Visual Web Development

Terry Marris January 2009

15 Arrays and Array Lists

We look at arrays and array lists.

15.1 Arrays

An array is a set of contiguous storage locations. (Contiguous means touching, continuous, no gaps.)

team

0	1	2	3
tom	dick	harry	ann

Here, the array is named team and it has four cells each containing a value. The cells are numbered in order from zero upwards. Each cell number is known as an index or subscript.

All the elements (or values) in an array must be of the same type. In our example, that type is String. Of course, you can have arrays of any type you like.

The element in location zero is tom. We write

```
team(0) = "tom"
```

The element in location three is ann. We write

```
team(3) = "ann"
```

The indices range from 0 up to 3 inclusive. There are four elements in the array. The last index is one less than the number of elements in the array. Always.

We can add an item to an array.

team

0	1	2	3	4
tom	dick	harry	ann	may

```
team(4) = "may"
```

Traverse through the array element by element.

```
For i = 0 To 4
    name = team(i)
    stringOutput = stringOutput + name
    If i < 4 Then
        stringOutput = stringOutput + ", "
    End If
Next
```

Here, variable i indexes the array.

Search for an element in the array. harry is in the team, but june is not.

```
For i = 0 To 4
    name = team(i)
    if name = "harry" Then
        return "found"
    End If
Next
return "not found"
```

Remove an element from the array.

team

0	1	2	3
tom	harry	ann	may

Sort the contents of an array e.g. into alphabetical order.

team

0	1	2	3
ann	harry	may	tom

The capacity of the array, the number of elements it could contain, is controlled by the programmer. We prefer to use an ArrayList, which is a kind of array, because its size is controlled automatically by Visual Basic.

15.2 Array Lists

An ArrayList is a kind of array. We use an ArrayList to maintain a list of team members. By maintain we mean operations such as add members to the list, remove members from the list and sort members into alphabetical order.

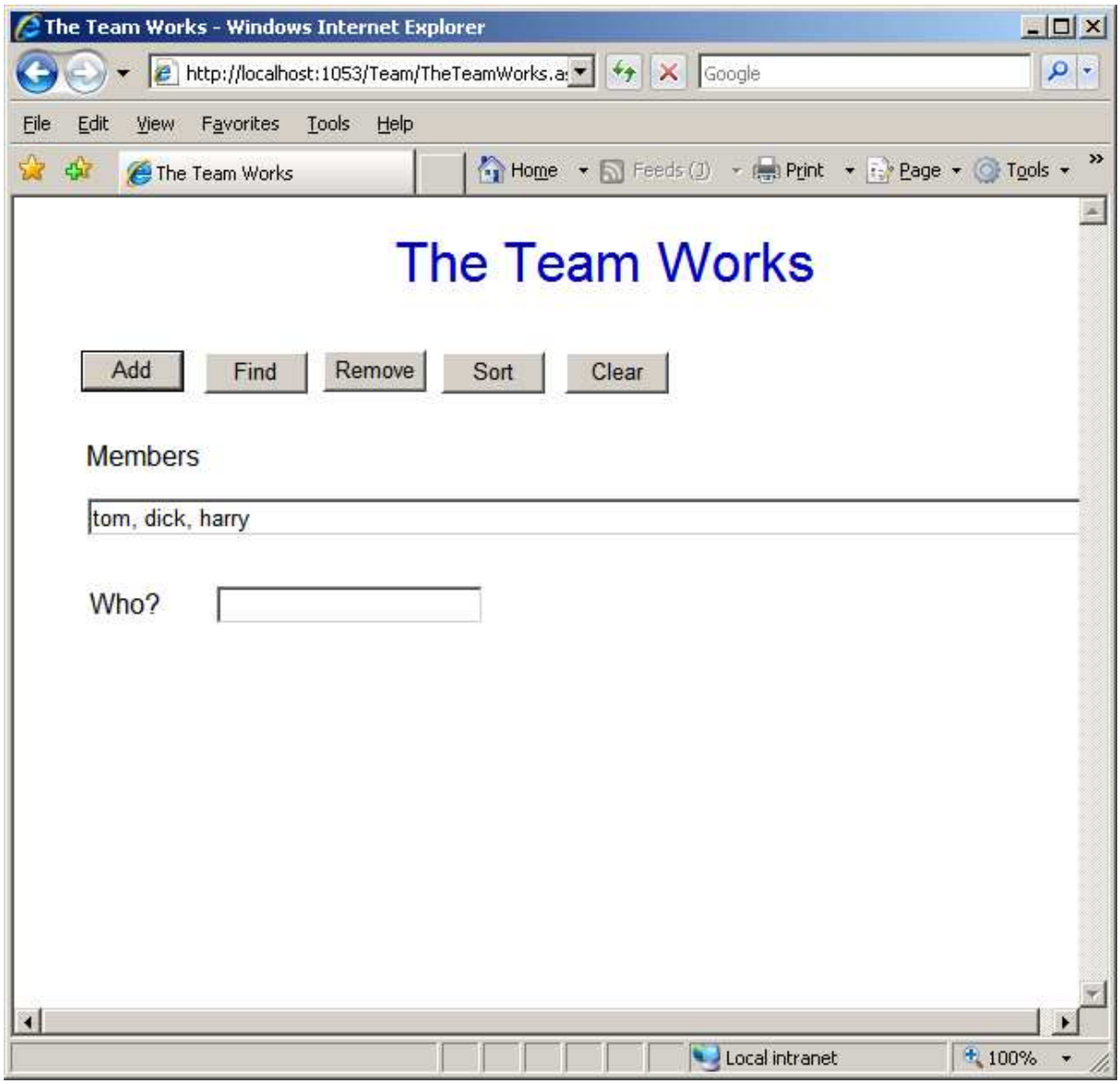
Use Case	TheTeamWorks
Goal	to maintain a list of team members
Pre-condition	
Post-condition	
Initiating Actor	the user

