Разработка приложений Windows Forms в среде .NET

Что такое платформа .NET?

- *.NET Framework – эта среда, представляющая собой дополнительный операционный слой, разделяющий приложения пользователя и базовые сервисы Windows (Win API)*
- *.NET Framework – это платформа для разработки и исполнения прикладных программ*
- Заметим, что термин “*платформа*” мы обычно применяем в двух разных смыслах. С одной стороны, это “концепция”, с другой – набор вполне конкретных объектов (файлов, документации и пр.). Эта двойственность в полной мере относится к .NET Framework

Структура .NET Framework…

Структура .NET Framework

- Как видно из рисунка, .NET Framework состоит из *двух главных компонентов*: ***библиотеки базовых классов*** (***BCL***) и ***Common Language Runtime*** (***CLR***) – среда исполнения NET-приложений), которые соответственно предназначены для решения следующих задач:
  - унификации библиотек функций для всех приложений, независимо от используемого языка программирования;
  - повышения управляемости и безопасности кода
- В этой среде ведется разработка и исполнение программ. Главным инструментом создания приложений является Visual Studio .NET. Для среды .NET корпорация Microsoft разработала четыре языка программирования: Visual C++ .NET, Visual Basic .NET, JScript .NET и Visual C#

Библиотека базовых классов…

- .NET Framework Base Class Library – библиотека базовых классов, на основе которых строятся все .NET-приложения
- Ранее подобный набор создавался для каждого языка программирования, теперь он — один для всех средств
- Такая унификация системы разработки нивелирует функциональные возможности разных языков, поэтому выбор инструмента в значительной степени зависит от пристрастия к тому или иному синтаксису
- .NET Framework Base Class Library – динамические библиотеки классов, являющиеся компонентом .NET Framework, а не пользовательского приложения!

Библиотека базовых классов…

- Классы библиотеки BCL разделены на ***пространства имен*** (***namespace***) – логическая группа типов, классов и других пространств имен
- Практически во всех программах .NET используется пространство имен **`System`**
  - Включает класс **`Object`**, от которого наследуются все остальные классы .NET
  - Включает классы для решения таких базовых задач как, как *сборка мусора* (*garbage collection*), *обработка исключений* (*exception handling*), *консольный ввод/вывод* (*console I/O*) и другие вспомогательные классы (математические классы, преобразование данных, работа с временем и датой)
- Рассмотрим основные пространства имен библиотеки BCL

Библиотека базовых классов…

- *Пространство имен* **`System.Collections`**
  - Классы для управления коллекциями объектов
- *Класс* **`System.Console`**
  - Используется для обмена данными с консолью
- *Пространство имен* **`System.Reflection`**
  - Одна из самых сильных концепций .NET – *отражение* (*reflection*), которое позволяет динамически обнаруживать информацию о типах и даже создавать, сохранять и выполнять код во время работы приложения
- *Класс* **`System.GC`**
  - Контроль за сборкой мусора. Сборщик мусора удаляет те объекты, на которые не осталось ни одной корректной ссылки

