

microsoft.public.vb.general.discussion: Re: Collection class that doesn't require scripting runtime ?

## Re: Collection class that doesn't require scripting runtime ?

**Source:** <http://www.tech-archive.net/Archive/VB/microsoft.public.vb.general.discussion/2004-02/4046.html>

---

**From:** Joe \ (joe\_at\_bftsi0.UUCP)

**Date:** 02/24/04

Date: Tue, 24 Feb 2004 10:19:16 -0800

"Alan Silver" <alan-silver@nospam.thanx> wrote in message  
<news:vLAAXHC5j2OAFwPE@nospamthankyou.spam>...

- > *Some time ago, I remember seeing a post that mentioned a fast collection*
- > *class that didn't require the scripting runtime DLL to be referenced,*
- > *but I don't have the URL any more.*
- >
- > *Anyone know where I could find one ? Basically I just want to store*
- > *pairs of key/item data, all strings, no objects. If there's a better way*
- > *of doing this, please advise me.*

If you don't need to access items by index, try something like this:

```
-----  
VERSION 1.0 CLASS  
BEGIN  
    MultiUse = -1    'True  
END  
Attribute VB_Name = "Dorktionary"  
Attribute VB_GlobalNameSpace = False  
Attribute VB_Creatable = True  
Attribute VB_PredeclaredId = False  
Attribute VB_Exposed = False  
' EXPERIMENTAL Class Dorktionary  
Option Compare Binary  
Option Explicit: DefObj A-Z  
Private Const Nil = 0  
Private Enum AddMode  
    AddAbsent  
    AddAlways  
    AddNever  
End Enum  
Private Type Node  
    Key As String  
    Item As Variant  
    LT As Long  
    GT As Long  
    Parent As Long  
    Priority As Single  
End Type  
Private Type Stash
```

Re: Collection class that doesn't require scripting runtime ?

## microsoft.public.vb.general.discussion: Re: Collection class that doesn't require scripting runtime ?

```
Compare As VbCompareMethod
Count As Long
Data() As Node
Head As Long
Free As Long
End Type
Private This As Stash
Public Sub Add(ByVal Key As String, ByVal Item As Variant)
    Dim Curr As Long: Curr = Find(Key, AddAlways)
    Debug.Assert Curr > Nil
    If Curr = Nil Then Error 5
    UpHeap Curr
    ' VarSwap This.Data(Curr).Item, Item
    If IsObject(Item) Then Set This.Data(Curr).Item = Item Else This.Data(Curr).Item = Item
End Sub
Public Sub AddItem(ByVal Item As Variant, ByVal Key As String)
    Dim Curr As Long: Curr = Find(Key, AddAlways)
    Debug.Assert Curr > Nil
    If Curr = Nil Then Error 5
    UpHeap Curr
    ' VarSwap This.Data(Curr).Item, Item
    If IsObject(Item) Then Set This.Data(Curr).Item = Item Else This.Data(Curr).Item = Item
End Sub
Public Property Get Compare() As VbCompareMethod
    Compare = This.Compare
End Property
Public Property Let Compare(ByVal NewValue As VbCompareMethod)
    Debug.Assert This.Count = 0
    If This.Count > 0 Then Error 5
    This.Compare = NewValue
End Property
Public Property Get Count() As Long
    Debug.Assert This.Count = CountAll(This.Head)
    Count = This.Count
End Property
' Exhaustive data structure integrity checker
' (but this doesn't check the free list!)
Private Function CountAll(ByVal Curr As Long) As Long
    If Curr = Nil Then Exit Function
    Debug.Assert This.Data(Curr).Priority > 0
    Debug.Assert This.Data(Curr).Parent = Nil Eqv This.Head = Curr
    Debug.Assert This.Data(Curr).Parent <> Curr And This.Data(Curr).LT <> Curr And This.Data(Curr).
    If This.Data(Curr).LT > Nil Then Debug.Assert This.Data(This.Data(Curr).LT).Parent = Curr
    If This.Data(Curr).GT > Nil Then Debug.Assert This.Data(This.Data(Curr).GT).Parent = Curr
    If This.Data(Curr).LT > Nil Then Debug.Assert This.Data(This.Data(Curr).LT).Priority <= This.Da
    If This.Data(Curr).GT > Nil Then Debug.Assert This.Data(This.Data(Curr).GT).Priority <= This.Da
    If This.Data(Curr).LT > Nil Then Debug.Assert StrComp(This.Data(This.Data(Curr).LT).Key, This.D
    If This.Data(Curr).GT > Nil Then Debug.Assert StrComp(This.Data(This.Data(Curr).GT).Key, This.D
    CountAll = 1 + CountAll(This.Data(Curr).LT) + CountAll(This.Data(Curr).GT)
End Function
Public Function Exists(ByVal Key As String) As Boolean
    Dim Curr As Long: Curr = Find(Key, AddNever)
    If Curr > Nil Then UpHeap Curr
    Exists = Curr > Nil
End Function
Private Function Find(Key As String, ByVal AddMode As AddMode) As Long
    Dim Curr As Long, Prev As Long, Left As Boolean
    Debug.Assert This.Head > Nil Eqv This.Count > 0
    Curr = This.Head
    Do Until Curr = Nil
        Debug.Assert This.Data(Curr).Parent = Prev
        Select Case StrComp(Key, This.Data(Curr).Key, This.Compare)
```

