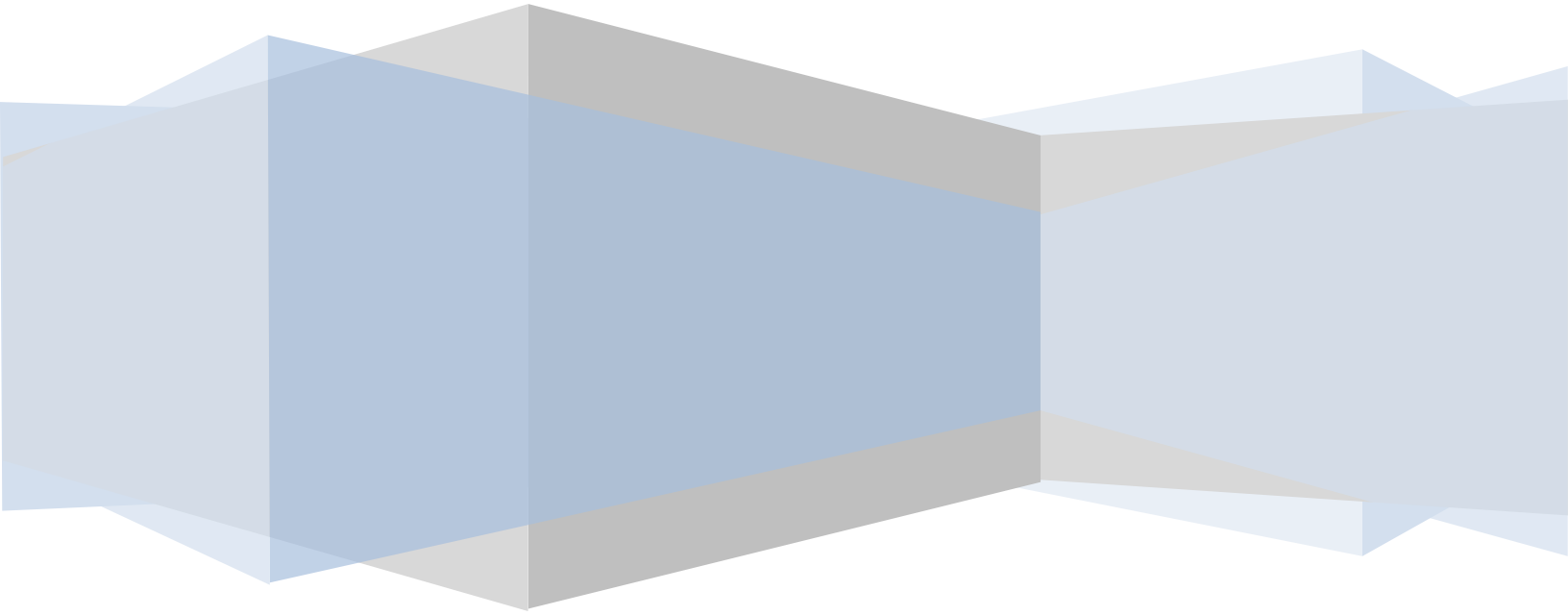
Visual Basic 2010 Essentials



Visual Basic 2010 Essentials – First Edition

Table of Contents

Chapter 1. About Visual Basic Essentials

Visual Basic Essentials is intended to be of use to both novices looking to learn Visual Basic, and to those proficient in other languages that plan to cross-train. The Visual Basic language, combined with the Visual Studio, provides a uniquely powerful, yet easy to learn development environment allowing even the absolute beginner to rapidly create and deploy Windows applications.

Visual Basic Essentials begins with instruction on designing Windows forms in Visual Studio including tasks such as designing menu systems and toolbars and wiring up event procedures. The book then introduces the basic concepts of the Visual Basic language covering concepts such as Visual Basic variable types, looping, flow control, functions and subroutines. Once the basics are covered, topics such as single and mutli-dimensional arrays, string handling, file I/O and date and time manipulation are explained. Finally, more advanced topics such as Visual Basic object oriented programming, database access and graphics drawing are detailed.

Throughout the book, liberal use is made of code excerpts providing practical examples of theory in action. It is intended that having read this online book, the programmer will be confidently developing Windows applications using Visual Basic and Visual Studio.

Chapter 2. Downloading and Installing Visual Studio

Everything you need to build Visual Basic applications is included with Microsoft Visual Studio. At the time of publication of this book the latest release of Visual Studio is Visual Studio 2010. The first step in learning Visual Basic is, therefore, to obtain a copy of Visual Studio.

2.1 Getting Visual Studio

There are a number of options in terms of getting a copy of Visual Studio. If you work for a company which has Visual Studio licenses available then the first step is to talk to your IT department to see if they can provide you with a license, or purchase one for you.

If you do not have access to a purchased Visual Studio license, or lack the funds to buy a copy, the best option, and actually the ideal option for those learning Visual Basic, is to download a trial version from Microsoft's web site. This will give you a 30 day trial period to use all the features of Visual Studio with the option to extend the trial to 90 days. Given that this book will teach you everything you need to know to program in Visual Basic the 90 day trial period will give you plenty of time to learn Visual Basic with time to spare to decide if you want purchase a fully licensed copy of Visual Studio.

