Daniel Bühling

SolidWorks Objekte und Dokumente

API Grundlagen und Dokumenthandling

3., komplett verbesserte und erweiterte Auflage

SolidWorks API-Programmierung
mit Visual Basic 2005 und SolidWorks 2007

Dezember 2007

Schulungsskript der Schuler Design Automation GmbH
Einführung – Erklärung - Nachschlagwerk

Die Informationen in diesem Skript werden ohne Rücksicht auf einen eventuellen Patentschutz veröffentlicht. Warennamen werden ohne Gewährleistung der freien Verwendbarkeit benutzt. Bei der Zusammenstellung von Texten und Abbildungen wurde mit größter Sorgfalt vorgegangen. Trotzdem können Fehler nicht vollständig ausgeschlossen werden.
Herausgeber und Autor können für fehlerhafte Angaben und deren Folgen weder juristische Verantwortung noch irgendeine Haftung übernehmen. Für Verbesserungsvorschläge und Hinweise auf Fehler sind Herausgeber und Autor dankbar.

Schuler Design Automation GmbH
Nimrodstraße 9
Gebäude 3, 1. Stock
D- 90441 Nürnberg

www.Schuler-DA.de
kontakt@schuler-da.de

Vorwort:

In diesem Skript erhalten Sie Grundkenntnisse in der SolidWorks Objektstruktur.
Sie lernen, wie Sie SolidWorks Objekte richtig deklarieren und initialisieren, damit für Ihre Aufgaben die richtigen Methoden, Ereignisse und Eigenschaften zur Verfügung stehen.
Dabei werden Ihnen Tipps und Tricks im richtigen Umgang mit den Objekten und der „SolidWorks und Zusatzanwendungen-API-Hilfethemen“ vermittelt, um effizient Programmieren zu können
Als weiteren Schwerpunkt wird Ihnen gezeigt, wie Sie Ihre SolidWorks Dokumente effektiv über die API handeln können.
Neben den Grundlagen, wie „Öffnen“, „Speichern“ und „Speichern unter“ eines Dokuments, werden Sie lernen, wie Sie an weitere Informationen eines Dokuments gelangen, um z.B. Referenzen zu bearbeiten.

Ziel dieses Skripts ist es Ihnen zu vermitteln, wie Sie mit der Programmiersprache Visual Basic eigene Anwendungen für SolidWorks erzeugen und die erforderlichen SolidWorks Objekte für Ihre Aufgabe nutzen.
Das Skript soll Ihnen zudem zeigen, wie Sie SolidWorks Dokumente über die API Schnittstelle bearbeiten und diese effektiv verwalten.

In diesem Skript wird eine Objektorientierte Programmierung mit Hilfe der Programmsprache Visual Basic 8 und dem Net Framework 2.0 angewendet. Nicht an allen Stellen in diesem Skript kann auf die Vorgehensweise sowie den Methoden und Eigenschaften des Net Frameworks eingegangen werden.
Aus diesem Grund ist es erforderlich Erfahrungen in diesem Themen der Programmierung mitzubringen, damit Sie dieses Skript sinnvoll durcharbeiten können.

Inhaltsverzeichnis:

1. Teil 1
Hallo SolidWorks, hier bin ich!
1.1. Projektverweise und ihre Bedeutung 7
1.2. Die SolidWorks Objektstruktur 10
1.3. Die „SolidWorks API-Hilfethemen“ 12
1.4. Die SolidWorks ProgId und Registryschlüssel 14
1.5. SolidWorks durch CreateObject belegen 16
1.6. SolidWorks durch GetObject belegen 29
1.7. CreateObject oder GetObject 34
1.8. Die richtige SolidWorks Version? 37
1.9. Die SwConst Library 39

2. Teil 2
Auf SolidWorks reagieren
2.1. SolidWorks Ereignisse verfügbar machen 45
2.2. Auf das Öffnen eines Dokuments reagieren 46
2.3. Auf das Speichern eines Dokuments reagieren 50
2.4. Ein Ereignis in SolidWorks stoppen 55

3. Teil 3
Dokumenthandling in SolidWorks
3.1. Arbeiten mit den Net-Klassen System.IO 57
3.2. Ein Dokument in SolidWorks öffnen 63
3.3. Ein Dokument in SolidWorks schließen 70
3.4. Das Speichern eines Dokuments 75
3.5. „Speichern unter“ eines Dokuments 79
3.6. Das Erzeugen eines Schnittstellenformats 83
3.7. Ermitteln des Speicherorts eines Dokuments 84
3.8. Ermitteln der Referenzen eines Dokuments 86
3.9. Ändern einer Referenz 98
3.10. Dokumenteigenschaften ermitteln 101
3.11. Dokumenteigenschaften manipulieren 111

4. Teil 4

Nützliche Tricks beim Arbeiten mit der SolidWorks API

4.1. Aufnahmen des Makrorekorders 117
4.2. Programmieren in der SolidWorks VBA 119
4.3. SolidWorks Optionen 121
4.4. Option “Standard Makroverzeichnis” ändern 124
4.5. Weitere Beispiele und Informationen über die SolidWorks API 126

5. Anhang

A Der Code des SldWorksDokument Projekt im Überblick 127
B Stichwortverzeichnis SolidWorks API Befehle 148

1 Hallo SolidWorks, hier bin ich!

SolidWorks bietet Automatismen, mit welchen Sie lästige und wiederkehrende Aufgaben effektiv lösen können.

Das Programmieren von neuen Funktionen oder das Zusammenfügen von mehreren Abläufen ist durch eine Visual Basic Anwendung ebenso möglich, wie das Erstellen von kompletten Bauteilen, Baugruppen und Zeichnungen.

Die SolidWorks Schnittstellen stellen fast die komplette SolidWorks Funktionalität dar, welche Sie ohne weitere Lizenzkosten in einem Visual Basic Programm verwenden können.

1.1 Projektverweise und Ihre Bedeutung

Damit Ihnen die SolidWorks API Methoden, Konstanten und Schnittstellen zur Verfügung stehen, benötigt ein Visual Basic Projekt verschiedene Projektverweise.

Durch einen Projektverweis zu einer Type Library stehen Ihrem Projekt neue Funktionen zur Verfügung.

SolidWorks bietet mehrere Type Librarys, welche Sie in einem Projekt verwenden können. Die wichtigsten Projektverweise sind:

Name	Pfad
SldWorks Type Library	..\SolidWorks\sldworks.tlb
SolidWorks constant type library	..\SolidWorks\swconst.tlb
SolidWorks exposed type libraries for add-in use	..\SolidWorks\swpublished.tlb

Tabelle 1.0
Die wichtigsten SolidWorks Verweise

Betrachten Sie die einzelnen Type Librarys genauer.

SldWorks Type Library
Die „SldWorks Type Library“ stellt Ihnen die SolidWorks API Funktionen zur Verfügung, welche in einer übersichtlichen Objektstruktur gegliedert sind.

Um die Funktionen zu verwenden, benötigen Sie eine Objektinstanz des SolidWorks Objekts, welches die Funktion enthält.

Mit den SolidWorks Objekten und der SolidWorks Objektstruktur werden Sie sich im weiteren Verlauf dieses Skripts beschäftigen und diese an vielen Beispielen verdeutlichen.

SolidWorks constant type library
In dieser Type Library stellt Ihnen SolidWorks verschiedene Enumerationsvariablen zur Verfügung.

Diese Enumerationsvariablen bieten Ihnen die Möglichkeit Ihren Programmcode lesbar zu gestalten und verschiedene Optionen übersichtlich in Worte darzustellen.

SolidWorks exposed type libraries for add-in use
Die Type Libraries for Add-in use stellt Ihnen Schnittstellen zur Verfügung, welche Sie in Ihren Projekt verwenden können.

Wenn Sie eine solche Schnittstelle in Ihren Visual Basic Projekt einfügen, stehen Ihnen Schnittstellenfunktionen zur Verfügung, mit welchen Sie zum Beispiel ein SolidWorks Add-In erstellen können.

Für das Projekt dieser SolidWorks API Grundlagenschulung benötigen Sie die beiden Projektverweise „SldWorks Type Library“ und „SolidWorks constant type library“.

Öffnen Sie ein neues „Visual Basic WindowsApplication“ Projekt. Diesem neuen Projekt geben Sie den Namen „SldWorksDokument“.
In den Projekteinstellungen unter „My Projekt“ des „Projektmappen-Explorers“ fügen Sie in das Verweis Register die benötigten Projektverweise ein.
Sollte der „Projektmappen-Explorer“ in Visual Studio nicht sichtbar sein, können Sie diesen durch den Menüeintrag “Ansicht > Projektmappen-Explorer“ oder der Tastenkombination „Strg + Alt + L“ anzeigen.

Öffnen Sie die Projekteinstellungen, indem Sie im „Projektmappen-Explorer“ auf „My Projekt“ klicken und in das Register „Verweis“ wechseln.
In diesem Register klicken Sie auf „Hinzufügen“ und wechseln im darauf folgenden Dialog „Verweis hinzufügen“ zu dem Register „COM“. Die SolidWorks Verweise sind in diesem Register vorhanden, da es sich bei den Type Libraries um ActiveX Komponenten handelt. Seit der Version SolidWorks 2007 stehen diese „COM“ Komponenten auch als Net Klassen zur Verfügung.

Wählen Sie in der Liste die bereits erwähnten Type Libraries aus und klicken auf „Ok“.
Nun stehen Ihnen die SolidWorks API Methoden und Konstanten in Ihrem Projekt zur Verfügung.

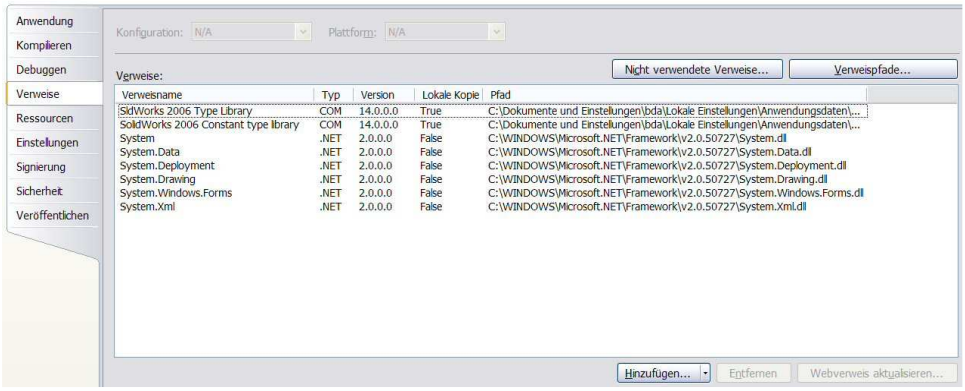


Abbildung 1.0 Hinzugefügte SolidWorks Verweise im Visual Studio

1.2 Die SolidWorks Objektstruktur

Einen Überblick über die SolidWorks Objektstruktur bekommen Sie durch den Objektbrowser im Visual Studio. Drücken Sie auf die Funktionstaste F2, oder wählen Sie im Menü „Ansicht > Objektbrowser“ aus.

Im Objektbrowser werden Ihnen alle Klassen angezeigt, welche Ihnen in Ihrem Visual Basic Projekt zur Verfügung stehen. Neben den Net Framework Namespaces System und dessen Klassen befinden sich dort auch die Namespaces SldWorks und SwConst, welche Sie durch die Projektverweise eingebunden haben.

Wenn Sie den Namespace SldWorks anklicken und dort wiederum die Klasse SldWorks auswählen, erhalten Sie einen Überblick über alle Methoden, Events und Propertys dieser Klasse.

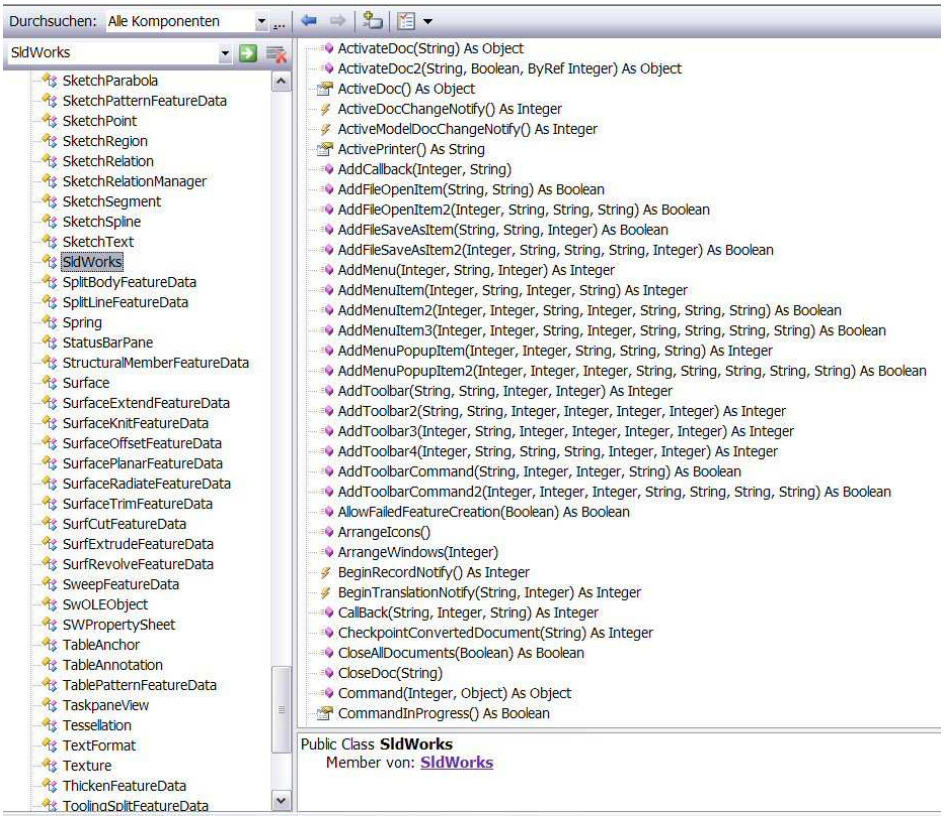


Abbildung 1.1
Objektbrowser in Visual Studio

Diese Klasse stellt die SolidWorks Application dar, mit welcher Sie in SolidWorks die weiteren benötigten Objekte abfragen und initialisieren können.
Diese Klasse ist für jedes SolidWorks API Programm der Einstieg. Deshalb werden Sie sich mit dieser Klasse noch umfangreich in dieser SolidWorks API Grundlagenschulung beschäftigen.

Auch die SolidWorks API Hilfe, mit welcher Sie sich im nächsten Kapitel beschäftigen, bietet uns eine Übersicht der SolidWorks Objektstruktur.

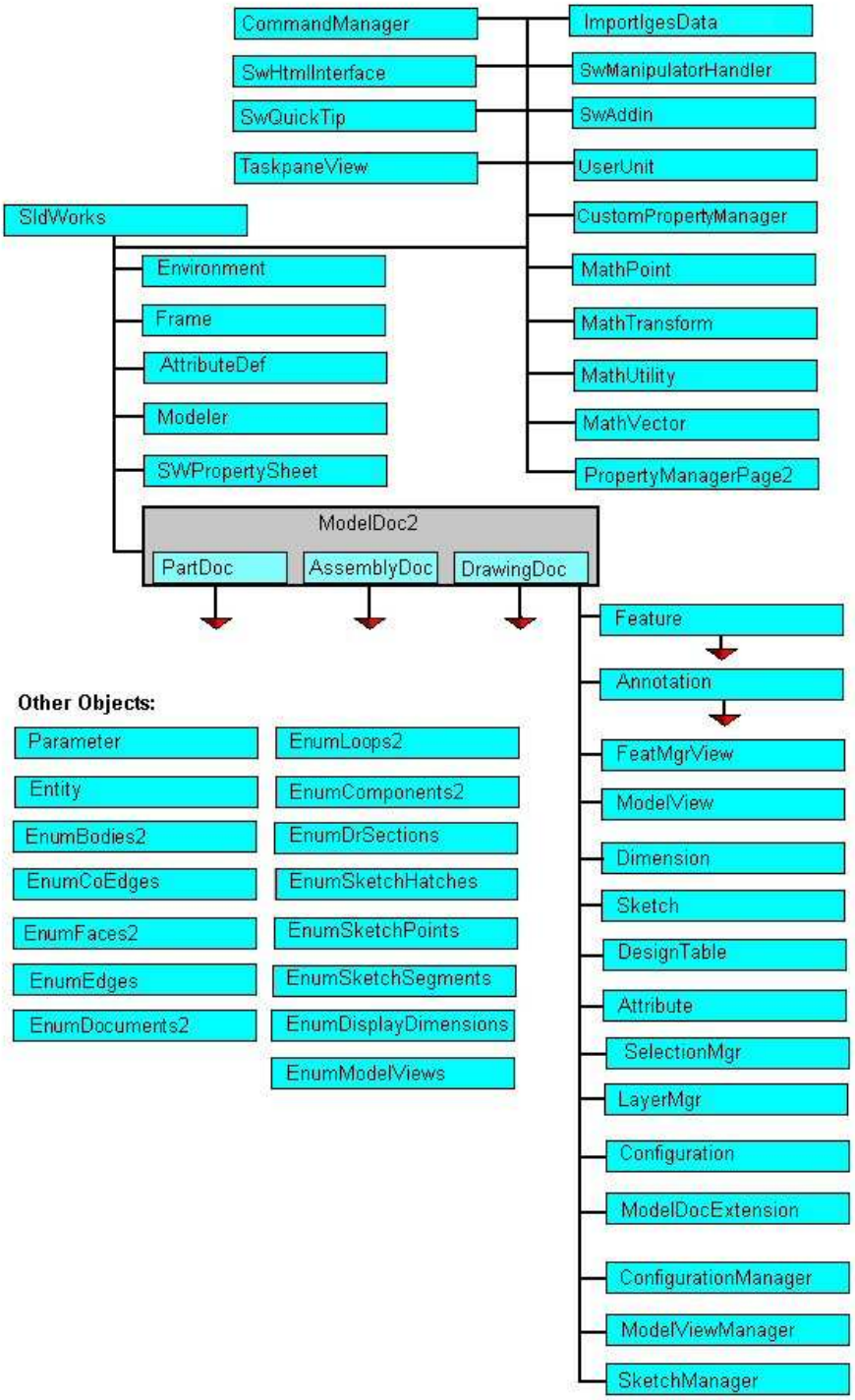


Abbildung 1.2
SolidWorks Objektstruktur als Diagramm

Dieses Objekt Diagramm zeigt die wichtigsten SolidWorks Objekte (Klassen) und Ihre Zusammengehörigkeit.
Es zeigt uns, wie die Objekte in gegenseitiger Verbindung stehen.

Die wichtigsten Objekte dieser SolidWorks API Grundlagenschulung sind die Objekte: SldWorks, ModelDoc2, PartDoc, AssemblyDoc und DrawingDoc.

Wie man am Beispiel des DrawingDoc Objekts erkennen kann, ist dieses Mitglied des ModelDoc2 Objekts, welches sich wiederum vom SldWorks Objekt ableitet. Auf diesen Zusammenhang wird im Kapitel 2 näher eingegangen.

1.3 Die „SolidWorks und Zusatzanwendungen-API-Hilfethemen“

Die SolidWorks API Hilfe ist die umfangreichste Quelle über die SolidWorks API Programmierung.
Kein Buch und keine Internetquelle bieten mehr Informationen und Beispiele als die API Hilfe. Alle API Methoden, Konstanten und Schnittstellen sind beschrieben und übersichtlich geordnet.
Damit Sie sich in der Online Hilfe zu Recht finden folgen ein paar Tipps:

Hilfe Themen dieser Online Hilfe können über das Inhalt-, Index- und Suchen-Register aufgerufen werden.
Im Inhalt Register ist das für Sie interessante Thema die SolidWorks API Help.
In dieser gibt es viele Codebeispiele im Unterthema „Examples and Projects“. Die SolidWorks Objekte sind übersichtlich im Unterthema APIs sortiert. Hier sind die Methoden, Events (Ereignisse), Propertys (Eigenschaften) und Schnittstellenfunktionen aller SolidWorks Objekte ausführlich mit Parameter, Rückgabewert und Hinweis beschrieben.

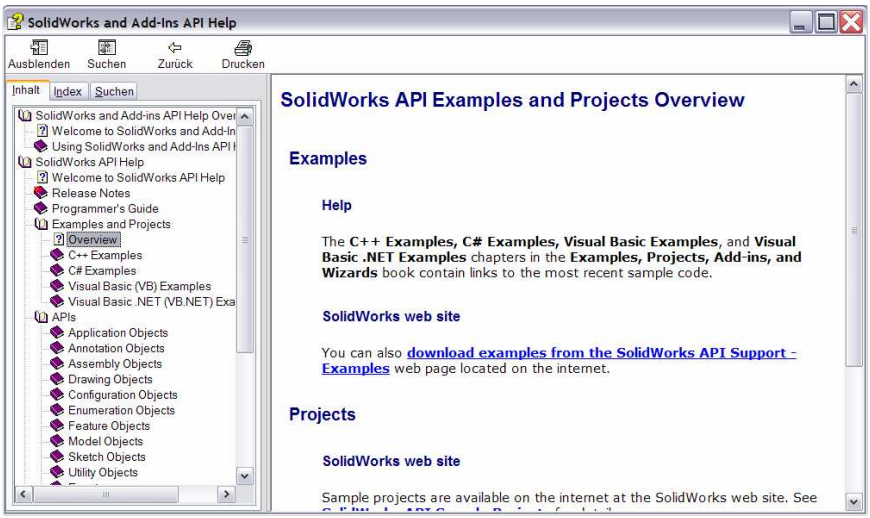


Abbildung 1.3 SolidWorks API Hilfe. Register Inhalt

Das Index Register eignet sich hervorragend, um ein bestimmtes Objekt zu durchsuchen. Wenn Sie zum Beispiel das SldWorks Objekt durchsuchen möchten, reicht es aus,wenn Sie „sld“ in das Textfeld eingeben. Wie Sie sehen, werden alle Inhalte des SldWorks Objekts übersichtlich angezeigt.

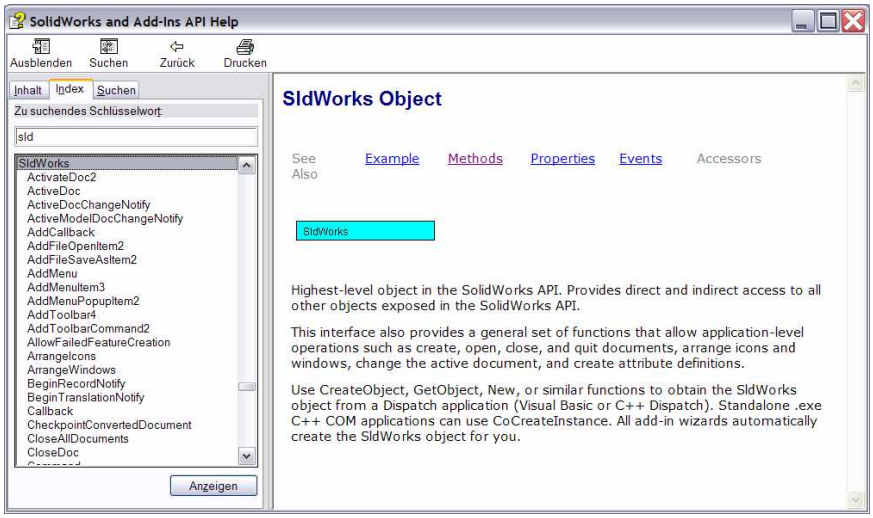


Abbildung 1.4 SolidWorks API Hilfe. Register Index

Im Suchen Register können Sie nach Befehlen oder bestimmten Begriffen suchen.
Eine Suche nach „Open“ wird Sie zum Beispiel zur OpenDoc6 Methode im SldWorks Objekt führen.

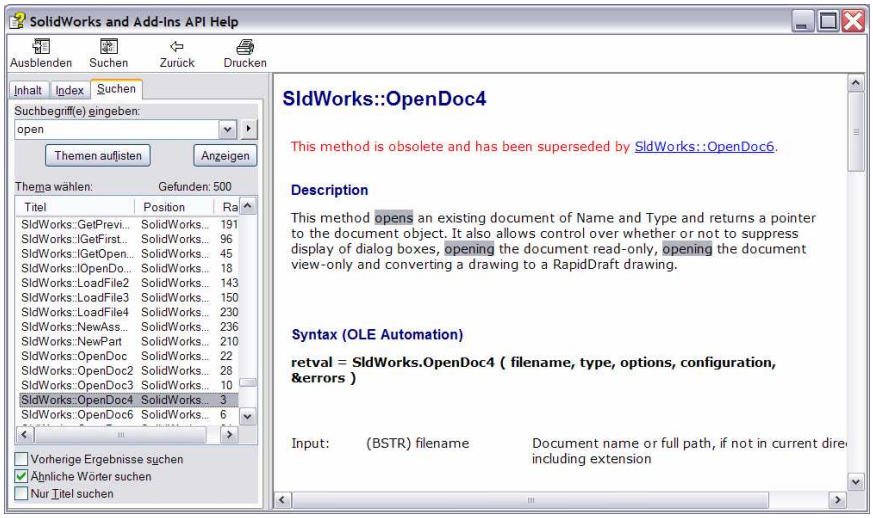


Abbildung 1.5 SolidWorks API Hilfe. Register Suchen

Das Beispiel „OpenDoc“ zeigt uns auch, dass in der Online Hilfe neben aktuellen auch alle alten Befehle aufgeführt werden. Dies wird Ihnen immer durch den Hinweis „This method is obsolete and has been superseded by ..“ im oberen Bereich des Hilfe Themas mitgeteilt.
Bedenken Sie, dass alte API Befehle nicht ungültig sind. Meist sind nur andere oder mehr Parameter der Grund für eine neue Methode.

1.4 Die SolidWorks ProgId und Registryschlüssel

Damit Sie in einem externen Visual Basic Projekt auf SolidWorks zugreifen können, benötigen Sie eine Objektinstanz zur SolidWorks Application.

Es gibt in VB.Net 2005 zwei Möglichkeiten eine solche Instanz zu erstellen. Die benötigten Methoden sind in den Net Klassen „Microsoft.VisualBasic“ enthalten und heißen „CreateObject“ und „GetObject“.

Bevor Sie sich mit diesen Methoden näher beschäftigen, schauen Sie die SolidWorks Registryeinträge in der Windows Registry, für ein besseres Verständnis, genauer an.

Alle verfügbaren SolidWorks Type Librarys wurden bei der Installation von SolidWorks auf Ihrem System registriert. Diese Registrierung erzeugt in der Windows Registry die benötigten Einträge.
Unter anderen können Sie in der Windows Registry die Klassen ID (CLSID, eindeutige Identifizierung der Klasse) der SolidWorks Type Library abfragen.
Diese Klassen ID ermitteln Sie am einfachsten mit Hilfe der ProgID (Namespace und Klassenname) der SolidWorks Type Library „SldWorks.Application“.

Öffnen Sie den Registrierungs-Editor, indem Sie zum Beispiel „RegEdit“ in das Windows Ausführenfenster eingeben.
Suchen Sie im Registrierungs-Editor nach „SldWorks.Application“. Für die Suche klicken Sie bitte auf den Registryschlüssel „HKEY_CLASSES_ROOT“. Starten Sie die Suche durch Strg+F oder im Menü „Bearbeiten“. Geben Sie im Suchen Dialog „SldWorks.Application“ ein und klicken Sie auf „Weitersuchen“.

Abbildung 1.6
Suchen Dialog des Registrierungs-Editor

Klicken Sie auf „Weitersuchen“ oder F3 bis Sie den Registryschlüssel „SldWorks.Application“ angezeigt bekommen. (Alternativ können Sie auch in der Baumstruktur des „HKEY_CLASSES_ROOT“ Schlüssel nach „SldWorks.Application“ suchen, indem Sie mit dem Scrollbalken zu diesem Schlüssel scrollen.)

Wenn Sie diesen Registryschlüssel gefunden haben sollten Sie ähnliches im Registrierungs-Editor angezeigt bekommen.

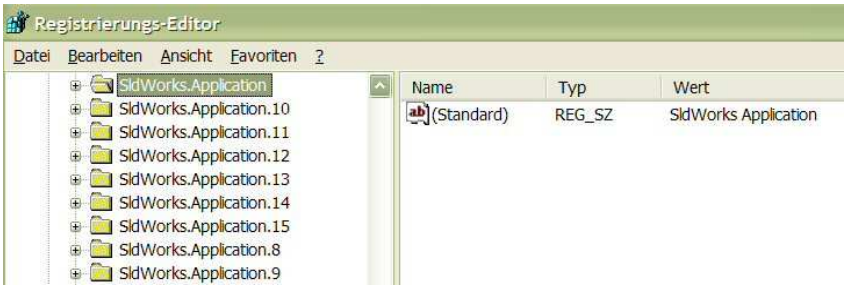


Abbildung 1.7
SolidWorks ProgId Registryschlüssel im Registrierungs-Editor

Sie sehen, dass es mehrere „SldWorks.Application“ Schlüssel gibt. Die Erklärung ist einfach.
Der Registryschlüssel „SldWorks.Application“ steht für die zuletzt auf Ihrem System gestartete SolidWorks Version.
Die Schlüssel „SldWorks.Application.x“ stehen für eine bestimmte SolidWorks Version.

Tabelle 1.1
SolidWorks ProgId und Ihre Versions-zugehörigkeit

Registryschlüssel	SolidWorks Version
SldWorks.Application.11	SolidWorks 2003
SldWorks.Application.12	SolidWorks 2004
SldWorks.Application.13	SolidWorks 2005
SldWorks.Application.14	SolidWorks 2006
SldWorks.Application.15	SolidWorks 2007
SldWorks.Application.16	SolidWorks 2008

Keine Angst, wenn in Ihrer Registry SolidWorks Versionsschlüssel fehlen. Diese sind von den installierten SolidWorks Versionen und verwendeten SolidWorks Add-Ins abhängig.

Der wichtige Eintrag in diesen Registryschlüssel ist der Unterschlüssel „CLSID“.
Dieser enthält die Information auf welche Type Library Ihr System zugreift, da eine CLSID (Klassen ID) eine Klassenbibliothek eindeutig auf Ihrem System definiert.

Betrachten Sie zur Verdeutlichung den Schlüssel „SldWorks.Application“ und dessen „CLSID“ Unterschlüssel. Mit Hilfe dieses Eintrags können Sie ermitteln, welche SolidWorks Version zuletzt gestartet wurde, indem Sie die CLSID mit denen der Versionsschlüssel vergleichen.

Die CLSID Einträge der Registryschlüssel „SldWorks.Application“ und „SldWorks.Application.14“ sind identisch.

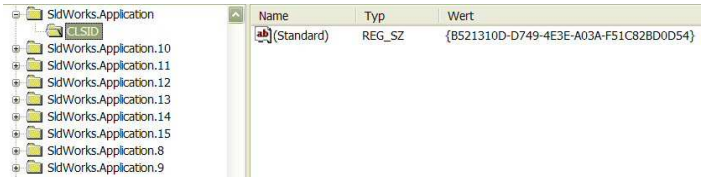


Abbildung 1.8 CLSID Eintrag der SolidWorks ProgId SldWorks. Application

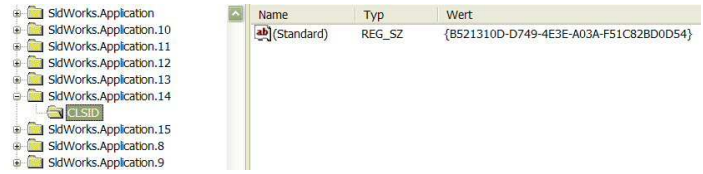


Abbildung 1.9 CLSID Eintrag der SolidWorks ProgId SldWorks. Application.14

Dies bedeutet zum Beispiel, dass die SolidWorks Version 2006 zuletzt gestartet wurde.

Mit diesem wichtigen Hintergrundwissen können Sie sich endlich dem Visual Studio zuwenden und die ersten Zeilen Programmcode eingeben.

1.5 SolidWorks durch CreateObject belegen

Bevor Sie in die SolidWorks API einsteigen, bereiten Sie das bereits geöffnete Projekt auf die kommenden Aufgaben vor.
Durch das Öffnen eines „WindowsApplication“ Projekts wurde automatisch das Formular „Form1“ hinzugefügt.
Dieses benennen Sie in „SchulungFrm“ um. Als nächstes fügen Sie in das Projekt eine Klasse ein und nennen diese „MySldWorksCls“.
In das Formular „SchulungFrm“ fügen Sie ein Label und zwei Buttons ein und ändern die folgenden Eigenschaften:

SchulungFrm:

Eigenschaft	Wert
ShowIcon	False
Size	400; 250
Text	SolidWorks API Schulung

Tabelle 1.2 Geänderte Eigenschaften der SchulungFrm

Label1:

Eigenschaft	Wert
Name	lblUeberschrift
AutoSize	False
Size	360; 60
Text	Schuler Design Automation GmbH SolidWorks API Schulung SolidWorks Objekte und Dokumente
TextAlign	MiddleCenter

Tabelle 1.3 Geänderte Eigenschaften des Label1

Button1:

Eigenschaft	Wert
Name	cmdGet
Size	130; 30
Text	GetObject

Tabelle 1.4 Geänderte Eigenschaften des Button1

Button2:

Eigenschaft	Wert
Name	cmdCreate
Size	130; 30
Text	CreateObject

Tabelle 1.5 Geänderte Eigenschaften des Button2

Diese Steuerelemente richten Sie im Formular aus, damit dieses ungefähr so aussieht.

Abbildung 1.10 Das SchulungFrm Fenster

Wie bereits erwähnt, benötigen Sie in einem externen Visual Basic Projekt eine Objektinstanz zur SolidWorks Application.
Für diese Objektinstanz benötigen Sie ein deklariertes SolidWorks Objekt. Das SolidWorks Application Objekt wird folgendermaßen deklariert:

```
Dim oSwApp As SldWorks.SldWorks
```

Code 1.0 Deklaration des SolidWorks Application Objekts

In diesem Objekt wir eine Instanz der SolidWorks Application gespeichert. Dafür stehen Ihnen die bereits erwähnten Methoden zur Verfügung. Beginnen Sie mit der Methode „CreateObject“.

Die Methode „CreateObject“ rufen Sie durch einen Klick auf den Button „cmdCreate“ auf. Deshalb benötigen Sie eine Ereignisprozedur für das Klick Ereignis des Buttons und eine Funktion, welche in der Ereignisprozedur aufgerufen wird.

Die Ereignisprozedur fügen Sie am einfachsten durch einen Doppelklick auf den Button im Visual Studio Designer hinzu.

```
Private Sub cmdCreate_Click( _
            ByVal sender As System.Object, _
            ByVal e As System.EventArgs) _
            Handles cmdCreate.Click

End Sub
```

Code 1.1
Ereignishandler des cmdCreate.Click Ereignis

Wechseln Sie in die Klasse „MySldWorksCls“ und deklarieren Sie dort eine Klassenweit verfügbare Variable „oSwAppCls“ vom Typ „SldWorks.SldWorks“.
Die Endung „…Cls“ wird vom Autor verwendet um Klassenweit verfügbare Variablen zu kennzeichnen.

```
Dim oSwAppCls As SldWorks.SldWorks
```

Code 1.2
Deklaration des Klassenweit verfügbaren SolidWorks Application Objekts

Diese Variable „oSwAppCls“ verwenden Sie in einer nur lesen Eigenschaft (Property) der Klasse „MySldWorks“, mir welcher der Inhalt dieser Variablen abgefragt und verwendet werden kann. Damit diese Eigenschaft von anderen Klassen Ihres Projekts verwendet werden kann, besitzt diese die Zugriffsebene „Friend“.

```
    Friend ReadOnly Property MySldWorks() _
                    As SldWorks.SldWorks
        Get
            Return oSwAppCls
        End Get
    End Property
```

Code 1.3
MySldWorks Eigenschaft, mit welcher der Inhalt der oSwAppCls abgefragt und verwendet werden kann

In die Klasse „MySldWorksCls“ fügen Sie darauf die Methode „SolidWorksInstanzCreate“ vom Typ SldWorks.SldWorks ein.

In dieser Methode fügen Sie eine Fehlerbehandlung (Try-Catch-Anweisung) ein.
Durch die Fehlerbehandlung werden Exceptions (Fehler) in eine MessageBox geleitet, in welcher Sie die Fehlermeldung und den Fehlerort anzeigt bekommen. Diese Art der Fehlerbehandlung wir in allen weiteren Methoden und Prozeduren verwendet.

Code 1.4
Funktionsrahmen der Methode SolidWorksBelegen Create

```
Friend Function SolidWorksInstanzCreate() _
                        As SldWorks.SldWorks

   Try

       Return oSwAppCls
   Catch ex As Exception
       Debug.Assert(False)
       MsgBox("Fehler: Wo: " & _
                ex.StackTrace & " Was: " & ex.Message)
       Return Nothing
   End Try
End Function
```

Als Rückgabewert dieser Methode wird die Klassenvariable „oSwAppCls“ Variable verwendet. Tritt ein Fehler in dieser Methode auf, wird der Rückgabewert Nothing festgelegt.

Die Methode „SolidWorksInstanzCreate“ rufen Sie in der Ereignisprozedur des „cmdCreate.Click“ Ereignis mit Hilfe einer Objektinstanz der Klasse „MySldWorksCls“ auf.

