

*Das Beste, was ich weiß, hab ich durch
Umgang, Erzählung, Lektüre gelernt.*

Theodor Fontane, (1819 - 1898)

Wer bin ich...

Zur Person

- Jörg Bottler, Lebensgemeinschaft, fünfjähriger Sohn
- 44 Jahre, gebürtiger Saarländer
- Dipl.Ing. E-Technik
- Hobbys: Theaterspielen, VBA-Programmierung und MS-Access Entwicklung.
- Wohnhaft in Waghäusel bei Heidelberg/Karlsruhe

Hauptberuf

- Teamleiter, Projektmanager und „VBA-Toolfrickler“ in der T-Systems SI am Standort Darmstadt

Nebentätigkeit (seit 1998)

- Entwicklung, Wartung und Vermarktung der Schulverwaltungssoftware OMNIA auf Basis von MS-Access in den Versionen 97-2010

Kontakt:

- joerg.bottler@omnia-ng.de



Warum bin ich hier...

- Habe selbst schon von der AEK profitiert
- Eigene Best-Practices vermitteln
- Einige Bücher zu Access gelesen

und das WICHTIGSTE

.....EWIGER RUHM !!!!

Warnung: Dies wird kein Lehrbuchvortrag



AEK 2012

Klassen, Events, Collections für den Hausgebrauch

Ein Plädoyer für den alltäglichen Einsatz unpopulärer Methoden

Referent: Jörg Bottler

Schulverwaltungssoftware OMNIA-NG

Historie

1998 Erste Version auf Basis von MS-Access 97

2008 Umstellung auf MS-Access 2007 (Ribbons)

2011 Umstellung auf MS-Access 2010, um auf den Stand der Technik zu bleiben (Backstage-Bereich)

HEUTE: Einsatz an 74 Schulen in Rheinland-Pfalz.
Einmal jährlich erfolgt ein kostenpflichtiges Update (220 €)

Technik

Klassische Frontend (accde) Backend (accdb)-Lösung

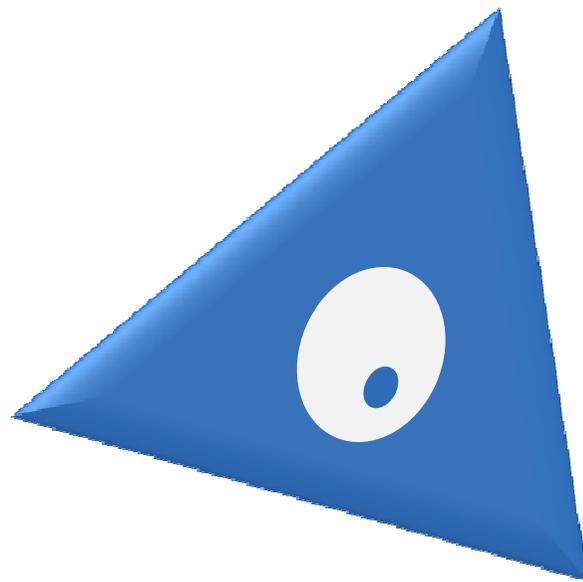
Installationsroutine mit Inno-Setup (Auslieferung als DVD oder Download)

80% der Kunden haben Access Runtime im Einsatz



Sehr kurze Live-Vorstellung

OMNIA-Schulverwaltungssoftware



Teil 1: Klassen

... für den Hausgebrauch

KLASSEN... in ACCESS...wie war das noch ?

LES
OBJEKTORIENTIERUNG
VEREERBUNG
METHODEN
LET
PROPERTY
KONSTRUKTOREN
INITIALIZE
ATTRIBUTE



Phasen der Strukturierung

Sinnhafte Zusammenlegung von Funktionen und Werten zu einem funktionalen Block zur einfachen Verwendung

Klassen

Zusammenfassung von nützlichen Prozeduren zur Wiederverwendung

Library-
Module

Funktionen aufsplitten in kleine Teilfunktionen =>Kein endloser Code in einer Prozedur oder Funktion

Subs & Functions



Persönliche Vorlieben beim Einsatz Klassen

Ich nutze:

- + Kapselung von inhaltlich zusammengehörigen Funktionalitäten
- + Verwaltung und Zugriff auf übergreifende Variablen / Informationen
- + Verständlichkeit von Code und Wieder-verwendbarkeit
- + IntelliSense-Fähigkeit

