

# Apparo Fast Edit

## Einsatz und Verwendung von Variablen



Inhaltsverzeichnis

<b>1</b>	<b><i>Definition</i></b>	<b>3</b>
<b>2</b>	<b><i>Einsatz von Variablen im Designer</i></b>	<b>5</b>
2.1	<b>Variablen in Lookup-Definitionen</b>	<b>5</b>
2.2	<b>Variablen in Überschriften, Hinweistexten, im Kopf- und Fußbereich</b>	<b>5</b>
2.3	<b>Variablen in Filterdefinitionen</b>	<b>5</b>
2.4	<b>Variablen in Variablen und bei der Überprüfung von Dateneingaben</b>	<b>6</b>
<b>3</b>	<b><i>Interne Variablen</i></b>	<b>9</b>
<b>4</b>	<b><i>Berichts-Variablen</i></b>	<b>10</b>
<b>5</b>	<b><i>SQL Variablen</i></b>	<b>11</b>
<b>6</b>	<b><i>Script-Variablen</i></b>	<b>13</b>
6.1	<b>Script Variablen in Datenbankverbindungen</b>	<b>13</b>
<b>7</b>	<b><i>Widget Referenz Variablen</i></b>	<b>14</b>
<b>8</b>	<b><i>Betriebssystemvariablen</i></b>	<b>15</b>
<b>9</b>	<b><i>Debuggen von Variablen</i></b>	<b>16</b>
9.1	<b>Definition</b>	<b>16</b>
9.2	<b>Variablenausgabe zu Debug-Zwecken</b>	<b>16</b>
9.3	<b>Debuggen von Script-Variablen</b>	<b>17</b>
9.4	<b>Debuggen von SQL-Variablen</b>	<b>19</b>

## 1 Definition

Variablen sind Platzhalter.

Sie geben einen oder mehrere Werte zurück.

Sie können fixe Werte enthalten, oder Berechnungen bzw. Abfragen dynamisch ausführen.

Zur Laufzeit, d.h. bei der Ausführung der Business Cases, wird der Wert der Variable berechnet und am jeweiligen Einsatzort ausgegeben.

**Syntax:** <%Variablenname%>

Variablen-Bezeichnungen sind case-sensitiv, d.h. Groß- und Kleinschreibung müssen beachtet werden.



Grundsätzlich unterschieden werden anwenderdefinierte Variablen und interne Variablen.

**Apparo Fast Edit unterstützt 6 verschiedene Variablenarten:**

- Interne (vordefinierte) Variablen
- SQL-Variablen
- Berichtsvariablen
- Script-Variablen
- Widget Referenz Variablen
- Betriebssystemvariablen

Variablen können in praktisch allen Einstellungen und in anderen Variablen verwendet werden

In Business Cases können Sie diese Variablen-Typen erstellen:

Wählen Sie den Typ der neuen Variable ✕

	Variablen Typ	Variablen Typ Beschreibung
	▶ <b>Script Variable</b>	Sie können JavaScript verwenden, um komplexere Berechnungen anzustellen, eigene Java Klassen aufzurufen, SQL Befehle auszuführen uvm. Das Ergebnis kann in Fast Edit in jeder anderen Variable verwendet werden.
	▶ <b>Report Variable</b>	Wenn Sie mit Berichtsmanagement einen Cognos Bericht und einen Business Case verlinken, dann finden Sie eine zusätzliche Spalte "FE_test". Damit können Sie den Inhalt der Variablen "test" definieren. Dieser Inhalt wird zum Business Case über die aufrufende URL transportiert. Um den Inhalt nutzen zu können, muss dort die Reportvariable "test" erstellt worden sein. Die Variable, bzw deren Inhalt können Sie zur Ausgabe im Business Case oder zur Weiterverarbeitung in allen Variablen verwenden
	▶ <b>SQL Variable (für alle Tabellen)</b>	SQL Variable zum Ausführen von Kommandos in allen Tabellen. Bei jeder Verwendung der Variable wird das dazugehörige SQL ausgeführt. Die Variable enthält den Inhalt der ersten Zeile, erste Spalte (je nach SQL-Kommando)
	▶ <b>SQL Variable (nur für die Zieltabelle)</b>	SQL Variable für die Business Case Zieltabelle. Alle im Business Case vorhandenen Filter werden berücksichtigt

✕ ABBRECHEN

## 2 Einsatz von Variablen im Designer

Viele Widget-Einstellungen können mit Variablen dynamisch gestaltet werden.

