Gerando e executando códigos em tempo de execução

28/11/2012 10:32

Não se confunda com o título, caro zumbi das neves! É isso mesmo: a execução do seu programa principal vai gerar e executar um outro código. A idéia deste post veio de um comentário recente que foi feito num post antigo aqui do site. A pergunta do milhão é: o que fazer quando o nível de dinamicidade está altíssimo e o field-symbol não é o bastante?

> Vegeta, ao avistar os níveis de dinamicidade
> deste post

Ah, o GENERATE SUBROUTINE POOL. Ainda hoje eu me lembro do dia em que eu vi o tal do "programa completamente dinâmico". Era um report que criava um código ABAP de acordo com o que o user escolhia na tela de seleção, e executava-o logo em seguida. Note que ele não gerava um programa estático (não existia um "nome de programa" na SE80), ele montava, executava e "eliminava" o código dentro da mesma "janela de execução". O ABAP Noob que existia dentro de mim ficou maravilhado com tal proeza.

Mas daí você me pergunta "tá Mauricio, mas para que diabos serve isso de gerar um programa dentro de um programa"? Se você já utilizou aquela classe bacanuda para gerar tabelas internas de forma dinâmica através do preenchimento um fieldcat (cl_alv_table_create), saiba que você já utilizou o GENERATE SUBROUTINE POOL por tabela! Lembre-se: não existe mágica no mundo da programação.

E ae, vamos aprender a usar esse negócio?

A idéia é beeem mais simples do que parece. Tudo que você precisa fazer é montar o seu FORM dentro de uma tabela interna. Exemplo:

```
REPORT zombie_dinamicidadeextrema.

DATA: t_code       TYPE TABLE OF string,
      l_line       LIKE LINE OF t_code,
      l_prog_temp  TYPE string.

l_line = 'PROGRAM eusoudinamico.'.
APPEND l_line TO t_code.

l_line = 'FORM escreva_algo.'.
APPEND l_line TO t_code.

l_line = 'WRITE ''algo''.'.
APPEND l_line TO t_code.
```

```
l_line = 'ENDFORM.'.
APPEND l_line TO t_code.
```

Depois, você deve utilizar o comando GENERATE SUBROUTINE POOL, passando como parâmetro a tabela que acabou de criar. Esse é o comando que vai criar o "programa" em tempo de execução.

```
GENERATE SUBROUTINE POOL t_code NAME l_prog_temp.
```

**Você ainda pode colocar outros parâmetros batutas nesse comando. F1 nele* 🙂

O parâmetro L_PROG_TEMP vai retornar o nome do programa temporário que foi criado. Agora é só fazer um PERFORM... IN PROGRAM para poder chamar o código.

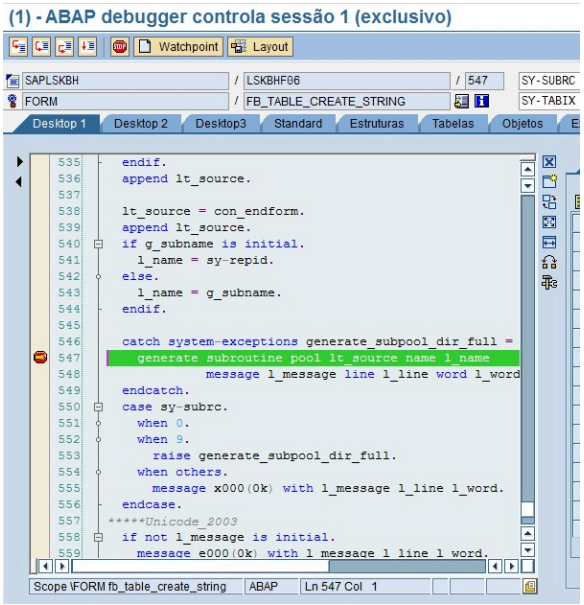
```
PERFORM ('ESCREVA_ALGO') IN PROGRAM (l_prog_temp).
```

E pronto! Você vai ver "algo" escrito na tela.

É importante dizer que o código que você colocar dentro de T_CODE deve estar sintaticamente correto, caso contrário, a variável L_PROG_NAME vai ficar vazia, o comando vai retornar um SY-SUBRC diferente de 0 e um DUMP vai ser disparado no PERFORM...IN PROGRAM.

Fácil, diz aí? E as possibilidades são infinitas! Quer um exemplo?

Como eu disse ali em cima, o método CL_ALV_TABLE_CREATE=>CREATE_DYNAMIC_TABLE é utilizado para criar tabelas dinâmicas. Neste post eu expliquei com exemplos, como fazer uma tabela dinâmica + select dinâmico. Pegue o código, copie num programa temporário e comece a depurar. Coloque um break-point no comando GENERATE SUBROUTINE POOL, e veja a *magia-que-não-é-mais-magia* acontecer:



Não é feitiçaria, é tecnologia! (só os FORTES irão ver o FORM dinâmico que declara a tabela sendo executado)

Uma curiosidade sobre essa técnica, é que existe um limite no número de programas que você pode criar em tempo de execução. Vou explicar como descobri isso com uma breve história:

Certa vez, um amigo meu tinha um programa que precisava fazer selects em mais ou menos 40 tabelas (ou algo assim), que eram mais ou menos parecidas entre si. Pensamos em utilizar a técnica de SELECTs dinâmicos, utilizando

tabelas de retorno geradas dinamicamente pelo método CL_ALV_TABLE_CREATE=>CREATE_DYNAMIC_TABLE. Ele codificou tudo, de um trabalhão... Mas, na hora dos testes, começamos a enfrentar um DUMP após cerca de 35 criações de tabelas dinâmicas. Eu ajudei-o a identificar o problema, e debugamos o standard até chegar no ponto onde o sistema busca esse valor, através de uma Kernel Call. Esse número máximo de Subroutine Pools gerados é fixo, e, até onde eu sei pela pesquisa que fiz na época, não dá para configurar.

A gente acabou pensando em alguma outra solução besta, que eu acho que deu certo. Mas também se não deu, aquela era a hora perfeita de eu dizer "*se vira colega*", já que o produto não era da minha responsabilidade 😈 . Obviamente eu não fiz isso (🙄) e demos um jeito.

Espero que tenham se divertido! Abraços e até a próxima!

Comentários

igor vilela — *15/10/2013 12:43*

eu passei por esse problema, a solução e vc gerar o código do programa e exportar ele pra memória, após isso vc da um submit em um outro programa, nesse outro programa vc pega o codigo exportado pra memoria e cria a subrotina. feito isso vc vai ter burlado o limite de criacao de subrotinas.

att.,
igor vilela

Diego — *21/07/2013 14:58*

Muito bom o post... mas há que se fazer um alerta.
Execução dinâmica de código é muito bom para alguns casos mas muito perigoso também.
Alguém com intenções duvidosas pode, com um simples debug, alterar o conteúdo do código dinâmico e destruir o SAP. Eu mesmo consegui inserir um comando em um form dinâmico que era capaz de me dar todos os acessos de SAP* sem alterar nenhuma tabela... tudo em tempo de execução.

Paulo Sales — *04/12/2012 16:03*

Muito útil! Valeu pelo post.

Kaio Almeida — *02/12/2012 00:22*

Muito bom o post!!