Artigo
Julio Esquerdo · Abr. 12, 2022 6min de leitura

Uso de variáveis locais no InterSystems IRIS

Julio Esquerdo e Lucio Mattos

Atualmente temos muitos recursos dentro do IRIS como por exemplo suporte REST, programação em Python, suporte a SQL, orientação a objetos, ML, IA, e uma infinidade de outras funcionalidades.

Mas aqueles que programam os códigos que serão utilizados, mesmo com os atuais recursos disponíveis, sabem que será necessário trazer dados para uma área de utilização, para posterior trabalho.

O IRIS tem suporte ao que chamamos de Variáveis Locais, que são estruturas armazenadas dentro do processo corrente, só estão disponíveis para o processo que as armazena e ao término do processo elas são eliminadas. Muitas vezes no decorrer do desenvolvimento de uma aplicação nos deparamos com algumas questões que podem ser facilmente resolvidas com o uso de variáveis locais.

A criação de uma variável local é muito simples:

```
>Set a=1
```

Pronto. Temos nossa variável criada. Ela não precisa ser previamente definida ou identificada. Ao longo do nosso processo, em qualquer ponto, podemos criar uma nova variável ou fazer referencia a uma já existente.

Para verificar a variável que criamos acima, podemos utilizar o comando ZWrite:

```
>ZWrite a
a=1
```

A princípio, a variável local não tem um tipo definido, sendo que podemos armazenar qualquer coisa dentro dela, um valor numérico ou uma string, e também podemos verificar o padrão dos dados armazenados com os verificadores de sintaxe:

```
>Write a?1N // Verifica se o conteúdo da variável é compativel com 1 Numérico
1

>Write a?1A // Verifica se o conteúdo da variável é compatível com 1 Alfabético
0
```

A variável local também pode ser indexada:

```
>Set a(1)=10
>Set a(2)=20
>ZWrite a
a(1)=10
a(2)=20
```

Podemos ter estruturas mais complexas armazenadas em variáveis locais:

```
>Set uf("RJ")="Rio de Janeiro"
>Set uf("ES")="Espírito Santo"
>Set uf("MG")="Minas Gerias"
>Set uf("SP")="São Paulo"
```

E podemos colocar mais informações na nossa variável, criando mais níveis de dados:

```
>Set uf("RJ","POP")=6748000
>Set uf("ES","POP")=3885000
>Set uf("MG","POP")=20870000
>Set uf("SP","POP")=40040000
```

E podemos recuperar facilmente a informação desejada dentro do nosso processo:

```
>Write uf("RJ,"POP")
6748000
```

Ou ainda, percorrer todas as entradas que existem para a nossa variável, e depois criar uma ordenação:

```
>Set n1="" For Set n1=$Order(uf(n1)) Quit:n1="" Set ufPop(uf(n1,"POP"),n1)=""
>Zwrite ufPop
ufPop(3885000,"ES")=""
ufPop(6748000,"RJ")=""
ufPop(20870000,"MG")=""
ufPop(40040000,"SP")=""
```

Podemos ainda verificar a existencia de uma determinada variável no nosso processo:

```
>Write $Data(uf("RS"))
0
```

E caso façamos referencia a uma variável que não existe, receberemos erro <UNDEFINED>:

```
>Write uf("AM")

WRITE uf("AM")
^
<UNDEFINED> *uf("AM")
```

Podemos utilizar as funções do IRIS como $ORDER, $QUERY, $ZORDER, $DATA e $GET para percorrer ou recuperar informações em variáveis locais.

As variáveis locais não são imutáveis. Podemos Incluir um novo item na nossa variável:

```
>Set uf("RS")="Rio Grande do Dul"
>ZWrite uf
uf("ES")="Espírito Santo"
uf("MG")="Minas Gerais"
uf("RJ")="Rio de Janeiro"
uf("RS")="Rio Grande do Dul"
uf("SP")="São Paulo"
```

Também podemos alterar uma informação na nossa variável:

```
>Set uf("RS")="Rio Grande do Sul"
>ZWrite uf
uf("ES")="Espírito Santo"
uf("MG")="Minas Gerais"
uf("RJ")="Rio de Janeiro"
uf("RS")="Rio Grande do Sul"
uf("SP")="São Paulo"
```

E podemos eliminar também:

```
>Kill uf("RS")
>ZWrite uf
uf("ES")="Espírito Santo"
uf("MG")="Minas Gerais"
uf("RJ")="Rio de Janeiro"
uf("SP")="São Paulo"
```

As variáveis locais podem auxiliar no nosso processo disponibilizando informações, organizando-as e melhorando a performance do nosso código.