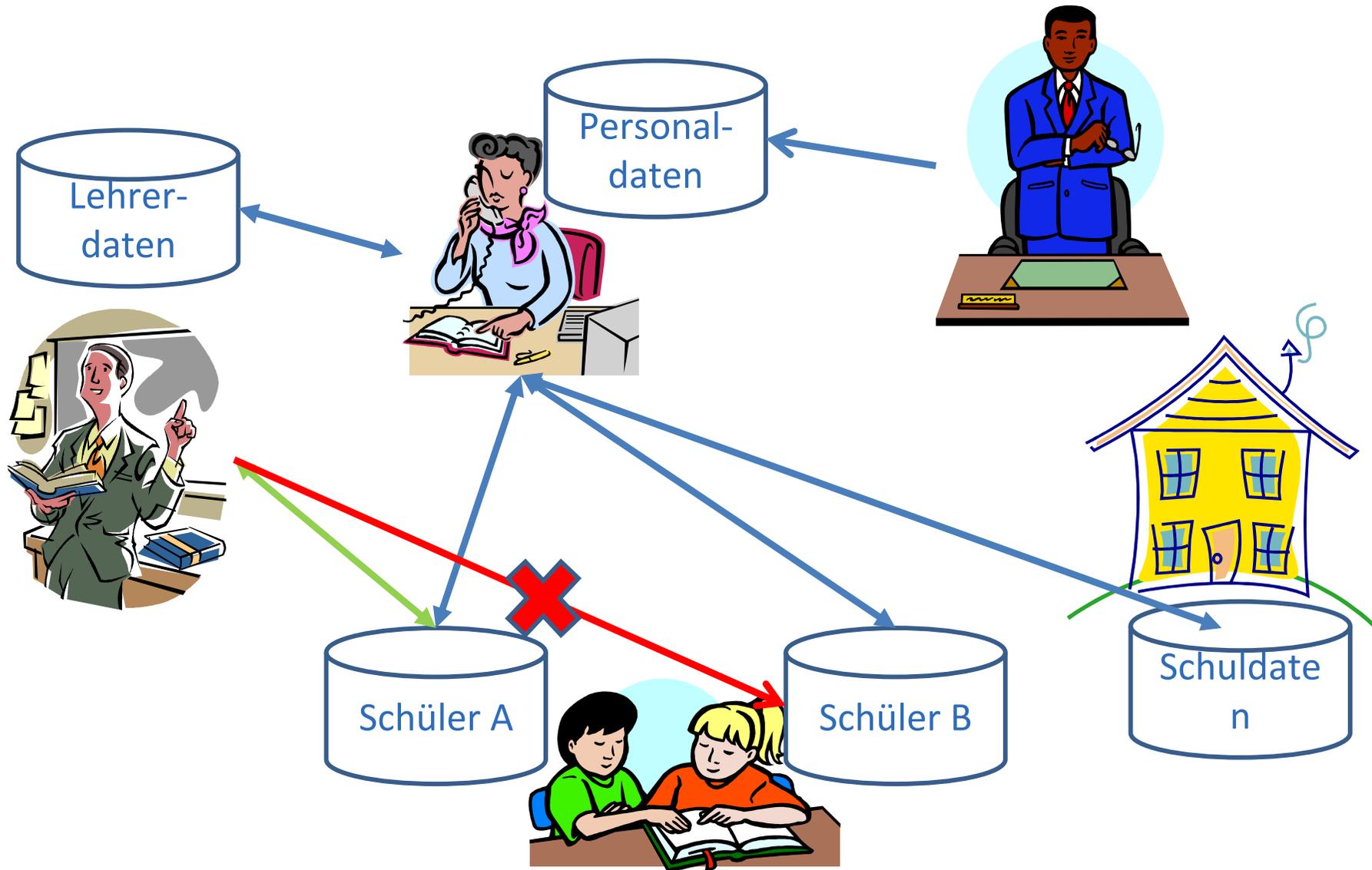
Ich brauche weniger:

- Vererbung (geht auch gar nicht)
- Sklavische Kapselung
- Let, Set Methoden
- Interfaces



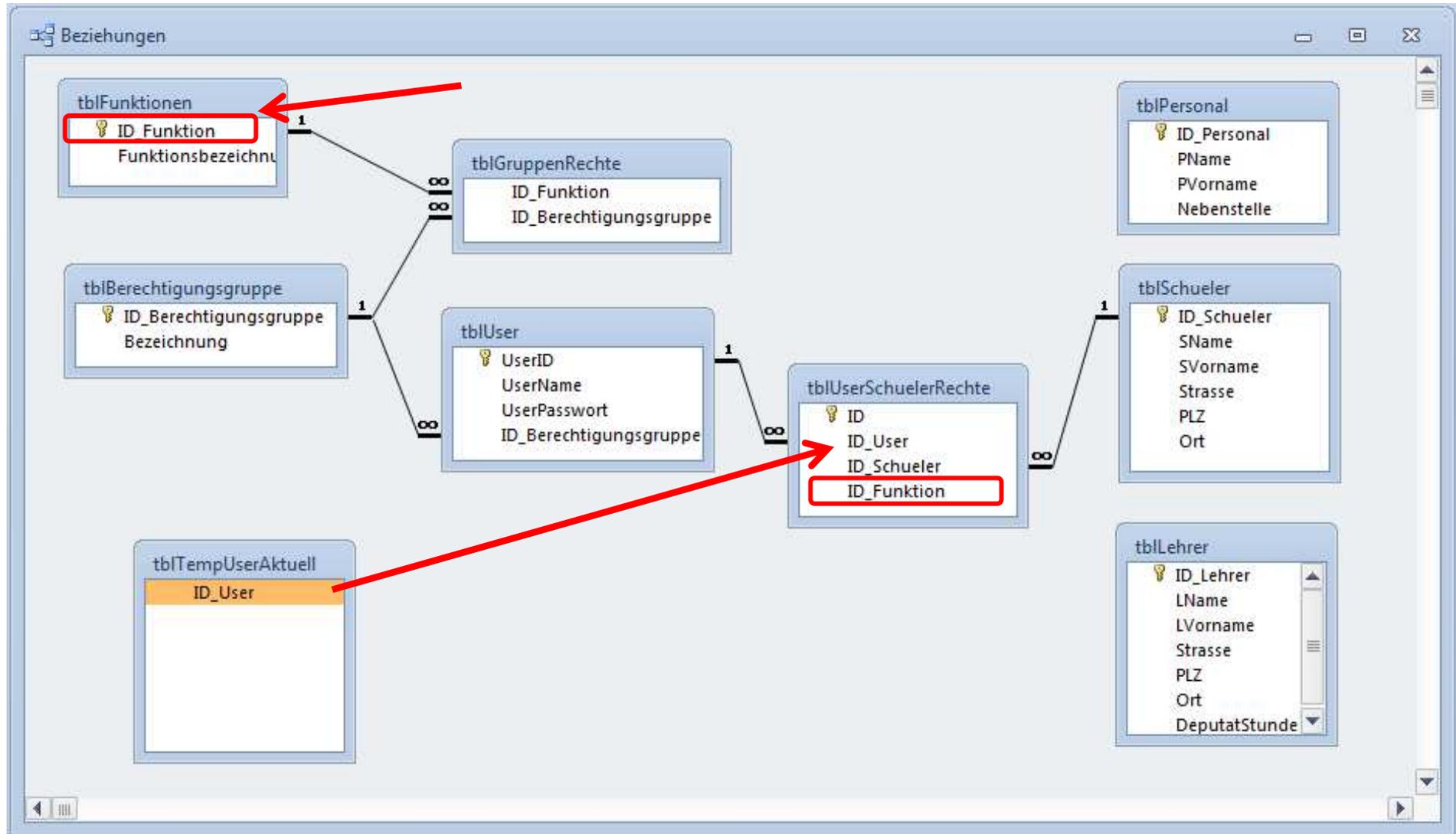
Beispiel 1 – Schwerpunkt Strukturierung:
Steuerungslogik & Berechtigungsthema
in Klassen auslagern

