Chapter 23 – Next Steps in Application Building

# Objectives

* Application Settings and Resources
* The My Namespace
* Working with Internationalization
* Working with Logging
* Introduction to Security in .NET
* Reflection

This chapter is going to look at some odds and ends that you really should know about but that didn’t necessarily fit in well with the previous chapters. We will examine different ways in which we can add functionality to our programs to really take us to another level in our application building.

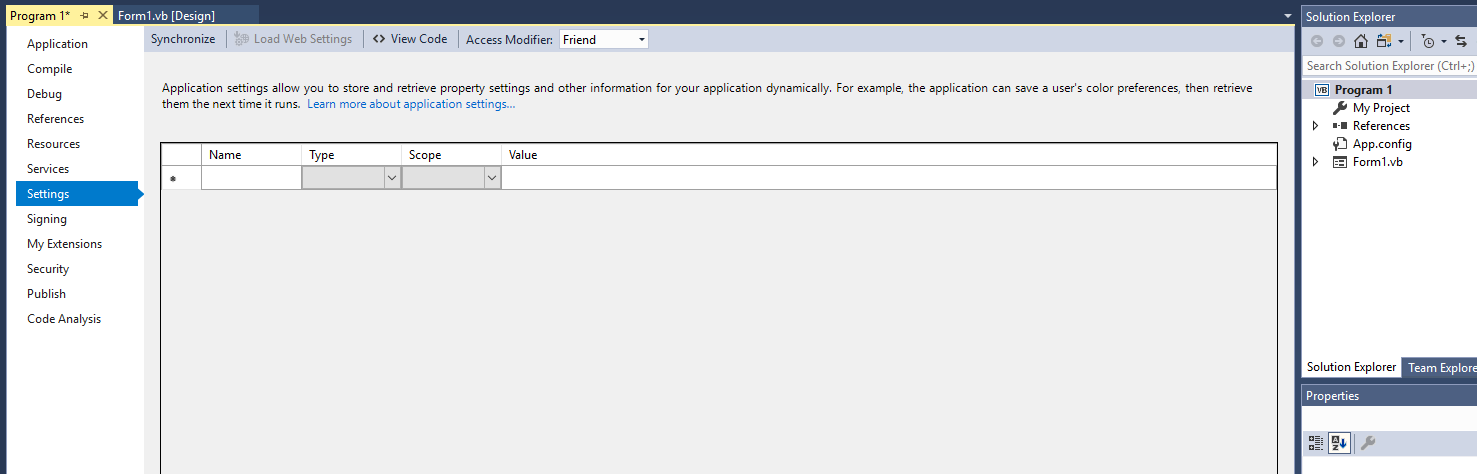
# Application Settings and Resources

Many times, we need to store some information pertinent to an application to make it work correctly. For example, when software is installed, we may provide the user with the ability to specify directories where certain components are supposed to exist, say the path to where we store certain files. Even in the projects we’ve built, there have been some items like database connection strings that we need to bundle with the application in order to make it work correctly.

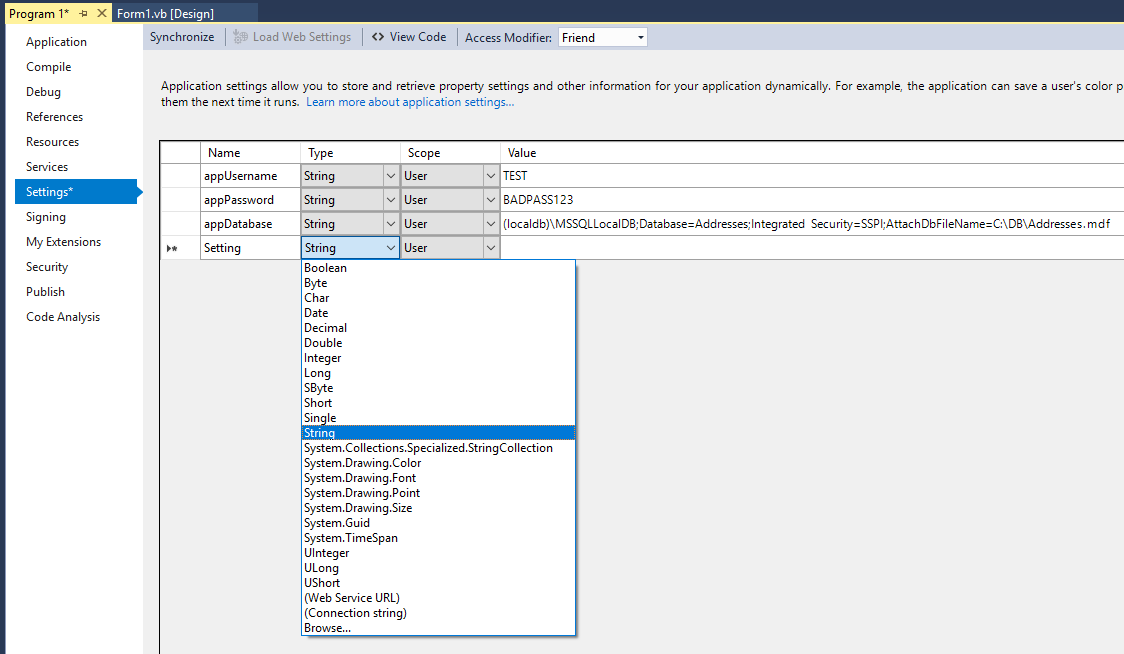
Placing these items in code doesn’t make it very easy to maintain – if there are any changes to the value, we have to modify source code and recompile. The old Windows way of doing things was to store .INI (or information files) with settings. The problem with this technique is that the .INI files were plain text and could be edited (read that as mangled) by a user. We also could store settings in the registry, but this is a tenuous business since the more we add to the registry, the more we can slow things down. Furthermore, if a registry gets corrupted, that’s bad news for the whole computer system and us.

.NET has solved much of this problem by providing a facility right within the application framework to store application settings, e.g. the settings get compiled and bundled as part of the application. Data that we want to maintain does not have to be text in nature; in fact, the strong type checking that .NET provides is available here as well. Multiple groups of settings can be maintained within an application, so per-application and per-user values can be stored. Finally, the settings are available in a simple key-value system.

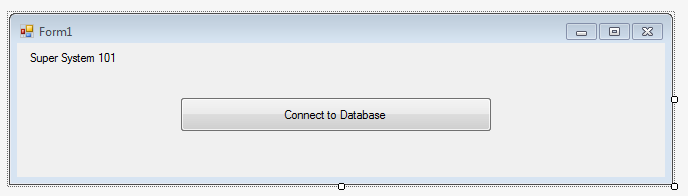
Let’s look at adding three text type settings to an application. We’ll hold a database connection string, a user name to connect to the database and a password. We’ll begin by starting a Windows Forms application. Right click on the project’s properties and you’ll see a “Settings” tab in the left hand pane:



We simply enter a name for the setting, pick a type (from the dropdown you can see there are lots of types) and then the scope (where the values are maintained for the whole application or for each user that works with the software). Now you understand how applications like Word are installed once but each user is able to customize how they work with and “see” the software. Here’s the form after entering some settings:



Let’s modify our main form so that it uses some of these values…



We have one Label (using the default name of Label1) and a Command button cmdConnect. When the application starts up, the user is not “logged” in or connected to the database until the Command button is pressed. At that point, we would start the connection process using the information that’s stored in the Settings that we created. Here’s the code to access them:

'Chapter 23 - Program 1

Public Class Form1

Private Sub cmdConnect\_Click(sender As Object, e As EventArgs)

Handles cmdConnect.Click

'The user is attempting to connect so we will need the

'connection string and credentials from the Settings

'To get to them, we simply use My.Settings and all of our settings show up!

Label1.Text = "Super System 101 -- Logged in as " & My.Settings.appUsername

'We would continue on with using their username (My.Settings.appUsername)

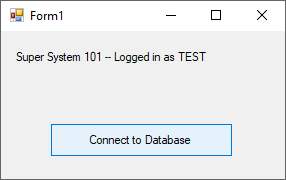
'and password (My.Settings.appPassword) to try to access

'the database that's named in My.Settings.appDatabase

End Sub

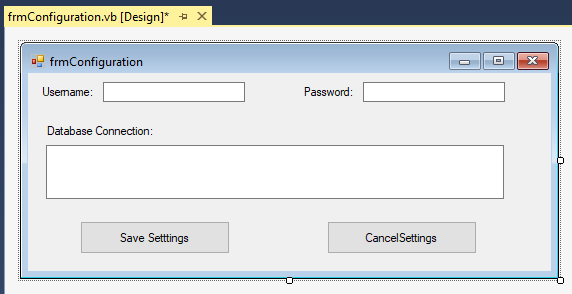
End Class

Running our application and pressing the Connect button gives us this:



Reading settings isn’t really a problem, but what if I need to change these settings after the software is deployed. Think about if we decide to change the path to the database, or if we need to use a different username/password combination? Well, I could build a form that simply has the settings placed in textboxes, allow those values to be edited and when a save button is pressed, the settings are stored back out to the application. Maybe that’s exactly what I want, but what if I also don’t want the end user being able to do that? Well let’s get a bit more sophisticated. What if we required that the sysadmin start the application up with a value on the command line? If the value is there, then we show the settings screen allowing changes to be made and then the app starts up as usual. If the command line value isn’t there, there’s no settings screen and the system starts up like usual. Sounds like a pretty good solution to me!

Let’s begin by creating a configuration screen:



Here’s the code behind the form:

'Chapter 23 - Program 2

Public Class frmConfiguration

Private Sub frmConfiguration\_Load(sender As Object, e As EventArgs) Handles Me.Load

'When the form loads, pulls the settings into textboxes so they

'can be changed if the user wants to change them

txtUsername.Text = My.Settings.appUsername

txtPassword.Text = My.Settings.appPassword

txtDatabase.Text = My.Settings.appDatabase

End Sub

Private Sub cmdSaveSettings\_Click(sender As Object, e As EventArgs)

Handles cmdSaveSettings.Click

'All we need to do is push the values from the textboxes back out

'to the settings!

My.Settings.appUsername = txtUsername.Text

My.Settings.appPassword = txtPassword.Text

My.Settings.appDatabase = txtDatabase.Text

'Although they are automatically saved when we assign

'a value to a setting, let's still call the

'save method -- we can stop automatic saving by setting

'My.Application.SaveMySettingsOnExit = False

My.Settings.Save()

'Let the user know we've saved the settings and exit

MessageBox.Show("Settings successfully saved!")

Me.Close()

End Sub

Private Sub cmdCancelSettings\_Click(sender As Object, e As EventArgs)

Handles cmdCancelSettings.Click

'The user doesn't want to save the changes, so exit

Me.Close()

End Sub

End Class

Now this configuration form will not show up when we launch the application. The main form will, which is the way we want things to behave. We only want the configuration form to show if a sysadmin is running the application to configure it. What will need to happen is for the admin to add a “–settings” value to the application’s command line when the program is run and then we will show the configuration screen.

