**КУРНОСЕНКО СОФЬЯ**

**№ 3 Регулярные выражения, атрибуты валидации, меню, панели инструментов, строки состояния**

**Задание**

1) Измените функциональность предыдущей лабораторной работы добавить

на форму меню с пунктами:

a. «Поиск» (подменю указанными в вариантах) В поиске, кроме поиска

на полное соответствие, реализовать поиск по на основе регулярных

выражений (диапазон, наличие букв на определенных позициях, число

повторений символов и т.п.). Результаты поисковых запросов можно

выводить в отельное окно. Сделайте отдельное окно для

конструирования поисковых запросов (в том числе и по нескольким

критериям).

b. «Сортировка по» (году, фамилии, специальности и т.п.) Для поиска,

сортировки и модификаций используйте LINQ.

c. «Сохранить» результаты поиска и сортировок в отдельных xml или

json-файл. Используйте сериализацию.

d. Добавьте пункт «О программе». При выборе пункта меню «О

программе» должно выводиться окно сообщений с версией и ФИО

разработчика.

2) Добавьте валидацию данных на основе атрибутов. При валидации вводимых

данных используйте функционал в виде атрибутов из пространства имен

System.ComponentModel.DataAnnotations и классов ValidationResult,

Validator и ValidationContext. Используйте атрибуты RegularExpression,

Range, свойство ErrorMessage и т.д

3) Создайте свой атрибут валидации (см. таблицу с вариантами).

4) Добавить панель инструментов с кнопками дублирующими команды меню

«поиск», «сортировки», «очистить», «удалить», «вперед», «назад». Добавить

возможность скрывать и закреплять панель инструментов.

5) Добавить строку состояния с тестовыми сообщениями о текущем количестве объектов и последнем выполненном действии, текущей датой и временем.

**8 вариант:**

Поиск по: названию, типу, диапазону цены. Сортировка по дате

производства, стране производителя, затем по названию.

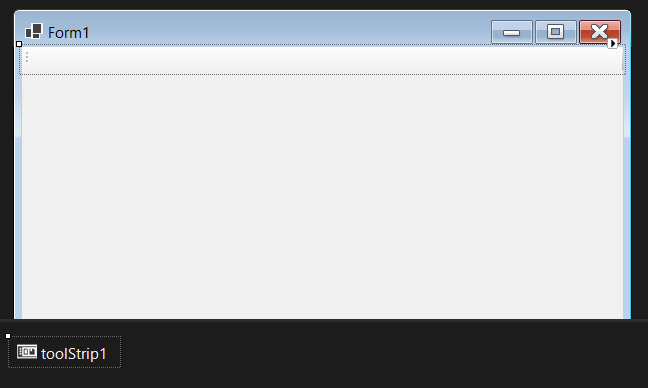
Пользовательский атрибут для валидации инвентарный номер.

**Меню и панели инструментов**

**Панель инструментов ToolStrip**

Элемент ToolStrip представляет панель инструментов. Каждый отдельный элемент на этой панели является объектом ToolStripItem.

ToolStrip в конструкторе выглядит так:



Ключевые свойства компонента ToolStrip связаны с его позиционированием на форме:

▪ *Dock*: прикрепляет панель инструментов к одной из сторон формы

▪ *LayoutStyle*: задает ориентацию панели на форме (горизонтальная, вертикальная, табличная)

▪ *ShowItemToolTips*: указывает, будут ли отображаться всплывающие подсказки для отдельных элементов панели инструментов

▪ *Stretch*: позволяет растянуть панель по всей длине контейнера

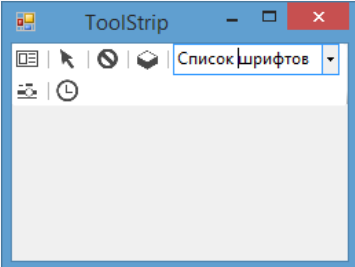
В зависимости от значения свойства *LayoutStyle* панель инструментов может располагаться по горизонтали, или в табличном виде:

▪ *HorizontalStackWithOverflow*: расположение по горизонтали с переполнением - если длина панели превышает длину контейнера, то новые элементы, выходящие за границы контейнера, не отображаются, то есть панель переполняется элементами