microsoft.public.vb.general.discussion: Re: Collection class that doesn't require scripting runtime ?

```
Case Is < 0
    Prev = Curr
    Curr = This.Data(Curr).LT
    Left = True
Case Is > 0
    Prev = Curr
    Curr = This.Data(Curr).GT
    Left = False
Case Else
    Exit Do
End Select
Loop
Select Case AddMode
Case AddAbsent
    If Curr <> Nil Then
        Find = Curr
        Exit Function
    End If
Case AddAlways
    If Curr <> Nil Then
        Find = Nil
        Exit Function
    End If
Case AddNever
    Find = Curr
    Exit Function
End Select
Debug.Assert Curr = Nil
Debug.Assert AddMode = AddAbsent Or AddMode = AddAlways
Curr = GetOne()
This.Data(Curr).Key = Key ' StrSwap This.Data(Curr).Key, Key
This.Data(Curr).Parent = Prev
If Prev = Nil Then
    Debug.Assert This.Head = Nil
    This.Head = Curr
ElseIf Left Then
    Debug.Assert This.Data(Prev).LT = Nil
    This.Data(Prev).LT = Curr
Else
    Debug.Assert This.Data(Prev).GT = Nil
    This.Data(Prev).GT = Curr
End If
This.Count = This.Count + 1
Find = Curr
End Function
Private Function GetOne() As Long
If This.Free = Nil Then
    If This.Head = Nil Then
        This.Free = Nil + 1
    Else
        This.Free = UBound(This.Data) + 1
    End If
    ReDim Preserve This.Data(Nil + 1 To This.Free + This.Free \ 2 + 4)
    Dim i As Long
    For i = 1 To This.Free - 1
        Debug.Assert This.Data(i).Priority > 0
    Next
    For i = This.Free To UBound(This.Data) - 1
        This.Data(i).Parent = i + 1
    Next
End If
Debug.Assert This.Data(This.Free).Priority = 0
```

## microsoft.public.vb.general.discussion: Re: Collection class that doesn't require scripting runtime ?

```
    Debug.Assert This.Data(This.Free).Parent >= Nil
    GetOne = This.Free
    This.Free = This.Data(This.Free).Parent
End Function
Public Property Get Item(ByVal Key As String) As Variant
Attribute Item.VB_UserMemId = 0
    Dim Curr As Long: Curr = Find(Key, AddNever)
    If Curr = Nil Then Exit Property
    UpHeap Curr
    With This.Data(Curr)
        If IsObject(.Item) Then Set Item = .Item Else Let Item = .Item
    End With
End Property
Public Property Let Item(ByVal Key As String, ByVal Item As Variant)
    Dim Curr As Long: Curr = Find(Key, AddAbsent)
    Debug.Assert Curr > Nil
    UpHeap Curr
    ' VarSwap This.Data(Curr).Item, Item
    If IsObject(Item) Then Set This.Data(Curr).Item = Item Else This.Data(Curr).Item = Item
End Property
Public Property Set Item(ByVal Key As String, ByVal Item As Variant)
    Dim Curr As Long: Curr = Find(Key, AddAbsent)
    Debug.Assert Curr > Nil
    UpHeap Curr
    ' VarSwap This.Data(Curr).Item, Item
    If IsObject(Item) Then Set This.Data(Curr).Item = Item Else This.Data(Curr).Item = Item
End Property
Public Function Items() As IUnknown
    Dim C As Collection
    Set C = New Collection
    Dim i As Long
    If This.Count > 0 Then
        For i = LBound(This.Data) To UBound(This.Data)
            If This.Data(i).Priority > 0 Then C.Add This.Data(i).Item
        Next i
    End If
    Set Items = C
End Function
Public Function Keys() As IUnknown
    Dim C As Collection
    Set C = New Collection
    Dim i As Long
    If This.Count > 0 Then
        For i = LBound(This.Data) To UBound(This.Data)
            If This.Data(i).Priority > 0 Then C.Add This.Data(i).Key
        Next i
    End If
    Set Keys = C
End Function
Public Function NewEnum() As IEnumVARIANT
Attribute NewEnum.VB_UserMemId = -4
Attribute NewEnum.VB_MemberFlags = "40"
    Dim C As Collection: Set C = Items
    Set NewEnum = C.[_NewEnum]
End Function
Public Sub Remove(ByVal Key As String)
    Dim Curr As Long: Curr = Find(Key, AddNever)
    Debug.Assert Curr > Nil
    If Curr = Nil Then Error 5
    This.Data(Curr).Priority = 0
    While This.Data(Curr).LT > Nil And This.Data(Curr).GT > Nil
        If This.Data(This.Data(Curr).LT).Priority > This.Data(This.Data(Curr).GT).Priority Then
```