Let’s go ahead and access the Application Events. By the way, notice that on the screen where the Application Events button is, there is a checkbox for “Save My.Settings on Shutdown” – it’s the same flag that I mentioned in the comments in our code above. Add the following code to the Startup event:

'Chapter 23 - Program 2

Imports Microsoft.VisualBasic.ApplicationServices

Namespace My

Partial Friend Class MyApplication

Private Sub MyApplication\_Startup(sender As Object, e As StartupEventArgs)

Handles Me.Startup

'Indicates we found/didn't find -SETTINGS in command line

Dim blnFoundSettings As Boolean = False

'Scan for -SETTINGS in each command line argument

For Each s As String In My.Application.CommandLineArgs

If s.ToUpper = "-SETTINGS" Then

blnFoundSettings = True

End If

Next

'If we found -SETTINGS, show configuration screen

If blnFoundSettings Then

frmConfiguration.ShowDialog()

End If

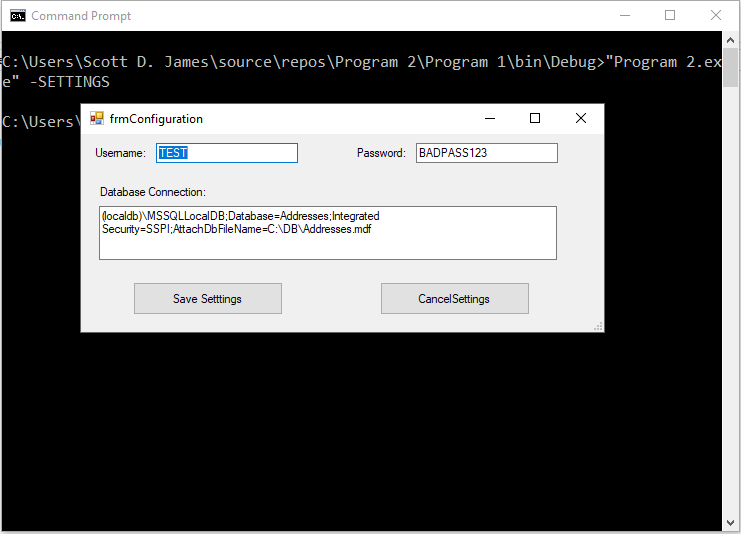
'Remember once we drop to here the main form is loaded

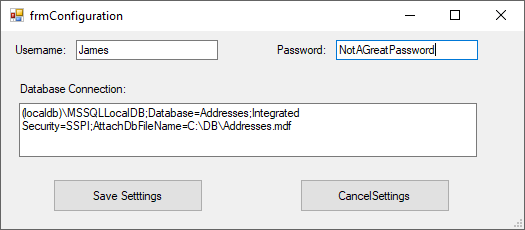
End Sub

End Class

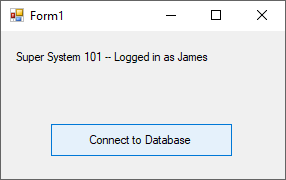
End Namespace

I am going to launch my application from a command window so I can provide an argument to the command line – I could also create a shortcut to do the same thing:



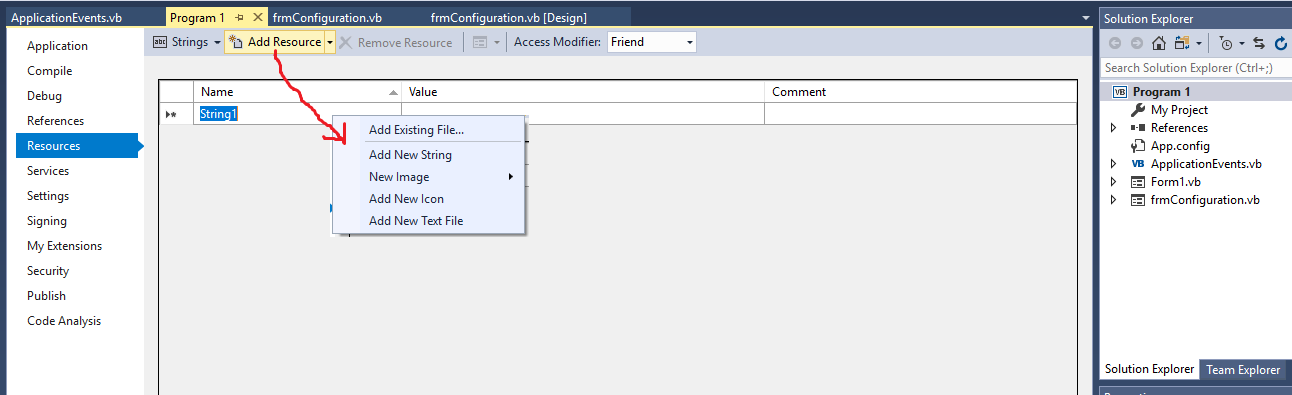
When the application launched, notice that we see the settings configuration screen. That’s because the -SETTINGS command line value was passed in indicating that the user is an administrator who may want to change some settings. I will change the username and password: 

Once I click Save the button and the accompanying “Settings Saved” MessageBox, the app starts and I connect as user James:



The next time the program is run without the command line argument, we start things up normally (no configuration screen). When we press the connect button, we are connected again as user James proving that the settings were saved. Remember to run the second copy from the command line as well; running from within the IDE will reset the settings to what you initially typed in the Settings window in the Project Properties.

In summary, you can see how we can easily create modifiable system settings and you learned about the command line as well. One other thing to point out before moving on is that there is a Resources tab in the Project settings just like the Settings tab. The difference here is that resources are items you want added to a project that should not be modifiable, like say icon files, background graphics, et cetera. The Resources tab is a really convenient way to bundle items into your application without having them floating around as individual files where they could be deleted or their path could get changed on you. The following is a screenshot of the Resources tab:



We can store pretty much anything in that we want as a resource. All resources are accessed in a manner that’s almost identical to our Settings. We just use My.Resources.*xxxx* to get to the resource we are interested in.

# The My Namespace

You’ll notice that throughout the previous chapter we would occasionally talk about My.Application and then some values within that namespace. Well, it’s part of a much larger collection of values that Microsoft makes immediately accessible to you and your VB programs, and in some cases, it makes our programming lives much easier. There is a whole big My universe. Let’s look at some of the major pieces that are available under My:

|  |  |
| --- | --- |
| *My item* | *What it has in it* |
| My.Application | Information about the current application, culture settings and so forth |
| My.Application.Info | Additional information about the application and its assembly |
| My.Applicaton.Log | We can generate logging information from the application |
| My.Computer | Resources within the local machine |
| My.Computer.Audio | Play sounds |
| My.Computer.Clipboard | Access to the system clipboard |
| My.Computer.Clock | Access to the current system date and time |
| My.Computer.Filesystem | Access to the drives, folders and files on the local filesystem |
| My.Computer.Filesystem.  Specialdirectories | Quick access to folders like Documents, Desktop, et cetera |
| My.Computer.Info | Information about the OS and other local resources |
| My.Computer.Keyboard | Access to the current state of the keyboard |
| My.Computer.Mouse | Access to the current state of the mouse |
| My.Computer.Network | Access to the current state of the network and some interaction tools |
| My.Computer.Ports | Access to the serial ports |
| My.Computer.Registry | Access to the registry and the ability to read/write keys |
| My.Forms | All forms defined within an application |
| My.Log | Used in ASP.NET applications to log activity |
| My.Request | Used in ASP.NET applications to access the Request object |
| My.Resources | Access to application specific resources included in the application |
| My.Response | Used in ASP.NET applications to access the Response object |
| My.Settings | Access to the application settings |
| My.User | Information about the who the current user is logged in as |
| My.WebServices | Used in ASP.NET applications to access the available web services |

So what’s the big deal? Well, My sometimes makes our work a little bit easier. Say for example that I want to play a system sound. Normally I’d have to look up the namespace that the SystemSound method lives in, include an Imports statement to it and what not just to call the method. My simplifies all of that, as in:

Public Class Form1

Private Sub Form1\_Load(sender As Object, e As EventArgs) Handles MyBase.Load

My.Computer.Audio.PlaySystemSound(Media.SystemSounds.Exclamation)

End Sub

End Class

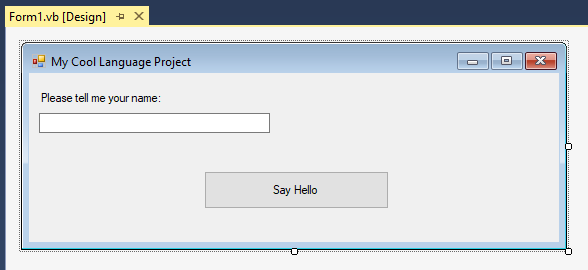
Is there anything that can be done with My that can’t be done in some other way? The answer to that is “no.” After all, My is just calling the .NET framework classes that are already there. What does My offer us? A tree-like representation of IntelliSense access to a lot of the common tasks we need to carry out. Take a little time to look at what’s bundled under My, but as you do, notice that much of what’s there we’ve already talked about. This means you already know which namespace these entities live in!

# Working with Internationalization

We can’t pretend that we live in a vacuum anymore. If we build a commercially successful piece of software, other people in foreign countries may want to use it. At a most basic level, we need to worry about things like date formats and currency formats. Fortunately, Windows provides us with culture-locale settings to correctly determine these values from the operating system. At a deeper level, however, it boils down to our applications needing to be able to display their messages, buttons and lists in other languages. This section will examine how to pull this off.

The way that .NET handles multiple languages is through resource files. Basically, we will create files that have information for each language that we are going to support placed in its own resource file. When the application is run, .NET will select the correct resource based on the locale settings of the current user.

I am going to create a US English, UK English, Spanish and German version of my project which is simply a form that asks for a username. When the command button is pressed on the form, a greeting is displayed that will need some of the locale information. All controls use their default names. Here’s the form:



Here’s the code:

'Chaper 23 - Program 3

Public Class Form1

Private Sub Button1\_Click(sender As Object, e As EventArgs) Handles Button1.Click

MessageBox.Show("Hi! " & TextBox1.Text & vbCrLf & " You have " &

FormatCurrency(1234.56) & " in your account " &

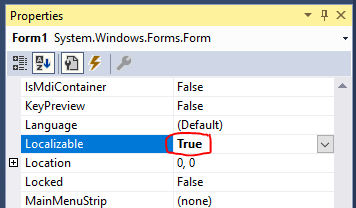
vbCrLf & "Today is " & FormatDateTime(Now(),

DateFormat.LongDate))

End Sub

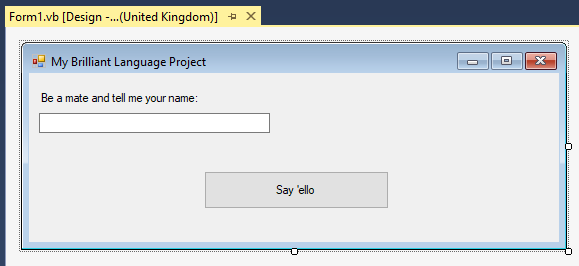
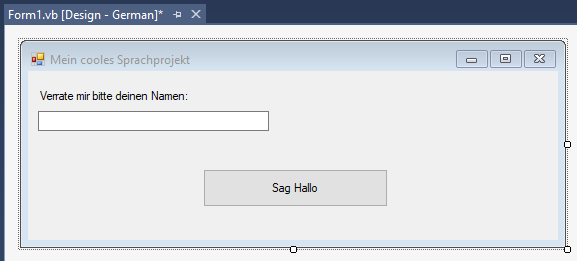
End Class

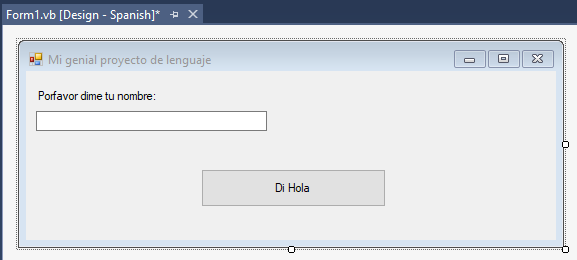
It’s important that you add all your controls to the form before you start to internationalize it. Once we have everything on the form the way we want it to appear, we set the Localizable property of our form to True:



Now if you missed a control, or want to add one later, you will have to change Localization back to False before being able to add the control. It’s just how VB keeps the language pieces intact. That’s why I recommend getting the form as perfect as possible in the first language before worrying about trying to internationalize…

We then set the Language property of the form to the locale (country/language) that we want to enter information for. We can change the Text property on the form to the new language version. We will then go to each control and change the Text on that control into the new language version to the language we are translating to. Each control will need to have the same work done to it… What this does is instructs Visual Studio to create a new resource file for each language; everything is handled for us.

