Tuhls' kleine RibbonUI-Fibel

Menüband anpassen – leicht gemacht

Der mit DATEI | OPTIONEN | MENÜBAND ANPASSEN zu öffnende Dialog MENÜBAND ANPASSEN lässt einige persönliche Anpassungen der Registerkarten im Menüband zu, hat allerdings eine Menge Einschränkungen. Die größte ist die Beschränkung auf den aktuellen Computer. Um angepasste Einstellungen »mitzunehmen«, ist eine ExportedUI-Datei auf dem anderen Computer zu installieren.

Per XML-Programmierung bringt eine Datei oder eine Vorlage die Anpassungen mit, ohne dass es zusätzlicher Maßnahmen bedarf. Darum ist diese Methode ideal zum Erstellen von Gruppenvorlagen.

Um so etwas zu realisieren, müssen Sie sich nicht großartig mit XML auskennen. Mit Hilfe spezieller Editor-Programme, ein paar Listen zum Nachschlagen und dieser Anleitung schaffen Sie das auch ohne Vorkenntnisse.

Warum noch eine RibbonUI-Anleitung?

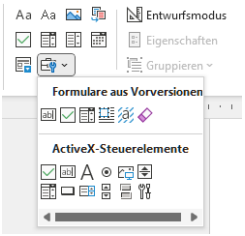
Ich musste mich im Rahmen eines Projekts mit der Ribbon-Programmierung auseinandersetzen und wühlte mich durch die Eingeweide des Internets. Mein Ergebnis: Es ist alles da, aber nur mühsam zu finden und noch mühsamer als Einsteiger zu verstehen. Darum machte ich mich nach Abschluss des Projekts daran, meine Erkenntnisse in einsteigergerechter Weise aufzuschreiben.

Es ist nur ein Einstieg, um ein paar Schaltflächen ins Menüband zu schmuggeln. Für weitergehende Wünsche sei auf die ausführlicheren Quellen verwiesen.

Hinweis

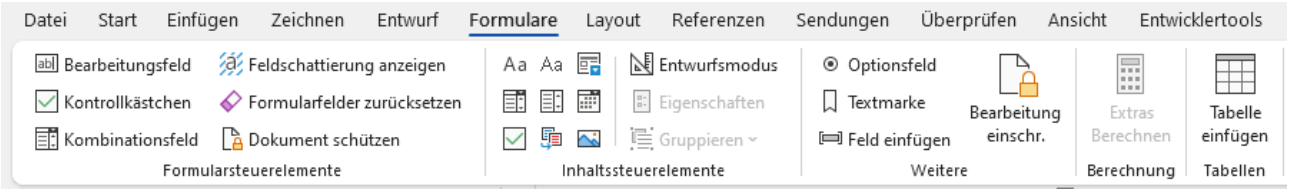
Diese Anleitung hat kein Inhaltsverzeichnis. Benutzen Sie bitte die Navigation Ihres PDF-Readers!

Ein Beispiel



Die Funktionen zur Formulargestaltung sind von Microsoft in die hinterste Word-Ecke verbannt worden: ENTWICKLERTOOLS | *Steuerelemente*: . Mit Klick auf endlich stehen einem alle Formularsteuerelemente zur Verfügung.

Mit ein paar XML-Zeilen (genau 44) bauen Sie sich eine eigene Registerkarte »Formulare«, die alle zur Formularbearbeitung erforderlichen Funktionen konzentriert bietet.[1]

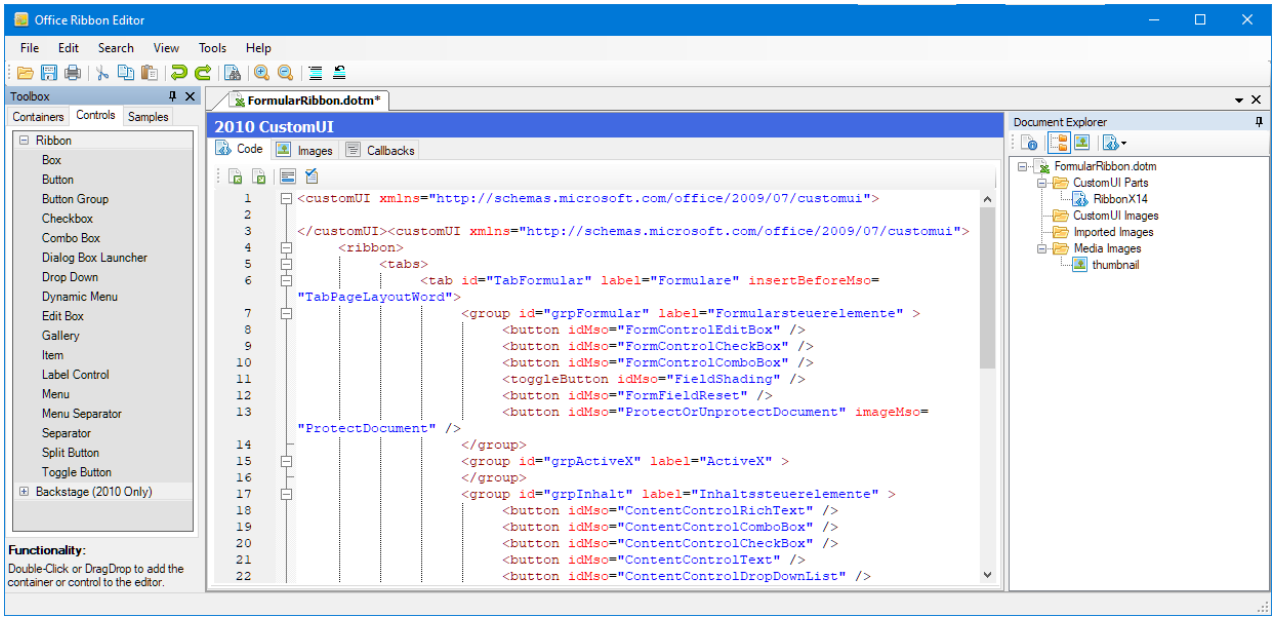


[1] Download und ausführliche Beschreibung: https://oerttel.net/kommunizieren/formulare/

