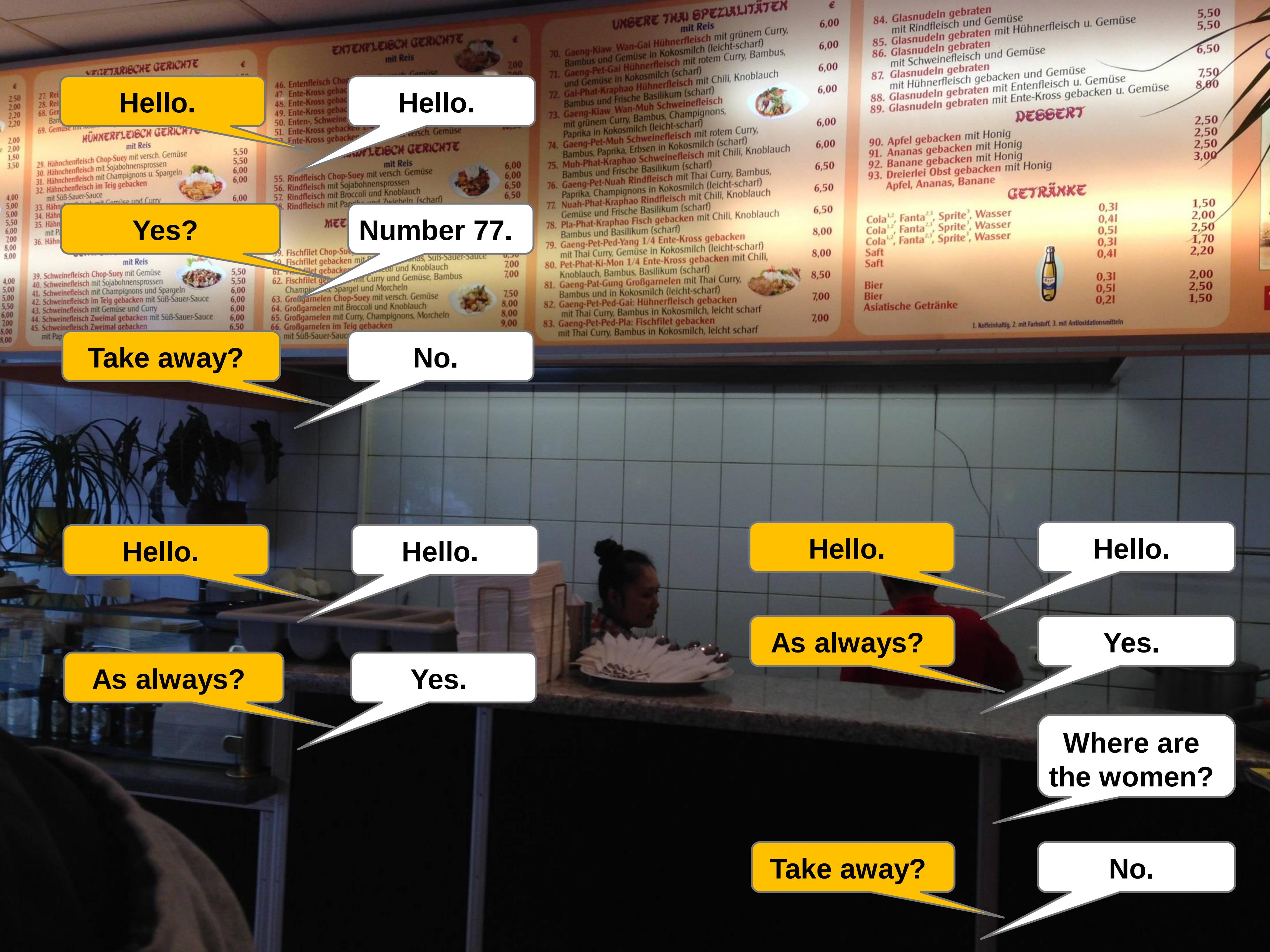
ADventas Consulting

ABAP DSL Workbench

SAP TechED 2016

Barcelona, November 2016

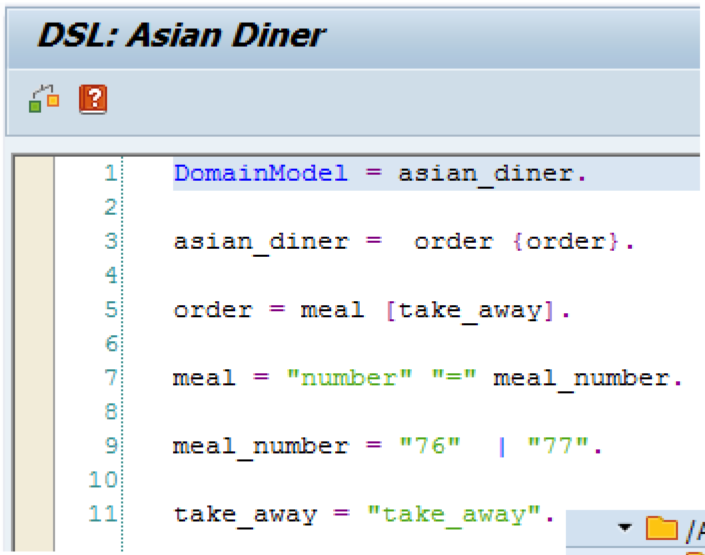

Hello.
Hello.
Yes?
Number 77.
Take away?
No.
Hello.
Hello.
As always?
Yes.
Hello.
Hello.
As always?
Yes.
Where are the women?
Take away?
No.