Here are the various language versions I created (thanks to Google Translate)…  



Now add the following code to the Application Events Startup:

'Chapter 23 - Program 3

Imports Microsoft.VisualBasic.ApplicationServices

Namespace My

Partial Friend Class MyApplication

Private Sub MyApplication\_Startup(sender As Object, e As StartupEventArgs)

Handles Me.Startup

Dim strLanguage As String

strLanguage =

InputBox("Enter your language choice: US, UK, SPANISH or GERMAN")

Select Case strLanguage.ToUpper

Case "UK"

System.Threading.Thread.CurrentThread.CurrentUICulture =

New Globalization.CultureInfo("en-GB")

System.Threading.Thread.CurrentThread.CurrentCulture =

New Globalization.CultureInfo("en-GB")

Case "SPANISH"

System.Threading.Thread.CurrentThread.CurrentUICulture =

New Globalization.CultureInfo("es-ES")

System.Threading.Thread.CurrentThread.CurrentCulture =

New Globalization.CultureInfo("es-ES")

Case "GERMAN"

System.Threading.Thread.CurrentThread.CurrentUICulture =

New Globalization.CultureInfo("de-DE")

System.Threading.Thread.CurrentThread.CurrentCulture =

New Globalization.CultureInfo("de-DE")

'Otherwise we will default to the default language which is US

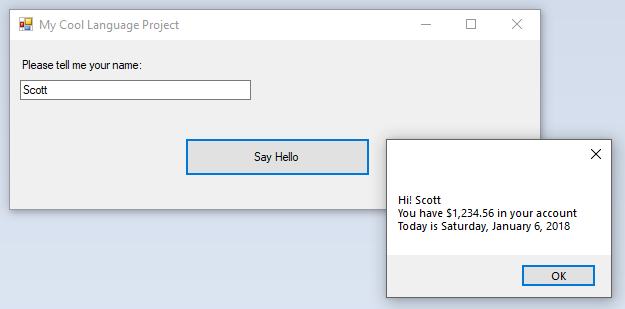
End Select

End Sub

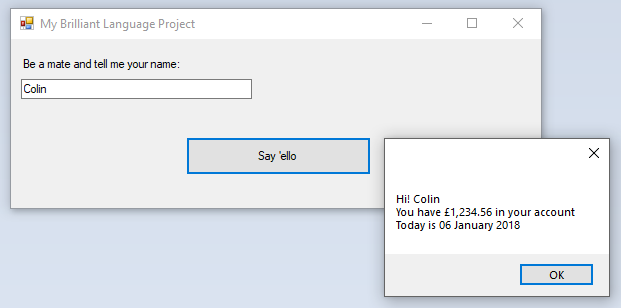
End Class

End Namespace

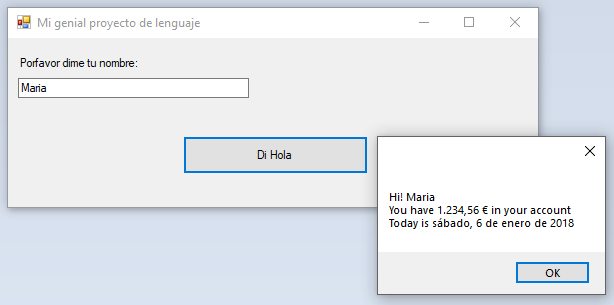
Here’s an example where I entered US as the language:



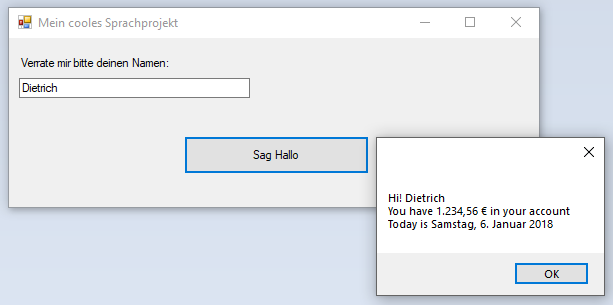
Running it again with UK:



Notice that the currency mark and the date format also changed! Here it is in Spanish:



See the , replace spots with our .? Also notice that it is Euros since we just picked Spanish and that defaults to Spain. Finally, our German version:



About the only real issue to deal with is that the MessageBox text stayed in the default language. This problem is tougher to solve than what we’ve already looked at. One of the common solutions used is to build up a database of localized messages that you want to display, for example:

|  |  |  |
| --- | --- | --- |
| *ItemName* | *Locale* | *ItemFormat* |
| Messagebox1text | Default | Hi! {0} You have {1] in your account Today is {2} |
| Messagebox1text | en-GB | Cheers! {0} You have {1} in the bank Today is {2} |
| Messagebox1text | es-ES | Hola! {0} Tienes {1} en tu cuenta Hoy es {2} |
| Messagebox1text | de-DE | Hallo! {0} Sie haben {1} auf ihrem konto Heute ist {2} |

We would then perform a SQL Query along the order of:

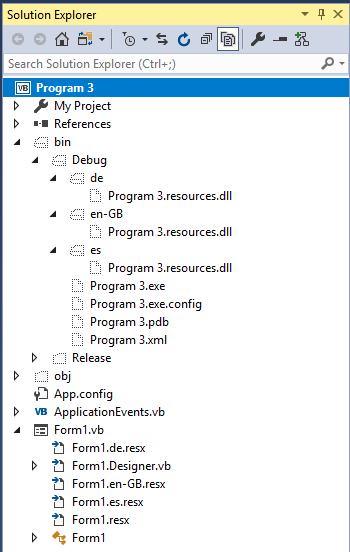
SELECT \* FROM LocalizationDatabase WHERE ITEMNAME = Messagebox1text and LOCALE = es-ES

Then update the MessageBox to:

MessageBox.Show(ItemFormat, TextBox1.Text, FormatCurrency(500.0),

FormatDateTime(Now(), DateFormat.ShortDate))

Now when the application runs, everything is truly localized! Your second choice would be to continue to handle things the way that Visual Studio does, which is through multiple Resource Files…take a look at the project with All Files Shown active:



You can see that there is a dll created in the final compilation for each locale. Internally there is also a resource file for each locale. We could place our own settings in those files and pull them out as well. It just depends on which way you think is cleaner.

# Working with Logging

Many times, when building applications, we want to be able to output either instrumentation or diagnostic messages. Sometimes we simply want to record what went on during an install, other times, if we detect a program crash of some sort, we may want to record what was going on prior to the crash. In any case, this brings up the issue of creating a log.

There are two forms of logging that we commonly use: writing to the Windows Event Log file system and using trace files. The event log file is maintained by the operating system and anything that we write to that log file can be viewed with the Log Viewer software. Let’s look at a simple program that throws an exception and then writes information out to the EventLog:

'Chapter 23 - Program 4

Module Module1

'Note: you must run this application as an administrator for it

'to work right. Compile the program to an .EXE and then open a

'command shell with administrator privileges and then run the

'program

Sub Main()

Dim myEventLog As New EventLog

Dim myEventLogEntryType As EventLogEntryType

Try

Throw (New ArgumentOutOfRangeException)

Catch ex As ArgumentOutOfRangeException

If Not EventLog.SourceExists("LogExampleApp") Then

EventLog.CreateEventSource("LogExampleApp", "System")

End If

myEventLog.Source = "LogExampleApp"

myEventLog.Log = "System"

myEventLogEntryType = EventLogEntryType.Information

myEventLog.WriteEntry("An Error Occurred: " & ex.Message,

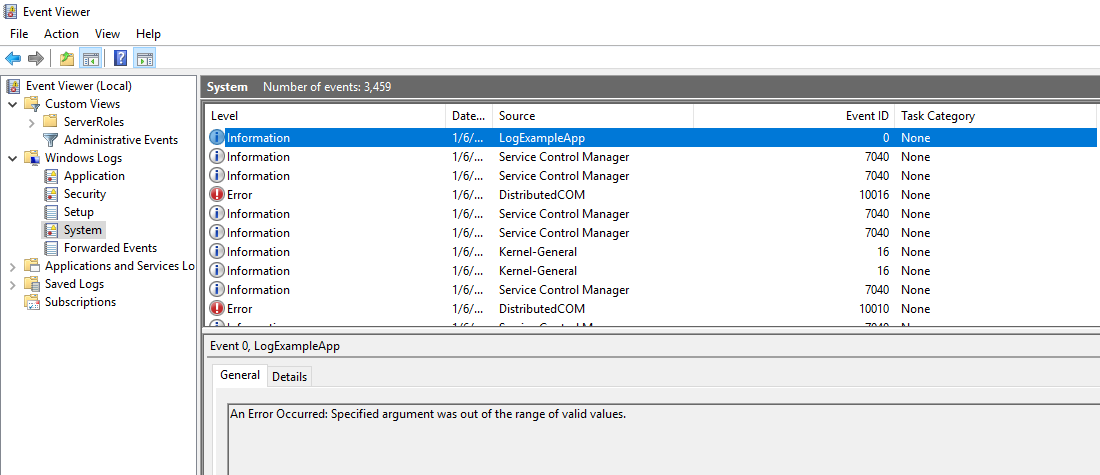
myEventLogEntryType)

End Try

End Sub

End Module

If we use the Control Panel🡪Administrative Tools🡪Event Viewer, we will see that we successfully wrote an error message out to the system log:



That was really almost too easy. In general, we’re not going to want to write every little thing out to the operating system event logs – those should be reserved for big showstopper type events.

Here are the methods/properties we are interested in on the EventLog class:

|  |  |
| --- | --- |
| *Item* | *Purpose* |
| CreateEventSource | Creates an event source in the specified log |
| Delete | Deleted an entire Event Log |
| DeleteEventSource | Deletes an event source and associated entries |
| Exists | Used to determine whether a specific Event Log exists |
| GetEventLogs | Retrieves a list of all Event Logs on a particular computer |
| Log | Property that specifies the log file to write to. Application is the default if not specified…remember there are three: Application, System and Security |
| Source | Property that specifies the source of the entry to be written |
| SourceExists | Used to determine whether a specific source exists in a log |
| WriteEntry | Writes a string to a specified log |

The second technique to performing application logging is using trace files. With this method, we own the file where things are written so we can log to our heart’s content. Here’s a sample program that shows how this works:

'Chapter 23 - Program 5

Imports Microsoft.VisualBasic.Logging

Module Module1

Sub Main()

Dim myLog As New Log("SimpleTraceLog")

'Make sure that the folder exists before running this

'program

myLog.DefaultFileLogWriter.Location = LogFileLocation.Custom

myLog.DefaultFileLogWriter.CustomLocation = "C:\VBLOG"

'We have to use MsgBox since this is a console application

MsgBox(myLog.DefaultFileLogWriter.FullLogFileName)

myLog.WriteEntry("Application starting up")

Try

Throw New ArgumentOutOfRangeException

Catch ex As ArgumentOutOfRangeException

myLog.WriteException(ex)

End Try

myLog.WriteEntry("Application shutting down")

myLog.DefaultFileLogWriter.Close()

End Sub

End Module

From the program’s code, you can see that we can set where we want our logging information to be written to. Once I’ve run the program, here’s a copy of what’s in C:\VBLOG\Program 4.log:

SimpleTraceLog Information 0 Application starting up

SimpleTraceLog Error 2 Specified argument was out of the range of valid values.

SimpleTraceLog Information 0 Application shutting down

Since all of this information is written locally, and not to the operating system’s logging facility, we can send whatever values we want out to the log file and not have to worry about overwhelming it. In the Logging namespace, there are two classes that we are interested in: Log and FileLogTraceListener.

Here are the properties for the Log class:

|  |  |
| --- | --- |
| *Property* | *Purpose* |
| [DefaultFileLogWriter](http://msdn.microsoft.com/en-us/library/microsoft.visualbasic.logging.log.defaultfilelogwriter.aspx) | Gets the [FileLogTraceListener](http://msdn.microsoft.com/en-us/library/microsoft.visualbasic.logging.filelogtracelistener.aspx) object that underlies the Log object. |
| [TraceSource](http://msdn.microsoft.com/en-us/library/microsoft.visualbasic.logging.log.tracesource.aspx) | Gets the [TraceSource](http://msdn.microsoft.com/en-us/library/system.diagnostics.tracesource.aspx) object that underlies the Log object. |

Here are the methods for the Log class:

|  |  |
| --- | --- |
| *Method* | *Purpose* |
| [InitializeWithDefaultsSinceNoConfigExists](http://msdn.microsoft.com/en-us/library/microsoft.visualbasic.logging.log.initializewithdefaultssincenoconfigexists.aspx) | Creates a new [FileLogTraceListener](http://msdn.microsoft.com/en-us/library/microsoft.visualbasic.logging.filelogtracelistener.aspx) object and adds it to the [Listeners](http://msdn.microsoft.com/en-us/library/system.diagnostics.tracesource.listeners.aspx) collection. |
| [WriteEntry(String)](http://msdn.microsoft.com/en-us/library/ms128188.aspx) | Writes a message to the application's log listeners. |
| [WriteException(Exception)](http://msdn.microsoft.com/en-us/library/ms128193.aspx) | Writes exception information to the application's log listeners. |

