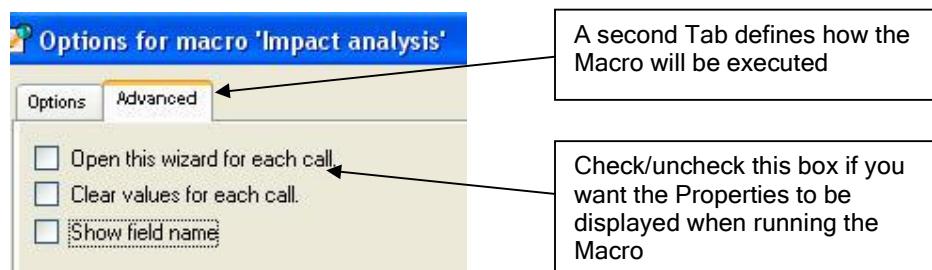
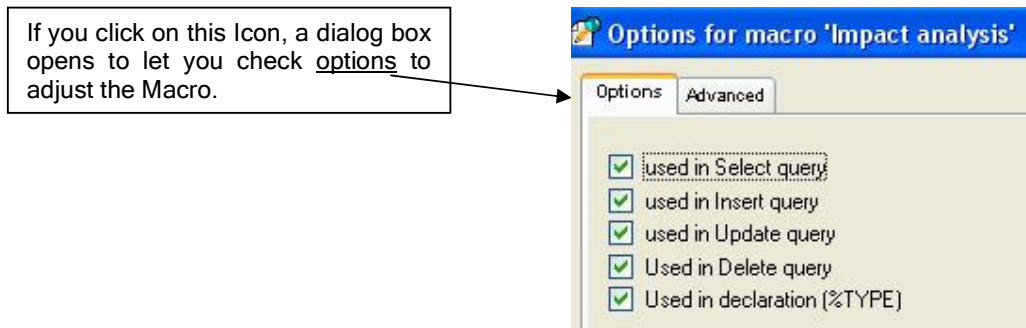
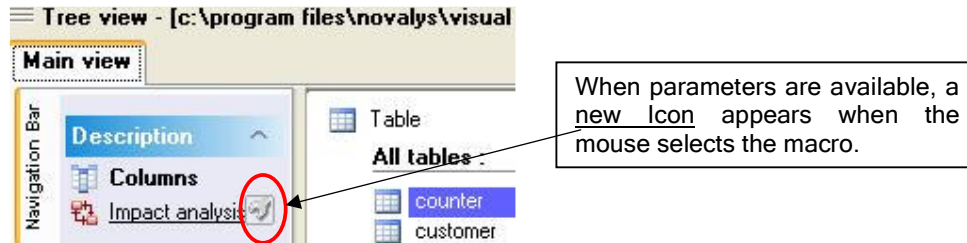


Introduction to Visual Expert 6.0 new features

1. NEW NAVIGATION BAR	2
2. NEW MACROS.....	3
2.1. LESS MACROS, MORE PARAMETERS	3
2.2. MACROS SHORTCUTS IN THE TREEVIEW MENU.....	3
2.3. FILTERING OBJECTS IN THE TREEVIEW.....	4
2.4. FINDING SPECIAL DATAWINDOWS	5
2.5. MAPPING DATAWINDOWS / DBMS	6
2.6. POWERBUILDER ⇔ STORED PROCEDURES DEPENDENCIES.....	6
2.7. INHERITANCE HIERARCHY (WINDOWS/UO/MENUS).....	7
3. NEW IMPACT ANALYSIS	8
3.1. RESULT DISPLAYED AT INSTRUCTION-LEVEL	8
3.2. IMPACT ANALYSIS EXAMPLES	9
3.2.1. <i>Impact Analysis on a TABLE</i>	9
3.2.2. <i>Impact Analysis on a DB COLUMN</i>	10
3.2.3. <i>Impact Analysis on a STORED PROCEDURE</i>	11
4. EXPLORING REFERENCES WITH VISUAL EXPERT 6.0	12
4.1. REFERENCES FOUND IN POWERBUILDER COMPONENTS.....	12
4.1.1. <i>DataWindows</i>	12
4.1.2. <i>PowerScripts</i>	13
4.1.3. <i>RPC FUNC</i>	14
4.2. REFERENCES FOUND IN PL/SQL OR T-SQL COMPONENTS.....	15
4.2.1. <i>Oracle Package</i>	15
4.2.2. <i>Stored Procedures</i>	16
4.2.3. <i>Other DB Code items</i>	17
4.2.4. <i>PL/SQL %TYPE references</i>	17
4.2.5. <i>Parameters & Local Variables</i>	17
5. SOURCE CODE VIEW	18
5.1. HYPERLINKS IN THE CODE	18
5.2. TITLE OF THE SOURCE CODE VIEW	18
5.3. SEARCH IN THE SOURCE CODE VIEW	19
6. MISCELLANEOUS.....	20
6.1. OBJECT PREVIEWS.....	20
6.2. TOOLTIPS.....	21
6.3. "SHORT LIST" VIEW	23
6.4. TECHNICAL REQUIREMENTS	24

1. New navigation bar

With Visual Expert 6.0, some treeview macros include parameters. These parameters let you adjust the result expected from the Macro.



2. New Macros

2.1. Less Macros, more parameters

All treeview macros have been redeveloped and reorganized

A few standard macros are now available for all languages (PB, PL/SQL, and T-SQL):

- Same concept and same Macro name from one language to another
- Each macro uses parameters to cover most needs
- These parameters depend on the type of component selected in the treeview

The screenshot shows the 'Transact-SQL' treeview with the following component lists:

Component list for PowerBuilder :	Count
Datawindow	54 components
PBL	13 components
Application object	1 component
Function object	8 components
Menu	17 components
Window	73 components
UserObject	410 components
Structure	41 components

Oracle PL/SQL :	Count
PL/SQL File	6 components
Package	2 components
Stored procedure	15 components
Trigger	1 component
Cursor	2 components

Transact SQL :	Count
T-SQL File (ASE)	6 components
Stored procedure	9 components

All stored procedures :

- OrdersbyEmployee
- Productsbycustomer
- on_customer_list

Callouts from the left sidebar:

- Definition:** details about the selected component (methods, variables, ancestor, SQL query, parameters...)
- References:** lists the items **called by** (referenced by) the selected component
- Impact Analysis:** lists the items **calling** (referencing) the selected component
- Called Hierarchy:** chain of methods calling each other in the application

2.2. Macros shortcuts in the Treeview Menu

The pop-menu in the treeview (right-click on an item in the treeview) now includes the main macros available for the selected item – no need to go to the navigation bar.

The screenshot shows a treeview with a right-click context menu open. The menu items are:

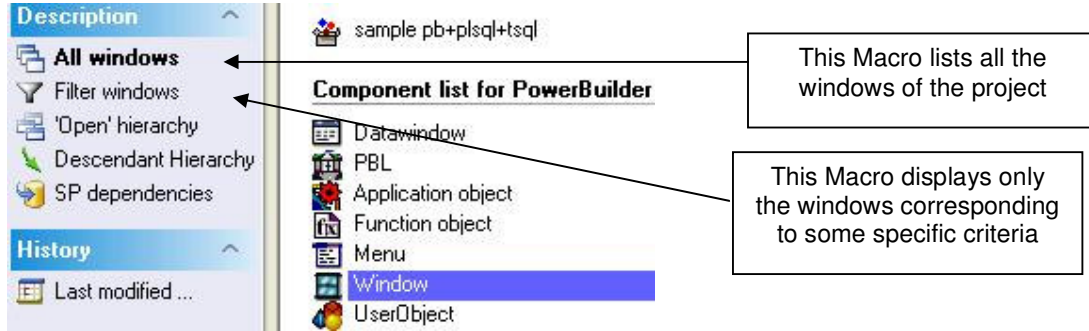
- DW Definition [+Ctrl to open Wizard]
- Impact analysis
- DB Columns
- Preview
- Locate
- Open in PB IDE
- Copy item name
- Select all

A callout box points to the 'DW Definition' macro with the text: "Main macros available in the treeview Pop-menu"

2.3. Filtering objects in the treeview

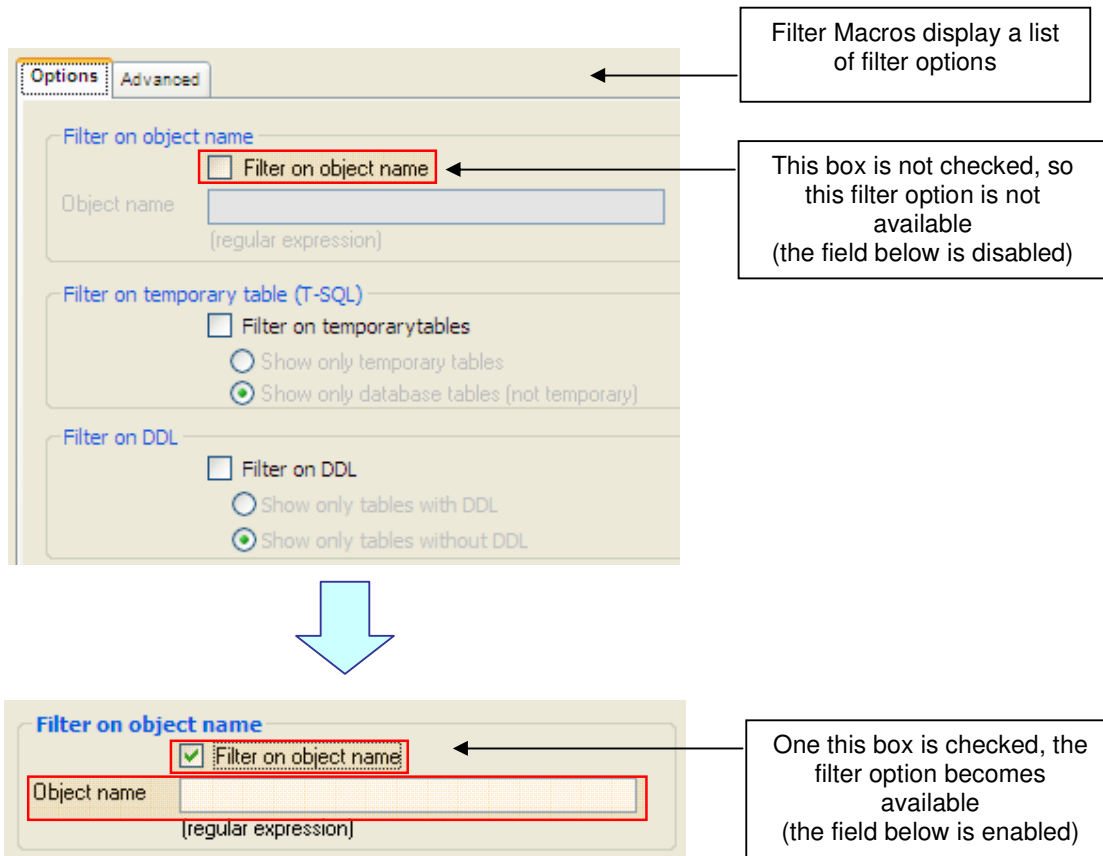
Objects can now be listed in the treeview with 2 macros:

