Funktionsgruppe zum Ausführen eines SE16XXL-Scripts

Es ist eine **neue Funktionsgruppe** implementiert worden, die es einem ABAP-Programm ermöglicht, ein Script auszuführen und das Ergebnis in Form einer **Referenz-Variable** zurückzubekommen. Das ABAP-Programm hat die volle Kontrolle über das Resultat – es kann unerwünschte Zeilen entfernen, den Inhalt anderer ergänzen, und sie schließlich auf beliebige Weise anzeigen, nicht unbedingt mithilfe von ALV. Es ist nicht einmal zwingend, das Ergebnis auszugeben – stattdessen kann das Programm das Resultat des Scripts zum Aufbauen einer eigenen Liste verwenden.

Die neue Funktionsgruppe heißt **/TFTO/TX_SFMI** (SFMI steht für **S**cript **F**unction **M**odule **I**nterface). Der Haupt-Funktionsbaustein ist **/TFTO/TX_SFMI_CALL_SCRIPT**.

Um dem Programmierer einen Einblick über die Verwendung der Funktionsgruppe zu vermitteln, sind folgende **Beispiel-Programme** implementiert worden:

- **/TFTO/SFMI_CALL_SIMPLE**
- **/TFTO/SFMI_CALL_EXAMPLE**
- **/TFTO/SFMI_CALL_EXTRA**

Anhand des Codings dieser Beispiel-Programme und der einzelnen Funktions-bausteine der Gruppe sollte es einem ABAP-Programmierer möglich sein, zu verstehen, wie die neue Funktionalität zu verwenden ist.

Zum Testen empfiehlt es sich, die Beispiel-Programme zu **kopieren** und die Namen der darin aufgerufenen Scripts mit eigenen zu **ersetzen**. Der ABAP-Debuggers kann dann sehr nützlich sein, um zu untersuchen, wie die Sache funktioniert.

In den nachfolgenden Seiten werden weitere Details über die Schnittstelle und der Wirkunsgweise der Funktionsgruppe erläutert.

Wichtige Anmerkung

In den neuesten SAP-Versionen ist die maximale Länge der Feldnamen für Datenbank-Tabellen von **16 auf 30** erhöht worden.

In SE16XXL sind die meisten listen das Ergebnis von mehreren Joins. Jede Tabelle eines Joins ist einem Alias (A, B, C usw.) zugeordnet. Den Feldnamen der Tabelle wird jeweils der Alias vorangestellt , wie A~MATNR (intern A-MATNR) usw.
Bis dato ergab diese Verlängerung Feldnamen von maximal 18 Stellen, weit unter der maximalen Länge von 30 Zeichen.

Die neue Situation verlangt eine maximale Namenslänge von 32 Stellen, die leider nicht verfügbar sind (die Feldnamen in den Haupt-Strukturen wie **DFIES**, **SLIS_FIELDCAT_ALV** usw. sind weiterhin nur 30 Stellen lang). Daher ergibt sich die Notwendigkeit, die überlangen Namen von **32 auf 30** Stellen zu verkürzen. Aus **A-TOTALREPLENISHMENTLEADDURATION** würde zum Beispiel **A-TOTALREPLENISHMENTLEADDURATN** entstehen.

Allerdings ist das Verkürzen von Namen eine **knifflige Angelegenheit**. Die Felder müssen in Zusammenhang mit den restlichen Feldern der jeweiligen Tabelle betrachtet werden. Zumindest theoretisch können viele Feldnamen existieren, die sich nur durch ein einziges Zeichen unterscheiden. Oder ein Feld kann genau den verkürzten Namen eines anderen Feldes aufweisen. Darüber hinaus können zu einem späteren Zeitpunkt neue Felder hinzugefügt werden, deren Namen mit den verkürzten Formen von bereits existierenden Feldern in Konflickt kommen können. All dies zeigt, dass ein Algorithmus, basierend auf den Feldnamen allein, nicht praktikabel ist. Eine Logik, die die Originalnamen und ihre verkürzte Formen in die Datenbank speichert, würde genauso scheitern, weil neue Felder zu einem späteren Zeitpunkt hinzukommen könnten, die sich nicht mit den bisherigen vertragen.

Die einzige gangbare Lösung scheint eine **dynamische** zu sein, d.h. die Verkürzung findet jedesmal neu statt mit dem Effekt, dass die verkürzten Formen von einem Mal zum anderen unterschiedlich ausfallen können. Um korrekt arbeiten zu können, müssen Scripts nur original unverkürzte Feldnamen enthalten. Die dynamischen Kurzformen dürfen nur in volatilen Daten – wie z.B. der ALV-Feldkatalog usw. – verwendet werden. An der Oberfläche sollte der Anwender von alledem nichts merken.

Nach diesen Ausführungen dürfte es klar sein, dass die obige Situation eine Auswirkung auf die SFMI-Schnittstelle haben muss. Das ist der Grund, wenn man die verschiedenen Funktionsbausteine untersucht, warum mehrere Felder von Länge 32 zu finden sind. Sie enthalten die Original-Feldnamen, während die normalen Felder mit Länge 30 die möglicherweise verkürzten Formen enthalten. Nachdem die verkürzten Formen unzuverlässing sind – sie können von einem Lauf zum anderen unterschiedlich ausfallen – sollte das aufrufende Programm nur die Originalnamen verwenden, und die Kurzformen anhand der Tabelle ET_SFMICOL ermitteln.