Die Editoren

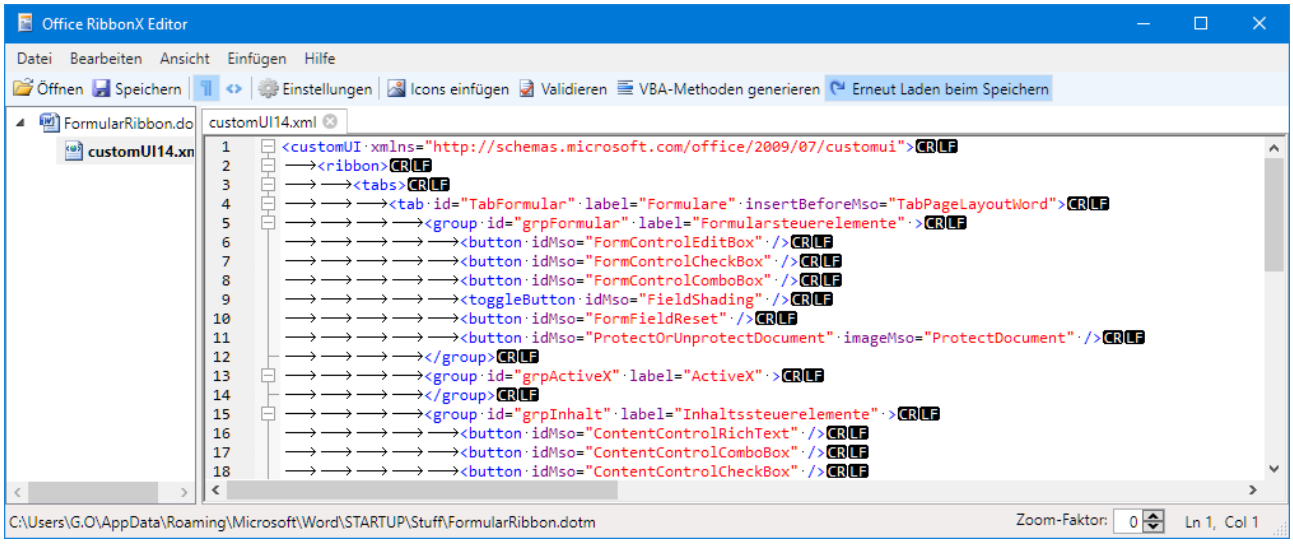
Microsoft hatte mal einen eigenen **Custom UI Editor for Microsoft Office** für die XML-Gestaltung des Menübandes im Angebot, doch dessen Vertrieb aus ungeklärten Gründen eingestellt. In den Tiefen des Netzes finden sich noch Download-Angebote, z. B. bei https://github.com/OfficeDev/office-custom-ui-editor; doch andere Editoren sind besser.

Die Freeware **Office Ribbon Editor** von *Leaf Creation* wird über diverse Download-Server vertrieben. Er bietet mehr Unterstützung als der doch eher schlichte Microsoft-Editor, z. B. indem korrekte Control-Platzhalter eingefügt werden, die Sie dann nur noch mit individuellen Daten füllen.



Der »Office Ribbon Editor« von Leaf Creations

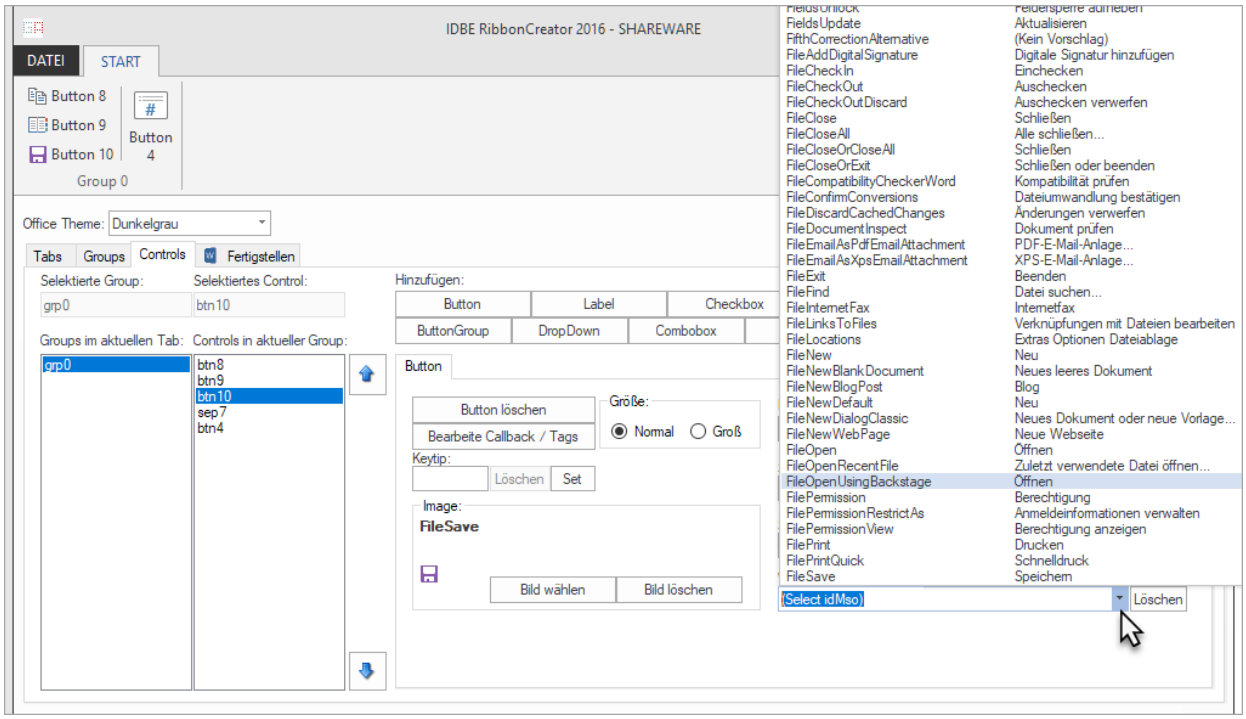
Der **Office RibbonX Editor** von *Fernando Andreu* (https://github.com/femandreu/office-ribbonx-editor) ähnelt dem Modell von Leaf Creation und ist ebenfalls Freeware. Er besitzt eine deutsche Benutzerführung, aber keine Assistenzfunktion beim Einfügen der Controls in korrekter Syntax.. Sie sollten einfach beide ausprobieren.



Der »Office RibbonX Editor« von Fernando Andreu

Dann gibt es von *Avenius* noch den **IDBE Ribbon Creator**, der allerlei Unterstützung bietet, aber ebenfalls Grundkenntnisse der Ribbon-Programmierung voraussetzt. Für Amateurzwecke ist er mit 32 € recht teuer, für Profis entbehrlich. Bei http://www.ribboncreator2016.de/ kann eine kostenlose Testversion bezogen werden. Seine besonderen Vorteile sind

- das Baukastensystem, das den XML-Code anhand der ausgewählten Controls sicher aufbaut,
- die Vorschau auf das Aussehen des gestalteten Menübands und
- die eingebauten Bezeichner für `idMso` und `imageMso` (rechts im Bild).



Der »IDBE Ribbon Creator« von Avenius

Hinweis

Die weiteren Beschreibungen beziehen sich auf den **»Office Ribbon Editor«** von **Leaf Creations**.

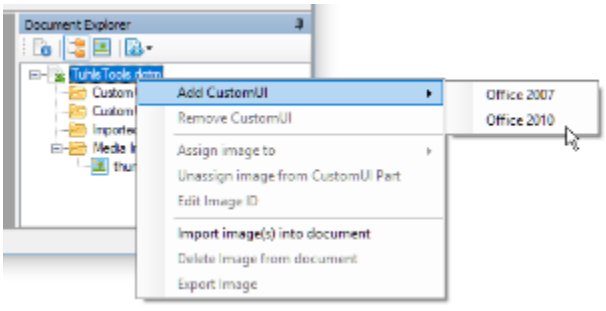
Custom UI erstellen

Nach dem Start des Editors laden Sie mit FILE ein Word-Dokument – vorzugsweise eine **Vorlagendatei.dotm**.