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)

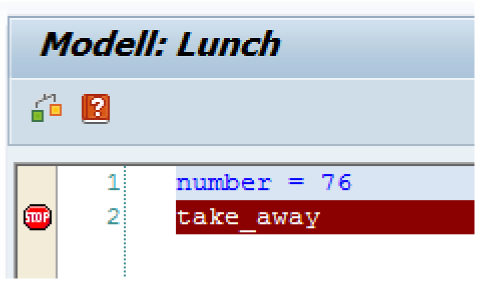
Domain-Specific Language **GPL - General Purpose Language**

ADventas Consulting



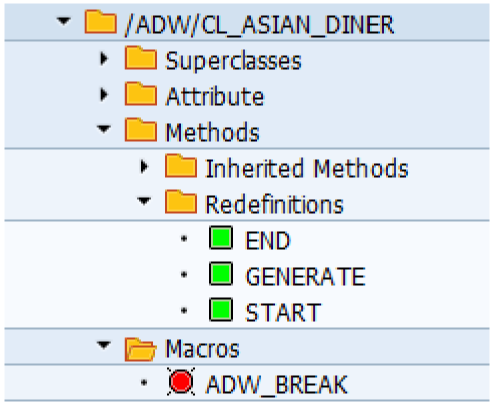
DSL: Asian Diner

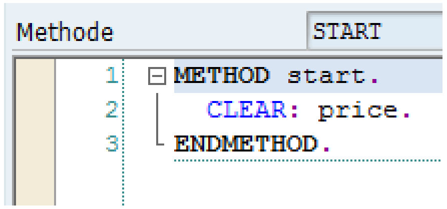
```
DomainModel = asian_diner.

asian_diner =  order {order}.

order = meal [take_away].

meal = "number" "=" meal_number.

meal_number = "76"  | "77".

take_away = "take_away".
```



Modell: Lunch

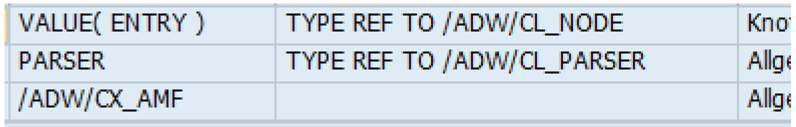
```
number = 76
take_away
```



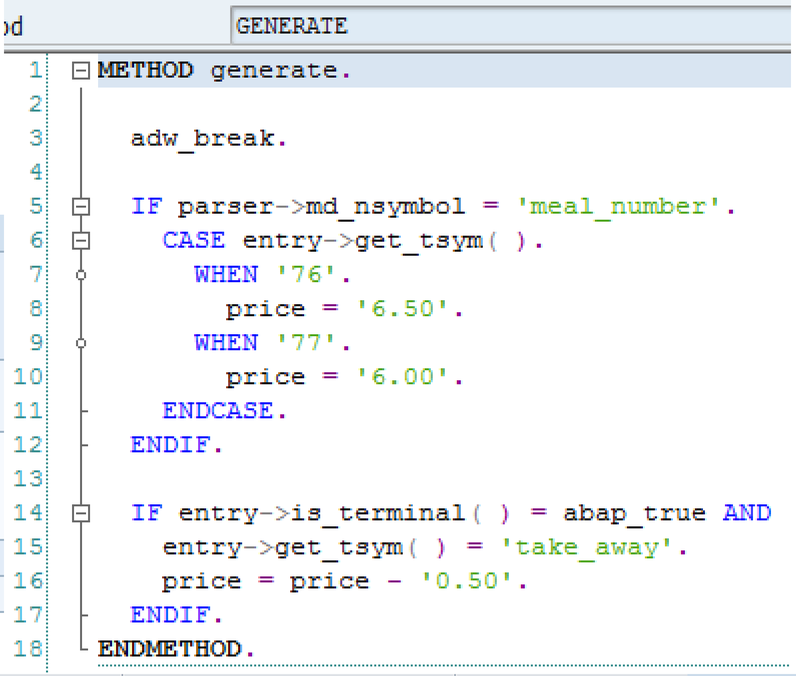


Methode START

```
METHOD start.
  CLEAR: price.
ENDMETHOD.
```

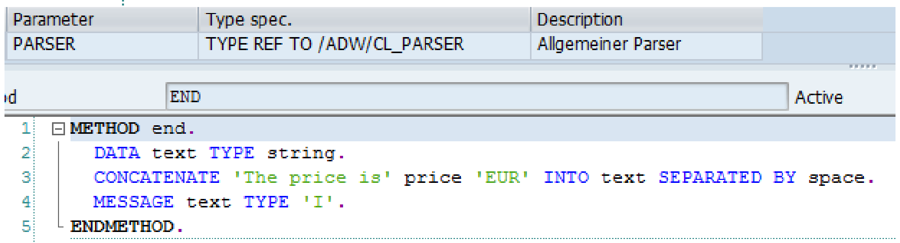


VALUE(ENTRY)	TYPE REF TO /ADW/CL_NODE	Kno
PARSER	TYPE REF TO /ADW/CL_PARSER	Allge
/ADW/CX_AMF		Allge



od GENERATE

```
METHOD generate.

  adw_break.

  IF parser->md_nsymbol = 'meal_number'.
   CASE entry->get_tsym( ).
     WHEN '76'.
       price = '6.50'.
     WHEN '77'.
       price = '6.00'.
   ENDCASE.
  ENDIF.

  IF entry->is_terminal( ) = abap_true AND
   entry->get_tsym( ) = 'take_away'.
   price = price - '0.50'.
  ENDIF.
ENDMETHOD.
```



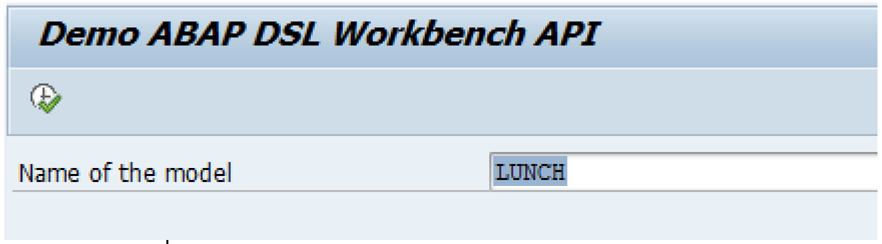
Parameter	Type spec.	Description
PARSER	TYPE REF TO /ADW/CL_PARSER	Allgemeiner Parser

od END Active

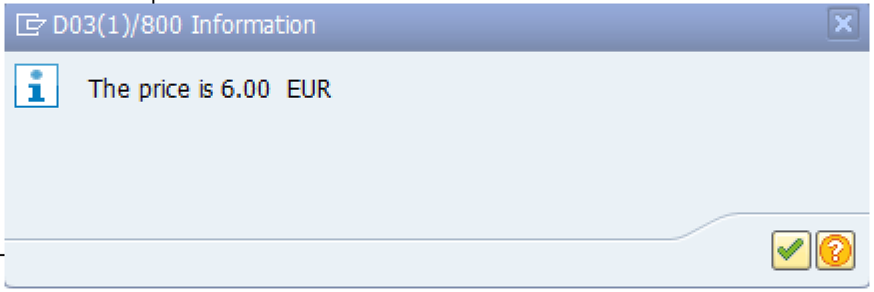
```
METHOD end.
  DATA text TYPE string.
  CONCATENATE 'The price is' price 'EUR' INTO text SEPARATED BY space.
  MESSAGE text TYPE 'I'.
ENDMETHOD.
```

