Moderne Workflow-Programmierung mit ABAP® Objects

Handbuch für Entwickler

von
Ulrich Mende

1. Auflage

Thematische Gliederung:

SAP

dpunkt.verlag 2012

Verlag C.H. Beck im Internet:
www.beck.de
ISBN 978 3 86490 013 6

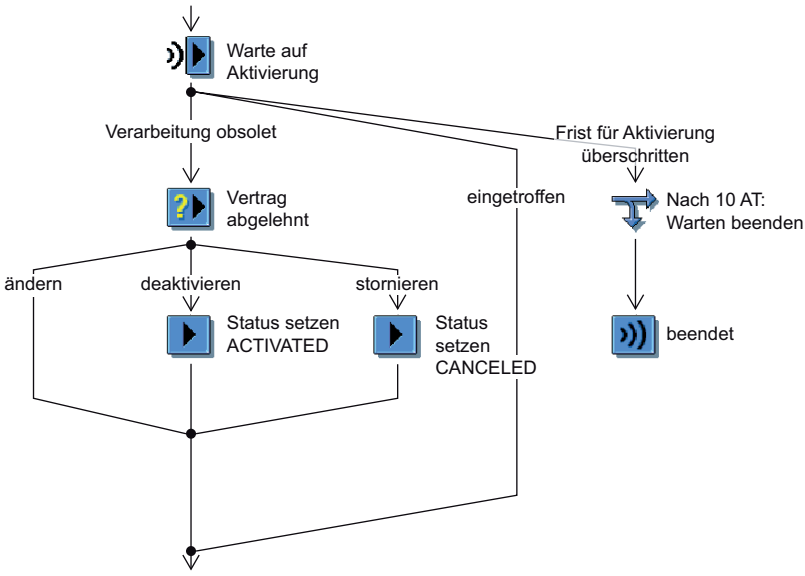


Abb. 10–48
Ereignisempfänger mit Fristüberwachung

Nach einer Fristüberschreitung setzt ein Steuerschritt das Warte-Workitem auf obsolet, wodurch dessen Obsoletzweig durchlaufen wird. In diesem befindet sich eine Sachbearbeiterentscheidung (SBE), die die Alternativen *Vertrag stornieren* und *Vertrag aktivieren* zulässt. In den entsprechenden Folgezweigen wird der Vertragsstatus dunkel (= ohne Dialog) gesetzt.

10.11 Klassenverwendung im Workflow ohne Standardaufgaben

10.11.1 Konzept

Viele Entwickler scheuen den Aufwand, für jede Workflow-Klasse extra das Interface IF_WORKFLOW zu implementieren und für jede (ggf. statische) Methode unbedingt eine eigene Standardaufgabe anzulegen. Es fragt sich daher, ob es nicht möglich ist, Klassen auch direkt im Workflow zu verwenden. Das geht tatsächlich, wobei unterschiedliche Varianten möglich sind.

Die folgende Tabelle zeigt, wie auf welche Komponenten einer Klasse im Workflow ohne Standardaufgaben zugegriffen werden kann. Wie sich zeigt, ist dabei nur teilweise eine Implementierung von IF_WORKFLOW notwendig.

Der statische Klassenzugriff, der einen Ausdruck über die %-Zeichen klammert, ist nur für statische Klassenkomponenten (statische Attribute und funktionale, statische Methoden) möglich.

Tab. 10–5 Klassenaufrufe im Workflow ohne Standardaufgaben

		Komponente statisch		Instanz-komponente	
		Attr.	Funkt. Meth.	Attr.	Funkt. Meth.
%...%					
■ Zugriff ohne Containerreferenz ■ kein Interface IF_WORKFLOW		ja kein F4	ja kein F4	nein	nein
&…&					
■ Zugriff mit Container-referenz	■ Referenz ungebunden ■ Interface ohne Implementierung	ja F4	**nein !**	nein	nein
■ Interface nötig: IF_WORKFLOW	■ Referenz gebunden ■ Interface mit Implementierung	ja F4	ja F4	ja F4	ja F4

Der Referenzzugriff, der einen Ausdruck über die &-Zeichen klammert, erfordert immer eine Referenz auf die Klasse im Workflow-Container. Dabei muss diese Referenz nicht einmal gebunden sein, um statische Attribute anzusprechen.

