Dynamische Suchhilfe

Suchhilfen geben immer wieder Anlass zu viel Diskussionen. Es gibt viele Wege zum Ziel; häufig sind diese jedoch umständlich oder zumindest nicht ideal. Für den Fall, dass du mal eine Suchhilfe brauchst, bei der die Felder erst zur Laufzeit bekannt sind, kannst du folgendes Vorgehen nutzen:

Suchhilfe-Exit Funktionsbaustein kopieren

Als erstes benötigst du einen Suchhilfe-Exit. Dieser Exit besteht aus einem Funktionsbaustein mit definierter Schnittstelle. Am besten kopierst du den Funktionsbaustein F4IF_SHLP_EXIT_EXAMPLE auf z. B. Z_F4IF_SHLP_EXIT_NEU1.

Hierfür benötigst du eine eigene Funktionsgruppe. Diese muss vor dem Kopieren vorhanden sein. In diesem Beispiel heißt der Funktionsbaustein z_f4ifshlp_exit_4_enno.

Funktionsbaustein Suchhilfe-Exit

```
FUNCTION z_f4ifshlp_exit_4_enno.
*"----------------------------------------------------------------------
*"*"Lokale Schnittstelle:
*" TABLES
*" SHLP_TAB TYPE SHLP_DESCT
*" RECORD_TAB STRUCTURE SEAHLPRES
*" CHANGING
*" REFERENCE(SHLP) TYPE SHLP_DESCR
*" REFERENCE(CALLCONTROL) LIKE DDSHF4CTRL STRUCTURE DDSHF4CTRL
*"----------------------------------------------------------------------

DATA: rc TYPE sy-subrc.

IF callcontrol-step = 'SELONE'.
EXIT.
ENDIF.

IF callcontrol-step = 'PRESEL'.
EXIT.
ENDIF.

*"----------------------------------------------------------------------
* STEP SELECT (Select values)
*"----------------------------------------------------------------------
```

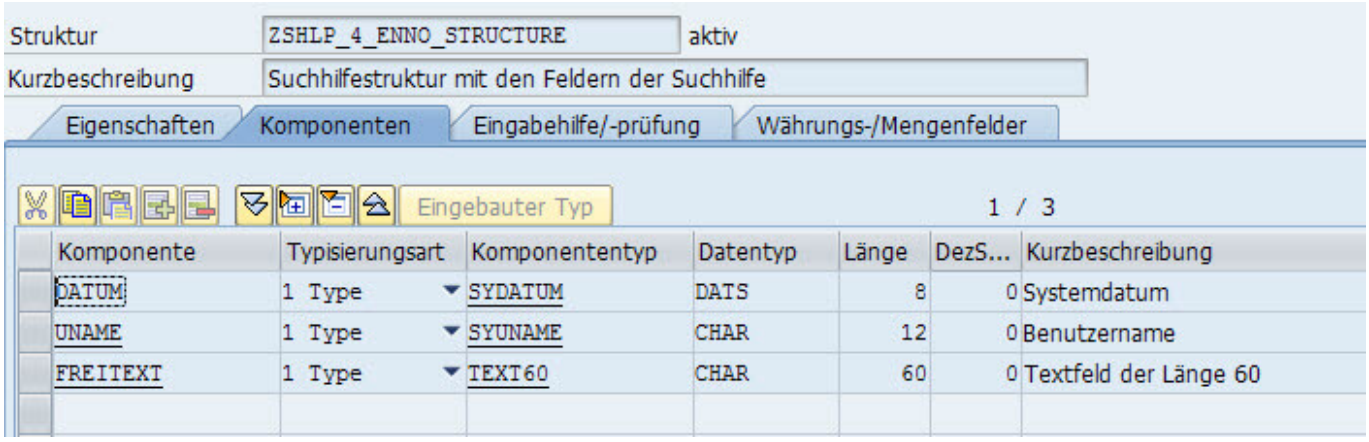
```
* This step may be used to overtake the data selection completely.
* To skip the standard seletion, you should return 'DISP' as following
* step in CALLCONTROL-STEP.
* Normally RECORD_TAB should be filled after this step.
* Standard function module F4UT_RESULTS_MAP may be very helpfull in this
* step.
IF callcontrol-step = 'SELECT'.
PERFORM shlp_4_enno USING shlp_tab[]
CHANGING callcontrol
shlp
record_tab[]
rc.
IF rc = 0.
callcontrol-step = 'DISP'.
ELSE.
callcontrol-step = 'EXIT'.
ENDIF.
EXIT. "Don't process STEP DISP additionally in this call.
ENDIF.

IF callcontrol-step = 'DISP'.
EXIT.
ENDIF.

ENDFUNCTION.
```

