Die Grenzen meiner Sprache bedeuten die Grenzen meiner Welt.
Ludwig Wittgenstein, Tractatus logico-philosophicus(1933)

Programmieren mit Visual Basic for Applications (VBA)

Erfahrungsberichte

So, one has again to warn the scientific community against using MS Excel for statistical purposes.

Knüsel, L. (2002). On the Reliability of Microsoft Excel XP for Statistical Purposes.

Die erwähnten Mängel und die Tatsache, das VBA-Word instabil ist, schliesst die Realisierung von wirklich großen Projekten von vornherein aus..... Nachdem wir uns nach dem Erscheinen von Microsoft Office2000 vermehrt Kunden gewidmet haben, welche von uns eine Microsoft Office-Lösung verlangten (..), sind wir heute froh, dieses Kundensegment wieder sukzessive abbauen zu können, denn es gibt für Softwareentwickler wirklich bessere und nervenschonendere Wege, um ein faires Einkommen zu erlangen.

Erfahrungsbericht gefunden unter: http://mypage.bluewin.ch/reprobst/WordFAQ/VBA.htm

Grundlegendes

Visual Basic for Applications (VBA) ist eine (fast) objektorientierte Makroprogrammiersprache zur Anpassung von Softwareprodukten der Firma Microsoft und kooperierender Partner.
VBA kann vom Sprachumfang in 2 inhaltliche Bereiche geteilt werden

- Sprachkern (gleichzusetzen mit Visual Basic): Schlüsselwörter, Befehle, Funktionen, Kontrollstrukturen, die für alle Anwendungen gelten
- Anwendungsspezifische Objekte, deren Eigenschaften, Ereignisse und Methoden

Komponenten der VBA-Entwicklungsumgebung

Das Code-Fenster

- Fenster in dem der Programmtext geschrieben werden kann

Das Fenster für den Projekt-Explorer

- Fenster in dem die Objekte des Projektes verwaltet werden

Das Eigenschaftenfenster

- Fenster in dem die Eigenschaften des im Projekt-Explorer ausgewählten Objektes angezeigt und verändert werden können

Programmierhilfen

- Objektkatalog, Debugger, Hilfefunktion, ...

Technische Vorbemerkungen

- Es gibt kein Zeichen um das Ende eines Befehls zu markieren. Am Ende einer Zeile muss der Befehl fertig sein oder mit einem Zeilenfortsetzungszeichen (Leerzeichen gefolgt von einem Unterstrich) fortgesetzt werden.
- Groß- und Kleinschreibung wird nicht unterschieden.
- Einzelne Zeilen werden als Kommentar erkannt, wenn sie mit einem Hochkomma ′ beginnen
- Da es für selbstgeschriebene Programme keine „Rückgängigmach"-Funktion gibt, sollte in der Testphase stets nur mit Kopien gearbeitet werden.

Geschichtliches

VBA geht auf die Programmiersprache Basic zurück, die einen einfachen Befehlssatz enthielt, um unter MS-DOS Programme zu schreiben. Als das Betriebssystem MS-DOS dann durch MS Windows abgelöst wurde, wurde auch Basic mit einer grafischen Oberfläche bestückt und zu Visual Basic umbenannt. Für Microsoft Office-Anwendungen stand ursprünglich eine Makrosprache zur Verfügung, welche sich z. B. im Falle von Word „WordBasic" nannte. Seit Office 97 wurde die objektorientierte Programmiersprache VBA eingeführt.

Deklaration des Datentyps von Variablen

Auch wenn es nicht unbedingt empfehlenswert ist: Der Anwender kann das Thema Datentypen ignorieren. Wird für eine Variable überhaupt kein Datentyp festgelegt, so wird automatisch der Datentyp **Variant** angenommen. Variablen vom Typ Variant können Zeichenfolgen, Datums-, Zeit-, boolesche oder numerische Werte enthalten und diese Werte automatisch umwandeln. Variant-Variablen benötigen mindestens 16 Bytes Speicherplatz, und der Zugriff erfolgt langsamer als bei festgelegten Datentypen. Da keine explizite Deklaration werden Variablen, die vom Programmierer nur versehentlich falsch geschrieben wurde, als neue Variablen betrachtet. Es folgt dann eine oft sehr aufwendige Fehlersuche ! Mit der Option Explicit am Modulanfang kann die Deklaration von Variablen erzwungen werden.

Für alle Nichtignoranten von Datentypen – VBA kennt folgende einfache Datentypen:

- boolean – true, false
- byte – ganze Zahl zwischen 0 und 255
- integer – ganze Zahl zwischen –32 und 32 767
- long – ganze Zahl zwischen –2 147 483 648 und 2 147 483 647
- single – Gleitkommazahl ca. von –3.4E38 bis 3.5E38 (8 Stellen Genauigkeit)
- double – reele Zahl ca. von –1.8E308 bis 1.8E308 (16 Stellen Genauigkeit)
- currency – Festkommazahl mit 15 Stellen vor und 4 Stellen nach dem Komma
- string (fester Länge) – bis zu 65 400 Zeichen
- string (dynamischer Länge) – bis ca. 2 Milliarden Zeichen
- date – Datum und Uhrzeit zwischen 01. 01. 0100 und 31.12.9999, Uhrzeit: 00:00 bis 23.59.59
- object – Adressen, die auf Objekte in der Anwendung verweisen können

Die Deklaration des Datentyps erfolgt mit dem Schlüsselwort DIM.

Beispiel	Erläuterung
Dim y As Date y = #1/5/1993#	Die Variable y vom Datentyp Date bekommt den 5. Januar 1993 als Wert zugewiesen (amerikanisches Datenformat).
Dim y as string*5 y=“Hallo“	Die Variable y wird als Zeichenkette mit fester Länge defniert (5 Zeichen) und bekommt anschließend den Wert „Hallo“ zugewiesen.
Dim ob As Object Set ob = Cells(1, 1) Cells(1, 1) = „Hallo“ MsgBox ob Set ob = Nothing MsgBox ob	Die Variable ob vom Typ Object wird mit der Set-Anweisung der Verweis auf die 1. Zelle im aktuellen Tabellenblatt zugewiesen. Die 1. MsgBox-Anweisung gibt daher den Wert „Hallo“ aus. Mit der 2. Set-Anweisung wird die Bindung wieder augehoben und daher ergibt die 2. MsgBox-Anweisung einen Fehler.

