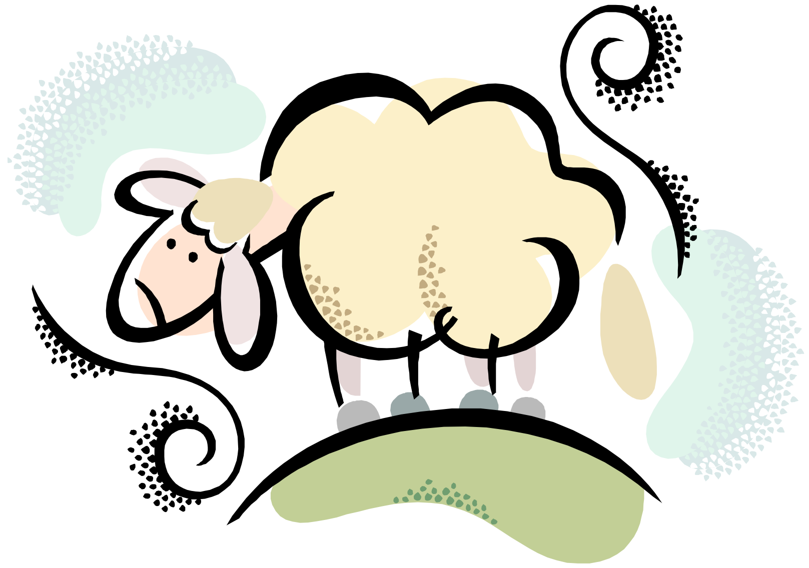
sheepsqueezers.com



# C# Programming VI: *Microsoft Visual C# 2010 Express Edition*

## Legal Stuff

sheepsqueezers.com

# .NET Lecture Series

sheepsqueezers.com

C#
Programming I:
Concepts of OOP

C#
Programming II:
Beginning C#

C#
Programming III:
Advanced C#

C#
Programming IV-1:
System
Namespace

C#
Programming IV-2:
System.Collections
Namespace

C#
Programming IV-3:
System.Collections.
Generic
Namespace

C#
Programming IV-4A:
System.Data
Namespace

C#
Programming IV-4B:
System.Data.Odbc
Namespace

C#
Programming IV-4C:
System.Data.OleDb
Namespace

C#
Programming IV-4D:
Oracle.DataAccess.Client
Namespace

C#
Programming IV-4E:
System.Data.SqlClient
Namespace

C#
Programming IV-4F:
System.Data.SqlTypes
Namespace

C#
Programming IV-5:
System.Drawing/(2D)
Namespace

C#
Programming IV-6:
System.IO
Namespace

C#
Programming IV-7:
System.Numerics

C#
Programming IV-8:
System.Text and
System.Text.
RegularExpressions
Namespaces

C#
Programming V:
Introduction
to LINQ

**C#
Programming VI:
Visual C# 2010
Express Edition**

C#
Self-
Inflicted
Project #1
Address
Cleaning

C#
Self-
Inflicted
Project #2
Large
Intersection
Problem

# Charting Our Course

## What is Visual C# 2010 Express Edition?

Visual C# 2010 Express Edition is an integrated development environment (IDE) used to program .NET Framework 4 graphical user interface (GUI) applications, console applications, and a variety of other program types such as services, Silverlight applications, Windows Phone applications, XBOX games, etc. without having to resort to the command line. This allows you to create rich applications using familiar controls such as the button control, tree control, grid control, etc.

In addition to the controls offered by Microsoft, several companies create their own controls which you can purchase and use in your applications. Companies such as Infragistics and DevExpress offer windows forms controls as well as controls for ASP.NET and more.

Visual C# 2010 Express Edition offers several features to ease the burden of programming such as IntelliSense, codesearch as well as tools to aid in debugging.

Unlike other Visual Studio products, the Express Edition is free, but is limited in its functionality as compared to purchased versions. You can view a chart of feature comparisons here: http://www.microsoft.com/visualstudio/en-us/products/2010-editions/product-comparison#expresscomparetable.

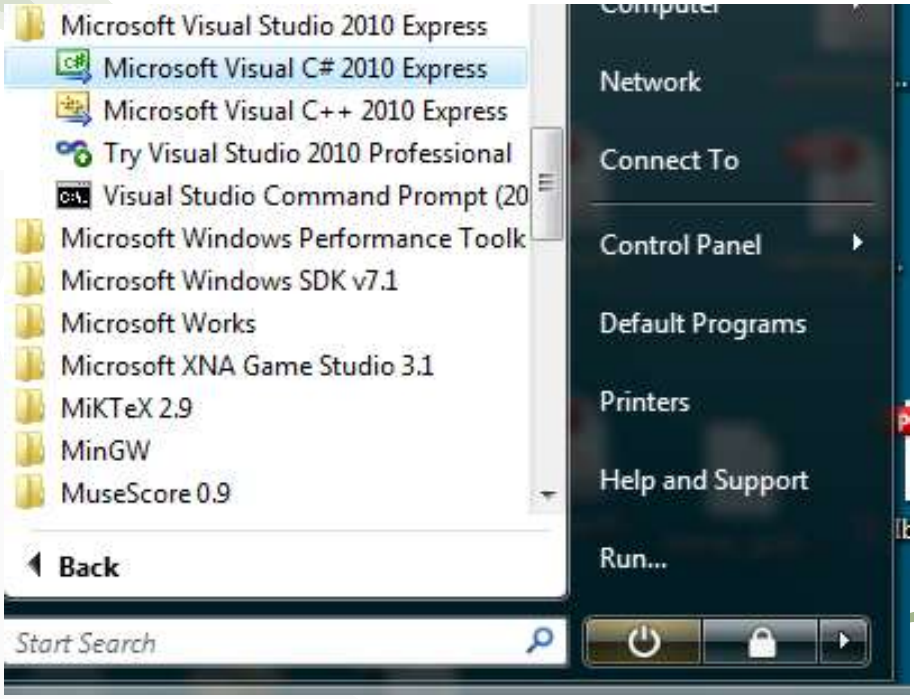
Regardless of its limitations, Visual C# 2010 Express Edition is a great tool to use to learn how to program in C#.

## Installing Visual C# 2010 Express Edition

In order to program C# using Visual C# 2010 Express Edition…well…you're going to have to download it and install it! ☺

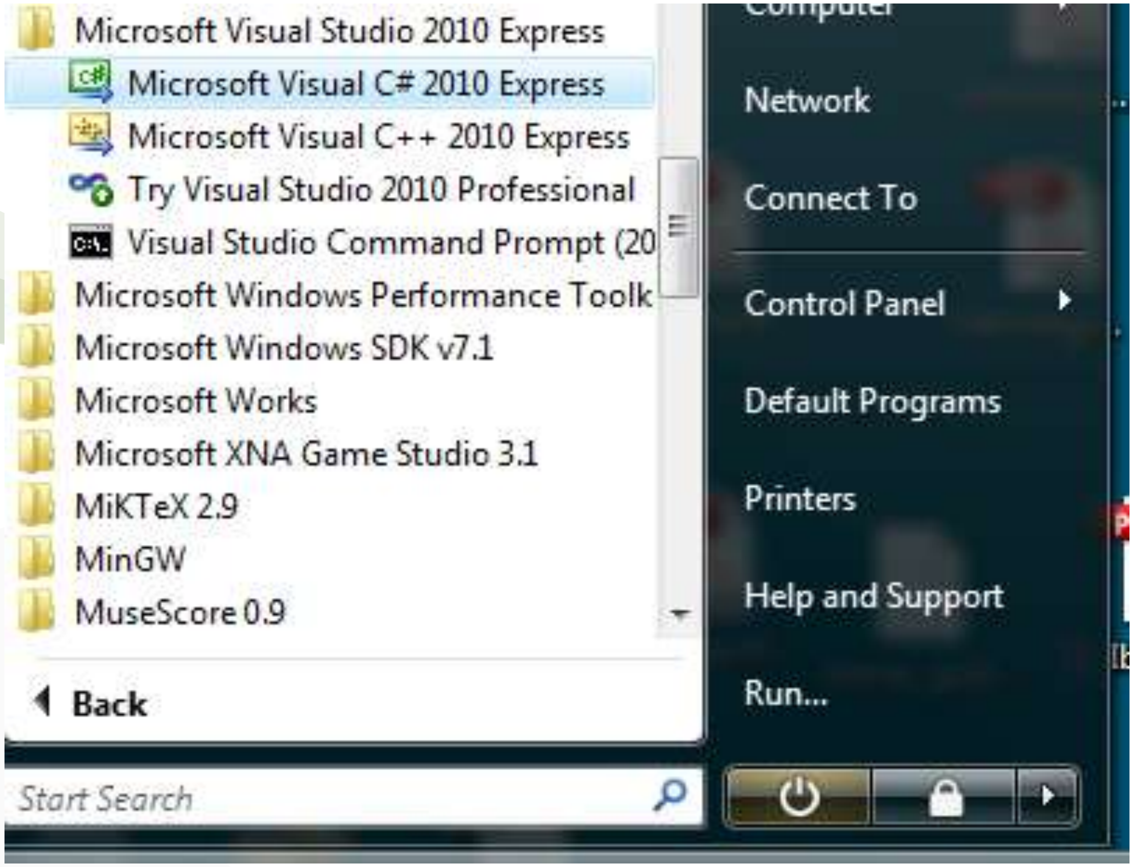
Please see the Pre-Requisites section in the *C# Programming II* presentation for instructions on how to download and install Microsoft Visual C# 2010 Express Edition. This presentation is located in the .NET Section in the Slidedecks folder on the www.sheepsqueezers.com website. (Please also install Microsoft Visual C++ 2010 Express Edition as well.)

To start Visual C# 2010 Express Edition, click on Start…All Programs…Microsoft Visual Studio 2010 Express…Microsoft Visual C# 2010 Express. Once the program starts, you will see a screen similar to that on the next slide.

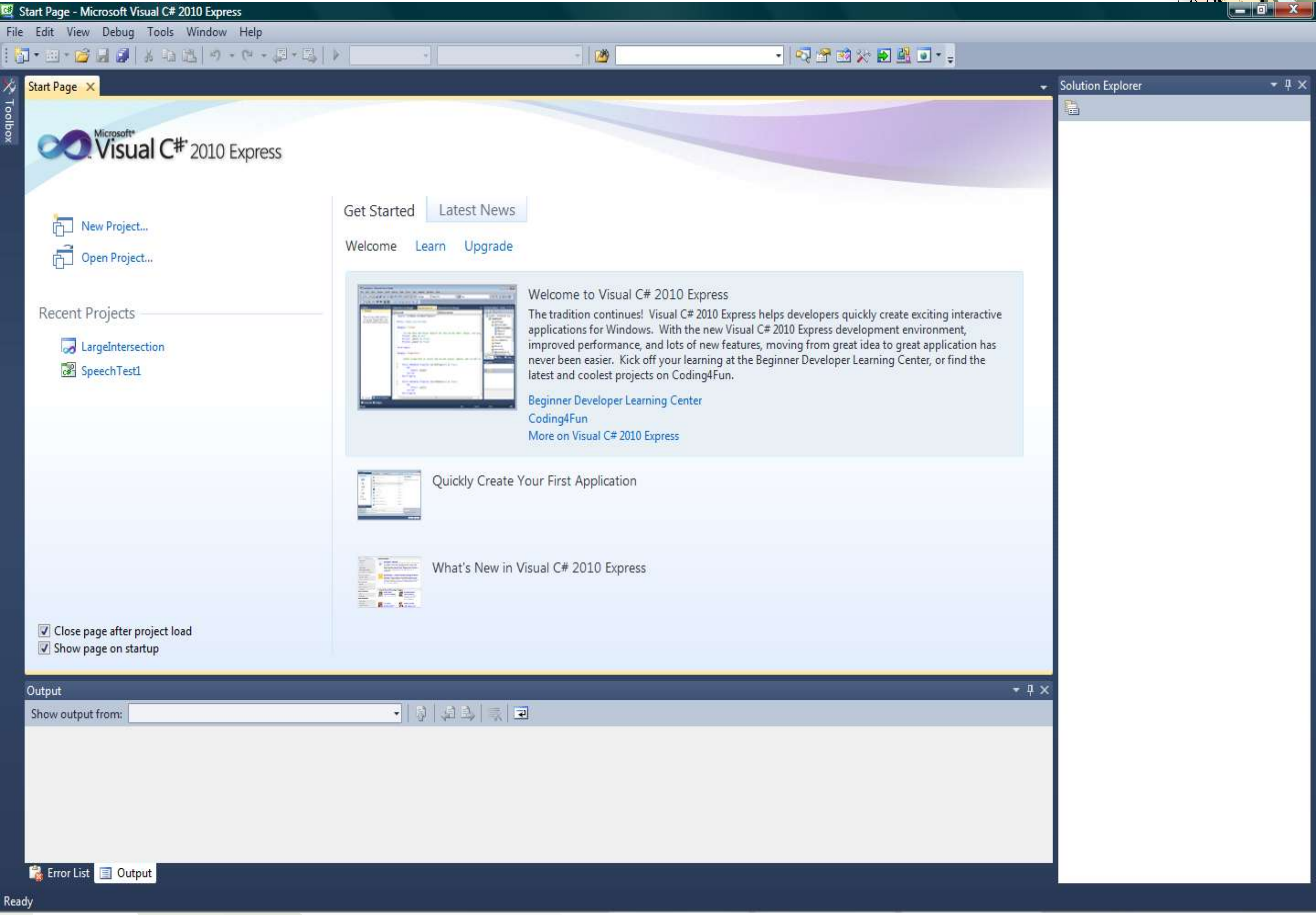

Microsoft Visual Studio 2010 Express
Microsoft Visual C# 2010 Express
Microsoft Visual C++ 2010 Express
Try Visual Studio 2010 Professional
Visual Studio Command Prompt (20
Microsoft Windows Performance Toolk
Microsoft Windows SDK v7.1
Microsoft Works
Microsoft XNA Game Studio 3.1
MiKTeX 2.9
MinGW
MuseScore 0.9
Back
Start Search
Computer
Network
Connect To
Control Panel
Default Programs
Printers
Help and Support
Run...

# Starting Visual C# 2010 Express Edition

To start Visual C# 2010 Express Edition, click on Start…All Programs…Microsoft Visual Studio 2010 Express…Microsoft Visual C# 2010 Express. Once the program starts, you will see a screen similar to that on the next slide (the Recent Projects section will be empty if you've never created a project with this software).


Microsoft Visual Studio 2010 Express
Microsoft Visual C# 2010 Express
Microsoft Visual C++ 2010 Express
Try Visual Studio 2010 Professional
Visual Studio Command Prompt (20
Microsoft Windows Performance Toolk
Microsoft Windows SDK v7.1
Microsoft Works
Microsoft XNA Game Studio 3.1
MiKTeX 2.9
MinGW
MuseScore 0.9
Back
Start Search
Network
Connect To
Control Panel
Default Programs
Printers
Help and Support
Run...

# Starting Visual C# 2010 Express Edition



Start Page - Microsoft Visual C# 2010 Express
File Edit View Debug Tools Window Help
Toolbox
Start Page
Solution Explorer
Microsoft Visual C# 2010 Express
New Project...
Open Project...
Recent Projects
LargeIntersection
SpeechTest1
Get Started
Latest News
Welcome
Learn
Upgrade
Beginner Developer Learning Center
Coding4Fun
More on Visual C# 2010 Express
Quickly Create Your First Application
What's New in Visual C# 2010 Express
Output
Show output from:
Error List
Output
Ready

Welcome to Visual C# 2010 Express

The tradition continues! Visual C# 2010 Express helps developers quickly create exciting interactive applications for Windows. With the new Visual C# 2010 Express development environment, improved performance, and lots of new features, moving from great idea to great application has never been easier. Kick off your learning at the Beginner Developer Learning Center, or find the latest and coolest projects on Coding4Fun.

☑ Close page after project load
☑ Show page on startup

## Starting Visual C# 2010 Express Edition

Be aware that, if you are attempting to create a stored procedure for SQL Server, Visual C# 2010 Express Edition does not come with the SQL Server Project Template. You will have to code your assemblies without the help of the template. For an example of how to do this, please see Appendix C/D in the document *Self-Inflicted C# Project #2: Large Intersection Problem* located in the .NET Section in the Documents folder on the www.sheepsqueezers.com website.

While on the subject of creating assemblies for use with SQL Sever, note that you need to use the version of Visual C# that corresponds with your SQL Server Database. That is, if you are running SQL Server 2005, then you need .NET 2 or below (Visual Studio 2005). If you are running SQL Server 2008, then you need .NET 3/3.5 or below (Visual Studio 2008). If you are running SQL Server 2010, you need .NET 4 or below (Visual Studio 2010). Attempting to use a more recent version of Visual C# will result in problems when you reference the assembly using the SQL `CREATE ASSEMBLY` statement in SQL Server Management Studio.

Now, on the IDE shown on the previous slide is broken up into several sections.

On the left is the Toolbox. Moving your mouse over the word Toolbox expands to show you a list of controls such as a button control, grid control, etc. Now, since we have not started a project, you won't see any controls yet.

On the right is the Solution Explorer. This shows you all of your projects, C# code, and additional files that support your project.

On the bottom, you will see two tabs: Error List and Output.

## Starting Visual C# 2010 Express Edition

The Error List shows you a list of errors (not that a tip-top programmer such as yourself gets any errors) when you compile your program.  Note that if you double-click on an error, you are brought to the exact location of the error in the code window.

The Output tab shows you a list of search results when you search for a word or phrase in your code.

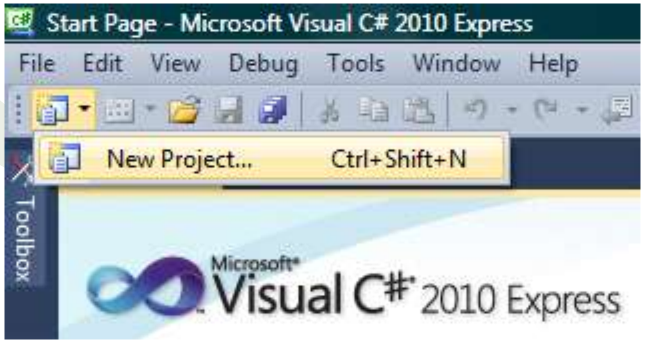
Now, let's move on to our first Visual C# project.

## Our First Visual C# Project

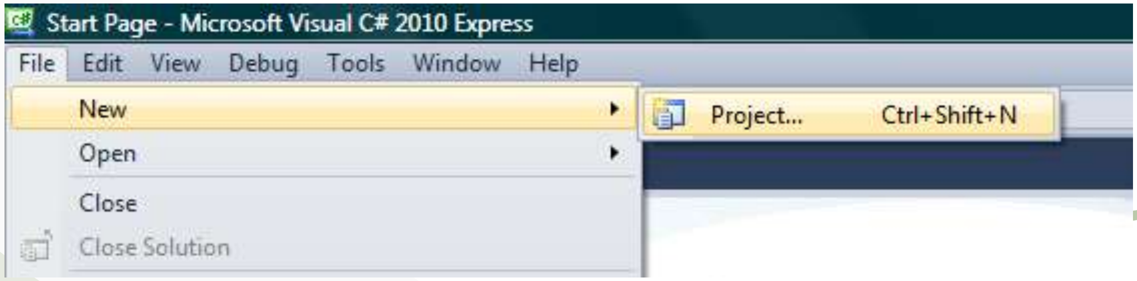
In this section, we create our first project. In all of the previous .NET presentations, we've been using the command line to compile the code, I thought that we should start with a console application, just for old time sake!

Open up Visual C# 2010 Express Edition. To create a new project, you can do one of three things:

1. Click on the New Project… link in the on the Start Page tab
2. Click on the New Project icon on the left of the toolbar:


Start Page - Microsoft Visual C# 2010 Express
File Edit View Debug Tools Window Help
New Project... Ctrl+Shift+N
Toolbox
Microsoft Visual C# 2010 Express

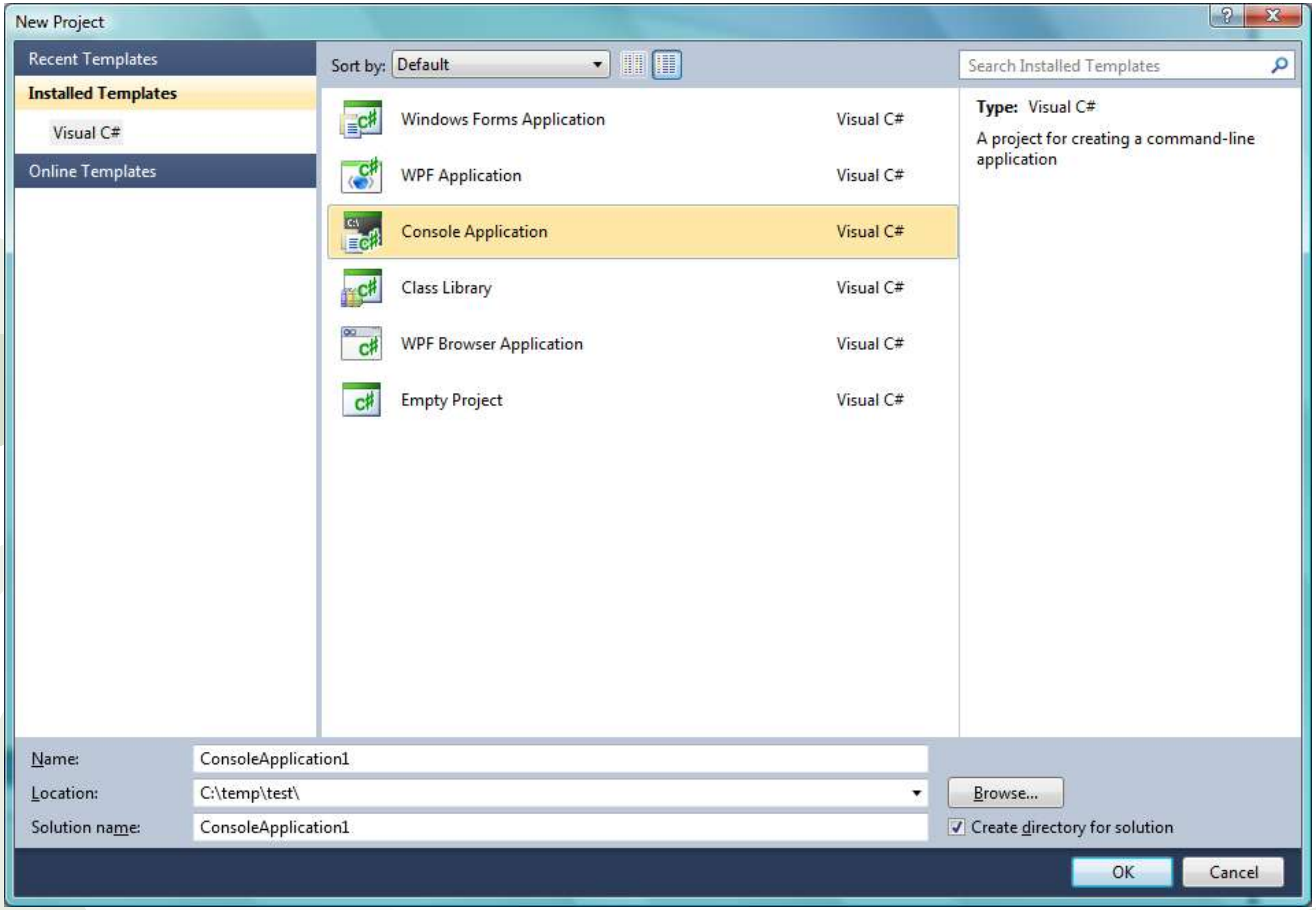
3. Click on the menu File → New → New Project…


Start Page - Microsoft Visual C# 2010 Express
File Edit View Debug Tools Window Help
New
Open
Close
Close Solution
Project... Ctrl+Shift+N

## Our First Visual C# Project

sheepsqueezers.com

Regardless of the method you choose, the New Project dialog box will be displayed, as shown below.



Windows Forms Application — Visual C#
WPF Application — Visual C#
Console Application — Visual C#
Class Library — Visual C#
WPF Browser Application — Visual C#
Empty Project — Visual C#

**Type:** Visual C#

A project for creating a command-line application

Name: ConsoleApplication1

Location: C:\temp\test\

Solution name: ConsoleApplication1

Browse...

