TDIWiz

Text-Data-Integration-Wizard

Administratorhandbuch

Version 2.2

Ausgabe Januar 2021

Ein Produkt der Sklenar GmbH, Wien

Einleitung 1

Sprachelemente von TDIWiz 2

Ausdrücke 2
Variable 4
Definition und Gültigkeitsbereich 4
Zuweisung von Werten 5
Systemvariable 5
Kommandos zur Steuerung der Prozedurverarbeitung 5
IF *logical condition*$_1$ THEN 5
DO [UNTIL *logical condition*] 6
DONODES nodes [*filter*] 6
SELECT CASE *exp* 6
Sonstige Kommandos 6
Call dll,class,function — Aufruf eigene DLL 6
Createdocument template — Einfügen generiertes Dokument 7
Enabled var,{True|False} — Eingabefeld (de)aktivieren 7
ErrMsg exp — Abbruch mit eigener Fehlermeldung 7
Exit [Do] — Ausstieg aus Prozedur bzw. DO-Schleife 7
VarDef name[/type] — Definition Variable 7
Funktionen 7
Abs(num.exp) — Absolutwert von exp 8
Avg(nodes[,exp][,filter]) — Berechnen Durchschnitt 8
CBool(exp) — Umsetzung zu logischem Wert 8
CDate(exp) — Umsetzung zu Datumswert 8
CDbl(exp) — Umsetzung zu Double-Wert 8
Chr(integer) — Characterdarstellung eines ASCII-Wertes 8
CInt(exp) — Umsetzung zu Integer-Wert 8
Count(nodes [,filter]) — Anzahl der zugeordneten XML-Knoten 8
DateAdd(interval,count,date) — Datumsaddition 8
DateDiff(interval,from,to) — Datumsdifferenz 8
EndsWith(exp,string) — Abfrage, ob exp mit string endet 9
Format(exp,format) — Formatierungs-Funktion 9
GetDocVar(variable) — Lesen Dokumentvariable 10
If(condition,true-exp[,false-exp]) — Bedingte Texteinfügung 10
InStr([start,]string,search) — Suchen Zeichenkette 10
InStrRev(string,search[,start]) — Suchen Zeichenkette 10
IsNumeric(exp) — Prüfung, ob nummerisch 10
Left(exp,count) — Anfang einer Zeichenkette 10
Len(exp) — Länge einer Zeichenkette 10
List(nodes[,exp][,del] [,lastdel] [,filter]) — Auflistung 10
LTrim(exp) — Führende Leerstellen eliminieren 10
Max(nodes[,exp][,filter]) — Maximalwert 11
Mid(exp,start[,count]) — Teilkette 11
Min(nodes[,exp][,filter]) — Minimalwert 11
Replace(exp,oldvalue,newvalue) — Zeichenkette ersetzen 11
Right(exp,count) — Ende einer Zeichenkette 11
Round(exp,count) — Dezimalwert runden 11
RTrim(exp) — Abschließende Leerstellen eliminieren 11
StartsWith(exp,string) — Abfrage, ob exp mit string beginnt 11
Sum(nodes[,exp][,filter]) — Summe der zugeordneten Textknoten 11
TMExists(name) — Prüfung, ob Textmodul existiert 11
Trim(exp) — RTrim + LTrim 11
UCase(exp) — Umsetzen auf Großbuchstaben 12

VariableExists(variable) Prüfung, ob Dokumentvariable existiert 12
Weekday(date) Tag der Woche 12

Installation 13

TDIWiz Programmschnittstelle 14

Die wichtigsten Methoden und Attribute des TDIWiz-Objekts 15

Einleitung

Für die meisten zu generierenden Dokumente findet man mit den Grundfunktionen von TDIWiz sein Auslangen. Es kommt aber auch vor, dass diese doch nicht reichen bzw. dass beim Design der Inputdaten auf die Struktur einzelner Vorlagen eingegangen werden müsste.

Um auch diese Anforderungen einfach und leicht wartbar abdecken zu können, steht die Möglichkeit zur Verfügung, Inhalte durch Prozeduren (z.B. Rechenfunktionen, Fallunterscheidungen, etc.) zu generieren.

In diesem Handbuch werden die dazu verfügbaren Sprachelemente und Funktionen beschrieben.

Zusätzlich werden in einem Kapitel die von der TDIWiz-Anwenderprogrammschnittstelle (API) unterstützten Methoden und Attribute beschrieben.

