Diese Excel-Anwendung wendet sich etwas abseits vom Amateurfunk an diejenigen, die sich intensiver mit der VBA-Programmierung von MS Excel beschäftigen. Sie ist vor vielen Jahren, noch zu Zeiten von Office 2000 entstanden, um Updates von VBA-Modulen in produktiven Excel-Anwendungen bei Kunden ohne viel Handarbeit durchführen zu können.

Diese Anwendung, modernisiert mit MS Excel 2019 (32 Bit) unter MS Windows 10 Pro (64 Bit, 22H2), nun etwas aufgehübscht und mit verbesserter Benutzerführung, bietet folgende Funktionen:

1. **Code Cleaner**

   Die seit den 1990er Jahren in MS Office verfügbare Visual Basic Entwicklungsumgebung (VBE) hat mit der Zeit wohl etwas Staub angesetzt. Im übelsten Falle kann es vorkommen, dass nach umfangreichen Bearbeitungen des VBA-Codes die Anwendung ihren Dienst einstellt oder auch nur ein merkwürdiges Verhalten an den Tag legt. Und das, wenn der VBA-Code absolut sauber ist.
   Manuell ist eine Bereinigung in der VBE möglich, indem alle VBA-Module einzeln gelöscht und dabei in einem Zug exportiert, um anschließend in eine neue und leere oder in die aktuelle Excel-Arbeitsmappe, nun ohne VBA-Code, einzeln zurück importiert werden. An Excel-Tabellen oder der Arbeitsmappe 'angehängter' Code muss ebenfalls exportiert, anschließend gelöscht und reimportiert werden. Das wäre reichlich mühsam.
   Der Code Cleaner erledigt das mit ein paar wenigen Mausklicks, ohne den Datenbestand in der Arbeitsmappe anzurühren. Der vorkompilierte P-Code, der wahrscheinliche Verursacher möglicher Probleme, wird bereinigt, nebenbei reduziert sich damit auch die Dateigröße.

2. **Austausch von VBA-Modulen zwischen zwei Excel-Arbeitsmappen**

   Um in produktiven Anwendungen einzelne Funktionen ohne Gefährdung des Datenbestandes mit einem Update der betreffenden Code-Module zu verbessern, können gezielt VBA-Module von einer Excel-Quelldatei in eine Excel-Zieldatei übertragen werden. Die zum Austausch vorhandenen Module lassen sich aus einer angezeigten Liste auswählen. Vollzug wird angezeigt.

3. **Import von VBA-Textdateien in eine Excel-Arbeitsmappe**

   Die VBA-Textdateien können mit dem manuellen Export aus der VBE stammen, aus den vom Code Cleaner exportierten Code-Files oder aus einer anderen Quelle. Der Dateiname muss entsprechend dem Modulnamen der Zieldatei passend gewählt werden.

Grundsätzlich sollten diese Funktionen an einer Kopie der zu bearbeitenden Excel-Arbeitsmappe ausgeführt werden.

Voraussetzungen:

Neben der Aktivierung aktiver Inhalte (Makros) ist eine sicherheitstechnisch kritische Einstellung im Trust Center vorzunehmen, wenn auch nur vorübergehend:

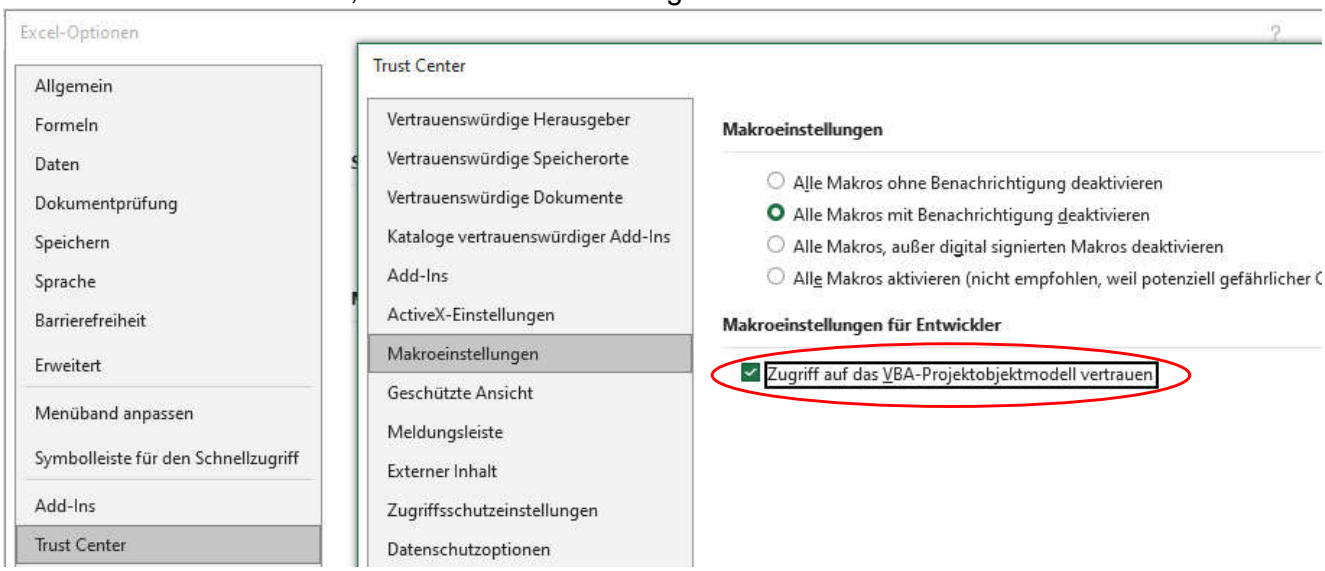


Abb. 0.1: Notwendige Einstellung im Trust Center.

Die Einstellung ist erreichbar:

- Excel-Menüband Datei – Optionen
- Dort (links ganz unten) "Trust Center"
- Button (rechts) "Einstellungen für das Trust Center"
- Dort (Leiste links) "Makroeinstellungen".

Vorsicht: Mit der rot gekennzeichneten Einstellung wird denkbaren Makro-Viren und -Trojanern Tür und Tor geöffnet. In vernetzten Systemen nur mit Bedacht vorzunehmen, wenn überhaupt, und wenn der Admin diese Sicherheitslücke noch offen gelassen haben sollte.
Aber: Diese Excel-Anwendung braucht diese Einstellung. "Normale" Excel-Anwendungen mit unterlegtem VBA-Code ohne Manipulationen des Codes brauchen diese Einstellung nicht. Ohne diese Einstellung erfolgt eine Meldung, dass der VBA-Code 'protected', also geschützt ist.

Auf eine weitere Voraussetzung bei der üblichen Programmierung der VBE, der zusätzlich zu setzende Verweis auf die "Microsoft Visual Basic for Application Extensibility 5.3", wurde mit einer etwas anderen Programmierung verzichtet. Die standardmäßig in Excel gesetzten Verweise reichen aus.

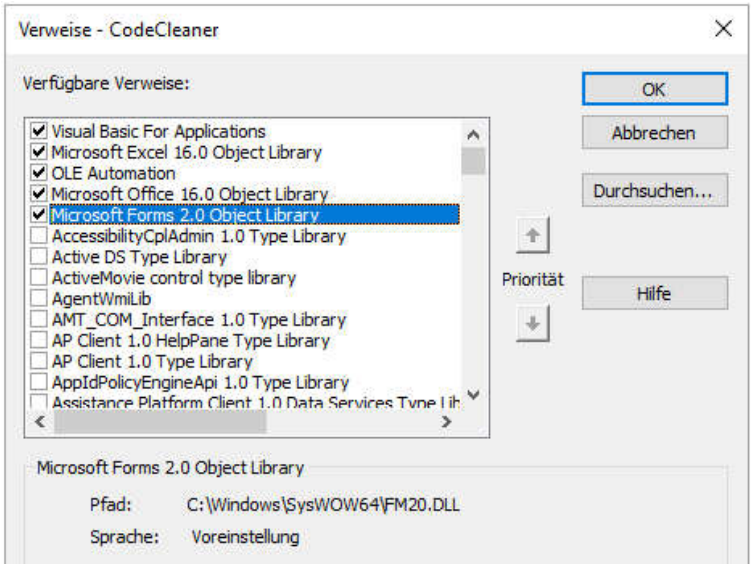


Abb. 0.2: Standardverweise für Excel 16.0 (Excel 2019).

Der VBA-Code ist prinzipiell auch für andere Programme aus der MS Office-Familie mit entsprechenden Anpassungen verwendbar. Hauptanwendung war damals aber MS Excel. Der VBA-Code ist wie immer ungeschützt einsehbar.