Aus den einfachen Datentypen können zusammengesetze Datentypen gebildet werden:

- Zusammenfassung von Elementen beliebigen Datentyps (≅RECORDS)
- Zusammenfassung aufeinanderfolgender, indizierter Elemente gleichen Datentyps (≅ARRAYS)

Beispiel	Erläuterung

Type Person
  Name As String
  Geburtstag As Date
  AnzKinder As Integer
End Type
Dim p As Person
p.Name = „Rudi“
	Der benutzerdefinierte Datentyp „Person“ bestehend aus einer Zeichenkette (für den Namen der Person), einem Datum (für das Geburtsdatum) und einer ganzen Zahl (für die Anzahl der Kinder) werden definiert. Die Variable p wird als Variable vom Typ Person deklariert. Anschließend wird dem Namen ein konkreter Wert zugewiesen.
	(Typdeklaration am Anfang des Moduls, noch vor der ersten Prozedur/Funktion.)

Dim Probe(1 To 100) As Single For i = 1 To 100 Probe(i) = 11 Next i	Das Datenfeld Probe bestehend aus 100 Fließkommazahlen wird deklariert. Anschließend wird in einer Schleife allen Probenwerten von Probe(1) bis Probe(100) der Wert 11 zugewiesen.
Dim Probe() As Integer X = 10 ReDim Probe(1 To X) As Integer For i = 1 To X Probe(i) = 11 Next i	Ein Feld dynamischer Länge. Zunächst wird Probe als Feld unbekannter Länge deklariert. Erst zur Laufzeit des Programms, wenn der Wert von X festgelegt wurde, wird mithilfe der Redim-Anweisung die Länge festgelegt. Die For-Schleife weist schließlich allen Probenwerten den Wert 11 zu.
Dim Probe(1 To 10, 1 To 20) As Integer Probe(10, 20) = 11	Definition eines 2-dimensionalen Feldes mit 10 Zeilen und 20 Spalten und Zuweisung eines Wertes an das letzte Element.

Das Datentyp-Konzept von VBA unterscheidet sich von dem anderer Programmiersprachen. Der Programmierer kann durch die Deklaration eines Datentyps für eine Variable den benötigten Speicherplatz und die Zugriffsgeschwindigkeit steuern. Es erfüllt nicht den Zweck, den Programmierer vor Flüchtigkeitsfehlern zu schützen. Im Gegenteil: Weist er einer Integer-Variablen den Wert 2,5 zu, wird ohne Fehlermeldung gerundet ! Auch auf andere stille Umwandlungen und Überraschungen sollte man gefasst sein, wie die folgende kleine Tabelle zeigt.

Beispiel	Erläuterung
Dim c As Integer c = „1“ + „5,2“	Zunächst werden die Zeichenketten „1“ + „5,2“ zur temporären Zeichenkette „15,2“ verkettet. Da die Variable c als ganze Zahl deklariert wurde, wird sie zu 15 gerundet. Das ERGEBNIS dieser Rechnung ist die Zahl c = 15 !
Dim c As String c = 1 + „5,2“	Auch wenn hier die Variable c als Zeichenkette definiert wurde, die Addition der Zahl 1 mit der Zeichenkette „5,2“ ergibt die temporäre Zahl 6,2 die dann zur Zeichenkette umgewandelt wird. Das ERGEBNIS dieser Rechnung ist die Zeichenkette c = „6,2“ !
Dim c As String * 5 c = „1234567890“	Die Variable c wird als Zeichenkette fester Länge definiert (5 Zeichen). Das Ergebnis der Zuweisung ist daher c=“12345“ !
Dim c As String * 5 c = „Hall“ d = c + „o“	Die Variable c wird als Zeichenkette fester Länge definiert (5 Zeichen). Da die Zeichenkette „Hall“ nur 4 Zeichen hat, wird bei der Zuweisung automatisch mit Leerzeichen aufgefüllt. Das Ergebnis der letzten Anweisung ist somit nicht „Hallo“, sondern „Hall o“ !
Dim a, b, c as integer	Vorsicht: diese Anweisung liefert c Integer, a, b Variant.

Operatoren

- Arithmetisch ^ * / + - \ (Ganzzahldivision) mod
- Zeichenketten + & ("15" & 3 ergibt "153")
- Vergleichsoperatoren = < > <= >= <>
- is *(Wenn zwei Objekte auf dasselbe Objekt verweisen, ist das Ergebnis True; andernfalls ist Ergebnis False.)*
- like zum Vergleich von Zeichenmuster: "abc" like "a*"
- Logische Operatoren and or not xor imp (Implikation) eqv (Äquivalenz)

Zuweisungen

- = Wertübergabe
- := optional bei der Parameterübergabe in Prozeduren
- set weist einer Variablen oder Eigenschaft einen Objektverweis zu

E/A-Operationen

- Eine Eingabe des Nutzers abfragen:

Anwort = InputBox(„Fragetext“)

- Den Nutzer über etwas informieren:

MsgBox „Text“ & Gehalt & „weiterer Text“

(Der Nutzer bekommt die Ausschrift „Text“ den Wert der Variablen Gehalt, gefolgt von „weiterer Text“.)