[✓] Create directory for solution

OK Cancel

## Our First Visual C# Project

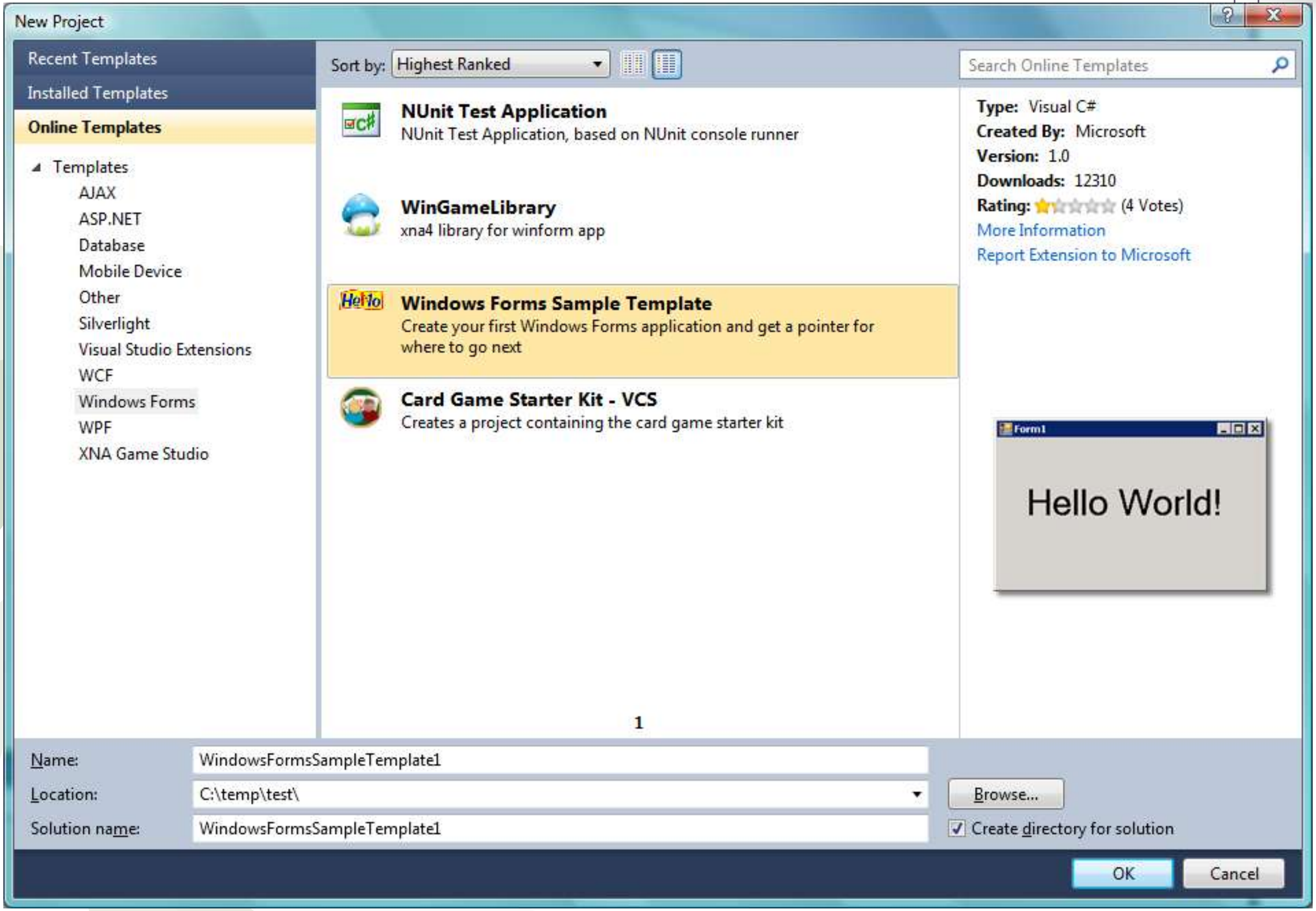
You'll note that there are several installed templates included to quickly help you create your project (unlike the daunting blank page shown by Notepad or TextPad we've been using up to now). The default installed templates are:

- ❑ Empty Project – This template is like the daunting blank page shown by…oh, you get it. You probably want to avoid this template unless you have big bollocks and a full prostate.
- ❑ Windows Forms Application – This is used to create a real-life honest-to-goodness graphical user interface (GUI) application just like the real programmers create. We'll talk about this later on in the presentation.
- ❑ Console Application – This is our familiar, yet somewhat dull, unmarried, and unable-to-find-love-in-the-big-city template used to create the applications just like the ones we've created in this lecture series up-to-now.
- ❑ Class Library – This template is used to create DLLs similar to the DLL we created in our *Self-Inflicted C# Project #2: Large Intersection Problem*.
- ❑ WPF Application/WPF Browser Application – Used with Windows Presentation Foundation…at this point I have absolutely no idea what that is…I'll have to come back to this later.

You also notice that you can click on the Online Templates link to see templates available from Microsoft. The next slide shows a few of those.

# Our First Visual C# Project

sheepsqueezers.com



New Project

Recent Templates
Installed Templates
**Online Templates**

Templates
- AJAX
- ASP.NET
- Database
- Mobile Device
- Other
- Silverlight
- Visual Studio Extensions
- WCF
- Windows Forms
- WPF
- XNA Game Studio

Sort by: Highest Ranked

Search Online Templates

**NUnit Test Application**
NUnit Test Application, based on NUnit console runner

**WinGameLibrary**
xna4 library for winform app

**Windows Forms Sample Template**
Create your first Windows Forms application and get a pointer for where to go next

**Card Game Starter Kit - VCS**
Creates a project containing the card game starter kit

1

**Type:** Visual C#
**Created By:** Microsoft
**Version:** 1.0
**Downloads:** 12310
**Rating:** ★☆☆☆☆ (4 Votes)
More Information
Report Extension to Microsoft

Name: WindowsFormsSampleTemplate1
Location: C:\temp\test\
Solution name: WindowsFormsSampleTemplate1

Browse...
✓ Create directory for solution

OK Cancel

# Our First Visual C# Project

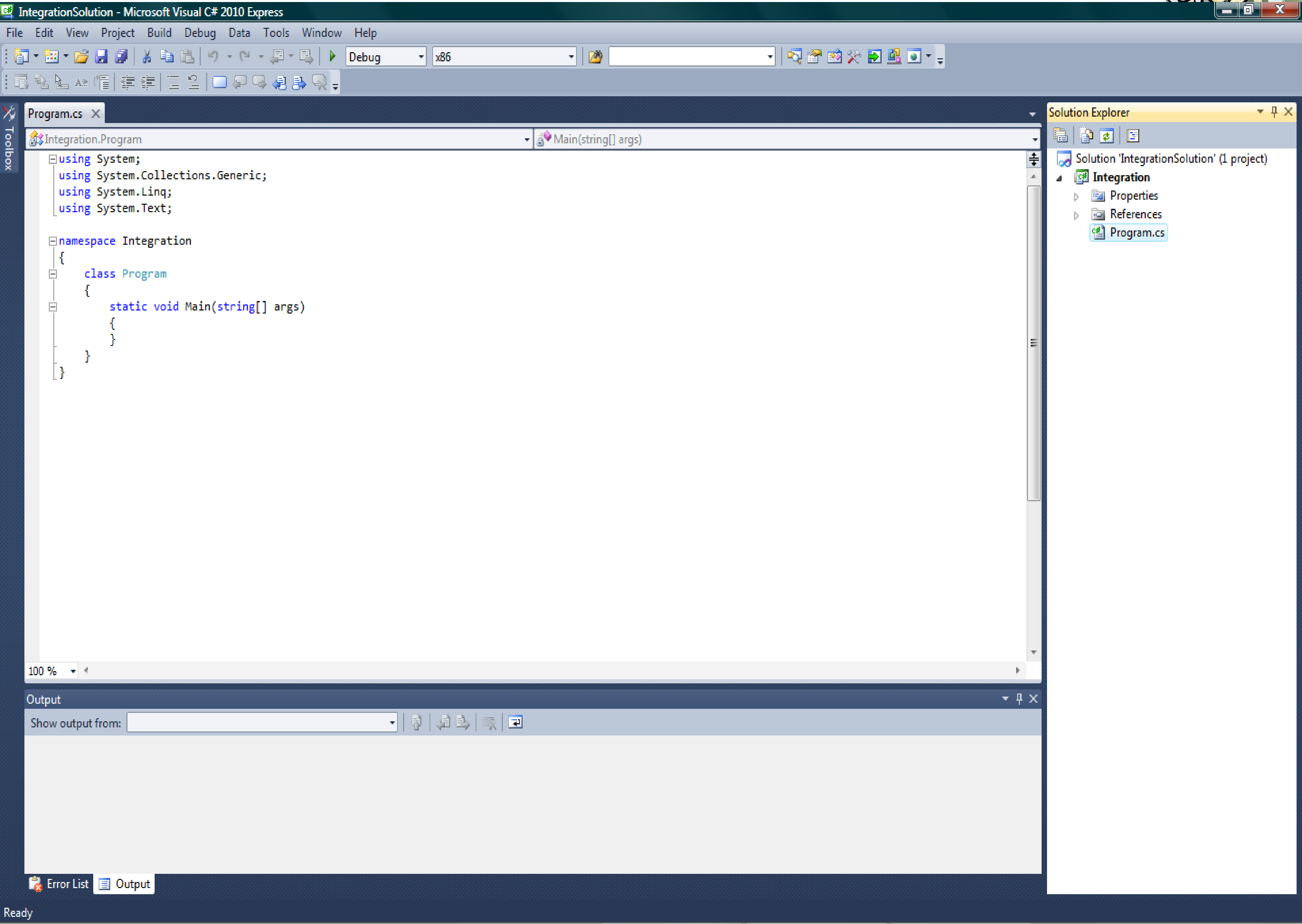
Now, click back on the Installed Templates link and then click on the Console Application template. You must fill in the input boxes shown at the bottom of the dialog box:

1. Name – This is the name of your project. Change this name to `Integration`.
2. Location – This is the location where your project will be stored on disk. As you see, my location is set to the `C:\temp\test` folder, but you can change this to any folder location you desire.
3. Solution Name – This is the name of the solution. A *solution* is a contain for one or more projects. Each project includes all of the code and extra little files you need such as images, XML files, etc. For now, change the Solution Name to `IntegrationSolution`.

You'll notice the checkbox to the right of the Solution Name input box. If this is checked, a separate directory is created with your solution name. Leave this box checked.

Next, click on the OK button to create your solution and project. Once Visual Studio finishes creating your project, you will see a screen similar to the following.

# Our First Visual C# Project



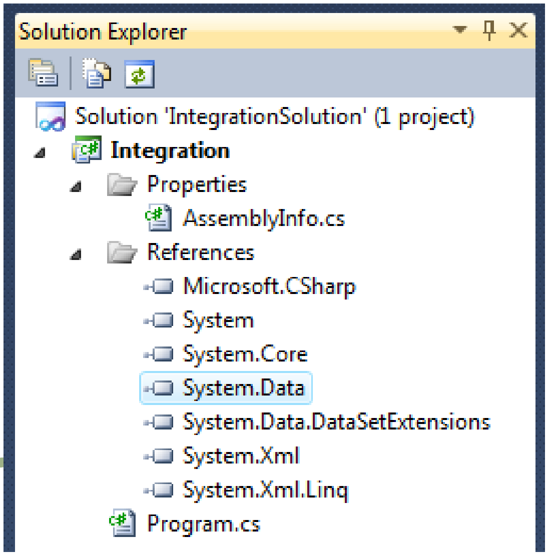
```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

namespace Integration
{
    class Program
    {
        static void Main(string[] args)
        {
        }
    }
}
```

# Our First Visual C# Project

Let's explore this screen before we move on.

The Solution Explorer, on the right, lists the name of the solution (IntegrationSolution) as well as the Integration project. Under the Integration project you will see the Properties folder, the References folder and the C# program named, in a gender-neutral fashion, `Program.cs`.

Recall that in order to use a particular namespace, such as `System.Data`, we have to code the statement `using System.Data;` in our C# program. That is still the case when using Visual Studio, but the References section shows you the names of the assemblies that go along with each namespace. Recall in the presentation on using `Oracle.DataAccess.Client` that we had to add a reference at the command line in order for the program to compile even though we coded `using Oracle.DataAccess.Client;` in the program. The reason we did that is because `Oracle.DataAccess.Client.dll` is not found in a normal .NET location. Now, if you expand the References section, you will see a list of the references you are given by default. By clicking on any one of the references listed, its properties will be revealed in the Properties section at the bottom right of Visual Studio. See next slide for an example of this. We talk more about adding references later on.

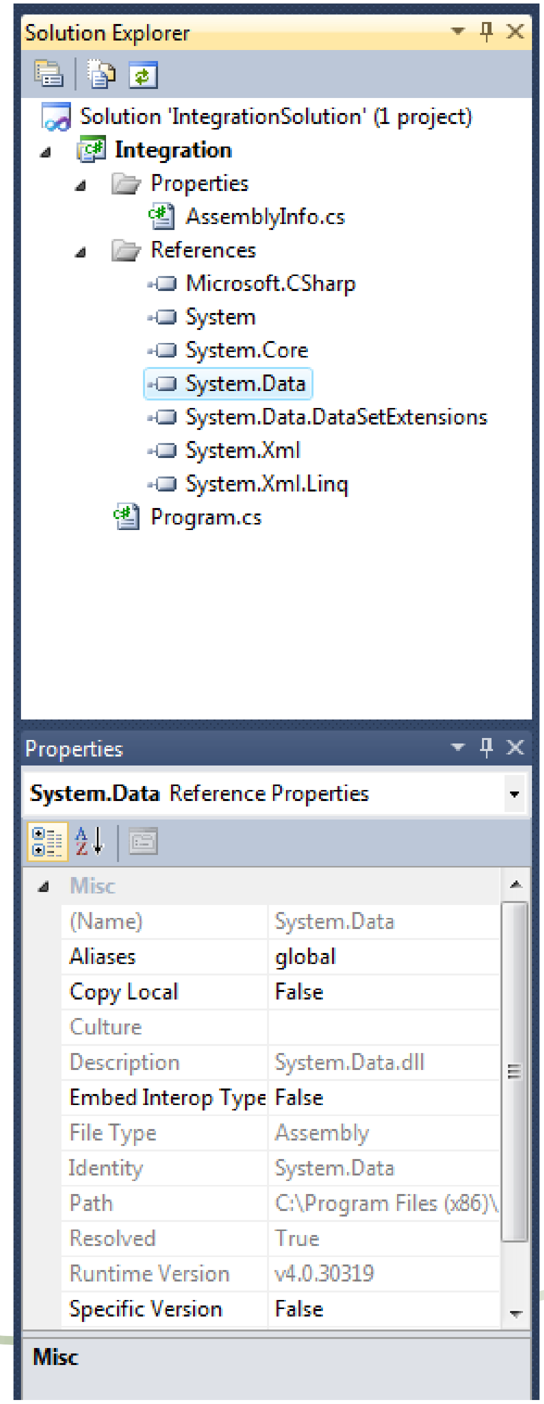


# Our First Visual C# Project

Clicking on `System.Data` in the References section to the right ➔

Reveals important properties of `System.Data` as well as smokin' recipe for baklava.

Note that if you click on one of the property names, a description of that property appears in the teensy-weensy grey box at the bottom.



| | |
|---|---|
| (Name) | System.Data |
| Aliases | global |
| Copy Local | False |
| Culture | |
| Description | System.Data.dll |
| Embed Interop Type | False |
| File Type | Assembly |
| Identity | System.Data |
| Path | C:\Program Files (x86)\ |
| Resolved | True |
| Runtime Version | v4.0.30319 |
| Specific Version | False |

Misc

# Our First Visual C# Project

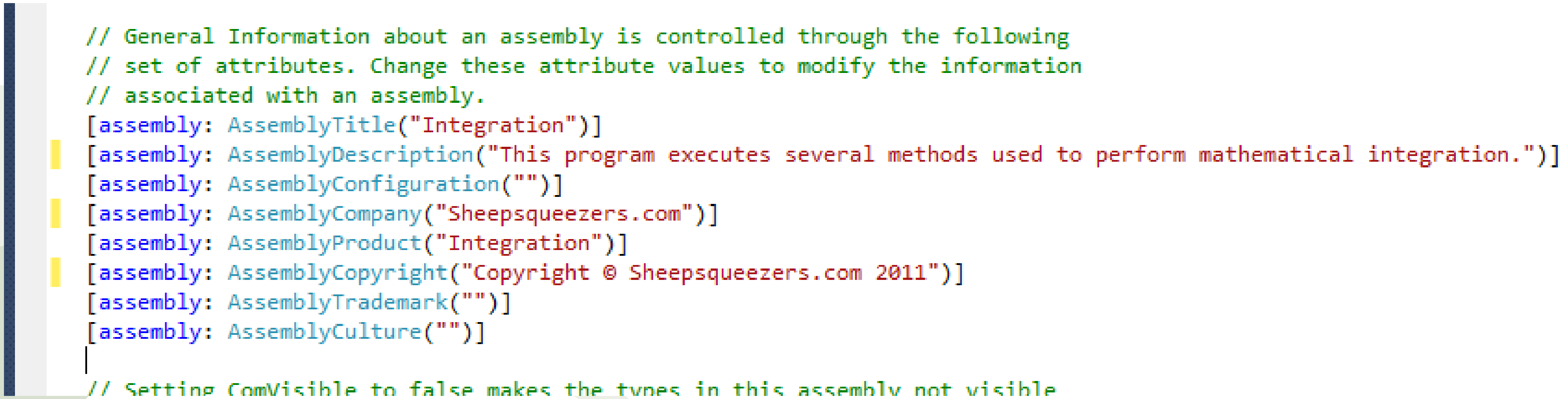
The Properties section shows you properties about the project itself. If you expand that section, you will see a single file, `AssemblyInfo.cs`, which when double-clicked, reveals the following code in the code window:

```
using System.Reflection;
using System.Runtime.CompilerServices;
using System.Runtime.InteropServices;

// General Information about an assembly is controlled through the following
// set of attributes. Change these attribute values to modify the information
// associated with an assembly.
[assembly: AssemblyTitle("Integration")]
[assembly: AssemblyDescription("")]
[assembly: AssemblyConfiguration("")]
[assembly: AssemblyCompany("Microsoft")]
[assembly: AssemblyProduct("Integration")]
[assembly: AssemblyCopyright("Copyright © Microsoft 2011")]
[assembly: AssemblyTrademark("")]
[assembly: AssemblyCulture("")]

// Setting ComVisible to false makes the types in this assembly not visible
// to COM components.  If you need to access a type in this assembly from
// COM, set the ComVisible attribute to true on that type.
[assembly: ComVisible(false)]

// The following GUID is for the ID of the typelib if this project is exposed to COM
[assembly: Guid("93f52d07-da0e-4ec2-bb46-819986a28f89")]

// Version information for an assembly consists of the following four values:
//
//      Major Version
//      Minor Version
//      Build Number
//      Revision
//
// You can specify all the values or you can default the Build and Revision Numbers
// by using the '*' as shown below:
// [assembly: AssemblyVersion("1.0.*")]
[assembly: AssemblyVersion("1.0.0.0")]
[assembly: AssemblyFileVersion("1.0.0.0")]
```

# Our First Visual C# Project

sheepsqueezers.com

Take note of the `assembly:` attributes. We learned about attributes in the *C# Programming III: Advanced C#* presentation. These attributes hold information about the project such as title, description, company, etc. as well as the version numbers of the assembly and file. Make sure to fill in the appropriate information for your project:



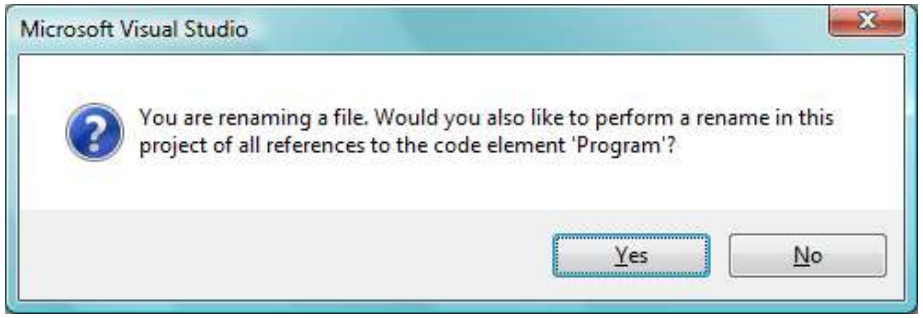
```
// General Information about an assembly is controlled through the following
// set of attributes. Change these attribute values to modify the information
// associated with an assembly.
[assembly: AssemblyTitle("Integration")]
[assembly: AssemblyDescription("This program executes several methods used to perform mathematical integration.")]
[assembly: AssemblyConfiguration("")]
[assembly: AssemblyCompany("Sheepsqueezers.com")]
[assembly: AssemblyProduct("Integration")]
[assembly: AssemblyCopyright("Copyright © Sheepsqueezers.com 2011")]
[assembly: AssemblyTrademark("")]
[assembly: AssemblyCulture("")]

// Setting ComVisible to false makes the types in this assembly not visible
```

As you see above, I changed the AssemblyDescription, the AssemblyCompany and the AssemblyCopyright. Note the yellow bars to the left. These indicate where changes have been made. Since the bars are yellow, this indicates that the changes have not been saved yet. If you click File…Save Properties\AssemblyInfo.cs, or click File…Save All, or click the Save icon on the toolbar, the yellow bars will be replaced with green bars. Next time you start Visual Studio, these green bars will not be displayed.

Next, let's look at the Program.cs file. If you click on Program.cs, the Properties window will show you some information about this file. In particular, the File Name property indicates the name of the file. Let's change this name.

# Our First Visual C# Project

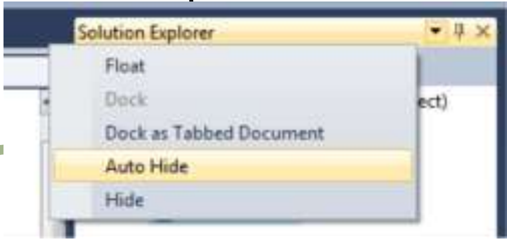
Go ahead and change that to `IntMethods.cs` by double-clicking the text `Program.cs` to the right of the property File Name and typing in `IntMethods.cs`. Finally, hit the Enter key for the change to take. You will see the following dialog box appear:


Microsoft Visual Studio

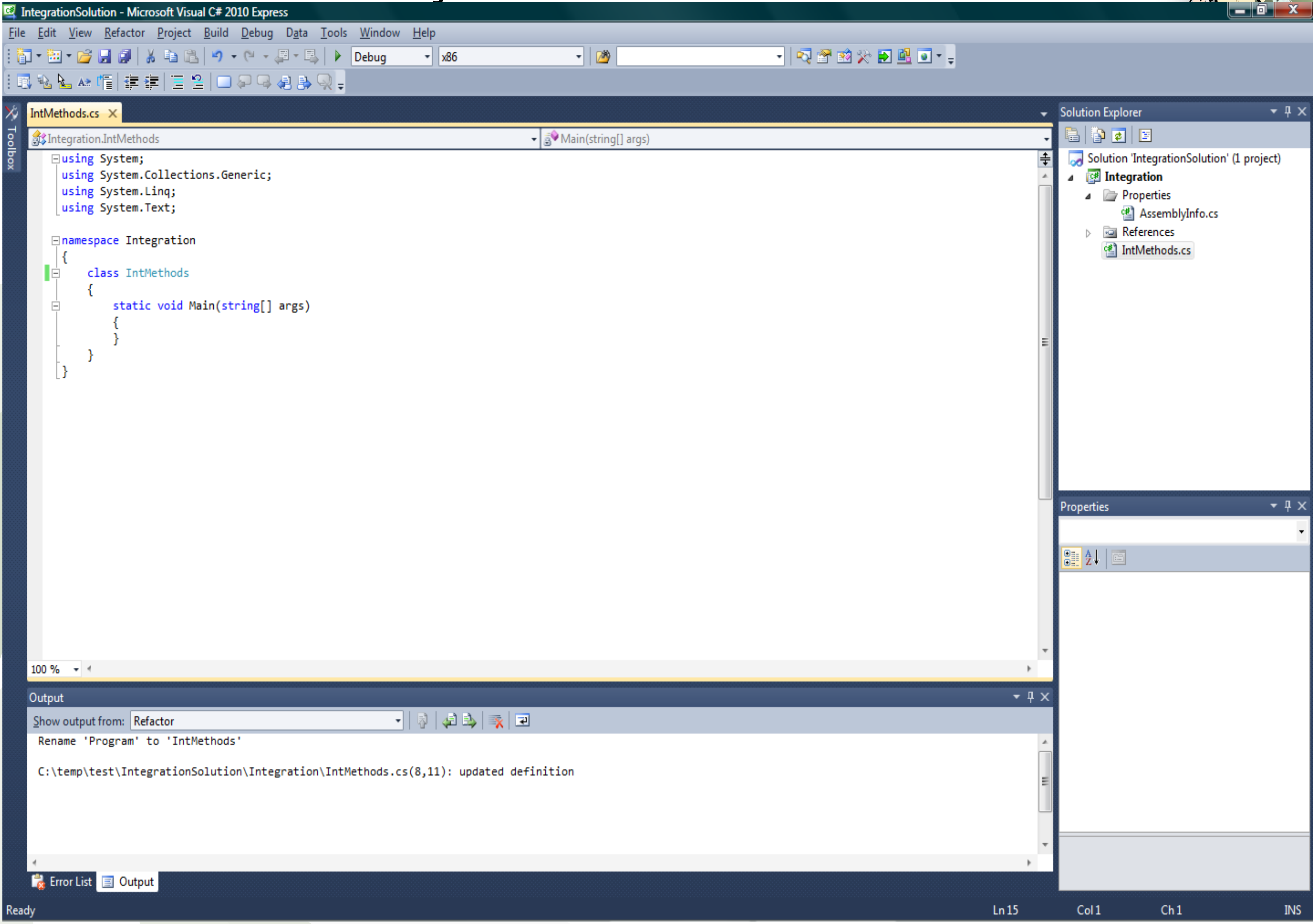
You are renaming a file. Would you also like to perform a rename in this project of all references to the code element 'Program'?

By clicking Yes, Visual Studio will rename all of the references to `Program.cs` to `IntMethods.cs` throughout…which is a good thing. Once complete, the Output window at the bottom which show you the results of this modification (called Refactor).