Wichtig

Die Word-Datei muss geschlossen sein, sonst kann der Editor sie nicht speichern.

In der Grundeinstellung sehen Sie links das Codefenster und rechts eine Struktur, die noch angereichert werden muss. Dazu bedarf es zuallererst einer CustomUI, die Sie beim Office Ribbon Editor mit im Rechtsklick auf den Dateinamen | ADD CUSTOMUI | OFFICE 2010 erstellen. Die Version 2010 gilt für alle Office-Versionen seither.



Im Bereich CUSTOMUI PARTS erscheint ein Eintrag RIBBONX 14, den Sie doppelklicken und damit das Codefenster öffnen (Abb. auf Seite 1).

In dieses Fenster gehört nun das Grundgerüst, der Rahmen einer CustomUI:

```
<customUI xmlns="http://schemas.microsoft.com/office/2009/07/customui">
<!--    Zwischen diese beiden Zeilen kommt alles, was Ihre persönliche Word-Oberfläche ausma-
        chen soll, die so genannten Container und Controls.
        Dieser Text zeigt, wie nicht zum Code gehörende Texte gekennzeichnet werden.-->
</customUI>
```

Dabei ist auf die Hierarchie des Menübands zu achten. Container (rot) enthalten andere, untergeordnete Container. Die untersten Containerebenen enthalten die Controls (grün), die ausführbaren Programmcode aufrufen.

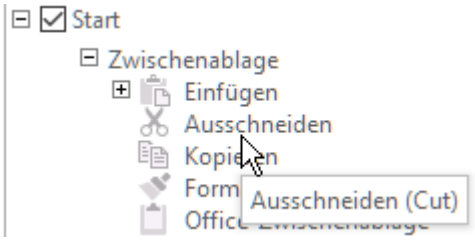
Controls sind erst unterhalb der Containerebene *group* anzutreffen. In einer Gruppe können noch Container wie *box, menu, gallery* oder *splitButton* zwischengeschoben werden, die zu den Controls gerechnet werden, in denen aber mehrere Controls vereint sind.

Wenn Sie also in Ihrem Menüband etwas verändern wollen, nehmen wir als Beispiel eine Ergänzung des Registers START, brauchen Sie zunächst diese Zeilen als Rückgrat Ihres Codes:

```
<customUI xmlns="http://schemas.microsoft.com/office/2009/07/customui">
    <ribbon>
        <tabs>
            <tab idMso="TabHome" >

            </tab>
        </tabs>
    </ribbon>
</customUI>
```

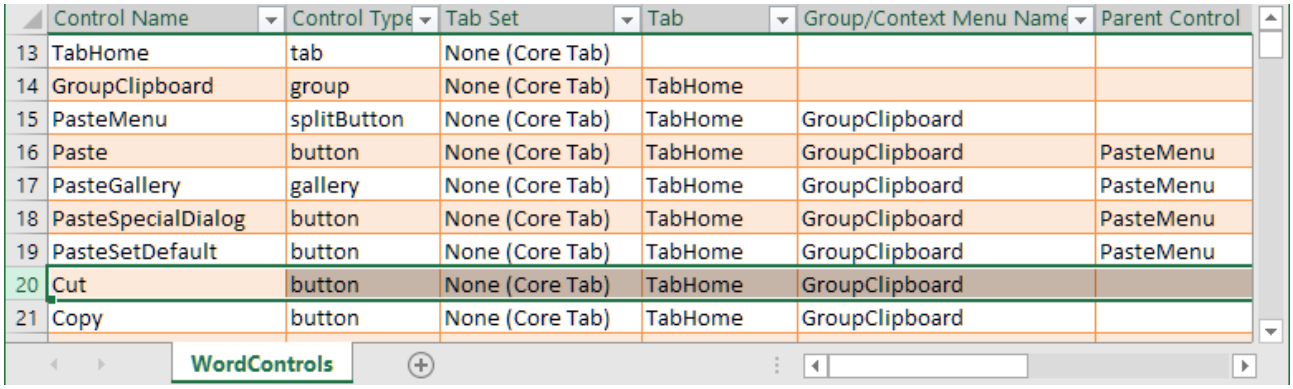
In der `tab`- Ebene beginnen die Schwierigkeiten. Jeder Eintrag benötigt einen Namen, genannt *ID*, der ihn eindeutig identifiziert. `idMso` ist immer dann zu verwenden, wenn es um Office-eigene Funktionen geht, hier also `<tab idMso="TabHome" >` für die Registerkarte START. Für selbstdefinierte Elemente steht dann nur `id="Name"`; da dürfen Sie einen Namen Ihrer Wahl verwenden, er darf aber nur einmal innerhalb einer CustomUI verwendet werden.

Die Namen für Befehle erfahren Sie aus einer Quickinfo, wenn Sie im Dialog MENÜBAND ANPASSEN mit dem Mauszeiger auf einem Eintrag verweilen. Der Name in der Klammer ist der interne Name des Befehls. Ausführlicher ist eine Zusammenstellung, die Sie direkt bei Microsoft herunterladen können:



https://www.microsoft.com/en-us/download/confirmation.aspx?id=50745

Hier sind in Excel-Tabellen nicht nur alle Namen von der Registerkarte bis zum Einzelbefehl aufgelistet, sondern auch ihre Typen, Einordnung in die Ribbon-Hierarchie und Vorkommen.



	Control Name	Control Type	Tab Set	Tab	Group/Context Menu Name	Parent Control
13	TabHome	tab	None (Core Tab)			
14	GroupClipboard	group	None (Core Tab)	TabHome		
15	PasteMenu	splitButton	None (Core Tab)	TabHome	GroupClipboard	
16	Paste	button	None (Core Tab)	TabHome	GroupClipboard	PasteMenu
17	PasteGallery	gallery	None (Core Tab)	TabHome	GroupClipboard	PasteMenu
18	PasteSpecialDialog	button	None (Core Tab)	TabHome	GroupClipboard	PasteMenu
19	PasteSetDefault	button	None (Core Tab)	TabHome	GroupClipboard	PasteMenu
20	Cut	button	None (Core Tab)	TabHome	GroupClipboard	
21	Copy	button	None (Core Tab)	TabHome	GroupClipboard	

WordControls

Wichtig

Diese Kenntnisse sind wichtig für den Einsatz der Container und Controls, denn XML ist da pingelig: Die Namen müssen

- exakt geschrieben sein (Groß-Klein-Schreibung beachten!) und
- zum angegebenen Container-/Controltyp passen.

Eigene Registerkarten und Gruppen positionieren

Grundsätzlich werden zusätzliche Register im Menüband und Gruppen in bestehenden Registern ans Ende angehängt. Wenn Sie dies nicht wollen, statten Sie die Container-/Gruppeneinträge mit dem zusätzlichen Parameter
`insertBeforeMso="TabName/GroupName"` oder
`insertAfterMso="TabName/GroupName"` aus.