Code 1.5
Aufruf der Methode SolidWorksBelegen Create im Ereignishandler des cmdCreate.Click Ereignis

```
Private Sub cmdCreate_Click( _
            ByVal sender As System.Object, _
            ByVal e As System.EventArgs) _
            Handles cmdCreate.Click
    Try
        'Eigene SolidWorks Klasse deklarieren
        Dim oMySldWorks As MySldWorksCls
        'Eigene SolidWorks Klasse initialisieren
        oMySldWorks = New MySldWorksCls
        'Create Methode ausführen
        oMySldWorks.SolidWorksInstanzCreate()
        'Eigene SolidWorks Klasse leeren
        oMySldWorks = Nothing
    Catch ex As Exception
        Debug.Assert(False)
        MsgBox("Fehler: Wo: " & _
                ex.StackTrace & " Was: " & ex.Message)
    End Try
End Sub
```

Beschäftigen Sie sich nun mit der Methode „SolidWorksInstanzCreate“. In dieser werden Sie die Besonderheiten und Eigenschaften der „CreateObject“ Methode ausprobieren.
Betrachten Sie die Methode „CreateObject“ zuerst etwas genauer. Der Visual Basic Hilfe können Sie die benötigten Informationen entnehmen:

Visual Basic
CreateObject
Befehlsreferenz

Mitglied der Net Klasse *Microsoft.VisualBasic*

Public Shared Function CreateObject(_
ByVal ProgId *As String, _*
Optional ByVal ServerName *As String= "") As Object*

CreateObject erstellt einen Verweis auf ein COM-Objekt und gibt diesen zurück. CreateObject kann in Visual Basic nur dann zur Erstellung von Klasseninstanzen verwendet werden, wenn diese explizit als COM-Komponenten verfügbar gemacht werden.

Als Parameter übergibt man dieser Methode folgende Werte:

- ProgId = Die Programm-ID des zu erstellenden Objekts.
- ServerName= Optional. Der Name des Netzwerkservers, auf dem das Objekt erstellt werden soll. Wenn der ServerName eine leere Zeichenfolge ("") ist, wird der lokale Computer verwendet

Die SolidWorks „ProgId“ haben Sie bereits kennen gelernt. Sie lautet für Ihren ersten Test „SldWorks.Application“. Denn optionalen Parameter „ServerName“ belassen Sie mit einer leeren Zeichenfolge, da Sie Ihren lokalen Computer verwenden möchten. Fügen Sie also folgenden Code in die Methode „SolidWorksInstanzCreate“ ein.

```
oSwAppCls = Microsoft.VisualBasic.CreateObject( _
                  ProgId:="SldWorks.Application")
```

Code 1.6
SolidWorks Objekt mit CreateObject initialisieren

Doch was ist nun? Die Methode erzeugt in Visual Studio folgende Fehlermeldung: "Option Strict On" lässt keine impliziten Konvertierungen von Object in SldWorks.SldWorks zu.

Visual Studio, weist Sie darauf hin, dass mit der Option „Option Stirct On“ eine explizite Typenumwandlung angegeben werden muss.

Hinweis:
Diese Option können Sie für das gesamte Projekt unter MyProject, Register Kompilieren angeben, oder für eine Klasse oberhalb des Beginns der Klasse (also vor der Codezeile Public Class…) durch Option Strict .. festlegen.
Diese Option sollte zur Codesicherheit immer auf „on“ stehen. Sie zwingt einem Programmier zu einem sauberen Programmierstill und vermeidet im voraus Fehler im Programmcode.

Eine Typenumwandlung wird in Visual Basic durch die Methode CType angegeben.

Visual Basic
CType
Befehlsreferenz

CType(expression, typename)

CType gibt das Ergebnis der expliziten Konvertierung eines Ausdrucks in einen angegebenen Datentyp, ein Objekt, eine Struktur, eine Klasse oder eine Schnittstelle zurück.

Als Parameter übergibt man dieser Methode folgende Werte:

- expression = Ein beliebiger gültiger Ausdruck. Wenn der Wert von *expression* außerhalb des für *typename* zulässigen Bereichs liegt, löst Visual Basic eine Ausnahme aus.
- typename = Ein beliebiger Ausdruck, der innerhalb einer As-Klausel in einer Dim-Anweisung zulässig ist, d.h. der Name eines beliebigen Datentyps, eines Objekts, einer Struktur, einer Klasse oder einer Schnittstelle.

Sie müssen diese Methode jedoch nicht eingeben, da Sie die überaus komfortable Visual Studio IDE (Entwicklungsumgebung) für diesen Vorgang ausnutzen können.
Wenn Sie mit der Maus über die blau unterkringelte Codezeile fahren, erscheint rechts ein Ausrufezeichensymbol.

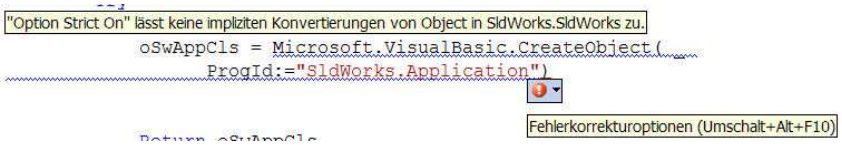


Abbildung 1.11
Fehlerhinweis im Visual Studio mit automatischer Fehler-korrekturoption

Wenn Sie auf dieses Symbol klicken, erscheint eine Fehlerkorrekturoption.

Abbildung 1.12
Visual Studio Fehler-korrekturoption Fenster

Klicken Sie einfach auf den Vorschlag dieser Fehlerkorrekturoption. Die Codezeile wird automatisch verbessert, ohne dass Sie etwas eingeben mussten.

Von der Visual Studio IDE automatisch verbesserter CreateObject Aufruf:

```
oSwAppCls = CType(Microsoft.VisualBasic.CreateObject( _
                        ProgId:="SldWorks.Application"), _
                        SldWorks.SldWorks)
```

Code 1.7
SolidWorks Objekt mit CreateObject initialisieren, incl. Typenumwandlung

Um zu kontrollieren, ob im oSwAppCls Objekt eine Instanz der SolidWorks Application vorhanden ist, verwenden Sie zwei einfache Eigenschaften der SldWorks.SldWorks Klasse, welche kurz erklärt werden.

SolidWorks Version ermitteln:

retval = SldWorks.RevisionNumber ()

SolidWorks API
SldWorks.RevisionNumber

Die Eigenschaft (Property) ist ReadOnly. Als Rückgabewert erhalten Sie die SolidWorks Version als String. Dieser String ist wie folgt aufgebaut: major.minor.minor.
Major = SolidWorks Versionjahr
Minor = Version des aktuellen ServicePack

Für die SolidWorks Version 2006, ServicePack 5.0 lautet die Rückgabe dieser Eigenschaft wie folgt: „14.5.0“.

SolidWorks Fenster anzeigen:

visibility = SldWorks.Visible (VB Get property)
SldWorks.Visible = visibility (VB Set property)

SolidWorks API
SldWorks.Visible

Diese Property stellt die Sichtbarkeit des SolidWorks Fensters ein. Als Parameter übergeben oder erhalten Sie einen Boolean Wert. Dieser ist „True“, wenn das SolidWorks Fenster angezeigt wird.

Diese beiden SolidWorks Propertys verwenden Sie in einer neuen Prozedur „VersionAnzeigen“ der Klasse „MySldWorksCls“.
Diese Prozedur besitzt also folgenden Inhalt:

Code 1.8
Die Prozedur VersionAnzeigen

```
Friend Sub VersionAnzeigen()
    Try
        'SolidWorks Fenster anzeigen
        oSwAppCls.Visible = True
        'SolidWorks Version
        'in einer MsgBox anzeigen
        MsgBox("SolidWorks Version: " & _
               oSwAppCls.RevisionNumber)
    Catch ex As Exception
        Debug.Assert(False)
        MsgBox("Fehler: Wo: " & _
            ex.StackTrace & " Was: " & ex.Message)
    End Try
End Sub
```

Die Prozedur „VersionAnzeigen“ wird in der Ereignisprozedur des „cmdCreate.Click“ Ereignis nach dem Aufruf der „SolidWorksInstanzCreate“ Methode aufgerufen. Am Ende der Ereignisprozedur ist sehr wichtig die eigene SolidWorks Klasse freizugeben (leeren).

Code 1.9
Aufruf der Prozedur VersionAnzeigen

```
'Create Methode ausführen
oMySldWorks.SolidWorksInstanzCreate()
'Version in einer MessageBox anzeigen
oMySldWorks.VersionAnzeigen()
'Eigene SolidWorks Klasse leeren
oMySldWorks = Nothing
```

Durch das Belegen des Objekts mit Nothing, wird die Instanz zur SolidWorks Application wieder freigegeben. Dies ist wichtig um SolidWorks richtig zu beendet und Speicherplatz freizugeben.

Bevor Sie nun die Methoden testen, schließen Sie bitte SolidWorks, falls Sie es geöffnet haben.
Starten Sie nun das Projekt und klicken Sie auf den „CreateObject“ Button.

Sie werden feststellen, dass die zuletzt gestartete SolidWorks Version geöffnet wird und in einer MessageBox die Version dieser SolidWorks Sitzung angezeigt wird.

Betrachten Sie die Vorgehensweise der verschiedenen Methoden Schritt für Schritt, indem Sie in die Ereignisprozedur des „cmdCreate.Click“ Ereignis einen Haltepunkt setzen.
Schließen Sie SolidWorks, und starten Sie das Projekt erneut. Klicken Sie auf den „CreateObject“ Button und wechseln Sie zu Visual Studio. Die Ausführung wurde am Haltepunkt gestoppt.

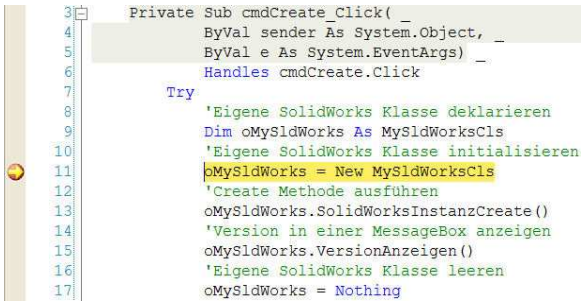


Abbildung 1.13
Haltepunkt mit Programmstop im Visual Studio

Schritt 1; Eigene SolidWorks Klasse initialisieren:

Klicken Sie auf F8, um die Codezeile auszuführen.
Wenn Sie nun den Mauszeiger auf die Variable oMySldWorks bewegen, können Sie sich den Inhalt dieser Variable anzeigen lassen.

Schritt 2; SolidWorks Instanz erzeugen:

Klicken Sie so lange auf F8, bis Sie die Codezeile der Abbildung 1.14 in der Methode „SolidWorksInstanzCreate“ sehen. Durch einen weiteren Klick führen Sie die „CreateObjekt“ Methode aus. Auch hier können Sie sich den Inhalt der Variable oSwAppCls anzeigen lassen. Diese Variable sollte mit einem Com-Objekt belegt sein. Die Variable ist somit mit einer Instanz auf die gerate erzeugte SolidWorks Application belegt.

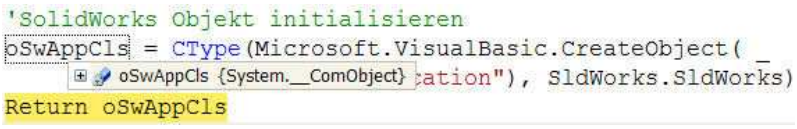


Abbildung 1.14
Inhalt der oSwAppCls Variablen in Visual Studio

Schritt 3; SolidWorks Fenster und Version in einer MsgBox anzeigen:

Klicken Sie wiederum solange auf F8, bis Sie die Methode „SolidWorksInstanzCreate“ verlassen und zur Prozedur „VersionAnzeigen“ gelangen.
Durch die Festlegung der Property „SldWorks.Visible“ mit dem Wert „True“, wird dass SolidWorks Fenster sichtbar und in der Windows Taskleiste angezeigt.
Wenn Sie nun noch mal auf F8 drücken, wird eine MessageBox geöffnet. Handelt es sich zum Beispiel um SolidWorks 2006 SP 5.0, hat die MessageBox folgenden Inhalt:

Abbildung 1.15
MsgBox mit SolidWorks Versionshinweis

Schritt 4: SolidWorks Klasse leeren / freigeben:

Als letztes wird die Variable „oMySldWorks“ in der Ereignisprozedur des „cmdCreate.Click“ Ereignis mit Nothing belegt. Diesen Variableninhalt können Sie sich im Visual Studio natürlich wieder anschauen.

Führen Sie nun die Funktion bis zum Ende aus, indem Sie auf F5 klicken, und schließen Sie das „SchulungFrm“ Formular um die Anwendung in Visual Studio zu beenden.

Nachdem die Vorgehensweise der Methoden ausführlich beschrieben wurde, werden Sie diese zum Verständnis der „CreateObject“ Methode in verschieden Zuständen und verschieden Parameter testen, um auch die letzten Fragen zu beantworten.

„CreateObjec“t bei geöffnetem SolidWorks

SolidWorks müsste durch Ihre Schritt für Schritt Analyse bereits geöffnet sein. Starten Sie also nur das Projekt, und klicken Sie auf den „CreateObject“ Button. Sie werden feststellen, dass sich kein weiteres SolidWorks öffnet, und Sie die gleiche Version in der MessageBox angezeigt bekommen.
Ist also SolidWorks bereits geöffnet, wird eine Instanz auf das bereits geöffnete SolidWorks erzeugt. Wichtig ist jedoch hierbei zu wissen, dass es dafür keine Garantie gibt. Dies bedeutet nichts anderes als, es kann sein, es kann aber auch nicht so sein.
Wie Sie eine aktuelle SolidWorks Sitzung sicher übernehmen, werden Sie im nächsten Kapitel erfahren.

„CreateObject“ mit Versionsübergabe

Im Kapitel 1.4 haben Sie gesehen, dass es neben der ProgId „SldWorks.Application“ weiter ProgId Einträge gibt (siehe Tabelle 1.1; SolidWorks ProgId und Ihre Versionszugehörigkeit).
Diese können dazu benutzt werden, um eine bestimmte SolidWorks Version zu öffnen.
Ändern Sie den „CreateObject“ Aufruf, indem Sie den Parameter ProgId eine SolidWorks Version angeben, welche auf Ihrem Computer installiert ist.

Für SolidWorks 2007 ist das zum Beispiel:

```
'SolidWorks Objekt (mit 2007) initialisieren
oSwAppCls = CType(Microsoft.VisualBasic.CreateObject( _
              ProgId:="SldWorks.Application.15"), _
              SldWorks.SldWorks)
```

Code 1.10
Geänderter CreateObject Aufruf

Wenn Sie das Projekt erneut ausführen, werden Sie feststellen, dass die übergebene SolidWorks Version geöffnet wird.
Dabei ist es egal, ob eine weitere SolidWorks Sitzung einer anderen Version geöffnet ist. Ist die Version bereits geöffnet, gelten dieselben Hinweise, welche im Abschnitt „CreateObject“ bei geöffnetem SolidWorks bereits erwähnt wurden.

Versuchen Sie jetzt das Öffnen einer nicht installierten SolidWorks Version. Auf dem Computer des Autors ist dies zum Beispiel SolidWorks 2003, also die ProgID „SldWorks.Application.11“
Ändern Sie dafür den „CreateObject“ Aufruf.

Code 1.11
Geänderter CreateObject Aufruf

```
'SolidWorks Objekt (mit 2003) initialisieren
oSwAppCls = CType(Microsoft.VisualBasic.CreateObject( _
              ProgId:="SldWorks.Application.11"), _
              SldWorks.SldWorks)
```

Beim Ausführen des Projekts ist es wieder egal, ob bereits eine SolidWorks Sitzung geöffnet ist.

Wie Sie vielleicht bereits vermuten, tritt ein Fehler in der Funktion auf, welcher uns in der MessageBox angezeigt wird.

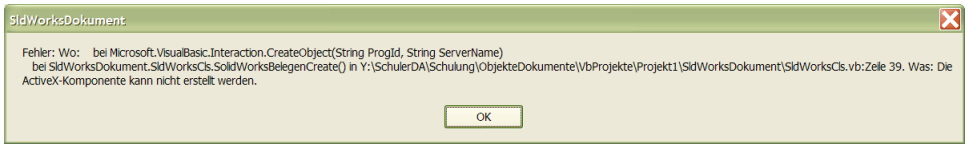


Abbildung 1.16
MsgBox mit der Fehlermeldung

Die Mitteilung der Fehlermeldung ist eindeutig.
„Die ActiveX-Komponente kann nicht erstellt werden.“
Es währe auch sehr merkwürdig, wenn dies gehen würde, da diese Komponente nicht installiert und somit nicht vorhanden ist.

Wichtig ist zu verstehen, dass nicht einfach eine beliebige SolidWorks Version geöffnet werden kann. Diese Version muss auf dem Computer installiert sein, auf welchem Sie Ihre Anwendung starten.

Das Erzeugen einer Instanz des SolidWorks Objekts durch die ProgId „SldWorks.Application“ öffnet immer eine unbestimmte SolidWorks Version. Dies kann nicht nur für Ihre Anwendung die falsche SolidWorks Version sein, sondern auch für den Anwender Ihres Programms eine ungewünschte SolidWorks Version.
Sie sehen, dass die Methode „CreateObject“ mehrere Nachteile besitzt. Sie ist zwar eine sinnvolle Methode SolidWorks zu öffnen, jedoch sollten Sie auf die Besonderheiten achten.

SolidWorks mit „CreateObject“ starten

Damit Ihnen ein weiteres Problem der „CreateObject“ Methode deutlich wird, schließen Sie bitte alle SolidWorks Sitzungen. Öffnen Sie die gewünschte SolidWorks Sitzung manuell. Wenn Sie keine SolidWorks Zusatzanwendung in SolidWorks aktiviert haben, aktivieren Sie bitte eine beliebige Zusatzanwendung.
Dazu können Sie im Zusatzanwendung Dialog, welchen Sie im Menü unter „Extras > Zusatzanwendungen“ öffnen, zum Beispiel die SolidWorks Utilities aktivieren.

Abbildung 1.17
SolidWorks Zusatzanwendung Dialog

Durch das Aktivieren dieser Zusatzanwendung wird eine weitere Symbolleiste in SolidWorks eingefügt.
Schließen Sie SolidWorks und öffnen Sie es erneut.
Wie Sie sehen, ist die Zusatzanwendung weiterhin aktiv und die Symbolleiste sichtbar.
Schließen Sie SolidWorks nun wieder und ändern Sie den „CreateObject“ Aufruf wieder so, dass Sie im ProgId Parameter „SldWorks.Application“ übergeben.

Starten Sie das Visual Basic Projekt, und bestätigen Sie die MessageBox mit dem Versionshinweis.

Betrachten Sie das SolidWorks Fenster etwas genauer. Die Zusatzanwendung wurde nicht geladen, obwohl Sie im Zusatzanwendung Dialog aktiviert ist. Besitzen Sie zudem SolidWorks Zusatzanwendungen von Drittanbieter, werden Sie feststellen, dass diese nicht mehr im Zusatzanwendung Dialog aufgelistet werden.

Dies liegt an der Eigenschaft, dass bei einer durch „CreateObject“ geöffnete SolidWorks Sitzung, keine Zusatzanwendungen geladen werden, da die jeweiligen Registryeinträge, welche dafür zuständig sind, nicht ausgewertet werden.

1.6 SolidWorks durch GetObject belegen

Die zweite Methode eine Instanz zur SolidWorks Application aufzubauen, ist die Methode „GetObject“.
Wie man dem Namen dieser Methode entnehmen kann, erzeugt diese keine neue Instanz, sondern übernimmt eine bereits geöffnete Sitzung.
Diese Vorgehensweise bietet mehrere Vorteile gegenüber der „CreateObject“ Methode.

- Der User wählt die SolidWorks Sitzung und somit die Version selbst aus.
- Alle Zusatzanwendungen werden durch den manuellen Start in SolidWorks geladen.
- Es wird keine unerwünschte SolidWorks Sitzung gestartet. *(Dieser Vorteil ist nicht zu unterschätzen, da somit ein User weiterhin der Chef an seinem Computer ist und nicht von einer Anwendung überstimmt wird. Besonders Windowsanwender sind in solchen Sachen sehr sensibel.)*

Jedoch gibt es auch bei der „GetObject“ Methode einiges zu beachten.

Damit Sie die „GetObject“ Methode testen können, verwenden Sie ähnlich wie bei der „CreateObject“ Methode, eine eigene Methode in der „MySolidWorksCls“ Klasse, welche Sie durch einen Klick auf dem „GetObject“ Button im „SchulungFrm“ Formular aufrufen.

Die neue Methode bekommt den Namen „SolidWorksInstanzGet“.
Der Funktionsrahmen dieser Methode sieht wie folgt aus:

```
Friend Function SolidWorksInstanzGet() _
                    As SldWorks.SldWorks
        Try

            Return oSwAppCls
        Catch ex As Exception
            Debug.Assert(False)
            MsgBox("Fehler: Wo: " & _
                ex.StackTrace & " Was: " & ex.Message)
            Return Nothing
        End Try
End Function
```

Code 1.12
Funktionsrahmen der Methode SolidWorks InstanzGet

Diese Methode wird, vergleichbar mit der „SolidWorksInstanzCreate“ Methode, in der Ereignisprozedur eines Klick Ereignis aufgerufen. Der unterschied liegt im Button, welcher das Klick Ereignis auslöst. Für die „SolidWorksInstanzGet“ Methode ist dies der Button „cmdGet“.
Den Inhalt der Variable „oSwAppCls“ der Klasse „MySldWorksCls“ testet Sie wieder mit Hilfe der Prozedur „VersionAnzeigen“.
Die Ereignisprozedur „cmdGet.Click“ besitzt somit folgenden Inhalt:

```
Private Sub cmdGet_Click( _
            ByVal sender As System.Object, _
            ByVal e As System.EventArgs) _
            Handles cmdGet.Click
        Try
            'Eigene SolidWorks Klasse deklarieren
            Dim oMySldWorks As MySldWorksCls
            'Eigene SolidWorks Klasse initialisieren
            oMySldWorks = New MySldWorksCls
            'Get Methode ausführen
            oMySldWorks.SolidWorksInstanzGet()
            'Version in einer MessageBox anzeigen
            oMySldWorks.VersionAnzeigen()
            'Eigene SolidWorks Klasse leeren
            oMySldWorks = Nothing
        Catch ex As Exception
            Debug.Assert(False)
            MsgBox("Fehler: Wo: " & _
                ex.StackTrace & " Was: " & ex.Message)
        End Try
End Sub
```

Code 1.13
Aufruf der Methode SolidWorks InstanzGet in der Prozedur des cmdGet.Click Ereignis

Der Funktionsrahmen der „SolidWorksInstanzGet“ Mehtode ist ähnlich dem der „SolidWorksInstanzCreate“ Methode. Einziger und wichtigster unterschied ist natürlich das initialisieren des SolidWorks Objekts. Dies geschied nun mit der „GetObject“ Methode, welche Sie sich zuerst näher betrachten.

Mitglied der Net Klasse *Microsoft.VisualBasic*

Visual Basic
GetObject
Befehlsreferenz

Public Function GetObject(_
Optional ByVal PathName As String = Nothing, _
Optional ByVal [Class] As String = Nothing) As Object

GetObject gibt einen Verweis auf ein von einer COM-Komponente bereitgestelltes Objekt zurück.

Als Parameter übergibt man dieser Funktion folgende Werte:

- PathName = Optional. Der vollständige Pfad und Name der Datei, die das abzurufende Objekt enthält. Wird PathName weggelassen, ist die Angabe von Class erforderlich.
- Class = Erforderlich, wenn PathName nicht angegeben wird. Eine Zeichenfolge, die die Klasse des Objekts darstellt. Das *Class*-Argument weist die folgende Syntax und die folgenden Bestandteile auf:
  appname.objecttype
  - *Appname* = Der Name der Anwendung, die das Objekt bereitstellt.
  - *Objecttype* = Typ oder Klasse des zu erstellenden Objekts.

Lassen Sie sich vom Parameter Class nicht verunsichern. Diesen hätte (*nach Meinung des Autors*) Microsoft besser als ProgId bezeichnen müssen. Denn in diesen Parameter wird eben eine solche ProgId übergeben.

Wenn Sie den „GetObject“ Befehl in die Methode „SolidWorksInstanzGet“ eingefügt haben, sollte diese wie folgt aussehen:

Code 1.14
Die Methode SolidWorks InstanzGet

```
Friend Function SolidWorksInstanzGet() _
                    As SldWorks.SldWorks
    Try
        oSwAppCls = _
            CType(Microsoft.VisualBasic.GetObject( _
            Class:="SldWorks.Application"), _
            SldWorks.SldWorks)
        Return oSwAppCls
    Catch ex As Exception
        Debug.Assert(False)
        MsgBox("Fehler: Wo: " & _
                ex.StackTrace & " Was: " & ex.Message)
        Return Nothing
    End Try
End Function
```

Mit Hilfe dieser Funktion werden Sie sich die Eigenschaften und Besonderheiten der „GetObject“ Methode näher anschauen.

„GetObject“ bei geöffnetem SolidWorks

Öffnen Sie bitte eine beliebige SolidWorks Sitzung, und starten Sie das Visual Basic Projekt „SldWorksDokument“. Klicken Sie im Formular bitte auf den „GetObject“ Button.

Wie gewünscht, wird durch die „GetObject“ Methode die aktuell geöffnete SolidWorks Sitzung übernommen und die Version dieser Sitzung in der MessageBox angezeigt.

„GetObject“ bei geschlossenem SolidWorks

Schließen Sie nun SolidWorks, und starten Sie unser Projekt.
Sie vermuten es wahrscheinlich schon. Da SolidWorks, und somit das Objekt „SldWorks.Application“ nicht zur Verfügung steht, kommt es zu einer Fehlermeldung.

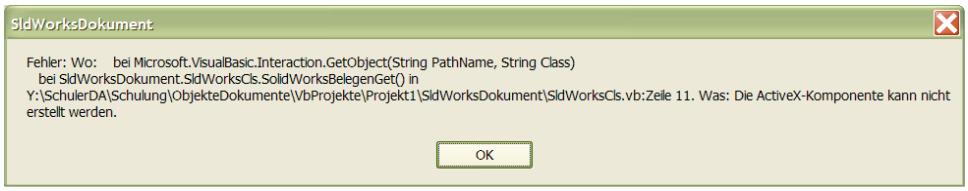


Abbildung 1.18
MsgBox mit der Fehlermeldung

Ein solcher Fehler tritt immer dann auf, wenn SolidWorks nicht geöffnet ist. Deshalb müssen Sie diesen Fehler mit einem entsprechenden Hinweis abfangen. Ein solcher Hinweis könnte zum Beispiel „Bitte öffnen Sie SolidWorks, und starten Sie darauf diese Anwendung erneut“ lauten.

„GetObject“ bei einem durch „CreateObject“ geöffnetem SolidWorks

Als nächstes möchte ich Ihnen noch eine Wechselwirkung zeigen. Schließen Sie dafür SolidWorks. Starten Sie das Projekt „SldWorksDokument“ und klicken im Formular bitte auf den „CreateObject“ Button. SolidWorks wird durch die „CreateObject“ Methode gestartet. Klicken Sie nun anschließend auf den „GetObject“ Button.
Es kommt zu einem Fehler, welcher durch unsere „Try-Catch-Anweisung“ wieder in der MessageBox angezeigt wird.

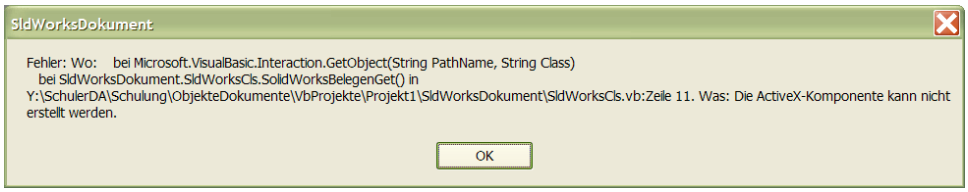


Abbildung 1.19
MsgBox mit der Fehlermeldung

Es handelt sich um die gleiche Fehlermeldung, so als ob SolidWorks nicht geöffnet ist.
Eine durch „CreateObject“ geöffnete SolidWorks Sitzung kann nicht mit „GetObject“ übernommen werden.

Diese Erkenntnis sollten Sie in Ihr Programm einfließen lassen und die oben bereits verfasste Hinweismeldung anpassen. „Bitte öffnen Sie SolidWorks, und starten Sie darauf diese Anwendung erneut. Sollte SolidWorks geöffnet sein, schließen Sie es bitte. Öffnen Sie SolidWorks, und starten Sie danach diese Anwendung“.

1.7 CreateObject oder GetObject

Mit welcher der beiden Methoden erstellt man eine Instanz zu SolidWorks? Diese Frage sollten Sie sich immer stellen, bevor Sie eine neue Anwendung beginnen.

Die Vor- und Nachteile beider Methoden wurden in den beiden Kapiteln an mehreren Beispielen beschrieben.

Der Befehl „GetObject“ bietet mehr Vor- als Nachteile.

Deshalb ist sie auch der Favorit der beiden Methoden.
Sie müssen in einer Funktion nur die Fehler abfangen, falls SolidWorks nicht oder mit „CreateObject“ geöffnet ist.

Eine solche Methode ist schnell umgesetzt und in jeder Anwendung allgemein einsetzbar.

Diese allgemeine Methode „SolidWorksInstanz“ folgt auf der folgenden Seite (aus Platzgründen in einer kleineren Schriftgröße).

Code 1.15
Allgemeine Methode um mit der GetObject Methode eine Instanz zur aktuellen SolidWorks Sitzung zu erzeugen

```
''' <summary>
''' Initialisiert das SolidWorks Objekt durch die Methode GetObject
''' und fragt in einer MsgBox nach wenn das Objekt nicht greifbar ist.
''' </summary>
''' <param name="bUserAbbruch">
''' Optionaler Rückgabewert, welcher mitteilt ob der User
''' die Aktion abgerochen hat oder ein Fehler aufgetreten ist.
''' </param>
''' <returns>Instanz der atkuellen SolidWorks Sitzung</returns>
''' <remarks></remarks>
Public Function SolidWorksInstanz( _
            Optional ByRef bUserAbbruch As Boolean = False) _
            As SldWorks.SldWorks
    Try
        'Variable für die MsgBox-Rückgabe
        Dim lDialogStatus As Microsoft.VisualBasic.MsgBoxResult
        Try
            'SolidWorks Objekt belegen
            oSwAppCls = CType(GetObject(, _
                    "SldWorks.Application"), _
                    SldWorks.SldWorks)
        Catch ex As Exception
        End Try
        'Wenn das SolidWorks Objekt nicht belegt ist...
        If oSwAppCls Is Nothing Then
           '... in einer MsgBox nachfragen
           lDialogStatus = MsgBox( _
             "Fehler beim Aufbau einer Schnittstelle zu SolidWorks." & _
             vbNewLine & _
             "Bitte öffnen Sie SolidWorks und klicken Sie auf ""Ok""." & _
             vbNewLine & _
             "Sollte SolidWorks bereits geöffnet sein, " & _
             "schließen Sie SolidWorks, " & _
             vbNewLine & _
             "öffnen Sie es erneut und klicken Sie dann auf ""Ok""." & _
             vbNewLine & _
             "Wenn Sie denn Startvorgang abbrechen " & _
             "möchten klicken Sie auf ""Abbrechen""", _
             MsgBoxStyle.Information Or MsgBoxStyle.OkCancel, _
             "Schuler Design Automation GmbH")
           'Rückgabewert der MsgBox auswerten
           Select Case lDialogStatus
                Case MsgBoxResult.Ok
                    'User hat SolidWorks geöffnet
                    'Funktion einfach nochmal ausführen
                    Return SolidWorksInstanz(bUserAbbruch)
                Case MsgBoxResult.Cancel
                    'User möchte die Anwednung abbrechen
                    bUserAbbruch = True
                    Return Nothing
            End Select
        End If
        Return oSwAppCls
    Catch ex As Exception
        Debug.Assert(False)
        MsgBox("Fehler: Wo: " & _
                ex.StackTrace & " Was: " & ex.Message)
        Return Nothing
    End Try
End Function
```

Die Methode „SolidWorksInstanz“ ist schnell erklärt.
Als Rückgabeparameter wird die SolidWorks Instanz zurückgeben.
Dieses soll in der Methode durch die aktuelle SolidWorks Sitzung initialisiert werden.
Wenn in der „GetObject“ Methode ein Fehler auftritt, wird dieser in der „Try-Catch-Anweisung“ abgefangen, jedoch nicht direkt ausgewertet. Danach wird der Inhalt des SolidWorks Objekts kontrolliert. Ist der Inhalt des Objekts Nothing wird der User in einer MessageBox auf diesen Zustand hingewiesen.
Er kann nun SolidWorks öffnen oder die Aktion abbrechen.

Fügen Sie diese Methode in Ihre „MySldWorksCls“ Klasse ein und passen Sie den Aufruf der Ereignisprozedur „cmdGet.Click“ an.

```
Private Sub cmdGet_Click( _
            ByVal sender As System.Object, _
            ByVal e As System.EventArgs) _
            Handles cmdGet.Click
        Try
            'Eigene SolidWorks Klasse deklarieren
            Dim oMySldWorks As MySldWorksCls
            'Eigene SolidWorks Klasse initialisieren
            oMySldWorks = New MySldWorksCls
            'Get Methode ausführen
            oMySldWorks.SolidWorksInstanz()
            'SolidWorks Instanz prüfen
            If oMySldWorks.MySldWorks IsNot Nothing Then
                'Version in einer MessageBox anzeigen
                oMySldWorks.VersionAnzeigen()
            End If
            'Eigene SolidWorks Klasse leeren
            oMySldWorks = Nothing
        Catch ex As Exception
            Debug.Assert(False)
            MsgBox("Fehler: Wo: " & _
                ex.StackTrace & " Was: " & ex.Message)
        End Try
End Sub
```

Code 1.16
Die Prozedur der Ereignis cmdGet.Click mit dem Aufruf der allgemeinen SolidWorksInstanz Methode

In der Methode „SolidWorksInstanz“ besitz der User die Möglichkeit die Aktion abzubrechen. Deshalb erfolgt für dem Aufruf der Methode „VersionAnzeigen“ eine Überprüfung der SolidWorks Instanz mit Hilfe der Eigenschaft „MySldWorks“.

Testen Sie die Methode. Sie werden den Vorteil der „GetObject“ Methode erkennen und diese Methode meist in Ihren Anwendungen einsetzen.

1.8 Die richtige SolidWorks Version?

Im Kapitel 1.3 haben Sie bereits erfahren, dass bei jeder neuen SolidWorks Version immer wieder neue API Methoden hinzukommen.
Deshalb ist es wichtig, die SolidWorks Version zu prüfen, egal für welche Methoden Sie sich entscheiden.
Dadurch können Sie in Ihrer Anwendung immer sicher sein, dass Methoden nur dann verwendet werden, wenn Sie in SolidWorks zur Verfügung stehen.

Es gibt viele SolidWorks API Funktionen, welche erst ab einer bestimmten SolidWorks Version zur Verfügung stehen.
Die Information, ab wann eine API Funktion in SolidWorks eingeführt wurde, können Sie der „SolidWorks Zusatzanwendungen-API-Hilfethemen“ entnehmen.

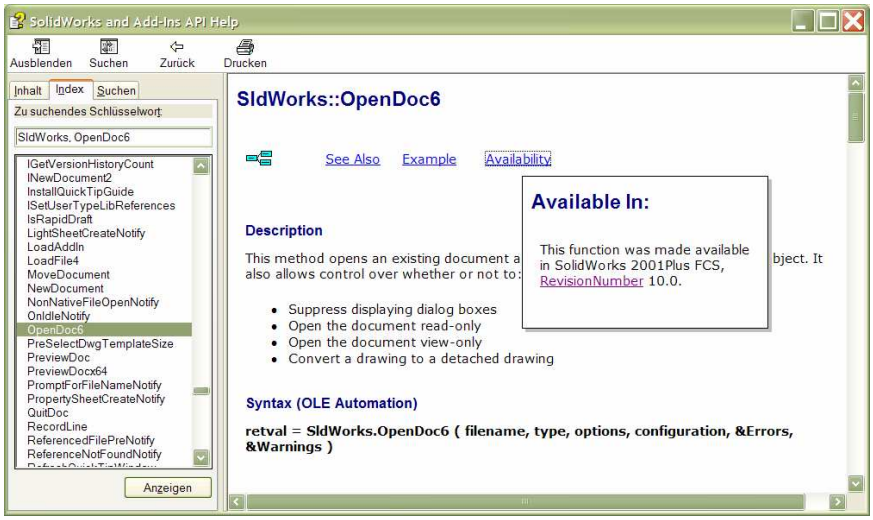


Abbildung 1.20
Hinweis wann eine Funktion in die SolidWorks API aufgenommen wurde in der SolidWorks Zusatz-anwendungen-API-Hilfethemen

Wenn Sie sich eine Methode in dieser Hilfe anzeigen lassen, verbirgt sich unter dem Link „Availability“ der Hinweis, in welcher SolidWorks Version dieser Funktion eingeführt wurde.
Bei der Funktion „SldWorks.OpenDoc6“ ist dies zum Beispiel die SolidWorks Version 2001Plus.

Diese Methode würde bei einer älteren SolidWorks Version zu einem Fehler führen.