Dieses Programm verändert den VBA-Code in Ihren Excel-Arbeitsmappen. Es wurde mit aller gebotenen Sorgfalt getestet und vielfach anhand der Ergebnisse überprüft.
Dennoch, die Verwendung erfolgt auf eigenes Risiko.
Eine Haftung für eventuelle Schäden ist ausgeschlossen.

Begrifflichkeiten

Die nachfolgend angesprochenen Komponenten (Module) stellen sich in der VBA-Entwicklungsumgebung so dar:

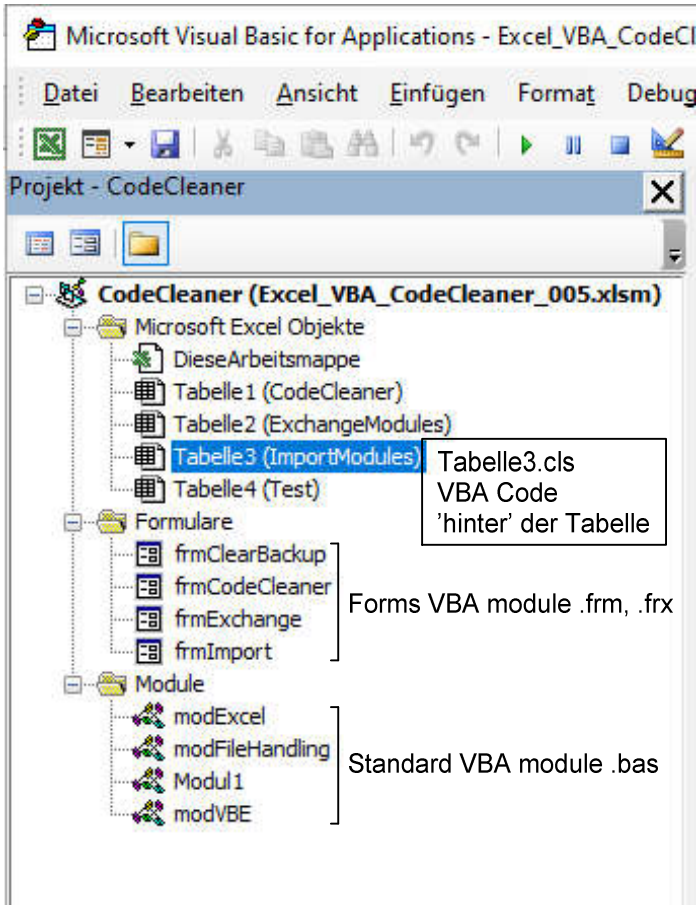


Abb. 0.3: die VBA-Komponenten im VBE Explorer.

1. **DieseArbeitsmappe und Tabellen**

   "Hinter" diesen Objekten kann VBA Code vorhanden sein, im vorliegenden Fall zur Code-Ausführung der Buttons in den Tabellen
   Tabelle1 [CodeName], gesetzter Registername "CodeCleaner" [Name]
   Tabelle2, gesetzter Registername "ExchangeModules"
   Tabelle3, gesetzter Registername "ImportModules".
   "Hinter" 'DieseArbeitsmappe' kann z.B. ein AutoOpen-Code vorgesehen werden, der bestimmte Funktionen beim Öffnen der Arbeitsmappe ausführt.
   VBA spricht die entsprechenden Module mit dem 'CodeName' an, in deutschsprachigen Systemen mit 'TabelleX' bzw. 'DieseArbeitsmappe'.
   Ein manueller Export aus der VBE erzeugt daraus eine Textdatei, z.B. 'Tabelle3.cls' oder 'DieseArbeitsmappe.cls'.

2. **Forms (UserForm) Module**

   Ein manueller Export aus der VBE erzeugt daraus eine Textdatei, z.B. 'frmCodeCleaner.frm' und eine Binärdatei 'frmCodeCleaner.frx', die mit 'frmCodeCleaner.frm' verlinkt ist.

3. **Standard Module**

   Ein manueller Export aus der VBE erzeugt daraus eine Textdatei, z.B. 'modExcel.bas'.

4. **Klassenmodule** (hier nicht verwendet)

   Ein manueller Export aus der VBE erzeugt daraus eine Textdatei, z.B. 'ClassModule.cls'.

Die Komponenten aus Ziffer 1 (DieseArbeitsmappe / Tabellen) werden anders behandelt als die Module aus den Ziffern 2 bis 4.