▪ *StackWithOverflow*: элементы располагаются автоматически с переполнением

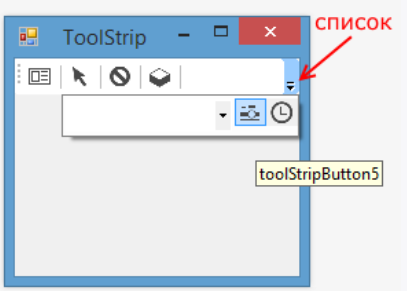
▪ *VerticalStackWithOverflow*: элементы располагаются вертикально с переполнением

▪ *Flow*: элементы располагаются автоматически, но без переполнения - если длина панели меньше длины контейнера, то выходящие за границы элементы переносятся, а панель инструментов растягивается, чтобы вместить все элементы



▪ *Table*: элементы позиционируются в виде таблицы

Если LayoutStyle имеет значения HorizontalStackWithOverflow / VerticalStackWithOverflow, то с помощью свойства *CanOverflow* мы можем задать поведение при переполнении. Так, если это свойство равно true (значение по умолчанию), то для элементов, не попадающих в границы ToolStrip, создается выпадающий список:



При значении false подобный выпадающий список не создается.

**Типы элементов панели и их добавление**

Панель ToolStrip может содержать объекты следующих классов:

▪ *ToolStripLabel*: текстовая метка на панели инструментов, представляет функциональность элементов Label и LinkLabel

▪ *ToolStripButton*: аналогичен элементу Button.Также имеет событие Click, с помощью которого можно обработать нажатие пользователя на кнопку

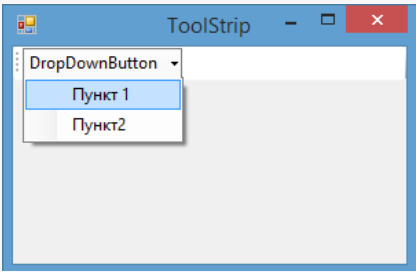
▪ *ToolStripSeparator*: визуальный разделитель между другими элементами на панели инструментов

▪ *ToolStripToolStripComboBox*: подобен стандартному элементу ComboBox

▪ *ToolStripTextBox*: аналогичен текстовому полю TextBox

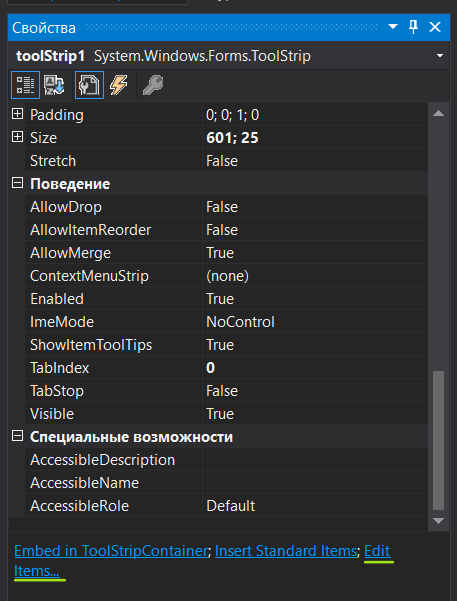
▪ *ToolStripProgressBar*: индикатор прогресса, как и элемент ProgressBar

▪ *ToolStripDropDownButton*: представляет кнопку, по нажатию на которую открывается выпадающее меню