Für Hinweise auf Fehler in diesem Handbuch, die uns trotz aller Sorgfalt möglicherweise unterlaufen sind, sind wir sehr dankbar!

Sklenar GmbH
SoftwareSolutions
Anton Baumgartner-Str. 44/C7/191
A-1230 Wien

E-Mail: info@sklenar.at

Sprachelemente von TDIWiz

Dieses Kapitel ist **nur dann** von Bedeutung, wenn **bedingte** Textteile oder **Prozeduren** verwendet werden!

Ausdrücke

Ausdrücke dienen zur Angabe von Parameterwerten bei Kommandos, Funktionsaufrufen und logischen Bedingungen

Ausdruck:

exp_1 [& exp_2] ... [& exp_n]

exp_n	{ nodes | variable | literal | spec.char | number | function | num.exp }

nodes

Der zugeordnete Dateninput wird für die Verarbeitung intern in XML-Daten konvertiert (Struktur wird im AddIn angezeigt). Diese können auf verschiedene Arten adressiert werden:

Knotenname

Die Suche nach den gewünschten Knoten hängt vom aktuellen „Kontext“ ab, nämlich ob der aktuelle Knoten innerhalb von Wiederholungsbereichen angesprochen wird oder nicht!

1. Innerhalb von geschachtelten Wiederholungsbereichen:
   Von innen nach außen werden die, den einzelnen Wiederholungs-bereichen zugeordneten XML-Knoten auf *Knotenname* durchsucht. Der Name muss **eindeutig** sein! Der Namens-Pfad muss soweit angegeben werden, dass dies der Fall ist (z.B.: Sales/Name und Support/Name im Wiederholungsbereich Customer)!

2. Außerhalb von Wiederholungsbereichen:
   Die Suche auf einen eindeutigen Knotennamen erfolgt im gesamten bzw. in dem durch *XMLFilter* eingeschränkten XML-Datenbereich.

Am einfachsten ist es, den eindeutigen Namen durch Doppelklick auf die unter XML-Data im linken Maskenbereich angezeigten Feldnamen zu übernehmen. Vor allem, da in diesem Fall auch die beim XML-Zugriff wichtige Groß-/Kleinschreibung des Namens beachtet wird!

- Zusätzlich kann in Sonderfällen innerhalb des Pfades auch auf einen **Index** oder durch Angabe eines **Filters** selektiert werden:

  Statt des Doppelklicks als Feldzuordnung wird aus dem Kontextmenü (rechte Maustaste) *Filter* aufgerufen.

  Das sich aus dem eingegebenen Pfadnamen ergebende Ergebnis muss eindeutig sein!

Beispiele
Customers[CompanyName="Around theHorn"]/ContactName
ContactName des Kunden Around theHorn
Customers[1]/ContactName
ContactName des 1.Kunden
Customers/ContactName[count()]
ContactName des letzten Kunden

XPath-Ausdruck

Als Alternative zum Zugriff mit Hilfe eines Filters kann auch über Xpath direkt auf Daten zugegriffen werden. XPath-Ausdrücke (XML-Abfragesprache) sind dadurch gekennzeichnet, dass sie mit „<“ beginnen und mit „>“ enden.

Variable, dynamisch zu bestimmende Teile des Ausdrucks können auf 2 Arten definiert werden:

1. Platzhalter (begrenzt durch „%“)
   <//Employees[EmployeeID=%$EmployeeID%]/FirstName>
   %$EmployeeID% wird durch den Wert der Variablen ersetzt!

2. der gesamte Ausdruck wird als Expression angegeben:
   <”//Employees[EmployeeID=” & $EmployeeID & “]/FirstName”>
   $EmployeeID wird durch den Variablenwert ersetzt!

Beide Fälle führen zum selben Ergebnis!

Innerhalb eines zusammengesetzten Ausdrucks oder einer Zeichenkette darf das Ergebnis von *nodes* nicht mehr als 1 Knoten beinhalten! Ausnahmen sind einige Funktionen (z.B. Max, Min, Sum, etc.) sowie das *DoNodes*-Kommando!