### Beispiele:

#### 2.1 Variablen in Lookup-Definitionen

Widget Typ	Zuordnung & Datenwerte	Widget-Verhalten	Visuelles	Visuelle Hilfetexte	Datenausgabeformat
'Sortieren nach'	<input type="text"/>				
Spaltenname	NAME_<%LANGUAGE%>	<input type="checkbox"/>	* Aktiviere Ausdrücke	<input type="checkbox"/>	
Vorgabewert	<%VARIABLE%>	für	alle Anwender	<input type="checkbox"/>	

Die zugeordnete Datenbankspalte setzt sich zusammen aus ‚Name\_‘ und dem Rückgabewert der verwendeten Sprache. Deutschen Anwendern wären der Spalte NAME\_DE zugeordnet und englische Anwender entsprechend der Spalte NAME\_EN

#### 2.2 Variablen in Überschriften, Hinweistexten, im Kopf- und Fußbereich

Spaltenbezeichner	
Sprache	Spaltenbezeichner
German	<%LABEL_DE%>
English	<%LABEL_EN%>

In diesem Beispiel wird die Überschrift der Spalte durch Variablen ausgegeben

#### 2.3 Variablen in Filterdefinitionen

Widget Typ	Zuordnung & Datenwerte	Widget-Verhalten	Lookup & Dropdown & Multiselect	Visuelles	Visuelle Hilfetexte	Datenausgabeformat
Datenbankverbindung	AFE3 Demo					
Lookup Tabelle	PRODUCT_PARTS					
Lookup-Tabellen Spalte für den Vergleich	PRODUCT_ID	<input type="checkbox"/>	Lesender Ausdruck			
Lookup-Tabellen Spalte für die Ausgabe	PRODUCT_NAME_<%LANGUAGE%>	<input type="checkbox"/>	Lesender Ausdruck			
Multiwert 'Sortiert nach'	Aufsteigend					
Lookup-Tabelle Sortierspalte	Verwenden derselben Spalte wie für die Anzeige des Wertes					
SQL 'where' Bedingung	<%FILTER_VAR%>					

Dynamischer SQL-Filter

## 2.4 Variablen in Variablen und bei der Überprüfung von Dateneingaben

### Beispiel für den Einsatz von Variablen bei der Überprüfung von Dateneingaben:

#### Überprüfung der Dateneingaben

Sie können hier mit Javascript eine Überprüfungsroutine definieren, die alle Werte der aktuellen Datenzeile überprüfen kann und bei Bedarf eine Fehlermeldung ausgibt. Sie finden hilfreiche Beispiele, wenn Sie das ?-Icon auf der rechten Seite auswählen.

```

var a = <%WIDGETWERT1%>;
var b = <%WIDGETWERT2%>;
var c = <%WIDGETWERT3%>;
var d = <%SQL_VARIABLE1%>;

// Leeres Ergebnis definieren, d.h. die Eingabe ist korrekt
var result = "";

if (c != 'A' && a > b) {
  if (<%LANGUAGE%> == 'en') {
    result = 'Product data is invalid';
  } else {
    result = 'Produktdaten sind falsch';
  }
}
if (d == 1234) {
  if (<%LANGUAGE%> == 'en') {
    result = 'Calculation is wrong';
  } else {
    result = 'Berechnung ist falsch';
  }
}
}
// Rückgabe des Ergebnisses / Fehlermeldung
result;

```

In diesem Beispiel wurden Widget Referenz Variablen, SQL-Variablen und interne Variablen verwendet.

### Beispiel für den Einsatz in anderen Variablen

#### Script Variablen:

Variablen Name

Variablenwert     Ausgabeformat

Script-Körper

Script-Sprache : javascript

```

var result = <%SQL_COUNT_VAR%> / 100;
result;

```

In diesem Beispiel wird in einer JavaScript Variable eine SQL-Variable verwendet.