Die im Folgenden vorgestellten Beispiele benutzen zunächst eine Klasse ZCL_CALC mit einer einfachen funktionalen, statischen Methode SUM (numerische Summenbildung). Zusätzlich greifen sie auf die im Abschnitt 11.1 vorgestellte Utility-Klasse ZCL_DATE zur Datumsberechnung auf Fabrikkalendern vor.

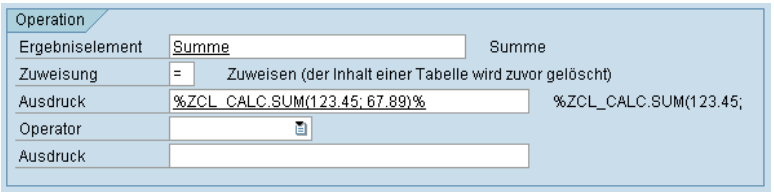
10.11.2 %-%-Zugriff ohne IF_WORKFLOW

Diese Zugriffsart ist nur für statische Konstanten und statische funktionale Methoden geeignet. (Im Abschnitt 10.11.6 wird gezeigt, wie auch nicht funktionale Methoden mit mehreren Exportparametern aufgerufen werden können.) Da es keine entsprechende Klassenreferenz im Workflow-Container gibt, kann man die Komponenten für den Ausdruck auch nicht mittels F4 aus dem Container wählen. Man muss also Namen und Signaturen der statischen Attribute und funktionalen Methoden genau kennen. Dafür kommt die Klasse vollständig ohne das Interface IF_WORKFLOW aus.

Bei Fehlern in der Eingabe erhält man allerdings detaillierte und sehr hilfreiche Hinweise, was an dem Ausdruck nicht stimmt.

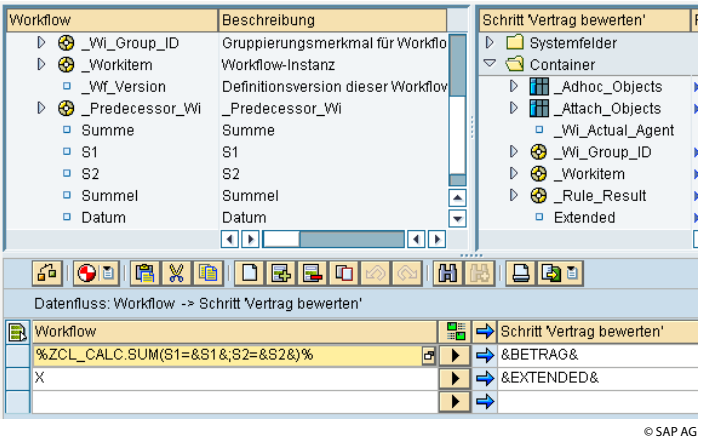
Die Abbildung 10–49 zeigt, wie eine statische Methode SUM einer Klasse ZCL_CALC (ohne Workflow-Interface) mit zwei konstanten Werten für die beiden Summanden in einer Containeroperation verwendet wird.

Abb. 10–49
Ausdruck mit funktionaler Methode in Containerzuweisung



In der folgenden Abbildung wird dieselbe Methode in einem Datenfluss benutzt, ihr werden aber als Summanden die beiden Containervariablen S1 und S2 übergeben.

Abb. 10–50
Funktionale Methode mit Parametervariablen im Datenfluss



10.11.3 &-&-Zugriff über ungebundene Containerreferenzen

Es leuchtet ein, dass ungebundene Referenzen nur auf statische Komponenten zugreifen können, da eben keine Instanz vorhanden ist. Leider können sie aber ausschließlich statische *Attribute* behandeln, mit statischen, funktionalen *Methoden* kommen sie nicht zurecht. Diese werden zwar per F4-Hilfe angeboten und können auch ausgewählt werden, liefern aber keine Werte. Die statischen Attribute werden mit ihren Werten bereits im Container angezeigt.

