

# Einzelne und wiederkehrende Termine mit MS OUTLOOK überwachen

Dr. Volker Thormählen, 15. August 2019

## Inhalt

Abbildungen .....	II
Listings .....	II
Tabellen.....	III
1 Aufgabenstellung .....	1
2 EXCEL-Projekt .....	1
3 Office-Automation.....	2
4 Eingabedaten der Prozeduren .....	2
4.1 Prozedur 'Create_Single_Appointments' .....	2
4.2 Prozedur 'Create_Recurrent-Appointments' .....	2
5 Ablauf der Prozeduren .....	3
5.1 Prozedur 'Create_Single_Appointments' .....	3
5.2 Prozedur 'Create_Recurrent_Appointments' .....	4
6 Ausgabedaten der Prozeduren .....	4
6.1 Prozedur 'Create_Single_Appointments' .....	4
6.2 Prozedur 'Create_Recurrent_Appointments' .....	4
7 Geordnete Terminliste nach EXCEL exportieren.....	7
7.1 Startdatum bestimmen .....	7
7.2 Zugehörige Prozedur .....	7
7.3 Erfolgsmeldung.....	7
7.4 Geordnete Terminliste .....	7
8 Termine eines best. Kalenderdatums nach EXCEL exportieren .....	8
8.1 Kalenderdatum bestimmen .....	8
8.2 Zugehörige Prozedur .....	8
8.3 Meldungen der Prozedur.....	8
8.4 Terminübersicht .....	9
9 Einzeltermine als privat markieren.....	10
9.1 Zeitgrenze für abendliche Einzeltermine .....	10
9.2 Einzeltermine am Wochenende .....	10
9.3 Zugehörige Prozedur .....	10
9.4 Erfolgsmeldung der Prozedur .....	11
10 OUTLOOK-Projekt.....	12
11 Terminüberwachung in ThisOutLookSession.....	12
11.1 Ereignisprozeduren.....	12
11.2 Ereignisprozeduren zur Erinnerungen an Termine .....	13
12 Erinnerungen an Termine als Notiz ausgeben .....	14
13 Schlafende Erinnerungen anzeigen .....	14
14 VBA-Quellcode .....	II
14.1 VBA-Funktionen.....	II
14.2 Prozedur 'Create_Single_Appointments' .....	III
14.3 Prozedur 'Create_Recurrent_Appointments' .....	V
14.4 Prozedur 'Find_Appointments' .....	VII
14.5 Prozedur 'Export_Appointments' .....	IX
14.6 Prozedur 'Update_Appointments' .....	XII
14.7 Ereignisprozeduren bei Erinnerung an einem Termin .....	XIV
14.8 Prozedur 'GetAllExistingReminders' .....	XVI
14.9 Prozedur 'GetListofSnoozedReminders' .....	XVII
15 Literatur.....	XVIII

## Abbildungen

Abbildung 1: EXCEL-Projekt .....	1
Abbildung 2: Verweise setzen .....	2
Abbildung 3: Aufbau und Inhalt des Tabellenblattes ' <i>Single_Appointments_Data</i> ' .....	2
Abbildung 4: Aufbau und Inhalt des Tabellenblattes ' <i>Recurrent_Appointments_Data</i> ' .....	2
Abbildung 5: Terminkonflikt anzeigen .....	3
Abbildung 6: Fehlermeldung bei potentiell doppeltem Kalendereintrag .....	4
Abbildung 7: Erfolgsmeldung nach Erstellung von 3 einzelnen Terminen .....	4
Abbildung 8: Erfolgsmeldung nach Erzeugung von 2 Serienterminen .....	4
Abbildung 9: Fehlermeldung bei doppelten Kalendereinträgen .....	5
Abbildung 10: Die Terminserie f. tägliche Termine d. Herrn Groß endet nach 4 Terminen .....	5
Abbildung 11: Terminserie f. wöchentliche Termine d. Fr. Klein endet nach 10 Terminen .....	6
Abbildung 12: Startdatum für die Terminliste vorgegeben .....	7
Abbildung 13: Erfolgsmeldung der Prozedur .....	7
Abbildung 14: Ergebnis: Nach Startdatum geordnete Terminliste .....	7
Abbildung 15: Kalenderdatum vorgeben .....	8
Abbildung 16: Zahl der gefundenen Kalendereinträge .....	8
Abbildung 17: Erfolgsmeldung der Prozedur ' <i>Export_Appointments</i> ' .....	9
Abbildung 18: Aufbau und Inhalt der Terminübersicht für ein best. Kalenderdatum .....	9
Abbildung 19: Zeitgrenze für abendliche Einzeltermine .....	10
Abbildung 20: Erfolgsmeldung der Prozedur ' <i>Update_Appointments</i> ' .....	11
Abbildung 21: OUTLOOK-Projekt .....	12
Abbildung 22: Begrüßungsmeldung durch MS OUTLOOK .....	12
Abbildung 23: Abschiedsmeldung durch MS OUTLOOK .....	13
Abbildung 24: Eine E-Mail an sich selbst senden, sobald die Erinnerung ausgelöst wird .....	13
Abbildung 25: Liste der vorhandenen Termin-Erinnerungen als OUTLOOK-Notiz ausgeben .....	14
Abbildung 26: Latente Erinnerung als OUTLOOK-Notiz .....	14

## Listings

Listing 1: VBA-Funktionen .....	II
Listing 2: VBA-Prozedur zum Erstellen von einzelnen Kalendereinträgen in OUTLOOK .....	IV
Listing 3: VBA-Prozedur zum Erstellen von wiederkehrenden Kalendereinträgen in OUTLOOK .....	VI
Listing 4: OUTLOOK Termine in einem bestimmten 14-Tage Zeitraum nach EXCEL exportieren .....	VIII
Listing 5: OUTLOOK Termine eines bestimmten Kalendertages nach EXCEL exportieren .....	XI
Listing 6: Einzeltermine am Abend oder am Wochenende als 'privat' kennzeichnen. ....	XIII
Listing 7: Ereignisprozeduren bei Erinnerung an einem Termin .....	XV
Listing 8: Alle Erinnerungen an Termine als OUTLOOK-Notiz anzeigen .....	XVI
Listing 9: Liste der latenten Erinnerungen erstellen .....	XVII

## Tabellen

Tabelle 1: Zusammenhang der EXCEL-Module und -Tabellen .....	1
Tabelle 2: Verschlüsselung der Verfügbarkeit in OUTLOOK .....	2
Tabelle 3: Verschlüsselung der Serienmuster in OUTLOOK .....	3
Tabelle 4: Verschlüsselung der Wochentage in OUTLOOK .....	3
Tabelle 5: Einzelheiten des OUTLOOK-Projekts.....	12

## 1 Aufgabenstellung

In diesem Beitrag werden VBA-Prozeduren besprochen, die erlauben, Termindaten zwischen EXCEL und OUTLOOK auszutauschen. Ein EXCEL- und ein OUTLOOK-Projekt werden vorgestellt, die jeweils einmalige bzw. wiederkehrende Termine im OUTLOOK-Standardkalender betreffen.

Das EXCEL-Projekt umfasst sechs VBA<sup>1</sup>-Objekte:

- *Create\_Single\_Appointments*: Prozedur zur automatischen Erzeugung von *einmaligen* Terminen bzw. Ereignissen im OUTLOOK-Standardkalender.
- *Create\_Recurrent\_Appointments*: Prozedur zur automatischen Erzeugung von *wiederkehrenden* Terminen bzw. Ereignissen im OUTLOOK-Standardkalender.:
- *Export\_Appointments*: OUTLOOK Termine eines Kalenderdatums nach EXCEL exportieren.
- *Find\_Appointments*: OUTLOOK Termine eines Zeitraums nach EXCEL exportieren.
- *Update\_Appointments* markiert Einzeltermine automatisch als *privat*, wenn ihr jeweiliger Beginn nach einer vom Benutzer vorgegebenen Zeitgrenze liegt *oder* wenn ein Einzeltermin am Wochenende beginnt<sup>2</sup>.
- *VBA\_Functions* enthält Funktionen, die von den genannten Prozeduren benötigt werden. Gliederungspunkt 2 beschreibt die dazugehörigen Einzelheiten.

Das OUTLOOK-Projekt umfasst drei VBA-Prozeduren. Gliederungspunkt 10 (s. Seite 12) enthält Näheres.

Die Nutzung dieser insgesamt neun VBA-Objekte ist vorteilhaft, um bei Kalendereinträgen die Fehlerquote durch Plausibilitätsprüfungen zu senken und den Arbeitsaufwand durch Stapelverarbeitung zu minimieren.

## 2 EXCEL-Projekt

Einen Überblick über das VBA-Projekt in 'Termine.xlsm' beinhaltet Abbildung 1:

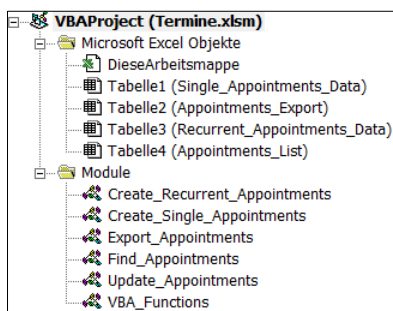


Abbildung 1: EXCEL-Projekt

Die Benennung der EXCEL-Module und -Tabellen lässt ihren Zusammenhang erkennen (s. Tabelle 1):

Name des Moduls	Name der Tabelle	Inhalt der Tabelle
Create_Recurrent_Appointments	Recurrent_Appointments_Data	Eingabedaten
Create_Single_Appointments	Single_Appointments_Data	Eingabedaten
Export_Appointments	Appointments_Export	Ausgabedaten
Find_Appointments	Appointments_List	Ausgabedaten
Update_Appointments	keine	keine
VBA_Functions	keine	keine

Tabelle 1: Zusammenhang der EXCEL-Module und -Tabellen

<sup>1</sup> Die Abkürzung **VBA** steht für **V**isual **B**asic for **A**pplications, eine proprietäre Programmiersprache der Firma Microsoft für die sog. *Office-Suite*.

<sup>2</sup> Für OUTLOOK-Benutzer sind an einem Wochenende beginnende Termine meistens *privat*.