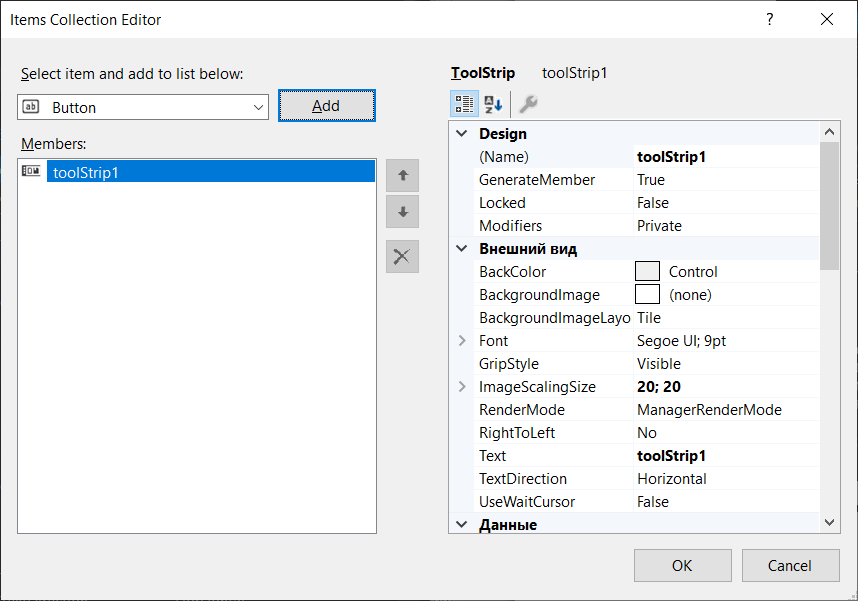


К каждому элементу выпадающего меню дополнительно можно прикрепить обработчик нажатия и обработать клик по этим пунктам меню.

**Чтобы добавить элементы** на панель инструментов в окне свойств нужно выбрать «Edit Items»:



В открывшемся окне можно добавлять и удалять элементы на панель, а также редактировать их свойства:



Также можно **добавлять новые элементы программно в коде**. Их расположение на панели инструментов будет соответствовать порядку добавления. Все элементы хранятся в ToolStrip в свойстве Items. Мы можем добавить в него любой объект класса ToolStripItem (то есть любой из выше перечисленных классов, так как они наследуются от ToolStripItem):

public partial class Form1 : Form

{

public Form1()

{

InitializeComponent();

ToolStripButton clearBtn = new ToolStripButton();

clearBtn.Text = "Clear";

// устанавливаем обработчик нажатия

clearBtn.Click += btn\_Click;

toolStrip1.Items.Add(clearBtn);

}

void btn\_Click(object sender, EventArgs e)

{

MessageBox.Show("Производится удаление");

}

}

Кроме того, здесь задается обработчик, позволяющий обрабатывать нажатия по кнопки на панели инструментов.

Элементы ToolStripButton, ToolStripDropDownButton и ToolStripSplitButton могут отображать как текст, так и изображения, либо сразу и то, и другое. Для управления размещением изображений в этих элементах имеются следующие свойства:

▪ *DisplayStyle*: определяет, будет ли отображаться на элементе текст, или изображение, или и то и другое.

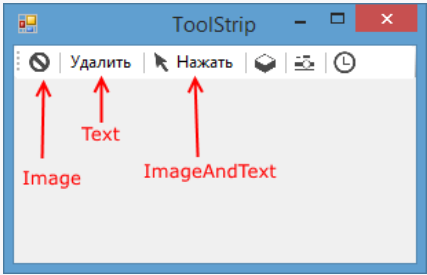
▪ *Image*: указывает на само изображение

▪ *ImageAlign*: устанавливает выравнивание изображения относительно элемента

▪ *ImageScaling*: указывает, будет ли изображение растягиваться, чтобы заполнить все пространство элемента

▪ *ImageTransparentColor*: указывает, будет ли цвет изображения прозрачным

Чтобы разместить изображение на кнопке, у свойства DisplayStyle надо установить значение Image. Если мы хотим, чтобы кнопка отображала только текст, надо указать значение Text, либо можно комбинировать два значения с помощью другого значения ImageAndText:



Все эти значения хранятся в перечислении ToolStripItemDisplayStyle. Также можно установить свойства в коде c#:

ToolStripButton clearBtn = new ToolStripButton();

clearBtn.Text = "Поиск";

clearBtn.DisplayStyle = ToolStripItemDisplayStyle.ImageAndText;

clearBtn.Image = Image.FromFile(@"D:\Icons\0023\search32.png");

