

**10. Access-Entwickler-Konferenz
Nürnberg 6./7.10.2007 und 20./21.10.2007**

Verweise

Grundlagen – Probleme – Lösungen

Thomas Möller, www.Team-Moeller.de

Vorstellung

- Thomas Möller
- dipl. Sparkassenbetriebswirt
- Arbeit mit Access seit 1997
- Seit 2000 hauptberuflich als Entwickler tätig
- **Schwerpunkte**
 - Access mit DB/2 als BackEnd
 - Beratungs-Tools
 - Reporting-Tools
- Nebenberuflich: Team-Moeller.de
 - Add-Ins
- Seit 1.1.2007 MVP für Access



Fahrplan

1. Grundlagen

Definition, Nutzen, Historie, Verweisdialog

2. Manipulation mit VBA

Objektmodell, Verweise auflisten, hinzufügen, löschen

3. Defekte Verweise

Disambiguation, Verweise auflösen, Lösungen manuell / per VBA, VerweisChecker

4. Late- / Early-Binding

Early-Bindung, Late-Binding, Verweisprobleme, Strategien

5. ActiveX

Definition, DLL Hell, Ersatzlösungen, Version Checker

6. Bibliotheks-DBs

Zweck, Zirkelbezüge, mdb und mde, RefLibPaths, VSS



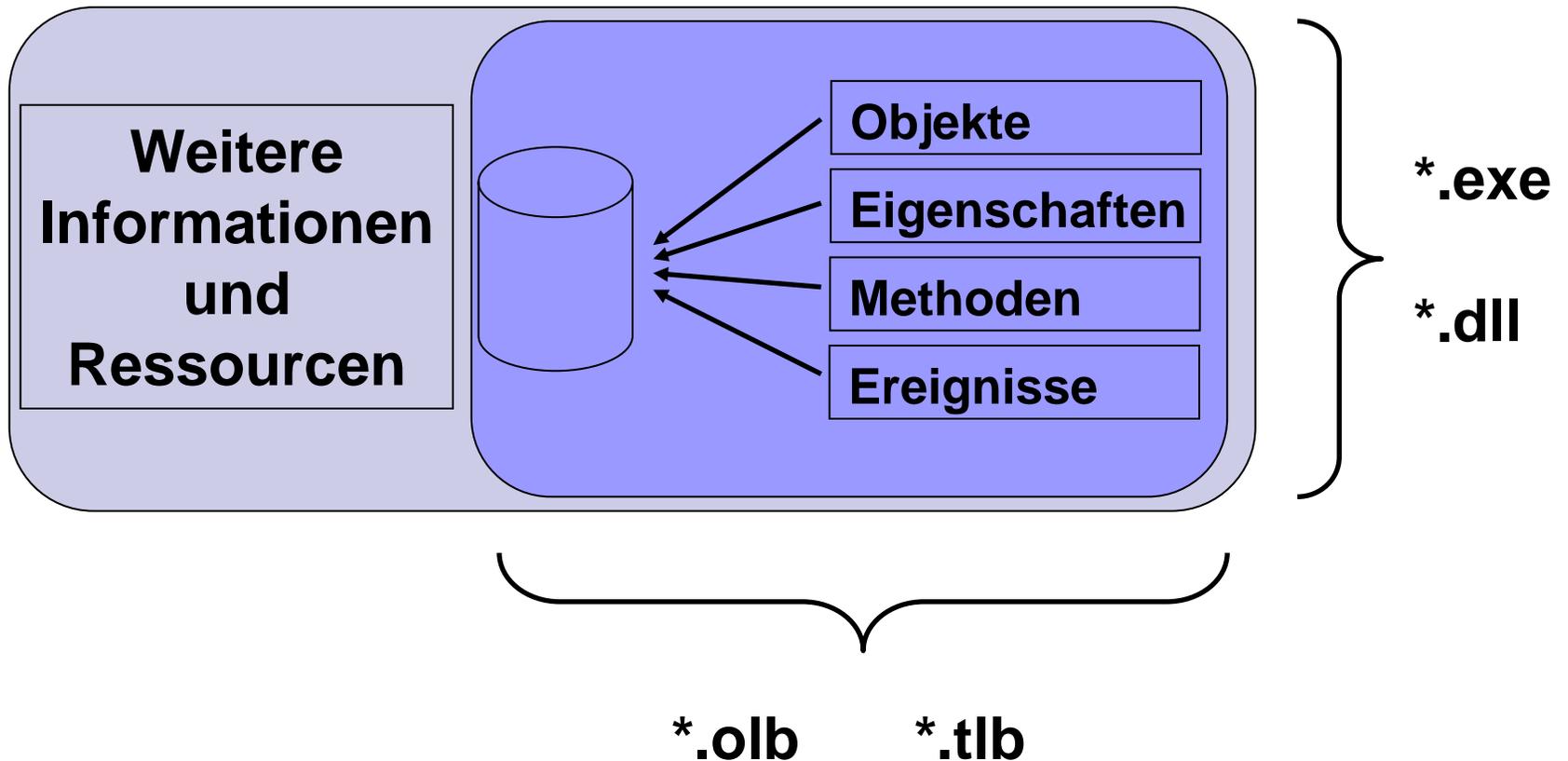
1. Grundlagen

- Was ist ein Verweis?
- Was bringen Verweise?
- Historische Entwicklung
- Welche Arten gibt es?
- Der Verweisdialog

Was ist eine Reference / ein Verweis?

- Das **Reference**-Objekt stellt einen Verweis auf die Typbibliothek einer anderen Anwendung oder eines anderen Projekts dar.
- Eine Typbibliothek ist eine Datei (oder eine Komponente einer Datei), die Standardbeschreibungen zur Automatisierung offen gelegter Objekte, Eigenschaften und Methoden enthält.
- Objektbibliotheksdateien (OLB) enthalten Typbibliotheken (Type Libraries oder TLBs), die als eigenständige Dateien bereitgestellt werden.

TypeLibrary (TLB)



Was bringen Verweise?

- Verweise machen externe Bibliotheken für VBA verfügbar.
- Der VBA-Editor kann auf Objekte, Eigenschaften und Methoden zugreifen.
- Der VBA-Editor kann einen Syntax-Check für Automationscode durchführen.
- Kontextsensitive Hilfe für die Schlüsselwörter fremder Objekte kann aufgerufen werden.
- Die Komponente kann mit dem Object-Browser (F2) untersucht werden.