Szenario: Benutzer in Schulverwaltung

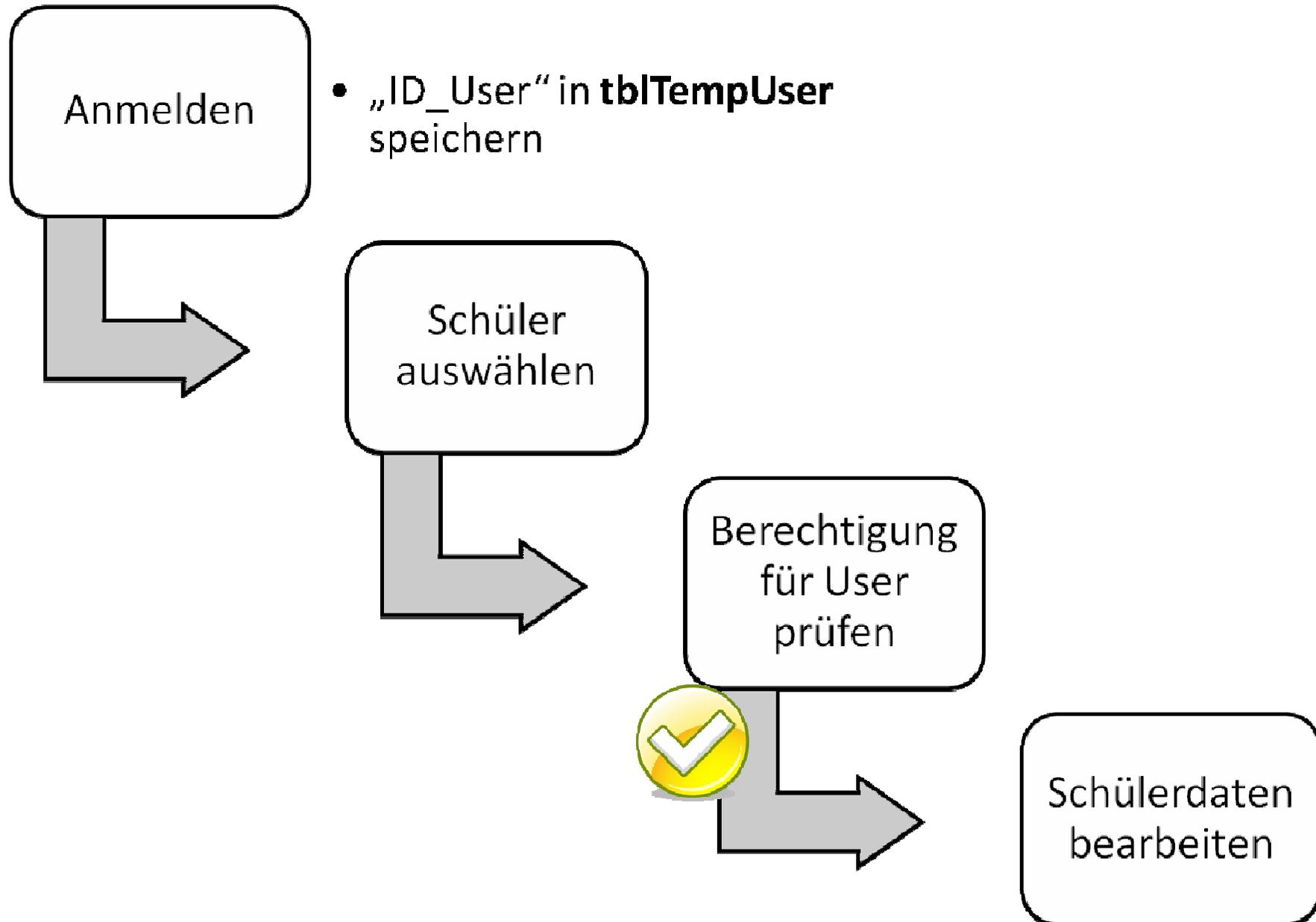


Datenmodell für Szenario

... pro User und Schüler definierbare Funktion (lesen, bearbeiten...)



Einfacher Programmablauf



Mögliche Variationen zur
Berechtigungsprüfung...