ACHTUNG: Bei Angabe des Knotennamens ist die Groß-/Kleinschreibung zu beachten!

variable — Bei Variablen muß ein „$“ vor den Namen gesetzt werden ($CustomerId).

literal — Literale sind durch Doppel-Hochkommas begrenzt (z.B.: "Example: ")

spec.char — Nicht darstellbare Zeichen können durch die Chr-Funktion (=> *Funktionen*) aufgerufen werden. Einige sind allerdings einfacher aufrufbar:

{EM}	Endemarke
{LF}, {NL}	Neue Zeile (Line Feed)
{CRLF}	Carriage Return + Line Feed
{TAB}	Tabulator

number — nummerische Konstante

function — => Kap. Funktionen

num.exp — Nummerische Werte, die durch Datenfelder, Variable, Zahlen oder Funktionswerte repräsentiert werden, verknüpft durch die Operatoren +, -, * und /. Durch das Setzen von Klammern kann die Reihenfolge der Abarbeitung beeinflußt werden!

Logische Bedingung:

logical expression$_1$ [{ AND | OR } log.exp$_2$] ... [{ AND | OR } log.exp$_n$]

logical expression: *[NOT] exp$_1$ [Vergleichsoperator exp$_2$]*

Vergleichsoperator: *{ = | < | > | <= | >= | <> | IN }*
Bei Operator **IN** können in einer Klammer mehrere Vergleichswerte angegeben werden:
Name in("Maier","Müller") entspricht
Name = "Maier" or Name = "Müller"

Variable

Variable stellen vor allem eine Möglichkeit dar, durch Anwender oder auch vom aufrufenden Programm her steuernd in die aktuelle Verarbeitung einzugreifen! Zusätzlich können damit zwischen den Prozeduren und logischen Bedingungen Informationen ausgetauscht werden.

Definition und Gültigkeitsbereich

Die Definition einer Variablen erfolgt entweder über die *SetVariable*-Methode des TDIWiz-Objekts, als Eingabevariable (=> Anwenderhandbuch) oder innerhalb einer Prozedur durch das Kommando

Vardef name$_1$[/type$_1$][,name$_2$[/type$_2$]] .. [,name$_n$[/type$_n$]]

name — eindeutige Bezeichnung

type — C(haracter; Standard), N(umeric), D(ate), B(oolean), I(nteger)

Gültigkeitsbereich:

- Vom aufrufenden Programm mittels *SetVariable* definierte Variable und Eingabevariable behalten bis zum Ende der Verarbeitung ihre Gültigkeit.
- In der Startprozedur definierte Variable bleiben für das aktuelle Element (einzelner Serienbrief oder gesamte Verarbeitung) erhalten.
- In sonstigen Prozeduren definierte Variable werden am Ende der Prozedur wieder frei gegeben!

ACHTUNG: Der Variablenname muss eindeutig sein! Die Definition einer bereits vorhandenen, noch gültigen Variablen führt zu einem Fehler!

Zuweisung von Werten

Die Zuweisung eines Wertes an eine Variable kann erfolgen durch

1. das aufrufende Programm durch die Methode ***SetVariable***
2. direkte Zuweisung - ***$variablename = exp***

Systemvariable

Einige Variable stehen auch ohne Definition zur Verfügung. Die Namen dieser „Systemvariablen“ dürfen für eigene Variable nicht verwendet werden:

ATT	Mail-Attachment
COUNT	Anzahl Sätze der aktuellen Datenquelle
DATE	Tagesdatum
ISMAIL	Wird von TDIWiz auf TRUE gesetzt, wenn ein E-Mail erzeugt wird
NR	Nummer des aktuellen Knotens (Wiederholungs- oder MailMerge-Knoten oder DONODES-Knoten)
NODENR	Aktueller berücksichtigter Knoten-Nummer; Knoten, die eine vorgegebene log.Bedingung nicht erfüllen, werden **nicht** mitgezählt
SAVEAS	Name unter dem das erstellte Dokument abgespeichert werden soll
SUBJECT	Betreff einer Mail-Nachricht
TEMPFOLDER	Temporärer Ordner
TEMPLATEFOLDER	Ordner der aktuellen Vorlage
TIME	Aktuelle Tageszeit

WICHTIG Falls die Textdatei ***TDISystemVariable.txt*** im Setup-Ordner existiert => für jeden Eintrag, der nicht einer Systemvariablen entspricht, wird eine Variable erzeugt!

Format: varname [Wert]

Kommandos zur Steuerung der Prozedurverarbeitung

Innerhalb von Prozeduren kann mit Hilfe von Kommandos der logische Ablauf der Verarbeitung gesteuert werden:

IF *logical condition*$_1$ **THEN**
...
[ELSEIF *logical condition*$_n$ **THEN]**
...
[ELSE]

...
END IF

Sobald eine *logical condition* erfüllt ist, werden die dahinter stehenden Befehle ausgeführt. Wenn keine Bedingung erfüllt ist, wird der ELSE-Zweig durchlaufen.