Die verschiedenen Datei Such-Funktionen bieten zunächst das Verzeichnis dieser Excel-Arbeitsmappe an. Der Code Cleaner legt auch hier die Sicherungsverzeichnisse an. Es ist daher zweckmäßig, die zu behandelnden Excel-Arbeitsmappen hier hinein zu kopieren.

1 Code Cleaner

Funktionsumfang:

1. Vermindert die Wahrscheinlichkeit von Laufzeitfehlern durch Bereinigung von Kompilierungsresten aus vorangegangenen Codebearbeitungen während der Erstellungsphase. Solch mögliche Instabilitäten sind nicht im Quellcode zu beheben, weil das Problem in einem lauffähigen Code selber nicht liegt,
2. vermindert die Wahrscheinlichkeit von Kompilierungskonflikten zwischen verschiedenen MS Office-Versionen,
3. beschleunigt die Ladezeit,
4. reduziert die Dateigröße.

Im Sheet "CodeCleaner" sind 2 Funktionen abrufbar:

1. Code cleaner
2. Clear backup files or folders

1.1 Code Cleaner

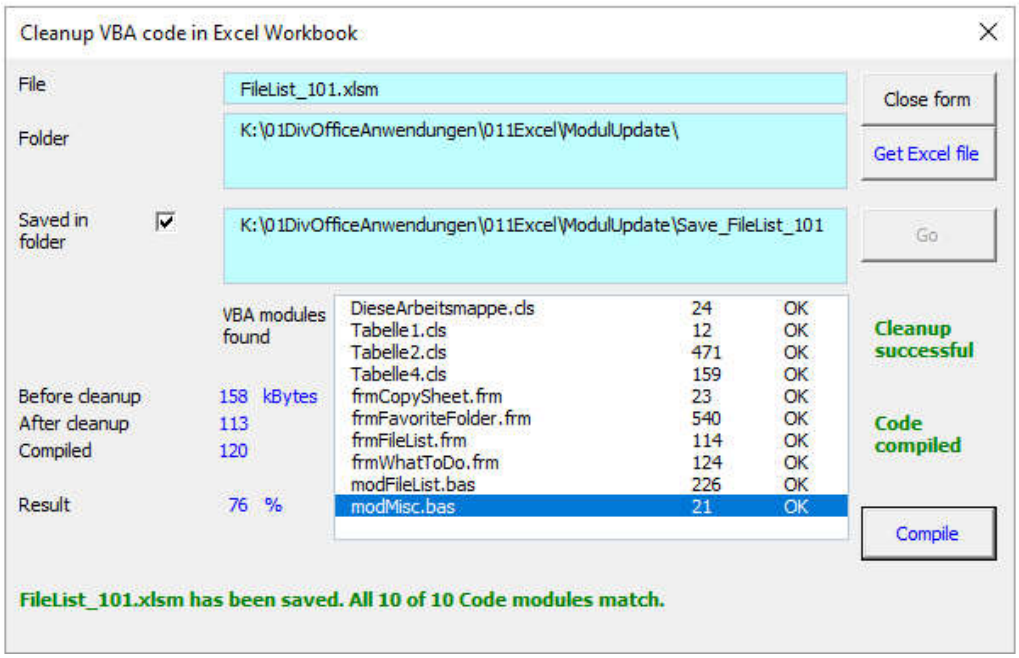


Abb. 1.1: Formular Code cleaner.

Die Zahlen in der Liste zeigen die Anzahl der Codezeilen je Modul an.