### SQL Variablen:

Variablen Name	<input type="text" value="SQL_var1"/>	*		
<table border="1"> <tr> <td>Variablenwert</td> <td>Ausgabeformat</td> </tr> </table>			Variablenwert	Ausgabeformat
Variablenwert	Ausgabeformat			
Datenbankverbindung : AFE3 Demo				
<table border="1"> <tr> <td>SQL Ausdruck</td> </tr> <tr> <td> <pre>select PRODUCT_COLOR_NAME from SAMPLE_CAR_COLOR where PRODUCT_COLOR_ID = &lt;%PRODUCT_ID%&gt;</pre> </td> </tr> </table>			SQL Ausdruck	<pre>select PRODUCT_COLOR_NAME from SAMPLE_CAR_COLOR where PRODUCT_COLOR_ID = &lt;%PRODUCT_ID%&gt;</pre>
SQL Ausdruck				
<pre>select PRODUCT_COLOR_NAME from SAMPLE_CAR_COLOR where PRODUCT_COLOR_ID = &lt;%PRODUCT_ID%&gt;</pre>				

Widget-Referenz Variablen werden häufig in SQL-Variablen verwendet. JavaScript-Variablen sind auch möglich.

### Beispiel für den Einsatz von Variablen als dynamisches Intervall:

Bei einem Widget vom Typ „Eingabefeld“ kann der erlaubte Wertebereich eingeschränkt werden:

<b>Überprüfung der Datenqualität</b>	
Anwenderdefinierte Java 6 Validator-Klasse	<input type="text" value="----"/> ▼
Intervall	Minimum erlaubt: <input type="text" value="&lt;%VAR_MIN_CALC%&gt;"/>
	Maximum erlaubt: <input type="text" value="&lt;%VAR_MAX_CALC%&gt;"/>

Beispiel für dynamische Intervalle, die die Werteeingabe anhand von Berechnungen einschränken.

### **Dynamische Werte werden via Variable realisiert:**

Unsere SQL Variable ist vom Typ eine SQL-Variable (nur für die Zieltabelle). Dieses hat den Vorteil, dass automatisch alle anwendergruppenabhängigen Filter verwendet werden, die in der Funktion „Datenausgabe filtern“ definiert wurden.

Die aktuelle Zeile wird durch den Wert im Widget PRODUCT\_ID identifiziert. D.h. PRODUCT\_ID ist ein Primärschlüssel.

Folgender Beispiel-SQL für die SQL-Variable wäre möglich:

```
SELECT min_value FROM targettable WHERE product_id = <%PRODUCT_ID%>
```

In diesem Fall bezieht sich <%PRODUCT\_ID%> auf das Widget PRODUCT\_ID im Business Case und liefert den aktuellen Wert.

Das SELECT liefert damit den Wert min\_value von der aktuellen Zeile und speichert ihn in der neuen SQL Variable „minimum“.

Das SQL wird jedes Mal ausgeführt, wenn auf die Variable „VAR\_MIN\_CALC“ zugegriffen wird.



### 3 Interne Variablen

Folgende Variable sind bereits vorab definiert und können sofort verwendet werden:

Variablen Name	Variablen Beschreibung
<%AFE_HOME_DIR%>	Dateiverzeichnispfad der Apparo Fast Edit-Installation
<%AFE_BC_NAME%>	Name des aktuell geöffneten Business Cases
<%AFE_CLIENT_ID%>	Enthält die Mandanten ID des aktuellen Mandantens
<%AFE_BC_ID%>	Kurzname (ID) des aktuell geöffneten Business Cases
<%SERVER_NAME%>	Name des Servers, auf dem Apparo Fast Edit läuft
<%USER_NAME%>	Name des eingeloggten Anwenders
<%USER_LOGIN%>	Eindeutiger Loginname des Anwenders
<%LANGUAGE%>	Kürzel der Sprache, in der Benutzeroberfläche angezeigt wird
<%CURRENT_DATE%>	Aktuelles Datum und Uhrzeit
<%DATE%>	Aktuelles Datum
<%TIMESTAMP%>	Aktuelles Datum und Uhrzeit
<%TIME_MS%>	Die Anzahl der Millisekunden seit dem 1.1.1970 (UNIX timestamp)
<%PRIMARY_KEY%>	Der Primärschlüssel der aktuellen Zeile
<%PRIMARY_KEYS%>	Kommagetrennte Liste der Primärschlüssel
<%ROW_EDIT_TYPE%>	Art der Datenänderung. Ausgabe ist vom Typ String
<%SELECTED_ROWS_COUNT%>	Anwendungsbeispiel: "Wollen Sie wirklich X Zeilen löschen?"
<%ROWS%>	Anzahl der aktuell sichtbaren Zeilen
<%BULK_UPDATED_ROWS%>	Anzahl der mit Massenupdate geänderten Zeilen
<%INSERTED_ROWS%>	Anzahl aller eingefügten Zeilen während Excel-Import
<%UPDATED_ROWS%>	Anzahl aller aktualisierten Zeilen beim Excel-Import
<%IMPORTED_ROWS%>	Anzahl aller Importierten Zeilen während des Excel-Imports
<%IMPORTED_FILE_NAME%>	Name der derzeit Importierten Datei (Excel-Import)
<%EXCEL_IMPORT_ID%>	Universell eindeutige Kennung (UUID) des Typs String jedes Excel-Import
<%EXPECTED_COLUMNS%>	Liste der erwarteten Spalten für Excel-Import
<%LINE%>	Fehlerzeile beim Excel Import, Beispiel: "Importfehler in Zeile <%LINE%>."
<%SAME_PK_ROWS%>	Hilfreich beim Anzeigen von Fehlermeldungen z.B. "Es gibt bereits eine Zeile mit dem gleichen Primärschlüssel. Anzahl <%SAME_PK_ROWS%>
<%UPLOADED_FILE_NAME%>	Name der hochgeladenen Datei (Datei-Upload/Download-Widget)
<%DELETED_FILE_NAME%>	Name der gelöschten Datei (Datei-Upload/Download-Widget)
<%RETURN_VALUE%>	In dieser Variable wird der Rückgabewert des Skriptes/Funktion gespeichert..

Wenn der Business Case Suchfelder verwendet, so werden für jedes Suchfeld automatisch die passenden Variablen definiert:

<%SEARCH_KEY_COLOR%>	Key-Wert für die Suche im Lookup Widget, zugeordnet zur Spalte 'COLOR'
<%SEARCH_VALUE_COLOR%>	Wert des Lookup-Such Widgets der Spalte 'COLOR' zugeordnet

#### 4 Berichts-Variablen

Wenn Sie mit Berichtsmanagement einen Cognos Bericht und einen Business Case verlinken, dann finden Sie eine zusätzliche Spalte "FE\_test". Damit können Sie den Inhalt der Variablen 'test' definieren.

Dieser Inhalt wird zum Business Case über die aufrufende URL transportiert.

Um den Inhalt nutzen zu können, muss dort die Reportvariable 'test' erstellt worden sein.

Die Variable, bzw. deren Inhalt können Sie zur Ausgabe im Business Case oder zur Weiterverarbeitung in allen Variablen verwenden.

**Variable für Business Case**

Variablen Name  \*

**Variablenwert** | **Ausgabeformat**

Vorgabewert

Der Vorgabewert wird nur verwendet, wenn der Bericht keinen Wert für diese Variable liefert.

Variablen Name  \*

**Variablenwert** | **Ausgabeformat**

Ausgabebetyp

Unter Ausgabeformat können Sie den Datentyp festlegen.

#### Beispiel für den Aufruf eines Business Cases aus einem Cognos Bericht (URL):

```
/ibmcognos/cgi-bin/cognos.cgi?b_action=xts.run&m=portal/bridge.xts&c_env=/portal/env.xml&c_mode=post&c_cmd=/KFE/pages/userInterface.jsf?bc=BCNAME&FE_Berichts_Var1=1234&backLink=%2Fcontent%2Ffolder%5B%40name%3D%27Apparo+Fast+Edit+Demonstration%27%5D
```

In der URL hat die Berichts-Variable Berichts\_Var1 den Wert 1234.