DO [UNTIL *logical condition***]**
...
LOOP [WHILE *logical condition***]**

Die zwischen DO und LOOP liegenden Befehle werden in einer Schleife immer wieder abgearbeitet, bis eine UNTIL-Bedingung erfüllt ist, eine WHILE-Bedingung nicht mehr erfüllt ist oder die Routine mit **EXIT DO** verlassen wird.

DONODES nodes [*filter***]**
...
NODESLOOP

Die zwischen DONODES und NODESLOOP liegenden Befehle werden in einer Schleife für alle der mit *nodes* definierten Knoten abgearbeitet. Durch **EXIT DONODES** kann die Routine vorzeitig verlassen wird.

SELECT CASE *exp*
CASE { $value_1$*..[,*$value_n$*]* **}**
...
[**CASE {** $value_1$*..[,*$value_n$*]* **}]**
...
[**CASE { ELSE }]**
...
END SELECT

Sobald ein CASE-Value dem Wert von *exp* entspricht, werden die dahinter stehenden Befehle ausgeführt. Wenn keine Bedingung erfüllt ist, wird der ELSE-Zweig durchlaufen.

Sonstige Kommandos

Kommandos werden als eigene Prozedur-Zeile angegeben.

Call dll,class,function **Aufruf eigene DLL**

Die DLL *dll* mit der Klasse *class* und der Funktion *function* wird aufgerufen. Bei dll kann die Endung „.dll“ weggelassen werden, wird kein Pfad angegeben, so wird der Installationsordner angenommen!

Übergabeparameter:

1. TDIWiz-Variable als Xml.XmlDocument
   <Variables>
   <Variable1>Inhalt1</Variable1>
   <Variable2>Inhalt2</Variable2>
   …….
   </Variables>

2. Schema der Eingabedaten
3. Eingabedaten als XmlDocument

Durch Änderung der Variablen bzw. durch das Hinzufügen von Variablen können Informationen an die aufrufende Prozedur übergeben werden.

Beispiel: Call Testprogram,Component,Testfunction

Createdocument template **Einfügen generiertes Dokument**

Die angegebene Vorlage wird verarbeitet, das Ergebnisdokument (ohne Kopf- und Fußteil) an der aktuellen Position eingefügt. Sonstige Textzeilen der Prozedur werden ignoriert.

Wenn die Zieladresse ein Bild-Inhaltssteuerelement ist, wird die 1. Seite des Ergebnis-dokuments (samt Kopf- und Fußteil) im JPG-Format eingefügt.

Enabled var,{True|False} **Eingabefeld (de)aktivieren**

Dieses Kommando ist nur innerhalb einer Variableneingabe-Prozedur möglich und steuert die Möglichkeiten der Eingabemaske. Es können damit Eingabefelder deaktiviert (False, 0) und wieder aktiviert (True, 1, -1) werden.

ErrMsg exp **Abbruch mit eigener Fehlermeldung**

Die TDIWiz -Verarbeitung wird mit der Fehlermeldung *exp* abgebrochen. Innerhalb einer Variableneingabe-Prozedur führt es nur zu einer Fehlermeldung.

Exit [Do] **Ausstieg aus Prozedur bzw. DO-Schleife**

VarDef name[/type] **Definition Variable**

Die Variable *name* vom Typ *type* (Std.= C(haracter)) wird definiert.

Beispiel:
vardef richtig/b
$richtig=true

ACHTUNG: Zeilen, die mit einem einfachen **Hochkomma (')** beginnen, werden als **Kommentar** interpretiert! Kommentare können aber auch am Ende einer Kommandozeile angefügt werden.

Funktionen

TDIWiz besitzt ein Vielzahl eingebauter Funktionen, die dem Administrator seine Arbeit erleichtern sollen. Jede Funktion gibt einen Wert (Zeichenkette, Zahl, logischer Wert) als Ergebnis zurück.

Einige der Funktionen existieren in ähnlicher Form auch in VBA (Visual Basic for Applications). Funktionen können in Ausdrücken innerhalb von Prozeduren und logischen Bedingungen angegeben werden.

Abs(num.exp) **Absolutwert von exp**

Beispiel: Abs(5) = Abs(-5) = 5

Avg(nodes[,exp][,filter]) **Berechnen Durchschnitt**

Der Durchschnittswert der nummerischen Textknoten *nodes* oder von *exp* wird zurückgegeben. Durch Angabe eines Filters kann eine Selektion vorgenommen werden.

