projekts.

```
Function ProcExists(docSrc As Document, strMdlNm As String, strProcNm As String) As Boolean
    ' Aufgabe: Existenzprüfung einer Prozedur
    ' Verweis erforderlich auf 'Microsoft Visual Basic for Applications Extensibility 5.3'
   Dim CodeMod    As VBIDE.CodeModule
   Dim lngLine    As Long
   ProcExists = False
   Err.Clear
   On Error Resume Next
   Set CodeMod = docSrc.VBProject.VBComponents(strMdlNm).CodeModule
   If Err.Number = 0 Then
      lngLine = CodeMod.ProcStartLine(strProcNm, vbext_pk_Proc)
      If Err.Number = 0 Then
         ProcExists = True
      End If
   End If
End Function
```

Listing 4: Funktion zur Existenzkontrolle der jeweiligen Bereinigungsprozedur

```
Private Sub cmdStart_Click()
    ' Bereinigunsprozedur(en) ausführen
    Dim docTgt          As Document ' Zieldokument
    Dim ctrl            As Control  ' Steuerelelement
    Dim strMdlNm        As String   ' Name des Moduls
    Dim strProcNm       As String   ' Name der Prozedur
    Dim strTgtFileNm    As String   ' Dateiname des Zieldokuments
    For Each ctrl In Me.Controls
        If TypeOf ctrl Is MSForms.Label Then
            strTgtFileNm = ctrl.Caption
            Exit For
        End If
    Next ctrl
    If Len(Trim(strTgtFileNm)) = vbNullString Then Exit Sub
    Set docTgt = Documents.Open(FileName:=strTgtFileNm, _
           AddToRecentFiles:=False, Visible:=True)
    '  Hervorhebungen im Zieldokument entfernen
    docTgt.Range.HighlightColorIndex = wdNoHighlight
    strMdlNm = "Tools"
    For Each ctrl In Me.Controls
        If TypeOf ctrl Is MSForms.CheckBox Then
            If ctrl.Value = True Then
                Select Case ctrl.Name
                    Case "CheckBox1"
                        Me.Hide
                        strProcNm = "ChangeUnderlingToItalic"
                        If ProcExists(docSrc, strMdlNm, strProcNm) = True Then
                            docTgt.Activate
                            ChangeUnderlingToItalic docTgt, ctrl.Caption
                        End If
                        Me.Show
                    Case "CheckBox2"
                        Me.Hide
                        strProcNm = "RaisedTextToSuperscript"
                        If ProcExists(docSrc, strMdlNm, strProcNm) = True Then
                            docTgt.Activate
                            RaisedTextToSuperscript docTgt, ctrl.Caption
                        End If
                        Me.Show
                    Case "CheckBox3"
                        Me.Hide
                        strProcNm = "RemoveExtraneousTabs"
                        If ProcExists(docSrc, strMdlNm, strProcNm) = True Then
                            docTgt.Activate
                            RemoveExtraneousTabs docTgt, ctrl.Caption
                        End If
                        Me.Show
                    Case "CheckBox4"
                        Me.Hide
                        strProcNm = "ReplaceMultipleSpaces"
                        If ProcExists(docSrc, strMdlNm, strProcNm) = True Then
                            docTgt.Activate
                            ReplaceMultipleSpaces docTgt, ctrl.Caption
                        End If
                        Me.Show
                    Case "CheckBox5"
                        Me.Hide
                        strProcNm = "DeleteWhiteSpacesAroundReturns"
                        If ProcExists(docSrc, strMdlNm, strProcNm) = True Then
                            docTgt.Activate
                            DeleteWhiteSpacesAroundReturns docTgt, ctrl.Caption
                        End If
                        Me.Show
                    Case "CheckBox6"
                        Me.Hide
                        strProcNm = "RemoveSpacesAroundHyphens"
                        If ProcExists(docSrc, strMdlNm, strProcNm) = True Then
```

```
                        docTgt.Activate
                        RemoveSpacesAroundHyphens docTgt, ctrl.Caption
                    End If
                    Me.Show
                Case "CheckBox7"
                    Me.Hide
                    strProcNm = "RemoveSpacingAroundEllipses"
                    If ProcExists(docSrc, strMdlNm, strProcNm) = True Then
                        docTgt.Activate
                        RemoveSpacingAroundEllipses docTgt, ctrl.Caption
                    End If
                    strProcNm = "InsertSpacingAroundEllipses"
                    If ProcExists(docSrc, strMdlNm, strProcNm) = True Then
                        docTgt.Activate
                        InsertSpacingAroundEllipses docTgt, ctrl.Caption
                    End If
                    Me.Show
                Case "CheckBox8"
                    Me.Hide
                    strProcNm = "ConvertQuotes"
                    If ProcExists(docSrc, strMdlNm, strProcNm) = True Then
                        docTgt.Activate
                        ConvertQuotes docTgt, ctrl.Caption
                    End If
                    Me.Show
                Case "CheckBox9"
                    ' Me.Hide
                    strProcNm = "PunctuationCheck"
                    If ProcExists(docSrc, strMdlNm, strProcNm) = True Then
                        docTgt.Activate
                        PunctuationCheck docTgt, ctrl.Caption
                    End If
                    ' Me.Show
                Case "CheckBox10"
                    Me.Hide
                    strProcNm = "ChangelsToOnes"
                    If ProcExists(docSrc, strMdlNm, strProcNm) = True Then
                        docTgt.Activate
                        ChangelsToOnes docTgt, ctrl.Caption
                    End If
                    Me.Show
                Case "CheckBox11"
                    Me.Hide
                    strProcNm = "ChangeOsToZeros"
                    If ProcExists(docSrc, strMdlNm, strProcNm) = True Then
                        docTgt.Activate
                        ChangeOsToZeros docTgt, ctrl.Caption
                    End If
                    Me.Show
                Case Else
                   MsgBox " Fehler in Fall-Strutur!", vbExclamation, "cmdStart"
                   Exit sub
            End Select
        End If
    End If
Next ctrl
End Sub
```

