Anwendung	# Excel + VBA
Autor	Dipl.-Ing. Harald Nahrstedt
Thema	**Ressourcenplanung**
Version	1.0
Erstellungsdatum	05.07.2019
Überarbeitung	
Ergänzung zu meinen Büchern	Die Welt der VBA-Objekte
Beschreibung	Mithilfe des Outlook-Kalenders lassen sich Ressourcen verplanen. In Verbindung mit Excel lassen sich darauf eine Datensicherung und -auswertung erstellen.

Neben den Objekten in Excel kommen hier auch die Objekte in Outlook zum Einsatz.

1 Termine im Outlook-Kalender nach Excel übertragen

Im ersten Schritt sollen die Termine aus dem Standard-Terminkalender von Outlook nach Excel übertragen werden.

Bild 1:

Kalender-Objekt im Outlook-Ordner

Durch die späte Objektbindung zu den Outlook-Objekten muss die Outlook-Anwendung auch nicht geöffnet sein. Voraussetzung zum Lesen der Kalendereinträge ist, dass der Kalender, wie in Bild 1 dargestellt, auch den Namen *Kalender* besitzt. Natürlich können auch andere Namen verwendet werden. Dann muss der Programmcode entsprechend angepasst werden.

Projekt: Ressourcenplanung_1.xlsm

Modul: modLesen

```
Private Sub TermineLesen()
   Dim wshTab        As Worksheet
   Dim objOutlook    As Object
   Dim objSpace      As Object
   Dim objFolder     As Object
   Dim objItem       As Object
   Dim lRow          As Long
   Dim sName         As String
   Dim sText         As String
   Dim vReturn       As Variant

   Set objOutlook = CreateObject("Outlook.Application")
   Set objSpace = objOutlook.GetNamespace("MAPI")
   Set objFolder = objSpace.GetDefaultFolder(9)
   sName = "Kalender"
   If SheetExists(sName) Then
      Set wshTab = Worksheets(sName)
   Else
      Set wshTab = Worksheets.Add(After:=Sheets(Sheets.Count))
```

```
      wshTab.Name = sName
   End If
   wshTab.Cells.ClearContents
   Load frmInfo
   frmInfo.Show 0
   lRow = 0
   For Each objItem In objFolder.Items
      With objItem
         lRow = lRow + 1
         wshTab.Cells(lRow, 1) = .Subject
         wshTab.Cells(lRow, 2) = .Location
         wshTab.Cells(lRow, 3) = .Start
         wshTab.Cells(lRow, 4) = .End
         wshTab.Cells(lRow, 5) = .Duration / 60
         wshTab.Cells(lRow, 6) = .Body
      End With
   Next
   Unload frmInfo
   wshTab.Visible = xlSheetHidden
   Set wshTab = Nothing
   Set objSpace = Nothing
   Set objOutlook = Nothing
   With Worksheets("Cover")
      .Activate
      .Unprotect "*"
      .Range("Sicherung") = Format(Date, "DD/MM/YYYY")
      .Protect "*"
   End With
   sText = Str(lRow) & " Termine gelesen !"
   vReturn = MsgBox(sText, vbInformation + vbOKOnly)
End Sub

Function SheetExists(sName As String) As Boolean
   On Error Resume Next
   SheetExists = Worksheets(sName).Index > 0
End Function
```

Mit der Methode *GetDefaultFolder* des Space-Objekts wird das Kalenderobjekt *objFolder* erstellt. Da das Space-Objekt über verschiedene Unterobjekte verfügt muss ein entsprechender Indes angegeben werden. Bei einer frühen Bindung kann die Outlook-Konstante *olFolderCalendar* gesetzt werden. Da hier aber die späte Bindung verwendet wird, kann nur der Wert der Konstanten, hier 9 verwendet werden. Tabelle 1 im Anhang zeigt weitere Outlook-Konstante.

Die Prozedur prüft, ob es ein Tabellenblatt gleichen Namens, wie ihn der Standard-Kalender hat, gibt. Diese Aufgabe übernimmt die Funktion *SheetExists*. Ist es nicht vorhanden, dann wird es erzeugt. Alle Einträge in diesem Tabellenblatt werden gelöscht, so dass eine Übertragung wiederholt werden kann. Da eine Übertragung bei vielen Daten länger dauert, wird während des Übertragungsprozesses ein Formular eingeblendet.