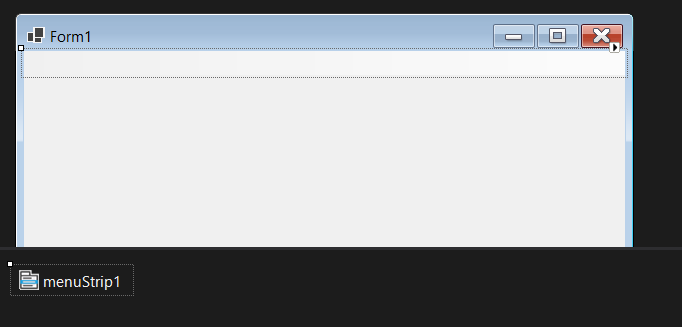
// добавляем на панель инструментов

toolStrip1.Items.Add(clearBtn);

**Создание меню MenuStrip**

Для создания меню в Windows Forms применяется элемент MenuStrip. Данный класс унаследован от ToolStrip и поэтому наследует его функциональность.

Вот так все выгладит в конструкторе:



Наиболее важные свойства компонента MenuStrip:

▪ Dock: прикрепляет меню к одной из сторон формы

▪ LayoutStyle: задает ориентацию панели меню на форме. Может принимать те же значения, что и соответствующее значения элемента ToolStrip,

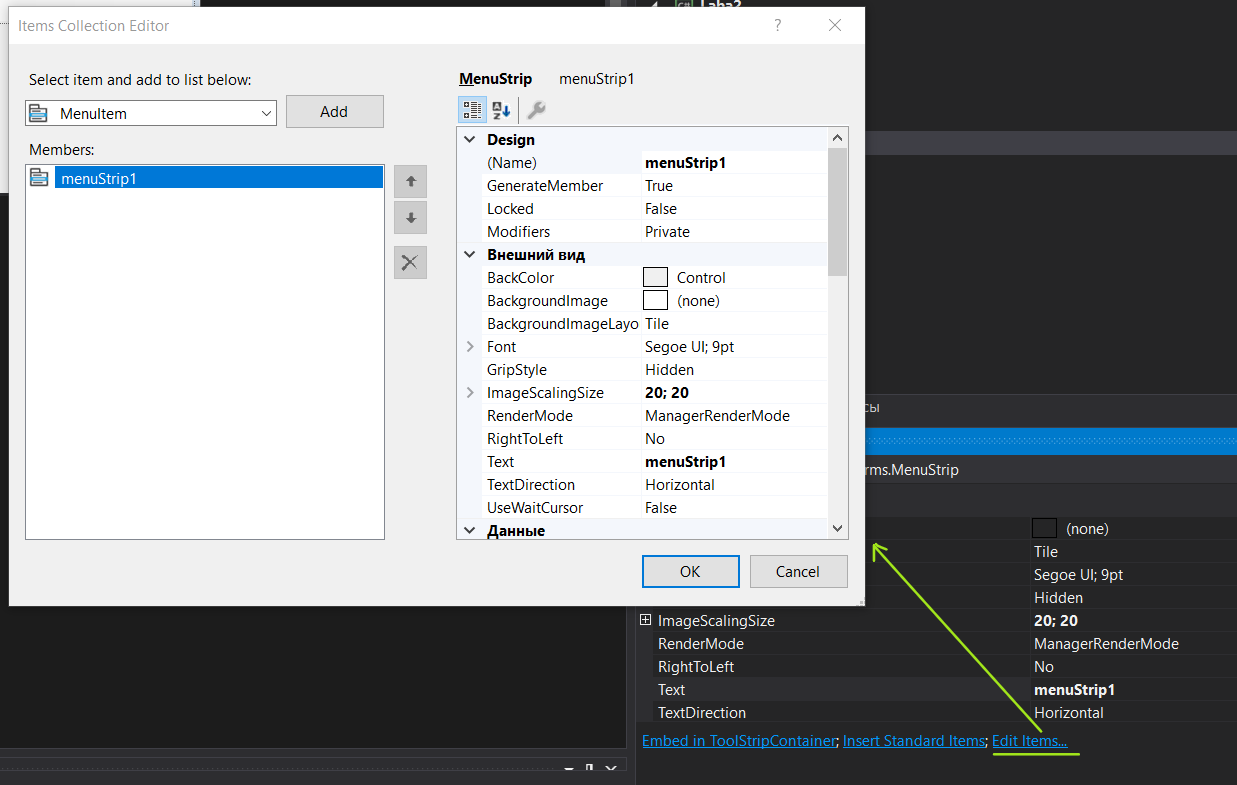
▪ ShowItemToolTips: указывает, будут ли отображаться всплывающие подсказки для отдельных элементов меню

