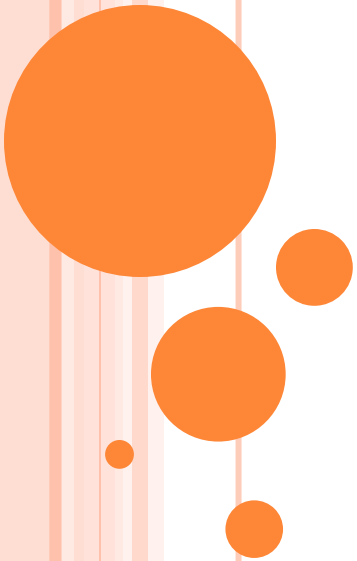


# Списки, файлы, строки

Лекция 5

# Элемент управленияListBox

**Свойства ListBox**

- Items -список элементов списка
- SelectiomMode- способ выбора элементов списка:
  - None-ни одного
  - One - один
  - multiSimple – множественный выбор
  - Multiextended – множественный выбор мышью
- multiColumn можно ли выводить текст в несколько колонок
- Sorted – есть ли сортировка
- SeiectedItem и SelectedIndex- свойства, возвращают выбранный элемент и его индекс.

# Свойства

- SelectedIndices – возвращает коллекцию индексов всех выделенных элементов;
- SelectedItems- возвращает коллекцию всех выделенных позиций;
- Items- все позиции списка;
- Text- текст выбранной позиции списка;

# Методы ListBox

- ADD- заполняет элементами список
- SetSelected –устанавливает или снимает выбор с элемента, с  заданным значением индекса
- Insert (index, item) –вставка элемента в указанное место.
- Remove (Value)- удаление элемента из списка.
- RemoveAt(index) – удаление элемента с указанным индексом.
- Clear () – метод, очистка всего списка.
- Findstring – поиск первого вхождения элемента, подобного указанному, в списке:
  - Findstring (String) – поиск с начала списка;
  - Findstring (S, Index) – поиск с указанной позиции

# События ListBox

- SeiectedIndexChanged, SeiectedValueChanged- - генерируется при смене выбранного элемента списка, поэтому его обработка позволяет отслеживать выбор нового элемента.

# Пример: выбор элемента списка и его отображение в метке

Private Sub ListBox1_SelectedlndexChanged()

- Label1.Text = "Элемент: " +ListBox1.Selectedltem
- Label2.Text = "Индекс: " + CStr(ListBox1.Selectedlndex)
- End Sub

# Примеры

- **Добавление элемента** в список, новый элемент вводится в текстовом поле

```
If (TextBoxl.Text,Trim() = String.Empty) Then
Return ListBox1.Items.Add(TextBoxl.Text)
```

- **Удаление выделенного элемента из списка**

```
If ListBox1.Selectedltems.Count < > 0 Then Return
ListBox1.Items.Remove (ListBox1. Selectedltem)
```

- Вставка элемента в список:

```
If (TextBoxl.Text.Trim() = String.Empty) Then Return
    If ListBoxl.Selectedltems.Count - 0 And _
ListBoxl.Items.Count > 0 Then Return
ListBoxl.Items.Insert(ListBoxl.Selectedlndex + 1,
_TextBoxl.Text)
```

# Поле со списком ComboBox

- DropDownStyle – Свойство, задает стиль отображения списка, его значения:
  - DropDown Разрешено редактирование содержимого поля ввода.
  - DropDownList Пользователь не может редактировать содержимое поля ввода.
  - Simple Разрешено редактирование содержимого поля ввода. Список всегда отображается

# Свойства

- SelectedItem- возвращает выбранный элемент из списка,
- SelectedIndex возвращает индекс выбранного элемента
- DroppedDown - устанавливает отображение списка в раскрытом состоянии.
- DropDownwidth - устанавливает ширину раскрывающего списка. Значение его не может быть меньше ширины самого элемента управления.

