# Контейнеры в Windows Forms

Для организации элементов управления в связанные группы существуют специальные элементы - контейнеры. Например, Panel, FlowLayoutPanel, SplitContainer, GroupBox. Ту же форму также можно отнести к контейнерам. Использование контейнеров облегчает управление элементами, а также придает форме определенный визуальный стиль.

Все контейнеры имеют свойство Controls, которое содержит все элементы данного контейнера. Когда мы переносим какой-нибудь элемент с панели инструментов на контейнер, например, кнопку, она автоматически добавляется в данную коллекцию данного контейнера. Либо мы также можем добавить элемент управления динамически с помощью кода в эту же коллекцию.

## Динамическое добавление элементов

Добавим на форму кнопку динамически. Для этого добавим событие загрузки формы, в котором будет создаваться новый элемент управления. Это можно сделать либо с помощью кода, либо визуальным образом.

С помощью перетаскивания элементов с Панели Инструментов можно легко добавить новые элементы на форму. Однако такой способ довольно ограничен, поскольку очень часто требуется динамически создавать (удалять) элементы на форме.

Для динамического добавления элементов создадим обработчик события загрузки формы в файле кода:

|  |  |
| --- | --- |
| 1  2  3 | private void Form1\_Load(object sender, EventArgs e)  {  } |

Теперь добавим в него код добавления кнопки на форму:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9 | private void Form1\_Load(object sender, EventArgs e)  {      Button helloButton = new Button();      helloButton.BackColor = Color.LightGray;      helloButton.ForeColor = Color.DarkGray;      helloButton.Location = new Point(10, 10);      helloButton.Text = "Привет";      this.Controls.Add(helloButton);  } |

Сначала создаем кнопку и устанавливаем ее свойства. Затем, используя метод Controls.Add мы добавляем ее в коллекцию элементов формы. Если бы мы это не сделали, мы бы кнопку не увидели, поскольку в этом случае для нашей формы ее просто не существовало бы.

С помощью метода Controls.Remove() можно удалить ранее добавленный элемент с формы:

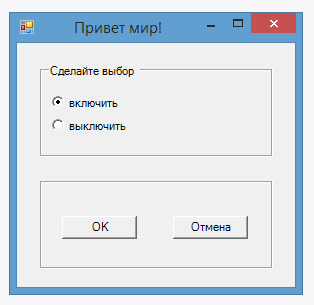
|  |  |
| --- | --- |
| 1 | this.Controls.Remove(helloButton); |

Хотя в данном случае в качестве контейнера использовалась форма, но при добавлении и удалении элементов с любого другого контейнера, например, GroupBox, будет применяться все те же методы.

## Элементы GroupBox, Panel и FlowLayoutPanel

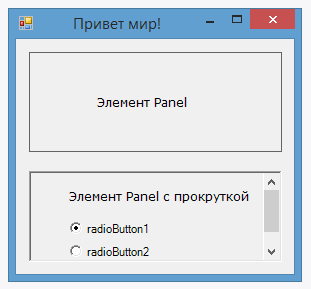
GroupBox представляет собой специальный контейнер, который ограничен от остальной формы границей. Он имеет заголовок, который устанавливается через свойство Text. Чтобы сделать GroupBox без заголовка, в качестве значения свойства Text просто устанавливается пустая строка.

Нередко этот элемент используется для группирования переключателей - элементов RadioButton, так как позволяет разграничить их группы.



Элемент Panel представляет панель и также, как и GroupBox, объединяет элементы в группы. Она может визуально сливаться с остальной формой, если она имеет то же значение цвета фона в свойстве BackColor, что и форма. Чтобы ее выделить можно кроме цвета указать для элемента границы с помощью свойства BorderStyle, которое по умолчанию имеет значение None, то есть отсутствие границ.

Также если панель имеет много элементов, которые выходят за ее границы, мы можем сделать прокручиваемую панель, установив ее свойство AutoScroll в true



Также, как и форма, GroupBox и Panel имеют коллекции элементов, и мы также можем динамически добавлять в эти контейнеры элементы. Например, на форме есть элемент GroupBox, который имеет имя groupBox1:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9 | private void Form1\_Load(object sender, EventArgs e)  {      Button helloButton = new Button();      helloButton.BackColor = Color.LightGray;      helloButton.ForeColor = Color.Red;      helloButton.Location = new Point(30, 30);      helloButton.Text = "Привет";      groupBox1.Controls.Add(helloButton);  } |

Для указания расположения элемента в контейнере мы используем структуру Point: new Point(30, 30);, которой в конструкторе передаем размещение по осям Х и Y. Эти координаты устанавливаются относительно левого верхнего угла контейнера - то есть в данном случае элемента GroupBox.

