ALVs Standard: Reutilizando o caos alemão

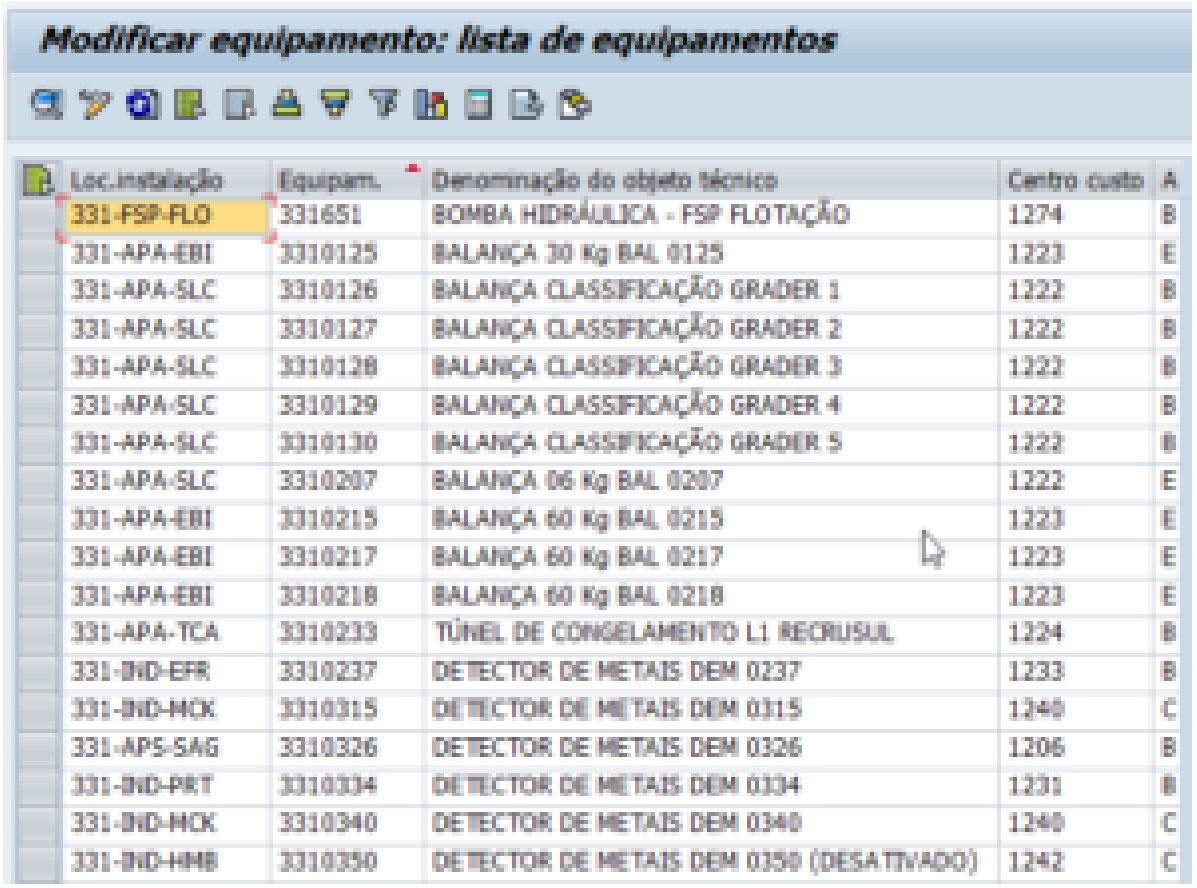
19/09/2013 12:00

Queridos zumbis, alguma vez vocês já passaram pela seguinte situação: o seu programa precisa usar as informações de algum relatório ALV, porém o relatório não está escrito de uma forma que faz com que o código fosse facilmente reutilizável?
Frustrante, não é?! 😕
Bom, dependendo do relatório, há sim uma chance de reutilizar a extração de dados e o código de processamento. Porém, meu amigo zumbi, infelizmente você vai descobrir que existem pouquíssimos relatórios standard que podem ser reutilizados com facilidade, principalmente quando começa os comentários em alemão e você tenta traduzi-los na esperança de entender alguma coisa (admita, você já fez isso!).

