

Re: Determine if an item in a collection has changed

Source:

<http://www.tech-archive.net/Archive/VB/microsoft.public.vb.general.discussion/2005-11/msg00848.html>

- *From:* "Dmitriy Antonov" <antonovdima@xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx>
 - *Date:* Tue, 15 Nov 2005 17:51:45 -0500
-

I am not sure I understand you questions correctly and I suspect it has nothing in common to your original post.

So for your general questions (as I understand them) I can give you my own general personal opinions, which probably different from opinions of many other peoples.

I don't remember that mentioned book has strong emphasis on ADO (if any). In fact, I still create my objects grounded on Lhotka's recommendations (while seriously revised). Any book has a big disadvantage – it never has complete solution for your own situation (and it applies to any scope of knowledge). And there is a reason for it – number of different situations greatly exceeds number of book authors (thanks god). I mean, if you read the book you understand that you'll definitely should make, at least some, unique design considerations in your application.

What people do with API – I don't know. Some use API, some not. I use (I mean Win API). Comparison between recordset and Win API, I would say, is not relevant. It is like if I try to compare my apartment and my car: I pay something for both of them and I use both but for very different purposes (even though they both have doors and windows). So I do use Win API, I do use recordsets as well as many other structures, including my own ones.

Parent–Child relationships in terms of OOP and circular references, as I think, is relatively frequent topic. BTW, building of collection classes is described in MSDN help.

The method of using object pointers to resolve circular references is described in Advanced Visual Basic 6 (by Matthew Curland). As far as I know he is referred as original source of this technique. There are a lot of advanced things in this book (unfortunately he overuses his own library, which usually complicates understanding of many topics). Again, I think, this method is used very frequently (you can find it in many free samples on Internet) and often referred in different VB conversations

Main idea – everyone's way is different, while it is always based on some known and commonly used principles. What's interesting, you often can get

Re: Determine if an item in a collection has changed

the same result using many different ways. And many of these ways can be considered as rational ones while none is generally an ideal one.

I don't know whether I answered your questions and I don't think you actually expected to get something more concrete.

Dmitriy.

"Damon Anderson" <someone@xxxxxxxxxxxxxx> wrote in message
news:ej2Pu8i6FHA.2608@xxxxxxxxxxxxxxxxxxxxxxxxxxxx

> Dmitriy-

>

> Thanks for the reply.

>

> How do others handle writing APIs for their application? Do people simply
> not use APIs? Rather, does everyone slog Recordsets around?

>

> While I would have thought what I am trying to do is pretty common, I
> haven't seen any postings or articles about this. The closest is Rocky
> Lhotka's VB6 Business Objects. While he's done a good job with it, I
> still feel that there is too much ADO exposed to the client.

>

> Thoughts?

>

> Craig

>

> "Dmitriy Antonov" <antonovdima@xxxxxxxxxxxxxxxxxxxxxxxxxxxx> wrote in message
> news:e3ywx4i6FHA.1248@xxxxxxxxxxxxxxxxxxxxxxxxxxxx

>> Yes, callbacks is an option. But you should be aware of one important
>> nuance: this will require your Person objects to hold a reference to its
>> parent, which by necessity holds a reference to each of its children (via
>> internal Collection object). This introduces circular reference, which
>> will hold all objects in memory even if you release all top-level
>> references to any of these objects. Memory leak is guaranteed (and often
>> very serious one).

>>

>> I know two methods to solve this problem.

>>

>> 1. The easiest, but not recommended. Add public method to your Persons
>> collection (something like CleanRefs). Within this method you should
>> remove all references from children to parent (generally you use simple
>> Friend method like ReleaseParent or directly Set Parent=Nothing, assuming
>> Parent is a property in Person class).

>> Why I don't recommend this method – because it requires explicit call to
>> that method just before destroying reference to Persons object from the
>> last holder (and only from the last one). It opens the door for human
>> mistakes, which are not immediately visible.