### 3 Office-Automation

Zum Verbinden der MS Office Komponenten EXCEL und OUTLOOK müssen die Verweise in der EXCEL-Arbeitsmappe 'Termine' wie folgt gesetzt werden (s. Abbildung 2):

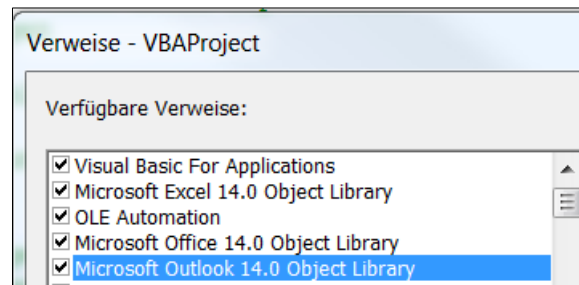


Abbildung 2: Verweise setzen

## 4 Eingabedaten der Prozeduren

### 4.1 Prozedur 'Create\_Single\_Appointments'

Die Prozedur 'Create\_Single\_Appointments' benötigt Eingabedaten aus dem Tabellenblatt 'Single\_Appointments\_Data' der Arbeitsmappe 'Termine' (s. Abbildung 3):

A	B	C	D	E	F	G	H
Betreff	Ort	Starttermin	Dauer(Min.)	Gebucht-Status	Erinnerungszeit(Min.)	Textkörper	Kategorie
Hr. Schmitz	Praxis	9.9.19 10:00	45	2	15	Sitzung mit	Orange Category
Fr. Müller	Praxis	10.9.19 13:00	15	2	15	Vorgespräch mit	Orange Category
Fr. Meier	Praxis	10.9.19 14:00	15	2	15	Vorgespräch mit	Orange Category
Hr. Franz	Praxis	10.9.19 14:00	15	2	15	Schlussgespräch mit	Orange Category

Abbildung 3: Aufbau und Inhalt des Tabellenblattes 'Single\_Appointments\_Data'

Die Spalte E des Tabellenblatts beinhaltet den sog. *Gebucht-Status* (d. h. die Verfügbarkeit eines Benutzers). Folgende Schlüssel werden dafür in OUTLOOK verwendet (s. Tabelle 2, Quelle: [1]):

Schlüssel	Name der Konstante	Beschreibung: Der Benutzer ...
0	olFree	... ist verfügbar
1	olTentative	Der Termin ist vorläufig
2	olBusy	... ist ausgelastet
3	olOutOfOffice	... befindet sich nicht im Büro
4	olWorkingElsewhere	... arbeitet sonst wo

Tabelle 2: Verschlüsselung der Verfügbarkeit in OUTLOOK

### 4.2 Prozedur 'Create\_Recurrent-Appointments'

Die Prozedur 'Create\_Recurrent\_Appointments' benötigt dementsprechend Eingabedaten aus dem Tabellenblatt 'Recurrent\_Appointments\_Data' der Arbeitsmappe 'Termine' (s. Abbildung 4):

Termine														
	A	B	C	D	E	F	G	H	I	J	K	L	M	N
1	Termin								Wiederholungen					
2	Betreff	Ort	Starttermin	Dauer (Min.)	Gebucht-Status	Erinnerungszeit(Min.)	Textkörper	Kategorie	Beginnt am	Beginnt um	Dauer(Min.)	Rhythmus	Wochentag	Wiederholungen
3	Hr. Groß	Praxis	10.9.19 9:00	15	2	15	Sitzung mit	Orange Category	10.09.19	9:00	45	täglich		4
4	Fr. Klein	Praxis	10.9.19 10:00	15	2	15	Sitzung mit	Orange Category	10.09.19	10:00	45	wöchentlich	Dienstag	10

Abbildung 4: Aufbau und Inhalt des Tabellenblattes 'Recurrent\_Appointments\_Data'

Die Spalte **L** des Tabellenblatts beinhaltet das Serienmuster. Folgende Serienmuster stehen in OUTLOOK 2010 zur Auswahl (s. Tabelle 3, Quelle: [2])

Serienmuster	Schlüsselt	Beschreibung
olRecursDaily	0	täglich
olRecursWeekly	1	wöchentlich
olRecursMonthly	2	monatlich
olRecursMontNth	3	jeden n-ten Monat
olRecursYearly	5	jährlich
olRecursYearNth	6	jedes n-te Jahr

**Tabelle 3: Verschlüsselung der Serienmuster in OUTLOOK**

Spalte **M** des Tabellenblatts beinhaltet den Wochentag beim Serienmuster 'wöchentlich'. Die Wochentage (*OlDaysOfWeek*) sind in OUTLOOK wie folgt verschlüsselt (s. Tabelle 4, Quelle: [3]):

Wochentag	Schlüssel	Konstante
Sonntag	1	olSunday
Montag	2	olMonday
Dienstag	4	olTuesday
Mittwoch	8	olWednesday
Donnerstag	16	olThursday
Freitag	32	olFriday
Samstag	64	olSaturday

**Tabelle 4: Verschlüsselung der Wochentage in OUTLOOK**

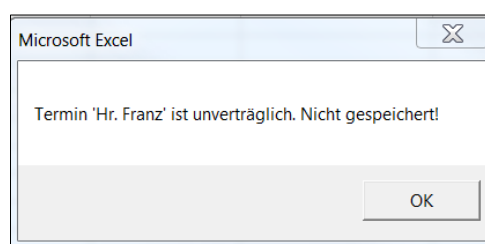
## 5 Ablauf der Prozeduren

### 5.1 Prozedur 'Create\_Single\_Appointments'

Der Ablauf dieser Prozedur kann anhand des kommentierten Quellcode nachvollzogen werden (s. Listing 2).

Wenn diese Prozedur versehentlich nochmals gestartet wird, entstehen keine Duplikate im Standardkalender von OUTLOOK. Verhindert wird dies durch den Aufruf der VBA-Funktion 'AppointmentExists' (s. Listing 1) vor der Speicherung des jeweiligen Kalendereintrags. Diese Funktion wird mit 3 Argumenten aufgerufen und gibt eine boolesche Variable zurück, je nachdem, ob der potentielle Kalendereintrag bereits vorhanden ist oder nicht.

Diese Prozedur benötigt auch die VBA-Funktion 'HasConflicts' (s. Listing 1). Durch ihren Aufruf wird verhindert, dass ein bereits gespeicherter Termin unverträglich ist mit einem neu hinzukommenden (s. Meldung in Abbildung 5).



**Abbildung 5: Terminkonflikt anzeigen**

## 5.2 Prozedur '*Create\_Recurrent\_Appointments*'

Der Quellcode dieser Prozedur befindet sich in Listing 3. Wenn ein/beide wiederkehrende Termine (s. Abbildung 4) bereits im Standardkalender von OUTLOOK enthalten ist/sind, wird/werden folgende Fehlermeldung/en ausgegeben (s. Abbildung 6). Die zuvor bereits erwähnte Funktion '*Appointment-Exists*' wird auch hier eingesetzt.

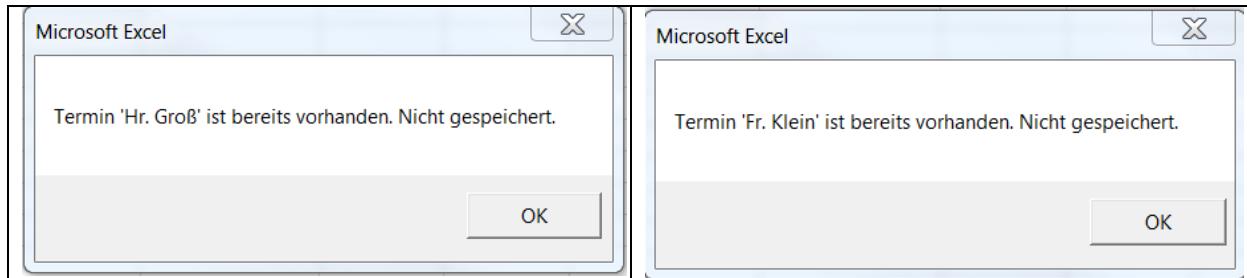


Abbildung 6: Fehlermeldung bei potentiell doppeltem Kalendereintrag

## 6 Ausgabedaten der Prozeduren

### 6.1 Prozedur '*Create\_Single\_Appointments*'

Abbildung 7 beinhaltet die Erfolgsmeldung dieser Prozedur:

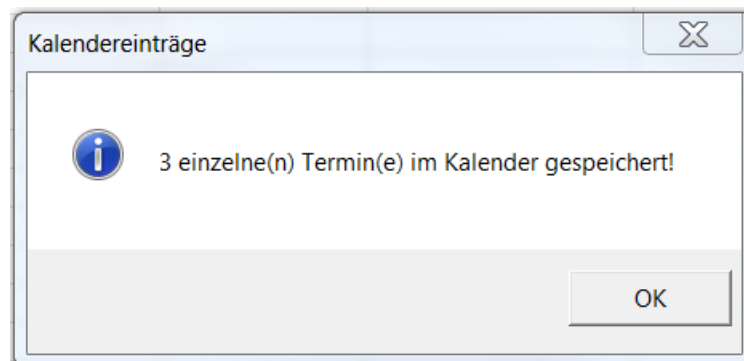


Abbildung 7: Erfolgsmeldung nach Erstellung von 3 einzelnen Terminen

### 6.2 Prozedur '*Create\_Recurrent\_Appointments*'

Abbildung 8 beinhaltet die Erfolgsmeldung dieser Prozedur.

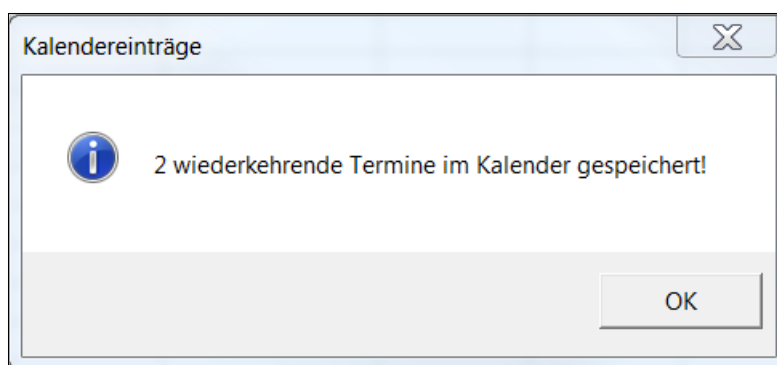


Abbildung 8: Erfolgsmeldung nach Erzeugung von 2 Serienterminen

Wenn die Prozedur potentielle Duplikate findet, lautet die Fehlermeldung (s. Abbildung 9):

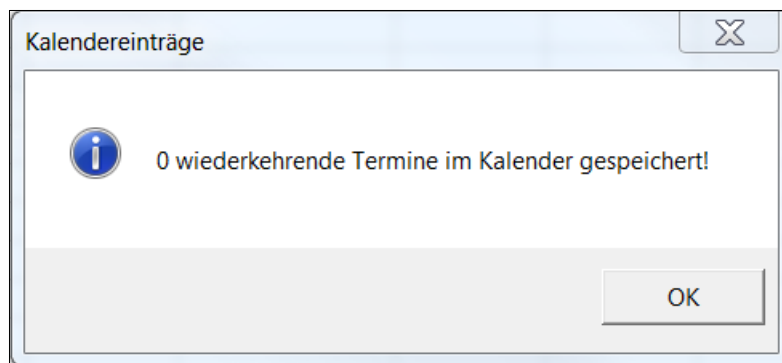


Abbildung 9: Fehlermeldung bei doppelten Kalendereinträgen

Die beiden folgenden Abbildungen (s. Abbildung 10 und Abbildung 11) weisen deutlich aus, welche wiederkehrenden Termine im OUTLOOK-Standardkalender durch diese Prozedur erzeugt werden.

