

MS ACCESS
custom dialogs & notifications V2
SIMPLE. ELEGANT. EFFECTIVE.

Let's make ACCESS sexy again!

Let's make ACCESS sexy again!



Modern UI in Access: Custom Dialogs & Mini Notifications – now even smarter, faster & more flexible!

Building a modern database with the classic MsgBox & InputBox is like a new Porsche – but with Beetle rims. It works, but looks outdated.

Time for the next step: Custom Dialogs & mini Notifications!

Why Custom Dialogs?

Imagine this: you've been working on an Access database for months. The logic is perfect, the performance is right, the security is top-notch. Then you show it to your customer – and the first thing they see is a **grey MsgBox** window from the 90s.

Here, I'll show you how to go **from grey to great with little effort**. If you want to impress users with an Access application today, you need more than standard dialogs. The new version of my **Custom Dialogs** is a complete **refactor** – even faster, even easier, even more powerful.

Comparison - Features

	Standard Dialogs	Custom Dialogs	Notifications
Formatted text	NO	YES	---
Images	NO	YES	---
Tables	NO	YES	---
Web content	NO	YES	---
Printable	NO	YES	---
Don't show again	NO	YES	---
Copy content	NO	YES	---
Delayed buttons	NO	YES	---
Colored header	NO	YES	---
Templates	NO	YES	---
Data selection	NO	YES	---
Auto close	NO	YES	YES
Custom Icons	NO	YES	YES
Build in	YES	NO	NO
Easy to use	YES	YES	YES
IntelliSense	YES	YES	YES

The stars of the show:

1. **myMsgBox** – the modern alternative to MsgBox
2. **myInputDialog** – more attractive than the original InputBox
3. **myComboBox** – like an InputBox but with data selection
4. **myListBox** – like myComboBox but as a list box (multiple selections possible).
5. **myErrorBox** – a dialog box for displaying error messages.
6. **myNotification** – discreet, brief feedback to the user.

MyComboBox and **myListbox** can be filled with values either as a values list or via SQL query. Multi-column entries are also possible.

💡 Not all feedback requires a dialog box:

myNotification provides discreet, **brief feedback**—displayed as a semi-transparent overlay across the entire screen, with a centrally placed icon. The notification stands out visually and closes automatically after a short time. This gives users immediate feedback without interrupting their workflow.

🎨 Two integrated editors – full control

✦ Dialog-Type Editor

With the Dialog Type Editor, you can customize the icon, set a header color, and even determine the color gradient of the header. This gives the dialogs a much more modern look and fits perfectly into the design of your application.

✦ Template Editor

With the Template Editor, you can create or edit complete dialog templates – **quickly, flexibly, and conveniently**. There you can:

- ✓ select the **dialog type** (myMsgBox, myInputBox, myComboBox, or myListBox),
- ✓ enter the **header & body text** (plain text or HTML),
- ✓ define the **button captions**,
- ✓ add a “**Do not show again**” checkbox,
- ✓ activate a **print button**,
- ✓ activate a **copy content button**,
- ✓ determine the **auto-close** time and the **delayed display of the buttons**,
- ✓ store **default values** for input, list, or combo boxes,
- ✓ enter **values for list boxes and combo boxes** directly. As a value list or as an SQL string

Particularly practical: an **online HTML editor** can be opened directly from the editor. With this **WYSIWYG** tool, simple plain text can be turned into cool HTML in no time at all, which you can then transfer directly to the template editor.

IntelliSense

With **modHelper_Enumeration**, IntelliSense support is finally here!

It was important to me that calling dialog boxes and notifications be **as easy as possible**—and that requires IntelliSense. That's why all dialog types (for both notifications and dialog boxes) are controlled via enumerations.

To **eliminate the effort of manually entering these enums**, I developed the **modHelper_Enumeration** module. New enum entries can be **dynamically created, edited, or deleted at runtime**—directly via the Type Editor. A real highlight: a module that can read, check, edit, or delete values from a module.