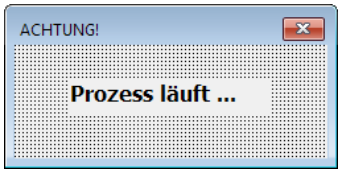

Bild 2:

Formular zum Prozessablauf

Übertragen werden nur die wichtigsten Eigenschaften eines Eintrags *objItem*. Weitere Eigenschaften finden wir im Objektkatalog von Outlook unter *Appointment*. Zum Schluss wird noch das aktuelle Datum im *Cover* eingetragen.

2 Termine aus der Excel-Tabelle zum Outlook-Kalender übertragen

Die im Excel-Tabellenblatt gesicherten Termine, können bei Bedarf geändert, gelöscht und ergänzt werden. Danach ist eine Übertragung nach Outlook nötig, auch auf ein Outlook auf einem anderen PC. Die nachfolgende Prozedur erfüllt diese Aufgabe.

Projekt: Ressourcenplanung_1.xlsm

Modul: modEintragen

```
Private Sub TermineEintragen()
   Dim wshTab        As Worksheet
   Dim objOutlook    As Object
   Dim objSpace      As Object
   Dim objFolder     As Object
   Dim objItem       As Object
   Dim lRow          As Long
   Dim lRowMax       As Long
   Dim lCount        As Long
   Dim sName         As String
   Dim vDate         As Variant
   Dim lExist        As Long
   Dim sText         As String
   Dim vReturn       As Variant

   Set objOutlook = CreateObject("Outlook.Application")
   Set objSpace = objOutlook.GetNamespace("MAPI")
   Set objFolder = objSpace.GetDefaultFolder(9)
   sName = "Kalender"
   If SheetExists(sName) Then
      Set wshTab = Worksheets(sName)
      lRowMax = wshTab.UsedRange.Rows.Count
'Termine vorhanden
      lExist = 0
      For lRow = 1 To lRowMax
         If Trim(wshTab.Cells(lRow, 3)) <> "" Then
            lExist = lExist + 1
         End If
      Next lRow
      sText = Str(lExist) & " Termine nach " & sName & " übertragen?"
      If MsgBox(sText, vbQuestion + vbYesNo) = vbNo Or _
         lExist = 0 Then
         Exit Sub
      End If
      Load frmInfo
      frmInfo.Show 0
```

```
'alle vorhandenen Termine löschen
      Do While objFolder.Items.Count > 0
         objFolder.Items(1).Delete
      Loop
'Termine übertragen
      lCount = 0
      For lRow = 1 To lRowMax
         If Trim(wshTab.Cells(lRow, 3)) <> "" Then
            Set objItem = objFolder.Items.Add
            With objItem
               .Subject = wshTab.Cells(lRow, 1)
               .Location = wshTab.Cells(lRow, 2)
               .Start = wshTab.Cells(lRow, 3)
               .End = wshTab.Cells(lRow, 4)
               .Duration = wshTab.Cells(lRow, 5) * 60
               .Body = wshTab.Cells(lRow, 6)
               .Save
            End With
            lCount = lCount + 1
            Set objItem = Nothing
         End If
      Next
      Set wshTab = Nothing
      Unload frmInfo
   End If
   Set objSpace = Nothing
   Set objOutlook = Nothing
   With Worksheets("Cover")
      .Activate
      .Unprotect "*"
      .Range("Übertragung") = Format(Date, "DD/MM/YYYY")
      .Protect "*"
   End With
   sText = Str(lCount) & " Termine übertragen !"
   vReturn = MsgBox(sText, vbInformation + vbOKOnly)
End Sub
```

Vor der Übertragung der Daten müssen die alten Einträge im Outlook-Kalender gelöscht werden. Auch dieser Termin wird im Cover vermerkt.

3 Alte Termine im Outlook-Kalender löschen

Mit der Zeit sammeln sich im Outlook-Kalender immer mehr Daten an. Im Cover der Anwendung ist daher ein Eintrag zum Terminzeitraum vorgesehen.

Bild 3:

Cover der Anwendung

Im Beispiel sind 5 Jahre angegeben. Die nachfolgende Prozedur liest diesen Eintrag (mit dem Bereichsnamen *Jahre*) und löscht alle Termine vor diesem Zeitraum, bezogen auf das aktuelle Jahr.