Here are the properties of the FileLogTraceListener class:

|  |  |
| --- | --- |
| *Property* | *Purpose* |
| [Append](http://msdn.microsoft.com/en-us/library/microsoft.visualbasic.logging.filelogtracelistener.append.aspx) | Determines whether to append the output to the current file or write it to a new file. |
| [Attributes](http://msdn.microsoft.com/en-us/library/system.diagnostics.tracelistener.attributes.aspx) | Gets the custom trace listener attributes defined in the application configuration file. (Inherited from [TraceListener](http://msdn.microsoft.com/en-us/library/system.diagnostics.tracelistener.aspx).) |
| [AutoFlush](http://msdn.microsoft.com/en-us/library/microsoft.visualbasic.logging.filelogtracelistener.autoflush.aspx) | Indicates whether or not the writing to the log file stream flushes the buffer. |
| [BaseFileName](http://msdn.microsoft.com/en-us/library/microsoft.visualbasic.logging.filelogtracelistener.basefilename.aspx) | Gets or sets the base name for the log files, which is used to create the full log-file name. |
| [CustomLocation](http://msdn.microsoft.com/en-us/library/microsoft.visualbasic.logging.filelogtracelistener.customlocation.aspx) | Gets or sets the log file directory when the [Location](http://msdn.microsoft.com/en-us/library/microsoft.visualbasic.logging.filelogtracelistener.location.aspx) property is set to [Custom](http://msdn.microsoft.com/en-us/library/ms128203.aspx). |
| [Delimiter](http://msdn.microsoft.com/en-us/library/microsoft.visualbasic.logging.filelogtracelistener.delimiter.aspx) | Gets or sets the delimiter used to delimit fields within a log message. |
| [DiskSpaceExhaustedBehavior](http://msdn.microsoft.com/en-us/library/microsoft.visualbasic.logging.filelogtracelistener.diskspaceexhaustedbehavior.aspx) | Determines what to do when writing to the log file and there is less free disk space available than specified by the[ReserveDiskSpace](http://msdn.microsoft.com/en-us/library/microsoft.visualbasic.logging.filelogtracelistener.reservediskspace.aspx) property. |
| [Encoding](http://msdn.microsoft.com/en-us/library/microsoft.visualbasic.logging.filelogtracelistener.encoding.aspx) | Gets or sets the encoding to use when creating a new log file. |
| [Filter](http://msdn.microsoft.com/en-us/library/system.diagnostics.tracelistener.filter.aspx) | Gets and sets the trace filter for the trace listener. (Inherited from [TraceListener](http://msdn.microsoft.com/en-us/library/system.diagnostics.tracelistener.aspx).) |
| [FullLogFileName](http://msdn.microsoft.com/en-us/library/microsoft.visualbasic.logging.filelogtracelistener.fulllogfilename.aspx) | Gets the current full log-file name. |
| [IncludeHostName](http://msdn.microsoft.com/en-us/library/microsoft.visualbasic.logging.filelogtracelistener.includehostname.aspx) | Indicates whether or not the host name of the logging machine should be included in the output. |
| [IndentLevel](http://msdn.microsoft.com/en-us/library/system.diagnostics.tracelistener.indentlevel.aspx) | Gets or sets the indent level. (Inherited from [TraceListener](http://msdn.microsoft.com/en-us/library/system.diagnostics.tracelistener.aspx).) |
| [IndentSize](http://msdn.microsoft.com/en-us/library/system.diagnostics.tracelistener.indentsize.aspx) | Gets or sets the number of spaces in an indent. (Inherited from [TraceListener](http://msdn.microsoft.com/en-us/library/system.diagnostics.tracelistener.aspx).) |
| [IsThreadSafe](http://msdn.microsoft.com/en-us/library/system.diagnostics.tracelistener.isthreadsafe.aspx) | Gets a value indicating whether the trace listener is thread safe. (Inherited from [TraceListener](http://msdn.microsoft.com/en-us/library/system.diagnostics.tracelistener.aspx).) |
| [Location](http://msdn.microsoft.com/en-us/library/microsoft.visualbasic.logging.filelogtracelistener.location.aspx) | Gets or sets location for the log files. |
| [LogFileCreationSchedule](http://msdn.microsoft.com/en-us/library/microsoft.visualbasic.logging.filelogtracelistener.logfilecreationschedule.aspx) | Determines which date to include in the names of the log files. |
| [MaxFileSize](http://msdn.microsoft.com/en-us/library/microsoft.visualbasic.logging.filelogtracelistener.maxfilesize.aspx) | Gets or sets the maximum allowed size of the log file, in bytes. |
| [Name](http://msdn.microsoft.com/en-us/library/system.diagnostics.tracelistener.name.aspx) | Gets or sets a name for this [TraceListener](http://msdn.microsoft.com/en-us/library/system.diagnostics.tracelistener.aspx). (Inherited from [TraceListener](http://msdn.microsoft.com/en-us/library/system.diagnostics.tracelistener.aspx).) |
| [NeedIndent](http://msdn.microsoft.com/en-us/library/system.diagnostics.tracelistener.needindent.aspx) | Gets or sets a value indicating whether to indent the output. (Inherited from [TraceListener](http://msdn.microsoft.com/en-us/library/system.diagnostics.tracelistener.aspx).) |
| [ReserveDiskSpace](http://msdn.microsoft.com/en-us/library/microsoft.visualbasic.logging.filelogtracelistener.reservediskspace.aspx) | Gets or sets the amount of free disk space, in bytes, necessary before messages can be written to the log file. |
| [TraceOutputOptions](http://msdn.microsoft.com/en-us/library/system.diagnostics.tracelistener.traceoutputoptions.aspx) | Gets or sets the trace output options. (Inherited from [TraceListener](http://msdn.microsoft.com/en-us/library/system.diagnostics.tracelistener.aspx).) |

Here are the methods of the FileLogTraceListener class:

|  |  |
| --- | --- |
| *Method* | *Purpose* |
| [Close](http://msdn.microsoft.com/en-us/library/microsoft.visualbasic.logging.filelogtracelistener.close.aspx) | Closes the underlying stream for the current log file and releases any resources associated with the current stream. (Overrides [TraceListener.Close](http://msdn.microsoft.com/en-us/library/system.diagnostics.tracelistener.close.aspx).) |
| [Fail(String)](http://msdn.microsoft.com/en-us/library/0kyffesz.aspx) | Emits an error message to the listener you create when you implement the [TraceListener](http://msdn.microsoft.com/en-us/library/system.diagnostics.tracelistener.aspx) class. (Inherited from [TraceListener](http://msdn.microsoft.com/en-us/library/system.diagnostics.tracelistener.aspx).) |
| [Flush](http://msdn.microsoft.com/en-us/library/microsoft.visualbasic.logging.filelogtracelistener.flush.aspx) | Flushes the underlying stream that writes to the current log file. (Overrides [TraceListener.Flush](http://msdn.microsoft.com/en-us/library/system.diagnostics.tracelistener.flush.aspx).) |
| [GetSupportedAttributes](http://msdn.microsoft.com/en-us/library/microsoft.visualbasic.logging.filelogtracelistener.getsupportedattributes.aspx) | Gets the custom XML configuration attributes supported by the trace listener. (Overrides[TraceListener.GetSupportedAttributes](http://msdn.microsoft.com/en-us/library/system.diagnostics.tracelistener.getsupportedattributes.aspx).) |
| [TraceData(TraceEventCache, String, TraceEventType, Int32, Object)](http://msdn.microsoft.com/en-us/library/ms128149.aspx) | Writes trace information, a data object, and event information to the output file or stream. (Overrides[TraceListener.TraceData (TraceEventCache, String, TraceEventType, Int32, Object)](http://msdn.microsoft.com/en-us/library/6w7c542z.aspx).) |
| [TraceEvent(TraceEventCache, String, TraceEventType, Int32)](http://msdn.microsoft.com/en-us/library/dz2kace8.aspx) | Writes trace and event information to the listener specific output. (Inherited from [TraceListener](http://msdn.microsoft.com/en-us/library/system.diagnostics.tracelistener.aspx).) |
| [TraceTransfer](http://msdn.microsoft.com/en-us/library/system.diagnostics.tracelistener.tracetransfer.aspx) | Writes trace information, a message, a related activity identity and event information to the listener specific output. (Inherited from [TraceListener](http://msdn.microsoft.com/en-us/library/system.diagnostics.tracelistener.aspx).) |
| Write | Writes the item to the listener you created |
| [WriteIndent](http://msdn.microsoft.com/en-us/library/system.diagnostics.tracelistener.writeindent.aspx) | Writes the indent to the listener you create when you implement this class, and resets the [NeedIndent](http://msdn.microsoft.com/en-us/library/system.diagnostics.tracelistener.needindent.aspx)property to false. (Inherited from [TraceListener](http://msdn.microsoft.com/en-us/library/system.diagnostics.tracelistener.aspx).) |
| WriteLine | Writes the item to the listener you created followed by a newline character. |

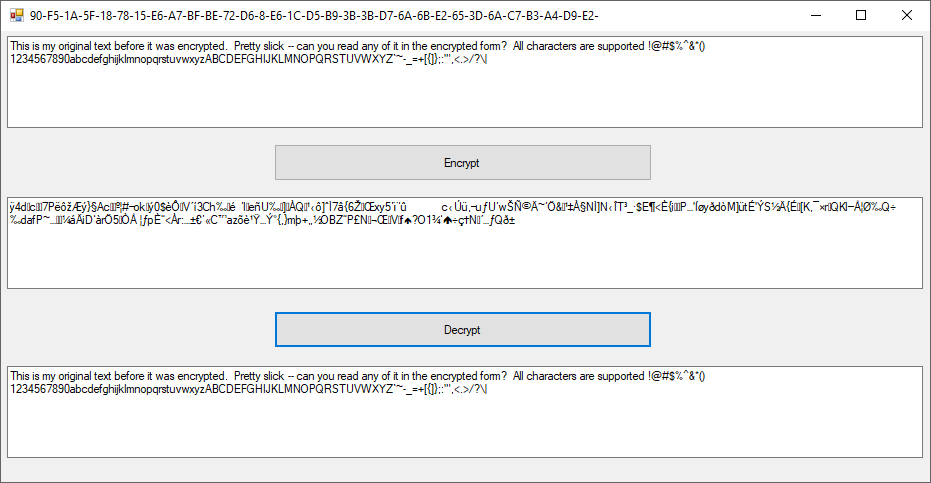
# Introduction to Security in .NET

Without getting into a whole lot of gory details, suffice it to say that .NET comes pretty well stocked if you want to perform encryption. We have a whole namespace set aside for it: System.Security.Cyptography. Both symmetric (the same key is used to encrypt and decrypt) and asymmetric encryption types (different keys are used for encrypting and decrypting) are supported.

Here are the Symmetric Cryptography algorithms that are supported:

* DES (Data Encryption Standard) uses a 56-bit block cipher. Fast, secure and small, it is inappropriate for highly sensitive data. DESCryptoServiceProvider is the class that handles this type of encryption.
* RC2 (Rivest Cipher 2) also uses a 56-bit block. This encryption standard was able to be exported outside of the U.S. – see the RC2CryptoServiceProvider class for details.
* Triple DES uses an underlying DES algorithm 3 times to generate more secure results than DES. It is available in TripleDESCryptoServiceProvider, but is not as secure as Rijndael.
* Rijndael uses between a 128-bit and 256-bit block cipher. It is related to a similar algorithm called AES (Advanced Encryption Standard) and is managed through the RijndaelManaged class. It is the most secure of the symmetric technologies.

Let’s write a simple symmetric key based text encryption program. We’ll build a simple form with three TextBoxes (txtSource, txtEncrypted, txtDecrypted in order from top to bottom) and two Command buttons (cmdEncrypt and cmdDecrypt) that round trip encrypt and decrypt:



Here’s the program code:

'Chapter 23 - Program 6

'We need access to streams since that's the way our cryptography works

Imports System.IO

Imports System.Security.Cryptography

Public Class Form1

'Let's make things as secure as possible

Dim myKey As New RijndaelManaged

'Our encrypted for will return as an array of bytes

Dim arrEncryptedVersion() As Byte

Private Sub cmdEncrypt\_Click(sender As Object, e As EventArgs)

Handles cmdEncrypt.Click

'Cryptography uses a stream approach: send in a stream

'and tell the cryptography provider what to do with it

Dim myEncryptionStream As CryptoStream

'After the encryption is done, we want to be able to

'get the encrypted output values...need a streamwriter

Dim myOutStream As StreamWriter

'We have to have some way to attach the crypto and

'stream writers together since we're not using files...