Historische Entwicklung

- Acc 2.0 Access Basic; keine Verweise
- Acc 95 Erste Version mit VBA. Verweise können nur zur Entwurfszeit angepasst werden.
- Acc 97 Verweise können per VBA angepasst werden.
- Acc 2002 Application-Object erhält Eigenschaft BrokenReference.
- Acc 2003 Meldet fehlende Verweise direkt beim Öffnen der Datenbank.
- Acc 2007 Sucht (beim erstmaligen Start) selbst nach fehlenden Verweisen.

Welche Dateitypen sind möglich?

Im Verweis-Dialog können folgende Dateitypen als Verweisziel ausgewählt werden:

- *.exe, *.dll Programmdateien
- *.tlb, *.olb Type-, Objektlibrary
- *.ocx ActiveX-Komponente
- *.mdb, *.mde Access-Datenbanken
 *.adp, *.ade, bzw. Access-Projekte
 *.mda, *.accdb,
 *.accde

Verweisdialog öffnen

- Wechsel in den VBA-Editor (ALT + F11)
- Extras / Verweise
- Verweisdialog öffnet sich

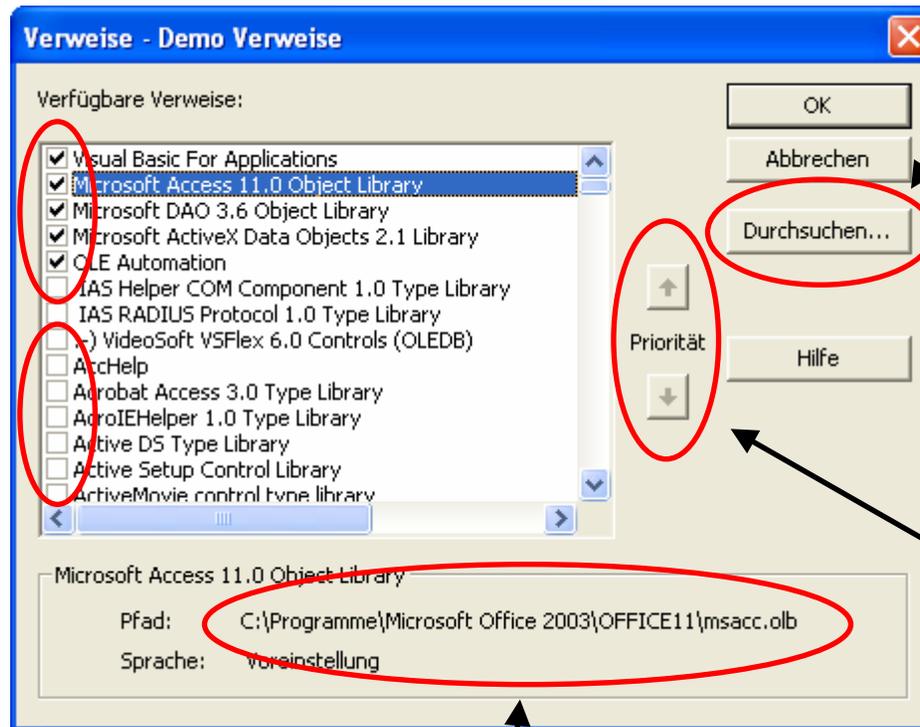


- Wenn Menüpunkt deaktiviert ist:
 - Codeausführung stoppen (Ausführen / Zurücksetzen)
- Wenn der Dialog nicht geöffnet wird
 - Eventuell Konflikt mit Installation von Crystal Report 8
<http://support.microsoft.com/kb/269383/en-us>

Verweisdialog: Funktionsweise

Ausgewählte Verweise

Verweis aus Datei einfügen



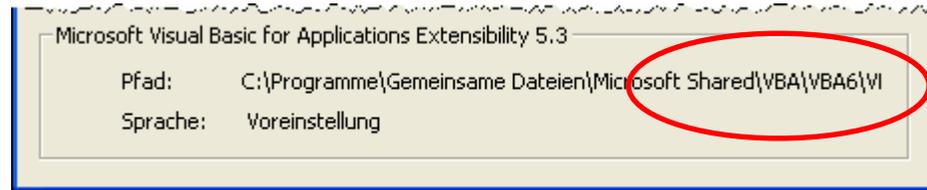
Verfügbare Verweise

Pfad zur TLB-Datei

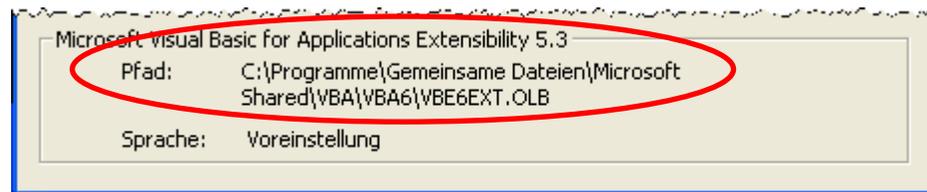
Priorität ändern

Verweisdialog verbessern

- Problem: Platz für Pfad nicht ausreichend



- Lösung: VBE Tools v2.0
(AddIn von Office Automation Ltd.)
<http://www.oaltd.co.uk/VBETools/Default.htm>
Größe für Pfad kann optional erweitert werden