Unterroutine

In dieser Unterroutine wird die Ergebnistabelle aufgebaut. Wir verwenden hier zwar eine feste Struktur (zshlp_4_enno_structure), aber das Ergebnis wird dynamisch an die Ergebnistabelle übergeben.
Die Struktur sieht folgendermaßen aus:




```
FORM shlp_4_enno USING it_shlp TYPE shlp_desct
CHANGING i_callcontrol TYPE ddshf4ctrl
i_shlp TYPE shlp_descr
xt_records TYPE ddshreslts
e_rc TYPE sy-subrc.
```

```
DATA: lt_data TYPE STANDARD TABLE OF zshlp_4_enno_structure WITH NON-UNIQUE
DEFAULT KEY,
*---------------------------------------------------------------------*
* Für jedes Feld das in der Selopt als Importparaemter gekennzeichnet ist
eine Range erstellen
* Kann man sicher auch dynamisch machen - aber dann wird's schwerer lesbar
*---------------------------------------------------------------------*
lt_r_datum TYPE RANGE OF zshlp_4_enno_structure-datum,
lt_r_uname TYPE RANGE OF zshlp_4_enno_structure-uname,
ls_data LIKE LINE OF lt_data,
ls_address TYPE bapiaddr3,
lt_return TYPE bapiret2_t.
.

FIELD-SYMBOLS: <ls_selopt_line> TYPE any.

*---------------------------------------------------------------------*
* Ranges für Daten-Selektion füllen
*---------------------------------------------------------------------*
LOOP AT i_shlp-selopt ASSIGNING FIELD-SYMBOL(<ls_selopt>).

CASE <ls_selopt>-shlpfield.
WHEN 'DATUM'.
APPEND INITIAL LINE TO lt_r_datum ASSIGNING <ls_selopt_line>.
WHEN 'UNAME'.
APPEND INITIAL LINE TO lt_r_uname ASSIGNING <ls_selopt_line>.
WHEN OTHERS.
CONTINUE.
ENDCASE.

MOVE-CORRESPONDING <ls_selopt> TO <ls_selopt_line>.

ENDLOOP.

*---------------------------------------------------------------------*
* Die so gefüllten Ranges jetzt für irgend eine Datenselektion nehmen
*---------------------------------------------------------------------*
SELECT bname AS uname, trdat AS datum
FROM usr02
WHERE bname IN @lt_r_uname
AND trdat IN @lt_r_datum
INTO CORRESPONDING FIELDS OF TABLE @lt_data.

LOOP AT lt_data ASSIGNING FIELD-SYMBOL(<ls_data>).

CALL FUNCTION 'BAPI_USER_GET_DETAIL'
EXPORTING
username = <ls_data>-uname
IMPORTING
address = ls_address
TABLES
```

