Der Begriff Ausdruck und die Filter

Sowohl beim Begriff Ausdruck als auch bei den Filtern werden jetzt komplexe Formulierungen unterstützt. Hier wird die „Sprache“ beschrieben, die dafür benutzt werden muss.

Die Sprache

Die Sprache benutzt Funktionen, Operatoren, Variablen und Konstanten. Die benutzten Begriffe werden über eine spezielle Funktion eingefügt und können jetzt beliebig komplex sein.

Es können mehrere Teilausdrücke in dem Ausdruck sein, jeweils mit „;“ abgeschlossen. Normalerweise reicht jedoch ein Teilausdruck.

Alle Teilausdrücke werden ausgewertet. Es wird der Wert des letzten Ausdrucks zurückgegeben.

a) Bemerkungen
... \\ eine Bemerkung ... Zeilenende
.... { eine Bemerkung } ....

b) Variablen

Um eine Variable anzulegen, weist man ihr einen Wert zu

```
a := 1;  // Dies ist eine ganze Zahl
b := 1.0; // Dies ist eine Fließkommazahl (immer mit “.” als Dezimaltrennzeichen
c := 'A text' // Dies ist eine Zeichenkette
c := "A text" // Dies auch
```

Variablen sind normalerweise local (für den Ausdruck). Mit SetGlobal kann man sie global (für den ganzen Vorgang) setzen.

Zuweisungen werden mit dem Operator ':=' durchgeführt

Die Variable 'Result' ist vordefiniert. Sie kann benutzt werden, um den Rückgabewert einer Formel explizit zu setzen.

c) Operationen

Die folgenden Operationen können benutzt werden.

+	-	*	/	% (=mod)		!=
<>	=	>=	<=	& und	| oder	xor

Zuweisungen benutzen ':='. Increment and Dekrement mit '+='uand '-='.

Es können Klammern benutzt werden'(' , ')'..

Kommas separieren Operationen auf der gleichen Ebene.
Der Ausdruck (a,b,c) hat den Wert c, aber a und b werden auch ausgeführt

Vordefinierte Funktionen:

Parameter in eckigen Klammern '[...]'sind optional.

Groß/Kleinschreibung ist wichtig

Bedingung:

if(a;b[;c]);

Führt b aus, wenn a wahr ist, ansonsten c Die Parameter werden mit ';' getrennt. Das Ergebnis ist b oder c.

Schleifen

while(a; b[; b2; b3; .... bn]);

Führt b(1..n) aus, solange a = wahr ist. Die Parameter werden mit ';' getrennt.

for(a; b; c[, c1 ... c2]);

Führt a zuerst aus. Dann wird c(1...n) ausgeführt, solange b = wahr ist. Bemerkung: die Parameter a,b,c werden mit ';' getrennt, die Liste (c1...cn) ist mit Komma getrennt.

break([a]);

Bricht eine Schleife ab ('while' oder 'for') wenn a wahr ist (wenn angegeben).

System Funktionen

NewGlobals(a [,a1...an];
SetGlobal(a [,a1...an];

Diese Funktionen deklarieren den Namen als globale Variable.
Globale Variablen gehören zur Auswertungseinheit und sind in größerem Kontext (ganzes Formular) nutzbar.

Alle Variablen müssen initialisiert werden, damit das System Weiss, von welchem Typ sie sind.

String Function

copy(a,b,c); kopiert c Zeichen des Textes a ab Position b
fillstr(a,b [,c]); setzt den Text a auf die feste Länge b. ‚c' wird zum Auffüllen benutzt..
char(a [a1 ...an]); Erzeugt einen Text mit den ANSI-Zeichen a..an.
IntToStr(a); Erzeugt einen Text aus der Ganzzahl a.