Two additional classes come into play:

clsDLG_Config – manages the complete configuration setup and the values of the dialog boxes.

clsDLG_Result – provides the returns, i.e., which button was clicked or which values were entered/selected in the ListBox, ComboBox, or InputBox.

This creates a structure that is both highly flexible and intuitive – with full IntelliSense power.

Some examples?

DialogBoxes:

As command:

```
myMsgBox "Header", "This is the body text", DLG_Info, "OK", "Cancel"
Debug.Print myComboBox("QUESTION", "Please choose a value.", DLG_Question, "SELECT dlg_type FROM tblDLG_Dialogs_types", "OK")
Debug.Print myInputBox("QUESTION", "Please enter a value.", DLG_Question, "Default value", "OK", "Cancel")
Debug.Print myListBox("QUESTION", "Please choose a value.", DLG_Mail, "1;2;3;4;3;5", "OK")
```

MessageBox with more options:

```
With DLG_Config
    .DialogType = DLG_Info
    .header = "HEADER Text"
    .Body = "HTML string"
    .Button1 = "OK"
    .Button2 = "Cancel"
    .Button3 = "Another button"
    .Printing = True
    .AutoClose = 20
    .ButtonDelay = 4
    .DialogWidth = 10000
    .DialogHeight = 5000
```

```
.Start_as_MsgBox  
End With
```

Listbox with more options (ComboBox is similar):

```
With DLG_Config  
.DialogType = DLG_Question  
.header = "HEADER Text"  
.Body = "HTML string"  
.Button1 = "OK"  
.ComboValue = "SELECT dlg_type, dlg_typeColor FROM tblDLG_Dialogs_types ORDER BY dlg_type"  
.ComboWidths = "5cm;0cm"  
.DefaultValue = "DLG_Info"  
.Start_as_ListBox  
End With
```

Inputbox using a template:

```
With DLG_Config  
.TemplateName = "delete"  
.TemplateValue1 = "Example text 1"  
.TemplateValue2 = "Example text 2"  
.DefaultValue = "Text xxx"  
.Start_as_InputBox  
End With
```

Notification:

```
myNotification DLG_Cancel, Black, 1
```

**Replacing build in standard dialogs**

I had the privilege of giving a presentation at the **Access User Group Europe**. During the session, someone asked whether it would be possible to create a function that could automatically replace the built-in standard dialogs — MsgBox and InputBox. **You asked for it, and here it is!**

My new version now supports exactly that. The function scans the entire VBE, detects all MsgBox and InputBox calls, displays them, and lets the user decide whether they should be replaced.

Playground: A preview of what else is possible 🚀

In addition to the actual dialogs, I have also included a playground. Here, I will briefly show you what else is possible with the new functions:

- ✓ Manipulate values in existing HTML code (e.g., for theme or language)
- ✓ Read out selected values again

In doing so, I also came across a small peculiarity:

The web browser control in Access currently only allows content to be read out if it has been loaded using the navigate function. However, if I write the innerHTML directly (as I do in the templates – because it's faster), the values cannot be read out.

👉 My workaround in the Playground: I create a temporary HTML file, load it, and then delete it again. It works – but the loading time is noticeably longer.

But there is a solution for a very quick load of the HTML file. It seems the web browser control does a DNS call before loading the local file. Sound stupid, but it is. So it takes a long time until the DNS server respond back, that he has no glue where your local file is located. But before Windows does the the DNS call, it looks always into the local hosts-file to find an entry for. If we add '127.0.0.1 msaccess' into this file, the HTML file is loaded immediately.


WITH THIS FIX I CONSIDER THE PLAYGROUND NOT ANYMORE AS A TEST ENVIRONMENT!!!

You will find the file typically at `C:\Windows\System32\drivers\etc\hosts`

My new version has now a build in feature to check if the entry exists. If not It will done.

Just give it a try

You can try out the dialogs and notifications directly using my testing form. This makes it easy to get to know them, test them, and incorporate them into your own application. Your users will thank you for it.

Are you developing with Access? **Then give it a try**—I look forward to your feedback or ideas for additional features! 