Variationen Berechtigungsprüfung

**a) Für jeden Schülerzugriff:
Lesen der Berechtigung aus der
Tabelle „tblUserSchuelerRechte“**

Vorteil:

- Kurzfristige Berechtigungsänderungen sind wirksam

Nachteile:

- Performance Probleme
- Konfliktpotenzial steigt im Multi-Userbetrieb
- Ggf. mehrmaliges Lesen der selben Information

**b) Lesen der Berechtigung aus
dem Hauptspeicher**

Vorteile:

- Einmaliges Lesen der Information bei der Anmeldung aus „tblUserSchuelerRechte“
- Einzelzugriff-Performance deutlich besser, da kein I/O notwendig

Nachteile:

- Änderung von Rechten ist erst nach neuerlicher Anmeldung wirksam



Wäre doch schön, wenn....

... das Berechtigungssystem einfach angepasst werden kann.

... Funktionalitäten zu Berechtigungsthemen einfach im Code verwendet werden können.

... das Ganze noch performant ist.



Lösung: Die Klasse clsSecurityManager

....auf in den Code....

Motivation zum Einsatz einer Klasse

Das Hauptprogramm ist dumm bzgl.
Berechtigungen...also brauchen wir jemanden.

Der Security-Manager:



- Überprüft den Nutzer beim Zutritt
- Weiß jederzeit, wer das Programm bedient und kann darüber Auskunft erteilen
- Kennt seine *Pappenheimer* und beantwortet Rechte-Anfragen des Hauptprogramms, wer was darf oder nicht.
- Meldet den Nutzer ab, bei Beendigung oder Logoff



Die wesentlichen Code-Teile

Klasse mit gewünschten Fähigkeiten und Attributen

clsManagerSecurity

```
Public Aktueller_Username As String
Public Aktuelle_UserID As Long

Public Function UserAbmeldung()
Public Function UserAnmeldung(...)
Public Function DarfUser(...)

Private Sub ResetUserInfos()
Private Sub GetAktUserInfos()
Private Function PasswortPruefung(...)
Private Sub AktualisiereUserInformationen(...)
Private Sub Loesche_SchuelerRechteInformation()
```

Erzeugung und zum Ansprechen des Objektes in globaler Methode:

MANAGER_SECURITY()

```
Public Function MANAGER_SECURITY() As clsManagerSecurity
Static myManager As clsManagerSecurity

If myManager Is Nothing Then
    Set myManager = New clsManagerSecurity
End If

Set MANAGER_SECURITY = myManager
Exit Function

End Function
```



Beispiel 2: Schwerpunkt Wiederverwendung
XML-Schnittstelle bedienen.

Szenario: XML-Schnittstelle des StaLa RLP

Auszug aus einer Ausgabe

```
<Klasse>
  <KlassenBez>5a</KlassenBez>
  <KlassenArt>2</KlassenArt>
  <KlassenStufe>5</KlassenStufe>
  <Schueler>
    <SchuelerId>2</SchuelerId>
    <Name>Acker</Name>
    <Vorname>Konstantin</Vorname>
    <KlassenStufe>5</KlassenStufe>
    <Geschlecht>m</Geschlecht>
    <GeburtsDatum>2000-10-22</GeburtsDatum>
    <Nationalitaet>0</Nationalitaet>
    <WohnortRLP>33800017</WohnortRLP>
    <Religion>2</Religion>
    <TeilnahmeReligion>2</TeilnahmeReligion>
    <EinschulungsJahr>2007</EinschulungsJahr>
    <FremdSprachen>
      <FremdSprache>
        <Sprache>7</Sprache>
        <StatusFach>5</StatusFach>
        <StatusErteil>1</StatusErteil>
        <Folge>1</Folge>
      </FremdSprache>
    </FremdSprachen>
  </Schueler>
</Klasse>
```



Programmatisch

- Schleifen über Schulklassen und Schüler
- Komplexe Tags, die mehrere Attribute haben
- Nur Ausgabe, wenn Wert vorhanden

Layout

- Beginnende und schließende Schleifen-Tags
- Einrückung muss so sein, dass Struktur erkennbar ist. (nicht zwingend aber hilfreich für Ergebniskontrolle)



Detailbetrachtung Auszug

Öffnende

„Komplexe“ Schleifen-Tags

Öffnendes Attribut-Tag

Schließendes Attribut-Tag

```
<Klasse>  
  <KlassenBez>5a</KlassenBez>  
  <KlassenArt>2</KlassenArt>  
  <KlassenStufe>5</KlassenStufe>  
  <Schueler>  
    <SchuelerId>2</SchuelerId>  
    <Name>Acker</Name>  
    <Vorname>Konstantin</Vorname>  
    ...
```

Schliessende

Schleifen-Tags

```
</Schueler>  
</Klasse>
```



Die Kandidaten für eine Klasse

Kandidaten für eine programmatische Klasse

```
<Klasse>  
  <KlassenBez>5a</KlassenBez>  
  <KlassenArt>2</KlassenArt>  
  <KlassenStufe>5</KlassenStufe>
```

```
<Schueler>
```

```
  <SchuelerId>2</SchuelerId>
```

```
  <Name>Acker</Name>
```

```
  <Vorname>Konstantin</Vorname>
```

```
</Schueler>
```

```
</Klasse>
```

clsXMLKlasse

clsXML

clsXMLSchueler



clsXML, clsXMLSchueler, clsXMLKlassen

....auf in den Code....

