RTTS, RTTI, RTTC e você: tudo a ver – Parte 2

29/04/2014 08:30

E aí, como vai aquele desenvolvimento bizarro em que você precisa identificar qual o tipo de uma determinada variável em tempo de execução? Hoje vamos explorar alguns exemplos do RTTS para fazer esse *trabalho sujo*.

Lembrando que na **Parte 1** você entende a teoria por trás do RTTS e aprende um pouco sobre casting e o operador "=?". Esta é a Parte 2 e, na Parte 3, iremos analisar como criar variáveis em tempo de execução.

Pegue a sua lupa de Sherlock Homes e vem comigo identificar essas variáveis safadas.

Analisando o tipo de uma váriável

Muita gente conhece o bom e velho DESCRIBE para tentar descobrir alguma coisa sobre as variáveis em tempo de execução. Nós mesmo já falamos um pouco do DESCRIBE TABLE por aqui. Porém, se você ler o HELP do describe, vai ver que a recomendação é usar o RTTS ao invés do DESCRIBE e variações, por conta da flexibilidade que as classes do RTTS proporcionam.

Como ignorar essas recomendação é algo *abapossauriano*, vamos fuçar um pouco no funcionamento do RTTS para analisar uma variável. No exemplo abaixo a variável "material" está referenciada ao DDIC. Experimente trocar de "mara-matnr" para outros tipos para ver diferentes resultados:

```
DATA: material  TYPE mara-matnr,
      texto     TYPE string.

DATA: relative_name TYPE string.

DATA: o_typedescr TYPE REF TO cl_abap_typedescr.

DATA: x031l TYPE TABLE OF x031l,
      x030l TYPE x030l.

* Criando o objeto para análise. Este objeto fará referência ao
* tipo da variável "material".
o_typedescr = cl_abap_typedescr=>describe_by_data( material ).

* Nome absoluto contém o nome + identificador. No caso, será \TYPE=MATNR
* que indica que este objeto é referente a um TYPE MATNR(elemento de dados).
* Experimente trocar o tipo do campo para PSTAT, e você verá que o TYPE muda
* para o elemento de dados, que é o PSTAT_D.
WRITE / o_typedescr->absolute_name.

* Nome utilizado para referência no programa.
relative_name = o_typedescr->get_relative_name( ).
WRITE / relative_name.

* Métodos que só funcionam se o tipo referenciado veio do DDIC.
* Se você trocar de "material" para "texto" na hora de chamar o
* método describe_by_data, irá tomar um DUMP na cabeça.
x031l[] = o_typedescr->get_ddic_object( ).
x030l   = o_typedescr->get_ddic_header( ).

* Tipo da Variável, neste caso C de CHAR
WRITE: / o_typedescr->type_kind.
```

```
* Tamanho da variável, para o elemento de dados MATNR - 18
WRITE: / o_typedescr->length.
```

Neste caso utilizamos o método *describe_by_data*, mas você pode utilizar outros métodos para buscar a instância, como o *describe_by_name*. No caso do e *describe_by_name*, você teria que passar o nome do tipo direto – "MATNR" (pelo *describe_by_data* voccê passa a variável).

Para quem não está acostumado com OO, este exemplo pode levar a uma pergunta importante...

Ué, mas onde foi parar o CREATE OBJECT?

Quando você precisa utilizar um objeto, nem sempre isso quer dizer que você precisará criá-lo. É extremamente comum no mundo do ABAP Objects delegar a atividade de criação para algum método de alguma classe. Tipo aqueles chefes que só delegam, delegam, delegam e não fazem nada.

Sério agora: no caso do exemplo anterior, quem irá executar o "CREATE OBJECT" é o método "**cl_abap_typedescr=>describe_by_data**". Esse método é um método estático, ou seja, não precisa ser acessado a partir de uma instância. Se você não entendeu nada, imagine que chamar um método estático público de uma classe da SE24, é igual a chamar uma função, no sentido de ser um pedaço de código que está em algum lugar e que pode ser acessado por qualquer programa.