При этом надо учитывать, что контейнером верхнего уровня является форма, а элемент groupBox1 сам находится в коллекции элементов формы. И при желании мы могли бы удалить его:

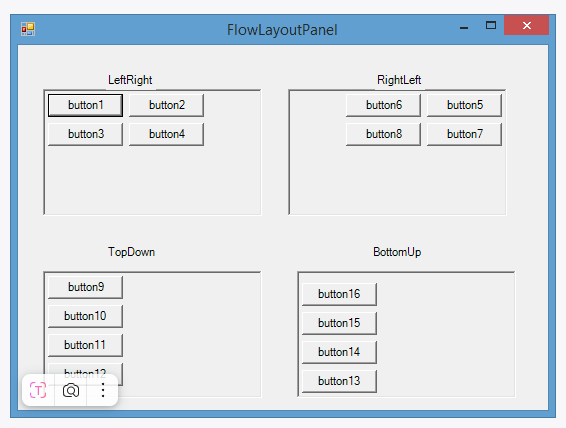
|  |  |
| --- | --- |
| 1 | this.Controls.Remove(groupBox1); |

### FlowLayoutPanel

Элемент FlowLayoutPanel является унаследован от класса Panel, и поэтому наследует все его свойства. Однако при этом добавляя дополнительную функциональность. Так, этот элемент позволяет изменять позиционирование и компоновку дочерних элементов при изменении размеров формы во время выполнения программы.

Свойство элемента **FlowDirection** позволяет задать направление, в котором направлены дочерние элементы. По умолчанию имеет значение LeftToRight - то есть элементы будут располагаться начиная от левого верхнего края. Следующие элементы будут идти вправо. Это свойство также может принимать следующие значения:

* **RightToLeft** - элементы располагаются от правого верхнего угла в левую сторону
* **TopDown** - элементы располагаются от левого верхнего угла и идут вниз
* **BottomUp** - элементы располагаются от левого нижнего угла и идут вверх

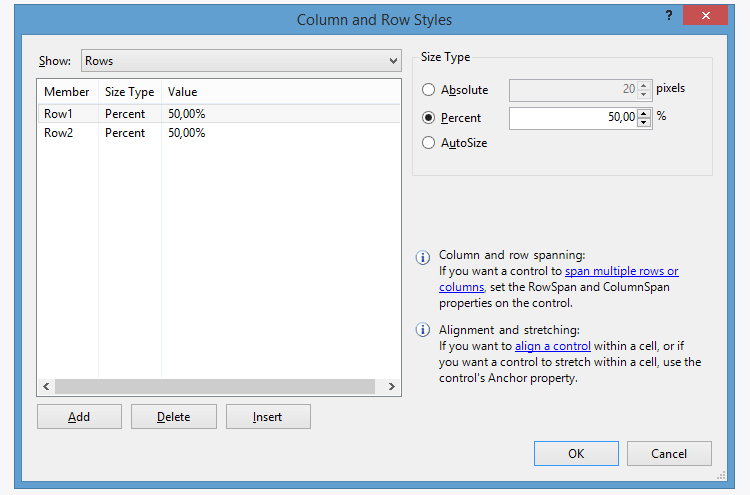


При расположении элементов важную роль играет свойство WrapContents. По умолчанию оно имеет значение True. Это позволяет переносить элементы, которые не умещаются в FlowLayoutPanel, на новую строку или в новый столбец. Если оно имеет значение False, то элементы не переносятся, а к контейнеру просто добавляются полосы прокрутки, если свойство AutoScroll равно true.

TableLayoutPanel

Элемент TableLayoutPanel также переопределяет панель и располагает дочерние элементы управления в виде таблицы, где для каждого элемента имеется своя ячейка. Если нам хочется поместить в ячейку более одного элемента, то в эту ячейку добавляется другой компонент TableLayoutPanel, в который затем вкладываются другие элементы.