# Методы

- FindString и FindStringExact – поиск элемента в списке
- BeginUpdate и EndUpdate обеспечивают гладкость процесса заполнения.

# Пример: добавление в список не повторяющихся данных:

- Private Sub ComboBoxl_KeyDown(ByVal sender As Object, ByVal e As System.Windows.Forms. KeyEventArgs) Handles ComboBoxl.KeyDown
-
- If e.KeyCode = Keys.Enter Then
- If Not ComboBoxl.Items.Contains(ComboBoxl.Text) Then
- ComboBoxl.Items.Add(ComboBoxl.Text)
- MessageBox.Show("Новый элемент добавлен в список")
- ComboBoxl.DroppedDown = True
- Else
- MessageBox,Show("Такой элемент уже существует")
- End If
- End If
- End Sub

# Функции выбора

- **Switch**(Var1,rez1,Var2,rez2…)-вычисляет значения Var и возвращает тот rez для которого Var=True
- Choose(Index, Rez1,Rez2, Rez3…)- в вычисляет значение переменной Index и возвращает RezN с вычисленным номером: если Index=2, то вернет Rez2

# Файлы

# Управляющие элементы

- **OpenFileDialog, *SaveFile*Dialog-** открывает диалоговое окно и предоставляет путь к файлу;
- ***Свойства:***
  - InitialDirectory имя папки, которая открывается при первом использовании окна;
  - Title - заголовок окна;
  - Filter – установка фильтра;
  - Filterlndex – № фильтра по умолчанию (если несколько фильтров).
- ***Методы:*** ShowDialog, FileName

# Открытие файла для чтения

FileOpen(номер_файла, путь, режим)

- **номер_файла** - это число от 1 до 255.
- **путь** - путь, по которому можно найти файл.
- **режим** - это ключевое слово, указывающее на то, как файл будет использоваться.
  - OpenMode.Input – чтение из файла и
  - OpenMode.Output- запись в файл)

# Функции для открытия файла

- LineInput – читает строку из текстового файла;
- EOF- проверяет на достижение конца файла;
- FileClose – закрывает файл.