- Ausgaben ins Direktfenster des Debuggers

Debug.print „Text“

Sprunganweisungen

- Goto label
- Exit Do
- Exit For
- Exit Function
- Exit Property
- Exit Sub

With-Anweisung

Mit der **With**-Anweisung kann man eine Reihe von Anweisungen für ein bestimmtes Objekt ausführen, ohne den Namen des Objekts mehrmals angeben zu müssen.

```
With Bezeichnungsfeld
   .Height = 2000
   .Width = 2000
   .Caption = "Schönen Tag noch"
End With
```

Kontrollstrukturen

Die folgende Tabelle enthält in VBA verfügbare Befehle zur Steuerung der Abarbeitungsreihenfolge. Dick hervorgehoben sind jeweils die Schlüsselworte.

Auswahlbefehl
• vollständige Alternative
	If Leistung = 1 Then
	Bonus = Einkommen * 0.1
	ElseIf Leistung = 2 Then
	Bonus = Einkommen * 0.09
	Else
	Bonus = 0
	End If

Auswahlbefehl • Fallauswahl	**Select Case** Monat **Case** 1, 3, 5, 7,8,10,12 ZahlDerTage =31 **Case** 2, 4, 6, 9, 11 ZahlDerTage =30 **Else** MsgBox "falsche Monatsangabe" **End Select**
Zählergesteuerter Zyklus	**For** Zaehler = 1 **To** 10 **Step** 1 .... *(Anweisungen werden genau 10x durchlaufen)* **Next** Zaehler

Zyklus über alle Elemente eines Aufzählungsobjektes	**For Each** Objekt **In** Auflistung
	If Objekt.Text = "Hallo" Then
	Gefunden = True
	Exit For
	End If
	Next

Kopfgesteuerte Schleifen (pre-checked loop)	Wiederholen von Anweisungen, wenn eine Bedingung dem Wert True entspricht **Do While** meineZahl > 10 meineZahl = meineZahl - 1 Zähler = Zähler + 1 **Loop** Wiederholen von Anweisungen, bis eine Bedingung dem Wert True entspricht **Do Until** meineZahl = 10 meineZahl = meineZahl - 1 Zähler = Zähler + 1 **Loop**
Fußgesteuerte Schleifen (post-checked loop)	Wiederholen von Anweisungen, wenn eine Bedingung dem Wert True entspricht **Do** meineZahl = meineZahl - 1 Zähler = Zähler + 1 **Loop While** meineZahl > 10 Wiederholen von Anweisungen, bis eine Bedingung dem Wert True entspricht **Do** meineZahl = meineZahl + 1 Zähler = Zähler + 1 **Loop Until** meineZahl = 10

VBA als prozedurale Programmiersprache

Funktionen, Prozeduren und Makros

- Kleinste selbständige Einheiten eines VBA-Programms sind Prozeduren und Funktionen. Funktionen geben einen einzelnen Wert zurück, genau wie die Excel-Funktionen Summe oder Cos.
- Makros sind Prozeduren, die von Excel automatisch generiert werden, indem die Aktionen des Nutzers aufgezeichnet werden. Manchmal kann man eine Aufgabe durch kleine manuelle Veränderungen an einem automatisch generierten Makro lösen. Auf jeden Fall helfen sie dem Anfänger, der sich auf diese Art mühelos Beispiele zum Abgucken generieren kann.
- Funktionen, Prozeduren und Makros werden als Teil des Projektes in sogenannten Modulen gespeichert. Bevor man mit dem Programmieren beginnt, muss zunächst ein neues Modul erzeugt werden. *(Option: Einfügen Modul in der Visual Basic Entwicklungsumgebung)*. Verfügt man bereits über von Excel aufgezeichnete Makros, so wurde das Modul bereits automatisch eingefügt.

Aufruf selbstgeschriebener Funktionen und Prozeduren

- Funktionen werden wie die Standard – Excel - Funktionen über die Option Einfügen/Funktion aufgerufen. (Kategorie: „benutzerdefiniert“)
- Prozedur werden genau wie die von Excel-automatisch generierten Makros über die Option Extras/Makro/Makros gestartet.

Wert- und Referenzparameter

- Byval: Wertparameter = Änderung in Prozedur/Funktion wird nicht an Aufrufer zurückgegeben.
- Byref: Referenzparameter = Übergabe als Verweis, Standard in VBA !

```
sub test1 (byval a as integer)
   a = 5
end sub

sub test2 (byref a as integer)
  a = 7
end sub

sub main ()
  b = 2

  test1 b
  debug.print b  ' Ausgabe -> 2

  test2 b
  debug.print b  ' Ausgabe -> 7
end sub
```

Gültigkeitsbereich und Lebensdauer von Variablen

Variablen, die nur in einer Prozedur oder Funktion bekannt sein sollen, deklariert man innerhalb derselben mit dem Schlüsselwort DIM. Variablen, die in allen Prozeduren/Funktionen eines Moduls gültig sein sollen, deklariert man im Deklarationsteil eines Moduls (Abschnitt vor der 1. Prozedur/Funktion.). Variablen, die darüber hinaus in allen Modulen Verwendung finden, muss man mit dem Schlüsselwort Public deklarieren. Wird eine Variable nicht deklariert, so ist sie standardmäßig nur lokal gültig. Die verschiedenen Deklarationsarten unterscheiden sich auch in der Dauer, während der eine Variable ihren Wert beibehält.

Dim x as Integer Public y as Integer Private z as Integer	Deklarationen auf Modulebene • Variable x ist nur im aktuellen Modul bekannt • Variable y ist in allen Modulen bekannt • private entspricht dim Auf Modulebene deklarierte Variablen erhalten ihren Wert, bis das Modul zurückgesetzt bzw. neu gestartet wird. *(Option: Ausführen/Zurücksetzen)*
Sub Prozedurname**()** Dim x as Integer Static y as Integer End Sub	Deklaration auf Prozedur bzw. Funktionsebene • Variable x ist eine lokale Variable und daher nur in der aktuellen Funktion/Prozedur bekannt. Sie entspricht nicht der auf Modulebene deklarierten globalen Variable x. Die Variable behält ihren Wert nur bis zur Beendigung der Prozedur bzw. Funktion. • Variable y ist zwar lokal, behält aber zwischen den Prozeduraufrufen ihren Wert bei. Er wird erst wieder beim Zurücksetzen des Moduls neu initialisiert.