Let's take ACCESS to the next level together. Because one thing is clear:

Standard was yesterday. Today, ACCESS is sexy.

MY DATABASE - DESIGN - TEMPLATE

PLAYGROUND

Theme: light
Language: German
Time intervals (min): 15

Calendar view showing dates from September 1 to 30, 2020. The date 17 is highlighted.

PRINT LBO

Printing the report
Number of copies: 1
The report will be printed on your default printer.
How many copies do you want to print?

PASSWORD

Password incorrect
You are not authorized to access this database!
The password you entered does not match that of the user `passwordWrong`.
You have 2 attempts left before the database is closed!

MS ACCESS
custom dialogs & notifications V2
SIMPLE. ELEGANT. EFFECTIVE.
Let's make ACCESS sexy again!

DIETERLE
programmierung

Let's make ACCESS sexy again!

 **Moderne UI in Access: Custom Dialogs & Mini Notifications – jetzt noch smarter, schneller & flexibler!**

Eine moderne Datenbank mit den klassischen MsgBox & InputBox zu bauen, ist wie ein **neuer Porsche – aber mit Käfer-Felgen**. Es funktioniert, aber wirkt nicht mehr zeitgemäß.

Zeit für den nächsten Schritt: Custom Dialogs & Mini Notifications!

Warum Custom Dialogs?

Stell dir Folgendes vor: Du arbeitest seit Monaten an einer Access-Datenbank. Die Logik ist perfekt, die Leistung stimmt, die Sicherheit ist erstklassig. Dann zeigst du sie deinem Kunden – und das Erste, was dieser sieht, ist ein **graues MsgBox-Fenster aus den 90er Jahren**.

Ich zeige dir hier, wie du **mit wenig Aufwand von grau zu großartig** kommst.

Wer heute Nutzer mit einer Access-Anwendung begeistern will, braucht mehr als Standarddialoge. Die neue Version meiner **Custom Dialogs** ist ein kompletter **Refactor** – noch schneller, noch einfacher, noch mächtiger.

Gegenüberstellung - Features

	Standard Dialogs	Custom Dialogs	Notifications
Formatierter Text	NO	YES	---
Bilder	NO	YES	---
Tabellen	NO	YES	---
Web Inhalte	NO	YES	---
Druckbar	NO	YES	---
Don't show again	NO	YES	---
Inhalt kopieren	NO	YES	---
Verzögerte Buttons	NO	YES	---
Farbige Kopfzeilen	NO	YES	---
Vorlagen	NO	YES	---
Daten Auswahl	NO	YES	---
Automatisch Schließen	NO	YES	YES
Eigene Icons	NO	YES	YES
Build in	YES	NO	NO
Easy to use	YES	YES	YES
IntelliSense	YES	YES	YES

Die Stars der Show:

1. **myMsgBox** – die moderne alternative zur MsgBox
2. **myInputDialog** – schöner als die original InputBox
3. **myComboBox** – wie eine InputBox jedoch mit Datenauswahl
4. **myListBox** – wie die myComboBox jedoch als ListBox (Mehrfachauswahl möglich).
5. **myErrorBox** – eine DialogBox um Fehlermeldungen anzuzeigen.
6. **myNotification** – dezentes kurzes Feedback an den User.

MyComboBox und **myListBox** können jeweils als Werteliste oder aber per SQL Abfrage mit Werten gefüllt werden. Auch Mehrspaltige Einträge sind möglich.

💡 Nicht jedes Feedback benötigt eine DialogBox:

myNotification liefert dezentes, kurzes **Feedback** – vollflächig halbtransparent über den Bildschirm gelegt, mit einem zentral platzierten Icon. Die Benachrichtigung hebt sich optisch ab und schließt automatisch nach kurzer Zeit. So erhalten Nutzer sofortige Rückmeldung, ohne aus ihrem Workflow gerissen zu werden.