Für `TabName/GroupName` setzen Sie die `idMso` der RegKarte/Gruppe ein, vor bzw. nach der Ihre benutzerdefinierte RegKarte/Gruppe einzufügen ist.

Controls einfügen

Wie auch im Dialog MENÜBAND ANPASSEN ist es hier nicht möglich, neue Funktionen in die bestehenden Gruppen einzuarbeiten. Es müssen eigene Gruppen angelegt werden, die die Funktionen aufnehmen.

```
<customUI xmlns="http://schemas.microsoft.com/office/2009/07/customui">
    <ribbon>
        <tabs>
            <tab idMso="TabHome" >
                <group id="grpEigene" label="Eigene" >
<!--                    In dieser Ebene sind eigene Funktionen möglich.-->
                </group>
            </tab>
        </tabs>
    </ribbon>
</customUI>
```

In der grünen Zeile sehen Sie auch, wie ein Kommentar vom Code abgegrenzt wird: Er beginnt mit `<!--` und wird abgeschlossen mit `-->`. Der Editor unterstützt das Kommentieren bzw. dessen Aufhebung mit den Schaltflächen  .

Wichtig

Für id- und label-Einträge dürfen nur kleine und große Buchstaben sowie Ziffern benutzt werden, keine Sonderzeichen. Sie dürfen nicht mit einer Ziffer beginnen.

Praxis: Word-Funktionen in andere Positionen bringen

Nehmen wir an, in diese Gruppe sollen die Funktionen

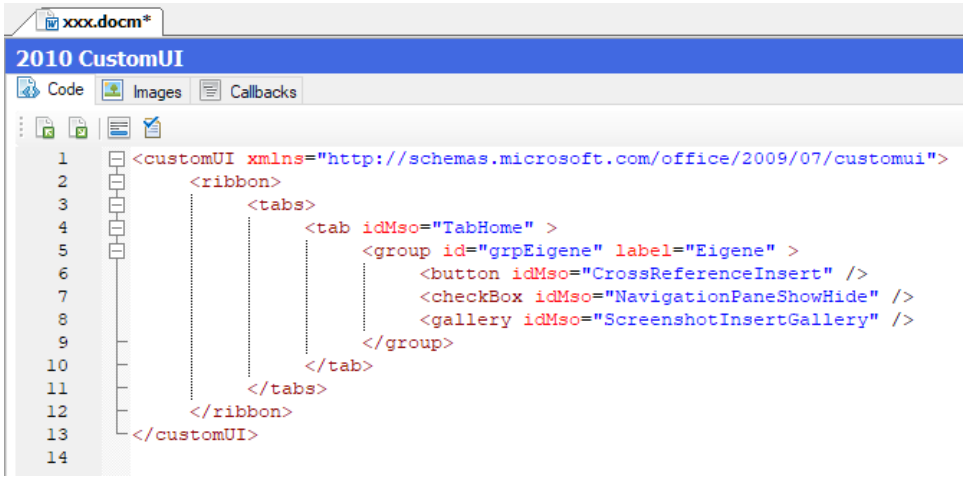
- QUERVERWEIS aus der Registerkarte VERWEISE/REFERENZEN und
- NAVIGATIONSBEREICH aus der Registerkarte ANSICHT
- SCREENSHOT aus der Registerkarte EINFÜGEN,

eingefügt werden, weil sie dort schneller aufzurufen sind.

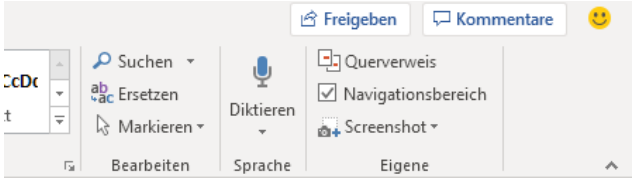
Für diese Befehle gibt es folgende Controls:

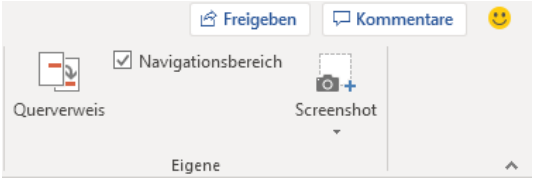
Control	Control-Typ	XML-Codezeilen
CrossReferenceInsert	button	`<button idMso="CrossReferenceInsert" />`
NavigationPaneShowHide	checkBox	`<checkBox idMso="NavigationPaneShowHide" />`
ScreenshotInsertGallery	gallery	`<gallery idMso="ScreenshotInsertGallery" />`

Diese drei Zeilen anstelle der grünen Kommentarzeile in den Code eingefügt, sehen dann so aus …



… und führen zu diesem Ergebnis im Menüband (links).

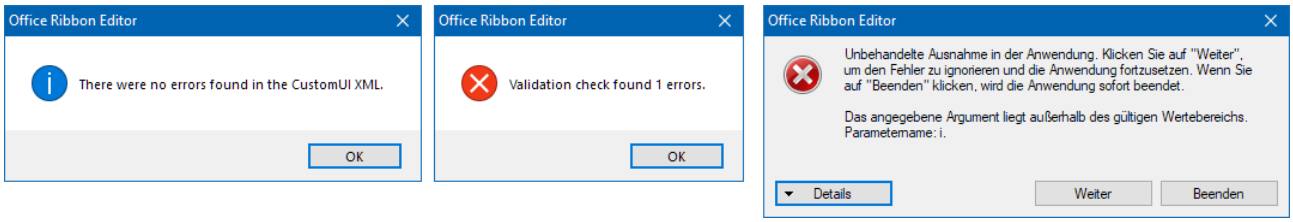




Standardmäßig werden kleine Schaltflächen zu dritt übereinander eingefügt. Wünschen Sie große Schaltflächen, ergänzen Sie die jeweilige Control-Zeile innerhalb der spitzen Klammern um `size="large"`. Das klappt allerdings nicht mit allen Controls; Checkboxen z. B. widersetzen sich einer Vergrößerung (Bild rechts).

Keine Angst vor Fehlermeldungen!

Klicken Sie regelmäßig die Schaltfläche über dem Codefenster an, um zu kontrollieren, ob der eingegebene Code korrekt ist.



Wenn die linke Meldung erscheint, ist alles in Ordnung, Sie können weitermachen.

