Subject: Re: !null values in arrays
Posted by David Fanning on Wed, 15 Dec 2010 18:20:13 GMT
View Forum Message <> Reply to Message

Gray writes:

> However, writing this post is worth it just to include that
> tautological error message.

Now I have to go looking for my glasses *and* my dictionary. :-(

Cheers,

David

--
David Fanning, Ph.D.
Fanning Software Consulting, Inc.
Coyote's Guide to IDL Programming: http://www.dfanning.com/
Sepore ma de ni thui. ("Perhaps thou speakest truth.")

Subject: Re: !null values in arrays
Posted by on Thu, 16 Dec 2010 11:30:47 GMT
View Forum Message <> Reply to Message

Hi Gray,

I see your point and agree to some extent. But I would also see the opposite arguments: what should then be the result of 3 * !NULL or 3 / !NULL ? Do you propose that !NULL is handled as a 0? This might often lead to more confusion, and for indexing purposes it is definitely not a 0.

The eventual advantage is probably not worth it...

Subject: Re: !null values in arrays
Posted by Paul Van Delst[1] on Thu, 16 Dec 2010 15:14:00 GMT
View Forum Message <> Reply to Message

Hello,

Axel M wrote:
> Hi Gray,
>

> I see your point and agree to some extent. But I would also see the
> opposite arguments: what should then be the result of 3 * !NULL or
> 3 / !NULL ? Do you propose that !NULL is handled as a 0? This might
> often lead to more confusion, and for indexing purposes it is
> definitely not a 0.

In IDL, storing a null in an otherwise homogeneous array makes said array a list. Why not use a list? The OP could still
call it an array.... :o)

Purely for perspective into the issue, ruby handles operations with null values thusly:

```
irb(main):001:0> 3 + nil
TypeError: nil can't be coerced into Fixnum
        from (irb):1:in `+'
        from (irb):1
```

and python similarly:

```
>>>  3 + None
Traceback (most recent call last):
  File "<stdin>", line 1, in ?
TypeError: unsupported operand type(s) for +: 'int' and 'NoneType'
```

cheers,

paulv

Subject: Re: !null values in arrays
Posted by penteado on Thu, 16 Dec 2010 15:39:54 GMT
View Forum Message <> Reply to Message

On Dec 16, 1:14 pm, Paul van Delst <paul.vande...@noaa.gov> wrote:
> Purely for perspective into the issue, ruby handles operations with null values thusly:
>
> irb(main):001:0> 3 + nil
> TypeError: nil can't be coerced into Fixnum
>   from (irb):1:in `+'
>   from (irb):1
>
> and python similarly:
>
>>>> 3 + None
>
> Traceback (most recent call last):

> File "<stdin>", line 1, in ?
> TypeError: unsupported operand type(s) for +: 'int' and 'NoneType'

Both appropriate. !null and NaN are not the same, and are not the same as 0. In fact, their lack in more primitive languages is an important hassle.

0 is not, intrinsically, the same as missing data. In particular, most mathematical operations on NaNs (say, adding a value to a NaN) should always return NaN to make sense. If an array has missing data, data cannot be magically created on the missing elements because, say, a scalar was added to the array.

NaN is special in the sense that it is a floating point value, part of the IEEE 754 standard, and properly recognized by hardware and other software. !null is a language construct, which necessarily varies among languages, as they vary in what constitutes a variable. It is an undefined variable, so one can assign to it, but not operate on it, the same way that one cannot operate on an undefined variable, but can create it by assignment.