Now, let's take a look at the code window. If you double-click on InvMethods.cs in the Solution Explorer, the code shows up in the code window which takes up a vast amount of the screen real estate. See next slide. Now, if you want to expand the screen real estate even more, you can set both the Solution Explorer and the Properties window to auto-hide. This will, in effect, hide these windows similar to the look of the Toolbox. To do this, locate the down-arrow in the Solution Explorer and click on Autohide. Do the same thing to the Properties window. See the slide after the next. This additional screen real estate is especially important when building GUI applications, because, really, who wants a small GUI.



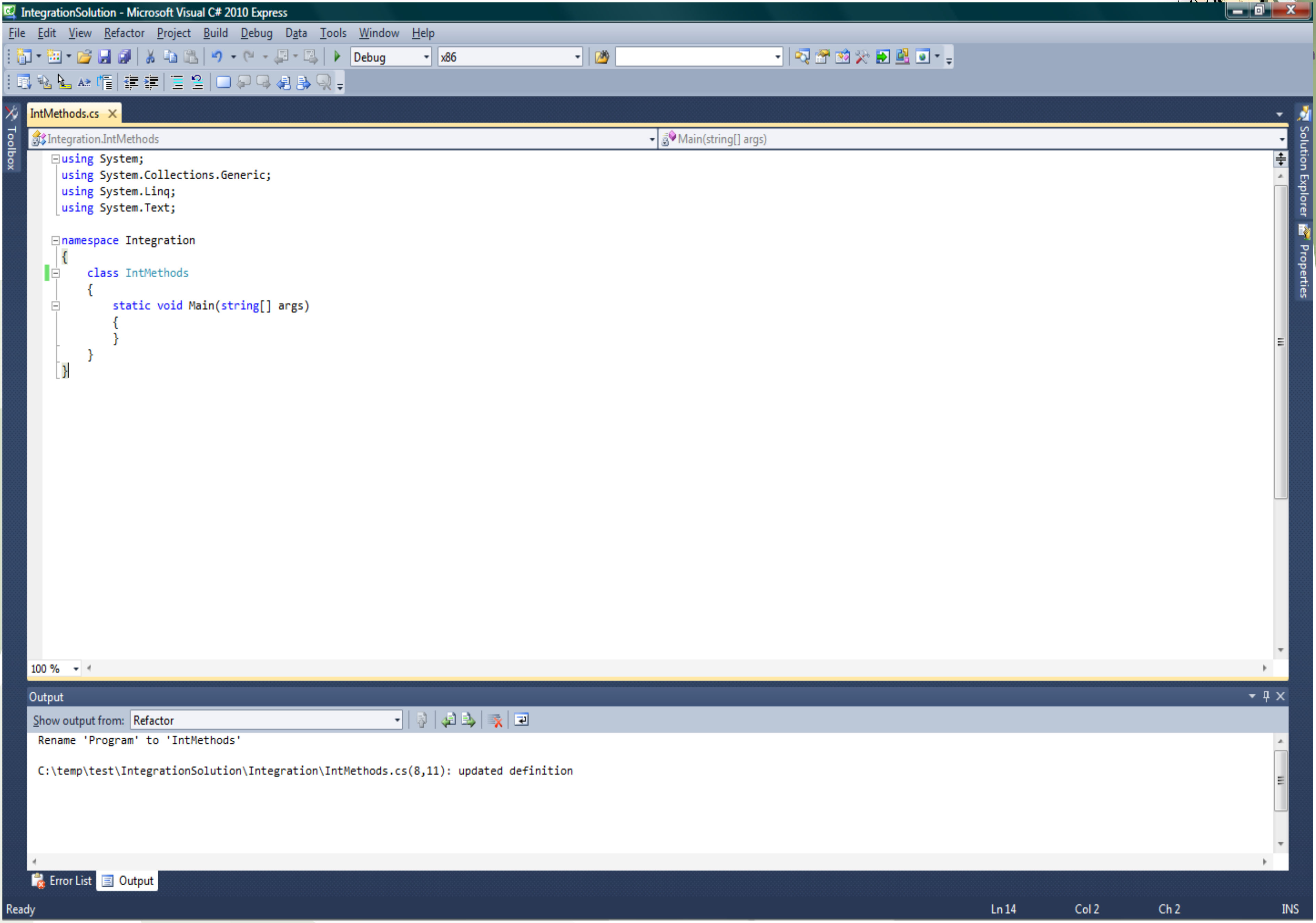
# Our First Visual C# Project



```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

namespace Integration
{
    class IntMethods
    {
        static void Main(string[] args)
        {
        }
    }
}
```

```
Rename 'Program' to 'IntMethods'

C:\temp\test\IntegrationSolution\Integration\IntMethods.cs(8,11): updated definition
```

# Our First Visual C# Project



```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

namespace Integration
{
    class IntMethods
    {
        static void Main(string[] args)
        {
        }
    }
}
```

Show output from: Refactor

```
Rename 'Program' to 'IntMethods'

C:\temp\test\IntegrationSolution\Integration\IntMethods.cs(8,11): updated definition
```

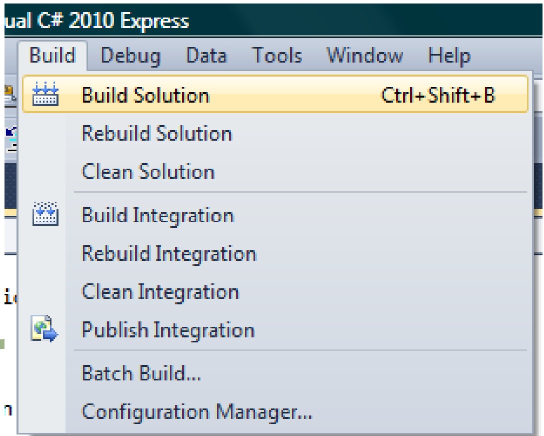
# Our First Visual C# Project

Now, let's add code to our project. Here is the simple integration code we used in the Programming C# II: Beginning C# presentation:

```
//Initialize my variables
Double SLICE_WIDTH = .0001D; //Width of rectangle
Double STARTING_X = 0D; //Starting X-Coordinate
Double ENDING_X = 10D; //Ending X-Coordinate
Double CURRENT_X = STARTING_X; //Current X value...used for loop
Double AREA_UNDER_CURVE = 0D; // Area under curve initialized to zero

//Compute the area under the curve x**2 + 1 from 0 to 10.
//From calculus, this value is 343.33.
do
{
 AREA_UNDER_CURVE += SLICE_WIDTH*(CURRENT_X * CURRENT_X + 1);
 CURRENT_X += SLICE_WIDTH;
} while(CURRENT_X <= ENDING_X);

//Print out the results
Console.WriteLine(AREA_UNDER_CURVE.ToString());
```

Copy this code from above and paste it into the Main static method. Save your project by clicking on the Save icon on the Toolbar (or clicking CTRL+S).

Next, let's compile our application. In order to do this you use one of the entries in the Build menu. As you can see to the right, you can either compile (build) by solution or project. In this case, let's Build the entire solution by clicking on Build Solution.

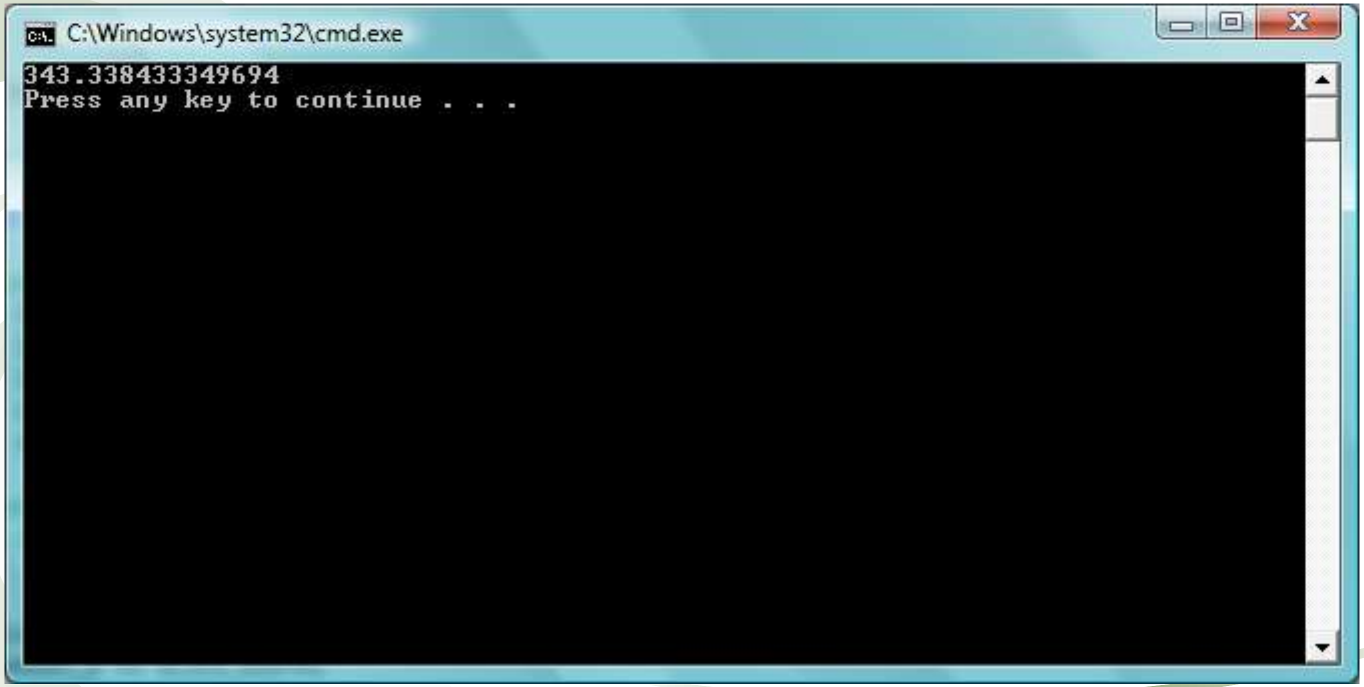


Take note of the output in the Output window. It shows that the build succeeded.

## Our First Visual C# Project

```
------ Build started: Project: Integration, Configuration: Debug x86 ------
  Integration -> C:\temp\test\IntegrationSolution\Integration\bin\Debug\Integration.exe
========== Build: 1 succeeded or up-to-date, 0 failed, 0 skipped ==========
```

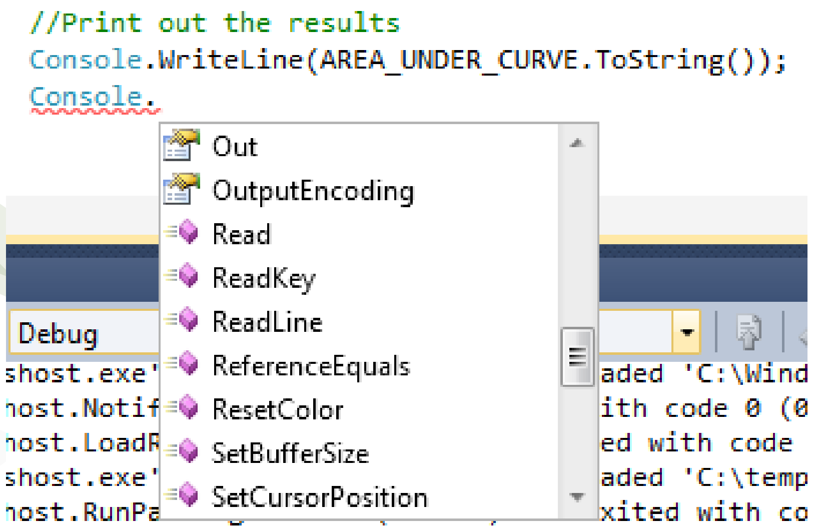
Next, to run the compiled console application, you can either click on the menu Debug → Start Debugging, the menu item Debug → Start without Debugging, click on the F5 button, or click the very small green arrow on the Toolbar. For us, click on Debug → Start without Debugging. You will see the command window popup with the results of our integration. Press any key to dismiss the command window.



```
343.338433349694
Press any key to continue . . .
```

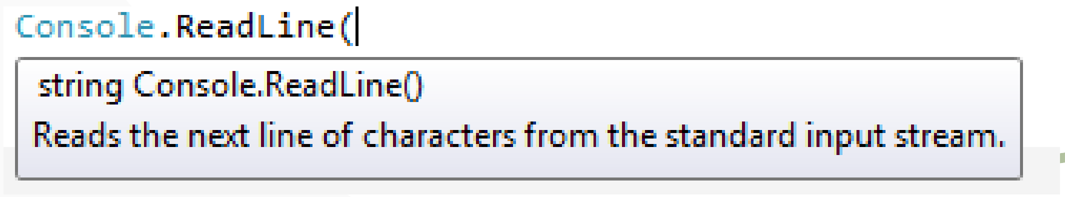
# Our First Visual C# Project

Now, the reason that we did not use Start with Debugging was because the output window flashes by so quickly. Let's add a `Console.Readline()` to the end of the program to prevent the window from disappearing so quickly. So, right after the line `Console.WriteLine(AREA_UNDER_CURVE.ToString());`, place a blank line and begin typing the word Console followed by the period. At this point, Visual Studio shows you a list of members related to the Console class, as shown below.



```
//Print out the results
Console.WriteLine(AREA_UNDER_CURVE.ToString());
Console.
```

If you scroll down to the R's, you will see Readline. Click on Readline. Next, type the left parenthesis. You will see the following informative text popup:

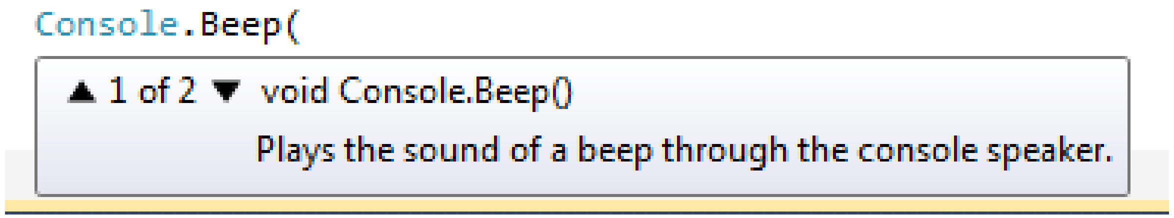


```
Console.ReadLine(
```

string Console.ReadLine()
Reads the next line of characters from the standard input stream.

## Our First Visual C# Project

Now, this informative text is very useful, especially if the method has more than one constructor. For example, if I wanted the program to beep when it finished, I could use the `Console.Beep()` method. But this method has two constructors, as shown below:



```
Console.Beep(
▲ 1 of 2 ▼ void Console.Beep()
           Plays the sound of a beep through the console speaker.
```

As you can see, there are 1 of 2 constructors for this method. If you move your cursor up or down, you can see the other constructors. Hit the ESC button to dismiss this popup.

Now, add the `Console.Readline();` to the program, click on the menu Build → Build Solution, then click on the F5 button to start the program. Your console application comes up and stays up until you hit the Enter key (which is what the `Console.Readline()` method is waiting for you to do).

Now, save this project and close Visual C#.

## Our Second Visual C# Project

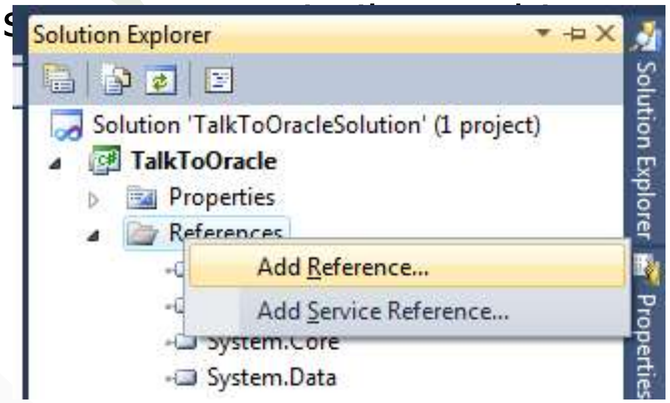
sheepsqueezers.com

Let's create another project. Open up Visual C# 2010 Express Edition and create a new console application. This time, give the application the name `TalkToOracle` and name the solution `TalkToOracleSolution`. Rename `Program.cs` to `OracleDataAccess.cs`.

Recall that in the presentation on the `Oracle.DataAccess.Client` namespace provided by Oracle, we had to compile our program using the command line reference switch, similar to this:

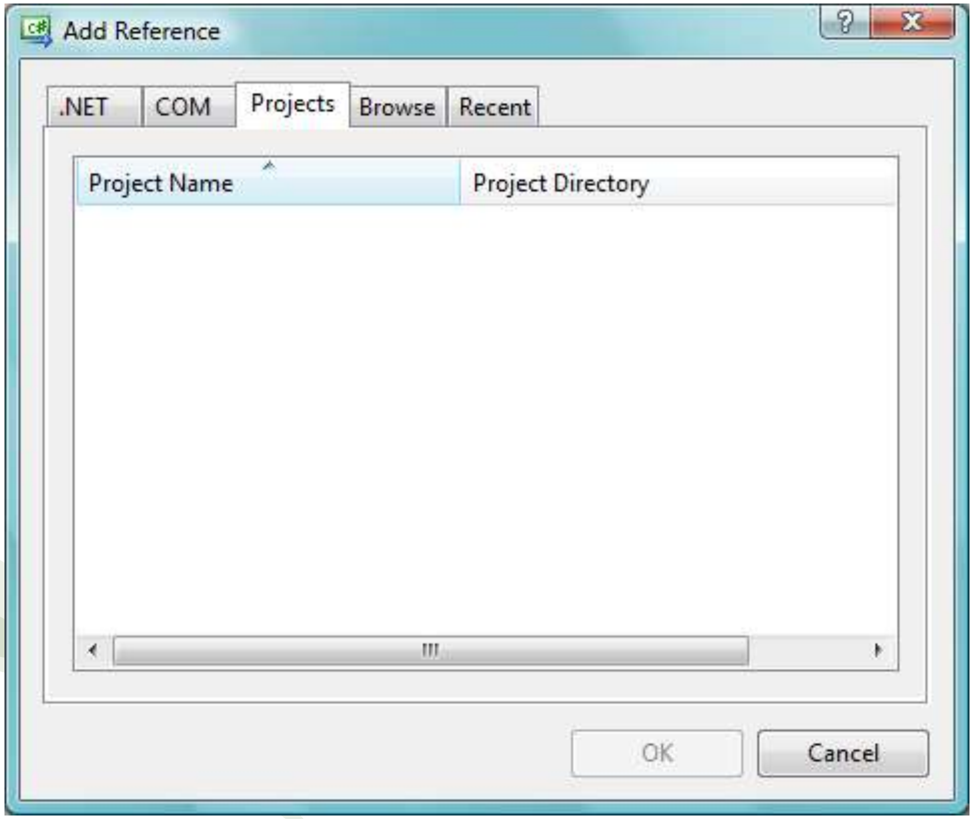
```
csc /r:C:\oracle\product\11.2.0\dbhome_1\ODP.NET\bin\4\Oracle.DataAccess.dll pgm.cs
```

Now, instead of using the /r switch, in Visual C# we add a reference to the class library Oracle.DataAccess.dll. In the Solution Explorer, expand the References section to show the default class libraries. Now, right click on the word References, and you will s



Click on Add Reference… and the Add Reference dialog box appears. There are several tabs available on this dialog and we will explain each one.

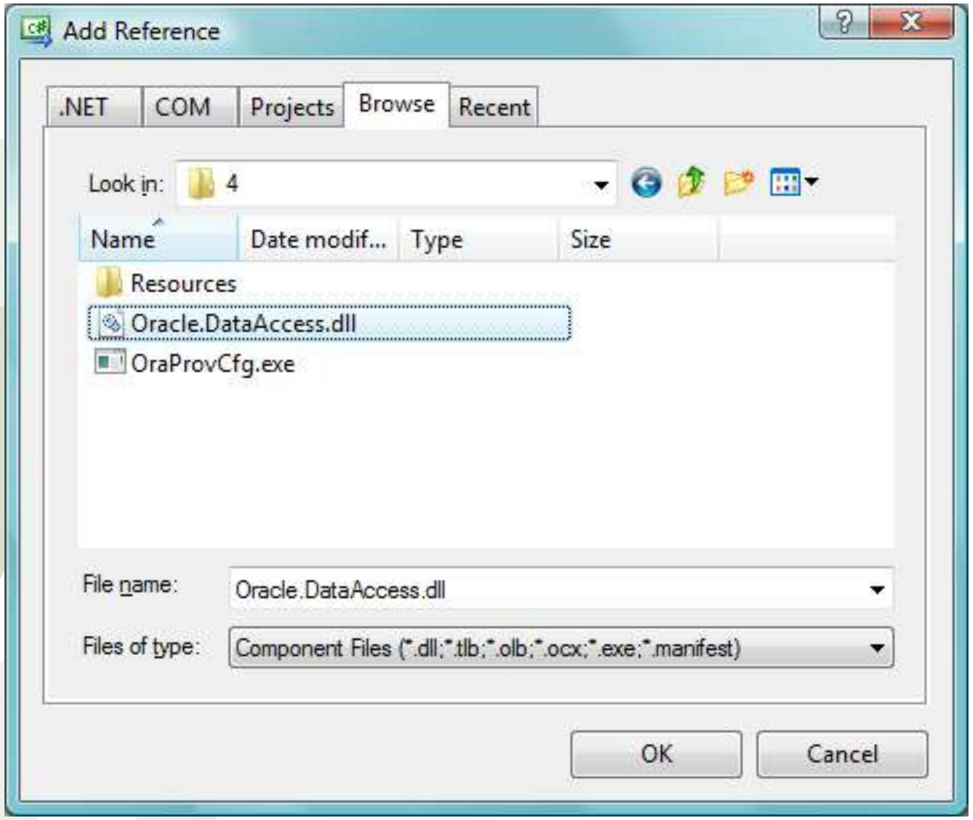
# Our Second Visual C# Project

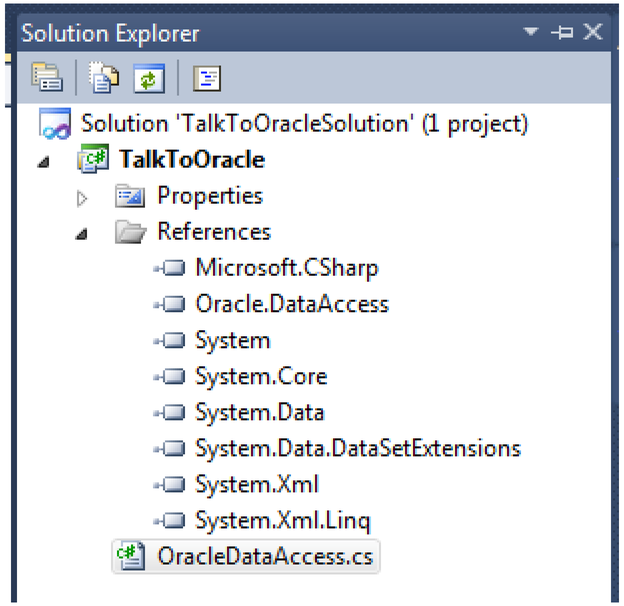


- .NET – This tab allows you to choose from a list of .NET framework namespaces such as `System.Numerics`, `System.Drawing`, etc.
- COM – This tab allows you to choose from a list of COM components such as `Microsoft Access 14.0 Object Library`, etc.
- Projects – This tab lists all reusable components created by you in your local environment. As you see, we haven't create reusable components.
- Browse – This tab allows you to browse for a specific class library.
- Recent – This tab lists items already added to our current and previous projects.

# Our Second Visual C# Project

Now, click on the Browse tab, and locate the `Oracle.DataAccess.dll` file which should be located in the folder (or similar folder): `C:\oracle\product\11.2.0\dbhome_1\ODP.NET\bin\4\` (shown below left). Click on the file `Oracle.DataAccess.dll` and then click OK.





After you click OK, you should see a reference to this class library in the References section (shown above right).

## Our Second Visual C# Project

Next, let's add some code to connect to the Oracle database, pull some data and show it at the command line. Add the following code to the Main routine:

```
String sConn = "Data Source=ORCL;User Id=scott;Password=tiger;";
using (OracleConnection oConn = new OracleConnection(sConn)) {

 oConn.Open();
 DataSet oDS = new DataSet();
 OracleCommand oCmd = new OracleCommand("SELECT * FROM EMP",oConn);
 OracleDataAdapter oDA = new OracleDataAdapter(oCmd);
 Int32 iRC = oDA.Fill(oDS,"EMP");

 //Print out data
 foreach(DataTable dtThisTable in oDS.Tables) {
  Console.Write("DataTable={0}\n",dtThisTable.TableName);
  foreach(DataRow drThisRow in dtThisTable.Rows) {
   Console.Write("\t");
   foreach(DataColumn dtThisColumn in dtThisTable.Columns) {
    Console.Write("{0} ",drThisRow[dtThisColumn]);
   }
   Console.WriteLine();
  }
 }

}

Console.Readline();
```

Add the following `using` statements at the top of the program:

```
using Oracle.DataAccess.Client;
using System.Data;
```

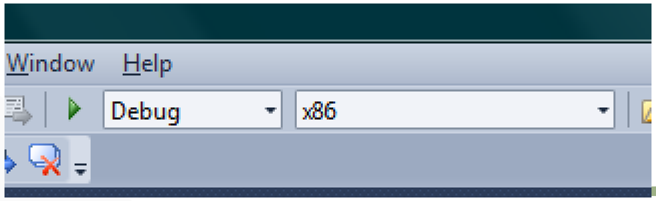
# Our Second Visual C# Project

Before we compile and run the program, notice that if you move the cursor to a left parenthesis or left curly bracket, the corresponding right parenthesis or right curly bracket it highlighted. This allows you to ensure that you are not missing a parenthesis or curly bracket.