In this chapter of Visual Basic Essentials we will assume that you are downloading and installing the trial version of Visual Studio 2010.

2.2 Downloading a Visual Studio Trial

To work through the examples in this book, Visual Studio 2010 Professional Edition is recommended.

Trial copies of Microsoft Visual Studio Professional Edition are available from the Visual Studio page of the Microsoft web site. From this page, you can choose to download a trial copy of Visual Studio 2010 using the *web installer* or as an *ISO image*.

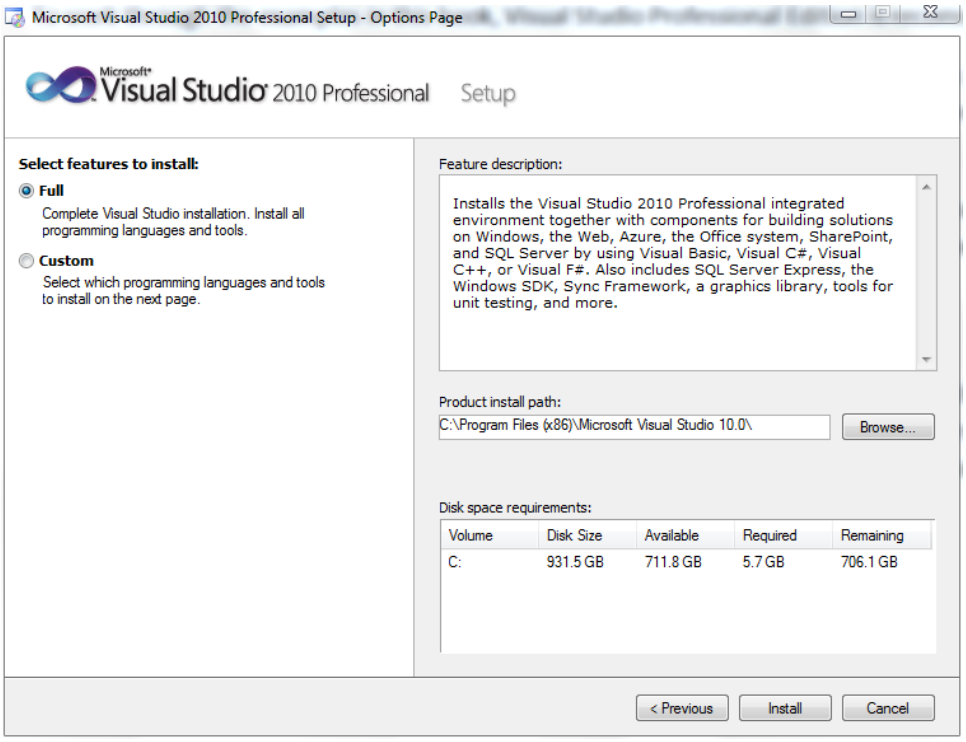
The ISO image is approximately 2.5GB in size. Whilst the web installer itself will download relatively quickly, the installer will subsequently download the remainder of the Visual Studio files and packages so a high speed internet connection is highly recommended regardless of the installation method chosen. That said, the web installer provides the ability to download only the features needed to develop Visual Basic applications, so will download and install faster than the ISO image.

The ISO image will either need to be burned to a DVD after it has downloaded, or be mounted as a virtual CD using a tool such as Virtual Clone Drive, or the ISO image may be unpacked using WinZip.

In the remainder of this chapter it will be assumed that the installation is being performed using the web installer.

