RTTS, RTTI, RTTC e você: tudo a ver – Parte 1 (bônus: Casting)

25/04/2014 08:30

Todo mundo adora fazer coisas "dinâmicas" em seus programas, certo? Para essa finalidade, a hierarquia de classes conhecida como RTTS tem muitas ferramentas que podem ajudá-lo a resolver problemas malucos onde você precisa criar/identificar variáveis dinâmicamente no seu programa (seja lá qual for o seu motivo).

Nesta série de posts vou explicar um pouco sobre como (mais) esse monte de siglas pode tornar-se uma arma valiosa para matar aquela EF zumbi que te persegue de noite, no banho, no carro e em todos os seus pensamentos.

Na **Parte 1** nós vamos tentar decifrar essa sopa de letrinhas, entender o que é down-cast e ver um exemplo simples(só para você não me xingar e dizer que o post é pura teoria).

Na Parte 2 vamos explorar algumas funções para analisar variáveis em tempo de execução.

Na **Parte 3**, vamos descobrir como utilizar as ferramentas do RTTS para criar outras variáveis.

Prepare a sua pipoca sabor cérebros e vamos lá!

Véio, RTTOQUE?

Não adianta reclamar de siglas amigo cambaleante, você escolheu ser programador. Você poderia ter jogado futebol, basquete, volêi, queimada, peteca... mas já que escolheu ficar aí sentando aguentando usuários chatos, vamos entender o que quer dizer cada uma dessas siglas:

RTTS – Runtime Type Services, ou "Paradas Legais para usar em Tempo de Execução". Consiste de 2 componentes:

- **RTTC: Runtime Type Creation:** Criação dinâmica em tempo de execução. Ex.: Criar uma nova variável do tipo CHAR 30.
- **RTTI: Runtime Type Identification:** Identificação de tipos em tempo de execução. Ex.: Descobrir que o campo ABC da estrutura WA_XYZ é um CHAR de 20 posições.

Nas minhas pesquisas descobri que, antigamente, só existiam ferramentas para identificação de tipos, que eram chamadas de RTTI. Com o tempo, a SAP implementou ferramentas para criação de variáveis em tempo de execução, e deu o nome de RTTC a elas. Foi então que surgiu a sigla "RTTS", que serve como identificador para essas ferramentas.

Muita gente fica assustada com RTTS pelo mesmo motivo de sempre: você acessa as ferramentas a partir de uma **hierarquia de classes.** Falou de SE24 os ABAPossauros tremem, eu sei. Mas vem comigo que a coisa é mais fácil do que parece.

Toma aí um exemplo simples para você ficar com vontade de aprender:

```
REPORT rtts_rtti_rttc_p_vc.

TYPES: tipo TYPE i.

DATA: variavel TYPE tipo,
      nome     TYPE string.
```

```
DATA: typedescr TYPE REF TO cl_abap_typedescr.

* Notem que estamos utilizando a variável para descrever o seu tipo (TYPE)
typedescr = cl_abap_typedescr=>describe_by_data( variavel ).

WRITE: / typedescr->type_kind. " Output: I (de Inteiro)

* Nome do tipo referenciado pela variável
nome = typedescr->get_relative_name( ).

WRITE: / nome. "Output: (nome do tipo da variavel).

* Experimente trocar o TYPE de integer para alguma outra coisa! (ex. String)
```

Classes do RTTS

Para acessar qualquer função do RTTS, você precisa utilizar a classe correspondente. O RTTS possui uma hierarquia de classes bem definida, que refletem os tipos e variáveis existentes dentro do SAP (types, estruturas, tabelas internas, objetos, etc). Dê uma olhada **neste link**, tem um mapa bem legal dos tipos existentes no ABAP.

Você não precisa decorar todas agora, mas é importante saber que essa hierarquia existe na SE24 mais próxima de você:

(retirado do help)

Para descomplicar um pouco essas siglas de RTTI e RTTC, uma mesma classe possui tanto as funções de descrição quando de criação. Ou seja, se eu quiser fazer alguma coisa (criar/descrever) relacionada a tabelas internas, vou utilizar a classe CL_ABAP_TABLEDESCR.

Sim, este é aquele momento em que você **esquece o que RTTI e RTTC** querem dizer, e **fica só com o RTTS** na cabeça. Aí, falei que era fácil 😀 .

Aqui você encontra uma descrição bem detalhada da hierarquia, com seus respectivos métodos RTTIs e RTTCs, além dos atributos de cada classe.

Importante notar que não existe uma classe RTTS específica para cada tipo de variável. Ex.: Todos os tipos de tabelas internas (STANDARD, SORTED, HASHED) podem ser tratados no RTTS pela classe de tabelas CL_ABAP_TABLEDESCR. O resto das classes RTTS também funcionam de forma semelhante.

Casting e o o fantasma do operador ?=

Em muitos exemplos da internet relacionados a RTTS você vai encontrar o operador ?= , e para não fazer as coisas por osmose, vamos tentar entender um pouco do que essa interrogação se trata.