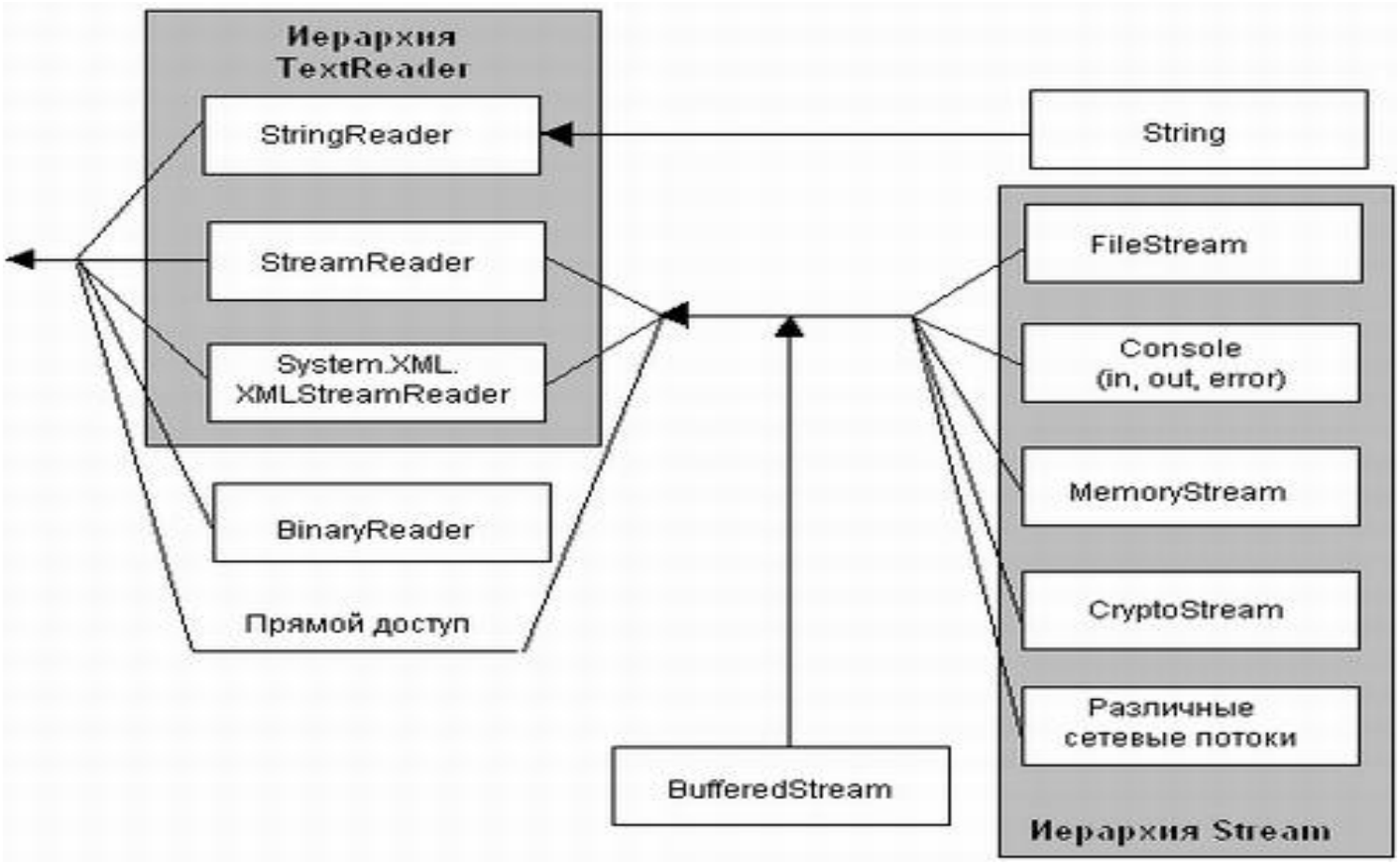
# Пример

```
Dim ff, tt as string
OpenFileDialog1.Title = "Select a File"
OpenFileDialog1.Filter = "файлы (*.TXT) | *.TXT«
OpenFileDialog1.ShowDialog()
        If OpenFileDialog1.FileName <> "" Then
           Try
              FileOpen(1, OpenFileDialog1.FileName, OpenMode.Input)
              Do Until EOF(1)
         ff = LineInput(1)
                  tt = tt & ff & vbCrLf
              Loop
              tst.Text = OpenFileDialog1.FileName
              tst.Text+ = tt
          Catch
              MsgBox("Ошибка открытия файла.")
           Finally
              FileClose(1)    'закрываем файл
           End Try
        End If
```

# Открытие файла для записи

- SaveFileDialog1.Filter = “(*.txt)|*.txt”
- SaveFileDialog1.ShowDialog()
- If SaveFileDialog1.FileName <> "" Then FileOpen(1, SaveFileDialog1.FileName,_ OpenMode.Output)
- PrintLine(1, tst.Text)
- FileClose(1)
- End If

# Ввод-вывод в .NET



# Класс StreamReader

Большинство классов, представляющих устройства ввода-вывода, являются производными от класса System. I0. Stream. Этот класс интерпретирует устройство как поток байтов (доступный для чтения или записи) и позволяет выполнять следующие операции:

- чтение одного или нескольких байтов данных;
- запись одного или нескольких байтов данных;
- асинхронное чтение или запись (с дополнительной возможностью оповещения о завершении операции);
- физическая запись данных из промежуточного буфера на устройство;
- переход к заданной позиции в потоке данных;
- закрытие потока (устройства) после завершения всех операций.

