Лабораторная работа
Разработка элементов управления Windows Forms

Платформа .NET Framework предоставляет широкие возможности по созданию элементов управления Windows Forms. При разработке элементов управления применяются два основных подхода: расширение функций существующего элемента управления путем наследования и разработка составных пользовательских элементов управления.

Класс Control

Базовым классом для всех элементов управления Windows Forms является класс Control. Он обеспечивает инфраструктуру, необходимую для визуального отображения в приложениях Windows Forms. Класс Control выполняет следующие задачи:

- Предоставление дескриптора окна.
- Управление маршрутизацией сообщений.
- Предоставление событий мыши и клавиатуры и многих других событий пользовательского интерфейса.
- Предоставление функций размещения в контейненре.
- Большое число функций, относящихся к визуальному представлению, таких как ForeColor, BackColor, Height, и Width.

Типы элементов управления

Согласно документации, Windows Forms поддерживает три типа определяемых пользователем элементов управления: составной, расширенный и пользовательский.

Составные элементы управления

Составной элемент управления представляет собой набор элементов управления Windows Forms, инкапсулированный в общий контейнер. Этот тип элемента управления часто называют пользовательским (User Control). Элементы управления, входящие в состав составных элементов управления, называются составляющими элементами управления.

Составной элемент управления сохраняет все функции, связанные с каждым составляющим элементом управления Windows Forms, и позволяет выборочно представлять и связывать их свойства. Например, составной элемент управления может служить для отображения адресных данных заказчика из базы данных. Составной элемент управления является производным от класса UserControl.

Использование составного элемента управления имеет смысл в случае, если требуется объединить функции нескольких элементов управления в один блок для повторного использования.

Расширенные элементы управления

Наследуемый элемент управления можно получить из любого существующего элемента Windows Forms. Такой подход позволяет сохранить все стандартные функции элемента управления Windows Forms, а затем расширить их путем добавления пользовательских свойств, методов или других функций. Например, можно создать производный от Button элемент управления, отслеживающий число раз его нажатия пользователем.

В некоторых элементах управления к графическому интерфейсу можно добавить пользовательское оформление путем переопределения метода OnPaint базового класса. Например в кнопке, отслеживающей нажатия, метод OnPaint можно переопределить для вызова базовой реализации OnPaint, а затем отобразить число нажатий в углу клиентской области элемента управления Button.

Наследование из элемента управления Windows Forms рекомендуется применять в следующих случаях:

- Если большинство необходимых функций аналогичны функциям уже существующего элемента управления Windows Forms.

- Если нестандартный графический интерфейс пользователя не требуется или необходимо разработать новый графический интерфейс для существующего элемента управления.

Пользовательские элементы управления

Другим способом разработки элемента управления является создание его с нуля путем наследования класса Control. Класс Control предоставляет все основные функции, необходимые элементам управления (включая события обработки мыши и клавиатуры), однако он не содержит функции для элемента управления и графический интерфейс.

Создание элемента управления путем наследования из класса Control более сложная задача, чем наследование из UserControl или существующего элемента управления Windows Forms. Поскольку задача реализации в больше мере возлагается на пользователя. Создаваемый элемент управления обладает большей гибкостью, чем составной или расширенный элемент управления, что позволяет подстроить его под конкретные потребности.

Для реализации элемента управления необходимо написать код для события OnPaint элемента управления, а также код для необходимой функции. Кроме того, можно переопределить метод WndProc и обработать сообщения Windows напрямую. Примером пользовательского элемента управления может служить элемент "Часы", который выглядит и действует аналогично часам со стрелками. Можно применить пользовательское оформление, чтобы заставить стрелки часов двигаться в зависимости от событий Tick внутреннего компонента Timer.

Наследование класса Control имеет смысл в следующих случаях:

- Если требуется определить пользовательское графическое представление элемента управления.
- Если требуется реализовать пользовательские функции, которые недоступны в стандартных элементах управления.

Упражнение 1. Наследование от стандартного элементов управления

Задание

Необходимо разработать элемент управления, позволяющий вводить текст и выделяющий значения, которые не могут быть интерпретированы как число, красным цветом.