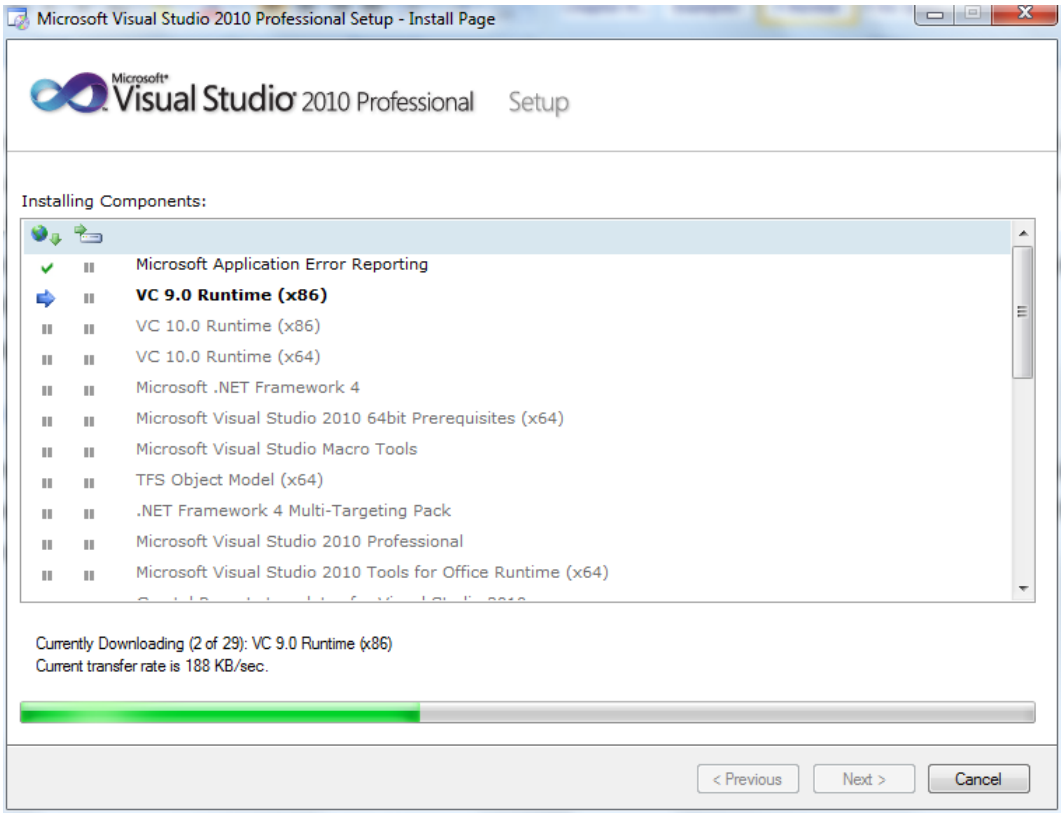
2.3 Installing Visual Studio

Once the web installer has downloaded, navigate to the location it has been saved to and double click on the executable to launch it. Once a number of files have been installed and the license terms accepted, a screen will appear providing the option to select the features to be installed:



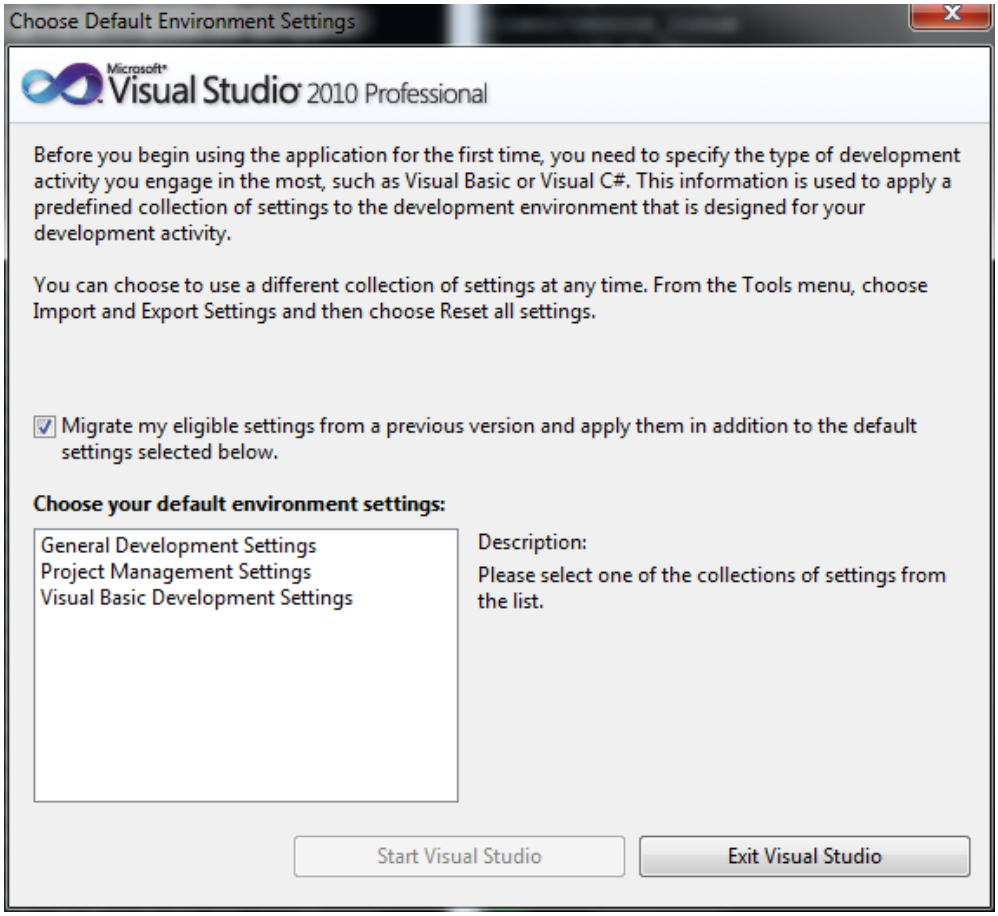
Visual Studio supports a number of different programming languages in addition to Visual Basic (such as C#, F# and C++). If you only plan to use Visual Basic then choose the custom installation option and uncheck the boxes for the other languages. This will both speed up the installation process and reduce the amount of space required to install Visual Studio. If you need to work with the other supported programming languages at a later date you can install additional language support.

