Learning VB.Net

Tutorial 10 – Collections

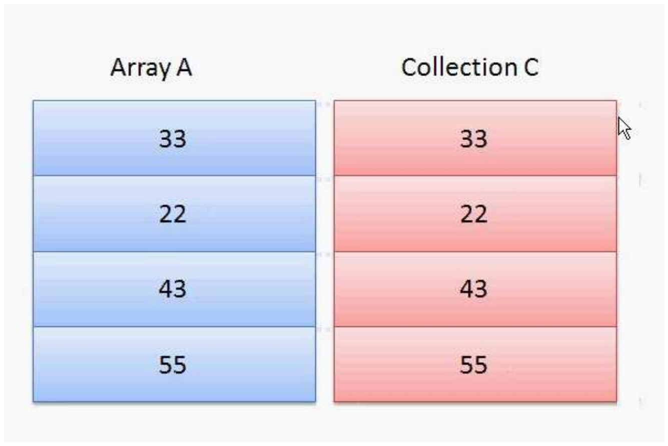
Hello everyone… welcome to vb.net tutorials. These are going to be very basic tutorials about using the language to create simple applications, hope you enjoy it. If you have any notes about it, please send them to notes@mka-soft.com I will be happy to answer them. Finally if you find these tutorials are useful, it would be nice from you to send a small donation via PayPal to donation@mka-soft.com.

Tutorial posted on 2010-Feb-03.

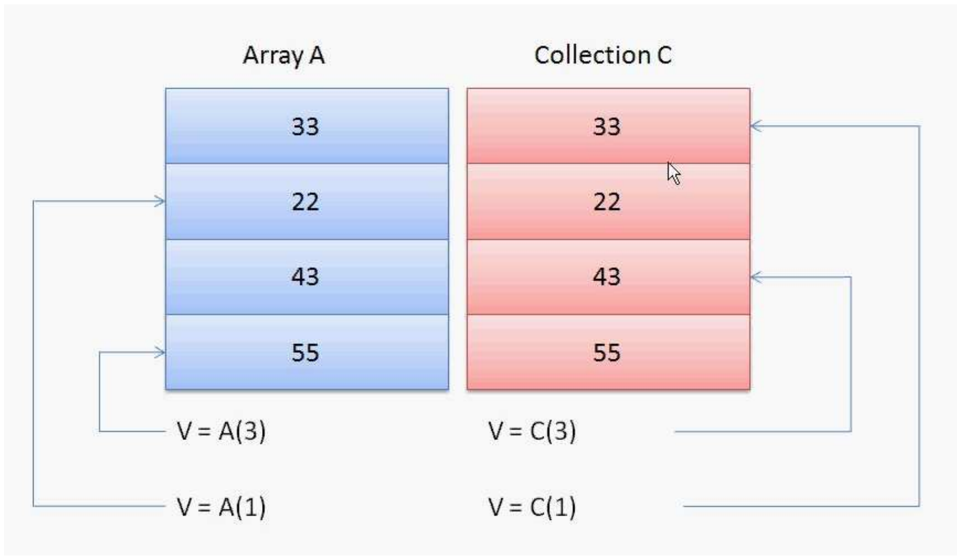
Collections

Last time we spoke about arrays, and saw how to work with them. Today we check out something similar and easier to use, and that is collections.

Collections are very similar to arrays. They are used to store a number of values (or variables), so that you can process them all.



There are a number of differences between arrays and collections. First difference is that the indexing for arrays starts with zero, while for a collection it starts with 1. To understand this, consider the image below:

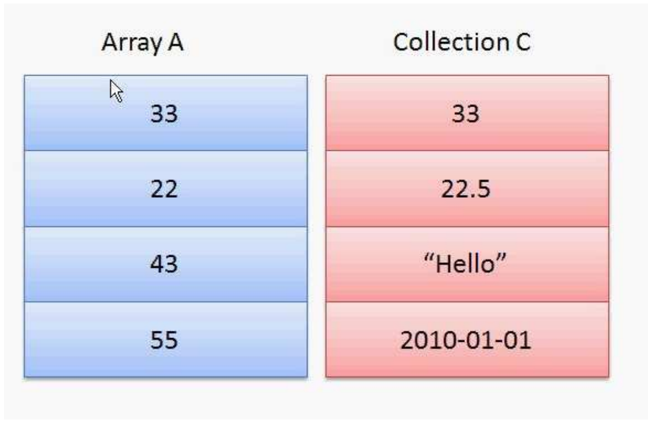


Now the statement V = A(3) will place the value 55 in V because the indexing start at zero in arrays. However the similar statement V = C(3) will place 43 instead because collection indexing is different. The same applies for the second statement.

