Introduction to .NET Framework and Visual Studio 2013 IDE

MIT 31043, Visual Programming
By: S. Sabraz Nawaz
Senior Lecturer in MIT
Department of MIT
Faculty of Management and Commerce
South Eastern University of Sri Lanka
.NET Framework

• The .NET Framework (now at version 4.6) is a revolutionary platform created by Microsoft for developing applications.
• .NET Framework runs on the Windows and Windows Phone operating systems, it is possible to find alternative versions that will work on other systems.
• .NET Framework enables the creation of desktop applications, Windows Store applications, web applications, web services, and pretty much anything else you can think of.
The .NET Framework has been designed so that it can be used from any language, including C# as well as C++, Visual Basic, JScript, and even older languages such as COBOL. For this to work, .NET-specific versions of these languages have also appeared, and more are being released all the time.

Not only do all of these have access to the .NET Framework, but they can also communicate with each other. It is possible for C# developers to make use of code written by Visual Basic programmers, and vice versa.
What’s in the .NET Framework?

• **.NET Framework** is a software framework developed by Microsoft that runs primarily on Microsoft Windows.
• It includes a large class library known as Framework Class Library (FCL) and provides language interoperability (each language can use code written in other languages) across several programming languages.
• Programs written for .NET Framework execute in a software environment known as Common Language Runtime (CLR) which is an application virtual machine that provides services such as security, memory management, and exception handling.
• FCL and CLR together constitute .NET Framework.
What’s in the .NET Framework?

- C#
- Visual Basic
- C++
- ...

- Common Language Specification
- .NET Framework Class Library
- Common Language Runtime

Visual Studio.NET
Common Language Specification

• An important goal of the .NET Framework is to support multiple languages.

• All languages are not created equal, so it is important to agree upon a common subset that all languages will support.

• The CLS is an agreement among language designers and class library designers about those features and usage conventions that can be relied upon.
The purpose of CLI is to provide a language-neutral platform for application development and execution, including functions for exception handling, garbage collection, security, and interoperability.

By implementing the core aspects of .NET Framework within the scope of CLI, this functionality will not be tied to a single language but will be available across the many languages supported by the framework. Microsoft's implementation of CLI is CLR.
Common Language Runtime

• The Common Language Runtime (CLR) is the virtual machine component of Microsoft's .NET framework and is responsible for managing the execution of .NET programs.

• In a process known as just-in-time compilation, the compiled code is converted into machine instructions that, in turn, are executed by the computer's CPU.

• The CLR provides additional services including memory management, type safety and exception handling.

• All programs written for the .NET framework, regardless of programming language, are executed by the CLR.

The .NET Framework class library is huge, comprising more than 2,500 classes. All this functionality is available to all the .NET languages. The library consists of four main parts:

1. Base class library (which includes networking, security, diagnostics, I/O, and other types of operating system services)
2. Data and XML classes
3. Windows UI
4. Web services and Web UI
Compilation in .NET

- Code in VB.NET: VB.NET compiler
- Code in C#: C# compiler
- Code in another .NET Language: Appropriate Compiler

Intermediate Language (IL) code

Just-in-time (JIT) execution by CLR
Intermediate Language (IL)

• .NET languages are not compiled to machine code. They are compiled to an Intermediate Language (IL).

• CLR accepts the IL code and recompiles it to machine code. The recompilation is just-in-time (JIT) meaning it is done as soon as a function or subroutine is called.
.NET Application Creation Process
Overview of .NET Framework release history

<table>
<thead>
<tr>
<th>Generation</th>
<th>Version number</th>
<th>CLR version</th>
<th>Release date</th>
<th>Development tool</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.0</td>
<td>1.0.3705.0</td>
<td>1.0</td>
<td>2002-02-13</td>
<td>Visual Studio .NET</td>
</tr>
<tr>
<td>1.1</td>
<td>1.1.4322.573</td>
<td>1.1</td>
<td>2003-04-24</td>
<td>Visual Studio .NET 2003</td>
</tr>
<tr>
<td>2.0</td>
<td>2.0.50727.42</td>
<td>2.0</td>
<td>2005-11-07</td>
<td>Visual Studio 2005</td>
</tr>
<tr>
<td>3.0</td>
<td>3.0.4506.30</td>
<td>2.0</td>
<td>2006-11-06</td>
<td>Microsoft Blend</td>
</tr>
<tr>
<td>3.5</td>
<td>3.5.21022.8</td>
<td>2.0</td>
<td>2007-11-19</td>
<td>Visual Studio 2008</td>
</tr>
<tr>
<td>4.0</td>
<td>4.0.30319.1</td>
<td>4</td>
<td>2010-04-12</td>
<td>Visual Studio 2010</td>
</tr>
<tr>
<td>4.5</td>
<td>4.5.50709.17929</td>
<td>4</td>
<td>2012-08-15</td>
<td>Visual Studio 2012</td>
</tr>
<tr>
<td>4.5.1</td>
<td>4.5.50938.18408</td>
<td>4</td>
<td>2013-10-17</td>
<td>Visual Studio 2013</td>
</tr>
<tr>
<td>4.5.2</td>
<td>4.5.51641</td>
<td>4</td>
<td>2014-05-05</td>
<td>Visual Studio 2013</td>
</tr>
</tbody>
</table>

Now Visual Studio 2015! ➔ 4.6
Microsoft Visual Studio

• **Microsoft Visual Studio** is an integrated development environment (IDE) from Microsoft. It is used to develop computer programs for Microsoft Windows, as well as web sites, web applications and web services.