Então, e agora?
Se você encontrar um relatório que tem a lógica que você precisa, pode ter uma outra maneira de reutilizá-lo sem precisar clonar programas ou ficar copiando código. Você pode chamar o relatório dentro do seu programa e tratá-lo como uma rotina. Essa técnica pode te ajudar a reduzir o esforço do desenvolvimento e reutilizar objetos que já existem.

Mas como isso funciona?
Depois da extração e processamento, os relatórios ALV exportam os dados para uma área de memória global e isso permite que o seu programa possa importá-los depois.
Quando um relatório ALV é executado, as informações em tempo de execução (layout, fieldcat, tabelas e etc.) são armazenadas pelos métodos da classe CL_SALV_BS_RUNTIME_INFO. Aliás, essa classe usa o EXPORT TO MEMORY para armazenar as informações. A classe lida com tudo isso internamente e fornece métodos para acessar essas informações, ou seja, não precisamos nos preocupar como elas são armazenadas.

Exemplo:
Vou usar como exemplo o relatório IE05, informando o número do inventário e centro na tela de seleção. Essa é a saída do relatório:



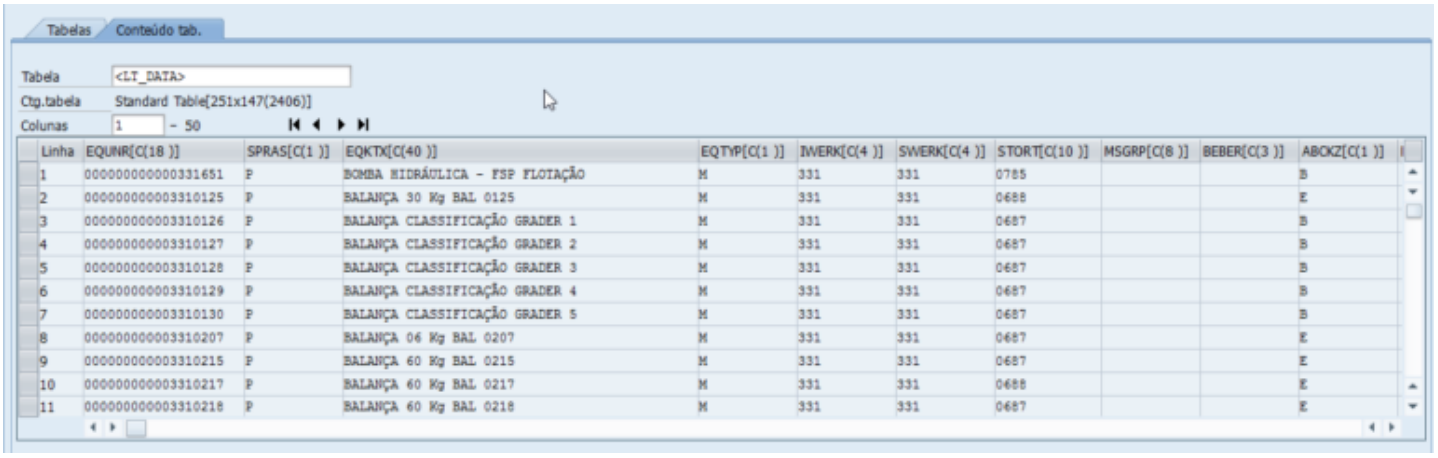
Loc.instalação	Equipam.	Denominação do objeto técnico	Centro custo	A
331-FSP-FLO	331651	BOMBA HIDRÁULICA - FSP FLOTAÇÃO	1274	B
331-APA-EBI	3310125	BALANÇA 30 Kg BAL 0125	1223	E
331-APA-SLC	3310126	BALANÇA CLASSIFICAÇÃO GRADER 1	1222	B
331-APA-SLC	3310127	BALANÇA CLASSIFICAÇÃO GRADER 2	1222	B
331-APA-SLC	3310128	BALANÇA CLASSIFICAÇÃO GRADER 3	1222	B
331-APA-SLC	3310129	BALANÇA CLASSIFICAÇÃO GRADER 4	1222	B
331-APA-SLC	3310130	BALANÇA CLASSIFICAÇÃO GRADER 5	1222	B
331-APA-SLC	3310207	BALANÇA 06 Kg BAL 0207	1222	E
331-APA-EBI	3310215	BALANÇA 60 Kg BAL 0215	1223	E
331-APA-EBI	3310217	BALANÇA 60 Kg BAL 0217	1223	E
331-APA-EBI	3310218	BALANÇA 60 Kg BAL 0218	1223	E
331-APA-TCA	3310233	TÚNEL DE CONGELAMENTO L1 RECRUSUL	1224	B
331-IND-EFR	3310237	DETECTOR DE METAIS DEM 0237	1233	B
331-IND-MCK	3310315	DETECTOR DE METAIS DEM 0315	1240	C
331-APS-SAG	3310326	DETECTOR DE METAIS DEM 0326	1206	B
331-IND-PRT	3310334	DETECTOR DE METAIS DEM 0334	1231	B
331-IND-MCK	3310340	DETECTOR DE METAIS DEM 0340	1240	C
331-IND-HMB	3310350	DETECTOR DE METAIS DEM 0350 (DESATIVADO)	1242	C

