

# **ABAP DSL Workbench**

## **SAP TechED 2016**

Barcelona, November 2016

| VEGETARISCHE GERICHTE                            |      | ENTENFLEISCH GERICHTE mit Reis             |      | UNGERE THAI SPEZIALITÄTEN mit Reis             |      | GLASNUDELN GEBRATEN   |      |
|--|------|--|------|--|------|---|------|
| 27. Reis   | 2,50 | 46. Entenfleisch Chop-Suey                 | 7,00 | 70. Gaeng-Klaw Wan-Gai Hühnerfleisch           | 6,00 | 84. Glasnudeln gebraten mit Rindfleisch                                   | 5,50 |
| 28. Reis   | 2,00 | 47. Ente-Kross gebacken                    | 7,00 | 71. Gaeng-Pet-Gai Hühnerfleisch                | 6,00 | 85. Glasnudeln gebraten mit Hühnerfleisch                                 | 5,50 |
| 68. Gemüse                                       | 2,20 | 48. Ente-Kross gebacken                    | 7,00 | 72. Gal-Phat-Kraphao Hühnerfleisch             | 6,00 | 86. Glasnudeln gebraten mit Schweinefleisch                               | 6,50 |
| Banane   | 2,20 | 49. Ente-Kross gebacken                    | 7,00 | 73. Gaeng-Klaw Wan-Muh Schweinefleisch         | 6,00 | 87. Glasnudeln gebraten mit Hühnerfleisch                                 | 7,50 |
| Bambus   | 2,20 | 50. Enten-, Schweinefleisch                | 7,00 | 74. Gaeng-Pet-Muh Schweinefleisch              | 6,00 | 88. Glasnudeln gebraten mit Entenfleisch                                  | 8,00 |
| 69. Gemüse                                       | 2,20 | 51. Ente-Kross gebacken                    | 7,00 | 75. Muh-Phat-Kraphao Schweinefleisch           | 6,50 | 89. Glasnudeln gebraten mit Ente-Kross                                    | 8,00 |
| mit Reis   |      | 52. Ente-Kross gebacken                    | 7,00 | 76. Gaeng-Pet-Nuah Rindfleisch                 | 6,50 | <b>DESSERT</b>  |      |
| 29. Hähnchenfleisch Chop-Suey                    | 5,50 | 53. Rindfleisch Chop-Suey                  | 6,00 | 77. Nuah-Phat-Kraphao Rindfleisch              | 6,50 | 90. Apfel gebacken mit Honig  | 2,50 |
| 30. Hähnchenfleisch mit Sojabohnensprossen       | 5,50 | 54. Rindfleisch mit Broccoli               | 6,00 | 78. Pla-Phat-Kraphao Fisch gebacken            | 8,00 | 91. Ananas gebacken mit Honig   | 2,50 |
| 31. Hähnchenfleisch mit Champignons u. Spargeln  | 6,00 | 55. Rindfleisch mit Broccoli und Knoblauch | 6,50 | 79. Gaeng-Pet-Ped-Yang 1/4 Ente-Kross gebacken | 8,00 | 92. Banane gebacken mit Honig   | 2,50 |
| 32. Hähnchenfleisch im Teig gebacken             | 6,00 | 56. Rindfleisch mit Broccoli und Knoblauch | 6,50 | 80. Pet-Phat-Ki-Mon 1/4 Ente-Kross gebacken    | 8,00 | 93. Dreierlei Obst gebacken   | 3,00 |
| 33. Hähnchenfleisch im Teig gebacken             | 6,00 | 57. Rindfleisch mit Broccoli und Knoblauch | 6,50 | 81. Gaeng-Pat-Gung Großgarnelen                | 8,50 | <b>GETRÄNKE</b>   |      |
| 34. Hähnchenfleisch im Teig gebacken             | 6,00 | 58. Rindfleisch mit Paprika und Zwiebeln   | 6,50 | 82. Gaeng-Pet-Ped-Gai: Hühnerfleisch gebacken  | 7,00 | Cola <sup>1,2</sup> , Fanta <sup>2,3</sup> , Sprite <sup>3</sup> , Wasser | 0,31 |
| 35. Hähnchenfleisch im Teig gebacken             | 6,00 | 59. Fischfilet Chop-Suey                   | 7,00 | 83. Gaeng-Pet-Ped-Pla: Fischfilet gebacken     | 7,00 | Cola <sup>1,2</sup> , Fanta <sup>2,3</sup> , Sprite <sup>3</sup> , Wasser | 0,41 |
| 36. Hähnchenfleisch im Teig gebacken             | 6,00 | 60. Fischfilet gebacken mit Gemüse         | 7,00 |  |      | Cola <sup>1,2</sup> , Fanta <sup>2,3</sup> , Sprite <sup>3</sup> , Wasser | 0,51 |
| 37. Hähnchenfleisch im Teig gebacken             | 6,00 | 61. Fischfilet gebacken mit Gemüse         | 7,00 |  |      | Saft  | 0,31 |
| 38. Hähnchenfleisch im Teig gebacken             | 6,00 | 62. Fischfilet gebacken mit Gemüse         | 7,00 |  |      | Saft  | 0,41 |
| 39. Schweinefleisch Chop-Suey                    | 5,50 | 63. Großgarnelen Chop-Suey                 | 7,50 |  |      | Bier  | 0,31 |
| 40. Schweinefleisch mit Sojabohnensprossen       | 5,50 | 64. Großgarnelen mit Broccoli              | 8,00 |  |      | Bier  | 0,51 |
| 41. Schweinefleisch mit Champignons und Spargeln | 6,00 | 65. Großgarnelen mit Curry                 | 8,00 |  |      | Asiatische Getränke   | 0,21 |
| 42. Schweinefleisch im Teig gebacken             | 6,00 | 66. Großgarnelen im Teig gebacken          | 9,00 |  |      |   | 1,50 |
| 43. Schweinefleisch mit Gemüse und Curry         | 6,00 |  |      |  |      |   | 2,00 |
| 44. Schweinefleisch Zweimal gebacken             | 6,00 |  |      |  |      |   | 2,50 |
| 45. Schweinefleisch Zweimal gebacken             | 6,50 |  |      |  |      |   | 2,20 |