Diese Berichts-Variable kann nun im Business Case verwendet werden oder weiterverarbeitet werden.

## 5 SQL Variablen

Es gibt 2 verschiedene Arten von SQL Variablen:

- **SQL-Variable (für alle Tabellen)**

SQL Variable zum Ausführen von Kommandos in allen Tabellen. Bei jeder Verwendung der Variable wird das dazugehörige SQL ausgeführt. Die Variable enthält den Inhalt der ersten Zeile, erste Spalte (je nach SQL-Kommando)

- **SQL-Variable (nur für die Zieltabelle)**

SQL Variable für die Business Case Zieltabelle. Alle im Business Case vorhandenen Filter werden berücksichtigt

**Beispiel:**

### Variable für Business Case

Variablen Name  \*

Variablenwert
Ausgabeformat

Datenbankverbindung : AFE3 Demo

SQL Ausdruck

```
select NAME from SAMPLE.CARS where |<%ID%>
```

+
-
\*
/
&
|
^
||
=
>
<
>=
<=
(
)
'

✓ OK
↩ ABBRECHEN

Der Hauptunterschied ist, dass eine **SQL-Variable (nur für die Zieltabelle)** automatisch:

- die Filter des Business Cases (siehe „Datenausgabe filtern“)
- alle gruppenabhängige Filter (siehe „Datenausgabe filtern“)
- Alle Filter-Widget-abhängigen Filter

verwendet.

Daher muss das SQL der Variable ebenfalls die Zieltabelle verwenden, damit die Filter auch die gleichen Spaltennamen vorfinden.

**SQL-Variablen (nur für die Zieltabelle)** sind sehr hilfreich für Kalkulationen, die sich auf die Zieltabelle beziehen – z. B. Summe aller verkauften Produkte – da alle verwendeten Filter automatisch berücksichtigt werden.

Da sich beim Einsatz von Filter-Widgets die Ausgabe verändert, muss normalerweise diese dynamische Filtereinschränkung ebenfalls berücksichtigt werden.

Bei einer **SQL-Variablen (nur für die Zieltabelle)** ist dies im Gegensatz zu einer **SQL-Variablen (für alle Tabellen) automatisch** der Fall.

**Eine SQL-Variable wird immer dann ausgeführt, wenn Sie verwendet wird.**

**Als Ergebnis wird der 1. Ergebniswert verwendet.**

## 6 Script-Variablen

Dieser Variablentyp hat normalerweise keine Verbindung zu einer Datenbank.  
Die Logik wird mit **JavaScript** definiert.

The screenshot shows a configuration window for a script variable. At the top, the 'Variablen Name' field is set to 'script\_var1'. Below this are two tabs: 'Variablenwert' and 'Ausgabeformat'. The 'Script-Körper' section is active, showing 'Script-Sprache : javascript'. The script body contains the following code:

```
var result = '<%=USER_NAME%>' + 'postfix';
result; //Rückgabe des Ergebnisses an die Variable
```

Der berechnete Wert wird in diesem Beispiel durch ‚result;‘ an die Variable übergeben

Widget Referenz Variablen, SQL-Variablen, Berichts-Variablen und Interne Variablen können verwendet werden.

Damit können Logik und SQL-Abfragen kombiniert werden.

### 6.1 Script Variablen in Datenbankverbindungen

Script-Variablen können auch in einer Datenbankverbindung verwendet werden.  
So kann z.B. der Login-Username berechnet werden.

Falls in einer Datenbankverbindung Script-Variablen verwendet werden, so wird automatisch das Verbindungs-Pooling abgeschaltet, da dann die Verbindungsdaten nicht mehr einheitlich sind.

Script-Variablen, die in einer Datenbankverbindung definiert wurden, können in allen Business Cases verwendet werden, die diese Datenbankverbindung verwenden.

## 7 Widget Referenz Variablen

Sie enthalten den Wert eines Widgets in der entsprechenden Zeile.  
Durch den Zeilenbezug können diese Variablen nur dort eingesetzt werden, wo dieser gegeben ist.  
So können z.B. nicht eingesetzt werden: im Kalkulationsbereich oder im Kopf- und Fußbereich.