Ein kleines Beispiel wird das oben Erwähnte verständlich machen.

Der Originalname sei 'A-TOTALREPLENISHMENTLEADDURATION'.

Die Tabelle ET_SFMICOL würde einen Eintrag wie folgt enthalten:

COLNAME32 = ' A_TOTALREPLENISHMENTLEADDURATION'
COLNAME = ' A_TOTALREPLENISHMENTLEADDURATN'
(das ist der Name des Feldes in der Ergebnis-Struktur – siehe nächste Seite)

ET_ALV_FCAT würde einen entsprechenden Eintrag enthalten mit:

FIELDNAME = ' A_TOTALREPLENISHMENTLEADDURATN'

Der einzige verlässliche Name ist COLNAME32 von Tabelle ET_SFMICOL, direkt vom Originalname abgeleitet.
COLNAME (und der entsprechende FIELDNAME von ET_ALV_FCAT) sollte nicht direkt verwendet werden – d.h. als Literal – im aufrufenden Programm, denn ihr Wert kann sich von Lauf zu Lauf verändern.

Eine letzte Bemerkung. Es scheint alles sehr kompliziert zu sein. Zum Glück für den Programmierer, sind die meisten Feldnamen weiterhin sehr kurz und damit von den obigen Ausführungen nicht tangiert. Nur für Feldnamen länger als 28 Zeichen muss etwas unternommen werden. In allen anderen Fällen - das sind die große Mehrheit - enthalten COLNAME32 und COLNAME denselben Wert und können damit beliebig verwendet werden.

Weitere Anmerkungen

- Die Ergebnisliste eines SE16XXL-Scripts hat eine komplizierte interne Struktur. Sollte diese interne Struktur als offizieller Bestandteil der Schnittstellenbeschreibung deklariert werden, könnten künftige Entwicklungen von SE16XXL beeinträchtigt werden. Aus diesem und anderen Gründen liefert der Funktionsbaustein eine andere Ergebnistruktur (ohne Unterstukturen) zurück. Um Namenskonflikte zu verhindern werden die Alias der Join-Einträge den einzelnen Feldnamen vorangestellt – A~MATNR (intern **A-MATNR**) wird **A_MATNR** usw. Die Felder einer einfachen Liste (ohne Join) werden auch mit "**A_**" versehen – aus **MATNR** wird also **A_MATNR**. Diese letzte Anpassung hat zusätzlich den Vorteil, dass ein nachträgliches Erweitern des Scripts durch Joins keine Änderung der Feldnamen im aufrufenden Programm mit sich zieht.

- Um Speicherplatz zu sparen wird das Script intern immer "**mit reduziertem Speicherbedarf**" ausgeführt. Daraus ergibt sich, dass in der Ergebnisliste lediglich die Felder zur Verfügung stehen, die für den korrekten Ablauf des Scripts notwendig sind.

- Nachdem die Struktur des Ergebnisses stark vom aufgerufenen Script, von den ausgewählten Listenfeldern und von anderen Faktoren abhängt, ist es für das aufrufende Programm nicht möglich, sie statisch zu deklarieren. Aus diesem Grund müssen alle Felder der Ergebnisliste dynamisch angesprochen werden mithilfe des ABAP-Befehls

  ASSIGN COMPONENT … OF STRUCTURE … TO …

  Zu diesem Zweck wird dem aufrufenden Programm nicht nur eine Referenz auf die (interne Tabelle der) Ergebnisliste zurückgeliefert, sondern auch eine Referenz auf einen Arbeitsbereich mit derselben Struktur. Dadurch kann das aufrufende Programm den relevanten Feldern geeignete Feldsymbole zuweisen, **bevor** ein Loop auf die Ergebnisliste gestartet wird.

- Einer der Gründe, ein Script aus einem ABAP-Programm aufzurufen, könnte die Notwendigkeit sein, die Ergebnisliste mit Werten **anzureichern**, die anderweitig nicht zu beschaffen sind. Zum Beispiel mithilfe von Funktionsbausteinen wie **CONVERT_TO_LOCAL_CURRENCY** um zusätzliche Spalten zu erstellen. In einer solchen Situation müsste die zurückgelieferte Ergebnisliste im Programm dupliziert werden, um die fehlenden Spalten zu ergänzen. Die Schnittstelle ist für diese Eventualitäten vorbereitet. Eine interne Tabelle **IT_EXTRA** kann dem Funktionsbaustein übergeben werden, mit der Definition der Zusatzspalten. Diese Spalten werden **leer** vom Funktionsbaustein zurückgeliefert, und müssen vom aufrufenden Programm nach der Ausführung des Scripts befüllt werden.