- The macro “**All <Object>**” lists all the objects of this type.
This macro is executed by default with a double-click at the root of the treeview.
- The macro “**Filter <Object>**” offers various parameters to filter each type of object.



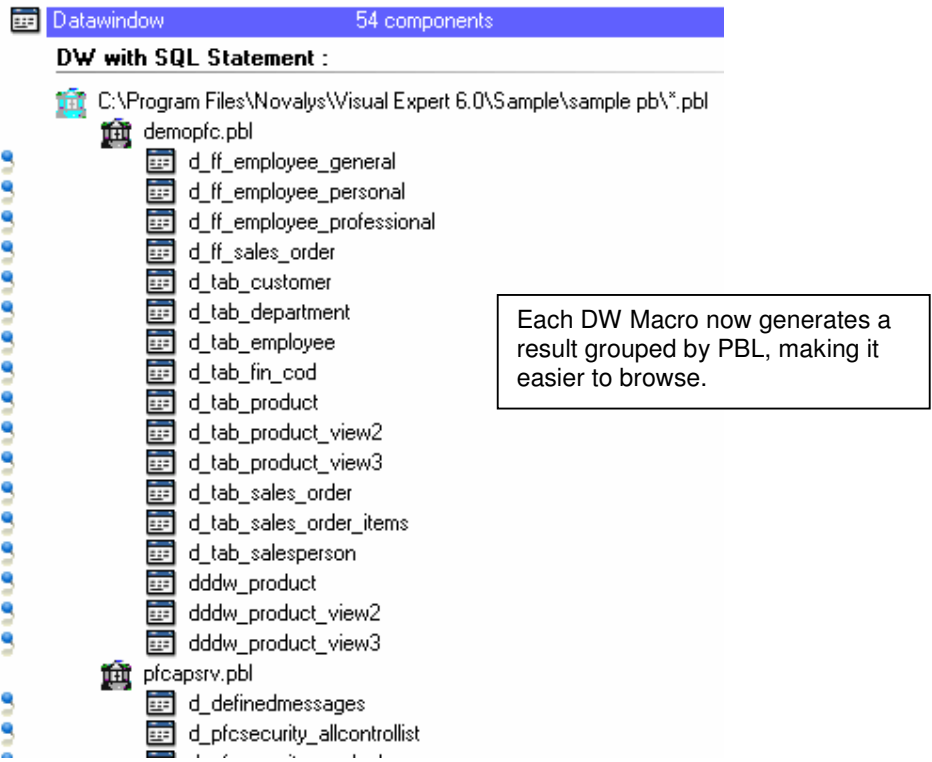
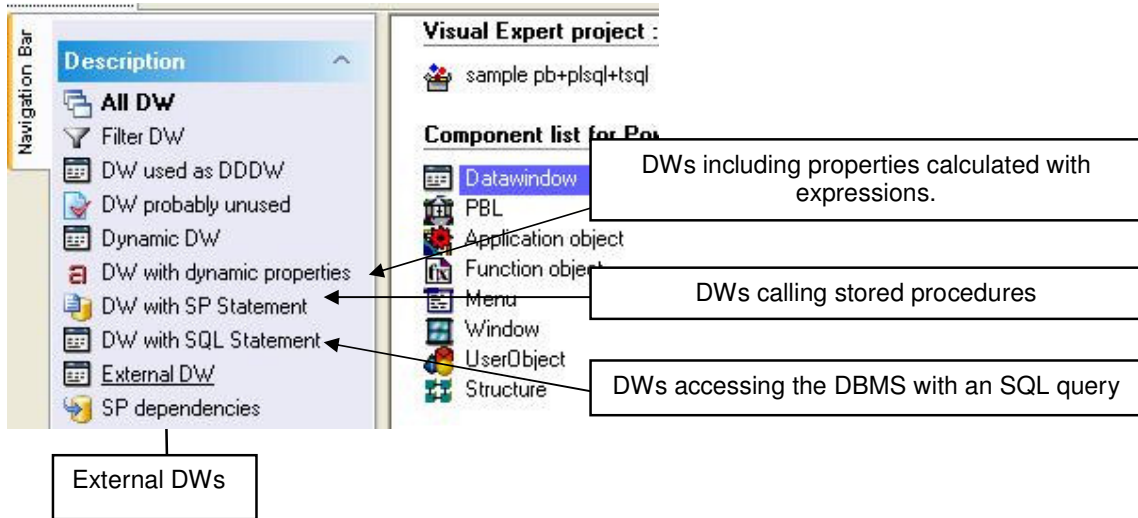
Each filter macro offers a list of filter options.

By default, these options are not activated. You can activate them by checking the corresponding box:



2.4. Finding Special DataWindows

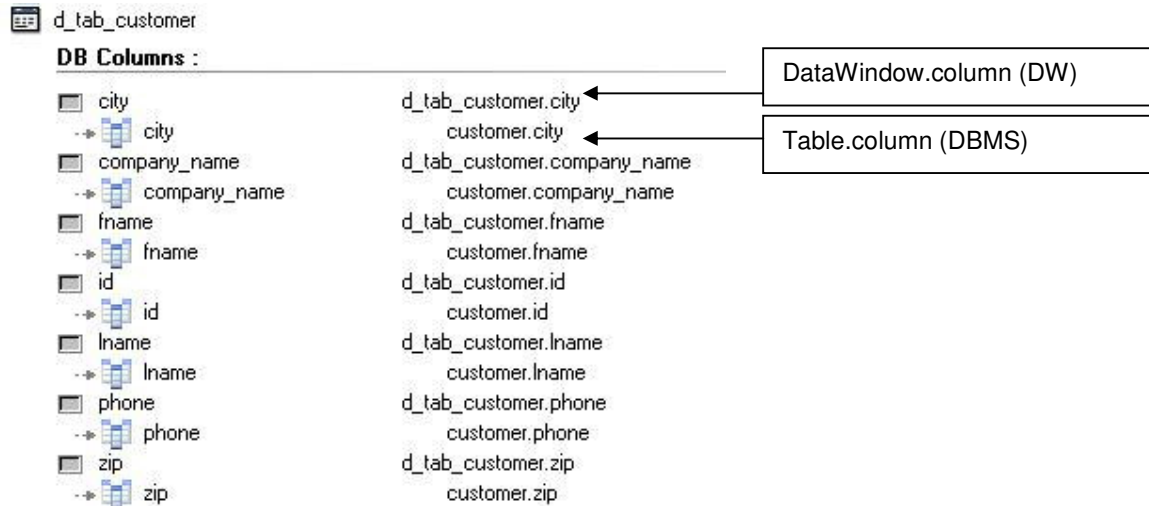
Several new macros have been added with Version 6.0, available at the root of the DWs:



2.5. Mapping DataWindows / DBMS

Visual Expert 6.0 also includes a new “DB columns” macro for DataWindows.

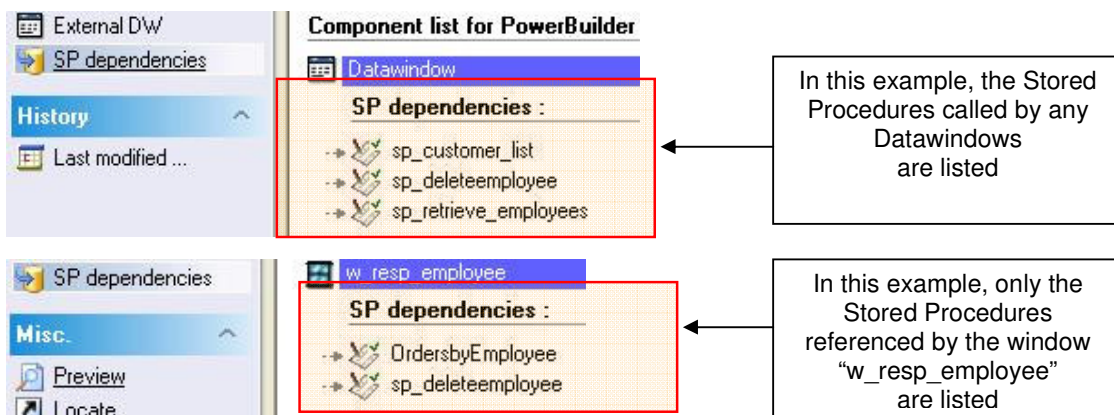
This macro displays the DW- DBMS mapping: it shows which table.column in the DBMS corresponds to each DW column.