**Syntax:** <%SPALTEN\_NAME%>

Edit-Widgets								
<input type="checkbox"/>	Spalte	Spaltenname	Widget	Titel	PK	RO	H	NN
<input type="checkbox"/>	1		▶ Platzhalter & Titel	▶			<input type="checkbox"/>	
<input type="checkbox"/>	2	▶ OFFICE_ID	▶ Eingabefeld	▶ Filiale	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
<input type="checkbox"/>	3	▶ PRODUCT_LINE_ID	▶ Lookup Auswahlfeld (für alle Tabellen)	▶ Produktlinie	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
<input type="checkbox"/>	4	▶ PRODUCT_ID	▶ Lookup Auswahlfeld (für alle Tabellen)	▶ Produkt	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
<input type="checkbox"/>	5	▶ MONTH_ID	▶ Lookup Auswahlfeld (für alle Tabellen)	▶ Monat	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
<input type="checkbox"/>	6	▶ SALES	▶ Eingabefeld	▶ Verkäufe	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
<input type="checkbox"/>	7	▶ STATUS_ID	▶ Lookup Auswahlfeld (für alle Tabellen)	▶ Mein Status	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
<input type="checkbox"/>	8	▶ STATE_REVISION_ID	▶ Lookup Auswahlfeld (für alle Tabellen)	▶ Revision Status	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>

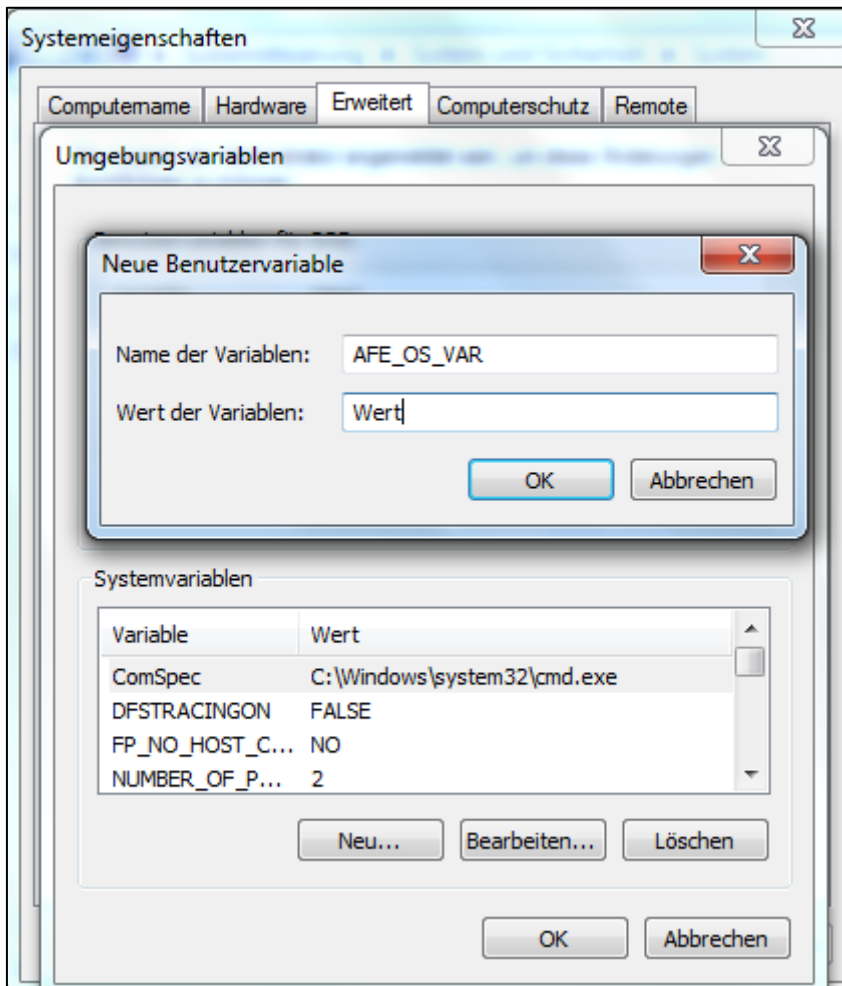
Beispiel für eine Widget Referenz Variablen: <%OFFICE\_ID%>

## 8 Betriebssystemvariablen

Diese Variablen werden im Betriebssystem definiert und können in allen Business Cases verwendet werden.