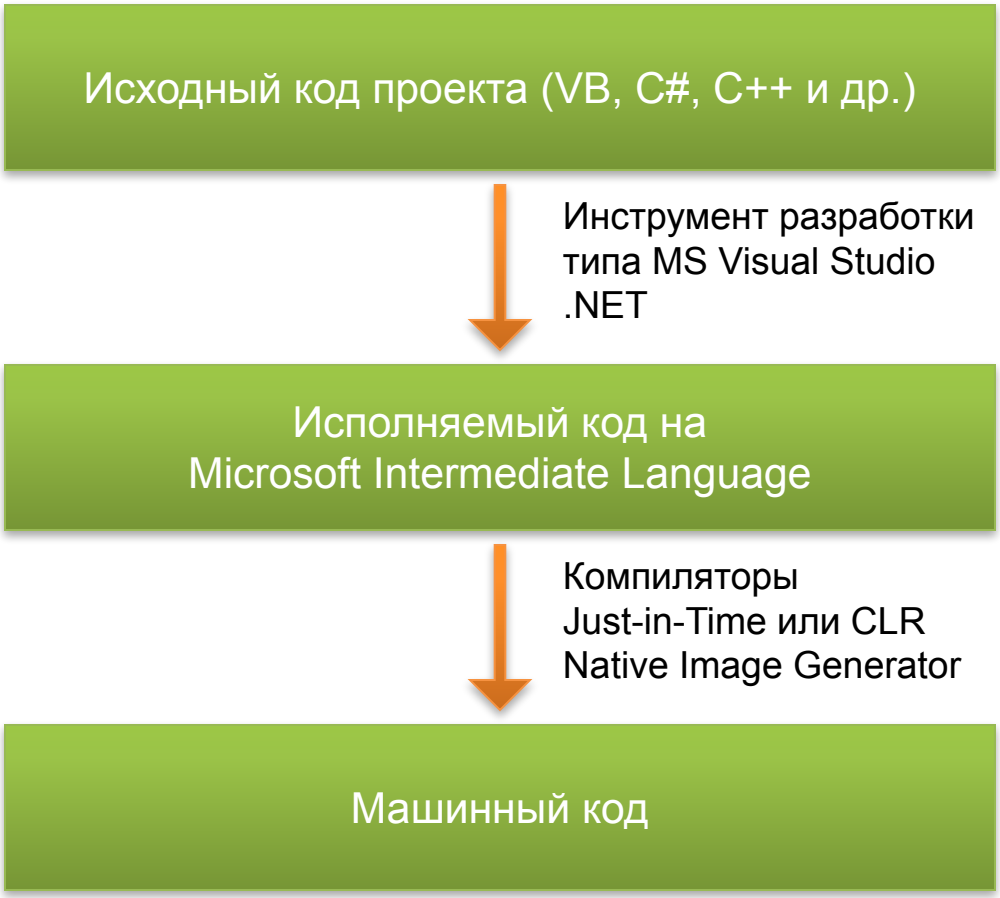
Библиотека базовых классов

- *Пространство имен* **`System.Net`**
  - Обмен данными по сети
- *Пространство имен* **`System.IO`**
  - Запись и чтение из потоков данных
- *Пространство имен* **`System.Windows.Forms`**
  - Большое количество классов для разработки графических приложений. Классы для создания форм, диалогов, собственных компонент и управления ими

Common Language Runtime...

- Среда исполнения .NET-программ CLR – это главный камень в фундаменте организации вычислительных процессов всей концепции .NET. Здесь решаются задачи повышения надежности и безопасности программ, а также платформенной независимости
- Фактически CLR исполняет программы, написанные только на *одном* стандартном языке ***Microsoft Intermediate Language*** (***MSIL***), который соответствует спецификациям ***Common Language Specification*** (***CLS***)
- Соответственно задача всех средств разработки .NET-приложений заключается в формировании результирующего исполняемого модуля на MSIL, но только реализованного уже в виде двоичного байт-кода

Common Language Runtime...



Common Language Runtime

- В отличие от классической схемы интерпретатора, используемой в том числе и в Java, CLR выполняет байт-код путем *предварительной компиляции* в машинный код отдельных фрагментов программы или приложения целиком
- Применяется Just-In-Time – компилятор, выполняющий преобразование MSIL в машинный код по мере обращения к процедурам (неиспользуемые фрагменты программы вовсе не компилируются!). Два главных преимущества по сравнению с машинным кодом:
  - Повышается безопасность программ
  - Просто решается вопрос адаптации программ к конкретной аппаратной платформе

Коротко о Visual C#

- Язык был создан Anders Hejlsberg (отец Delphi)
- С историей создания языка можно ознакомиться по адресу:

  http://www.levenez.com/lang/history.html
- Язык C# вобрал в себя опыт трех основных языков:
  - C++
  - Delphi
  - Java
- Язык C# разрабатывался как наиболее подходящий для разработки Windows-приложений на базе платформы .NET Framework

Варианты от сторонних разработчиков

- Mono [http://www.mono-project.com/Main_Page]
  - Открытая реализация платформы .NET
  - Поддерживается Windows, Linux, Mac OS X, Solaris, Unix
  - Спонсируется корпорацией Novell
- Sharp Develop [http://www.icsharpcode.net/OpenSource/SD]
  - Открытая IDE для разработки приложений на языке C#
  - Позволяет разрабатывать для .NET SDK и Mono
  - Целиком написана на C#

Введение в Windows Forms

Понятие Windows Forms

Windows Forms - технология для платформы .NET Framework в форме набора библиотек, упрощающих выполнение типичных задач приложений (чтение и запись в файловую систему и т.п.).