Zur Erinnerung: Bedenken Sie, dass alte API Befehle nicht ungültig sind. Meist sind nur andere oder mehr Parameter der Grund für eine neue Methode. Alte Methoden funktionieren auch in neuen SolidWorks Version, auch wenn es dort in der API bereits neuere Befehle gibt.

Die benötigte API Methode haben Sie bereits kennen gelernt.

Die folgende Eigenschaft „VersionInJahr“ der Klasse „MySldWorksCls“ wandelt den Rückgabewert der API Methode „SldWorks.RevisionNumber“ in einen lesbaren und vergleichbaren Rückgabewert um, welcher Ihnen die SolidWorks Hauptversion, also das Jahr, ausgibt.

Für diesen Zweck wird der „String“, welchen wir von der Methode „SldWorks.RevisionNumber“ erhalten, an den Trennpunkten geteilt und die Hauptversionsnummer in eine „Integer“ Variable umgewandelt. Diese Variable wird mit 1992 addiert und als Rückgabewert für die Eigenschaft verwendet.

Das Ergebnis dieser Aufgabenstellung sieht in einer Visual Basic Eigenschaft wie folgt aus:

```
Friend ReadOnly Property VersionInJahr() As Integer
        Get
            Dim iHauptversion As Integer
            Dim sVersionen() As String
            'SolidWorks Instanz prüfen
            If oSwAppCls Is Nothing Then
                '-1 als ungültige Rückgabe verwenden
                Return -1
            Else
                'Version ermitteln und an den
                'Trennpunkten teilen
                sVersionen = _
                    oSwAppCls.RevisionNumber.Split( _
                    CChar("."))
                'Nur die Hauptversion weiter verwenden
                iHauptversion = CInt(sVersionen(0))
                'Version in Jahreszahl umrechnen
                iHauptversion = iHauptversion + 1992
                'Die Hauptversion in Jahren
                'als Rückgabe verwenden
                Return iHauptversion
            End If
        End Get
End Property
```

Code 1.17
Die Eigenschaft VersionInJahr welche uns die SolidWorks Hauptversion in Jahren ausgibt

Diese eigene Eigenschaft können Sie in der Prozedur „VersionAnzeigen“ der „MySldWorksCls“ Klasse, anstelle der SolidWorks API Eigenschaft, verwenden:

```
            'SolidWorks Version
            'in einer MsgBox anzeigen
            MsgBox("SolidWorks Version: " & _
                    Me.VersionInJahr)
```

Code 1.18
Die Eigenschaft VersionInJahr in der MsgBox der VersionAnzeigen Prozedur verwenden

1.9 Die SwConst Library

In Kapitel 1.1 haben Sie neben der bereits vorgestellten „SldWorks Type Library“ auch die „SolidWorks constant type library“ (kurz „SwConst“) in das SldWorksDokument Projekt hinzugefügt.

Der Sinn dieser Type Library wird durch ein sehr einfaches Beispiel deutlich.

Ein paar Hinweise vorweg, die „SwConst“ enthält keinerlei Methoden oder Propertys. Sie besteht viel mehr aus einer Fülle von Konstanten, oder besser gesagt mehreren Enumerationsvariablen. Diese beschreiben Optionen oder Einstellungen, welche Sie durch Methoden der „SldWorks Type Library“ festlegen können. Anstelle von Zahlen geben Sie durch eine Enumerationsvariable eine lesbare Konstante an, welche die Option, Einstellung oder den Rückgabewert der SolidWorks Methode in kurzen Wörtern beschreibt. Somit wird bereits Ihr Programmcode lesbar, und es lassen sich Fehler und unnötiges Suchen (welche Option war noch mal die 1?) vermeiden.

Als Beispiel Methode verwenden Sie die SolidWorks Funktion zum Anzeigen einer einfachen MessageBox.

SolidWorks API
SldWorks.
SendMsgToUser2

result = SldWorks.SendMsgToUser2 (message, icon, buttons)

Als Parameter übergibt man dieser Funktion folgende Werte:

- message = String mit der Mitteilung, welche in der MessageBox angezeigt wird.
- icon = Enumerationsvariable des Typs swMessageBoxIcon_e, welche das angezeigte Icon in der MessageBox festlegt.
- Buttons = Enumerationsvariable des Typs swMessageBoxBtn_e, welche die angezeigten Buttons der MessageBox festlegen.

Als Rückgabe der SldWorks.SendMsgToUser2 Methode erhalten Sie eine Enumerationsvariable des Typs swMessageBoxResult_e, welche Ihnen die Antwort des User (also den gedrückten Button) zurückgibt.

Tauschen Sie in Ihrer Prozedur „VersionAnzeigen“ den MessageBox Aufruf aus und verwenden Sie anstelle einer VB-MessageBox eine SolidWorks-MessageBox.
Damit Sie eine auswertbare Antwort bekommen möchten, zeigen Sie in der MessageBox einen „Ja“ und „Nein“ Button an, was bei einer Versionsmitteilung natürlich nicht üblich ist.

Ändern Sie die Prozedur zunächst so, indem Sie die „SwConst“ Enumerationsvariablen zunächst nicht verwenden.

Code 1.19
Aufruf der SldWorks.SendMsgToUser2 ohne SwConst

```
'Rückgabewert der SolidWorks
'MessageBox
Dim iSwMsgBox As Integer
'SolidWorks Fenster anzeigen
oSwAppCls.Visible = True
'SolidWorks Version
'in einer MsgBox anzeigen
iSwMsgBox = oSwAppCls.SendMsgToUser2( _
    "SolidWorks Version: " & _
    Me.VersionInJahr, 2, 5)
Select Case iSwMsgBox
    Case 6
        'Hinweis in einer MsgBox
        MsgBox("Der User hat auf Ja geklickt")
    Case 3
        'Hinweis in einer MsgBox
        MsgBox("Der User hat auf Nein geklickt")
    Case Else
        Debug.Assert(False)
End Select
```

Testen Sie diese Funktionalität der SldWorks.SendMsgToUser2 Methode in dem Sie das Projekt starten und einen Button klicken.

Alles funktioniert Prima, jedoch ist der Programmcode mit den Parametern 2 und 5, sowie die Auswertung mit 6 und 3 nicht wirklich sehr informativ.

Würde die Meldung „Der User hat auf Ja geklickt“ fehlen, könnten Sie nur durch Debuggen der Prozedur feststellen, was die Antwort 3 in der „iSwMsgBox“ Variablen bedeutet.

Verändern Sie deshalb in der Prozedur „VersionAnzeigen“ den Aufruf der Methode „SldWorks.SendMsgToUser“ so, dass Sie nun die „SwConst“ Enumerationsvariablen verwenden.

Code 1.20
Aufruf der SldWorks.SendMsgToUser2 mit SwConst

```
'Rückgabewert der SolidWorks
'MessageBox
Dim eSwMsgBox As SwConst.swMessageBoxResult_e
'SolidWorks Fenster anzeigen
oSwAppCls.Visible = True
'SolidWorks Version
'in einer MsgBox anzeigen
eSwMsgBox = CType(oSwAppCls.SendMsgToUser2( _
    "SolidWorks Version: " & _
    Me.VersionInJahr, _
    SwConst.swMessageBoxIcon_e.swMbInformation, _
    SwConst.swMessageBoxBtn_e.swMbYesNo), _
    SwConst.swMessageBoxResult_e)
Select Case eSwMsgBox
    Case SwConst.swMessageBoxResult_e.swMbHitYes
        'Hinweis in einer MsgBox
        MsgBox("Der User hat auf Ja geklickt")
    Case SwConst.swMessageBoxResult_e.swMbHitNo
        'Hinweis in einer MsgBox
        MsgBox("Der User hat auf Nein geklickt")
    Case Else
        Debug.Assert(False)
End Select
```

Auch diese MessageBox funktioniert. Wenn Sie jedoch den Programmcode betrachten, ist der Unterschied der Codelesbarkeit mehr als deutlich.

Dies ist nicht der einzige Vorteil, der „SwConst“. Durch den Inhalt der Enumerationsvariable „SwConst.swMessageBoxIcon_e“ wissen Sie, welche Optionen bei der Icon Auswahl gültig sind. Dadurch wird eine falsche Angabe ausgeschlossen.

Sie sollten sich diesen Programmierstil angewöhnen.
In allen Beispielen dieses Skripts werden die SwConst Enumerationsvariablen verwendet, soweit dieses möglich ist.

2 Auf SolidWorks reagieren

Nicht selten ist es in einer Anwendung interessant zu wissen, welche Aktion ein User in SolidWorks ausführt bzw. startet. Damit dies eine Anwendung überwachen kann, stellt uns SolidWorks eine Reihe von Events zur Verfügung, mit deren Hilfe Sie auf bestimmte Aktionen hingewiesen werden.

Bekanntlich sind die Themen dieses Skripts SolidWorks Objekte und Dokumente. Es wäre also in diesem Zusammenhang interessant durch ein Event (Ereignis) zu erfahren, wenn in SolidWorks ein Dokument geöffnet oder gespeichert wird.

Dieses realisieren Sie in einem neuem Formular „SldWorksEventFrm“, welches Sie in unser vorhandenes Projekt einfügen.
In der Klasse dieses Formulars werden Sie anschließend die SolidWorks Ereignisse überwachen und auswerten. Dies ist, vor allem für eine konsequente objektorientierte Programmierung, auch in der Klasse „MySldWorksCls“ und einer neuen Dokument Klasse möglich. Es wird in diesem Kapitel auf eine solche Klasse verzichtet, um Ihnen die Ereignisse leicht verständlich zu erklären.

Das Formular „SldWorksEventFrm“ wird für die kommenden Aufgaben mit einigen Steuerelementen ausgestattet.
Fügen Sie bitte ein Label, zwei Buttons und eine ListBox ein und verändern Sie die Eigenschaften dieser Steuerelemente nach den folgenden Tabellen:

SldWorksEventFrm:

Eigenschaft	**Wert**
ShowIcon	False
Size	400; 300
Text	SolidWorks API Schulung

Tabelle 2.0 Geänderte Eigenschaften des SldWorksEventFrm Formular

Label1:

Eigenschaft	**Wert**
Name	lblHinweis
Text	Ermittelte SolidWorks Events:

Tabelle 2.1 Geänderte Eigenschaften des Label1

Button1:

Eigenschaft	**Wert**
Name	cmdStart
Size	100; 30
Text	Start

Tabelle 2.2 Geänderte Eigenschaften des Button1

Button2:

Eigenschaft	Wert
Name	cmdBeenden
Size	100; 30
Text	Beenden

Tabelle 2.3 Geänderte Eigenschaften des Button2

ListBox1:

Eigenschaft	Wert
Name	lstEreignis
Size	370; 140

Tabelle 2.4 Geänderte Eigenschaften des ListBox1

Diese Steuerelemente richten Sie im Formular bitte aus, damit dieses ungefähr so aussieht.

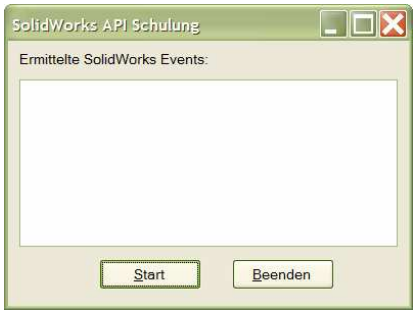


Abbildung 2.0 Das Formular SldWorksEventFrm

Damit das Formular beim Starten des „SldWorksDokument“ Projekts angezeigt wird, müssen Sie die Einstellung des Startformulars anpassen.
Öffnen Sie die Projekteinstellungen unter „MyProject“, und wechseln Sie in das Register „Anwendung“. Dort wählen Sie als Startformular das „SldWorksEventFrm“ aus.

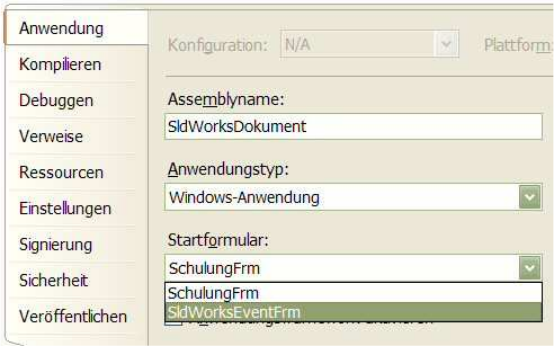


Abbildung 2.1 Das Startformular in den Projekt-einstellungen ändern

Nach diesen Vorbereitungen können Sie mit der Eventprogrammierung in SolidWorks beginnen.

2.1 SolidWorks Ereignisse verfügbar machen

Der Dim Anweisung können Sie den optionalen Bestandteil WithEvents übergeben.
Die Dim Anweisung deklariert und reserviert Speicherplatz für eine oder mehrere Variablen.

Wenn eine Variable mit WithEvents definiert ist, können Sie deklarativ angeben, dass eine Methode die Ereignisse der Variablen mit dem Handles-Schlüsselwort behandelt.
WithEvents kann nur auf Klassen- oder Modulebene verwendet werden. Dies bedeutet, dass der Deklarationskontext für eine WithEvents-Variable, eine Klasse oder ein Modul sein muss und keine Quelldatei, kein Namespace, keine Struktur und keine Prozedur sein kann.

In Ihrem Beispiel deklarieren Sie das SolidWorks Objekt in der Formularklasse so, dass dieses in der gesamten Klasse zur Verfügung steht. Dies bedeutet, dass Sie die Dim Anweisung direkt nach dem einleiteten der Formularklasse einfügen.

```
Public Class SldWorksEventFrm

    Dim WithEvents oSwAppCls As SldWorks.SldWorks

End Class
```

Code 2.0
Deklarieren des SolidWorks Objekts mit dem WithEvents Bestandteil

Hinweis:
Nicht alle SolidWorks Objekte besitzen Events. Ob und welche Events ein SolidWorks Objekt besitzen, können Sie der Online Hilfe oder dem Objektbrowser im Visual Studio entnehmen.

Durch die Deklaration mit WithEvents stehen Ihnen in der Formularklasse „SldWorksEventFrm“ alle Ereignisse der „SldWorks.SldWorks“ Klasse zur Verfügung.

2.2 Auf das Öffnen eines Dokuments reagieren

Damit Sie auf das Öffnen eines Dokuments reagieren können, benötigen Sie ein Ereignis der „SldWorks.SldWorks“ Klasse, welches Ihnen das Öffnen eines Dokuments mitteilt.
Der API Online Hilfe können Sie im Register Inhalt entnehmen, dass Ihnen SolidWorks sogar drei Ereignisse für diese Aktion zur Verfügung stehen.

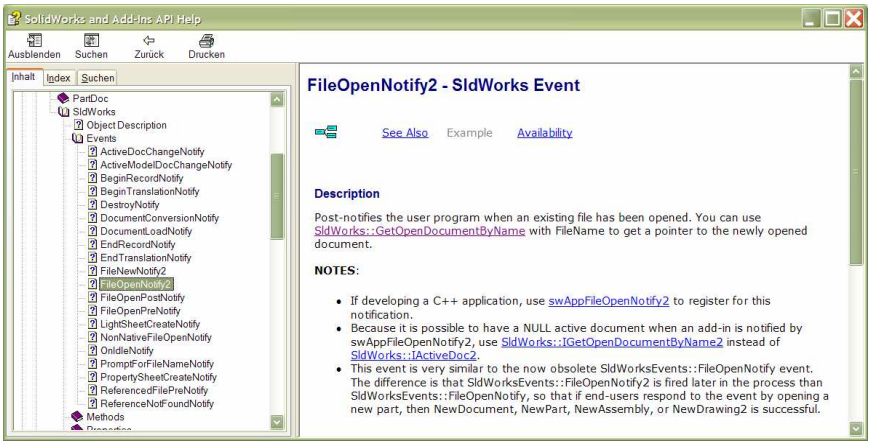


Abbildung 2.2
Die Events des SldWorks.SldWorks Objekts

Es handelt sich um die Ereignisse „FileOpenNotify2“, „FileOpenPostNotify“ und „FileOpenPreNotify“.
Diese Ereignisse werden, wie die Bezeichnungen verraten, nacheinander in folgender Reihenfolge aufgerufen:

1. FileOpenPreNotify – Das Öffnen dieses Dokuments hat noch nicht begonnen.
2. FileOpenNotify2 – Standard Ereignis, welche das Öffnen eines Dokuments mitteilt.
3. FileOpenPostNotify – Das Öffnen des Dokuments ist abgeschlossen.

Das verwendete Ereignis ist das Standardereignis „FileOpenNotify2“.

status = FileOpenNotify2 (FileName)

SolidWorks API
SldWorks.
FileOpenNotify2

Als Parameter erhalten Sie von diesem Ereignis folgenden Wert:

- FileName = Dateipfad des SolidWorks Dokuments, welches in SolidWorks geöffnet wird.

Dieses, sowie alle anderen Ereignisse, fügen Sie in die Formularklasse ein, indem Sie in der linken ComboBox des Codefensters das deklarierte „SldWorks.SldWorks“ Objekt, in diesem Beispiel „oSwAppCls“, auswählen.
Wenn Sie das Objekt in der linken ComboBox ausgewählt haben, stehen Ihnen alle Ereignisse dieses Objekts in der rechten ComboBox zur Verfügung. Wählen Sie in der rechten ComboBox das „FileOpenNotify2“ Ereignis aus.

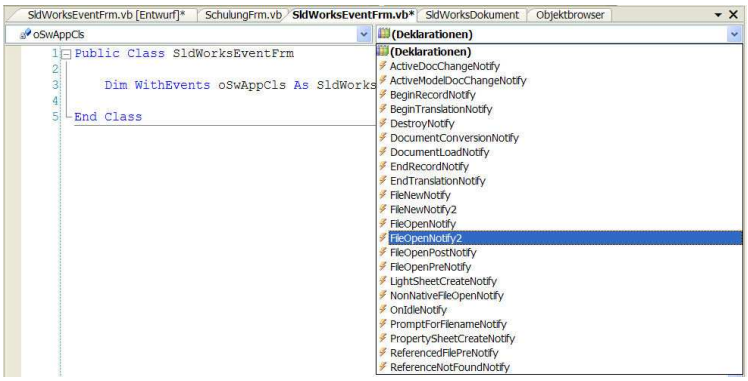


Abbildung 2.3
Das Einfügen des Events FileOpenNotify2 mit Hilfe des Visual Studios

Wenn Sie den Eintrag in der ComboBox ausgewählt haben, wird die Ereignisprozedur automatisch in die Klasse eingefügt.

```
Private Function oSwAppCls_FileOpenNotify2( _
            ByVal FileName As String) As Integer _
            Handles oSwAppCls.FileOpenNotify2

End Function
```

Code 2.1
Ereignisprozedur des FileOpenNotify2 Ereignis

Die Parameter werden für dieses Ereignis so angelegt, wie Sie das „FileOpenNotify2“ Ereignis benötigt.
Der für Visual Basic wichtigste Teil dieses Händlers ist „Handles oSwAppCls.FileOpenNotify2“.

Damit wir dieses Ereignis testen können, müssen Sie in die „SldWorksEventFrm“ Formularklasse zwei weitere Ereignisse für den „cmdStart“ und „cmdBeenden“ Button einfügen.

Im „cmdStart.Click“ Ereignis initialisieren Sie das deklarierte „oSwAppCls“ Objekt mit der bekannten Methode „SolidWorksInstanz“ aus der „MySldWorksCls“.

Code 2.2 Ereignisprozedur des cmdStart.Click Ereignis

```
Private Sub cmdStart_Click( _
    ByVal sender As System.Object, _
    ByVal e As System.EventArgs) _
    Handles cmdStart.Click
    Try
        'Eigene SolidWorks Klasse deklarieren
        Dim oMySldWorks As MySldWorksCls
        'Eigene SolidWorks Klasse initialisieren
        oMySldWorks = New MySldWorksCls
        'Get Methode ausführen
        oSwAppCls = oMySldWorks.SolidWorksInstanz()
        'Eigene SolidWorks Klasse leeren
        oMySldWorks = Nothing
    Catch ex As Exception
        Debug.Assert(False)
        MsgBox("Fehler: Wo: " & _
            ex.StackTrace & " Was: " & ex.Message)
    End Try
End Sub
```

Im „cmdBeenden.Click“ Ereignis geben Sie das „oSwAppCls“ Objekt wieder frei (leeren) und schließen das Formular, damit die Anwendung beenden wird.

Code 2.3 Ereignisprozedur des cmdBeenden.Click Ereignis

```
Private Sub cmdBeenden_Click( _
    ByVal sender As System.Object, _
    ByVal e As System.EventArgs) _
    Handles cmdBeenden.Click
    Try
        'SolidWorks freigeben
        oSwAppCls = Nothing
        'Formular schließen
        Me.Close()
    Catch ex As Exception
        Debug.Assert(False)
        MsgBox("Fehler: Wo: " & _
            ex.StackTrace & " Was: " & ex.Message)
    End Try
End Sub
```

Wenn das „SldWorksDokument“ Projekt gestartet und auf den „Start“ Button des „SldWorksEventFrm“ geklickt wird, initialisiert das Ereignis des Buttons die Variable „oSwAppCls“ mit der aktuellen SolidWorks Sitzung.

Ab dem Moment, in der diese Variable mit der SolidWorks Instanz belegt ist, kann die Formularklasse alle Ereignisse empfangen. Verarbeitet werden jedoch nur die Ereignisse, welche in der Klasse über einen Ereignishändler, definiert durch das Schlüsselwort „Handles“, verfügen.

Da die Ereignisprozedur des „FileOpenNotify2“ Ereignis leer ist, bekommen Sie davon nichts mit. In anderen Worten es passiert nichts.

Als Ziel soll in der Ereignisprozedur „FileOpenNotify2“ der übergebene Parameter „Filename“ als neuer Eintrag in die ListBox „lstEreignis“ einfügen werden.

Um dies zu realisieren, benötigen Sie jedoch eine Einstellung, die im „Form.Load“ Ereignis des „SldWorksEventFrm“ festgelegt wird.

Dies ist deshalb nötig, da es sich bei einem Zugriff auf die ListBox „lstEreignis“ des „SldWorksEventFrm“ Formulars im „SldWorks.FileLoadNotify2“ Ereignis um einen ungültigen Thread übergreifenden Zugriff handelt.
Damit sich das Augenmerk dieses Skripts nicht auf die Threadprogrammierung mit Visual Basic richtet, sondern auf die SolidWorks API Programmierung, fügen wir ein „Form.Load“ Ereignis in die Klasse ein und legen in diesem Ereignis fest, dass Thread übergreifende Zugriffe erlaubt sind.
(Aus Platzgründen in kleinerer Schriftgröße)

Code 2.4
Ereignisprozedur des MyBase.Load Ereignis

```
Private Sub SldWorksEventFrm_Load( _
            ByVal sender As System.Object, _
            ByVal e As System.EventArgs) _
            Handles MyBase.Load
   Try
        'Unerlaubte Thread übergreifende Zugriffe erlauben
        System.Windows.Forms.Form.CheckForIllegalCrossThreadCalls = False
   Catch ex As Exception
       Debug.Assert(False)
       MsgBox("Fehler: Wo: " & _
                ex.StackTrace & " Was: " & ex.Message)
   End Try
End Sub
```

Dies ist nicht die feine englische Art. Es gibt über die Invoke Methode eine offizielle Möglichkeit einen solchen Aufruf und Zugriff zu realisieren. Wegen des bereits erwähnten Grunds möchten wir jedoch an dieser Stelle darauf verzichten und nicht nur ein Auge zudrücken.

Hinweis:
Weitere Hinweise erhalten Sie in der zahlreichen Fachliteratur und in zahlreichen Beiträgen im Internet. Hier nur eine kleine Auswahl:

- http://www.galileocomputing.de/openbook/visual_basic/Kapitel_11-001.htm
- http://www.microsoft.com/germany/msdn/library/net/MultithreadProgrammierungMitVisualBasicNET.mspx

► Link
Mehr Informationen über Multithread Programmierung

Fügen Sie nun in der Ereignisprozedur des „FileOpenNotify2“ Ereignis einen Aufruf hinzu, um den Dateinamen in die Liste der ListBox hinzuzufügen.

```
Private Function oSwApp_FileOpenNotify2( _
            ByVal FileName As String) As Integer _
            Handles oSwApp.FileOpenNotify2

        Try
            'Den übergebenen Dateinamen
            'in die ListBox einfügen
            '(Ungültiger Thread
            'übergreifender Zugriff)
            Me.lstEreignis.Items.Add(FileName)

        Catch ex As Exception
            Debug.Assert(False)
            MsgBox("Fehler: Wo: " & _
                ex.StackTrace & " Was: " & ex.Message)
        End Try

End Function
```

Code 2.5
Inhalt der Ereignisprozedur des FileOpenNotify2 Ereignis

Wenn Sie das Projekt „SldWorksDokument“ starten und im Formular „SldWorksEventFrm“ auf den „Start“ – Button klicken empfängt Ihr Formular alle Ereignisse. Wenn Sie anschließend in SolidWorks ein Dokument öffnen, wird der Dateipfad dieses Dokuments in die Liste des Formulars angetragen.

Somit haben Sie auf ein Ereignis von SolidWorks reagiert und dieses ausgewertet.

2.3 Auf das Speichern eines Dokuments reagieren

Bevor Sie auf das Ereignis einer Speicheraktion reagieren, müssen Sie sich in die SolidWorks Dokument Klassen einarbeiten.
Diese enthalten die benötigen Events und werden deshalb benötigt.

Ein geöffnetes SolidWorks Dokument wird durch das „SldWorks.ModelDoc2“ Objekt dargestellt. Dieses lässt sich je nach Dokumenttyp in die Objekte „SldWorks.PartDoc“, „SldWorks.AssemblyDoc“ und „SldWorks.DrawingDoc“ umwandeln.

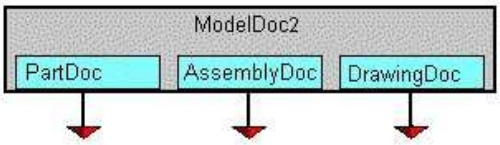


Abbildung 2.4
Objektstruktur des ModelDoc2 Objekts

Die SolidWorks „SldWorks.ModelDoc2“ Klasse wird zum Beispiel durch die Eigenschaft „ActiveDoc“ oder die Methode „GetOpenDocumentByName“ der „SldWorks.SldWorks“ Klasse initialisiert. Es gibt weitere Methoden über die eine Initialisierung erfolgen kann. Wir betrachten uns im Moment nur die Eigenschaft (Property) „SldWorks.ActiveDoc“.

ActiveDoc = SldWorks.ActiveDoc (VB Get property)

SolidWorks API
SldWorks.
ActiveDoc

Diese Property gibt Ihnen als Rückgabewert eine Instanz des „ModelDoc2“ Objekts zurück, von dem Dokument, welches aktuell in SolidWorks aktiv (aktives Dokumentfenster) ist.

Für das Beispiel dieses Kapitels können Sie das „FileOpenNotify2“ Ereignis nutzen, um in dieser das „SldWorks.ModelDoc2“ Objekt zu initialisieren.

```
Private Function oSwAppCls_FileOpenNotify2( _
            ByVal FileName As String) As Integer _
            Handles oSwAppCls.FileOpenNotify2
        Try
            'Den übergebenen Dateinamen
            'in die ListBox einfügen
            '(Ungültiger Thread
            'übergreifender Zugriff)
            Me.lstEreignis.Items.Add("Öffnen: " & FileName)
            'ModelDoc2 deklarieren
            Dim oSwModel As SldWorks.ModelDoc2
            'Dokumenten Objekt belegen
            oSwModel = CType(oSwAppCls.ActiveDoc, _
                             SldWorks.ModelDoc2)
            'Nicht mehr benötigtes Objekt freigeben
            oSwModel = Nothing
        Catch ex As Exception
            Debug.Assert(False)
            MsgBox("Fehler: Wo: " & _
                ex.StackTrace & " Was: " & ex.Message)
        End Try
End Function
```

Code 2.6
Belegen des ModelDoc2 Objekts in der Ereignisprozedur des FileOpenNotify2 Ereignis

Als nächstes müssen Sie den geöffneten Dokumenttyp ermitteln, um mit Hilfe dieser Information ein spezielleres Dokumenten Objekt zu belegen.
Ist zum Beispiel in SolidWorks eine Baugruppe geöffnet, dessen Instanz in einem „SldWorks.ModelDoc2“ Objekt steckt, können Sie dieses nicht in ein „PartDoc“ oder „DrawingDoc“ umwandeln. Dies würde in Ihrer Anwendung einen Fehler auslösen. Nur die Umwandlung in ein „SldWorks.AssemblyDoc“ Objekt ist in einem solchen Fall richtig.

Den Dokumententyp können Sie durch eine Methode der „SldWorks.ModelDoc2“ Klasse ermitteln.

retval = ModelDoc2.GetType

SolidWorks API
ModelDoc2.
GetType

Als Parameter erhalten wir von dieser Methode folgenden Wert:

- retval = Enumerationsvariable des Typs swDocumentTypes_e

Die Rückgabe dieser Methode können Sie direkt in einer Select Case Anweisung auswerten.

```
Select Case oSwModel.GetType
    Case SwConst.swDocumentTypes_e.swDocPART
    Case SwConst.swDocumentTypes_e.swDocASSEMBLY
    Case SwConst.swDocumentTypes_e.swDocDRAWING
    Case Else
        Debug.Assert(False)
End Select
```

Code 2.7
Select Case Anweisung um den Dokumetentyp auszuwerten

Damit Sie die spezielleren Dokumentobjekte initialisieren und dessen Ereignisse nutzen können, müssen Sie diese, wie die „oSwAppCls“ Variable zuvor, in der gesamten Formularklasse zur Verfügung stellen und mit dem Bestandteil WithEvents deklariert werden.

```
Dim WithEvents oSwPartCls As SldWorks.PartDoc
Dim WithEvents oSwAssemblyCls As SldWorks.AssemblyDoc
Dim WithEvents oSwDrawingCls As SldWorks.DrawingDoc
```

Code 2.8
Deklaration der SolidWorks Dokument Objekte

Diese Objekte belegen Sie in der bereits bestehenden Select Case Anweisung, je nach Dokumententyp.

```
'ModelDoc2 deklarieren
Dim oSwModel As SldWorks.ModelDoc2
'Dokumenten Objekt belegen
oSwModel = CType(oSwAppCls.ActiveDoc, _
    SldWorks.ModelDoc2)
Select Case oSwModel.GetType
    Case SwConst.swDocumentTypes_e.swDocPART
        oSwPartCls = CType(oSwModel, _
            SldWorks.PartDoc)
    Case SwConst.swDocumentTypes_e.swDocASSEMBLY
        oSwAssemblyCls = CType(oSwModel, _
            SldWorks.AssemblyDoc)
    Case SwConst.swDocumentTypes_e.swDocDRAWING
        oSwDrawingCls = CType(oSwModel, _
            SldWorks.DrawingDoc)
    Case Else
        Debug.Assert(False)
End Select
'Nicht mehr benötigtes Objekt freigeben
oSwModel = Nothing
```

Code 2.9
Vollständige Verarbeitung des ModelDoc2 Objekt in der Ereingisprozedur FileOpenNotify2

Mit dieser Select Case Anweisung belegen Sie bei einem SolidWorks Dokument immer das entsprechende Dokument Objekt in dem Ereignis, welches aufgerufen wird, wenn ein Dokument in SolidWorks geöffnet wird.
Sie können also immer auf das Speichern dieses zuletzt geöffneten Dokuments reagieren.

Das benötigte Ereignis heißt in allen Dokument Objekten „FileSaveNotify“.

SolidWorks API
PartDoc.
FileSaveNotify
AssemblyDoc.
FileSaveNotify
DrawingDoc.
FileSaveNotify

status = FileSaveNotify (FileName)

Als Parameter erhalten Sie von diesem Ereignis folgenden Wert:

- FileName = Dateipfad des SolidWorks Dokuments, unter welches es gespeichert wird

In diesem Ereignis fügen Sie, ähnlich wie beim Öffnen eines Dokuments, einen Eintrag in die Liste der Listbox „lstEreignis“ ein. Damit Sie diesem mit einem Öffnen Eintrag unterscheiden können, ändern Sie zunächst das FileOpenNotify2 Ereignis.

Code 2.10
Erweiterung des Listeneintrags

```
'Den übergebenen Dateinamen
'in die ListBox einfügen
'(Ungültiger Thread
'übergreifender Zugriff)
Me.lstEreignis.Items.Add("Öffnen: " & FileName)
```

Fügen Sie bitte nun für alle drei SolidWorks Dokument Objekte, also „oSwPartCls“, „oSwAssemblyCls“ und „oSwDrawingCls“, das „FileSaveNotify“ Ereignis in unsere Klasse „SldWorksEventFrm“ ein. Diese Ereignisprozeduren befüllen Sie mit einem jeweils identischen Inhalt, welche am Beispiel des Ereignisses „oSwPartCls. FileSaveNotify“ dargestellt wird.

Code 2.11
Ereignisprozedur des FileSaveNotify Ereignis

```
Private Function oSwPartCls_FileSaveNotify( _
            ByVal FileName As String) As Integer _
            Handles oSwPartCls.FileSaveNotify
    Try
        'Den übergebenen Dateinamen
        'in die ListBox einfügen
        '(Ungültiger Thread
        'übergreifender Zugriff)
        Me.lstEreignis.Items.Add("Speichern: " & FileName)
    Catch ex As Exception
        Debug.Assert(False)
        MsgBox("Fehler: Wo: " & _
            ex.StackTrace & " Was: " & ex.Message)
    End Try

End Function
```

Damit Sie die Anwendung sauber beenden fügen Sie als letztes das Freigeben der Dokument Objekte in die „cmdBeenden.Click“ Ereignisprozedur ein.

```
Private Sub cmdBeenden_Click( _
        ByVal sender As System.Object, _
        ByVal e As System.EventArgs) _
        Handles cmdBeenden.Click
        'SolidWorks freigeben
        oSwPartCls = Nothing
        oSwDrawingCls = Nothing
        oSwAssemblyCls = Nothing
        oSwAppCls = Nothing
        'Formular schließen
        Me.Close()
End Sub
```

Code 2.12 Ereignisprozedur des cmdBeenden.Click Ereignisses, in welchem die SolidWorks Objekte freigegeben werden

Nun können Sie die Anwendung wieder testen, indem Sie Dokumente in SolidWorks öffnen und speichern. Vergessen Sie dabei nicht die Verarbeitung der Ereignisse mit einem Klick auf den „Start“ - Button zu starten. Sie werden sehen, wie sich Ihre Aktionen auf die Einträge der ListBox auswirken.

2.4 Ein Ereignis in SolidWorks stoppen

Sie können ein Ereignis in SolidWorks nicht nur auswerten, sondern auch viele Ereignisse stoppen bzw. verändern.
Sie können zum Beispiel das Öffnen eines Dokuments verhindern, indem Sie das Ereignis abfragen und dessen Status verändern.

Das möchten Sie doch sofort mal ausprobieren!

Das Öffnen eines Dokuments lässt sich im „FileOpenPreNotify“ stoppen. Welches Ereignis war das noch mal? Das „FileOpenPreNotify“ Ereignis wird vor dem Öffnen eines Dokuments ausgelöst.

status = FileOpenPreNotify (FileName)

SolidWorks API SldWorks. FileOpenPreNotify

Als Parameter erhalten Sie von diesem Ereignis folgenden Wert:

- FileName = Dateipfad des SolidWorks Dokuments, unter welches es geöffnet werden soll.

Der Status dieses Ereignisses ist Standard auf Null gesetzt. Möchten Sie dieses Ereignis stoppen, müssen Sie diesen Status in 1 oder true (hat bei der Umwandlung in Integer den Wert 1) als Rückgabewert übergeben. Wenn Sie das Ereignis nicht verhindern möchten, lassen Sie den Status einfach unverändert.

Um nicht jedes Öffnen eines Dokuments zu verhindern, fragen Sie dies am besten mit einer MessageBox ab.

Fügen Sie also in unsere Klasse „SldWorksEventFrm“ das Ereignis „oSwAppCls.FileOpenPreNotify“ ein und befüllen dieses mit folgendem Inhalt:

Code 2.13
Inhalt der Ereignisprozedur des FileOpenPreNotify2 Ereignis

```
Private Function oSwAppCls_FileOpenPreNotify( _
            ByVal FileName As String) As Integer _
            Handles oSwAppCls.FileOpenPreNotify
        Try
            If MsgBox( _
            "Es wurde versucht in SolidWorks" & _
            vbNewLine & "die Datei: " & FileName & _
            " zu öffnen: " & vbNewLine & _
            "Möchten Sie dies verhindern?", _
            MsgBoxStyle.Question Or MsgBoxStyle.YesNo, _
            "Öffnen der Datei abbrechen") = _
            MsgBoxResult.Yes Then
                'Ereignis abbrechen
                Return CInt(True)
            End If
        Catch ex As Exception
            Debug.Assert(False)
            MsgBox("Fehler: Wo: " & _
               ex.StackTrace & " Was: " & ex.Message)
        End Try
End Function
```

Wenn Sie das Projekt nun testen, werden Sie feststellen, dass ein Dokument nicht geöffnet wird, wenn Sie dies durch die MessageBox angeben.

Sie sehen, dass die SolidWorks Events ein sehr umfangreiches und effektives Instrument sind, um auf Aktionen des Users zu reagieren oder diese in Erfahrung zu bringen.
Zudem haben Sie an dem letzten Beispiel gelernt, wie Sie eine Aktion unterbinden können.