▪ Stretch: позволяет растянуть панель по всей длине контейнера

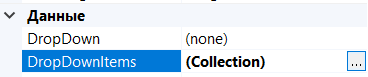
▪ TextDirection: задает направление текста в пунктах меню

MenuStrip выступает своего рода контейнером для отдельных пунктов меню, которые представлены объектом *ToolStripMenuItem*.

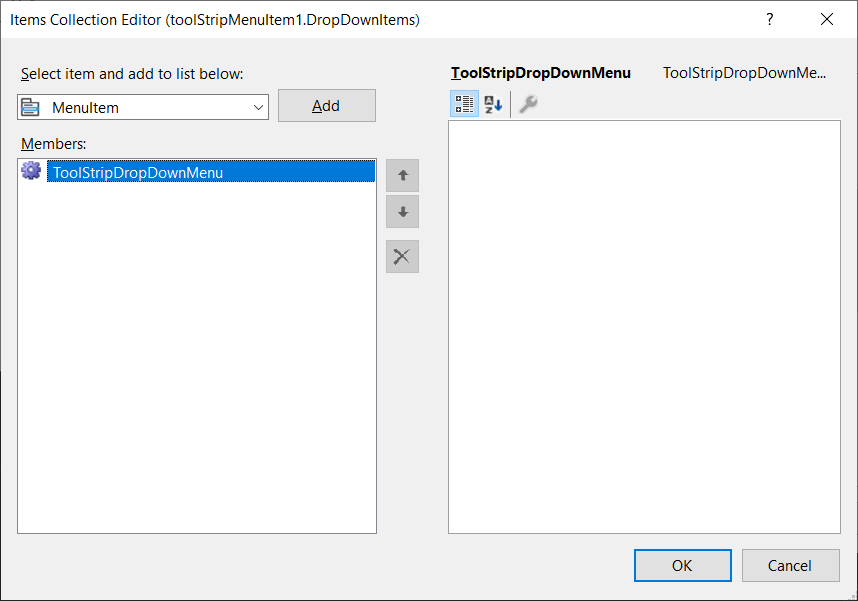
Добавить новые элементы в меню можно в конструкторе:



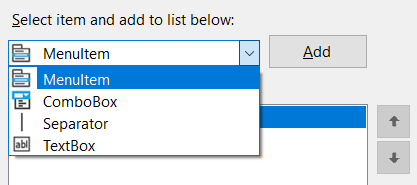
Для создания выпадающего списка справа в свойствах выбираем пункт DropDownItems и нажимаем на троеточие, появившееся правее:



Откроется окно:



Здесь мы ещё разделитель ставить можем:



**Для добавления доступно три вида элементов**: MenuItem (объект ToolStripMenuItem), ComboBox и TextBox. Таким образом, в меню мы можем использовать выпадающие списки и текстовые поля, однако, как правило, эти элементы применяются в основном на панели инструментов. Меню же обычно содержит набор объектов ToolStripMenuItem.

Также мы можем добавить пункты меню в коде C#:

public partial class Form1 : Form

{

public Form1()

{

InitializeComponent();

ToolStripMenuItem fileItem = new ToolStripMenuItem("Файл");

fileItem.DropDownItems.Add("Создать");

fileItem.DropDownItems.Add(new ToolStripMenuItem("Сохранить"));

menuStrip1.Items.Add(fileItem);

ToolStripMenuItem aboutItem = new ToolStripMenuItem("О программе");

aboutItem.Click += aboutItem\_Click;

menuStrip1.Items.Add(aboutItem);

}

void aboutItem\_Click(object sender, EventArgs e)

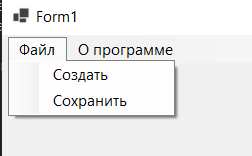
{

MessageBox.Show("О программе");

}

}

При запуске формы:



ToolStripMenuItem в конструкторе принимает текстовую метку, которая будет использоваться в качестве текста меню. Каждый подобный объект имеет коллекцию DropDownItems, которая хранит дочерние объекты ToolStripMenuItem. То есть один элемент ToolStripMenuItem может содержать набор других объектов ToolStripMenuItem. И таким образом, образуется иерархическое меню или структура в виде дерева.