Projekt: Ressourcenplanung_1.xlsm

Modul: modLöschen

```
Private Sub TermineLöschen()
   Dim objOutlook    As Object
   Dim objSpace      As Object
   Dim objFolder     As Object
   Dim objItem       As Object
   Dim lRow          As Long
   Dim lCount        As Long
   Dim sName         As String
   Dim vJahr         As Variant
   Dim vOld          As Variant
   Dim sText         As String
   Dim vReturn       As Variant

   vJahr = Year(Date)
   vOld = vJahr - Val(Worksheets("Cover").Range("Jahre"))
   sText = "Alle Termine vor" & Str(vOld) & " löschen ?"
   If MsgBox(sText, vbQuestion + vbYesNo) = vbYes Then
      Set objOutlook = CreateObject("Outlook.Application")
      Set objSpace = objOutlook.GetNamespace("MAPI")
      Set objFolder = objSpace.GetDefaultFolder(9)
      Load frmInfo
      frmInfo.Show 0
      lCount = 0
      For Each objItem In objFolder.Items
         With objItem
            If Year(.Start) < vOld Then
               lCount = lCount + 1
               objItem.Delete
```

```
            End If
         End With
      Next
      Unload frmInfo
      Set objSpace = Nothing
      Set objOutlook = Nothing
      sText = Str(lCount) & " Termine vor" & Str(vOld) & " gelöscht !"
      vReturn = MsgBox(sText, vbInformation + vbOKOnly)
   End If
End Sub
```

4 Ressourcen als eigene Kalender anlegen

Zusätzlich zum vorhandenen Standard-Kalender in Outlook gibt es die Möglichkeit, weitere Outlook-Kalender zu erstellen. Dazu wird das Kontextmenü des Standard-Kalenders aufgerufen - Klick mit der rechten Maustaste auf den Ordner Kalender.

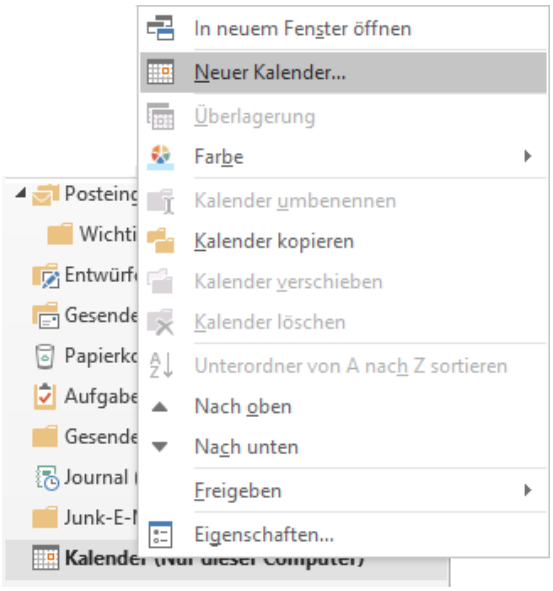

Bild 4:

Kontextmenü zum Kalender

Mit der Methode *Neuer Kalender…* im Kontextmenü kann dann ein neuer Unterordner erstellt werden. Der Name kann auch nach der Erstellung über das Kontextmenü geändert werden.

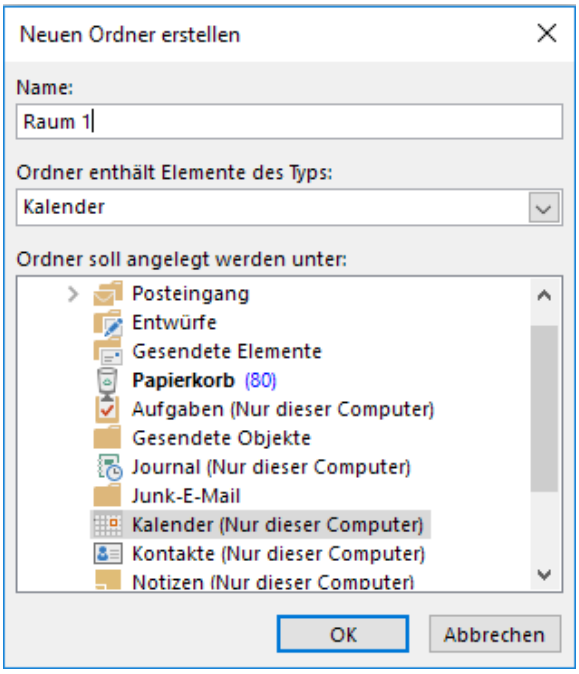

Bild 5:

Neuen Ordner erstellen

In diesem Beispiel verwende ich vier Räume und ihre Belegung (Bild 6). Es können aber auch andere Ressourcen wie Personen, Fahrzeuge, Rechner, etc. verwendet werden.

