ListBox

The purpose of the ListBox form is to show you some basic uses of list boxes. The ListBox form allows you to enter the names of some of your friends, and organize them in a list. Two textboxes are used to enter your friends first and last names. A large listbox is provided to organize a list of names. This list can be automatically sorted with a checkbox option. As friends are entered in the list, the total number of friends are automatically counted. There are other functions used to remove or change existing names in the list.

The Names ListBox is used to present the names of your friends. The list is organized to display first name followed by last name. The **Sorted** property on the ListBox is set by the status of the Sorted checkbox at the bottom of the form.

The "First Name" textbox is used to enter the first name of one of your friends. If you click the TAB key after keying in the first name, the insertion point will jump to the last name box automatically.

The "Last Name" textbox is used to enter the last name of one of your friends. If you click the TAB key after keying in the last name, the "focus" will jump to the Append button. Clicking the Enter key will automatically enter the name into the ListBox, clear the two textboxes, then move the insertion point into the first name box.

The Sorted checkbox is used to toggle the **Sorted** property of the lstNames ListBox. An "X" displayed in the checkbox will automatically sort the first names in the ListBox in alphabetical order.

The Totals label is used to display the total number of friends currently contained in the ListBox. The total count is calculated by the **UpdateTotals** method. When the count reaches a point where there are more names in the list than can be displayed, a scrollbar on the ListBox is automatically added and a hidden message is automatically displayed at the bottom of the form.

A hidden label at the bottom of the form is used to display a message about the number of friend you have. It is displayed only when the number of friends in the list exceed the number of lines displayed in the list.

The label is used to provide a brief instruction on how to run the application. Other labels are used on this form to describe various controls, such as First Name, Last Name and List of Friends. Controls on a form that are not called from program code, normally do not have a *function oriented* name. In this case, the default name Label1 is never changed through program code. The label name **lblTotal** is changed through program code by having its **Caption** set through the **UpdateTotals** method. Therefore the label's name property is set to a function oriented name.

Adding Names to the List

The Append button is used to take a first and last name entry, combine them, and add the complete name to the ListBox. A check is performed to make sure valid name entries exist. After the name is added to the list, the Total is updated. After the name is inserted, the first and last name textboxes are automatically cleared. The following code shows how this is done for the Append button.

```
Sub btnAppend_Click()
    If txtFirstName.Text == "" Or txtLastname.Test == "" Then
      InfoBox.Msg("Name information is incomplete.")
    Else
      lstNames.AdItem(txtFirstName.Text & " " & txtLastName.text)
      txtFirstName.Text = ""
      txtLastName.Text = ""
      txtFirstName.SetFocus
      UpdateTotals
    End If
End Sub
```

Removing Names from the List

The **Delete** button executes a method that is programmed to remove the currently selected name entry from the ListBox. If no entry is currently selected, no action will occur The following Click method can be found in the Method Editor's method ComboBox.

```
Sub btnDelete_Click()
    lstNames.RemoveItem(lstNames.ListIndex)
    UpdateTotals
End Sub
```

This instruction uses two ListBox standard methods named RemoveItem and ListIndex. The output of ListIndex is to return a line number that corresponds with the currently selected list entry. For example, if the first line in the ListBox were selected, the output of **ListIndex** would be 0 for line 1, 1 for line 2, etc. The method **RemoveItem** takes the selected line number and removes it from the list. Once this is done, the **UpdateTotals** sub procedure is executed to update the number of list items.

Clearing the List

The Reset button is used to clear the contents of the ListBox as well as the two individual textboxes. The Total label is also cleared. The Reset button executes the following method:

```
Sub btnReset_Click()
    lstNames.Clear
    txtFirstName.Text = ""
    txtLastName.Text = ""
    UpdateTotals
End Sub
```

The ListBox **Clear** method is executed first. This is a standard method available on ListBoxes and has the function of clearing all entries from the list. You can see the list of standard methods on any control simply by selecting it as the current control in the Method Editor. Below is a picture of the **lstNames** control displaying a list of standard methods.