• To read more
http://en.wikipedia.org/wiki/Microsoft_Visual_Studio
Integrated Development Environment (IDE)
Solution Explorer

• The Solution Explorer window contains a list of the items in the current solution.

• A solution can contain multiple projects, and each project can contain multiple items.

• The Solution Explorer displays a hierarchical list of all the components, organized by project.
Form

- A form is a basic building block of a Visual C# project. Eventually, you'll be creating projects that consist of many forms.
- Each form is really a class. Usually, a class consists of many methods. Some methods are related to user events such as click and keypress.
- The first default form of a VC project is often saved with the file name of Form1.cs. The class that goes along with this form is also named Form1.
Properties Window

- This window displays all the properties of the selected component and its settings.
- Every time you place a control on a form, you switch to this window to adjust the appearance of the control.
To create a new project:

• Choose **File > New > Project** from the menu. This displays the New Project dialog box.

• In the **New Project** dialogue box, click on **Visual C#** under project types list and click on **Windows Forms Application** under **Templates** list.

• Type **SampleApp** in the **Name** box.

• Accept the default location or select the one that you like.

• Click OK, and now you will see the Integrated Development Environment (IDE) with a blank form.
To create a new project:
Placing objects on a form

- Textboxes, labels, and buttons are three very useful kinds of objects to use in a VC program.
- There are several ways to place objects on a form.
  - You can double-click a Toolbox icon in order to place that object on the form.
  - You can drag a Toolbox icon onto a form to place an object there.
  - You can single-click the Toolbox icon to select it and then click and drag on the form itself to place and size an object.
Exercise

- Add the 03 Labels, 03 textBoxes, and 04 Buttons.
- Set their properties as follows.

<table>
<thead>
<tr>
<th>Control</th>
<th>Property</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Form1</td>
<td>Name</td>
<td>calculatorForm</td>
</tr>
<tr>
<td></td>
<td>Text</td>
<td>Simple Calculator</td>
</tr>
<tr>
<td></td>
<td>MaximizeBox</td>
<td>False</td>
</tr>
<tr>
<td>Label1</td>
<td>Text</td>
<td>First Number</td>
</tr>
<tr>
<td>Label2</td>
<td>Text</td>
<td>Second Number</td>
</tr>
<tr>
<td>Label3</td>
<td>Text</td>
<td>Result</td>
</tr>
</tbody>
</table>
## Exercise

<table>
<thead>
<tr>
<th>Control</th>
<th>Property</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Text1</td>
<td>Name</td>
<td>firstNumberTextbox</td>
</tr>
<tr>
<td>Text2</td>
<td>Name</td>
<td>secondNumberTextbox</td>
</tr>
<tr>
<td>Text3</td>
<td>Name</td>
<td>resultTextbox</td>
</tr>
<tr>
<td>Button1</td>
<td>Name</td>
<td>addButton</td>
</tr>
<tr>
<td></td>
<td>Text</td>
<td>Add</td>
</tr>
<tr>
<td>Button2</td>
<td>Name</td>
<td>subtractButton</td>
</tr>
<tr>
<td></td>
<td>Text</td>
<td>Subtract</td>
</tr>
<tr>
<td>Button3</td>
<td>Name</td>
<td>clearButton</td>
</tr>
<tr>
<td></td>
<td>Text</td>
<td>Clear</td>
</tr>
<tr>
<td>Button4</td>
<td>Name</td>
<td>exitButton</td>
</tr>
<tr>
<td></td>
<td>Text</td>
<td>Exit</td>
</tr>
</tbody>
</table>
Form at runtime

![Simple Calculator](image)

- First Number: 1500
- Second Number: 1525
- Results: 3025
Double click on `addButton` and type the code

```csharp
private void addButton_Click(object sender, EventArgs e)
{
    int firstNumber, secondNumber, results;
    firstNumber = int.Parse(firstNumberTextBox.Text);
    secondNumber = int.Parse(secondNumberTextBox.Text);
    results = firstNumber + secondNumber;
    resultsTextBox.Text = results.ToString();
}
```
Double click on `subtractButton` and type the code

```csharp
private void subtractButton_Click(object sender, EventArgs e)
{
    int firstNumber, secondNumber, results;
    firstNumber = int.Parse(firstNumberTextBox.Text);
    secondNumber = int.Parse(secondNumberTextBox.Text);
    results = firstNumber - secondNumber;
    resultsTextBox.Text = results.ToString();
}
```
Double click on **clearButton** and type the code

```
private void clearButton_Click(object sender, EventArgs e)
{
    firstNumberTextBox.Text = "";
    secondNumberTextBox.Text = "";
    resultsTextBox.Text = "";
    firstNumberTextBox.Focus();
}
```

Double click on **exitButton** and type the code

```
private void exitButton_Click(object sender, EventArgs e)
{
    this.Close();
}
```
Run it!!!

- Press **F5** on the keyboard to run the program and check its operations.
- Enter a value in the first textbox
- Enter a value in the second textbox
- If you click Add button, both values will be added and result will be show in the third textbox. If you click Subtract button, second value will be deducted from the first value and the result will be displayed.
- If you click on Clear button, all texts in the textboxes will be erased and the cursor will be blinking in the first box.
- If you click on Exit button, the program will be terminated.