Another important difference is the data type. All Array elements has the same data type. So if you have an array:

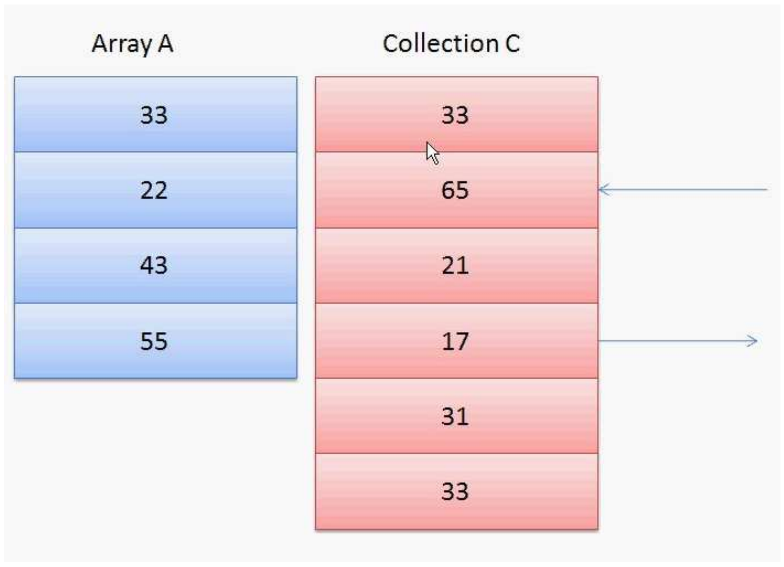
```
Dim A(0 To 9) As Integer
```

then A(0), A(1), A(2)… A(9) are all Integers. Collections on the other hand do not require this. You can store integers, reals, strings, bytes,… etc. in the same collection. This can be illustrated below:

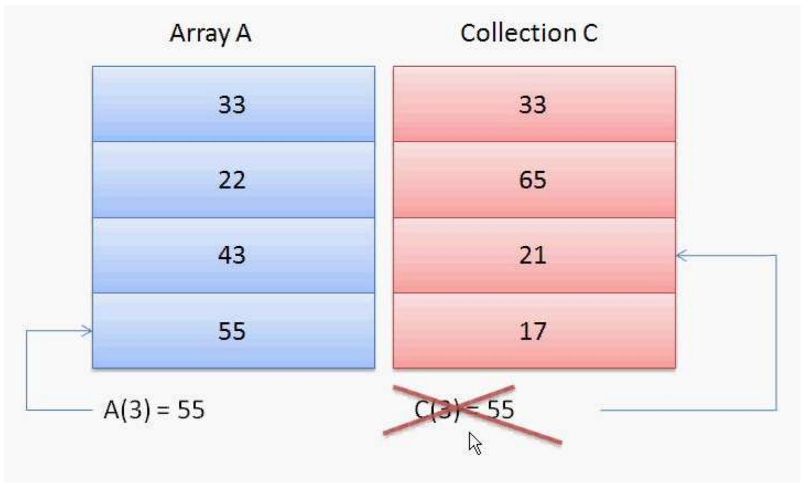


Usually you will use the same data type for all elements of the collection; however you still have the option to use different data types whenever you need to.

Another difference between arrays and collections is that collections can add elements and remove elements directly and change in size without any need for some kind of processing, while arrays are fixed in size, and you cannot insert values at specific locations at random.



Finally array elements can be updated and overwritten, while collections are not. To make this clear check out this:



In this example if you write C(3) = 55 you get an error, that is because collection does not allow you to update or overwrite the content of an element. However there is a way to overcome this. We will discuss this later.

Now let us see how to define a collection. There are two ways to define a collection:

```
    Dim C As New Collection
```

In this example you create a collection object that is ready to be used. So you can add, remove elements, get the number of items, or do whatever you want with the collection directly. Another way is:

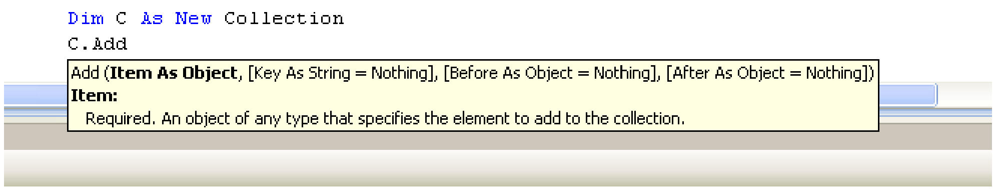
```
    Dim C As Collection
```

here C is not ready yet to be used. It does not point to a collection yet. You can never use it. To be able to use it later on in the code you should write:

```
    C = New Collection
```

This will allow you to use the collection without any problem. The first method of defining a collection is the one you will probably use the most.