Hello.

Hello.

Yes?

Number 77.

Take away?

No.

Hello.

Hello.

Hello.

Hello.

As always?

Yes.

As always?

Yes.

Where are the women?

Take away?

No.

## What is a Domain-Specific Language (DSL)?

“A Domain-Specific Language (DSL) is a small programming language, which focuses on a particular domain.” (see Xtext manual, p.2)



**Domain-Specific Language**



**GPL - General Purpose Language**

### DSL: Asian Diner

```

1 DomainModel = asian_diner.
2
3 asian_diner = order {order}.
4
5 order = meal [take_away].
6
7 meal = "number" "=" meal_number.
8
9 meal_number = "76" | "77".
10
11 take_away = "take_away".

```

### Modell: Lunch

```

1 number = 76
2 take_away

```

- ▼ /ADW/CL\_ASIAN\_DINER
  - ▶ Superclasses
  - ▶ Attribute
  - ▼ Methods
    - ▶ Inherited Methods
    - ▼ Redefinitions
      - END
      - GENERATE
      - START
  - ▼ Macros
    - ADW\_BREAK

#### Methode START

```

1 METHOD start.
2   CLEAR: price.
3 ENDMETHOD.

```

|                |                            |       |
|----------------|----------------------------|-------|
| VALUE( ENTRY ) | TYPE REF TO /ADW/CL_NODE   | Know  |
| PARSER         | TYPE REF TO /ADW/CL_PARSER | Allge |
| /ADW/CX_AMF    |                            | Allge |

#### Methode GENERATE

```

1 METHOD generate.
2
3   adw_break.
4
5   IF parser->md_nsymbol = 'meal_number'.
6     CASE entry->get_tsym( ).
7       WHEN '76'.
8         price = '6.50'.
9       WHEN '77'.
10        price = '6.00'.
11     ENDCASE.
12   ENDIF.
13
14   IF entry->is_terminal( ) = abap_true AND
15     entry->get_tsym( ) = 'take_away'.
16     price = price - '0.50'.
17   ENDIF.
18 ENDMETHOD.

```

| Parameter | Type spec.                 | Description        |
|-----------|----------------------------|--------------------|
| PARSER    | TYPE REF TO /ADW/CL_PARSER | Allgemeiner Parser |

#### Methode END

```

1 METHOD end.
2   DATA text TYPE string.
3   CONCATENATE 'The price is' price 'EUR' INTO text SEPARATED BY space.
4   MESSAGE text TYPE 'I'.
5 ENDMETHOD.

