



18 Internationale Anwendungen

Einige Firmen in Deutschland, Österreich und der Schweiz verwenden eine englische Oberfläche, um leichter mit ihrer Mutter- oder Tochterfirma in den USA kommunizieren zu können. In anderen Firmen wird englischsprachigen Benutzern eine englische Oberfläche zur Verfügung gestellt, andere haben eine deutsche Oberfläche. Und hieraus ergeben sich einige Schwierigkeit für die Programmierung.

So habe ich beispielweise in einer Applikation das letzte Änderungsdatum

```
Const DATUM As Date = "09.08.2007"
```

um es an mehreren Stellen verwenden zu können:

```
MsgBox "Dieses Makro ist durch Urheberrechte geschützt, " & vbCrLf & _  
"jede unerlaubte Vervielfältigung wird zivilrechtlich " & _  
vbCrLf & vbTab & "und strafrechtlich verfolgt " & vbCrLf & vbCrLf & vbCrLf & _  
"                Entwicklung und Design:" & vbCrLf & _  
"                Rene Martin" & vbCrLf & _  
"                Copyright 2001, Version 3.1 " & vbCrLf & _  
"                Letzte Änderung: " & DATUM, "Info"
```

Dies klappte prima, bis ich eines Tages das Datum in den 31.10.2007 änderte – ein Kompilierungsfehler auf einigen Rechnern (den englischsprachigen) war das Ergebnis.

18.1 Trennzeichen

Daraus ergibt sich das zweite Problem. Wenn eine Anwendung auf mehreren unterschiedlich-sprachigen Rechnern laufen soll, kann es sein, dass das Trennzeichen entweder ein Komma (USA, Großbritannien) oder ein Semikolon (übriges Europa) ist.

Die Sprache kann abgefragt werden:

```
Public Function Sprache() As String

    Dim strSprache As String

    With Application.LanguageSettings

        If .LanguagePreferredForEditing(msoLanguageIDGerman) _
            Or .LanguagePreferredForEditing(msoLanguageIDGermanLiechtenstein) _
            Or .LanguagePreferredForEditing(msoLanguageIDGermanLuxembourg) _
            Or .LanguagePreferredForEditing(msoLanguageIDSwissGerman) Then

            Sprache = "deutsch"

        Else

            Sprache = "englisch"

        End If

    End With

End Function
```

Es ist nicht sehr elegant, aber man kann Fehler, die eine Funktion liefern auch mit der Funktion Evaluate abfangen:

```
Sub Test()

    MsgBox LiefertFunktionFehler("VLOOKUP(A1;E1:F36;2;FALSCH)")

End Sub

Function LiefertFunktionFehler(FunktionsName As String) As Boolean
```

```
LiefertFunktionFehler = IsError(Application.Evaluate(FunktionsName))
```

```
End Function
```

Wird in eine UserForm eine Zahl mit Dezimaltrennzeichen eingegeben, braucht man sich normalerweise keine Gedanken machen, da das Dezimaltrennzeichen der länderspezifischen Oberfläche korrekt in eine US-amerikanische Zahl in VBA umgewandelt wird:

```
Dim curGeldBetrag As Currency

If Me.txtGeldbetrag.Value = "" Then

    MsgBox "Bitte geben Sie einen Betrag ein!", vbInformation

    Me.txtGeldbetrag.SetFocus

    Exit Sub

ElseIf IsNumeric(Me.txtGeldbetrag.Value) = False Then

    MsgBox "Bitte geben Sie nur Zahlen ein!", vbInformation

    Me.txtGeldbetrag.SetFocus

    Me.txtGeldbetrag.SelStart = 0

    Me.txtGeldbetrag.SelLength = Len(Me.txtGeldbetrag.Value)

    Exit Sub

End If
```

```
curGeldBetrag = Me.txtGeldbetrag.Value
```

Eine explizite Umwandlung in den Datentyp Currency ist nicht nötig, erhöht jedoch die Lesbarkeit und erleichtert das Auffinden von Fehlern:

Schwieriger wird es hingegen, wenn diese Zahl als Text gespeichert und manipuliert werden muss. Um dann ein deutsches „12,5“ und ein US-amerikanisches „12.5“ in ein VBA-kompatibles „12.5“ zu verwandeln, könnte man das Ergebnis der Zeichenkette

```
Application.DecimalSeparator
```

beziehungsweise

```
Application.ThousandsSeparator
```

verwenden. Sie liefern länderspezifisch das in der Systemsteuerung eingestellte Tausendertrennzeichen, beziehungsweise Dezimaltrennzeichen.

