# Visual Web Development

Terry Marris  October 2007

## *16  Strings*

We look at some methods for processing strings.

### 16.1  Strings

A string is a sequence of characters.  Characters include upper and lower case letters, digits, space, tab and punctuation symbols.

A string literal (also known as a string constant) is enclosed within quotation marks.

```
Dim strLiteral As String =   "Error message:"
```

To include quotation marks within the string itself we use two quotation marks side-by-side.

```
Dim strQuote As String =
                "She said: ""This is a fine mess you've got me into!""  to her friend"
```

## 16.2 String Operations

We describe a few operations that can be performed on strings. Strings are treated as arrays of Char indexed from zero upwards.

Working with Dim str As String = "Catastrophe":

| **Method** | **Example** | **Returns** |
|---|---|---|
| Chars(index) | str.Chars(0) | C |
| Concat(string) | str.Concat( " rules OK") | Catastrophe rules OK |
| EndsWith(string) | str.EndsWith("he") | True |
| FirstIndexOf(string) | str.FirstIndexOf("t") | 2 |
| IndexOf(string) | str.IndexOf("t") | 2 |
| | strIndexOf("x") | -1 (x is not in Catastrophe) |
| Insert(index, count) | str.Insert(9, "!") | Catastrophe! |
| LastIndexOf(string) | str.LastIndexOf("t") | 5 |
| Length() | str.Length() | 11 |
| Remove(index, count) | str.Remove(3, 7) | Cat |
| Replace(old, new) | str.Replace("e", "ic") | Catastrophic |
| StartsWith(string) | str.StartsWith("C") | True |
| Substring(from, length) | str.Substring(4, 5) | strop |
| ToLower() | str.ToLower() | catastrophe |
| ToUpper() | str.ToUpper() | CATASTROPHE |
| Trim() | str.Trim() | str with spaces removed from beginning and end |

Split() splits a string into an array of strings.

```
Dim strArray() As String = Split("someone@example.com", "@") returns
strArray(0) = "someone"
strArray(1) = "example.com"
```

Join() joins an array of strings into one string.

```
Dim strArray() As String
strArray(0) = "someone"
strArray(1) = "example.com"
Dim str As String = Join(strArray, "@")
```

leaves str = "someone@example.com"

## 16.3 Exercise

1. Design, write and test a program that will demonstrate any five operations on strings.

## 16.4 Validation

Validation involves checking whether data complies with a set of rules. For example, a rule might be: you must be at least 17 to qualify for a car driving licence.

```
If age < 17 Then
    report = "Too young"
Else
    report = "Age ok"
End If
```

The rules for a valid e-mail address are slightly more complicated (see http://tools.ietf.org/html/rfc2822 for example). We shall arbitrarily declare an e-mail address to be valid if it conforms to the following rules:

| | | |
|---|---|---|
| **1** | There must be a user name and a domain name. | pgreen@xmail.com<br>user name: pgreen<br>domain name: xmail.com |
| **2** | The allowable characters are abcdefghijklmnopqrstuvwxyz.-_@. | pgreen@xmail.com |
| **3** | The user name and the domain name must be separated by an @. | pgreen@xmail.com |
| **4** | The user name cannot end with a full stop and the domain name cannot begin with a full stop. | pgreen•@•xmail.com |
| **5** | The domain name must contain at least one full stop. | pgreen@xmail•com |
| **6** | There must be at least two characters after the last full stop in the domain name. | pgreen@xmail.**com** |

First, we remove any spaces from the beginning and end of the e-mail address, and convert it all to lower case.

```
Function tidy(ByVal emailAddress As String) As String
    Dim str As String = emailAddress.ToLower()
    str = str.Trim()
    Return str
End Function
```

Then we check that only valid characters are used. We go through the e-mail address character by character. For each character we look for its index in okChars. If we cannot find it then we know we have an invalid character.

```
Function validChars(ByVal emailAddress As String) As String
    Dim okChars As String = "abcdefghijklmnopqrstuvwxyz_-@."
    For i As Integer = 0 To emailAddress.Length() - 1
        If okChars.IndexOf(emailAddress(i)) < 0 Then
            Return "validChars: invalid character in e-mail address"
        End If
    Next
    Return "OK"
End Function
```

We notice that an underscore is included as a valid character, but underscores are not used in domain names (but they can appear in user names). This is a probelm that needs fixing.