Antes de mais nada, dê uma lida no post do Mauro falando sobre herança. O conceito de casting está diretamente relacionado aos conceitos de herança.

Este operador serve para que você consiga fazer um down cast, ou seja, para que você consiga mover um **objeto de uma classe pai** e passar para um outro objeto que faz referência a **uma classe filha**.

Vamos imaginar a seguinte hierarquia de classes:

Se eu tiver um objeto do tipo CL_IPHONE, eu irei conseguir enviá-lo a outra variável que faça referência a classe CL_CELULAR. O relacionamento "Iphone é um Celular" é válido (e facilmente compreendido). Como a classe CL_IPHONE é filha de CL_CELULAR, a associação é feita sem problemas, pelo operador "=". Veja só:

```
CLASS cl_celular DEFINITION.
ENDCLASS.

CLASS cl_iphone DEFINITION INHERITING FROM cl_celular.
ENDCLASS.

CLASS cl_galaxy DEFINITION INHERITING FROM cl_celular.
ENDCLASS.

CLASS cl_lumia DEFINITION INHERITING FROM cl_celular.
ENDCLASS.

DATA: o_celular TYPE REF TO cl_celular, "Classe Pai de Todas as outras
      o_iphone  TYPE REF TO cl_iphone,
      o_galaxy  TYPE REF TO cl_galaxy,
      o_lumia   TYPE REF TO cl_lumia.

CREATE OBJECT o_iphone.

* Pai = Filho
o_celular = o_iphone.
```

Esse é o chamado up-casting, ou seja: você está enviando um **objeto do tipo "filho" para uma variável do tipo "pai"**.

Mas e o contrário? E se eu quiser pegar a referencia de CL_IPHONE que está na variável O_CELULAR, e quiser enviar **de volta** a uma variável do tipo CL_IPHONE?

```
CLASS cl_celular DEFINITION.
ENDCLASS.

CLASS cl_iphone DEFINITION INHERITING FROM cl_celular.
ENDCLASS.

CLASS cl_galaxy DEFINITION INHERITING FROM cl_celular.
ENDCLASS.
```

```
CLASS cl_lumia DEFINITION INHERITING FROM cl_celular.
ENDCLASS.

DATA: o_celular TYPE REF TO cl_celular, "Classe Pai de Todas as outras
      o_iphone  TYPE REF TO cl_iphone,
      o_galaxy  TYPE REF TO cl_galaxy,
      o_lumia   TYPE REF TO cl_lumia.

CREATE OBJECT o_iphone.

* Pai = Filho
o_celular = o_iphone.

* Filho = Pai .... funciona?
o_iphone = o_celular.
```

Aí você fala "ok Mauricio, é bem óbvio que isso funcionaria, certo? **Errado**! O verificador de sintaxe irá dizer que essa operação não é possível, com uma mensagem que pode parecer absurda: "*O_CELULAR não pode ser convertido para o tipo de O_IPHONE*".

MAS COMO NÃO MEUUUU??? EU ACABEI DE PASSAR ALI EM CIMA!!!!!11!!!!!!!1!1!!ONZE

Acalme-se jovem zumbiwan. Quem garante que o que está em O_CELULAR é mesmo um O_IPHONE? E se fosse um O_GALAXY? E se fosse um O_LUMIA? E se fosse um O_QUALQUERCELULARDAFACEDATERRA?

```
CLASS cl_celular DEFINITION.
ENDCLASS.

CLASS cl_iphone DEFINITION INHERITING FROM cl_celular.
ENDCLASS.

CLASS cl_galaxy DEFINITION INHERITING FROM cl_celular.
ENDCLASS.

CLASS cl_lumia DEFINITION INHERITING FROM cl_celular.
ENDCLASS.

DATA: o_celular TYPE REF TO cl_celular, "Classe Pai de Todas as outras
      o_iphone  TYPE REF TO cl_iphone,
      o_galaxy  TYPE REF TO cl_galaxy,
      o_lumia   TYPE REF TO cl_lumia.

CREATE OBJECT o_iphone.
CREATE OBJECT o_galaxy.
CREATE OBjeCT o_lumia.

* Pai = Filho
o_celular = o_galaxy.
o_celular = o_iphone.
o_celular = o_lumia.

" o_celular pode fazer referência a qualquer um dos filhos!
" e agoooraaaa???
```

É nesse momento que você precisa utilizar o operador ?=. É como se o verificador de sintaxe falasse para você: "eu sei lá o que tem dentro de O_CELULAR, mas já que você tá falando que é um O_IPHONE, eu acredito!":

```
CLASS cl_celular DEFINITION.
ENDCLASS.

CLASS cl_iphone DEFINITION INHERITING FROM cl_celular.
ENDCLASS.
```

```
CLASS cl_galaxy DEFINITION INHERITING FROM cl_celular.
ENDCLASS.

CLASS cl_lumia DEFINITION INHERITING FROM cl_celular.
ENDCLASS.

DATA: o_celular TYPE REF TO cl_celular, "Classe Pai de Todas as outras
      o_iphone  TYPE REF TO cl_iphone,
      o_galaxy  TYPE REF TO cl_galaxy,
      o_lumia   TYPE REF TO cl_lumia.

CREATE OBJECT o_iphone.

* Pai = Filho
o_celular = o_iphone.

* Filho = Pai.
o_iphone ?= o_celular.

* Com o ?=, o verificador permite que você faça a operação
* por sua conta e risco. Se estiver errado, vai dar DUMP
* e ele vai rir da sua cara.
```