Sämtliche Sprachen- und PC-abhängige Zeichen werden in der Sammlung

`Application.International`

aufgelistet. So gibt beispielsweise

`MsgBox Application.International(xlDecimalSeparator)`

ein Komma zurück.

Tabelle 18.1 Die Sammlung `Application.International`

Index	Typ	Bedeutung	Ergebnis
Eckige und geschweifte Klammern			
<code>xlLeftBrace</code> , <code>xlRightBrace</code>	String	Das Zeichen, das in einer Matrix anstelle der linken oder rechten geschweiften Klammer verwendet wird.	{ oder }
<code>xlLeftBracket</code> , <code>xlRightBracket</code>	String	Das Zeichen, das in Z1S1-Bezügen anstelle der linken oder rechten eckigen Klammer verwendet wird.	[oder]
<code>xlLowerCaseColumnLetter</code>	String	Ein Kleinbuchstabe für Spaltenangaben	s
<code>xlUpperCaseColumnLetter</code>	String	Ein Großbuchstabe für Spaltenangaben	S
<code>xlLowerCaseRowLetter</code>	String	Ein Kleinbuchstabe für Zeilenangaben	z
<code>xlUpperCaseRowLetter</code>	String	Ein Großbuchstabe für Zeilenangaben (in Z1S1-Bezügen)	S
Länder- und regionale Einstellungen			
<code>xlCountryCode</code>	Long	Die landesspezifische/regionale Version von Microsoft Excel	49 - deutsch
<code>xlCountrySetting</code>	Long	Die aktuelle Länder- und regionale Einstellung in der Windows-Systemsteuerung	49 – Deutschland 41 – Schweiz 43 - Österreich
<code>xlGeneralFormatName</code>	String	Der Name des Standard-Zahlenformats	Standard
Währung			
<code>xlCurrencyBefore</code>	Boolean	True, wenn das Währungssymbol dem Betrag vorangeht, False, wenn es nachgestellt ist.	False
<code>xlCurrencyCode</code>	String	Das Währungssymbol	€
<code>xlCurrencyDigits</code>	Long	Die Anzahl der Nachkommastellen in Währungsformaten	2
<code>xlCurrencyLeadingZeros</code>	Boolean	True, wenn führende Nullen bei Nullbeträgen angezeigt werden	True

Index	Typ	Bedeutung	Ergebnis
xlCurrencyMinusSign	Boolean	True, wenn vor negativen Zahlen ein Minuszeichen angegeben wird. False, wenn Klammern verwendet werden	True
xlCurrencyNegative	Long	Währungsformat für negative Währungswerte: 0 = (Symbolx) oder (xSymbol) 1 = -Symbolx oder -xSymbol 2 = Symbol-x oder x-Symbol 3 = Symbolx- oder xSymbol, wobei Symbol das Währungssymbol des Landes oder der Region darstellt. Beachten Sie, dass die Position des Währungssymbols durch xlCurrencyBefore bestimmt wird.	1
xlCurrencySpaceBefore	Boolean	True, wenn vor dem Währungssymbol ein Leerzeichen eingefügt wird	True
xlCurrencyTrailingZeros	Boolean	True, wenn nachstehende Nullen bei Nullbeträgen angezeigt werden	True
xlNoncurrencyDigits	Long	Die Anzahl der Nachkommastellen für Nicht-Währungsformate	2
Datum und Uhrzeit			
xl24HourClock	Boolean	True beim 24-Stunden-Zeitformat, False beim 12-Stunden-Zeitformat	True
xl4DigitYears	Boolean	True, wenn Jahreszahlen mit 4 Ziffern angezeigt werden. False, wenn nur 2 Ziffern verwendet werden	True
xlDateOrder	Long	Die Reihenfolge der Datumselemente: 0 = Monat-Tag-Jahr 1 = Tag-Monat-Jahr 2 = Jahr-Monat-Tag	1
xlDateSeparator	String	Das Datumstrennzeichen	.
xlDayCode	String	Das Tagessymbol	T
xlMonthCode	String	Das Monatssymbol	M
xlYearCode	String	Das Jahressymbol	J
xlHourCode	String	Das Stundensymbol	h
xlMinuteCode	String	Das Minutensymbol	m
xlSecondCode	String	Das Sekundensymbol	s
xlDayLeadingZero	Boolean	True, wenn bei Tagen eine führende Null angezeigt wird	True