Once the desired features have been selected, the installer will download and install the corresponding packages. The download and installation process is reported in real-time as illustrated in the following figure:



2.4 Starting Visual Studio for the First Time

Once the Visual Studio installation is complete you are ready to start it up. Find Visual Studio in the Windows Start menu and select it. The first time Visual Studio is run it will ask you which programming language you would like to use as the default:



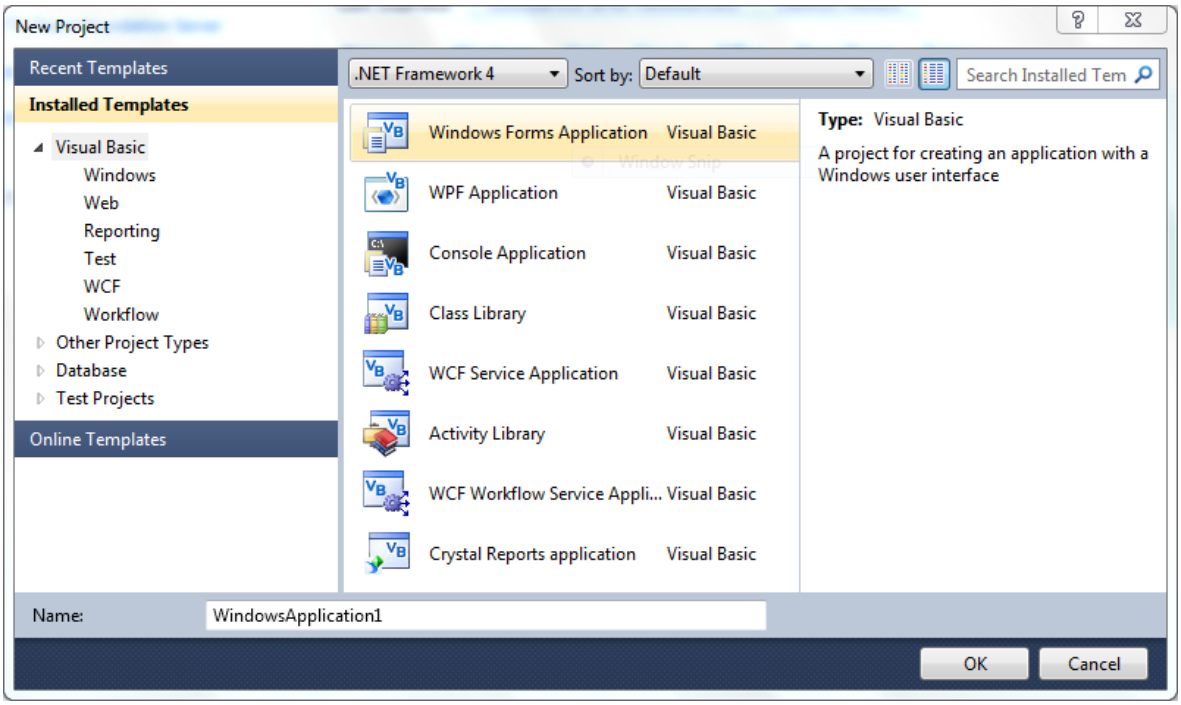
Select *Visual Basic Development Settings* from the list and click on the *Start Visual Studio* button. After loading some user settings, the main Visual Studio 2010 window should appear and you are now ready to start learning Visual Basic.

Chapter 3. Creating a New Visual Basic Project

Visual Studio uses the concept of *projects* to contain the files and resources required to build an application using Visual Basic. Typically there will be one Visual Studio project per individual application you develop. In this chapter we will look at creating a new Visual Studio project.