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.
```


Demo ABAP DSL Workbench API

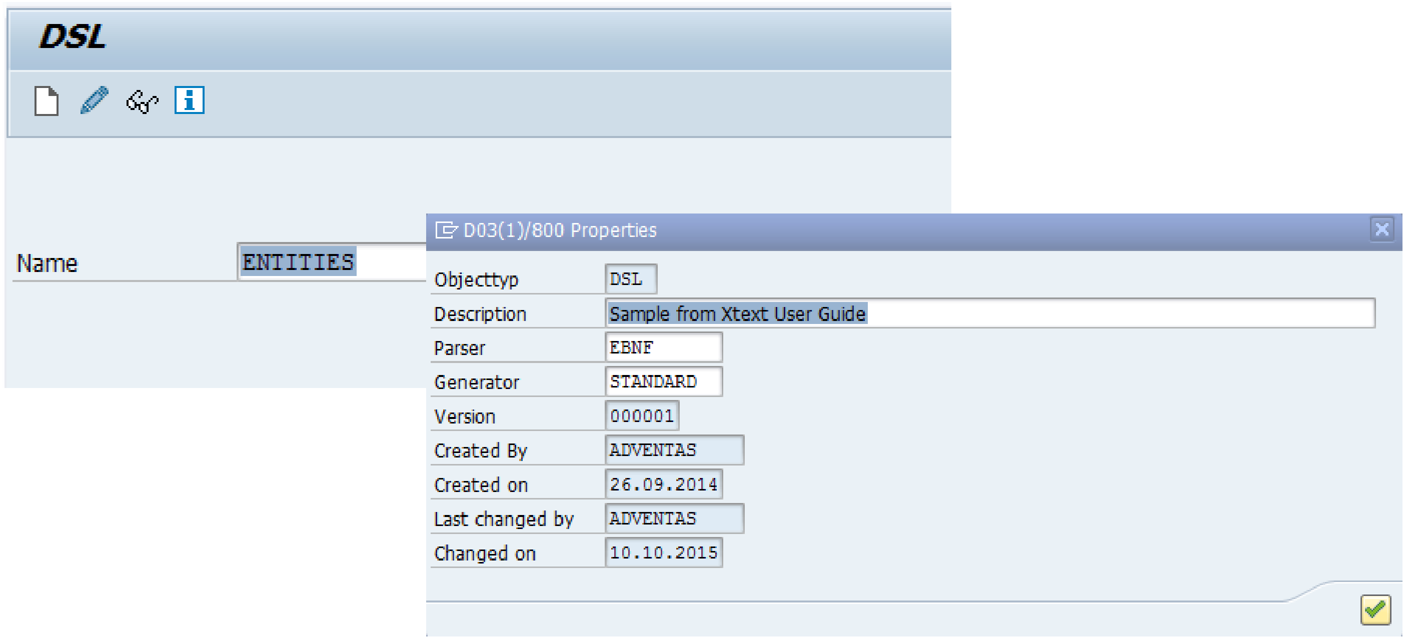
Name of the model LUNCH



The price is 6.00 EUR

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

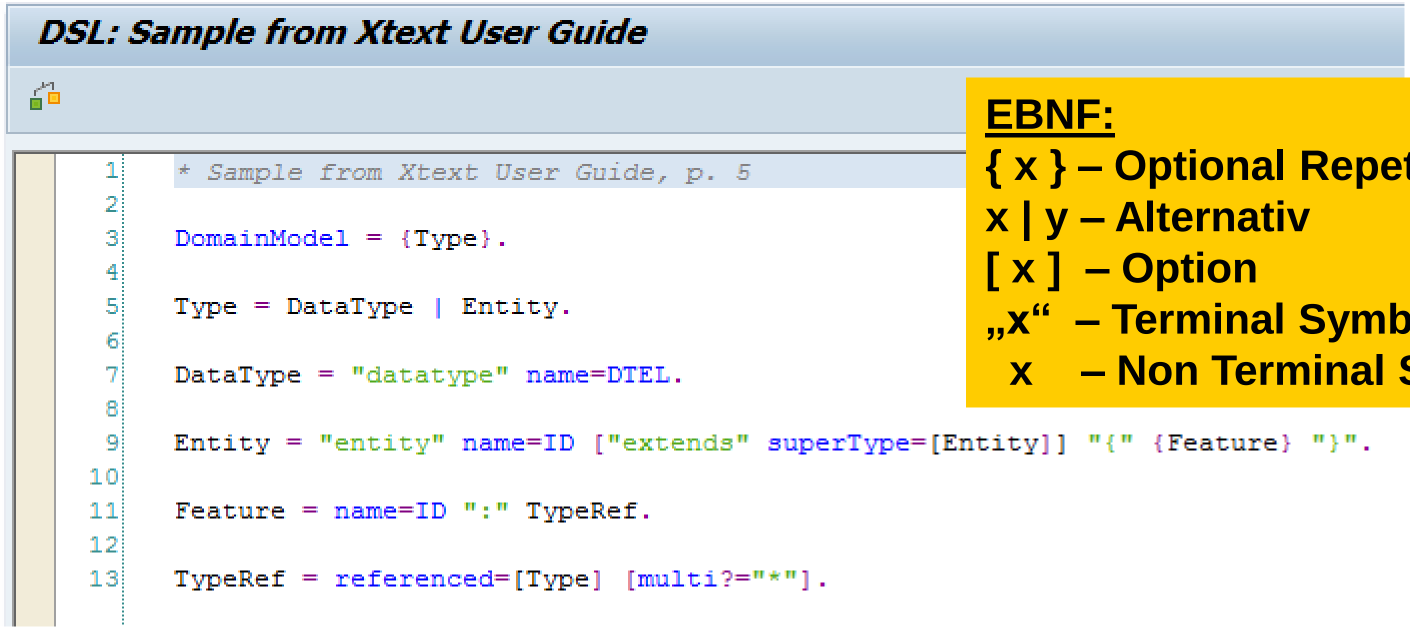
First you have to create a DSL by referring to an EBNF.



Name	ENTITIES
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
Changed on	10.10.2015

- 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

```
* Sample from Xtext User Guide, p. 5

DomainModel = {Type}.