Чтобы установить нужное число строки столбцов таблицы, мы можем использовать свойства Rows и Columns соответственно. Выбрав один из этих пунктов в окне Properties (Свойства), нам отобразится следующее окно для настройки столбцов и строк:



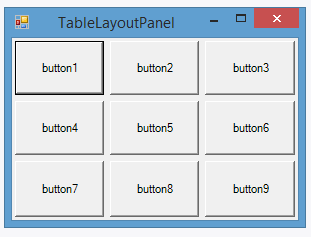
В поле Size Type мы можем указать размер столбцов / строк. Нам доступны три возможные варианта:

* Absolute: задается абсолютный размер для строк или столбцов в пикселях
* Percent: задается относительный размер в процентах. Если нам надо создать резиновый дизайн формы, чтобы ее строки и столбцы, а также элементы управления в ячейках таблицы автоматически масштабировались при изменении размеров формы, то нам нужно использовать именно эту опцию
* AutoSize: высота строк и ширина столбцов задается автоматически в зависимости от размера самой большой в строке или столбце ячейки

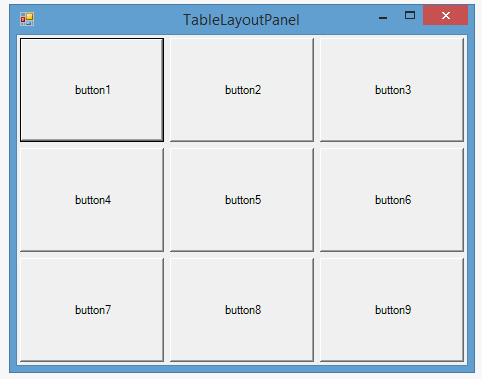
Также мы можем комбинировать эти значения, например, один столбец может быть фиксированным с абсолютной шириной, а остальные столбцы могут иметь ширину в процентах.

В этом диалоговом окне мы также можем добавить или удалить строки и столбцы. В тоже время графический дизайнер в Visual Studio не всегда сразу отображает изменения в таблице - добавление или удаление строк и столбцов, изменение их размеров, поэтому, если изменений на форме никаких не происходит, надо ее закрыть и потом открыть заново в графическом дизайнере.

Итак, например, у меня имеется три столбца и три строки размер у которых одинаков - 33.33%. В каждую ячейку таблицы добавлена кнопка, у которой установлено свойство Dock=Fill.



Если я изменю размеры формы, то автоматически масштабируются и строки и столбцы вместе с заключенными в них кнопками:



Что довольно удобно для создания масштабируемых интерфейсов.

В коде динамически мы можем изменять значения столбцов и строк. Причем все столбцы представлены типом **ColumnStyle**, а строки - типом **RowStyle**:

|  |  |
| --- | --- |
| 1  2  3  4  5 | tableLayoutPanel1.RowStyles[0].SizeType = SizeType.Percent;  tableLayoutPanel1.RowStyles[0].Height = 40;    tableLayoutPanel1.ColumnStyles[0].SizeType = SizeType.Absolute;  tableLayoutPanel1.ColumnStyles[0].Width = 50; |

Для установки размера в ColumnStyle и RowStyle определено свойство SizeType, которое принимает одно из значений одноименного перечисления SizeType

Добавление элемента в контейнер TableLayoutPanel имеет свои особенности. Мы можем добавить его как в следующую свободную ячейку или можем явным образом указать ячейку таблицы:

|  |  |
| --- | --- |
| 1  2  3  4  5 | Button saveButton = new Button();  // добавляем кнопку в следующую свободную ячейку  tableLayoutPanel1.Controls.Add(saveButton);  // добавляем кнопку в ячейку (2,2)  tableLayoutPanel1.Controls.Add(saveButton, 2, 2); |

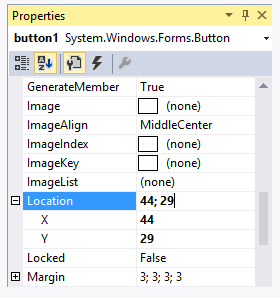
В данном случае добавляем кнопку в ячейку, образуемую на пересечении третьего

столбца и третьей строки. Правда, если у нас нет столько строк и столбцов, то система автоматически выберет нужную ячейку для добавления.