Beispiel: Avg("Customers",UnitPrice*Number,Country = "Germany")

CBool(exp) **Umsetzung zu logischem Wert**

Beispiel: CBool(0) = False, CBool(num.exp) = True, wenn num.exp <> 0

CDate(exp) **Umsetzung zu Datumswert**

CDbl(exp) **Umsetzung zu Double-Wert**

WICHTIG: Bei XML-Textknoten wird als Dezimaltrennzeichen immer ein „**.**“ erwartet!

Chr(integer) **Characterdarstellung eines ASCII-Wertes**

Beispiel: Chr(10) entspricht Line Feed.

CInt(exp) **Umsetzung zu Integer-Wert**

Wenn *exp* keinen ganzzahligen Wert darstellt, wird das Ergebnis gerundet!

Count(nodes [,filter]) **Anzahl der zugeordneten XML-Knoten**

Die Anzahl der Knoten *nodes* wird zurückgegeben. Bei fehlender Angabe eines Knotens, wird die Knotenanzahl des aktuellen Wiederholungsbereichs oder der aktuellen DONODES-Schleife zurückgegeben. Durch Angabe eines Filters kann eine Selektion vorgenommen werden.
Beispiel: Count("Customers",Country="Germany") ergibt die Anzahl deutscher Kunden

DateAdd(interval,count,date) **Datumsaddition**

interval => VBA

yyyy	Jahr
m	Monat
d	Tag
ww	Woche
h	Stunde
n	Minute
s	Sekunde

DateDiff(interval,from,to) **Datumsdifferenz**

Interval => DateAdd

EndsWith(exp,string) **Abfrage, ob exp mit string endet**

Ergibt **True**, wenn *exp* mit *string* endet, sonst **False**.

Format(exp,format) **Formatierungs-Funktion**

Bei der Formatierung wird unterschieden:

1. Datumswert (*format* beginnt mit *(D)*)

   Beispiele mit dem Datumswert 28.08.2018 19:10:15

format	**Ergebnis**
(D)d oder (D)Standard	28.08.2018
(D)D	Dienstag, 28. August 2018
(D)f	Dienstag, 28. August 2018 19:10
(D)F	Dienstag, 28. August 2018 19:10:15
(D)g	28.08.2018 19:10
(D)G	28.08.2018 19:10:15
(D)m	28. August
(D)o	2018-08-28T19:10:15.0000000
(D)R	Tue, 28 Aug 2018 19:10:15 GMT
(D)s	2018-08-28T19:10:15
(D)t	19:10
(D)T	19:10:15
(D)u	2018-08-28 19:10:15Z
(D)U	Dienstag, 28. August 2018 17:10:15
(D)y	August 2018
(D)dddd, dd.MMMM yyyy	Dienstag, 28.August 2018
(D)ddd, dd.MMMM yyyy	Di., 28.August 2018
(D)dd-MM-yy	28-08-18

2. Nummerischer Wert

   Beispiele mit dem nummerischen Wert 1234567,8;

format	**Ergebnis**
N oder n oder Standard	1.234.567,80
C	€ 1.234.567,80
P	123.456.780,00 %
#,000.00	1.234.567,80
#,000.##	1.234.567,8

   Die Formatierung erfolgt im Prtogramm mit der ToString-Methode.

3. Logischer Wert (*format* beginnt mit =)

   Die Formatierungsanweisung ist in diesem Fall ausnahmsweise **ohne** Hochkommas anzugeben und muss mit „=“ beginnen. „True“- und „False“-Strings sind durch „/“ zu

trennen. Falls noch eine dritte Zeichenkette – wiederum durch „/“ getrennt – angegeben wurde, so kommt diese zum Tragen, falls es sich nicht um einen logischen Wert handelt!

Beispiel: *Format(IsCorrect, =correct/incorrect)* => in Abhängigkeit davon, ob der Textknoten *IsCorrect* den logischen Wert wahr oder falsch hat, ist das Funktionsergebnis „*correct*“ bzw. „*incorrect*“!

GetDocVar(variable) **Lesen Dokumentvariable**

Der Wert der gewünschten WORD-Dokumentvariablen wird übergeben. Existiert die Variable nicht, so wird ein Leerstring übergeben!

If(condition,true-exp[,false-exp]) **Bedingte Texteinfügung**

Mit dieser Funktion kann abhängig von einer logischen Bedingung Text erzeugt werden.

Beispiel: "Sehr geehrte" & If(Sex="M","r Herr "," Frau ") & ContactName & "!" ergibt eine Briefanrede. If-Funktionen können auch mehrstufig sein!