**Syntax:** <%AFE\_BEZEICHNER%>

Beispiel für die Definition unter Windows:



## 9 Debuggen von Variablen

Insbesondere bei der Verwendung von Script- und SQL-Variablen treten mit zunehmender Komplexität häufiger logische und/oder syntaktische Fehler auf. Dieses Kapitel gibt Ihnen einen Einblick, wie Sie die Fehler erkennen oder finden können.

### 9.1 Definition

Mit Debuggen wird in der Informatik die Suche und Beseitigung von Fehlern bezeichnet.

### 9.2 Variablenausgabe zu Debug-Zwecken

Wenn Variablen bei Ihrer Verwendung nicht ausgegeben werden, z.B. wenn Sie in anderen Variablen eingesetzt oder für die Parameterübergabe in Script- oder DB-Prozeduraufrufen verwendet werden, ist es häufig schwierig Fehler in Ihnen zu entdecken.

Es empfiehlt sich daher diese Variablen direkt im Business Case, zumindest in der Entwicklungsphase, auszugeben.

#### Variablenausgabe im Kopfbereich

Variablen ohne Zeilenbezug können Sie direkt im Kopfbereich ausgeben:

**Apparo Fast Edit**

**++DEBUG++**  
**BC\_NAME: SB\_VARIABLE\_SQL\_TBC**  
**Sprache: de**

Count:32  
 Count\_target:1

Search\_key\_sql:102  
 Search\_key\_sql\_target:102  
 Search\_value\_sql:TEST102  
 Search\_value\_sql\_target:TEST102

SUCHEN
FILTER ZURÜCKSETZEN

ÄNDERN

	SALES_ID *	SALES_NAME	SALES_DESCR
<input type="checkbox"/>	102,00	TEST102	102



## Variablenausgabe im Edit-Bereich

Variablen mit Zeilenbezug, das sind in der Regel Widget Referenz Variablen oder Script- und SQL-Variablen, die Widget Referenz Variablen enthalten, können nicht im Kopfbereich zu Debug-Zwecken ausgegeben werden, da Widget Referenz Variablen immer den Inhalt des Widgets enthalten, in deren Zeile sie verwendet werden.

Apparo Fast Edit					
SALES_ID *	SALES_NAME	SALES_DESCR	++DEBUG++		
102,00	TEST102	102	VAR ID*100: 10200.0	VAR ID+NAME: 10200.0TEST102	RandomColor: HEADER
103,00	103	103	VAR ID*100: 10300.0	VAR ID+NAME: 10300.0103	RandomColor: HEADER
104,00	104	104	VAR ID*100: 10400.0	VAR ID+NAME: 10400.0104	RandomColor: HEADER

### 9.3 Debuggen von Script-Variablen

Neben der inhaltlichen Prüfung durch Ausgabe im Kopfbereich, gibt es noch weitere Möglichkeiten zur Prüfung auf Fehler:

#### Syntax-Prüfung bei Script-Variablen

JavaScript-Variablen bieten die Möglichkeit die enthaltenen JavaScript-Befehle auf Syntax-Fehler zu überprüfen. Bei Fehlern wird die entsprechende Zeile markiert und eine Beschreibung des Fehlers wird unterhalb des Script-Bereichs angezeigt.

Im folgenden Beispiel fehlt das Semikolon am Ende der ersten Zeile:

Script-Definition

Script-Sprache : javascript

```

1 var x = <%SALES_ID%> * 100
2 //Rückgabegabe des berechneten Wertes
3 x;
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20

```

**SYNTAX-ÜBERPRÜFUNG**

**Prüfungsfehler:**

Line: 1 - Expected ';' and instead saw 'x'.  
 » /\*global afe\*/ var x = 123 \* 100

Ist die Syntax ohne Fehler, gibt die Fehlerprüfung eine entsprechende Meldung aus:



### Fehlerausgabe im Log

Fehler in JavaScript-Variablen werden auch immer im AFE Log ausgegeben.  
Sie finden die Datei „afe.log“ im Verzeichnis [APPARO-HOME]/FastEdit/logs/