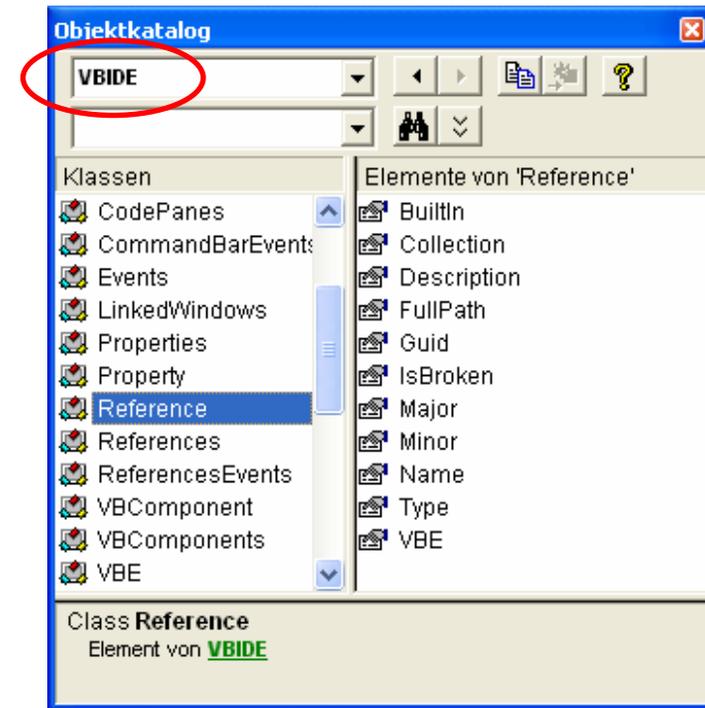
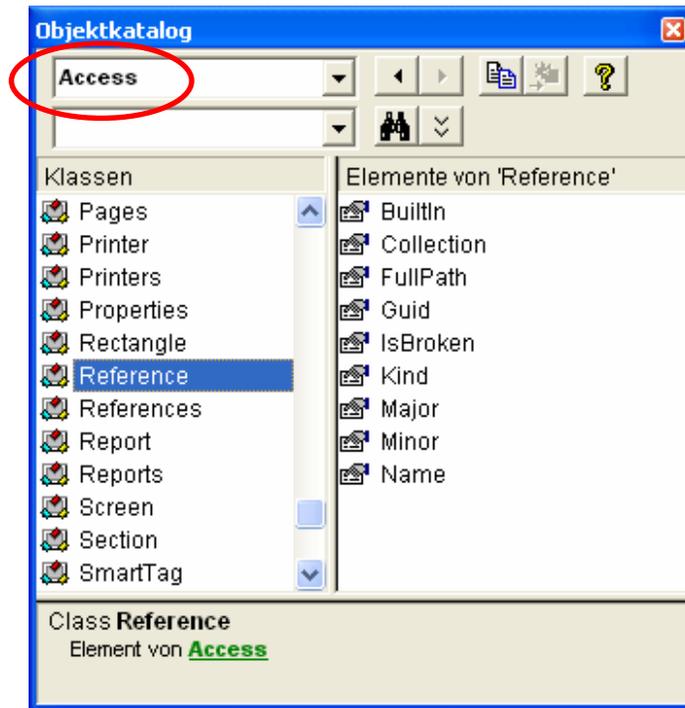


2. Manipulation mit VBA

- Eigenschaften / Methoden / Ereignisse
- Verweise auflisten
- Verweis vorhanden?
- Verweis hinzufügen
 - Add From String
 - Add From GUID
- Verweis löschen
- Eigenschaft BrokenReference

Verwendete Bibliotheken

- Zwei Bibliotheken mit Objekten zu Verweisen:



- Eigenschaften und Methoden nahezu identisch
- Verwendung von „Access“ spart einen Verweis!

References-Auflistung

■ Eigenschaften

Count Anzahl der aktivierten Verweise

■ Methoden

AddFromFile Hinzufügen aus Datei

AddFromGUID Hinzufügen mit GUID

Remove Verweis entfernen

■ Ereignisse

ItemAdded Verweis wurde hinzugefügt

ItemRemoved Verweis wurde entfernt

Reference-Objekt

■ Eigenschaften

- BuiltIn Standardverweis, Verweis eingebaut
- Fullpath Vollständiger Pfad und Dateiname
- GUID GUID (aus Registry)
- IsBroken Gibt an, ob Verweis ungültig ist
- Kind Art des Verweises (TLB, VB-Projekt)
- Major Hauptversionsnummer
- Minor Nebenteil der Versionsnummer
- Name Name des Verweises

- Description Beschreibung aus Verweis-Dialog
nur in „VB-IDE“ verfügbar

Verweise auflisten

- Variable vom Typ Reference deklarieren
- Schleifendurchlauf mit For-Each:

```
'Variablen deklarieren  
Dim ref As Access.Reference  
  
For Each ref In Application.References  
    Me.lstVerweise.AddItem ref.Name  
Next
```

C
O
D
E

Verweis vorhanden?

```
'Variablen deklarieren
Dim ref As Access.Reference
Dim strVerweis As String

'Initialisieren
strVerweis = Me!txtVerweis

For Each ref In Application.References
    If ref.Name = strVerweis Then
        MsgBox "Verweis ist vorhanden"
        Exit Sub
    End If
Next
```

C
O
D
E

Verweis hinzufügen (I)

```
'Variablen deklarieren  
Dim strDateiPfad As String
```

```
'Initialisieren  
strDateiPfad = Me!txtDateiPfad
```

```
Application.References.AddFromFile _  
    strDateiPfad
```

C
O
D
E

 Fehlermeldung, wenn Verweis schon vorhanden

Verweis hinzufügen (II)

```
'Variablen deklarieren  
Dim strGUID As String
```

```
'Initialisieren  
strGUID = Me!txtGUID
```

```
Application.References.AddFromGuid _  
    strGUID, 0, 0
```

C
O
D
E

 Fehlermeldung, wenn Verweis schon vorhanden

Verweis hinzufügen (III)

- Wie den GUID ermitteln?
 - Gewünschten Verweis einfügen und die Eigenschaft GUID auslesen
- Angabe der Versions-Nr.
 - Ermöglicht Verweis auf beliebige Version
 - Übergabe von „0, 0“ sorgt dafür, dass neuester Verweis verwendet wird

Verweis löschen

'Variablen deklarieren

```
Dim ref As Access.Reference
```

```
Dim strVerweis As String
```

'Initialisieren

```
strVerweis = Me!txtVerweis
```

```
Set ref = Application.References(strVerweis)
```

```
Application.References.Remove ref
```

C
O
D
E



Fehlermeldung, wenn Verweis nicht vorhanden

BrokenReference

- Eigenschaft des Application-Objekts
- Ist seit Access 2002 vorhanden
- Gibt an, ob in der aktuellen Datenbank getrennte Verweise vorhanden sind
- Ermöglicht durch die Abfrage einer einzigen Eigenschaft zu prüfen, ob in der Datenbank Probleme mit Verweisen vorliegen:

```
If Application.BrokenReference = False Then  
    MsgBox "Alle Verweise sind ok."  
End If
```

3. Defekte Verweise

- „Disambiguation“
- Wie löst Access Verweise auf?
- Manuelle Lösungen
- Lösungen per VBA
- Ideen von Michael Kaplan
- VerweisChecker