FloatToStr(a) wandelt einen Fließkommawert in einen Text
FloatToCurr(a [,b] [,c]) Wandelt einen Fließkommawert in eine Währungsdarstellung mit b Nachkommastellen und einer Länge von c..
ShowMessage (a [,a1 ... an]) zeigt eine Meldung mit den Texten a + a1 ... + an
Input(a[,b] [,c]) Zeigt eine Eingabebox an. a ist der Fragetext, b die Überschrift und c der Vorgabewert.
RememberBox(a,b[,c] ,d) Fragt den Benutzer **einmal** nach einer Eingabe. 'a' ist der Name des Wertes, b die Frage, c (optional) ist die Überschrift und d die Vorbesetzung. Die interne Liste wird durch die Funktion *WPEval.Init gelöscht*.

beginswith(a,b) wahr, wenn a mit b anfängt
beginswithNC(a,b) wahr, wenn a mit b anfängt (ohne Groß/Kleinschreibung)
includes(a,b) wahr, wenn b in a vorkommt (ohne Groß/Kleinschreibung)
includesNC(a,b) wahr, wenn b in a vorkommt anfängt (ohne Groß/Kleinschreibung)

in(e, c1[..cn] wahr wenn e in der Liste der c1..cn vorkommt
inNC(e, c1[..cn] wahr wenn e in der Liste der c1..cn vorkommt (ohne Groß/Kleinschreibung)

exchg(a,i1,o1[i2,o2..iN,oN]) tauscht die jeweils iN gegen oN aus, wenn a = iN, ansonsten ist das Ergebnis a

LookUp(a,d,c1,v1[,cN,vN>]) gibt den zu a passenden Wert aus der Liste (c,v) zurück, ansonsten d

Numerische Funktionen

RoundDigits(a,b); Runded a auf b Stellen

Außerdem gibt es die mathematischen Funktionen: **Sin**(a), **Cos**(a), **Min**(a,b), **Max**(a,b), **Abs**(a), **Sqr**(a)

Datums Funktionen

DateValue(a); Verwandelt einen Datumstext in die numerische interne Form

DateFormat (a,b); gibt ein Datum im internen Format als formatierten Text aus

Datecalc(a,b) errechnet ein neues Datum aus a mit Hilfe des Ausdrucks in b

b <op>:<einheit>,<anzahl>[;<op>:<einheit>,<anzahl>…..]

<op>

Plus
Minus
FirstOf //erster Tag
LastOf //letzter Tag
FirstDOWOf //erster Wochentag von
LastDOWOf //letzter Wochentag von
PriorDOW //vorheriger Wochentag
NextDOW //nächster Wochentag
Today //heute/jetzt

<einheit>

Year
6Months
4Months
3Months
Month
Week
Day
su,mo,tu,we,th,fr,sa
h,m,s

Beispiel:

Datecalc(„13.1.2015“,“FirstOF:month,1“) ergibt 1.1.2015
Datecalc(„13.1.2015“,“Plus:day,3“) ergibt 16.1.2015

Begriffe

Token (token,format)

Token Definition des Tokens, wird vom Dialog automatisch über Feld einfügen erzeugt

format
‚S' Zeichenkette
‚I' ganze Zahl
‚F' Fliesskommazahl
‚B' logisch (wahr,falsch)
‚D' Datum/Zeit

wenn es fehlt, dann ‚S' Zeichenkette

Alte Funktionen, bitte nur die Neuen benutzen, die Alten sind aus Kompatibilitäts-Gründen noch da.

~~**T** (id,p1,p2,……) evaluates Token~~

~~Id Token ID, can include expression for redirection like "Grp:Bezeichnung"~~
~~P1 first Parameter~~
~~P2..pn next parameters or format if begins with @ (does not work in every situation)~~

~~**TN** (id,p1,p2,……) evaluates Token as numeric value~~

~~Id Token ID, can include expression for redirection like "Grp:Bezeichnung"~~
~~P1..PN Parameter~~

~~**TF** (id,p1,p2,……,fo) evaluates Token as numeric value~~

~~Id Token ID, can include expression for redirection like "Grp:Bezeichnung"~~
~~P1..pn parameters~~
~~fo format for expression~~