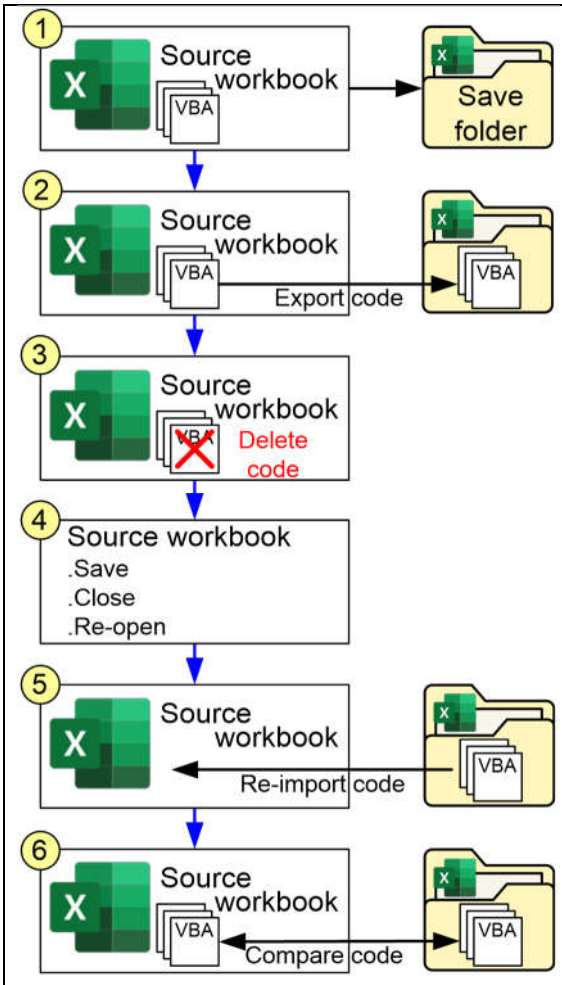
Der in allen nachfolgenden Formularen vorhandene Button "Close form" stellt sicher, dass die jeweils geöffneten Excel-Dateien geordnet geschlossen werden, insbesondere die veränderte Zieldatei abgespeichert wird. Ein Schließen der Formulare über das "X" rechts oben wird deshalb unterbunden.

a) Get Excel file

Um sicher zu gehen, sollte die zu behandelnde Datei vorher in das Verzeichnis des CodeCleaners kopiert werden.
Lokalisierung der zu behandelnden Excel-Datei (Source workbook) mit dem File-Such-Dialog.
Unter dem Verzeichnis des Code Cleaners wird ein Sicherungsverzeichnis …\Save_+Name der Excel-Datei angelegt, angezeigt in "Saved in folder".

b) Go

Die Bearbeitung des Cleanup wird angestoßen, der Ablauf der Einzelfunktionen sieht folgendermaßen aus:



Eine Kopie des Source workbook wird mit dem Namen "Bak_"+Name des Source workbooks im Sicherungsverzeichnis abgespeichert.

Alle VBA Code-Module, auch Code-Komponenten zu Tabellen oder Arbeitsmappen, werden in das Sicherungsverzeichnis als Textfiles exportiert (anders als beim manuellen Export reiner Code ohne Declaration Header)

Im Source workbook werden alle VBA Code-Module gelöscht.

Das Source workbook, nun ohne VBA-Code, wird abgespeichert, geschlossen und wieder geöffnet.

Die exportierten VBA Code-Module werden aus dem Sicherungsverzeichnis zurück in das Source workbook reimportiert. Bibliotheksverweise bleiben unberührt.

Der VBA-Code im Source workbook wird zeilenweise mit dem Code im Sicherungsverzeichnis verglichen. Übereinstimmungen je Modul mit "OK" und in der unteren Zeile grün angezeigt.
Bei Nicht-Übereinstimmungen mit "??" je Modul und einer roten Meldung in der unteren Zeile.

Abb. 1.2: Ablaufplan der Codebereinigung hinter dem Button "Go"

Wenn der Wischmopp einmal durchgefahren ist, reduziert sich die Dateigröße mehr oder weniger deutlich, insbesondere dann, wenn die Arbeitsmappe bereits eine längere Geschichte diverser Code-Bearbeitungen hinter sich hat.

c) Compile

Der Code ist im nun bereinigten Source workbook lauffähig. Excel "übersetzt", d.h. interpretiert den Code zeilenweise unmittelbar während der Laufzeit. Mit der Funktion "Compile" wird zunächst der Code auf Syntax- und Verweis-Fehler überprüft. Weiterhin wird intern, in einem nicht sichtbaren Speicherbereich der Excel-Datei, der Quellcode in einen sog. P-Code vorübersetzt, den Excel ausführt. Bei anderen Programmen der MS Office-Familie ist es vergleichbar. Das verbessert etwas die Laufzeit, erhöht aber die Dateigröße geringfügig.

Microsoft empfiehlt, bei Wechsel der Excel-Version immer ein Compile vorzunehmen.

1.2 Clear backup files or folders

Nach der Bearbeitung mit dem Code Cleaner werden die dabei exportierten Dateien normalerweise nicht mehr gebraucht. Mit dieser Funktion können sie wieder gelöscht werden. Sollte etwas schief gelaufen sein, steht die Kopie des Source Workbook ("Bak_...xlsm") noch zur Verfügung.