```

## You can use the API of the workbench to call it from your own programs

```
REPORT /adw/api_sample.  
  
DATA workbench_api TYPE REF TO /adw/cl_api.  
  
PARAMETERS:  
  model TYPE /adw/objdir-name MEMORY ID /adw/mmd OBLIGATORY.  
  
INITIALIZATION.  
  CREATE OBJECT workbench_api.  
  
START-OF-SELECTION.  
  DATA msg TYPE REF TO /adw/cx_adw.  
  
TRY.  
  workbench_api->run( model ).  
  CATCH /adw/cx_adw INTO msg.  
  MESSAGE msg TYPE 'S'.  
ENDTRY.
```

The screenshot shows two windows from the ABAP IDE. The top window, titled "Demo ABAP DSL Workbench API", contains a form with a "Name of the model" field where the value "LUNCH" is entered. The bottom window, titled "D03(1)/800 Information", displays the output message "The price is 6.00 EUR". A vertical line connects the "Name of the model" field to the output message, indicating the data flow.

- The input and output data is controlled by the generator class
- DSL or model can be provided during run time



**First you have to create a DSL by referring to an EBNF.**

The screenshot shows the ADventas DSL Workbench interface. At the top, there is a header bar with the text 'DSL' and several icons (a document, a pencil, a link, and an information icon). Below this, there is a main workspace area. In the foreground, a dialog box titled 'D03(1)/800 Properties' is open. The dialog box has a 'Name' field containing 'ENTITIES'. Below the name field, there are several fields for properties: 'Objecttyp' (DSL), 'Description' (Sample from Xtext User Guide), 'Parser' (EBNF), 'Generator' (STANDARD), 'Version' (000001), 'Created By' (ADVENTAS), 'Created on' (26.09.2014), 'Last changed by' (ADVENTAS), and 'Changed on' (10.10.2015). A green checkmark icon is visible in the bottom right corner of the dialog box.

- You enter a name of the DSL
- You put in a description and
- You choose a definition language / parser
- You choose a generator – STANDARD first and later the generator you have implemented for the DSL you defined.

## To define the DSL you use an extended EBNF.

*DSL: Sample from Xtext User Guide*

```

1  * Sample from Xtext User Guide, p. 5
2
3  DomainModel = {Type}.
4
5  Type = DataType | Entity.
6
7  DataType = "datatype" name=DTEL.
8
9  Entity = "entity" name=ID ["extends" superType=[Entity]] "{" {Feature} "}".
10
11 Feature = name=ID ":" TypeRef.
12
13 TypeRef = referenced=[Type] [multi?="*"].

```

### EBNF:

{ x } – Optional Repetition

x | y – Alternativ

[ x ] – Option

„x“ – Terminal Symbol

x – Non Terminal Symbol

### The extensions are:

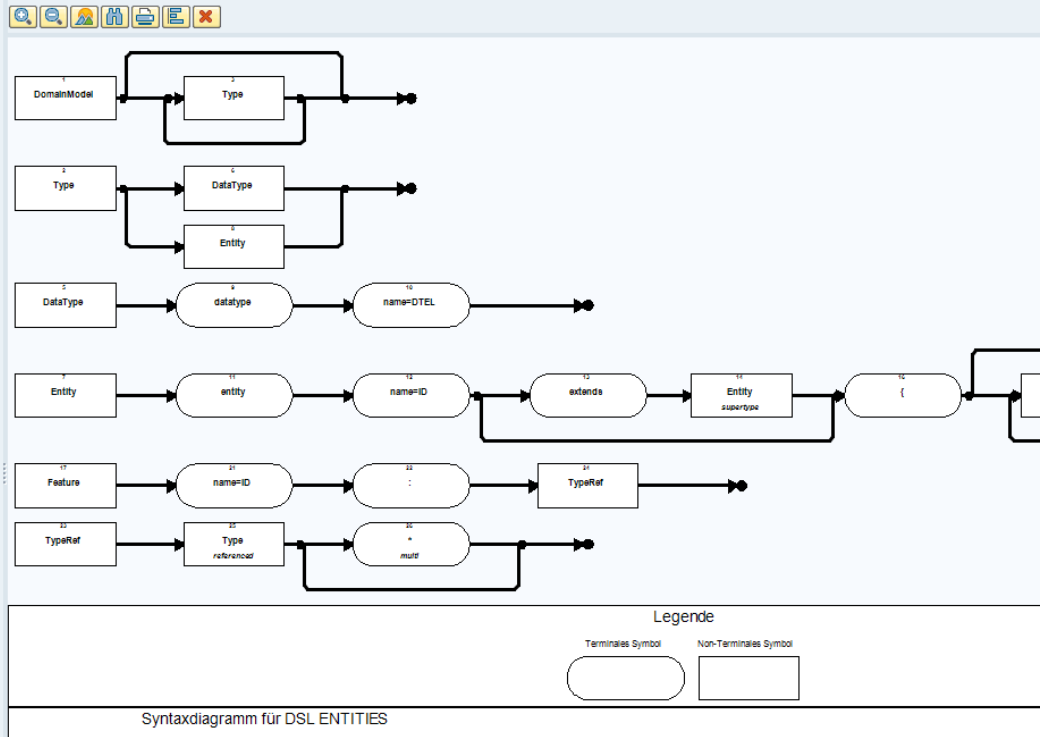
- *superType* - inheritance, *referenced* – cross-reference und *multi* - multiplicity as in xtext
- You can refer to DDIC types e.g. like *name=DTEL*
- Character strings of fixed or variable length are supported e.g. *name=c(4)* or *name=STRING* as well as digits e.g. *name=d(4)*.