FAZIT: Für mich wichtigste Vorteile von Klassen

Strukturierung von Code

- „Verantwortung“, Funktionen und Werte an Objekte übergeben - > Security-Manager, File-Manager, Ablauf-Manager

Write and Forget

- Ich kann vergessen, wie ein Objekt seine Funktionen umsetzt: Ich nutze diese nur noch
(Ausnahme: Wenn Funktion geändert werden soll)

IntelliSense

- Über IntelliSense bekomme ich die Fähigkeiten und Inhalte meines Objektes zugeliefert



Teil 2: Collections

... für den Hausgebrauch

Collections für Dummys in 5 Minuten

Ein Collection-Objekt ist eine geordnete Folge (oder auch Auflistung) von Elementen, auf die als Einheit (***mittels eines Schlüssels***) Bezug genommen werden kann.



Vergleich Arrays und Collections

Array

Inhalt :

- Elemente müssen gleichen Datentyps sein.

Erweiterung um ein Element:

- Über „ReDim Preserve“ Anweisung

Löschen eines Elementes:

- Feldinhalt an Indexposition löschen.

Nutzung mit Programm-Schleifen:

- Schleifen über Index

Zugriff auf bestimmtes Element:

- Über Feldindex oder Suche nach Inhalt, mittels Durchlaufen des Feldes

Collection

Inhalt:

- Elemente können verschiedenen Datentyp haben.

Erweiterung um ein Element:

- Über Add-Methode

Löschen eines Elementes:

- Über Remove-Methode

Nutzung mit Programm-Schleifen:

- Schleifen über Index oder For each...

Zugriff auf bestimmtes Element:

- Zugriff über **Key-Value** liefert sofort Objekt zurück.



Collection-Beispiel: **Personenliste**

Erstellen, Hinzufügen und Löschen von Personen

....ab in den Code....

Wichtigste Codezeilen

Deklaration

- *Public mMeinePersonenliste As Collection*

Definition

- Set mMeinePersonenliste = New Collection

Add-Methode mit Key-Wert (String)

- mMeinePersonenliste.Add ErstePerson, "" & ID

Zugriff per Schlüssel auf ein Element in der Liste

- Set myPerson =
mMeinePersonenliste(CStr(parID))

Löschen eines Elements aus der Liste über Key

- mMeinePersonenliste.Remove CStr(parIDPerson)



Beispiele für Collections als nützliche Helferlein...

Vermeidung von mehrfachen Datenbankzugriffen

- Beispiel Userrechte für 1000 Schüler beim Anmeldeprozess einlesen. Spätere Abfrage der Berechtigung nur über die Collection und die ID des Schülers

Summierung von Werten auf einem bestimmten Begriff (z.B. beim Auswerten von Zeitsheets)

- Verschiedene Summen als Collection
 - Tätigkeit = Schlüssel , Summe = Inhalt



Wo Licht ist, ist auch Schatten....

Nachteile/Risiken bei der Verwendung von Collections

- Die Information liegt im Speicher und ist damit nicht persistent
- Bei nicht abgefangenen Fehlern geht der Inhalt verloren - > Vorkehrungen für nochmaliges Einlesen treffen
- Inhalte können nicht einfach betrachtet werden, wie z.B. in einer Tabelle oder Abfrage.
- Datenmenge ist ggf. durch Arbeitsspeicher begrenzt



Teil 3: Events

... für den Hausgebrauch

Kurz zur Theorie

Quelle-Wikipedia:

Ein **Ereignis** ([engl. event](#)) dient....zur Steuerung des [Programmflusses](#).

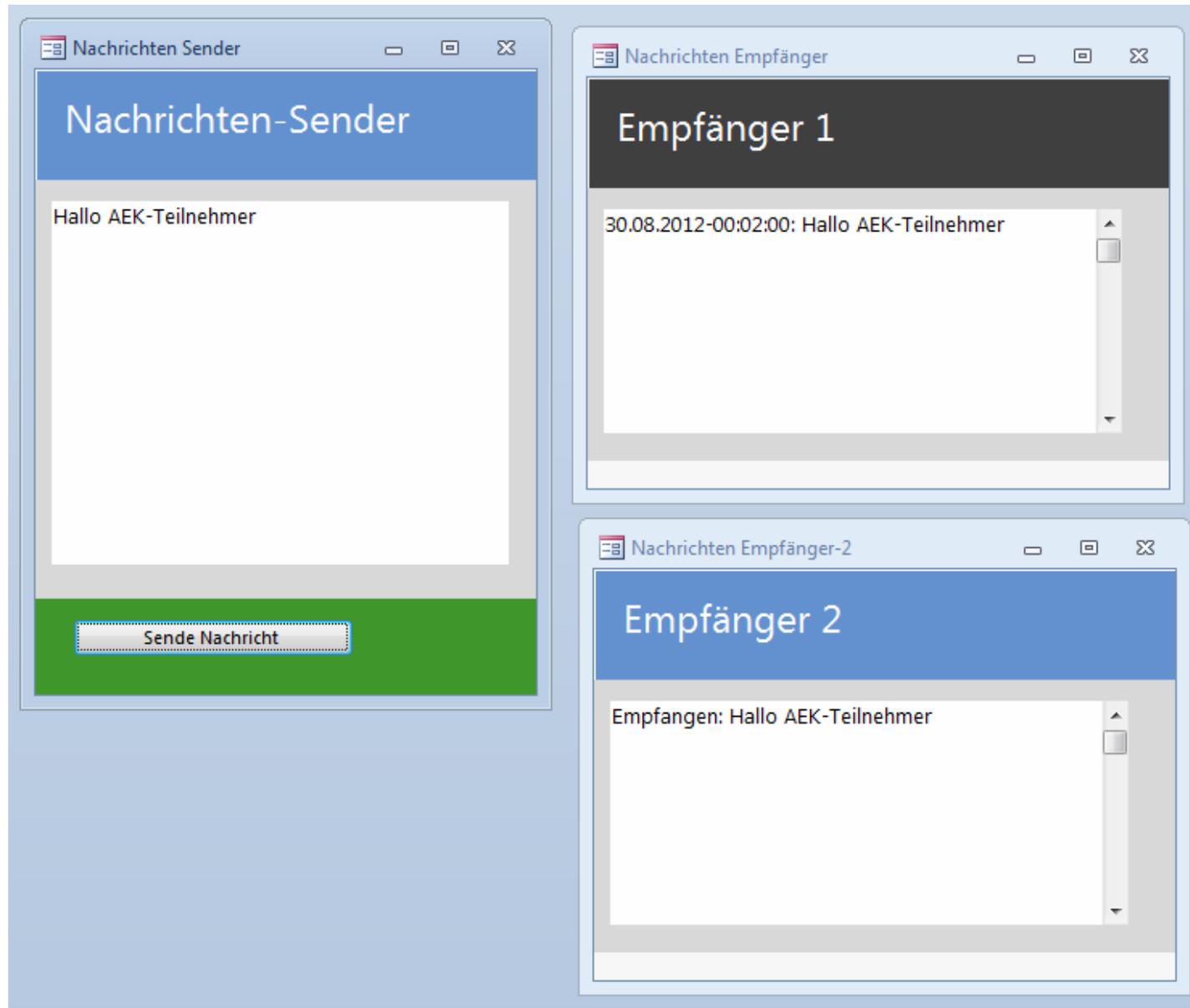
Das Programm wird nicht linear durchlaufen, sondern es werden spezielle *Ereignisbehandlungsroutinen* (engl. *listener, observer, event handler*) immer dann ausgeführt, wenn ein bestimmtes *Ereignis* auftritt.