Listing 5: Prozeduren zur Bereinigung des Zieldokuments ausführen

7 Elf Prozeduren zur Bereinigung des Zieldokuments

```
Sub ChangeUnderlingToItalic(docTgt As Document, strCaption As String)
    Unterstreichung in Kursivschrift ändern und Unterstreichung aufheben.
    Dim intCount    As Integer
    Dim intChanges  As Integer
    Dim lngRetVal   As Long
    Dim objRng      As Range
    Application.ScreenUpdating = False
    Set objRng = docTgt.Range(Start:=0, End:=0)
    With objRng.Find
        .Text = ""
        .ClearFormatting
        With .Font
            .Underline = wdUnderlineSingle
            .Italic = False
        End With
        .Forward = True
        .Wrap = wdFindContinue
        .Format = True
        .MatchCase = False
        .MatchWholeWord = False
        .MatchWildcards = False
        .MatchSoundsLike = False
        .MatchAllWordForms = False
        .Execute
        Do While .Found = True
            intCount = intCount + 1
            objRng.HighlightColorIndex = wdTurquoise
            lngRetVal = MsgBox("Ersetzen?" & vbCr & _
               objRng.Text, vbYesNo + vbQuestion, strCaption)
            If lngRetVal = vbYes Then
                intChanges = intChanges + 1
                With objRng.Font
                    .Underline = wdUnderlineNone
                    .Italic = True
                End With
            End If
            objRng.Collapse Direction:=wdCollapseEnd
            .Execute
        Loop
    End With
    MsgBox CStr(intCount) & " Fundstelle(n), davon " & CStr(intChanges) & "
ersetzt!", vbExclamation, strCaption
    Application.ScreenUpdating = True
End Sub
```