## microsoft.public.vb.general.discussion: Re: Collection class that doesn't require scripting runtime ?

```
        UpOne This.Data(Curr).LT
    Else
        UpOne This.Data(Curr).GT
    End If
Wend
Debug.Assert This.Data(Curr).LT = Nil Or This.Data(Curr).GT = Nil
Dim Prev As Long: Prev = This.Data(Curr).Parent
Debug.Assert Prev = Nil Eqv This.Head = Curr
Dim Chld As Long: Chld = Nil
If This.Data(Curr).LT > Nil Then
    Chld = This.Data(Curr).LT
    This.Data(Chld).Parent = Prev
ElseIf This.Data(Curr).GT > Nil Then
    Chld = This.Data(Curr).GT
    This.Data(Chld).Parent = Prev
End If
If Prev = Nil Then
    Debug.Assert Chld = Nil Eqv This.Count = 1
    This.Head = Chld
ElseIf This.Data(Prev).LT = Curr Then
    This.Data(Prev).LT = Chld
Else
    Debug.Assert This.Data(Prev).GT = Curr
    This.Data(Prev).GT = Chld
End If
This.Count = This.Count - 1
With This.Data(Curr)
    .Key = vbNullString
    .Item = Empty
    .LT = Nil
    .GT = Nil
    .Parent = This.Free
    Debug.Assert .Priority = 0
End With
This.Free = Curr
End Sub
Public Sub RemoveAll()
    Erase This.Data
    This.Count = 0
    This.Head = Nil
    This.Free = Nil
End Sub
Private Sub UpHeap(ByVal Curr As Long)
    Debug.Assert Curr > Nil
    Dim Priority As Single: Priority = Rnd + 1
    Debug.Assert Priority > 0
    If Priority <= This.Data(Curr).Priority Then Exit Sub
    This.Data(Curr).Priority = Priority
    Do
        Dim Prev As Long: Prev = This.Data(Curr).Parent
        If Prev = Nil Then
            Exit Do
        ElseIf Priority < This.Data(Prev).Priority Then
            Exit Do
        Else
            UpOne Curr
        End If
    Loop
End Sub
Private Sub UpOne(ByVal Curr As Long)
    Debug.Assert Curr > Nil
    Debug.Assert This.Data(Curr).Parent > Nil
```

microsoft.public.vb.general.discussion: Re: Collection class that doesn't require scripting runtime ?

```
Dim Prev As Long: Prev = This.Data(Curr).Parent
Dim PPrv As Long: PPrv = This.Data(Prev).Parent
Debug.Assert This.Data(Curr).Priority >= This.Data(Prev).Priority
Debug.Assert This.Data(Prev).LT = Curr Or This.Data(Prev).GT = Curr
Dim Chld As Long
If This.Data(Prev).LT = Curr Then
    Chld = This.Data(Curr).GT
    This.Data(Curr).GT = Prev
    This.Data(Prev).LT = Chld
Else
    Debug.Assert This.Data(Prev).GT = Curr
    Chld = This.Data(Curr).LT
    This.Data(Curr).LT = Prev
    This.Data(Prev).GT = Chld
End If
If Chld > Nil Then This.Data(Chld).Parent = Prev
This.Data(Prev).Parent = Curr
This.Data(Curr).Parent = PPrv
If PPrv = Nil Then
    Debug.Assert This.Head = Prev
    This.Head = Curr
ElseIf This.Data(PPrv).LT = Prev Then
    This.Data(PPrv).LT = Curr
Else
    Debug.Assert This.Data(PPrv).GT = Prev
    This.Data(PPrv).GT = Curr
End If
End Sub
-----
--
Joe Foster <mailto:jlfoster%40znet.com>      On the cans? <http://www.xenu.net/>
WARNING: I cannot be held responsible for the above      They're coming to
because my cats have apparently learned to type.        take me away, ha ha!
```