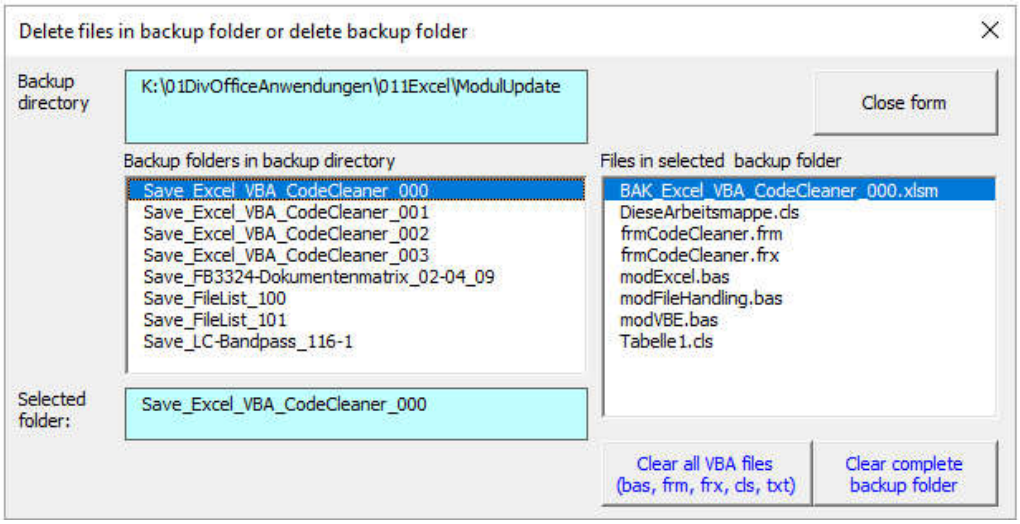


Abb. 1.3: Backup files oder Verzeichnisse löschen.

Alle mit dem Code Cleaner im Backup Directory angelegten "Save"-Verzeichnisse werden in der linken Liste angezeigt. Ein Klick auf einen Eintrag in der Verzeichnisliste zeigt in der rechten Liste die dort gespeicherten VBA-Modul-Files. Entweder alle VBA-Modul-Files außer das Backup workbook "Bak_...xlsm" löschen oder das gesamte ausgewählte "Save-Verzeichnis.

2 VBA-Module zwischen Excel-Dateien austauschen

Funktionsumfang:

1 Importieren von VBA-Modulen aus einer Excel-Anwendung in eine noch leere Excel-Datei,
2 Austauschen, sprich Update, vorhandener Module