>>

>> 2. More complex. Instead of references to parent objects, all children

Re: Determine if an item in a collection has changed

```
>> hold its pointer (obtained using ObjPtr). You can resolve the object from
>> that pointer using CopyMemory. E.g.
>>
>> Public Function ObjectFromPtr(IPtr As Long) As Object
>> If IPtr <= 0 Then Exit Function
>> Dim obj As Object
>> CopyMemory ByVal ObjPtr(obj), IPtr, 4&
>> Set ObjectFromPtr = obj
>> CopyMemory ByVal ObjPtr(obj), 0&, 4&
>> Set obj = Nothing
>> End Function
>>
>> Be careful here, if pointer points to already destroyed object then GPF
>> or something equally bad is guaranteed.
>>
>> Since it is possible that some of the children objects can be held
>> somewhere outside collection it is still recommended to assign zero to
>> all those pointers in collection's Terminate event, but there is no need
>> to call anything explicitly and thus all objects will be successfully
>> destroyed after the last reference is released.
>>
>> Dmitriy.
>>
>>
>> "Craig Buchanan" <someone@xxxxxxxxxxxxxx> wrote in message
>> news:%23HOCegi6FHA.3544@xxxxxxxxxxxxxxxxxxxxxxxxxxxx
>>> Dmitriy-
>>>
>>> Thanks for the response.
>>>
>>> Let's say that i have two classes: Persons and Person.
>>>
>>> The Persons class is a Collection class. It has two relevant methods:
>>> Add and Save. The Add requires a Person reference. The Save persists
>>> all objects in the collection that have changed (either by a batch
>>> update/insert or calling each Person instances Save method).
>>>
>>> The Person class is a custom object. It has two relevant, Friend
>>> members: IsDirty property and Save method. If I make a change to the
>>> Person or create a new person, IsDirty is true. The Save method manages
>>> the insert/update.
>>>
>>> I could look at the state of the Person instance in the Add method and
>>> set internal state accordingly. However, I'm more concerned about
>>> changes to an item already in the collection. For instance:
>>>
>>> ...
>>> Dim PP as New Persons
>>> PP.Load 'load from db
>>>
>>> Dim P as Person: Set P = PP("#" & 10)
```

Re: Determine if an item in a collection has changed

```
>>> With P
>>> .FirstName="Craig"
>>> ...
>>> End With
>>> Set P = Nothing
>>>
>>> Set P = New Person
>>> With P
>>> ...
>>> End With
>>>
>>> PP.Add P
>>> Set P = nothing
>>>
>>> PP.Save
>>>
>>> ...
>>>
>>> Obviously, in PP.Save, i could enumerate each Person instance and call
>>> the .Save method if the instance needed persistance, but i'd rather be
>>> smart about it.
>>>
>>> As i'm not creating an explicit WithEvents reference for each Person in
>>> the Persons class, raising an event is out. I guess that leaves me with
>>> classbacks.
>>>
>>> Am I understanding the situation correctly?
>>>
>>> Craig
>>>
>>> "Dmitriy Antonov" <antonovdima@xxxxxxxxxxxxxxxxxxxxxxxxxxxx> wrote in
>>> message news:eALBbp05FHA.4036@xxxxxxxxxxxxxxxxxxxxxxxxxxxx
>>>>
>>>> "Craig Buchanan" <someone@xxxxxxxxxxxxxx> wrote in message
>>>> news:%23dZ2o0z5FHA.3188@xxxxxxxxxxxxxxxxxxxxxxxxxxxx
>>>>> It's there an easy way to determine if one or more objects in a
>>>>> collection has changed, without enumerating the collection and
>>>>> checking each object individually? Is this a case for call backs?
>>>>>
>>>>> Thanks for the help.
>>>>>
>>>>>
>>>>> If you mean an object of type Collection then it depends, what kind of
>>>>> items it contains. There is no way to notify about such changes if it
>>>>> contains items of primitive datatypes (yes you still can enumerate them
>>>>> and compare with some previous state as long as you preserve that state
>>>>> in some way). If it contains Objects created from classes, where such
>>>>> functionality is implemented than you can use that functionality. If
>>>>> you are talking about objects, which you create (or going to create)
>>>>> then you have many different options of doing this. You can add some
>>>>> property, like IsDirty or Changed, into your class and/or add event,
```