3 Dokumenthandling in SolidWorks

In fast allen SolidWorks API Anwendungen ist das Öffnen, Speichern und Auswerten von SolidWorks Dokumenten ein zentrales Thema. Arbeitet Ihre Anwendung zum Beispiel mit Masterdokumenten, werden diese in SolidWorks geöffnet, bearbeitet, gespeichert und anschließend in SolidWorks geschlossen.
Damit diese Grundlagen kein Hindernis für Ihre Aufgabe sind, werden Ihnen in diesem Kapitel alle nötigen Befehle der SolidWorks API erklärt um Ihre Dokumente effektiv händeln zu können.
Neben den Befehlen der SolidWorks API ist der richtige Umgang mit den Net-Klassen wichtig, welche für das File System Ihres Computers stehen.

3.1 Arbeiten mit den Net-Klassen System.IO

Dieses Kapitel soll bei weitem kein Visual Basic Buch ersetzen und auch keine Visual Basic Befehlsreferenz sein. Vielmehr geht es darum, Ihnen die wichtigsten und in diesem Kapitel benötigten Methoden zu zeigen.

Hinweis:
Weitere Hinweise erhalten Sie in der zahlreichen Fachliteratur und in zahlreichen Beiträgen im Internet. Hier nur eine kleine Auswahl:

- http://www.galileocomputing.de/openbook/visual_basic/Kapitel_12-001.htm
- http://www.vbarchiv.net/workshop/workshop72.php
- http://www.vbarchiv.net/workshop/workshop73.php

► **Link**
Mehr Informationen über die Net-Klassen System.IO

System.IO.File

Existenz einer Datei prüfen:

```
Public Shared Function Exists ( _
        path As String) As Boolean
```

Visual Basic
System.IO.
File.Exists
Befehlsreferenz

Als Parameter übergibt man folgenden Wert:

- path = Die zu überprüfende Datei.

Rückgabewert:
„true“, wenn der Aufrufer über die erforderlichen Berechtigungen verfügt und path den Namen einer vorhandenen Datei enthält, andernfalls false.

Eine Datei verschieben:

Public Shared Sub Move (_
sourceFileName As String, _
destFileName As String)

Visual Basic
System.IO.
File.Move
Befehlsreferenz

Als Parameter übergibt man dieser Funktion folgende Werte:

- sourceFileName = Der Name der zu verschiebenden Datei.
- destFileName = Der neue Pfad für die Datei.

Eine Datei löschen:

Public Shared Sub Delete (_
path As String)

Visual Basic
System.IO.
File.Delete
Befehlsreferenz

Als Parameter übergibt man dieser Funktion folgenden Wert:

- path = Der Name der zu löschenden Datei.

System.IO.Directory

Für die Bearbeitung von Verzeichnissen steht Ihnen die Net Klasse System.IO.Directory zur Verfügung. Diese enthält fast dieselben Befehle wie die System.IO.File Klasse. Anstelle von Dateien werden in dieser Klasse Verzeichnisse bearbeitet.

Neues Verzeichnis erzeugen:

Public Shared Function CreateDirectory (_
path As String) As DirectoryInfo

Visual Basic
System.IO.
Directory.
CreateDirectory
Befehlsreferenz

Als Parameter übergibt man dieser Funktion folgenden Wert:

- path = Der zu erstellende Verzeichnispfad.

Als Rückgabewert erhalten Sie eine Instanz des vom Parameter path angegebene DirectoryInfo Objekt.

System.IO.Path

Die System.IO.Path Klasse enthält Methoden, um bestimmte Bestandteile eines Dateipfads einfach zu ermitteln.

Extension des Dateipfads ermitteln:

Public Shared Function GetExtension (_
path As String) As String

Visual Basic
System.IO.
Path.GetExtension
Befehlsreferenz

Als Parameter übergibt man dieser Funktion folgenden Wert:

- path = Die Pfadzeichenfolge, aus der die Erweiterung abgerufen werden soll.

Rückgabewert
Eine String, der die Erweiterung des angegebenen Pfades (einschließlich ".") enthält.

Dateinamen mit Extension ermitteln:

Public Shared Function GetFileName (_
path As String) As String

Visual Basic
System.IO.
Path.GetFileName
Befehlsreferenz

Als Parameter übergibt man dieser Funktion folgenden Wert:

- path = Die Pfadzeichenfolge, aus der der Dateiname und die Erweiterung abgerufen werden sollen.

Rückgabewert
Ein String, bestehend aus den Zeichen nach dem letzten Verzeichniszeichen in path.

Dateinamen ohne Extension ermitteln:

Public Shared Function GetFileNameWithoutExtension (_
path As String) As String

Visual Basic
System.IO. Path.
GetFileName
WithoutExtension
Befehlsreferenz

Als Parameter übergibt man dieser Funktion folgenden Wert:

- path =Der Pfad der Datei.

Rückgabewert
Ein String, der die von GetFileName zurückgegebene Zeichenfolge enthält, ohne den letzten Punkt (.) und alle folgenden Zeichen.

Verzeichnis eines Dateipfads ermitteln:

Visual Basic
System.IO. Path.
GetDirectoryName
Befehlsreferenz

Public Shared Function GetDirectoryName (_
path As String) As String

Als Parameter übergibt man dieser Funktion folgenden Wert:

- path = Der Pfad einer Datei oder eines Verzeichnisses.

Rückgabewert
Ein String, die Verzeichnisinformationen für path enthält.

Sie sehen, die Methoden der Net-Klassen System.IO sind keine Hexerei. Die meisten Methoden erklären sich bereits aus dem Methodennamen. Mit den System.IO Klassen stellt Ihnen die Net Klassenbibliothek ein umfangreiches Werkzeug zur Verfügung, mit welchem Sie Ihre Dateien mühelos manipulieren und bearbeiten können.

Wie bereits in den zuvor behandelten Kapiteln, starten Sie die SolidWorks API Methoden mit Hilfe eines Formulars, welches Sie zunächst erstellen müssen und als Startformular in Ihrem Projekt „SldWorksDokument“ festlegen.

Fügen Sie bitte in Ihr Projekt ein neues Formular mit den Namen „SldWorksDokumenteFrm“ ein. In dieses Formular fügen Sie ein Label, einen OpenFileDialog und zunächst zwei Buttons ein (das Formular wird im laufe dieses Kapitels erweitert). Passen Sie folgende Eigenschaften der Steuerelemente an:

SldWorksDokumenteFrm:

Tabelle 3.0
Geänderte Eigenschaften des SldWorks DokumenteFrm Formular

Eigenschaft	**Wert**
ShowIcon	False
Size	330; 240
Text	SolidWorks API Schulung

Label1:

Tabelle 3.1
Geänderte Eigenschaften des Label1

Eigenschaft	**Wert**
Name	lblUeberschrift
Text	Schuler Design Automation GmbH SolidWorks API Schulung SolidWorks Objekte und Dokumente
Size	300; 60
TextAlign	MiddleCenter

Button1:

Eigenschaft	Wert
Name	cmdOeffnen
Size	110; 30
Text	Öffnen

Tabelle 3.2 Geänderte Eigenschaften des Button1

Button2:

Eigenschaft	Wert
Name	cmdSchliessen
Size	110; 30
Text	Schließen

Tabelle 3.3 Geänderte Eigenschaften des Button2

Ordnen Sie diese Steuerelemente so an, dass Ihr Formular ungefähr wie die Abbildung aussieht.

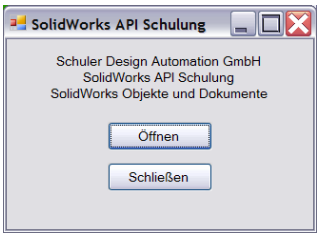


Abbildung 3.0 Das Formular SldWorks DokumenteFrm

Vergessen Sie nicht das Formular „SldWorksDokumenteFrm“ als Startformular unter „MyProject“ festzulegen.

In die Ereignisprozedur des „cmdOeffnen.Click“ Ereignis fügen Sie bitte folgenden Code ein, um eine SolidWorks Datei durch einen „Öffnen-Dialog“ auswählen zu können. Dabei erlauben Sie die Dateitypen „*.sldprt“, „*sldasm“ und „*.slddrw“. Mit Hilfe dieses Dialogs können Sie ein beliebiges SolidWorks Dokument auswählen, welches Sie in SolidWorks öffnen.

Code 3.0
Ereignisprozedur cmdOeffnen.Click

```
Private Sub cmdOeffnen_Click( _
            ByVal sender As System.Object, _
            ByVal e As System.EventArgs) _
            Handles cmdOeffnen.Click
 Try
   'Dialog aufrufen
   With Me.OpenFileDialog1
     'Keine Auswahlvorgabe
     .FileName = ""
     'Welche Dateitypen dürfen
     'in diesem Dialog ausgewählt werden
    .Filter = "SolidWorks Bauteil (*.sldprt)|*.sldprt|" & _
         "SolidWorks Baugruppe (*.sldasm)|*.sldasm|" & _
         "SolidWorks Zeichnung (*.slddrw)|*.slddrw"
     'Titel das Dialogs
     .Title = "SolidWorks Dokument öffnen"
   End With
   'Wenn eine Datei geöffnet werden soll
   If Me.OpenFileDialog1.ShowDialog() = _
        Windows.Forms.DialogResult.OK Then
     'Ausgewältes SolidWorks Dokument
     Debug.Print( _
         Me.OpenFileDialog1.FileName)
   End If
 Catch ex As Exception
   Debug.Assert(False)
   MsgBox("Fehler: Wo: " & _
      ex.StackTrace & " Was: " & ex.Message)
 End Try
End Sub
```

Wenn Sie das Projekt starten und auf den „Öffnen“ Button klicken, sollten Sie einen „Öffnen Dialog“ angezeigt bekommen.

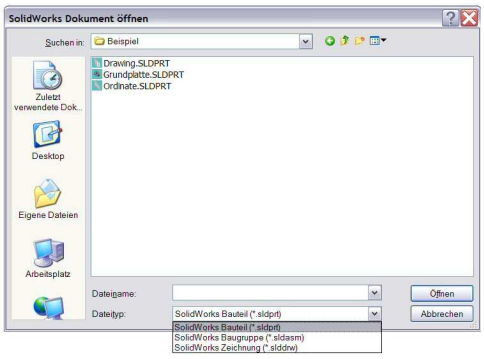


Abbildung 3.1
Windows Datei öffnen Dialog

Nach dem Sie nun alle Vorbereitungen getroffen haben, können Sie in die SolidWorks API einsteigen.

3.2 Ein Dokument in SolidWorks öffnen

Das Öffnen eines SolidWorks Dokuments erfolgt durch die Methode „OpenDoc6“.

SolidWorks API
SldWorks.
OpenDoc6

retval = SldWorks.OpenDoc6 (filename, type, options, configuration, Errors, Warnings)

Als Parameter übergeben Sie dieser Funktion folgende Werte:

- Filename = Dateipfad des SolidWorks Dokuments, welches Sie öffnen möchten.
- Type = SolidWorks Dokumententyp. Definiert durch die Enumerationsvariablen swDocumentTypes_e
- Options = Option beim Öffnen des Dokuments, wie Silent oder ReadOnly. Definiert durch die Enumerationsvariablen swOpenDocOptions_e
- Configuration = Konfiguration des SolidWorks Dokuments, welches geöffnet werden soll.
- Errors = Rückgabewert Errormeldung. Definiert durch die Enumerationsvariablen swFileLoadError_e
- Warnings = Rückgabewert Warnungsmeldung. Definiert durch die Enumerationsvariablen swFileLoadWarning_e

Als Rückgabe erhalten Sie eine Instanz des gerade geöffnetem SolidWorks Dokuments in einem „SldWorks.ModelDoc2“ Objekt.

Soviel zur Theorie, aber nun endlich wieder zur Praxis. Erstellen Sie in der Klasse „MySldWorksCls“ eine Methode „DokumentOeffnen“ des Typs „SldWorks.ModelDoc2“.

Dieser Methode fügen Sie einen Parameter hinzu um den Dateipfad des Dokumentes festzulegen, welches geöffnet wird.

Diese Methode wird Sie so allgemein wie möglich gestaltet, damit Sie diese immer verwenden können, wenn Sie ein Dokument in SolidWorks öffnen möchten.

In die Methode selbst fügen Sie drei Variable ein, welche Sie mit den Rückgabewerten der „OpenDoc6“ Methode befüllen werden.

Code 3.1 Funktionsrahmen der Funktion DokumentOeffnen

```
Friend Function DokumentOeffnen( _
        ByVal sDateipfad As String) As SldWorks.ModelDoc2
    Try
            'Benötigte Variablen
            Dim oSwModel As SldWorks.ModelDoc2
            Dim eError As SwConst.swFileLoadError_e
            Dim eWarning As SwConst.swFileLoadWarning_e
    Catch ex As Exception
            Debug.Assert(False)
            MsgBox("Fehler: Wo: " & _
                ex.StackTrace & " Was: " & ex.Message)
            Return Nothing
    End Try
End Function
```

Fügen Sie als nächstes eine weitere Variable in die Methode ein, welche den Dokumenttyp enthält. Diesen ermitteln Sie durch die Extension des Dateipfads.
Damit der übergebene Parameter und die Klassenvariable „oSwAppCls“ mit sinnvollen Inhalt belegt sind, fügen Sie zwei einfache Abfragen in die Methode ein, um diese zu überprüfen.

Code 3.2 Parameter überprüfen

```
            'SolidWorks Dokumenttyp
            Dim eDocTyp As SwConst.swDocumentTypes_e
            Dim sExtension As String
            'Existenz der übergeben Datei prüfen
            If IO.File.Exists(sDateipfad) = False Then
                'Fehler erzeugen
                Throw New Exception( _
                    "Die Datei: " & sDateipfad & _
                    " ist nicht vorhanden.")
            End If
            'SolidWorks Application Objekt prüfen
            If oSwAppCls Is Nothing Then
                'Fehler erzeugen
                Throw New Exception( _
                    "Kein SolidWorks Application " & _
                    "Objekt vorhanden")
            End If
```

Als nächstes müssen Sie den Dokumententyp ermitteln.

Doch bevor Sie dies tun, müssen Sie für die Klasse „MySldWorksCls“ eine wichtige Option festlegen.

Durch die Option „Option Compare Text“, welche Sie (wie jede Visual Basic Option) vor der „MySolidWorksCls“ Klasse einfügen, wird festgelegt, dass ein String nicht nach Groß- und Kleinschreibung unterschieden wird.
Dies ist wichtig, da Sie nicht sicher sein können, ob die Extension des Dateipfads in Groß- oder Kleinbuchstaben vorliegt.

Diese Option können Sie ebenfalls in den Projekteinstellungen unter „MyProject“ festlegen.

```
Option Strict On
Option Compare Text

Public Class MySldWorksCls
```

Code 3.3 Kompilierungs-optionen in Visual Basic

Um den Dokumententyp zu ermitteln, prüfen Sie die zuvor ermittelte Extension in einer Select Case Anweisung.

```
'Dokumenttyp ermitteln
sExtension = IO.Path.GetExtension( _
                        sDateipfad)
Select Case sExtension
    Case ".sldprt"
        eDocTyp = _
        SwConst.swDocumentTypes_e.swDocPART
    Case ".sldasm"
        eDocTyp = _
        SwConst.swDocumentTypes_e.swDocASSEMBLY
    Case ".slddrw"
        eDocTyp = _
        SwConst.swDocumentTypes_e.swDocDRAWING
    Case Else
        'Fehler erzeugen
        Throw New Exception( _
        "Ungültiger Dokumenttyp")
End Select
```

Code 3.4 Dokumententyp ermitteln und in einer Select Case Anweisung auswerten

Es wurden nun alle Parameter überprüft und der Dokumententyp ermittelt. Es ist nun also an der Zeit die SolidWorks API Methode für das Öffnen eines Dokuments aufzurufen.

```
'Überprüfungen erfolgreich
'Dokument in SolidWorks öffnen
oSwModel = oSwAppCls.OpenDoc6( _
sDateipfad, _
eDocTyp,
SwConst.swOpenDocOptions_e.swOpenDocOptions_Silent, _
"", _
CType(eError, SwConst.swFileLoadError_e), _
CType(eWarning, SwConst.swFileLoadWarning_e))
```

Code 3.5 Aufruf der OpenDoc6 Methode

Sie sehen, dass der Aufruf der „OpenDoc6“ Methode sehr umfangreich ist. Schon aus diesem Grund, aber auch wegen der Parameter Überprüfung und der noch folgenden Auswertung, ist eine allgemeine Öffnen Methode mehr als sinnvoll.
Bevor wir aber zur Auswertung der von SolidWorks erhalten Rückgabewerte kommen, betrachten wir uns die übergebenen Parameter etwas genauer.

Filename:
Als Filenamen haben Sie den Parameter „sDateipfad“ übergeben, welchen Sie zuvor auf Existenz und Dokumenttyp geprüft haben.
Type:
Als Type haben Sie die Variable „eDocTyp“ übergeben. Befüllt haben Sie diese Variable durch die Verarbeitung der Extension des im „sDateipfad“ enthaltenen Dateipfads.
Options:
Als Options wurde die Enumerationskonstante „SwConst.swOpenDocOptions_e.swOpenDocOptions_Silent“ übergeben. Diese bewirkt, dass das Dokument ohne Meldung in SolidWorks geöffnet wird. Dies ist zu 99% die sinnvollste Option, welche Sie bei einer Anwendung angeben. Schließlich soll dieses Dokument automatisch geöffnet werden.
Möglich wäre zudem die Option „swOpenDocOptions_ReadOnly“, um das Dokument schreibgeschützt zu öffnen. Sie können alle verfügbaren Optionen in der API Online Hilfe nachschlagen.
Configuration:
Bei Ihrem Dokument soll die Standardkonfiguration geöffnet werden. Dies erreichen Sie, indem Sie dem Parameter Configuration einen leeren String übergeben.
Errors:
Als Errors wurde die Variable „eError“ übergeben. Diese wird anschließend ausgewertet.
Warnings:
Als Warning wurde die Variable „eWarning „übergeben. Diese wird ebenfalls anschließend ausgewertet

Nun erfolgt die Auswertung der Rückgabewerte.
Der wichtigste Rückgabewert ist natürlich die Instanz zum „SldWorks.ModelDoc2“ Objekt. Ist dieses Nothing, können Sie davon ausgehen, dass etwas nicht geklappt hat.
Informationen, was in diesem Fall nicht geklappt hat, erhalten Sie in der Variable „eError“.

Der Autor traut es sich kaum zu sagen, aber die Informationen der „eWarning“ Variablen können Sie zunächst ignorieren.
Diese enthält Informationen, wie zum Beispiel ob das Dokument schreibgeschützt ist oder eine Referenz des Dokuments nicht gefunden wurde.
Natürlich können Sie diese Informationen nicht immer ignorieren. Es ist zum Beispiel sehr sinnvoll diese Information als optionalen Rückgabeparameter in die Methode aufzunehmen, um diesen, wenn erforderlich, auszuwerten.
In diesem Beispiel verzichten Sie zunächst auf diese Möglichkeit, werden jedoch am Ende auf diese zurückkommen.

```
'Rückgabewerte auswerten
If oSwModel Is Nothing Then
    'Fehler erzeugen
    Throw New Exception( _
    "Das Dokument: " & _
    sDateipfad & _
    " konnte nicht in SolidWorks geöffnet werden. " & _
    "Fehlerantwort: " & eError.ToString)
End If
'Es hat alles geklappt
Return oSwModel
```

Code 3.6
Rückgabewerte der OpenDoc6 Methode auswerten

Wenn alles in der Methode „DokumentOeffnen“ geklappt hat, wird als Rückgabe dieser Funktion die erzeugte Instanz der „SldWorks.ModelDoc2“ Klasse übergeben.

Nun fehlt noch der Aufruf dieser Methode.
Diesen fügen Sie in die bereits vorhandene Ereignisprozedur des „cmdOeffnen.Click“ Ereignisses ein.
In dieser haben Sie ja bereits einen „Öffnen Dialog“ eingefügt (nur für den Fall, dass Sie sich nach dem OpenDoc6 Marathon nicht mehr daran erinnern).
Wir fügen den Aufruf unserer Methode in die If Anweisung ein, welche ausgeführt wird, falls der Anwender im „Öffnen Dialog“ auf den „Öffnen“ Button klickt.

```
'Wenn eine Datei geöffnet werden soll
If Me.OpenFileDialog1.ShowDialog() = _
     Windows.Forms.DialogResult.OK Then
     'Ausgewältes SolidWorks Dokument
     Debug.Print( _
          Me.OpenFileDialog1.FileName)
     'Eigene SolidWorks Klasse deklarieren
     Dim oMySldWorks As MySldWorksCls
     'Eigene SolidWorks Klasse initialisieren
     oMySldWorks = New MySldWorksCls
     'SolidWorks initialisieren
     oMySldWorks.SolidWorksInstanz()
     'Dokument in SolidWorks öffnen
     oMySldWorks.DokumentOeffnen( _
          Me.OpenFileDialog1.FileName)
     'Eigene SolidWorks Klasse leeren
     oMySldWorks = Nothing
End If
```

Code 3.7
Aufruf der DokumentOeffnen Methode in der cmdOeffnen.Click Ereignisprozedur

Sie initialisieren durch die bereits bekannte Methode das SolidWorks Application Objekt. Wenn dies erfolgreich geschehen ist, rufen Sie die Methode „DokumentOeffnen“ auf.

Jetzt ist es soweit. Testen Sie das Projekt ausführlich. Herzlichen Glückwunsch, Sie haben per API ein Dokument in SolidWorks geöffnet. Zudem haben Sie eine relativ allgemeine Öffnen Methode geschrieben, welche Sie in Ihren Anwendungen verwenden können.

Bevor Sie mit dem nächsten Kapitel fortfahren, werde ich Sie noch etwas mit der OpenDoc6 Methode ärgern.

Die Methode „DokumentOeffnen“ wird optimiert, um sie für eine allgemeine Verwendung nahezu perfekt zu gestalten.
Deshalb fügen Sie zwei optionale Parameter ein. Einen Übergabeparameter „eOpenDocOptions“ um die Option bei Öffnen des Dokuments festlegen zu können. Als zweites einen Rückgabeparameter „eOpenWarning“, um diesen bei Bedarf auswerten zu können.
Außerdem leiten Sie in der Methode aufgetretene Fehler im Catch Abschnitt der „Try-Catch-Anweisung“ weiter, um die Informationen dieser erzeugten Fehler bereitzustellen.

Die überarbeitete Methode „DokumentOeffnen“ im Überblick folgt, aus Platzgründen, auf der nächsten Seite und in kleinerer Schriftgröße.

```
Friend Function DokumentOeffnen( _
     ByVal sDateipfad As String, _
     Optional ByVal eOpenDocOptions As SwConst.swOpenDocOptions_e = _
          SwConst.swOpenDocOptions_e.swOpenDocOptions_Silent, _
     Optional ByRef eOpenWarning As SwConst.swFileLoadWarning_e = 0) _
     As SldWorks.ModelDoc2
        Try
            'Benötigte Variablen
            Dim oSwModel As SldWorks.ModelDoc2
            Dim eError As SwConst.swFileLoadError_e
            'SolidWorks Dokumenttyp
            Dim eDocTyp As SwConst.swDocumentTypes_e
            Dim sExtension As String
            'Existenz der übergeben Datei prüfen
            If IO.File.Exists(sDateipfad) = False Then
                'Fehler erzeugen
                Throw New Exception( _
                    "Die Datei: " & sDateipfad & _
                    " ist nicht vorhanden.")
            End If
            'SolidWorks Application Objekt prüfen
            If oSwAppCls Is Nothing Then
                'Fehler erzeugen
                Throw New Exception( _
                    "Kein SolidWorks Application " & _
                    "Objekt vorhanden")
            End If
            'Dokumenttyp ermitteln
            sExtension = IO.Path.GetExtension( _
                                    sDateipfad)
            Select Case sExtension
                Case ".sldprt"
                    eDocTyp = _
                    SwConst.swDocumentTypes_e.swDocPART
                Case ".sldasm"
                    eDocTyp = _
                    SwConst.swDocumentTypes_e.swDocASSEMBLY
                Case ".slddrw"
                    eDocTyp = _
                    SwConst.swDocumentTypes_e.swDocDRAWING
                Case Else
                    'Fehler erzeugen
                    Throw New Exception( _
                    "Ungültiger Dokumenttyp")
            End Select
            'Überprüfungen erfolgreich
            'Dokument in SolidWorks öffnen
            oSwModel = oSwAppCls.OpenDoc6( _
                sDateipfad, _
                eDocTyp, eOpenDocOptions, _
                "", _
                CType(eError, SwConst.swFileLoadError_e), _
                CType(eOpenWarning, SwConst.swFileLoadWarning_e))
            'Rückgabewerte auswerten
            If oSwModel Is Nothing Then
                'Fehler erzeugen
                Throw New Exception( _
                "Das Dokument: " & _
                sDateipfad & _
                " konnte nicht in SolidWorks geöffnet werden. " & _
                "Fehlerantwort: " & eError.ToString)
            End If
            'Es hat alles geklappt
            Return oSwModel
        Catch ex As Exception
            Debug.Assert(False)
            MsgBox("Fehler: Wo: " & _
                ex.StackTrace & " Was: " & ex.Message)
            Throw New Exception( _
                "Fehler beim Öffnen des Dokuments: " & _
                sDateipfad & ".")
            Return Nothing
        End Try
End Function
```

Code 3.8
Allgemeine Methode für das Öffnen eines SolidWorks Dokuments

3.3 Ein Dokument in SolidWorks schließen

Das Schließen eines Dokuments ist schnell erklärt. Die benötigte Methode hat nicht so viele Parameter wie die „OpenDoc6“ Methode. Um genau zu sein, besitzen die Methoden einen oder keinen Parameter.
Jedoch gibt es auch beim Schließen eines Dokuments einiges zu beachten. Das liegt vor allem an den vier verschiedenen Methoden ein Dokument zu schließen, wobei zwei dieser Methode allem Anschein nach, nicht wirklich ausgereift sind.

Ich stelle Ihnen zunächst die Methode vor, welche bis lang immer funktioniert hat und in allen Anwendungen des Autors verwendet wird.

SldWorks.CloseDoc (Name)

SolidWorks API
SldWorks.
CloseDoc

Als Parameter übergibt man dieser Funktion folgenden Wert:

- Name = Name (Titel) des Dokuments, welches geschlossen wird.

In den Remarks der API Online Hilfe finden Sie einen Hinweis auf das Verhalten einer SolidWorks Hintergrundsitzung. Nach ausgiebigen Tests kann ich Ihnen jedoch sagen, dass Sie diesem Hinweis in der Online Hilfe keine Aufmerksamkeit schenken müssen. Es gab bislang keine Anwendung, in welcher diese Methode nicht funktioniert hat.
Sollten Sie andere Erfahrungen gemacht haben, freut sich der Autor jetzt schon auf Ihr Feedback.

Um diese Methode, und im weiteren Verlauf auch alle anderen, zu testen, fügen Sie eine Ereignisprozedur für das Klick Ereignis unseres „cmdSchliessen“ Buttons in unsere Formularklasse ein.

Code 3.10
Grundgerüst der cmdSchliessen.
Click
Ereignisprozedur

```
Private Sub cmdSchliessen_Click( _
    ByVal sender As System.Object, _
    ByVal e As System.EventArgs) _
    Handles cmdSchliessen.Click
    Try
        'Eigene SolidWorks Klasse deklarieren
        Dim oMySldWorks As MySldWorksCls
        'Eigene SolidWorks Klasse initialisieren
        oMySldWorks = New MySldWorksCls
        'SolidWorks initialisieren
        oMySldWorks.SolidWorksInstanz()

        'Eigene SolidWorks Klasse leeren
        oMySldWorks = Nothing
    Catch ex As Exception
        Debug.Assert(False)
        MsgBox("Fehler: Wo: " & _
            ex.StackTrace & " Was: " & ex.Message)
    End Try
End Sub
```

In dieser Ereignisprozedur initialisieren Sie, wie immer, mit der bereits bekannten Methode das SolidWorks Application Objekt. Was jetzt noch fehlt ist das aktuelle „ModelDoc2“ Objekt. Dieses ermitteln Sie in einer „Nur lesen“ Eigenschaft der „MySldWorksCls“ Klasse. In dieser initialisieren Sie die SldWorks.ModelDoc2 Klasse mit Hilfe der bereits bekannten Methode „SldWorks.ActiveDoc“.

Code 3.11
AktuellesDokument Eigenschaft

```
Friend ReadOnly Property AktuellesDokument() As _
        SldWorks.ModelDoc2
    Get
        'SolidWorks Instanz prüfen
        If oSwAppCls Is Nothing Then
            'Nothing als ungültige Rückgabe verwenden
            Return Nothing
        Else
            'Aktuelles Dokument zurückgeben
            Return CType( _
                oSwAppCls.ActiveDoc, _
                SldWorks.ModelDoc2)
        End If
    End Get
End Property
```

Als nächstes fügen Sie eine neue Methode in die Klasse „MySldWorksCls“ ein, um ein aktuell in SolidWorks geöffnetes Dokument zu schließen.

Code 3.12
Methode für das Schließen des aktuell in SolidWorks geöffneten Dokuments

```
Friend Sub DokumentSchliessen()
    Try

    Catch ex As Exception
        Debug.Assert(False)
        MsgBox("Fehler: Wo: " & _
                ex.StackTrace & " Was: " & ex.Message)
        Throw New Exception( _
        "Fehler beim Schliessen des aktuellen Dokuments.")
    End Try
End Sub
```

Das aktuell in SolidWorks geöffnete Dokument ermitteln Sie in der Methode „DokumentSchliessen“ mit der Eigenschaft „AktuellesDokument“.
Darauf rufen Sie die „SldWorks.CloseDoc“ Methode auf und übergeben dieser den Titel des geöffneten Dokuments.
Den Titel des geöffneten Dokuments ermittelt die folgende API Methode:

retval = ModelDoc2.GetTitle ()

SolidWorks API
ModelDoc2.GetTitle

Als Rückgabewert erhalten Sie den Dokumenttitel des Dokuments.

Bei der Rückgabe dieser API Methode ist zu beachten, dass diese je nach Einstellung variieren kann. Diese Einstellung legen Sie jedoch nicht in SolidWorks, sondern im Windows Explorer fest. Es handelt sich um die Ordneroption „Erweiterungen bei bekannten Datentypen ausblenden“. Diese Ordneroption legt auch in SolidWorks fest, ob die Extension Bestandteil des Titel ist.

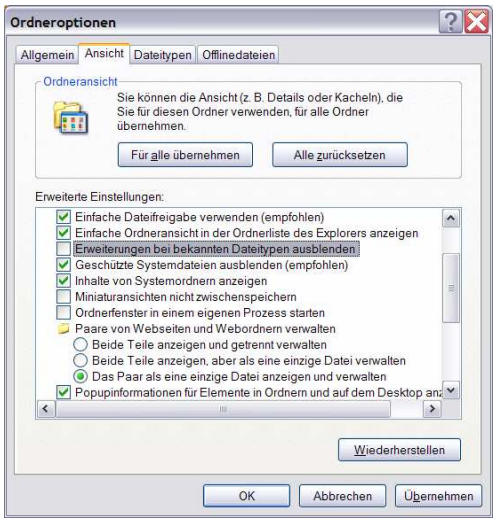


Abbildung 3.2
Ordneroptionen des Windows Explorers

Für das Schließen eines geöffneten Dokuments hat diese Besonderheit keine Auswirkung.

Jedoch sollten Sie darauf achten, wenn Sie den Titel eines Dokuments auswerten. Im Verlauf dieses Kapitels werden Sie Möglichkeiten kennen lernen wie man einen Dokumentpfad so aufschlüsseln kann, damit man immer das gewünschte Ergebnis ermittelt.

Fügen Sie also in die Methode „DokumentSchließen“ eine Abfrage der SolidWorks Instanz ein. Ist diese vorhanden rufen Sie die Methode „CloseDoc“ auf:

Code 3.13
Aufruf der CloseDoc API Methode in der Methode Dokument Schliessen

```
Friend Sub DokumentSchliessen ()
    Try
        'SolidWorks Instanz prüfen
        If oSwAppCls Is Nothing Then
            'Fehler
            Throw New Exception( _
            "Keine SolidWorks Instanz vorhanden.")
        Else
            'Aktuell in SolidWorks
            'aktives Dokument schliessen
            oSwAppCls.CloseDoc( _
                Me.AktuellesDokument.GetTitle)
        End If
    Catch ex As Exception
        Debug.Assert(False)
        MsgBox("Fehler: Wo: " & _
            ex.StackTrace & " Was: " & ex.Message)
        Throw New Exception( _
        "Fehler beim Schliessen " & _
        "des aktuellen Dokuments.")
    End Try
End Sub
```

Die Methode „DokumentSchliessen“ muss noch in der Ereignisprozedur des „cmdSchliessen.Click“ ausgeführt werden.

Code 3.14
Aufruf der Methode Dokument Schliessen

```
        'Eigene SolidWorks Klasse deklarieren
        Dim oMySldWorks As MySldWorksCls
        'Eigene SolidWorks Klasse initialisieren
        oMySldWorks = New MySldWorksCls
        'SolidWorks initialisieren
        oMySldWorks.SolidWorksInstanz()
        'Aktuelles Dokument schliessen
        oMySldWorks.DokumentSchliessen()
        'Eigene SolidWorks Klasse leeren
        oMySldWorks = Nothing
```

Betrachten Sie nun die anderen drei Methoden.

SldWorks.QuitDoc (Name)

SolidWorks API
SldWorks.QuitDoc

Als Parameter übergibt man auch hier den Dokumenttitel.
Diese Methode funktioniert ebenso zufrieden stellend wie die Methode „CloseDoc“. Der genaue Unterschied dieser beiden Methoden bleibt dem Autor ein Rätsel.
Bei den folgenden Funktionen wird es aber noch besser. Diese erledigen einfach nicht das was man vermutet und lösen sogar eine Ausnahme (Fehler) aus.

ModelDoc2.Quit

SolidWorks API
ModelDoc2.Quit

SolidWorks zeigt bei dieser Methode keine Reaktion.

ModelDoc2.Close

SolidWorks API
ModelDoc2.Close

Im Visual Studio wir eine „System.NotImplementedException“ Ausnahme ausgelöst.
Diese Ausnahme wird ausgelöst, wenn eine angeforderte Methode oder Operation nicht implementiert ist.

Damit Sie selbst ein Bild über die verschiedenen Methoden bekommen, probieren Sie alle in der Methode „DokumentSchliessen“ aus. Auch die Property „SldWorks.UserControl“ wurde in diese aufgenommen. Nur leider werden Sie bei keiner Einstellung ein anderes Ergebnis sehen, auch wenn dies in der API Online Hilfe angedeutet wird.

Code 3.14
Das Schließen eines Dokuments durch alle vier Methoden

```
Für das Testen der verschiedenen
'SolidWorks API Methoden
'UserControl Einstellung festlegen
oSwAppCls.UserControl = True
'Dokument mit ModelDoc2.Close schließen
Me.AktuellesDokument.Close()
'Dokument mit ModelDoc2.Quit schließen
Me.AktuellesDokument.Quit()
'Dokument mit SldWorks.CloseDoc schließen
oSwAppCls.CloseDoc(Me.AktuellesDokument.GetTitle)
'Dokument mit SldWorks.QuitDoc schließen
oSwAppCls.QuitDoc(Me.AktuellesDokument.GetTitle)
```

Damit Sie die einzelnen Methoden testen können, schließen Sie die anderen Aufrufe als Kommentar einfach aus. Testen Sie das Schließen eines SolidWorks Dokuments ausführlich und in allen Varianten.
Es kommt nicht sehr oft vor, dass sich API Methoden in SolidWorks anderes verhalten, als es in der API Online Hilfe beschrieben wird. Beim Schließen eines Dokuments ist es leider so.

3.4 Das Speichern eines Dokuments

Nach zwei umfangreicheren Kapiteln, beschäftigt sich dieses Kapitel mit dem Speichern eines SolidWorks Dokuments. Wichtig ist es beim Speichern eines Dokuments festzustellen, ob das Dokument bereits gespeichert wurde.

Die API Methode um ein Dokument zu speichern.

retval = ModelDoc2.Save3 (Options, Errors, Warnings)

SolidWorks API
ModelDoc2.Save3

Als Parameter übergeben Sie dieser Funktion folgende Werte:

- Options = Option beim Speichern des Dokuments, wie Silent. Definiert durch die Enumerationsvariablen swSaveAsOptions_e
- Errors = Rückgabewert Errormeldung. Definiert durch die Enumerationsvariablen swFileSaveError_e
- Warnings = Rückgabewert Warnungsmeldung. Definiert durch die Enumerationsvariablen swFileSaveWarning_e

Als Rückgabe erhalten Sie einen Boolean Wert, welcher Ihnen den Erfolg der Aktion mitteilt.

Auch für diese Aufgabe wird eine allgemeine Prozedur in der Klasse „MySldWorksCls“ erzeugt, welche Sie universell verwenden können.

Code 3.15
Funktionsrahmen der Prozedur Dokument Speichern

```
Friend Sub DokumentSpeichern()
    Try
        'SolidWorks Instanz prüfen
        If oSwAppCls Is Nothing Then
            'Fehler
            Throw New Exception( _
            "Keine SolidWorks Instanz vorhanden.")
        Else

        End If
    Catch ex As Exception
        Debug.Assert(False)
        MsgBox("Fehler: Wo: " & _
            ex.StackTrace & " Was: " & ex.Message)
        Throw New Exception( _
        "Fehler beim Speichern " & _
        "des aktuellen Dokuments.")
    End Try
End Sub
```

Bevor wir diese Methode testen, müssen Sie das „SldWorksDokumenteFrm“ Formular um zwei Buttons erweitern.