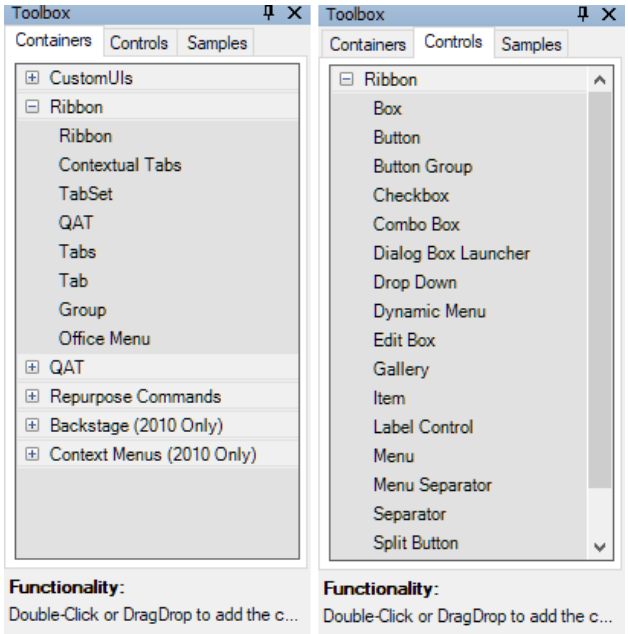
Bei der mittleren Meldung zeigt ein Fenster am unteren Rand des Editors zusätzliche Hinweise zum Fundort des Fehlers an, was häufig aber auch nicht weiterhilft, weil die Fehlerbeschreibung zwar logisch, der Fehler aber nicht unbedingt nicht unbedingt in der angegebenen Zeile zu finden ist.

Die Fehlermeldung rechts sieht schlimmer aus als sie ist. Klicken Sie *auf jeden Fall auf* ✕ und machen Sie sich an die Fehlersuche.

Die häufigsten Ursachen für Fehlermeldungen sind

- Schreibfehler bei Containern oder Controls,
- doppelte ID-Vergabe,
- nicht sauber geschlossene Container,
- sonstige Tippfehler, schon ein vergessenes Leerzeichen gereicht zum Alarm.

Das saubere Schließen von Containern und Sammel-Controls lässt sich vermeiden, indem Sie eine Hilfsfunktion des Editors verwenden: Blenden Sie mit VIEW | TOOLBOX links eine Sidebar ein, die das korrekte Eröffnen und Abschließen erleichtert.



So fügt zum Beispiel ein Doppelklick auf MENU drei Codezeilen an der Cursorposition ein:

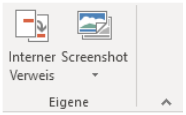
```
<menu id="" >


</menu>
```

Die Daten der Menüs sind zu ergänzen und darunter die Menübefehle einzutragen. Der formale Abschluss steht schon da.

Schaltflächen umgestalten

Sollten Ihnen die Standard-Darstellungen der Schaltflächen nicht gefallen, lässt sich dem abhelfen. Durch Einfügen eines Parameters `label="Schaltflächenname"` ändern Sie die Bezeichnung der Schaltfläche ab. Das Schaltflächensymbol (Icon) wechseln Sie mit `image="Bild"` oder `imageMso="Bild"`. Die Abbildung zeigt die Codeänderung und deren Auswirkung im Menüband.

```
<customUI xmlns="http://schemas.microsoft.com/office/2009/07/customui">
    <ribbon>
        <tabs>
            <tab idMso="TabHome" >
                <group id="grpEigene" label="Eigene" >
                    <button idMso="CrossReferenceInsert" label="Interner Verweis" size="large" />
                    <gallery idMso="ScreenshotInsertGallery" imageMso="PhotoAlbumInsert" size="large" />
                </group>
            </tab>
        </tabs>
    </ribbon>
</customUI>
```



Während Sie beim Namen völlig frei sind, gibt es für die Icons die Bedingung, dass sie erreichbar sein müssen. `imageMso` sind immer erreichbar, weil Bestandteil von Word. Die Bezeichnungen stimmen mit jenen der Controls überein; auf der Site https://bert-toolkit.com/imagemso-list.html gibt es eine Tabelle mit Mustern und XML-gerechten Strings zum Kopieren.

Eigene Icons verwenden

Für mit `image="Bild"` zu verwendende eigene Kreationen können als Pixelgrafiken der gängigen Formate JPG, TIF, PNG und GIF in die Word-Datei importiert werden, indem Sie vom Code-Fenster auf das Register IMAGES umschalten. Mit der Schaltfläche , gefolgt von FROM FILE SYSTEM öffnen Sie einen Dateiauswahl-Dialog.

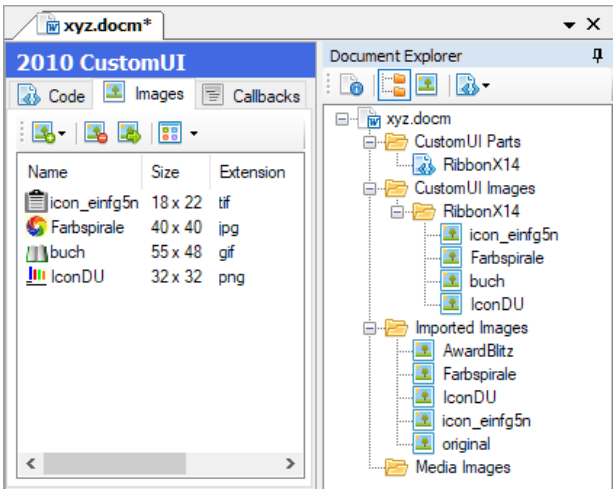
Die Icons sollten annähernd quadratisch und nicht zu groß sein. Word skaliert sie intern auf ein quadratisches Format mit 16×16 Pixel (px) für kleine Schaltflächen oder 32×32 px für große. Icons, die zu weit ab vom quadratischen Format sind, werden verzerrt, und zu große belasten unnötig die Dateigröße.

Nach der Auswahl werden die Icons gleich dreimal im Editor gelistet: einmal im IMAGES-Fenster und gleich doppelt im EXPLORER-Fenster in den Rubriken CUSTOMUI IMAGES und IMPORTED IMAGES. Sie können die Namen der Icons nach dem Import noch ändern. Wird der Name im IMAGES-Fenster oder in der Rubrik CUSTOMUI IMAGES geändert, wirkt sich das nicht auf den Namen in der Rubrik IMPORTED IMAGES aus.

Dort bleibt der Name vom Import erhalten (siehe im Screenshot Eintrag »buch«, der als »original« importiert wurde).

Ebenso bleiben in im IMAGES-Fenster oder in CUSTOMUI IMAGES gelöschte Icons in IMPORTED IMAGES erhalten (siehe »AwardBlitz«).

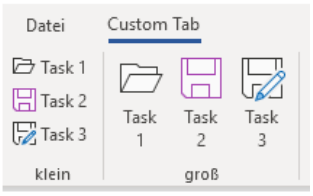
Den Namen des Icons fügen Sie mit `image="Iconname"` in die Control-Zeile ein. **Die Dateiendung ist wegzulassen!**



Icon-Größe festlegen

Ergänzen Sie eine Zeile um den Parameter `size="large"`, wird die Schaltfläche über die volle Höhe des Menübands angezeigt.

```
<tab id="customTab" label="Custom Tab">
    <group id="grpStd" label="klein">
        <button id="task1s" label="Task 1" imageMso="FileOpen"/>
        <button id="task2s" label="Task 2" imageMso="FileSave"/>
        <button id="task3s" label="Task 3" imageMso="FileSaveAs"/>
    </group>
    <group id="grpStd2" label="groß">
        <button id="task1l" label="Task 1" imageMso="FileOpen" size="large" />
        <button id="task2l" label="Task 2" imageMso="FileSave" size="large" />
        <button id="task3l" label="Task 3" imageMso="FileSaveAs" size="large" />
    </group>
</tab>
```