3.1 Creating a Project in Visual Studio

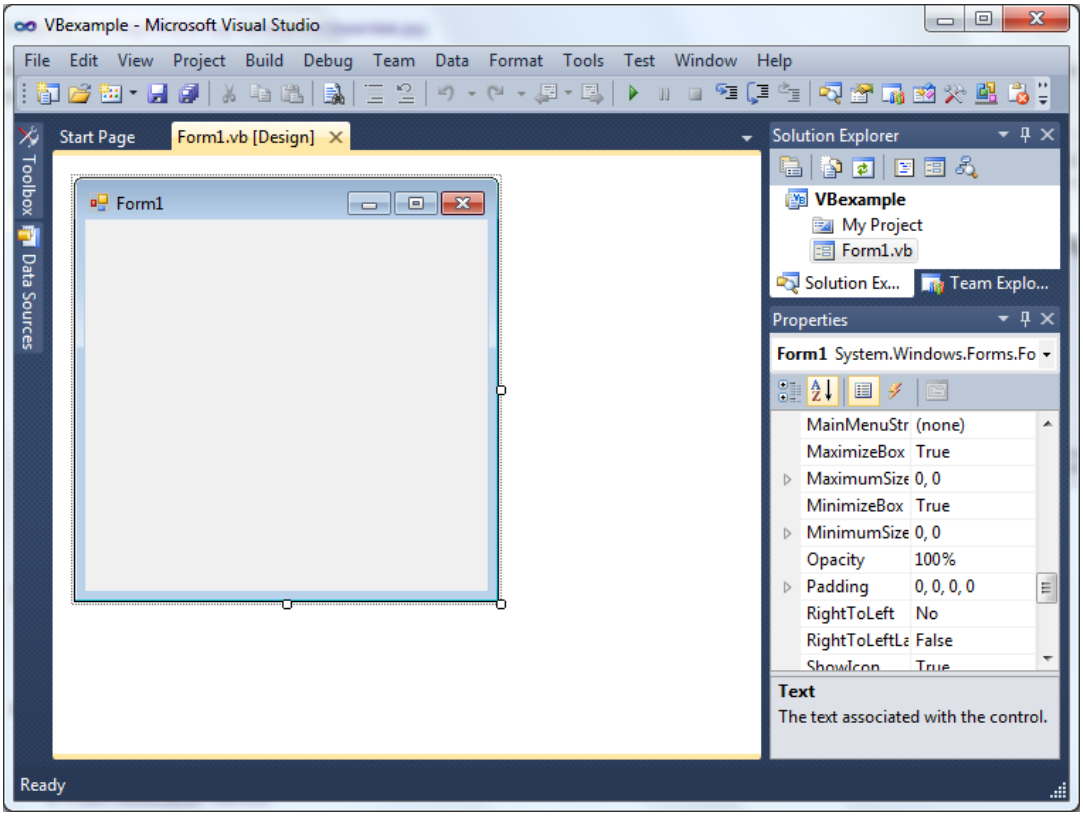
Now that Visual Studio is installed it is time to create a new project. If it is not already running, begin by starting Visual Studio from the Windows Start menu. When the main Visual Studio Window appears click on the *File* menu and select the *New Project* option. Visual Studio will subsequently display the "New Project" dialog illustrated in the following figure:



The *New Project* dialog provides a number of options with regard to the type of project being created and the version of the .NET Framework to use as the foundation for that project. In this example we are going to build a Windows Application using .NET Framework 4. A Windows Application is a graphical application (in other words an application that appears in a window and contains buttons, text fields and all the other graphical items we expect to see when we run Microsoft Windows applications).

Select the *Windows Application* icon from the New Project dialog and give your new project a name by typing it into the *Name:* field (for example you might want to name your project "VBexample"). Click on the *OK* button to create the project. Once the project is created Visual

Studio will display the new project, which at this point consists of a single form, ready for us to start adding visual components:



If the main window does not appear as illustrated above, select the *Window -> Reset Window Layout* menu option to reset the window to the default settings. Since it is easy to lose windows and tabs in Visual Studio during the learning phase this is a useful feature to keep in mind as you work through the remaining chapters of this book.

Now that we have created a new project in Visual Studio it is time to move on to the next chapter and create a simple Visual Basic application. Before doing so, however, select the *File -> Close Project* menu item and click on the *Discard* button in the resulting dialog to remove our test project from your system.

Chapter 4. A Simple Visual Basic Example

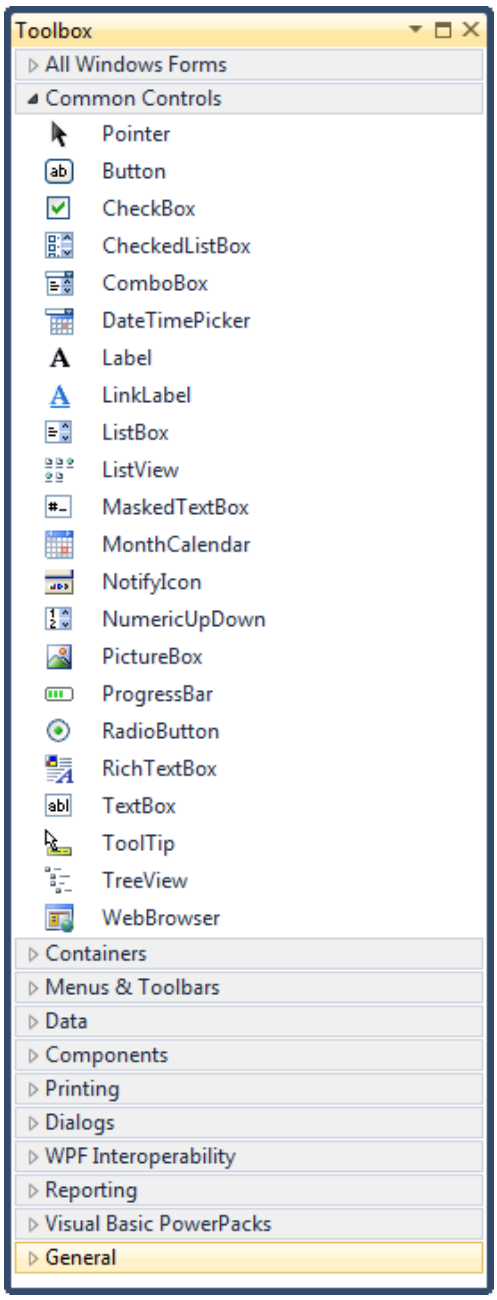
It is often helpful when learning a new programming language to start out with a very simple example. In this chapter we will create a small sample Visual Basic application. The purpose of this example is to provide an early confidence boost for those completely new to Visual Basic by showing just how easy it is to create an application and verifying that the Visual Studio 2010 environment is correctly installed and configured.