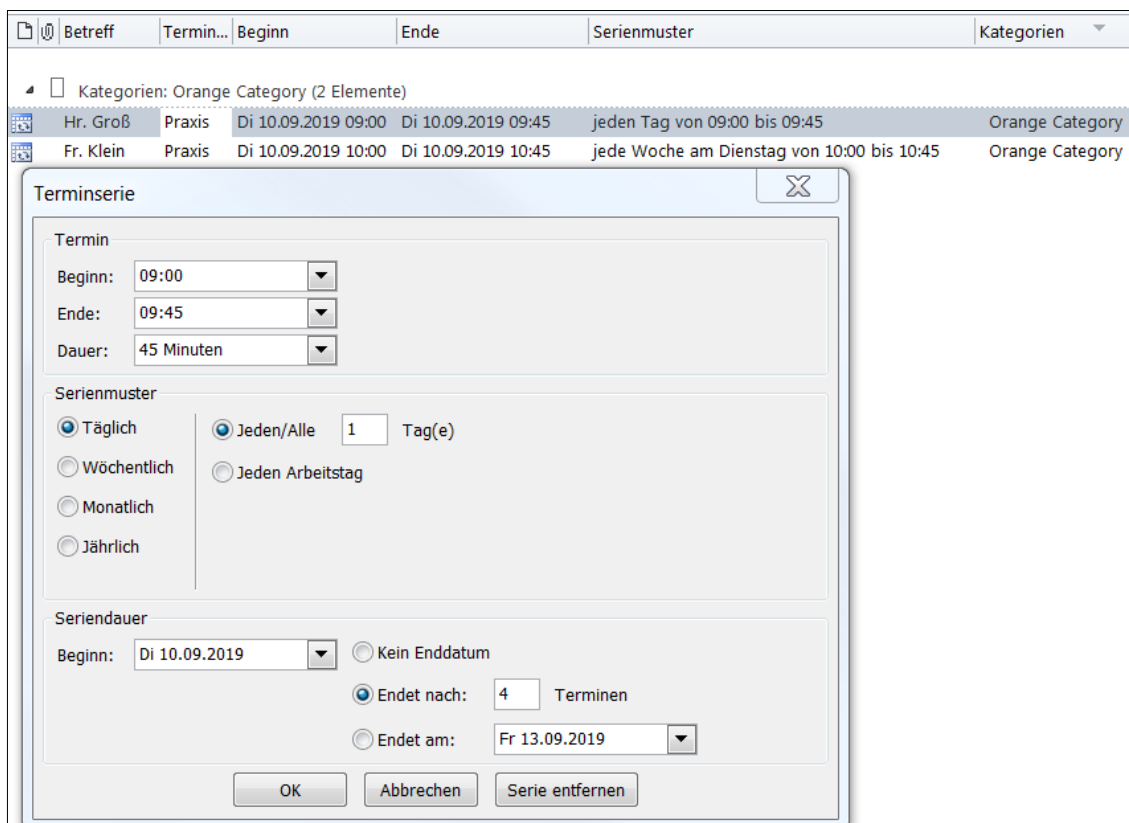


Abbildung 10: Die Terminserie f. tägliche Termine d. Herrn Groß endet nach 4 Terminen



Betreff	Termin...	Beginn	Ende	Serienmuster	Kategorien
Kategorien: Orange Category (2 Elemente)					
Hr. Groß	Praxis	Di 10.09.2019 09:00	Di 10.09.2019 09:45	jeden Tag von 09:00 bis 09:45	Orange Category
Fr. Klein	Praxis	Di 10.09.2019 10:00	Di 10.09.2019 10:45	jede Woche am Dienstag von 10:00 bis 10:45	Orange Category

Terminserie

Termin

Beginn: 10:00
Ende: 10:45
Dauer: 45 Minuten

Serienmuster

☐ Täglich

Jede/Alle 1 Woche(n) am

☐ Sonntag
☐ Montag
☒ Dienstag
☐ Mittwoch

☐ Wöchentlich
☐ Donnerstag
☐ Freitag
☐ Samstag

☐ Monatlich
☐ Jährlich

Seriendauer

Beginn: Di 10.09.2019

☐ Kein Enddatum
☒ Endet nach: 10 Terminen
☐ Endet am: Di 12.11.2019

OK

Abbrechen

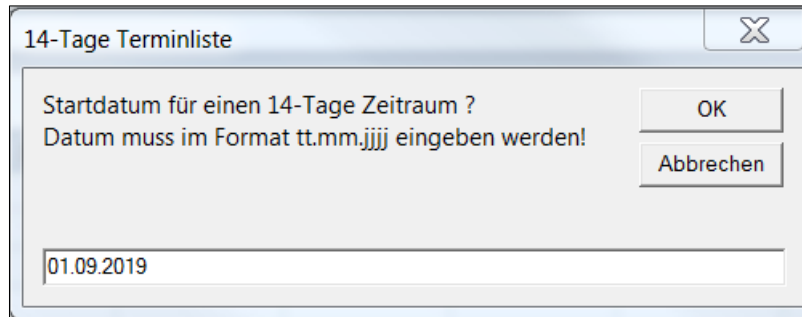
Serie entfernen

Abbildung 11: Terminserie f. wöchentliche Termine d. Fr. Klein endet nach 10 Terminen

## 7 Geordnete Terminliste nach EXCEL exportieren

Nunmehr wird demonstriert (s. Listing 4), wie eine geordnete Terminliste für einen 14-Tage Zeitraum erstellt und in ein bestimmtes EXCEL-Tabellenblatt exportiert werden kann<sup>3</sup>. Das zugehörige Startdatum kann vom Benutzer frei bestimmt werden (s. Abbildung 12).

### 7.1 Startdatum bestimmen



14-Tage Terminliste

Startdatum für einen 14-Tage Zeitraum ?  
Datum muss im Format tt.mm.jjjj eingegeben werden!

OK  
Abbrechen

01.09.2019

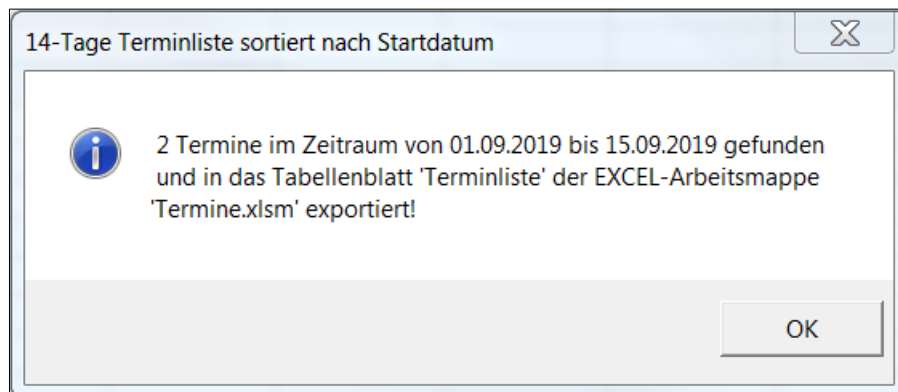
Abbildung 12: Startdatum für die Terminliste vorgegeben

### 7.2 Zugehörige Prozedur


Der Quellcode der zugehörigen Prozedur ist ausgiebig kommentiert (s. Listing 4).

### 7.3 Erfolgsmeldung

Am Ende der Prozedur wird eine Erfolgsmeldung ausgegeben (s. Abbildung 13):



14-Tage Terminliste sortiert nach Startdatum

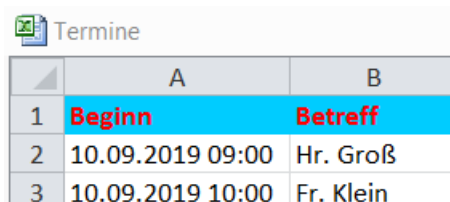
 2 Termine im Zeitraum von 01.09.2019 bis 15.09.2019 gefunden und in das Tabellenblatt 'Terminliste' der EXCEL-Arbeitsmappe 'Termine.xlsm' exportiert!

OK

Abbildung 13: Erfolgsmeldung der Prozedur

### 7.4 Geordnete Terminliste

Nach erfolgreichem Ablauf der Prozedur enthält das Tabellenblatt 'Appointments\_List' der EXCEL-Arbeitsmappe 'Termine.xlsm' die gewünschte Information (s. Abbildung 14):



	A	B
1	Beginn	Betreff
2	10.09.2019 09:00	Hr. Groß
3	10.09.2019 10:00	Fr. Klein

Abbildung 14: Ergebnis: Nach Startdatum geordnete Terminliste

<sup>3</sup> In Anlehnung an [5].

## 8 Termine eines best. Kalenderdatums nach EXCEL exportieren

Die Prozedur 'Export\_Appointments' (s. Listing 5) kann genutzt werden, um die Termine eines ausgewählten Kalenderdatums in ein bestimmtes EXCEL-Tabellenblatt zu exportieren. Das zugehörige Kalenderdatum kann vom Benutzer frei bestimmt werden (s. Abbildung 15).

### 8.1 Kalenderdatum bestimmen

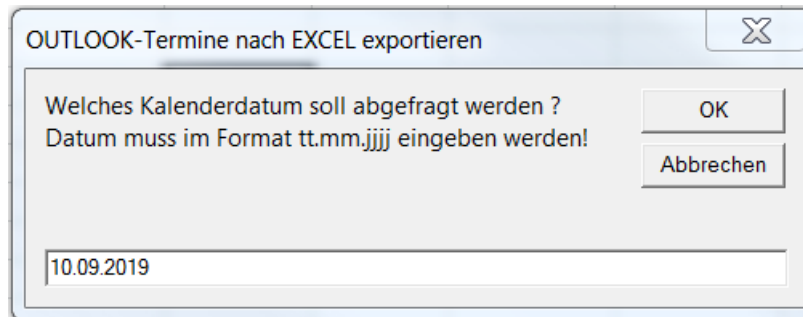


Abbildung 15: Kalenderdatum vorgeben

### 8.2 Zugehörige Prozedur

Der Quellcode der Prozedur 'Export\_Appointments' ist kommentiert (s. Listing 5).