Pelo lado do design do RTTS, imagine que você está fazendo a seguinte requisição: ***"Hey RTTS, me devolva a instância certa, preenchida com todos os dados dessa variável que estou te enviando"***. Saber disso, por hora, é o suficiente 🙂 .

Para os curiosos de plantão, existe uma construção clássica (design pattern) em OO que se chama *Factory*. Ela abusa de métodos estáticos para que a classe crie a instância da melhor maneira possível. Dê uma olhada neste link para saber como funciona, com exemplos em ABAP. Diz a lenda que eu vou fazer um post disso algum dia, mas é só uma lenda urbana e pode ser mentira, tipo bicho papão, saci pererê, geisy arruda...

Descobrindo os campos de uma estrutura

Agora que você entendeu como instanciar objetos do RTTS, e como fazer análises em uma variável simples, está na hora de utilizar a maravilinda análise de estruturas.

Manja quando você tem aquela estrutura maldita e precisa descobrir se ela tem o campo que você precisa, na posição que você precisa, do tipo que você precisa, do elemento de dados que você precisa, aaaaaaaAAaAaAAa? Pois seus problemas acabaram! 🙄 Descobrir os campos de uma estrutura, sua posição e outras parafernálias nunca foi tão fácil!

```
DATA: estrutura TYPE mara.

DATA: componentes TYPE cl_abap_structdescr=>component_table.

DATA: structdescr   TYPE REF TO cl_abap_structdescr,
      campodescr    TYPE REF TO cl_abap_datadescr.

FIELD-SYMBOLS <component> LIKE LINE OF cl_abap_structdescr=>components.

structdescr ?= cl_abap_structdescr=>describe_by_data( estrutura ).

* Aqui você recebe a lista de todos os componentes da estrutura, com outros
* objetos já referenciados de acordo com o tipo do campo. (abra no debug para entender)
```

```
* Com isso você pode saber O QUE VOCÊ QUISER sobre QUALQUER CAMPO da estrutura.
* Não é feitiçaria, é SAPLoLogia (só pra rimar)
componentes = structdescr->get_components( ).

* Vamos analisar um campo específico da estrutura:
campodescr = structdescr->get_component_type( 'MATNR' ). "Troque o campo para experimentar!"
WRITE: /01 'MATNR',
       10 'É do tipo',
       20 campodescr->type_kind.

*Ah, mas tudo que eu quero é um lista dos campos e já era!
LOOP AT structdescr->components ASSIGNING  <component>.
*   Tá, toma aí
    WRITE: /01 <component>-name,
            20 <component>-decimals,
            30 <component>-length,
            40 <component>-type_kind,
            50 sy-tabix. "posição na estrutura
ENDLOOP.
```

Viu ali o operador =? entrando em cena? Isso acontece porque o método "*describe_by_data*" está declarado na classe pai CL_ABAP_TYPEDESCR, e esse método retorna um parâmetro do tipo TYPEDESCR. Porém, como programadores espertos que somos, sabemos que **AQUELE** "*describe_by_data*" da classe CL_ABAP_STRUCTDESCR vai retornar uma instância da classe CL_ABAP_STRUCTDESCR. É aí que o =? entra em jogo, e dizemos para o verificador de sintaxe não encher o saco porque a gente sabe o que está fazendo.

AI MEU DEUS MAURICIO, NÃO ENTENDI NADA DESSE ÚLTIMO PARÁGRAFO!!!1!!1!!1!!

Leia a Parte 1. Se mesmo assim não entendeu, pergunta pra gente aí nos comentários 🙂 .

Com a classe CL_ABAP_STRUCTDESCR você descobre os campos da estrutura, mas para analisar um campo específico, vai ter que usar a classe para o tipo específico daquele campo. Debugue o exemplo que você vai me entender!