# Методы класса TextReader

- Close- закрывает поток и освобождает системные ресурсы
- Peek- возвращает следующий символ в потоке без смещения указателя
- Read- читает один символ из потока
- ReadLine- читает строку
- ReadToEnd- читает файл целиком

# Методы класса TextWrite

- Close – закрывает поток и освобождает ресурсы
- Write- записывает в поток любые базовые типы данных в текстовом формате
- Writine - записывает в поток любые базовые типы данных в текстовом формате, за которыми записывается CRLF

# Класс StreamReader

- StreamReader класс из библиотеки .NET Framework, для его использования необходимо подключить библиотеку
- Imports System.IO
- Пример, вывод текстового файла в текстовое поле

```
Dim St As StreamReader
    St = New StreamReader("D:\r.txt")
    tst.Text = St.ReadToEnd
    St.Close()
    tst.Select(0, 0)
```

# Запись в файл

```
Dim St As StreamWriter
    St = New StreamWriter("D:\r1.txt")
    St.Write(tst.Text)
    St.Close()
    tst.Select(0, 0)
```

# Классы для работы с файлами

- **System.IO.File и System.IO.FileInfo**
- Классы предназначены для выполнения различных операций с файлами, в том числе создания, удаления, копирования, перемещения и проверки существования файлов. Также с их помощью можно открывать файлы (функции открытия файлов возвращают объекты System. I0. FileStream, используемые при последующих операциях чтения и записи).

# Классы для работы с файлами

- **System. IO.FileSystemInfo**
- Базовый класс для классов System.I0.DirectoryInfo и System. I0. FileInfo. Используется при перемещении в иерархии каталогов для получения информации о каталогах и файлах.

# Классы для работы с файлами

- **System.IO.FileSystemWatcher**
- Класс предназначен для отслеживания событий файловой системы (создания, удаления и модификации файлов).

# Проверка наличия файла

```
Private Function SourceFileExists() As Boolean
        If Not (System.IO.File.Exists(tst.Text)) Then
            MsgBox("The source fil e does not exist!", MsgBoxStyle.Exclamation)
        Else
            SourceFileExists = True
            Return (SourceFileExists)
        End If
    End Function
```

# Копирование файлов

- Файлы копируются с помощью метода
- Copy () объекта System.IO. File
- **If Not {SourceFileExists()} Then Exit Sub**
- **System.I0.File.Copy(tst.Text, tstn.Text)**
- **MsgBox(“The file has been successfully copied.")**

# Перемещение файлов

- При перемещении файла он удаляется из папки, в которой находится, и помещается в новую. При этом можно оставить ему прежнее имя, можно изменить. Перемещение файла выполняется методом Move () объекта System.IO. File.
- **If Not (SourceFileExists{)) Then Exit Sub**
- **System.I0.File.Move(tst.Text, tstn.Text}**
- **MsgBox("The file has been successfully moved.")**

# Переименование файла

- Когда файл переименовывается, то с его содержимым ничего не происходит. Он остается в той же папке, изменяется только его имя. Для переименования файла используется метод Move( )
- Для этого надо указать имя файла и оставить его путь без изменений.

# Удаление файлов

- Метод Delete () физически удаляет файлы
- **If Not SourceFileExists() Then Exit Sub**
- **If MsgBox (“Are you sure you want to delete the source file?", MsgBoxStyle.Question Or MsgBoxStyle.YesNo) = MsgBoxResult.Yes Then**
- **System.I0.FileDelete(tst.Text)**
- **MsgBox("The file has been successfully deleted.")**
- **End If**

# Свойства файла

Сведения о файле доступные через объект File

| Свойства | Описание |
| --- | --- |
| GetCreationTime | Возвращает дату и время создания файла |
| GetLastAccessTime | Возвращает дату и время последнего обращения к файлу |
| GetLastWriteTime | Возвращает дату и время последнего изменения файла |

# Работа с папками, используя объект Directory

- создать папку

**System.IO.Directory.CreateDirectory("c:\my direct")**

- существует ли папка

**MsgBox(System.IO.Directory.Exists("c:\temp")**

- переместить папку

**System.IO.Directory.Move("c:\dir1","d:\dir2")**

- Удалить папку
- **System.I0.Directory.Delete("c:\temp")**