Listing 6: Unterstreichung in Kursivschrift ändern und Unterstreichung aufheben

```
Sub RaisedTextToSuperscript(docTgt As Document, strCaption As String)
  'Drei Punkte höher gestellten Text in Hochstellung umwandeln und Schriftgrad erhöhen.
  Dim objRng         As Range
  Dim intCount       As Integer
  Dim intChanges     As Integer
  Dim lngRetVal      As Long
  Application.ScreenUpdating = False
  Set objRng = docTgt.Range(Start:=0, End:=0)
  With objRng.Find
    .ClearFormatting
    .Replacement.ClearFormatting
    .Text = ""
    With .Font
        .Position = 3
    End With
    .Forward = True
    .Wrap = wdFindContinue
    .Format = True
    .MatchCase = False
    .MatchWholeWord = False
    .MatchWildcards = False
    .MatchSoundsLike = False
    .MatchAllWordForms = False
    .Execute
    Do While .Found = True
        intCount = intCount + 1
        ' objRng.HighlightColorIndex = wdTurquoise
        lngRetVal = MsgBox("Ersetzen?" & vbCr & _
           objRng.Text, vbYesNo + vbQuestion, strCaption)
        If lngRetVal = vbYes Then
            intChanges = intChanges + 1
            With objRng.Font
                .Size = 14
                .Position = 0
                .Superscript = True
            End With
        End If
        objRng.Collapse Direction:=wdCollapseEnd
        .Execute
    Loop
  End With
  Application.ScreenUpdating = True
  MsgBox CStr(intCount) & " Fundstelle(n), davon " & CStr(intChanges) & " ersetzt!",
vbExclamation, strCaption
End Sub
```

Listing 7: Drei Punkte höher gestellten Text in Hochstellung umwandeln und Schriftgrad erhöhen.

```
Sub RemoveExtraneousTabs(docTgt As Document, strCaption As String)
    ' Zusätzliche Tabulatoren entfernen.
    Dim objRng        As Range
    Dim intCount      As Integer
    Dim intChanges    As Integer
    Dim lngRetVal     As Long
    Application.ScreenUpdating = False
    Set objRng = docTgt.Range(Start:=0, End:=0)
    With objRng.Find
        .ClearFormatting
        .Text = "^t{2;}"
        .Forward = True
        .Wrap = wdFindContinue
        .Format = False
        .MatchCase = False
        .MatchWholeWord = False
        .MatchWildcards = True
        .MatchSoundsLike = False
        .MatchAllWordForms = False
        .Execute
        Do While .Found = True
            intCount = intCount + 1
            objRng.HighlightColorIndex = wdTurquoise
            lngRetVal = MsgBox("Ersetzen?" & vbCr & _
               objRng.Text, vbYesNo + vbQuestion, strCaption)
            If lngRetVal = vbYes Then
                intChanges = intChanges + 1
                objRng.Text = Chr(9)
            End If
            objRng.Collapse Direction:=wdCollapseEnd
            .Execute
        Loop
     End With
    MsgBox CStr(intCount) & " Fundstelle(n), davon " & CStr(intChanges) & "
ersetzt!", vbExclamation, strCaption
    Application.ScreenUpdating = True
End Sub
```

Listing 8: Zusätzliche Tabulatoren entfernen

```
Sub ReplaceMultipleSpaces(docTgt As Document, strCaption As String)
    ' Multiple Leerzeichen durch 1 Leerzeichen ersetzen.
    Dim objRng        As Range
    Dim intCount      As Integer
    Dim intChanges    As Integer
    Dim lngRetVal     As Long
    Application.ScreenUpdating = False
    Set objRng = docTgt.Range(Start:=0, End:=0)
    With objRng.Find
        .ClearFormatting
        .Text = "[ ]{2;}"
        .Forward = True
        .Wrap = wdFindContinue
        .Format = False
        .MatchCase = False
        .MatchWholeWord = False
        .MatchWildcards = True
        .MatchSoundsLike = False
        .MatchAllWordForms = False
        .Execute
        Do While .Found = True
            intCount = intCount + 1
            objRng.HighlightColorIndex = wdTurquoise
            lngRetVal = MsgBox("Ersetzen?" & vbCr & _
               objRng.Text, vbYesNo + vbQuestion, strCaption)
            If lngRetVal = vbYes Then
                intChanges = intChanges + 1
                objRng.Text = Space(1)
            End If
            objRng.Collapse Direction:=wdCollapseEnd
            .Execute
        Loop
    End With
    MsgBox CStr(intCount) & " Fundstelle(n), davon " & CStr(intChanges) & "
ersetzt!", vbExclamation, strCaption
    Application.ScreenUpdating = True
End Sub
```