2.6. PowerBuilder ↔ Stored Procedures dependencies

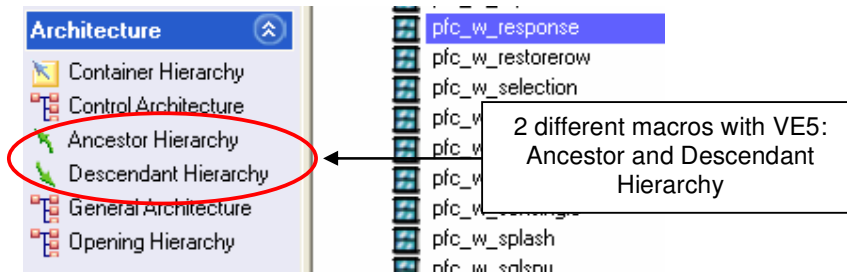
A new Macro called « SP dependencies » lists all stored procedures referenced by PB objects:

- This macro is available at the root of the treeview for each type of PB components
- It is also available for PB objects displayed in the treeview

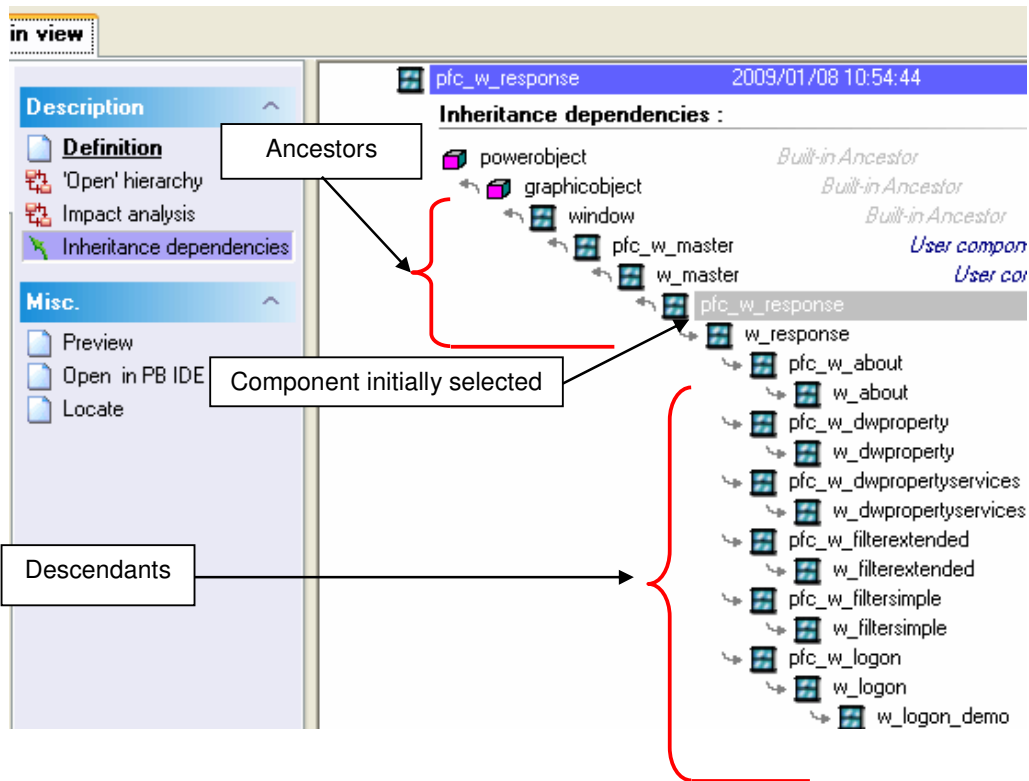


2.7. Inheritance hierarchy (Windows/UO/Menus)

Visual Expert 5.x offered 2 different macros: Descendants + Ancestors



With Visual Expert 6.0, the same Macro displays **both** Ancestors and Descendants:



3. New Impact Analysis

3.1. Result displayed at instruction-level

Visual Expert used to perform impact Analysis across the entire application and provide a component-level result in the treeview. Visual Expert 6.0 now provides a fine-grain result at instruction level!

This icon means that this item belongs to the Impact Analysis result

This symbol means that this event is referencing the table "Customer"

This indicates how many references were found in this code, and the number the reference currently highlighted

customer
Impact analysis :
dempfc
d_ff_sales_order
Data Source
d_tab_customer
Data Source
d_tab_sales_order
Data Source
Pack_DataManager.plsql
DataManager
OrdersbyEmployee
SQL c1
Productsbycustome
SQL c1
sp_customer_list.sql
sp_customer_list
C:\Program Files\Novalys\Vis
DataManager.tsq
{Create procedure
OrdersbyEmpl
SQL c1
{Create procedure
Productsbycu:
SQL c1
en_customer_list.sql

```
39      return uf_GetUpperCase(lname) ||  
40  end uf_GetFullname;  
41  
42  
43  procedure OrdersbyEmployee( Employee_I  
44  is  
45  Emp_fname  varchar2;  
46  Emp_lname  varchar2;  
47  
48  cursor c1 is  
49  SELECT sales_order.id Oid,  
50         sales_order.order_date Od  
51         DECODE(sales_order.fin_co  
52         'r1', 'Revenue', 'e1',  
53         sales_order.region Oregio  
54         customer.fname Cfname,  
55         customer.lname Clname,  
56         sum (sales_order_items.qu  
57  FROM customer,  
58         employee,  
59         product,  
60         sales_order_items,  
61         sales_order  
62  WHERE ( employee.emp_id = Employee  
63         ( employee.emp_id = sales  
64         ( customer.id = sales_ord  
65         ( sales_order_items.id =  
66         (product.id = sales_order  
67  GROUP BY Oid,Odate,  
68         sales_order.fin_code_id,
```

Find | references: 1 of 12

54 55 57 64 70 129 133 134 143 147 151 152

Each line including a reference are marked with this symbol.

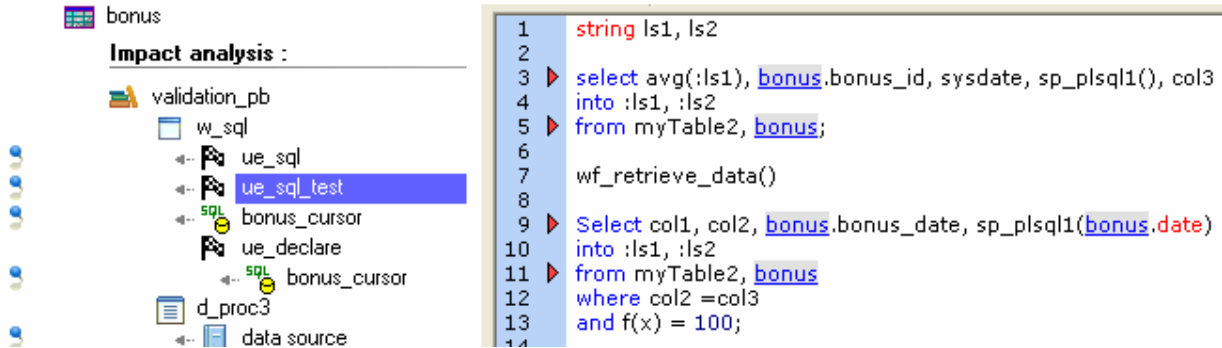
This index lists all the line of code with a reference. If you click on a number, the source code will scroll down to display this line.

You can navigate from one reference to the next by pressing [Enter]. Pressing [Shift+Enter] will take you back to the previous reference. The line with the current reference is highlighted in Yellow.

3.2. Impact Analysis Examples

3.2.1. Impact Analysis on a TABLE

Example 1: Table « bonus » used in a **PowerBuilder Script**



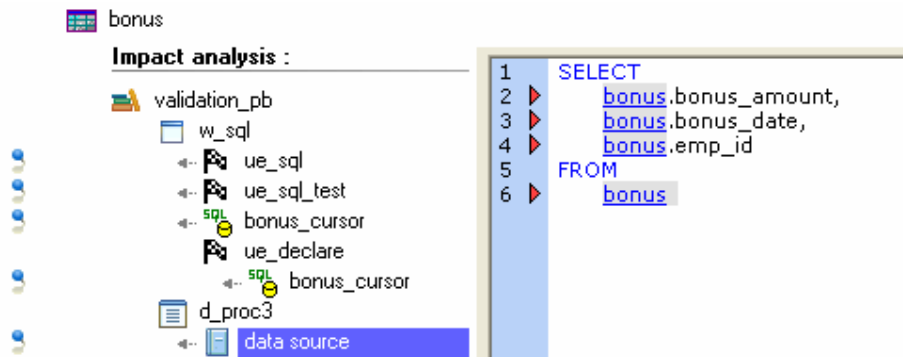
bonus

Impact analysis :

- validation_pb
 - w_sql
 - ue_sql
 - ue_sql_test**
 - bonus_cursor
 - ue_declare
 - bonus_cursor
 - d_proc3
 - data source

```
1 string ls1, ls2
2
3 ▶ select avg(:ls1), bonus.bonus_id, sysdate, sp_plsql1(), col3
4 into :ls1, :ls2
5 ▶ from myTable2, bonus;
6
7 wf_retrieve_data()
8
9 ▶ Select col1, col2, bonus.bonus_date, sp_plsql1(bonus.date)
10 into :ls1, :ls2
11 ▶ from myTable2, bonus
12 where col2 =col3
13 and f(x) = 100;
14
```