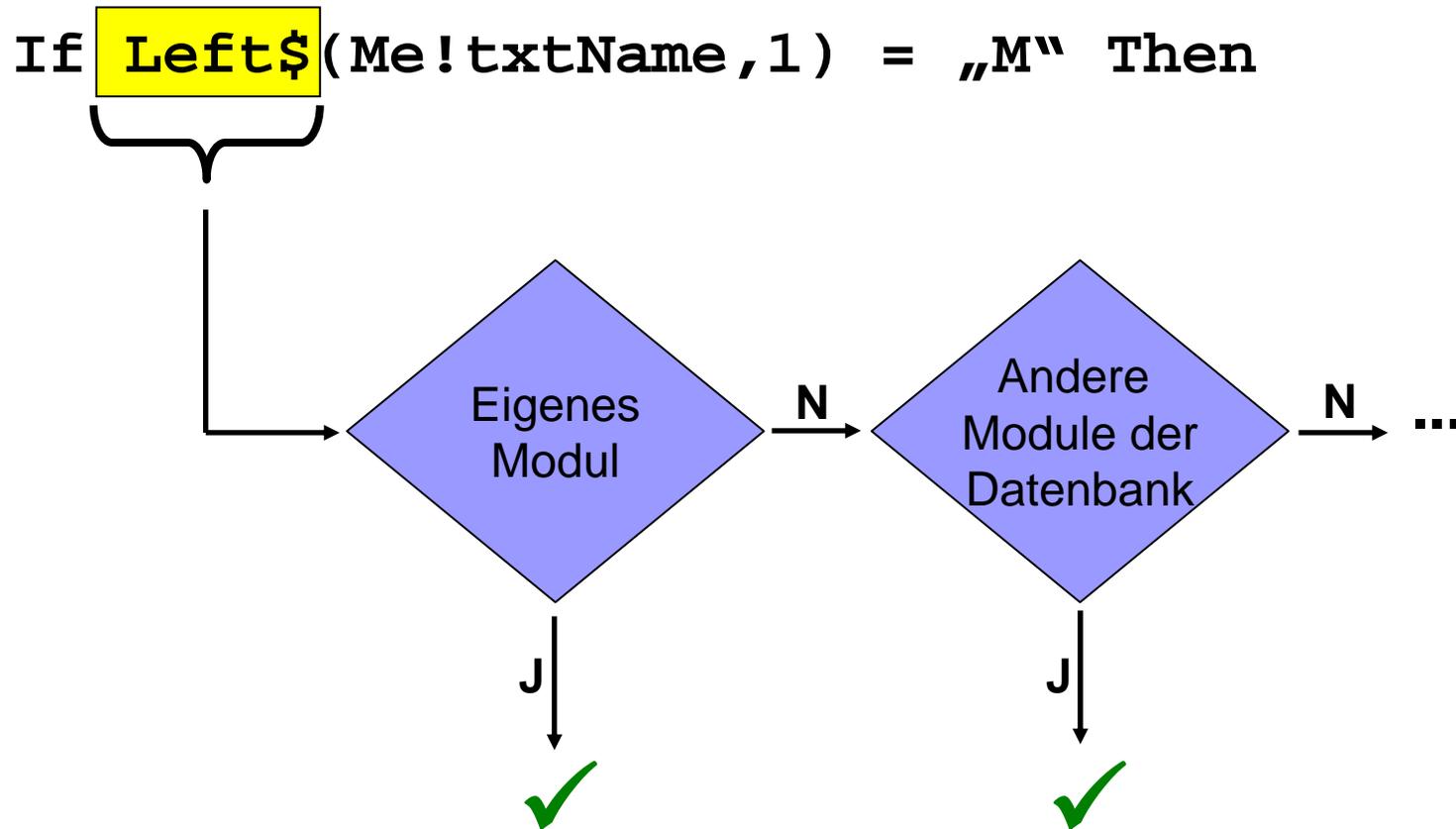
„Disambiguation“

- Prozess, die passende Bibliothek zu finden:
- Access trifft auf Objekt / Methode
- Access sucht passende Bibliothek
 - Objekte der Datenbank
 - Durchsuchen der Verweise
 - Suche in Reihenfolge der Priorität der Verweise
 - VBA und Access werden als letztes durchsucht
 - Ein fehlerhafter Verweis führt zum Problem

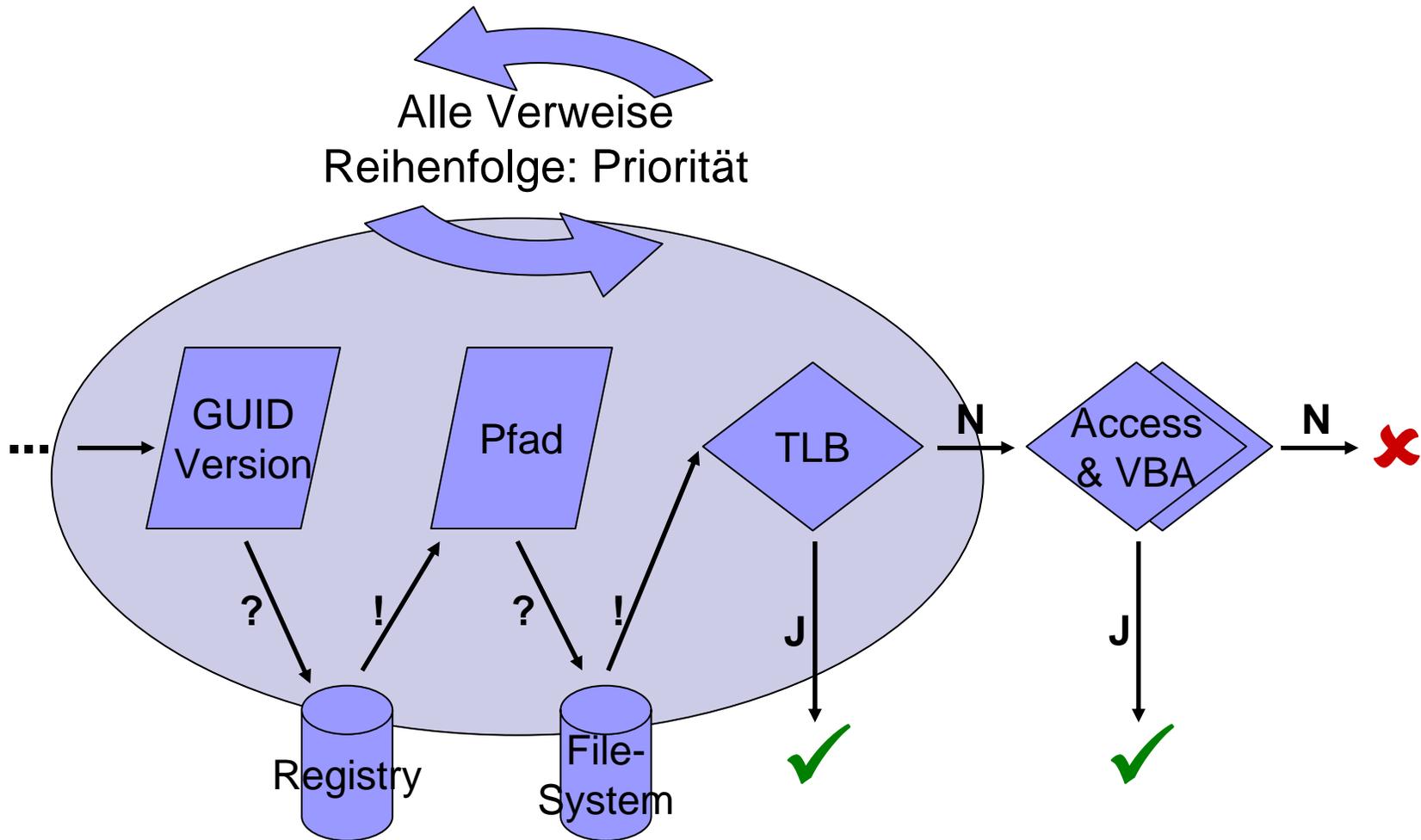
- Vermeidung: Bibliothek beim Befehl angeben
 - VBA.Left\$ oder Access.Application.Run