Listing 9: Multiple Leerzeichen durch 1 Leerzeichen ersetzen

```
Sub DeleteWhiteSpacesAroundReturns(docTgt As Document, strCaption As String)
    ' Leerstelle(n) vor/nach Absatzmarken entfernen.
    Dim objRng      As Range
    Dim intCount    As Integer, intChanges As Integer, intLoop As Integer
    Dim lngRetVal   As Long, lngHighLight  As Long
    Dim varFindList As Variant
    Application.ScreenUpdating = False
    ' Leerstelle(n) vor/nach Absatzmarken entfernen
    varFindList = Array("^w^13", "^13^w") ' Suchbegriffe
    lngHighLight = Options.DefaultHighlightColorIndex
    For intLoop = LBound(varFindList) To UBound(varFindList)
        Set objRng = docTgt.Range(Start:=0, End:=0)
        With objRng.Find
            .ClearFormatting
            .Text = varFindList(intLoop)
            .Forward = True
            .Wrap = wdFindContinue
            .Format = False
            .MatchCase = False
            .MatchWholeWord = False
            .MatchWildcards = False
            .MatchSoundsLike = False
            .MatchAllWordForms = False
            .Execute
            Do While .Found = True
                intCount = intCount + 1
                objRng.HighlightColorIndex = wdYellow
                lngRetVal = MsgBox("Ersetzen?" & vbCr & _
                   objRng.Text, vbYesNo + vbQuestion, strCaption)
                If lngRetVal = vbYes Then
                    intChanges = intChanges + 1
                    objRng.Text = Chr(13)
                End If
                With objRng
                    .HighlightColorIndex = wdNoHighlight
                    .Collapse Direction:=wdCollapseEnd
                End With
                .Execute
            Loop
        End With
    Next intLoop
    Options.DefaultHighlightColorIndex = lngHighLight
    MsgBox CStr(intCount) & " Fundstelle(n), davon " & _
           CStr(intChanges) & " ersetzt!", vbExclamation, strCaption
    Application.ScreenUpdating = True
End Sub
```

Listing 10: Leerstelle(n) vor/nach Absatzmarken entfernen

```
Sub RemoveSpacesAroundHyphens(docTgt As Document, strCaption As String)
    ' Ein oder mehr Leerzeichen vor/hinter Bindestrich entfernen.
    Dim objRng        As Range      ' Textbereich
    Dim strListSep    As String     ' Listentrennzeichen
    Application.ScreenUpdating = False
    strListSep = Application.International(wdListSeparator) 'Listentrennzeichen ermitteln
    Set objRng = docTgt.Content
    objRng.Select
    With Selection
        .HomeKey Unit:=wdStory
        With .Find
            .ClearFormatting
            With .Replacement
                .ClearFormatting
                .Text = "\1\3"
            End With
            .Forward = True
            .MatchWildcards = True
            .Text = "(*-)([ ]{1" & strListSep & "})(*)"
        End With
        .Find.Execute Replace:=wdReplaceAll

        .HomeKey Unit:=wdStory
        With .Find
            .ClearFormatting
            With .Replacement
                .ClearFormatting
                .Text = "\1\3\4"
            End With
            .Forward = True
            .MatchWildcards = True
            .Text = "(*)([ ]{1" & strListSep & "})(-)(*)"
        End With
        .Find.Execute Replace:=wdReplaceAll
    End With
    Application.ScreenUpdating = True
    MsgBox "Normales Ende: " & strCaption
End Sub
```