Index	Typ	Bedeutung	Ergebnis
xlMDY	Boolean	True, wenn im langen Datumsformat die Datumsreihenfolge Monat-Tag-Jahr lautet. False, wenn die Reihenfolge Tag-Monat-Jahr ist	False
xlMonthLeadingZero	Boolean	True, wenn bei als Zahlen dargestellten Monaten führende Nullen angezeigt werden	True
xlMonthNameChars	Long	Gibt aus Gründen der Abwärtskompatibilität immer drei Zeichen zurück. Abgekürzte Monatsnamen werden aus Microsoft Windows gelesen und können eine beliebige Länge besitzen	3
xlTimeSeparator	String	Das Uhrzeittrennzeichen	:
xlTimeLeadingZero	Boolean	True, wenn bei Uhrzeiten führende Nullen angezeigt werden	True
xlWeekdayNameChars	Long	Gibt aus Gründen der Abwärtskompatibilität immer drei Zeichen zurück. Abgekürzte Wochentagsnamen werden aus Microsoft Windows gelesen und können eine beliebige Länge besitzen	3
Maßeinheitensysteme			
xlMetric	Boolean	True, wenn das metrische System verwendet wird. False, wenn das englische Maßsystem verwendet wird	True
xlNonEnglishFunctions	Boolean	True, wenn Funktionen nicht in Englisch angezeigt werden	True
Trennzeichen			
xlAlternateArraySeparator	String	Das alternative Trennzeichen für Matrix-Elemente, das verwendet werden soll, wenn das aktuelle Matrix-Trennzeichen mit dem Dezimaltrennzeichen identisch ist	@
xlColumnSeparator	String	Das Zeichen, mit dem Spalten in einer Matrix getrennt werden	\
xlDecimalSeparator	String	Das Dezimaltrennzeichen	,
xlThousandsSeparator	String	Das Null- oder 1.000er-Trennzeichen	.
xlListSeparator	String	Das Listentrennzeichen	;
xlRowSeparator	String	Das Zeichen, mit dem Zeilen in einer Matrix getrennt werden	;

Tabelle 18.2 Liste der unterstützten Werte von xlCountryCode

Sprache	Ländercode	Land
---------	------------	------

Arabic	966	Saudi Arabia
Czech	42	Czech Republic
Danish	45	Denmark
Dutch	31	The Netherlands
English	1	The United States of America
Farsi	982	Iran
Finnish	358	Finland
French	33	France
German	49	Germany
Greek	30	Greece
Hebrew	972	Israel
Hungarian	36	Hungary
Indian	91	India
Italian	39	Italy
Japanese	81	Japan
Korean	82	South Korea
Norwegian	47	Norway
Polish	48	Poland
Portuguese(Brazil)	55	Brazil
Portuguese	351	Portugal
Russian	7	Russian Federation
Simplified Chinese	86	China
Spanish	34	Spain
Swedish	46	Sweden
Thai	66	Thailand
Traditional Chinese	886	Taiwan
Turkish	90	Turkey
Urdu	92	Pakistan
Vietnamese	84	Vietnam

18.2 Länderspezifische Datumsprobleme

Schwierig wird die Eingabe eines Datums. Da das deutsche „06.12.2007“ in den USA nicht den Nikolaustag bedeutet, sondern der russische Unabhängigkeitstag also der 12. Juni, muss eine andere Eingabevariante gesucht werden.

Die Lösung: Entweder Sie binden ein Kalendersteuerelement ein, das ein korrektes Datum liefert, oder Sie trennen das Datum in seine drei Bestandteile: Tag, Monat und Jahr auf und setzen dann die drei Komponenten mit der Funktion `DateSerial` zusammen:

```
datDatum = DateSerial( _  
  
CInt(Me.txtJahr.Value), CInt(Me.txtMonat.Value), CInt(Me.txtTag.Value))
```

Hinweis

Auch hier gilt: Die Funktion `CInt` ist eigentlich überflüssig.

Umgekehrt sollten Sie beachten, dass das Datumstrennzeichen in den USA und Großbritannien der Schrägstrich „/“ ist, während die europäische Norm einen Punkt vorsieht.

Sie sollten auch auf länderspezifische Abfragen wie:

```
If Format(Date, "dddd") = "Sonntag" Then
```

verzichten, da sie natürlich nicht in englischsprachigen Ländern laufen. Selbst

```
If Format(Date, "MMMM") = "Januar" Then
```

funktioniert noch nicht einmal in Österreich, weil dort der erste Monat „Jänner“ heißt.