D
E
M
O

„Disambiguation“



„Disambiguation“



Fehlerhafter Verweis

- Wann ist ein Verweis fehlerhaft?
 - nicht dieselbe Version im System registriert
 - die registrierte Datei wird nicht gefunden

- Gründe:
 - Ältere Version ist / wurde installiert
 - Verweis ist nicht registriert / Registrierung wurde aufgehoben
 - Registrierte Datei wurde umbenannt / verschoben / gelöscht
 - Eine neue Version hat die binäre Kompatibilität durchbrochen

Wo erfolgt Disambiguation?

- VBA
- (Jet) Expression Service

- Was ist der (Jet) Expression Service?
- Dient zur Auswertung von Ausdrücken in
 - Abfragen
 - Steuerelementen
 - Feldeigenschaften

Unterschied in MDB und MDE (?)

- in MDB (nicht kompiliert)
 - Code wird modulweise geladen
 - gesamtes Modul wird interpretiert
 - dabei werden passende Objekte gesucht
 - ein fehlerhafter Verweis führt zum Problem
- in MDE (und kompilierter MDB)
 - Code ist bereits kompiliert
 - keine Suche nach passenden Objekten zur Laufzeit
 - Fehler beim Laden eines fehlerhaften Verweises
- Im Ergebnis kein Unterschied

Wie löst Access Verweise auf?

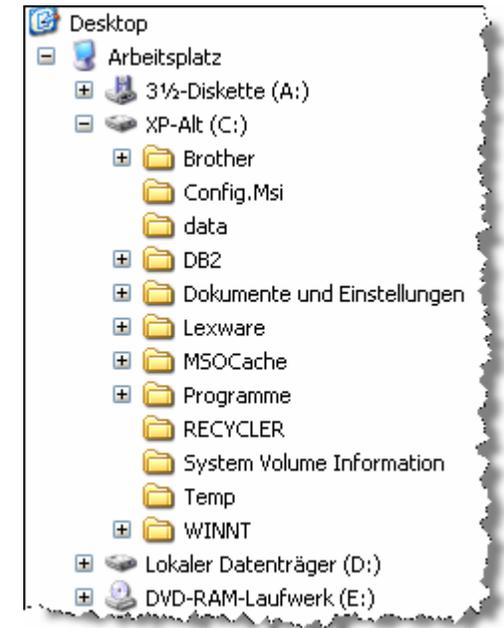
Vorgehen gemäß [KB 824255](#):

1. Prüfung, ob Verweis bereits geladen
2. Wird Verweis in RefLibPaths gefunden?
 - HKLM\Software\Microsoft\Office\x.0\Access
 - Name des Wertes: Name der DB
 - Wert: Kompletter Pfad zur Datenbank
3. Verwendung der SearchPath API

Wenn Verweis nicht gefunden wird, dann Fehler!

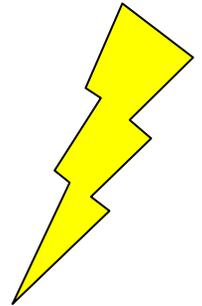
SearchPath - API

- Anwendungsverzeichnis
 - Verzeichnis von MSaccess.exe
- Aktuelles Verzeichnis
 - Wenn man Datei / Öffnen klickt
- System-Verzeichnis
 - \System und \System32
- Windows-Verzeichnis
- Umgebungsvariable PATH
 - Alle eingetragenen Pfade
- Verzeichnis, in dem Datenbank gespeichert ist
 - Incl. Unterverzeichnisse



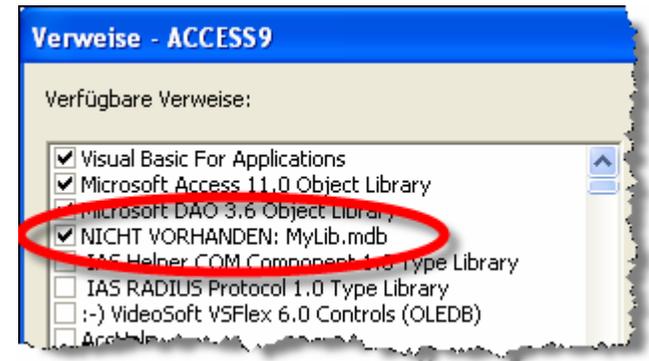
Fehlermeldungen / Symptome

- Fehlermeldung:
 - Funktion steht in Ausdrücken nicht zur Verfügung
 - Fehler beim Kompilieren:
Projekt oder Bibliothek nicht gefunden
- Steuerelementinhalt:
 - #Name
- Fehlermeldung:
 - In diesem Steuerelement befindet sich kein Objekt
 - Objekt unterstützt diese Eigenschaft oder Methode nicht
- Fehlermeldung:
 - Benutzerdefinierter Typ nicht definiert



Manuelle Lösungen (I)

- VBA-Editor öffnen
 - ALT + F11
- Verweis-Dialog öffnen
 - Extras / Verweise
- Prüfen, ob Verweis >NICHT VORHANDEN<
- Defekten Verweis entfernen / abwählen
- Prüfen, ob Verweis benötigt wird
 - Debuggen / Kompilieren...
- Eventuell Verweis wieder aktivieren



Manuelle Lösungen (II)

- Wenn kein Verweis fehlt:
 - Beliebigen Verweis einfügen, Dialog verlassen, Dialog wieder öffnen, Verweis wieder entfernen
- Alternativ:
 - Alle Verweise entfernen, Dialog verlassen, Dialog öffnen, Verweise wieder einfügen
- Letzte Möglichkeit:
 - Alles in neue Datenbank importieren