### 8.3 Meldungen der Prozedur

Die Prozedur zeigt zunächst an, wie viele Kalendereinträge auf das eingegebene Kalenderdatum entfallen (s. Abbildung 16):

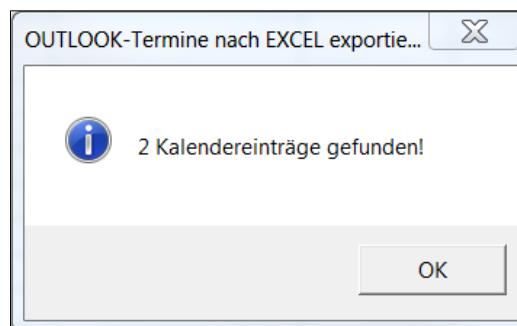


Abbildung 16: Zahl der gefundenen Kalendereinträge

Wenn keine relevanten Kalendereinträge gefunden werden, wird dies durch eine entsprechende Meldung angezeigt.

Im Erfolgsfall wird abschließend folgende Meldung angezeigt (s. Abbildung 17):

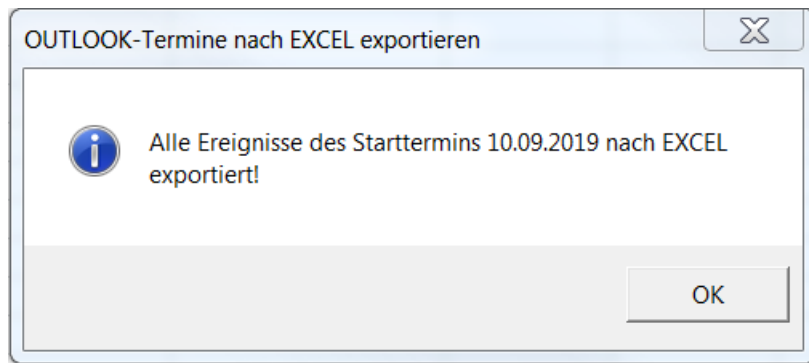


Abbildung 17: Erfolgsmeldung der Prozedur 'Export\_Appointments'

## 8.4 Terminübersicht

Als Ergebnis erzeugt die Prozedur 'Export\_Appointments' ein EXCEL-Tabellenblatt mit folgendem Aufbau und Inhalt (s. Abbildung 18) für den gewählten Kalendertag (s. Abbildung 15). Angezeigt werden die wiederkehrenden Termine von Herrn Groß und Frau Klein mit 'Beginn' am 10.09.2019. Die Spalte 'Rhythmus' weist den Turnus des Serienmusters aus. Die Spalte 'Häufigkeit' beinhaltet dementsprechend die Zahl der Wiederholungen.

Termine									
	A	B	C	D	E	F	G	H	I
1	Beginn	Dauer(Min.)	Ende	Ort	Betreff	Textkörper	Rhythmus	Wochentag	Häufigkeit
2	10.09.2019 09:00	45	13.09.2019	Praxis	Hr. Groß	Sitzung mit Hr. Groß	täglich		4
3	10.09.2019 10:00	45	12.11.2019	Praxis	Fr. Klein	Sitzung mit Fr. Klein	wöchentlich	Dienstag	10

Abbildung 18: Aufbau und Inhalt der Terminübersicht für ein best. Kalenderdatum

## 9 Einzeltermine als privat markieren

Die Prozedur *'Update\_Appointments'* (s. Listing 6) setzt die Eigenschaft *'privat'* von Einzelterminen im Standardkalender automatisch auf *'true'* wenn folgende Bedingungen erfüllt sind:

- Es handelt sich um einen Einzeltermin<sup>4</sup>
- Dieser Einzeltermin ist noch nicht als *'privat'* markiert
- Der entsprechende Beginn liegt vor einer frei gewählten abendlichen Zeitgrenze (z.B. 17:30 Uhr) *oder* beginnt an einem Wochenende.

Die erwähnte Zeitgrenze wird zu Beginn der Prozedur *'Update\_Appointments'* abgefragt.

### 9.1 Zeitgrenze für abendliche Einzeltermine

Die Zeitgrenze für abendliche Einzeltermine wird im Dialog vom Benutzer bestimmt (s. Abbildung 19):

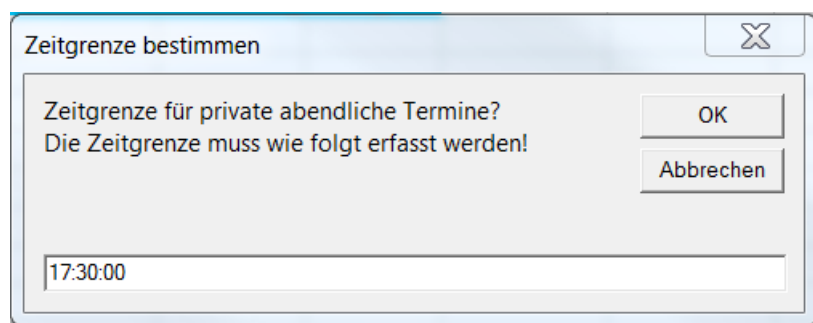


Abbildung 19: Zeitgrenze für abendliche Einzeltermine

### 9.2 Einzeltermine am Wochenende

Ob ein Termin am Wochenende beginnt, wird mit der Funktion *'IsWeekendAppointment'* (s. Listing 1) bestimmt. Diese Funktion wird von der Prozedur *'Update\_Appointments'* (s. Listing 6) aufgerufen.

### 9.3 Zugehörige Prozedur

Der Quellcode der Prozedur *'Update\_Appointments'* ist ausgiebig kommentiert (s. Listing 6).

---

<sup>4</sup> Also **kein** Serientermin.

## 9.4 Erfolgsmeldung der Prozedur

Am Ende gibt die Prozedur eine Erfolgsmeldung (s. Abbildung 20) aus, weil ein gefundener Einzeltermin *nach* der vorgegebenen Zeitgrenze (s. Abbildung 19) *oder* am Wochenende beginnt.

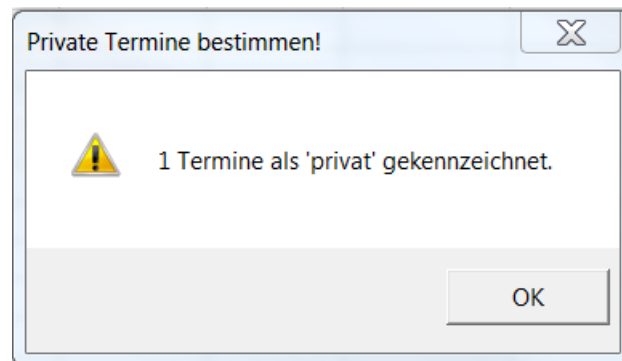


Abbildung 20: Erfolgsmeldung der Prozedur 'Update\_Appointments'

## 10 OUTLOOK-Projekt

Im Folgenden werden drei VBA-Prozeduren vorgeführt, die nicht zum EXCEL-Projekt (s. Abbildung 1) gehören, sondern zu einem OUTLOOK-Projekt (s. Abbildung 21):

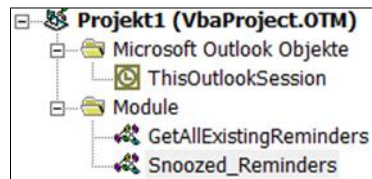


Abbildung 21: OUTLOOK-Projekt

Tabelle 5 enthält die Einzelheiten des OUTLOOK-Projekts:

Name des Moduls	Aufgabe
ThisOutlookSession	Terminüberwachung mit Ereignisprozeduren
GetAllExistingReminders	Termine mit vorhandenen Erinnerungen als Notiz anzeigen
Snoozed_Reminders	Schlafende (d. h. latente) Erinnerungen als Notiz anzeigen

Tabelle 5: Einzelheiten des OUTLOOK-Projekts

## 11 Terminüberwachung in ThisOutlookSession

### 11.1 Ereignisprozeduren

Die Überwachung der Termine im OUTLOOK-Standardkalender kann automatisiert werden durch Ereignisprozeduren (sog. Ereignishandler), die in das *ThisOutlookSession*-Modul von OUTLOOK eingefügt werden müssen. Ereignisse sind immer einem bestimmten Objekt zugeordnet. Das *Application*-Objekt ist das oberste Objekt und steht immer zur Verfügung (das heißt, es ist stets vorhanden).

Folgende Begrüßungsmeldung kann beispielsweise durch das Ereignis '*Application\_Startup*' automatisch angezeigt werden, wenn die OUTLOOK-Anwendung gestartet wird.

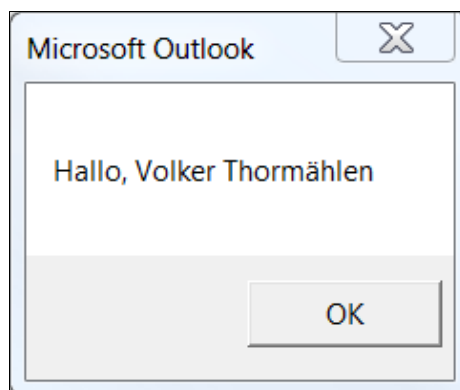


Abbildung 22: Begrüßungsmeldung durch MS OUTLOOK

Dementsprechend kann das Ereignis '*Application\_Quit*' folgende Meldung automatisch ausgeben, wenn die Anwendung beendet wird.

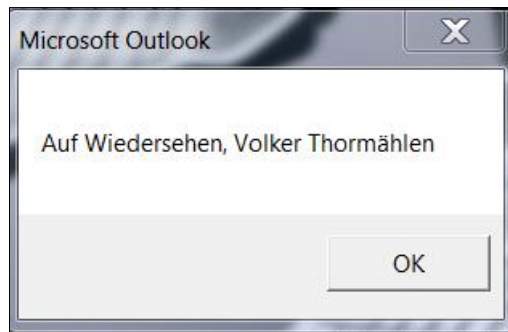


Abbildung 23: Abschiedsmeldung durch MS OUTLOOK

## 11.2 Ereignisprozeduren zur Erinnerungen an Termine

Das eingebaute Ereignis *Reminders.ReminderFire* wird von OUTLOOK immer dann ausgelöst, wenn eine Erinnerung an einen Termin im Standardkalender entsprechend definiert wurde. Dieses Ereignis tritt ein, bevor die Erinnerung ausgeführt und angezeigt wird. Es kann genutzt werden, um den jeweiligen Benutzer per E-Mail an sich selbst über den Eintritt des Ereignisses zu alarmieren (s. Abbildung 24).