After the syntax check the syntax diagram is shown. It helps you to develop the DSL.

DSL: Sample from Xtext User Guide

```

1  * Sample from Xtext User Guide, p. 5
2
3  DomainModel = {Type}.
4
5  Type = DataType | Entity.
6
7  DataType = "datatype" name=DTEL.
8
9  Entity = "entity" name=ID ["extends" superType=[Entity]] "(" (Feature) ")"
10
11 Feature = name=ID ":" TypeRef.
12
13 TypeRef = referenced=[Type] [multi?="*"].
  
```





## In another transaction you can use the DSL .

**Model**

📄 ✎ 68

Name

🔍 D03(1)/800 Eigenschaften

|                 |                 |
|-----------------|-----------------|
| Objektyp        | MMD             |
| Titel           | Conference Tool |
| DSL             | ENTITIES        |
| Version         | 000001          |
| Anleger         | ADVENTAS        |
| Erstellung.dat. | 26.09.2014      |
| letzter Änderer | ADVENTAS        |
| Änderun.datum   | 09.10.2015      |

**Modell: Conference Tool**

```

1  datatype sap_bool
2  datatype Bool
3  datatype String
4
5  entity Session {
6      Title: String
7      IsTutorial: Bool
8  }
9
10 entity Person {
11     Name: String
12 }
13
14 entity Speaker extends Person {
15     Sessions: Session*
16 }
17
18 entity Conference {
19     Name: String
20     Attendees: Person*
21     Speakers: Speaker*
22 }

```

- Enter a description
- Choose the DSL the model refers to
- Start modelling

**During syntax check the syntax tree is build and shown afterwards to the user.**

The screenshot displays the ADventas DSL Workbench interface. The main window is titled "Modell: Conference Tool". On the left, a code editor shows the following DSL code:

```
1 datatype sap_bool
2 datatype Bool
3 datatype String
4
5 entity Session {
6   Title: String
7   IsTutorial: Bool
8 }
9
10 entity Person {
11   Name: String
12 }
13
14 entity Speaker extends Person {
15   Sessions: Session*
16 }
17
18 entity Conference {
19   Name: String
20   Attendees: Person*
21   Speakers: Speaker*
22 }
```

On the right, a syntax tree is displayed. The root node is "CONFERENCE", which contains several "Type" nodes. Each "Type" node contains "DataType" nodes, which in turn contain "datatype" and "sap\_bool" nodes. The "Type" nodes also contain "Entity" nodes, which contain "entity", "Session", and "{" nodes. The "Entity" nodes also contain "Feature" nodes, which contain "Title", ":", and "TypeRef" nodes. The "TypeRef" nodes contain "Type" nodes.