🎨 Zwei integrierte Editoren – volle Kontrolle

✦ Dialog-Typen Editor

Mit dem Dialog-Typen-Editor kannst du das Icon anpassen, eine Header-Farbe festlegen und sogar den Farbverlauf des Headers bestimmen. Dadurch wirken die Dialoge deutlich moderner und fügen sich perfekt ins Design deiner Anwendung ein.

✦ Template Editor

Mit dem Template-Editor erstellst oder bearbeitest du komplette Dialog-Vorlagen – **schnell, flexibel und komfortabel**.

Dort kannst du:

- ✓ den **Dialog-Typ** wählen (myMsgBox, myInputBox, myComboBox oder myListBox),
- ✓ **Überschrift & BodyText** (Plaintext oder HTML) eintragen,
- ✓ die **Button-Beschriftungen** festlegen,
- ✓ eine „**Do not show again**“-Checkbox hinzufügen,
- ✓ einen **Druck-Button** aktivieren,
- ✓ einen **Kopiere Inhalt-Button** aktivieren,
- ✓ die **Auto-Close-Zeit** sowie die **verzögerte Anzeige der Buttons** bestimmen,
- ✓ **Standardwerte** für Input-, List- oder ComboBox hinterlegen,
- ✓ **Werte für ListBox und ComboBox** direkt einpflegen. Als Werteliste oder als SQL String

Besonders praktisch: Direkt aus dem Editor heraus lässt sich ein **Online-HTML-Editor** öffnen. Mit diesem **WYSIWYG-Tool** wird aus einfachem Plaintext im Handumdrehen cooles HTML, das du anschließend direkt in den Template-Editor übernimmst.

IntelliSense

Mit **modHelper_Enumeration** gibt's endlich IntelliSense-Support!

Mir war wichtig, dass die Aufrufe von Dialogboxen und Notifications **so einfach wie möglich sind** – und dafür braucht es **IntelliSense**. Deshalb werden alle Dialogtypen (sowohl für Notifications als auch für Dialogboxen) über Enumerationen gesteuert.

Damit man diese **Enums nicht mehr mühsam von Hand eintragen muss**, habe ich das Modul **modHelper_Enumeration** entwickelt. Neue Enum-Einträge lassen sich **dynamisch zur Laufzeit erstellen, bearbeiten oder löschen** – direkt über einen komfortablen Editor. Ein echtes Highlight: Ein Modul, das Werte aus einem Modul auslesen, prüfen, bearbeiten oder löschen kann.

Zusätzlich kommen zwei Klassen ins Spiel:

clsDLG_Config – verwaltet das komplette Konfigurations-Setup und die Werte der Dialogboxen.

clsDLG_Result – liefert die Rückgaben, also welcher Button geklickt wurde oder welche Werte in ListBox, ComboBox oder InputBox eingetragen/ausgewählt wurden.

So entsteht eine maximal flexible, aber gleichzeitig intuitive Struktur – mit voller IntelliSense-Power.

Ein paar Beispiele?

Als Command:

```
myMsgBox "Header", "This is the body text", DLG_Info, "OK", "Cancel"
Debug.Print myComboBox("QUESTION", "Please choose a value.", DLG_Question, "SELECT dlg_type FROM tblDLG_Dialogs_types", "OK")
Debug.Print myInputBox("QUESTION", "Please enter a value.", DLG_Question, "Default value", "OK", "Cancel")
Debug.Print myListBox("QUESTION", "Please choose a value.", DLG_Mail, "1;2;3;4;3;5", "OK")
```

MessageBox mit mehr Optionen:

```
With DLG_Config
    .DialogType = DLG_Info
    .header = "HEADER Text"
    .Body = "HTML string"
    .Button1 = "OK"
    .Button2 = "Cancel"
    .Button3 = "Another button"
    .Printing = True
```



```
.AutoClose = 20
.ButtonDelay = 4
.DialogWidth = 10000
.DialogHeight = 5000
.Start_as_MsgBox
End With
```