Type = DataType | Entity.

DataType = "datatype" name=DTEL.

Entity = "entity" name=ID ["extends" superType=[Entity]] "{" {Feature} "}".

Feature = name=ID ":" TypeRef.

TypeRef = referenced=[Type] [multi?="*"].
```

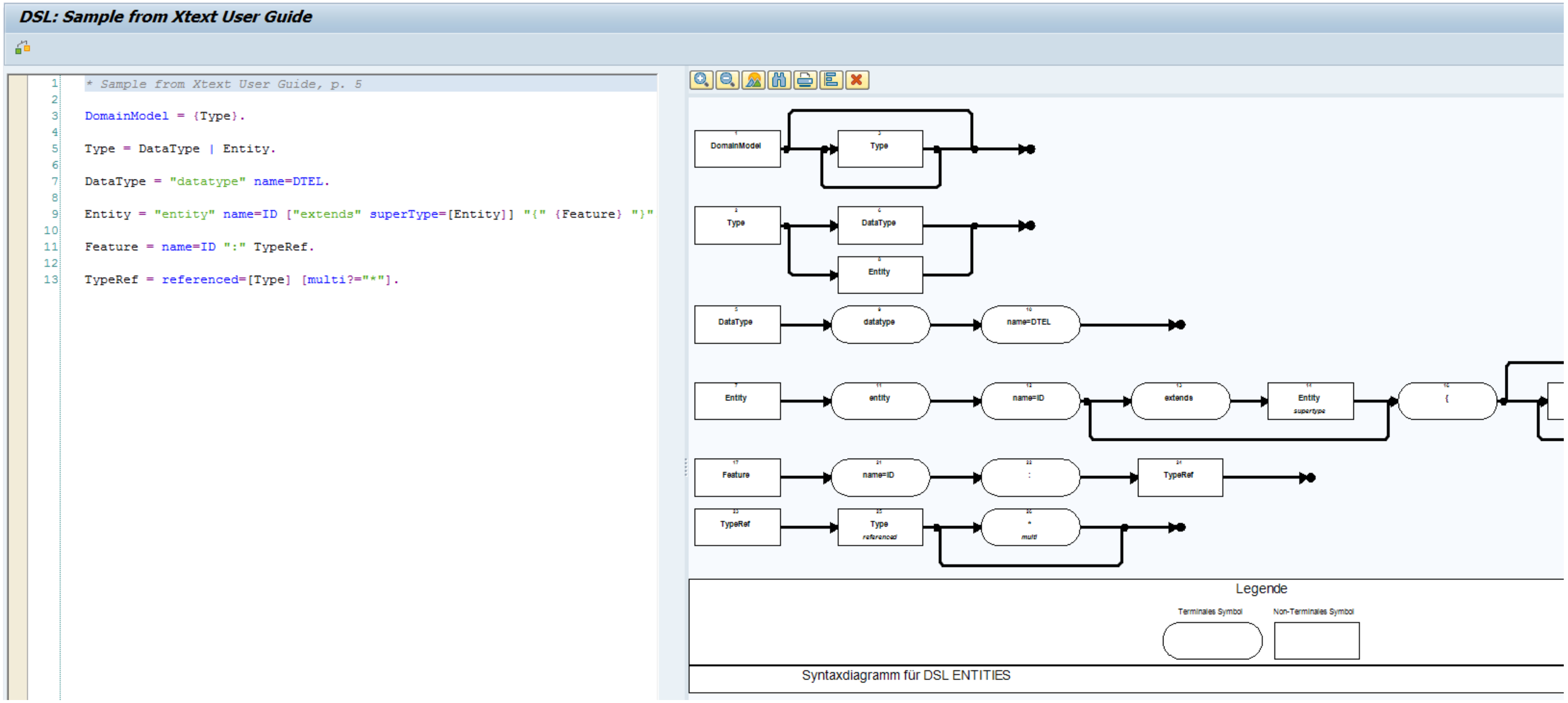
EBNF:
{ x } – Optional Repetition
x | y – Alternativ
[x] – Option
„x“ – Terminal Symbol
x – Non Terminal Symbol

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)*.

ADventas Consulting

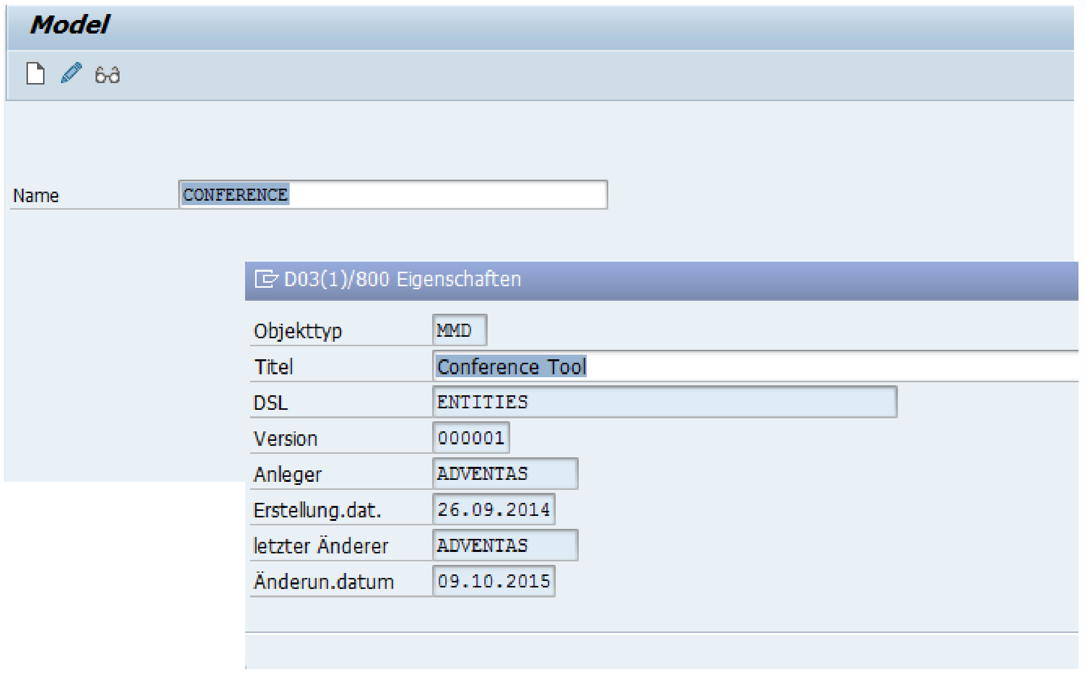
After the syntax check the syntax diagram is shown. It helps you to develop the DSL.



DSL: Sample from Xtext User Guide

```
* Sample from Xtext User Guide, p. 5

DomainModel = {Type}.

