

ФЕДЕРАЛЬНОЕ АГЕНТСТВО ПО ОБРАЗОВАНИЮ
ГОСУДАРСТВЕННОЕ ОБРАЗОВАТЕЛЬНОЕ
УЧРЕЖДЕНИЕ
ВЫСШЕГО ПРОФЕССИОНАЛЬНОГО ОБРАЗОВАНИЯ
«ВОРОНЕЖСКИЙ ГОСУДАРСТВЕННЫЙ
УНИВЕРСИТЕТ»

**РАЗРАБОТКА КОМПОНЕНТОВ
И ЭЛЕМЕНТОВ УПРАВЛЕНИЯ WINFORMS
ДЛЯ ПЛАТФОРМЫ .NET FRAMEWORK**

Учебно-методическое пособие для вузов

Составители:
Д. И. Соломатин,
А. В. Копытин

Издательско-полиграфический центр
Воронежского государственного университета
2010

СОДЕРЖАНИЕ

Содержание.....	3
1. Компонентно-ориентированный подход к разработке ПО	4
2. Работа с компонентами в среде Visual Studio	5
3. Иерархия классов компонентов в .NET FCL	6
4. Варианты разработки элементов управления	12
4.1. Пример элемента управления HelloWorldControl	13
5. Организация проекта в Visual Studio	14
5.1. Отладка компонентов в режиме дизайна	17
6. Разработка пользовательских элементов управления.....	18
6.1. Свойства.....	22
6.2. События.....	24
6.3. Атрибуты	25
6.4. Пиктограмма компонента	28
7. Разработка специализированных элементов управления в виде наследников от Control с прорисовкой внешнего вида.....	30
7.1. Отрисовка элемента управления	30
7.2. Борьба с мерцанием элементов управления при перерисовке	35
7.3. Управление размерами элемента управления.....	36
7.4. Описание событий с дополнительными параметрами.....	38
7.5. Отслеживание действий пользователей	40
8. Разработка специализированных элементов управления в виде наследников от существующих элементов управления	42
9. Разработка компонентов с коллекциями элементов	46
10. Расширенная поддержка времени разработки	48
10.1. Конверторы типов.....	49
10.2. Дизайнеры типов.....	51
10.3. Дизайнеры компонентов	51
Заключение	51
Практические задания.....	51
Список литературы	55

Формы (а также пользовательские элементы управления и т. п.), созданные в Visual Studio, состоят из 2 исходных *.cs-файлов, например Form1.cs и Form1.Designer.cs. При этом класс формы Form1 описывается одновременно в этих двух файлах в виде разделяемого класса (partial class).

В файле Form1.Designer.cs объявляются переменные, которые указывают на дочерние компоненты формы. Также в данном файле дизайнером форм создается метод InitializeComponent, в который помещается код на языке C# (для C#-проектов) инициализации формы, создания дочерних компонентов и их инициализации (установке свойств и т. п.). При выполнении приложения метод InitializeComponent, собственно, и создает форму в том виде, в котором она была сохранена при редактировании.

Процесс перевода отредактированной формы в код, необходимый для создания формы, будем называть *сериализацией формы в код (создания формы)*. При разработке новых компонентов можно определенным образом влиять на то, как какие-то свойства компонента будут сериализоваться в код, речь об этом пойдет ниже.

Файлы *.Designer.cs не предназначены для ручного редактирования, т. к., если допустить ошибку, Visual Studio не сможет восстановить форму из кода.

Во втором файле Form1.cs содержится конструктор формы, в котором обязательно содержится вызов метода InitializeComponent:

```
public Form1() {
    InitializeComponent();
}
```

Также в файле Form1.cs описывается код обработчиков событий компонентов и элементов управления и другая логика, связанная с поведением формы.

Такое разделение кода формы (пользовательского элемента управления и т. п.) позволяет не засорять основной файл (Form1.cs), в котором описывается логика поведения формы, объемным кодом инициализации самой формы, описания, создания и инициализации дочерних элементов формы и т. п.

3. ИЕРАРХИЯ КЛАССОВ КОМПОНЕНТОВ В .NET FCL

В .NET FCL компоненты и классы образуют следующую иерархию классов:

```
System.Object
  System.MarshalByRefObject
    System.ComponentModel.Component
      System.IO.FileSystemWatcher
      System.Windows.Forms.CommonDialog
```

```

System.Windows.Forms.ColorDialog
...
System.Windows.Forms.Control
System.Windows.Forms.ButtonBase
System.Windows.Forms.Button
System.Windows.Forms.CheckBox
System.Windows.Forms.RadioButton
System.Windows.Forms.Label
System.Windows.Forms.ListControl
System.Windows.Forms.ComboBox
System.Windows.Forms.ListBox
System.Windows.Forms.PictureBox
System.Windows.Forms.ScrollableControl
System.Windows.Forms.ContainerControl
System.Windows.Forms.Form
System.Windows.Forms.PropertyGrid
System.Windows.Forms.UserControl
...
System.Windows.Forms.Panel
...
System.Windows.Forms.ScrollBar
...
System.Windows.Forms.ImageList
System.Windows.Forms.Menu
System.Windows.Forms.Timer
System.Windows.Forms.ToolBarButton
...

```

В данной иерархии, естественно, представлены не все компоненты и элементы управления .NET FCL. Жирным шрифтом выделены базовые классы, на которых строится компонентная модель .NET Framework, остальные классы всего лишь реализуют специфическую функциональность конкретных элементов.

System.Object

Базовый класс для всех классов в .NET Framework.

System.MarshalByRefObject

Реализует возможность взаимодействия с объектом из другого домена приложения, чаще всего используется в технологии .NET Remoting.

System.ComponentModel.Component

Базовый класс для описания компонентов. Класс Component является частью компонентной модели .NET Framework, при этом среда Visual Studio позволяет размещать компоненты, например на формы приложения, зада-

вать свойства компонентов в инспекторе свойств, разрабатывать специальные редакторы компонентов и т. п.

В таблице ниже приведены некоторые свойства Component (подробнее – см. документацию):

Container	Возвращает контейнер IContainer, содержащий компонент Component
DesignMode	Указывает, находится ли компонент в режиме разработки (в противном случае – в режиме выполнения)
Events	Список обработчиков событий, привязанных к компоненту
Site	Получает или задает экземпляр ISite для компонента

System.Windows.Forms.Control

Базовый класс для описания элементов управления.

В таблице ниже приведены некоторые свойства Control (подробнее – см. документацию):

Anchor	Позволяет привязать размеры элемента управления с размера контейнера, содержащего этот элемент
BackColor	Цвет фона элемента управления
BackgroundImage	Фоновое изображение элемента управления
Left, Top	Позиция элемента управления относительно контейнера
Width, Height	Размеры элемента управления
Location	Left + Top
Size	Width + Height
Bounds	Location + Size
CanFocus	Определяет возможность элемента управления получить фокус ввода (элемент управления, имеющий фокус ввода, получает события от клавиатуры)
Capture	Определяет, была ли мышь «захвачена» данным элементом управления (т. е. события мыши обрабатываются данным элементом управления)
ClientRectangle	Прямоугольник клиентской части элемента управления, т. е. той области, за перерисовку которой отвечает элемент управления самостоятельно (в клиентскую часть компонента не входит, например, рамка компонента, заголовок окна для окон и т. п.)
ClientSize	Размеры клиентской области