Listing 11: Ein oder mehr Leerzeichen vor/hinter Bindestrich entfernen

```
Sub RemoveSpacingAroundEllipses(docTgt As Document, strCaption As String)
    ' Leerstelle(n) vor/nach Auslassungszeichen entfernen.
    Dim objRng      As Range
    Dim intCount    As Integer, intChanges As Integer, intLoop As Integer
    Dim lngRetVal   As Long, lngHighLight  As Long
    Dim strListSep  As String
    Dim varFindList As Variant
    Application.ScreenUpdating = False
    strListSep = Application.International(wdListSeparator)
    varFindList = Array("[ ]{1" & strListSep & "}^0133", "^0133[ ]{1" & strListSep & "}")
    lngHighLight = Options.DefaultHighlightColorIndex
    For intLoop = LBound(varFindList) To UBound(varFindList)
        Set objRng = docTgt.Range(Start:=0, End:=0)
        With objRng.Find
            .ClearFormatting
            .Text = varFindList(intLoop)
            .Forward = True
            .Wrap = wdFindContinue
            .Format = False
            .MatchCase = False
            .MatchWholeWord = False
            .MatchWildcards = True
            .MatchSoundsLike = False
            .MatchAllWordForms = False
            .Execute
            Do While .Found = True
                intCount = intCount + 1
                objRng.HighlightColorIndex = wdYellow
                lngRetVal = MsgBox("Ersetzen?" & vbCr & _
                   objRng.Text, vbYesNo + vbQuestion, strCaption)
                If lngRetVal = vbYes Then
                    intChanges = intChanges + 1
                    With objRng
                        If intLoop = 0 Then
                            .Text = LTrim(.Text)
                        Else
                            .Text = Trim(.Text)
                        End If
                    End With
                End If
                With objRng
                    .HighlightColorIndex = wdNoHighlight
                    .Collapse Direction:=wdCollapseEnd
                End With
                .Execute
            Loop
        End With
    Next intLoop
    Options.DefaultHighlightColorIndex = lngHighLight
    Application.ScreenUpdating = True
    MsgBox CStr(intCount) & " Fundstelle(n), davon " & _
           CStr(intChanges) & " ersetzt!", vbExclamation, strCaption
End Sub
```

Listing 12: Leerstelle(n) vor/nach Auslassungszeichen entfernen

```
Sub InsertSpacingAroundEllipses(docTgt As Document, strCaption As String)
    ' Fehlende Leerstelle vor/nach Auslassungszeichen einfügen.
    Dim objRng      As Range
    Dim intCount    As Integer, intChanges As Integer, intLoop As Integer
    Dim lngRetVal   As Long, lngHighLight  As Long
    Dim varFindList As Variant
    Application.ScreenUpdating = False
    varFindList = Array("[A-z0-9]^0133", "^0133[A-z0-9]") ' Suchbegriffe
    lngHighLight = Options.DefaultHighlightColorIndex
    For intLoop = LBound(varFindList) To UBound(varFindList)
        Set objRng = docTgt.Range(Start:=0, End:=0)
        With objRng.Find
            .ClearFormatting
            .Text = varFindList(intLoop)
            .Forward = True
            .Wrap = wdFindContinue
            .Format = False
            .MatchCase = False
            .MatchWholeWord = False
            .MatchWildcards = True
            .MatchSoundsLike = False
            .MatchAllWordForms = False
            .Execute
            Do While .Found = True
                intCount = intCount + 1
                objRng.HighlightColorIndex = wdYellow
                lngRetVal = MsgBox("Ersetzen?" & vbCr & _
                   objRng.Text, vbYesNo + vbQuestion, strCaption)
                If lngRetVal = vbYes Then
                    intChanges = intChanges + 1
                    With objRng
                        .HighlightColorIndex = wdNoHighlight
                        If intLoop = 0 Then
                            With objRng
                                .Collapse Direction:=wdCollapseStart
                                .Select
                            End With
                            With Selection
                                .MoveRight Unit:=wdCharacter, Count:=1, Extend:=wdMove
                                .InsertAfter Space(1)
                            End With
                        Else
                            With objRng
                                .Collapse Direction:=wdCollapseEnd
                                .Select
                            End With
                            With Selection
                                .MoveLeft Unit:=wdCharacter, Count:=1, Extend:=wdMove
                                .InsertBefore Space(1)
                            End With
                        End If
                    End With
                Else
                    objRng.HighlightColorIndex = wdNoHighlight
                End If
                .Execute
            Loop
        End With
    Next intLoop
    Options.DefaultHighlightColorIndex = lngHighLight
    Application.ScreenUpdating = True
    MsgBox CStr(intCount) & " Fundstelle(n), davon " & _
           CStr(intChanges) & " ersetzt!", vbExclamation, strCaption
End Sub
```