Example 2: Table « bonus » used in a **PowerBuilder Datawindow**



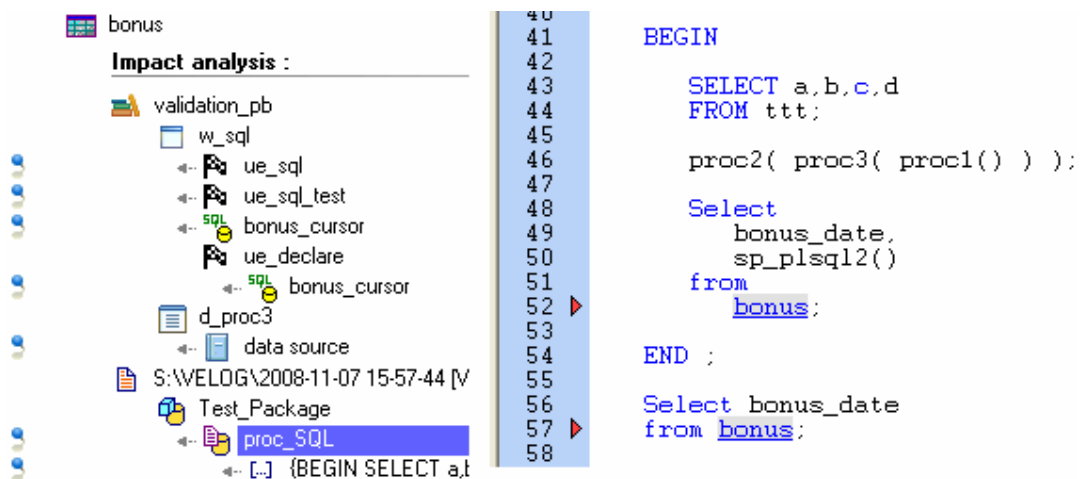
bonus

Impact analysis :

- validation_pb
 - w_sql
 - ue_sql
 - ue_sql_test
 - bonus_cursor
 - ue_declare
 - bonus_cursor
 - d_proc3
 - data source**

```
1 SELECT
2 ▶ bonus.bonus_amount,
3 ▶ bonus.bonus_date,
4 ▶ bonus.emp_id
5 FROM
6 ▶ bonus
```

Example 3: Table « bonus » used in a **Stored Procedure**



bonus

Impact analysis :

- validation_pb
 - w_sql
 - ue_sql
 - ue_sql_test
 - bonus_cursor
 - ue_declare
 - bonus_cursor
 - d_proc3
 - data source
 - S:\VELOG\2008-11-07 15-57-44 [V]
 - Test_Package
 - proc_SQL**

```
40
41 BEGIN
42
43 SELECT a,b,c,d
44 FROM ttt;
45
46 proc2( proc3( proc1() ) );
47
48 Select
49 bonus_date,
50 sp_plsql2()
51 from
52 ▶ bonus;
53
54 END ;
55
56 Select bonus_date
57 ▶ from bonus;
58
```

3.2.2. Impact Analysis on a DB COLUMN

Example 1: Column « bonus.bonus_date » used in a **PowerBuilder Script**

The screenshot shows the PowerBuilder IDE. On the left, the 'Columns' pane lists 'bonus_amount' and 'bonus_date'. Below it, the 'Impact analysis' pane shows a tree view where 'ue_sql_test' is selected. On the right, a code editor displays the following SQL script:

```

1  string ls1, ls2
2
3  select avg(:ls1), bonus.bonus_id, sysdat
4  into :ls1, :ls2
5  from myTable2, bonus;
6
7  wf_retrieve_data()
8
9  ▶ Select col1, col2, bonus.bonus_date, sp_
10 into :ls1, :ls2
11 from myTable2, bonus
12 where col2 = col3
13 and f(x) = 100;
14
15 ls2 = ls1
16
17 ▶ Select bonus_date, sp_plsql2()
18 into :ls1, :ls2
19 from bonus;

```

Example 2: Column « bonus.bonus_date » used in a **PowerBuilder Datawindow**

The screenshot shows the PowerBuilder IDE. On the left, the 'Columns' pane lists 'bonus_amount' and 'bonus_date'. Below it, the 'Impact analysis' pane shows a tree view where 'data source' is selected. On the right, a code editor displays the following SQL query:

```

1  SELECT
2  bonus.bonus_amount,
3  ▶ bonus.bonus_date,
4  bonus.emp_id
5  FROM
6  bonus

```

Example 3: Column « bonus.bonus_date » used in a **Stored Procedure**

The screenshot shows the PowerBuilder IDE. On the left, the 'Columns' pane lists 'bonus_amount' and 'bonus_date'. Below it, the 'Impact analysis' pane shows a tree view where 'proc_SQL' is selected. On the right, a code editor displays the following stored procedure definition:

```

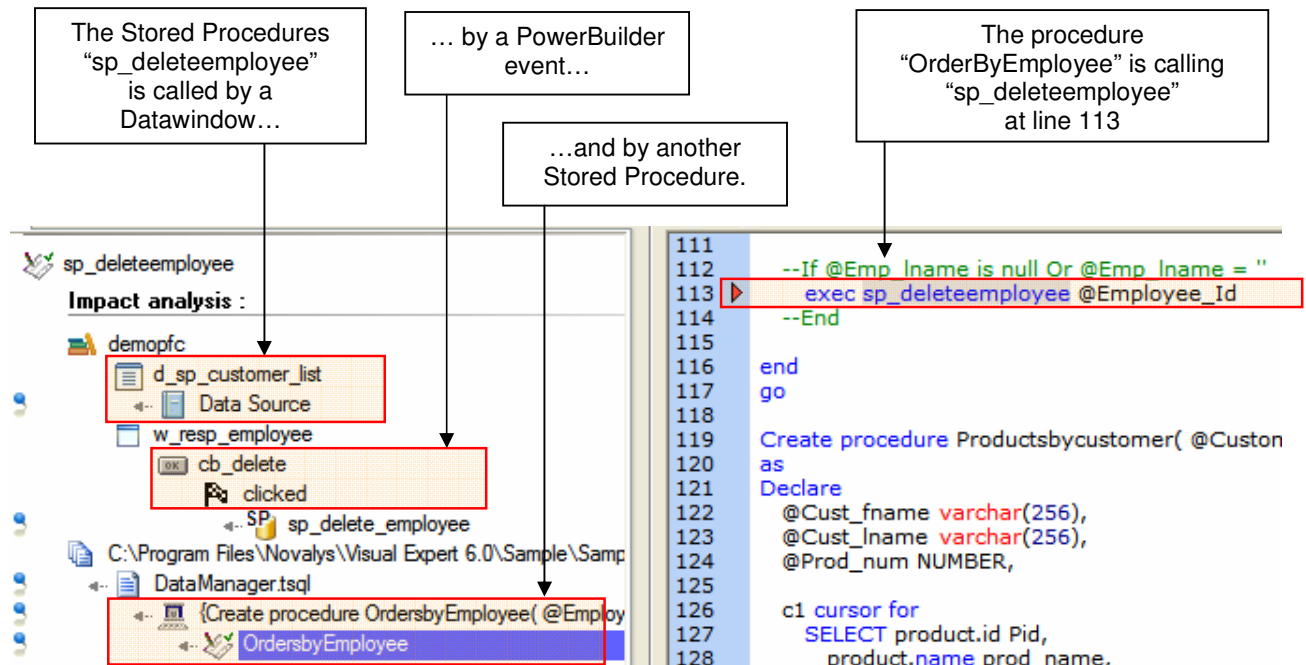
40
41  BEGIN
42
43  SELECT a,b,c,d
44  FROM ttt;
45
46  proc2( proc3( p
47
48  Select
49  ▶ bonus_date,
50  sp_plsql2()
51  from
52  bonus;
53
54  END ;
55
56  ▶ Select bonus_date
57  from bonus;
58

```

3.2.3. Impact Analysis on a STORED PROCEDURE

Example: Impact Analysis for the stored procedure « sp_deleteemployee »:

Visual Expert will find all references to stored procedures and display a result at instruction-level:

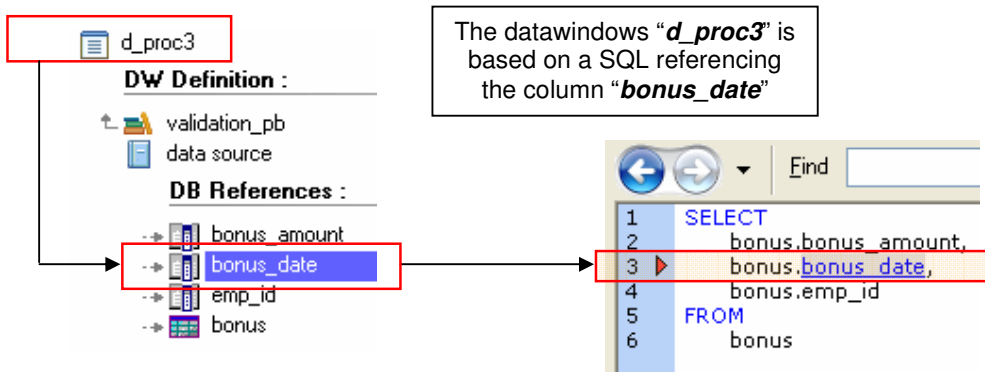


4. Exploring references with Visual Expert 6.0

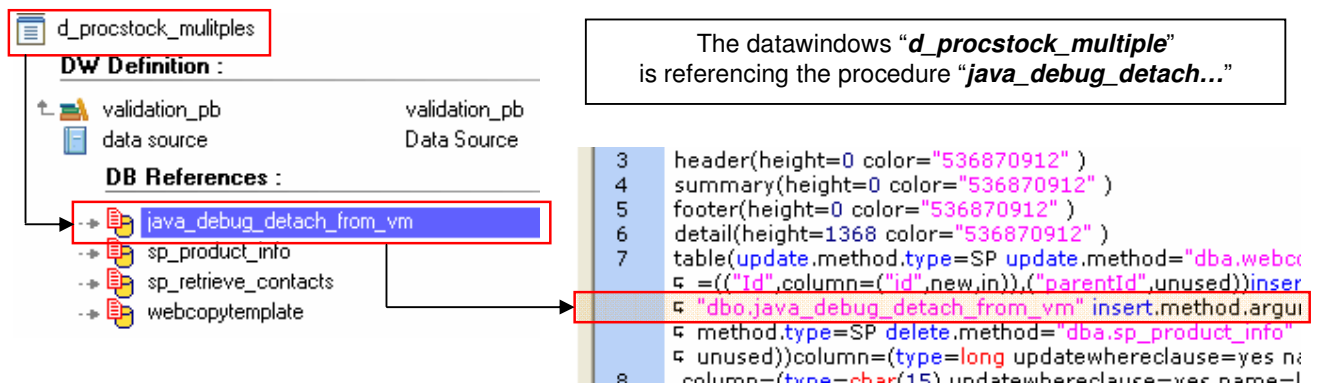
4.1. References found in PowerBuilder components

4.1.1. DataWindows

When a DataWindow is based on a SQL DataSource, the table and columns referenced are listed in the treeview and highlighted in the source:



Both « update » and « select » procedures are displayed as referenced by the DataWindow:



4.1.2. PowerScripts

All items referenced by a Script are listed in the treeview and highlighted in the code.