VBA als objektorientierte Programmiersprache

Objektorientierte Prinzipien

1. Bildung von Klassen von Objekten („Objekttypen“)
2. Kapselung (encapsulation): Objekte haben Eigenschaften, die nur mit den dafür vorgesehenen Methoden verändert werden können.
3. Vererbung (inheritance): Bildung von Klassenhierarchien, Subklassen erben Eigenschaften und Verhalten ihrer Oberklassen und können erweiterte Funktionalität bekommen (Spezialisierung)
4. Polymorphismus: Subklassen können geerbte Funktionen überschreiben (method overloading)

Verwendung der Objekte der VBA-Anwendung

VBA stellt die Objekte der jeweiligen Anwendung zur Verfügung. Die Objekte der Anwendung sind in einer Objekthierarchie gegliedert. Um auf ein Objekt zuzugreifen ist die Kenntnis der Position eines Objektes in der Hierarchie notwendig.

Für MS Word bedeutet dies z.B. dass ein Dokument Tabellen enthalten kann, welche wiederum aus Spalten bestehen. Wir haben es daher mit einem Objekttyp Document, deren Element Tables und Unterelement Columns zu tun. Da ein Dokument viele Tabellen enthalten kann, spricht man auch von einer Sammlung (engl. Collection). Auf die einzelnen Tabellen kann man über ihren Index zugreifen. Die folgende Anweisung löscht die 1. Spalte der 1. Tabelle im Dokument:

```
ActiveDocument.Tables(1).Columns(1).Delete
```

Alle Tabellen in der Sammlung Tables (*Collections kann man immer am „s“ im Namen erkennen*) können bequem mit einer speziellen Schleife bearbeitet werden. Die folgende Anweisung löscht die 1. Spalte in allen Tabellen des aktuellen Dokuments.

```
For Each ht In ActiveDocument.Tables
  ht.Columns(1).Delete
Next
```

Die meisten Eigenschaften lassen sich in VBA direkt verändern. Z. B. kann mit der folgenden Anweisung der Hintergrund der 1. Spalte der 1. Tabelle blau gefärbt werden.

```
ActiveDocument.Tables(1).Columns(1).Shading.BackgroundPatternColorIndex = 2
```

Arbeiten mit Collections

Collection = vordefinierte Klasse zur Speicherung beliebig vieler Werte.

```
dim c as new Collection
c.add 1
c.add 5
c.add 17
c.add "Hallo, Welt"
for each el in c
   debug.print el
next
debug.print c.Count, c(2)
```

Nach Abarbeitung erscheint im Direktfenster folgende Ausgabe:

```
1
 5
 17
Hallo, Welt
 4          5
```

Definition eigener Objekte in VBA

Klassenmodule ermöglichen eingeschränkte objektorientierte Programmierung.

Java-Klasse	VBA-Klasse

`class Punkt {`
`double x, y;`
`}`
	• Einfügen eines Klassenmoduls im VB-Editor (Option Einfügen/Klassenmodul)
	• Vergabe des Namens Punkt im Eigenschaftenfenster
	• Im Code-Fenster diesen Text eingeben:
	`public x as double, y as double`

Nutzung einer selbstdefinierten Klasse in VBA

Die folgende Prozedur gibt im Direktfenster (*einschaltbar über Menü Ansicht im VB-Editor*) die Zeichenfolge 2 3 4 aus.

```
sub PunktTest ()
  dim p as Punkt
  set p = new Punkt
  p.x = 2 : p.y = 2

  dim q as new Punkt
  q.x = 3 : q.y = 3

  dim r as Object
  set r = q
  r.y = 4

  debug.print p.x, q.x, q.y

  'Freigabe einer Objektreferenz:
  set p = nothing
end sub
```

Kapselung in VBA

Kapselung der in Klasse deklarierten Variablen; Zugriffsmethoden mit Syntax einer Zuweisung/eines Variablenzugriffs.

Beispiel: Klassenmodul Kreisklasse

```
private p as Punkt
private r as double

property let radius (byval r0 as double)
r = r0
end property

property get radius () as double
radius = r
end property

property set mittelpunkt (byval p0 as Punkt)
set p = p0
end property
```

Zugriff auf gekapselte Eigenschaften über ihre Methoden

```
Sub KreisTest()
  Dim k As New Kreis
  Dim p As New Punkt
  p.x = 2: p.y = 2
  Set k.mittelpunkt = p
  k.radius = 3.5
```

```
  Debug.Print k.mittelpunkt.x, k.radius, k.berechneFlaeche
End Sub
```

Polymorphismus

Vererbung wird nicht unterstützt, Polymorphismus möglich über den Datentyp Objekt, da deren Prüfung erst zur Laufzeit erfolgt.

Beispiel: Zwei Klassen A und B:
KlasseA:

```
function name () as string
name = "A"
end function
```

KlasseB:

```
function name () as string
name = "B"
end function
```

Anwendung

```
Sub Polymorphismus()
  Dim c As New Collection, q As Object
  For i = 1 To 10
     If Rnd < 0.5 Then
        Set q = New KlasseA
     Else
       Set q = New KlasseB
     End If
     c.Add q
  Next i
  For Each q In c
      Debug.Print q.name
  Next
End Sub
```

Erläuterung: Die Funktionsaufrufe q.name sind abhängig vom jeweiligen Typ, der zuvor entsprechend einer Zufallszahl gesetzt wurde.