Finally, save the program and compile it by clicking the menu Build → Build Solution. If your machine is a 64-bit machine like mine, then you will receive the following error:

```
------ Rebuild All started: Project: TalkToOracle, Configuration: Debug x86 ------
warning CS1607: Assembly generation -- Referenced assembly 'Oracle.DataAccess.dll' targets a different
processor

Compile complete -- 0 errors, 1 warnings
  TalkToOracle -> C:\temp\test\TalkToOracleSolution\TalkToOracle\bin\Debug\TalkToOracle.exe
========== Rebuild All: 1 succeeded, 0 failed, 0 skipped ==========
```

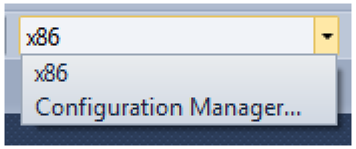
This is due to the fact that the `Oracle.DataAccess.dll` I downloaded from Oracle's website is the 64-bit assembly and not the 32-bit version. Now, you will notice that at the top of the Visual C# IDE, the text "x86" appears in a drop-down box:



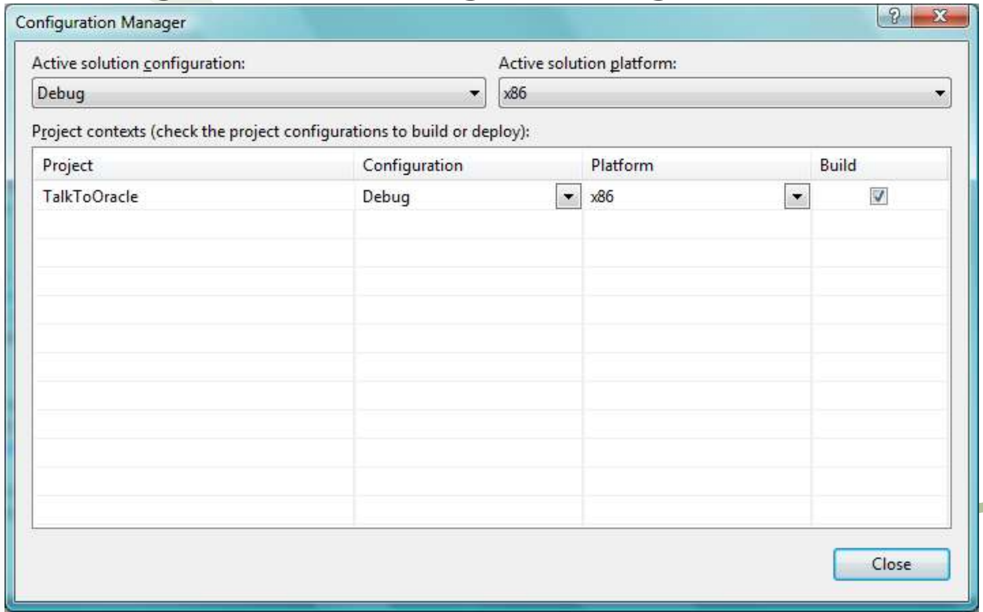
# Our Second Visual C# Project

This x86 indicates that we are building x86 assemblies. The reason that we did not have this problem when we compiled at the command line with `csc.exe` is because `csc.exe` defaults to a target of `anycpu`. We could have just as easily requested a 32-bit assembly by using the `/platform:x86` switch at the command line and we would have received an error as well there.

To rectify this situation, we have to change the x86 to x64. If you click on the arrow associated with the x86 drop-down box, you will notice two options: x86 and Configuration Manager…(shown below).



Since x64 or AnyCPU is not available, click on the Configuration Manager… menu item. This starts the Configuration Manager dialog box.


Configuration Manager

Active solution configuration: Debug

Active solution platform: x86

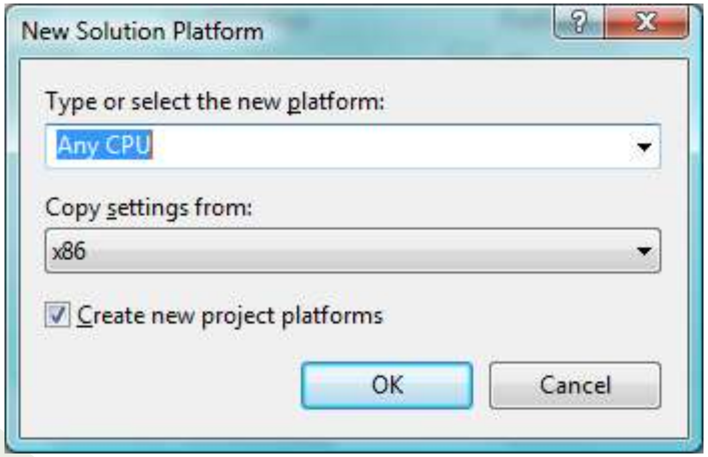
Project contexts (check the project configurations to build or deploy):

| Project | Configuration | Platform | Build |
|---|---|---|---|
| TalkToOracle | Debug | x86 | ✓ |

# Our Second Visual C# Project

sheepsqueezers.com

On the Configuration Manager dialog box, we need to set up the AnyCPU option. Click on the drop-down box below the text `Active solution platform:`. and select `<New…>`. The `New Solution Platform` dialog box appears and looks like this:


New Solution Platform

Type or select the new platform: Any CPU

Copy settings from: x86

☑ Create new project platforms

The drop-down box under the text `Type or select the new platform:` contains the following choices: `Any CPU`, `Itanium` and `x64`. Select Any CPU. The `Copy settings from:` option of x86 is fine. Also, leave the checkbox checked. Click OK.

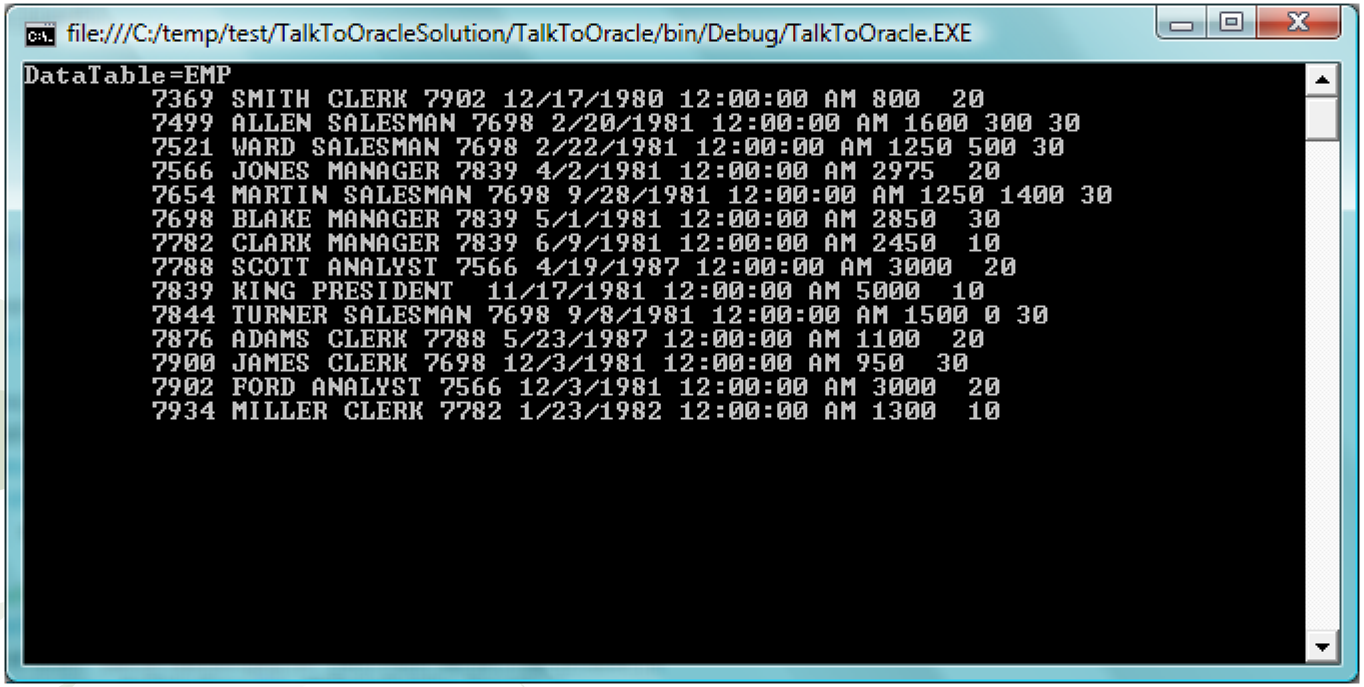
You are brought back to the Configuration Manager dialog box only this time you now have the option of Any CPU. Go ahead and repeat the steps above so that you have the ability to choose x64 or Itanium. Finally, choose your target platform (Any CPU should be fine) and click Close.

When you are brought back to the IDE, you will notice that the drop-down box at the top shows Any CPU. This drop-down box now also lists all of the choices we could possibly want!

## Our Second Visual C# Project

sheepsqueezers.com

Finally, Build the solution and then hit the F5 button. You should see the following output:


file:///C:/temp/test/TalkToOracleSolution/TalkToOracle/bin/Debug/TalkToOracle.EXE

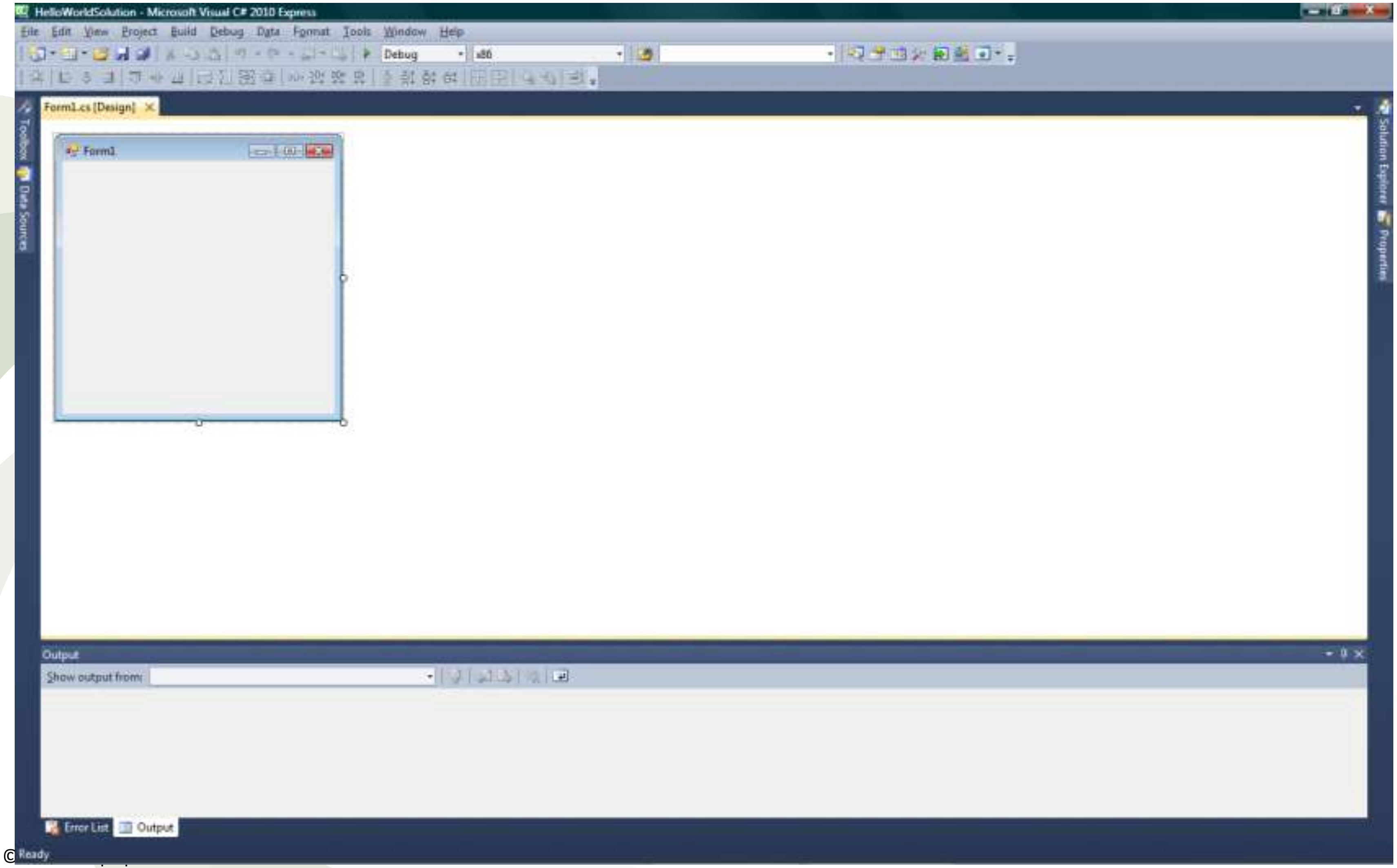
```
DataTable=EMP
        7369 SMITH CLERK 7902 12/17/1980 12:00:00 AM 800  20
        7499 ALLEN SALESMAN 7698 2/20/1981 12:00:00 AM 1600 300 30
        7521 WARD SALESMAN 7698 2/22/1981 12:00:00 AM 1250 500 30
        7566 JONES MANAGER 7839 4/2/1981 12:00:00 AM 2975  20
        7654 MARTIN SALESMAN 7698 9/28/1981 12:00:00 AM 1250 1400 30
        7698 BLAKE MANAGER 7839 5/1/1981 12:00:00 AM 2850  30
        7782 CLARK MANAGER 7839 6/9/1981 12:00:00 AM 2450  10
        7788 SCOTT ANALYST 7566 4/19/1987 12:00:00 AM 3000  20
        7839 KING PRESIDENT  11/17/1981 12:00:00 AM 5000  10
        7844 TURNER SALESMAN 7698 9/8/1981 12:00:00 AM 1500 0 30
        7876 ADAMS CLERK 7788 5/23/1987 12:00:00 AM 1100  20
        7900 JAMES CLERK 7698 12/3/1981 12:00:00 AM 950  30
        7902 FORD ANALYST 7566 12/3/1981 12:00:00 AM 3000  20
        7934 MILLER CLERK 7782 1/23/1982 12:00:00 AM 1300  10
```

Hit the Enter key to dismiss the dialog box (`Console.Readline()` is waiting for you to do this).

Save and close this project.

# Our Third (and Visually Stunning) C# Project

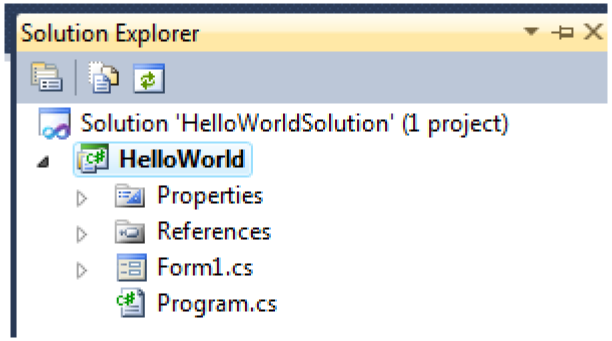
Okay, enough of the black-and-white-hey-it's-the-1950s-man console crap, let's create a GUI application. Create a new project, but this time instead of choosing Console Application, choose Windows Forms Application. Name the project HelloWorld and name the solution HelloWorldSolution. This time, instead of a code window, you will see a blank form:



## Our Third (and Visually Stunning) C# Project

When you open the Toolbox, this time you will see a list of several components available to you. Components such as buttons, checkboxes, treeviews, etc.

Now, if you open up the Solution Explorer, you will see something like the following:



Notice that there is a `Program.cs`, just like with our console applications, but there is a `Form1.cs`. `Form1.cs` is just the graphical appearance of the form. Under `Form1.cs` is another file called `Form1.Designer.cs`. This file contains C# code used to initialize the form as well as dispose of the form. Note that in the file Program.cs, you have the familiar Main program, but it contains additional lines by default to show your form:

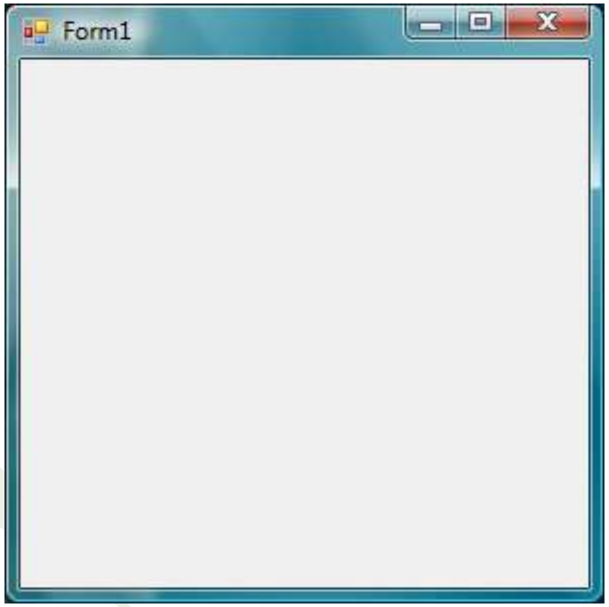
```
Application.EnableVisualStyles();
Application.SetCompatibleTextRenderingDefault(false);
Application.Run(new Form1());
```

The method `Application.Run(new Form1());` is used to show the form when the program executes.

Before we move on, rename `Program.cs` to `HelloWorld.cs` and `Form1.cs` to `frmHW.cs`.

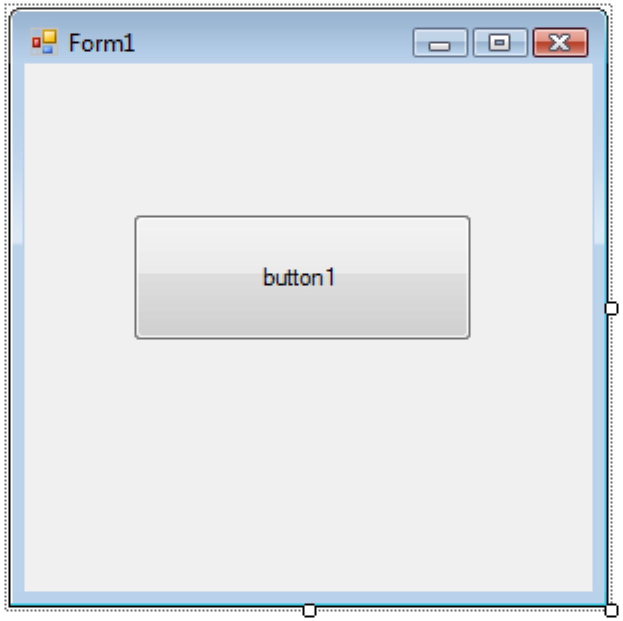
## Our Third (and Visually Stunning) C# Project

Next, let's compile this program and click the F5 button. Alternatively, you can just click the F5 button and this will compile the program and then execute it! You will see a blank form appear on your screen.



Next, let's put a button on the form that will display a popup dialog box with the words *Hello, World!* in it. Expand the Toolbox, click on the Button component and draw a button on the form. (You may have to click on a blank area of the IDE after you've clicked on Button so that the Toolbox retracts to display the form again.) You should see something like this (see next slide).

# Our Third (and Visually Stunning) C# Project



You can make the button as large or as small as you want to.

Next, change the name of the button from `button1` to `btnHelloWorld`. You do this by right-clicking on the button and clicking on Properties menu item in the popup menu. This will display the Properties window on the right side of the IDE. If you scroll down to the Design section you will see an property labeled `(Name)`. This is the programmatic name of the button. Change this to `btnHelloWorld`. Hit the Enter key or tab to another property to make this change take effect.

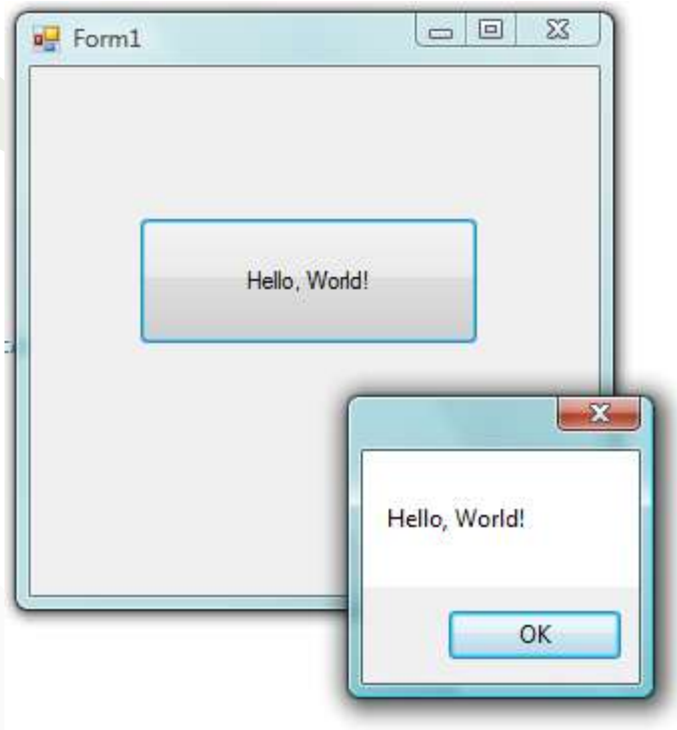
Next, let's change the text appearing on the face of the button. Scroll to the Appearance section and look for the Text property. Change the text from `button1` to `Hello, World!`. Hit the Enter key or tab to another property to make this change take effect.

## Our Third (and Visually Stunning) C# Project

Next, let's code the popup dialog box. If you double-click on the button, you will taken to the code window associated with the form (`frmHW.cs`). The IDE places a blank method in that code window. This method is actually click event for the button. Notice that it is named `btnHelloWorld_Click`. Now, in this method, place the following code:

```
MessageBox.Show("Hello, World!");
```

Next, click the F5 button and when the form appears this time, click on the button labeled *Hello, World!*. You should see something like this:



Hello, World!

## Our Third (and Visually Stunning) C# Project

Click OK to dismiss the popup message box. Click on the X button to close the form and you'll be back in the IDE.

Now, let's add a second button. This button will be used to close the form rather than having us click on the X button. Add another button to the form, name is `btnClose` and change the text to `Close!`. Double-click this button and you will be taken to its own Click event handler. In this method, enter the following code:

```
this.Close();
```

The `this`, in this case, refers to the form itself. If you type in **`this.`** you will be presented with several Intellisense options to choose from. Run this program by hitting the F5 button. If you click the `Close!` button, the form will be closed and you will be returned to the IDE.

## Our First Boo-Boo

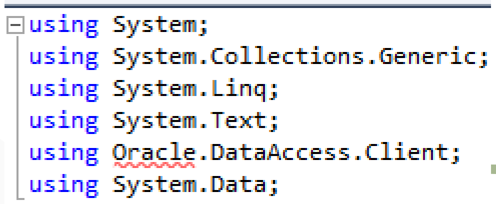
So far, we haven't see what happens when we make a programming mistake. Well, that's going to change right now, mister!

Recall that we added a reference to our the `Oracle.DataAccess.dll` and added the `using` statement using `Oracle.DataAccess.Client;` to the top of our code. What happens if we leave off the using statement? The result is this error message on the Error List tab at the bottom of the IDE:

```
The type or namespace name 'OracleConnection' could not be found (are you missing a using directive or an
assembly reference?)
```

Now, this error message occurred even before compiling the program. The moment we used the class `OracleConnection` without the using statement, the error message appears and a red underline appear under the text `OracleConnection` in the code window.

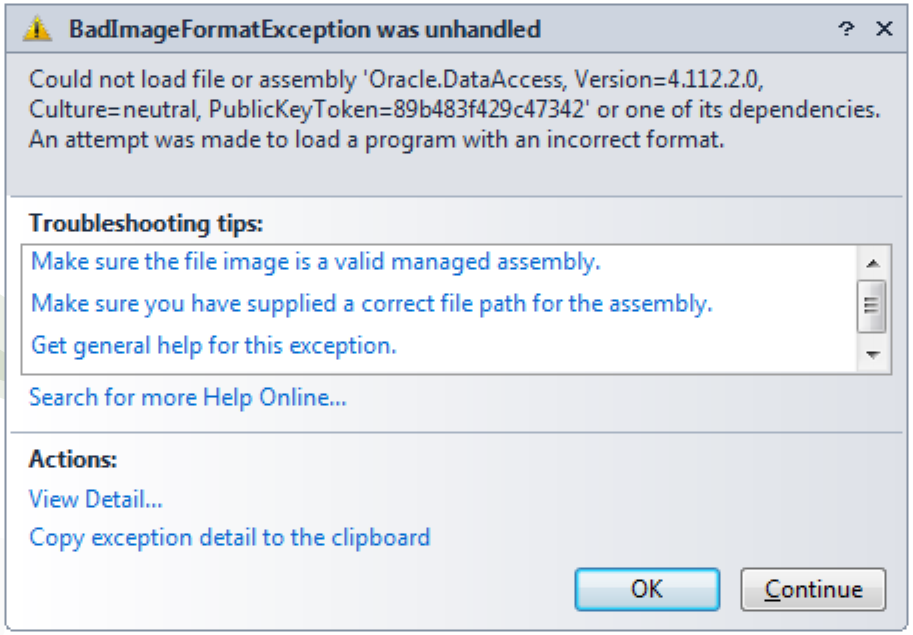
If, instead, we forgot to add the *reference*, but did code the `using` statement, the IDE would not recognize the `Oracle` namespace and give place an red underline under the text Oracle (as shown below). Also, the remaining code referencing Oracle-related classes, etc. would also have red underline under them.



```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using Oracle.DataAccess.Client;
using System.Data;
```

## Our First Boo-Boo

Recall that we had to compile the Oracle program with Any CPU or x64 because my laptop is running 64-bit Windows and the `Oracle.DataAccess.dll` is a 64-bit class library. If we attempt to compile our program as 32-bit, the compilation will succeed, but we receive the following error message when we run the program:



BadImageFormatException was unhandled

Could not load file or assembly 'Oracle.DataAccess, Version=4.112.2.0, Culture=neutral, PublicKeyToken=89b483f429c47342' or one of its dependencies. An attempt was made to load a program with an incorrect format.