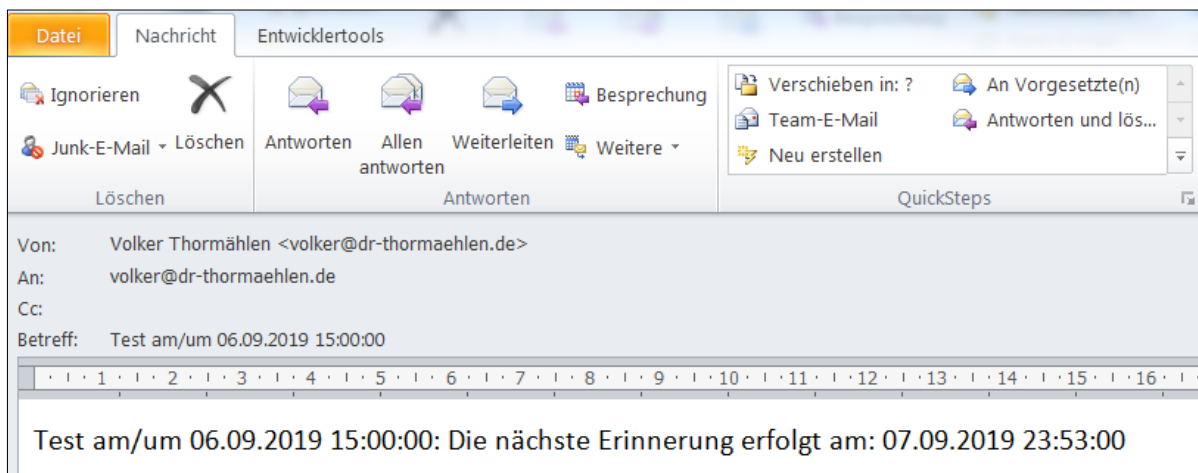


Abbildung 24: Eine E-Mail an sich selbst senden, sobald die Erinnerung ausgelöst wird

Listing 7 beinhaltet den vollständigen Quellcode der Prozeduren für das auslösende Ereignis '*Reminders.ReminderFire*'.



## 12 Erinnerungen an Termine als Notiz ausgeben

Eine Liste aller Termine mit gesetzten Erinnerungen (s. Abbildung 25) lässt sich mit der Prozedur 'GetAllExistingReminders' (s. Listing 8) anzeigen.



Abbildung 25: Liste der vorhandenen Termin-Erinnerungen als OUTLOOK-Notiz ausgeben

## 13 Schlafende Erinnerungen als Notiz ausgeben

Eine OUTLOOK-Notiz aller Termine mit latenten (d. h. 'schlafenden') Erinnerungen (s. Abbildung 26) lässt sich mit der Prozedur 'GetListofSnoozedReminders' (s. Listing 9) erzeugen.

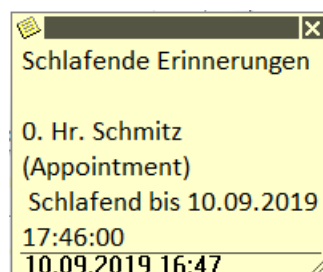


Abbildung 26: Latente Erinnerung als OUTLOOK-Notiz

## 14 VBA-Quellcode

### 14.1 VBA-Funktionen

```
Function AppointmentExists(ByVal objOutlook As Object, ByVal dtmStart As Date, ByVal  
strSubject As String) As Boolean  
    ' Aufgabe: Doppelte Kalendereinträge vermeiden.  
    Dim objFolder As Object ' MAPI-Verzeichnis  
    Dim objAppt As Object ' Kalendereintrag  
    AppointmentExists = True  
    Set objFolder = objOutlook.GetNamespace("MAPI").GetDefaultFolder(9) ' 9 => olFolderCalendar  
    For Each objAppt In objFolder.Items  
        If objAppt.Class = olAppointment Then  
            If objAppt.Start = dtmStart And objAppt.Subject = strSubject Then Exit Function  
        End If  
    Next objAppt  
    AppointmentExists = False  
End Function  
  
Function HasConflicts(ByVal objOutlook As Object, olAppt As AppointmentItem) As Boolean  
    ' Aufgabe: Konfligierende Kalendereinträge verhindern.  
    Dim objFolder As Object ' MAPI-Verzeichnis  
    Dim objAppt As Object ' Kalendereintrag  
    Set objFolder = objOutlook.GetNamespace("MAPI").GetDefaultFolder(9) ' 9 => olFolderCalendar  
    For Each objAppt In objFolder.Items  
        If ((objAppt.BusyStatus <> olFree) And (objAppt <> olAppt)) Then  
            ' Wenn das aktuelle Ereignis beginnt, bevor das übergebene endet,  
            ' muss es beendet sein, bevor das übergebene beginnt.  
            If (objAppt.Start < olAppt.End) And (objAppt.End > olAppt.Start) Then  
                HasConflicts = True ' Terminkonflikt gefunden  
                Exit Function  
            End If  
        End If  
    Next objAppt  
    HasConflicts = False  
End Function  
  
Function IsWeekendAppointment(ByVal dtmStart As Date) As Boolean  
    ' Aufgabe: Termin am Wochenende (Sa., So.) erkennen.  
    Select Case Weekday(dtmStart, vbSunday)  
        Case vbSaturday, vbSunday  
            IsWeekendAppointment = True  
        Case Else  
            IsWeekendAppointment = False  
    End Select  
End Function
```

Listing 1: VBA-Funktionen

## 14.2 Prozedur 'Create\_Single\_Appointments'

```

Sub Create_Single_Appointments()
' Aufgabe: Einzelne Kalendereinträge erstellen
Dim objWS As EXCEL.Worksheet ' Tabellenblatt
Dim rng As EXCEL.Range ' Bereich im Tabellenblatt
Dim objOL As OUTLOOK.Application ' OUTLOOK-Anwendung
Dim olItems As OUTLOOK.Items ' Kalendereinträge
Dim olAppt As OUTLOOK.AppointmentItem ' Termin als Kalendereintrag
Dim bolCreated As Boolean ' Statusschalter f. OUTLOOK
Dim dtmStart As Date ' Startdatum
Dim intCount As Integer ' Terminzähler
Dim lngRow As Long ' Zeile im Tabellenblattbereich
Dim strBetreff As String ' Betreff

' OUTLOOK starten
On Error Resume Next
Set objOL = GetObject(, "OUTLOOK.Application")
If Err.Number <> 0 Then
    Err.Clear
    Set objOL = CreateObject("OUTLOOK.Application")
    bolCreated = True
Else
    bolCreated = False
End If
' Fehlerbehandlungsroutine setzen
On Error GoTo Error_Point
' Bildschirmaktualisierung ausschalten
Application.ScreenUpdating = False
' EXCEL-Tabellenblatt bestimmen
Set objWS = ThisWorkbook.Sheets("SingleAppointments")
' Dieses Tabellenblatt aktivieren
objWS.Activate
' Die Spalten A bis H dienen zur Erstellung von Kalendereinträgen
Set rng = Range("A:H")
' Terminzähler mit Anfangswert belegen
intCount = 0
For lngRow = 2 To rng.Cells(rng.Rows.Count, 8).End(xlUp).Row
    ' Termin erstellen
    Set olAppt = objOL.CreateItem(olAppointmentItem)
    ' Eigenschaften des Termins setzen
    With olAppt
        strBetreff = rng.Cells(lngRow, 1)
        .Subject = strBetreff ' Betreff: Name
        .Location = rng.Cells(lngRow, 2) ' Besprechungsort
        dtmStart = rng.Cells(lngRow, 3)
        .Start = dtmStart ' Beginnt am/um
        .Duration = rng.Cells(lngRow, 4) ' Dauer in Minuten
        If Trim(rng.Cells(lngRow, 5).Value) = "" Then ' Gebucht-Status
            .BusyStatus = olBusy ' Status: besetzt
        Else
            .BusyStatus = rng.Cells(lngRow, 5) ' Status: besetzt
        End If
        If rng.Cells(lngRow, 6).Value > 0 Then
            .ReminderMinutesBeforeStart = rng.Cells(lngRow, 6) ' Vorlaufzeit in Minuten
            .ReminderSet = True ' Erinnerung: ja
            .ReminderPlaySound = True ' akustisches Signale
        Else
            .ReminderSet = False ' Erinnerung: nein
            .ReminderPlaySound = False ' kein akustisches Signale
        End If
        .Body = rng.Cells(lngRow, 7) & " " & rng.Cells(lngRow, 1) ' Beschreibung + Name
        .AllDayEvent = False ' Tagertermin: nein
        .Categories = rng.Cells(lngRow, 8) ' Kategorie
        .Sensitivity = olPrivate ' Vertraulichkeit: privat
    ' Duplikate und Konflikte in den Kalendereinträgen finden
    If Not AppointmentExists(objOL, dtmStart, strBetreff) Then
        If Not HasConflicts(objOL, olAppt) Then
            intCount = intCount + 1 ' Terminzähler erhöhen
            .Close olSave ' Kalendereintrag speichern
        Else
            MsgBox "Termin '" & .Subject & "' ist unverträglich. Nicht gespeichert!"
        End If
    Else
        MsgBox "Termin '" & .Subject & "' ist schon vorhanden. Nicht gespeichert!"
    End If
    End With
Next lngRow

```

```

    MsgBox Str(intCount) & " Termine im Kalender gespeichert!", vbInformation, "Kalendereinträge"
Exit_Point:
On Error Resume Next
' Bildschirmaktualisierung einschalten
Application.ScreenUpdating = True
If bolCreated Then
    objOL.Quit
End If
' Objektvariablen freigeben
Set objOL = Nothing
Set olAppt = Nothing
Set olItems = Nothing
' Prozedur sofort verlassen
Exit Sub
Error_Point:
MsgBox "Laufzeitfehler (" & Err.Number & ") " & Err.Description, vbCritical, "Termine
importieren"
Resume Exit_Point
End Sub