## Размеры элементов и их позиционирование в контейнере

### Позиционирование

Для каждого элемента управления мы можем определить свойство Location, которое задает координаты верхнего левого угла элемента относительно контейнера. При переносе элемента с панели инструментов на форму это свойство устанавливается автоматически. Однако потом в окне Свойств мы можем вручную поправить координаты положения элемента:

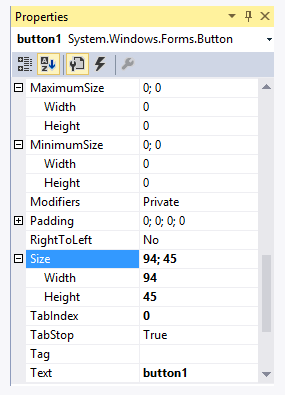


Также мы можем установить позицию элемента в коде:

|  |  |
| --- | --- |
| 1  2  3  4 | private void Form1\_Load(object sender, EventArgs e)  {      button1.Location = new Point(50, 50);  } |

### Установка размеров

С помощью свойства **Size** можно задать размеры элемента:



Дополнительные свойства MaximumSize и MinimumSize позволяют ограничить минимальный и максимальный размеры.

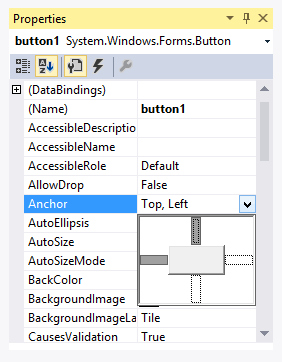
Установка свойств в коде:

|  |  |
| --- | --- |
| 1  2  3  4 | button1.Size = new Size { Width = 50, Height = 25 };  // установка свойств по отдельности  button1.Width = 100;  button1.Height = 35; |

### Свойство Anchor

Дополнительные возможности по позиционировании элемента позволяет определить свойство Anchor. Это свойство определяет расстояние между одной из сторон элемента и стороной контейнера. И если при работе с контейнером мы будем его растягивать, то вместе с ним будет растягиваться и вложенный элемент.

По умолчанию у каждого добавляемого элемента это свойство равно Top, Left:



Это значит, что если мы будем растягивать форму влево или вверх, то элемент сохранит расстояние от левой и верхней границы элемента до границ контейнера, в качестве которого выступает форма.

Мы можем задать четыре возможных значения для этого свойства или их комбинацию:

* Top
* Bottom
* Left
* Right

Например, если мы изменим значение этого свойства на противоположное - Bottom, Right, тогда у нас будет неизменным расстояние между правой и нижней стороной элемента и формой.

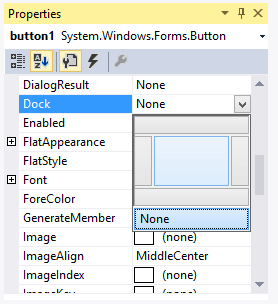
При этом надо отметить, что данное свойство учитывает расстояние до границ контейнера, а не формы. То есть если у нас на форме есть элемент Panel, а на Panel расположена кнопка, то на кнопку будет влиять изменение границ Panel, а не формы. Растяжение формы будет в этом случае влиять только, если оно влияет на контейнер Panel.

Чтобы задать это свойство в коде, надо использовать перечисление **AnchorStyles**:

|  |  |
| --- | --- |
| 1  2  3 | button1.Anchor = AnchorStyles.Left;  // задаем комбинацию значений  button1.Anchor = AnchorStyles.Left | AnchorStyles.Top; |

### Свойство Dock

Свойство Dock позволяет прикрепить элемент к определенной стороне контейнера. По умолчанию оно имеет значение None, но также позволяет задать еще пять значений:



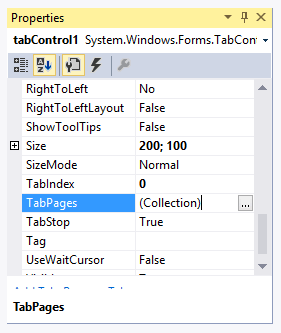
* Top: элемент прижимается к верхней границе контейнера
* Bottom: элемент прижимается к нижней границе контейнера
* Left: элемент прижимается к левой стороне контейнера
* Right: элемент прикрепляется к правой стороне контейнера
* Fill: элемент заполняет все пространство контейнера

## Панель вкладок TabControl и SplitContainer

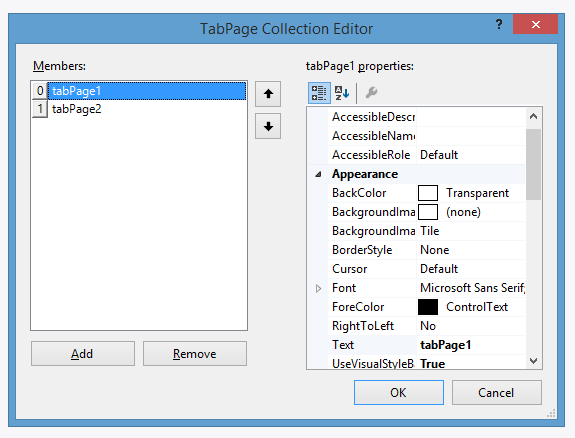
### TabControl

Элемент TabControl позволяет создать элемент управления с несколькими вкладками. И каждая вкладка будет хранить некоторый набор других элементов управления, как кнопки, текстовые поля и др. Каждая вкладка представлена классом **TabPage**.