Dim myMemoryStream As New MemoryStream

'Create a new CryptoStream pointed at the underlying

'memory stream

myEncryptionStream = New CryptoStream(myMemoryStream, myKey.CreateEncryptor(),

CryptoStreamMode.Write)

'Create a new output stream to the encryption stream

myOutStream = New StreamWriter(myEncryptionStream)

'Simply write the plain text to the output stream,

'which in turn sends it to the encryption stream

myOutStream.WriteLine(txtSource.Text)

'Close these streams up...

myOutStream.Close()

myEncryptionStream.Close()

'We can now fetch the encrypted stream from the memory

'stream where the cryptography provider placed it

arrEncryptedVersion = myMemoryStream.ToArray()

myMemoryStream.Close()

'Since the encrypted form is a series of bytes,

'convert each to a character and print it...

txtEncrypted.Text = ""

For Each aByte In arrEncryptedVersion

txtEncrypted.Text &= Chr(aByte)

Next

'Show the key as a series of hex bytes in the form title

Me.Text = ""

For Each abyte In myKey.Key

Me.Text &= Hex(Int(abyte)) & "-"

Next

End Sub

Private Sub cmdDecrypt\_Click(sender As Object, e As EventArgs)

Handles cmdDecrypt.Click

Dim myMemoryStream As New MemoryStream

Dim myEncryptionStream As CryptoStream

Dim myInputStream As StreamReader

'Pretty much the opposite of the encryption process...

myMemoryStream = New MemoryStream(arrEncryptedVersion)

myEncryptionStream = New CryptoStream(myMemoryStream, myKey.CreateDecryptor,

CryptoStreamMode.Read)

myInputStream = New StreamReader(myEncryptionStream)

'Read the decrypted text from the input stream

txtDecrypted.Text = myInputStream.ReadLine

myInputStream.Close()

myEncryptionStream.Close()

myMemoryStream.Close()

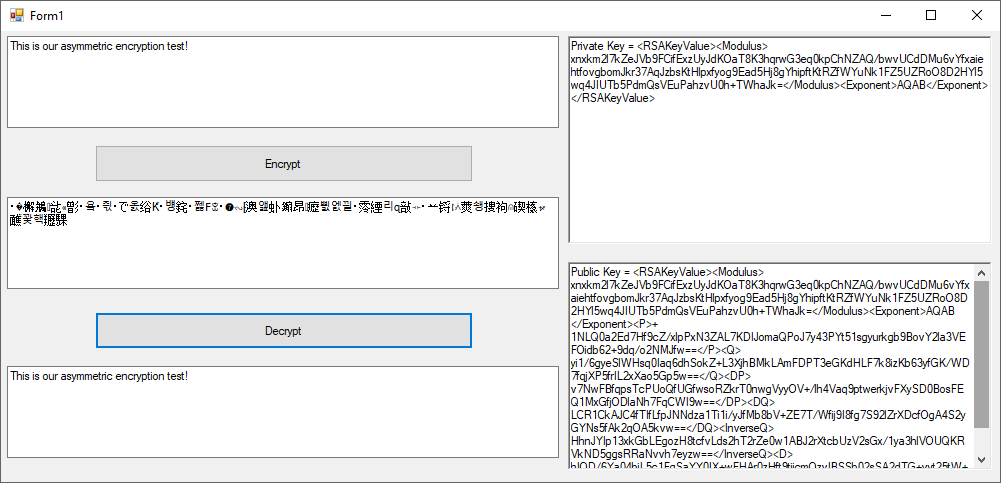
End Sub

End Class

If you are interested in Asymmetric cryptography, .NET has two providers:

* DSA (Digital Signature Algorithm) developed by the U.S. Government and supported through DSACryptoServiceProvider
* RSA (Rivest, Shamir and Adleman), an older asymmetric scheme in the RSACryptoServiceProvider class

Here’s the last application recast as an asymmetric application. All the control names stay the same as the last program, but we add two RichTextBoxes (rtbPrivateKey and rtbPublicKey on the right side, top to bottom):



Our application code looks like this:

'Chapter 23 - Program 7

'We need access to streams since that's the way our cryptography works

Imports System.IO

Imports System.Text

Imports System.Security.Cryptography

Public Class Form1

'Create a UnicodeEncoder to convert between byte array and string.

Dim ByteConverter As New UnicodeEncoding()

'Create byte arrays to hold source, encrypted, and decrypted data.

Dim dataToEncrypt As Byte() = ByteConverter.GetBytes("Data to Encrypt")

Dim encryptedData() As Byte

Dim decryptedData() As Byte

'Create a new RSACryptoServiceProvider object.

Dim RSA As New RSACryptoServiceProvider()

'Export the key information to an RSAParameters object.

'Pass false to export the public key information or pass

'true to export public and private key information.

Dim RSAParams As RSAParameters = RSA.ExportParameters(True)

Private Sub cmdEncrypt\_Click(sender As Object, e As EventArgs)

Handles cmdEncrypt.Click

'Convert the original text into a byte array

dataToEncrypt = ByteConverter.GetBytes(txtSource.Text)

'Import the RSA Key information.

RSA.ImportParameters(RSAParams)

'Pass the data to ENCRYPT and a boolean flag specifying

'no OAEP padding.

encryptedData = RSA.Encrypt(dataToEncrypt, False)

'Convert the byte array into a string for display

txtEncrypted.Text = ByteConverter.GetString(encryptedData)

rtbPrivateKey.Text = "Private Key = " & RSA.ToXmlString(False)

rtbPublicKey.Text = "Public Key = " & RSA.ToXmlString(True)

End Sub

Private Sub cmdDecrypt\_Click(sender As Object, e As EventArgs)

Handles cmdDecrypt.Click

'Pass the data to DECRYPT and a boolean flag specifying

'no OAEP padding.

decryptedData = RSA.Decrypt(encryptedData, False)

'Convert the decrypted byte array back into a string

txtDecrypted.Text = ByteConverter.GetString(decryptedData)

End Sub

End Class

Notice that we are able to access to the two keys involved in the asymmetric cryptography (the private and the public key) and display them. They are encrypted of course…

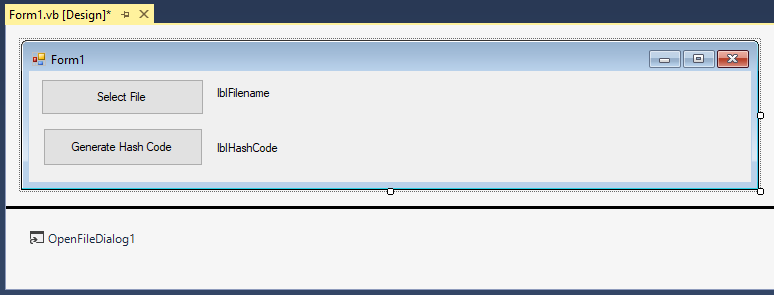
## Hash Codes

Many times, we need to calculate a value that can be recalculated to verify data content. For example, how do you know that a file’s contents haven’t been changed since the file was created? Hash codes are calculated using a user specified key and then can be recomputed as needed to ensure that the original object is still identical to the form it was in when the hash code was created.

.NET again supports several different hashing types:

* HMAC (Hash-based Message Authentication Code) using the SHA-1 hash function. It uses a 160-bit hash code and it is available through HMACSHA1 – it is fairly easy to break.
* MAC (Message Authentication Code) using the Triple-DES algorithm. This has a key of 128 or 192 bits in length and the generated value is 64 bits in length. You can find it in MACTripleDES.
* MD5 (Message-Digest Algorithm Number 5) is another Rivest algorithm. This is a very fast hash technique with 128 bit long hash value, but it has been shown to have some security flaws. This is the MD5CryptoServiceProvider class.
* SHA1Managed uses the SHA-1 algorithm like HMAC, but it is written natively in .NET. Several of the other hashes are written using pre-.NET DLLs. There are also variants called SHA256Managed, SHA384Managed and SHA512Managed; the difference is the hash size and the computational time to construct them.

Let’s write a simple application that will let us calculate the hash code of any file. We’ll then make a change to the file to prove the hash value changes. Our form consists of two Command buttons (cmdSelectFile and cmdGenerateHashCode), an OpenFileDialog (using the default name) and two Labels (lblFilename and lblHashCode). This is what the form looks like:



Here’s the code:

'Chapter 23 - Program 8

Imports System.IO

Imports System.Security.Cryptography

Public Class Form1

Private Sub cmdSelectFile\_Click(sender As Object, e As EventArgs)

Handles cmdSelectFile.Click

'Pick a file...

OpenFileDialog1.ShowDialog()

lblFilename.Text = OpenFileDialog1.FileName

End Sub

Private Sub cmdGenerateHashCode\_Click(sender As Object, e As EventArgs)

Handles cmdGenerateHashCode.Click

'Pick the algorithm to use

Dim myHashAlgorithm = New SHA1Managed

'Our hash will be returned as an array

Dim arrHashValue() As Byte

'Open the file with a FileStream

Using myFile As New FileStream(lblFilename.Text, FileMode.Open, FileAccess.Read)

'Send the file stream to the hasher

arrHashValue = myHashAlgorithm.ComputeHash(myFile)

'Display the results

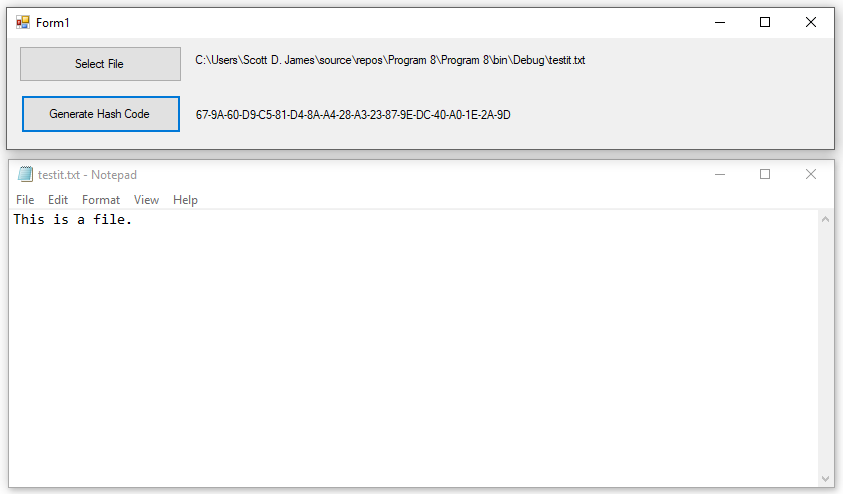
lblHashCode.Text = BitConverter.ToString(arrHashValue)

End Using

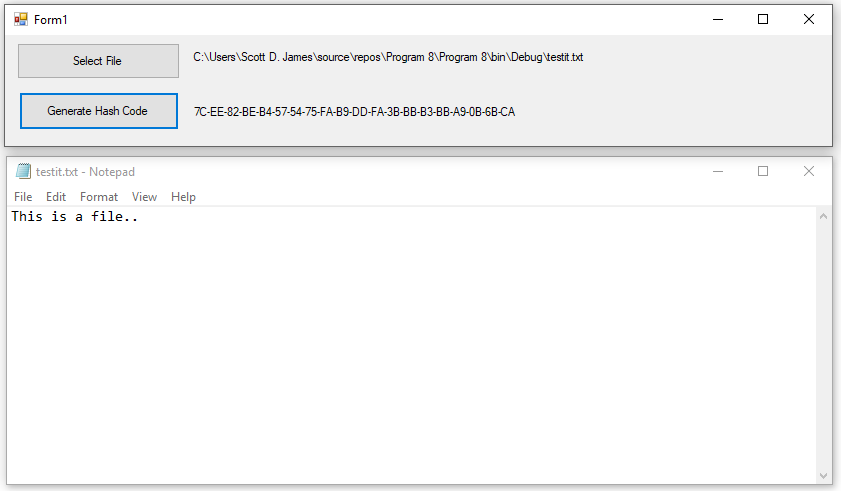
End Sub

End Class

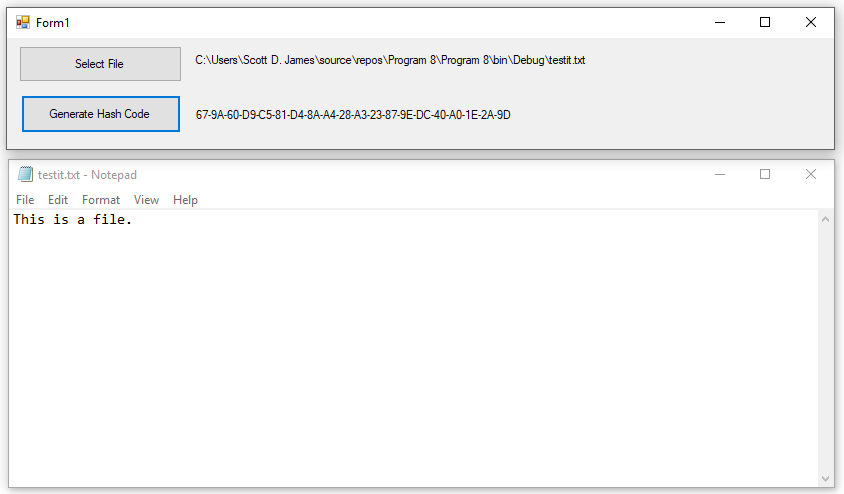
Here’s the first execution on the original file:



Let’s make one little change (an additional period at the end of the sentence):



Notice that the hash changed… Let’s put the file back to the original version and try the hash one more time:



Pretty easy way to detect any changes to data integrity! We can also send a password in, “seeding” the hash value – that way only people that know the password would be able to receive the proper hash value – see the KeyedHashAlgorithm class for more information.

