

Элементы выбора

VB.NET включает несколько стандартных элементов управления, обеспечивающих возможность выбора:

CheckBox (чек-бокс)

CheckBox показывает, включено или нет определенное условие. В приложении флажки используются, чтобы дать пользователю возможность выбора типа «правда/ложь» или «да/нет». Так как флажки работают независимо друг от друга, пользователь может установить любое их число одновременно.

Свойство **Checked** показывает, находится ли чек-бокс в отмеченном состоянии, или нет. Если в отмеченном, то значение этого свойства True

RadioButten (радиокнопка)

Это набор элементов для выбора одного из многих вариантов. В отличие от чек-боксов, радиокнопки обязательно объединяются в группы (не менее двух в группы). Объединяются они с помощью элементов Рамка группы (GroupBox) или панели (Panel).

Радиокнопки могут быть помещены на форму без рамки или панели. Но они все равно образуют группу. В этом случае сама форма является объединяющим объектом.

Объединение радиокнопок в группу означает, что он можно выбрать одну и только одну возможность из заданного набора.

Свойство Checked у нажатой кнопки True, у остальных – False.

Панели списков ListBox и ComboBox.

Элементы выбора панель списков ListBox и ComboBox представляют собой элементы, позволяющие пользователю делать выбор из гораздо большего количества вариантов, чем флажки и переключатели.

Элемент управления Панель списка ListBox

Свойство Items осуществляет доступ к коллекции элементов списка. В режиме разработки в окне Properties можно занести коллекцию элементов списка. Это окно открывается при нажатии на многоточие в свойстве Items.

Метод **Add** добавляет элемент в коллекцию Items в конец списка. Нумерация начинается с 0.

```
ListBox1.Items.Add("строка")
```

```
listBox1.Items.Add(textBox1.Text);
```

```
// Копирует все элементы из listBox1 в listBox2 (добавляя их в
конец списка listBox2)
listBox2.Items.AddRange(listBox1.Items);
```

Метод **Insert** добавляет элемент на конкретное место. Аргументами метода Insert являются текст и номер элемента списка, Например:

добавляется элемент в начало списка

```
ListBox1.Items.Insert (0, "Иванов")
```

Добавляет текст из текстового поля в начало списка (индекс 0)

```
listBox1.Items.Insert(0, textBox1.Text);
```

Метод **RemoveAt** удаляет элемент с определенным номером. Например:

удалить первый элемент списка

```
ListBox1.Items.RemoveAt(0)
```

'удалить из списка выделенный элемент

```
listBox1.Items.RemoveAt(listBox1.SelectedIndex)
```

Метод **Item** предоставляет значение элемента коллекции с конкретным номером

```
Label1.Text = ListBox1.Items.Item(2)
```

Метод **Clear** удаляет все элементы из коллекции

'ОЧИСТИТЬ СПИСОК

```
ListBox1.Items.Clear()
```

Метод **Count** возвращает число элементов в коллекции.

```
listBox1.Items.Count.ToString()
```

Состояние кнопки будет обновляться при изменении содержимого поля

// Кнопка "Добавить" активна, если в поле есть хоть 1 символ для textBox1

```
button1.Enabled = (textBox1.Text != String.Empty);
```

Основные свойства, методы и события элемента **ListBox**

Свойства	Описание
Items	Осуществляет доступ к коллекции элементов списка в ListBox.
SelectedIndex	Индекс выделенного элемента списка..Если в списке не выбран ни один элемент, возвращается - 1
SelectedItem	Предоставляет текст пункта, который выделен в текущий момент
Text	Предоставляет текст пункта, который выделен в данный момент
Sorted	Позволяет отсортировать список. Если True, то отсортированный список
Методы	
ClearSelected	Отменяет все выделения в списке

SetSelected	Используется для выделения или снятия выделения заданного пункта списка
-------------	---

Пример

```
string a;
int n;
n = ListBox1.Items.Count
ListBox1.SetSelected(n - 1, True)
a = ListBox1.Text
MessageBox.Show("количество элементов = " & n & vbCrLf & "
добавили " & a)
```

Элемент управления ComboBox.

Эти разновидности списков работают одинаково; отличие состоит только в том, что комбинированное окно содержит дополнительное текстовое окно, в котором отображается выбранный элемент.

Поддерживает все свойства, методы и события элемента управления ListBox. Но имеет свои, дополнительные.

Свойства	Описание
DropDownStyle	Это свойство предоставляет стиль ComboBox: Dimple (простой); DropDown – можно ввести собственное значение (принят по умолчанию). DropDownStyle – нельзя ввести собственное значение Simple – имеет вид ListBox, но с присоединенным текстовым полем. Можно ввести значение в текстовое поле, либо выбрать требуемое значение из списка, которое будет перенесено в текстовое поле
MaxDropDownItems	Предоставляет максимальное количество пунктов, которые будут выведены в спускающейся части списка

Извлечь выделенный элемент

```
Label1.Text = ComboBox1.SelectedItem
```

Извлечь третий элемент

```
Label1.Text = ComboBox1.Items.Item(2)
```

Извлечь случайный элемент из списка

```
Label1.Text = ComboBox1.Items.Item(rn.Next(ComboBox1.Items.Count))
```

Окно вывода сообщений MessageBox

Окно сообщений в .NET поддерживается классом `MessageBox`. Для отображения окна сообщений достаточно вызвать определенный в этом классе статический метод `Show()`.