Agora vou chamar o relatório dentro do meu programa, passando os mesmos valores para os parâmetros de seleção e importar os dados para uma tabela interna:

```
DATA: lt_selscreen TYPE TABLE OF rsparams,
      wa_selscreen TYPE rsparams.

FIELD-SYMBOLS <lt_data>   TYPE ANY TABLE.
DATA lr_data              TYPE REF TO data.

* Monta a tabela com os dados do parâmetro de seleção
wa_selscreen-selname = 'INVNR'.    "Nome do campo
wa_selscreen-kind    = 'S'.        "Tipo (P - parameter /S - select-options)
wa_selscreen-sign    = 'I'.
wa_selscreen-option  = 'BT'.
wa_selscreen-low     = '120000'.
wa_selscreen-high    = '161000'.
APPEND wa_selscreen TO lt_selscreen.
CLEAR wa_selscreen.

wa_selscreen-selname = 'SWERK'. "Nome do campo
wa_selscreen-kind    = 'S'.     "Tipo (P - parameter /S - select-options)
wa_selscreen-sign    = 'I'.
wa_selscreen-option  = 'EQ'.
wa_selscreen-low     = '331'.
APPEND wa_selscreen TO lt_selscreen.
CLEAR wa_selscreen.

cl_salv_bs_runtime_info=>set(
  EXPORTING display  = abap_false
            metadata = abap_false
            data     = abap_true ).

* Executa o relatório e importa a tabela de saida
SUBMIT riequi20
  WITH SELECTION-TABLE lt_selscreen
```

```
  AND RETURN.
TRY.
    cl_salv_bs_runtime_info=>get_data_ref(
      IMPORTING r_data = lr_data ).
    ASSIGN lr_data->* TO <lt_data>.
  CATCH cx_salv_bs_sc_runtime_info.
    MESSAGE 'Não é possível recuperar os dados ALV' TYPE 'E'.

ENDTRY.

cl_salv_bs_runtime_info=>clear_all( ).
```

Resultado da tabela preenchida:



Linha	EQUNR[C(18)]	SPRAS[C(1)]	EQKTX[C(40)]	EQTYP[C(1)]	IWERK[C(4)]	SWERK[C(4)]	STORT[C(10)]	MSGRP[C(8)]	BEBER[C(3)]	ABCKZ[C(1)]
1	000000000000331651	P	BOMBA HIDRÁULICA - FSP FLOTAÇÃO	M	331	331	0785			B
2	000000000003310125	P	BALANÇA 30 Kg BAL 0125	M	331	331	0688			E
3	000000000003310126	P	BALANÇA CLASSIFICAÇÃO GRADER 1	M	331	331	0687			B
4	000000000003310127	P	BALANÇA CLASSIFICAÇÃO GRADER 2	M	331	331	0687			B
5	000000000003310128	P	BALANÇA CLASSIFICAÇÃO GRADER 3	M	331	331	0687			B
6	000000000003310129	P	BALANÇA CLASSIFICAÇÃO GRADER 4	M	331	331	0687			B
7	000000000003310130	P	BALANÇA CLASSIFICAÇÃO GRADER 5	M	331	331	0687			B
8	000000000003310207	P	BALANÇA 06 Kg BAL 0207	M	331	331	0687			E
9	000000000003310215	P	BALANÇA 60 Kg BAL 0215	M	331	331	0687			E
10	000000000003310217	P	BALANÇA 60 Kg BAL 0217	M	331	331	0688			E
11	000000000003310218	P	BALANÇA 60 Kg BAL 0218	M	331	331	0687			E

Simples, não é? Vou explicar os métodos e os parâmetros utilizados pela classe CL_SALV_BS_RUNTIME_INFO:

SET () – método que inicializa a classe.
• DISPLAY – Defina como “abap_false” para forçar o relatório ALV ser executado em “background”, ou seja, o ALV não será exibido.
• METADATA – Defina como “abap_false” para evitar que o layout, fieldcat, etc, sejam exportados para a memória... a gente não precisa deles neste cenário.
• DATA – Defina como “abap_true” para forçar a exportação da tabela de dados para a memória.

GET_DATA_REF () – Se você não conhece qual é a estrutura da tabela, ou prefere usar uma tabela dinâmica, este método pode ser usado para acessar a referência da tabela.
• R_DATA – parâmetro de exportação da tabela.

CLEAR_ALL () – este método limpa todas as áreas de memória, redefinindo os flags que foram definidos no método SET ().

Essa técnica pode ser utilizada para acessar os dados dos relatórios que usam qualquer tipo de ALV (ALV Grid, ALV List, ALV Tree), mas, obviamente, não vai funcionar para relatórios WRITE ou qualquer coisa que não seja SAPGUI ALV.
É isso aí, espero que isso possa ajudá-los algum dia! Até mais e que a força anti-zumbi esteja com vocês!!

Comentários

> **Felipe** — *20/04/2020 10:39*
>
> Me ajudou muito 🙂

marcos — *27/11/2019 18:59*

Boa Noite,

muito boa dica, eu já utilizo esta forma de obter dados de alv porém estou com uma dificuldade.

Quando o relatório que utilizo este método já faz uso dele para outro alv o resultado passa a ser exibido em tela.

Já passou por isso ?

Reginaldo Russo — *08/10/2015 17:09*

Pessoal, também é possível utilizar o CALL TRANSACTION no lugar do SUBMIT.

Alex — *19/12/2014 10:18*

Como faço para os dados que estão armazenados lr_data tipo atribuir um campo a uma variavel local.EX:

matnr = -MATNR

> **Ruan Carvalhaes** — *22/07/2015 08:18*
>
> Basta rebobinar o disquete do programa principal e processar a rebimboca da parafuseta.

Sandra — *07/07/2014 17:12*

Muito legal a dica . Me ajudou muito.

Jorge Bastos — *12/02/2014 07:03*

Ótima dica Lady Dai!
Parabéns!!!

Daniel jesus — *09/01/2014 08:29*

Sensacional!
Parabéns Daiane!

Mauro Laranjeira — *25/09/2013 11:27*

Show de Bola Dai... Parabéns =D

Kazu — *23/09/2013 09:30*

Muito bom Daiane! valew pela informação !

Vinicius Ostan — *20/09/2013 13:48*

Amazing!!!

Muito bom..
Acabei de passar por tal situação, pena não ter lido este tópico antes... 😕

Na próxima vou lembrar do zombie.

Obrigado

Renan A. Ferreira — *19/09/2013 13:50*

Pãtz, ótima dica Daiane!
Valeu!