Решение

Наследование от стандартного элемента TextBox является наиболее удачным решением, т.к. стандартная функциональность дополняется новыми возможностями.

1. Создайте новый проект на базе шаблона «Windows Control Library». Назовите решение «WinFormsControlLab», а проект «LabControls».
2. Удалите из проекта созданный по умолчанию элемент управления «UserControl1.cs».
3. Добавьте в проект новый элемент на основе шаблона «Component Class» и назовите его NumberBox.

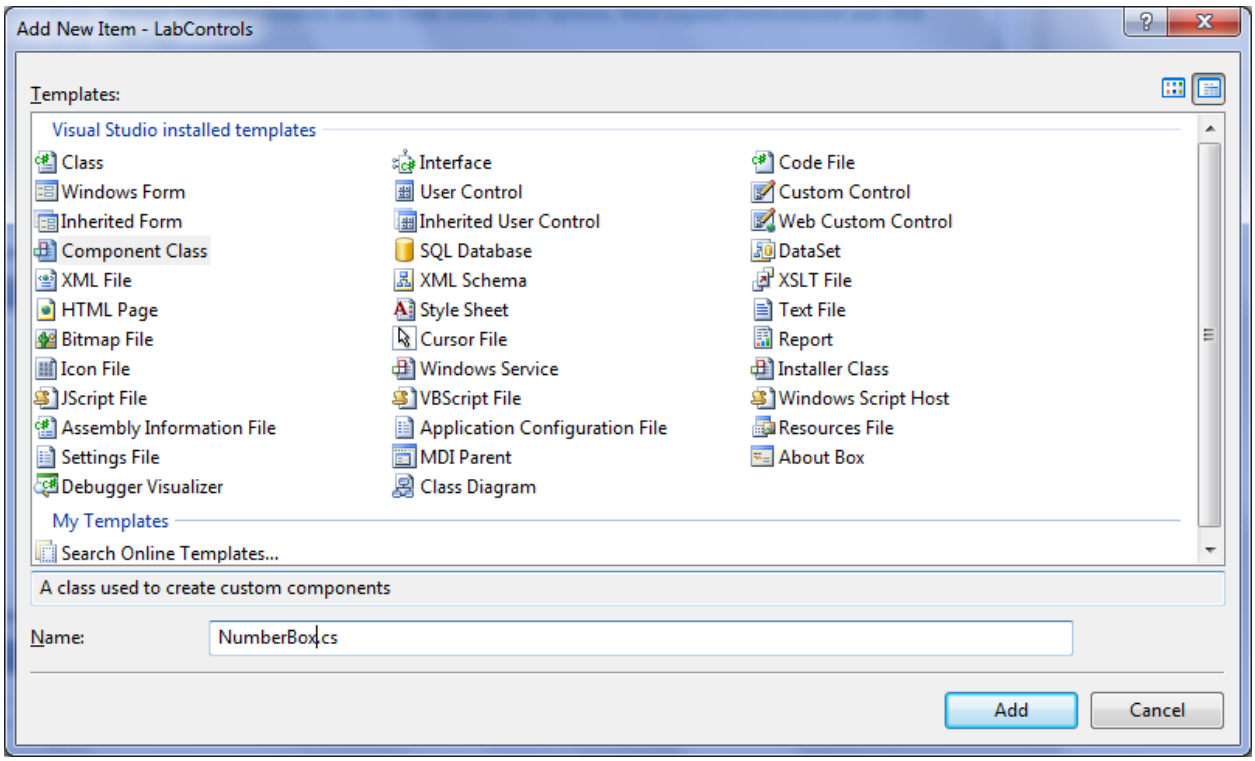

Рис. 1.

4. В программном коде добавленного элемента замените наследование от класса Component наследованием от класса . Не забудьте добавить строку использования пространства имен.

```
using System.Windows.Forms;
```

5. Переопределите метод OnTextChanged. Добавьте в него проверку на число. Если текст не является числом, он отображается красным цветом.

```
protected override void OnTextChanged(EventArgs e)
{
    double x;
    if (!double.TryParse(Text, out x))
        ForeColor=Color.Red;
    else
        ForeColor = Color.Black;
    base.OnTextChanged(e);
}
```

6. Добавьте в решение новый проект на основе шаблона «Windows Application» и назовите его «TestControlsApplication». Этот проект мы будем использовать для тестирования создаваемого элемента управления.