Die Eigenschaften dieser Buttons ändern Sie bitte mit dem Inhalt der folgenden Tabellen.

Button1:

Eigenschaft	Wert
Name	cmdSpeichern
Size	120; 30
Text	Speichern

Tabelle 3.4 Geänderte Eigenschaften des Button1

Button2:

Eigenschaft	Wert
Name	cmdSpeichernUnter
Size	120; 30
Text	Speichern unter

Tabelle 3.5 Geänderte Eigenschaften des Button2

Wenn Sie die Buttons im Formular ausrichten, sollte Ihr Formular wie folgt aussehen.

Abbildung 3.3 Das Formular SldWorks DokumenteFrm

Um ein Dokument in SolidWorks zu speichern, benötigen Sie eine Instanz auf die „SldWorks.ModelDoc2“ Klasse, welche durch die Methode „AktuellesDokument“ belegt wird.
Fügen Sie in die „cmdSpeichern.Click“ Ereignisprozedur wieder den Aufruf der „SolidWorksInstanz“ und „DokumentSpeichern“ Prozedur aus Ihrer Klasse „MySldWorksCls“ ein.

```
Private Sub cmdSpeichern_Click( _
     ByVal sender As System.Object, _
     ByVal e As System.EventArgs) _
     Handles cmdSpeichern.Click
     Try
         'Eigene SolidWorks Klasse deklarieren
         Dim oMySldWorks As MySldWorksCls
         'Eigene SolidWorks Klasse initialisieren
         oMySldWorks = New MySldWorksCls
         'SolidWorks initialisieren
         oMySldWorks.SolidWorksInstanz()
         'Aktuelles Dokument speichern
         oMySldWorks.DokumentSpeichern()
         'Eigene SolidWorks Klasse leeren
         oMySldWorks = Nothing
     Catch ex As Exception
         Debug.Assert(False)
         MsgBox("Fehler: Wo: " & _
             ex.StackTrace & " Was: " & ex.Message)
     End Try
 End Sub
```

Code 3.16
Ereignisprozedur des cmdSpeichern.Click Ereignis

Die Prozedur „DokumentSpeichern“ erhält, ähnlich der „DokumentOeffnen“ Methode, zwei optionale Parameter. Diese Parameter sind für die Übergabe der Speicheroption und die Auswertung der Speicherwarnungen zuständig.

Also Standardeinstellung des optionalen Parameters „eSaveAsOptions“ wird wieder die Option Silent übergeben. Falls die Methode „Save3“ nicht erfolgreich ausgeführt wird (Funktionsstatus = „false“), geben Sie den Error Rückgabewert „eError“ in einer Fehlermeldung zurück.
Die komplette allgemeine Prozedur „DokumentSpeichern“ der Klasse „MySldWorksCls“ folgt auf der nächsten Seite.

Code 3.17
Allgemeine Prozedur für das Speichern des aktuellen Dokuments

```
Friend Sub DokumentSpeichern( _
Optional ByVal eSaveAsOptions As SwConst.swSaveAsOptions_e = _
    SwConst.swSaveAsOptions_e.swSaveAsOptions_Silent, _
Optional ByRef eSaveWarning As SwConst.swFileSaveWarning_e = 0)
    Try
        'SolidWorks Instanz prüfen
        If oSwAppCls Is Nothing Then
            'Fehler
            Throw New Exception( _
            "Keine SolidWorks Instanz vorhanden.")
        Else
            'Rückgabewerte
            Dim bStatus As Boolean
            Dim eError As SwConst.swFileSaveError_e
            'Aktuelles Dokument speichern
            bStatus = Me.AktuellesDokument.Save3( _
                   eSaveAsOptions, _
                   CType(eError, SwConst.swFileSaveError_e), _
                   CType(eSaveWarning, SwConst.swFileSaveWarning_e))
            'Rückgaben auswerten
            If bStatus = False Then
                Throw New Exception( _
                "Aktuelles Dokument wurde nicht gespeichert." & _
                vbNewLine & "Errormeldung der API Methode: " & _
                eError.ToString & ".")
            End If
        End If
    Catch ex As Exception
        Debug.Assert(False)
        MsgBox("Fehler: Wo: " & _
            ex.StackTrace & " Was: " & ex.Message)
        Throw New Exception( _
        "Fehler beim Speichern " & _
        "des aktuellen Dokuments.")
    End Try
End Sub
```

Beachten Sie beim Testen das ein Dokument bereits gespeichert sein muss, wenn Sie dieses, zum Beispiel nach einer Veränderung, speichern möchten.
Wurde ein Dokument noch nicht gespeichert, müssen Sie dies mit „Speichern unter“ zuvor tun.

3.5 „Speichern unter“ eines Dokuments

Das „Speichern unter“ erfolgt in der SolidWorks API durch die Methode:

SolidWorks API
ModelDoc2.
SaveAs4

retval = ModelDoc2.SaveAs4 (Name, Version, _
Options, Errors, Warnings)

Als Parameter übergeben wir dieser Funktion folgende Werte:

- Name = Dateipfad, unter welchem das Dokument gespeichert wird.
- Version = Format, definiert durch die Enumerationsvariablen swSaveAsVersion_e
- Options = Option beim Speichern des Dokuments, wie Silent. Definiert durch die Enumerationsvariablen swSaveAsOptions_e
- Errors = Rückgabewert Errormeldung. Definiert durch die Enumerationsvariablen swFileSaveError_e
- Warnings = Rückgabewert Warnungsmeldung. Definiert durch die Enumerationsvariablen swFileSaveWarning_e

Als Rückgabe erhalten Sie einen Boolean Wert, der Ihnen den Erfolg der Methode mitteilt.

Damit Sie das „Speichern unter“ sinnvoll ausprobieren können, fügen Sie in Ihr Formular „SldWorksDokumenteFrm“ eine „SaveFileDialog“ - Komponente ein. Mit diesem Dialog wählen Sie aus, wohin Sie das Dokument speichern möchten.

Die neue Prozedur Ihrer „MySldWorksCls“ Klasse „DokumentSpeichernUnter“ besitzt einen identischen Funktionsrahmen der allgemeinen „DokumentSpeichern“ Prozedur. Neben den beiden optionalen Parametern wird zudem ein Speicherpfad benötigt. Dieser Speicherpfad legt fest, wohin und wie Ihr SolidWorks Dokument gespeichert wird.

Code 3.18
Die Prozedur Dokument SpeichernUnter

```
Friend Sub DokumentSpeichernUnter( _
    ByVal sSpeicherpfad As String, _
    Optional ByVal eSaveAsOptions As _
    SwConst.swSaveAsOptions_e = _
    SwConst.swSaveAsOptions_e.swSaveAsOptions_Silent, _
    Optional ByRef eSaveWarning As _
    SwConst.swFileSaveWarning_e = 0)
        Try
            'SolidWorks Instanz prüfen
            If oSwAppCls Is Nothing Then
                'Fehler
                Throw New Exception( _
                "Keine SolidWorks Instanz vorhanden.")
            Else
                'Rückgabewerte
                Dim bStatus As Boolean
                Dim eError As SwConst.swFileSaveError_e

            End If
        Catch ex As Exception
            Debug.Assert(False)
            MsgBox("Fehler: Wo: " & _
                ex.StackTrace & " Was: " & ex.Message)
            Throw New Exception( _
            "Fehler beim ""Speichern unter"" " & _
            "des aktuellen Dokuments.")
        End Try
End Sub
```

Diese Prozedur wird wie immer im Klickereignis des zugeordneten Buttons verwendet.
Wie bei Öffnen eines Dokuments, besitz der Anwender nun die Möglichkeit den Speicherort des SolidWorks Dokument in einem Dialog zu wählen.
Den Dateityp geben Sie mit Hilfe des SolidWorks Dokumententyps vor, um die Auswahl im „SaveFileDialog“ zu vereinfachen.

Das „Speichern unter“ erfolgt in der If-Anweisung, wenn das Dokument gespeichert werden soll.
Den Speicherpfad erhalten wir aus der „FileName“ Eigenschaft des „SaveFileDialogs“.

Code 3.19
Ereignisprozedur des cmdSpeichern Unter.Click Ereignis

```
Private Sub cmdSpeichernUnter_Click( _
    ByVal sender As System.Object, _
    ByVal e As System.EventArgs) _
    Handles cmdSpeichernUnter.Click
    Try
        Dim eSldDocTyp As SwConst.swDocumentTypes_e
        'Eigene SolidWorks Klasse deklarieren
        Dim oMySldWorks As MySldWorksCls
        'Eigene SolidWorks Klasse initialisieren
        oMySldWorks = New MySldWorksCls
        'SolidWorks initialisieren
        oMySldWorks.SolidWorksInstanz()
        'Einstellungen im SaveFileDialog
        With Me.SaveFileDialog1
            'Vorgabe ist der Titel
            'des SolidWorks Dokuments
            .FileName = oMySldWorks.AktuellesDokument.GetTitle
            'Welche Dateitypen dürfen
            'in diesem Dialog ausgewählt werden
            .Filter = "SolidWorks Bauteil (*.sldprt)|*.sldprt|" & _
                      "SolidWorks Baugruppe (*.sldasm)|*.sldasm|" & _
                      "SolidWorks Zeichnung (*.slddrw)|*.slddrw"
            'Dateityp durch den
            'Dokumenttyp des Dokuments vorbelegen
            eSldDocTyp = CType( _
                oMySldWorks.AktuellesDokument.GetType, _
                SwConst.swDocumentTypes_e)
            Select Case eSldDocTyp
                Case SwConst.swDocumentTypes_e.swDocPART
                    .FilterIndex = 0
                Case SwConst.swDocumentTypes_e.swDocASSEMBLY
                    .FilterIndex = 1
                Case SwConst.swDocumentTypes_e.swDocDRAWING
                    .FilterIndex = 2
            End Select
            'Titel das Dialogs
            .Title = "SolidWorks Dokument speichern"
        End With
        'Wenn die Datei gespeichert werden soll
        If Me.SaveFileDialog1.ShowDialog() = _
                Windows.Forms.DialogResult.OK Then
            'Speichern unter Methode
            'der eigenen Klasse ausführen
            oMySldWorks.DokumentSpeichernUnter( _
               Me.SaveFileDialog1.FileName)
        End If
        'Eigene SolidWorks Klasse leeren
        oMySldWorks = Nothing
    Catch ex As Exception
        Debug.Assert(False)
        MsgBox("Fehler: Wo: " & _
            ex.StackTrace & " Was: " & ex.Message)
    End Try
End Sub
```

Jetzt fehlt nur der eigentliche „Speichern unter“ Aufruf in der Prozedur „DokumentSpeichernUnter“, die Methode „ModelDoc2.SaveAs4“.

Als Version übergeben wir das Format swSaveAsCurrentVersion, dies ist die normale Standardeinstellung und in 99% aller Fälle die richtige Einstellung.
Die restlichen Parameter sollten Ihnen von der ModelDoc2.Save3 Methode bekannt sein.

Code 3.20
Aufruf der SaveAs4 Methode

```
'Rückgabewerte
Dim bStatus As Boolean
Dim eError As SwConst.swFileSaveError_e
'Aktuelles Dokument speichern
bStatus = Me.AktuellesDokument.SaveAs4( _
    sSpeicherpfad, _
    SwConst.swSaveAsVersion_e.swSaveAsCurrentVersion, _
    eSaveAsOptions, _
    CType(eError, SwConst.swFileSaveError_e), _
    CType(eSaveWarning, SwConst.swFileSaveWarning_e))
'Rückgaben auswerten
If bStatus = False Then
    Throw New Exception( _
    "Aktuelles Dokument wurde nicht gespeichert." & _
    vbNewLine & "Errormeldung der API Methode: " & _
    eError.ToString & ".")
End If
```

An diesem Beispiel können Sie erkennen, das die API Methode SaveAs4 selbst nicht besonders umfangreich ist.
Die Prozedur wird durch den Speicherdialog und der Fehlerauswertung „aufgebläht“.
Dafür bietet die Ereignisprozedur einen gewissen Grundkomfort, indem man den Speicherpfad auswählen kann.

Die SaveAs4 Methode führt uns direkt zum nächsten Thema. Mit dieser Methode können neben SolidWorks- auch Schnittstellenformate gespeichert werden.

3.6 Das Erzeugen eines Schnittstellenformats

SolidWorks bietet eine Vielzahl von Schnittstellenformate wie „dxf“, „Step“ oder „Parasolid“. Solche Dateien lassen sich problemlos durch die API Methode ModelDoc2.SaveAs4 erzeugen.
Entscheidend für das Format ist die Extension im Parameter „Name“ der Speicher unter Methode „SaveAs4“.
Möchten Sie zum Beispiel ein PDF Dokument erzeugen, erreichen Sie dies durch die Extension *.pdf.

Auch wenn es in diesem Skript nicht behandelt wird, sollte es kurz erwähnt werden.
Öffnen kann man ein Schnittstellenformat durch die Methode „SldWorks.LoadFile4“. Dieses Thema ist aber so umfangreich, dass es in einer Grundlagenschulung nichts zu suchen hat.

Vielleicht auch interessant, durch ein SolidWorks Add-In lassen sich Schnittstellenformate durch die Methode „SldWorks.AddFileSaveAsItem2“ in SolidWorks einfügen.
Bekanntestes Beispiel war sicherlich das SolidWorks 2005 Add-In „Save As PDF“, welches das Erstellen von PDF’s ermöglichte.

Damit Sie in Ihrem Projekt Schnittstellenformate erzeugen können, erweitern Sie die zulässigen Dateitypen.
Als Beispiel werden die Dateiformate „*.pdf“ und „*.jgp“ verwendet, da diese aus allen SolidWorks Dokumententypen erzeugt werden können.

Code 3.21
Erweiterter Dateitypfilter des Speicherdialogs

```
'Welche Dateitypen dürfen
'in diesem Dialog ausgewählt werden
.Filter = "SolidWorks Bauteil (*.sldprt)|*.sldprt|" & _
 "SolidWorks Baugruppe (*.sldasm)|*.sldasm|" & _
 "SolidWorks Zeichnung (*.slddrw)|*.slddrw|" & _
 "PDF (*.pdf)|*.pdf|" & _
 "JPEG (*.jpg)|*.jpg"
```

Es wird Sie sicherlich sehr erfreuen, wenn ich Ihnen mitteile, dass dies alle benötigten Änderungen waren. Testen Sie einfach das Projekt.

Beim Erzeugen von Schnittstellenformaten ist darauf zu achten, dass die gewünschte Konvertierung in SolidWorks möglich ist.
Eine der wichtigsten Programmierregeln in SolidWorks lautet:
Was in SolidWorks manuell nicht möglich ist, lässt sich auch durch eine API Anwendung nur sehr selten realisieren.

3.7 Ermitteln des Speicherorts eines Dokuments

Wenn Sie ein Dokument bearbeiten möchten, ist es oft nötig den Speicherort dieses Dokuments zu kennen. Besonders, wenn Sie ein bereits geöffnetes Dokument in SolidWorks übernehmen, wissen Sie dessen Speicherort zunächst nicht.

retval = ModelDoc2.GetPathName ()

SolidWorks API
ModelDoc2.
GetPathName

Als Rückgabe erhalten Sie den vollständigen Speicherpfad des Dokuments, sofern dieses gespeichert ist.

Diesen Speicherpfad können Sie durch die Methoden der Net-Klasse „System.IO.Path“ aufschlüsseln.
Es ist sinnvoller einen Dateinamen durch diese Vorgehensweise zu ermitteln, als über „ModelDoc2.GetTitle“. Wie bereits erwähnt, ist die Rückgabe dieser Methode durch eine Option im Windows Explorer abhängig. Die Methode „ModelDoc2.GetPathName“ liefert jedoch immer den vollständigen Pfad.

Die Aufschlüsslung des Speicherpfads erfolgt in vier Eigenschaften der „MySldWorksCls“ Klasse. In diesen Eigenschaften wird jeweils das aktuelle Dokument ermittelt und darauf dessen Speicherpfad. Beachten Sie, dass die Eigenschaften aufeinander aufbauen um doppelten Programmcode zu vermeiden.

Code 3.22
Eigenschaften um den Speicherpfad und dessen Bestandteile zu ermitteln

```
Friend ReadOnly Property AktuellesDokumentSpeicherpfad() As String
    Get
        'Aktuelles Dokument prüfen
        If Me.AktuellesDokument Is Nothing Then
            'Leerer String als ungültige Rückgabe verwenden
            Return ""
        Else
            'Speicherpfad zurückgeben
            Return Me.AktuellesDokument.GetPathName
        End If
    End Get
End Property

Friend ReadOnly Property AktuellesDokumentExtension() As String
    Get
        'Speicherpfad aufschlüsseln
        Return IO.Path.GetExtension(Me.AktuellesDokumentSpeicherpfad)
    End Get
End Property

Friend ReadOnly Property AktuellesDokumentVerzeichnis() As String
    Get
        'Speicherpfad aufschlüsseln
        Return IO.Path.GetDirectoryName( _
            Me.AktuellesDokumentSpeicherpfad)
    End Get
End Property

Friend ReadOnly Property AktuellesDokumentDateiName() As String
    Get
        'Speicherpfad aufschlüsseln
        Return IO.Path.GetFileName( _
            Me.AktuellesDokumentSpeicherpfad)
    End Get
End Property
```

Anschließend ein Beispielaufruf, welcher den Speicherpfad aufschlüsselt und in einer MessageBox ausgibt. Die eigene SolidWorks Klasse „MySldWorksCls“ und SolidWorks muss zuvor initialisiert, sowie in SolidWorks ein Dokument geöffnet sein.

```
'Beispielaufruf für die Ermittlung
'und Aufschlüßlung des Speicherpfad
'Ausgabe in einer MessageBox
MsgBox("Aktuelles Dokument" & _
     vbNewLine & "Pfad: " & _
     oMySldWorks.AktuellesDokumentSpeicherpfad & _
     vbNewLine & "Name: " & _
     oMySldWorks.AktuellesDokumentDateiName & _
     vbNewLine & "Extension: " & _
     oMySldWorks.AktuellesDokumentExtension & _
     vbNewLine & "Verzeichnis: " & _
     oMySldWorks.AktuellesDokumentVerzeichnis)
```

Code 3.23 Speicherpfads eines Dokuments ermitteln und aufschlüsseln

Diesen Codeabschnitt können Sie zum Beispiel in eine Ereignisprozedur des „SldWorksDokumenteFrm“ einfügen.

Wird der Code 3.23 in die Ereignisprozedur „cmdSchließen.Click“ eingefügt, wird die MessageBox angezeigt, bevor das Dokument geschlossen wird. In dieser wird der Speicherpfad aufgeschlüsselt und angezeigt.

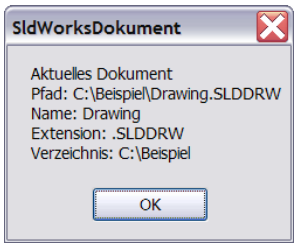


Abbildung 3.4 Aufgeschlüsselter Speicherpfad in der MessageBox

3.8 Ermitteln der Referenzen eines Dokuments

Das Thema Referenzen dürfte für jeden SolidWorks User kein unbekanntes sein.
Auch bei der Programmierung eigener Anwendungen sind Referenzen eines SolidWorks Dokuments wichtig.
Ein kleines Beispiel:
Sie möchten mit Hilfe einer Anwendung eine SolidWorks Baugruppe kopieren. Sie besitzen aber nur den Pfad der Baugruppe. Sie müssen also aus der Baugruppe die Referenzen ermitteln, um die in der Baugruppe verbauten Dokumente zu kopieren. Wenn Sie die Baugruppe und alle benötigten Dokumente kopiert haben, müssen Sie zudem die Referenzen an den neuen Speicherort anpassen.

Sie sehen, dass es nicht unwahrscheinlich ist, dass Sie in einer Anwendung mit Referenzen in Berührung kommen.

Damit Sie sich mit den API Methoden vertraut machen, mit denen Sie die Referenzen eines SolidWorks Dokuments manipulieren können, bearbeiten Sie ein kleines Beispiel.

Entpacken Sie bitte das Zip-Archiv der Beispielbaugruppe Schraubstock in ein beliebiges Verzeichnis. Anschließend ermitteln Sie dessen Referenzen und listen diese in einer ListBox auf. Darauf passen Sie die Referenzen im folgenden Kapitel an die neuen Speicherorte an.

Für dieses Beispiel über die SolidWorks Referenzen benötigen Sie ein neues Formular, welches Sie bitte in das Projekt „SldWorksDokument“ einfügen. Dieses Formular nennen Sie bitte „SldWorksReferenzenFrm“. Alle weiteren Einstellungen entnehmen Sie bitte den Tabellen:

SldWorksReferenzenFrm:

Eigenschaft	Wert
ShowIcon	False
Size	440; 350
SizeGripStyle	Show
Text	SolidWorks API Schulung

Tabelle 3.6 Geänderte Eigenschaften des SldWorks ReferenzenFrm Formular

Label1:

Eigenschaft	Wert
Name	lblUeberschrift
Text	Schuler Design Automation GmbH SolidWorks API Schulung SolidWorks Objekte und Dokumente

Tabelle 3.7 Geänderte Eigenschaften des Label1

Button1:

Eigenschaft	Wert
Name	cmdErmitteln
Anchor	Bottom, Right
Size	100; 30
Text	Ermitteln

Tabelle 3.8 Geänderte Eigenschaften des Button1

Button2:

Eigenschaft	Wert
Name	cmdAnpassen
Anchor	Bottom, Right
Size	100; 30
Text	Anpassen

Tabelle 3.9 Geänderte Eigenschaften des Button2

Button3:

Eigenschaft	Wert
Name	cmdSchliessen
Anchor	Bottom, Right
Size	100; 30
Text	Schließen

Tabelle 3.10 Geänderte Eigenschaften des Button3

ListBox1:

Eigenschaft	Wert
Name	lstReferenzen
Anchor	Top, Bottom, Left, Right
Size	410; 160

Tabelle 3.11 Geänderte Eigenschaften des ListBox1

Diese Steuerelemente richten Sie im Formular aus, damit dieses ungefähr so aussieht.

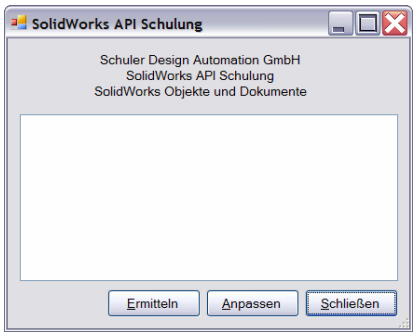


Abbildung 3.5 Das Formular SldWorks ReferenzenFrm

Als nächsten Schritt entpacken Sie das Schraubstock Archiv in ein beliebiges Verzeichnis.
In diesem Skript wird dieses Archiv in das Verzeichnis „C:\Baugruppen“ entpackt. Die Baugruppe Schraubstock besteht aus der eigentlichen Baugruppe Schraubstock.sldasm, sowie den Dokumenten:

- Backen.sldprt
- FesterBacken.sldprt
- Fuehrung.sldprt
- Grundplatte_neu.sldprt
- LoserBacken.sldprt
- Spindel.sldprt
- Spindelfuehrung.sldprt

Die Baugruppe Schraubstock werden Sie auslesen und die Referenzen ermitteln.

Die SolidWorks API bietet zwei identische Methoden, um die Referenzen eines Dokuments zu ermitteln. Die Methoden befinden sich in der „SldWorks.ModelDoc2“ und „SldWorks.SldWorks“ Klasse, wobei beim Letzteren ein Pfad übergeben wird, um das Dokument zu definieren.
Betrachten wir die Methode der „SldWorks“ Klasse:

SolidWorks API
SldWorks.
GetDocument
Dependencies2

retval = SldWorks.GetDocumentDependencies2 (document, _
traverseFlag, searchFlag, addReadOnlyInfo)

Als Parameter übergeben Sie dieser Funktion folgende Werte:

- document = Pfad des Dokuments.
- traverseFlag = Legt die Tiefe der Suche fest.
  true –alle abhängigen Dokumente durchsuchen
  flase – nur die höchste Ebene durchsuchen
- searchFlag = SolidWorks Suchregeln verwenden. Bei false wird am letzten Speicherort des Dokuments gesucht
- addReadOnlyInfo = true wenn sie die „Read-Only“ Informationen des Dokuments ermitteln möchten

Als Rückgabe erhalten Sie ein so genanntes SaveArray. In diesem SaveArray sind zwei oder drei Informationen enthalten, je nach Wert des Parameters „addReadOnlyInfo“. Der Dokumentname und –pfad des referenzierten Dokuments oder der Dokumentname und –pfad des referenzierten Dokuments sowie dessen „Nur lesen“ Status als Boolean Wert.
Wie Sie dieses SaveArray weiter verarbeiten, werden Sie in der Beispielfunktion lernen.

Nun die Methode der „ModelDoc2“ Klasse:

SolidWorks API
ModelDoc2.
GetDependencies2

retval = ModelDoc2.GetDependencies2 (traverseFlag, _
searchFlag, addReadOnlyInfo)

Sie sehen, dass die Parameter dieser Methode identisch mit der „SldWorks.GetDocumentDependencies2“ sind. Nur der Parameter „document“ fehlt, da das Dokument durch die Instanz des „ModelDoc2“ Objekt bereits festgelegt ist. Auch bei der Rückgabe dieser Methode handelt es sich wieder um ein SaveArray.

Es gibt bei beiden Methoden jedoch einen wichtigen Unterschied, welchen Sie unbedingt beachten sollten. Wenn Sie die Referenzen durch die Methode „GetDependencies2“ der „ModelDoc2“ Klasse ermitteln, ist das Dokument in SolidWorks geöffnet. Dies hat zur Folge, dass SolidWorks die Referenzen durch Suchpfade und interne Vorgehensweisen bereits angepasst hat.

Nach diesem theoretischem Teil und den Vorbereitungen machen Sie sich ans Werk, die Referenzen der Baugruppe Schraubstock zu ermitteln. Für das Ermitteln von Referenzen schreiben Sie eine allgemeine Methode in die „MySldWorksCls“ Klasse.

Code 3.24 Funktionsrahmen der ReferenzenEines Dokuments Ermitteln Methode

```
Friend Function ReferenzenEinesDokumentsErmitteln( _
  Optional ByVal sDokumentPfad As String = "", _
  Optional ByVal bAuchReadOnlyInfo As Boolean = False) As _
  System.Collections.Generic.List(Of String)
        Try
            'Dummy SaveArray
            Dim oDummyArray() As Object
            'Referenzliste
            Dim scReferenzen As _
                System.Collections.Generic.List(Of String)
            'SolidWorks Application Objekt prüfen
            If oSwAppCls Is Nothing Then
                'Fehler erzeugen
                Throw New Exception( _
                "Keine SolidWorks Instanz vorhanden.")
            End If
            'Existenz der übergeben Datei prüfen
            If IO.File.Exists(sDokumentPfad) = False Then
                'Fehler erzeugen
                Throw New Exception( _
                    "Die Datei: " & sDokumentPfad & _
                    " ist nicht vorhanden.")
            End If
            Return scReferenzen
        Catch ex As Exception
            Debug.Assert(False)
            MsgBox("Fehler: Wo: " & _
                ex.StackTrace & " Was: " & ex.Message)
            Return Nothing
        End Try
End Function
```

Als optionale Parameter besitzt die Funktion „ReferenzenEinesDokumentsErmitteln“ den Dokumentpfad und die Einstellung „bAuchReadOnlyInfo“, die festlegt welche Informationen ermittelt werden. Der Parameter „DokumentPfad“ wird im ersten Funktionsabschnitt, wie die SolidWorks Instanz geprüft.
Als Rückgabeparameter dient eine generische String Auflistung.

Nach der Parameterüberprüfung rufen Sie nun die SolidWorks API Methode auf, um die Referenzen zu ermitteln.

```
'Referenzierte Dokumente ermitteln
oDummyArray = CType( _
    oSwAppCls.GetDocumentDependencies2( _
    sDokumentPfad, True, _
    False, bAuchReadOnlyInfo), [Object]())
'Überprüfen ob das Dokument
'überhaupt Referenzen hat!
If IsArray(oDummyArray) = False Then
    'Wenn keine Referenzen gefunden wurden,
    'war das kein Fehler der Funktion!
    Return Nothing
End If
```

Code 3.25 Aufruf der SldWorks. GetDocument Dependencies2 Methode

Als Parameter übergeben Sie der „SldWorks.GetDocumentDependencies2“ Methode den Dokumentpfad, sowie die Optionen der Referenzermittlung. Durch diese Optionen wird festgelegt, dass wir alle abhängigen Dokumente auslesen und die SolidWorks Suchregeln nicht verwenden möchten. Zudem werden die ReadOnly Informationen nicht ermittelt.
Mit der Rückgabe befüllen wir das DummyArray, welches die „CType“ Umwandlung benötigt.

Nun geht es darum, die Informationen dieses SaveArray aufzuschlüsseln. Die Informationen der Dokumentnamen werden ignoriert, da die String Liste nur die Dokumentpfade enthalten soll. Sollte die Option „addReadOnlyInfo“ auf den Wert „True“ stehen, müssen Sie die Verarbeitung des SaveArray anpassen und zudem die „Nur lesen“ Information ignorieren.

Betrachten wir uns zum Verständnis das SaveArray in der Visual Studio Schnellüberwachung.
Dazu müssen Sie die Methode „ReferenzenEinesDokumentsErmitteln“ debuggen.
Dies geschieht im „cmdErmitteln.Click“ Ereignis, wobei Ihnen die Inhalte der Ereignisprozedur bekannt sein müssten.

```
   Private Sub cmdErmitteln_Click( _
            ByVal sender As System.Object, _
            ByVal e As System.EventArgs) _
            Handles cmdErmitteln.Click
    Try
       Dim scReferenzen As _
                System.Collections.Generic.List(Of String)
       Dim sSchraubstockPfad As String
       'Baugruppe festlegen
       sSchraubstockPfad = _
                "C:\Baugruppen\Schraubstock" & _
                "\Schraubstock.sldasm"
       'Eigene SolidWorks Klasse deklarieren
       Dim oMySldWorks As MySldWorksCls
       'Eigene SolidWorks Klasse initialisieren
       oMySldWorks = New MySldWorksCls
       'SolidWorks initialisieren
       oMySldWorks.SolidWorksInstanz()
       'Referenzen des Dokuments speichern
       scReferenzen = _
              oMySldWorks.ReferenzenEinesDokumentsErmitteln( _
                sSchraubstockPfad)
       'Rückgabe prüfen
       If Not scReferenzen Is Nothing Then

       End If
       'Eigene SolidWorks Klasse leeren
       oMySldWorks = Nothing
    Catch ex As Exception
       Debug.Assert(False)
       MsgBox("Fehler: Wo: " & _
                ex.StackTrace & " Was: " & ex.Message)
    End Try
End Sub
```

Code 3.26
Ereignisprozedur des cmdErmitteln.Click Ereignis

Wir fügen in die Funktion „ReferenzenEinesDokumentsErmitteln“, bei der Auswertung des „oDummyArray“, einen Haltepunkt ein, und lassen uns den Inhalt dieser Variablen in der Schnellüberwachung anzeigen.

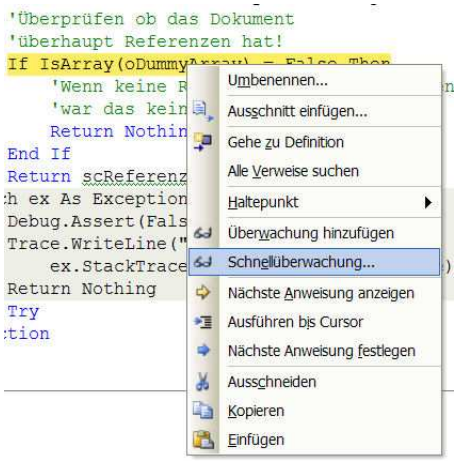


Abbildung 3.6
Die Variable oDummyArray in der Schnell-überwachung anzeigen

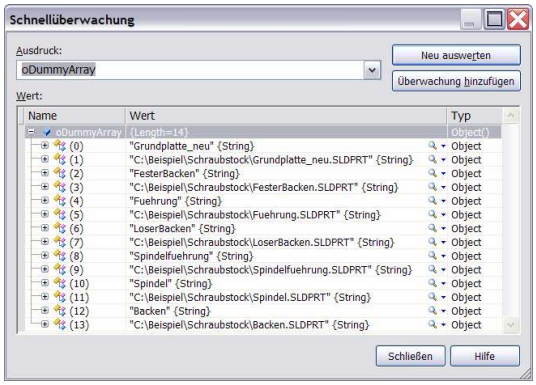


Abbildung 3.7
Die Variable oDummyArray im Schnell-überwachung Dialog

Wie Sie im Schnellüberwachung Dialog erkennen können, ist bei dem Parameter „addReadOnlyInfo“ = „False“, jeder gerade Index dieses SaveArray ein Dokumentname und jeder ungerade Index ein Dokumentpfad.

Für unser Ziel müssen wir also nur die ungeraden Indizes beachten, um die Dokumentpfade zu ermitteln.
Den Wert des Parameter „addReadOnlyInfo“ können Sie jedoch nicht außer ach lassen, da sich dieser auf den Inhalt des SaveArrays auswirkt.
Deshalb wird in der Methode „ReferenzenEinesDokumentsErmitteln“ eine weitere Variable benötigt, welcher diesen Parameter in der Verarbeitung des SaveArrays berücksichtigt.

Die Verarbeitung des SaveArrays erfolgt in einer For-Next-Schleife, in welcher die String Liste mit dem Dokumentpfad befüllt wird.
Die Schleife beginnt immer mit dem Index 1 und wird abhängig vom Parameter „addReadOnlyInfo“ und der dadurch beeinflussten Variable „iStep“ in zweier oder dreier Schritten durchlaufen.

Code 3.27
Auswertung des SaveArrays und befüllen der String Liste

```
'Parameter addReadOnlyInfo berücksichtigen
If bAuchReadOnlyInfo Then
    iStep = 3
Else
    iStep = 2
End If
'Stringliste initialisieren
scReferenzen = New _
    System.Collections.Generic.List(Of String)
'SaveArray aufschlüsseln
'und Dokumentpfade verarbeiten
For i As Integer = 1 To _
    oDummyArray.GetUpperBound(0) Step iStep
    scReferenzen.Add(CStr(oDummyArray(i)))
Next
```

Was nun fehlt, ist das Anzeigen der Referenzen in der ListBox. Hierfür muss die Ereignisprozedur erweitert werden.

```
'Referenzen des Dokuments speichern
scReferenzen = _
    oMySldWorks.ReferenzenEinesDokumentsErmitteln( _
    sSchraubstockPfad)
'Rückgabe prüfen
If Not scReferenzen Is Nothing Then
    'Listbox befüllen
    Me.lstReferenzen.Items.Clear()
    For Each s As String In scReferenzen
        Me.lstReferenzen.Items.Add(s)
    Next
End If
```

Code 3.28
Erweiterung der Ereignisprozedur cmdErmitteln.Click um die Referenzen in der ListBox anzuzeigen

Wenn Sie nun das Projekt starten und die Funktion testen, werden die Referenzen des Schraubstocks in der Listbox angezeigt.

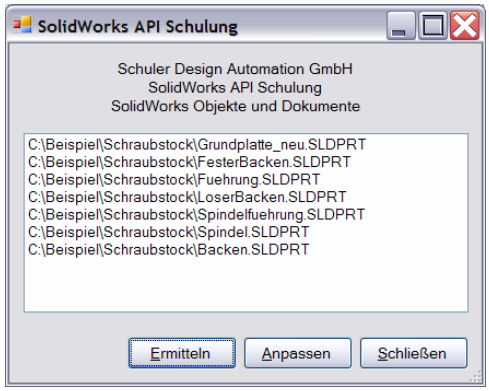


Abbildung 3.8
Die Referenzen des Schraubstocks im SldWorks ReferenzenFrm

Die Referenzen selbst sollten auch bei Ihnen auf „C:\Beispiel\Schraubstock“ verweisen. Dies ist der Speicherort, in welchem die SolidWorks Dokumente des Zip-Archivs erzeugt wurden.

Als nächsten folgt die zweite Methode „ModelDoc2.GetDependencies2“, dessen Rückgabe Sie sicherlich überraschen wird.

Um dies zu realisieren, erweitern Sie die Methode „ReferenzenEinesDokumentsErmitteln“, damit Sie die Ermittlung der Referenzen eines Dokuments an den übergebenen Parameter abhängig machen.

Die Methode stellt somit wieder eine allgemeine Lösung dar. Sie sollten die Methode durch Ihre bisherigen Kenntnisse problemlos verstehen.
Ist der optionale Parameter „sDokumentPfad“ leer, also belegt mit einem Leer String, wird versucht die Referenzen mit Hilfe der „ModelDoc2“ Klasse zu ermitteln, welche in diesem Fall mit dem aktuell in SolidWorks aktiven Dokument initialisiert wird.