Troubleshooting tips:

Make sure the file image is a valid managed assembly.

Make sure you have supplied a correct file path for the assembly.

Get general help for this exception.

Search for more Help Online...

Actions:

View Detail...

Copy exception detail to the clipboard

If you receive the exception `BadImageFormatException` it's a good bet that one or more of your class libraries has a different bittage than the rest. In order to determine the bittage, you can use the `dumpbin.exe` program installed when you installed Microsoft Visual C++ 2010 Express (you did install that, didn't you?). Dumpbin produces a section labeled `FILE HEADER VALUES` and the first line labeled `machine` indicates the bittage of the assembly:

## Our First Boo-Boo

```
dumpbin /headers C:\oracle\product\11.2.0\dbhome_1\ODP.NET\bin\4\Oracle.DataAccess.dll
```

produces the following (in part):

```
Microsoft (R) COFF/PE Dumper Version 10.00.30319.01
Copyright (C) Microsoft Corporation.  All rights reserved.

Dump of file C:\oracle\product\11.2.0\dbhome_1\ODP.NET\bin\4\Oracle.DataAccess.dll

PE signature found

File Type: DLL

FILE HEADER VALUES
            8664 machine (x64)
               2 number of sections
        4CEA1964 time date stamp Mon Nov 22 02:19:00 2010
               0 file pointer to symbol table
               0 number of symbols
              F0 size of optional header
            2022 characteristics
                   Executable
                   Application can handle large (>2GB) addresses
                   DLL
```

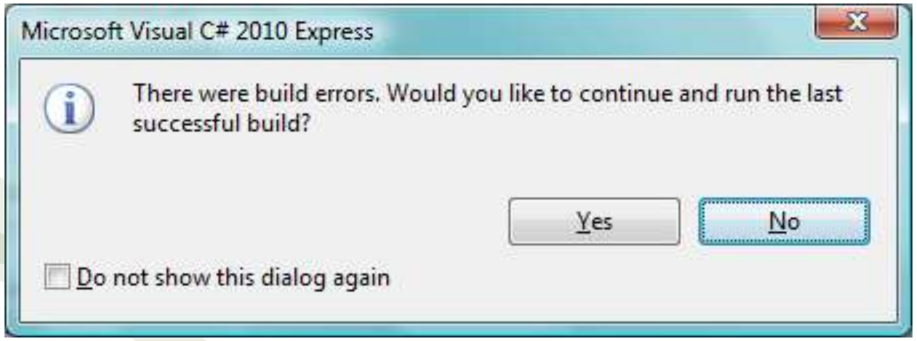
As you can see above in red, the `Oracle.DataAccess.dll` is a compiled for an x64 (64-bit) machine.

Note that if you compile this program using Any CPU, it works fine, but I assume that you should compile for the software's intended target platform (x86, x64, Itanium) rather than a generic platform (Any CPU).

## Our First Boo-Boo

Now, if you do receive an error, warning or message within the Error List tab, you can double-click the line and the code window takes you to the line that caused the error.  Probably a good idea to examine the first error or warning message given in the Error List tab rather than anything towards the end!

If you leave off a semi-colon, you may receive the following dialog box if you just hit the F5 button to compile/execute the program:



Microsoft Visual C# 2010 Express

There were build errors. Would you like to continue and run the last successful build?

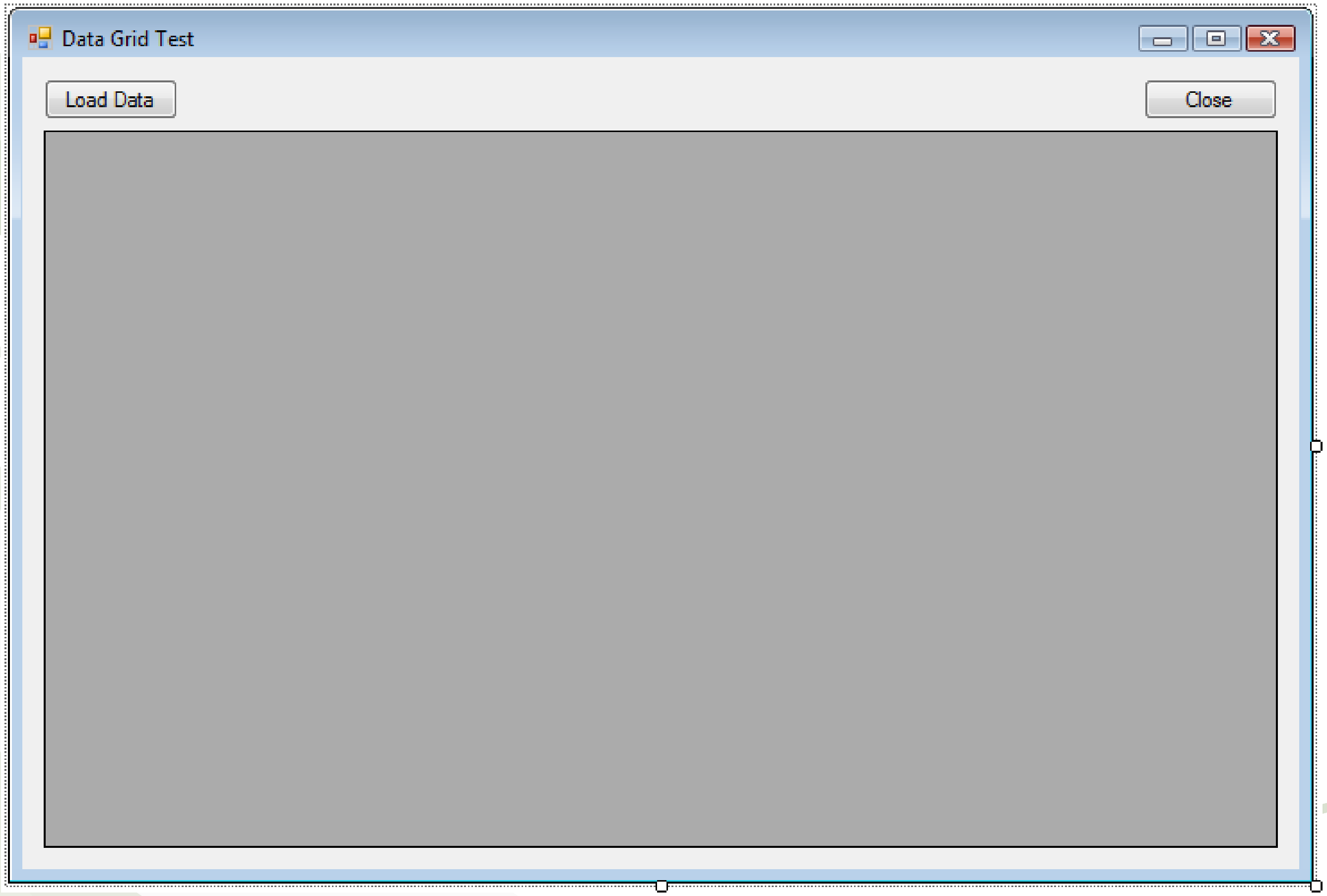
Division by zero errors have been a problem since the abacus, but in .NET it seems that division by zero is handled differently if you are using a `Double` than if you are using an `Int32`.  If you using a `Double`, dividing by zero results in the value `Infinity`.  No error message is returned, but the results of your computations will probably be a bit off.  If you are using `Int32`, you will get a `DivideByZeroException` exception.  Regardless, you should test your denominators for zero-ness before doing the division.

If you leave off a semi-colon, you may receive the following dialog box if you just hit the F5 button to compile/execute the program:

## Our Fourth (and Visually Stunning) C# Project

In this section, I'd like to explore the DataGrid control a little bit. I want to read in the data from my Oracle table and populate a DataGrid with that data.

You know the drill by now: create a new solution for a Windows Form Application, rename `Program.cs`, rename the `Form1`, etc. etc. Add two buttons to the form: one button is labeled `Load Data` and the other is `Close`. Drop a `DataGrid` on the form. You should have something similar to this form:



## Our Fourth (and Visually Stunning) C# Project

Now, the `DataGrid` has a `DataSource` property you can set with a `DataTable`. So, we're going to pull our data from the database into a `DataTable` and then populate the `DataGrid` by updating the `DataSource` property.

Don't forget to target the x64 or Any CPU platforms if you are not running x86 with the 32-bit `Oracle.DataAccess.dll`!

Here is the code for the Click event of the Load Data button:

```
btnLoadData.Enabled = false;
DataTable dtMyData = DataGridTest.GetOraData();
dataGridView1.DataSource = dtMyData;
btnLoadData.Enabled = true;
```

Notice that when a user clicks on the `Load Data` button, I am disabling it so that the use cannot click on it more than one. Once the data has been loaded, I enable the button again. Notice that I am returning a `DataTable` and calling the `GetOraData` method to load the data from Oracle. Note that the code to pull the data from Oracle is NOT in the code for the form events! The code is in what is called `Program.cs` by default!! The code is on the next slide.

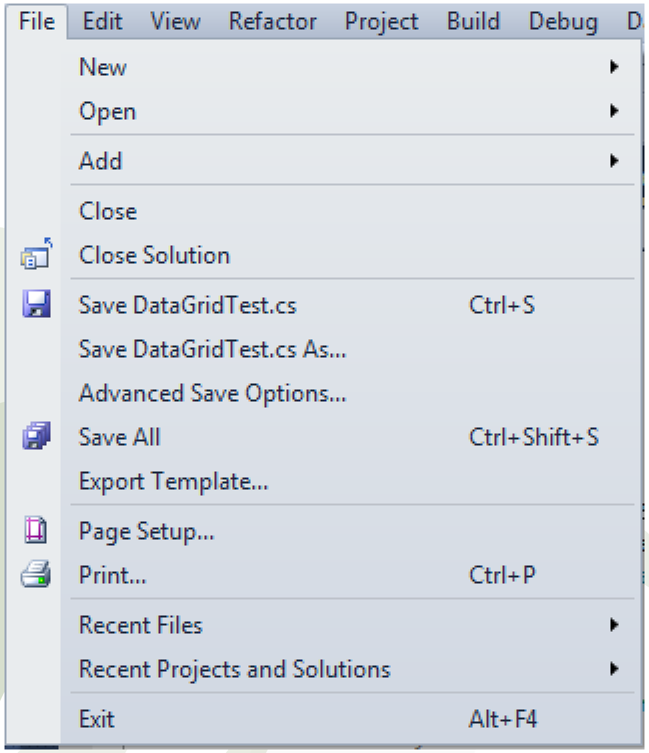
# Our Fourth (and Visually Stunning) C# Project

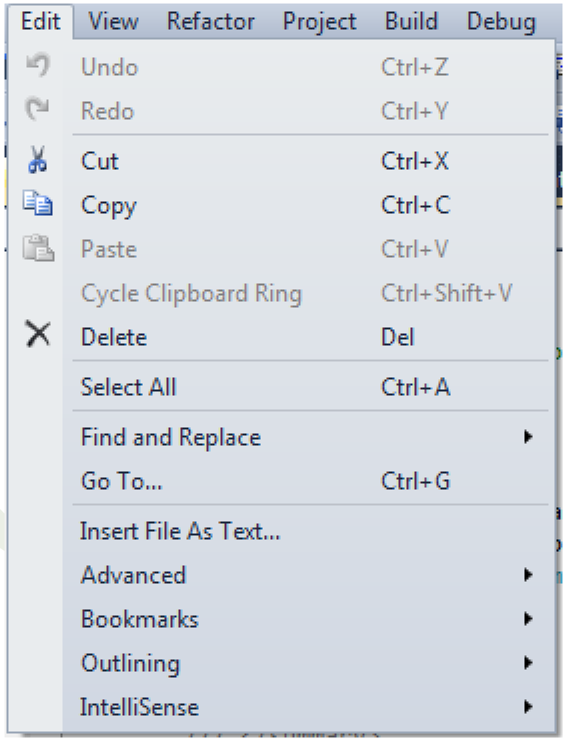
```
/// <summary>
/// This method pulls data from the Oracle database and returns a DataTable.
/// </summary>
public static DataTable GetOraData()
{
    String sConn = "Data Source=ORCL;User Id=scott;Password=tiger;";
    OracleConnection oConn = new OracleConnection(sConn);
    oConn.Open();
    DataSet oDS = new DataSet();
    OracleCommand oCmd = new OracleCommand("SELECT * FROM EMP", oConn);
    OracleDataAdapter oDA = new OracleDataAdapter(oCmd);
    Int32 iRC = oDA.Fill(oDS, "EMP");
    oConn.Close();
    return(oDS.Tables["EMP"]);
}
```

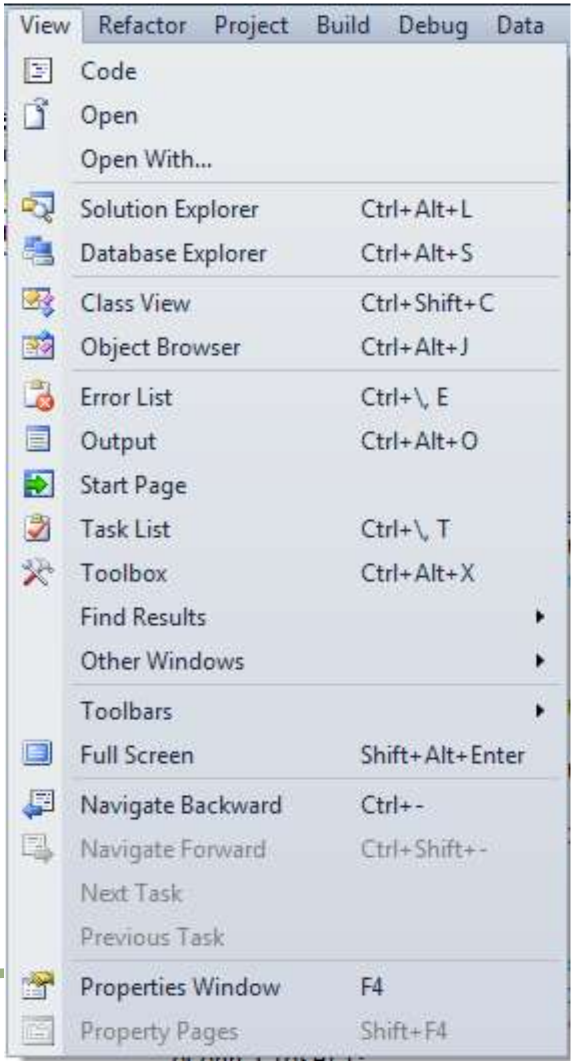
Notice that I am still using a `DataSet`, but I am returning the `EMP` table by using the `DataTableCollection`: `oDS.Tables["EMP"];`.

# Exploring the Visual C# Menus

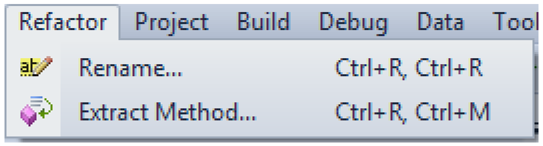
In this section, I'd like to explore the menus and menu items available within the IDE. Since each menu takes up a lot of room visually, I have placed them on this and the subsequent few pages. Detailed descriptions then follow.

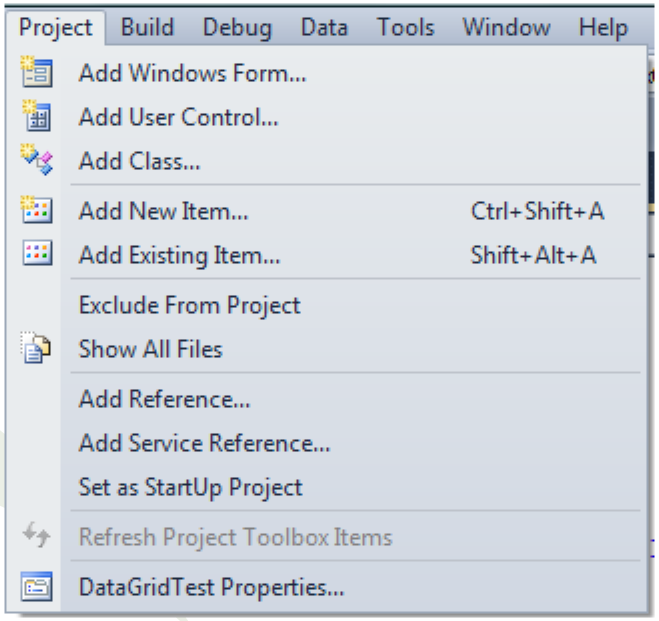


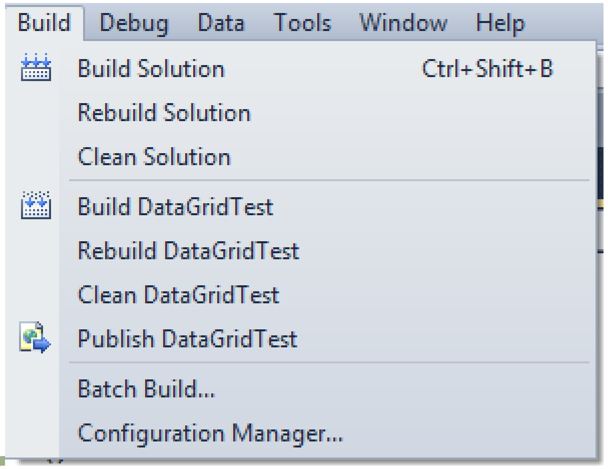




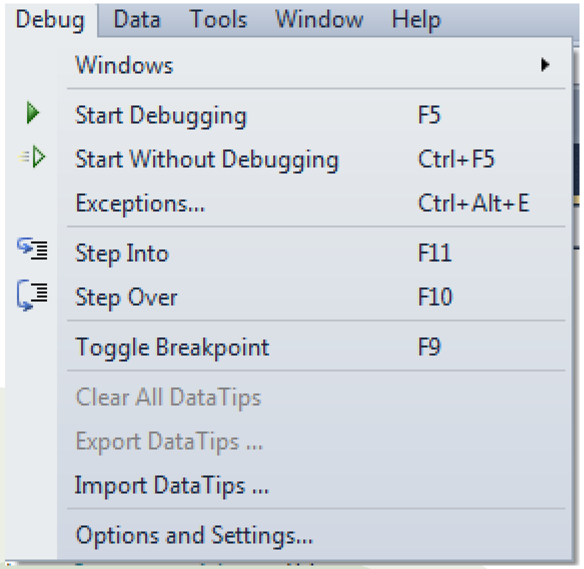
# Exploring the Visual C# Menus

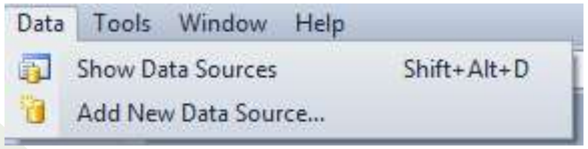

Refactor Project Build Debug Data Tool
Rename... Ctrl+R, Ctrl+R
Extract Method... Ctrl+R, Ctrl+M

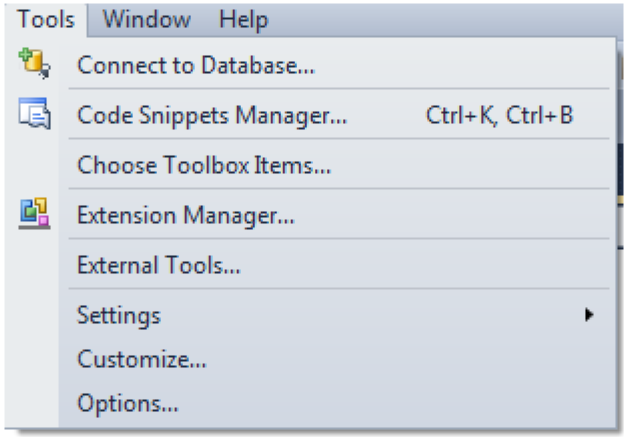

Project Build Debug Data Tools Window Help
Add Windows Form...
Add User Control...
Add Class...
Add New Item... Ctrl+Shift+A
Add Existing Item... Shift+Alt+A
Exclude From Project
Show All Files
Add Reference...
Add Service Reference...
Set as StartUp Project
Refresh Project Toolbox Items
DataGridTest Properties...



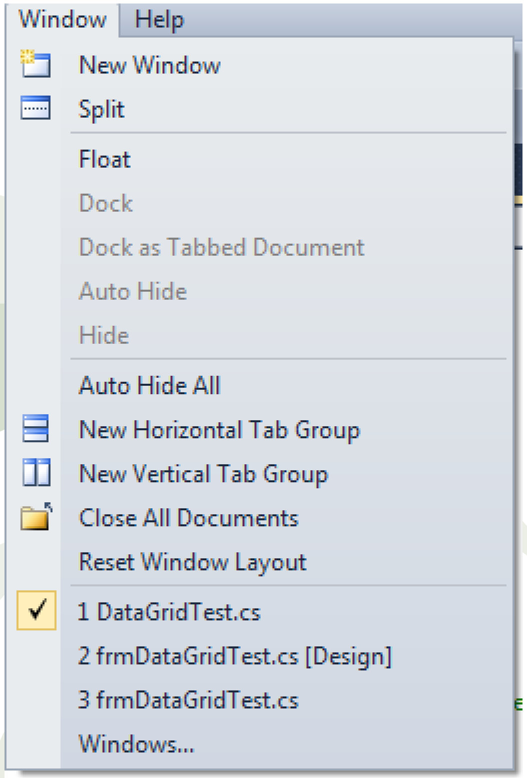
# Exploring the Visual C# Menus

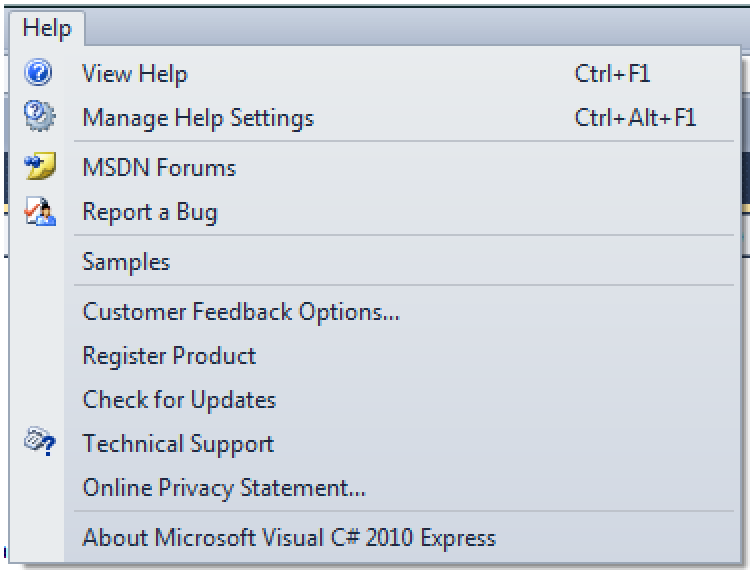






# Exploring the Visual C# Menus




Help
View Help Ctrl+F1
Manage Help Settings Ctrl+Alt+F1
MSDN Forums
Report a Bug
Samples
Customer Feedback Options...
Register Product
Check for Updates
Technical Support
Online Privacy Statement...
About Microsoft Visual C# 2010 Express

# Exploring the Visual C# Menus

## The File Menu

The File menu consists of the following menu items:

1. File → New → Project… – this menu item allows you to create a new project within your solution.
2. Open → Project/Solution… – this menu item allows you to open an existing project or solution. You have the option to add to the current solution or close the current solution in favor the what you are opening.
3. Open → File… -- this menu item allows you to open an existing file. Note that the file, even if it is a C# program, is not added to the Solution Explorer.
4. File → Close – this menu item closes the currently open active code or design window.
5. File → Close Solution – this menu item closes the solution completely.
6. Save *file.cs* – this menu item saves *file.cs*. Note that this includes the form.
7. Save *file.cs* As… – this menu item allows you to save *file.cs* to another name.
8. Advanced Save Options… – this menu item brings up the Advanced Save Options dialog box which allows you to the encoding of your program as well as which line ending to use (Carriage Return/Line Feed, etc.).
9. Save All – this menu item allows you to save all of the files within your project. Note that if a file has been changed, an asterisk appears to the right of the filename on the tab associated with a particular code window.
10. Export Template... – this menu item brings up the Export Template Wizard which allows you to save a project as a template for use when creating a new project.

# Exploring the Visual C# Menus

## The File Menu (continued)

11. Page Setup… – this menu item brings up the Page Setup dialog box which allows you to set up the orientation, margins, etc. for use when printing our your code.
12. Print… – this menu item brings up the oh-so-familiar-I-can-see-it-in-my-sleep Print dialog box used to print your code to a printer.
13. Recent Files → – this menu item lists recent files you've opened so you can open them again quickly and painlessly.
14. Recent Projects and Solutions → – this menu item lists recently opened projects and/or solutions so you can quickly and very painfully open them again.