Beispiele (deren Programmierung weniger als 15 Minuten erfordert)

Excel-Prozedur: MarkAlleUnterSchwellwert

Prozedur, die in einer Excel-Tabelle alle Zellen orange hinterlegt, die kleiner als 5 sind.

```
Sub MarkAlleUnterSchwellwert()
AnzZeilen = ActiveSheet.UsedRange.Rows.Count
AnzSpalten = ActiveSheet.UsedRange.Columns.Count

For spalte = 1 To AnzSpalten
      For zeile = 1 To AnzZeilen
            If Cells(zeile, spalte) < 5 Then
                  Cells(zeile, spalte).Interior.ColorIndex = 40
            End If
      Next
Next
End Sub
```

Wordprozedur: HyperlinksKiller

Prozedur, welche alle Hyperlinks im aktuellen Worddokument in normalen Text umwandelt.

```
Sub HyperlinksKiller()
With ActiveDocument
  For i = 1 To .Hyperlinks.Count
    .Hyperlinks(1).Delete
  Next
End With
End Sub
```

Accessprozedur DB_INFO

Erzeugt ein ASCII-File mit einer Auflistung der Tabellen der aktuellen Datenbank zusammen mit jeweiligen Feldern.

```
Sub DB_Info()
Set fs = CreateObject("Scripting.FileSystemObject")
zieldateiname = CurrentDb.Name & ".txt"
Set ziel = fs.CreateTextFile(zieldateiname, True)

ziel.Writeline "Aktuelles Datenbankfile: " & CurrentDb.Name
ziel.Writeline

For i = 0 To CurrentDb.TableDefs.Count - 1
  hstr = "Tabelle " & i & ": " & CurrentDb.TableDefs(i).Name & "Felder:"
  For j = 0 To CurrentDb.TableDefs(i).Fields.Count - 1
      hstr = hstr & " " & CurrentDb.TableDefs(i).Fields(j).Name
  Next j
  hstr = hstr & ")"
  ziel.Writeline hstr
Next i
ziel.Close
End Sub
```

Vielleicht nützliche Links

Skripte/Tutorials

- ACCESS: http://www.pi-linz.ac.at/ (Rubrik Veröffentlichungen / Downloads)
- EXCEL: http://xtremes.stat.math.uni-siegen.de/~michael/lehre/prakws03/skript.pdf
- EXCEL und WORD: http://www.fernuni-hagen.de/URZ/urzbib/ls_broschueren.html
- VBA in 21 Tagen http://www.ti5.tu-harburg.de/manual/vba5/httoc.htm
- RRZN Handbücher (erhältlich bei der ZEDAT)

BEISPIELE, BEISPIELE,BEISPIELE

- http://mypage.bluewin.ch/reprobst/WordFAQ/VBA.htm (kritische Seite zu VBA für Word)
- http://www.jumper.ch/
- http://people.freenet.de/a.seeberger/index.html
- http://www.excel-cd.de/ (Kopieren der Beispiele nur über manuelles Abtippen)
- http://www.vb-fun.de/vb/index.htm (eher eine VB-Seite, aber auch VBA)
- http://www.excel-inside.de/
- http://www.fh-potsdam.de/~Bauing/fachgebiete/vba/startseite.htm (für Bauingenieure)
- http://www.arstechnica.de/computer/msoffice/vbaindex.html
- http://www.excel-vba.de/
- http://www.excel-center.de/index.php

Foren

- http://www.office-loesung.de/
- http://www.excel-center.de/forum/
- http://www.herber.de/forum/

Quellen

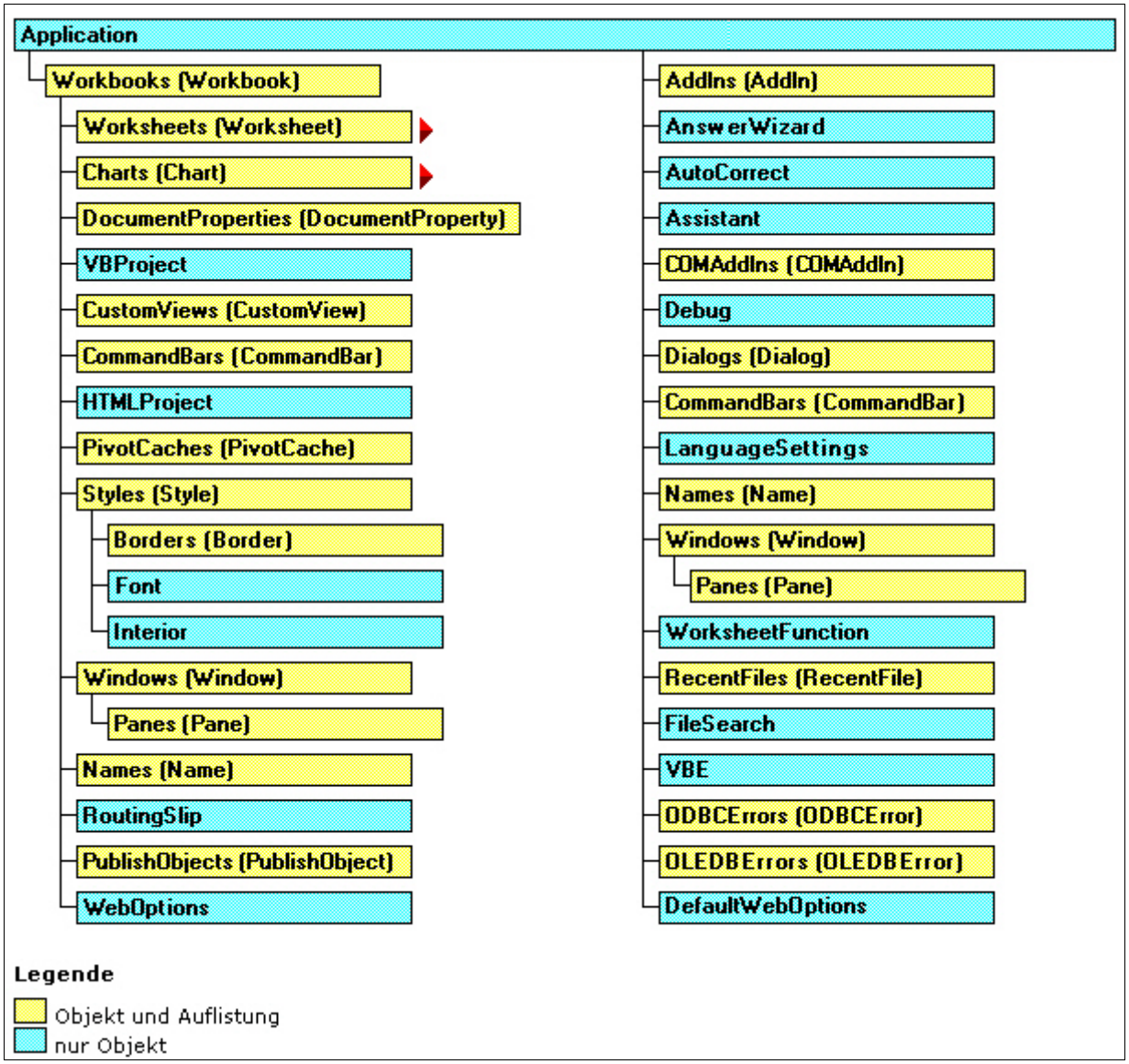
- viele der oben angeführten Links haben zum Inhalt dieses Dokuments beigetragen
- ein Teil wurde den Hilfetexten von VBA entnommen
- alle objektorientierten Beispiele entstammen: http://xtremes.stat.math.uni-siegen.de/~michael/lehre/prakws03/skript.pdf. Vielen Dank für die gute Arbeit !
- W. Doberenz, T. Kowalski: Microsoft Access-Programmierung. Microsoft Press, 2001.
- Access – Automatisierung, Programmierung. RRZN-Heft, 3. Auflage, 1999.