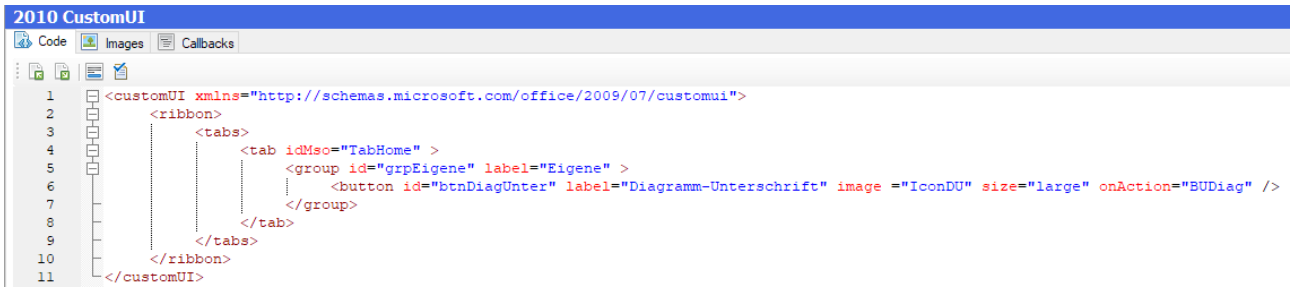
Eigene Makros aufrufen

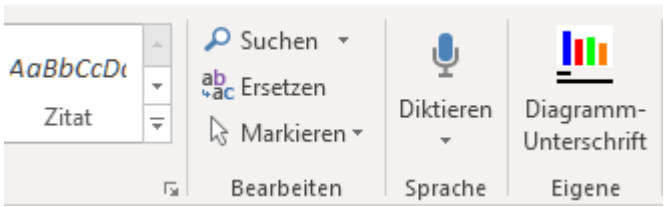
Wie beim normalen Anpassen des Menübands lassen sich auch selbst erstellte VBA-Makros für komplexere Aufgaben mit dem Editor ins Menüband einbringen. Dafür muss die Control-Zeile ein Argument `onAction="Markroname"` enthalten.

Beispiel

Ein Makro namens »BUDiag « soll mit einer Schaltfläche aufgerufen werden. Die XML-Zeile dazu könnte so aussehen:

```
<button id="btnDiagUnter" label="Diagramm-Unterschrift" image ="IconDU"
onAction="BUDiag " />
```





Wichtig

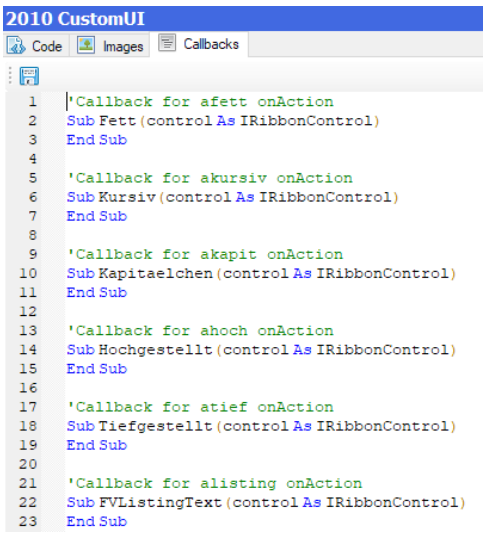
Um vom Menüband aus aufgerufen zu werden, muss der Makroname den Klammerzusatz **(control As IRibbonControl)** erhalten. Ein so deklariertes Makro ist weder über den Makro-Dialog noch mit F5 und F8 zu starten, es kann nur über das Menüband aufgerufen werden. Für schrittweise Testläufe können Stoppstellen im Code gesetzt werden.

Tipp

Verwenden Sie das vom Menüband aufgerufene Makro einfach durch einen bloßen Launcher für das eigentliche Makro, also:

```
001  Sub DUeinfg(control As IRibbonControl)
002  Call DUeinfugen
003  End Sub
004  Sub DUeinfugen()
005  ' Diagrammunterschrift
006      With CaptionLabels("Diagramm")
007          .IncludeChapterNumber = True
008          .ChapterStyleLevel = 1
009          .Separator = wdSeparatorPeriod
010      End With
011      With Selection
012          .Paragraphs(1).Range.Select
013          .Collapse Direction:=wdCollapseStart
014          .InsertCaption Label:="Diagramm", TitleAutoText:="", Title:="", _
015          Position:=wdCaptionPositionBelow, ExcludeLabel:=0
016          .Style = "Beschriftung"
017          .TypeText Text:=": "
018      End With
019  End Sub
```

Callbacks erzeugen



Damit Sie nicht vergessen, die bezogenen Makros auch anzulegen, können Sie mit der Schaltfläche
GENERATE VBA CALLBACKS
den Editor veranlassen, alle `onAction`-Aufrufe zusammenzutragen und in der Registerkarte CALLBACKS aufzulisten.

Quickinfos

Zu den Schaltflächen können QuickInfos eingeblendet werden, wenn der Mauszeiger darüber zur Ruhe kommt. Dazu dienen die Parameter `screentip="Fette Überschrift"` und `supertip="Normaler Text"`.

In QuickInfos können Sie alle Zeichen verwenden, aber im Editor lassen sich die meisten nicht direkt eingeben. Stattdessen ist die Nummer des Zeichens in `&#` und `;` eingeschlossen einzutragen, zum Beispiel `&#60;` für <. Ebenso sind die Zeichen, die für den XML-Code benötigt werden, zu ersetzen: `&#34;` für " und `&#38;` für &.

Die wichtigsten Controls

Wenn Sie die Listen der Controls durchsehen, werden Sie auf eine große Vielfalt stoßen, von denen nur einige für selbstgebaute Word-Menübänder benötigt werden. Übersichten und Erläuterungen zu *allen* Controls finden Sie auf der offiziellen Site von Microsoft, im RibbonX-Workshop von René Holtz und im RibbonX-Workshop von »matrix-edv«, wobei leider angemerkt werden muss, dass die Liste von Microsoft ist recht lieblos zusammengeschustert ist. Die anderen beiden sind aufschlussreicher.

Die folgende Auswahl der wichtigsten Controls, reicht für die meisten Umgestaltungen aus.

Wichtig

Achten Sie penibel auf die Groß-/Kleinschreibung!

Die unter »Syntax« angegebenen Parameter sind die Mindestausstattung, die Sie nach Belieben um `label`, `image`, `size` etc. ergänzen können.

onLoad

in der customUI-Zeile (erste Zeile überhaupt) sorgt dafür, dass das angeführte Makro beim Aufruf der mit diesem RibbonX-Code ausgestatteten Datei ausgeführt wird.