Changing Existing Names

Changing an entry in a ListBox involves combining some existing methods. The following click method is defined for the Change button.

```
Sub btnChange_Click()
    Dim first_name As String
    Dim last_name As String
    Dim list_entry As String

     list_entry = lstNames.ItemString(lstNames.ListIndex)

     ' parse the list entry to get the first and last names
     ' format for GetToken <string> <token> <separator>
     first_name = GetToken(list_entry, 1, " ")
     last_name = GetToken(list_entry, 2, " ")

 ' copy the first and last names into the edit boxes
 txtFirstName.Text = first_name
     txtLastName.Text = last_name

     ' Remove the current list entry
    btnDelete_Click

     ' Select the first name
     txtFirstName.SetFocus
     txtFirstName.SelStart = 0
     txtFirstName.SelLength = Len(txtFirstName.Text)
End Sub
```

The Change function involves taking a currently selected line from the ListBox and placing the first and last names in their corresponding TextBox for editing. In order to separate the first and last names from the selected list entry, a function named **GetToken** was written. This function takes the list item entry, a token count value, and a token separator. In this case, the separator between first and last names is a blank space " ".

After this is done, the line item must be removed from the list. This is done by calling btnDelete_Click. This method is already defined to work with the **Delete** button. It can be called directly from our Change method. Also note that the Delete method will already call the **UpdateTotals** sub procedure, therefore this call does not have to be added to the Change method.

After the line has been removed from the list and the totals updated, the insertion point is placed at the beginning of the first name textbox and the first name is selected. This is done by the **SelStart** and **SelLength** methods of the TextBox control.

Inserting Names into the List

Inserting an entry into a ListBox is very similar to adding an entry. The following click method is defined for the **Insert** button.

```
Sub btnInsert_Click()
Dim name_entry As String

If txtFirstName.Text == "" Or txtLastName.Text == "" Then
   InfoBox.Msg("Name information is incomplete.")
ElseIf lstNames.ListIndex == -1 Then
      InfoBox.Msg("No name entry selected in list.")
   Else
      name_entry = txtFirstName.Text & " " & txtLastName.Text
      lstNames.InsertItem(name_entry, lstNames.ListIndex)
      txtFirstName.Text = ""
      txtLastName.Text = ""
      UpdateTotals
   End If
End Sub
```

The Insert button is used to take a first and last name entry, combine them, and insert the complete name ino the ListBox. The name is inserted on the line above the currently selected name in the list. After the name is inserted, the first and last name textboxes are automatically cleared.

The key difference between the Append function and the Insert function is the difference in functionality between the **AppendItem** method and **InsertItem** method. The AppendItem method automatically adds an item to the end of a list, while the InsertItem method takes an entry and an index number (line number) and inserts the item just before that line.

Sorting Names in the List

If you click the checkbox to True, an "X" appears in the box and the first names in the ListBox are automatically sorted in alphabetical order. When additional names are added, they will be added to the list in alphabetically. If you then click the checkbox to False, the checkbox will appear blank, but the names will not go back to their original order. When additional names are added to the list, they will either be appended or inserted into the list, depending on which button was clicked.

The program code associated with the chkSorted click method is shown below.

```
Sub chkSorted_Click()
   If chkSorted.Value == 0 Then
                lstNames.Sorted = "False"
   Else
                lstNames.Sorted = "True"
   End If
      lstNames.Refresh
End Sub
```

The functionality is pretty straight forward. The Value property of the chkSorted checkbox can be set to 0 or 1. If set to 0, the value is False. If set to 1, the value is true. The Value property of the ckeckbox is controlled by clicking on the checkbox control. A value of True will display an "X" in the checkbox while a value of False will leave the checkbox blank.

The method above shows that if the chkSorted checkbox value is 0, the Sorted property of the ListBox lstNames is set to "False." If the chkSorted checkbox value is 1, the Sorted property of the ListBox lstNames is set to "True." Names in the ListBox are automatically sorted in alphabetical order when the Sorted property is set to True. The last line of the program code is used to "Refresh" the list and update its order.

ListBox Sample Help

Sample Description: **ListBox**

Points of Interest Application

- Adding Names to the List
- Removing Names from the List
- Clearing Names from the List
- Changing Existing Names
- Inserting Names into the List
- Sorting Names in the List

Controls

- ListBox
- TextBox
- Button

For Help on Help, Press F1