Type = DataType | Entity.

DataType = "datatype" name=DTEL.

Entity = "entity" name=ID ["extends" superType=[Entity]] "{" {Feature} "}"

Feature = name=ID ":" TypeRef.

TypeRef = referenced=[Type] [multi?="*"].
```

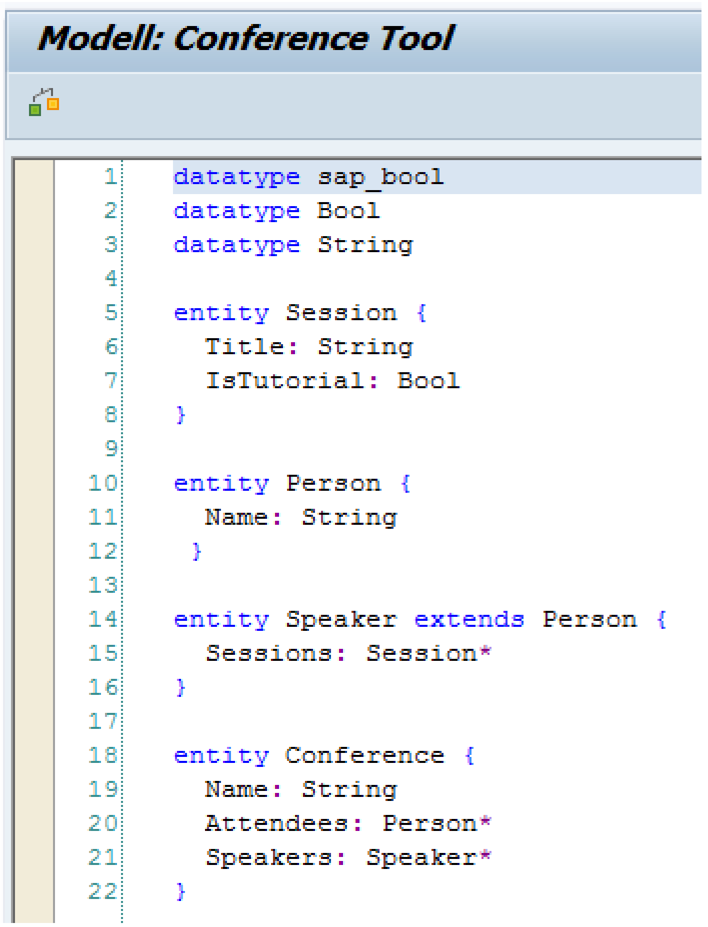
In another transaction you can use the DSL .



Name	CONFERENCE

D03(1)/800 Eigenschaften

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

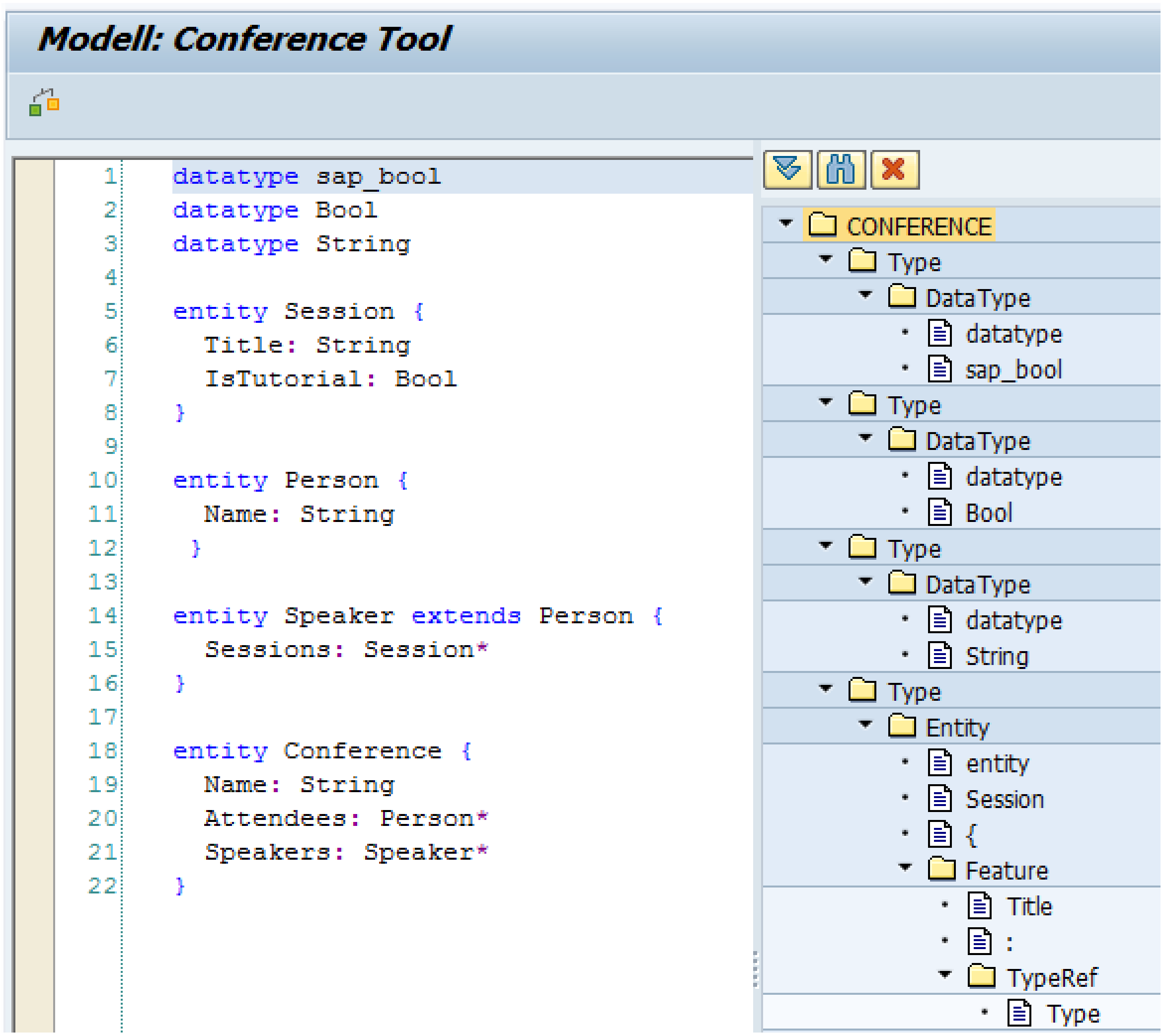