Syntax:

```
<customUI xmlns="http://schemas.microsoft.com/office/2009/07/customui"
onLoad="Startmakro">
```

dialogBoxLauncher fügt der Gruppe das Symbol ⇲ hinzu

Syntax:

```
<dialogBoxLauncher>
    <button id="bLaunch" onAction="Makroname"/>
</dialogBoxLauncher>
```

labelControl nutzt die erste Zeile in der Gruppe für eine zusätzliche Überschrift

Syntax: `<labelControl id="lbl1" label="Überschrift" />`

button häufigstes Control, ruft unmittelbar Funktionen auf, siehe oben

Syntax: `<button id="bTuwas" onAction="Makroname"/>`
oder: `<button idMso="Cut" />`

toggleButton schaltet zwischen zwei Zuständen hin und her.

Syntax: `<toggleButton id="tbUmschalt" getPressed="Makro1" onAction="Makro2"/>`
oder: `<toggleButton idMso="ParagraphMarks" />`

Die Aktivschaltung wird durch farbige Hinterlegung angezeigt; sie muss beim Makro-Aufruf mit der `onAction`-Funktion koordiniert sein:

- Der Zustand ist mit dem `getPressed` angegebenen Makro festzulegen, dazu muss das Makro eine Variable übergeben.

  ```
  Sub Makro1(control As IRibbonControl, ByRef returnedVal)
  ```

- Das mit `onAction` aufgerufene Makro muss eine Variable vom Typ Boolean übergeben.

  ```
  Sub Makro2(control As IRibbonControl, pressed As Boolean)
  ```

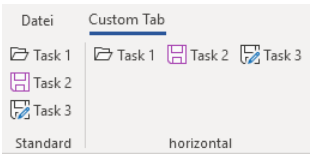
checkBox schaltet zwischen zwei Zuständen hin und her.

Syntax: `<checkBox id="cbCheck" getPressed="Makro1" onAction="Makro2"\>`
oder: `<checkBox idMso="ViewRulerWord" />`

Die Aktivschaltung wird durch das Ankreuzfeld angezeigt; sie muss beim Makro-Aufruf mit der `onAction`-Funktion koordiniert sein. Die Variablenübergabe ist dieselbe wie bei `toggleButton`.

box fasst mehrere Buttons zusammen. Damit kann vom spaltenweisen Aufbau jeweils dreier kleiner Buttons abgewichen werden

```
<box id="box1" boxStyle="horizontal">
  <button id="b1" label="Task 1" imageMso="FileOpen"/>
  <button id="b2" label="Task 2" imageMso="FileSave"/>
  <button id="b3" label="Task 3" imageMso="FileSaveAs"/>
</box>
```



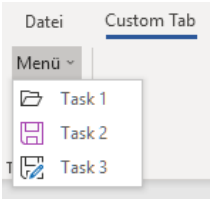
Mit `boxStyle="horizontal"` lässt sich Aufbaurichtung der eingeschlossenen Buttons ändern.

separator senkrechte Linie , um Buttongruppen innerhalb der Gruppe optisch zu trennen

Syntax: `<separator id="sep1" />`

menu klappt beim Anklicken eine Liste mit Controls aus

```
<menu id="men1" label="Menü">
  <button id="b1" label="Task 1" imageMso="FileOpen"/>
  <button id="b2" label="Task 2" imageMso="FileSave"/>
  <button id="b3" label="Task 3" imageMso="FileSaveAs"/>
</menu>
```

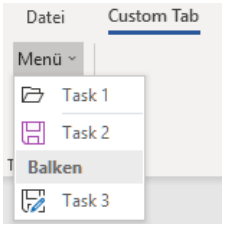


menuSeparator teilt mit einer Querlinie die Menüeinträge in Gruppen

Syntax: `<menuSeparator id="menSep1" />`

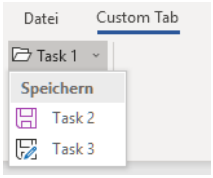
kann um den Parameter `title="Name"` ergänzt werden, dann zeigt das Menü einen Teilungsbalken mit dem Label als Zwischenüberschrift an.

```
<menu id="men1" label="Menü">
     <button id="b1" label="Task 1" imageMso="FileOpen"/>
  <menuSeparator id="menSep1" />
     <button id="b2" label="Task 2" imageMso="FileSave"/>
  <menuSeparator id="menSep2" title="Balken"/>
     <button id="b3" label="Task 3" imageMso="FileSaveAs"/>
</menu>
```



splitButton geteilte Schaltfläche, bei der der obere oder linke Teil direkt eine Funktion aufruft, während ⌄ ein Menü aufklappt.

```
<splitButton id="split1" >
      <button id="b1" label="Task 1" imageMso="FileOpen"/>
   <menu id="men1" >
      <menuSeparator id="sepSplit" title="Speichern" />
      <button id="b2" label="Task 2" imageMso="FileSave"/>
      <button id="b3" label="Task 3" imageMso="FileSaveAs"/>
   </menu>
</splitButton>
```



gallery ruft ein Mso-Control vom Typ `gallery` auf

Syntax: `<gallery idMso="BreaksGallery" />`

Standard-Ribbons verändern

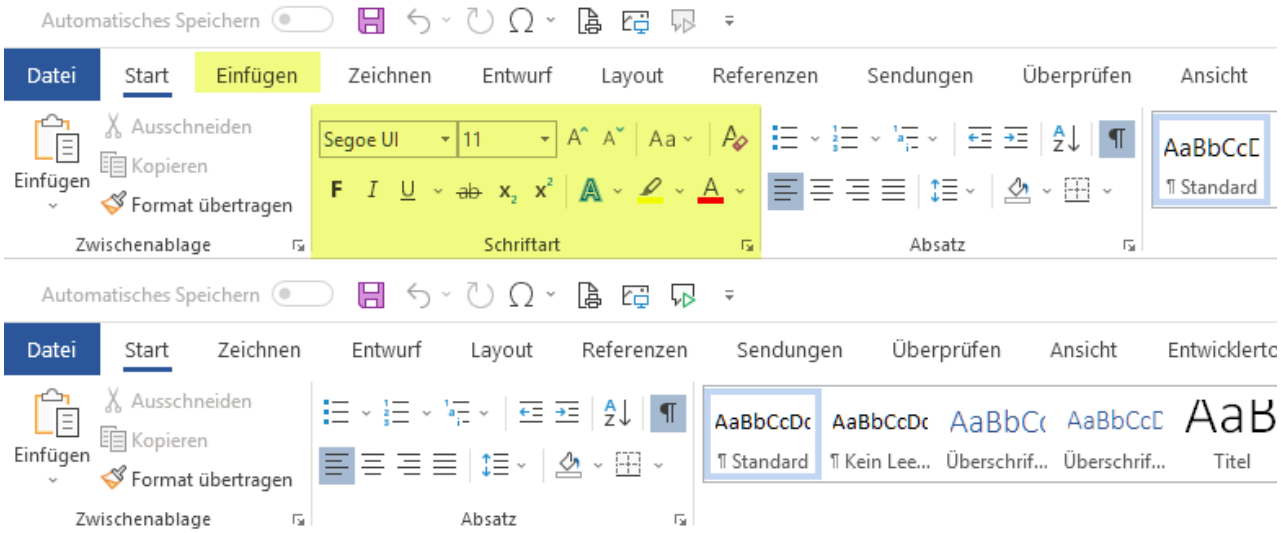
Die XML-Programmierung bietet Ihnen alle Möglichkeiten, in die Funktionen der Standard-Ribbons einzugreifen.