Um caso: Imagine um processo que percorra a tabela de vendas de uma empresa e busque uma informação baseado na UF de entrega, por exemplo o nome dor extenso desta UF. Se o processo for buscar essa informação em uma estrutura armazenada em disco, mesmo com as capacidades de buffer do IRIS, esta operação pode acabar gerando uma demanda muito grande de I/O.

Nosso processo pode, logo no início, montar uma estrutura local com as informações das UFs e o processo passar a pegar tais informações desta variável local. Isto economizará muito acesso a disco.

Podemos também armazenar diversas informações em uma mesma variável, por exemplo, as vendas por UF mês a mês:

```
>Set ufVenda("RJ")="34.01,56.98,34.90,45.67,62.63,87.96,68.66,67.36,68.31,65.90,60.98,58.66"
>ZWrite ufVenda
ufVenda("RJ")="34.01,56.98,34.90,45.67,62.63,87.96,68.66,67.36,68.31,65.90,60.98,58.66"
```

E para resgatar um certo valor, por exemplo, as vendas de Agosto (mês 8):

```
>Write $Piece(ufVenda("RJ"),",",8)
67.36
```

Do ponto de vista de armazenamento, dentro da variável existe uma string. Nós a delimitamos com o caracter "," e cada espaço delimitado traz o valor de venda de um dado mês. O espaço 1 Janeiro, o espaço 2 Fevereiro e assim por diante. A função $Piece recupera aquele estaço delimitado conforme os parâmetros passados.

Também podemos utilizar o comando MERGE para juntar duas variáveis locais:

```
>Set ufA("RJ")="Rio de Janeiro"
>Set ufA("SP")="São Paulo"
>Set ufB("ES")="Espirito Santo"
>Set ufB("MG")="Minas Gerais"
>ZWrite ufA
ufA("RJ")="Rio de Janeiro"
ufA("SP")="São Paulo"zw ufB
>ZWrite ufB
ufB("ES")="Espirito Santo"
ufB("MG")="Minas Gerais"
>Merge ufA=ufB
>ZWrite ufA
ufA("ES")="Espirito Santo"
ufA("MG")="Minas Gerais"
ufA("RJ")="Rio de Janeiro"
ufA("SP")="São Paulo"
```

Podemos também usar as funções de Lista ($LIST, $LISTBUILD, $LISTDATA, $LISTFIND, $LISTFROMSTRING, $LISTGET, $LISTLENGTH, $LISTNEXT, $LISTSAME, $LISTTOSTRING, $LISTVALID) com variáveis locais, e as variáveis locais também armazenam objetos do IRIS, o que facilita muito nosso trabalho.

E, sim, existe um limite de tamanho máximo para uma variável local, que é 3.641.144 caracteres. Acima disso

receberemos erro <MAXSTRING>. Veja o exemplo com a variável var que foi criada com 3641144 caracteres, ou seja, o limite máximo possível para uma variável:

```
>Write $Length(var) // verifica o tamanho de uma variável
3641144

>Set var=var_"0" // concatena o caracter "0" a variavel var

SET var=var_"0"
^
<MAXSTRING>
>
```

É importante lembrar que o espaço disponível para a criação das variáveis locais é aquele disponível para o nosso processo. E também que quando nosso processo terminar as nossas variáveis serão eliminadas.

Como complemento ao estudo das variáveis locais, no IRIS os dados podem ser modelados e armazenados como Globais (arrays multidimensionais), objetos, tabelas, documentos, etc.

O interessante é que as estruturas de dados não necessitam de prévia declaração, definição ou alocaçãode espaço, elas passam a existir à medida que os dados são inseridos.

Outra curiosidade é que as estruturas de dados que suportam o paradigma {chave/valor} são intercambiáveis, inclusive com tipos de dados dinâmicos, como JSON. Observe as estruturas listadas a seguir:

```
Array multidimensional local
  cliente("NOME") = "João"
  cliente("ENDERECO") = "Rio de Janeiro"

Array multidimensional global
  ^cliente("NOME") = "João"
  ^cliente("ENDERECO") = "Rio de Janeiro"

JSON
  cliente = {"NOME":"João","ENDERECO":"Rio de Janeiro"}

  {global:"cliente",
  subscripts:"NOME",
  data:"João"
  }
  {global:"cliente",
  subscripts:"ENDERECO",
  data:"Rio de Janeiro"
  }
```

Então é isso. Aproveitem os recursos que o IRIS oferece para a trabalho com variáveis locais!

Um abraço!

#Concurso #Caché #InterSystems IRIS

URL de origem:https://pt.community.intersystems.com/post/uso-de-vari%C3%A1veis-locais-no-intersystems-iris