## SecureString

This is one more class you should know about in an attempt to keep potential crackers from accessing strings when your application is running. Normal strings, even if they are just in memory, are subject to being located and deciphered. If we want to prevent this, all we need to do is declare our sensitive string variables of type SecureString.

Internally, we don’t see anything different, but .NET is encrypting on the fly when the strings are needed. Here’s a sample application using the SecureString class:

'Chapter 23 - Program 9

Imports System.Security

Module Module1

Public Sub Main()

Dim myPassword As New SecureString

Console.WriteLine("Please enter password: ")

myPassword = ReadString()

Console.WriteLine()

Console.WriteLine("What's in mypassword is now secure and inaccessible : " &

myPassword.ToString)

Console.WriteLine("Once the data is in a SecureString, it can only be sent " &

"to other items that accept secure strings...")

Console.ReadLine()

End Sub

Function ReadString() As SecureString

'SecureString only handles one character at a time...

Dim aSecureString As New SecureString

Dim key As ConsoleKeyInfo

Do

key = Console.ReadKey(True)

' Append the character to the password.

aSecureString.AppendChar(key.KeyChar)

Console.Write("\*")

' Exit if Enter key is pressed.

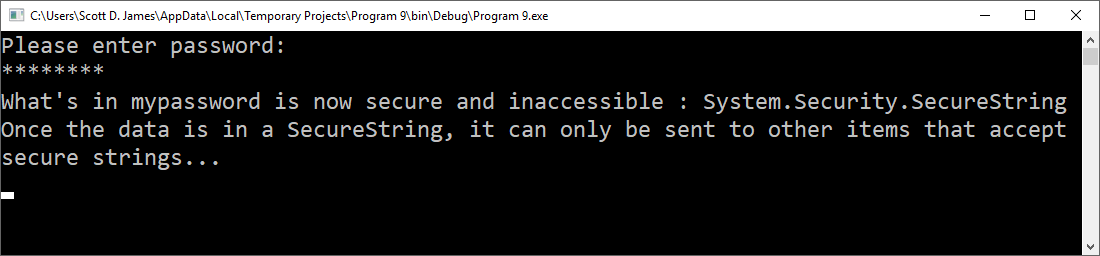
Loop While key.Key <> ConsoleKey.Enter

Return aSecureString

End Function

End Module

Here’s the program when it’s run:



# Reflection

The last thing to talk about in this chapter is reflection. You’ve been using it for some time now, but we haven’t specifically discussed it. Reflection is the ability for .NET to look at itself at runtime and figure out things like data types. Didn’t you kind of wonder how generics and LINQ could determine what type of stuff they ought to return? That’s reflection at work and there is an entire namespace dedicated to it.

One of the main reflection methods that we care about is the GetType method which all objects inherit. This is where a data type “gives up” what type it is to the program…

'Chapter 23 - Program 10

Class clsClass

Public aField As Integer

Public anotherField As New Dictionary(Of String, String)

Public afinalField As New ArgumentOutOfRangeException

End Class

Module Module1

Sub Main()

Dim myString As String = "This"

Dim aClass As New clsClass

Debug.WriteLine(myString.GetType)

Debug.WriteLine(aClass.GetType)

Debug.WriteLine(aClass.aField.GetType)

Debug.WriteLine(aClass.anotherField.GetType)

Debug.WriteLine(aClass.afinalField.GetType)

'We can also get type information this way:

Dim t As Type = GetType(clsClass)

Debug.WriteLine(t.ToString)

End Sub

End Module

Here’s the output:

System.String

Program\_10.clsClass

System.Int32

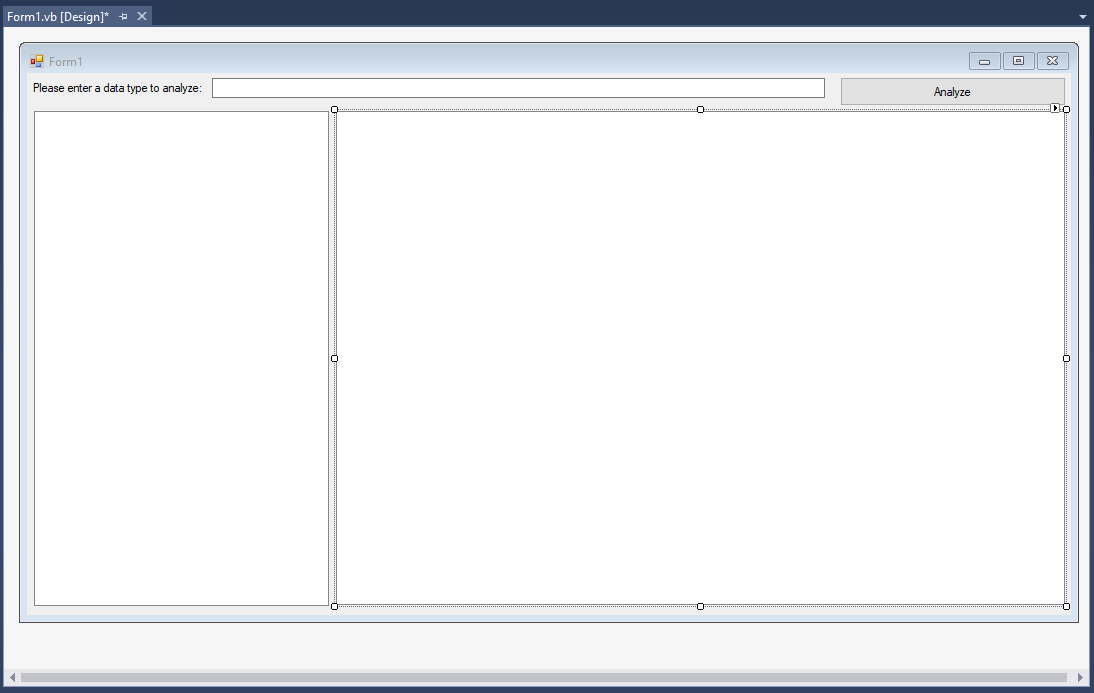
System.Collections.Generic.Dictionary`2[System.String,System.String]

System.ArgumentOutOfRangeException

Program\_10.clsClass

Now that we know that we have access to this type of information, let’s build a reflection analyzer application using a couple of controls that we haven’t played with yet: the TreeView and the ListView. Your application will work the following way. You will ask the user to input a data type and then the TreeView will show any members in that data type. Anytime the user clicks on a member, the details of that member will appear in the ListView.

Here’s the completed form. All controls used their default names. The TreeView control is the left control, while the ListView is the right control:



Here’s the source code to the application:

'Chapter 23 - Program 11

'We need to include the reflection namespace...

Imports System.Reflection

Public Class Form1

Private Sub Form1\_Load(sender As Object, e As EventArgs) Handles MyBase.Load

'Make sure that the ListView is in the "details" form, not

'an iconized view (remember there are several views that

'are available in a ListView)

ListView1.View = View.Details

End Sub

Private Sub Button1\_Click(sender As Object, e As EventArgs) Handles Button1.Click

'Get a copy of the type so we can reflect on it...

Dim myType As Type = Type.GetType(TextBox1.Text)

'These variables will hold the various items the type contains

Dim dtMethods() As MethodInfo

Dim dtFields() As FieldInfo

Dim dtProperties() As PropertyInfo

Dim dtInterfaces

'We need these to add category nodes and informational subnode

'pieces to the TreeView

Dim aNode, aSubNode As TreeNode

'Clear both the TreeView and the ListView objects out

TreeView1.Nodes.Clear()

ListView1.Items.Clear()

ListView1.Columns.Clear()

'Try to get any methods that this type might have

Try

dtMethods = myType.GetMethods()

'If we have some

If dtMethods.Count > 0 Then

'Add a "Methods" category node

aNode = New TreeNode

aNode.Name = "Methods"

aNode.Text = "Methods"

TreeView1.Nodes.Add(aNode)

'Under that category add all of the method names

'as subnodes -- we can then expand/contract what's

'shown

For Each myMethodInfo In dtMethods

aSubNode = New TreeNode

aSubNode.Name = myMethodInfo.Name

aSubNode.Text = myMethodInfo.Name

TreeView1.Nodes(aNode.Name).Nodes.Add(aSubNode)

Next

End If

Catch ex As Exception

End Try

'Now we do the same thing to see if the type has any fields

Try

dtFields = myType.GetFields()

If dtFields.Count > 0 Then

aNode = New TreeNode

aNode.Name = "Fields"

aNode.Text = "Fields"

TreeView1.Nodes.Add(aNode)

For Each myFieldInfo In dtFields

aSubNode = New TreeNode

aSubNode.Name = myFieldInfo.Name

aSubNode.Text = myFieldInfo.Name

TreeView1.Nodes(aNode.Name).Nodes.Add(aSubNode)

Next

End If

Catch ex As Exception

End Try

'Check for properties...

Try

dtProperties = myType.GetProperties()

If dtProperties.Count > 0 Then

aNode = New TreeNode

aNode.Name = "Properties"

aNode.Text = "Properties"

TreeView1.Nodes.Add(aNode)

For Each myPropertyInfo In dtProperties

aSubNode = New TreeNode

aSubNode.Name = myPropertyInfo.Name

aSubNode.Text = myPropertyInfo.Name

TreeView1.Nodes(aNode.Name).Nodes.Add(aSubNode)

Next

End If

Catch ex As Exception

End Try

'Finally, look for interfaces

Try

dtInterfaces = myType.GetInterfaces()

If dtInterfaces.length > 0 Then

aNode = New TreeNode

aNode.Name = "Interfaces"

aNode.Text = "Interfaces"

TreeView1.Nodes.Add(aNode)

For Each myInterfaceInfo In dtInterfaces

aSubNode = New TreeNode

aSubNode.Name = myInterfaceInfo.Name

aSubNode.Text = myInterfaceInfo.Name

TreeView1.Nodes(aNode.Name).Nodes.Add(aSubNode)

Next

End If

Catch ex As Exception

End Try

End Sub

Private Sub TreeView1\_DoubleClick(sender As Object, e As EventArgs)

Handles TreeView1.DoubleClick

'These have the same purpose as they did above -- if the user clicks

'on a particular category, we need to pull the details

Dim myType As Type = Type.GetType(TextBox1.Text)

Dim dtMethods() As MethodInfo

Dim dtFields() As FieldInfo

Dim dtProperties() As PropertyInfo

Dim dtInterfaces

'This will let us put a new item into the ListView

Dim aListViewItem As ListViewItem

'We need to be able to determine which node in the TreeView

'was double clicked on in order to pull the right "category" of

'information for the ListView

Dim theNode As New TreeNode

'We want to know what category a ListView item belongs to

Dim strCategory As String

'So when the user double clicks on a TreeView node...

'grab a copy of it

theNode = TreeView1.SelectedNode

'Make sure that the user clicked on something in the TreeView

If theNode IsNot Nothing Then

'If they did -- they may have double clicked on a category

'or on an item -- we want the category so check to see if there

'is a parent; if so, we want the parent's name...