Возможности приложений Windows Forms в среде разработки Microsoft Visual Studio .NET:

- вывод информации;
- ввод данных пользователем;
- обмен информацией с удаленными компьютерами через сетевое соединение

Библиотека Windows Forms

- Библиотека Windows Forms представляет собой платформу для разработки графических приложений на базе .NET Framework
- Данная библиотека содержит набор простых в использовании и расширяемых классов, позволяющих создавать приложения с развитым интерфейсом
- Ключевым понятием является ***форма*** (***form***). *Форма – это область экрана, обычно прямоугольная, посредством которой программа предоставляет пользователю информацию и получает от него необходимые входные данные*

Понятие формы…

- Существуют различные виды форм:
  - *Стандартные окна (Standard windows);*
  - *Окна с многодокументальным интерфейсом (MDI-windows);*
  - *Диалоговые окна (Dialog boxes);*
  - *Поверхности для рисования (Display surfaces)*
- Самый простой способ создания интерфейса – поместить ***элементы управления*** на поверхность формы. Формы предоставляют большое число ***свойств*** и ***методов***, определяющих их внешний вид и поведение, и ***событий***, определяющих реакцию на действия пользователя. *Устанавливая необходимые свойства и разрабатывая обработчики, мы создаем конкретное приложение*

Понятие формы

- Как любые другие объекты в среде .NET Framework *формы являются экземплярами классов*. Форма, которая проектируется в дизайнере среды разработки, является классом, наследованным от класса **`Form`**. При запуске приложения создается *экземпляр данного класса*. Как и любые другие объекты, формы можно наследовать друг от друга, добавляя новую функциональность или меняя поведение
- Наряду с этим формы являются также *элементами управления*, поскольку сам класс **`Form`** наследован от базового класса **`Control`**
- Для разработки форм можно использовать обычный редактор кода, однако удобнее пользоваться дизайнером

Создание простейшей формы…

- Для примера создадим простейшее графическое приложение с использованием Windows Forms. Выполните следующие шаги:
  - Запустите среду MS Visual Studio 2005
  - Выполните команду **File | New | Project…** и введите имя нового приложения (например, Hello World)
  - С панели компонентов **Toolbox** перетащите на форму элемент управления **`Button`**
  - Выделите кнопку с помощью левой кнопки мыши и в редакторе свойств **Properties** свойству **`Text`** присвойте значение **“Say Hello”**, а свойству **`Name`** значение **“button”**

Создание простейшей формы

- Выполните следующие шаги (продолжение):
  - Дважды щелкните на кнопке, что перейти к созданию кода *обработчика события*, возникающего при *щелчке на кнопке*
  - В редактор кода введите следующую команду:

```
private void ButtonClick(object sender, EventArgs e)
{
    MessageBox.Show("Hello, World!");
}
```

  - Откомпилируйте и запустите приложение с помощью команды **Debug | Start Debugging** или клавиши **F5**
- *Мы создали простейшее приложение с единственным обработчиком события!*

Элементы управления

- Итак, создание приложения с использованием Windows Forms сводится к *созданию форм*, *добавлению* на них *элементов управления* и *разработке обработчиков событий*
- ***Элементы управления*** (***controls***) – *это объекты, которые находятся внутри объектов формы*. Каждый элемент управления имеет набор свойств, методов и событий для выполнения определенных целей. Элементы управления служат для отображения сведений или ввода пользовательских данных
- Можно добавлять элементы управления *в дизайнере* или написать код для добавления элементов управления *динамически во время работы приложения*