## The Edit Menu

The Edit Menu consists of the following menu items:

1. Undo – when enabled, allows you to undo what you typed in. (Sorry, this option does not extend to those situations that you've created for yourself in the course of living your life. Try voodoo instead.) (Short-cut: CTRL-Z)
2. Redo – when enabled, allows you to undo what you've undone. (Short-cut: CTRL-Y)
3. Cut – this menu item allows you to cut selected text from a code window or a component from a form. (Short-cut: CTRL-X)
4. Copy – this menu item allows you to copy selected text from a code window or a component from a form. (Short-cut: CTRL-C)

# Exploring the Visual C# Menus

## The Edit Menu (continued)

5. Paste – this menu item allows you to paste text or components selected by using the Copy menu item. (Short-cut: CTRL-V)
6. Cycle Clipboard Ring – Each time you copy a piece of text using the Copy menu item or the CTRL-C short-cut, the entry is added to the clipboard ring. If you'd like to scan through that ring in order to find a piece of code you've hopefully copied at one point, you can click CTRL-SHIFT-Y continuously on a blank line in the code window until you come across the code you want. Then, hit the ESC button to keep the code in the window. (Sadly, hitting the Enter key will cause the desired code to disappear from the code window…poof!)
7. Delete – this menu item deletes selected text or a component. Similar to Cut. (Short-cut: DEL Button)
8. Select All – this menu item selects all text in a code window or all components on a form. (Short-cut: CTRL-A)
9. Find and Replace → Quick Find – this menu item brings up the Find and Replace dialog box which allows you to quickly find a piece of text.
10. Find and Replace → Quick Replace – this menu item brings up the Find and Replace dialog box which allows you to quickly replace text with other text.
11. Find and Replace → Find in File – this menu item allows you to search for text within a specific file.
12. Find and Replace → Replace in File – this menu item allows you to replace text within a specific file with other text.

# Exploring the Visual C# Menus

<u>The Edit Menu (continued)</u>

13. Find and Replace → Find Symbol – this menu item allows you to search for a specific class, method, property, etc. within your solution or even the references. You can also specify whether to search within your code or within a specific .NET Framework version.
14. Go To… – this menu item allows you to quickly go to a specific line number in your code.
15. Insert File As Text… – this menu item allows you to insert text (residing in a file) into your program at the point your cursor is located.
16. Advanced → Format Document – this menu item
17. Advanced → Format Selection – this menu item
18. Advanced → Tabify Selected Lines – this menu item turns the spaces in the selected code into tabs.
19. Advanced → Untabify Selected Lines – this menu item turns the tabs in the selected code into spaces.
20. Advanced → Make Uppercase – this menu item uppercases the selected text.
21. Advanced → Make Lowercase – this menu item lowercases the selected text.
22. Advanced → Delete Horizontal White Space – this menu item deletes the whitespace (spaces, tabs, etc.) at the beginning of the selected text. Effectively, the code becomes flush against the left edge of the code window.
23. Advanced → View White Space – this menu item allows you to see the white space (spaces, tabs, etc.) in your code. Spaces are shown as dashes, and tabs are shown as arrows.

# Exploring the Visual C# Menus

## The Edit Menu (continued)

24. Advanced → Word Wrap – this menu item allows code that is too long for the code window to wrap around so you don't have to scroll left and right.
25. Advanced → Incremental Search – this menu item quickly find what you are looking for in the code if you know the first few letters. Type CTRL-I, then type the first few letters and the IDE searches for them. Hit ESC to end.
26. Advanced → Comment/Uncomment Selection – this menu item comments out or uncomments out the selected code.
27. Advanced → Increase Line Indent – this menu item increases the indentation for the selected code.
28. Advanced → Decrease Line Indent – this menu item decreases the indentation for the selected code.
29. Bookmarks → Toggle Bookmark – this menu item marks the line where the cursor is presently located with a bookmark.
30. Bookmarks → Enable Bookmark – this menu item allows you to toggle enabling or disabling a bookmark. When a specific bookmark is enabled, that bookmark will be seen by the next or previous bookmark commands. If the bookmark is disabled, the next or previous bookmark commands will skip that bookmark.
31. Bookmarks → Previous Bookmark – this menu item searches for the previous bookmark.
32. Bookmarks → Next Bookmark – this menu item searches for the next bookmark.
33. Bookmarks → Clear Bookmarks – this menu item clears all of the bookmarks.

# Exploring the Visual C# Menus

## The Edit Menu (continued)

34. Bookmarks → Previous Bookmark in Document – this menu item
35. Bookmarks → Next Bookmark in Document – this menu item
36. Bookmarks → Add Task List Shortcut – this menu item will add a note to the task list. We talk about task lists when we go through the View menu.
37. Outlining → Hide Selection – this menu item allows you to hide a selection of code and then expand it by clicking on a plus sign (+) or collapse it by clicking on a minus sign (-). To hide code, select the code from just before the left curly brace to just after the last curly brace. This will allow the method name to appear with an ellipse (…) to the right indicating collapsed code.
38. Outlining → Toggle Outlining Expansion – this menu item alternately expands or collapses outlined code your cursor is on. (Short-cut: CTRL-MM)
39. Outlining → Toggle All Outlining – this menu item expands/collapses **all** of the code set up as outlined. (Short-cut: CTRL-ML)
40. Outlining → Stop Outlining – this menu item removes all outlines and shows all of the code. (Short-cut: CTRL-MP)
41. Outlining → Stop Hiding Current – this menu item removes the outlining effect for the code your cursor is on. (Short-cut: CTRL-MU)
42. Outlining → Collapse to Definitions – this menu item collapses all outlines to their smallest outline. (Short-cut: CTRL-MO)
43. Intellisense → Generate → Method/Constructor/Property/Field/Enum Member/New Type… – these menu items allow you to generate code for a method, constructor, property, field, enumerated type or a new type

# Exploring the Visual C# Menus

## The Edit Menu (continued)

44. Intellisense → Implement Interface – this menu item allows you to implement an interface.
45. Intellisense → Implement Abstract Class – this menu item allows you to implement an abstract class.
46. Intellisense → Organize Usings – these menu items allow you to remove unneeded usings, sort them in order, or both (remove and sort).
47. Intellisense → List Members – this menu item allows you to list the members of the .NET Framework depending on where your cursor is at the moment. If your cursor is on, say, DataTable, then members of the DataTable class are shown on the screen. Scroll down the list and hit the tab key when you find what you want. (Short-cut: CTRL-J)
48. Intellisense → Parameter Info – parameter information is normally shown when you type the left parenthesis of a method. You can use this menu item to invoke this feature manually by using this menu item.
49. Intellisense → Quick Info – normally, popup information appears when you hover your mouse cursor over the item. You can invoke this manually using this menu item.
50. Intellisense → Complete Word – normally, when you type a member that already exists, Intellisense attempts to help you out by displaying members in a list with similar spellings. You can manually invoke this by using this menu item.

## Exploring the Visual C# Menus

### The Edit Menu (continued)

51. Intellisense → Toggle Completion Mode – When you type in the code window, Intellisense attempts to help you out by making suggestions. There are two completion modes: Completion Mode and Suggestion Mode. See Microsoft's MSDN page on this topic for more: http://msdn.microsoft.com/en-us/library/exbffbc2.aspx.
52. Intellisense → Insert Snippet – this menu item allows you to quickly insert code snippets for statements such as while, switch, etc. Hit Tab to insert the snippet. (Short-Cut: CTRL-KX)
53. Intellisense → Surround With – this menu item surrounds select code with blocks of code like a do {…} while, for-loop, etc. (Short-Cut: CTRL-KS)

### The View Menu

The View menu consists of the following menu items:

1. View → Code – this menu item shows the code when you are in the design window.
2. View → Designer – this menu item shows you the design window when you are in the code window.
3. View → Open – this menu item opens the code when you are in design mode.
4. View → Open With… – this menu item displays the Open With… dialog box which allows you to open the code in Notepad, XML Editor, etc.

## Exploring the Visual C# Menus

The View Menu (continued)

5. View → Solution Explorer – this menu item opens the Solution Explorer.
6. View → Database Explorer – this menu item opens the Database Explorer window.
7. View → Class View – this menu item opens the Class View for the project.
8. View → Object Browser – this menu item opens the Object Browser which allows you to search and peek through the namespaces, classes, methods, etc. (Short-Cut: CTRL-ALT-J)
9. View → Error List – this menu item brings the Error List tab located on the bottom of the IDE to the foreground.
10. View → Output – this menu item brings the Output tab located on the bottom of the IDE to the foreground.
11. View → Start Page – this menu item displays the Start Page (the page you see when you first come into Visual C#)
12. View → Task List – this menu item brings the Task List tab located on the bottom of the IDE to the foreground. We talk more about the Task List later on in the presentation.
13. View → Toolbox – this menu item displays the Toolbox.
14. View → Find Results → Find Results – this menu item displays the Find Results tab at the bottom of the IDE.
15. View → Find Results → Find Symbol Results – this menu item displays the Find Symbol Results tab at the bottom of the IDE.

# Exploring the Visual C# Menus

## The View Menu (continued)

16. View → Other Windows → Web Browser – this menu item displays a Web Browser from within Visual Studio. (Short-Cut: CTRL-ALT-R)
17. View → Other Windows → Document Outline – this menu item opens the Document Outline window when you are displaying a form. It shows the logical structure of the form as a hierarchy. This displays nothing when you are in a code window.
18. View → Toolbars – these menu items allow you to show or hide toolbars. By default, the Standard Toolbar and the Text Toolbar are displayed.
19. View → Full Screen – this menu item allows you to view the Visual Studio IDE in full screen mode. (Short-Cut: SHIFT-ALT-ENTER)
20. View → Navigate Backward/Forward – The IDE will place invisible markers in your code window(s) when you perform certain actions such as searching, using the Go To line # feature, dropping in text from a file, etc. Clicking on View Navigate Backward/Forward takes you to this wonderful places in your code.
21. View → Next/Previous Task – these menu items take you to the next or previous task. We talk about Tasks later on.
22. View → Properties – this menu item displays the Properties pages for a Solution or for a Project.

## Exploring the Visual C# Menus

### The Refactor Menu

The Refactor menu is only available when you are in the code window and consists of the following menu items:

1. Refactor → Rename… – this menu item displays the Rename dialog box which allows you to rename a symbol to a new name. Note that you must first select the text of the symbol you want to rename, then you can click on this menu item!
2. Refactor → Extract Method… – this menu item displays the Extract Method dialog box allowing you to create a method from the selected code. For example, if you select the following code: `DataSet oDS = new DataSet();`, then the result of the refactoring is as follows: (1) the code above is replace by `DataSet oDS = MyNewMethod();` and (2) the following method is injected:

```
private static DataSet MyNewMethod()
{
    DataSet oDS = new DataSet();
    return oDS;
}
```

### The Project Menu

The Project menu consists of the following menu items:

1. Project → Add Windows Form… – this menu item actually displays the Add New Item dialog box with the Windows Form option already highlighted. Your job is to give it a name and then click the OK button.

# Exploring the Visual C# Menus

## The Project Menu (continued)

2. Project → Add User Control… – this menu item actually displays the Add New Item dialog box with the User Control option already highlighted. Your job is to give it a name and then click the OK button.
3. Project → Add Class… – this menu item actually displays the Add New Item dialog box with the Class option already highlighted. Your job is to give it a name and then click the OK button.
4. Project → Add New Item… – this menu item displays the aforementioned Add New Item dialog box only this time nothing is highlighted, so you're going to have to do some work here and highlight something, give it a name and then click OK.
5. Project → Add Existing Item… – this menu item displays the Add Existing Item dialog box (which is just the File Open dialog box with a fancy name) and it allows you to add an item (such as C# code) into your project. Note that this added item is included in the Solution Explorer.
6. Project → Exclude From Project/Include In Project – this menu item allows you to tell Visual Studio that a particular C# code module is to be excluded from the project. Note that the file is neither deleted from the project nor is it removed from disk, it is just not included in the build of the project. Once excluded, you can included it again by clicking on Project → Include In Project.
7. Project → Show All Files – this menu item will show all of the excluded files (see above) in the Solution Explorer window. This is a toggle switch, so clicking it again will hide all of the excluded files.

# Exploring the Visual C# Menus

## The Project Menu (continued)

8. Project → Add Reference… – this menu item displays the Add Reference dialog box. (We discusses this earlier.) The selection is added to the References section in the Solution Explorer.
9. Project → Add Service Reference – this menu item displays the Add Service Reference dialog box. This, too, will be added to the References section in the Solution Explorer.
10. Project → Set as StartUp Project – Since a solution can contain multiple projects, you can choose one to be the start up project.
11. Project → Refresh Project Toolbox Items – this menu item refreshes the toolbox. See this article for more: [http://dan9298.blogspot.com/2009/01/visual-studio-and-auto-toolbox-populate.html](http://dan9298.blogspot.com/2009/01/visual-studio-and-auto-toolbox-populate.html).
12. Project → [ProjectName] Properties – this menu item displays the Properties pages for the project named [ProjectName]. We talk more about these properties pages later on in the presentation.

## The Build Menu

The Build menu contains the following menu items:

1. Build → Build Solution – this menu item builds the solution; that is, all projects that are in the solution are built. If a project has not changed since the previous build, then it is not built.
2. Build → Rebuild Solution – this menu item performs a Clean then a Build.

## Exploring the Visual C# Menus

The Build Menu (continued)

3. Build → Clean Solution – this menu item removes any intermediate files leaving the project-specific files only.
4. Build → Build [ProjectName] – this menu item builds a specific project.
5. Build → Rebuild [ProjectName] – this menu item cleans and bulids a specific project.
6. Build → Clean [ProjectName] – this menu item cleans a specific project.
7. Build → Publish [ProjectName] – this menu item starts the Publish Wizard which allows you to publish your project for download from a website, installation from CD-ROM/DVD-ROM or from a UNC path or file share.
8. Build → Batch Build… – this menu item displays the Batch Build dialog box which allows you to choose one or more build types such as debug mode, release mode, etc.  This is different from the other options on the Build menu in that it allows you to build more than one "thing" with the click of a button rather than having to continually change the from Debug to Release and from x86 to x64.
9. Build → Configuration Manager – this menu item displays the Configuration Manager dialog box.  We've discusses this dialog box before.

# Exploring the Visual C# Menus

## The Debug Menu

The Debug menu contains the following menu items:

1. Debug → Window → Output – this menu item brings up the Output tab at the bottom of the IDE.
2. Debug → Window → Immediate – this menu item brings up the Immediate tab at the bottom of the IDE.
3. Debug → Start Debugging – this menu item starts the program with debugging turned on. (Short-Cut: F5)
4. Debug → Start without Debugging – this menu item starts the program with debugging turned off. (Short-Cut: CTRL-F5)
5. Debug → Exceptions… – this menu brings up the Exceptions dialog box. This allows you to choose amongst the Common Language Runtime Exceptions and Managed Debugging Assistants which will cause a break in the program if one or more occurs either when the program does not handle the error or when the error is thrown. You also have the option to add a new exception.
6. Debug → Step Into – this menu item allows you to step into code when a breakpoint has been reached. (Short-Cut: F11)
7. Debug → Step Over – this menu item allows you to step over code when a breakpoint has been reached. (Short-Cut: F12)
8. Debug → Toggle Breakpoint – this menu item allows you to set the breakpoint on the line where the cursor is located, or alternately unset the breakpoint.

## Exploring the Visual C# Menus

### The Debug Menu (continued)

9. Debug → Clear All Data Tips – this menu item clears all data tips.
10. Debug → Export Data Tips – this menu item allows you to export all of the data tips within your project.
11. Debug → Import Data Tips – this menu item brings up the Import Data Tips dialog box (just the File…Open dialog box) which allows you to import data tips saved by the export feature.
12. Debug → Options and Settings… – this menu item brings up the Options dialog box which allows you to make changes to a variety of IDE-related settings such as having line numbers in the text editor, etc. We discuss the Options dialog box later in the presentation.

### The Data Menu

The Data Menu contains the following menu items:

1. Data → Show Data Sources – this menu item brings up the Data Sources pane at the left of the IDE and it lists all of the data sources your have created within your project(s).
2. Data → Add New Data Source… – this menu item brings up the Data Source Configuration Wizard which allows you to create a data source from a database, service or other object. The Wizard also allows you to create a new connection to a database.

# Exploring the Visual C# Menus

## The Tools Menu

The Tools Menu contains the following menu items:

1. Tools → Connect to Database… – this menu item brings up the Database Explorer pane at the left of the IDE as well as the Choose Data Source dialog box.
2. Tools → Code Snippets Manager… – this menu item brings up the Code Snippets Manager which allows you to manage and add additional snippets, remove an existing snippet and import snippets. Note that snippets related to Visual C# are files located in `C:\Program Files (x86)\Microsoft Visual Studio 10.0\VC#\Snippets\1033\Visual C#` and the extension is `.snippet`. User-create snippets are located in `C:\Users\Scott\Documents\Visual Studio 2010\Code Snippets\Visual C#\My Code Snippets`.
3. Tools → Choose Toolbox Items… – this menu item brings up the Choose Toolbox Items dialog box which allows you to add or remove .NET, COM, WPF and Silverlight components in the Toolbox.
4. Tools → Extension Manager – this menu item brings up the Extension Manager which allows you to add extensions to your project such as templates, controls and tools available either locally or online. Many of these extensions are free or free editions of for-purchase products (such as LightningChart Basic, TX Text Control, Telerik Rad Calendar and more).
5. Tools → External Tools – this menu item brings up the External Tools dialog box which allows you to add your favorite programs to Visual Studio such as TextPad, etc. Your tools are add to the Tools menu as a menu item.

# Exploring the Visual C# Menus

## The Tools Menu (continued)

6. Tools → Settings → Basic Settings – this menu item shows basic settings for the IDE.
7. Tools → Settings → Expert Settings – this menu item show expert settings for the IDE. I recommend this setting.
8. Tools → Settings → Reset… – this menu item resets all of your settings.
9. Tools → Settings → Import and Export Settings – this menu item allows you to export your settings which can then be imported into Visual Studio at another location (such as at home, at work, at the local pub, etc.).
10. Tools → Customize – this menu item brings up the Customize dialog box which allows you to modify toolbars, create a custom toolbar, and rearrange menus, toolbars, and context menus.
11. Tools → Options – this menu item brings up the Options menu. This is the same as the Debug→ Options and Settings dialog. Make sure to click the Show All Settings checkbox in order to see all of the options.

## The Window Menu

The Window menu consists of the following menu items:

1. Window → New Window – this menu item duplicates the code window you are on.

The remaining menu items are familiar to Windows users and we won't go into them here.

## Exploring the Visual C# Menus

The Help Menu

The Help menu consists of the following menu items:

1. Help → View Help – this menu item brings up the Help webpage for Visual C# in an external browser.
2. Help → Manage Help Settings – this menu item brings up the Help Library Manager which allows you to choose online or local help, check for updates online, install content from online, install content from disk, and remove content (not that you would ever do that!!).
3. Help → MSDN Forums – this menu item brings up the Visual Studio General Forums webpage within Visual Studio (rather than an external browser).
4. Help → Report a Bug – this menu item brings up the Microsoft Connect site for Visual Studio which allows you to submit feedback for Visual Studio 2010 and the .NET Framework 4. This opens up in Visual Studio (rather than an external browser).
5. Help → Samples – this menu item brings up the Visual Studio 2010 Code Samples webpage within Visual Studio's browser.
6. Help → Customer Feedback Options… – this menu item allows you to choose whether to participate in the Customer Experience Improvement Program.
7. Help → Register Product – this menu item allows you to register your Visual C# 2010 Express Edition.
8. Help → Check for Updates – this menu item allows you to check for product updates.

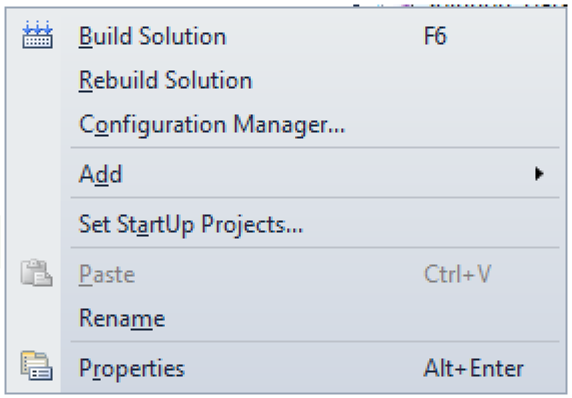
# Exploring the Visual C# Menus

The Help Menu (continued)

9. Help → Technical Support – this menu item brings up the Technical Support Options for Visual Studio webpage in an external browser.
10. Help → Online Privacy Statement… – this menu item brings up the Online Privacy Statement webpage in Visual Studio's internal browser.
11. Help → About Microsoft Visual C# 2010 Express – this menu item brings up the About dialog box for the product.

## Exploring the Context Menus

Context menus, or popup menus, are displayed by right-clicking on certain items within Visual Studio. For example, the Solution Explorer context menu is displayed when you right-click on the solution name within the Solution Explorer pane. The Code Window context menu is displayed when you right-click within the code window. The Form Window context menu is displayed when you right-click on a form. A lot of these context menus are just repeats of the menus that appear in the menus at the top of the IDE. For example, the Solution Explorer context menu, shown below, has the familiar choices: Build Solution, Rebuild Solution, Configuration Manager, etc.

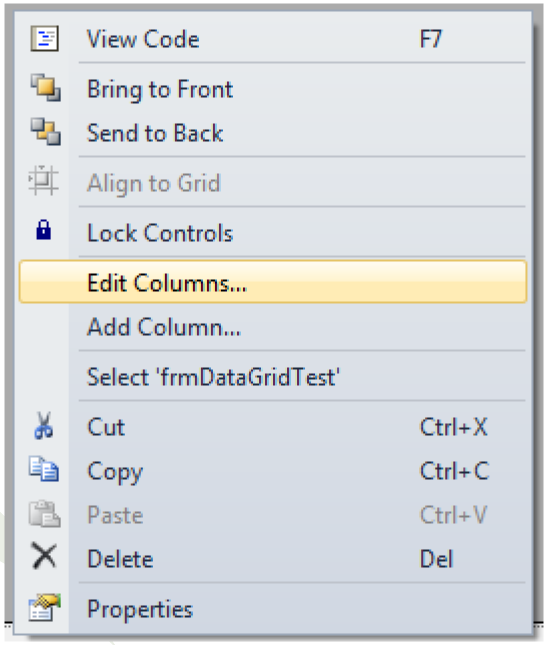


Because the context menus are repeats of menu items appearing in the menus at the top of the IDE, we won't go into all of them, but I do want to point out some nice menu items in certain context menus.

For example, while in a code window, the context menu has the option to Run To Cursor. This menu item allows you to run the code up to the point where your cursor is located. Once the cursor has been reached, you are in Debug Mode and you can Step Into or Step Over code, etc.

## Exploring the Context Menus

While on a form, the context menu that appears depends on the control your mouse point is hovering over at the moment. For example, if you right-click on the DataGrid, you will see the following context menu:



Now, this context menu is similar to other context menus for controls, except that it contains two additional menu items: Edit Columns… and Add Column… This is true of other controls as well, such as the Tab Control whose context menu contains the standard menu items as well as the Add Tab and Remove Tab menu items.

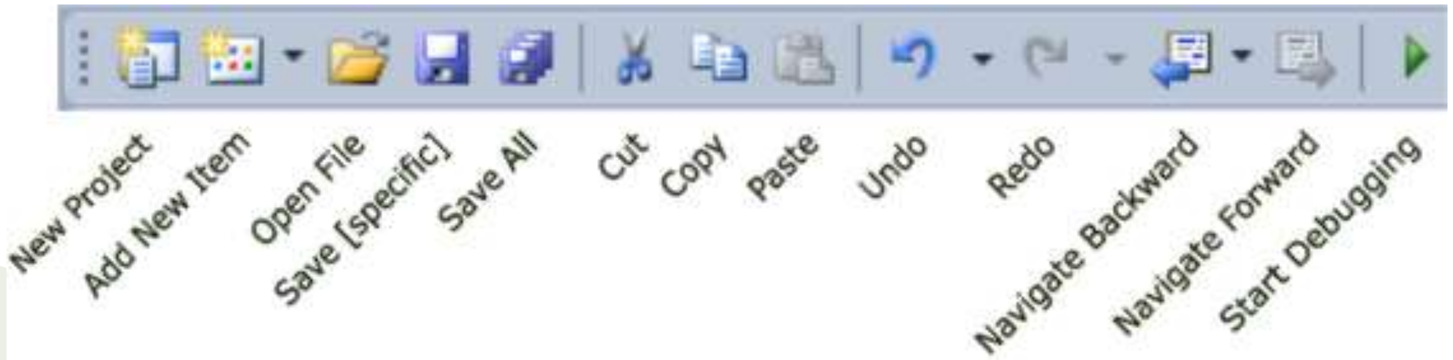
## Exploring the Context Menus

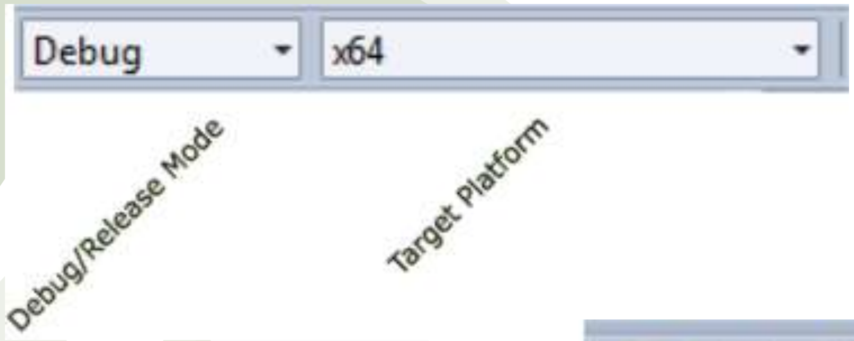
Another nice feature is the Add menu which appears on the context menu for a project listed in the Solution Explorer. The Add Menu features the following menu items: New Item, Existing Item, New Folder, Windows Form, User Control, Class. These are nice if you want to add items quickly to your project.