Wie generiert man eigene Events ?

... ab in den Code...

Code-Beispiel: Sender & 2 Empfänger



Wichtigste Code-Fragmente des Sender-Codes

Sender-Klasse: clsManagerNachrichten

'Event-Definition

```
Public Event NachrichtAnAlle(Nachrichtentext As String)
```

...

'Funktion zum Broadcasten des Events

```
Public Sub EventAusloesen_NachrichtAnAlle(parNachricht as string)
```

```
{
```

```
RaiseEvent NachrichtAnAlle(parNachricht)
```

```
}
```

Sender-Instanz über zentrale Routine rückliefern:

```
Public Function MANAGER_Nachrichten() As clsManagerNachrichten
```

Formular, das Sender-Objekt nutzt -

frmNachrichtenSender:

```
MANAGER_Nachrichten().EventAusloesen_NachrichtAnAlle Me.txtNachricht
```



Wichtigste Code-Fragmente des Empfängers

Formular: frmNachrichtenEmpfaenger

‘Formular mit Sender-Objekt Typ und dessen Events „vertraut“ machen
Public WithEvents SenderObjekt As clsManagerNachrichten

‘Formular auf konkretes Sender-Objekt einstellen in Form-Load Methode

```
Private Sub Form_Load()  
    Set SenderObjekt = MANAGER_Nachrichten()  
    ....  
End Sub
```

‘Auf ein bestimmtes Event des Objektes reagieren

```
Private Sub SenderObjekt_NachrichtAnAlle(Nachrichtentext As String)
```



Gründe zur Nutzung eigener Events

- Flexible Reaktionsmöglichkeiten auf **ein** Ereignis
- 1... n Abnehmer/Consumer auf **ein** Ereignis
- Implementierung des Codes ist gut getrennt durch verschiedene Implementierer möglich, da das Werfen und Verarbeiten des Events getrennt voneinander codierbar sind.



Gefahren & Nachteile bei Nutzung eigener Events

Nachteile:

- Der Programmablauf als Ganzes ist nicht mehr leicht erkennbar
- Das Debugging wird erschwert
- Synchronisationsprobleme können auftreten, wenn mehrere Consumer das selbe Event verarbeiten
- Bei unbehandelter Fehlerroutine geht Sender-Bezug verloren, ohne dass man es merkt

UND ACHTUNG:

- Die Routine, die das Event wirft, wartet, bis alle Consumer mit der Verarbeitung des Events fertig sind. (fire and forget ist also nicht)



Erfahrungsbeispiel: Ein überfrachtetes,
langsames Formular wird wieder schlank

Phase 1: Access 97 - Alles in einem Formular

Daten des Schöler

Schölerliste

Abadian	Florian
Abanon	Kai Uwe
Abba	Julia
Aber	Linda
Aber	Tobias
Aberes	Klaus Michael
Beispiel	Markus
Ebertor	Bianca
Muster	Marion
Stein	Alexander
Stelder	Andre
Stolski	Alice
Strang	Wilhelm
Ukar	Marcell
Ukhardt	Sabine
Ukuler	Jasmin
Ukunos	Bernd
Ukuz	Alexander
Ulbricht	Sabrina
Ülünkir	Jennifer
Ümaker	Joachim
Ümakun	Witali
Umbauer	Ralf
Ümberg	Dennis Thoma
Ümunkel	Sarah
Ümunkel	Steffen
Ümürer	Simon
Ümürta	Silvia
Umut	Mike
Umuth	Sandra
Ümütur	Florian
Undola	Sabrina
Unkert	Christian
Unterhard	Michael
Urbusch	Alexander
Urkhardt	Andreas
Urnisch	Kristina
Urutzki	Daniel
Üter	Natascha
Ütkron	Jessica
Üturq	Jasmin

Name: **Stelder**
Vorname: **Andre**
Nr: **364**

Klasse:
Status:

Aktuelles Jahr
 Kommdendes Jahr

Personendaten | Sorgeberechtigte | Statistik 1 | Statistik 2 | Sprachen +MEU | Behinderung | Joker | Merkmale | Notiz | Historie | Kurse

Kontaktdaten

Name:
Vorname:
Anrede:
Titel:
Straße:
PLZ:
Ort:
WohnortKey: ?
TelNr:
TelNr_2:
FaxNr:
eMail:
Notfall_Kontakt:
Notfall_Telefon:

Primäre Statistikdaten

GebDat:
GebOrt:
Geschlecht:
StaatsAng:
Religion:
TeilnAnRelUnt:

Schulhintergrund

Schulbesuchsjahr:
Einschulungsjahr:
Schul. Vorbildung:
Kommt von Schule:
Sprachprobleme:



Phase 1: Access 97 - Alles in einem Formular

The screenshot displays the Microsoft Access 97 interface. On the left, a 'Schülerliste' (Student List) table is visible, containing names and surnames. The 'Stelder Andre' entry is highlighted. On the right, a form titled 'Daten des Schüler' (Student Data) is open, showing fields for personal information, contact data, and statistics. The form is divided into several sections: 'Kontaktdaten' (Contact Data), 'Primäre Statistikdaten' (Primary Statistical Data), and 'Schülerhintergrund' (Student Background). The form contains numerous fields, including text boxes, dropdown menus, and radio buttons. A 'Kopieren vom 1. Sorgeberechtigten' (Copy from 1st Guardian) button is located at the bottom of the form.

Name	Nachname
Abadian	Florian
Abanon	Kai-Uwe
Abba	Julia
Aber	Linda
Aber	Tobias
Aberes	Klaus Michael
Beispiel	Markus
Ebertor	Bianca
Muster	Marion
Stein	Alexander
Stelder	Andre
Stolski	Alice
Strang	Wilhelm
Ukar	Marcell
Ukhar	Stefan
Ukuler	Jasmin
Ukunos	Benid
Ukuz	Alexander
Ulbricht	Sabrina
Ülunkir	Christoph
Ümaker	Stephan
Ümakun	Witali
Umbauer	Ralf
Ümber	Denise/Toma
Ümunk	Birgit
Ümun	Severin
Ümürer	Simon
Ümürta	Silvia
Ümut	Mike
Ümuth	Sandra
Ümütur	Florian
Ündola	Sabrina
Ünkert	Christian
Ünterhard	Michael
Ürbusch	Alexander
Ürkhart	Andreas
Ürmisch	Kristina
Ürutski	Daniel
Üter	Natascha
Ütkron	Jessica
Üturq	Jasmin

11 Register

LISTBOX mit Schülern

Pro Register ca 15-20 Felder

Darunter bis zu 10 Combo-Boxen

mit eigener Datenquelle und/oder Unterformulare



Performance-Probleme und Lösung beim Umstieg auf 2007

Symptom: Im Netzwerk wird das ganze Formular sehr langsam.

Festgestellte Ursachen:

- a) Vielzahl von Listboxen mit Zuweisung über VBA-Code

FillcomboBox
cbxStaatsangehörigkeit...

- b) Datenquelle des Formulars über einzelnen Datensatz zuweisen:

Me.Recordsource = ...
WHERE ID=.....

Abhilfe in 2007:

a) Datenquelle der Listboxen direkt im Steuerelement hinterlegen. Keine VBA Zuweisung

b) Für Formular die komplette Tabelle als Datenquelle hinterlegen und dann mit RecordsetClone arbeiten:

```
Set rst = Me.RecordsetClone
```

```
rst.FindFirst "ID=...."
```

```
Me.Bookmark =rst.Bookmark
```



Phase 2: Entkoppeln von Listbox vom Datenform

The screenshot displays two windows from a software application. The left window, titled 'Schüler Liste', shows a list of active students with search filters and a table of names and class numbers. The right window, titled 'Daten des Schüler', shows a detailed form for a selected student, 'Ukar Marcell', with fields for personal data, contact information, and primary statistics.

Schüler Liste (Left Window):

Schüler: Aktive

Namensuche...

Klassensuche...

Ein Click Modus

Markieren ? AKT Anordnen

Abadian	Florian	S 7a
Abanon	Kai Uwe	S 9a
Abba	Julia	S 9b
Aber	Linda	B 9c
Aber	Tobias	S 7b
Aberes	Klaus Michael	S 8a
Stein	Alexander	B 8c
Stelder	Andre	S 10b
Stolski	Alice	S 8a
Strang		
Ukar		
Ukhardt		
Ukuler		
Ukunos		
Ukuz		
Ulbricht		
Ülünkir		
Ümaker		
Ümakun		
Umbauer		
Ümberg		
Ümunkel		
Ümunkel		
Ümürer		
Ümürta		
Umut		
Umuth		
Ümütur		
Undola		
Unkert		
Unterhard		
Urbusch	Alexander	8b
Urkhardt	Andreas	B 7c
Urnisch	Kristina	S 8a

Daten des Schüler (Right Window):

Name: Ukar
Vorname: Marcell
Nr: 57
Klasse: 5c
Status: Aktiv

Aktuelles Jahr
 Kommendes Jahr

Personendaten | **Sorgeberechtigte** | Statistik 1 | Statistik 2 | Sprachen +MEU | Behinderung | Joker | Merkmale | Notiz | Historie | Kurse

Kontaktdaten

Name: Ukar
Vorname: Marcell
Anrede:
Titel:

Primäre Statistikdaten

GebDat: 11.05.2001
GebOrt: Musterstadt
Geschlecht: m
StaatsAng: Deutschland

Listbox mit Schülern in einem eigenen
Formular
-> Entkoppeln der Listbox vom Daten-
Formular