Bild 6:

Neuen Ordner erstellen

5 Termine in eigenen Kalendern nach Excel-Tabellen übertragen

Der Aufbau der Prozedur ist ähnlich wie beim Standardkalender, allerding müssen alle eigenen Kalender erfasst und in eigenen Tabellenblättern abgespeichert werden. Diese befinden sich in der Objektliste Folders unter dem Standard-Kalender.

Projekt: Ressourcenplanung_2.xlsm

Modul: modLesen

```
Private Sub TermineLesen()
   Dim wshTab        As Worksheet
   Dim objOutlook    As Object
```

```
   Dim objSpace      As Object
   Dim objFolder     As Object
   Dim objItem       As Object
   Dim lRow          As Long
   Dim lCount        As Long
   Dim sName         As String
   Dim sText         As String
   Dim vReturn       As Variant

   Set objOutlook = CreateObject("Outlook.Application")
   Set objSpace = objOutlook.GetNamespace("MAPI")
   lCount = 0
   For Each objFolder In objSpace.Session.GetDefaultFolder(9).Folders
      sName = objFolder.Name
      If SheetExists(sName) Then
         Set wshTab = Worksheets(sName)
      Else
         Set wshTab = Worksheets.Add(After:=Sheets(Sheets.Count))
         wshTab.Name = sName
      End If
      wshTab.Cells.ClearContents
      lRow = 0
      For Each objItem In objFolder.Items
         With objItem
            lCount = lCount + 1
            lRow = lRow + 1
            wshTab.Cells(lRow, 1) = .Subject
            wshTab.Cells(lRow, 2) = .Location
            wshTab.Cells(lRow, 3) = .Start
            wshTab.Cells(lRow, 4) = .End
            wshTab.Cells(lRow, 5) = .Duration / 60
            wshTab.Cells(lRow, 6) = .Body
         End With
      Next
      wshTab.Visible = xlSheetHidden
      Set wshTab = Nothing
   Next
   Set objSpace = Nothing
   Set objOutlook = Nothing
   With Worksheets("Cover")
      .Activate
      .Unprotect "*"
      .Range("Sicherung") = Format(Date, "DD/MM/YYYY")
      .Protect "*"
   End With
   sText = Str(lCount) & " Termine gelesen !"
   vReturn = MsgBox(sText, vbInformation + vbOKOnly)
End Sub

Function SheetExists(sName As String) As Boolean
   On Error Resume Next
   SheetExists = Worksheets(sName).Index > 0
End Function
```

Mithilfe einer *For-Each-Next*-Schleife werden die eigenen Kalender aus der Liste gelesen und in ein Tabellenblatt gleichen Namens gespeichert.

6 Termine aus Excel-Tabellen nach eigenen Kalendern übertragen

Für alle vorhandenen eigenen Kalender in Outlook werden entsprechende Tabellenblätter in Excel gesucht und wenn vorhanden deren Daten übertragen.

Projekt: Ressourcenplanung_2.xlsm

Modul: modEintragen

```
Private Sub TermineLesen()
   Dim wshTab        As Worksheet
   Dim objOutlook    As Object
   Dim objSpace      As Object
   Dim objFolder     As Object
   Dim objItem       As Object
   Dim lRow          As Long
   Dim lCount        As Long
   Dim sName         As String
   Dim sText         As String
   Dim vReturn       As Variant

   Set objOutlook = CreateObject("Outlook.Application")
   Set objSpace = objOutlook.GetNamespace("MAPI")
   lCount = 0
   For Each objFolder In objSpace.Session.GetDefaultFolder(9).Folders
      sName = objFolder.Name
      If SheetExists(sName) Then
         Set wshTab = Worksheets(sName)
      Else
         Set wshTab = Worksheets.Add(After:=Sheets(Sheets.Count))
         wshTab.Name = sName
      End If
      wshTab.Cells.ClearContents
      lRow = 0
      For Each objItem In objFolder.Items
         With objItem
            lCount = lCount + 1
            lRow = lRow + 1
            wshTab.Cells(lRow, 1) = .Subject
            wshTab.Cells(lRow, 2) = .Location
            wshTab.Cells(lRow, 3) = .Start
            wshTab.Cells(lRow, 4) = .End
            wshTab.Cells(lRow, 5) = .Duration / 60
            wshTab.Cells(lRow, 6) = .Body
         End With
      Next
      wshTab.Visible = xlSheetHidden
      Set wshTab = Nothing
   Next
   Set objSpace = Nothing
   Set objOutlook = Nothing
   With Worksheets("Cover")
      .Activate
      .Unprotect "*"
      .Range("Sicherung") = Format(Date, "DD/MM/YYYY")
      .Protect "*"
   End With
   sText = Str(lCount) & " Termine gelesen !"
   vReturn = MsgBox(sText, vbInformation + vbOKOnly)
End Sub

Function SheetExists(sName As String) As Boolean
```

```
   On Error Resume Next
   SheetExists = Worksheets(sName).Index > 0
End Function
```