InStr([start,]string,search) **Suchen Zeichenkette**

Wenn die Suchzeichenkette *search* in *string* ab der Position *start* (Std.: 1) enthalten ist, gibt die Funktion als Ergebniswert die Position des 1.Zeichens zurück, sonst den Wert 0.

Beispiel: InStr("0123123","1") ergibt 2

InStrRev(string,search[,start]) **Suchen Zeichenkette**

Wie InStr, die Suche erfolgt aber in umgekehrter Richtung.

Beispiel: InStrRev("0123123","1") ergibt 5

IsNumeric(exp) **Prüfung, ob nummerisch**

Left(exp,count) **Anfang einer Zeichenkette**

Beispiel: Left(“12345“,3) liefert als Ergebnis „123“

Len(exp) **Länge einer Zeichenkette**

List(nodes[,exp][,del] [,lastdel] [,filter]) **Auflistung**

Der Inhalt der Textknoten *nodes* wird gelistet. Zur Trennung wird *del* (Std.: ", ") eingefügt, vor dem letzten Element *lastdel*. Durch Angabe eines Filters kann selektiert werden.

Beispiel: *List("Customers",Name1 & "/" & PostalCode , ", ", " und ",Country = "Germany“)*
Die einzelnen Namen und Postleitzahlen werden übergeben, wobei vor dem letzten Element statt einem Komma “ und “ zur Trennung eingefügt wird!

LTrim(exp) **Führende Leerstellen eliminieren**

Beispiel: LTrim(" abc") ergibt „abc“

Max(nodes[,exp][,filter]) **Maximalwert**

Die Maximalwert von *exp* im Bereich der selektierten Textknoten *nodes* wird zurückgegeben.

Mid(exp,start[,count]) **Teilkette**

Ab Position *start* werden *count* Zeichen von *exp* übergeben. Fehlt die Angabe von *count*, so wird der gesamte Rest übergeben:

Beispiel: Mid("123456",3,2) ergibt „34“

Min(nodes[,exp][,filter]) **Minimalwert**

Die Minimalwert von *exp* im Bereich der selektierten Textknoten *nodes* wird zurückgegeben.

Replace(exp,oldvalue,newvalue) **Zeichenkette ersetzen**

In der Zeichenkette *exp* werden alle vorkommenden *oldvalue* durch *newvalue* ersetzt.

Beispiel: Replace("123451","1","") ergibt „2345“

Right(exp,count) **Ende einer Zeichenkette**

Beispiel: Right("12345",2) ergibt „45“

Round(exp,count) **Dezimalwert runden**

count gibt die Zahl der Kommastellen an, auf die gerundet werden soll.

Beispiel: Round(123.456,2) ergibt „123,46“

RTrim(exp) **Abschließende Leerstellen eliminieren**

Beispiel: RTrim("abc ") ergibt „abc“

StartsWith(exp,string) **Abfrage, ob exp mit string beginnt**

Ergibt **True**, wenn *exp* mit *string* beginnt, sonst **False**.

Sum(nodes[,exp][,filter]) **Summe der zugeordneten Textknoten**

Die Summe von *exp* im Bereich der selektierten Textknoten *nodes* wird zurückgegeben.

Beispiel: sum("OrderDetails",(UnitPrice * Quantity) * (1 - Discount))

TMExists(name) **Prüfung, ob Textmodul existiert**

Prüft, ob der Textmodul *name* existiert

Trim(exp) **RTrim + LTrim**

Sowohl führende und als auch abschließende Leerstellen werden eliminiert.

Beispiel: Trim(" abc ") ergibt „abc“

UCase(exp) **Umsetzen auf Großbuchstaben**

Beispiel: UCase("abcd") ergibt „ABCD“

VariableExists(variable) **Prüfung, ob Dokumentvariable existiert**

Ergibt **True**, wenn die TDIWiz-Variable *variable* existiert, sonst **False**.

Weekday(date) **Tag der Woche**

Gibt den Wochentag von *date* als Ziffer zurück:

1 = Sonntag
2 = Montag
3 = Dienstag
4 = Mittwoch
5 = Donnerstag
6 = Freitag
7 = Samstag

Beispiel: weekday("19.10.2015") ergibt 2 (= Montag)

Installation

Die Installations-Zip-Files sind entweder ***TDIWizSetup Client 2.2.0.exe*** oder ***TDIWizSetup Server 2.2.0.exe***.