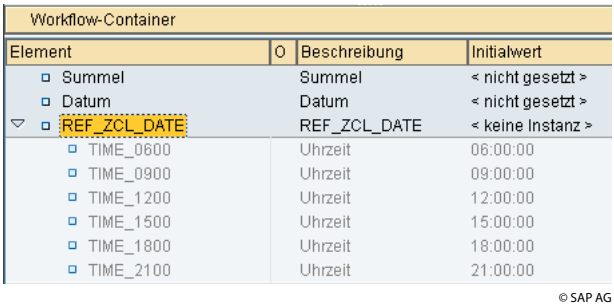


Abb. 10–51
Ungebundene Klassenreferenz im Container

10.11.4 &-&-Zugriff über gebundene Containerreferenzen

Mit echten, gebundenen Containerreferenzen kann man auf alle Komponenten zugreifen, es besteht immer eine Eingabehilfe, sowohl für Attribute als auch für funktionale Methoden. Natürlich muss das Interface IF_WORKFLOW für entsprechende Instanziierung unterstützt werden.

Ein Trick für Utility-Klassen, die ausschließlich statische, funktionale Methoden besitzen, besteht darin, eine Pseudoinstanziierung zu nutzen. Dies wird am Beispiel der Utility-Klasse ZCL_DATE (siehe Abschnitt 11.1) dargestellt. Folgende Punkte sind zu erledigen, um die Klasse und ihre statischen, funktionalen Methoden im Workflow ohne Standardaufgaben und mit komfortabler F4-Hilfe zu nutzen:

- Klasse hat keinen CONSTRUCTOR, da nur Pseudoinstanziierung für Workflow.
- Interface IF_WORKFLOW in der Klasse eintragen.
- Statische Membervariable MO_SELF Type Ref To ZCL_DATE anlegen.
- Keine Schlüsselattribute
- Coding für Interfacemethoden FIND_BY_LPOR und LPOR wie in Listing 10–12 eintragen.
- Klassenreferenz im Workflow eintragen und initialisieren.

```
METHOD BI_PERSISTENT~FIND_BY_LPOR.
* statisch: DATA: mo_self Type Ref To ZCL_DATE
  if not mo_self is bound.
    create object mo_self. " kein CONSTRUCTOR!
  endif.
  result = mo_self.
ENDMETHOD.
```

Listing 10–12
Pseudoinstanziierung

```
METHOD BI_PERSISTENT~LPOR.
   result-catid  = 'CL'.
   result-typeid = 'ZCL_DATE'.
   result-instid = '0'. " beliebig
ENDMETHOD.
```

Die LPOR-INSTID ist beliebig, sie wird gar nicht verwendet. Bei der Instanziierung (= Initialisierung) im Workflow kann man einfach 0 eingeben. Danach erscheint die Referenz im Workflow-Container als gebunden und man kann die statischen Methoden per F4-Hilfe an der Referenz auswählen und bekommt das vollständige Muster inklusive Signatur zum Ausfüllen in den Ausdruck eingestellt.

Insbesondere, wenn man eine Utility-Klasse häufig verwenden möchte, ist der geringe Mehraufwand durch die elegante F4-Hilfe gerechtfertigt.

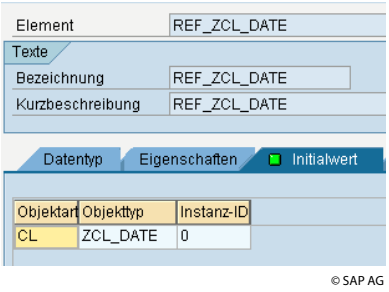


Abb. 10–52 *Instanziierung (= Initialisierung) der Pseudoreferenz der Utility-Klasse*

Abbildung 10–53 zeigt einen Referenzausdruck in einem Datenfluss.