```

**Listing 2: VBA-Prozedur zum Erstellen von einzelnen Kalendereinträgen in OUTLOOK**

## 14.3 Prozedur 'Create\_Recurrent\_Appointments'

```

Sub Create_Recurrent_Appointments()
' Aufgabe: Wiederkehrende Kalendereinträge erstellen
Dim objWS As EXCEL.Worksheet ' Tabellenblatt
Dim rng As EXCEL.Range ' Bereich im Tabellenblatt
Dim objOL As OUTLOOK.Application ' OUTLOOK-Anwendung
Dim olItems As OUTLOOK.Items ' Kalendereinträge
Dim olAppt As OUTLOOK.AppointmentItem ' Termin als Kalendereintrag
Dim olPattern As OUTLOOK.RecurrencePattern ' Serienmuster

Dim bolCreated As Boolean ' Statusschalter f. OUTLOOK
Dim dtmStart As Date ' Startdatum
Dim intCount As Integer ' Terminzähler
Dim intRhythmus As Integer ' Rhythmus als ganzzahliger Wert
Dim intWeekday As Integer ' Wochentag als ganzzahliger Wert
Dim lngRow As Long ' Zeile im Tabellenblattbereich
Dim strBetreff As String ' Betreff
Dim varRhythmus As String ' Rhythmus
Dim strWeekday As String ' Wochentag (b. wöchentlichem Rhythmus)
Dim varRhythmus As Variant ' Rhythmus (täglich bzw. wöchentlich)
Dim varWeekday As Variant ' Wochentag

' OUTLOOK starten
On Error Resume Next
Set objOL = GetObject(, "OUTLOOK.Application")
If Err.Number <> 0 Then
    Err.Clear
    Set objOL = CreateObject("OUTLOOK.Application")
    bolCreated = True
Else
    bolCreated = False
End If
' Fehlerbehandlungsroutine setzen
On Error GoTo Error_Point
' Bildschirmaktualisierung ausschalten
Application.ScreenUpdating = False
' EXCEL-Tabellenblatt bestimmen
Set objWS = ThisWorkbook.Sheets("Recurrent_Appointments_Data")
' Dieses Tabellenblatt aktivieren
objWS.Activate
' Die Spalten A bis H dienen zur Erstellung von Kalendereinträgen
Set rng = Range("A:N")
' Terminzähler mit Anfangswert belegen
intCount = 0
For lngRow = 3 To rng.Cells(rng.Rows.Count, 14).End(xlUp).Row
    varRhythmus = rng.Cells(lngRow, 12).Value
    If Not IsNumeric(varRhythmus) Then
        Select Case varRhythmus
            Case "täglich"
                intRhythmus = 0
            Case "wöchentlich"
                intRhythmus = 1
        End Select
    Else
        MsgBox "Rhythmus " & varRhythmus & " ist unbekannt", vbCritical, "Serientermin"
    End If
    varWeekday = rng.Cells(lngRow, 13).Value
    If Len(Trim(varWeekday)) > 0 Then
        Select Case varWeekday
            Case "Sonntag"
                intWeekday = 1
            Case "Montag"
                intWeekday = 2
            Case "Dienstag"
                intWeekday = 4
            Case "Mittwoch"
                intWeekday = 8
            Case "Donnerstag"
                intWeekday = 16
            Case "Freitag"
                intWeekday = 32
            Case "Samstag"
                intWeekday = 64
        End Select
    End If
End For
End Sub

```

```

' Serientermin erstellen
Set olAppt = objOL.CreateItem(olAppointmentItem)
' Serienmuster erstellen
Set olPattern = olAppt.GetRecurrencePattern
' Eigenschaften des Serientermins setzen
With olPattern
    .RecurrenceType = intRhythmus
    If intRhythmus = 1 Then
        .DayOfWeekMask = intWeekday
    End If
    .Occurrences = rng.Cells(lngRow, 14)
    .PatternStartDate = rng.Cells(lngRow, 9)
    .StartTime = rng.Cells(lngRow, 10)
    .Duration = rng.Cells(lngRow, 11)
End With
With olAppt
    strBetreff = rng.Cells(lngRow, 1)
    .Subject = strBetreff
    .Location = rng.Cells(lngRow, 2)
    dtmStart = rng.Cells(lngRow, 3)
    .Start = dtmStart
    .BusyStatus = olBusy
    If rng.Cells(lngRow, 6).Value > 0 Then
        .ReminderMinutesBeforeStart = rng.Cells(lngRow, 6)
        .ReminderSet = True
        .ReminderPlaySound = True
    Else
        .ReminderSet = False
        .ReminderPlaySound = True
    End If
    .Body = rng.Cells(lngRow, 7) & " " & rng.Cells(lngRow, 1)
    .AllDayEvent = False
    .Categories = rng.Cells(lngRow, 8)
    .Sensitivity = olPrivate
    ' Duplikate in den Kalendereinträgen finden
    If Not AppointmentExists(objOL, dtmStart, strBetreff) Then
        intCount = intCount + 1
        .Close olSave
    Else
        MsgBox "Termin '" & .Subject & "' ist bereits vorhanden. Nicht gespeichert!"
    End If
End With
Next lngRow
' Erfolgsmeldung
MsgBox Str(intCount) & " wiederkehrende Termine im Kalender gespeichert!", vbInformation,
"Kalendereinträge"
Exit_Point:
On Error Resume Next
' Bildschirmaktualisierung einschalten
Application.ScreenUpdating = True
If bolCreated Then
    objOL.Quit
End If
' Objektvariablen freigeben
Set objOL = Nothing
Set olAppt = Nothing
Set olItems = Nothing
' Prozedur sofort verlassen
Exit Sub
Error_Point:
MsgBox "Laufzeitfehler (" & Err.Number & ") " & Err.Description, vbCritical, "Termine
importieren"
Resume
Resume Exit_Point
End Sub

```

**Listing 3: VBA-Prozedur zum Erstellen von wiederkehrenden Kalendereinträgen in OUTLOOK**

## 14.4 Prozedur 'Find\_Appointments'

```

Sub Find_Appointments()
' Aufgabe: OUTLOOK Termine in einem bestimmten 14-Tage Zeitraum nach EXCEL exportieren
Const intDays As Integer = 14 ' Zeitraum in Tagen (z. B. 14 Tage)
' EXCEL Objekte
Dim objWS As EXCEL.Worksheet ' EXCEL-Tabellenblatt
' OUTLOOK Objekte
Dim objOL As OUTLOOK.Application ' OUTLOOK-Anwendung
Dim olAppt As OUTLOOK.AppointmentItem ' Termin
Dim olFolder As OUTLOOK.Folder ' Verzeichnis
Dim olItems As OUTLOOK.Items ' Kalendereinträge
Dim olItemsFilter As OUTLOOK.Items ' gefilterte Kalendereinträge
Dim olNS As OUTLOOK.Namespace ' Wurzelobjekt f. Verzeichnisse
' Sonstige Deklarationen
Dim bolCreated As Boolean ' OUTLOOK-Status
Dim lngCount As Long ' Terminzähler
Dim dtmStart As Date ' Startdatum
Dim dtmEnd As Date ' Enddatum
Dim strFilter As String ' Filter für Starttermine
Dim varDate As Variant ' Datum
' Startdatum für einen 14-Tage Zeitraum eingeben
varDate = Format(Now(), "dd.mm.yyyy")
varDate = InputBox("Startdatum für einen " & intDays & "-Tage Zeitraum ?" & Chr$(13) & _
"Datum muss im Format tt.mm.jjjj eingeben werden!", intDays & "-Tage Terminliste", varDate)
If IsDate(varDate) Then
    dtmStart = Format(varDate, "dd.mm.yyyy")
    dtmEnd = DateAdd("d", intDays, dtmStart)
Else
    MsgBox "Abbruch: Ungültiges Datum!", vbExclamation, "OUTLOOK-Termine"
    GoTo Exit_Point
End If
' OUTLOOK starten
On Error Resume Next
Set objOL = GetObject(, "OUTLOOK.Application")
If Err.Number <> 0 Then
    Err.Clear
    Set objOL = CreateObject("OUTLOOK.Application")
    bolCreated = True
Else
    bolCreated = False
End If
' Fehlerbehandlungsroutine setzen
On Error GoTo Error_Point
' Bildschirmaktualisierung ausschalten
Application.ScreenUpdating = False
' EXCEL-Tabellenblatt bestimmen
Set objWS = ThisWorkbook.Sheets("Terminliste")
' Dieses Tabellenblatt aktivieren
objWS.Activate
' Löscht alle Zellen im aktiven Tabellenblatt
With Cells
    ' Alle Zellen im aktiven Tabellenblatt löschen
    .ClearContents
    ' ... und Farbe des Innenbereichs löschen
    .Interior.ColorIndex = xlNone
End With
' Spaltenköpfe bestimmen
Cells(1, 1).Value = "Beginn"
Cells(1, 2).Value = "Betreff"
' 1. Tabellenzeile formatieren
With objWS.Range("A1:B1")
    .Font.Bold = True
    .Font.ColorIndex = 3 ' Schriftfarbe: rot
    .Font.Size = 11 ' Schriftgrad: 11
    .Interior.ColorIndex = 33 ' Farbe des Innenbereichs: blau
End With
' Mit der Items.Find-Methode einen Filter für den nächsten
' 14-Tage-Zeitraum erstellen.
strFilter = "[Start] >= '" & Format(dtmStart, "mm/dd/yyyy hh:mm AMPM") & "' " & _
"& '" And [End] <= '" & Format(dtmEnd, "mm/dd/yyyy hh:mm AMPM") & "' '"
Set olNS = objOL.GetNamespace("MAPI")
Set olFolder = olNS.GetDefaultFolder(olFolderCalendar)
Set olItems = olFolder.Items

```

```

' Wiederkehrende Termine einbeziehen
olItems.IncludeRecurrences = True
' Kalendereinträge terminlich eingrenzen
Set olItemsFilter = olItems.Restrict(strFilter)
' gefilterte Termine nach Startdatum sortieren
olItemsFilter.Sort "[Start]"
' Kalendereinträge ausgeben
lngCount = 0 ' Terminzähler
For Each olAppt In olItemsFilter
    lngCount = lngCount + 1
    With olAppt
        Cells(lngCount + 1, 1) = Format(.Start, "dd.mm.yyyy hh:nn") ' Starttermin
        Cells(lngCount + 1, 2) = .Subject ' Betreff
    End With
Next olAppt
MsgBox Str(lngCount) & " Termine im Zeitraum von " & dtmStart & " bis " & dtmEnd & _
    " gefunden" & vbCrLf & " und in das Tabellenblatt '" & _
    objWS.Name & "' der EXCEL-Arbeitsmappe '" & ThisWorkbook.Name & "' exportiert!",
    vbInformation, str(intDays) & "-Tage Terminliste sortiert nach Startdatum"
Exit_Point:
On Error Resume Next
' Bildschirmaktualisierung einschalten
Application.ScreenUpdating = True
If bolCreated Then
    objOL.Quit
End If
' Objektvariablen freigeben
Set objWS = Nothing
Set objOL = Nothing
Set olNS = Nothing
Set olAppt = Nothing
Set olFolder = Nothing
Set olItems = Nothing
Set olItemsFilter = Nothing
' Prozedur sofort verlassen
Exit Sub
Error_Point:
MsgBox "Laufzeitfehler (" & Err.Number & ") " & Err.Description, vbCritical, "Termine
listen"
Resume Exit_Point
End Sub