Добавление элементов управления…

- *Чтобы нарисовать элемент управления на форме*:
  - Откройте форму с помощью двойного щелчка левой кнопкой мыши в окне **Solution Explorer**
  - В панели компонентов **Toolbox** щелкните элемент управления, который требуется добавить на форму
  - Щелкните место на форме, в котором должен располагаться верхний левый угол элемента, и перетащите указатель на место, в котором должен располагаться правый нижний угол
  - Элемент добавляется на форму в указанное место с указанными размерами

Добавление элементов управления

- *Чтобы перетащить элемент управления в форму*:
  - Откройте форму с помощью двойного щелчка левой кнопкой мыши в окне **Solution Explorer**
  - В панели компонентов **Toolbox** щелкните элемент управления и перетащите его на форму. Элемент добавляется в форму в указанное место с *размером по умолчанию*
- Чтобы добавить элемент управления с размером по умолчанию в верхний левый угол формы, щелкните его два раза в панели компонентов **Toolbox**

“Невидимые” элементы управления…

- В панели компонентов доступны *невидимые элементы управления* (или *компоненты*). Компоненты не предоставляют интерфейс пользователя и не отображаются в дизайнере
- *Компоненты добавляются аналогично элементам управления*:
  - Откройте форму с помощью двойного щелчка левой кнопкой мыши в окне **Solution Explorer**
  - В панели компонентов **Toolbox** щелкните компонент и перетащите его на форму
  - Компонент появится в области изменяемого размера внизу формы. После добавления компонент можно выделить и задать его свойства, как для любого элемента управления в форме

“Невидимые” элементы управления

- *Обычно компоненты добавляются на форму во время выполнения*. Это общий сценарий в силу того, что компоненты не имеют визуального выражения в отличие от элементов управления, имеющих интерфейс пользователя
- В следующем примере выполняется добавление компонента **Timer** во время выполнения и задание его свойства **Interval**:

```
private void ButtonClick(object sender, EventArgs e)
{
    Timer timer = new Timer();

    timer.Interval = 1000;
}
```

Обработчики событий…

- ***Обработчик события*** (***event handler***) – *это фрагмент кода, который выполняется при возникновении того или иного события (например, нажатие на кнопке, изменение текста и изменение положения бегунка)*. Каждый элемент управления имеет свой набор событий, на которые он способен реагировать
- Назначать событиям обработчики можно *в дизайнере* или же непосредственно в редакторе кода
- Сами обработчики событий прописываются в редакторе кода

Обработчики событий

- *Чтобы добавить обработчик некоторого события*:
  - Щелкните в дизайнере левой кнопкой мыши интересующий вас элемент управления или невизуальный компонент
  - Перейдите к окну **Properties** и смените режим отображения свойств на режим отображения событий (кнопка с изображением молнии)
  - Выберите интересующее вас событие и дважды щелкните по нему левой кнопкой мыши
  - В результате этих действий будет сгенерирован пустой обработчик выбранного события, и на экране появится редактор кода

Типичные элементы управления…

- *Редактирование текста* – **`TextBox`**
  - Отображает текст, введенный во время разработки, который может редактироваться пользователями во время выполнения, а также может быть изменен программными средствами
- *Отображение текста только для чтения* – **`Label`**
  - Отображает текст, недоступный для непосредственного редактирования пользователем
- *Выбор из списка* – **`ComboBox`**
  - Отображает раскрывающийся список
- *Выбор из списка* – **`NumericUpDown`**
  - Отображает список чисел, который можно прокручивать с помощью кнопок со стрелками

Типичные элементы управления…

- *Вывод графики* – **PictureBox**
  - Отображает в рамке графические файлы, например точечные рисунки или значки
- *Задание значений* – **CheckBox**
  - Отображает флажок и надпись для текста. Используется для задания параметров
- *Задание значений* – **RadioButton**
  - Выводит кнопку, которая может быть включена или выключена
- *Задание значений* – **Trackbar**
  - Позволяет задавать значения на шкале, перемещая по ней ползунок