- Nur möglich in MDB, nicht in MDE oder Runtime

Lösungen per VBA (I)

- Defekte Verweise ermitteln und Info an den User geben
 - BrokenReference-Eigenschaft auswerten
 - Schleife über alle Verweise und IsBroken-Eigenschaft auswerten
- In MDB, MDE und RT möglich

D
E
M
O

Lösungen per VBA (II)

- Verweisliste aktualisieren
- Vorgehen gemäß [KB 194374 \(de\)](#)
 - einen Verweis löschen
 - diesen Verweis erneut einfügen
 - alle Module kompilieren
- Funktioniert in MDB
- Funktioniert nicht in MDE und RunTime

D
E
M
O

Lösungen per VBA (III)

- Alle defekten Verweise aktualisieren
- Vorgehen gemäß [KB 194374 \(en-us\)](#)
 - alle defekten Verweise löschen
 - diese Verweise erneut einfügen
 - alle Module kompilieren
- Funktioniert in MDB
- Funktioniert nicht in MDE und RunTime

D
E
M
O

Lösungen per VBA (IV)

- Alle verwendeten Verweise beim Start aktualisieren
 - alle Verweise löschen
 - aktuelle Version der Verweise erneut einfügen
 - AddFromGUID mit Parameter 0, 0 verwenden
 - Alle Module kompilieren
- Funktioniert in MDB
- Funktioniert nicht in MDE und RunTime

D
E
M
O

Garantie für funktionierende Verweise (I)

- Überlegungen von Michael Kaplan
 - <http://www.trigeminal.com/usenet/usenet026.asp?1033>
- Einzeiliges AutoExec-Makro verwenden
 - Nur Funktion (FixRefs) zur Prüfung Verweise starten
- Nur explizite Funktionsaufrufe
 - Vermeidung von Disambiguation
- Funktion FixRefs in gesondertem Modul
 - vermeidet Risiken durch Disambiguation
- Keine anderen Libraries verwenden
 - wenn doch, dann Late-Binding
- Keine Formulare laden
 - insbesondere keine mit OCX

Garantie für funktionierende Verweise (II)

- Expression Service nicht verwenden
 - Verweisprobleme wegen Disambiguation
- Nicht trödeln
 - Access lädt Teile nachträglich
- Alle Verweise prüfen
 - Nicht über Reference-Auflistung und IsBroken
 - stattdessen Objekte mit Late-Binding erstellen
- Defekte Verweise reparieren
 - nicht mit Shell RegSvr32 sondern mit API
- Prüfung, ob Reparatur erfolgreich
 - sonst Info an User und Start abbrechen

VerweisChecker

- Lösung von Terry Kreft u. Dev Ashish
 - <http://www.mvps.org/access/modules/mdl0022.htm>
- Kann als AddIn verwendet werden
- Lässt sich auch in Code einbinden
- Beim Start der Anwendung wird auf defekte Verweise geprüft
- „Schweizer Messer“ für Probleme mit Verweisen
- Lösung funktioniert in MDE nur teilweise
- Für Kommunikation mit Enduser geeignet?

D
E
M
O

Fazit

- Verweise erleichtern die Arbeit beim Programmieren.
- Verweise schaffen im Gegenzug eine Abhängigkeit vom ausführenden Rechner.
- Je mehr Verweise verwendet werden, desto größer ist die Abhängigkeit.
- Probleme mit Verweisen können nur bedingt mit VBA beseitigt werden.
- Verzicht auf Verweise anstreben!





4. Late- / Early-Binding

- Early-Binding
- Late-Binding
- Problem mit Verweisen
- Strategien

Early Binding

- Verweis wird gesetzt

- Instanziierung wie folgt:

```
Dim wdApp As Word.Application  
Set wdApp = New Word.Application
```

- Vorteile:

- Bessere Performance
- IntelliSense-Unterstützung
- Syntax-Prüfung beim Kompilieren
- Verwendung eingebauter Konstanten

- Nachteil:

- Es wird ein Verweis benötigt

Late Binding

- Es wird kein Verweis gesetzt
- Instanziierung wie folgt:

```
Dim objApp As Object  
Set objApp = CreateObject(„Word.Application“)
```
- Nachteile:
 - Schlechtere Performance
 - Kein IntelliSense
 - Keine Syntax-Prüfung beim Kompilieren
 - Konstanten müssen selbst deklariert werden
- Vorteil:
 - Unabhängig von Verweisen

Problem mit Verweisen

- Ausgangslage:
 - Access Anwendung mit Verweis auf Word
- Ausführung auf Rechner mit gleicher Version
 - Keine Probleme zu erwarten
- Ausführung auf Rechner mit neuerer Version
 - Verweis wird selbständig angepasst
 - Hierbei kann es zu Fehlern kommen
- Ausführung auf Rechner mit älterer Version
 - Verweis kann nicht angepasst werden
 - Neue Version war bei der Erstellung der älteren Version noch nicht bekannt

Strategien

■ Early-Binding

- Macht Programmieren bequem
- Problem mit Verweisen

■ Late-Binding

- Keine Unterstützung beim Programmieren
- Keine Probleme mit Verweisen
- Unabhängig von Versionen

■ Lösung

- Programmierung mit Early-Bindung
- Auslieferung mit Late-Bindung



Von Early- zu Late-Binding (I)

- Verweis entfernen
- Deklaration aller Objekt-Variablen anpassen
 - `Dim objWord As Object`
- Instanziierung der Objekte mit `GetObject` oder `CreateObject`
 - `Set objWord = GetObject(, „Word.Application“)`
- Konstanten durch Wert ersetzen oder selber definieren
 - `Const wdDoNotSaveChanges = 0`
- Evtl. optional vorhandene Parameter ergänzen

Von Early- zu Late-Binding (II)

- Werte für Konstanten ermitteln
 - Objektkatalog oder
 - Direktfenster

- Code-Zeilen für Early-Binding
 - Nur auskommentieren, nicht entfernen
 - Macht Wartungsarbeiten einfacher

- Alternative
 - Bedingte Kompilierung einsetzen

5. ActiveX

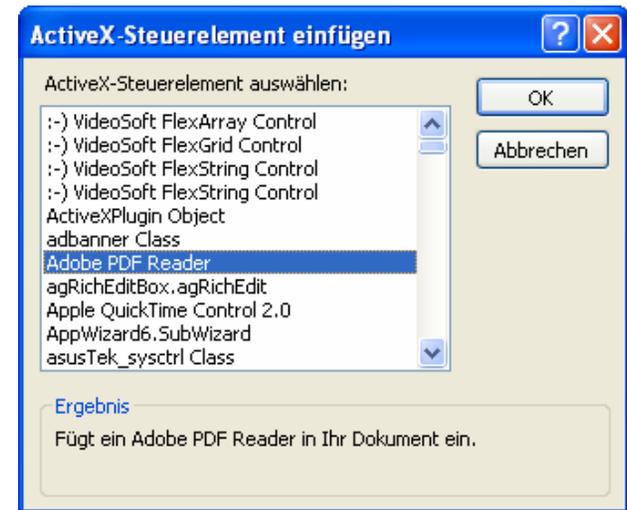
- Was ist ein ActiveX / *.ocx?
- Praktische Hinweise
- DLL Hell
- Ersatzlösungen (statt OCX)
- OCX/DLL Version Checker

Was ist ein ActiveX / *.ocx?