Если передать при добавление строку текста, то для нее неявным образом будет создан объект ToolStripMenuItem: fileItem.DropDownItems.Add("Создать")

Назначив обработчики для события Click, мы можем обработать нажатия на пункты меню: aboutItem.Click += aboutItem\_Click.

**Отметки пунктов меню**

Свойство CheckOnClick при значении true позволяет на клику отметить пункт меню. А с помощью свойства Checked можно установить, будет ли пункт меню отмечен при запуске программы.

Еще одно свойство CheckState возвращает состояние пункта меню - отмечен он или нет. Оно может принимать три значения: Checked (отмечен), Unchecked (неотмечен) и Indeterminate (в неопределенном состоянии)

Например, создадим ряд отмеченных пунктов меню и обработаем событие установки / снятия отметки:

public partial class Form1 : Form

{

public Form1()

{

InitializeComponent();

ToolStripMenuItem fileItem = new ToolStripMenuItem("Файл");

ToolStripMenuItem newItem = new ToolStripMenuItem("Создать") { Checked = true, CheckOnClick = true };

fileItem.DropDownItems.Add(newItem);

ToolStripMenuItem saveItem = new ToolStripMenuItem("Сохранить") { CheckOnClick = true };

saveItem.CheckedChanged += menuItem\_CheckedChanged; // назначаем обработчик

fileItem.DropDownItems.Add(saveItem);

menuStrip1.Items.Add(fileItem);

}

void menuItem\_CheckedChanged(object sender, EventArgs e)

{

ToolStripMenuItem menuItem = sender as ToolStripMenuItem;

if (menuItem.CheckState == CheckState.Checked)

MessageBox.Show("Отмечен");

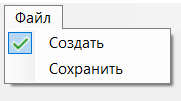
else if (menuItem.CheckState == CheckState.Unchecked)

MessageBox.Show("Отметка снята");

}

}

Итог:



**Клавиши быстрого доступа**

Если нам надо быстро обратиться к какому-то пункту меню, то мы можем использовать клавиши быстрого доступа. Для задания клавиш быстрого доступа используется свойство *ShortcutKeys*:

public partial class Form1 : Form

{

public Form1()

{

InitializeComponent();

ToolStripMenuItem fileItem = new ToolStripMenuItem("Файл");

ToolStripMenuItem saveItem = new ToolStripMenuItem("Сохранить") { Checked = true, CheckOnClick = true };

saveItem.Click += saveItem\_Click;

saveItem.ShortcutKeys = Keys.Control | Keys.P;

fileItem.DropDownItems.Add(saveItem);

menuStrip1.Items.Add(fileItem);

}

void saveItem\_Click(object sender, EventArgs e)

{

MessageBox.Show("Сохранение");

}

}

Клавиши задаются с помощью перечисления *Keys*. В данном случае по нажатию на комбинацию клавиш Ctrl + P, будет срабатывать нажатие на пункт меню "Сохранить".

С помощью **изображений** мы можем разнообразить внешний вид пунктов меню. Для этого мы можем использовать следующие свойства:

▪ DisplayStyle: определяет, будет ли отображаться на элементе текст, или изображение, или и то и другое.