```
datatype sap_bool
datatype Bool
datatype String

entity Session {
  Title: String
  IsTutorial: Bool
}

entity Person {
  Name: String
 }

entity Speaker extends Person {
  Sessions: Session*
}

entity Conference {
  Name: String
  Attendees: Person*
  Speakers: Speaker*
}
```

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

ADventas Consulting

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

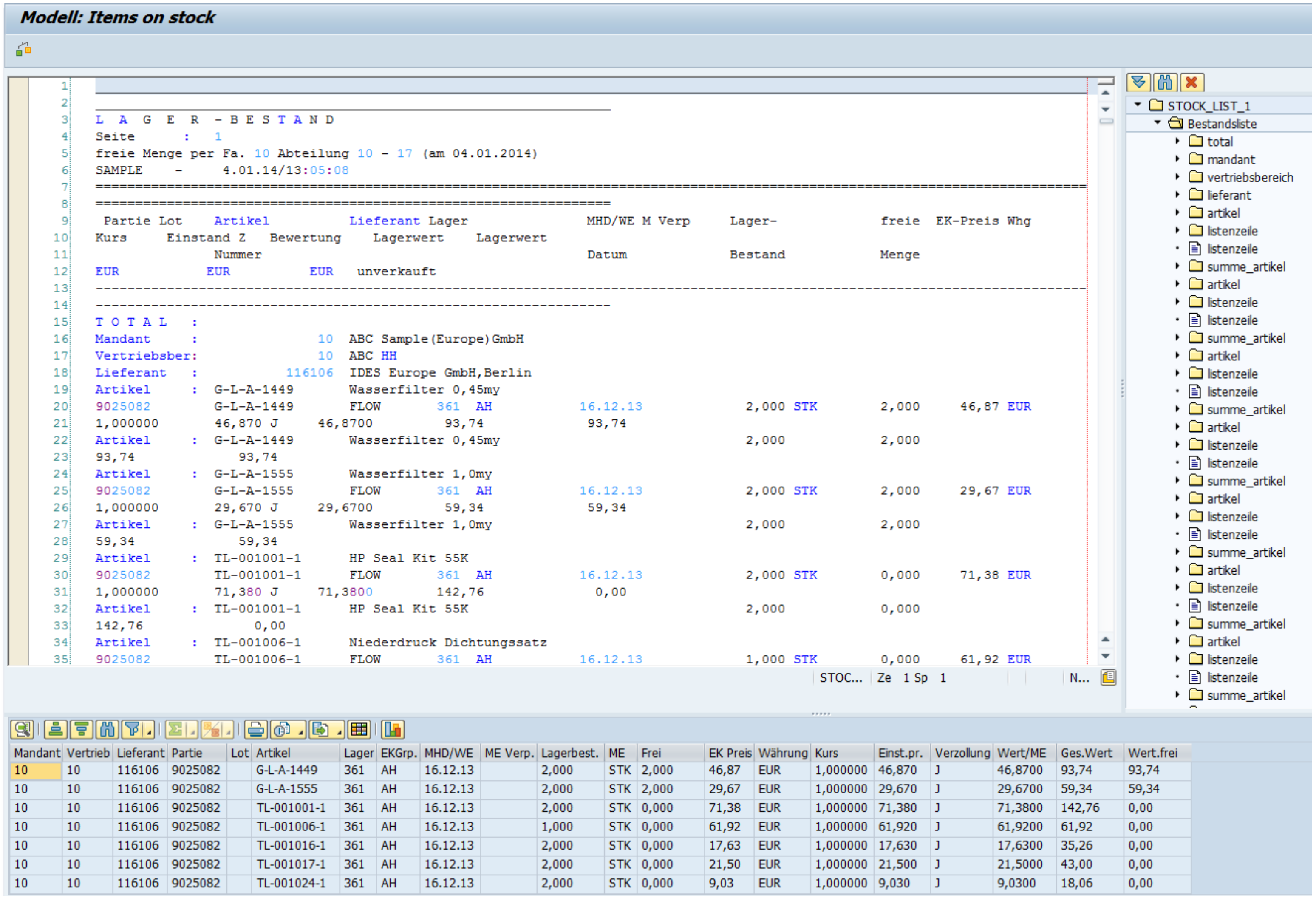

Modell: Conference Tool

```
datatype sap_bool
datatype Bool
datatype String

entity Session {
  Title: String
  IsTutorial: Bool
}

entity Person {
  Name: String
 }

entity Speaker extends Person {
  Sessions: Session*
}

entity Conference {
  Name: String
  Attendees: Person*
  Speakers: Speaker*
}
```