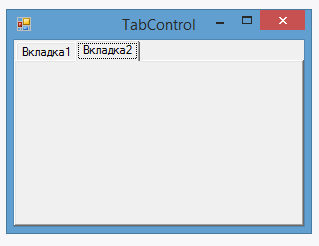
Чтобы настроить вкладки элемента TabControl используем свойство **TabPages**. При переносе элемента TabControl с панели инструментов на форму по умолчанию создаются две вкладки - tabPage1 и tabPage2. Изменим их отображение с помощью свойства TabPages:



Нам откроется окно редактирования/добавления и удаления вкладок:



Каждая вкладка представляет своего рода панель, на которую мы можем добавить другие элементы управления, а также заголовок, с помощью которого мы можем переключаться по вкладкам. Текст заголовка задается с помощью свойства Text.



#### Управление вкладками в коде

Для добавления новой вкладки нам надо ее создать и добавить в коллекцию tabControl1.TabPages с помощью метода Add:

|  |  |
| --- | --- |
| 1  2  3  4 | //добавление вкладки  TabPage newTabPage = new TabPage();  newTabPage.Text = "Континенты";  tabControl1.TabPages.Add(newTabPage); |

Удаление так же просто:

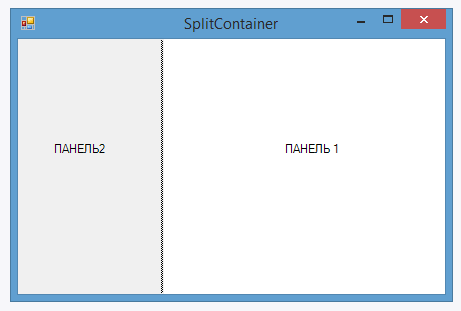
|  |  |
| --- | --- |
| 1  2  3  4  5 | // удаление вкладки  // по индексу  tabControl1.TabPages.RemoveAt(0);  // по объекту  tabControl1.TabPages.Remove(newTabPage); |

Получая в коллекции tabControl1.TabPages нужную вкладку по индексу, мы можем ей легко манипулировать:

|  |  |
| --- | --- |
| 1  2 | // изменение свойств  tabControl1.TabPages[0].Text = "Первая вкладка"; |

### SplitContainer

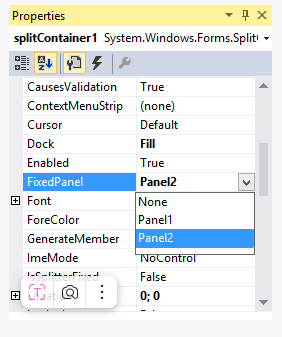
Элемент SplitContainer позволяет создавать две разделенные сплитером панели. Изменяя положение сплитера, можно изменить размеры этих панелей.



Используя свойство Orientation, можно задать горизонтальное или вертикальное отображение сплитера на форму. В данном случае это свойство принимает значения Horisontal и Vertical соответственно.

В случае, когда надо запретить изменение положения сплиттера, то можно присвоить свойству IsSplitterFixed значение true. Таким образом, сплитер окажется фиксированным, и мы не сможем поменять его положение.

По умолчанию при растяжении формы или ее сужении также будет меняться размер обеих панелей сплитконтейнера. Однако мы можем закрепить за одной панелью фиксированную ширину (при вертикальной ориентации сплиттера) или высоту (при горизонтальной ориентации сплиттера). Для этого нам надо установить у элемента SplitContainer свойство FixedPanel. В качестве значения оно принимает панель, которую надо зафиксировать:



Чтобы изменить положение сплитера в коде, мы можем управлять свойством **SplitterDistance**, которое задает положение сплиттера в пикселях от левого или верхнего края элемента SplitContainer. А с помощью свойства SplitterIncrement можно задать шаг, на который будет перемещаться сплиттер при движении его с помощью клавиш-стрелок.

Чтобы скрыть одну из двух панелей, мы можем установить свойство Panel1Collapsed или Panel2Collapsed в true