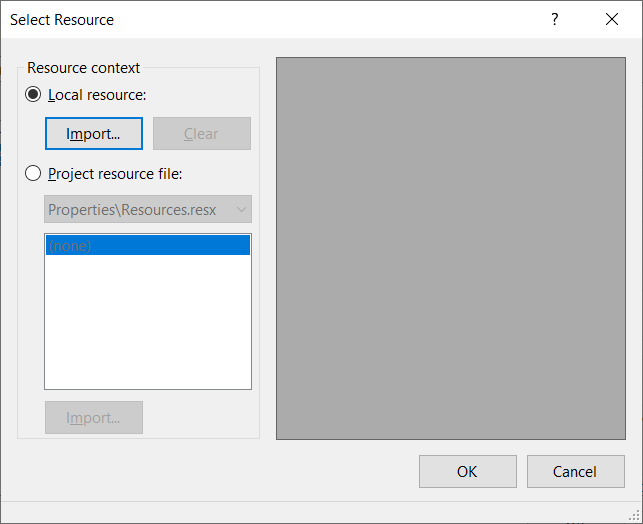
▪ Image: указывает на само изображение

▪ ImageAlign: устанавливает выравнивание изображения относительно элемента

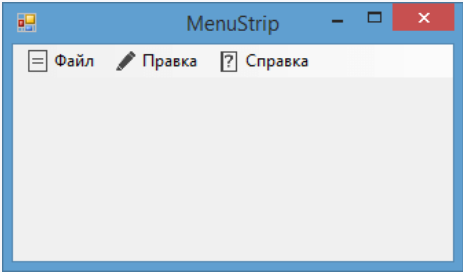
▪ ImageScaling: указывает, будет ли изображение растягиваться, чтобы заполнить все пространство элемента

▪ ImageTransparentColor: указывает, будет ли цвет изображения прозрачным

Если изображение для пункта меню устанавливает в режиме дизайнера, то нам надо выбрать в окне свойство пункт Image, после чего откроется окно для импорта ресурса изображения в проект:



Чтобы указать, как разместить изображение, у свойства DisplayStyle надо установить значение Image. Если мы хотим, чтобы кнопка отображала только текст, то надо указать значение Text, либо можно комбинировать два значения с помощью другого значения ImageAndText. По умолчанию изображение размещается слева от текста:



Также можно установить изображение динамически в коде:

fileToolStripMenuItem.Image = Image.FromFile(@"D:\Icons\0023\block32.png");

**Строка состояния StatusStrip**

StatusStrip представляет строку состояния, во многом аналогичную панели инструментов ToolStrip. Строка состояния предназначена для отображения текущей информации о состоянии работы приложения.

При добавлении на форму StatusStrip автоматически размещается в нижней части окна приложения (как и в большинстве приложений). Однако при необходимости мы сможем его иначе позиционировать, управляя свойством Dock, которое может принимать следующие значения:

▪ Bottom: размещение внизу (значение по умолчанию)

▪ Top: прикрепляет статусную строку к верхней части формы

▪ Fill: растягивает на всю форму

▪ Left: размещение в левой части формы

▪ Right: размещение в правой части формы

▪ None: произвольное положение

StatusStrip может содержать различные элементы.

Можно обратиться на панели свойств к свойству Items компонента StatusStrip и открывшемся окне добавить и настроить все элементы:

▪ StatusLabel: метка для вывода текстовой информации. Представляет объект ToolStripLabel

▪ ProgressBar: индикатор прогресса. Представляет объект ToolStripProgressBar

▪ DropDownButton: кнопка с выпадающим списком по клику. Представляет объект ToolStripDropDownButton

▪ SplitButton: еще одна кнопка, во многом аналогичная DropDownButton. Представляет объект ToolStripSplitButton

Также мы можем добавить элементы программно. Создадим небольшую программу. Определим следующий код формы:

public partial class Form1 : Form

{

ToolStripLabel dateLabel;

ToolStripLabel timeLabel;

ToolStripLabel infoLabel;

Timer timer;

public Form1()

{

InitializeComponent();

infoLabel = new ToolStripLabel();

infoLabel.Text = "Текущие дата и время:";

dateLabel = new ToolStripLabel();

timeLabel = new ToolStripLabel();

statusStrip1.Items.Add(infoLabel);

statusStrip1.Items.Add(dateLabel);

statusStrip1.Items.Add(timeLabel);

timer = new Timer() { Interval = 1000 };

timer.Tick += timer\_Tick;

timer.Start();

}

void timer\_Tick(object sender, EventArgs e)

{

dateLabel.Text = DateTime.Now.ToLongDateString();

timeLabel.Text = DateTime.Now.ToLongTimeString();

}

}

Здесь создаются три метки на строке состояния и таймер. После создания формы таймер запускается, и срабатывает его событие Tick, в обработчике которого устанавливаем текст меток.