```

**Listing 4: OUTLOOK Termine in einem bestimmten 14-Tage Zeitraum nach EXCEL exportieren**



## 14.5 Prozedur 'Export\_Appointments'

```

Sub Export_Appointments()
' Aufgabe: OUTLOOK-Termine nach Excel exportieren
Const conMsg As String = "OUTLOOK-Termine nach EXCEL exportieren"
' Excel Objekte
Dim objWS As Excel.Worksheet ' Tabellenblatt
' OUTLOOK Objekte
Dim olAppt As Outlook.AppointmentItem ' Termin
Dim objOL As Outlook.Application ' Anwendung
Dim olNS As Outlook.Namespace ' Wurzelobjekt für Verzeichnisse
Dim olFolder As Outlook.MAPIFolder ' Verzeichnis
Dim olItems As Outlook.Items ' Kalendereinträge
Dim olPattern As Outlook.RecurrencePattern ' Serienmuster
' Sonstige Variablen
Dim bolCreated As Boolean ' OUTLOOK-Status
Dim dtmStart As Date ' Startdatum
Dim dtmEnd As Date ' Enddatum
Dim dtmToday As Date ' Tagesdatum
Dim intRow As Integer ' Zeilenindex
Dim lngCount As Long ' Terminzähler
Dim objDateRng As Object ' Datumsbereich
Dim strDoW As String ' Wochentag
Dim strPattern As String ' Serienmuster as String
Dim strFilter As String ' Filter
Dim varDate As Variant ' Datum als Variant-Wert
' OUTLOOK starten
On Error Resume Next
Set objOL = GetObject(, "Outlook.Application")
If Err.Number <> 0 Then
    Err.Clear
    Set objOL = CreateObject("Outlook.Application")
    bolCreated = True
Else
    bolCreated = False
End If
' Fehlerbehandlungsroutine setzen
On Error GoTo Error Point
' EXCEL-Tabellenblatt bestimmen
Set objWS = ThisWorkbook.Sheets("Appointments_Export")
' Dieses Tabellenblatt aktivieren
objWS.Activate
With Cells
    ' Alle Zellen im aktiven Tabellenblatt löschen
    .ClearContents
    ' ... und Farben löschen
    .Interior.ColorIndex = xlNone
End With
' Spaltenköpfe belegen
Cells(1, 1).Value = "Beginn"
Cells(1, 2).Value = "Dauer (Min.)"
Cells(1, 3).Value = "Ende"
Cells(1, 4).Value = "Ort"
Cells(1, 5).Value = "Betreff"
Cells(1, 6).Value = "Textkörper"
Cells(1, 7).Value = "Serienmuster"
Cells(1, 8).Value = "Wochentag"
Cells(1, 9).Value = "Häufigkeit"
' 1. Tabellenzeile formatieren
With objWS.Range("A1:I1")
    .Font.Bold = True
    .Font.ColorIndex = 3 ' Schriftfarbe: rot
    .Font.Size = 11 ' Schriftgrad: 11
    .Interior.ColorIndex = 33 ' Farbe des Innenbereichs: blau
End With
' Bildschirmaktualisierung ausschalten
Application.ScreenUpdating = False
' Tagesdatum vorschlagen
Select Case Weekday(Now + 1, vbMonday)
    Case Is > 5
        dtmToday = Now + 3
    Case Else
        dtmToday = Now + 1
End Select
' Datum abfragen
varDate = Format(dtmToday, "dd.mm.yyyy")

```

```

varDate = InputBox("Welches Kalenderdatum soll abgefragt werden ?" & Chr$(13) & "Datum muss im
Format tt.mm.jjjj eingegeben werden!", conMsg, varDate)
If IsDate(varDate) Then
    dtmStart = Format(varDate, "dd.mm.yyyy")
    dtmEnd = dtmStart + 1
Else
    MsgBox "Abbruch: Ungültiges Datum!", vbExclamation, conMsg
    GoTo Exit_Point
End If
Set olNS = objOL.GetNamespace("MAPI")
Set olFolder = olNS.GetDefaultFolder(olFolderCalendar)
Set olItems = olFolder.Items
lngCount = olItems.Count
If lngCount = 0 Then
    MsgBox "Keine Kalendereinträge gefunden", vbCritical, conMsg
    GoTo Exit_Point
Else
    MsgBox lngCount & " Kalendereinträge gefunden!", vbInformation, conMsg
End If
' Filter definieren
strFilter = "[Start] >= '" & dtmStart & "' And [End] <= '" & dtmEnd & "'"
' Einträge ab Zeile 2
intRow = 2
Set objDateRng = olFolder.Items.Restrict(strFilter)
For Each olAppt In objDateRng
    With olAppt
        'Termindaten eintragen
        Cells(intRow, 1) = Format(.Start, "dd.mm.yyyy hh:mm") ' Termin
        Cells(intRow, 2) = Int(.Duration) ' Dauer (Minuten)
        Set olPattern = .GetRecurrencePattern ' Serienattribute
        Select Case olPattern.RecurrenceType
            Case 0
                strPattern = "täglich"
            Case 1
                strPattern = "wöchentlich"
            Case 2
                strPattern = "monatlich"
            Case 3
                strPattern = "jeden n-ten Monat"
            Case 5
                strPattern = "jährlich"
            Case 6
                strPattern = "jedes n-te Jahr"
            Case Else
                strPattern = ""
        End Select
        If Format(olPattern.PatternEndDate, "dd.mm.yyyy") <> Format(DateValue("31.12.4500"),
            "dd.mm.yyyy") Then
            ' Enddatum
            Cells(intRow, 3) = Format(DateValue(olPattern.PatternEndDate), "dd.mm.yyyy")
            ' Serienmuster
            Cells(intRow, 7) = strPattern
        Else
            Cells(intRow, 3) = Format(.Start + (((1 / 24) / 60) * .Duration), "hh:nn")
        End If
        Cells(intRow, 4) = .Location ' Ort
        Cells(intRow, 5) = .Subject ' Betreff
        Cells(intRow, 6) = .Body ' Textkörper
        Select Case olPattern.DayOfWeekMask
            Case Is = olSunday
                strDoW = "Sonntag"
            Case Is = olMonday
                strDoW = "Montag"
            Case Is = olTuesday
                strDoW = "Dienstag"
            Case Is = olWednesday
                strDoW = "Mittwoch"
            Case Is = olThursday
                strDoW = "Donnerstag"
            Case Is = olFriday
                strDoW = "Freitag"
            Case Is = olSaturday
                strDoW = "Samstag"
            Case Else
                strDoW = ""
        End Select
    End With

```

```

        If Len(Trim(strDoW)) > 0 Then
            Cells(intRow, 8) = strDoW ' Wochentag
        End If
        Cells(intRow, 9) = olPattern.Occurrences ' Häufigkeit
        intRow = intRow + 1
    End With
Next olAppt
MsgBox "Alle Ereignisse des Starttermins " & dtmStart & " nach EXCEL exportiert!",
vbInformation, conMsg
Exit_Point:
    On Error Resume Next
    Application.ScreenUpdating = True
    If bolCreated Then
        objOL.Quit
    End If
    ' Objektvariablen freigeben
    Set objOL = Nothing
    Set olNS = Nothing
    Set olAppt = Nothing
    Set olFolder = Nothing
    ' Prozedur sofort verlassen
Exit Sub
Error_Point:
    MsgBox "Laufzeitfehler (" & Err.Number & ") " & Err.Description, vbCritical, conMsg
    Resume Exit_Point
End Sub

```

**Listing 5: OUTLOOK Termine eines bestimmten Kalendertages nach EXCEL exportieren**

## 14.6 Prozedur 'Update\_Appointments'

```

Sub Update_Appointments()
' Aufgabe: Einzeltermine am Abend oder am Wochenende als 'privat' kennzeichnen.
' Konstanten
Const conMsg As String = "Private Termine bestimmen!"
Const conLimit As Date = #5:30:00 PM# ' Zeitgrenze für private Einzeltermine"
' OUTLOOK Objekte
Dim objOL As Outlook.Application ' OUTLOOK-Anwendung
Dim olFolder As Outlook.Folder ' Verzeichnis
Dim olItems As Outlook.Items ' Kalendereinträge
Dim objItem As Object ' Kalendereintrag
' Sonstige Deklarationen
Dim bolCreated As Boolean ' OUTLOOK-Status
Dim bolFlag As Boolean ' boolescher Schalter
Dim dtmStartDate As Date ' Startdatum eins Termins
Dim dtmStartTime As Date ' Startzeit eines Termins
Dim dtmLimit As Date ' Zeitgrenze f. private Einzeltermin
Dim lngCount As Long ' Terminzähler
Dim varLimit As Variant ' Zeitgrenze f. priv. Termin als Variant-Wert
' Zeitgrenze f. private Termine am Abend bestimmen.
varLimit = InputBox("Zeitgrenze für private abendliche Termine?" & Chr$(13) & _
    "Die Zeitgrenze muss wie folgt erfasst werden!", "Zeitgrenze bestimmen", conLimit)
If IsDate(varLimit) Then
    dtmLimit = FormatDateTime(varLimit, vbShortTime)
Else
    MsgBox "Abbruch: Ungültige Zeitgrenze!", vbExclamation, "Zeitgrenze für private Termine"
    GoTo Exit_Point
End If
' OUTLOOK starten
On Error Resume Next
Set objOL = GetObject(, "Outlook.Application")
If Err.Number <> 0 Then
    Err.Clear
    Set objOL = CreateObject("Outlook.Application")
    bolCreated = True
Else
    bolCreated = False
End If
' Fehlerbehandlungsroutine setzen
On Error GoTo Error_Point
' Verzeichnis bestimmen
Set olFolder = objOL.GetNamespace("MAPI").GetDefaultFolder(olFolderCalendar)
Set olItems = olFolder.Items
lngCount = olItems.Count
' Zahl der Kalendereinträge prüfen
If lngCount = 0 Then
    MsgBox "Keine Kalendereinträge gefunden", vbCritical, conMsg
    GoTo Exit_Point
End If
lngCount = 0
For Each objItem In olItems
    With objItem
        ' Termin prüfen: Schalter aus
        bolFlag = False
        If .RecurrenceState = olApptNotRecurring Then
            If .Sensitivity <> olPrivate Then
                dtmStartDate = FormatDateTime(.Start, vbShortDate)
                If IsWeekendAppointment(dtmStartDate) = True Then
                    ' Wochenendtermin: Schalter an
                    bolFlag = True
                Else ' Kein Wochenendtermin
                    dtmStartTime = FormatDateTime(.Start, vbShortTime)
                    If dtmStartTime > dtmLimit Then
                        ' Abendlicher Termin: Schalter an
                        bolFlag = True
                    End If
                End If
            End If
        End If
    End With
    lngCount = lngCount + 1
End For
End Sub