4.1 Creating a New Project

The first step is to create a new project to contain our example Visual Basic Application. Start Visual Studio and select *File -> New project*. From the new project dialog select *Windows Application* and name the project *myVBapp* and click on *Ok* to create the new project. Once the new project is created, Visual Studio will display a blank form ready for us to design the user interface of the application.

4.2 Adding Controls to the Form

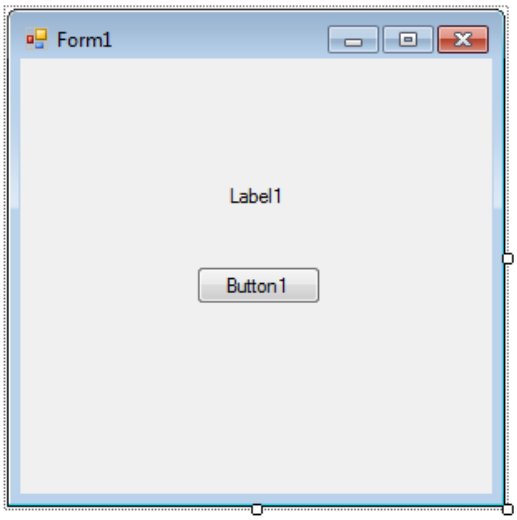
For the purposes of our example Visual Basic application we are going to add two controls to our form; a push button and a label. To achieve this we first need to access the Visual Studio Toolbox. Along the left hand side of the Visual Studio main window you should see a tab labeled *Toolbox*. Click on this tab to display the *Toolbox*. It should appear as follows:



This Toolbox contains all the controls that may be used to build a Graphical User Interface for a Windows application. The toolbox will auto-hide by default, that is it will disappear when the mouse pointer is moved away from it. To make it permanently visible click on the push pin icon at the top of the toolbox window. Once it is pinned in place, it will no longer auto-hide. It is also possible to detach the toolbox so that it will float and can be positioned anywhere on the desktop. To do so, simply click on the toolbox title area and drag it to the desired screen location. To re-dock the panel to its original location, click on the small down arrow located in the title bar and select the *Dock* option from the resulting menu.

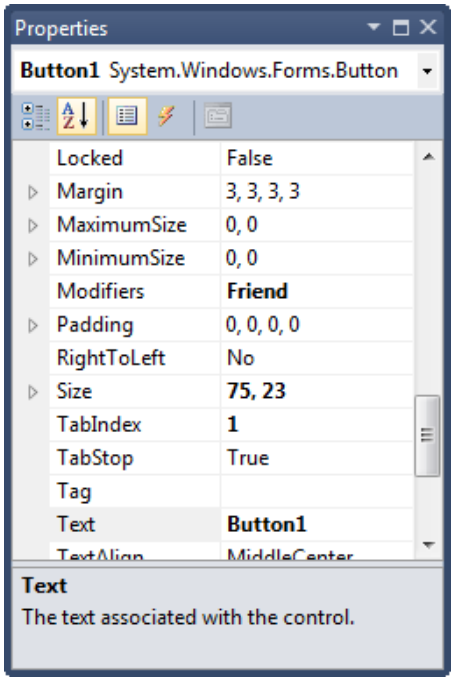
Controls are added to the Form by clicking on the required control in the Toolbox and dragging it to the desired location on the Form. To add a label to the form, click on the Label control in the Toolbox, drag it to the center of the Form and release the mouse button. The new label will then appear in the Form at the point you dropped it.

Next we need to add a button. Grab a Button from the Toolbox and drag and drop it on the Form. Use the mouse to move the controls around the Form until you have a layout you are happy with, for example:



4.3 Setting Control Properties

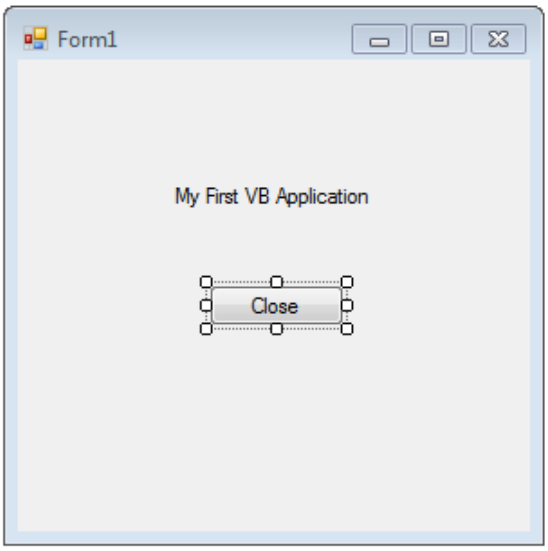
Now that we have added the controls to our Form we need to change some of the characteristics of these controls. This is done by changing the *Properties* of the controls. Properties are a group of settings that allow the appearance and behavior of controls to be changed. For example, there is a property for changing the text displayed on a label, the color of a button, the font of the text on a button and so on. Properties of a control are changed using the Visual Studio *Properties* panel which is, by default, displayed in the bottom right hand corner of the main dialog:



The property panel is *context sensitive.* In other words, the properties displayed at any one time are related to the currently selected control in the Form. If you click on the Label and then the Button in your Form you will see the properties panel change to reflect the current selection.