(Die Methode wurde aus Platzgründen in einer kleineren Schriftart auf der folgenden Seite abgedruckt)

Code 3.29
Die allgemeine Methode Referenzen EinesDokuments Ermitteln

```
''' <summary>
''' Ermittelt die Referenzen eines SolidWorks Dokument
''' und gibt diese als String Liste zurück.
''' Das Dokument kann durch den Optionalen Parameter
''' sDokumentPfad angegeben werden. Ist dies nicht der Fall
''' werden die Referenzen des aktuell in
''' SolidWorks geöffneten Dokuments ermittelt.
''' </summary>
''' <param name="sDokumentPfad">Optionaler Parameter.
''' Wird ein Dokumentpfad übergeben,
''' werden die Referenzen dieses Dokuments ermitteln</param>
''' <param name="bAuchReadOnlyInfo">Optionaler Parameter.
''' Bestimmt ob die ReadOnly Informationen ermittelt werden.
''' Diese sind jedoch nie in der Stringliste enthalten</param>
''' <returns>Referenzen des ausgelesenen Dokuments.
''' Wenn dieses keine Referenzen besitz Nothing</returns>
''' <remarks></remarks>
Friend Function ReferenzenEinesDokumentsErmitteln( _
    Optional ByVal sDokumentPfad As String = "", _
    Optional ByVal bAuchReadOnlyInfo As Boolean = False) As _
    System.Collections.Generic.List(Of String)
    Try
        Dim iStep As Integer
        'Dummy SaveArray
        Dim oDummyArray() As Object
        'Referenzliste
        Dim scReferenzen As _
            System.Collections.Generic.List(Of String)
        'SolidWorks Application Objekt prüfen
        If oSwAppCls Is Nothing Then
            'Fehler erzeugen
            Throw New Exception( _
            "Keine SolidWorks Instanz vorhanden.")
        End If
        'Existenz der übergeben Datei prüfen
        If IO.File.Exists(sDokumentPfad) = False Then
            'Über das ModelDoc2 ermitteln
            oDummyArray = CType( _
                Me.AktuellesDokument.GetDependencies2( _
                True, False, False), [Object]())
        Else
            'Über die SolidWorks Application
            oDummyArray = CType( _
                oSwAppCls.GetDocumentDependencies2( _
                sDokumentPfad, True, _
                False, False), [Object]())
        End If
        'Überprüfen ob das Dokument
        'überhaupt Referenzen hat!
        If IsArray(oDummyArray) = False Then
            'Wenn keine Referenzen gefunden wurden,
            'war das kein Fehler der Funktion!
            Return Nothing
        End If
        'Parameter addReadOnlyInfo berücksichtigen
        If bAuchReadOnlyInfo Then
            iStep = 3
        Else
            iStep = 2
        End If
        'Stringliste initialisieren
        scReferenzen = New _
            System.Collections.Generic.List(Of String)
        'SaveArray aufschlüsseln
        'und Dokumentpfade verarbeiten
        For i As Integer = 1 To _
            oDummyArray.GetUpperBound(0) Step iStep
            scReferenzen.Add(CStr(oDummyArray(i)))
        Next
        Return scReferenzen
    Catch ex As Exception
        Debug.Assert(False)
        MsgBox("Fehler: Wo: " & _
         ex.StackTrace & " Was: " & ex.Message)
        Return Nothing
    End Try
End Function
```

Damit Sie die Ermittlung der Referenzen über die Methode „ModelDoc2.GetDependencies2“ realisieren, müssen Sie die Ereignisprozedur des „cmdErmitteln.Click“ Ereignis anpassen.
Dies ist schnell erledigt, indem Sie die Variable sSchraubstockPfad nicht mit dem eigentlichen Speicherpfad der Baugruppe Schraubstock, sondern mit Nothing, belegen.
Bevor Sie das Projekt starten und die Methode testen, muss der Schraubstock natürlich in SolidWorks geöffnet werden.

Betrachten Sie die ermittelten Referenzen etwas genauer.

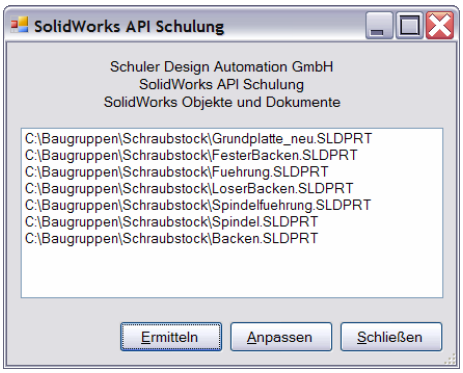


Abbildung 3.9
Die Referenzen des Schraubstocks im SldWorks ReferenzenFrm

Alle Referenzen verweisen nun auf den Speicherort „C:\Baugruppen\Schraubstock“.
Wieso das, Sie haben doch nichts geändert? Richtig!
Sie können die Funktion wieder mit der Angabe eines Dokumentpfads testen. Sie werden feststellen, dass die Referenzen in diesem Fall wieder auf „C:\Beispiel\Schraubstock“ verweisen.

Dies liegt einfach daran, ob das Dokument in SolidWorks geöffnet ist oder nicht. Denn beim Öffnen des Dokuments passt SolidWorks die Referenzen automatisch an, um die Baugruppe überhaupt vollständig zu laden.
Der Speicherort einer Baugruppe ist immer ein Suchpfad für SolidWorks, in denen nach Referenzen gesucht wird.

Die Referenzen bleiben allerdings bis zum Speichern unverändert. Wenn Sie also den Schraubstock speichern, werden die Referenzen darauf auch auf „C:\Baugruppen\Schraubstock“ verweisen, wenn diese nicht in SolidWorks geöffnet ist.

Es gibt allerdings auch eine API Methode, um die Referenzen in einem SolidWorks Dokument zu ändern.
Löschen Sie bitte deshalb den Schraubstock und entpacken das Zip-Archiv erneut, damit die Referenzen wieder auf „C:\Beispiel\Schraubstock“ verweisen.

3.9 Ändern einer Referenz

Das Ändern einer Referenz wird durch folgende API Methode realisiert:

SolidWorks API
SldWorks.Replace Referenced Document

```
retval = SldWorks.ReplaceReferencedDocument ( _
                referencingDocument, _
                referencedDocument, newReference)
```

Als Parameter übergeben Sie dieser Funktion folgende Werte:

- referencingDocument = Pfad des Dokuments, welches die Referenz besitz. Dieses Dokument darf in SolidWorks nicht geöffnet sein.
- referencedDocument = Alter Referenzpfad, welcher ersetz werden soll.
- newReference = Neuer Pfad, durch welchem die alte Referenz ersetzt wird.

Damit eine Referenz ersetzt wird, muss das Dokument, welches die Referenz besitzt, vorhanden sein. Der alte Referenzpfad muss im Dokument hinterlegt und die neue Referenz vorhanden sein. Wichtig ist, dass in SolidWorks keinerlei Überprüfung erfolgt. Es ist SolidWorks also egal, ob die neue Referenz Sinn macht oder nicht. Dies bedeutet, dass Sie ohne weiteres eine Platte durch eine Welle ersetzen können, auch dann, wenn die Platte mit Verknüpfungen in einer Baugruppe verbaut ist. Die einzige Einschränkung liegt darin, dass nur gleiche Dokumenttypen ausgetauscht werden können. Also nur eine Baugruppe durch eine Baugruppe oder ein Bauteil durch ein Bauteil.

In der Schraubstock Baugruppe möchten Sie die Referenzen ersetzen, wenn diese auf „C:\Beispiel\Schraubstock“ verweisen. Für diesen Zweck ermitteln Sie die Referenzen des Schraubstocks, überprüfen diese und setzen diese auf die Bauteile im Verzeichnis „C:\Baugruppe\Schraubstock“.

Für das Auslesen der Referenzen benutzen Sie die gleichen Funktionsaufrufe wie in der „cmdErmitteln.Click“ Ereignisprozedur. Die Referenzen werden allerdings nicht in der Liste angezeigt, sondern überprüft und ggf. verbessert.

Fügen Sie also in das „cmdAnpassen.Click“ Ereignis folgenden bekannten Code ein:

```
Private Sub cmdAnpassen_Click( _
    ByVal sender As System.Object, _
    ByVal e As System.EventArgs) Handles cmdAnpassen.Click
    Try
        Dim scReferenzen As _
                System.Collections.Generic.List(Of String)
        Dim sSchraubstockPfad As String
        'Baugruppe festlegen
        sSchraubstockPfad = _
                "C:\Baugruppen\Schraubstock" & _
                "\Schraubstock.sldasm"
        'Eigene SolidWorks Klasse deklarieren
        Dim oMySldWorks As MySldWorksCls
        'Eigene SolidWorks Klasse initialisieren
        oMySldWorks = New MySldWorksCls
        'SolidWorks initialisieren
        oMySldWorks.SolidWorksInstanz()
        'Referenzen des Dokuments speichern
        scReferenzen = _
           oMySldWorks.ReferenzenEinesDokumentsErmitteln( _
           sSchraubstockPfad)
        'Rückgabe prüfen
        If Not scReferenzen Is Nothing Then

        End If
        'Eigene SolidWorks Klasse leeren
        oMySldWorks = Nothing
    Catch ex As Exception
        Debug.Assert(False)
        MsgBox("Fehler: Wo: " & _
             ex.StackTrace & " Was: " & ex.Message)
    End Try
End Sub
```

Code 3.30
Ereignisprozedur cmdAnpassen.Click

Bei der Referenzenauswertung prüfen Sie das Verzeichnis, auf welches eine Referenz verweist. Ist dies „C:\Beispiel\Schraubstock“, ersetzen Sie die alte Referenz durch das entsprechende Dokument in „C:\Baugruppe\Schraubstock“.

```
'Rückgabe prüfen
If Not scReferenzen Is Nothing Then
    'Referenzen auswerten
    For Each sReferenz As String In scReferenzen
        If System.IO.Path.GetDirectoryName(sReferenz) = _
                "C:\Beispiel\Schraubstock" Then
            oMySldWorks.MySldWorks.ReplaceReferencedDocument( _
                sSchraubstockPfad, sReferenz, _
                "C:\Baugruppe\Schraubstock\" & _
                System.IO.Path.GetFileName(sReferenz))
        End If
    Next
End If
```

Code 3.31
Auswertung der ermittelten Referenzen und ersetzen bei einer erfüllten Bedingung

Der API Methode „SldWorks.ReplaceReferencedDocument“ übergeben Sie den Schraubstockpfad, den alten ermittelten Referenzpfad und den neuen Referenzpfad. Dieser setzt sich aus dem neuen Verzeichnis und dem alten Dokumentnamen zusammen.

Wenn Sie diese Methode testen, ermitteln Sie zuvor die Ausgangsreferenzen des Schraubstocks. Passen Sie die Referenzen darauf an, und ermitteln Sie die Referenzen erneut. Diese sollten danach auf „C:\Baugruppe\Schraubstock“ verweisen.

Gehen Sie beim Ersetzen von Referenzen immer mit großer Sorgfalt vor, da Sie umfangreiche Baugruppen durch wenige Zeilen Code unbrauchbar machen können.

3.10 Dokumenteigenschaften ermitteln

Neben den Referenzen eines Dokuments sind die Eigenschaften ein häufig interessantes Thema. Dokumenteigenschaften sind Dokumentvariablen, welche allgemeine Informationen oder Merker für Ihre Anwendung beinhalten können.

Diese Dokumentvariablen können Sie mit Hilfe einer Instanz auf die „SldWorks.ModelDoc2“ Klasse ermitteln. Dabei gibt es Unterschiede zwischen den Dokument- und Konfigurationseigenschaften. Also Eigenschaften, welche für dass gesamte Dokument oder einer bestimmten Konfiguration eines Dokuments gelten.
Da Konfigurationen in dieser SolidWorks API Grundlagenschulung nicht berücksichtigt werden, befasst sich dieses und das folgende Kapitel nur mit den konfigurationsunabhängigen Dokumenteigenschaften.

Als Ziel dieses Kapitels wird die Aufgabe realisiert, dass alle Dokumenteigenschaften des aktuell in SolidWorks geöffneten Dokuments in einer Listbox, zusammen mit ihrem Inhalt, angezeigt werden.

Für diese Aufgabe benötigen Sie wieder ein neues Formular, welches Sie bitte in das Projekt „SldWorksDokument“ einfügen. Dieses Formular nennen Sie bitte „SldWorksEigenschaftenFrm“. Alle weiteren Einstellungen entnehmen Sie bitte den Tabellen:

SldWorksReferenzenFrm:

Eigenschaft	Wert
ShowIcon	False
Size	440; 350
SizeGripStyle	Show
Text	SolidWorks API Schulung

Tabelle 3.12 Geänderte Eigenschaften des SldWorks EigenschaftenFrm Formular

Label1:

Eigenschaft	Wert
Name	lblUeberschrift
Text	Schuler Design Automation GmbH SolidWorks API Schulung SolidWorks Objekte und Dokumente

Tabelle 3.13 Geänderte Eigenschaften des Label1

Button1:

Eigenschaft	Wert
Name	cmdErmitteln
Anchor	Bottom, Right
Size	100; 30
Text	Ermitteln

Tabelle 3.14 Geänderte Eigenschaften des Button1

Button2:

Eigenschaft	Wert
Name	cmdAnpassen
Anchor	Bottom, Right
Size	100; 30
Text	Anpassen

Tabelle 3.15 Geänderte Eigenschaften des Button2

Button3:

Eigenschaft	Wert
Name	cmdSchliessen
Anchor	Bottom, Right
Size	100; 30
Text	Schließen

Tabelle 3.16 Geänderte Eigenschaften des Button3

ListBox1:

Eigenschaft	Wert
Name	lstEigenschaften
Anchor	Top, Bottom, Left, Right
Size	410; 160

Tabelle 3.17 Geänderte Eigenschaften der ListBox1

Wie Sie an der Abbildung des neuen Formulars „SldWorksEigenschaftenFrm“ erkennen können, ist dieses Formular identisch aufgebaut, wie das „SldWorksReferenzenFrm“ Formular.

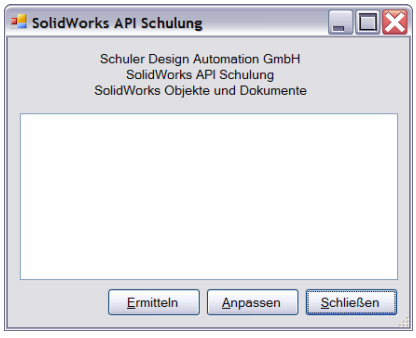


Abbildung 3.10 Das neue Startformular SldWorks EigenschaftenFrm

Auch von der Vorgehensweise werden Sie wieder ähnlich vorgehen. Durch einen Klick auf den Button „Ermitteln“ lesen Sie alle Dokumenteigenschaften eines SolidWorks Dokuments aus. Mit Hilfe des Buttons „Anpassen“ werden Sie im nächsten Kapitel die Dateieigenschaften manipulieren.
Vergessen Sie bitte nicht, das Startformular auf das „SldWorksEigenschaftenFrm“ Formular umzustellen.

Die Ermittlung der Dokumenteigenschaften bringt für die SolidWorks API Grundlagenschulung einen positiven Nebeneffekt mit sich.

In der SolidWorks Version 2007 sind neue API Methoden in die SolidWorks Type Library hinzugekommen, mit welchen die Dokumenteigenschaften ermittelt werden.

Neben diesen neuen API Methoden und die dadurch benötigten SolidWorks Objekte werden Sie sich in diesem Kapitel auch mit den älteren API Methoden beschäftigen, welche immer dann zum Einsatz kommen müssen, wenn Ihre Anwendung in einer älteren SolidWorks Version verwendet wird.

Aus diesem Grund werden Sie die Eigenschaft „VersionInJahr“ des Kapitels 1.8 nutzen. Die verschiedenen Möglichkeiten werden in zwei voneinander unabhängigen Methoden in Ihrer Klasse „MySldWorksCls“ verwendet.
In einer allgemeinen Methode wird am Schluss des Kapitels die richtige Methode für die richtige SolidWorks Version verwendet, indem zwischen den Versionen, vor und ab SolidWorks 2007, unterschieden wird.

Zu Beginn dieses Kapitels beschäftigen Sie sich mit den API Methoden, welche bis SolidWorks 2006 aktuell waren.

In der Ereignisprozedur „cmdErmitteln.Click“ erfolgt wie in den vorangegangenen Kapiteln die Initialisierung der SolidWorks Application und später der eigentliche Aufruf der neuen Methode „DokumentEigenschaftenUndWertErmittelnVorSw2007“.

Der Funktionsrahmen dieser Methode fügen Sie in Ihre Klasse „MySldWorksCls“ ein:

Code 3.32 Funktionsrahmen der Methode Dokument EigenschaftenUnd WertErmitteln VorSw2007

```
Friend Function DokumentEigenschaftenUndWertErmittelnVorSw2007( _
            Optional ByVal sKonfiguration As String = "") As _
            System.Collections.Generic.Dictionary( _
            Of String, String)
        Try
            'Dictionary für die ermittelten
            'Dokumenteigenschaften und dessen Wert
            Dim scEigenschaftenUndWert As _
                System.Collections.Generic.Dictionary( _
                Of String, String)
            'Stringarray für die vorhandenen
            'Dokumenteigenschaften des Dokuments
            Dim sEigenschaften() As String
            'SolidWorks Application Objekt prüfen
            If oSwAppCls Is Nothing Then
                'Fehler erzeugen
                Throw New Exception( _
                "Keine SolidWorks Instanz vorhanden.")
            End If

            Return scEigenschaftenUndWert
        Catch ex As Exception
            Debug.Assert(False)
            MsgBox("Fehler: Wo: " & _
             ex.StackTrace & " Was: " & ex.Message)
            Return Nothing
        End Try
End Function
```

Die ermittelten Dokumenteigenschaften werden in der Methode „DokumentEigenschaftenUndWertErmittelnVorSw2007“ in der generischen Dictionary mit dem Name „scEigenschaftenUndWert“ gespeichert. Im Schlüssel (Key) wird der Name der Dokumenteigenschaft und im Wert (Value) wird der Inhalt der Dokumenteigenschaft abgelegt.

Die SolidWorks API Methode für das Ermitteln der vorhandenen Dokumenteigenschaften in einem SolidWorks Dokument lautet:

retval = ModelDoc2.GetCustomInfoNames2 (configuration)

Als Parameter übergeben Sie dieser Funktion folgende Werte:

- configuration = Konfiguration, welche die Eigenschaften enthält oder ein Leerstring für konfigurationsunabhängige Dokumenteigenschaften.

SolidWorks API ModelDoc2. GetCustom InfoNames2

Als Rückgabe erhalten Sie ein SaveArray mit den Namen aller Dokumenteigenschaften.

Das SaveArray können Sie bei der Methode „ModelDoc2.GetCustomInfoNames2“ direkt in ein Stringarray umwandeln.

Dieses Stringarray beinhaltet darauf die Namen aller Eigenschaften für das aktuelle SolidWorks Dokument. Mit Hilfe einer For-Next-Schleife werden alle Eigenschaften gespeichert und dessen Wert ermittelt.

Code 3.33
Alle Dokument-eigenschaften ermitteln und in einer Schleife berücksichtigen

```
'Dokumenteigenschaften des aktuellen
'Dokuments ermitteln
sEigenschaften = CType( _
     Me.AktuellesDokument.GetCustomInfoNames2( _
     sKonfiguration), String())
'Dictionary initialisieren
scEigenschaftenUndWert = New _
     System.Collections.Generic.Dictionary( _
     Of String, String)
'Alle Eigenschaften berücksichtigen
'und dessen Wert ermitteln
For iEigenschaft As Integer = _
     0 To sEigenschaften.GetUpperBound(0)

Next
```

Der Inhalt (Wert) einer Dokumenteigenschaft wird über die SolidWorks API mit der folgenden Methode ermittelt:

SolidWorks API
ModelDoc2.
GetCustom
InfoValue

retval = ModelDoc2. GetCustomInfoValue(_
configuration, FieldName)

Als Parameter übergeben Sie dieser Funktion folgende Werte:

- configuration = Konfiguration, welche die Eigenschaften enthält oder ein Leerstring für konfigurationsunabhängige Dokumenteigenschaften.
- FieldName = Name der Dokumenteigenschaft

Als Rückgabe erhalten Sie den Inhalt der übergebenen Dokumenteigenschaften.

Der Inhalt der Dokumenteigenschaft wird direkt ermittelt und als Wert in der Dictionary abgelegt.

Code 3.34
Alle Werte der Dokument-eigenschaften ermitteln

```
'Alle Eigenschaften berücksichtigen
'und dessen Wert ermitteln
For iEigenschaft As Integer = _
            0 To sEigenschaften.GetUpperBound(0)
    'Eigenschaften und dessen ermitteln
    'Wert als Eintrag in die Dictionary
    scEigenschaftenUndWert.Add( _
        sEigenschaften(iEigenschaft), _
        Me.AktuellesDokument.GetCustomInfoValue( _
        sKonfiguration, sEigenschaften(iEigenschaft)))
Next
```

Das SolidWorks Bauteil Grundplatte der Baugruppe Schraubstock besitz einige Dokumenteigenschaften.
Um die Methode „DokumentEigenschaftenUndWertErmittelnVorSw2007“ zu testen, öffnen Sie bitte dieses SolidWorks Dokument.

Wenn Sie das Projekt starten und im Formular „SldWorksEigenschaftenFrm“ auf den Button „Ermitteln“ klicken, werden alle Namen und Werte der Dokumenteigenschaften ermittelt und in der ListBox dargestellt.

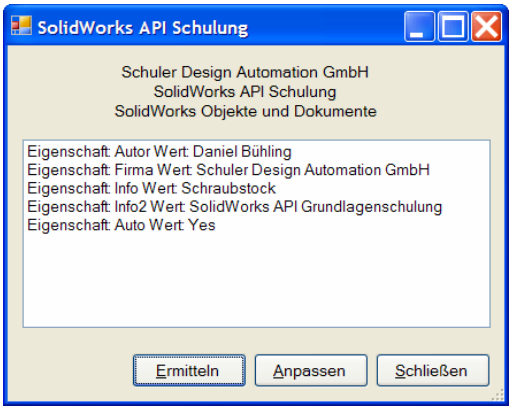


Abbildung 3.11
Die Ermittelten Dokument-eigenschaften des Dokuments Grundplatte

Sollten Sie die Methode in SolidWorks 2007 getestet haben, werden Sie sich sicherlich fragen, weshalb für diese SolidWorks Version eine andere Methode benötigt wird.
Wie im Kapitel 1.3 und 1.8 beschrieben sind alte Befehle keineswegs ungültig oder falsch. Es sollten jedoch die neueren Befehle verwendet werden. Nur dort bekommen Sie eine Unterstützung vom SolidWorks API Support.

Die Methoden für die Ermittlung der Dokumenteigenschaften in SolidWorks 2007 benötigen in zwei Variablen Instanzen weiterer SolidWorks Klassen.

Die „SldWorks.ModelDocExtension“ Klasse

```
'Erweiterungsobjekt eines
'SolidWorks Dokuments
Dim oModelDocExt As SldWorks.ModelDocExtension
```

Code 3.35
Objekt der ModelDoc Extension Klasse

Die „ModelDocExtension“ ist eine Erweiterung der „SldWorks.ModelDoc2“ Klasse. Viele neue Methoden werden vom „ModelDoc2“ in die „ModelDocExtension“ Klasse verschoben

Eine Instanz auf die „ModelDocExtension“ Klassen wird mit einer Eigenschaft der „ModelDoc2“ Klasse erstellt:

modelDocExt = ModelDoc2.Extension

SolidWorks API
ModelDoc2.
Extension

Als Rückgabe „modelDocExt“ erhalten Sie eine Instanz der „SldWorks.ModelDocExtension“ Klasse des SolidWorks Dokuments, welches durch die Instanz der „SldWorks.ModelDoc2“ Klasse repräsentiert wird.

Die Dokumenteigenschaften werden in SolidWorks 2007 auch durch eine eigene Klasse dargestellt, auch in SolidWorks bekommt die objektorientierte Programmierung einen immer höheren Stellenwert.
Die Klasse für die Eigenschaften lautet:
„SldWorks.CustomPropertyManager”.

```
'Objekt für die Eigenschaften
'eines Dokuments
Dim oCustomPropertyManager As _
      SldWorks.CustomPropertyManager
```

Code 3.36
Objekt der
CustomProperty
Manager Klasse

Die Instanz auf diese Klasse wird durch die folgende Eigenschaft der “ModelDocExtension” Klasse erzeugt.

Retval = ModelDocExtension.CustomPropertyManager (_
ConfigName)

SolidWorks API
ModelDoc
Extension.
CustomProperty
Manager

Als Parameter übergeben Sie dieser Funktion folgende Werte:

- ConfigName = Konfiguration, welche die Eigenschaften enthält oder ein Leerstring für konfigurationsunabhängige Dokumenteigenschaften.

Als Rückgabe erhalten Sie eine Instanz der „SldWorks.CustomPropertyManager“ Klasse, welches für die Dokumenteigenschaften der übergebenen Konfiguration steht.

Die Methode für SolidWorks 2007, in welcher Sie diese beiden Klassen und Eigenschaften verwenden, wird ebenfalls in der Klasse „MySldWorksCls“ erzeugt und bekommt den Namen „DokumentEigenschaftenUndWertErmittelnAbSw2007“.

Diese Methode besitzt folgenden Funktionsrahmen:

Code 3.37
Funktionsrahmen der Methode Dokument EigenschaftenUnd WertErmitteln AbSw2007

```
Friend Function DokumentEigenschaftenUndWertErmittelnAbSw2007( _
            Optional ByVal sKonfiguration As String = "") As _
            System.Collections.Generic.Dictionary( _
            Of String, String)
        Try
            'Dictionary für die ermittelten
            'Dokumenteigenschaften und dessen Wert
            Dim scEigenschaftenUndWert As _
                System.Collections.Generic.Dictionary( _
                Of String, String)
            'Stringarray für die vorhandenen
            'Dokumenteigenschaften des Dokuments
            Dim sEigenschaften() As String
            'Inhalt der Dokumenteigenschaft
            Dim sInhalt As String = Nothing
            'Erweiterungsobjekt eines
            'SolidWorks Dokuments
            Dim oModelDocExt As SldWorks.ModelDocExtension
            'Objekt für die Eigenschaften
            'eines Dokuments
            Dim oCustomPropertyManager As _
                SldWorks.CustomPropertyManager
            'SolidWorks Application Objekt prüfen
            If oSwAppCls Is Nothing Then
                'Fehler erzeugen
                Throw New Exception( _
                "Keine SolidWorks Instanz vorhanden.")
            End If
            'Erweiterungsobjekt des
            'Dokuments initialisieren
            oModelDocExt = Me.AktuellesDokument.Extension
            'Dokumenteigenschaftenmanger
            'des aktuell Dokuments initialisieren
            oCustomPropertyManager = _
                oModelDocExt.CustomPropertyManager( _
                sKonfiguration)

            Return scEigenschaftenUndWert
        Catch ex As Exception
            Debug.Assert(False)
            MsgBox("Fehler: Wo: " & _
             ex.StackTrace & " Was: " & ex.Message)
            Return Nothing
        End Try
End Function
```

In der Methode “DokumentEigenschaftenUndWertErmittelnAbSw2007“ wurden bereits alle benötigen Variablen deklariert und initialisiert. Neu in dieser Methode ist neben den Variablen für die SolidWorks Klassen „CustomPropertyManager“ und „ModelDocExtension“ die Variable „sInhalt“, in welcher später der Inhalt einer Dokumenteigenschaft zwischengespeichert wird.

Was in der neuen Methode noch fehlt, ist die Ermittlung der Namen der vorhandenen Dokumenteigenschaften und dessen Inhalt. Die Vorgehensweise ist dabei identisch wie in der ersten Methode.
Die benötigen SolidWorks API Befehle lauten:

SolidWorks API
CustomProperty Manager. GetNames

retval = CustomPropertyManager.GetNames ()

Als Rückgabe erhalten Sie ein SaveArray mit den Namen aller Dokumenteigenschaften.

SolidWorks API
CustomProperty Manager.Get2

CustomPropertyManager.Get2 (_
FieldName, valOut, reesolvedValOut)

Als Parameter übergeben Sie dieser Funktion folgende Werte:

- FieldName = Name der Dokumenteigenschaft

Als Rückgabe erhalten Sie:

- valOut = Den Inhalt der übergebenen Dokumenteigenschaften.
- reesolvedValOut = Evaluierter Inhalt der übergebenen Dokumenteigenschaften.

Der evaluierte Inhalt wird in der Methode „DokumentEigenschaftenUndWertErmittelnAbSw2007“ ignoriert. Deshalb belegen Sie diesen Parameter mit „Nothing“.
Die beiden Befehle werden in der Methode, also wie folgt aufgerufen und verwendet:

Code 3.38
Ermittlung der Eigenschaftsnamen und dessen Inhalte mit Hilfe der CustomProperty Manager Klasse

```
sEigenschaften = CType( _
    oCustomPropertyManager.GetNames, String())
'Dictionary initialisieren
scEigenschaftenUndWert = New _
    System.Collections.Generic.Dictionary( _
    Of String, String)
'Alle Eigenschaften berücksichtigen
'und dessen Wert ermitteln
For iEigenschaft As Integer = _
    0 To sEigenschaften.GetUpperBound(0)
    'Inhalt der Eigenschaft ermitteln
    oCustomPropertyManager.Get2( _
        sEigenschaften(iEigenschaft), sInhalt, Nothing)
    'Eigenschaften und dessen ermitteln
    'Wert als Eintrag in die Dictionary
    scEigenschaftenUndWert.Add( _
        sEigenschaften(iEigenschaft), _
        sInhalt)
Next
```

Wenn Sie den Aufruf der Methode in der „cmdErmitteln.Click“ Ereignisprozedur an die neue Methode …AbSw2007 anpassen, können Sie diese wieder testen.

Damit Ihnen in der Klasse „MySldWorksCls“ wieder eine allgemeine Methode zur Verfügung steht, fehlt Ihnen noch eine Methode, welche die richtige Methode für die Ermittlung der Dokumenteigenschaften abhängig von der SolidWorks Version verwendet.

Diese Methode nennen Sie „DokumentEigenschaftenUndWertErmitteln“. Der Typ und die Parameter dieser Methode sind mit den vorherigen Methoden identisch. Die SolidWorks Version wird mit Hilfe der Eigenschaft „VerisonInJahr“ der „MySldWorksCls“ Klasse ermittelt und in einer If-Anweisung ausgewertet.

Code 3.39 Versionsabhängige Ermittlung der Dokument-eigenschaften

```
Friend Function DokumentEigenschaftenUndWertErmitteln( _
        Optional ByVal sKonfiguration As String = "") As _
        System.Collections.Generic.Dictionary( _
         Of String, String)
    Try
        'Dictionary für die ermittelten
        'Dokumenteigenschaften und dessen Wert
        Dim scEigenschaftenUndWert As _
                System.Collections.Generic.Dictionary( _
                Of String, String)
        'Je nach SolidWorks Version die
        'richtige Methode ausführen
        If Me.VersionInJahr >= 2007 Then
            'Ab SolidWorks 2007
            scEigenschaftenUndWert = _
            Me.DokumentEigenschaftenUndWertErmittelnAbSw2007( _
            sKonfiguration)
        Else
            'Für ältere SolidWorks Versionen
            scEigenschaftenUndWert = _
            Me.DokumentEigenschaftenUndWertErmittelnVorSw2007( _
            sKonfiguration)
        End If
        Return scEigenschaftenUndWert
    Catch ex As Exception
        Debug.Assert(False)
        MsgBox("Fehler: Wo: " & _
             ex.StackTrace & " Was: " & ex.Message)
        Return Nothing
    End Try
End Function
```

Wenn Sie diese Methode für die Ermittlung der Dokumenteigenschaften verwenden, verwendet Ihre Anwendung immer die aktuellen SolidWorks API Befehle.

3.11 Dokumenteigenschaften manipulieren

Die Dokumenteigenschaften des Bauteils Grundplatte werden in diesem Kapitel manipuliert.
Als Ziel sollen Sie eine Eigenschaft löschen, eine neue erzeugen und einen Eigenschaftsinhalt ändern.

Diese Manipulation lässt sich nicht einfach in einer allgemeinen Methode realisieren, da diese vom jeweiligen Dokument und dessen Eigenschaften abhängig sind. Deshalb wird diese Methode in der Formularklasse „SldWorksEigenschaftenFrm“ eingefügt.
Wie im Kapitel 3.10 werden beide Möglichkeiten der Manipulation der Dokumenteigenschaften in zwei unabhängigen Methoden verwendet.

Beginnen Sie zunächst mit der Methode für eine SolidWorks Version vor 2007. Die benötigten API Befehle für diese Aufgaben sind:

retval = ModelDoc2.AddCustomInfo3 (configuration, FieldName, _
FieldType, FieldValue)

SolidWorks API
ModelDoc2.
AddCustomInfo3

Als Parameter übergeben Sie dieser Funktion folgende Werte:

- configuration = Konfiguration, welche die Eigenschaften enthält oder ein Leerstring für konfigurationsunabhängige Dokumenteigenschaften.
- FieldName = Name der Dokumenteigenschaft.
- FieldType = Typ der Dokumenteigenschaft. Vergleichbar mit dem Typ einer Variablen. Definiert werden die Eigenschaftstypen durch die Enumerationsvariable swCustomInfoType_e.
- FieldValue = Inhalt der Dokumenteigenschaft.

Als Rückgabewert erhalten Sie einen Status, welcher Ihnen den Erfolg der Methode mitteilt.

retval = ModelDoc2.DeleteCustomInfo2(configuration, FieldName)

SolidWorks API
ModelDoc2.
DeleteCustomInfo2

Als Parameter übergeben Sie dieser Funktion folgende Werte:

- configuration = Konfiguration, welche die Eigenschaften enthält oder ein Leerstring für konfigurationsunabhängige Dokumenteigenschaften.
- FieldName = Name der Dokumenteigenschaft

Als Rückgabewert erhalten Sie einen Status, welcher Ihnen den Erfolg der Methode mitteilt.

value = ModelDoc2.CustomInfo2 (configuration, fieldName)
ModelDoc2.CustomInfo2 (configuration, fieldname) = value

SolidWorks API
ModelDoc2.
CustomInfo2

Als Parameter übergeben Sie dieser Funktion folgende Werte:

- configuration = Konfiguration, welche die Eigenschaften enthält oder ein Leerstring für konfigurationsunabhängige Dokumenteigenschaften.
- FieldName = Name der Dokumenteigenschaft

Diese Eigenschaft fragt den Inhalt einer Dokumenteigenschaft ab, oder legt diesen fest.

Die drei SolidWorks API Befehle werden Sie in einer neuen Prozedur „AnpassenVorSw2007“ verwenden, welche in der Ereignisprozedur „cmdAnpassen.Click“ aufgerufen wird, um die Dokumenteigenschaften zu manipulieren. Die benötigten Instanzen erstellen Sie mit Ihrer Klasse „MySldWorksCls“.

Code 3.40
Ereignisprozedur
cmdAnpassen.Click

```
Private Sub cmdAnpassen_Click( _
        ByVal sender As System.Object, _
        ByVal e As System.EventArgs) _
        Handles cmdAnpassen.Click
    Try
        'Eigene SolidWorks Klasse deklarieren
        Dim oMySldWorks As MySldWorksCls
        'Eigene SolidWorks Klasse initialisieren
        oMySldWorks = New MySldWorksCls
        'SolidWorks initialisieren
        oMySldWorks.SolidWorksInstanz()

        'Eigene SolidWorks Klasse leeren
        oMySldWorks = Nothing
    Catch ex As Exception
        Debug.Assert(False)
        MsgBox("Fehler: Wo: " & _
         ex.StackTrace & " Was: " & ex.Message)
    End Try
End Sub
```

Als Parameter der Prozedur „AnpassenVorSw2007“ wird eine Instanz Ihrer SolidWorks Klasse „MySldWorksCls“ übergeben. Die API Befehle für die Manipulation der Dokumenteigenschaften werden so verwendet, dass Sie nur für das Bauteil Grundplatte gültig sind, da die Namen der vorhandenen Dokumenteigenschaften nicht ermittelt oder überprüft werden.

```
Private Sub AnpassenVorSw2007( _
        ByVal oMySldWorks As MySldWorksCls)
    Try
        'Dokumenteigenschaft löschen
        oMySldWorks.AktuellesDokument.DeleteCustomInfo2( _
            "", "Auto")
        'Dokumenteigenschaft ändern
        oMySldWorks.AktuellesDokument.CustomInfo2( _
            "", "Autor") = "Schulungsteilnehmer"
        'Neue Dokumenteigenschaft hinzufügen
        oMySldWorks.AktuellesDokument.AddCustomInfo3( _
            "", "Datum", _
            SwConst.swCustomInfoType_e.swCustomInfoDate, _
            Now.ToString)
    Catch ex As Exception
        Debug.Assert(False)
        MsgBox("Fehler: Wo: " & _
         ex.StackTrace & " Was: " & ex.Message)
    End Try
End Sub
```

Code 3.41
Prozedur für das Manipulieren der Dokument-eigenschaften

Die Prozedur „AnpassenVorSw2007“ wird im Klick Ereignis des Buttons „cmdAnpassen“ aufgerufen und das initialisierte „oMySldWorks“ Objekt übergeben.

```
'Dokumenteigenschaft manipulieren
Me.AnpassenVorSw2007(oMySldWorks)
```

Code 3.42
Aufruf der Manipulations prozedur

Wenn Sie die Anwendung testen und die Dokumenteigenschaften manipulieren, werden Sie den Unterschied dieser feststellen, wenn Sie die Dokumenteigenschaften vor und nach der Manipulation ermitteln.