If theNode.Parent IsNot Nothing Then

strCategory = theNode.Parent.Name

Else

'...otherwise we can directly pull the category name

strCategory = theNode.Name

End If

'Now set up the ListView with the correct details

Select Case strCategory

Case "Methods"

'Clear out everything in the ListView

ListView1.Items.Clear()

ListView1.Columns.Clear()

'Build up the column headings

ListView1.Columns.Add("Name", 100, HorizontalAlignment.Center)

ListView1.Columns.Add("Is Abstract?", 100,

HorizontalAlignment.Center)

ListView1.Columns.Add("Is Constructor?", 100,

HorizontalAlignment.Center)

ListView1.Columns.Add("Is Final?", 100, HorizontalAlignment.Center)

ListView1.Columns.Add("Is Generic?", 100, HorizontalAlignment.Center)

ListView1.Columns.Add("Is Private?", 100, HorizontalAlignment.Center)

ListView1.Columns.Add("Is Public?", 100, HorizontalAlignment.Center)

ListView1.Columns.Add("Is Static?", 100, HorizontalAlignment.Center)

ListView1.Columns.Add("Is Virtual?", 100, HorizontalAlignment.Center)

ListView1.Columns.Add("Return Type", 100, HorizontalAlignment.Center)

'Pull the methods again, and now get the details

dtMethods = myType.GetMethods()

For Each myMethod In dtMethods

aListViewItem = New ListViewItem

'The first item added ends up in the first column

aListViewItem.Text = myMethod.Name

ListView1.Items.Add(aListViewItem)

'All other columns are added as subitems to the first

aListViewItem.SubItems.Add(myMethod.IsAbstract)

aListViewItem.SubItems.Add(myMethod.IsConstructor)

aListViewItem.SubItems.Add(myMethod.IsFinal)

aListViewItem.SubItems.Add(myMethod.IsGenericMethod)

aListViewItem.SubItems.Add(myMethod.IsPrivate)

aListViewItem.SubItems.Add(myMethod.IsPublic)

aListViewItem.SubItems.Add(myMethod.IsStatic)

aListViewItem.SubItems.Add(myMethod.IsVirtual)

aListViewItem.SubItems.Add(myMethod.ReturnType.ToString)

Next

'Repeat the logic for fields...

Case "Fields"

ListView1.Items.Clear()

ListView1.Columns.Clear()

ListView1.Columns.Add("Name", 100, HorizontalAlignment.Center)

ListView1.Columns.Add("Field Type", 100, HorizontalAlignment.Center)

ListView1.Columns.Add("Is Literal?", 100, HorizontalAlignment.Center)

dtFields = myType.GetFields

For Each myFieldInfo In dtFields

aListViewItem = New ListViewItem

aListViewItem.Text = myFieldInfo.Name

ListView1.Items.Add(aListViewItem)

aListViewItem.SubItems.Add(myFieldInfo.FieldType.ToString)

aListViewItem.SubItems.Add(myFieldInfo.IsLiteral)

Next

'...properties

Case "Properties"

ListView1.Items.Clear()

ListView1.Columns.Clear()

ListView1.Columns.Add("Name", 100, HorizontalAlignment.Center)

ListView1.Columns.Add("Can Read?", 100, HorizontalAlignment.Center)

ListView1.Columns.Add("Can Write?", 100, HorizontalAlignment.Center)

ListView1.Columns.Add("Type", 100, HorizontalAlignment.Center)

dtProperties = myType.GetProperties

For Each myPropertyInfo In dtProperties

aListViewItem = New ListViewItem

aListViewItem.Text = myPropertyInfo.Name

ListView1.Items.Add(aListViewItem)

aListViewItem.SubItems.Add(myPropertyInfo.CanRead)

aListViewItem.SubItems.Add(myPropertyInfo.CanWrite)

aListViewItem.SubItems.Add(myPropertyInfo.PropertyType.ToString)

Next

'...and finally, interfaces

Case "Interfaces"

ListView1.Items.Clear()

ListView1.Columns.Clear()

ListView1.Columns.Add("Name", 100, HorizontalAlignment.Center)

dtInterfaces = myType.GetInterfaces()

For Each myInterfaceInfo In dtInterfaces

aListViewItem = New ListViewItem

aListViewItem.Text = myInterfaceInfo.name

ListView1.Items.Add(aListViewItem)

Next

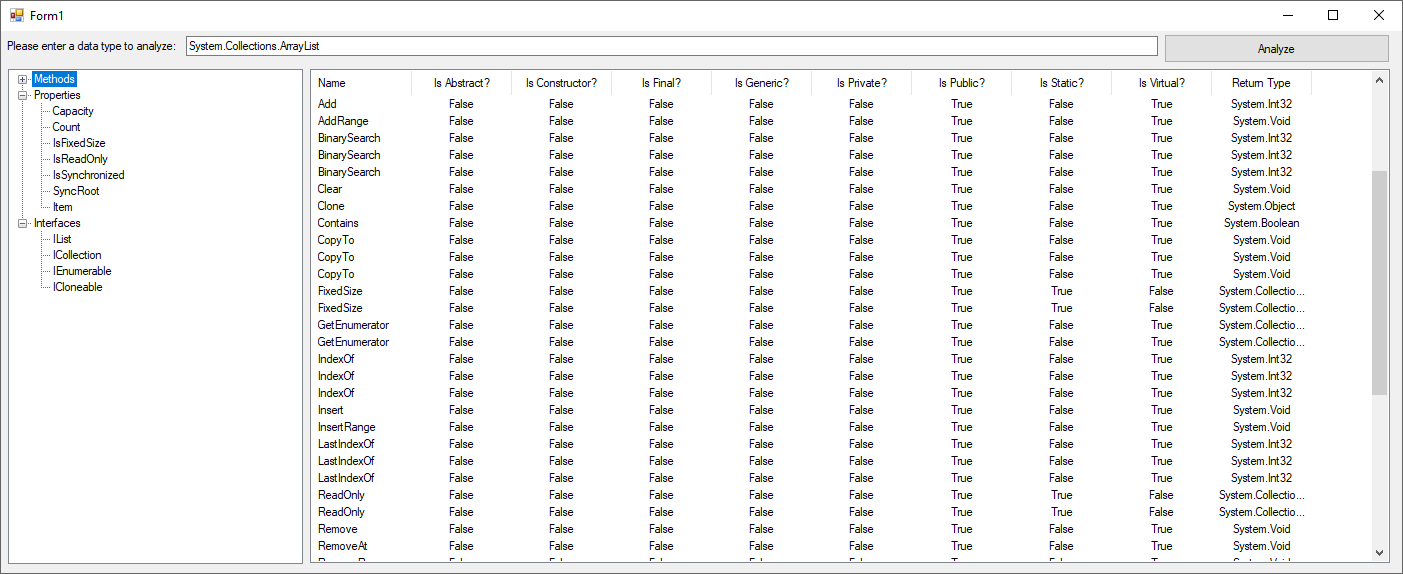
End Select

End If

End Sub

End Class

When we execute the program, type in a class name and press the Analyze button, we can explore what’s in a class using reflection:



Pretty neat, eh? This idea of reflection is not really that new – databases have had the ability to share metadata regarding their schema for a long time. This same idea was just extended to programming languages and we’re now able to get an application to tell us about what makes it up.

I added the ListView and TreeView controls here because they were really appropriate for the project due to the complexity of the detail data and the tree-like structure of the high level components making up a class. Neither control is overly difficult to use and if you pay close attention to the form you get the idea of what’s going on. We’ll end this section with an examination of these two controls.

Here’s the relevant TreeView properties:

|  |  |
| --- | --- |
| *Property* | *Purpose* |
| [CheckBoxes](http://msdn.microsoft.com/en-us/library/system.windows.forms.treeview.checkboxes.aspx) | Gets or sets a value indicating whether check boxes are displayed next to the nodes in the TreeView control. |
| [FullRowSelect](http://msdn.microsoft.com/en-us/library/system.windows.forms.treeview.fullrowselect.aspx) | Gets or sets a value indicating whether the selection highlight spans the width of the TreeView control. |
| [HideSelection](http://msdn.microsoft.com/en-us/library/system.windows.forms.treeview.hideselection.aspx) | Gets or sets a value indicating whether the selected tree node remains highlighted even when the TreeView has lost the focus. |
| [HotTracking](http://msdn.microsoft.com/en-us/library/system.windows.forms.treeview.hottracking.aspx) | Gets or sets a value indicating whether a tree node label takes on the appearance of a hyperlink as the mouse pointer passes over it. |
| [ImageIndex](http://msdn.microsoft.com/en-us/library/system.windows.forms.treeview.imageindex.aspx) | Gets or sets the image-list index value of the default image that is displayed by the tree nodes. |
| [ImageKey](http://msdn.microsoft.com/en-us/library/system.windows.forms.treeview.imagekey.aspx) | Gets or sets the key of the default image for each node in the TreeView control when it is in an unselected state. |
| [ImageList](http://msdn.microsoft.com/en-us/library/system.windows.forms.treeview.imagelist.aspx) | Gets or sets the [ImageList](http://msdn.microsoft.com/en-us/library/system.windows.forms.imagelist.aspx) that contains the [Image](http://msdn.microsoft.com/en-us/library/system.drawing.image.aspx) objects that are used by the tree nodes. |
| [Indent](http://msdn.microsoft.com/en-us/library/system.windows.forms.treeview.indent.aspx) | Gets or sets the distance to indent each child tree node level. |
| [ItemHeight](http://msdn.microsoft.com/en-us/library/system.windows.forms.treeview.itemheight.aspx) | Gets or sets the height of each tree node in the tree view control. |
| [LabelEdit](http://msdn.microsoft.com/en-us/library/system.windows.forms.treeview.labeledit.aspx) | Gets or sets a value indicating whether the label text of the tree nodes can be edited. |
| [LineColor](http://msdn.microsoft.com/en-us/library/system.windows.forms.treeview.linecolor.aspx) | Gets or sets the color of the lines connecting the nodes of the TreeView control. |
| [Nodes](http://msdn.microsoft.com/en-us/library/system.windows.forms.treeview.nodes.aspx) | Gets the collection of tree nodes that are assigned to the TreeView control. |
| [Padding](http://msdn.microsoft.com/en-us/library/system.windows.forms.treeview.padding.aspx) | Gets or sets the spacing between the TreeView control's contents and its edges. |
| [PathSeparator](http://msdn.microsoft.com/en-us/library/system.windows.forms.treeview.pathseparator.aspx) | Gets or sets the delimiter string that the tree node path uses. |
| [Scrollable](http://msdn.microsoft.com/en-us/library/system.windows.forms.treeview.scrollable.aspx) | Gets or sets a value indicating whether the TreeView control displays scroll bars when they are needed. |
| [SelectedImageIndex](http://msdn.microsoft.com/en-us/library/system.windows.forms.treeview.selectedimageindex.aspx) | Gets or sets the image list index value of the image that is displayed when a tree node is selected. |
| [SelectedImageKey](http://msdn.microsoft.com/en-us/library/system.windows.forms.treeview.selectedimagekey.aspx) | Gets or sets the key of the default image shown when a [TreeNode](http://msdn.microsoft.com/en-us/library/system.windows.forms.treenode.aspx) is in a selected state. |
| [SelectedNode](http://msdn.microsoft.com/en-us/library/system.windows.forms.treeview.selectednode.aspx) | Gets or sets the tree node that is currently selected in the TreeView control. |
| [ShowLines](http://msdn.microsoft.com/en-us/library/system.windows.forms.treeview.showlines.aspx) | Gets or sets a value indicating whether lines are drawn between tree nodes in the TreeView control. |
| [ShowNodeToolTips](http://msdn.microsoft.com/en-us/library/system.windows.forms.treeview.shownodetooltips.aspx) | Gets or sets a value indicating ToolTips are shown when the mouse pointer hovers over a [TreeNode](http://msdn.microsoft.com/en-us/library/system.windows.forms.treenode.aspx). |
| [ShowPlusMinus](http://msdn.microsoft.com/en-us/library/system.windows.forms.treeview.showplusminus.aspx) | Gets or sets a value indicating whether plus-sign (+) and minus-sign (-) buttons are displayed next to tree nodes that contain child tree nodes. |
| [ShowRootLines](http://msdn.microsoft.com/en-us/library/system.windows.forms.treeview.showrootlines.aspx) | Gets or sets a value indicating whether lines are drawn between the tree nodes that are at the root of the TreeView. |
| [Sorted](http://msdn.microsoft.com/en-us/library/system.windows.forms.treeview.sorted.aspx) | Gets or sets a value indicating whether the tree nodes in the TreeView are sorted. |
| [StateImageList](http://msdn.microsoft.com/en-us/library/system.windows.forms.treeview.stateimagelist.aspx) | Gets or sets the image list that is used to indicate the state of the TreeView and its nodes. |
| [Text](http://msdn.microsoft.com/en-us/library/w4c3zsc1.aspx) | Gets or sets the text of the TreeView. |
| [TopNode](http://msdn.microsoft.com/en-us/library/system.windows.forms.treeview.topnode.aspx) | Gets or sets the first fully-visible tree node in the TreeView control. |
| [TreeViewNodeSorter](http://msdn.microsoft.com/en-us/library/system.windows.forms.treeview.treeviewnodesorter.aspx) | Gets or sets the implementation of [IComparer](http://msdn.microsoft.com/en-us/library/system.collections.icomparer.aspx) to perform a custom sort of the TreeView nodes. |
| [VisibleCount](http://msdn.microsoft.com/en-us/library/system.windows.forms.treeview.visiblecount.aspx) | Gets the number of tree nodes that can be fully visible in the TreeView control. |