## Example 2 – Converting a stock list

**Modell: Items on stock**

1  
2  
3 **L A G E R - B E S T A N D**  
4 Seite : 1  
5 freie Menge per Fa. 10 Abteilung 10 - 17 (am 04.01.2014)  
6 SAMPLE - 4.01.14/13:05:08  
7  
8 -----  
9 Partie Lot Artikel Lieferant Lager MHD/WE M Verp Lager- freie EK-Preis Whg  
10 Kurs Einstand Z Bewertung Lagerwert Lagerwert Datum Bestand Menge  
11 EUR EUR EUR unverkauft  
12 -----  
13  
14  
15 **T O T A L :**  
16 Mandant : 10 ABC Sample (Europe) GmbH  
17 Vertriebsber: 10 ABC HH  
18 Lieferant : 116106 IDES Europe GmbH,Berlin  
19 Artikel : G-L-A-1449 Wasserfilter 0,45my  
20 9025082 G-L-A-1449 FLOW 361 AH 16.12.13 2,000 STK 2,000 46,87 EUR  
21 1,000000 46,870 J 46,8700 93,74 93,74  
22 Artikel : G-L-A-1449 Wasserfilter 0,45my  
23 93,74 93,74  
24 Artikel : G-L-A-1555 Wasserfilter 1,0my  
25 9025082 G-L-A-1555 FLOW 361 AH 16.12.13 2,000 STK 2,000 29,67 EUR  
26 1,000000 29,670 J 29,6700 59,34 59,34  
27 Artikel : G-L-A-1555 Wasserfilter 1,0my  
28 59,34 59,34  
29 Artikel : TL-001001-1 HP Seal Kit 55K  
30 9025082 TL-001001-1 FLOW 361 AH 16.12.13 2,000 STK 0,000 71,38 EUR  
31 1,000000 71,380 J 71,3800 142,76 0,00  
32 Artikel : TL-001001-1 HP Seal Kit 55K  
33 142,76 0,00  
34 Artikel : TL-001006-1 Niederdruck Dichtungssatz  
35 9025082 TL-001006-1 FLOW 361 AH 16.12.13 1,000 STK 0,000 61,92 EUR

STOC... Ze 1 Sp 1 N...

| Mandant | Vertrieb | Lieferant | Partie  | Lot | Artikel     | Lager | EKGrp. | MHD/WE   | ME Verp. | Lagerbest. | ME  | Frei  | EK Preis | Währung | Kurs     | Einst.pr. | Verzollung | Wert/ME | Ges.Wert | Wert.frei |
|---------|----------|-----------|---------|-----|-------------|-------|--------|----------|----------|------------|-----|-------|----------|---------|----------|-----------|------------|---------|----------|-----------|
| 10      | 10       | 116106    | 9025082 |     | G-L-A-1449  | 361   | AH     | 16.12.13 |          | 2,000      | STK | 2,000 | 46,87    | EUR     | 1,000000 | 46,870    | J          | 46,8700 | 93,74    | 93,74     |
| 10      | 10       | 116106    | 9025082 |     | G-L-A-1555  | 361   | AH     | 16.12.13 |          | 2,000      | STK | 2,000 | 29,67    | EUR     | 1,000000 | 29,670    | J          | 29,6700 | 59,34    | 59,34     |
| 10      | 10       | 116106    | 9025082 |     | TL-001001-1 | 361   | AH     | 16.12.13 |          | 2,000      | STK | 0,000 | 71,38    | EUR     | 1,000000 | 71,380    | J          | 71,3800 | 142,76   | 0,00      |
| 10      | 10       | 116106    | 9025082 |     | TL-001001-1 | 361   | AH     | 16.12.13 |          | 1,000      | STK | 0,000 | 61,92    | EUR     | 1,000000 | 61,920    | J          | 61,9200 | 61,92    | 0,00      |
| 10      | 10       | 116106    | 9025082 |     | TL-001016-1 | 361   | AH     | 16.12.13 |          | 2,000      | STK | 0,000 | 17,63    | EUR     | 1,000000 | 17,630    | J          | 17,6300 | 35,26    | 0,00      |
| 10      | 10       | 116106    | 9025082 |     | TL-001017-1 | 361   | AH     | 16.12.13 |          | 2,000      | STK | 0,000 | 21,50    | EUR     | 1,000000 | 21,500    | J          | 21,5000 | 43,00    | 0,00      |
| 10      | 10       | 116106    | 9025082 |     | TL-001024-1 | 361   | AH     | 16.12.13 |          | 2,000      | STK | 0,000 | 9,03     | EUR     | 1,000000 | 9,030     | J          | 9,0300  | 18,06    | 0,00      |

## Example 2 – Defining the elements of a stock list

DomainModel = Bestandsliste.