Listbox mit mehr Optionen (ComboBox ist gleich):

```
With DLG_Config
.DialogType = DLG_Question
.header = "HEADER Text"
.Body = "HTML string"
.Button1 = "OK"
.ComboValue = "SELECT dlg_type, dlg_typeColor FROM tblDLG_Dialogs_types ORDER BY dlg_type"
.ComboWidths = "5cm;0cm"
.DefaultValue = "DLG_Info"
.Start_as_ListBox
End With
```

Inputbox mit Verwendung von einem Template:

```
With DLG_Config
.TemplateName = "delete"
.TemplateValue1 = "Example text 1"
.TemplateValue2 = "Example text 2"
.DefaultValue = "Text xxx"
.Start_as_InputBox
End With
```

Notification:

```
myNotification DLG_Cancel, Black, 1
```

💡 Ersetzen von integrierten Standarddialogen

Ich hatte die Ehre, einen Vortrag bei der **Access User Group Europe** zu halten. Während der Sitzung fragte jemand, ob es möglich wäre, eine Funktion zu erstellen, die die integrierten Standarddialoge – MsgBox und InputBox – automatisch ersetzen könnte. **Ihr habt danach gefragt, und hier ist sie!**

Meine neue Version unterstützt genau das. Die Funktion durchsucht die gesamte VBE, erkennt alle MsgBox- und InputBox-Aufrufe, zeigt sie an und lässt den Benutzer entscheiden, ob sie ersetzt werden sollen.

🚀 Playground: Eine Vorschau was alles noch möglich ist

Neben den eigentlichen Dialogen habe ich auch einen Playground mitgeliefert. Hier zeige ich kurz, was mit den neuen Funktionen noch alles möglich ist:

- ✅ Werte in bestehendem HTML-Code (z. B. für Theme oder Sprache) manipulieren
- ✅ gewählte Werte wieder auslesen

Dabei bin ich auch auf eine kleine Besonderheit gestoßen:

Das Webbrowser-Control in Access erlaubt aktuell nur das Auslesen von Inhalten, wenn diese per Navigate-Funktion geladen wurden. Schreibe ich hingegen den innerHTML direkt (so wie ich es in den Templates mache – weil es schneller ist), lassen sich die Werte nicht auslesen.

👉 Mein Workaround im Playground: Ich erstelle eine temporäre HTML-Datei, lade sie ein und lösche sie danach wieder. Funktioniert – aber die Ladezeit ist spürbar länger.

Es gibt jedoch eine Lösung für das sehr schnelle Laden der HTML-Datei. Es scheint, dass die Webbrowser-Steuerung vor dem Laden der lokalen Datei einen DNS-Aufruf durchführt. Das klingt dumm, ist es aber auch. Es dauert also lange, bis der DNS-Server antwortet, dass er keine Ahnung hat, wo sich Ihre lokale Datei befindet. Bevor Windows jedoch den DNS-

Aufruf durchführt, sucht es immer in der lokalen Hosts-Datei nach einem Eintrag. Wenn wir „127.0.0.1 msaccess“ in diese Datei einfügen, wird die HTML-Datei sofort geladen.


MIT DIESER KORREKTUR BETRACHTETE ICH DEN PLAYGROUND NICHT MEHR ALS TESTUMGEBUNG!!!

Sie finden die Datei in der Regel unter `C:\Windows\System32\drivers\etc\hosts`

Meine neue Version verfügt nun über eine integrierte Funktion, um zu überprüfen, ob der Eintrag vorhanden ist. Ist dies nicht der Fall, wird er erstellt.

Einfach ausprobieren

Anhand meiner Testing-Form kannst du die Dialoge und Notifications direkt ausprobieren. Das macht es einfach, sie kennenzulernen, zu testen.– und sie in deine eigene Anwendung zu übernehmen. Deine Nutzer werden es dir danken.

Du entwickelst mit Access? **Dann probier's aus** – ich freu mich auf dein Feedback oder Ideen für weitere Funktionen! 

Lasst uns ACCESS gemeinsam auf die nächste Stufe bringen. Denn eines ist klar:

Standard war gestern. Heute ist ACCESS sexy. 