Listing 13: Fehlende Leerstelle vor/nach Auslassungszeichen einfügen

```
Sub ConvertQuotes(docTgt As Document, strCaption As String)
    Gerade in typografische Anführungsstriche umwandeln oder umgekehrt.
    Dim bolOption   As Boolean
    Dim lngRetVal   As Long, lngLoop As Long
    Dim objRng      As Range
    Dim varFind     As Variant, varRepl As Variant
    Application.ScreenUpdating = False
    lngRetVal = MsgBox("Auf 'Ja' klicken, um typografische in gerade
Anführungszeichen umzuwandeln." & vbCr & _
                       "Auf 'Nein' klicken, um gerade in typografische
Anführungszeichen umzuwandeln.", _
                       vbYesNo, strCaption)
    bolOption = Options.AutoFormatAsYouTypeReplaceQuotes
    If lngRetVal = vbYes Then
        varFind = Array(ChrW(132), ChrW(147), ChrW(148), ChrW(171), ChrW(130), ChrW(145), ChrW(146))
        varRepl = Array(Chr(34), Chr(34), Chr(34), Chr(34), Chr(39), Chr(39), Chr(39))
        Options.AutoFormatAsYouTypeReplaceQuotes = False
        For lngLoop = LBound(varFind) To UBound(varFind)
            Set objRng = docTgt.Range
            With objRng.Find
                Do While .Execute(varFind(lngLoop))
                    With objRng
                        .Text = varRepl(lngLoop)
                        .Collapse wdCollapseEnd
                    End With
                Loop
            End With
        Next lngLoop
    Else
        Options.AutoFormatReplaceQuotes = True
        docTgt.Range.AutoFormat
    End If
    Options.AutoFormatAsYouTypeReplaceQuotes = bolOption
    Application.ScreenUpdating = True
    MsgBox "ConvertQuotes: Normales Ende!", vbExclamation, strCaption
End Sub
```

Listing 14: Gerade in typografische Anführungsstriche umwandeln oder umgekehrt

```
Sub PunctuationCheck(docTgt As Document, strCaption As String)
'Zeichensetzung prüfen
Application.ScreenUpdating = False
Dim arrFind     As Variant
Dim lngLoop     As Long
Dim strListSep  As String     ' Listentrennzeichen

strListSep = Application.International(wdListSeparator) 'Listentrennzeichen ermitteln
Options.DefaultHighlightColorIndex = wdYellow

arrFind = Array(".^s.^s.", "[\(]*[\)]", "-", "\(*\)", "[^02^34^39^=^+\-]", _
"[0-9A-z]^0130", ".^0130", ",^0130", "[\!]^0130", "[\?]^0130", "^0130;", "^0130:", _
"^0130^+", "^+^0130", "^0145^0130", _
"[0-9A-z]^0132", ".^0132", ",^0132", "[\!]^0132", "[\?]^0132", "^0132;", "^0132:", _
"^0132^+", "^+^0132", "^0147^0132", _
"[ ^13]^0145[0-9A-z]", "^0145^+", "^+^0145", "^0130^0145", _
"^0147[0-9A-z]", "^0147^+", "^+^0147", _
"s^0145[!^s]", "^=", "^+", "^39{1;}", "[ ]{2;}", _
",.", ".,", ";,", ",;", ",:", ":,", _
"[\?],", ",[\?]", "[\!],", ",[\!]", _
",[.;:\?\!]", "[.;:\?\!],")

Options.DefaultHighlightColorIndex = wdYellow

With ActiveDocument.Content.Find
    .ClearFormatting
    .Replacement.ClearFormatting
    For lngLoop = 0 To UBound(arrFind)
        .Execute FindText:=arrFind(lngLoop), ReplaceWith:="^&", _
         Replace:=wdReplaceAll, Wrap:=wdFindContinue, _
         MatchWildcards:=True
        ' MsgBox arrFind(lngLoop)
    Next lngLoop
End With

' In Fußnoten ausführen, falls vorhanden
If ActiveDocument.Range.Footnotes.Count > 0 Then
    With ActiveDocument.StoryRanges(wdFootnotesStory).Find
        For lngLoop = 0 To UBound(arrFind)
            .Execute FindText:=arrFind(lngLoop), ReplaceWith:="^&", _
             Replace:=wdReplaceAll, Wrap:=wdFindContinue, _
             MatchWildcards:=True
            .Replacement.Highlight = True
        Next lngLoop
    End With
End If

' Zum ersten Highlight in der Hauptgeschichte gehen
If Selection.Range = ActiveDocument.StoryRanges(wdMainTextStory) = False Then
    ActiveDocument.StoryRanges(wdMainTextStory).Select
    Selection.HomeKey wdStory
    With Selection.Find
        .ClearFormatting
        .Text = ""
        .Highlight = True
        .Execute Forward:=True, Wrap:=wdFindContinue, _
            Format:=True, MatchWildcards:=False, Replace:=wdReplaceNone
    End With
    Selection.Collapse
Else
    Selection.HomeKey wdStory
    With Selection.Find
        .ClearFormatting
        .Text = ""
        .Highlight = True
        .Execute Forward:=True, Wrap:=wdFindContinue, _
            Format:=True, MatchWildcards:=False, Replace:=wdReplaceNone
```

```
    End With
    Selection.Collapse
End If
Application.ScreenUpdating = True
MsgBox "Normales Ende: Interpunktion prüfen"
End Sub
```