Wichtig: Zur Installation das Setup-Programm als Administrator aufrufen!

Voraussetzung für den Einsatz von TDIWiz ist das .NET-Framework 4.5.

Die Installation erfolgt standardmäßig im Sub-Ordner „\Sklenar\TDIWiz“ des Programmordners.

Folgende Bibliotheken/Dateien werden u.a. bei der Installation übernommen:

Systembibliotheken

TDIWiz benötigt einige Systembibliotheken (z.B. Interop-DLL für Word bei AddIn, etc.). Bei der Installation werden diese in den Installationsordner übernommen.

Text.Data.Integration.Wizard.dll + Resources-DLL

Verarbeitungskomponente; wird immer benötigt und in den Installationsordner übernommen.

TDIWizAddIn.dll samt zugehöriger Dateien (nur bei Client-Setup)

Word-Addin, das zum Aufruf des Administrations-Aufgabenbereichs dient und im Administrator-Setup enthalten ist. Nur ab MS Word 2007 und nur mit der 32-Bit Version nutzbar!

Diverses

TDIOC.dll	DLL für Word-Aktionen in der Verarbeitungskomponente
TDICONV.dll	DLL für die Konvertierung des Ergebnisses in andere Formate
TDIQR.dll	DLL für die Umsetzung in QR-Code

Die Deinstallierung von TDIWiz muss über die Systemverwaltung aufgerufen werden.

WICHTIG Zur Nutzung der Textmodul-Funktion muss im Installationsordner die Datei *TDIDBConnection.ini* mit DB-Verbindungsdaten zu einer Datenbank existieren, die die Tabelle TDITextModules enthält! Der Download *TDIWiz Testvorlagen* enthält sowohl ini-Datei (muss in den Installationsordner kopiert werden) als auch eine Access-DB mit einer Textmodul-Tabelle. Diese könnte natürlich auch aber in eine SQL-Server-DB übernommen werden.

TDIWiz Programmschnittstelle

Die eigentliche Arbeit, die Erstellung neuer Dokumente oder E-Mails, wird durch die TDIWiz-Komponente erledigt, einer .NET-DLL. Als Aufruf-Parameter werden nur die Angabe des Musterdokuments sowie der Input-Daten benötigt! Im Sonderfall, dass nur Variable zur Erstellung des neue Dokuments genutzt werden, kann auch die Angabe der Input-Daten entfallen.

Da MS Word für die Verarbeitung nicht benötigt wird, kann TDIWiz auch serverseitig genutzt und problemlos in Browser-basierte Applikationen integriert werden.

Das erstellte Dokument hat wieder das Word-Format. Durch Angabe im Feld „Speichern unter“ der Eigenschaftsmaske kann aber eine Umsetzung in andere Formate (z.B. PDF, XPS, HTML, etc.) bewirkt werden. Am Client erfolgt die Konvertierung über MS Word, am Server oder bei speziellen Formaten (z.B. PNG, JPEG) hingegen über die Fremdsoftware Gembox!

Die nachfolgenden Beispiele sind in VisualBasic dargestellt!

ACHTUNG: Zuvor ist es unbedingt notwendig einen Verweis auf die .NET-DLL ***Text.Data.Integration.Wizard.dll*** zu erstellen!

Durch

Dim tdiwiz As New TDIWiz.Component

wird ein TDIWiz-Objekt erzeugt, dessen Methoden und Attribute angesprochen werden können.

Die wichtigsten Methoden und Attribute des TDIWiz-Objekts

Beim Aufruf von TDIWiz stehen die Funktionen **CreateDocument** und **SetVariable** zur Verfügung:

tdiwiz.CreateDocument(Template[, SaveAs])

oder

tdiwiz.CreateDocument(Dokument[, SaveAs])

Die angegebene Dokumentendatei ***Template*** oder das Dokumentenobjekt ***Dokument*** wird mit den Daten befüllt, die bei der Vorlagendefinition oder direkt beim Aufruf über das Attribut **Data** (Name der Datenquelle oder XML-Daten als Zeichenkette) zugeordnet wurden.

Das Ergebnisdokument wird unter ***SaveName*** abgespeichert oder als Objekt zurückgegeben, wobei der Dateityp (docx, dotx) **nicht** geändert wird!

Durch Angabe des Attributs ***XMLFilter*** kann im Xpath-Format eine **Einschränkung** der XML-Eingabedaten und damit eine einfachere Adressierung erreicht werden (z.B.: //Customers[CustomerID='BERGS']).