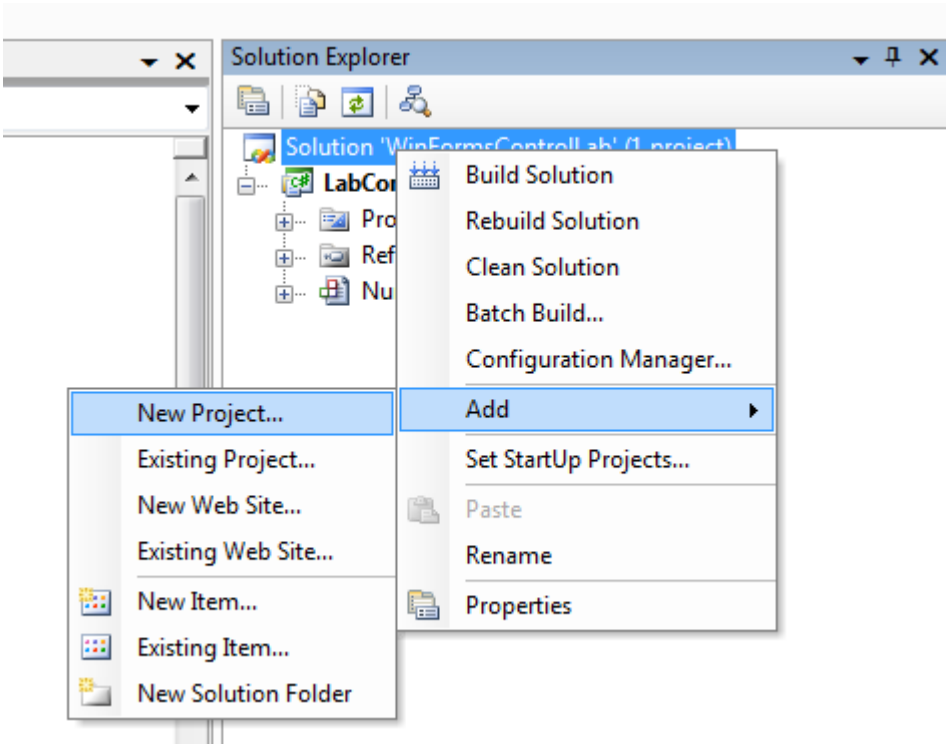

Рис. 2.

7. С помощью команды «Set as StartUp Project» назначте проект TestControlsApplication для запуска при отладке.
8. После построения решения («Build|Build Solution») на панели элементов управления должен появиться разрабатываемый элемент управления. Добавьте на форму проекта «TestControlsApplication» несколько элементов управления NumberBox.

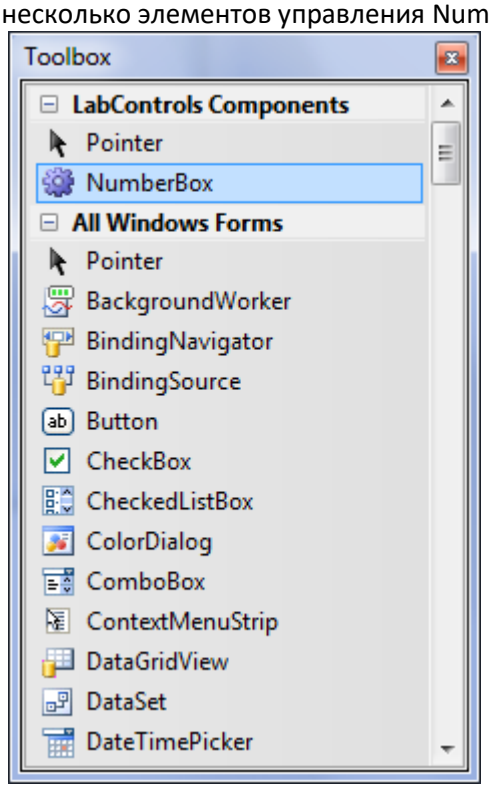

Рис. 3.