The event **"ue_sql_test"** is referencing the column **"bonus_date"**...

References:

- bonus_date
- bonus_id
- col1
- col2
- col3
- date
- x
- f
- sp_plsql1
- sp_plsql2
- SQL sql_1
- SQL sql_2
- SQL sql_3
- bonus
- myTable2

```

9  ▶ Select col1, col2, bonus.bonus_date, sp_plsql1()
10 into :ls1, :ls2
11 from myTable2, bonus
12 where col2 = col3
13 and f(x) = 100;
14
15 ls2 = ls1
16
17 ▶ Select bonus_date, sp_plsql2()
18 into :ls1, :ls2
19 from bonus;

```

...as well as the Stored Procedure **"sp_plsql1"**...

References:

- bonus_date
- bonus_id
- col1
- col2
- col3
- date
- x
- f
- sp_plsql1
- sp_plsql2

```

3  ▶ select avg(:ls1), bonus.bonus_id, sysdate,
   4  sp_plsql1(), col3
   5  into :ls1, :ls2
   6  from myTable2, bonus;
   7  wf_retrieve_data()
   8
   9  ▶ Select col1, col2, bonus.bonus_date,
   10 sp_plsql1(bonus.date)
   11 into :ls1, :ls2
   12 from myTable2, bonus
   13 where col2 = col3
   14 and f(x) = 100;

```

...and the local variable **"ls1"**

References:

- ue_sql
- ue_sql_test
- ls1
- ls2

```

3  ▶ select avg(:ls1), bonus.bonus_id, sysdate, sp_plsql1()
4  ▶ into :ls1, :ls2
5  from myTable2, bonus;
6
7  wf_retrieve_data()
8
9  Select col1, col2, bonus.bonus_date, sp_plsql1(bonus.date)
10 ▶ into :ls1, :ls2
11 from myTable2, bonus
12 where col2 = col3
13 and f(x) = 100;
14
15 ▶ ls2 = ls1
16
17 Select bonus_date, sp_plsql2()
18 ▶ into :ls1, :ls2
19 from bonus;

```

4.1.3. RPC FUNC

An RPCFUNC declaration is another solution for PowerBuilder to use a stored procedure.

This declaration is similar to the declaration of an external dll function. The RPCFUNC keyword indicates that a stored procedure is called (and not a dll function).

Visual Expert considers RPCFUNC methods like any other method of the component:

When an RPCFUNC method or dll is selected, Visual Expert displays its declaration:

w_rpcfunc_test	25	function string my_plsql_proc1 (string toto) rpcfunc alias for sp_plsql1
Definition :	26	function string my_plsql_proc2 (string toto) rpcfunc alias for sp_plsql2
	27	
validation_pb	28	// déclarations de fonctions externes de DLL
activate	29	//
clicked	30	▶ function string my_dll_func1 (string toto1, string toto2) library "myDLL.d
close	31	function string my_dll_func2 (string toto1, string toto2) library "myDLL.d
open	32	
f(x) my_dll_func1	33	
f(x) my_dll_func2	34	end prototypes
f(x) my_plsql_proc1	35	forward prototypes
f(x) my_plsql_proc2	36	public function integer uf_method1 (integer p1)
f(x) uf_method1	37	
f(x) uf_method2		

Visual Expert also displays the references to RPCFUNC or dll functions when they are selected:

open	20	
f(x) my_dll_func1	21	type prototypes
f(x) my_dll_func2	22	
f(x) my_plsql_proc1	23	// déclarations de procédures stockées sous forme de méthode
References :	24	↳ (RPCFUNC)
sp_plsql1	25	▶ function string my_plsql_proc1 (string toto) rpcfunc alias for sp_plsql1
f(x) my_plsql_proc2	26	function string my_plsql_proc2 (string toto) rpcfunc alias for sp_plsql2
f(x) uf_method1	27	
f(x) uf_method2	28	// déclarations de fonctions externes de DLL
	29	//
open	30	▶ function string my_dll_func1 (string toto1, string toto2) library
f(x) my_dll_func1	24	//
References :	25	function string my_plsql_proc1 (string toto) rpcfunc alias for sp_plsql1
external_func1	26	function string my_plsql_proc2 (string toto) rpcfunc alias for sp_plsql2
f(x) my_dll_func2	27	
f(x) my_plsql_proc1	28	// déclarations de fonctions externes de DLL
f(x) my_plsql_proc2	29	//
f(x) uf_method1	30	▶ function string my_dll_func1 (string toto1, string toto2) library
f(x) uf_method2	31	↳ "myDLL.dll" alias for external_func1
	32	function string my_dll_func2 (string toto1, string toto2) library
	33	↳ "myDLL.dll" alias for external_func2

4.2. References found in PL/SQL or T-SQL components

4.2.1. Oracle Package

A Package is related to any item referenced by the package itself or its procedures, types, etc... As a result, a Package may reference lots of items. You can list all referenced items in the treewiew and select one of them: the source code will automatically highlight the references to this item.

The Package "Package_Reference" is referencing the column "column1"...

Package_Reference
References :

- column1
- column2
- proc1
- proc2
- proc3
- table1
- local2

```
94 PROCEDURE proc3(  
95   param1 IN table1.column1%  
96   param2 IN table1.column2%  
97 )  
98 AS  
99  
100   l_varlocal_1 table1.column1%type;  
101  
102 BEGIN  
103  
104   SELECT column1, proc1()  
105   FROM table1;
```

...as well as the Procedure "Proc2"...

Package_Reference
References :

- column1
- column2
- proc1
- proc2
- proc3
- table1
- local2

```
84 BEGIN  
85  
86   SELECT column1, column2 , proc1()  
87   FROM table1;  
88   proc2( proc3( proc1() ) );  
89  
90 END;
```

...and the variable "local2"...

Package_Reference
References :

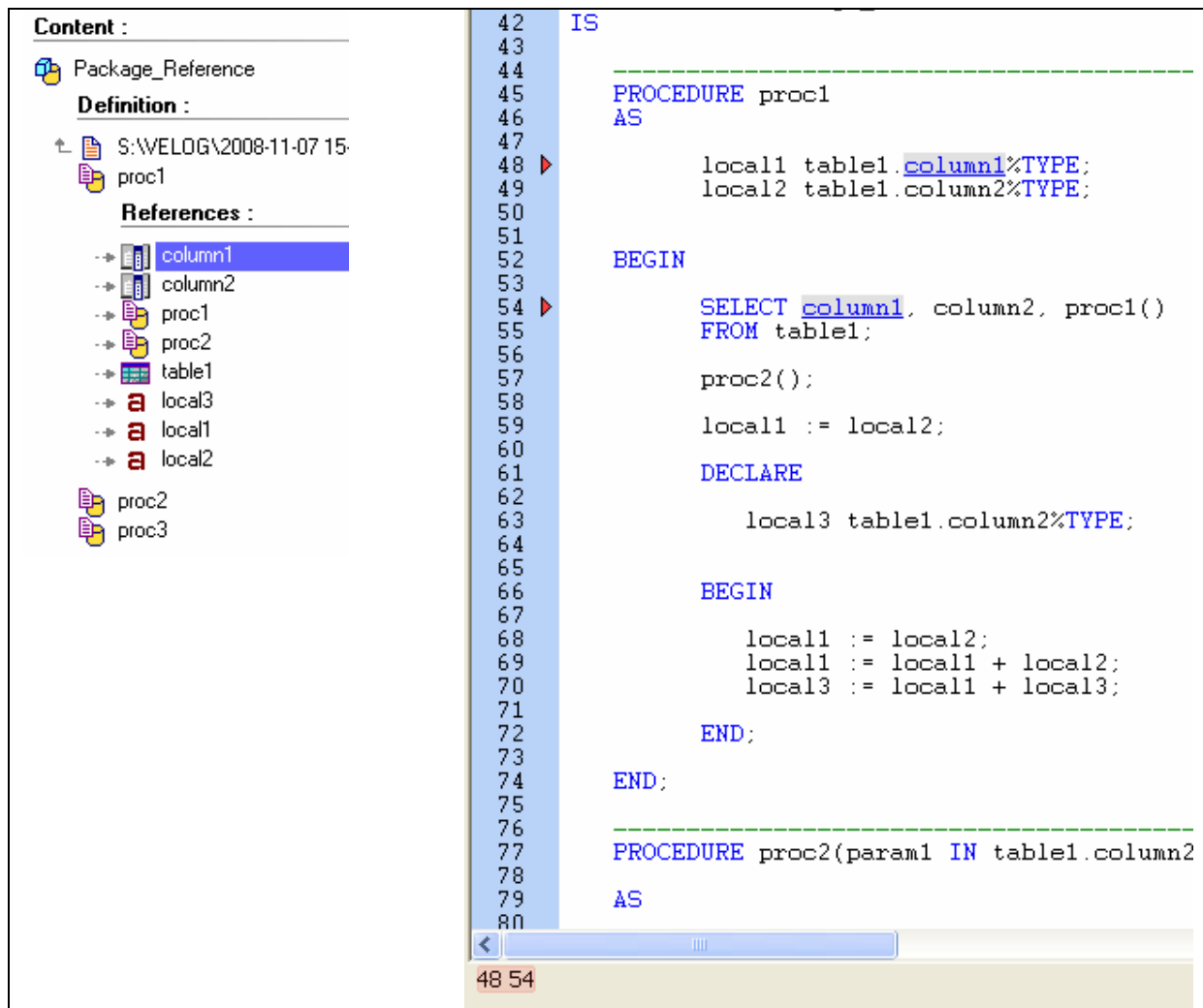
- column1
- column2
- proc1
- proc2
- proc3
- table1
- local2

```
59   local1 := local2;  
60  
61 DECLARE  
62  
63   local3 table1.column2%TYPE;  
64  
65  
66 BEGIN  
67  
68   local1 := local2;  
69   local1 := local1 + local2;  
70   local3 := local1 + local3;  
71  
72 END;
```