Listing 15: Zeichensetzung prüfen

```
Sub ChangelsToOnes(docTgt As Document, strCaption As String)
    l-Zeichen, die keine eins darstellen, in 1 umwandeln.
    Dim intChanges  As Integer
    Dim intHits     As Integer
    Dim objRng      As Range
    Set objRng = docTgt.Range(0, 0)
    With objRng.Find
        .ClearFormatting
        .Text = "[0-9]{1;}l[0-9]{1;}"
        .Replacement.Text = ""
        .Forward = True
        .Wrap = wdFindContinue
        .MatchWildcards = True
        .Format = False
        .MatchCase = False
        .MatchWholeWord = False
        .MatchAllWordForms = False
        .MatchSoundsLike = False
        .Execute
        Do While .Found = True
            intHits = intHits + 1
            objRng.Select
            If MsgBox("Ersetzen? " & objRng.Text, vbQuestion + vbYesNo, strCaption) = vbYes Then
               objRng.Text = Replace(objRng.Text, "l", "1", Compare:=vbTextCompare)
               intChanges = intChanges + 1
            End If
            Selection.Collapse Direction:=wdCollapseEnd
            .Execute
        Loop
    End With
    Set objRng = ActiveDocument.Range(0, 0)
    With objRng.Find
        .Text = "l[0-9]{1;}"
        .Replacement.Text = ""
        .Forward = True
        .Wrap = wdFindContinue
        .MatchWildcards = True
        .Format = False
        .Execute
        Do While .Found = True
            intHits = intHits + 1
            If MsgBox("Ersetzen? " & objRng.Text, vbQuestion + vbYesNo, strCaption) = vbYes Then
               objRng.Text = Replace(objRng.Text, "l", "1", Compare:=vbTextCompare)
               intChanges = intChanges + 1
            End If
            Selection.Collapse Direction:=wdCollapseEnd
            .Execute
        Loop
    End With
    Set objRng = ActiveDocument.Range(0, 0)
    With objRng.Find
        .Text = "[0-9]{1;}l"
        .Replacement.Text = ""
        .Forward = True
        .Wrap = wdFindContinue
        .MatchWildcards = True
        .Format = False
        .Execute
        Do While .Found = True
            intHits = intHits + 1
            If MsgBox("Ersetzen? " & objRng.Text, vbQuestion + vbYesNo, strCaption) = vbYes Then
               objRng.Text = Replace(objRng.Text, "l", "1", Compare:=vbTextCompare)
               intChanges = intChanges + 1
            End If
            Selection.Collapse Direction:=wdCollapseEnd
            .Execute
        Loop
    End With
```

```
    MsgBox CStr(intHits) & " Fundstelle(n), davon " & CStr(intChanges) & " ersetzt!",
vbExclamation, strCaption
End Sub
```