7 Alte Termine in eigenen Kalendern löschen

Auch in dieser Anwendung verfügt das Cover um eine Zelle *Jahre*. Entsprechend diesem Eintrag werden alte Termine in allen eigenen Kalendern gelöscht.

Projekt: Ressourcenplanung_2.xlsm

Modul: modLöschen

```
Private Sub TermineLöschen()
   Dim objOutlook    As Object
   Dim objSpace      As Object
   Dim objFolder     As Object
   Dim objItem       As Object
   Dim lRow          As Long
   Dim sName         As String
   Dim vJahr         As Variant
   Dim vOld          As Variant
   Dim sText         As String
   Dim vReturn       As Variant

   vJahr = Year(Date)
   vOld = vJahr - Val(Worksheets("Cover").Range("Jahre"))
   sText = "Alle Termine vor" & Str(vOld) & " löschen ?"
   If MsgBox(sText, vbQuestion + vbYesNo) = vbYes Then
      Set objOutlook = CreateObject("Outlook.Application")
      Set objSpace = objOutlook.GetNamespace("MAPI")
      For Each objFolder In objSpace.Session.GetDefaultFolder(9).Folders
         For Each objItem In objFolder.Items
            With objItem
               If Year(.Start) < vOld Then
                  MsgBox .Subject
                  objItem.Delete
               End If
            End With
         Next
      Next
      Set objSpace = Nothing
      Set objOutlook = Nothing
      sText = "Alle Termine vor" & Str(vOld) & " gelöscht !"
      vReturn = MsgBox(sText, vbInformation + vbOKOnly)
   End If
End Sub
```

8 Anhang

Tabelle 1: OlDefaultFolders-Konstante

Konstante	Wert	Beschreibung
olFolderCalendar	9	Ordner Kalender
olFolderConflicts	19	Ordner Konflikte (Unterordner des Ordners Synchronisierungsprobleme). Nur bei einem Exchange-Konto verfügbar.
olFolderContacts	10	Ordner Kontakte
olFolderDeletedItems	3	Ordner Gelöschte Objekte
olFolderDrafts	16	Ordner Entwürfe
olFolderInbox	6	Ordner Posteingang
olFolderJournal	11	Ordner Journal
olFolderJunk	23	Ordner Junk-E-Mail
olFolderLocalFailures	21	Ordner Lokale Fehler (Unterordner des Ordners Synchronisierungsprobleme). Nur bei einem Exchange-Konto verfügbar.
olFolderManagedEmail	29	Ordner der obersten Ebene in der Gruppe Verwaltete Ordner. Weitere Informationen zu verwalteten Ordnern finden Sie in der Hilfe in Microsoft Outlook. Nur bei einem Exchange-Konto verfügbar.
olFolderNotes	12	Ordner Notizen
olFolderOutbox	4	Ordner Postausgang
olFolderSentMail	5	Ordner Gesendete Elemente
olFolderServerFailures	22	Ordner Serverfehler (Unterordner des Ordners Synchronisierungsprobleme). Nur bei einem Exchange-Konto verfügbar.
olFolderSuggestedContacts	30	Ordner Vorgeschlagene Kontakte
olFolderSyncIssues	20	Ordner Synchronisierungsprobleme. Nur bei einem Exchange-Konto verfügbar.
olFolderTasks	13	Ordner Aufgaben
olFolderToDo	28	Ordner Aufgaben
olPublicFoldersAllPublicFolders	18	Ordner Alle öffentlichen Ordner im Speicher Exchange Öffentliche Ordner. Nur bei einem Exchange-Konto verfügbar.
olFolderRssFeeds	25	Ordner RSS-Feeds