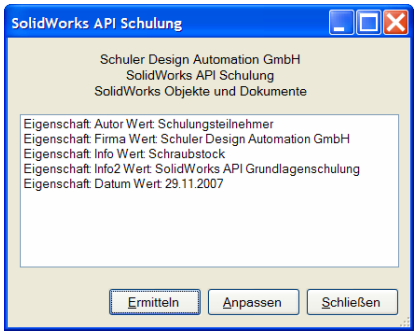


Abbildung 3.12
Geänderte Dokument-eigenschaften im Formular EigenschaftenFrm

Die SolidWorks API Befehle für die Manipulation der Dokumenteigenschaften für SolidWorks 2007 befinden sich alle in der Klasse „SldWorks.CustomPropertyManager“ und lauten wie folgt:

Retval = CustomPropertyManager.Add2 (_
FieldName, FieldType, FieldValue)

SolidWorks API
CustomProperty Manager.Add2

Als Parameter übergeben Sie dieser Funktion folgende Werte:

- FieldName = Name der Dokumenteigenschaft.
- FieldType = Typ der Dokumenteigenschaft. Vergleichbar mit dem Typ einer Variablen. Definiert werden die Eigenschaftstypen durch die Enumerationsvariable swCustomInfoType_e.
- FieldValue = Inhalt der Dokumenteigenschaft.

Als Rückgabewert erhalten Sie einen Status, welcher Ihnen den Erfolg der Methode mitteilt.

retval = CustomPropertyManager.Delete (FieldName)

SolidWorks API
CustomProperty Manager.Delete

Als Parameter übergeben Sie dieser Funktion folgende Werte:

- FieldName = Name der Dokumenteigenschaft

Als Rückgabewert erhalten Sie einen Status, welcher Ihnen den Erfolg der Methode mitteilt.

retval = CustomPropertyManager.Set (FieldName, FieldValue)

SolidWorks API
CustomProperty Manager.Set

Als Parameter übergeben Sie dieser Funktion folgende Werte:

- FieldName = Name der Dokumenteigenschaft.
- FieldValue = Inhalt der Dokumenteigenschaft.

Als Rückgabewert erhalten Sie einen Status, welcher Ihnen den Erfolg der Methode mitteilt.

Diese Methoden verwenden Sie in der Prozedur „AnpassenAbSw2007“. Vergessen Sie dabei nicht die benötigten Instanzen der „ModeDocExtension“ und „CustomPropertyManager“ Klasse.

Code 3.43 Prozedur für das Manipulieren der Dokument-eigenschaften in SolidWorks 2007

```
    Private Sub AnpassenAbSw2007( _
            ByVal oMySldWorks As MySldWorksCls)
        Try
            'Erweiterungsobjekt eines
            'SolidWorks Dokuments
            Dim oModelDocExt As SldWorks.ModelDocExtension
            'Objekt für die Eigenschaften
            'eines Dokuments
            Dim oCustomPropertyManager As _
                SldWorks.CustomPropertyManager
            'Erweiterungsobjekt des
            'Dokuments initialisieren
            oModelDocExt = oMySldWorks.AktuellesDokument.Extension
            'Dokumenteigenschaftenmanger
            'des aktuell Dokuments initialisieren
            oCustomPropertyManager = _
                oModelDocExt.CustomPropertyManager( _
                "")
            'Dokumenteigenschaft löschen
            oCustomPropertyManager.Delete("Auto")
            'Dokumenteigenschaft ändern
            oCustomPropertyManager.Set( _
                "Autor", "Schulungsteilnehmer")
            'Neue Dokumenteigenschaft hinzufügen
            oCustomPropertyManager.Add2( _
                "Datum", _
                SwConst.swCustomInfoType_e.swCustomInfoDate, _
                Now.ToString)
        Catch ex As Exception
            Debug.Assert(False)
            MsgBox("Fehler: Wo: " & _
             ex.StackTrace & " Was: " & ex.Message)
        End Try
    End Sub
```

Die beiden Methoden …Vor- und …AbSw2007 werden in der Ereignisprozedur „cmdAnpassen.Click“ versionsabhängig verwendet. Damit Sie dies erreichen, verwenden Sie die gleiche Abfrage wie beim Ermitteln der Dokumenteigenschaften.

Code 3.44 Versions-abhängiger Aufruf der Manipulations prozedur

```
'Dokumenteigenschaft manipulieren
'Je nach SolidWorks Version die richtige Methode ausführen
If oMySldWorks.VersionInJahr >= 2007 Then
    'Ab SolidWorks 2007
    Me.AnpassenAbSw2007(oMySldWorks)
Else
    'Für ältere SolidWorks Versionen
    Me.AnpassenVorSw2007(oMySldWorks)
End If
```

4 Nützliche Tricks beim Arbeiten mit der SolidWorks API

4.1 Aufnahmen des Makrorekorders

Mit Absicht wird in diesem Skript erst im letzten Kapitel auf den Makrorekorder eingegangen.
Sicherlich hatten Sie bereits Kontakt mit Makros und sind evtl. auch durch diese auf die SolidWorks API Programmierung aufmerksam geworden.
Wenn Sie das Skript aufmerksam bearbeitet haben, sind Ihnen hoffentlich Unterschiede zu den bekannten Programmzeilen im Makrorekorder aufgefallen.

Die wichtigsten Unterschiede:

- Die SolidWorks VBA (Visual Basic for Applicationen) ist Stand Visual Basic 6 und somit nicht mehr aktuell (Was nicht bedeutet, das sich in der VBA nicht fast alles Realisieren lässt).
- Anwendungen der SolidWorks VBA können nur als Makro gespeichert werden.
- Die Aufnahmen des Makrorekorders beschränken sich meist auf die Ebene der SolidWorks Application Objekte also:
    - SldWorks.SldWorks
    - SldWorks.ModelDoc2
    - SldWorks.PartDoc
    - SldWorks.AssemblyDoc
    - SldWorks.DrawingDoc
    - SldWorks.ModelDocExtension

  Nicht selten hat der Makrorekorder aber auch Befehle anderer Objekte aufgezeichnet.
- Der aufgezeichnete Programmcode ist unsauber deklariert. Das SolidWorks Application Objekt wir zum Beispiel immer als Object deklariert.
- Der Code des Makrorekorders enthält (zwar sehr selten, aber es kommt vor) nicht vorhandene und somit falsche Methoden
- Die Konstanten der SwConst Library werden nicht benutzt.

Der Makrorekorder ist jedoch ein sehr sinnvolles Instrument, um den einen oder anderen API Befehl in Erfahrung zu bringen.

Als Beispiel werden Sie das Öffnen der Schraubstock Baugruppe mit dem Makrorekorder aufzeichnen.

Code 4.0
Makrorekorder
Aufzeichnung

```
Dim swApp As Object
Dim Part As Object
Dim SelMgr As Object
Dim boolstatus As Boolean
Dim longstatus As Long, longwarnings As Long
Dim Feature As Object
Sub main()

Set swApp = Application.SldWorks

Set Part = swApp.OpenDoc6( _
"C:\Baugruppen\Schraubstock\Schraubstock.SLDASM", 2, 0, _
"", longstatus, longwarnings)
Set Part = swApp.ActivateDoc2( _
"Schraubstock.SLDASM", False, longstatus)
swApp.ActiveDoc.ActiveView.FrameLeft = 0
swApp.ActiveDoc.ActiveView.FrameTop = 0
swApp.ActiveDoc.ActiveView.FrameState = 1
swApp.ActiveDoc.ActiveView.FrameState = 1
End Sub
```

Ohne die Kenntnisse dieses Skripts, besonders dem Kapitel 3.2, hätten Sie den Befehl zum Öffnen eines Dokuments ermittelt.

Sie sehen auch wie unsauber dieser Code aufgezeichnet wurde und sich deshalb sehr schlecht lesen und bearbeiten lässt.

Sie können den Schraubstock in SolidWorks wieder schließen und das Makro ausführen. Bei aller Kritik an dem Programmstil des Makrorekorders, funktionsfähig ist er.

4.2 Programmieren in der SolidWorks VBA

Bei der SolidWorks VBA handelt es sich um eine fast vollwertige Endwicklungsumgebung. Der einzige Haken ist jedoch, dass ein Makro nicht in eine exe oder dll Datei kompiliert werden kann. Das Erzeugen einer SolidWorks Zusatzanwendung ist also mit der SolidWorks VBA nicht möglich.

Einfügen lassen sich in ein VBA Projekt alle bekannten und am meisten benötigten Projektelemente wie Module, Klassen und Formulare.
Fehlende Komponenten, wie die „Microsoft Windows Common Controls“ können ebenso in ein VBA Projekt eingefügt und somit in einem Formular benutz werden.
Dies erledigen Sie im Menü „Einfügen“.

Auch Verweise (zum Beispiel auf Microsoft Excel) lassen sich in das Projekt einfügen. Klicken Sie hierfür auf den Menüeintrag „Extras > Verweise“.

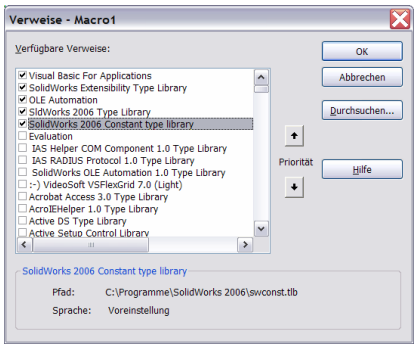


Abbildung 4.0
Verweise des SolidWorks VBA Projekts

Die benötigten SolidWorks Verweise sind bereits Bestandteil des VBA Projekts.

Deshalb spricht nichts dagegen diese im Programmcode wirkungsvoll zu verwenden.
Verbessern Sie den aufgezeichneten Programmcode für das Öffnen des Schraubstocks, so wie Sie es in diesem Skript gelernt haben. Auch die Konstanten der SwConst Library werden benutz.

Code 4.1
Verbesserter Code des Makrorekorders

```
Dim swApp As SldWorks.SldWorks
Dim Part As SldWorks.ModelDoc2
Dim boolstatus As Boolean
Dim longstatus As Long
Dim ewarnings As SwConst.swFileLoadWarning_e
Dim eerrors As SwConst.swFileLoadError_e

Sub main()

    Set swApp = Application.SldWorks

    Set Part = swApp.OpenDoc6( _
        "C:\Baugruppen\Schraubstock\Schraubstock.SLDASM", _
        swDocumentTypes_e.swDocASSEMBLY, _
        swOpenDocOptions_e.swOpenDocOptions_Silent, _
        "", eerrors, ewarnings)

End Sub
```

Ist dieser Code nicht schlanker und besser lesbar als der vom Makrorekorder aufgezeichnete Code?
Auf eine Auswertung der Rückgabewerte habe ich an dieser Stelle verzichtet. Dieses können Sie im Kapitel 3.2. nachschlagen.

Sie können dieses Makro ein weiteres Mal testen. Es funktioniert wie gewünscht.

4.3 SolidWorks Optionen

SolidWorks bietet einem User bekanntlich eine Fülle von Optionen, welche zum Großteil in SolidWorks Menü unter „Extras > Optionen“ eingestellt werden.

Diese Optionen bewirken in manchen Fällen eine andere Vorgehensweise in SolidWorks oder enthalten wichtige Informationen.

Diese Optionen lassen sich per API ermitteln und ändern.
Es gibt zwei Hauptoptionsgruppen welche sich wiederum in fünf grundsätzliche Optionsgruppen aufteilen lassen.
Die beiden Hauptgruppen sind die SolidWorks- (System) und Dokumentoptionen. Diese Optionen unterteilen sich jeweils in die verschieden Variablentypen:

- Toggle (Boolean)
- Integer
- Double
- String
- StringList

Diese Optionen lassen sich je nach SolidWorks- oder Dokumentoption in der „SldWorks.SldWorks“ oder „SldWorks.ModelDoc2“ Klasse abfragen und ändern.
Die dabei benötigten Methoden sind in beiden Klassen identisch.
Vorgestellt wird zuerst die Methode für die Toogle (Boolean) Einstellungen der „SldWorks.SldWorks“ Klasse.
Alle anderen Optionen lassen sich auf sehr ähnliche Weise abfragen und ermitteln, weshalb auf eine weitere Erklärung dieser Methoden verzichtet wird. Für eine andere Optionsgruppe passen Sie einfach das Schlüsselwort „Toggle“ durch die gewünschte Optionsgruppe an.

Ermitteln von Optionen

SolidWorks API
SldWorks / ModelDoc2 GetUserPreference Toggle

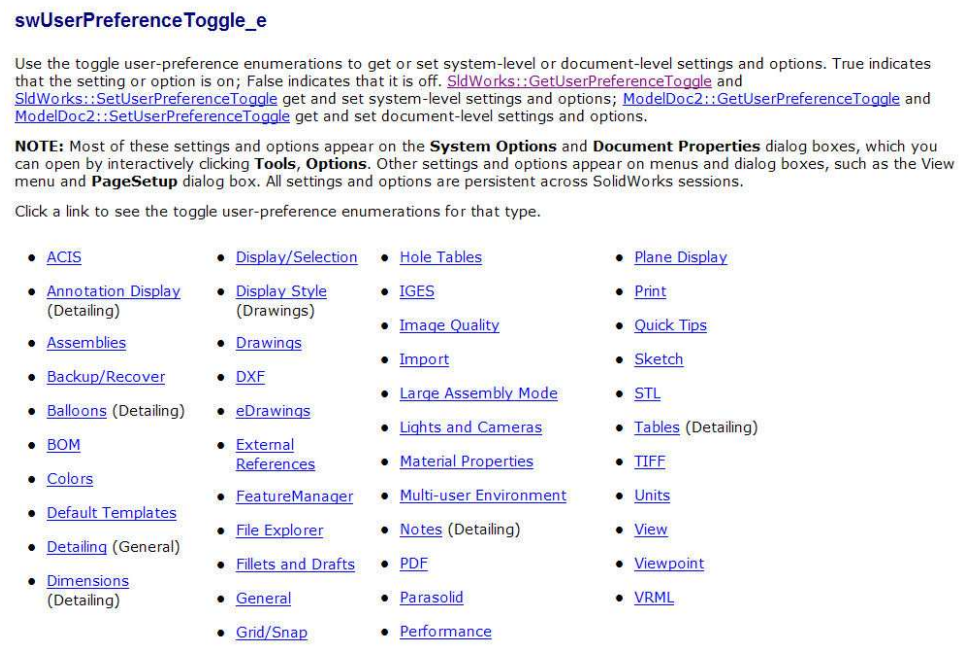
```
retval = SldWorks.GetUserPreferenceToggle ( _
                                    userPreferenceToggle)
```

Als Parameter übergeben Sie:

- userPreferenceToggle = die Emurationsvariable „swUserPreferencesToggle_e“. Diese Emurationsvariable sollten Sie in der API Online Hilfe nachschlagen, um die richtige Option zu ermitteln. Es sind dort alle verfügbaren Optionen übersichtlich in Untergruppen aufgelistet.

Als Rückgabe erhalten Sie eine Variable des Variablentyps, welche von der Optionsgruppe repräsentiert wird. Bei Toggle ist dies Boolean.

Betrachten Sie sich das Thema „swUserPreferencesToggle_e“ in der API Online Hilfe etwas genauer.



swUserPreferenceToggle_e

Use the toggle user-preference enumerations to get or set system-level or document-level settings and options. True indicates that the setting or option is on; False indicates that it is off. SldWorks::GetUserPreferenceToggle and SldWorks::SetUserPreferenceToggle get and set system-level settings and options; ModelDoc2::GetUserPreferenceToggle and ModelDoc2::SetUserPreferenceToggle get and set document-level settings and options.

NOTE: Most of these settings and options appear on the **System Options** and **Document Properties** dialog boxes, which you can open by interactively clicking **Tools**, **Options**. Other settings and options appear on menus and dialog boxes, such as the View menu and **PageSetup** dialog box. All settings and options are persistent across SolidWorks sessions.

Click a link to see the toggle user-preference enumerations for that type.

- ACIS
- Annotation Display (Detailing)
- Assemblies
- Backup/Recover
- Balloons (Detailing)
- BOM
- Colors
- Default Templates
- Detailing (General)
- Dimensions (Detailing)
- Display/Selection
- Display Style (Drawings)
- Drawings
- DXF
- eDrawings
- External References
- FeatureManager
- File Explorer
- Fillets and Drafts
- General
- Grid/Snap
- Hole Tables
- IGES
- Image Quality
- Import
- Large Assembly Mode
- Lights and Cameras
- Material Properties
- Multi-user Environment
- Notes (Detailing)
- PDF
- Parasolid
- Performance
- Plane Display
- Print
- Quick Tips
- Sketch
- STL
- Tables (Detailing)
- TIFF
- Units
- View
- Viewpoint
- VRML

ACIS

	Type of Setting		
Enumeration	**System-Level**	**Document-Level**	**Comments**
swTranslateNameAttribFromKernelBody	Yes	No	Not used.

Abbildung 4.1
Hilfethema swUserPreferences Toggle

In diesem Hilfethema werden Ihnen die Usereinstellungen in verscheidende Themenbereiche, zum Beispiel Drawing, DXF …, angezeigt, damit Sie diese schneller finden.
Die eigentlichen Emurationskonstanten werden in einer Tabelle angezeigt. Wichtig ist hierbei den Hinweis zu berücksichtigen, ob es sich bei einer Option um eine System-Level (SolidWorks-Option) oder Document-Level (Dokumentoption) handelt.

Das Ändern von Optionen erfolgt auf eine ähnliche Weise.

SolidWorks API
SldWorks / ModelDoc2 SetUserPreference Toggle

SldWorks.SetUserPreferenceToggle (_
userPreferenceValue, onFlag)

Als Parameter übergeben Sie dieser Methode:

- userPreferenceValue = Emurationsvariable „swUserPreferencesToggle _e“, welche die Option repräsentiert
- onFlag = Die neue gewünschte Optionseinstellung. Bei einer Option vom Typ Toggle einen Booleanwert, also True oder False.

Bedenken Sie aber, dass Sie die Optionen eines Users immer sichern sollten wenn Sie diese ändern.
Auch ist es wichtig, bei einem Programmabsturz dafür zu sorgen, dass Sie die Optionen wieder so einstellen, wie Sie diese vorgefunden haben. Sie werden andernfalls eine Menge Probleme mit einem Solidworks User bekommen, wenn Sie dessen Einstellungen einfach verändern und nicht zurücksetzen.

Im folgenden Kapitel werden Sie eine Useroption ändern, welche das Standardverzeichnis für SolidWorks Makro festlegt.

4.4 Option „Standard Makroverzeichnis“ ändern

Damit Sie ein Beispiel über die SolidWorks Optionen erstellen, ändern Sie in diesem Kapitel die SolidWorks-Option, welche das Standardverzeichnis für SolidWorks Makros festlegt.

Die Option speichert ein Verzeichnis, also kann es sich beim Optionstyp nur um einen String handeln. Im Hilfethema „swUserPreferenceStringValue_e“ finden Sie den Themenbereich „File Locations“. In diesem Abschnitt befindet sich auch die benötigte Emurationskonstante „swFileLocationsMacros“.
Die Konstante besitzt somit folgenden vollständigen Namen:

```
SwConst.swUserPreferenceStringValue_e.swFileLocationsMacros
```

Code 4.2
Emurations-konstante des standard Markoverzeichnis

Erstellen Sie eine Prozedur „MakroVerzeichnis“ in der Formularklasse “SchulungFrm”, in der die vorhandene Einstellung in einer MessageBox angezeigt und darauf geändert wird. Als neues Makroverzeichnis verwenden Sie einfach Ihr Desktopverzeichnis.

Vergessen Sie bitte nicht für das Testen der Prozedur, das Formular „SchulungFrm“ als Startformular einzustellen.

Als Parameter der Prozedur „MakroVerzeichnis“ wird eine Instanz Ihrer SolidWorks Klasse „MySldWorksCls“ übergeben. Die Prozedur als solches rufen Sie in der Ereignisprozedur „cmdGet.Click“ auf, wenn eine Instanz zu SolidWorks erstellt wurde.

```
'SolidWorks Instanz prüfen
If oMySldWorks.MySldWorks IsNot Nothing Then
     'Makroverzeichnis ermitteln und ändern
     Me.MakroVerzeichnis(oMySldWorks)
     'Version in einer MessageBox anzeigen
     oMySldWorks.VersionAnzeigen()
End If
```

Code 4.3
Auruf der MakroVerzeichnis Prozedur nach dem initialisieren der SolidWorks Application

Die API Methode für das Ermitteln und Ändern einer SolidWorks-Option vom Typ String lautet:

retval = SldWorks. GetUserPreferenceStringValue (_
userPreference)

SolidWorks API
SldWorks.
GetUserPreference
StringValue

Als Parameter übergeben Sie die Emurationsvariable „swUserPreferenceStringValue_e“.

Als Rückgabe erhalten Sie einen String mit dem Inhalt der Option.

retval =SldWorks. SetUserPreferenceStringValue (_
userPreference, userPrefVal)

SolidWorks API
SldWorks.
SetUserPreference
StringValue

Als Parameter übergeben Sie dieser Methode:

- userPreference = Emurationsvariable „swUserPreferenceStringValue_e“, welche die Option repräsentiert
- userPrefVal = Die neue gewünschte Optionseinstellung.

Als Rückgabewert erhalten Sie einen Status, ob die Aktion erfolgreich war.

```
Private Sub MakroVerzeichnis(ByVal oMySldWorks As MySldWorksCls)
    Try
        'Aktuelles Makroverzeichnis
        MsgBox("Aktuelles Makroverzeichnis: " & _
        vbNewLine & _
        oMySldWorks.MySldWorks.GetUserPreferenceStringValue( _
        SwConst.swUserPreferenceStringValue_e.swFileLocationsMacros))
        'Makroverzeichnis in den eigenen Desktop ändern
        oMySldWorks.MySldWorks.SetUserPreferenceStringValue( _
        SwConst.swUserPreferenceStringValue_e.swFileLocationsMacros, _
        System.Environment.GetFolderPath( _
        Environment.SpecialFolder.DesktopDirectory))
    Catch ex As Exception
        Debug.Assert(False)
        MsgBox("Fehler: Wo: " & _
               ex.StackTrace & " Was: " & ex.Message)
    End Try
End Sub
```

Code 4.4
Die Prozedur
MakroVerzeichnis

Wenn Sie jetzt die SolidWorks Application zweimal mit Hilfe des „GetObject“ Buttons initialisieren, bekommen Sie zunächst Ihr Ausgangsverzeichnis und beim zweiten Mal Ihr Desktopverzeichnis als SolidWorks „Standard Makroverzeichnis“ angezeigt.

Abbildung 4.2
Ausgangs- und
verändertes
Makroverzeichnis

4.5 Weitere Beispiele und Informationen über die SolidWorks API

Bevor ich Sie gleich auf viele interessante und informative Internetseiten verweise, möchte ich Sie nochmals auf die „SolidWorks und Zusatzanwendungen-API-Hilfethemen“, also die API Online Hilfe und den Makrorekorder aufmerksam machen. Suchen Sie einen bestimmten Befehl? Vielleicht nimmt Ihn der Makrorekorder auf! Alle Informationen zu diesem Befehl erhalten Sie in der API Online Hilfe!

Visual Basic 2005:

Tipp, Tricks und OpenSource Projekte:

- http://www.vbarchiv.net
- http://www.activevb.de
- http://www.vb-power.net

Kostenlose Openbocks:

- http://www.galileocomputing.de/openbook/visual_basic/

SolidWorks API:

Offizielle SolidWorks Seiten auf Deutsch und Englisch:

- http://www.solidworks.com/pages/services/APIDownloads.html?pid=121
- http://www.solidworks.de/pages/services/APIDownloads.html?pid=39
- http://www.solidworks.de/pages/services/CodeExamples.html?PID=39

Inoffizielle Hilfeseite von Stefan Berlitz:

- http://solidworks.cad.de

Wiki Plattform über CAD:

- http://www.cad42.de/index.php/SolidWorks_API

Sonstiges:

- http://mysite.verizon.net/mjlombard/
- http://www.lennyworks.com/solidworks/default.asp?ID=20

Foren:

Alles Rund um das Thema CAD, SolidWorks und Visual Basic

- http://www.cad.de

Visual Basic Forum mit Info Board

- http://www.vbarchiv.net

5 Anhang

A Der Code des SldWorksDokument Projekts

Klasse MySldWorksCls

```
Option Strict On
Option Compare Text

Public Class MySldWorksCls

    Dim oSwAppCls As SldWorks.SldWorks

    Friend ReadOnly Property MySldWorks() _
                    As SldWorks.SldWorks
        Get
            Return oSwAppCls
        End Get
    End Property

    Friend Function SolidWorksInstanzCreate() _
                        As SldWorks.SldWorks
        Try
            'SolidWorks Objekt initialisieren
            oSwAppCls = CType(Microsoft.VisualBasic.CreateObject( _
                  ProgId:="SldWorks.Application"), SldWorks.SldWorks)
            Return oSwAppCls
        Catch ex As Exception
            Debug.Assert(False)
            MsgBox("Fehler: Wo: " & _
                ex.StackTrace & " Was: " & ex.Message)
            Return Nothing
        End Try
    End Function

    Friend Sub VersionAnzeigen()
        Try
            'Rückgabewert der SolidWorks
            'MessageBox
            Dim eSwMsgBox As SwConst.swMessageBoxResult_e
            'SolidWorks Fenster anzeigen
            oSwAppCls.Visible = True
            'SolidWorks Version
            'in einer MsgBox anzeigen
            eSwMsgBox = CType(oSwAppCls.SendMsgToUser2( _
                "SolidWorks Version: " & _
                Me.VersionInJahr, _
                SwConst.swMessageBoxIcon_e.swMbInformation, _
                SwConst.swMessageBoxBtn_e.swMbYesNo), _
                SwConst.swMessageBoxResult_e)
            Select Case eSwMsgBox
                Case SwConst.swMessageBoxResult_e.swMbHitYes
                    'Hinweis in einer MsgBox
                    MsgBox("Der User hat auf Ja geklickt")
                Case SwConst.swMessageBoxResult_e.swMbHitNo
                    'Hinweis in einer MsgBox
                    MsgBox("Der User hat auf Nein geklickt")
                Case Else
                    Debug.Assert(False)
            End Select

        Catch ex As Exception
            Debug.Assert(False)
            MsgBox("Fehler: Wo: " & _
                ex.StackTrace & " Was: " & ex.Message)
        End Try
    End Sub
```

```
Friend Function SolidWorksInstanzGet() _
                 As SldWorks.SldWorks
    Try
        oSwAppCls = CType(Microsoft.VisualBasic.GetObject( _
            Class:="SldWorks.Application"), SldWorks.SldWorks)
        Return oSwAppCls
    Catch ex As Exception
        Debug.Assert(False)
        MsgBox("Fehler: Wo: " & _
            ex.StackTrace & " Was: " & ex.Message)
        Return Nothing
    End Try
End Function

''' <summary>
''' Initialisiert das SolidWorks Objekt durch die Methode GetObject
''' und fragt in einer MsgBox nach wenn das Objekt nicht greifbar ist.
''' </summary>
''' <param name="bUserAbbruch">
''' Optionaler Rückgabewert, welcher mitteilt ob der User
''' die Aktion abgerochen hat oder ein Fehler aufgetreten ist.
''' </param>
''' <returns>Instanz der atkuellen SolidWorks Sitzung</returns>
''' <remarks></remarks>
Public Function SolidWorksInstanz( _
        Optional ByRef bUserAbbruch As Boolean = False) _
        As SldWorks.SldWorks
    Try
        'Variable für die MsgBox-Rückgabe
        Dim lDialogStatus As Microsoft.VisualBasic.MsgBoxResult
        Try
            'SolidWorks Objekt belegen
            oSwAppCls = CType(GetObject(, _
                "SldWorks.Application"), _
                SldWorks.SldWorks)
        Catch ex As Exception
        End Try
        'Wenn das SolidWorks Objekt nicht belegt ist...
        If oSwAppCls Is Nothing Then
         '... in einer MsgBox nachfragen
         lDialogStatus = MsgBox( _
         "Fehler beim Aufbau einer Schnittstelle zu SolidWorks." & _
         vbNewLine & _
         "Bitte öffnen Sie SolidWorks und klicken Sie auf ""Ok""." & _
         vbNewLine & _
         "Sollte SolidWorks bereits geöffnet sein, " & _
         "schließen Sie SolidWorks, " & _
         vbNewLine & _
         "öffnen Sie es erneut und klicken Sie dann auf ""Ok""." & _
         vbNewLine & _
         "Wenn Sie denn Startvorgang abbrechen " & _
         "möchten klicken Sie auf ""Abbrechen""", _
         MsgBoxStyle.Information Or MsgBoxStyle.OkCancel, _
         "Schuler Design Automation GmbH")
         'Rückgabewert der MsgBox auswerten
         Select Case lDialogStatus
             Case MsgBoxResult.Ok
                 'User hat SolidWorks geöffnet
                 'Funktion einfach nochmal ausführen
                 Return SolidWorksInstanz(bUserAbbruch)
             Case MsgBoxResult.Cancel
                 'User möchte die Anwednung abbrechen
                 bUserAbbruch = True
                    Return Nothing
         End Select
        End If
        Return oSwAppCls
    Catch ex As Exception
        Debug.Assert(False)
        MsgBox("Fehler: Wo: " & _
            ex.StackTrace & " Was: " & ex.Message)
        Return Nothing
    End Try
End Function
```

```
    Friend ReadOnly Property VersionInJahr() As Integer
        Get
            Dim iHauptversion As Integer
            Dim sVersionen() As String
            'SolidWorks Instanz prüfen
            If oSwAppCls Is Nothing Then
                '-1 als ungültige Rückgabe verwenden
                Return -1
            Else
                'Version ermitteln und an den
                'Trennpunkten teilen
                sVersionen = _
                    oSwAppCls.RevisionNumber.Split( _
                    CChar("."))
                'Nur die Hauptversion weiter verwenden
                iHauptversion = CInt(sVersionen(0))
                'Version in Jahreszahl umrechnen
                iHauptversion = iHauptversion + 1992
                'Die Hauptversion in Jahren
                'als Rückgabe verwenden
                Return iHauptversion
            End If
        End Get
    End Property

    ''' <summary>
    ''' Öffnet das übergebene SolidWorks Dokument in
    ''' SolidWorks und gibt eine Instanz auf das
    ''' SldWorks.ModelDoc2 Objekt zurück
    ''' </summary>
    ''' <param name="sDateipfad">
    ''' Dateipfad des SolidWorks Dokuments,
    ''' welches geöffnet werden soll</param>
    ''' <param name="eOpenDocOptions">
    ''' Optionaler Übergabeparameter,
    ''' welcher die Öffnenoption für SolidWorks enthält</param>
    ''' <param name="eOpenWarning">
    ''' Optionaler Rückgabeparameter,
    ''' welchen den Warningstatus von SolidWorks enthält</param>
    ''' <returns>Instanz eines ModelDoc2 Objekts
    ''' des geöffneten Dokuments. Bei einem Fehler Nothing</returns>
    ''' <remarks></remarks>
    Friend Function DokumentOeffnen( _
       ByVal sDateipfad As String, _
       Optional ByVal eOpenDocOptions As SwConst.swOpenDocOptions_e = _
       SwConst.swOpenDocOptions_e.swOpenDocOptions_Silent, _
       Optional ByRef eOpenWarning As SwConst.swFileLoadWarning_e = 0) As
       SldWorks.ModelDoc2
        Try
            'Benötigte Variablen
            Dim oSwModel As SldWorks.ModelDoc2
            Dim eError As SwConst.swFileLoadError_e
            'SolidWorks Dokumenttyp
            Dim eDocTyp As SwConst.swDocumentTypes_e
            Dim sExtension As String
            'Existenz der übergeben Datei prüfen
            If IO.File.Exists(sDateipfad) = False Then
                'Fehler erzeugen
                Throw New Exception( _
                    "Die Datei: " & sDateipfad & _
                    " ist nicht vorhanden.")
            End If
            'SolidWorks Application Objekt prüfen
            If oSwAppCls Is Nothing Then
                'Fehler erzeugen
                Throw New Exception( _
                    "Kein SolidWorks Application " & _
                    "Objekt vorhanden")
            End If
```

```
            'Dokumenttyp ermitteln
            sExtension = IO.Path.GetExtension( _
                                    sDateipfad)
            Select Case sExtension
                Case ".sldprt"
                    eDocTyp = _
                    SwConst.swDocumentTypes_e.swDocPART
                Case ".sldasm"
                    eDocTyp = _
                    SwConst.swDocumentTypes_e.swDocASSEMBLY
                Case ".slddrw"
                    eDocTyp = _
                    SwConst.swDocumentTypes_e.swDocDRAWING
                Case Else
                    'Fehler erzeugen
                    Throw New Exception( _
                    "Ungültiger Dokumenttyp")
            End Select
            'Überprüfungen erfolgreich
            'Dokument in SolidWorks öffnen
            oSwModel = oSwAppCls.OpenDoc6( _
                sDateipfad, _
                eDocTyp, eOpenDocOptions, _
                "", _
                CType(eError, SwConst.swFileLoadError_e), _
                CType(eOpenWarning, SwConst.swFileLoadWarning_e))
            'Rückgabewerte auswerten
            If oSwModel Is Nothing Then
                'Fehler erzeugen
                Throw New Exception( _
                "Das Dokument: " & _
                sDateipfad & _
                " konnte nicht in SolidWorks geöffnet werden. " & _
                "Fehlerantwort: " & eError.ToString)
            End If
            'Es hat alles geklappt
            Return oSwModel
        Catch ex As Exception
            Debug.Assert(False)
            MsgBox("Fehler: Wo: " & _
                ex.StackTrace & " Was: " & ex.Message)
            Throw New Exception( _
                "Fehler beim Öffnen des Dokuments: " & _
                sDateipfad & ".")
            Return Nothing
        End Try
    End Function

    Friend ReadOnly Property AktuellesDokument() As SldWorks.ModelDoc2
        Get
            'SolidWorks Instanz prüfen
            If oSwAppCls Is Nothing Then
                'Nothing als ungültige Rückgabe verwenden
                Return Nothing
            Else
                'Aktuelles Dokument zurückgeben
                Return CType( _
                    oSwAppCls.ActiveDoc, _
                    SldWorks.ModelDoc2)
            End If
        End Get
    End Property
```

```
    Friend Sub DokumentSchliessen()
        Try
            'SolidWorks Instanz prüfen
            If oSwAppCls Is Nothing Then
                'Fehler
                Throw New Exception( _
                "Keine SolidWorks Instanz vorhanden.")
            Else
                'Aktuell in SolidWorks
                'aktives Dokument schließen
                oSwAppCls.CloseDoc( _
                    Me.AktuellesDokument.GetTitle)
                'Für das Testen der verschiedenen
                'SolidWorks API Methoden_
                'UserControl Einstellung festlegen
                'oSwAppCls.UserControl = True
                'Dokument mit ModelDoc2.Close schließen
                'Me.AktuellesDokument.Close()
                'Dokument mit ModelDoc2.Quit schließen
                'Me.AktuellesDokument.Quit()
                'Dokument mit SldWorks.CloseDoc schließen
                'oSwAppCls.CloseDoc(Me.AktuellesDokument.GetTitle)
                'Dokument mit SldWorks.QuitDoc schließen
                'oSwAppCls.QuitDoc(Me.AktuellesDokument.GetTitle)
            End If
        Catch ex As Exception
            Debug.Assert(False)
            MsgBox("Fehler: Wo: " & _
                ex.StackTrace & " Was: " & ex.Message)
            Throw New Exception( _
            "Fehler beim Schliessen " & _
            "des aktuellen Dokuments.")
        End Try
    End Sub

    Friend Sub DokumentSpeichern( _
        Optional ByVal eSaveAsOptions As SwConst.swSaveAsOptions_e = _
        SwConst.swSaveAsOptions_e.swSaveAsOptions_Silent, _
        Optional ByRef eSaveWarning As SwConst.swFileSaveWarning_e = 0)
        Try
            'SolidWorks Instanz prüfen
            If oSwAppCls Is Nothing Then
                'Fehler
                Throw New Exception( _
                "Keine SolidWorks Instanz vorhanden.")
            Else
                'Rückgabewerte
                Dim bStatus As Boolean
                Dim eError As SwConst.swFileSaveError_e
                'Aktuelles Dokument speichern
                bStatus = Me.AktuellesDokument.Save3( _
                       eSaveAsOptions, _
                       CType(eError, SwConst.swFileSaveError_e), _
                       CType(eSaveWarning, SwConst.swFileSaveWarning_e))
                'Rückgaben auswerten
                If bStatus = False Then
                    Throw New Exception( _
                    "Aktuelles Dokument wurde nicht gespeichert." & _
                    vbNewLine & "Errormeldung der API Methode: " & _
                    eError.ToString & ".")
                End If
            End If
        Catch ex As Exception
            Debug.Assert(False)
            MsgBox("Fehler: Wo: " & _
                ex.StackTrace & " Was: " & ex.Message)
            Throw New Exception( _
            "Fehler beim Speichern " & _
            "des aktuellen Dokuments.")
        End Try
    End Sub
```

```
Friend Sub DokumentSpeichernUnter( _
ByVal sSpeicherpfad As String, _
Optional ByVal eSaveAsOptions As SwConst.swSaveAsOptions_e = _
SwConst.swSaveAsOptions_e.swSaveAsOptions_Silent, _
Optional ByRef eSaveWarning As SwConst.swFileSaveWarning_e = 0)
    Try
        'SolidWorks Instanz prüfen
        If oSwAppCls Is Nothing Then
            'Fehler
            Throw New Exception( _
            "Keine SolidWorks Instanz vorhanden.")
        Else
            'Rückgabewerte
            Dim bStatus As Boolean
            Dim eError As SwConst.swFileSaveError_e
            'Aktuelles Dokument speichern
            bStatus = Me.AktuellesDokument.SaveAs4( _
                   sSpeicherpfad, _
                   SwConst.swSaveAsVersion_e.swSaveAsCurrentVersion, _
                   eSaveAsOptions, _
                   CType(eError, SwConst.swFileSaveError_e), _
                   CType(eSaveWarning, SwConst.swFileSaveWarning_e))
            'Rückgaben auswerten
            If bStatus = False Then
                Throw New Exception( _
                "Aktuelles Dokument wurde nicht gespeichert." & _
                vbNewLine & "Errormeldung der API Methode: " & _
                eError.ToString & ".")
            End If
        End If
    Catch ex As Exception
        Debug.Assert(False)
        MsgBox("Fehler: Wo: " & _
            ex.StackTrace & " Was: " & ex.Message)
        Throw New Exception( _
        "Fehler beim ""Speichern unter"" " & _
        "des aktuellen Dokuments.")
    End Try
End Sub

Friend ReadOnly Property AktuellesDokumentSpeicherpfad() As String
    Get
        'Aktuelles Dokument prüfen
        If Me.AktuellesDokument Is Nothing Then
            'Leerer String als ungültige Rückgabe verwenden
            Return ""
        Else
            'Speicherpfad zurückgeben
            Return Me.AktuellesDokument.GetPathName
        End If
    End Get
End Property

Friend ReadOnly Property AktuellesDokumentExtension() As String
    Get
        'Speicherpfad aufschlüsseln
        Return IO.Path.GetExtension(Me.AktuellesDokumentSpeicherpfad)
    End Get
End Property

Friend ReadOnly Property AktuellesDokumentVerzeichnis() As String
    Get
        'Speicherpfad aufschlüsseln
        Return IO.Path.GetDirectoryName( _
            Me.AktuellesDokumentSpeicherpfad)
    End Get
End Property

Friend ReadOnly Property AktuellesDokumentDateiName() As String
    Get
        'Speicherpfad aufschlüsseln
        Return IO.Path.GetFileName(Me.AktuellesDokumentSpeicherpfad)
    End Get
End Property
```

```
    ''' <summary>
    ''' Ermittelt die Referenzen eines SolidWorks Dokument
    ''' und gibt diese als String Liste zurück.
    ''' Das Dokument kann durch den Optionalen Parameter
    ''' sDokumentPfad angegeben werden. Ist dies nicht der Fall
    ''' werden die Referenzen des aktuell in
    ''' SolidWorks geöffneten Dokuments ermittelt.
    ''' </summary>
    ''' <param name="sDokumentPfad">Optionaler Parameter.
    ''' Wird ein Dokumentpfad übergeben,
    ''' werden die Referenzen dieses Dokuments ermitteln</param>
    ''' <param name="bAuchReadOnlyInfo">Optionaler Parameter.
    ''' Bestimmt ob die ReadOnly Informationen ermittelt werden.
    ''' Diese sind jedoch nie in der Stringliste enthalten</param>
    ''' <returns>Referenzen des ausgelesenen Dokuments.
    ''' Wenn dieses keine Referenzen besitz Nothing</returns>
    ''' <remarks></remarks>
    Friend Function ReferenzenEinesDokumentsErmitteln( _
        Optional ByVal sDokumentPfad As String = "", _
        Optional ByVal bAuchReadOnlyInfo As Boolean = False) As _
        System.Collections.Generic.List(Of String)
        Try
            Dim iStep As Integer
            'Dummy SaveArray
            Dim oDummyArray() As Object
            'Referenzliste
            Dim scReferenzen As _
                System.Collections.Generic.List(Of String)
            'SolidWorks Application Objekt prüfen
            If oSwAppCls Is Nothing Then
                'Fehler erzeugen
                Throw New Exception( _
                "Keine SolidWorks Instanz vorhanden.")
            End If
            'Existenz der übergeben Datei prüfen
            If IO.File.Exists(sDokumentPfad) = False Then
                'Über das ModelDoc2 ermitteln
                oDummyArray = CType( _
                    Me.AktuellesDokument.GetDependencies2( _
                    True, False, False), [Object]())
            Else
                'Über die SolidWorks Application
                oDummyArray = CType( _
                    oSwAppCls.GetDocumentDependencies2( _
                    sDokumentPfad, True, _
                    False, False), [Object]())
            End If
            'Überprüfen ob das Dokument
            'überhaupt Referenzen hat!
            If IsArray(oDummyArray) = False Then
                'Wenn keine Referenzen gefunden wurden,
                'war das kein Fehler der Funktion!
                Return Nothing
            End If
            'Parameter addReadOnlyInfo berücksichtigen
            If bAuchReadOnlyInfo Then
                iStep = 3
            Else
                iStep = 2
            End If
            'Stringliste initialisieren
            scReferenzen = New _
                System.Collections.Generic.List(Of String)
            'SaveArray aufschlüsseln
            'und Dokumentpfade verarbeiten
            For i As Integer = 1 To _
                oDummyArray.GetUpperBound(0) Step iStep
                scReferenzen.Add(CStr(oDummyArray(i)))
            Next
            Return scReferenzen
        Catch ex As Exception
            Debug.Assert(False)
            MsgBox("Fehler: Wo: " & _
             ex.StackTrace & " Was: " & ex.Message)
            Return Nothing
        End Try
    End Function
```

```
    Friend Function DokumentEigenschaftenUndWertErmitteln( _
            Optional ByVal sKonfiguration As String = "") As _
            System.Collections.Generic.Dictionary(Of String, String)
        Try
            'Dictionary für die ermittelten
            'Dokumenteigenschaften und dessen Wert
            Dim scEigenschaftenUndWert As _
                System.Collections.Generic.Dictionary( _
                Of String, String)
            'Je nach SolidWorks Version die richtige Methode ausführen
            If Me.VersionInJahr >= 2007 Then
                'Ab SolidWorks 2007
                scEigenschaftenUndWert = _
                    Me.DokumentEigenschaftenUndWertErmittelnAbSw2007( _
                    sKonfiguration)
            Else
                'Für ältere SolidWorks Versionen
                scEigenschaftenUndWert = _
                    Me.DokumentEigenschaftenUndWertErmittelnVorSw2007( _
                    sKonfiguration)
            End If

            Return scEigenschaftenUndWert
        Catch ex As Exception
            Debug.Assert(False)
            MsgBox("Fehler: Wo: " & _
             ex.StackTrace & " Was: " & ex.Message)
            Return Nothing
        End Try
    End Function

    Private Function DokumentEigenschaftenUndWertErmittelnVorSw2007( _
            Optional ByVal sKonfiguration As String = "") As _
            System.Collections.Generic.Dictionary(Of String, String)
        Try
            'Dictionary für die ermittelten
            'Dokumenteigenschaften und dessen Wert
            Dim scEigenschaftenUndWert As _
                System.Collections.Generic.Dictionary( _
                Of String, String)
            'Stringarray für die vorhandenen
            'Dokumenteigenschaften des Dokuments
            Dim sEigenschaften() As String
            'SolidWorks Application Objekt prüfen
            If oSwAppCls Is Nothing Then
                'Fehler erzeugen
                Throw New Exception( _
                "Keine SolidWorks Instanz vorhanden.")
            End If
            'Dokumenteigenschaften des aktuellen
            'Dokuments ermitteln
            sEigenschaften = CType( _
                Me.AktuellesDokument.GetCustomInfoNames2( _
                sKonfiguration), String())
            'Dictionary initialisieren
            scEigenschaftenUndWert = New _
                System.Collections.Generic.Dictionary( _
                Of String, String)
            'Alle Eigenschaften berücksichtigen
            'und dessen Wert ermitteln
            For iEigenschaft As Integer = _
                        0 To sEigenschaften.GetUpperBound(0)
                'Eigenschaften und dessen ermitteln
                'Wert als Eintrag in die Dictionary
                scEigenschaftenUndWert.Add( _
                    sEigenschaften(iEigenschaft), _
                    Me.AktuellesDokument.GetCustomInfoValue( _
                    sKonfiguration, sEigenschaften(iEigenschaft)))
            Next
            Return scEigenschaftenUndWert
        Catch ex As Exception
            Debug.Assert(False)
            MsgBox("Fehler: Wo: " & _
             ex.StackTrace & " Was: " & ex.Message)
            Return Nothing
        End Try
    End Function
```

```
    Private Function DokumentEigenschaftenUndWertErmittelnAbSw2007( _
      Optional ByVal sKonfiguration As String = "") As _
      System.Collections.Generic.Dictionary(Of String, String)
        Try
            'Dictionary für die ermittelten
            'Dokumenteigenschaften und dessen Wert
            Dim scEigenschaftenUndWert As _
                System.Collections.Generic.Dictionary( _
                Of String, String)
            'Stringarray für die vorhandenen
            'Dokumenteigenschaften des Dokuments
            Dim sEigenschaften() As String
            'Inhalt der Dokumenteigenschaft
            Dim sInhalt As String = Nothing
            'Erweiterungsobjekt eines
            'SolidWorks Dokuments
            Dim oModelDocExt As SldWorks.ModelDocExtension
            'Objekt für die Eigenschaften
            'eines Dokuments
            Dim oCustomPropertyManager As _
                SldWorks.CustomPropertyManager
            'SolidWorks Application Objekt prüfen
            If oSwAppCls Is Nothing Then
                'Fehler erzeugen
                Throw New Exception( _
                "Keine SolidWorks Instanz vorhanden.")
            End If
            'Erweiterungsobjekt des
            'Dokuments initialisieren
            oModelDocExt = Me.AktuellesDokument.Extension
            'Dokumenteigenschaftenmanger
            'des aktuell Dokuments initialisieren
            oCustomPropertyManager = _
                oModelDocExt.CustomPropertyManager( _
                sKonfiguration)
            sEigenschaften = CType( _
                oCustomPropertyManager.GetNames, String())
            'Dictionary initialisieren
            scEigenschaftenUndWert = New _
                System.Collections.Generic.Dictionary( _
                Of String, String)
            'Alle Eigenschaften berücksichtigen
            'und dessen Wert ermitteln
            For iEigenschaft As Integer = _
                        0 To sEigenschaften.GetUpperBound(0)
                'Inhalt der Eigenschaft ermitteln
                oCustomPropertyManager.Get2( _
                    sEigenschaften(iEigenschaft), sInhalt, Nothing)
                'Eigenschaften und dessen ermitteln
                'Wert als Eintrag in die Dictionary
                scEigenschaftenUndWert.Add( _
                    sEigenschaften(iEigenschaft), _
                    sInhalt)
            Next
            Return scEigenschaftenUndWert
        Catch ex As Exception
            Debug.Assert(False)
            MsgBox("Fehler: Wo: " & _
             ex.StackTrace & " Was: " & ex.Message)
            Return Nothing
        End Try
    End Function

End Class
```