- Es ist im obigen Szenario klar, dass das aufrufende Programm bestimmte Spalten der Ergebnisliste benötigt, um den Inhalt der Zusatzspalten zu produzieren. Falls diese Spalten für das korrekte Funktionieren des Script entbehrlich sind und der Anwender sie nicht für die Liste ausgewählt hat, könnte es tatsächlich sein, dass sie im Ergebnis fehlen. Um diese Eventualität zu verhindern, kann das aufrufende Programm dem Funktionsbaustein eine Liste der notwendigen Felder (**IT_NEEDED**) mitgeben. Diese Felder werden dann in jeden Fall im Ergebnis präsent sein, unabhängig vom Benutzerverhalten. Ein spezieller Funktions-baustein **(/TFTO/TX_SFMI_DEFINE_NEEDED**) ist mitgeliefert, um den Programmierer mit den notwendigen ABAP-Befehlen zu unterstützen.

- Wie bereits anfangs erwähnt, braucht das aufrufende Programm die Ergebnisliste nicht anzuzeigen, und wenn doch, nicht unbedingt mithilfe von ALV.
  Sollte das Programm dennoch dies tun, liefert der Funktionsbaustein **/TFTO/TX_SFMI_CALL_SCRIPT** einen kompletten **ALV-Feldkatalog** zurück, gültig sowohl für den Funktionsbaustein **REUSE_ALV_LIST_DISPLAY** als auch für **REUSE_ALV_GRID_DISPLAY**. Dieser Feldkatalog (**ET_ALV_FCAT**) enthält alle beteiligten Felder ausser zwei (SELBOX und COLOR), die zum Markieren und Färben der Ergebniszeilen vorgesehen sind.
  Die Liste der Feldgruppierungen (nur für Joins) und mögliche Sortier-Kriterien werden auch zurückgeliefert.

- In der SE16XXL Ergebnisliste kann der Anwender jederzeit die Einstellungen ändern, z.B. von Feldnamen auf Feldbezeichner als Überschriften wechseln, oder die Konvertierungsexits ein- bzw. ausschalten.
  Diese Flexibilität steht nicht zur Verfügung, wenn das Script von einem Programm aufgerufen wird. In diesem Fall müssen die Einstellungen **vor** dem Aufruf festgelegt werden. Entweder werden die Benutzer-Einstellungen per Programm vordefiniert, ohne dass der Anwender sie ändern kann, oder es wird dem Benutzer überlassen, sie vor dem Programmlauf nach Belieben anzupassen. In jeden Fall werden die Einstellungen an den Funktionsbaustein in Form einer Range-Tabelle (IR_USETT) übergeben. Auf diese Weise kann diese Information in eine unsichtbare SELECT-OPTION des aufrufenden Programms hinterlegt werden, die dadurch in eine Variante gespeichert werden kann. All das ist nur für die ALV-Ausgabe relevant – die Ergebnisliste bleibt immer gleich.

- Der Mechanismus einer Range-Tabelle wird auch für andere Informationen verwendet, die dem Funktionsbaustein übergeben werden, wie z.B. die Listenfelder, die Sortier-Kriterien usw. Es ist damit ohne Probleme möglich, diese Daten in eine Programm-Variante zu speichern.
  Es soll nicht versucht werden, diese Range-Tabellen direkt zu füllen, sondern nur mithilfe der mitgelieferten Funktionsbausteine der Funktionsgruppe.

- Auch wenn in den meisten Fällen nur ein Script aufgerufen werden wird, ist es denkbar, ein Programm zu implementieren, das mehrere Scripts hintereinander aufruft, um daraus ein kombiniertes Ergebnis zu produzieren.

- Unter der Annahme, dass das aufrufende Programm sich um die Berechtigungen des Anwenders kümmern wird, werden seitens des Funktionsbausteins keine Prüfungen in Bezug auf das auszuführende Script durchgeführt. Das bedeutet, dass das Programm jedes existierende Script aufrufen kann, sowohl global als auch benutzerspezifisch – sogar private Scripts von anderen Benutzern. Die letzterwähnte Option hat den Vorteil, das beteiligte Script von (un)beabsichtigten Manipulationen seitens anderer Personen zu schützen.

- Bezüglich Zugriffsrechte für Tabellen und Felder bleiben diese **vollständig in Kraft**. Es können mithilfe eines ABAP-Programms nur Daten aus der Datenbank herausgelesen werden, die sonst auch mit SE16XXL zu bekommen sind. Das gilt auch für die Berechtigungs-Prüfungen auf Satzebene (inklusive spezielle Berechtigungen). Das Ausführen eines Scripts mithilfe eines ABAP-Programms **umgeht nicht** die Prüfung der Daten.

- Das aufgerufene Script könnte eine oder mehrere Filter-Operationen enthalten, die zu einer Serie von ausgeblendeten Sätzen im Ergebnis führen, die in SE16XXL mithilfe der Operation "Ausgeblendete Sätze wieder anzeigen" erneut zum Vorschein gebracht werden können. Diese ausgeblendeten Sätze werden in Zusammenhang mit SFMI ignoriert – nur die Sätze, die am Ende des Script-Laufes sichtbar sind, werden berücksichtigt und zurückgeliefert.