

Macros e os comandos que não existem (ou existem?).

15/02/2011 15:00

Antes de tudo, um fato que **ainda** assusta alguns ABAPers: você sabia que o comando "break ", **não é um COMANDO de verdade?**



Mas Hein?!?

Eu explico: o comandinho mágico "break " é, na verdade, uma **MACRO!** Dúvida? Então abre aí a tabela **TRMAC** na SE16, e coloque o nome "BREAK" no campo name:

NAME	NUMM	LINE
BREAK	000	* USER specific BREAK-POINT
BREAK	001	if sy-uname = '&1'
BREAK	002	break-point
BREAK	003	endif

Comandos da MACRO break

Muito bonito, muito interessante... **mas que diabos são macros?**

Bom, vale dizer aqui que eu conheci as macros de uma maneira meio triste... Quando eu estava começando no mundo do ABAP, eu fiquei um tempão olhando para aquele "validate_execution" dentro de uma EXIT da VA, tentando entender o que aquela coisa fazia.. até procurei no google :oops:. Daí um amigo mais experiente me falou "isso é uma MACRO cabeção!", e me explicou o seguinte:

- **MACROS** são sequências de comandos pré-definidos dentro dos comandos **DEFINE...END-OF-DEFINITION**.
- A idéia é ter criar uma espécie de "novo comando", que você pode re-utilizar no resto do seu código.
- Os comandos ABAP definidos dentro de MACROS não podem ser debuggados, portanto, devem conter poucas linhas.
- As macros podem ser definidas globalmente, utilizando a tabela TRMAC (via sm30). **Eu não recomendo utilizar isso, e nem recomendo que qualquer alteração seja feita nas macros já existentes nesta tabela. Se você apagar algo e programas standard começarem a dar erro de sintaxe, a culpa é só sua :P!**

Pois bem, vejamos o exemplo abaixo da criação de uma macro local:

```
REPORT    zombie_macro.

* Variável contadora de comida de zumbi.
DATA: v_cerebros_consumidos TYPE i.

* Criação da MACRO
*-----*
DEFINE comer_cerebros.

* Parâmetros em MACROS são marcados com &1...&9 Tome cuidado pois
* ele vai validar a sintaxe abaixo usando o valor do parâmetro.
  v_cerebros_consumidos = v_cerebros_consumidos + &1.

END-OF-DEFINITION.

*-----*
* START-OF-SELECTION
*-----*
START-OF-SELECTION.

* Sim, os zumbis estão com fome! :)
* Note que você não vai conseguir debuggar...
  comer_cerebros 4.
  comer_cerebros 8.
  comer_cerebros 2.
  comer_cerebros 5.
  comer_cerebros 20.

* Vamos ver se somou mesmo.
WRITE v_cerebros_consumidos.
```

Beleza, funciona... Mas depois desse exemplo de zueira, você deve estar se perguntando: **tá, pra que é que eu posso usar isso?**

Lembra da minha história? Da macro “validade_execution”? Ela rodava uma validação para saber se a exit estava ativa ou não numa transação de configuração do cliente (não tem SMOD/CMOD para a SAVE_DOCMT_PREPARE da VA...), e adivinha: os ABAPs espertinhos não podiam debuggar, porque você não pode debuggar macros! Logo, se a gente colocar um AUTHORITY-CHECK dentro de uma MACRO...

```
REPORT    zombie_macro.

*-----*
* MACROS
*-----*
DEFINE valida_autorizacao.

  authority-check object 'Z_CENTRO'
    id 'WERKS' field &1.

END-OF-DEFINITION.

DATA: wa_marc TYPE marc.

*-----*
* START-OF-SELECTION
*-----*
START-OF-SELECTION.

* Select qualquer, só para testar
SELECT *
FROM marc
```

```

INTO wa_marc
UP TO 1 ROWS.
ENDSELECT.

IF sy-subrc = 0.

*   Authority Check inquebrável!
    valida_autorizacao wa_marc-werks.

ENDIF.

```

Já que não dá pra debugar, você acaba com a graça dos ABAPers que curtem pular autorizações. 🙄

Agora falando sério **mesmo**: eu usava macros para fazer algumas conversões ou mesmo contas simpleszinhas. Acho que ajuda e deixa o código mais limpo. Ao mesmo tempo, se for para fazer alguma lógica mais complexa ou que tenha base na regra de negócio, crie um FORM ou um Método. É mais fácil de debugar, analisar problemas ou mesmo alterar.

E por hoje chega! **break mauricio.cruz**

Abraços!

Comentários

Vinicius Silva — 17/07/2014 13:28

Ressuscitando esse zumbi...

Outra vantagem da macro mas que deve ser usada com uma BOA quantidade de cuidado é que ele não exige declaração do tipo das estruturas que forem usadas como parâmetros...

Suponhamos duas estruturas que possuem o campo "DT_FIM", você quer atualizar o campo das duas estruturas, até aí o perform resolve... mas e se cada uma possui uma estrutura diferente? Embora você consiga trabalhar com a estrutura, para acessar o campo o perform exigirá a declaração do tipo da estrutura, perdendo a dinâmica da coisa.

```

DEFINE ajusta_data.

&1-DT_FIM = &1-DT_INICIO + 30.

END-OF-DEFINITION.

```

```

ajusta_data wa1.

ajusta_data wa2.

```

Pronto, problema resolvido. Por motivos óbvios, todas as estruturas que forem passadas na macro devem ter os campos usados na mesma.

Rafael Paes — 11/03/2013 10:50

Olá.
Sempre estou por aqui e hoje resolvi curtir!
Ótimo trabalho! Grande Abraço!

Gisele Oliveira — 04/09/2012 11:38

Gente, desde que conheci este site minha vida de ABAPer nunca mais foi a mesma. 😊
O conteúdo é sempre muito útil e didático.
Fora que é uma diversão ler os textos.
Sempre deixo escapar no mínimo um sorrisinho quando vejo fotos como a do macaco acima ou a do Nhonho na outra página. rs
Posso dizer que já perdi a conta de quantas coisas aprendi aqui.
Hoje descobri como usar essa tal de macro. 😊
Muito bom mesmo!
Valeu!

Rafael Tavares — 21/02/2011 05:30

As macros têm outra característica bastante interessante que permite que se monte os nomes das variáveis dinamicamente.

Se eu tiver no meu código variáveis com os nomes:

```
DATA: wa_zombie1 TYPE type_zombie,  
wa_zombie2 TYPE type_zombie.
```

Posso criar uma macro para limpar qualquer campo da tabela:

```
DEFINE limpa_campo.  
CLEAR wa_&1-&2.  
END-OF_DEFINITION.
```

As chamadas dessa macro deveriam passar sempre os valores para se completar o nome do campo. Por exemplo:
limpa_campo: zombie1 cabeca,
zombie2 tronco.

A primeira chamada da macro vai limpar o campo 'wa_zombie1-cabeca' e a segunda chamada vai limpar o campo 'wa_zombie2-tronco'.

Isso pode evitar que sejam necessários alguns FIELD_SYMBOLS em casos específicos.

Mauricio Cruz — 21/02/2011 05:37

Curti o clima zumbi do seu comentário hahaha.

Valeu Rafael! 😊