Eu achei bem bizarro esse negócio quando estava aprendendo, mas se você se imaginar no lugar do verificador de sintaxe, faz bastante sentido aquela interrogação ali. Serve até como um warning, como se o verificador estivesse dizendo: “eu entendi o que você quer fazer maaaaaas… não tenho como ter certeza se isso vai dar certo.”.

Finalizando

Hoje você descobriu que RTTS existe, que serve para fazer maluquices em tempo de execução, e que o operador “?=” serve para fazer down-casting. Nos vemos nos próximos posts, onde vamos analisar alguns exemplos de aplicação. A regra da dúvida continua valendo, ou seja, teve dúvida ~~você que se vire~~ poste nos comentários.

Espero que tenham gostado. Abraços a todos que já jogaram a culpa no programador (é com você que estou falando, verificador de sintaxe! 😈).

Comentários

> **Robson** — *11/12/2014 12:00*
>
> Fala Maurício, blz?
>
> Na verdade meu comentário foi em cima dos “castings”, esqueci mencionar… rs
>
> O que pra mim não ficou claro é em que situação surge a necessidade de utilizar. Acredito que com um exemplo real fique mais claro.
>
> Abraço!

> **Robson** — *10/12/2014 14:00*
>
> Você mostrou “como” fazer, mas para mim não ficou claro “porque” ou “quando” fazer. Talvez utilizando um exemplo real ficasse mais claro.

Se ficar complicado responder aqui (ou complementar o post), fica a sugestão da criação de um novo.

Valeu!

> **Mauricio Cruz** — *11/12/2014 09:31*
>
> Eae Robson!
>
> Concordo que na parte 1 me perdi um pouco na hora de explicar a motivação por trás do aprendizado do RTTS. Eu me empolguei tanto explicando que falhei ao mostrar exemplos claros...
>
> As partes 2 e 3 contém exemplos mais palpáveis do que é possível fazer com o RTTS. Se mesmo assim ficar com dúvidas, comente lá nos posts beleza?
>
> Valeu o comentário, vou levar em consideração para explicações futuras.
>
> Abraços!

Custodio — *30/04/2014 22:05*

Que bom ver post novo por aqui!

Muito bom o assunto, o conteudo e o otimo comentario do Fabio.

Bjundas

> **Mauricio Cruz** — *01/05/2014 09:50*
>
> Valeu mano da Austrália. Abs!

Fábio Pagoti — *29/04/2014 17:23*

O Zumbie saiu da terra novamente! Fico feliz por isso.

Falando um pouco mais do operador ?= (ou ?MOVE) note que incluir o ? simplesmente diz ao interpretador que a tentativa de atribuição deve ser realizada e que o código é valido.

Contudo, a valor das referência pode significar uma incompatibilidade de tipos (quando você tenta jogar um Galaxy em um IPhone por exemplo). Nestes casos apesar do seu código ativar ele dá um dump que pode ser tratado usando a cláusula catch em CX_SY_MOVE_CAST_ERROR.

Lembrando que o conceito de casting também é válido para interfaces e pode ser tratado da mesma forma.

> **Mauricio Cruz** — *01/05/2014 09:50*
>
> É meu, estamos acordando! 🙂
>
> Valeu por complementar o post Fábio, abraços!

Mateus Oliveira — *29/04/2014 13:15*

> Parabéns Maurício.
> Além do conteúdo muito bem explicado os comentários foram hilários.

> **Mauricio Cruz** — *29/04/2014 13:18*
>
> Valeus Mateus, fico feliz que tenha gostado! Abraços!

> **Henrique Dias** — *28/04/2014 14:43*
>
> Ótimo post, só faltou a tradução do RTTC e RTTI.

> **Mauricio Cruz** — *29/04/2014 07:02*
>
> Mano Henrique, eu já falei pra você parar de acessar o site bêbado. A explicação das siglas está logo no começo do post! haha Abs!

> **Henrique Dias** — *29/04/2014 13:19*
>
> Faltou a tradução na manolagem igual ao RTTS:
>
> RTTS – Runtime Type Services, ou “Paradas Legais para usar em Tempo de Execução”.
>
> RTTC: Runtime Type Creation: Criação dinâmica em tempo de execução.
>
> RTTI: Runtime Type Identification: Identificação de tipos em tempo de execução.

> **Mauricio Cruz** — *29/04/2014 13:20*
>
> Ah manjei, foi mal! É que acabou a inspiração na hora 😕