To begin with, we will change the text of the Label control. Select the Label control in the form and then scroll down the list of properties until you find *Text*. This is the property which defines the text to be displayed on the currently selected Label control. Change this from the current value ('Label1') to read *My First VB Application*. Notice that as soon as you change this property the Label in the Form changes to reflect the new property setting.

Select the Button control in the Form and change the *Text* Property for this control to read *Close*. Re-position the controls in the Form if necessary. You should now have a Form which looks something like the following:



4.4 Creating an Event Handler

The next step is to make the *Close* button do something when it is pressed. Before we do that, however, we need to give the button a more meaningful name. Visual Studio has given the button a default name of *Button1*. While this is fine for a small design, it will quickly become difficult to work with such names in larger applications containing many buttons. With the Button selected in the Form, scroll up to the top of the properties list and change *(Name)* from *Button1* to *closeButton*.

Having changed the name we can now add an event to the button. Double click on the *Button* in the Form to display the event code for the *closeButton* control. Visual Studio will display the following code:

```
Private Sub closeButton_Click(ByVal sender As System.Object, ByVal e As
     System.EventArgs) Handles closeButton.Click


    End Sub
```

This is a Visual Basic *subroutine* where code is placed to define what happens when a *Click* event is detected on the button (i.e. the user clicks on the button in the user interface of our application). In this example we want the application to close when the *closeButton* is pressed. To achieve this we add a single line of Visual Basic code to the *closeButton_Click()* sub-routine as follows so that the code calls the *Close()* method to exit the application:

```
Private Sub closeButton_Click(ByVal sender As System.Object, ByVal e As
```

```
                System.EventArgs) Handles closeButton.Click
        Close()
    End Sub
```

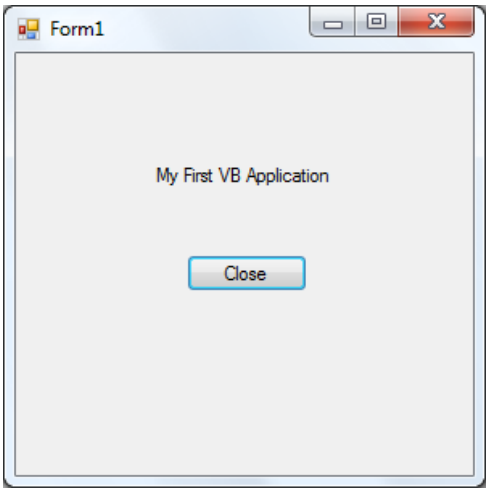
To return to the form design screen, simply click on the *Form1.vb [Design]* tab.

4.5 Building and Running a Visual Basic Application

Now that we have completed the design and implementation of a simple Visual Basic application we can compile and run it. To build the application, select *Build myVBapp* from the Visual Studio *Build* menu. Assuming there are no problems the application should compile without any errors (the message *Build succeeded* should appear in the status bar at the bottom of the Visual Studio screen).

Once built, the application can be run by selecting *Start Debugging* from the *Debug* menu. An even quicker way of building and running the application is to simply press **F5**. This will compile and run the application without having to use any menu options. As you develop Visual Basic applications in Visual Studio you will find yourself using the **F5** shortcut more than any other key on your keyboard.

After a few seconds the application should appear just as you designed it in the Visual Studio designer:



Try out the Click event on the *closeButton* control by clicking on the *Close* button to close the application. Congratulations - you have designed, built and run your first Visual Basic application.

Before proceeding, select the *File -> Close Project* menu option and select *Save* from the resulting dialog to save your project.

Chapter 5. Visual Basic and Forms

The Windows Form is a vital component in the development of any Windows-based application. Forms essentially provide the windows that make up a Windows application. In fact, the terms window and form are often used interchangeably. Forms allow the Visual Basic developer to create windows and layout controls (such as buttons, labels etc) in those forms to provide the application's user interface.

In the *Designing Forms in Visual Basic* we will look at how to layout controls inside a form. Before we reach that stage, however, there are a surprising number of ways in which the form itself can be modified and configured. We will cover these options in detail in this chapter.

5.1 Creating a New Form

Throughout this chapter we will work with a form in a new project. Begin by starting Visual Studio and creating a new Windows Application project (see *Creating a New Visual Basic Project* for details of how to do this) and name the new project *VBforms*.

Once the new project is created you will see a Form in Visual Studio ready for you to begin work.