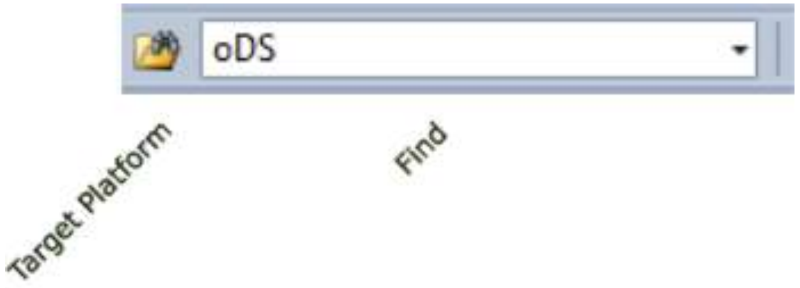
One nice feature is the New Folder menu item. This allows you to add a folder to your project, but that folder actually appears on disk as well under the project itself. It's probably a nice place to keep image files, etc.

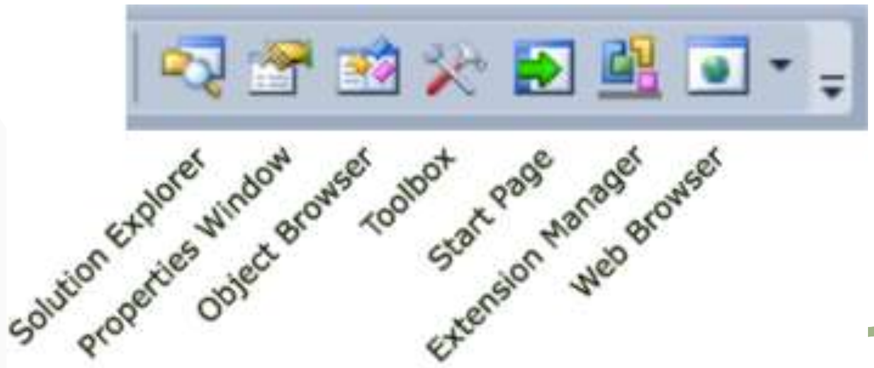
## Exploring the Standard Toolbars

To make your life easier, it's probably a good idea to familiarize yourself with the toolbars that appear within the IDE.  Below is a description of the standard toolbars.

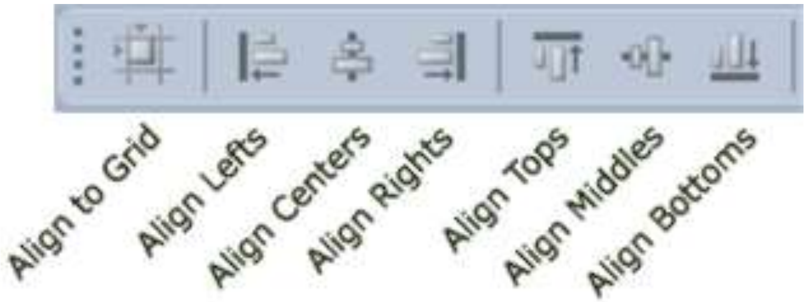


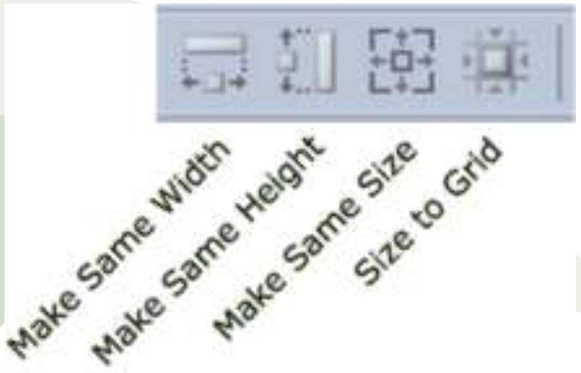


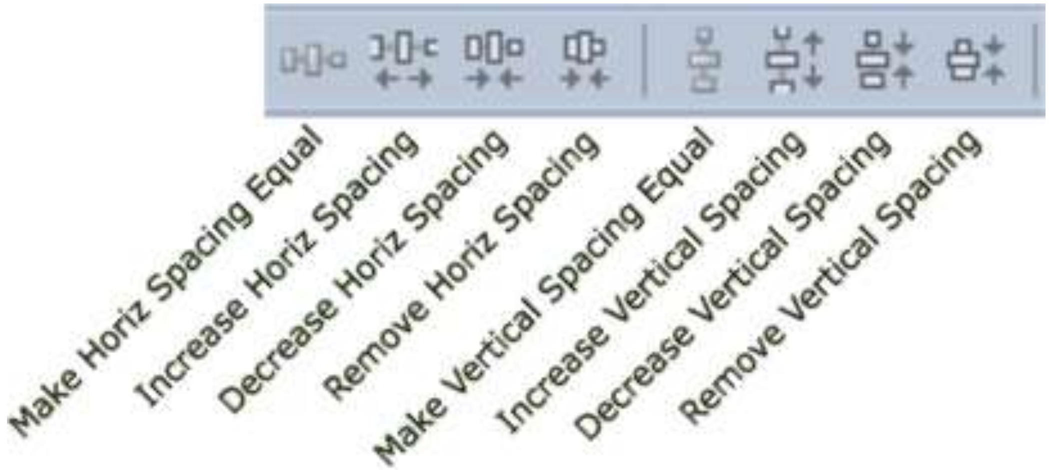


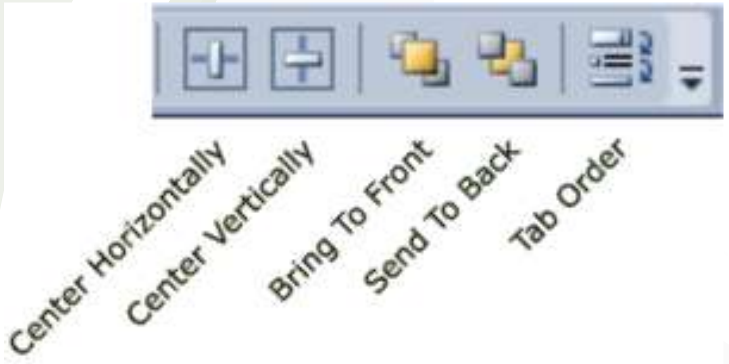


# Exploring the Standard Toolbars

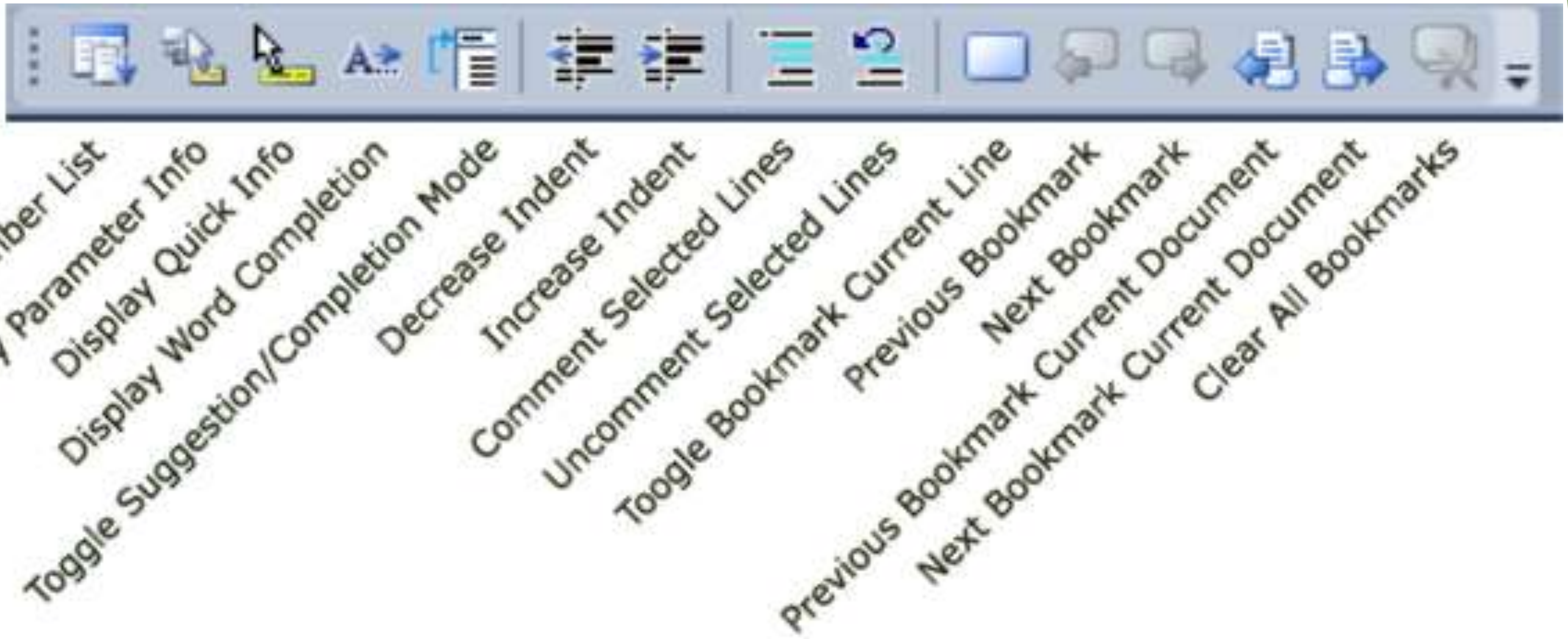








# Exploring the Standard Toolbars



## Understanding Tasks and the Task List

If you are working on a sizable project, unless you have an excellent memory, you're going to have to jot down notes to yourself on what you need to add to the code, what you have to fix, etc.  In Visual Studio, the Task List feature can save you from being inundated with sticky notes all over your office.

There are two varieties of Task List : Comments and User Tasks.

You can gather together the text within your code's comments based on certain keywords that appear at the beginning of the comment itself.  For example, normally you'd write a comment like this:

```
//Damn!  I have to add the code to send emails to everyone!
```

Instead, if you write the comment like this…

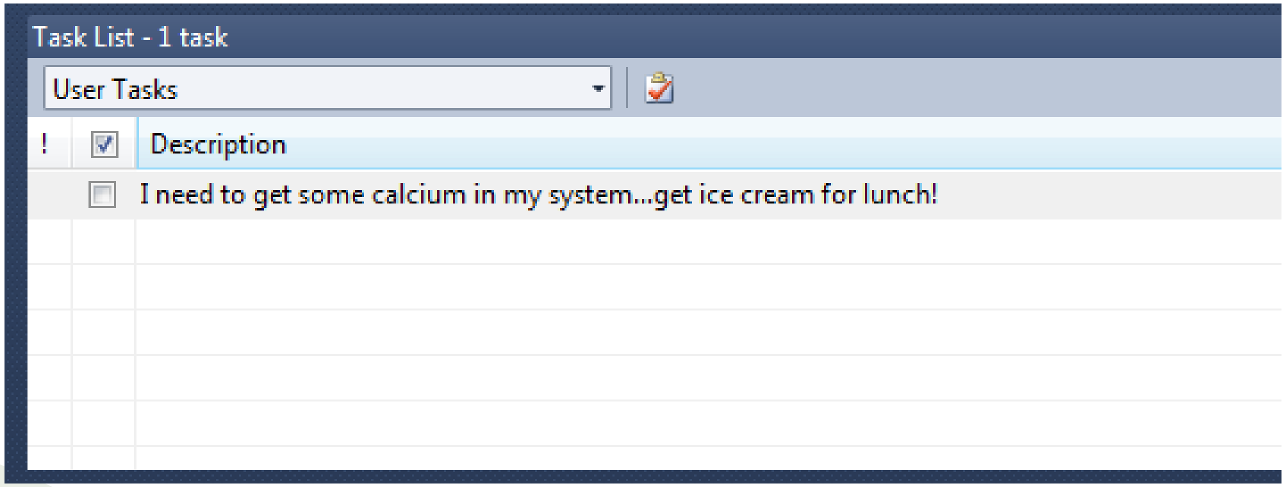
```
//TODO: I have to add the code to send emails to everyone!
```

…this reminder is placed in the Task List (View…Task List) as a reminder.

The second variety is the User Tasks.  You can add a user task to the Task List by selecting User Tasks from the drop-down box in the Task List window, clicking the Create User Task button (just to the right of the drop-down box), and adding your own task.  See next slide for an example of this.

# Understanding Tasks and the Task List

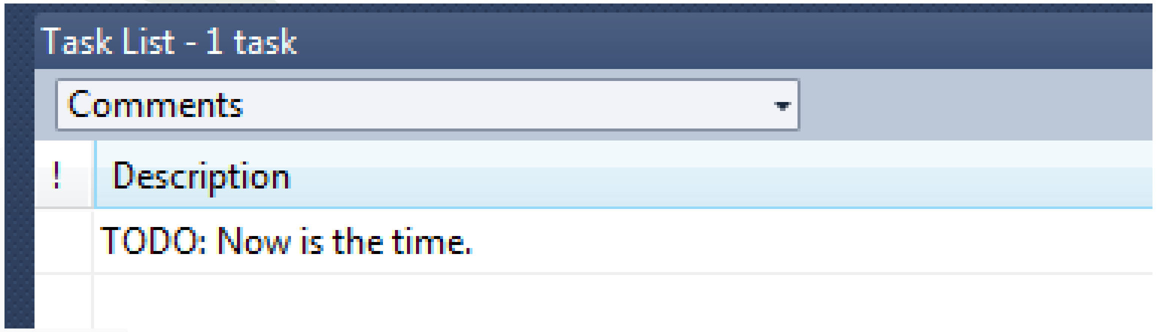
sheepsqueezers.com


Task List - 1 task

User Tasks

| ! | ☑ | Description |
|---|---|---|
| | ☐ | I need to get some calcium in my system...get ice cream for lunch! |

To see the comment variety of the Task List, click the drop-down box and select Comment. You will see something like the following:

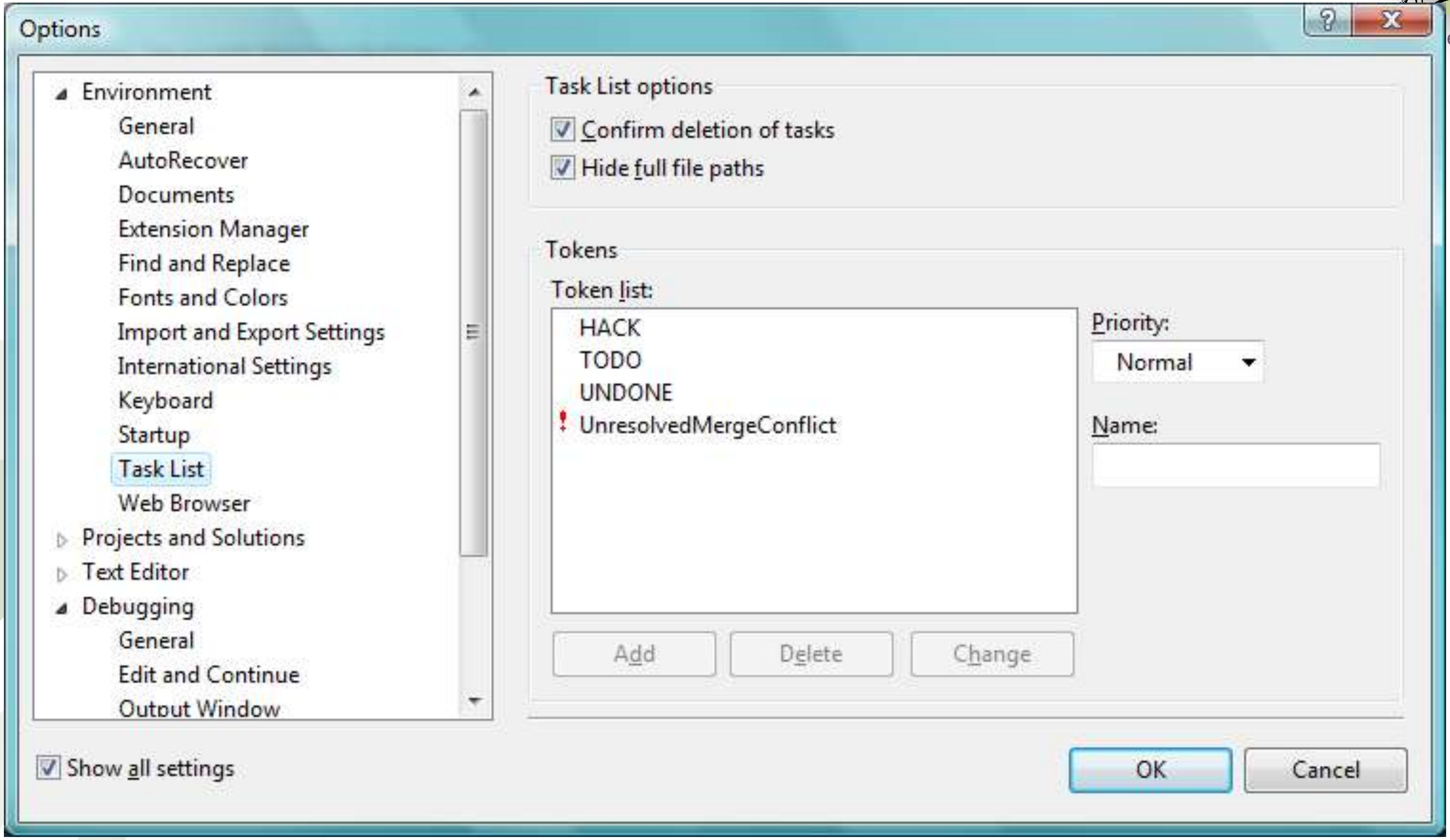

Task List - 1 task

Comments

| ! | Description |
|---|---|
| | TODO: Now is the time. |

Now, you're not limited to the word TODO. By default, the tokens TODO, HACK and UNDONE. You can add your own tokens to this list by going to the Options menu and clicking on Task List under the Environment section. Fill in the name of the token in the input box and click Add to add it to the token list. You can also Delete or Change any token. See next slide for an example of this.

# Understanding Tasks and the Task List



Options

- Environment
  - General
  - AutoRecover
  - Documents
  - Extension Manager
  - Find and Replace
  - Fonts and Colors
  - Import and Export Settings
  - International Settings
  - Keyboard
  - Startup
  - Task List
  - Web Browser
- Projects and Solutions
- Text Editor
- Debugging
  - General
  - Edit and Continue
  - Output Window

Task List options

- [x] Confirm deletion of tasks
- [x] Hide full file paths

Tokens

Token list:

- HACK
- TODO
- UNDONE
- ! UnresolvedMergeConflict

Priority: Normal

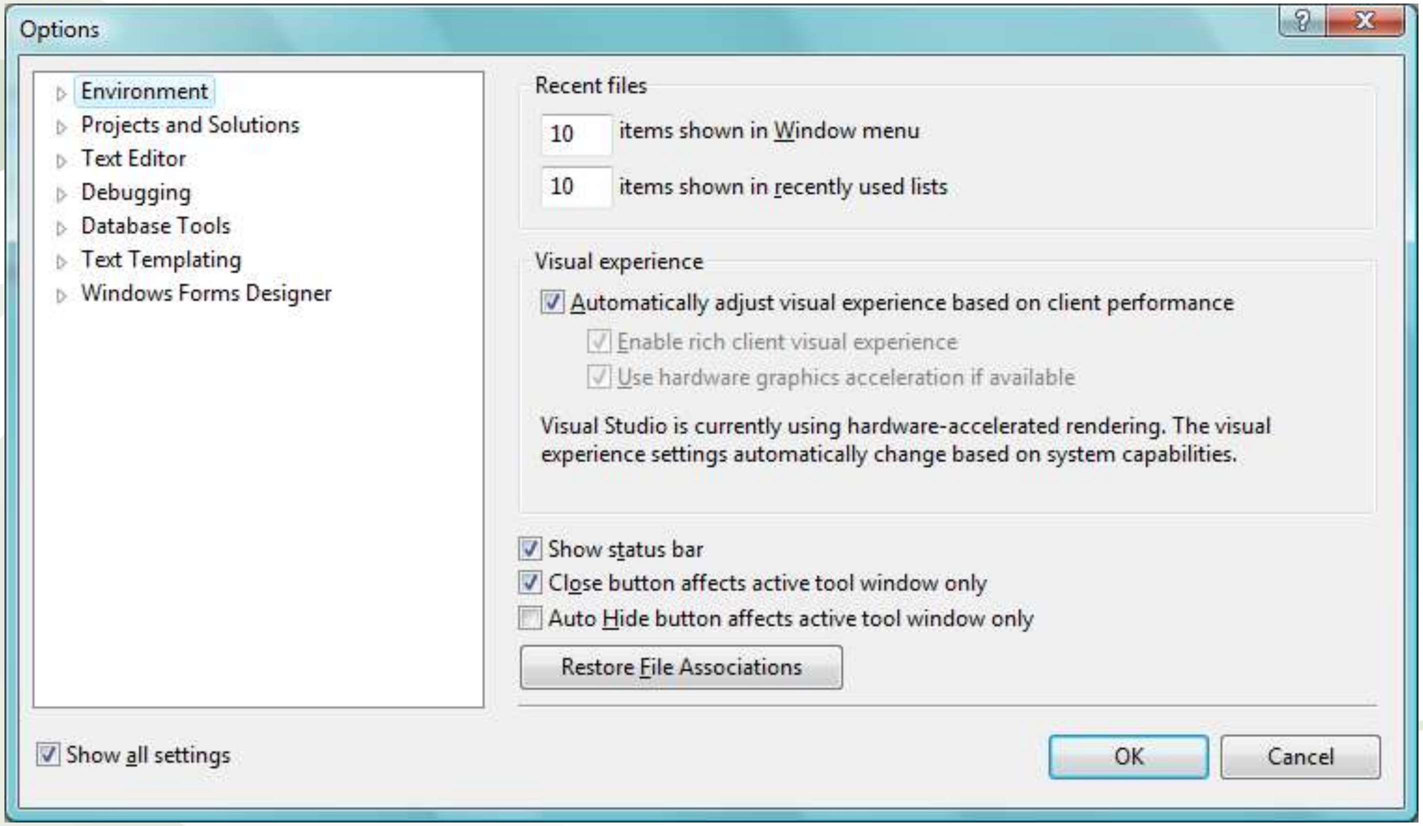
Name:

- [x] Show all settings

Back in the Task List window, if you are viewing the User Tasks, you can indicate that you are done with that task by clicking the checkbox to the left of the task. For Comments, if you remove the comment in the code, or just remove the token from the comment, that task is removed from the Task List.

## The Options Dialog Box

What's life without options, right? Well, Visual C# has a lot of options to offer in the Options dialog box (shown below). You can gain access to this dialog box from either the Options menu item off the Tools menu, or the Options and Settings menu item off the Debug menu. Both methods bring up the same dialog, but the Debugging section is opened if you click on Debug → Options and Settings. As you can see below, there are seven major sections of options available from the Options dialog: Environment, Projects and Solutions, Text Editor, Debugging, Database Tools, Text Templating, and Windows Forms Designer.



10 items shown in Window menu
10 items shown in recently used lists

☑ Automatically adjust visual experience based on client performance
☑ Enable rich client visual experience
☑ Use hardware graphics acceleration if available

Visual Studio is currently using hardware-accelerated rendering. The visual experience settings automatically change based on system capabilities.

☑ Show status bar
☑ Close button affects active tool window only
☐ Auto Hide button affects active tool window only

☑ Show all settings

## The Options Dialog Box

Note that each of these seven sections contains several subsections. For instance, the Projects and Solutions section contains the General subsection and the Build and Run subsection.

Also, be aware that when you click on a section name (such as Environment, or Text Editor), it's the first subsection that appears on the right side. That is, when you click on the Environment section, the General subsection – the first subsection appearing below the word Environment – appears to the right. Put it another way: each section name is just a header and contains no dialog information itself.

While we won't go through every option available to you (that would be insane!), we do want to point out some nice options you should be aware of.

The Environment section consists of the following subsections:

1. General – allows you to set how many items can appear in the Windows menu and if the status bar should be shown, among others.
2. AutoRecover – allows you to set autorecovery options such as the number of minutes that needs to pass before an autosave is performed and how many days to keep the autorecovery information.
3. Documents – allows you to set document-related information.
4. Extension Manager – allows you to enable/disable extensions within Visual Studio Gallery and to automatically check for updates of installed extensions.
5. Find and Replace – options related to the Find and Replace menu.

## The Options Dialog Box

6. Fonts and Colors – options allowing you to set the font and colors for items appearing in the code window
7. Import and Export Settings – allows you to specify a file where your exported settings are saved.
8. International Settings – allows you to choose your language.
9. Keyboard – Keyboard and shortcut options.
10. Startup – options related to the startup of Visual C# such as whether to show the Start Page or not and how often to refresh the page.
11. Task List – Task List related options.  See the section on using the Task List earlier in this presentation.
12. Web Browser – Browser-related options such as a default home page and search page.