Formular SchulungFrm

```
Public Class SchulungFrm

    Private Sub cmdCreate_Click( _
            ByVal sender As System.Object, _
            ByVal e As System.EventArgs) _
            Handles cmdCreate.Click
        Try
            'Eigene SolidWorks Klasse deklarieren
            Dim oMySldWorks As MySldWorksCls
            'Eigene SolidWorks Klasse initialisieren
            oMySldWorks = New MySldWorksCls
            'Create Methode ausführen
            oMySldWorks.SolidWorksInstanzCreate()
            'Version in einer MessageBox anzeigen
            oMySldWorks.VersionAnzeigen()
            'Eigene SolidWorks Klasse leeren
            oMySldWorks = Nothing
        Catch ex As Exception
            Debug.Assert(False)
            MsgBox("Fehler: Wo: " & _
                ex.StackTrace & " Was: " & ex.Message)
        End Try
    End Sub

    Private Sub cmdGet_Click( _
            ByVal sender As System.Object, _
            ByVal e As System.EventArgs) _
            Handles cmdGet.Click
        Try
            'Eigene SolidWorks Klasse deklarieren
            Dim oMySldWorks As MySldWorksCls
            'Eigene SolidWorks Klasse initialisieren
            oMySldWorks = New MySldWorksCls
            'Get Methode ausführen
            oMySldWorks.SolidWorksInstanz()
            'SolidWorks Instanz prüfen
            If oMySldWorks.MySldWorks IsNot Nothing Then
                'Makroverzeichnis ermitteln und ändern
                Me.MakroVerzeichnis(oMySldWorks)
                'Version in einer MessageBox anzeigen
                oMySldWorks.VersionAnzeigen()
            End If
            'Eigene SolidWorks Klasse leeren
            oMySldWorks = Nothing
        Catch ex As Exception
            Debug.Assert(False)
            MsgBox("Fehler: Wo: " & _
                ex.StackTrace & " Was: " & ex.Message)
        End Try
    End Sub

    Private Sub MakroVerzeichnis(ByVal oMySldWorks As MySldWorksCls)
        Try
        'Aktuelles Makroverzeichnis
        MsgBox("Aktuelles Makroverzeichnis: " & vbNewLine & _
        oMySldWorks.MySldWorks.GetUserPreferenceStringValue( _
        SwConst.swUserPreferenceStringValue_e.swFileLocationsMacros))
        'Makroverzeichnis in den eigenen Desktop ändern
        oMySldWorks.MySldWorks.SetUserPreferenceStringValue( _
        SwConst.swUserPreferenceStringValue_e.swFileLocationsMacros, _
        System.Environment.GetFolderPath( _
        Environment.SpecialFolder.DesktopDirectory))
        Catch ex As Exception
            Debug.Assert(False)
            MsgBox("Fehler: Wo: " & _
                ex.StackTrace & " Was: " & ex.Message)
        End Try
    End Sub

End Class
```

Formular SldWorksEventFrm

```
Public Class SldWorksEventFrm

    Dim WithEvents oSwAppCls As SldWorks.SldWorks
    Dim WithEvents oSwPartCls As SldWorks.PartDoc
    Dim WithEvents oSwAssemblyCls As SldWorks.AssemblyDoc
    Dim WithEvents oSwDrawingCls As SldWorks.DrawingDoc

    Private Function oSwAppCls_FileOpenNotify2( _
            ByVal FileName As String) As Integer _
            Handles oSwAppCls.FileOpenNotify2

        Try
            'Den übergebenen Dateinamen
            'in die ListBox einfügen
            '(Ungültiger Thread
            'übergreifender Zugriff)
            Me.lstEreignis.Items.Add("Öffnen: " & FileName)
            'ModelDoc2 deklarieren
            Dim oSwModel As SldWorks.ModelDoc2
            'Dokumenten Objekt belegen
            oSwModel = CType(oSwAppCls.ActiveDoc, _
                             SldWorks.ModelDoc2)
            Select Case oSwModel.GetType
                Case SwConst.swDocumentTypes_e.swDocPART
                    oSwPartCls = CType(oSwModel, _
                            SldWorks.PartDoc)
                Case SwConst.swDocumentTypes_e.swDocASSEMBLY
                    oSwAssemblyCls = CType(oSwModel, _
                            SldWorks.AssemblyDoc)
                Case SwConst.swDocumentTypes_e.swDocDRAWING
                    oSwDrawingCls = CType(oSwModel, _
                            SldWorks.DrawingDoc)
                Case Else
                    Debug.Assert(False)
            End Select
            'Nicht mehr benötigtes Objekt freigeben
            oSwModel = Nothing
        Catch ex As Exception
            Debug.Assert(False)
            MsgBox("Fehler: Wo: " & _
                ex.StackTrace & " Was: " & ex.Message)
        End Try

    End Function

    Private Function oSwAppCls_FileOpenPreNotify( _
            ByVal FileName As String) As Integer _
            Handles oSwAppCls.FileOpenPreNotify
        Try
            If MsgBox( _
            "Es wurde versucht in SolidWorks" & _
            vbNewLine & "die Datei: " & FileName & _
            " zu öffnen: " & vbNewLine & _
            "Möchten Sie dies verhintern?", _
            MsgBoxStyle.Question Or MsgBoxStyle.YesNo, _
            "Öffnen der Datei abbrechen") = _
            MsgBoxResult.Yes Then
                'Ereignis abbrechen
                Return CInt(True)
            End If
        Catch ex As Exception
            Debug.Assert(False)
            MsgBox("Fehler: Wo: " & _
               ex.StackTrace & " Was: " & ex.Message)
        End Try
    End Function
```

```
Private Sub cmdBeenden_Click( _
    ByVal sender As System.Object, _
    ByVal e As System.EventArgs) _
    Handles cmdBeenden.Click
    Try
        'SolidWorks freigeben
        oSwPartCls = Nothing
        oSwDrawingCls = Nothing
        oSwAssemblyCls = Nothing
        oSwAppCls = Nothing
        'Formular schließen
        Me.Close()
    Catch ex As Exception
        Debug.Assert(False)
        Trace.WriteLine("Fehler: Wo: " & _
            ex.StackTrace & " Was: " & ex.Message)
    End Try
End Sub

Private Sub cmdStart_Click( _
    ByVal sender As System.Object, _
    ByVal e As System.EventArgs) _
    Handles cmdStart.Click
    Try
        'Eigene SolidWorks Klasse deklarieren
        Dim oMySldWorks As MySldWorksCls
        'Eigene SolidWorks Klasse initialisieren
        oMySldWorks = New MySldWorksCls
        'Get Methode ausführen
        oSwAppCls = oMySldWorks.SolidWorksInstanz()
        'Eigene SolidWorks Klasse leeren
        oMySldWorks = Nothing
    Catch ex As Exception
        Debug.Assert(False)
        Trace.WriteLine("Fehler: Wo: " & _
            ex.StackTrace & " Was: " & ex.Message)
    End Try
End Sub

Private Sub SldWorksEventFrm_Load( _
        ByVal sender As System.Object, _
        ByVal e As System.EventArgs) _
        Handles MyBase.Load
    Try
    'Unerlaubte Thread übergreifende Zugriffe erlauben
    System.Windows.Forms.Form.CheckForIllegalCrossThreadCalls = False
    Catch ex As Exception
        Debug.Assert(False)
        Trace.WriteLine("Fehler: Wo: " & _
            ex.StackTrace & " Was: " & ex.Message)
    End Try
End Sub

Private Function oSwAssemblyCls_FileSaveNotify( _
        ByVal FileName As String) As Integer _
        Handles oSwAssemblyCls.FileSaveNotify

    Try
        'Den übergebenen Dateinamen
        'in die ListBox einfügen
        '(Ungültiger Thread
        'übergreifender Zugriff)
        Me.lstEreignis.Items.Add("Speichern: " & FileName)
    Catch ex As Exception
        Debug.Assert(False)
        MsgBox("Fehler: Wo: " & _
            ex.StackTrace & " Was: " & ex.Message)
    End Try

End Function
```

```
    Private Function oSwDrawingCls_FileSaveNotify( _
            ByVal FileName As String) As Integer _
            Handles oSwDrawingCls.FileSaveNotify

        Try
            'Den übergebenen Dateinamen
            'in die ListBox einfügen
            '(Ungültiger Thread
            'übergreifender Zugriff)
            Me.lstEreignis.Items.Add("Speichern: " & FileName)
        Catch ex As Exception
            Debug.Assert(False)
            MsgBox("Fehler: Wo: " & _
                ex.StackTrace & " Was: " & ex.Message)
        End Try

    End Function

    Private Function oSwPartCls_FileSaveNotify( _
            ByVal FileName As String) As Integer _
            Handles oSwPartCls.FileSaveNotify

        Try
            'Den übergebenen Dateinamen
            'in die ListBox einfügen
            '(Ungültiger Thread
            'übergreifender Zugriff)
            Me.lstEreignis.Items.Add("Speichern: " & FileName)
        Catch ex As Exception
            Debug.Assert(False)
            MsgBox("Fehler: Wo: " & _
                ex.StackTrace & " Was: " & ex.Message)
        End Try

    End Function

End Class
```

Formular SldWorksDokumenteFrm

```
Public Class SldWorksDokumenteFrm

    Private Sub cmdOeffnen_Click(ByVal sender As System.Object, _
            ByVal e As System.EventArgs) Handles cmdOeffnen.Click
        Try
            'Dialog aufrufen
            With Me.OpenFileDialog1
                'Keine Auswahlvorgabe
                .FileName = ""
                'Welche Dateitypen dürfen
                'in diesem Dialog ausgewählt werden
                .Filter = "SolidWorks Bauteil (*.sldprt)|*.sldprt|" & _
                     "SolidWorks Baugruppe (*.sldasm)|*.sldasm|" & _
                     "SolidWorks Zeichnung (*.slddrw)|*.slddrw"
                'Titel das Dialogs
                .Title = "SolidWorks Dokument öffnen"
            End With
            'Wenn eine Datei geöffnet werden soll
            If Me.OpenFileDialog1.ShowDialog() = _
                 Windows.Forms.DialogResult.OK Then
                'Ausgewältes SolidWorks Dokument
                Debug.Print( _
                    Me.OpenFileDialog1.FileName)
                'Eigene SolidWorks Klasse deklarieren
                Dim oMySldWorks As MySldWorksCls
                'Eigene SolidWorks Klasse initialisieren
                oMySldWorks = New MySldWorksCls
                'SolidWorks initialisieren
                oMySldWorks.SolidWorksInstanz()
                'Dokument in SolidWorks öffnen
                oMySldWorks.DokumentOeffnen( _
                        Me.OpenFileDialog1.FileName)
                'Eigene SolidWorks Klasse leeren
                oMySldWorks = Nothing
            End If
        Catch ex As Exception
            Debug.Assert(False)
            MsgBox("Fehler: Wo: " & _
                ex.StackTrace & " Was: " & ex.Message)
        End Try
    End Sub

    Private Sub cmdSchliessen_Click(ByVal sender As System.Object, _
        ByVal e As System.EventArgs) Handles cmdSchliessen.Click
        Try
            'Eigene SolidWorks Klasse deklarieren
            Dim oMySldWorks As MySldWorksCls
            'Eigene SolidWorks Klasse initialisieren
            oMySldWorks = New MySldWorksCls
            'SolidWorks initialisieren
            oMySldWorks.SolidWorksInstanz()
            'Beispielaufruf für die Ermittlung
            'und Aufschlüßlung des Speicherpfad
            'Ausgabe in einer MessageBox
            MsgBox("Aktuelles Dokument" & _
                vbNewLine & "Pfad: " & _
                oMySldWorks.AktuellesDokumentSpeicherpfad & _
                vbNewLine & "Name: " & _
                oMySldWorks.AktuellesDokumentDateiName & _
                vbNewLine & "Extension: " & _
                oMySldWorks.AktuellesDokumentExtension & _
                vbNewLine & "Verzeichnis: " & _
                oMySldWorks.AktuellesDokumentVerzeichnis)
            'Aktuelles Dokument schliessen
            oMySldWorks.DokumentSchliessen()
            'Eigene SolidWorks Klasse leeren
            oMySldWorks = Nothing
        Catch ex As Exception
            Debug.Assert(False)
            MsgBox("Fehler: Wo: " & _
                ex.StackTrace & " Was: " & ex.Message)
        End Try
    End Sub
```

```
    Private Sub cmdSpeichern_Click( _
        ByVal sender As System.Object, _
        ByVal e As System.EventArgs) _
        Handles cmdSpeichern.Click
        Try
            'Eigene SolidWorks Klasse deklarieren
            Dim oMySldWorks As MySldWorksCls
            'Eigene SolidWorks Klasse initialisieren
            oMySldWorks = New MySldWorksCls
            'SolidWorks initialisieren
            oMySldWorks.SolidWorksInstanz()
            'Aktuelles Dokument speichern
            oMySldWorks.DokumentSpeichern()
            'Eigene SolidWorks Klasse leeren
            oMySldWorks = Nothing
        Catch ex As Exception
            Debug.Assert(False)
            MsgBox("Fehler: Wo: " & _
                ex.StackTrace & " Was: " & ex.Message)
        End Try
    End Sub

    Private Sub cmdSpeichernUnter_Click( _
        ByVal sender As System.Object, _
        ByVal e As System.EventArgs) _
        Handles cmdSpeichernUnter.Click
        Try
            Dim eSldDocTyp As SwConst.swDocumentTypes_e
            'Eigene SolidWorks Klasse deklarieren
            Dim oMySldWorks As MySldWorksCls
            'Eigene SolidWorks Klasse initialisieren
            oMySldWorks = New MySldWorksCls
            'SolidWorks initialisieren
            oMySldWorks.SolidWorksInstanz()
            'Einstellungen im SaveFileDialog
            With Me.SaveFileDialog1
                'Vorgabe ist der Titel
                'des SolidWorks Dokuments
                .FileName = oMySldWorks.AktuellesDokument.GetTitle
                'Welche Dateitypen dürfen
                'in diesem Dialog ausgewählt werden
                .Filter = "SolidWorks Bauteil (*.sldprt)|*.sldprt|" & _
                    "SolidWorks Baugruppe (*.sldasm)|*.sldasm|" & _
                    "SolidWorks Zeichnung (*.slddrw)|*.slddrw|" & _
                    "PDF (*.pdf)|*.pdf|" & _
                    "JPEG (*.jpg)|*.jpg"
                'Dateityp durch den
                'Dokumenttyp des Dokuments vorbelegen
                eSldDocTyp = CType( _
                    oMySldWorks.AktuellesDokument.GetType, _
                    SwConst.swDocumentTypes_e)
                Select Case eSldDocTyp
                    Case SwConst.swDocumentTypes_e.swDocPART
                        .FilterIndex = 0
                    Case SwConst.swDocumentTypes_e.swDocASSEMBLY
                        .FilterIndex = 1
                    Case SwConst.swDocumentTypes_e.swDocDRAWING
                        .FilterIndex = 2
                End Select
                'Titel das Dialogs
                .Title = "SolidWorks Dokument speichern"
            End With
```

```
            'Wenn die Datei gespeichert werden soll
            If Me.SaveFileDialog1.ShowDialog() = _
                    Windows.Forms.DialogResult.OK Then
                'Speichern unter Methode
                'der eigenen Klasse ausführen
                oMySldWorks.DokumentSpeichernUnter( _
                   Me.SaveFileDialog1.FileName)
            End If
            'Eigene SolidWorks Klasse leeren
            oMySldWorks = Nothing
        Catch ex As Exception
            Debug.Assert(False)
            MsgBox("Fehler: Wo: " & _
                ex.StackTrace & " Was: " & ex.Message)
        End Try
    End Sub

End Class
```

Formular SldWorksReferenzenFrm

```
Public Class SldWorksReferenzenFrm

    Private Sub cmdErmitteln_Click( _
           ByVal sender As System.Object, _
           ByVal e As System.EventArgs) _
           Handles cmdErmitteln.Click
        Try
            Dim scReferenzen As _
                System.Collections.Generic.List(Of String)
            Dim sSchraubstockPfad As String
            'Baugruppe festlegen
            sSchraubstockPfad = _
                "C:\Baugruppen\Schraubstock" & _
                "\Schraubstock.sldasm"
            'Eigene SolidWorks Klasse deklarieren
            Dim oMySldWorks As MySldWorksCls
            'Eigene SolidWorks Klasse initialisieren
            oMySldWorks = New MySldWorksCls
            'SolidWorks initialisieren
            oMySldWorks.SolidWorksInstanz()
            'Referenzen des Dokuments speichern
            scReferenzen = _
                oMySldWorks.ReferenzenEinesDokumentsErmitteln( _
                sSchraubstockPfad)
            'Rückgabe prüfen
            If Not scReferenzen Is Nothing Then
                'Listbox befüllen
                Me.lstReferenzen.Items.Clear()
                For Each s As String In scReferenzen
                    Me.lstReferenzen.Items.Add(s)
                Next
            End If
            'Eigene SolidWorks Klasse leeren
            oMySldWorks = Nothing
        Catch ex As Exception
            Debug.Assert(False)
            MsgBox("Fehler: Wo: " & _
             ex.StackTrace & " Was: " & ex.Message)
        End Try
    End Sub

    Private Sub cmdSchliessen_Click(ByVal sender As System.Object, _
           ByVal e As System.EventArgs) Handles cmdSchliessen.Click
        Try
            Me.Close()
        Catch ex As Exception
            Debug.Assert(False)
            MsgBox("Fehler: Wo: " & _
             ex.StackTrace & " Was: " & ex.Message)
        End Try
    End Sub
```

```
    Private Sub cmdAnpassen_Click( _
            ByVal sender As System.Object, _
            ByVal e As System.EventArgs) Handles cmdAnpassen.Click
        Try
            Dim scReferenzen As _
                System.Collections.Generic.List(Of String)
            Dim sSchraubstockPfad As String
            'Baugruppe festlegen
            sSchraubstockPfad = _
                "C:\Baugruppen\Schraubstock" & _
                "\Schraubstock.sldasm"
            'Eigene SolidWorks Klasse deklarieren
            Dim oMySldWorks As MySldWorksCls
            'Eigene SolidWorks Klasse initialisieren
            oMySldWorks = New MySldWorksCls
            'SolidWorks initialisieren
            oMySldWorks.SolidWorksInstanz()
            'Referenzen des Dokuments speichern
            scReferenzen = _
                oMySldWorks.ReferenzenEinesDokumentsErmitteln( _
                sSchraubstockPfad)
            'Rückgabe prüfen
            If Not scReferenzen Is Nothing Then
                'Referenzen auswerten
                For Each sReferenz As String In scReferenzen
                    If System.IO.Path.GetDirectoryName(sReferenz) = _
                            "C:\Beispiel\Schraubstock" Then
                       oMySldWorks.MySldWorks.ReplaceReferencedDocument( _
                            sSchraubstockPfad, sReferenz, _
                            "C:\Baugruppe\Schraubstock\" & _
                            System.IO.Path.GetFileName(sReferenz))
                    End If
                Next

            End If
            'Eigene SolidWorks Klasse leeren
            oMySldWorks = Nothing
        Catch ex As Exception
            Debug.Assert(False)
            MsgBox("Fehler: Wo: " & _
             ex.StackTrace & " Was: " & ex.Message)
        End Try
    End Sub

End Class
```

Formular SldWorksEigenschaftenFrm

```
Public Class SldWorksEigenschaftenFrm

    Private Sub cmdErmitteln_Click( _
            ByVal sender As System.Object, _
            ByVal e As System.EventArgs) _
            Handles cmdErmitteln.Click
        Try
            'Alle Dokumenteigenschaften mit ihrem Wert
            Dim scEigenschaftenUndWert As _
                System.Collections.Generic.Dictionary(Of String, String)
            Dim sEinzelneEigenschaftUndWert As String
            'Eigene SolidWorks Klasse deklarieren
            Dim oMySldWorks As MySldWorksCls
            'Eigene SolidWorks Klasse initialisieren
            oMySldWorks = New MySldWorksCls
            'SolidWorks initialisieren
            oMySldWorks.SolidWorksInstanz()
            'Dokumenteigenschaften ermitteln
            scEigenschaftenUndWert = _
                oMySldWorks.DokumentEigenschaftenUndWertErmitteln()
            'Rückgabe prüfen
            If Not scEigenschaftenUndWert Is Nothing Then
                'Listbox befüllen
                Me.lstEigenschaften.Items.Clear()
                For Each s As String In scEigenschaftenUndWert.Keys
                    sEinzelneEigenschaftUndWert = _
                        "Eigenschaft: " & s & " Wert: " & _
                        scEigenschaftenUndWert.Item(s)
                    Me.lstEigenschaften.Items.Add( _
                        sEinzelneEigenschaftUndWert)
                Next
            End If
            'Eigene SolidWorks Klasse leeren
            oMySldWorks = Nothing
        Catch ex As Exception
            Debug.Assert(False)
            MsgBox("Fehler: Wo: " & _
             ex.StackTrace & " Was: " & ex.Message)
        End Try
    End Sub

    Private Sub cmdSchliessen_Click(ByVal sender As System.Object, _
            ByVal e As System.EventArgs) Handles cmdSchliessen.Click
        Try
            Me.Close()
        Catch ex As Exception
            Debug.Assert(False)
            MsgBox("Fehler: Wo: " & _
             ex.StackTrace & " Was: " & ex.Message)
        End Try
    End Sub
```

```
Private Sub cmdAnpassen_Click( _
        ByVal sender As System.Object, _
        ByVal e As System.EventArgs) Handles cmdAnpassen.Click
    Try
        'Eigene SolidWorks Klasse deklarieren
        Dim oMySldWorks As MySldWorksCls
        'Eigene SolidWorks Klasse initialisieren
        oMySldWorks = New MySldWorksCls
        'SolidWorks initialisieren
        oMySldWorks.SolidWorksInstanz()
        'Dokumenteigenschaft manipulieren
        'Je nach SolidWorks Version die richtige Methode ausführen
        If oMySldWorks.VersionInJahr >= 2007 Then
            'Ab SolidWorks 2007
            Me.AnpassenAbSw2007(oMySldWorks)
        Else
            'Für ältere SolidWorks Versionen
            Me.AnpassenVorSw2007(oMySldWorks)
        End If
        'Eigene SolidWorks Klasse leeren
        oMySldWorks = Nothing
    Catch ex As Exception
        Debug.Assert(False)
        MsgBox("Fehler: Wo: " & _
         ex.StackTrace & " Was: " & ex.Message)
    End Try
End Sub

Private Sub AnpassenVorSw2007( _
        ByVal oMySldWorks As MySldWorksCls)
    Try
        'Dokumenteigenschaft löschen
        oMySldWorks.AktuellesDokument.DeleteCustomInfo2( _
            "", "Auto")
        'Dokumenteigenschaft ändern
        oMySldWorks.AktuellesDokument.CustomInfo2( _
            "", "Autor") = "Schulungsteilnehmer"
        'Neue Dokumenteigenschaft hinzufügen
        oMySldWorks.AktuellesDokument.AddCustomInfo3( _
            "", "Datum", _
            SwConst.swCustomInfoType_e.swCustomInfoDate, _
            Now.ToString)
    Catch ex As Exception
        Debug.Assert(False)
        MsgBox("Fehler: Wo: " & _
         ex.StackTrace & " Was: " & ex.Message)
    End Try
End Sub
```

```
    Private Sub AnpassenAbSw2007( _
            ByVal oMySldWorks As MySldWorksCls)
        Try
            'Erweiterungsobjekt eines
            'SolidWorks Dokuments
            Dim oModelDocExt As SldWorks.ModelDocExtension
            'Objekt für die Eigenschaften
            'eines Dokuments
            Dim oCustomPropertyManager As _
                SldWorks.CustomPropertyManager
            'Erweiterungsobjekt des
            'Dokuments initialisieren
            oModelDocExt = oMySldWorks.AktuellesDokument.Extension
            'Dokumenteigenschaftenmanger
            'des aktuell Dokuments initialisieren
            oCustomPropertyManager = _
                oModelDocExt.CustomPropertyManager( _
                "")
            'Dokumenteigenschaft löschen
            oCustomPropertyManager.Delete("Auto")
            'Dokumenteigenschaft ändern
            oCustomPropertyManager.Set( _
                "Autor", "Schulungsteilnehmer")
            'Neue Dokumenteigenschaft hinzufügen
            oCustomPropertyManager.Add2( _
                "Datum", _
                SwConst.swCustomInfoType_e.swCustomInfoDate, _
                Now.ToString)
        Catch ex As Exception
            Debug.Assert(False)
            MsgBox("Fehler: Wo: " & _
             ex.StackTrace & " Was: " & ex.Message)
        End Try
    End Sub

End Class
```

B Stichwortverzeichnis SolidWorks API Befehle

SldWorks.SldWorks

SldWorks.RevisionNumber 22
SldWorks.Visible 22
SldWorks.SendMsgToUser2 39
SldWorks.FileOpenNotify2 46
SldWorks.ActiveDoc 51
SldWorks.FileOpenPreNotify 55
SldWorks.OpenDoc6 63
SldWorks.CloseDoc 70
SldWorks.QuitDoc 74
SldWorks.GetDocumentDependencies2 89
SldWorks.ReplaceReferencedDocument 98
SldWorks.GetUserPreferenceToggle 121
SldWorks.SetUserPreferenceToggle 123
SldWorks.GetUserPreferenceStringValue 125
SldWorks.SetUserPreferenceStringValue 125

SldWorks.ModelDoc2

ModelDoc2.GetType 52
ModelDoc2.GetTitle 71
ModelDoc2.Quit 74
ModelDoc2.Close 74
ModelDoc2.Save3 75
ModelDoc2.SaveAs4 79
ModelDoc2.GetPathName 84
ModelDoc2.GetDependencies2 89
ModelDoc2.GetCustomInfoManes2 104
ModelDoc2.GetCustomInfoValue 105
ModelDoc2.Extension 107
ModelDoc2.AddCustomInfo3 111
ModelDoc2.DeleteCustomInfo2 111
ModelDoc2.CustomInfo2 112

SldWorks.PartDoc

PartDoc.FileSaveNotify 54

SldWorks.AssemblyDoc

AssemblyDoc.FileSaveNotify 54

SldWorks.DrawingDoc

DrawingDoc.FileSaveNotify 54

SldWorks.ModelDocExtension

ModelDocExtension.CustomPropertyManager 107

SldWorks.CustomPropertyManager

CustomPropertyManager.GetNames 109
CustomPropertyManager.Get2 109
CustomPropertyManager.Add2 114
CustomPropertyManager.Delete 114
CustomPropertyManager.Set 114