\* Fields in the list

```

Mandant           = name=c(2).
Mandantename      = name=c(22).
Vertriebsbereich  = name=c(20).
VBNR              = name=c(2).
Lieferantenname   = name=c(30).
Artikeltext       = name=c(30).
ARTNR            = name=c(14).
LFNR              = name=c(6).
Partie_Lot        = name=d(11).
Lieferantenkurzname = name=c(10).
Lager             = name=c(3).
EK_Gruppe         = "AH" | "90" | "AB" | "HH" | "54" | "EHM54" | "KW_52" | "BF_54" | "ML_54" | "AH_24" |
                    "52" | "JPK52" | "24" | "DS" | "KW" | "KP" | "MP" | "BF" | "WOE54" | "AH_90" | "AH_GMA" |
                    "25" | "EMH25" | "EHM25" | "EMH56" | "EHM56" | "MAP51" | "MAP" | "HH" | "ML" | "TM" | "CF" | "SH" |
                    "MBS" | "JPK24" | "WRE54" | "AB_90" | "DS_57" | "DS_24" | "KW_54" | "MP_54" | "KP_54" | "TM_25" |
                    "SH_57" | "MAP57" | "HH_57" | "CF_80" | "SJ_54" | "80" | "HH_54" | "EHM" | "MAP54" | "CF_57".

MHD_WE           = name=c(8).
m_verp           = "MT" | "KG" | "STK" | "LTR" | "1SP" | "2BB" | "1BS" | "2SP" | "BU" | "1BB" | "SAP".
Gewichtseinheit   = name=c(2).
EK_Preis         = name=STRING.
Whr              = "EUR" | "USD" | "UDD" | "DEM" | "AUD" | "GBP".
Kurs              = name=c(8).
Einstandspreis    = name=STRING.
Verzollung        = name=c(1).
Bewertung         = name=STRING.
Lagerwert_je_me   = name=STRING.
Lagerbestand      = name=STRING.
freie_menge       = name=STRING.
Lagerwert_gesamt  = name=STRING.
Lagerwert_freie   = name=STRING.
ME               = "MT" | "KG" | "STK" | "LTR" | "MTR".

```

## Example 2 – Defining the structure of a stock list

```

* line of the list
listenzeile      = Partie_Lot ARTNR Lieferantenkurzname Lager [EK_Gruppe] MHD_WE [m_verp] Lagerbestand [ME]
                  freie_menge EK_Preis [Whr] Kurs Einstandspreis Verzollung
                  Lagerwert_je_me Lagerwert_gesamt Lagerwert_freie.

* Summ
summe_artikel    = "Artikel"      ":" ARTNR Artikeltext Lagerbestand freie_menge Lagerwert_gesamt Lagerwert_freie.
summe_lieferant  = "Lieferant"    ":" LFNR Lieferantennamen Lagerbestand freie_menge Lagerwert_gesamt Lagerwert_freie.
summe_vertriebsbereich = "Vertriebsber" ":" VBNR Vertriebsbereich Lagerbestand freie_menge Lagerwert_gesamt Lagerwert_freie.
summe_mandant    = "Mandant"      ":" Mandant Mandantennamen Lagerbestand freie_menge Lagerwert_gesamt Lagerwert_freie.
gesamt_summe     = Lagerbestand freie_menge Lagerwert_gesamt Lagerwert_freie.
gesamt_gruppen   = "Gesamtsumme über alle Gruppen" ":" Lagerbestand freie_menge Lagerwert_gesamt Lagerwert_freie.

* Headers
total            = "T" "O" "I" "A" "L" ":".
mandant          = "Mandant"      ":" Mandant Mandantennamen.
vertriebsbereich = "Vertriebsber" ":" VBNR Vertriebsbereich.
lieferant        = "Lieferant"    ":" LFNR Lieferantennamen.
artikel          = "Artikel"      ":" ARTNR Artikeltext.

Bestandsliste = { [total] [mandant]
                  { vertriebsbereich
                    { lieferant
                      { artikel
                        listenzeile {listenzeile}
                        summe_artikel
                      }
                    }
                    summe_lieferant
                  }
                  summe_vertriebsbereich
                }
                [summe_mandant]
              }.