Die des Problems Lösung sieht vor mit Zahlen zu arbeiten, also:

```
If Weekday(Date:=Date, FirstDayOfWeek:=vbMonday) = 7 Then
```

```
If Month(Date) = 12 Then
```

Das gleiche Problem haben Sie auch bei der Abfrage von Wahrheitswerten in Zellen: Eine Abfrage:

```
If ActiveCell.Value = "Wahr" Then
```

ist sehr problematisch, dagegen läuft:

```
If ActiveCell.Value = True Then
```

länderunabhängig. Umgekehrt liefert:

```
MsgBox "Das Ergebnis lautet: " & True
```

den Text „Das Ergebnis lautet: Wahr“.

18.3 Formeln

Während die Zeile

```
MsgBox ActiveCell.FormulaLocal
```


=SUMME(A1:A3) liefert, so ergibt die Zeile

```
MsgBox ActiveCell.Formula
```

immer – und zwar unabhängig von der eingestellten Sprache

```
=SUM(A1:A3)
```

Das bedeutet, dass Sie auf die Eigenschaft `FormulaLocal` verzichten sollten, wenn die Zielsprache des Rechners nicht klar ist. Zwar wird aus der englischen

```
=SUM(A1:A3)
```

die Funktion `=SUMME(A1:A3)` auf einem deutschen Rechner, jedoch wandelt Excel die Funktion

```
=SUMME(A1:A3)
```

nicht in `=SUM(A1:A3)` auf einem englischen Rechner um.

Hinweis

Beachten Sie auch, dass das Trennzeichen im Englischen das Komma ist und nicht das Semikolon wie auf deutschen Rechnern. Eine Funktion

```
=WENN(A1="";"";A1*19%)
```

muss folglich formuliert werden:

```
=IF(A1="", "", A1*19%)
```

Wie schon bereits erwähnt, sollten Sie die Eigenschaften `FormulaLocal` `NumberFormatLocal` verwenden; also nicht:

```
ActiveCell.FormulaLocal = "=Summe(A1:C1)"
```

oder:

```
ActiveCell.NumberFormatLocal = "#.##0,00"
```

Dies führt auf Rechnern mit einer englischen Oberfläche zu Problemen. Verwenden Sie also die US-amerikanische Schreibweise:

```
ActiveCell.FormulaR1C1 = "=IF(RC[-1]=0,0,RC[-1]*0.19)"
```

Hierbei tritt jedoch das nächste Problem auf. Wenn Sie der oberen Formel nicht hartcodiert den Wert 0.19, der zu einem deutschen 0,19 „umgerechnet“ wird eingeben wollen oder können (beispielsweise, weil diese Formel Teil einer Funktion ist, die einen Wert erhält, dann dürfen Sie nicht schreiben:

```
ActiveCell.FormulaR1C1 = "=IF(RC[-1]=0,0,RC[-1]*" & dbl_MWSt & ")"
```

da dies zu einem Fehler führt. Der Grund: der Wert 0.19 wird für die deutsche Oberfläche in 0,19 verwandelt, in die Formel

```
=IF(RC[-1]=0,0,RC[-1]*" & dbl_MWSt & ")"
```

eingesetzt, so dass dies nun die Zeichenfolge

```
=IF(RC[-1]=0,0,RC[-1]*0,19)
```

ergibt. Und das führt zu einem Fehler. Die Lösung: wandeln Sie die Gleitkommazahl mit der Funktion Str in eine Zeichenkette um. Sie wird automatisch das Komma in einen korrekten Punkt umschreiben:

```
ActiveCell.FormulaR1C1 = "=IF(RC[-1]=0,0,RC[-1]*" & Str(dbl_MWSt) & ")"
```



Achtung

Die Funktion CStr liefert hier jedoch keinen korrekten Wert!

Problematisch wird die Eingabe bei der Funktion TEXT. Sie verwandelt eine Zahl in einen Text, was an einigen Stellen nötig ist. Wenn Sie die Codezeile

```
ActiveCell.FormulaR1C1 = "=TEXT(RC[-1],""0,00"")"
```

auf einem deutschsprachigen Rechner laufen lassen, so erhalten Sie keine Probleme, in den USA wird jedoch ein Fehler erzeugt. Die Lösung könnte wie folgt aussehen:

```
ActiveCell.FormulaR1C1 = "=TEXT(RC[-1],""0" & _  
Application.International(xlDecimalSeparator) & "00"")"
```

Vielleicht fragen Sie sich, wer denn so etwas braucht. Ganz einfach: Wenn Sie in eine Excel-Zelle die Funktion

```
= "München, den "&HEUTE()
```

eingeben, dann steht in der Zelle

```
München, den 39425
```

Damit das Datum korrekt als Datum dargestellt wird, muss es mit der Funktion TEXT konvertiert werden:

```
= "München, den "&TEXT(HEUTE();"TT.MM.JJJJ")
```