```
CONFERENCE
  Type
    DataType
      datatype
      sap_bool
  Type
    DataType
      datatype
      Bool
  Type
    DataType
      datatype
      String
  Type
    Entity
      entity
      Session
      {
      Feature
        Title
        :
        TypeRef
          Type
```

ADventas Consulting

Example 2 – Converting a stock list



Modell: Items on stock

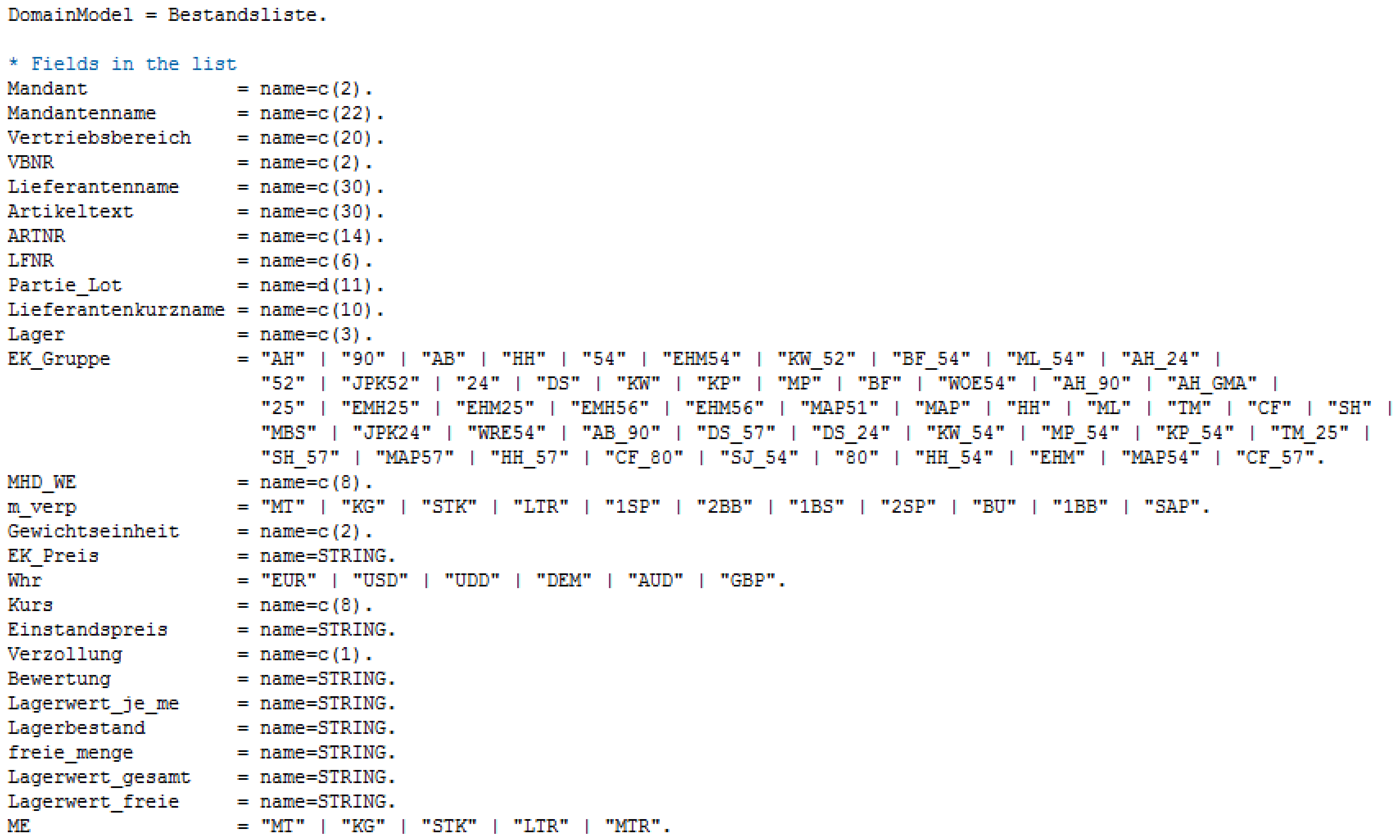
```
____________________________________________________________________
L  A  G  E  R  - B E S T A N D
Seite       :   1
freie Menge per Fa. 10 Abteilung 10 - 17 (am 04.01.2014)
SAMPLE    -     4.01.14/13:05:08
=====================================================================================================================================
====================================================================
 Partie Lot    Artikel          Lieferant Lager                MHD/WE M Verp     Lager-            freie  EK-Preis Whg
Kurs     Einstand Z   Bewertung    Lagerwert    Lagerwert
               Nummer                                          Datum             Bestand           Menge
EUR           EUR          EUR   unverkauft
-------------------------------------------------------------------------------------------------------------------------------------
--------------------------------------------------------------------
T O T A L   :
Mandant     :               10  ABC Sample(Europe)GmbH
Vertriebsber:               10  ABC HH
Lieferant   :           116106  IDES Europe GmbH,Berlin
Artikel     :  G-L-A-1449       Wasserfilter 0,45my
9025082        G-L-A-1449       FLOW       361  AH        16.12.13          2,000 STK        2,000     46,87 EUR
1,000000       46,870 J     46,8700       93,74           93,74
Artikel     :  G-L-A-1449       Wasserfilter 0,45my                         2,000            2,000
93,74             93,74
Artikel     :  G-L-A-1555       Wasserfilter 1,0my
9025082        G-L-A-1555       FLOW       361  AH        16.12.13          2,000 STK        2,000     29,67 EUR
1,000000       29,670 J     29,6700       59,34           59,34
Artikel     :  G-L-A-1555       Wasserfilter 1,0my                          2,000            2,000
59,34             59,34
Artikel     :  TL-001001-1      HP Seal Kit 55K
9025082        TL-001001-1      FLOW       361  AH        16.12.13          2,000 STK        0,000     71,38 EUR
1,000000       71,380 J     71,3800      142,76            0,00
Artikel     :  TL-001001-1      HP Seal Kit 55K                             2,000            0,000
142,76              0,00
Artikel     :  TL-001006-1      Niederdruck Dichtungssatz
9025082        TL-001006-1      FLOW       361  AH        16.12.13          1,000 STK        0,000     61,92 EUR
```

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

- STOCK_LIST_1
  - Bestandsliste
    - total
    - mandant
    - vertriebsbereich
    - lieferant
    - artikel
    - listenzeile
    - listenzeile
    - summe_artikel
    - artikel
    - listenzeile
    - listenzeile
    - summe_artikel
    - artikel
    - listenzeile
    - listenzeile
    - summe_artikel
    - artikel
    - listenzeile
    - listenzeile
    - summe_artikel
    - artikel
    - listenzeile
    - listenzeile
    - summe_artikel
    - artikel
    - listenzeile
    - listenzeile
    - summe_artikel
    - artikel
    - listenzeile
    - listenzeile
    - summe_artikel

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-001006-1	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



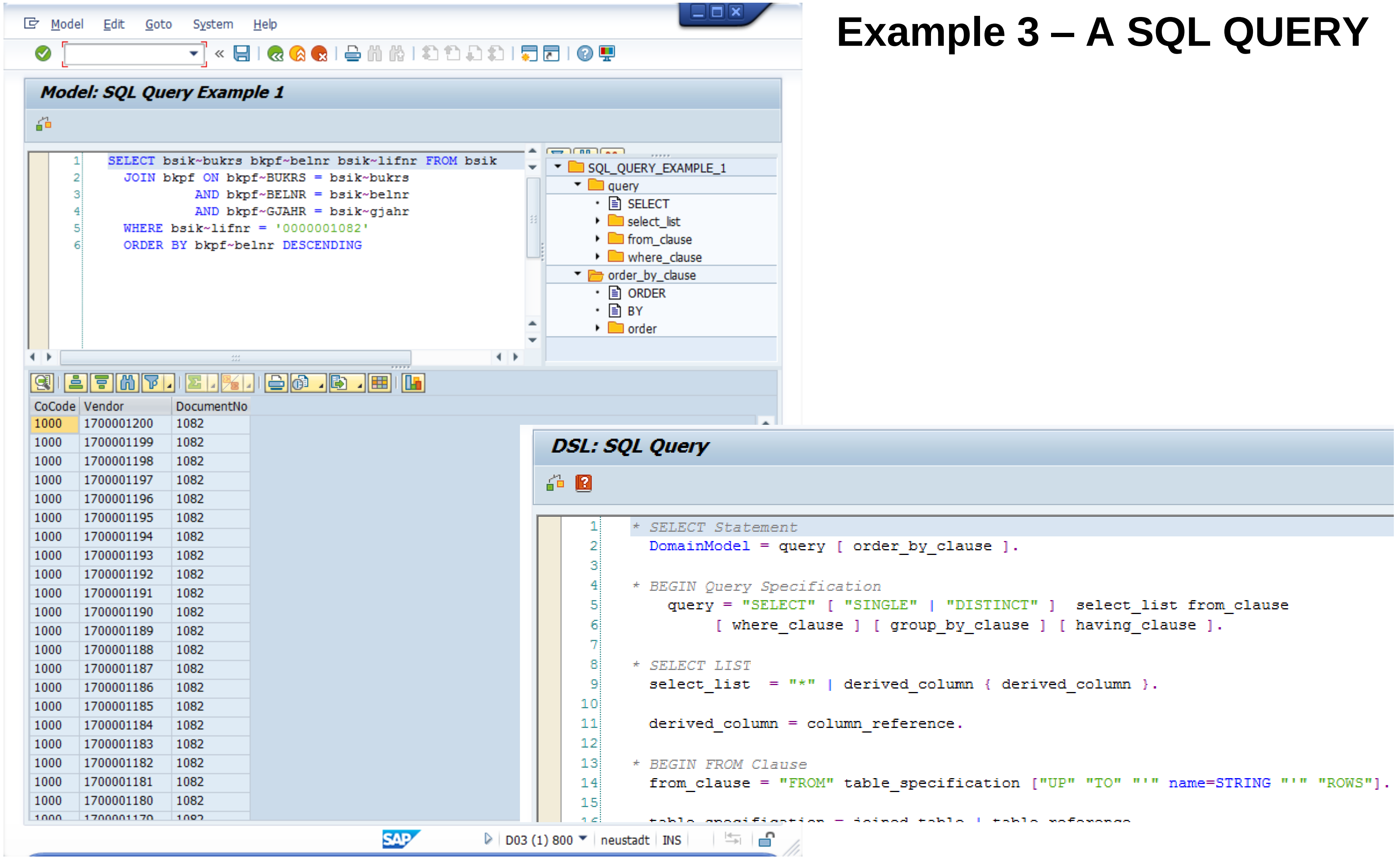
```
DomainModel = Bestandsliste.