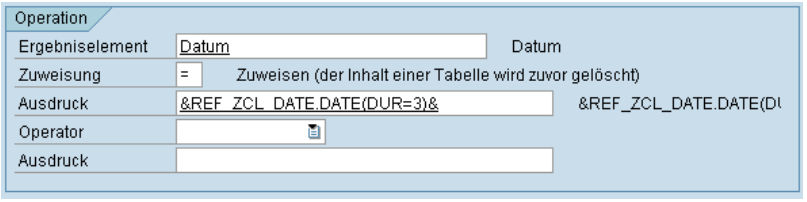


Abb. 10–53 *Kompletter (ergänzter) Ausdruck*

In der Abbildung 10–54 ist die aufgeklappte Referenz in einem Datenfluss zu sehen. Man kann die Methode DATE auswählen und muss nur die Dauer ergänzen.

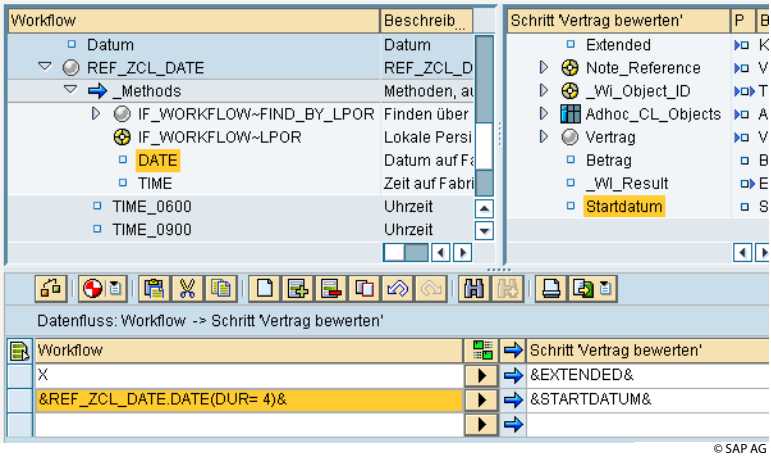


Abb. 10–54
Ausdruck mit funktionaler Methode via Referenz im Datenfluss

10.11.5 Geschachtelter Zugriff mit %-% und &-&

Es ist verständlich, dass sich mehrere %-%-Paare oder mehrere &-&-Paare untereinander in einem Ausdruck nicht vertragen. Anders als bei Klammerpaaren haben sie ja keinen öffnenden und schließenden Teil.

So wird der Ausdruck &ZCL_DATE.DATE(DUR=&Tage&)& abgewiesen. Der Interpreter weiß einfach nicht, ob das zweite & das schließende zum ersten ist oder ob ein neues Paar beginnt.

Möglich ist allerdings die mehrfache, alternierende Schachtelung von %-%-Paaren mit &-&-Paaren, wie folgende Abbildung einer Bedingung zeigt.

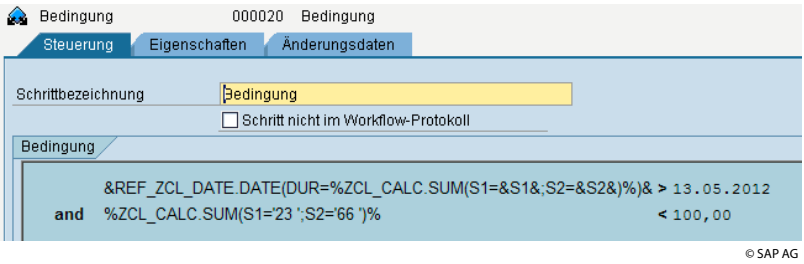


Abb. 10–55
Geschachtelter Ausdruck in einer Bedingung

In Bedingungen können Klassenausdrücke nur über den einfachen Zeileneditor eingegeben werden.

Es fragt sich natürlich, ob solche komplexen, geschachtelten Ausdrücke sinnvoll sind. In vielen Fällen scheint es besser, eine Zwischenvariable mit Zwischenwerten zu füllen. Das erleichtert später auch das Lesen der Workflow-Protokolle. Wer könnte schon auf Anhieb sagen, ob das Datum im o.g. Ausdruck wirklich nach dem 13.05.2012 liegt.

10.11.6 Methoden mit mehreren Exportparametern

In den vorangegangenen Teilkapiteln wurde angenommen, dass die aufgerufenen Methoden funktional waren, also genau einen RETURNING-Parameter und keine Exportparameter haben. Diese Annahme wird durch die Verwendung in Ausdrücken (Datenflüsse, Containerzuweisungen) auch nahegelegt.