Используя этот метод, можно создать 18 разных форм. Они будут отличаться друг от друга набором параметров, который входит в каждую форму

Примеры форм:

- `MessageBox.Show(Text [, Caption [, Buttons [, Icon]]]);`
- `Text` — текст сообщения;
- `Caption` — текст в заголовке окна;
- `Buttons` — указывает доступные кнопки. Комбинации кнопок задаются переменными класса `MessageBoxButtons` и могут иметь следующие значения:

`MessageBoxButtons.OK`,
`MessageBoxButtons.OKCancel`,
`MessageBoxButtons.YesNo`,
`MessageBoxButtons.YesNoCancel` — да, нет, отмена,
`MessageBoxButtons.RetryCancel`,
`MessageBoxButtons.AbortRetryIgnore` — прервать, повтор, пропустить.

`Icon` — указывает вид пиктограммы, которая будет отображаться в диалоговом окне.

Вид пиктограммы определяется из набора констант `MessageBoxIcon`;

Константы из набора `MessageBoxIcon`:

`MessageBoxIcon.Error` — пиктограмма "Ошибка" — белая буква X на фоне красного кружочка

`MessageBoxIcon.Question` — "Вопросительный знак"

`MessageBoxIcon.Exclamation` — восклицательный знак на фоне желтого треугольника

`MessageBoxIcon.Information` — пиктограмма "Информация" — буква "i" в кружочке

```
MessageBox.Show("Hello World", "Sample", _
MessageBoxButtons.OKCancel, MessageBoxIcon.Exclamation)
```

```
// Перенаправление выбранного элемента из listBox1 в listBox2
```

```
if (listBox2.SelectedItem == null)
{
    MessageBox.Show("Необходимо выбрать элемент из
списка!", "Sample", MessageBoxButtons.OK,
MessageBoxIcon.Information);
}
else
{
    listBox2.Items.Add(listBox1.Text);
    listBox1.Items.RemoveAt(listBox1.SelectedIndex);
}
```

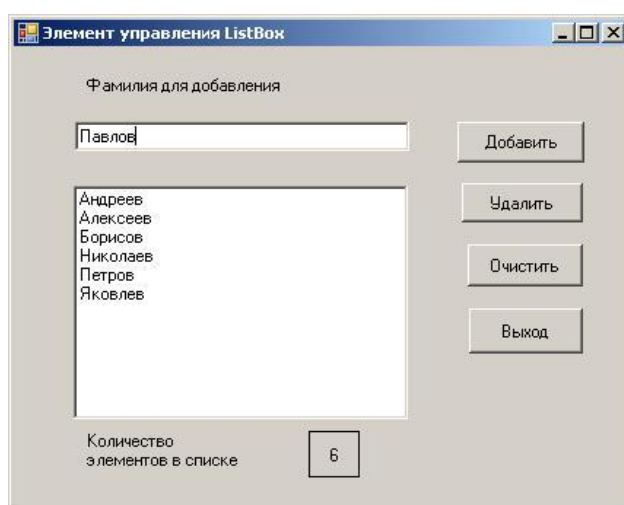
```
// Перенаправление всех элементов из listBox1 в listBox2
```

```
listBox1.Items.Add(listBox2.Text);  
listBox2.Items.RemoveAt(listBox2.SelectedIndex);
```

Задание 2. Элементы выбора

Задача 1.

Необходимо разработать приложение, в котором пользователь может ввести в текстовое поле (textBox) имя клиента, которое будет добавлено в список listBox, если нажать кнопку **Добавить**. Пользователь может удалить из списка элемент, выбрав его и нажав кнопку **Удалить**. Кнопкой **Очистить** можно удалить все элементы из списка. В метке отображается число клиентов в списке.



Сформируйте список при разработке формы в окне свойств (свойство Items элемента ListBox).

Задача 2. Добавить в задачу 1 следующие функции. Сделать кнопку **Удалить** недоступной, если список пуст, или элемент не выбран. Сделать доступной кнопку **Добавить**, если в текстовое поле введен хотя бы 1 символ.

Сформируйте список при загрузке формы в процедуре Load (свойство Items элемента ListBox).

Подсказка:

Чтобы сделать кнопки неактивными используйте `enable=false`;

Задача 3. Разработать форму с двумя списками. С помощью кнопок необходимо перемещать один выбранный элемент из списка в список или все элементы одновременно. В левый список можно добавить элементы через текстовое окно.

Выбор из списка

Петров
Андреев
Почкин
Белкин
Егоров

>

<

>>

<<

Добавить в список

Выход