Bom, espero que agora a hierarquia de classes esteja fazendo um pouco mais de sentido para você 😛

Análises em Tabelas

E, por fim, vamos analisar uma tabela interna. Acho que você já sacou que a classe CL_ABAP_TABLEDESCR vai fazer uma análise geral na tabela, mas você vai precisar utilizar a hierarquia de classes para analisar o campos (CL_ABAP_DATADESCR e derivados) de uma linha específica (CL_ABAP_STRUCTDESCR).

```
DATA: tabela TYPE TABLE OF mara.

DATA: tabledescr  TYPE REF TO cl_abap_tabledescr,
      structdescr TYPE REF TO cl_abap_structdescr.

FIELD-SYMBOLS <key> LIKE LINE OF tabledescr->key.

tabledescr ?= cl_abap_tabledescr=>describe_by_data( tabela ).

* Vamos fuçar na linha da tabela
structdescr ?= tabledescr->get_table_line_type( ).
* Note que o programa faz um downcast do tipo que GET_TABLE_LINE_TYPE retorna para o
* objeto STRUCTDESCR.
* E aqui você pode usar o exemplo de estruturas para fuçar no objeto STRUCTDESCR.
* [insira o código do exemplo anterior aqui]

* Campos da tabela
```

```
LOOP AT tabledescr->key ASSIGNING <key> .
  WRITE / sy-tabix.    "Posição do Campo
  WRITE 20 <key>-name. "Números da Mega-Sena (quem sabe assim algúem testa este exemplo!)
ENDLOOP.

* Tem mais um monte de atributos que descrevem outras coisas da tabela
* interna. Explore!
```

É meu, tô falando... esse tal RTTS é pior que paparazzi na praia de copacabana.

Guarde sua lupa porque por hoje chega pessoal. Espero que os exemplos estejam claros o suficiente para que você possa fazer seus próprios experimentos com as classes do RTTS. Não adianta ficar lendo, tem que executar, testar e estudar!

Abraços a todos aqueles que usam a boina do Sherlock Holmes.

Comentários

Leandro Tentoni — *26/08/2016 17:50*

Fala Abaper's!!

seguinte, utilizei as dicas aqui e tenho outra relacionada:
* Métodos que só funcionam se o tipo referenciado veio do DDIC.
* Se você trocar de "material" para "texto" na hora de chamar o
* método describe_by_data, irá tomar um DUMP na cabeça.

Para evitar o Dump é só verificar se campo verificado vem do ddic, da seguinte forma:

"Recebe o nome do dominio (somente para campos do dicionário)
o_typedescr = cl_abap_typedescr=>describe_by_data(material).
"Verifica se tipo é do dicionário
CHECK o_typedescr->is_ddic_type( ) EQ abap_true.

é isso ai, espero ter ajudado!

Abraços!

> **Leandro Tentoni** — *26/08/2016 17:53*
>
> Desconsiderem essa parte:
> ""Recebe o nome do dominio (somente para campos do dicionário)" hehehehehh
>
> abraços

Fawcs — *29/04/2014 09:10*

Esses dias eu estava com preguiça de procurar no google o nome dessas classes quando apareceu esse tweet no meu feed 😉 foi uma mão na roda!

> Eu quera analisar uns dados e como eu sou preguiçoso demais pra aprender as funcionalidades todas do excel... fiz um programa que eu copiava uma planilha e ele gerava um alv pra mim igualzinho 😀
>
> > **Mauricio Cruz** — *29/04/2014 09:13*
> >
> > Aí sim heim Fawcs! Fico feliz de ter ajudado. RTTS está longe de ser algo novo, mas é bem legal e merece esta sequência de posts.
> >
> > Abraços! 🙂
> >
> > > **Fawcs** — *29/04/2014 09:15*
> > >
> > > É mano, ABAP Zombie superando o Google!
> > >
> > > Falando em algo novo... Sabe se existe algum repositório ou uma maneira de saber a partir de qual versão uma classe/função está disponível no SAP? No momento a maneira que eu tenho de descobrir isso é logando num ambiente 4.7 e procurando!
> > >
> > > > **Mauricio Cruz** — *29/04/2014 09:17*
> > > >
> > > > Putz não sei... deve dar para saber olhando a data de criação do objeto abap.
> > > >
> > > > Mas só pesquisando para saber!