9. Запустите приложение и убедитесь в его работоспособности.

Упражнение 2. Разработка пользовательского элемента управления

Задание

Необходимо разработать элемент управления, позволяющий указывать путь к файлу в файловой системе. Должны присутствовать возможности как выбора файла с помощью стандартного диалогового окна, так и непосредственного указания пути строкой.

Решение

В данном случае необходимо использовать решение на базе UserControl, т.к. элемент управления комбинирует несколько стандартных элементов управления. Командная кнопка используется для вызова диалога выбора файла, а TextBox для вывода выбранного файла и ввода текста пользователем.

1. Добавьте в проект «LabControls» новый элемент на базе шаблона UserControl и назовите его «FilePathSelect».

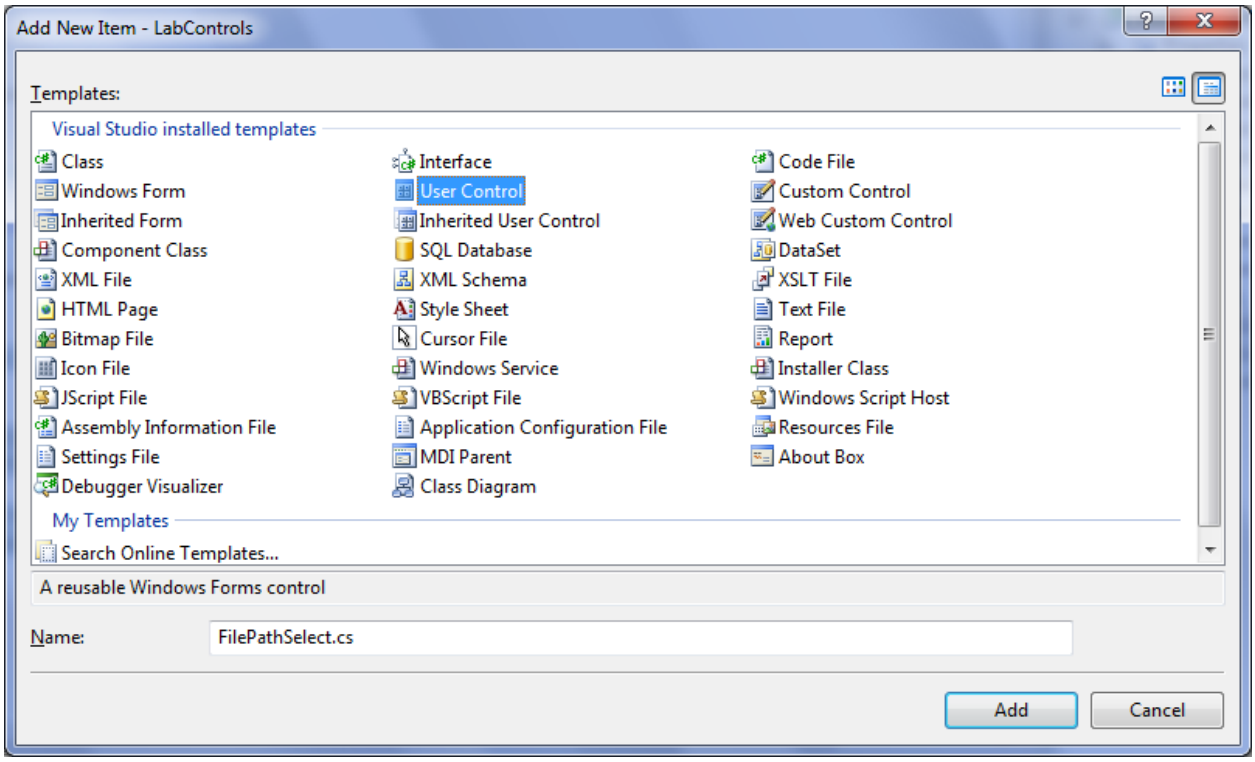

Рис. 4.