4.2.2. Stored Procedures

Visual Expert can list the items referenced by a stored procedure.

This is the same concept as the Oracle Packages, with a scope limited to a stored procedure:



The screenshot displays the Visual Expert interface. On the left, a tree view under 'Content' shows a 'Package_Reference' section with a 'Definition' for 'proc1' located at 'S:\VELOG\2008-11-07 15'. Below this, a 'References' list includes 'column1', 'column2', 'proc1', 'proc2', 'table1', 'local3', 'local1', and 'local2'. On the right, the SQL code for 'proc1' is shown, starting with 'PROCEDURE proc1 AS' and ending with 'END;'. The code includes local variable declarations and assignments. A vertical line of numbers (42-80) is on the left of the code, and a horizontal line of numbers (48-54) is at the bottom. The number '48' is highlighted in red, corresponding to the 'column1' reference in the tree view.

```
42 IS
43
44 -----
45 PROCEDURE proc1
46 AS
47
48     local1 table1.column1%TYPE;
49     local2 table1.column2%TYPE;
50
51
52 BEGIN
53
54     SELECT column1, column2, proc1()
55     FROM table1;
56
57     proc2();
58
59     local1 := local2;
60
61 DECLARE
62
63     local3 table1.column2%TYPE;
64
65
66 BEGIN
67
68     local1 := local2;
69     local1 := local1 + local2;
70     local3 := local1 + local3;
71
72 END;
73
74 END;
75
76 -----
77 PROCEDURE proc2(param1 IN table1.column2
78 AS
79
80
```

48 54

4.2.3. Other DB Code items

Visual Expert also analyses Views, Types and Cursors.
It supports references to tables and columns with on the instruction %Type

4.2.4. PL/SQL %TYPE references

Visual Expert supports all %TYPE references. Parent items (Views, Types, and Cursors) and the table/column referenced. For instance:

Content : Package_Reference Definition : S:\VELOG\2008-11-07 proc1 References : column1 column2 proc1 proc2 table1 local3 local1 local2 proc2 proc3	44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66	<pre>PROCEDURE proc1 AS local1 table1.column1%TYPE; local2 table1.column2%TYPE; BEGIN SELECT column1, column2, proc1() FROM table1; proc2(); local1 := local2; DECLARE local3 table1.column2%TYPE; BEGIN</pre>
Content : Package_Reference Definition : S:\VELOG\2008-11-07 proc1 proc2 proc3 References : column1 column2 proc1 proc2 proc3 table1 l_varlocal1_inBloc l_varlocal2_inBloc l_varlocal_1	93 94 95 96 97 98 99 100 101 102 103 104 105 106 107 108 109 110 111 112 113 114 115	<pre>PROCEDURE proc3(param1 IN table1.column1%TYPE, param2 IN table1.column2%TYPE) AS l_varlocal_1 table1.column1%type; BEGIN SELECT column1, proc1() FROM table1; BEGIN SELECT column1, column2, proc1() FROM table1; proc2(proc3(proc1())); END ;</pre>

4.2.5. Parameters & Local Variables

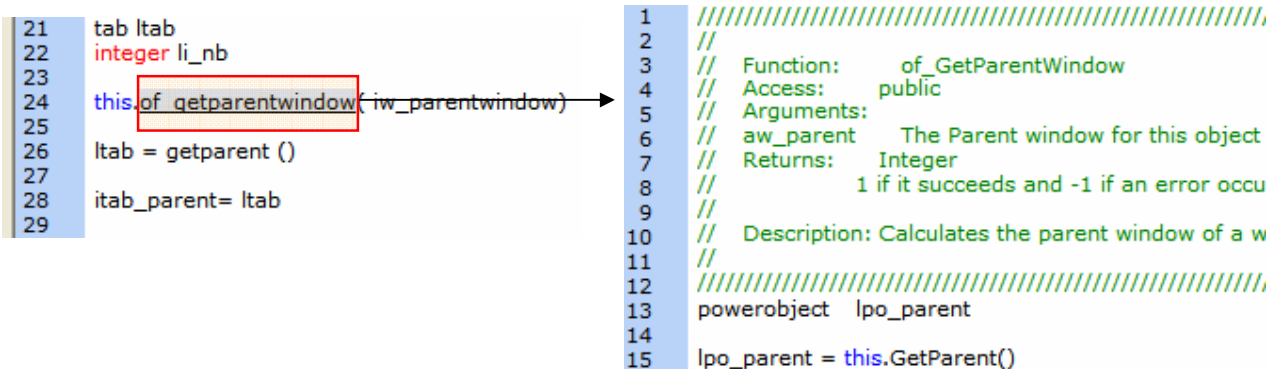
Variables & Parameters are referenced as any other DB Item. Each reference will be highlighted in the source code.

5. Source code view

5.1. Hyperlinks in the code

Hyperlinks are now available in the source code, both for methods and variables:

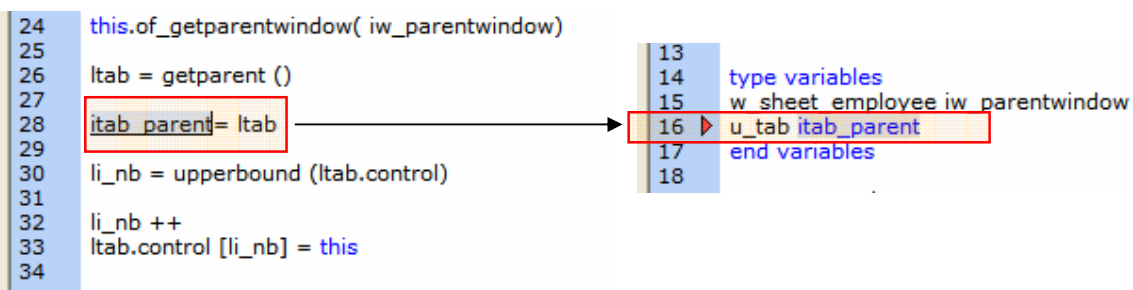
A click on a referenced method will display its source code (Go to definition):



```
21 tab ltab
22 integer li_nb
23
24 this.of_getparentwindow(iw_parentwindow)
25
26 ltab = getparent ()
27
28 itab_parent= ltab
29
```

```
1 ////////////////////////////////////////////////////////////////////
2 //
3 // Function: of_GetParentWindow
4 // Access: public
5 // Arguments:
6 // aw_parent The Parent window for this object
7 // Returns: Integer
8 // 1 if it succeeds and -1 if an error occur
9 //
10 // Description: Calculates the parent window of a w
11 //
12 ////////////////////////////////////////////////////////////////////
13 powerobject lpo_parent
14
15 lpo_parent = this.GetParent()
```

A click on a referenced variable will display its declaration (Go to declaration):



```
24 this.of_getparentwindow( iw_parentwindow)
25
26 ltab = getparent ()
27
28 itab_parent= ltab
29
30 li_nb = upperbound (ltab.control)
31
32 li_nb ++
33 ltab.control [li_nb] = this
34
```

```
13
14 type variables
15 w_sheet employee iw_parentwindow
16 u_tab itab_parent
17 end variables
18
```

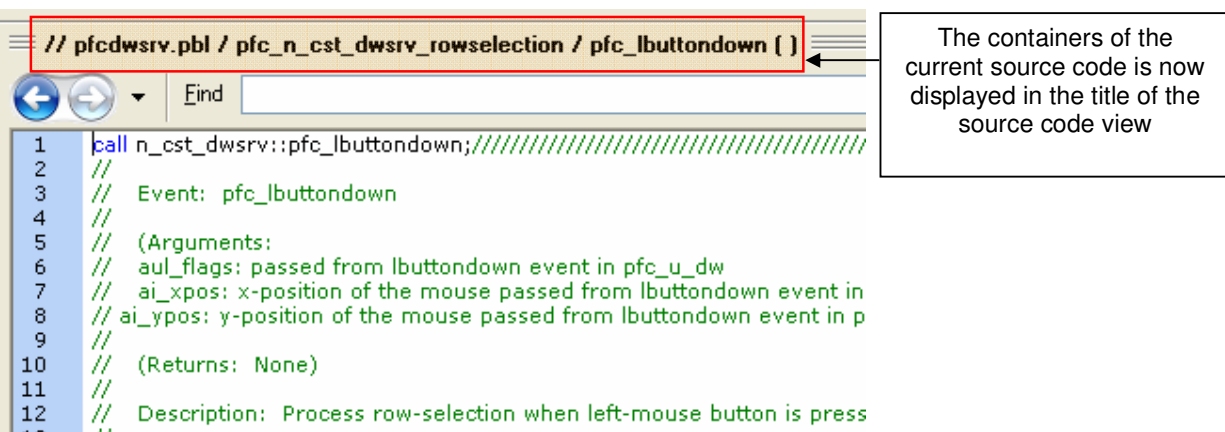
5.2. Title of the source code view

The title of the source code view now displays the containers of the current code.