Типичные элементы управления

- *Диалоговые окна* – **`OpenFileDialog`**
  - Диалоговое окно для поиска и выбора файла
- *Диалоговые окна* – **`SaveFileDialog`**
  - Диалоговое окно для сохранения файла
- *Элементы управления меню* – **`MainMenu`**
  - Интерфейс режима разработки для создания меню
- *Команды* – **`MainMenu`**
  - Используется для запуска, остановки или прерывания процесса
- *Группировка элементов управления* – **`Panel`** и **`GroupBox`**
  - Группирует набор элементов управления в прокручиваемую без надписи и непрокручиваемую с надписью рамку

Пример: приложение для просмотра изображений

Описание приложения

- В заключение рассмотрим пример простого приложения для просмотра изображений
- Функциональность приложения весьма примитивна:
  - Открыть графический файл с помощью стандартного диалога открытия файла
  - Прокрутить изображение с помощью стандартных полос прокрутки
  - Выйти из приложения
- Для загрузки графический файлов будет использоваться стандартный класс **`Bitmap`**, который позволяет загрузить большинство графических форматов (включая GIF и JPG)

Создание нового приложения

- Выполните следующие шаги:
  - Запустите среду MS Visual Studio 2005
  - Выполните команду **File | New | Project…** и введите имя нового приложения (например, Image Viewer)
  - Щелкните по форме приложения и в редакторе свойств **Properties** свойству **`Text`** присвойте значение **“`Image Viewer`”**, а свойству **`Name`** задайте значение **“`MainForm`”**
  - Убедитесь в работоспособности “пустого” приложения, запустив его на выполнение с помощью команды **Debug | Start Debugging** или клавиши **F5**

Создание главного меню

- Для добавления к программе главного меню выполните следующие шаги:
  - В панели компонентов щелкните левой кнопкой мыши элемент управления **`MenuStrip`** и перетащите его на форму – внизу окна дизайнера появится область с новым компонентом
  - Выберите в дизайнере левой кнопкой мыши добавленное меню и в редакторе свойств **Properties** свойству **`Name`** задайте значение **“`menuStrip`”**
  - С помощью дизайнера добавьте к меню команды **File | Open** и **File | Exit**
  - Перейдите в редактор свойств **Properties** и пунктам меню **File**, **File | Open** и **File | Exit** в качестве значений свойства **Name** задайте соответственно значения “**`menuItemFile`**”, “**`menuItemOpen`**”, “**`menuItemExit`**”

Создание обработчиков событий…

- Библиотека BCL содержит класс **`Bitmap`**, который берет на себя выполнение практически всей нудной работы по обработке графических файлов и настолько прост в использовании, на сколько это вообще возможно
- Для открытия файлов воспользуемся стандартным диалогом открытия файлов. Для этой цели необходимо создать экземпляр класса **`OpenFileDialog`** и вызвать его метод **`ShowDialog`**. Если пользователь выберет какой-либо файл, то метод **`ShowDialog`** вернет значение **`DialogResult.OK`**

Создание обработчиков событий…

- Дважды щелкните в дизайнере на пункте меню **Open** и в редакторе кода пропишите следующий обработчик:

```
private Bitmap image = null;

private void MenuItemOpenClick(object sender, EventArgs e)
{
    OpenFileDialog dialog = new OpenFileDialog();

    if (dialog.ShowDialog() == DialogResult.OK)
    {
        image = new Bitmap(dialog.FileName);

        AutoScroll = true;
        AutoScrollMinSize = image.Size;

        Invalidate();
    }
}
```

Создание обработчиков событий…

- Щелкните в дизайнере на форме, в списке событий выберите событие **Paint** и задайте для него следующий обработчик:

```
private void MainFormPaint(object sender, PaintEventArgs e)
{
    if (null != image)
    {
        e.Graphics.DrawImage(image,
                             AutoScrollPosition.X,
                             AutoScrollPosition.Y,
                             image.Width, image.Height);
    }
}
```

Создание обработчиков событий

- Наконец, для пункта меню Exit задайте следующий обработчик:

```
private void MenuItemExitClick(object sender, EventArgs e)
{
    Close();
}
```

- Откомпилируйте и запустите приложение с помощью команды **Debug | Start Debugging** или клавиши **F5**
- После успешного запуска выполните команду **File | Open** и затем выберите один из графических файлов – изображение появится в окне