Here’s the relevant TreeView methods:

|  |  |
| --- | --- |
| *Method* | *Purpose* |
| [BeginUpdate](http://msdn.microsoft.com/en-us/library/system.windows.forms.treeview.beginupdate.aspx) | Disables any redrawing of the TreeView. |
| [CollapseAll](http://msdn.microsoft.com/en-us/library/system.windows.forms.treeview.collapseall.aspx) | Collapses all the tree nodes. |
| [EndUpdate](http://msdn.microsoft.com/en-us/library/system.windows.forms.treeview.endupdate.aspx) | Enables the redrawing of the TreeView. |
| [ExpandAll](http://msdn.microsoft.com/en-us/library/system.windows.forms.treeview.expandall.aspx) | Expands all the tree nodes. |
| [GetNodeAt(Point)](http://msdn.microsoft.com/en-us/library/1yxbz43s.aspx) | Retrieves the tree node that is at the specified point. |
| [GetNodeCount](http://msdn.microsoft.com/en-us/library/system.windows.forms.treeview.getnodecount.aspx) | Retrieves the number of tree nodes, optionally including those in all subtrees, assigned to the TreeView control. |
| [HitTest(Point)](http://msdn.microsoft.com/en-us/library/z6sb88bz.aspx) | Provides node information, given a point. |
| [Sort](http://msdn.microsoft.com/en-us/library/system.windows.forms.treeview.sort.aspx) | Sorts the items in TreeView control. |

These are the commonly used ListView properties:

|  |  |
| --- | --- |
| *Property* | *Purpose* |
| [AllowColumnReorder](http://msdn.microsoft.com/en-us/library/system.windows.forms.listview.allowcolumnreorder.aspx) | Gets or sets a value indicating whether the user can drag column headers to reorder columns in the control. |
| [AutoArrange](http://msdn.microsoft.com/en-us/library/system.windows.forms.listview.autoarrange.aspx) | Gets or sets whether icons are automatically kept arranged. |
| [CheckBoxes](http://msdn.microsoft.com/en-us/library/system.windows.forms.listview.checkboxes.aspx) | Gets or sets a value indicating whether a check box appears next to each item in the control. |
| [CheckedIndices](http://msdn.microsoft.com/en-us/library/system.windows.forms.listview.checkedindices.aspx) | Gets the indexes of the currently checked items in the control. |
| [CheckedItems](http://msdn.microsoft.com/en-us/library/system.windows.forms.listview.checkeditems.aspx) | Gets the currently checked items in the control. |
| [Columns](http://msdn.microsoft.com/en-us/library/system.windows.forms.listview.columns.aspx) | Gets the collection of all column headers that appear in the control. |
| [FullRowSelect](http://msdn.microsoft.com/en-us/library/system.windows.forms.listview.fullrowselect.aspx) | Gets or sets a value indicating whether clicking an item selects all its subitems. |
| [GridLines](http://msdn.microsoft.com/en-us/library/system.windows.forms.listview.gridlines.aspx) | Gets or sets a value indicating whether grid lines appear between the rows and columns containing the items and subitems in the control. |
| [Groups](http://msdn.microsoft.com/en-us/library/system.windows.forms.listview.groups.aspx) | Gets the collection of [ListViewGroup](http://msdn.microsoft.com/en-us/library/system.windows.forms.listviewgroup.aspx) objects assigned to the control. |
| [HeaderStyle](http://msdn.microsoft.com/en-us/library/system.windows.forms.listview.headerstyle.aspx) | Gets or sets the column header style. |
| [HotTracking](http://msdn.microsoft.com/en-us/library/system.windows.forms.listview.hottracking.aspx) | Gets or sets a value indicating whether the text of an item or subitem has the appearance of a hyperlink when the mouse pointer passes over it. |
| [HoverSelection](http://msdn.microsoft.com/en-us/library/system.windows.forms.listview.hoverselection.aspx) | Gets or sets a value indicating whether an item is automatically selected when the mouse pointer remains over the item for a few seconds. |
| [InsertionMark](http://msdn.microsoft.com/en-us/library/system.windows.forms.listview.insertionmark.aspx) | Gets an object used to indicate the expected drop location when an item is dragged within a ListView control. |
| [Items](http://msdn.microsoft.com/en-us/library/system.windows.forms.listview.items.aspx) | Gets a collection containing all items in the control. |
| [LabelEdit](http://msdn.microsoft.com/en-us/library/system.windows.forms.listview.labeledit.aspx) | Gets or sets a value indicating whether the user can edit the labels of items in the control. |
| [LabelWrap](http://msdn.microsoft.com/en-us/library/system.windows.forms.listview.labelwrap.aspx) | Gets or sets a value indicating whether item labels wrap when items are displayed in the control as icons. |
| [LargeImageList](http://msdn.microsoft.com/en-us/library/system.windows.forms.listview.largeimagelist.aspx) | Gets or sets the [ImageList](http://msdn.microsoft.com/en-us/library/system.windows.forms.imagelist.aspx) to use when displaying items as large icons in the control. |
| [ListViewItemSorter](http://msdn.microsoft.com/en-us/library/system.windows.forms.listview.listviewitemsorter.aspx) | Gets or sets the sorting comparer for the control. |
| [MultiSelect](http://msdn.microsoft.com/en-us/library/system.windows.forms.listview.multiselect.aspx) | Gets or sets a value indicating whether multiple items can be selected. |
| [Scrollable](http://msdn.microsoft.com/en-us/library/system.windows.forms.listview.scrollable.aspx) | Gets or sets a value indicating whether a scroll bar is added to the control when there is not enough room to display all items. |
| [SelectedIndices](http://msdn.microsoft.com/en-us/library/system.windows.forms.listview.selectedindices.aspx) | Gets the indexes of the selected items in the control. |
| [SelectedItems](http://msdn.microsoft.com/en-us/library/system.windows.forms.listview.selecteditems.aspx) | Gets the items that are selected in the control. |
| [ShowGroups](http://msdn.microsoft.com/en-us/library/system.windows.forms.listview.showgroups.aspx) | Gets or sets a value indicating whether items are displayed in groups. |
| [ShowItemToolTips](http://msdn.microsoft.com/en-us/library/system.windows.forms.listview.showitemtooltips.aspx) | Gets or sets a value indicating whether ToolTips are shown for the [ListViewItem](http://msdn.microsoft.com/en-us/library/system.windows.forms.listviewitem.aspx) objects contained in theListView. |
| [SmallImageList](http://msdn.microsoft.com/en-us/library/system.windows.forms.listview.smallimagelist.aspx) | Gets or sets the [ImageList](http://msdn.microsoft.com/en-us/library/system.windows.forms.imagelist.aspx) to use when displaying items as small icons in the control. |
| [Sorting](http://msdn.microsoft.com/en-us/library/system.windows.forms.listview.sorting.aspx) | Gets or sets the sort order for items in the control. |
| [StateImageList](http://msdn.microsoft.com/en-us/library/system.windows.forms.listview.stateimagelist.aspx) | Gets or sets the [ImageList](http://msdn.microsoft.com/en-us/library/system.windows.forms.imagelist.aspx) associated with application-defined states in the control. |
| [TileSize](http://msdn.microsoft.com/en-us/library/system.windows.forms.listview.tilesize.aspx) | Gets or sets the size of the tiles shown in tile view. |
| [TopItem](http://msdn.microsoft.com/en-us/library/system.windows.forms.listview.topitem.aspx) | Gets or sets the first visible item in the control. |
| [View](http://msdn.microsoft.com/en-us/library/system.windows.forms.listview.view.aspx) | Gets or sets how items are displayed in the control. |
| [VirtualListSize](http://msdn.microsoft.com/en-us/library/system.windows.forms.listview.virtuallistsize.aspx) | Gets or sets the number of [ListViewItem](http://msdn.microsoft.com/en-us/library/system.windows.forms.listviewitem.aspx) objects contained in the list when in virtual mode. |
| [VirtualMode](http://msdn.microsoft.com/en-us/library/system.windows.forms.listview.virtualmode.aspx) | Gets or sets a value indicating whether you have provided your own data-management operations for the ListView control. |

Here’s the relevant listview methods:

|  |  |
| --- | --- |
| *Method* | *Purpose* |
| [ArrangeIcons](http://msdn.microsoft.com/en-us/library/6t4c7x4t.aspx) | Arranges items in the control when they are displayed as icons based on the value of the [Alignment](http://msdn.microsoft.com/en-us/library/system.windows.forms.listview.alignment.aspx) property. |
| [AutoResizeColumn](http://msdn.microsoft.com/en-us/library/system.windows.forms.listview.autoresizecolumn.aspx) | Resizes the width of the given column as indicated by the resize style. |
| [AutoResizeColumns](http://msdn.microsoft.com/en-us/library/system.windows.forms.listview.autoresizecolumns.aspx) | Resizes the width of the columns as indicated by the resize style. |
| [BeginUpdate](http://msdn.microsoft.com/en-us/library/system.windows.forms.listview.beginupdate.aspx) | Prevents the control from drawing until the [EndUpdate](http://msdn.microsoft.com/en-us/library/system.windows.forms.listview.endupdate.aspx) method is called. |
| [Clear](http://msdn.microsoft.com/en-us/library/system.windows.forms.listview.clear.aspx) | Removes all items and columns from the control. |
| [EndUpdate](http://msdn.microsoft.com/en-us/library/system.windows.forms.listview.endupdate.aspx) | Resumes drawing of the ListView control after drawing is suspended by the [BeginUpdate](http://msdn.microsoft.com/en-us/library/system.windows.forms.listview.beginupdate.aspx) method. |
| [EnsureVisible](http://msdn.microsoft.com/en-us/library/system.windows.forms.listview.ensurevisible.aspx) | Ensures that the specified item is visible within the control, scrolling the contents of the control if necessary. |
| [FindItemWithText(String)](http://msdn.microsoft.com/en-us/library/y3h4x385.aspx) | Finds the first [ListViewItem](http://msdn.microsoft.com/en-us/library/system.windows.forms.listviewitem.aspx) that begins with the specified text value. |
| [FindNearestItem(SearchDirectionHint, Point)](http://msdn.microsoft.com/en-us/library/sbk6x183.aspx) | Finds the next item from the given point, searching in the specified direction. |
| [GetChildAtPoint(Point)](http://msdn.microsoft.com/en-us/library/a6zktd23.aspx) | Retrieves the child control that is located at the specified coordinates. |
| [GetItemAt](http://msdn.microsoft.com/en-us/library/system.windows.forms.listview.getitemat.aspx) | Retrieves the item at the specified location. |
| [GetItemRect(Int32)](http://msdn.microsoft.com/en-us/library/03sahk6t.aspx) | Retrieves the bounding rectangle for a specific item within the ListView control. |
| [HitTest(Point)](http://msdn.microsoft.com/en-us/library/w8ssfa8a.aspx) | Provides item information, given a point. |
| [RedrawItems](http://msdn.microsoft.com/en-us/library/system.windows.forms.listview.redrawitems.aspx) | Forces a range of [ListViewItem](http://msdn.microsoft.com/en-us/library/system.windows.forms.listviewitem.aspx) objects to be redrawn. |
| [Sort](http://msdn.microsoft.com/en-us/library/system.windows.forms.listview.sort.aspx) | Sorts the items of the ListView. |
| [UpdateExtendedStyles](http://msdn.microsoft.com/en-us/library/system.windows.forms.listview.updateextendedstyles.aspx) | Updates the extended styles applied to the ListView control. |