Main Success Scenario

1. user selects from a list of options: add, find, remove or sort team members
2. system reports on the success of the requested operation
3. system displays current team members
4. exit success

Exceptions

15.3 User Interface



The user enters a name in the text box labelled Who? and then clicks the Add button. The name is then added to the Members list. Find reports whether a named member is in the list or not. Remove removes a member from the list - if it is in the list in the first place. Sort sorts the list into ascending alphabetical order. Clear empties the list of members.

15.4 Array List Operations

Global Variables

We declare and initialise an array list.

```
Partial Class _Default
    Inherits System.Web.UI.Page

    ' GLOBAL DECLARATIONS
    Dim alTeam As New ArrayList()
```

We place the declaration near the top of the file because access to the array list, named alTeam, is to be shared among several functions and procedures.

Page Load and Unload Events

We need to preserve the contents of the array list as pages are loaded and unloaded between button clicks.

```
    Protected Sub Page_Unload(ByVal sender As Object, ByVal e As
System.EventArgs) Handles Me.Unload
        Session("TeamMembersList") = alTeam
    End Sub

    Protected Sub Page_Load(ByVal sender As Object, ByVal e As
System.EventArgs) Handles Me.Load
        If IsPostBack = True Then
            alTeam = Session("TeamMembersList")
        Else
            alTeam = New ArrayList()
        End If
        txtWho.Focus()
        btnOK.Visible = False
    End Sub
```

On start up we set the focus on the input txtWho, and hide the OK button.

Display Array List

We need to display the contents of the array list, in a text box, with each member separated from the next by a comma.

```
Function showTeam() As Integer
    txtTeam.Text = ""
    Dim lastIndex As Integer = alTeam.Count() - 1
    For i As Integer = 0 To lastIndex
        Dim strName As String = Convert.ToString(alTeam(i))
        txtTeam.Text = txtTeam.Text + strName
        If i < lastIndex Then
            txtTeam.Text = txtTeam.Text + ", "
        End If
    Next
    Return 0
End Function
```

We work through the team list item by item. We convert each item from the list into a String (because items are automatically stored in lists as Objects). We add each name onto the output text box. We add a comma after each name except for the last name.

We write showTeam() as a function (it could have been written as a procedure) as a matter of personal preference. The value returned, 0 to indicate success, can be ignored at the point of call - see later.

Search

The result of every search is either *success* or *failure*.

We look for a member when given a name. The function findMember() shown below, returns either Found if the given name, strName, is in the list, or Not found if the given name is not in the list.

```
Function findMember(ByVal strName As String) As String
    Dim lastIndex As Integer = alTeam.Count() - 1
    For i As Integer = 0 To lastIndex
        Dim strAName As String = Convert.ToString(alTeam(i))
        If strAName.Equals(strName) Then
            Return "Found"
        End If
    Next
    Return "Not found"
End Function
```

We work through the team list item by item. We convert each item from the list into a String (strAName) and then see whether it is the same as the one we are looking for

(strName). If it is the same we immediately suspend the search with Return "Found". If it is not the same we go on to the next item in the list. If we reach the end of the list without having returned, it can only be because the name we are looking for is not there. Only then do we return "Not found".

A problem with this function is that case is significant; Tom and tom are regarded as two different items. One way to resolve this problem (if it is a problem) is to convert both strName and strAName entirely to lower case before comparing them for equality.

```
strName = LCase(strName)
strAName = LCase(strAName)
If strAName.Equals(strName) Then ...
```

Add

We can either write the code, in detail from scratch, to add a new member or we can use an array list method to do the job for us. Here, we use the array list Add() method. So the code behind the Add button is:

```
Protected Sub btnAdd_Click(ByVal sender As Object, ByVal e As
            System.EventArgs) Handles btnAdd.Click
    Dim strName As String = txtWho.Text.ToString()
    alTeam.Add(strName)
    showTeam()
    txtWho.Text = ""
End Sub
```

We get the new member from txtWho, convert it to a String, and then add it to the array list, alTeam. Although we add a String object to the array list, it is an Object that is stored. The conversion takes place automatically. Then we display the team contents and clear the input text box.

You may remember that showTeam() was written as a function that returned 0 for success? Well, that returned value is ignored here.