2. Расположите на элементе управления командную кнопку, присвоив ей имя btnSelectPath, и TextBox с именем txtPath.

Рис. 5.

3. Спомощью ствойства Anchor настройте поведение элементов так, чтобы они адекватно изменяли свои размеры при изменении размеров радительского элемента управления.
4. На событие кнопки Click добавьте код выбора имени файла с помощью стандартного диалогового окна.

```
private void btnSelectPath_Click(object sender, EventArgs e)
{
    OpenFileDialog ofd = new OpenFileDialog();
    ofd.FileName = txtPath.Text;
    if (ofd.ShowDialog() == DialogResult.OK)
        txtPath.Text = ofd.FileName;
}
```

5. Для доступа к выбранному пути добавим к разрабатываемый класс свойство FileName.

```
public string FileName
{
    get
    {
        return txtPath.Text;
    }
    set
    {
        txtPath.Text = value;
    }
}
```

6. Добавьте на форму проекта «TestControlsApplication» несколько элементов управления FilePathSelect. Обратите внимание нато, что свойство FileName отображается в окне свойств «Properties». Попробуйте его изменить, изменения должны отображатся в элементе управления.
7. Запустите приложение и убедитесь в его работоспособности.

Задание 1. Разработка пользовательского элемента для выбора цвета

Задание

Необходимо разработать элемент управления, позволяющий задавать цвет в системе RGB указанием интенсивности его составляющих: красного, зеленого и синего цветов.

Рис. 6.

Требования

a. Интенсивность цвета может быть задана как в десятичной системе исчисления, так и в шестнадцатеричной. Режим ввода задается специальными переключателями (Dec и Hex).
b. Просмотр цвета должен осуществляться в интерактивном режиме, непосредственно при изменении значений в текстовых полях.
c. При вводе значений должен осуществляться контроль вводимых символов. В десятичном режиме допустимы только цифровые символы, в шестнадцатеричном режиме допустимы цифровые символы и символы A-F без учета регистра.
d. При вводе значений должен контроль попадания значения в диапазон от 0 до 255 (FF). При выходе числа за пределы должно подставиться максимально близкое допустимое число (0 или 255 (FF)).
e. При смене режима текущее значение должно автоматически преобразоваться.
f. Для ввода параметров должен использоваться специально разработанный элемент управления, наследующий от стандартного TextBox. Вся функциональность по обработке вводимых символов должна быть заключена в этом элементе.
g. Элемент должен иметь свойство типа Color для установки/получения текущего цвета. После установки свойства изменения должны отобразиться (в полях ввода и в поле просмотра цвета).
h. Элемент должен иметь событие, вызываемое при смене цвета в нем.
i. Элемент должен поддерживать вставку данных из буфера обмена.

Рекомендации

1. Для работы с цветом используется класс Color из пространства имен System.Drawing. Получить необходимый цвет из трех его составляющих возможно при помощи метода Color.FromArgb (Int32, Int32, Int32). Для доступа к компонентам цвета (красному, зеленому, синему) используются поля: R, G, B.
2. Для преобразования числа из строкового представления в шестнадцатеричной системе в число можно использовать метод ToInt32 класса Convert: Convert.ToInt32(строка, 16). Класс Convert будет полечен и при других преобразованиях. Другой вариант преобразования Int32.Parse(строка, System.Globalization.NumberStyles.HexNumber).
3. Вывести число строковое представление числа в шестнадцатеричной системе поможет вызов метода ToString("X") или String.Format("{0:X}", число);
4. Игнорировать ввод нецифровых символов возможно при помощи переопределения метода OnKeyPress.

```
protected override void OnKeyPress(KeyPressEventArgs e)
{
    if (!char.IsDigit(e.KeyChar) && !char.IsControl((e.KeyChar)))
        e.Handled = true;
    base.OnKeyPress(e);
}
```

Задание 2. Разработка пользовательского элемента для отображения текущего времени

Задание

Необходимо разработать элемент управления, отображающий текущее время с помощью часовой и минутной стрелок (аналоговые часы). При разработке использовать фреймворк WPF. Должна быть использована только векторная графика (не использовать растровый фон). Компонент должен поддерживать масштабирование.