It helps to understand where is located the current code.

For instance:

- //{PBL}/{composant}/{contrôle}/{méthode}
- //{text file}/{Package}/{Procédure}



```
// pfc_dwsrv.pbl / pfc_n_cst_dwsrv_rowselection / pfc_lbuttondown [ ]
```

```
1 call n_cst_dwsrv::pfc_lbuttondown;//////////////////////////////////////////////////////////////////
2 //
3 // Event: pfc_lbuttondown
4 //
5 // (Arguments:
6 // aul_flags: passed from lbuttondown event in pfc_u_dw
7 // ai_xpos: x-position of the mouse passed from lbuttondown event in
8 // ai_ypos: y-position of the mouse passed from lbuttondown event in p
9 //
10 // (Returns: None)
11 //
12 // Description: Process row-selection when left-mouse button is press
13 //
```

5.3. Search in the source code view

The search feature in the source code view has been redeveloped:

- When entering a string in the “find” field, the code automatically scrolls down to display the first occurrence found.
- The number of occurrences found in this code is automatically displayed.
- Each line containing a reference is marked with a bullet
- You can type [enter]/[Shift Enter] to navigate up/down in the list of occurrences, or click on the up/down button.
- An Index shows the number of all lines containing an occurrence of the string.

While you type a string, the code automatically scrolls down to the first occurrence

...and the number of occurrences found is

The up/down buttons will jump to the next/previous occurrence. Same result when typing [enter/Shift enter]

```

43 procedure OrdersbyEmployee( Employee_Id in NUMBER )
44 is
45     Emp_fname varchar2;
46     Emp_lname varchar2;
47
48     cursor c1 is
49     SELECT sales_order.id Oid,
50           sales_order.order_date Odate,
51           DECODE(sales_order.fin_code_id,
52                 'r1','Revenue','e1','Expense'),
53           sales_order.region Oregion,
54           customer.fname Cfname,
55           customer.lname Clname,
56           sum (sales_order_items.quantity*unit_price) Oa
57     FROM customer,
58          employee,
59          product,
60          sales_order_items,
61          sales_order
62     WHERE ( employee.emp_id = Employee_Id ) and

```

This Index shows all line number with an occurrence. A click on a number scroll to the

Each line with an occurrence of the string is marked with this

Each occurrence found is selected with a square

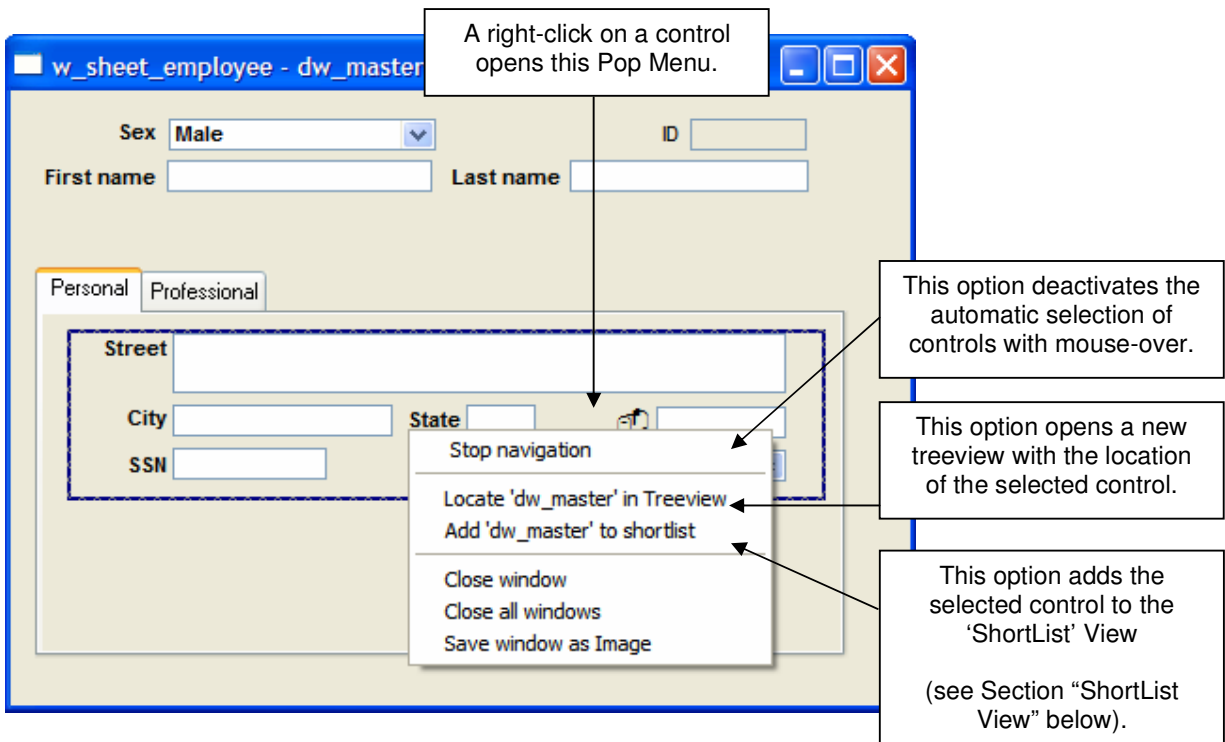
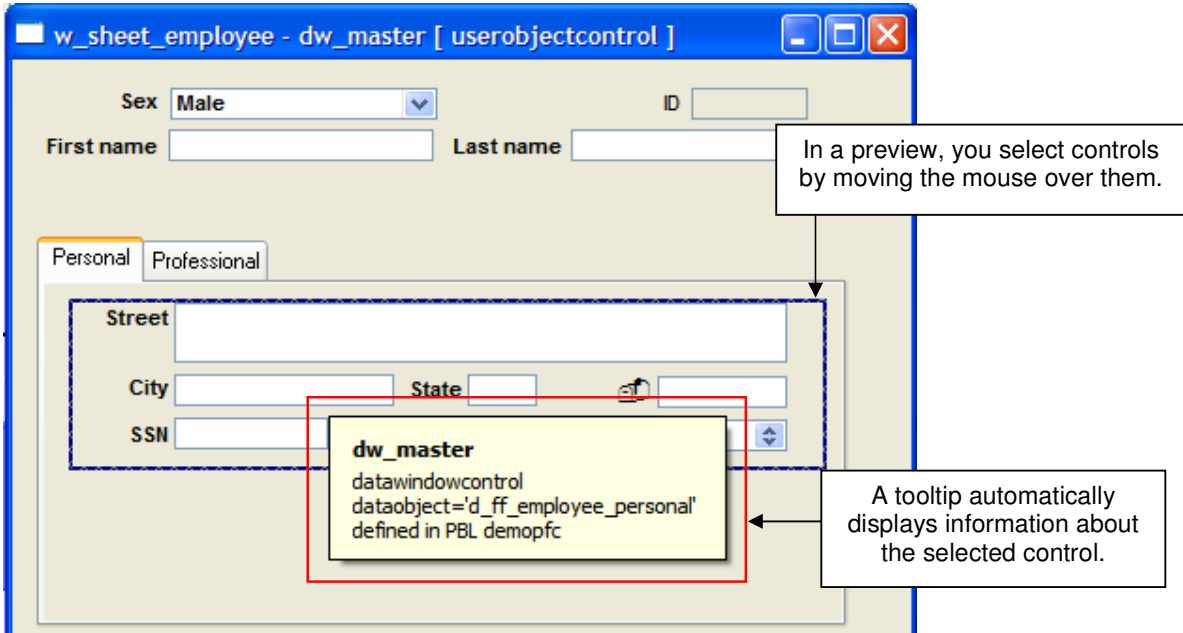
The occurrences currently visible in this view are selected with a blue

When navigating up/down in the occurrences, the current

6. Miscellaneous

6.1. Object Previews

While exploring your application, you can use the Macro "Preview" to display a graphical preview of PowerBuilder Windows, Datawindows or Visual UserObjects:



6.2. Tooltips

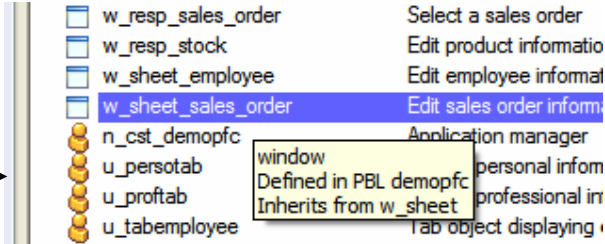
When you move the mouse over code item, Visual Expert displays ToolTips:
They provide additional information about this item (object, method, variable, table...).

Such tooltips are available:

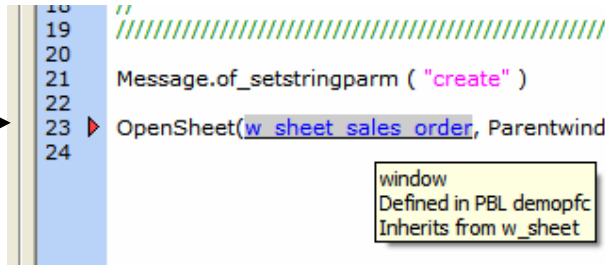
- For the items listed in the treeview
- For the items referenced in the source code view
- For the controls displayed in a preview (see the section *Object Preview*)

Tooltips display additional information for...