Now in order to add elements to the collection you use the add method.



```
Dim C As New Collection
C.Add
```

Add (**Item As Object**, [Key As String = Nothing], [Before As Object = Nothing], [After As Object = Nothing])
Item:
Required. An object of any type that specifies the element to add to the collection.

When you write C.Add the compiler shows you the parameters that you should provide. The ones in the square brackets are optional. The parameters are:

- Item: is the value you want to store
- Key: you can provide a text value to quickly access the elements of collection instead of providing numbers to access them. For example the code below will store the value 40 in V:

```
Dim C As New Collection
C.Add(33, "Smith")
C.Add(40, "Michel")
C.Add(77, "John")

Dim V As String
V = C("Michel")
```

- Before: is the index of the item you want the new element to be inserted before.
- After: is the index of the item you want the new element to be inserted after.

Next is removing elements. This simple, you just provide the index of the element you want to remove:

```
C.Remove(3)
```

This removes the 3rd element from the collection.

Also getting the number of elements is as easy. You use the count function.

```
I = C.Count
```

Finally we describe how to simulate the functionality replacing or updating an item in the array using collection.

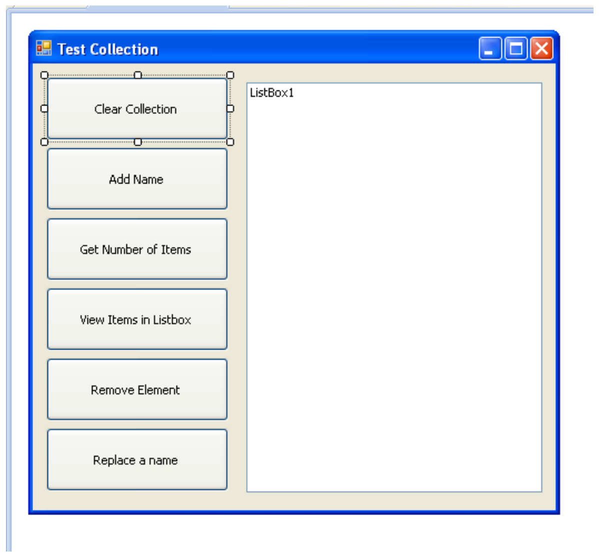
If we write :

```
C.Add(N, , , 7)
C.Remove(7)
```

This will have exactly the same effect as:

```
A(6) = N
```

So this is a basic introduction about collections. Next is a simple application demonstrating the use of collections. Create a windows form, and make it similar to what you see below:



Next write the code below:

```
Public Class Form1

    Dim MyCollection As New Collection

    Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles Button1.Click
        ' this method clears all the elements in the collection
        MyCollection.Clear()
    End Sub

    Private Sub Button2_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles Button2.Click
        ' read a name
        Dim Name As String
        Name = InputBox("enter a name")

        ' add the name into the list
        MyCollection.Add(Name)
    End Sub

    Private Sub Button3_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles Button3.Click
        MsgBox("the number of items is:" & MyCollection.Count)
    End Sub

    Private Sub Button4_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles Button4.Click
        ' clear old content
        ListBox1.Items.Clear()

        ' insert the items into the list box
        Dim I As Integer
        For I = 1 To MyCollection.Count
            ListBox1.Items.Add(MyCollection(I))
        Next

    End Sub

    Private Sub Button5_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles Button5.Click
        ' get element position
        Dim I As Integer
        I = InputBox("enter the element number you want to remove:")
        MyCollection.Remove(I)
    End Sub

    Private Sub Button6_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles Button6.Click
        ' get element position
        Dim I As Integer
        Dim N As String
        I = InputBox("enter the element number:")
        N = InputBox("enter the new name:")
        MyCollection.Add(N, , , I)
        MyCollection.Remove(I)

    End Sub

End Class
```

The code is very simple and at your level you should understand it very easily. If you have any problem with it just send me notes about it. Next tutorial we will start working with functions, and see how it makes coding much easier for us. If you have any notes or questions send them to notes@mka-soft.com.

Thank you.

mkaatr.