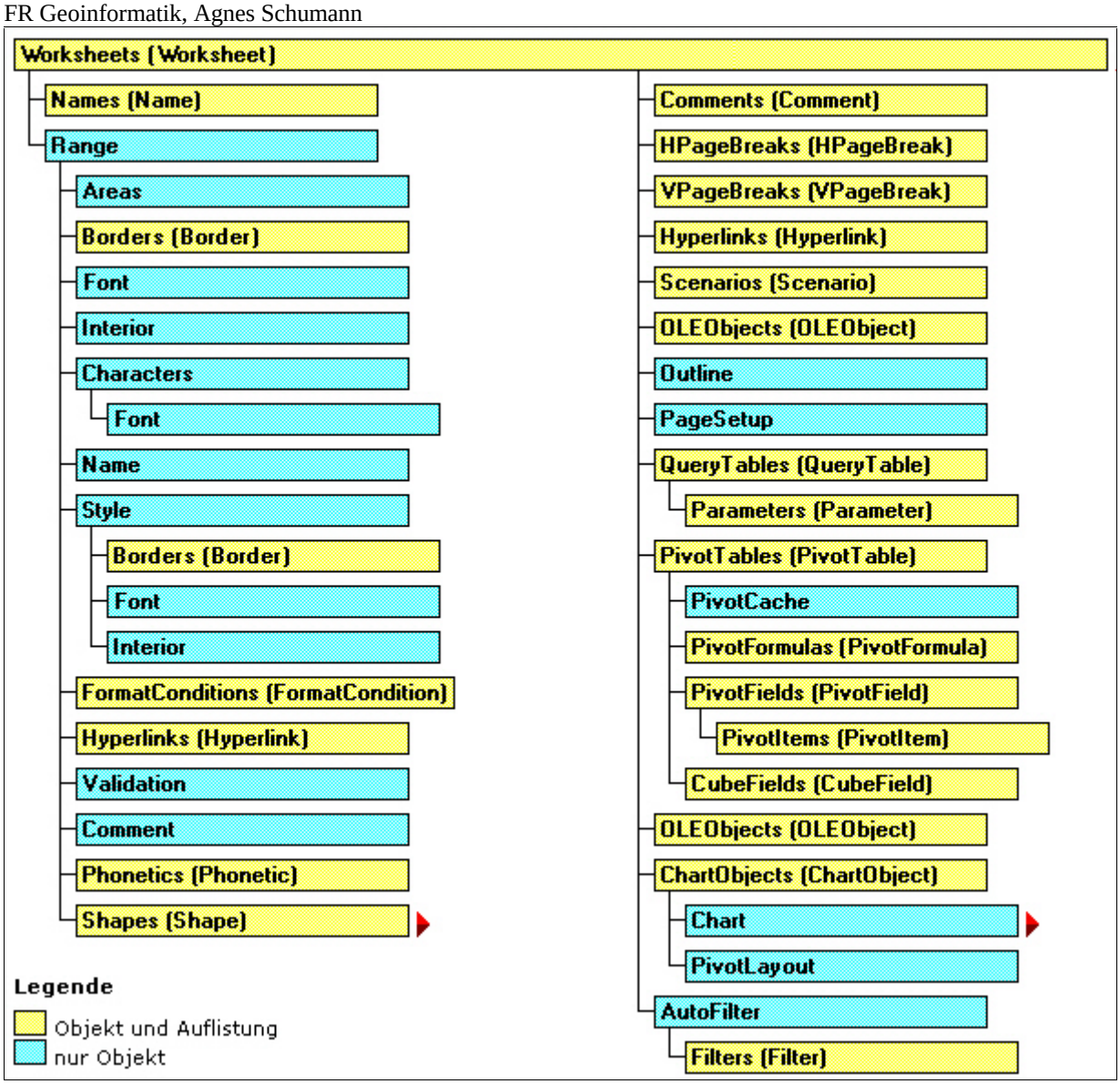
Anhang

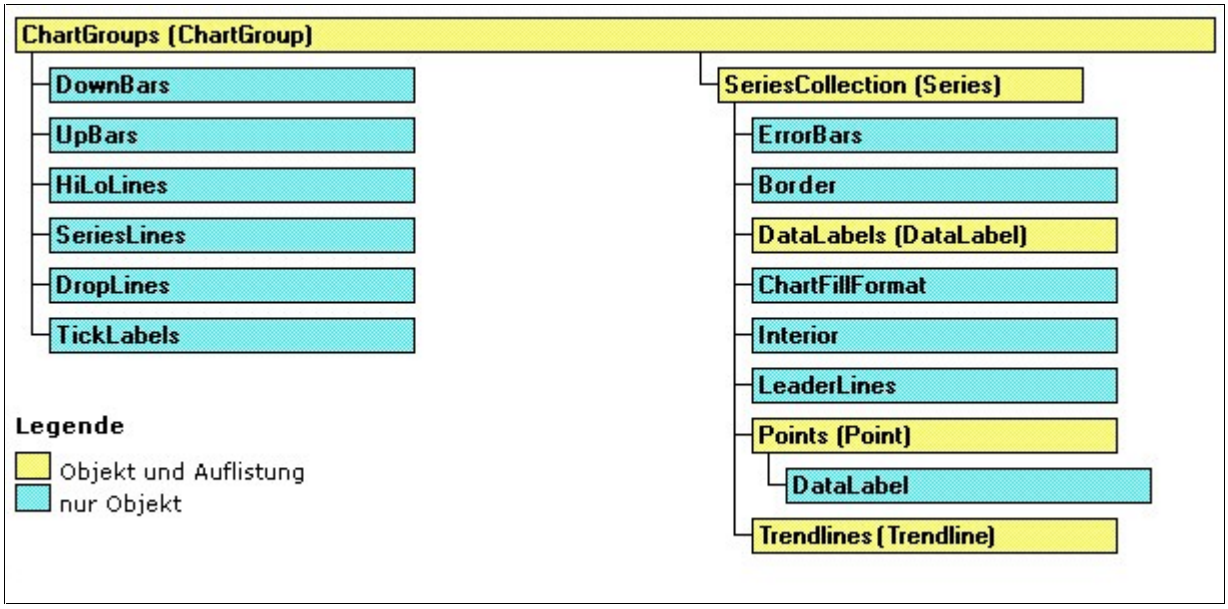
Excel-Objekthierarchie

Excel besteht aus mehr als 200 hierarchisch angeordneten Objekten, deren Eigenschaften und Methoden frei zugänglich sind. Objekte höherer Stufen beinhalten Objekte untergeordneter Stufen. Um auf ein Objekt zuzugreifen ist die Kenntnis der Position eines Objektes in der Hierarchie notwendig. Nachfolgend ein Auszug aus der Objekthierarchie von Excel.

(Menü VBA-Hilfe, Schlüsselwort: „Excel Objekte“ im Antwort-Assistenten oder „Microsoft_Excel“ im Suchindex eingeben und anschließend das Thema „Microsoft Excel-Objekte auswählen.)




Worksheets (Worksheet)
Names (Name)
Range
Areas
Borders (Border)
Font
Interior
Characters
Font
Name
Style
Borders (Border)
Font
Interior
FormatConditions (FormatCondition)
Hyperlinks (Hyperlink)
Validation
Comment
Phonetics (Phonetic)
Shapes (Shape)
Comments (Comment)
HPageBreaks (HPageBreak)
VPageBreaks (VPageBreak)
Hyperlinks (Hyperlink)
Scenarios (Scenario)
OLEObjects (OLEObject)
Outline
PageSetup
QueryTables (QueryTable)