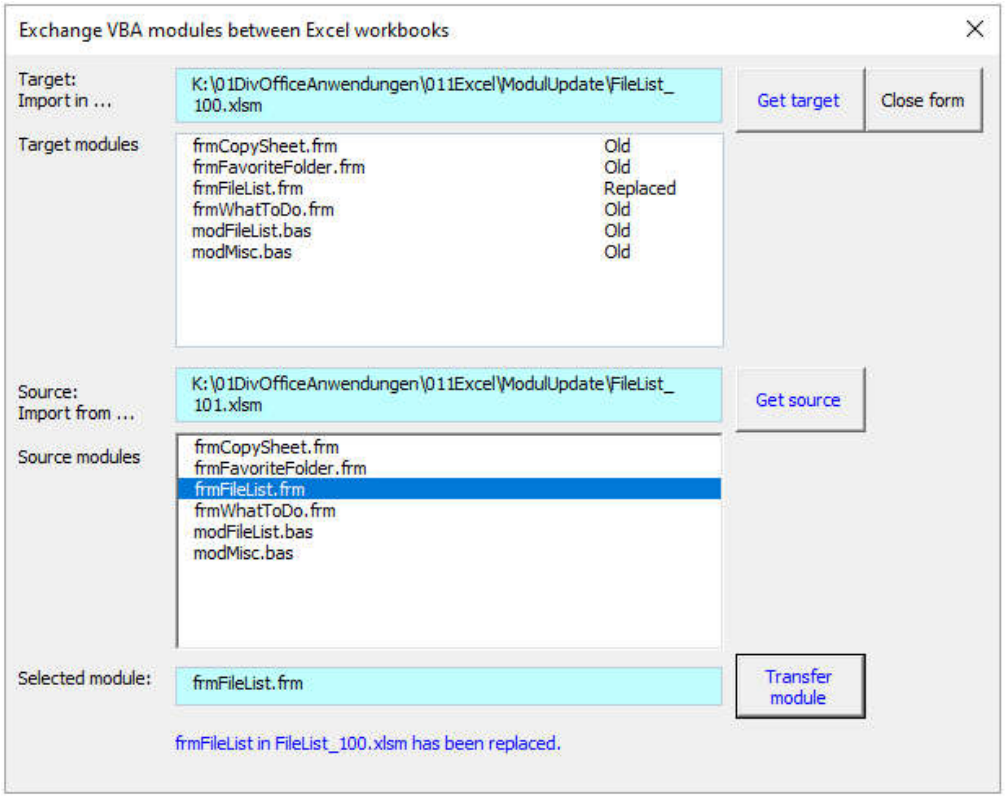


Abb. 2.1: Formular Exchange VBA modules.

a) Get Target

Die zu bearbeitende Zieldatei (Target) mit dem File-Such-Dialog lokalisieren. Die VBA-Module werden in der Liste "Target modules" angezeigt.

b) Get Source

Entsprechend für die Quelldatei (Source), aus der Module in die Zieldatei zu übertragen sind.

In beiden Fällen nur Standardmodule (.bas), Klassenmodule (.cls) und Forms-Module (.frm) angezeigt, nicht (!) ggf. in Tabellen (Sheets) oder im Workbook (DieseArbeitsmappe / ThisWorkbook) angehängter VBA-Code.

Das zu exportierende Modul in der Liste "Source modules" markieren, erscheint darunter in "Selected module".

c) Transfer module

Exportiert das gewählte Modul von Source nach Target.
Ist ein gleichnamiges Modul bereits im Target vorhanden, wird gefragt, ob dieses überschrieben werden soll. Wenn ja, erfolgt Vollzugsmeldung "Replaced" in der Target Moduliste.
Wenn das gewählte Modul noch nicht im Target vorhanden ist, wird es eingefügt, Kennzeichnung "Inserted" in der Target Modulliste.
Forms-Module liegen als 2 Dateien vor: .frm und .frx. .frx wird automatisch mit übertragen.

3 VBA Codefiles in eine Excel-Datei importieren

Funktionsumfang:

1 Import von VBA Code, der als Textdatei vorliegt, in eine Excel-Zieldatei.
Die Textdatei kann etwa aus den vom Code Cleaner ausgelagerten Modulen, aus manuell aus der VBE exportierten Modulen oder aus einer anderen Quelle stammen.

2 Ergänzend zur Anwendung aus Abschnitt 2 sind hier auch Code-Komponenten zu Tabellen oder Arbeitsmappen zum Import vorgesehen. Hier ist zu beachten, dass der Dateiname der Konvention von CodeNamen der Tabellen und der Arbeitsmappe entspricht.