SAP bietet aber eine Möglichkeit an, auch nicht funktionale Methoden, die ggf. auch mehrere Exportparameter besitzen, aufzurufen. Dies wird für die folgende Methode ZCL_CALC.POWER demonstriert, die einen Import- und zwei Exportparameter hat.

Listing 10–13
Nicht funktionale Methode mit zwei alternativen Exportparametern

```
*****************************************************
* Parameters:
* I       Importing    TYPE  INT4  Natürliche Zahl
* SQUARE  Exporting    TYPE  INT4  Quadrat
* CUBIC   Exporting    TYPE  INT4  Kubik
*****************************************************
method POWER.
   square = i ** 2.
   cubic  = i ** 3.
endmethod.
```

Der nachfolgende Datenfluss »ernennt« über die Zuweisung

_RESULT = CUBIC

den Exportparameter CUBIC zum RETURNING-Parameter. Dessen Wert wird dann in der Zuweisung verwendet (s. Abb. 10–56). Dieses Verfahren funktioniert sinngemäß auch für den anderen Parameter SQUARE.

Gleichzeitig lassen sich die beiden Exportparameter leider nicht verwenden, folgende Anweisung wird abgewiesen:

Leider nicht möglich!

```
ZCL_CALC.POWER(I=&S1&;_RESULT=CUBIC;SQUARE=&QUADRAT&)
```

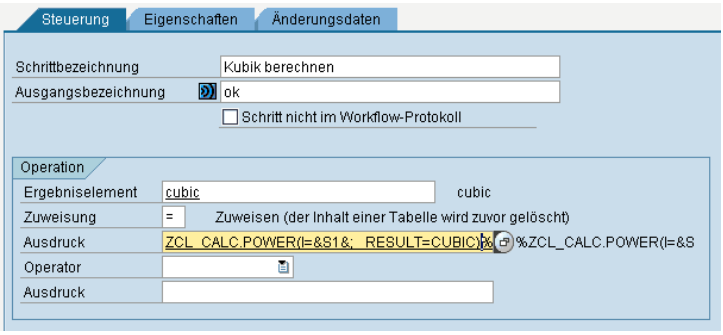


Abb. 10–56
Aufruf der Methode ZCL_CALC.POWER in einem Datenfluss

Das ist nicht ganz verständlich, denn wenn man via I=&S1& Importparameter mit Werten aus dem Container versorgen kann, dann könnte man ja sicher auch mit SQUARE=&QUADRAT& die Werte der Exportparameter in den Container zurückschreiben. Vielleicht befürchtet man dadurch unkontrollierbare Seiteneffekte im Container.

10.11.7 Objektreferenzen on-the-Flight

Wenn eine Aufgabe als Importparameter Objektreferenzen verlangt und man schon zur Definitionszeit weiß, welche Objektinstanz zu übergeben ist, so kann man Ausdrücke der folgenden Form in Datenflüssen benutzen:

[BO.ZVERTRAG.00000008]	für BOR-Objekte
[CL.ZCL_VERTRAG.00000008]	für Klasseninstanzen

Das hat den Vorteil, dass man keine Objektreferenz im Quellcontainer haben muss.

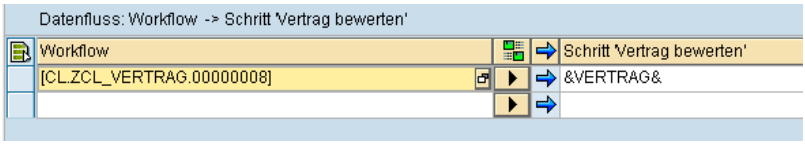


Abb. 10–57
Objekt on-the-Flight im Datenfluss

Containervariablen kann man in diese Ausdrücke aber nicht einbinden, folgender Ausdruck ist ungültig:

[CL.ZCL_VERTRAG. &VerNr&] F A L S C H !

Die Ausdrücke müssen zur Definitionszeit vollständig bekannt sein.