5.2 Changing the Form Name

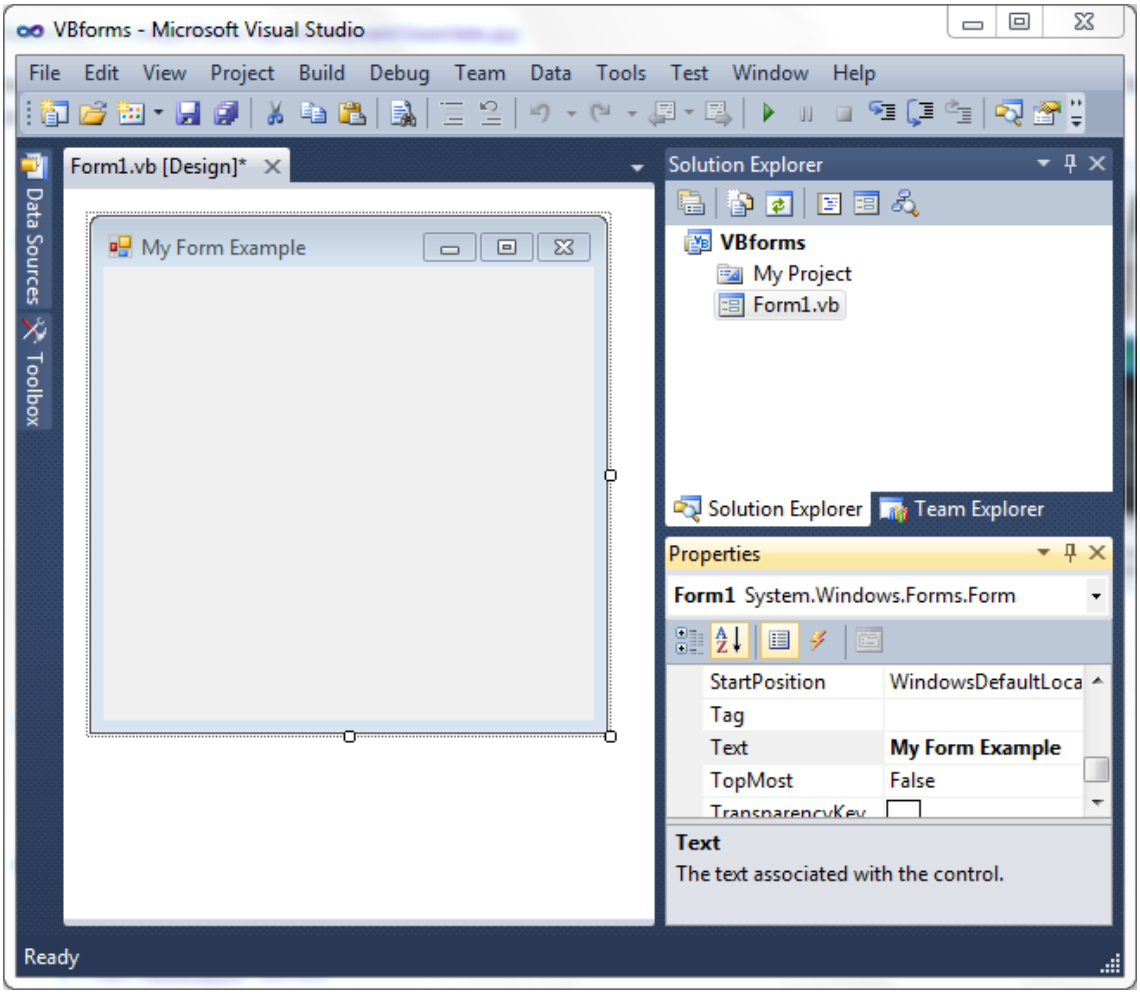
All objects in a Visual Basic application need a name so that they can be referenced in the code. When a new object is added to an application in Visual Studio it is assigned a default name which usually consists of the object type and a number. For example the first form object in an application is named *Form1*, the second *Form2*, and so on.

To change the name of a Form to something more meaningful, simply click in any area of the Form in Visual Studio and change the *(Name)* value in the *Properties* panel.

5.3 Changing the Form Title

Each form represents an application window. The text displayed in the title bar for each window should be changed to display something meaningful. This should either be the name of application, or a description of the form's function (for example *Order Entry* or *Sales Report*).

The value of the text to be displayed in the window title is defined by the form's *Text* property. To change the title of the form, therefore, select the *Text* value in the *Properties* panel and change it to a new value (for example, 'My Form Example'):



5.4 Changing the Form Background Color

The background of any form may be changed either by specifying a new color, or by using a background image. The background color is controlled by the *BackColor* property of the form. By default this property is set to *Control*. This specifies that the form should use a system defined default color. This color varies between different Windows versions, and changes when the user changes the overall theme of their Windows desktop environment.

To change the background color, select the form in Visual Studio and locate the *BackColor* property in the *Properties* panel. There are a number of ways to specify a color.

- Color by name - Simply type in a color name into the *BackColor* value field (for example *Red*, *Yellow*, *Cyan* etc).

- Color by RGB value - Colors may be specified by entering the hexadecimal RGB values (for example #FFFFFF for white, #000000 for black and so on). RGB values may also be specified using decimal values (for example 255,255,255 for white, 0,0,0 for black etc).