Listing 16: l-Zeichen, die keine eins darstellen, in 1 umwandeln

```
Sub ChangeOsToZeros(docTgt As Document, strCaption As String)
   ' O-Zeichen, die keine Null darstellen, in 0 umwandeln.
    Dim intChanges  As Integer
    Dim intHits     As Integer
    Dim objRng      As Range
    Set objRng = docTgt.Range(0, 0)
    With objRng.Find
        .ClearFormatting
        .Text = "[0-9]{1;}O[0-9]{1;}"
        .Replacement.Text = ""
        .Forward = True
        .Wrap = wdFindContinue
        .MatchWildcards = True
        .Format = False
        .MatchCase = False
        .MatchWholeWord = False
        .MatchAllWordForms = False
        .MatchSoundsLike = False
        .Execute
        Do While .Found = True
            intHits = intHits + 1
            objRng.Select
            If MsgBox("Ersetzen? " & objRng.Text, vbQuestion + vbYesNo, strCaption) = vbYes Then
               objRng.Text = Replace(objRng.Text, "O", "0", Compare:=vbTextCompare)
               intChanges = intChanges + 1
            End If
            Selection.Collapse Direction:=wdCollapseEnd
            .Execute
        Loop
    End With
    Set objRng = ActiveDocument.Range(0, 0)
    With objRng.Find
        .Text = "O[0-9]{1;}"
        .Replacement.Text = ""
        .Forward = True
        .Wrap = wdFindContinue
        .MatchWildcards = True
        .Format = False
        .Execute
        Do While .Found = True
            intHits = intHits + 1
            If MsgBox("Ersetzen? " & objRng.Text, vbQuestion + vbYesNo, strCaption) = vbYes Then
               objRng.Text = Replace(objRng.Text, "O", "0", Compare:=vbTextCompare)
               intChanges = intChanges + 1
            End If
            Selection.Collapse Direction:=wdCollapseEnd
            .Execute
        Loop
    End With
    Set objRng = ActiveDocument.Range(0, 0)
    With objRng.Find
        .Text = "[0-9]{1;}O"
        .Replacement.Text = ""
        .Forward = True
        .Wrap = wdFindContinue
        .MatchWildcards = True
        .Format = False
        .Execute
        Do While .Found = True
            intHits = intHits + 1
           If MsgBox("Ersetzen? " & objRng.Text, vbQuestion + vbYesNo, strCaption) = vbYes Then
               objRng.Text = Replace(objRng.Text, "O", "0", Compare:=vbTextCompare)
               intChanges = intChanges + 1
            End If
            Selection.Collapse Direction:=wdCollapseEnd
            .Execute
        Loop
```

```
    End With
    MsgBox CStr(intHits) & " Fundstelle(n), davon " & CStr(intChanges) & " ersetzt!", vbExclamation, strCaption
End Sub
```

Listing 17: O-Zeichen, die keine Null darstellen, in 0 umwandeln.

8 Erweiterung

Weitere Prozeduren zur Bereinigung eines Word-Dokuments können in die präsentierte Lösung mit relativ wenig Arbeitsaufwand eingebaut werden, beispielweise die Paarigkeitsprüfung[2] von Klammern (runde, eckige, geschweifte, …) und Anführungszeichen (gerade, typografische, französische …) oder die Existenz eines Satzzeichens (.?!) am Ende eines vollständigen Satzes im Haupttext und/oder in anderen Textbereichen (sog. *Story Ranges*[3]) eines Word-Dokuments, beispielsweise in Fußnoten.

[2] Volker Thormählen, Unpaarige Anführungszeichen und Klammern interaktiv korrigieren, URL: http://www.dr-thormaehlen.de/WORD/MismatchedPairs.pdf, gefunden am 29.April 2020.

[3] Der Begriff *Story Range* bezeichnet den Teil eines Word-Dokuments, der durch seinen Typ (*WdStoryType*) identifiziert ist.