Find

The code behind the Find button is just as straightforward.

```
Protected Sub btnFind_Click(ByVal sender As Object, ByVal e As
                  System.EventArgs) Handles btnFind.Click
    Dim strName As String = txtWho.Text.ToString()
    Dim strReport As String = findMember(strName)
    txtWho.Text = txtWho.Text + " " + strReport
    btnOK.Visible = True
End Sub
```

We retrieve the name entered by the user in txtWho. Then we call the findMember() function we described earlier. The value returned by our findMember() function, Found or Not Found, is stored in report and then displayed.

When the report is displayed, the OK button is shown, and then made invisible when it is clicked.

```
Protected Sub btnOK_Click(ByVal sender As Object, ByVal e As
            System.EventArgs) Handles btnOK.Click
    txtWho.Text = ""
    btnOK.Visible = False
End Sub
```

Remove

The code behind the Remove button is shown below.

```
Protected Sub btnRemove_Click(ByVal sender As Object, ByVal e As
                  System.EventArgs) Handles btnRemove.Click
    Dim strName As String = txtWho.Text.ToString()
    alTeam.Remove(strName)
    showTeam()
    txtWho.Text = ""
End Sub
```

We retrieve the name entered by the user. The array list Remove() method then removes the given name from the array list - if it can. Then we display the team contents.

Sort

The code behind the Sort button is beautifully simple. We say: Hey, alTeam, go and sort yourself. Then we display the sorted team.

```
Protected Sub btnSort_Click(ByVal sender As Object, ByVal e As
                    System.EventArgs) Handles btnSort.Click
    alTeam.Sort()
    showTeam()
End Sub
```

Clear

The clear button empties the array list of team members.

```
    Protected Sub btnClear_Click(ByVal sender As Object, ByVal e As
                        System.EventArgs) Handles btnClear.Click
        alTeam.Clear()
        showTeam()
    End Sub
```

Entire VB Code

The entire VB code is shown below.

```
Partial Class _Default
    Inherits System.Web.UI.Page

    ' GLOBAL DECLARATIONS
    Dim alTeam As New ArrayList()

    Protected Sub Page_Unload(ByVal sender As Object, ByVal e As
System.EventArgs) Handles Me.Unload
        Session("TeamMembersList") = alTeam
    End Sub

    Protected Sub Page_Load(ByVal sender As Object, ByVal e As
System.EventArgs) Handles Me.Load
        If IsPostBack = True Then
            alTeam = Session("TeamMembersList")
        Else
            alTeam = New ArrayList()
        End If
        txtWho.Focus()
        btnOK.Visible = False
    End Sub

    Function showTeam() As Integer
        txtTeam.Text = ""
        Dim lastIndex As Integer = alTeam.Count() - 1
        For i As Integer = 0 To lastIndex
            Dim strName As String = Convert.ToString(alTeam(i))
            txtTeam.Text = txtTeam.Text + strName
            If i < lastIndex Then
                txtTeam.Text = txtTeam.Text + ", "
            End If
        Next
        Return 0
    End Function
```

```
    Function findMember(ByVal strName As String) As String
        Dim lastIndex As Integer = alTeam.Count() - 1
        For i As Integer = 0 To lastIndex
            Dim strAName As String = Convert.ToString(alTeam(i))
            If strAName.Equals(strName) Then
                Return "Found"
            End If
        Next
        Return "Not found"
    End Function

    Protected Sub btnAdd_Click(ByVal sender As Object, ByVal e As
System.EventArgs) Handles btnAdd.Click
        Dim strName As String = txtWho.Text.ToString()
        If strName.Length() > 0 Then
            alTeam.Add(strName)
        End If
        showTeam()
        txtWho.Text = ""
    End Sub

    Protected Sub btnFind_Click(ByVal sender As Object, ByVal e As
System.EventArgs) Handles btnFind.Click
        Dim strName As String = txtWho.Text.ToString()
        Dim strReport As String = findMember(strName)
        txtWho.Text = txtWho.Text + " " + strReport
        btnOK.Visible = True
    End Sub

    Protected Sub btnOK_Click(ByVal sender As Object, ByVal e As
System.EventArgs) Handles btnOK.Click
        txtWho.Text = ""
        btnOK.Visible = False
    End Sub

    Protected Sub btnSort_Click(ByVal sender As Object, ByVal e As
System.EventArgs) Handles btnSort.Click
        alTeam.Sort()
        showTeam()
    End Sub

    Protected Sub btnRemove_Click(ByVal sender As Object, ByVal e As
System.EventArgs) Handles btnRemove.Click
        Dim strName As String = txtWho.Text.ToString()
        alTeam.Remove(strName)
        showTeam()
        txtWho.Text = ""
    End Sub

    Protected Sub btnClear_Click(ByVal sender As Object, ByVal e As
System.EventArgs) Handles btnClear.Click
        alTeam.Clear()
        showTeam()
    End Sub
End Class
```

15.5 Exercises

1. Try out the program that maintains a list of team members as shown above. Resolve the case issue.

2. A problem with the find method is that it concludes on finding the first match. What if there were two entries in the list with the same name? Resolve the problem.

15.6 Conclusion

We have seen that an array list can be viewed as a regular array or as an object with its own useful methods. Next we look at Strings.