Ersetzen Sie die Zeile zur Eröffnung der Ribbon-Ebene `<ribbon >` durch
`<ribbon startFromScratch="true" >`,
erreichen Sie, dass nur Ihre eigenen Registerkarten im Menüband erscheinen.

Teile der Standard-Ribbons ausblenden

Nehmen wir an, Sie möchten die Gruppe SCHRIFTART in der Registerkarte START ausblenden:

```
<tab idMso="TabHome" >
   <group idMso="GroupFont" visible="false" />
</tab>
<tab idMso="TabInsert" visible="false" />
```



Mit dieser Methode lassen sich ganze Registerkarten und Gruppen ausblenden, aber keine Controls, wie bei MENÜBAND ANPASSEN.

Workaround

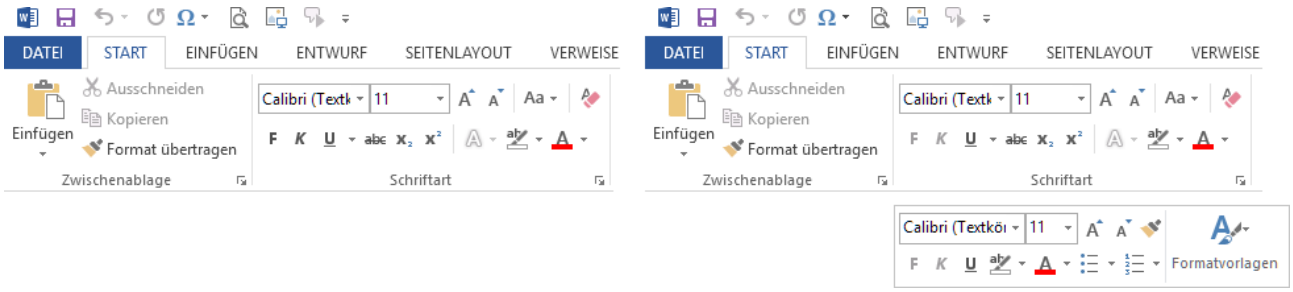
Blenden Sie eine komplette Gruppe aus und legen Sie eine eigene Gruppe an, die nur die gewünschten Mso-Controls enthält.

Schaltfläche »ausgrauen« und funktionsunfähig machen

Es ist aber möglich, die Funktion einer Schaltfläche zu sperren.

Wollen Sie die Schaltflächen für **fett** und *kursiv* in der Gruppe SCHRIFTART zwar erhalten, aber funktionslos schalten, funktioniert dies nicht durch Abschalten innerhalb der Gruppe, sondern mit Hilfe der `command`-Anweisungen. Dazu bedarf es eines gesonderten Abschnitt `<commands>` *noch vor* dem Abschnitt `<ribbon>` in der XML-Struktur. Davon sind dann aber auch *alle* Schaltflächen *derselben* Funktion an anderer Stelle betroffen, zum Beispiel in der Minisymbolleiste.

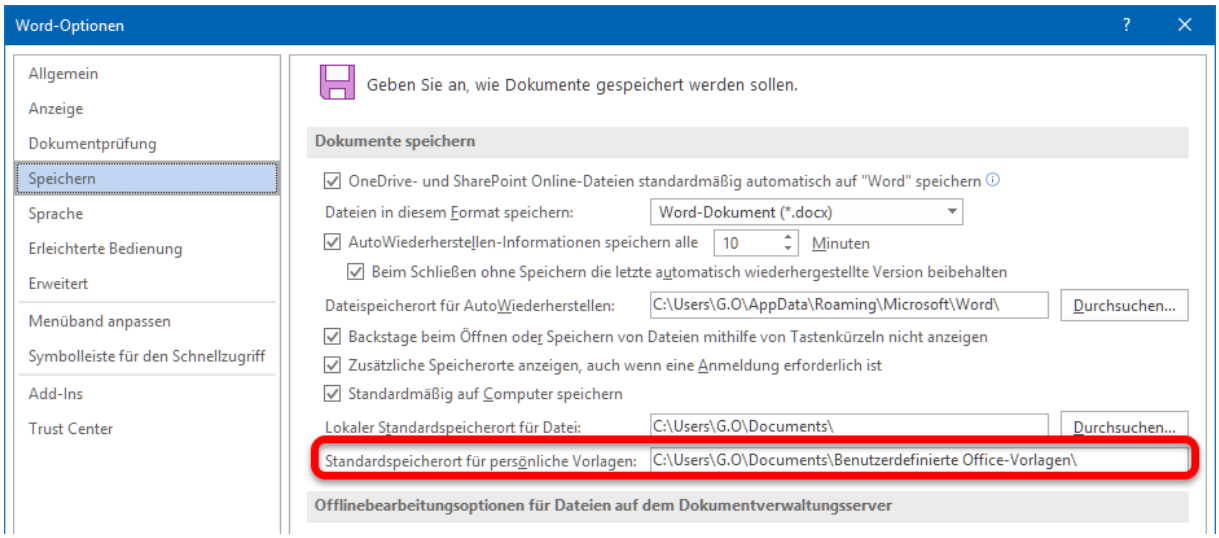
```
<customUI xmlns="http://schemas.microsoft.com/office/2009/07/customui">
<commands>
    <command idMso="Bold" enabled="false" />
    <command idMso="Italic" enabled="false" />
</commands>
<ribbon startFromScratch="false">
…
```



Wohin mit der .dotm?

Je nachdem, wie weitgehend Ihre Einflussnahmen auf das Menüband sein sollen, kommen folgenden Speicherorte in Betracht:

Für **komplette Vorlagen** ist der in DATEI | OPTION | SPEICHERN festgelegte **Standardspeicherort für persönliche Vorlagen** der passende Speicherort.



Ist die Vorlage dort gespeichert, wird Sie Ihnen in DATEI | NEU | PERSÖNLICH angeboten.

Soll die Vorlage die **Standard-Vorlage ergänzen** oder verändern, speichern Sie sie im **Startup-Ordner für Word**, den Sie am sichersten mit dem Makro-Editor ermitteln:

1. Starten Sie den Makro-Editor mit Alt+F11
2. Öffnen Sie mit ANSICHT | DIREKTFENSTER den Direkteingabebereich.
3. Tragen Sie dort ein `?Application.StartupPath`, gefolgt von ↵.
4. Der Pfad wird angezeigt.

Im Startup-Ordner gespeicherte Vorlagen werden beim Start von Word automatisch mitgeladen und der XML-Code für das Menüband ausgeführt.

Ausführliche Quellen

Liste Controls

https://www.microsoft.com/en-us/download/confirmation.aspx?id=50745

Liste ImageMso

https://bert-toolkit.com/imagemso-list.html

Beschreibungen der Controls

https://msdn.microsoft.com/en-us/library/dd926139(v=office.12).aspx

https://www.rholtz-office.de/ribbonx/start

http://matrix-edv.de/rx/ws/19.html