Phase 3: Optionsgruppe und UFos

Optionsgruppe ersetzt Registerkarten

Ein Unterformular-Rahmen bildet Container für zu ladende Unterformulare je nach Optionsauswahl

Format	Daten	Ereignis	Andere	Alle
Herkunftsobjekt				frmSchuelerSubSprachen
Verknüpfen nach	Schueler_Nr			
Verknüpfen von	Schueler_Nr			
Leeren Hauptentwurf filtern	Ja			
Aktiviert	Ja			
Gesperrt	Nein			



Phase 3: Erreichte Vorteile

- Die Anzeige der Listbox zur Schüлераuswahl ist getrennt von der Detail-Anzeige eines Schülerdatensatzes
- Dadurch, dass das Register-Steuererelement in eine Optionsgruppe umgewandelt wurde, werden nur die Steuerlemente des jeweiligen Unterformulars geladen

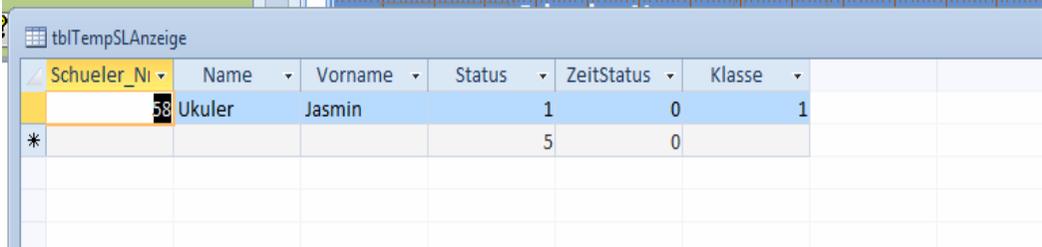
Reduktion von 200 -> 10-15 Steuererelemente



Phase 4: Hauptform auf TempTab & Events

Datenquelle für Hauptform auf TempTab

- Das Hauptform soll nur eine kleine Tabelle laden, die die Kerndaten (ID, Name, Vorname) eines Schülers aufnimmt.



The screenshot shows a data table with the following columns: Schueler_Nr, Name, Vorname, Status, ZeitStatus, and Klasse. The first row is highlighted in blue and contains the values 58, Ukuler, Jasmin, 1, 0, and 1. A second row is partially visible with an asterisk in the first column and the value 5 in the Status column.

Schueler_Nr	Name	Vorname	Status	ZeitStatus	Klasse
58	Ukuler	Jasmin	1	0	1
*			5	0	

- Aktualisierung von Hauptformular und Unterformulare(n) über **Event-Benachrichtigung**, wenn neuer Schüler ausgewählt wird
 - Vereinfachung der Programmierung
 - Direkte Kopplung zum Haupt & Unterformular entfällt



Phase 5: Security-Prüfung über Collections

Security Prüfung vorher

- Nachsehen in Tabellen, ob Berechtigung entsprechend eingetragen, bei jedem Schülerwechsel

Änderung in Phase 5.

- Einlesen der Berechtigungen des Users in Collections bei Anmeldung
- Security Prüfung erfolgt nur noch über eine Anweisung. „Ist in Collection“



Finale performante Lösung

The image shows a screenshot of a school management software interface. On the left, a window titled 'Schüler Liste' displays a list of students with columns for name, class, and other details. A blue box highlights the list, with the text 'Listbox getrennt von Details' overlaid. On the right, a window titled 'Daten des Schüler' shows the detailed record for a student named 'Ulbricht'. A blue box highlights the name and ID, with the text 'Daten aus TmpTab' overlaid. Below the name, there are several tabs for different data categories: 'Personendaten', 'Sorgeberechtigte', 'Sprachen+MEU', 'Kurse/Unterricht', 'Behinderung', 'Jokerfelder', 'Merkmale', and 'Notiz'. The 'Personendaten' tab is active, showing fields for 'Name', 'Vorname', 'Anrede', 'Titel', 'Straße', 'PLZ', 'Ort', 'Wohnortkey', 'Bundesland', and 'Wohnstaat'. A blue box highlights the 'Wohnortkey' field, with the text 'Unterformulare pro Option' overlaid. Below the 'Wohnortkey' field, there is a blue arrow pointing to the 'Wohnortkey' field, with the text 'Events zur Aktualisierung' overlaid. On the right side of the 'Personendaten' tab, there are 'Primäre Statistikdaten' fields for 'GebDat', 'GebOrt', 'Geschlecht', 'StaatsAng', 'Religion', and 'TeilAnRegInt'. Below these, there are 'Schulhintergrund' fields for 'Schulbesuchsjahr', 'Einschulungsjahr', 'Schul. Vorbildung', 'Kommt von Schule/Kiga', and 'Sprachprobleme'.

Listbox getrennt von Details

Daten aus TmpTab

Unterformulare pro Option

Events zur Aktualisierung



Nachteile

- Verteilung auf verschiedene Subforms erhöht Suchaufwand in Access-Navigationsleiste
- Die Fehlersuche ist erschwert
- Synchronisationsprobleme zwischen Hauptformular und Unterformularen, je nach Programmzustand

Vorteile

- Deutliche und wahrnehmbare Performance-Verbesserung, insbesondere im Netz-Einsatz
- Die Programmlogik ist vereinfacht für Hauptform und Subforms



Bücher von André Minhorst

- Access 2010 Grundlagenbuch für Entwickler
- Access 2007 Das Praxisbuch für Entwickler



Fragen ?

Vielen Dank !