```

## Example 3 – A SQL QUERY

Model: SQL Query Example 1

```

1 SELECT bsik~bukrs bkp~belnr bsik~lifnr FROM bsik
2 JOIN bkp ON bkp~BUKRS = bsik~bukrs
3     AND bkp~BELNR = bsik~belnr
4     AND bkp~GJAHR = bsik~gjahr
5 WHERE bsik~lifnr = '0000001082'
6 ORDER BY bkp~belnr DESCENDING
    
```

| CoCode | Vendor     | DocumentNo |
|--------|------------|------------|
| 1000   | 1700001200 | 1082       |
| 1000   | 1700001199 | 1082       |
| 1000   | 1700001198 | 1082       |
| 1000   | 1700001197 | 1082       |
| 1000   | 1700001196 | 1082       |
| 1000   | 1700001195 | 1082       |
| 1000   | 1700001194 | 1082       |
| 1000   | 1700001193 | 1082       |
| 1000   | 1700001192 | 1082       |
| 1000   | 1700001191 | 1082       |
| 1000   | 1700001190 | 1082       |
| 1000   | 1700001189 | 1082       |
| 1000   | 1700001188 | 1082       |
| 1000   | 1700001187 | 1082       |
| 1000   | 1700001186 | 1082       |
| 1000   | 1700001185 | 1082       |
| 1000   | 1700001184 | 1082       |
| 1000   | 1700001183 | 1082       |
| 1000   | 1700001182 | 1082       |
| 1000   | 1700001181 | 1082       |
| 1000   | 1700001180 | 1082       |
| 1000   | 1700001170 | 1082       |

SAP | D03 (1) 800 | neustadt | INS

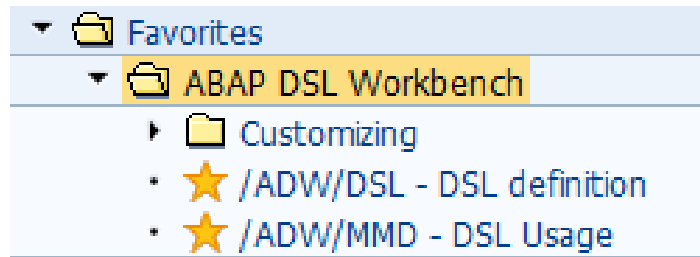
DSL: SQL Query

```

1 * SELECT Statement
2   DomainModel = query [ order_by_clause ].
3
4 * BEGIN Query Specification
5   query = "SELECT" [ "SINGLE" | "DISTINCT" ] select_list from_clause
6         [ where_clause ] [ group_by_clause ] [ having_clause ].
7
8 * SELECT LIST
9   select_list = "*" | derived_column { derived_column }.
10
11 derived_column = column_reference.
12
13 * BEGIN FROM Clause
14   from_clause = "FROM" table_specification [ "UP" "TO" "" name=STRING "" "ROWS" ].
15
16 table_specification = joined_table | table_reference
    
```



**The ABAP DSL Workbench is a toolset, which allows you to define and execute your own DSLs.**



**The ABAP DSL Workbench has the following components:**

- A scanner and parser to define a Domain-Specific Language
- A scanner and general parser for to be able to use the defined DSL
- An editor, with keyword highlighter and keyword completion. You can also set breakpoints to debug the parsing regarding a specific line.
- A general code generator which can be used to generate ABAP code and other development objects.
- The workbench itself is written in ABAP based on SAP Netweaver 7.00 and thus integrates very easily into the ABAP Stack. It is available in English und German.

## Why and when should we create a Domain-Specific Language?

“(…) a DSL is a thin veneer over a model, where the model might be a library or framework.” (see M. Fowler, “Languages and Semantic Model”, p. 16)

- Domain Experts can read the DSL and thus understand what the system thinks it’s doing.
- Change in Execution Context – The same DSL could be used in an ABAP Context as well as in an Java Context.
- Alternative Computational Model – Declarative programming instead of imperative programming.
- (see M. Fowler, “Domain-Specific Languages”, p. 33f.)
- Improvement of Development Productivity - Reduce manual typing by generating infrastructural code, hide a complex API.

**Thank you for your attention! Questions?**

**If you want the slides or if you want to try out the workbench please contact me directly.**

**Please don't forget to fill out the Feedback!**

ADventas Consulting  
Peter Langner  
Kattjahren 8  
22359 Hamburg

Tel. 040 60 55 94 01  
Fax 040 60 55 94 00  
Mobile 0151 12 21 48 67

Peter.Langner@adventas.de  
<http://www.adventas.de>