Parameters (Parameter)
PivotTables (PivotTable)
PivotCache
PivotFormulas (PivotFormula)
PivotFields (PivotField)
PivotItems (PivotItem)
CubeFields (CubeField)
OLEObjects (OLEObject)
ChartObjects (ChartObject)
Chart
PivotLayout
AutoFilter
Filters (Filter)
Legende
Objekt und Auflistung
nur Objekt


ChartGroups (ChartGroup)
DownBars
UpBars
HiLoLines
SeriesLines
DropLines
TickLabels
SeriesCollection (Series)
ErrorBars
Border
DataLabels (DataLabel)
ChartFillFormat
Interior
LeaderLines
Points (Point)
DataLabel
Trendlines (Trendline)
Legende
Objekt und Auflistung
nur Objekt

Deutsch - Englisch

Excel 5.0 und 7.0 (95) bot deutsche Begriffe für die VBA-Makrosprache an. Seit Excel 8.0 (97) sind im VBA-Code nur noch englische Schlüsselwörter erlaubt. Alte deutschsprachige Dateien werden beim Öffnen automatisch übersetzt. Übersetzungsfehler müssen manuell beseitigt werden. (siehe http://www.roehrenbacher.at/download/erc_vbaliste.htm oder http://home.arcor.de/setup/). Hier ist eine Liste der Tabellenfunktionen, da deren Bezeichnungen oft nur in deutsch bekannt sind.