Bei der Verwendung unserer Beispielvariablen im Kopfbereich würden Sie den folgenden Fehler im Log finden:

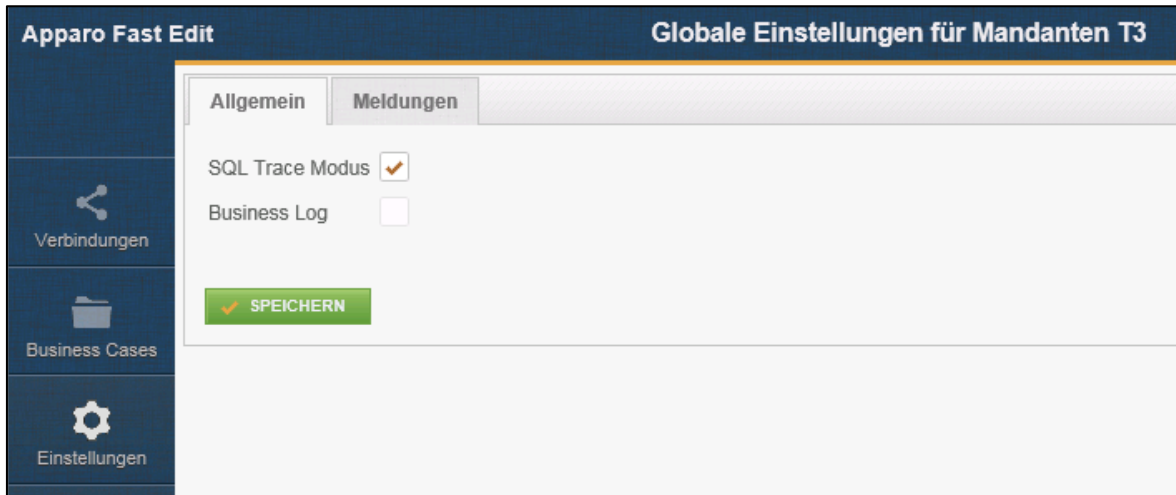
```
2015-05-07 14:02:03,547 [ajp-apr-9800-exec-8] WARN ScriptVariableResolver - The exception is:
sun.org.mozilla.javascript.internal.EvaluatorException: syntax error (<Unknown source>#1) in <Unknown
source> at line number 1
2015-05-07 14:02:03,547 [ajp-apr-9800-exec-8] WARN ScriptVariableResolver - Error in script variable
named '100'
2015-05-07 14:02:03,547 [ajp-apr-9800-exec-8] WARN ScriptVariableResolver - The script body is:
var x = * 100;
//Rückgabegabe des berechneten Wertes
x;
```

*Die Widget Referenz Variable konnte hier wegen des fehlenden Zeilenbezuges nicht aufgelöst werden und erzeugt so eine fehlerhafte Formel.*

## 9.4 Debuggen von SQL-Variablen

Fehler in SQL-Variablen werden bei der Ausgabe im Kopf- oder Edit-Bereich angezeigt, sowie in der Datei afe.log.

Eine weitere und bessere Möglichkeit Probleme mit SQL-Variablen zu identifizieren bietet das SQL-Trace Log, welches im Designer aktiviert werden muss:



Sie finden die Datei „afeSQL.csv“ im Verzeichnis [APPARO-HOME]/FastEdit/logs/

Im Trace werden alle SQL-Anfragen in Tabellenform mit den folgenden Informationen gespeichert:

*Zeitstempel, Mandant, Business Case, Anwendername, Ausführungszeitdauer, SQL Kommando*

07.05.2015 15:26	T3	sb_variable_sql_TBC1	de	0:00:00.000	Select count(SALES_ID) from TESTING.SAMPLE_SALES
---------------------	----	----------------------	----	-------------	--

Typische Fehlerausgaben sind:

**Select cuont (SALES\_ID) from TESTING.SAMPLE\_SALES ORA-00904: "CUONT": invalid identifier**

Oder

**Select SALES\_DESCR from ".SAMPLE\_SALES where SALES\_ID = " ORA-00903: invalid table name**