

Re: Enumerated types

Source:

<http://www.tech-archive.net/Archive/Access/microsoft.public.access.formscoding/2007-05/msg00112.html>

- *From:* Peter Hallett <PeterHallett@xxxxxxxxxxxxxxxxxxxxxxxxxxxxxx>
 - *Date:* Wed, 2 May 2007 11:33:02 -0700
-

Thanks for the prompt reply, Scott. I now understand enumerated types rather better. It seems as if accessing an array by index is, after all, the best way of going about the job but I have to initialize the thing and it all looks clumsy. It would be almost as easy to forget the For...Next loop and set the required values in four sequential explicit statements. That, too, doesn't look too elegant but it would be much worse with 40 values to set, instead of 4.

Come back Kernighan & Ritchie. All is forgiven. I must be getting old.

--

Peter Hallett

"Scott McDaniel" wrote:

On Wed, 2 May 2007 10:35:01 -0700, Peter Hallett
<PeterHallett@xxxxxxxxxxxxxxxxxxxxxxxxxxxxxx> wrote:

I have clearly failed to understand the syntax of enumerated types. Having declared :-

```
Enum ACGroup
A = 1
B
C
D
End Enum
```

I cannot then access the members of the group. With J as a variable byte, the compiler baulks at ACGroup.J or ACGroup(J). I simply wish to return A&D for values of J between 1 and 4, respectively.

Be definition an "enumeration" would return only numeric values. AFAIK, there is no way to have an Enum return Text

Re: Enumerated types

values. You could build a class or function for this, as you probably know, but there is no way to do this:

```
ACGroup(1)
```

and have that return "A"

There are other ways of doing this, of course, and, in this trivial example, probably better ways, but it would be nice to have this option available.

One alternative I could try, in the mean time, is to declare a string array and access its members using J as an index. As an old (=elderly) C programmer, I also tried `For stString = A To D &`, but, hardly surprisingly, VBA was having none of it. Would anyone be kind enough to clarify the situation for me?

Don't believe you can use the `For-Next` or `For Each` syntax with an array, unless you work with the `Ubound` or `Lbound` of the array (although I could be wrong about that). You could do this in a Standard Module:

```
[General Declaration]
```

```
dim arr() As String
```

```
Dim i as integer
```

```
Function LoadArray() As Boolean
```

```
  '/redim and load the array
```

```
  redim arr(3)
```

```
  arr(0)="A"
```

```
  arr(1)="B"
```

```
  arr(2)="C"
```

```
  arr(3)="D"
```

```
End Function
```

```
Function GetArrayValue(ArrayIndex as Integer) as String
```

```
  If ArrayIndex <= ubound(arr) Then GetArrayValue= arr(ArrayIndex)
```

```
End If
```

You'd have to call `LoadArray` when the app started, then anywhere you needed a value from the array just call

```
GetArrayValue(xx)
```

Scott McDaniel

```
scott@xxxxxxxxxxxxxxxxxxxxxxxxxxxx
```

Re: Enumerated types

www.infotrakker.com