Deutsch A-L	Englisch A-L	Deutsch M-Z	Englisch M-Z
ABRUNDEN	ROUNDDOWN	MAX	MAX
ABS	ABS	MDET	MDETERM
ACHSENABSCHNITT	INTERCEPT	MEDIAN	MEDIAN
ADRESSE	ADDRESS	MIN	MIN
ANZAHL	COUNT	MINUTE	MINUTE
ANZAHL2	COUNTA	MINV	MINVERSE
ANZAHLLEEREZELLEN	COUNTBLANK	MITTELABW	AVEDEV
ARCCOS	ACOS	MITTELWERT	AVERAGE
ARCCOSHYP	ACOSH	MMULT	MMULT
ARCSIN	ASIN	MODALWERT	MODE
ARCSINHYP	ASINH	MONAT	MONTH
ARCTAN	ATAN	MTRANS	TRANSPOSE
ARCTAN2	ATAN2	N	N
ARCTANHYP	ATANH	NBW	NPV
AUFRUFEN	CALL	NEGBINOMVERT	NEGBINOMDIST
AUFRUNDEN	ROUNDUP	NICHT	NOT
BEREICH.VERSCHIEBEN	OFFSET	NORMINV	NORMINV
BEREICHE	AREAS	NORMVERT	NORMDIST
BESTIMMTHEITSMASS	RSQ	NV	NA
BETAINV	BETAINV	OBERGRENZE	CEILING
BETAVERT	BETADIST	ODER	OR
BINOMVERT	BINOMDIST	PEARSON	PEARSON
BOGENMASS	RADIANS	PI	PI
BW	PV	POISSON	POISSON
CHIINV	CHIINV	POTENZ	POWER
CHITEST	CHITEST	PRODUKT	PRODUCT
CHIVERT	CHIDIST	QIKV	MIRR
CODE	CODE	QUADRATESUMME	SUMSQ
COS	COS	QUANTIL	PERCENTILE
COSHYP	COSH	QUANTILSRANG	PERCENTRANK
DATUM	DATE	QUARTILE	QUARTILE
DATWERT	DATEVALUE	RANG	RANK
DBANZAHL	DCOUNT	RECHTS	RIGHT
DBANZAHL2	DCOUNTA	REGISTER.KENNUMMER	REGISTER.ID
DBAUSZUG	DGET	REST	MOD
DBMAX	DMAX	RGP	LINEST
DBMIN	DMIN	RKP	LOGEST
DBMITTELWERT	DAVERAGE	RMZ	PMT
DBPRODUKT	DPRODUCT	RÖMISCH	ROMAN
DBSTDABW	DSTDEV	RUNDEN	ROUND
DBSTDABWN	DSTDEVP	SÄUBERN	CLEAN
DBSUMME	DSUM	SCHÄTZER	FORECAST
DBVARIANZ	DVAR	SCHIEFE	SKEW
DBVARIANZEN	DVARP	SEKUNDE	SECOND
DIA	SYD	SIN	SIN
DM	DOLLAR	SINHYP	SINH
ERSETZEN	REPLACE	SPALTE	COLUMN
EXP	EXP	SPALTEN	COLUMNS
EXPONVERT	EXPONDIST	STABW	STDEV
FAKULTÄT	FACT	STABWN	STDEVP
FALSCH	FALSE	STANDARDISIERUNG	STANDARDIZE
FEHLER.TYP	ERROR.TYPE	STANDNORMINV	NORMSINV
FEST	FIXED	STANDNORMVERT	NORMSDIST
FINDEN	FIND	STEIGUNG	SLOPE

FINV	FINV	STFEHLERYX	STEYX
FISHER	FISHER	STUNDE	HOUR
FISHERINV	FISHERINV	SUCHEN	SEARCH
FTEST	FTEST	SUMME	SUM
FVERT	FDIST	SUMMENPRODUKT	SUMPRODUCT
GAMMAINV	GAMMAINV	SUMMEWENN	SUMIF
GAMMALN	GAMMALN	SUMMEX2MY2	SUMX2MY2
GAMMAVERT	GAMMADIST	SUMMEX2PY2	SUMX2PY2
GANZZAHL	INT	SUMMEXMY2	SUMXMY2
GDA	DDB	SUMQUADABW	DEVSQ
GDA2	DB	SVERWEIS	VLOOKUP
GEOMITTEL	GEOMEAN	T	T
GERADE	EVEN	TAG	DAY
GESTUTZTMITTEL	TRIMMEAN	TAGE360	DAYS360
GLÄTTEN	TRIM	TAN	TAN
GRAD	DEGREES	TANHYP	TANH
GROSS	UPPER	TEIL	MID
GROSS2	PROPER	TEILERGEBNIS	SUBTOTAL
GTEST	ZTEST	TEXT	TEXT
HARMITTEL	HARMEAN	TINV	TINV
HÄUFIGKEIT	FREQUENCY	TREND	TREND
HEUTE	TODAY	TTEST	TTEST
HYPGEOMVERT	HYPGEOMDIST	TVERT	TDIST
IDENTISCH	EXACT	TYP	TYPE
IKV	IRR	UND	AND
INDEX	INDEX	UNGERADE	ODD
INDIREKT	INDIRECT	UNTERGRENZE	FLOOR
INFO	INFO	VARIANZ	VAR
ISTBEZUG	ISREF	VARIANZEN	VARP
ISTFEHL	ISERR	VARIATION	GROWTH
ISTFEHLER	ISERROR	VARIATIONEN	PERMUT
ISTKTEXT	ISNONTEXT	VDB	VDB
ISTLEER	ISBLANK	VERGLEICH	MATCH
ISTLOG	ISLOGICAL	VERKETTEN	CONCATENATE
ISTNV	ISNA	VERWEIS	LOOKUP
ISTTEXT	ISTEXT	VORZEICHEN	SIGN
ISTZAHL	ISNUMBER	WAHL	CHOOSE
JAHR	YEAR	WAHR	TRUE
JETZT	NOW	WAHR	TRUE
KAPZ	PPMT	WAHRSCHBEREICH	PROB
KGRÖSSTE	LARGE	WECHSELN	SUBSTITUTE
KKLEINSTE	SMALL	WEIBULL	WEIBULL
KLEIN	LOWER	WENN	IF
KOMBINATIONEN	COMBIN	WERT	VALUE
KONFIDENZ	CONFIDENCE	WIEDERHOLEN	REPT
KORREL	CORREL	WOCHENTAG	WEEKDAY
KOVAR	COVAR	WURZEL	SQRT
CRITBINOM	KRITBINOM	WVERWEIS	HLOOKUP
KURT	KURT	ZÄHLENWENN	COUNTIF
KÜRZEN	TRUNC	ZEICHEN	CHAR
LÄNGE	LEN	ZEILE	ROW
LIA	SLN	ZEILEN	ROWS
LINKS	LEFT	ZEIT	TIME
LN	LN	ZEITWERT	TIMEVALUE
LOG	LOG	ZELLE	CELL
LOG10	LOG10	ZINS	RATE
LOGINV	LOGINV	ZINSZ	IPMT
LOGNORMVERT	LOGNORMDIST	ZUFALLSZAHL	RAND
		ZW	FV
		ZZR	NPER