Mit **Dokument** wird ein WordprocessingDocument-Objekt übergeben. Man erhält damit die Möglichkeit, auch Vorlagen, die außerhalb des Filesystems abgespeichert sind (z.B. Sharepoint-Dokumente) in Verbindung mit TDIWiz zu nutzen. Das übergebene Objekt enthält anschließend das Ergebnisdokument!

CreateDocument gibt einen Boolean-Wert zurück, der anzeigt, ob die Verarbeitung ohne Fehler durchgeführt werden konnte. Im Fehlerfall enthält das Attribut ***ErrDescription*** die Beschreibung des Fehlers.

tdiwiz.SetVariable VariableName, [VariableType], VariableValue)

Durch diese Methode können Variablenwerte an TDIWiz übergeben werden:

Als Standardtyp (=> Kap.Variable!) wird „C“ (Zeichenkette) angenommen.

ACHTUNG: Das **Setzen** der **Attribute** und **Variablen** muss **vor** dem *CreateDocument*-Aufruf erfolgen!

Beispiel für den Aufruf von TDIWiz **(VisualBasic-Code):**

```
Dim td As New TDIWiz.Component                    TDIWiz-Objekt erzeugen
td.MailMergeNode = "Customers"                    Knoten für Serienbrief
td.SetVariable "Test", "C", "Inhalt Testvariable"  Character-Variable
td.Data = "c:\tmp\tdiinp.xml"                     Eingabedaten
td.CreateDocument("c:\tmp\tditemplate.docx", "c:\tmp\tdiresult.docx")   Verarbeitung
If td.ErrDescription <> "" Then                          OK ?
   MsgBox td.ErrDescription                       Nein => Fehlermeldung
End If
Set td = Nothing                                  TDIWiz-Objekt frei geben
```

Zusätzlich stellt das TDIWiz-Objekt folgende Attribute und Methoden zur Verfügung:

SetXMLParameters(WPDocument, XMLSchema, XMLData)
oder
SetXMLParameters(Name, XMLSchema, XMLData)
Der angegebenen Vorlage (Datei oder Word-Objekt) werden Schema und Daten hinzugefügt. Damit kann die Vorlage völlig entkoppelt vom Zugriff auf entfernt liegende Daten designt und getestet werden.

ClearXMLParameters(WPDocument)
oder
ClearXMLParameters(Name)
Im Dokument abgespeicherte Daten werden wieder entfernt.

RefreshTextmodules(WPDocument)
oder
RefreshTextmodules(Name)
Im angegebenen Dokument (File oder Objekt) werden enthaltene Textmodule aktualisiert.

ErrDescription
Fehlertext

TDIDBConnection
Connection-String zur TDIDB-Datenbank (Textmodule)

Logging
Wie im Fehlerfall wird generell, die Vorlage samt den aktuellen Daten und den verwendeten Variablen als Word-Dokument im Sub-Ordner TDIWiz des Temp-Ordners (Suche auf %temp% im Explorer) unter dem Namen **TDITemplate[vorlagenname][.fehlercode].docx**

abgespeichert. Diese Datei hilft uns im Fehlerfall oder bei falschem Ergebnis bei der Analyse ohne Zugriff zum betroffenen Rechner zu benötigen!

MailMergeNode
Angabe des Knotens, auf dem ein Serienbrief oder -Mail basiert (z.B. Customers, wenn für jeden Kunden ein Brief erstellt werden soll).

MailMergeCondition
Bedingung unter der der einzelne Serienbrief erstellt werden soll.

NoForms
Eingabevariable sollen nicht über eine Bildschirmmaske abgefragt werden. Sinnvoll bei Server-Programmen um zu verhindern, dass eventuell nicht an der Programmschnittstelle übergebene Variable zu einem Problem führen!

In Verbindung mit dem Versenden von E-Mails gibt es noch die folgenden Attribute

MailAddress – entweder Feldname oder direkte Angabe der Mail-Adresse
Subject - Betreff
MailAttachments – Angabe der angehängten Dateien (durch Komma getrennt)
MailCC - CC
MailBcc - Bcc
NumberOfMails - gibt die Anzahl der erstellten E-Mail zurück
NoMailAddress – Anzahl der Serienelemente, für die keine Mail-Adresse existiert
OnlyDraft – die erstellte E-Mail wird in den Outlook-Entwurfsordner gestellt (nur Client!)
ReadReceiptRequested – Lesebestätigung anfordern
Nur beim Versenden über SMTP (Serverversion):
SMTPCredentials
SMTPSLL
SMTPPort