```

```

        ' Wenn Schalter an, dann ...
    If bolFlag Then
        .Sensitivity = olPrivate
        .Save
        lngCount = lngCount + 1
    End If
End With
Next objItem
MsgBox Str(lngCount) & " Termine als 'privat' gekennzeichnet.", vbExclamation, conMsg
Exit_Point:
On Error Resume Next
If bolCreated Then
    objOL.Quit
End If
' Objektvariablen freigeben.
Set objOL = Nothing
Set olFolder = Nothing
Set olItems = Nothing
Set objItem = Nothing
' Prozedur sofort verlassen.
Exit Sub
Error_Point:
MsgBox "Laufzeitfehler (" & Err.Number & ") " & Err.Description, vbCritical, conMsg
Resume Exit_Point
End Sub

```

**Listing 6: Einzeltermine am Abend oder am Wochenende als 'privat' kennzeichnen.**

## 14.7 Ereignisprozeduren bei Erinnerung an einem Termin

```
Option Explicit
Public WithEvents objReminders As Outlook.Reminders
Const EMAIL_TO = "volker@dr-thormaehlen.de"

Private Sub Application_Startup()
    ' Aufgabe: Element anzeigen, das vom Erinnerungsereignis
    ' bei jeder Ausführung einer Erinnerung ausgelöst wird.
    MsgBox "Hallo, " & Application.GetNamespace("MAPI").CurrentUser
    Set objReminders = Outlook.Application.Reminders
End Sub

Private Sub Application_Quit()
    ' Aufgabe: Schließt alle aktuell geöffneten Fenster.
    MsgBox "Auf Wiedersehen, " & Application.GetNamespace("MAPI").CurrentUser
    Set objReminders = Nothing
End Sub

Private Sub objReminders_BeforeRemindersShow(Cancel As Boolean)
    ' Anzeige des Erinnerungsfensters unterbinden.
    Cancel = True
End Sub

Private Sub objReminders_ReminderFire(ByVal objReminder As Reminder)
    ' Aufgabe: Ein Outlook-Element anzeigen,
    ' sobald eine entsprechende Erinnerung ausgeführt wird.
    Dim strSubject As String ' Betreff
    Dim strBody As String ' Textkörper
    On Error GoTo Error_Point
    With objReminder
        ' Gehört die Erinnerung zu einem Termin?
        If .Item.Class <> olAppointment Then GoTo Exit_Point
        ' Den zugehörigen Termin anzeigen.
        .Item.Display
        strSubject = .Caption & " am/um " & FormatDateTime(.Item.Start, vbGeneralDate)
        strBody = strSubject & ": Die nächste Erinnerung erfolgt am: " & _
            CStr(.NextReminderDate)
    End With
    ' Erinnerung an einen Termin per E-Mail versenden.
    Call E_Mail_Reminder(EMAIL_TO, strSubject, strBody)
    If objReminder.IsVisible Then
        ' Zeitspanne, um die eine Erinnerung verzögert werden soll.
        ' Standardwert ist 5 Minuten.
        objReminder.Snooze 60 ' Minuten
    End If
Exit_Point:
    Exit Sub
Error_Point:
    MsgBox "Laufzeitfehler (" & Err.Number & ") " & Err.Description, vbCritical, _
        "Termine überwachen"
    Resume Exit_Point
End Sub
```

```

Private Sub E Mail Reminder(EMAIL_TO, strSubject As String, strBody As String)
' Aufgabe: Erinnerung per E-Mail versenden.
Dim olMail As Outlook.MailItem
On Error GoTo Error_Point
Set olMail = Application.CreateItem(olMailItem)
With olMail
    .To = EMAIL_TO
    .Subject = strSubject
    .Body = strBody
    .Recipients.ResolveAll
    .Send
End With
Exit_Point:
Set olMail = Nothing
Exit Sub
Error_Point:
MsgBox "Laufzeitfehler (" & Err.Number & ") " & Err.Description, vbCritical, _
    "E-Mail senden"
Resume Exit_Point
End Sub

```

**Listing 7: Ereignisprozeduren bei Erinnerung an einem Termin**

## 14.8 Prozedur 'GetAllExistingReminders'

```
Sub GetAllExistingReminders()  
    ' Aufgabe: Alle Erinnerungen als OUTLOOK-Notiz anzeigen.  
    ... ' Quelle: In Anlehnung an [4]  
    Dim olReminders As Outlook.Reminders ' Erinnerungen  
    Dim olReminder As Object ' Erinnerung  
    Dim olNote As Outlook.NoteItem ' Notiz  
    Dim strReminderDetails As String ' Einzelheiten einer Erinnerung  
    Dim lngCount As Long ' Zähler f. gefundenen Erinnerungen  
    On Error GoTo Error_Point  
    Set olReminders = Outlook.Application.Reminders  
    lngCount = olReminders.Count  
    If lngCount = 0 Then  
        MsgBox "Keiner Erinnerungen gefunden.", vbInformation, "Erinnerungen"  
    Else  
        MsgBox CStr(lngCount) & " Erinnerungen gefunden.", vbInformation, "Erinnerungen"  
        For Each olReminder In olReminders  
            strReminderDetails = strReminderDetails & olReminder.Caption & _  
                " (" & TypeName(olReminder.Item) & ") ---> " &  
FormatDateTime(olReminder.NextReminderDate, vbGeneralDate) & vbCrLf  
        Next olReminder  
        ' Notiz erzeugen  
        Set olNote = Outlook.Application.CreateItem(olNoteItem)  
        With olNote  
            .Top = 300  
            .Left = 400  
            .Width = 500  
            .Height = 600  
            .Body = "Vorhandene Erinnerungen an Termine:" & vbCrLf & vbCrLf & strReminderDetails  
            .Categories = "Erinnerungen"  
            .Display  
        End With  
        ' .PrintOut  
    End With  
    End If  
Exit_Point:  
    Set olReminders = Nothing  
    Set olReminder = Nothing  
    Set olNote = Nothing  
    Exit Sub  
Error_Point:  
    MsgBox "Laufzeitfehler (" & Err.Number & ") " & Err.Description, vbCritical, _  
        "Notiz ausgeben"  
    Resume Exit_Point  
End Sub
```

Listing 8: Alle Erinnerungen an Termine als OUTLOOK-Notiz anzeigen



## 14.9 Prozedur 'GetListofSnoozedReminders'

```
Sub GetListofSnoozedReminders()  
    ' Aufgabe: Liste der schlafenden Erinnerungen erstellen.  
    ' Quelle: In Anlehnung an [5]  
    Dim objReminders As Outlook.Reminders  
    Dim objReminder As Object  
    Dim objNote As Outlook.NoteItem  
    Dim strList As String  
    Dim lngCount As Long  
    ' Alle Erinnerungen ermitteln  
    Set objReminders = Outlook.Reminders  
    For Each objReminder In objReminders  
        ' Latente Erinnerungen bestimmen  
        If (objReminder.OriginalReminderDate <> objReminder.NextReminderDate) = True Then  
            strList = strList & lngCount & ". " & objReminder.Caption &  
                " (" & Replace(TypeName(objReminder.Item), "Item", "") & ")" & vbCrLf &  
                " Schlafend bis " & _  
                objReminder.NextReminderDate & vbCrLf & vbCrLf  
            lngCount = lngCount + 1  
        End If  
    Next objReminder  
    ' Liste der schlafenden Erinnerungen als Notiz anzeigen  
    If lngCount > 0 Then  
        Set objNote = Outlook.Application.CreateItem(olNoteItem)  
        With objNote  
            .Body = "Schlafende Erinnerungen" & vbCrLf & vbCrLf & strList  
            .Display  
        End With  
    Else  
        MsgBox "Keine schlafenden Erinnerungen gefunden.", vbExclamation  
    End If  
End Sub
```

Listing 9: Liste der latenten Erinnerungen erstellen

## 15 Literatur

- [1] Microsoft, „OlBusyStatus-Aufzählung (Outlook),“ 08 06 2017. [Online]. Available: <https://docs.microsoft.com/de-de/office/vba/api/outlook.olbusystatus>. [Zugriff am 05 08 2019].
- [2] Microsoft, „OlRecurrenceType-Aufzählung (Outlook),“ 08 06 2017. [Online]. Available: <https://docs.microsoft.com/de-de/office/vba/api/outlook.olrecurrencetype>. [Zugriff am 05 08 2019].
- [3] Microsoft, „OlDaysOfWeek-Aufzählung (Outlook),“ 08 06 2017. [Online]. Available: <https://docs.microsoft.com/de-de/office/vba/api/outlook.oldaysofweek>. [Zugriff am 05 08 2019].
- [4] S. Zhang, „How to Quickly Get a List of All the Existing Reminders in Your Outlook,“ 02 03 2017. [Online]. Available: <https://www.datanumen.com/blogs/quickly-get-list-existing-reminders-outlook/>. [Zugriff am 05 08 2019].
- [5] S. Zhang, „How to Quickly Get a List of the Snoozed Reminders in Your Outlook,“ 13 09 2017. [Online]. Available: <https://www.datanumen.com/blogs/quickly-get-list-snoozed-reminders-outlook/>. [Zugriff am 05 08 2019].
- [6] Microsoft, „Durchsuchen des Kalenders nach Terminen in einem Datumsbereich, die ein bestimmtes Wort im Betreff enthalten,“ 2017 6 8. [Online]. Available: <https://docs.microsoft.com/de-de/office/vba/outlook/how-to/search-and-filter/search-the-calendar-for-appointments-within-a-date-range-that-contain-a-specific>. [Zugriff am 05 08 2019].
- [7] S. Zhang, „How to Auto Mark Weekend Appointments as Private in Outlook,“ 09 04 2018. [Online]. Available: <https://www.datanumen.com/blogs/how-to-auto-mark-weekend-appointments-as-private-in-outlook/>. [Zugriff am 05 08 2019].
- [8] S. Zhang, „How to Auto Mark All Evening Appointments as Private in Your Outlook,“ 03 05 2018. [Online]. Available: <https://www.datanumen.com/blogs/how-to-auto-mark-all-evening-appointments-as-private-in-your-outlook/>. [Zugriff am 05 08 2019].
- [9] A. Harcourt, „Automatically rejecting appointments in Microsoft Outlook 2007,“ 01 08 2009. [Online]. Available: [https://www.uglybugger.org/software/post/automatically\\_rejecting\\_appointments\\_in\\_microsoft\\_outlook\\_2007](https://www.uglybugger.org/software/post/automatically_rejecting_appointments_in_microsoft_outlook_2007). [Zugriff am 05 08 2019].
- [10] Slipstick Systems, „Send an Email When a Reminder Fires,“ 30 05 2018. [Online]. Available: <https://www.slipstick.com/developer/send-email-outlook-reminders-fires/>. [Zugriff am 05 08 2019].