* Fields in the list
Mandant              = name=c(2).
Mandantenname        = 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 Lieferantenname Lagerbestand freie_menge Lagerwert_gesamt Lagerwert_freie.
summe_vertriebsbereich = "Vertriebsber" ":" VBNR Vertriebsbereich Lagerbestand freie_menge Lagerwert_gesamt Lagerwert_freie.
summe_mandant          = "Mandant"      ":" Mandant Mandantenname 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" "T" "A" "L" ":".
mandant          = "Mandant"      ":" Mandant Mandantenname.
vertriebsbereich = "Vertriebsber" ":" VBNR  Vertriebsbereich.
lieferant        = "Lieferant"    ":" LFNR  Lieferantenname.
artikel          = "Artikel"      ":" ARTNR Artikeltext.

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

ADventas Consulting

Example 3 – A SQL QUERY



Model: SQL Query Example 1

```
SELECT bsik~bukrs bkpf~belnr bsik~lifnr FROM bsik
  JOIN bkpf ON bkpf~BUKRS = bsik~bukrs
           AND bkpf~BELNR = bsik~belnr
           AND bkpf~GJAHR = bsik~gjahr
  WHERE bsik~lifnr = '0000001082'
  ORDER BY bkpf~belnr DESCENDING
```

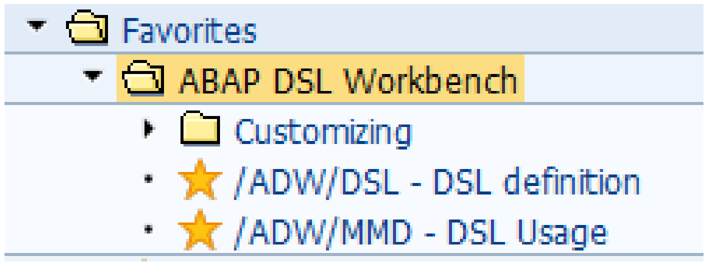
```
SQL_QUERY_EXAMPLE_1
  query
    SELECT
    select_list
    from_clause
    where_clause
  order_by_clause
    ORDER
    BY
    order
```

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	1700001179	1082

DSL: SQL Query

```
* SELECT Statement
  DomainModel = query [ order_by_clause ].

* BEGIN Query Specification
    query = "SELECT" [ "SINGLE" | "DISTINCT" ]  select_list from_clause
         [ where_clause ] [ group_by_clause ] [ having_clause ].

* SELECT LIST
  select_list  = "*" | derived_column { derived_column }.

  derived_column = column_reference.

* BEGIN FROM Clause
  from_clause = "FROM" table_specification ["UP" "TO" "'" name=STRING "'" "ROWS"].
```

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


Favorites
ABAP DSL Workbench
Customizing
/ADW/DSL - DSL definition
/ADW/MMD - DSL Usage

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