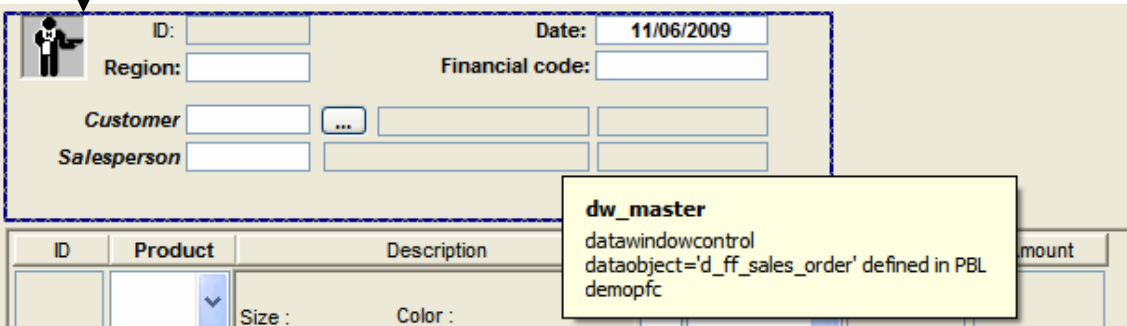
...Items listed in the treeview



Items referenced in the code



Controls displayed in a preview



The image illustrates three scenarios where Visual Expert displays tooltips:

- Items listed in the treeview:** A tooltip for `w_sheet_sales_order` shows its description "Edit sales order inform" and inheritance information: "window", "Defined in PBL demopfc", and "Inherits from w_sheet".
- Items referenced in the code:** A tooltip for `w_sheet_sales_order` in the code `OpenSheet(w_sheet_sales_order, Parentwindi` shows the same inheritance information.
- Controls displayed in a preview:** A tooltip for `dw_master` in a data entry form shows its description "datawindowcontrol" and dataobject information: "dataobject='d_ff_sales_order' defined in PBL demopfc".

Tooltips examples:

```

14 //
15 // Date Version
16 //
17 // 06/07/2000 1.0 Initial version
18 //
19 ///////////////////////////////////////////////////////////////////
20
21 w_sheet_sales_order lw_window
22 n_cst
23 window
24 // Checked
25 Inherits from w_sheet
26 IF IsNull(ai_row) or NOT IsValid(gnv_app.inv_mru) THEN
27 Return -1
28 END IF
29

```

PowerBuilder Object:
type, ancestor and PBL

```

2 m_mymenu ll_menu
3 cb_1 ll_var_cb
4 dw_1 ll_var_dw
5 userobject ll_var_uo
6 w_menu ll_win1
7 window ll_win2
8
9 ll_menu.m_menu1.m_menu_CC.checked = true
10 cb_1.text = "ok"
11 w_menu.uo_1.cb_1.text = "ok"
12 w_menu.dw_1.dataobject = "memo"
13
14 uo_1.cb_1.text = "memo"
15 dw_1.dataobject = "memo"
16
17 m_menu.text="ok"

```

Nested controls:
container, ancestor and PBL

```

2 m_mymenu ll_menu
3 cb_1 ll_var_cb
4 dw_1 ll_var_dw
5 userobject ll_var_uo
6 w_menu ll_win1
7 window ll_win2
8
9 ll_menu.m_menu1.m_menu_CC.checked = true
10 cb_1.text = "ok"
11 w_menu.uo_1.cb_1.text = "ok"
12 w_menu.dw_1.dataobject = "hello!"

```

DataWindowControl:
All dataobjects used are listed (both define in PB painter and dynamically associated by code)

```

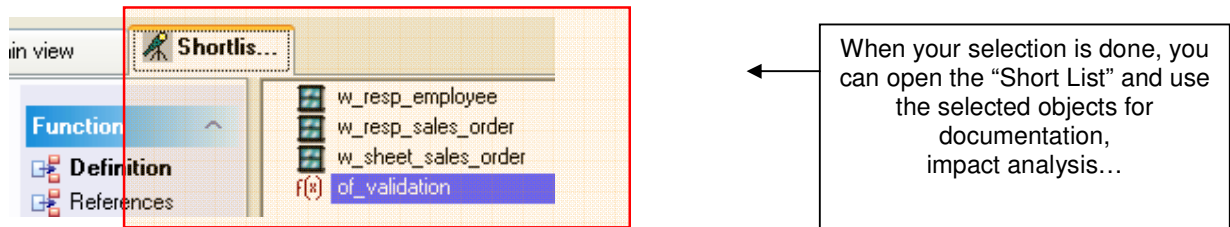
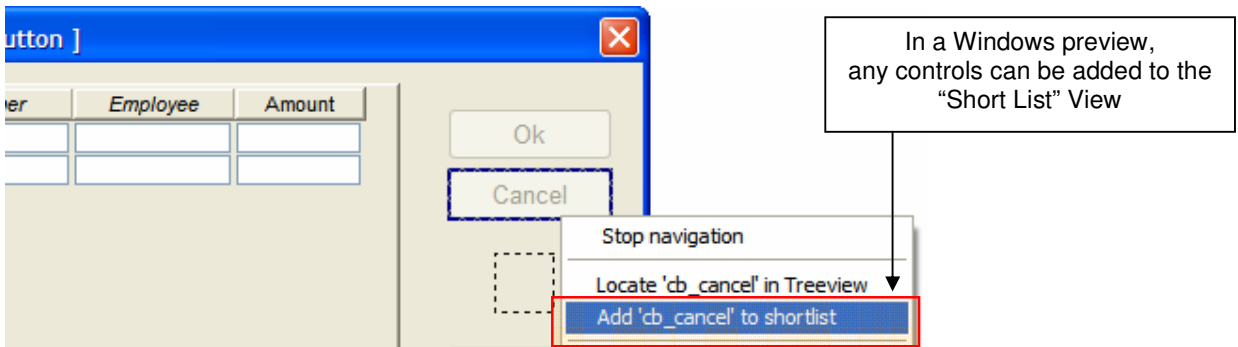
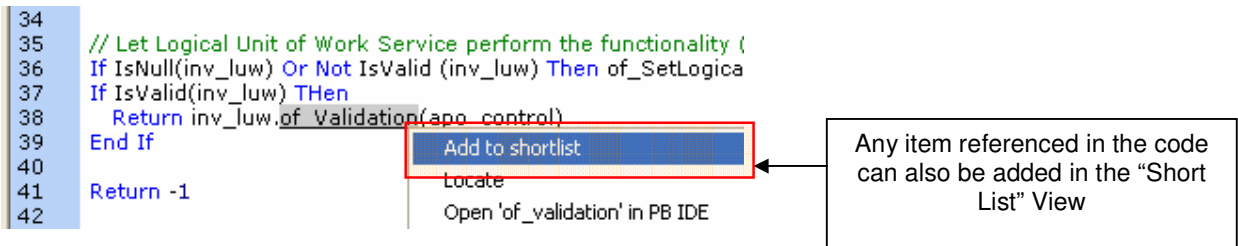
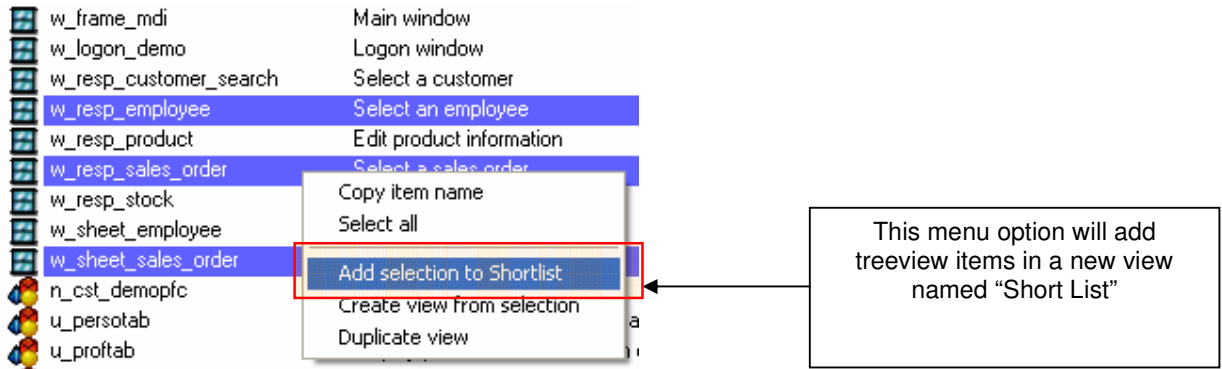
2 m_mymenu ll_menu
3 cb_1 ll_var_cb
4 dw_1 ll_var_dw
5 userobject ll_var_uo
6 w_menu ll_win1
7 window ll_win2
8
9 ll_menu.m_menu1.m_menu_CC.checked = true
10 cb_1.text = "ok"
11 w_menu.uo_1.cb_1.text = "ok"
12 w_menu.dw_1.dataobject = "hello!"
13
14 uo_1.cb_1.text = "ok"

```

Variable:
scope (local), type ('dw_1'), and information about the type.

6.3. “Short List” View

You can now select one by one code items and group them in a specific view named “ShortList”:



6.4. Technical requirements

- Polymorphism is now supported: it required Visual Expert exceptions to be declared
- Visual Expert GUI has been migrated to PB11 for future integration of .NET controls in VE
- PB11 Pre-processing supported: C# code is identified and ignored for now.
- PB 11.5 is now supported (syntax analysis, integration with PB 11.5 IDE...)