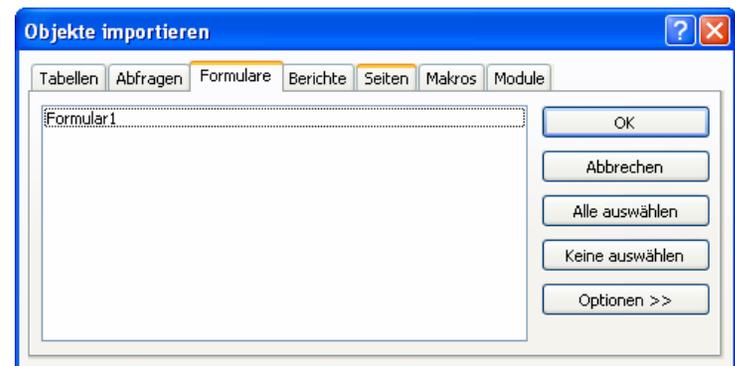
- Zusatz-Steuererelement
- In externer Bibliothek
- Muss registriert werden
 - Per Setup
 - Extras / ActiveX-Steuererelemente / Registrieren
 - RegSvr32.exe C:\DeinPfad\DeinActiveX.ocx
 - Declare Function DllRegisterServer Lib "DeinActiveX.OCX" () As Long
- Steht dann systemweit zur Verfügung

Praktische Hinweise

- Einfügen eines ActiveX-Steuererelements
- Verweis wird automatisch gesetzt!



- Importieren eines Formulars mit ActiveX-Steuererelement
- Verweis wird nicht übernommen



DLL Hell

- Problem bei der Nutzung gemeinsam verwendeter OCXe (z.B. Common Controls)
- Viele Programme verwenden diese Steuerelemente
- Jedes Programm bringt bei der Installation seine eigene Version mit
- Probleme vorprogrammiert

- OCX häufig nur Wrapper für API-Calls
- Verzicht auf OCX anstreben



Ersatzlösungen

Steuerelement	Ersatzlösung
Kalendersteuerelement	Kalender Links v. Tony Toews
CommonDialog <ul style="list-style-type: none"><input type="checkbox"/> Dateiauswahl<input type="checkbox"/> Farbauswahl<input type="checkbox"/> Schriftartenauswahl	FileDialog v. Karsten Pries ChooseColor-Dialog (T.Kreft / S.Lebans) ChooseFont-Dialog (T.Kreft / S. Lebans)
ListView, TreeView, ImageList, Slider, Progressbar, ...	“Programmer´s Libraries” (\$) von Johann Pumhösl http://www.access-paradies.de/kunden/tools_fuer_access.php

OCX/DLL Version Checker

- Tool von Tony Toews

 - <http://www.granite.ab.ca/access/ocxdllversionchecker.htm>

- Dient zum Vergleich der DLL-Versionen

- Vorgehen

 - Verwendete DLLs eintragen
 - Versionen ermitteln
 - MDE erstellen
 - beim Kunden ausführen
 - Unterschiede werden angezeigt

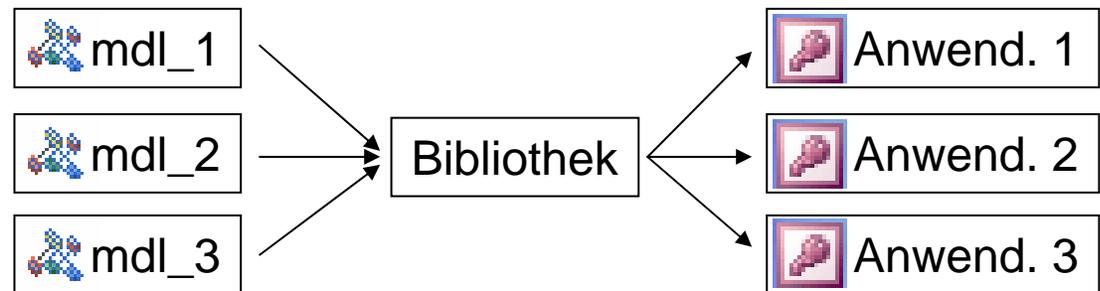
**D
E
M
O**

6. Bibliotheks-DBs

- Sinn und Zweck
- Verweise auf Verweise
- MDB und MDE
- Vorsicht mit MDEs
- Praktische Hinweise
- Verweisprobleme vermeiden
- Alternative: Visual SourceSafe

Sinn und Zweck

- Alle Module mit Standardfunktionen werden in eine Datenbank ausgelagert (Bibliothek)
- Bibliothek wird als Verweis in Anwendungen eingebunden



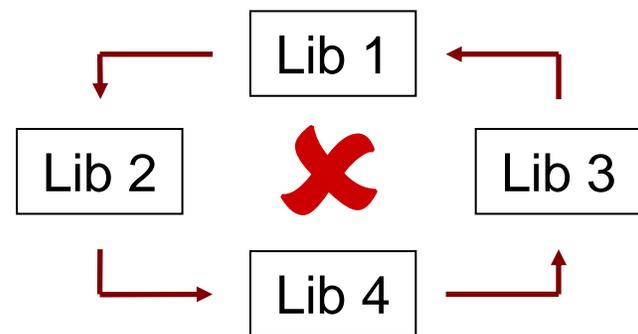
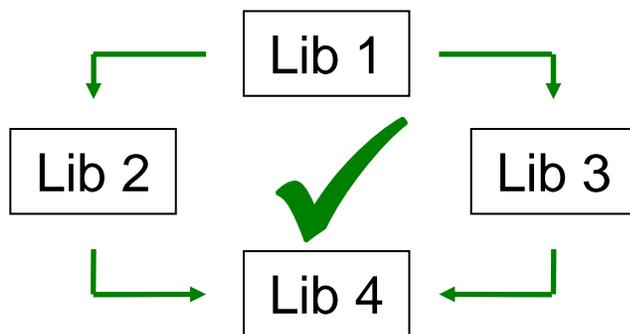
- Vorteile:

- Alle Standardfunktionen sind an einer Stelle verfügbar.
- Erweiterungen / Fehlerbereinigungen müssen nur an einer Stelle erfolgen

Verweise auf Verweise

- Verweis auf Datenbank, die wiederum Verweis auf Datenbank enthält
 - generell möglich
 - wird schnell unübersichtlich

■ Achtung: Zirkelbezüge



MDB und MDE

- Verweise sind auf *.mdb und auf *.mde möglich

Anwendung	Verweist auf	Funktioniert?
*.mdb	*.mdb	✓
*.mde	*.mdb	✗
*.mdb	*.mde	✓
*.mde	*.mde	✓

- Verweis auf *.mde funktioniert immer

Vorsicht mit MDEs

■ Konstellation

- Anwendung liegt als *.mde vor
- Library liegt ebenfalls als *.mde vor
- Library ist als Verweis in Anwendung eingebunden

■ Änderung an Library

- Code in Library wird geändert / angepasst
- Library wird erneut als *.mde erstellt

■ Problem:

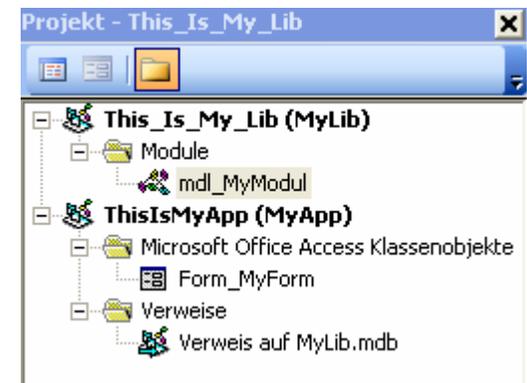
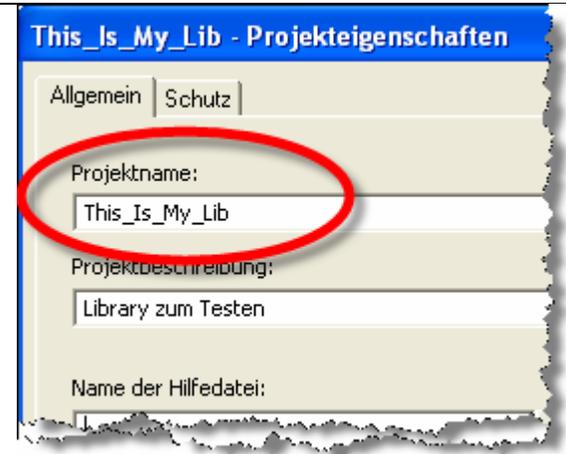
- Fehler, wenn Anwendung Code aus (geänderter) Library aufruft

■ Lösung:

- MDE der Anwendung ebenfalls erneut erstellen.

Praktische Hinweise

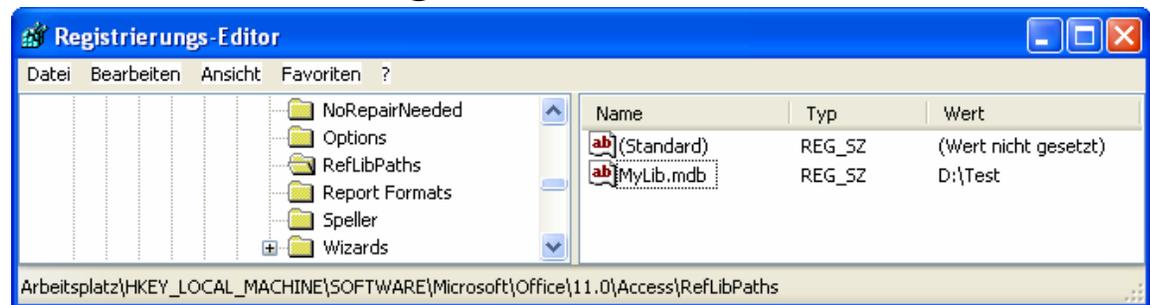
- Name des VBA-Projektes ist der Name des Verweises
- Name des Verweises muss eindeutig sein
- Auf Module der Library kann über Projekt-Explorer (Strg+R) zugegriffen werden
- Achtung: Änderungen können nicht gespeichert werden!!!



Verweisprobleme vermeiden

- Library im Ordner der Anwendung ablegen
 - wird automatisch gefunden
 - wird auch in relativ gleicher Position zur Anwendung gefunden
- RefLibPaths verwenden
 - Eintrag in Registry
 - dient zum Auffinden von Verweisen
 - wird beim **Start** von Access geladen

D
E
M
O



Alternative: Visual SourceSafe

■ Vorgehen:

- Objekte können ***gemeinsam genutzt*** werden
- Änderungen an Objekt werden automatisch in allen anderen Anwendungen nachgeführt

■ Vorteil:

- Anwendung unabhängig von Bibliotheksverweisen

■ Nachteile:

- Developer-Version wird benötigt
- Visual SourceSafe wird benötigt



Anhang

- Links, Links, ...

Links, Links, ...

■ Access Reference Problems

□ <http://www.accessmvp.com/djsteele/AccessReferenceErrors.html>

■ Reference Wizard

□ <http://www.mvps.org/access/modules/mdl0022.htm>

■ How to guarantee that reference will work

□ <http://www.trigeminal.com/usenet/usenet026.asp?1033>

■ Solving Problems with Library References

□ <http://www.allenbrowne.com/ser-38.html>

■ MS KB: So löst Access VBA-Verweise auf

□ <http://support.microsoft.com/kb/824255/de>

■ MS KB: Funktion steht in Ausdrücken nicht zur Verfügung

□ <http://support.microsoft.com/kb/194374/de>

Links, Links, ...

- DLL Hell – DLL Heaven

- <http://www.papwalker.com/dllhell/index.html>

- Early vs. Late-Binding

- <http://word.mvps.org/FAQs/InterDev/EarlyvsLateBinding.htm>

- OCX/DLL Version Checker

- <http://www.granite.ab.ca/access/ocxdllversionchecker.htm>

- ActiveX FAQ

- http://www.avenius.com/?Support:ActiveX_FAQ

- RefLibPaths Schritt-für-Schritt

- http://smsconsulting.spb.ru/shamil_s/topics/testrefs.htm

- Access 2002 Developer's Handbook

- Seiten 796 ff. und Seiten 1144 ff.