Artigo
Anonymous (não verificado) · Mar. 18, 2021 3min de leitura

Aproveitando ao máximo $ Query

Encontrei um caso de uso interessante do ObjectScript hoje com uma solução geral que gostaria de compartilhar.

Caso de uso:

Eu tenho uma matriz JSON (especificamente, no meu caso, uma matriz de problemas de Jira) que desejo agregar em alguns campos - digamos, categoria, prioridade e tipo de problema. Em seguida, desejo nivelar os agregados em uma lista simples com o total de cada um dos grupos. Claro, para a agregação, faz sentido usar uma matriz local na forma:

```
agg(category, priority, type) = total
```

De modo que, para cada registro na matriz de entrada, posso apenas:

```
Do $increment(agg(category, priority, type))
```

Mas depois de fazer a agregação, quero colocá-la em uma forma mais fácil de iterar, como uma matriz subscrita por inteiro:

```
summary = n
summary(1) = $listbuild(total1, category1, priority1, type1)
...
summary(n) = $listbuild(totalN, categoryN, priorityN, typeN)
```

Solução Básica:

A abordagem simples é ter três loops For aninhados com $ Order - por exemplo:

```
Set category = ""
For {
    Set category = $Order(agg(category))
    Quit:category=""

    Set priority = ""
    For {
        Set priority = $Order(agg(category,priority))
        Quit:priority=""

        Set type = ""
        For {
            Set type = $Order(agg(category,priority,type),1,total)
            Quit:type=""

            Set summary($i(summary)) = $listbuild(total,category,priority,type)
        }
    }
```

Foi com isso que comecei, mas é muito código, e se eu tivesse mais dimensões para agregar sobre ele ficaria complicado rapidamente. Isso me fez pensar - existe uma solução geral para realizar a mesma coisa? Acontece que existe!

Melhor solução com $ Query:

Decidi que usar $ query ajudaria. Observe que esta solução assume uma profundidade uniforme de subscritos / valores em todo o array local; coisas estranhas aconteceriam se essa suposição fosse violada.

```
ClassMethod Flatten(ByRef deep, Output flat) [ PublicList = deep ]
{
    Set reference = "deep"
    For {
        Set reference = $query(@reference)
        Quit:reference=""
        Set value = $listbuild(@reference)
        For i=1:1:$qlength(reference) {
            Set value = value_$listbuild($qsubscript(reference,i))
        }
        Set flat($i(flat)) = value
    }
}
```

Portanto, o snippet acima é substituído por:

```
Do ..Flatten(.agg,.summary)
```

Algumas coisas a serem observadas sobre esta solução:

deep precisa estar na PublicList para que $ query possa operar nela em cada iteração, a referência é alterada para fazer referência ao próximo conjunto de subscritos em profundidade que tem um valor - por exemplo, o valor pode ser: deep ("foo", "bar") $ qlength retorna o número de subscritos em referência $ qsubscript retorna o valor do i'ésimo subscrito de referência Quando as listas $ listbuild são concatenadas, o resultado é uma lista $ listbuild válida com as listas combinadas (isso é muito melhor do que usar qualquer outro delimitador!) Resumo $ query, $ qlength e $ qsubscript são úteis para lidar com matrizes globais / locais de profundidade arbitrária.

Leitura Adicional

$Query:
https://docs.intersystems.com/irisforhealthlatest/csp/docbook/DocBook.UI.Page.cls?KEY=RCOSFQUERY

$QSubscript: https: //docs.intersystems.com/irisforhealthlatest/csp/docbook/Doc.View.c ...

$QLength: https: //docs.intersystems.com/irisforhealthlatest/csp/docbook/Doc.View.c ...

#Caché

URL de origem:https://pt.community.intersystems.com/post/aproveitando-ao-m%C3%A1ximo-query