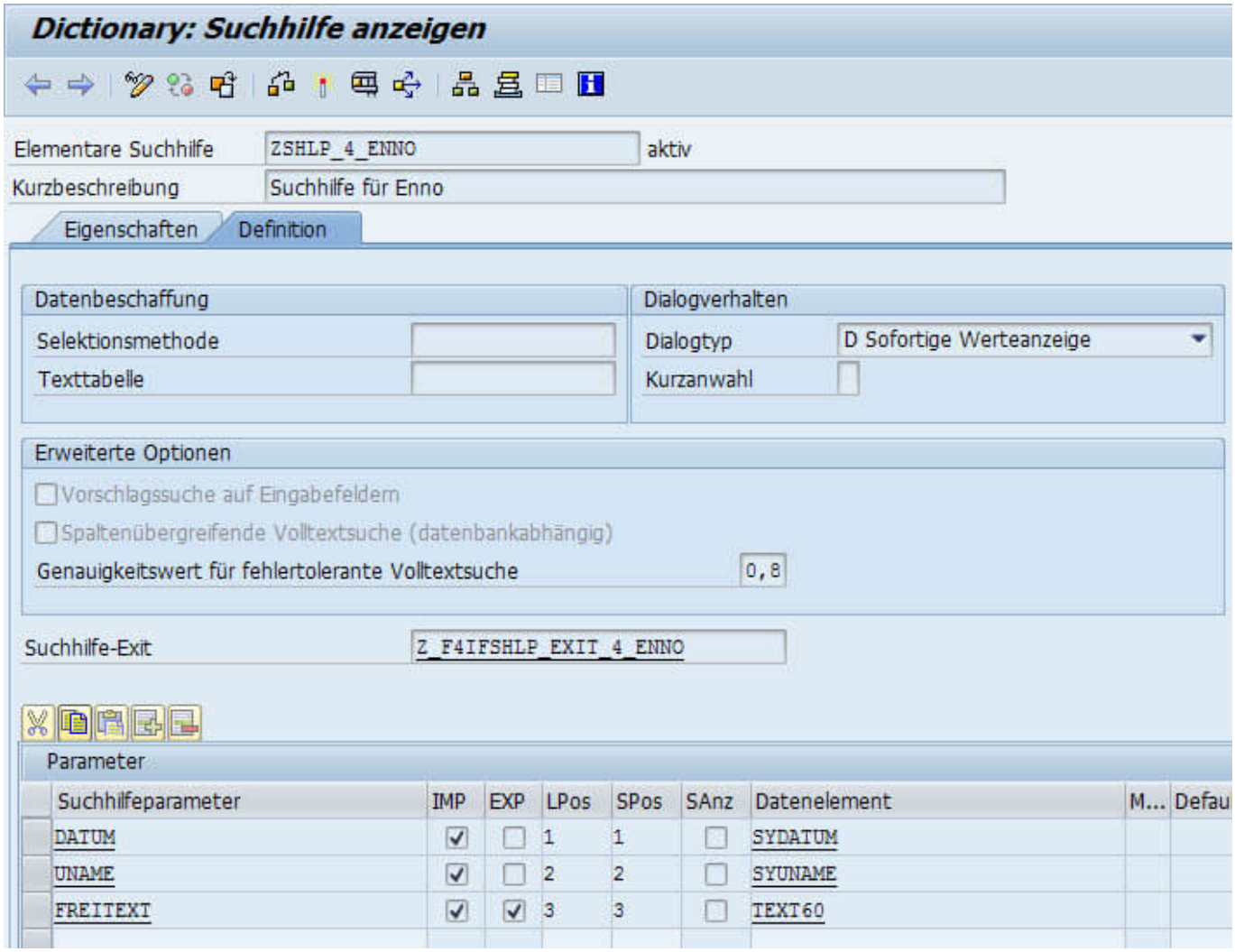
```
return = lt_return.
<ls_data>-freitext = ls_address-fullname.

ENDLOOP.

*---------------------------------------------------------------------*
* Und SAP das Ganze so aufbereiten lassen, wie man es kennt
*---------------------------------------------------------------------*
CALL FUNCTION 'F4UT_RESULTS_MAP'
EXPORTING
source_structure = 'ZSHLP_4_ENNO_STRUCTURE'
TABLES
shlp_tab = it_shlp
record_tab = xt_records[]
source_tab = lt_data[]
CHANGING
shlp = i_shlp
callcontrol = i_callcontrol
EXCEPTIONS
OTHERS = 0.
ENDFORM.
```

Suchhilfe anlegen

In Transaktion SE11 musst du nun noch eine Suchhilfe anlegen, die den Suchhilfe-Exit verwendet. Normalerweise gibst du in der Suchhilfe direkt eine *Selektionsmethode* an, also die Tabelle oder den View, aus dem gelesen wird. Da wir die Suchhilfe dynamisch aufbauen, bleibt dieses Feld leer und wir tragen nur den Baustein für den Suchhilfe-Exit ein:



Das war auch schon alles. Die Suchhilfe kann direkt getestet werden:

Anwendungsbeispiele

Der Anwendungsfall, für den man eine dynamische Suchhilfe benötigt, ist sicherlich nicht allzu oft vorhanden. Es ist aber schön, es zu können, wenn man mal vor dieser Herausforderung steht. Eine Möglichkeit wäre zum Beispiel jeweils andere Felder zu zeigen, je nachdem welche Partnerart der Anwender für die Selektion ausgewählt hat: Kunde, Lieferant oder Sachbearbeiter. Auch könnte ich mir vorstellen, dass zu Materialnummern unterschiedliche Felder je Materialart oder Anwendergruppe angezeigt werden.

Vielen Dank an Stefan für das Beispiel!!