Jedoch funktioniert die Codezeile

```
ActiveCell.FormulaR1C1 = _  
""München, den ""&TEXT(TODAY();"TT.MM.JJJJ"")"
```

nur in Deutschland, Österreich und der Schweiz – in den USA ist ein Fehler die Folge. Man müsste folglich die Funktion etwas erweitern in:

```
Dim T As String, M As String, J As String, Sep As String  
T = String(2, Application.International(xlDayCode))  
M = String(2, Application.International(xlMonthCode))
```

```
J = String(4, Application.International(xlYearCode))

Sep = Application.International(xlDateSeparator)

ActiveCell.FormulaR1C1 = _

    ""&"München, den "&TEXT(TODAY(),"") & T & Sep & M & Sep & J & ""&"""
```

Ebenso problematisch ist natürlich die VBA-Funktion `Format`, die eine Zeichenkette zurückgibt. Wird ihr Ergebnis überprüft, müssen länderspezifische Einstellungen beachtet werden. Die Zeile:

```
If Format(Date, "dd.MM.yyyy") = "06.01.2008" Then
```

funktioniert nicht in den USA! Analog wird:

```
If FormatCurrency(Expression:=12345, NumDigitsAfterDecimal:=2, _

    IncludeLeadingDigit:=vbTrue, UseParensForNegativeNumbers:=vbTrue, _

    GroupDigits:=vbTrue) = "12.345,00 €" Then
```

schon in der Schweiz als falsch ausgewertet werden. Analog verhalten sich `FormatNumber` und `FormatDateTime`.

Hinweis

Die bedingte Formatierung (`FormatConditions`) verlangt immer die US-amerikanische Schreibweise, auch wenn der Makrorekorder etwas anderes aufzeichnet!

18.4 Fazit

Internationale Anwendungen zu schreiben ist sehr schwierig. Vor allem, weil wir durch unsere kulturelle Prägung gerne vergessen, dass Dezimaltrennzeichen, Datumsangaben und andere Einstellungen in anderen Ländern – vor allem in Großbritannien und den USA – anders aussehen. Bei folgenden Punkten ist also Vorsicht geboten. Sie verlangen ein genaues und gutes Testen:

- Eingabe von Formeln
- Eingabe von Dezimalzahlen
- Eingabe und Auswertung von Datumsangaben
- Die Methode `OpenText`

verlangt ebenso eine Überarbeitung:

```
Workbooks.OpenText Filename:= _

    "C:\Dokumente und Einstellungen\Rene Martin\Lebmittl.txt", _
```

```
Origin:=xlWindows, StartRow:=1, DataType:=xlDelimited, _  
TextQualifier:=xlDoubleQuote, ConsecutiveDelimiter:=False, _  
Tab:=True, Semicolon:=False, Comma:=False, Space:=False, _  
Other:=False, FieldInfo:=Array(Array(1, 1), Array(2, 1)), _  
DecimalSeparator=".", ThousandsSeparator="," , _  
TrailingMinusNumbers:=True
```

- Die oben genannten Probleme spielen auch bei der Programmierung des Autofilters, Spezialfilters und der Pivottable eine Rolle
- Beachten Sie, dass die Papiergröße in den USA eine andere ist als in Deutschland – die Einstellung `Application.ActiveSheet.PageSetup.PaperSize` unterscheidet zwischen DIN-A-4 und Letter

Tipp

Mit einigen Tipps können die wichtigsten Probleme umgangen werden:

- Übergeben Sie keine Strings an Excel, wenn es sich um Zahlen, Datums- oder Uhrzeitangaben handelt. Konvertieren Sie diese Zahlen mit `CDate`, `CInt`, `Cdbl`, `CCur`, ... in den entsprechenden Typ
- Fragen Sie Werte nicht als Zeichenketten ab, sondern als Zahlen.
- Fragen Sie die entsprechenden Trennzeichen mit der Sammlung `Application.International` ab und bauen Sie so die Zahlenformate, Funktionen und Eingaben zusammen.
- Beim Lesen oder Schreiben verwendet Excel bei `.Formula`, `.NumberFormat` und so weiter die US-amerikanische Schreibweise – unabhängig von der Ländereinstellung.

Hinweis

Übrigens: Ich habe – nachdem der Fehler in der Zeile

```
Const DATUM As Date = "31.10.2007"
```

auf englischsprachigen Rechnern aufgetreten war, die Zeile geändert in:

```
Const DATUM As String = "09.08.2007"
```

Dies funktioniert prima, da ich dieses Datum lediglich in einem Meldungsfenster anzeigen und nicht als Datum weiterverarbeite.