We check the number of @ symbols. There should be just one. We go through the e-mail address character-by-character adding 1 to a counter for each @ symbol we find.

```
Function atSymbolCount(ByVal emailAddress As String) As String
    Dim atCount As Integer = 0
    For i As Integer = 0 To emailAddress.Length() - 1
        If emailAddress(i) = "@" Then
            atCount = atCount + 1
        End If
    Next
    If atCount = 1 Then
        Return "OK"
    End If
    Return "atSymbolCount: error with @"
End Function
```

We split the e-mail address into two parts: the user name and domain with suffix. The user name cannot end with a full stop and the domain-with-suffix cannot start with a full stop. If the Split() method cannot find the two parts separated by an @ we have an error. If the length of either part is zero we also have an error.

```
Function dotLocation(ByVal emailAddress As String) As String
    Const fullStop As String = "."
    Dim array() As String = Split(emailAddress, "@")
    If array.Length() <> 2 Then
        Return "dotLocation: error with email address format"
    End If
    Dim userName As String = array(0)
    Dim domainName As String = array(1)
    If userName.Length() = 0 Or domainName.Length() = 0 Then
        Return "dotLocation: error with email address format"
    End If
    If userName(userName.Length() - 1) <> fullStop And
         domainName(0) <> fullStop Then
        Return "OK"
    End If
    Return "dotLocation: error with full stops"
End Function
```

We split the email address into its user name and domain name parts. The domain name must have at least one full stop. We go through the domain name character-by-character, counting the full stops we come across.

```
Function dotCount(ByVal emailAddress As String) As String
    Const fullStop As String = "."
    Dim array() As String = Split(emailAddress, "@")
    If array.Length() <> 2 Then
        Return "dotCount: error with email address format"
    End If
    Dim userName As String = array(0)
    Dim domainName As String = array(1)
    Dim dots As Integer = 0
    For i As Integer = 0 To domainName.Length() - 1
        If domainName(i) = fullStop Then
            dots = dots + 1
        End If
    Next
    If dots <> 0 Then
        Return "OK"
    End If
    Return "dotCount: error with domain name"
End Function
```

Again we split the email address into its user name and domain name parts. We find the location of the last full stop and then count the number of characters that follow; there must be at least two.

```
Function suffixLength(ByVal emailAddress As String) As String
    Const fullStop As String = "."
    Dim array() As String = Split(emailAddress, "@")
    If array.Length() <> 2 Then
        Return "suffixLength: error with email address format"
    End If
    Dim userName As String = array(0)
    Dim domainName As String = array(1)
    Dim charCount As Integer = 0
    Dim indexOfLastDot As Integer = domainName.LastIndexOf(fullStop)
    For i As Integer = indexOfLastDot + 1 To domainName.Length() - 1
        charCount = charCount + 1
    Next
    If charCount >= 2 Then
        Return "OK"
    End If
    Return "suffixLength: error with suffix"
End Function
```

The entire email address checking function can now be assembled. We remember that the dimension of an array is given by its last index value. We go through the report array item by item. If we find an item that is not OK we immediately return with a diagnostic.

```
Function emailAddressCheck(ByVal emailAddress As String) As String
    emailAddress = tidy(emailAddress)
    Dim report(4) As String
    report(0) = atSymbolCount(emailAddress)
    report(1) = validChars(emailAddress)
    report(2) = dotLocation(emailAddress)
    report(3) = dotCount(emailAddress)
    report(4) = suffixLength(emailAddress)
    For i As Integer = 0 To report.Length() - 1
        If report(i) <> "OK" Then
            Return report(i)
        End If
    Next
    Return "OK"
End Function
```

The whole process is triggered by the user entering an e-mail address in a text box, txtMail, and the diagnostic messages are shown in txtReport.

```
Protected Sub btnOK_Click(ByVal sender As Object, ByVal e As
System.EventArgs) Handles btnOK.Click
    txtReport.Text = emailAddressCheck(txtEmail.Text)
End Sub
```

## 16.5 Exercises

1. Complete the program, shown above, that validates an e-mail address. Find and document the errors it contains.

2. The e-mail validation code provided above is not complete. Devise additional rules and then write and check code that implements these rules. For example: you cannot have two full stops side by side, the user name cannot be empty, ...

## 16.6 Conclusion

We have looked at some string processing functions and examined a non-trivial validation problem. Next, we look at classes.