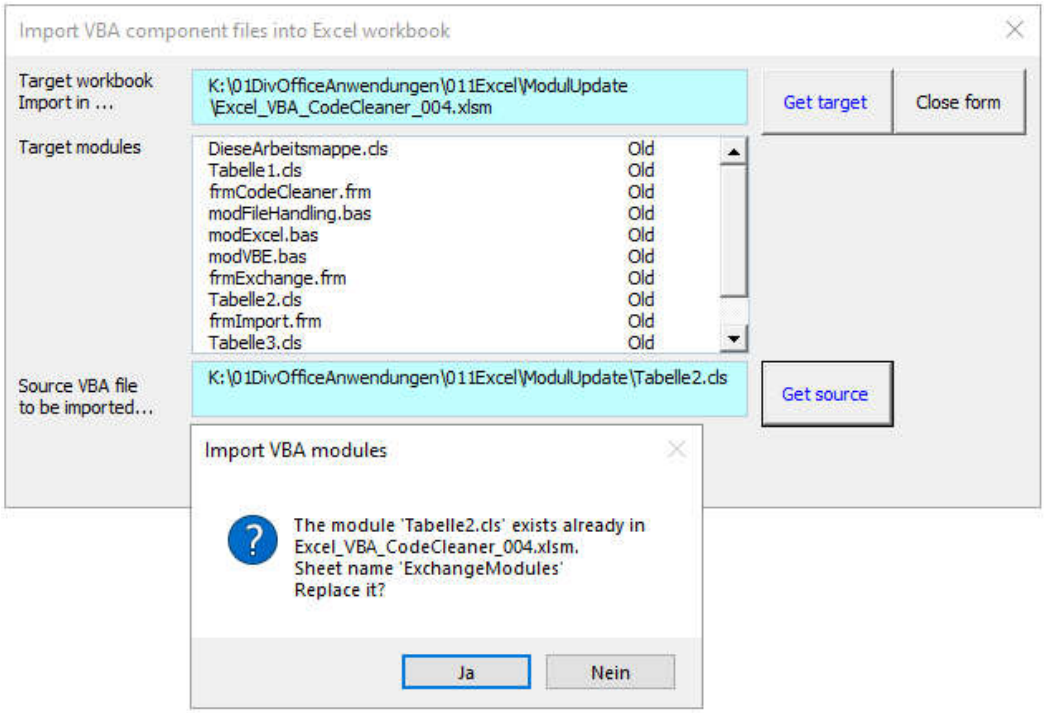


Abb. 3.1: Import von VBA Textdateien.

Das Vorgehen unterscheidet sich nach Art des Code-Moduls.

a) Code zu Tabellen und Arbeitsmappen

Der Dateiname der Quelldatei muss (!) mit dem CodeNamen des Zielobjekts in der Zieldatei übereinstimmen.

- Eine Quelldatei 'DieseArbeitsmappe.cls' oder z.B. in englischsprachigen Systemen 'ThisWorkbook.cls' enthält den Code für die Arbeitsmappe.
- Eine Quelldatei 'Tabelle2.cls' oder z.B. in englischsprachigen Systemen 'Sheet2.cls' enthält den Code für Tabelle2 bzw. Sheet2.
- In beiden Fällen ist der CodeName von Arbeitsmappe oder Tabelle maßgeblich, s.o. Abb. 0.3. Das Programm sucht anhand des Dateinamens nach den CodeNamen in der Liste 'Target modules' (Abb. 3.1).
- Die CodeNamen von Tabellen, "Tabelle" bzw. englisch "Sheet", und von Arbeitsmappen, "DieseArbeitsmappe" bzw. englisch "ThisWorkbook", können je nach Sprachversion im Sheet "ImportModules" angepasst werden.

Nachdem der vorhandene Code zu dem Objekt mit dem passenden CodeName in der Zieldatei gelöscht wurde, wird der Code-Text zeilenweise aus der Quelldatei in das Objekt (Tabelle / Arbeitsmappe) übertragen. Im Beispiel Abb. 3.1 ist es der Code zu Tabelle 2 (CodeName). Der im Tabellenregister festgelegte Name 'ExchangeModules' wird zur Sicherheit bei der Frage nach dem Überschreiben (Abb. 3.1) mit angegeben.

Manuell aus der VBE exportierte VBA-Module enthalten im Dateikopf einen Declaration Header, der mit einer oder mehreren "Attribute"-Zeilen endet. Dieser Erklärungsteil wird bis zum letzten "Attribute" nicht mit importiert.

Der Vollzug wird in der Liste 'Target modules' mit "Replaced" angezeigt.

Es kann nur Code zu in der Zieldatei vorhandenen Tabellen importiert werden. Ein Zufügen von Tabellencode zu einer im Target workbook nicht vorhandenen Tabelle, z.B. Einfügen einer Datei "Tabelle5.cls" im Beispiel in Abb. 3.1 wird nicht zugelassen.

b) Code zu UserForms (.frm), Standardmodulen (.bas), Klassenmodulen (.cls)

Diese Code-Module werden als Einheit übertragen. Sie müssen neben dem Quelltext auch den Declaration Header enthalten, wie er aus dem manuellen Export aus der VBE erzeugt wird. UserForm-Module liegen zweifach, als .frm und als .frx vor.

Bei Übereinstimmung des Quell-Dateinamens mit dem Modulnamen in der Liste 'Target modules' wird das Modul nach Löschen in der Zieldatei durch das importierte ersetzt. Vollzugsmeldung wie oben "Replaced" in der Liste 'Target modules'.

Ist das Modul nicht in der Zieldatei vorhanden, wird es neu eingefügt, Vollzugsmeldung "Inserted" in der Liste 'Target modules'.

4 Trust Center zurücksetzen

Zur Sicherheit sollte nach Abschluss der Arbeiten die Einstellung "Zugriff auf das VBA-Projektobjektmodell vertrauen" (Abb. 0.1) wieder zurückgesetzt werden: Haken wegnehmen.