The Projects and Solutions section consists of the following subsections:

1. General – settings to change the default projects, templates and item templates locations, among others.
2. Build and Run – settings to change the number of parallel projects to build at once, whether to save all changes to the project(s) before a build, etc.

The Text Editor section consists of the following subsections:

1. General – settings to allow/disallow drag-and-drop, the display of horizontal and vertical scrollbars, etc.

## The Options Dialog Box

2. File Extension – allows you to include non-standard file extensions and how they are to be opened.
3. All Languages – settings to apply across the board for all languages such as C#, VB, etc.  as well as tab settings.
4. C# – settings for C#-specific language.  The Formatting subsection allows you to control the indentation of code, new lines are used with code, etc.

The Debugging section consists of the following subsections:

1. General – this section contains a lot of options such as to enable/disable the exception assistant, enable/disable highlighting of source line for breakpoints or current line, etc.
2. Edit and Continue – settings related to Edit and Continue, etc.
3. Output Window – settings allowing you to turn on or off what is displayed in the Output Window.  By default, everything is turned on.
4. Symbols – settings related to Symbols.

The Database Tools section consists of the following subsections:

1. General – settings related to view and table designers as well as editors and execution of scripts.
2. Data Connections – settings related to how many rows can be displayed in the Output Window, and the desired SQL Server Instance name.
3. O/R Designer – this section has one setting allowing object names to be plural.

## The Options Dialog Box

4. Query and View Designers – settings such as how long to wait until a query is killed, panes shown by default, etc.
5. Table and Database Designers – settings related to table and database designers.

The Text Templating section consists of the following subsection:

1. Text Templating – this section contains one option: Show Security Message which is set to true.

The Windows Forms Designer section consists of the General subsection and the Data UI Customization subsection. See the following webpage for more information: http://msdn.microsoft.com/query/dev10.query?appId=Dev10IDEF1&l=EN-US&k=k(VS.TOOLSOPTIONSPAGES.WINDOWSFORMSDESIGNER.DATA_UI_CUSTOMIZATION);k(TargetFrameworkMoniker-%22.NETFRAMEWORK%2cVERSION%3dV4.0%22);k(DevLang-CSHARP)&rd=true.

Now, the following options changes are what I use:

1. Environment → Extension Manager – check the checkbox Automatically check for updates to installed extensions.
2. Projects and Solutions → General – check the checkbox for Show advanced build configurations.

*…continued…*

# The Options Dialog Box

3. Projects and Solutions → Build and Run – check the checkbox for For new solutions use the currently selected project as the startup project. (I may regret this option later on, but it sounds good now!)
4. Text Editor → All Languages → General – check the checkbox for Line numbers.
5. Text Editor → C# → Formatting → Spacing – check the checkbox for Insert space after keywords in control flow statements (this is my preference).
6. Text Editor → C# → Formatting → Spacing – check the checkbox for Insert space after comma (this is my preference).
7. Text Editor → C# → Formatting → Spacing – check the checkbox for Insert space after cast (this is my preference).
8. Debugging → General – check the checkbox for Highlight entire source line for breakpoints and current statement.
9. Database Tools → Data Connections – check the checkbox for Limit SQL results sent to Output window to: and then fill in the desired number. 500 is the default and that's fine.

## Property Pages for a Solution

If you go to the Solution Explorer and right-click on the solution name, the context menu pops up. If you click on Properties, the Solution 'SolutionName' Property Pages dialog box is displayed. This dialog box consists of two sections: Common Properties and Configuration Properties. The Configuration Properties we've talked about before, so we won't go into it here. The Common Properties section consists of the following subsections:

1. Startup Project – this section allows you to choose either a single project as the startup project or multiple startup projects.
2. Project Dependencies – this section shows you dependencies for each of your projects.
3. Debug Source Files – this section allows you to add additional or remove existing directories containing source code.

## Property Pages for a Project

If you go to the Solution Explorer and right-click on a project name, the context menu pops up. If you click on Properties, the project properties dialog appears as a tab in the main IDE window. Note that there is no dialog box to popup, per se.

The project properties consists of the following tabs:

1. Application – this section allows you to specify the assembly name, the namespace name, the target .NET Framework, the type of output (windows application, console application, or class library, the startup object as well as assembly information such as version and file information. You can also specify an icon to be used with the application.
2. Build – this section allows you to specify the configuration and platform as well as additional compilation symbols (such as DEBUG), errors and warning levels, as well as where to place the output.
3. Build Events – have no idea what this is! Must come back to this.
4. Debug – this section allows you to specify the configuration and platform settings, command line arguments, working directory, etc.
5. Resources – this section allows you to add resources such as images, strings, icons, audio, text files, etc. into your project.
6. Settings – "Application settings allow you to store and retrieve property settings and other information for your application dynamically. For example, the application can save a user's color preferences, then retrieve them the next time it runs. See more at: http://msdn.microsoft.com/query/dev10.query?appId=Dev10IDEF1&l=EN-US&k=k(APPLICATIONSETTINGSOVERVIEW);k(TargetFrameworkMoniker-%22.NETFRAMEWORK%2cVERSION%3dV4.0%22)&rd=true.

## Property Pages for a Project

7. Reference Paths – will have to come back to this.
8. Signing – this section allows you to sign the ClickOnce manifests as well as sign an assembly with a strong name key file.
9. Security – this section allows you to specify if your application is a full trust application or partial trust application.
10. Publish – this section allows you to specify the location where you will publish your finalized application, the install mode and settings as well as the publishing version number.

## Our Fifth (and Visually Stunning) C# Project

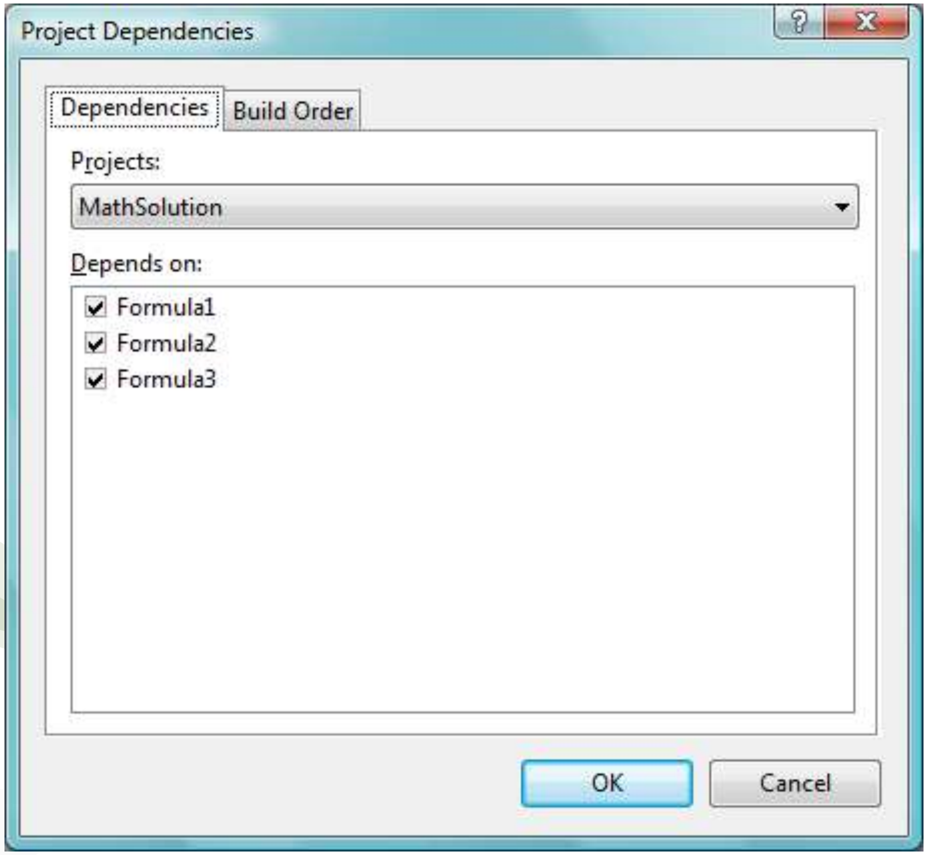
This next project I'd like to create a solution, and instead of just one project, I'd like to have multiple projects. I'd like one of those projects to be a Windows Forms Application, while the other projects are to be Class Libraries. Each class library will contain a single mathematical formula that will return a Double upon being passed one or more parameters. I'd like the Windows Forms Application project to contain an input box to hold a numeric value, and several buttons, each button will be associated with a specific Class Library which will be associated with a mathematical formula.

As usual, start Visual C# and create a new solution called the MathSolution and choose Windows Forms Application when asked.

Next, add a project to the solution by right-clicking on the solution name in the Solution Explorer and clicking Add → New Project… When the New Project dialog box appears, select Class Library and name it Formula1. Repeat this two more times adding Formula2 and Formula3.

Now, before we code anything, let's set the Project Dependencies. Since we are building three class libraries (DLLs) all of which will be used by the Windows Forms Application MathSolution, we need to tell Visual Studio this fact. Right-click on the project MathSolution, then click on the Project Dependencies… menu item on the context menu. This will bring up the Project Dependencies dialog box (as shown on the next page).

# Our Fifth (and Visually Stunning) C# Project


Project Dependencies

Dependencies | Build Order

Projects: MathSolution

Depends on:
- [✓] Formula1
- [✓] Formula2
- [✓] Formula3

Ensure that the project MathSolution is chosen from the Projects drop-down box. Next, ensure that all of the dependencies are checked within the Depends on: window. Click OK.

Next, let's add code to our three class libraries.

## Our Fifth (and Visually Stunning) C# Project

In the Solution Explorer, in the Formula1 project, change the name Class1.cs to clsFormula1.cs. Do the same for Formula2 (clsFormula2.cs) and Formula3 (clsFormula3.cs).

Next, double-click on clsFormula1.cs and replace the code within that module with the code below:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

namespace Formula1
{
    public static class clsFormula1
    {
        public static Double Formula1(Double dValue)
        {
            return( Math.Pow(dValue,2) + 1 );
        }

    }
}
```

Take note that I've changed the class to a static class and the single method within that class is also static.

Do the same thing for the remaining two modules:

# Our Fifth (and Visually Stunning) C# Project

## clsFormula2.cs:

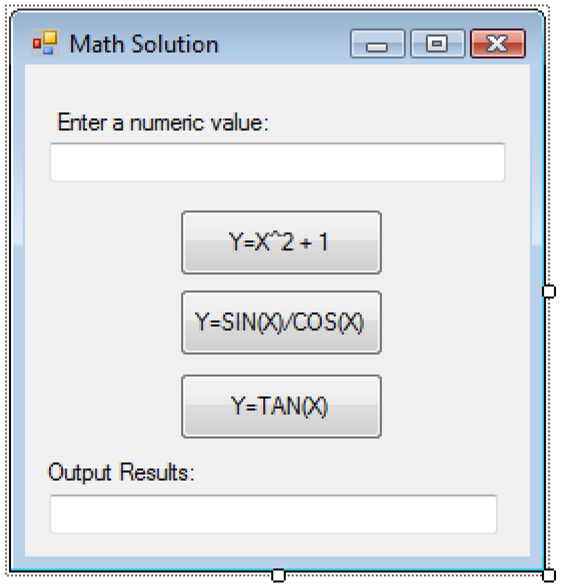
```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

namespace Formula2
{
    public static class clsFormula2
    {
        public static Double Formula2(Double dValue)
        {
            return ( Math.Sin(dValue) / Math.Cos(dValue) );
        }
    }
}
```

## clsFormula3.cs:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

namespace Formula3
{
    public static class clsFormula3
    {
        public static Double Formula3(Double dValue)
        {
            return ( Math.Tan(dValue) );
        }
    }
}
```

# Our Fifth (and Visually Stunning) C# Project

Next, modify the form to look like the form shown below:



Rename the input box at the top to txtInputValue and rename the input box at the bottom to txtOutputValue.

Rename the first button to btnFormula1, the second button to btnFormula2, and btnFormula3. Add the text that you see on the button faces to the Text property of each button.

Next, in the C# code for the form, you must add the three using lines as shown below. Note that Formula1, Formula2 and Formula3 are the *namespace names*!

```
using Formula1;
using Formula2;
using Formula3;
```

# Our Fifth (and Visually Stunning) C# Project

Finally, in the same code module, you must update the Click events for each of the three buttons. Click each button in turn in order for Visual Studio to insert the blank Click events. Then, update each one to look like the code below:

```
private void btnFormula1_Click(object sender, EventArgs e)
{
    txtOutputValue.Text = clsFormula1.Formula1(Convert.ToDouble(txtInputValue.Text)).ToString();
}

private void btnFormula2_Click(object sender, EventArgs e)
{
    txtOutputValue.Text = clsFormula2.Formula2(Convert.ToDouble(txtInputValue.Text)).ToString();
}

private void btnFormula3_Click(object sender, EventArgs e)
{
    txtOutputValue.Text = clsFormula3.Formula3(Convert.ToDouble(txtInputValue.Text)).ToString();
}
```

Notice that we refer to each of the static methods as *class-name.method-name* in the code above. This is the standard way of calling static methods (like calling `Math.Pow`, etc.).

Next, compile the code by right-clicking the solution name in the Solution Explorer and clicking Build Solution on the context menu. This will compile the code. Hit F5 and your GUI should appear.

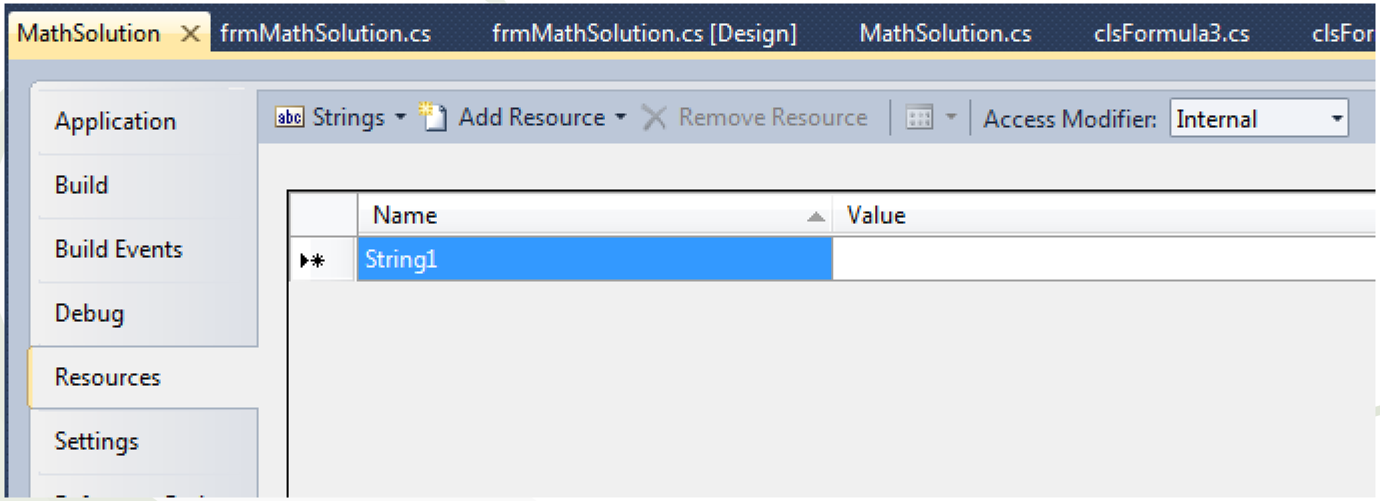
Naturally, we should add error checking, etc. to this code, but the goal was to show you how to create a Windows Forms Application with multiple Class Libraries.

## Using Resources

Recall that if you right-click on a project in Solution Explorer and then click on the Properties menu item, the Properties page appears to the left. This page has a tab marked Resources which allows you to add a variety of resources into your project such as images, text files, audio files, icons, strings, etc. You can also add blank images for your use later on.

This seems like a great idea since the file is embedded within your program and not strewn about on disk where some drunken user might delete it thinking it's an alien plot to take over his computer.

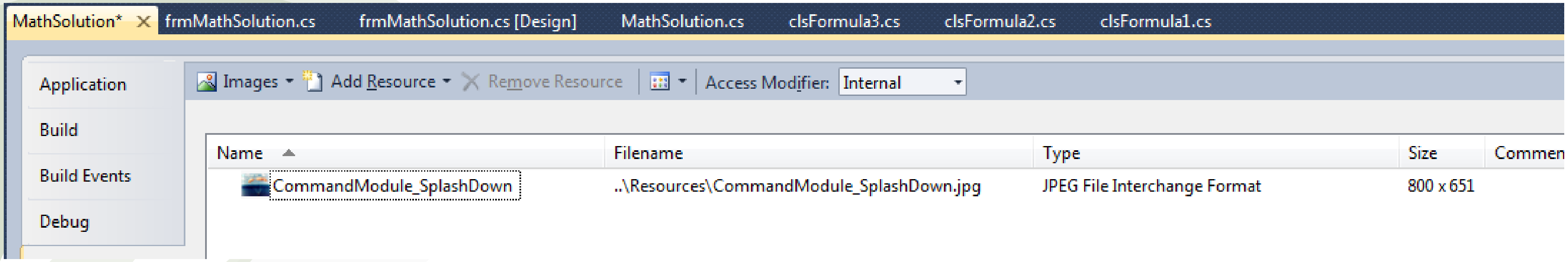
Below is an image of the Resources tab. The drop-down box on the left (showing the word Strings) allows you to select the category of resource to be displayed below. Right now, Strings resources are shown, but you can choose from Strings, Images, Audio, Icons, Files, Other.



## Using Resources

Now, based on your choice in this drop-down box, the drop-down box to the right of it allows you to add a resource based on an existing file, or you can add an new string, new text file, or new image.

Let's add an image to our project as a resource. Click the drop-down box on the left and select Images. Next, using the Add Resource drop-down box, add an existing image on your computer. Hey, hey, keep it clean, guys! I just happened to have a picture of an Apollo Command Module splashing down in the ocean on my computer, so that's what I added. Here is what the entry looks like in the Resources tab:

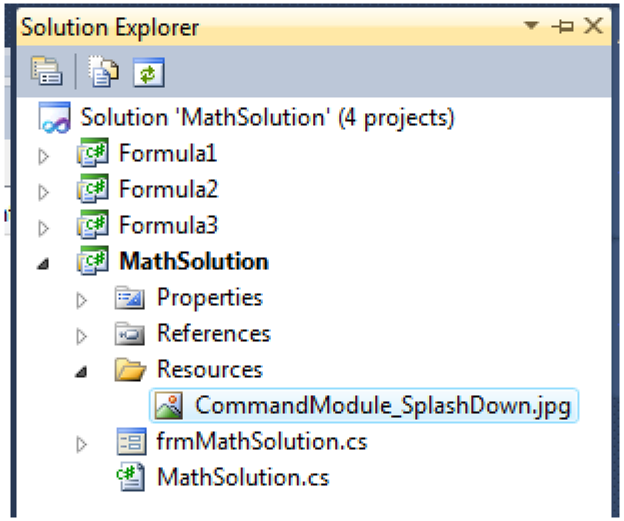


Access Modifier: Internal

| Name | Filename | Type | Size | Commen |
|---|---|---|---|---|
| CommandModule_SplashDown | ..\Resources\CommandModule_SplashDown.jpg | JPEG File Interchange Format | 800 x 651 | |

If you go back to the Solution Explorer pane, you will see an entry for this resource under the project within a folder named Resources. (See next slide.)

## Using Resources



Solution 'MathSolution' (4 projects)
- Formula1
- Formula2
- Formula3
- MathSolution
  - Properties
  - References
  - Resources
    - CommandModule_SplashDown.jpg
  - frmMathSolution.cs
  - MathSolution.cs

Now, add a `PictureBox` control to a form and call it pbImage. Next, depending on the size of the image, you may want to have the `PictureBox` automatically size it to fit in the size of the `PictureBox` you have on your form. To do this, go to the `SizeMode` property of the `PictureBox` and select `StretchImage` from the drop-down box.

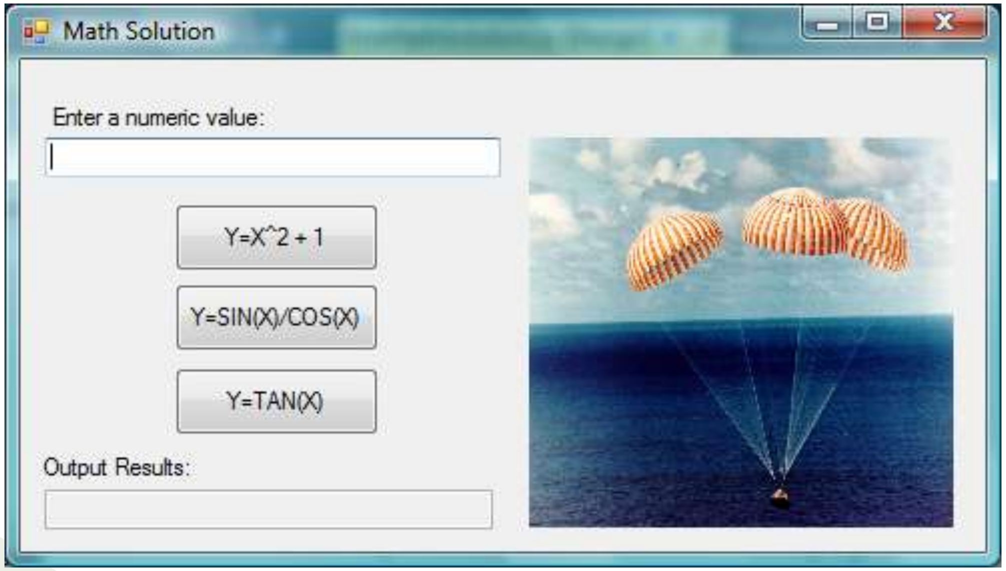
Now, let's populate this image when the form load, so we will be using the Load event of the form. Double-click on the form and you will be taken to the form's Load event. Add the following code to this event:

```
pbImage.Image = Properties.Resources.CommandModule_SplashDown;
```

Now, the name of your image is probably different, but if you use Intellisense, you should be able to find the image with no problem.

Note: Look into `resgen.exe`.

## Using Resources



As you see above, the image is loaded when the form appears and is resized to fit within the `PictureBox`.

## Using Application Settings

Similar to Resources, the Application Settings give you the ability to save user choices from within the program without having to resort to those silly databases, wacky text files on disk, etc.

Recall that if you right-click on a project in Solution Explorer and then click on the Properties menu item, the Properties page appears to the left.  This page has a tab marked Settings which allows you to add a variety of settings to your project.

Note that there are two types of scope for your Settings: Application or User.  An Application scope is applied across the application regardless of the user whereas the User scope saves each setting for an individual user without you having to code for it!

Based on Microsoft's MSDN website:

*The Application Settings feature of Windows Forms makes it easy to create, store, and maintain custom application and user preferences on the client computer. With Windows Forms application settings, you can store not only application data such as database connection strings, but also user-specific data, such as user application preferences. Using Visual Studio or custom managed code, you can create new settings, read them from and write them to disk, bind them to properties on your forms, and validate settings data prior to loading and saving.*

*Application settings enables developers to save state in their application using very little custom code, and is a replacement for dynamic properties in previous versions of the .NET Framework. Application settings contains many improvements over dynamic properties, which are read-only, late-bound, and require more custom programming. The dynamic property classes have been retained in .NET Framework version 2.0, but they are just shell classes that thinly wrap the application settings classes.*

## Using Application Settings

*Your Windows Forms applications will often require data that is critical to running the application, but which you do not want to include directly in the application's code. If your application uses a Web Service or a database server, you may want to store this information in a separate file, so that you can change it in the future without re-compiling. Similarly, your applications may require storing data that is specific to the current user. Most applications, for example, have user preferences that customize the application's appearance and behavior.*

*Application settings addresses both needs by providing an easy way to store both application-scoped and user-scoped settings on the client computer. Using Visual Studio or a code editor, you define a setting for a given property by specifying its name, data type, and scope (application or user). You can even place related settings into named groups for easier use and readability. Once defined, these settings are persisted and read back into memory automatically at run time. A pluggable architecture enables the persistence mechanism to be changed, but by default, the local file system is used.*

*Application settings works by persisting data as XML to different configuration (.config) files, corresponding to whether the setting is application-scoped or user-scoped. In most cases, the application-scoped settings are read-only; because they are program information, you will typically not need to overwrite them. By contrast, user-scoped settings can be read and written safely at run time, even if your application runs under partial trust. For more information about partial trust, see Security in Windows Forms Overview.*

*Settings are stored as XML fragments in configuration files. Application-scoped settings are represented by the <application.Settings> element, and generally are placed in app.exe.config, where app is the name of your main executable file. User-scoped settings are represented by the <userSettings> element and are placed in user.config, where user is the user name of the person currently running the application. You must deploy the app.exe.config file with your application; the settings architecture will create the user.config files on demand the first time the application saves settings for that user. You can also define a <userSettings> block within app.exe.config to provide default values for user-scoped settings.*

## Using Application Settings

### Note the following important information:

*Application settings has no built-in facility for encrypting information automatically. You should never store security-related information, such as database passwords, in clear text. If you want to store such sensitive information, you as the application developer are responsible for making sure it is secure. If you want to store connection strings, we recommend that you use Windows Integrated Security and not resort to hard-coding passwords into the URL. For more information, see Code Access Security and ADO.NET.*

### Continuing on…

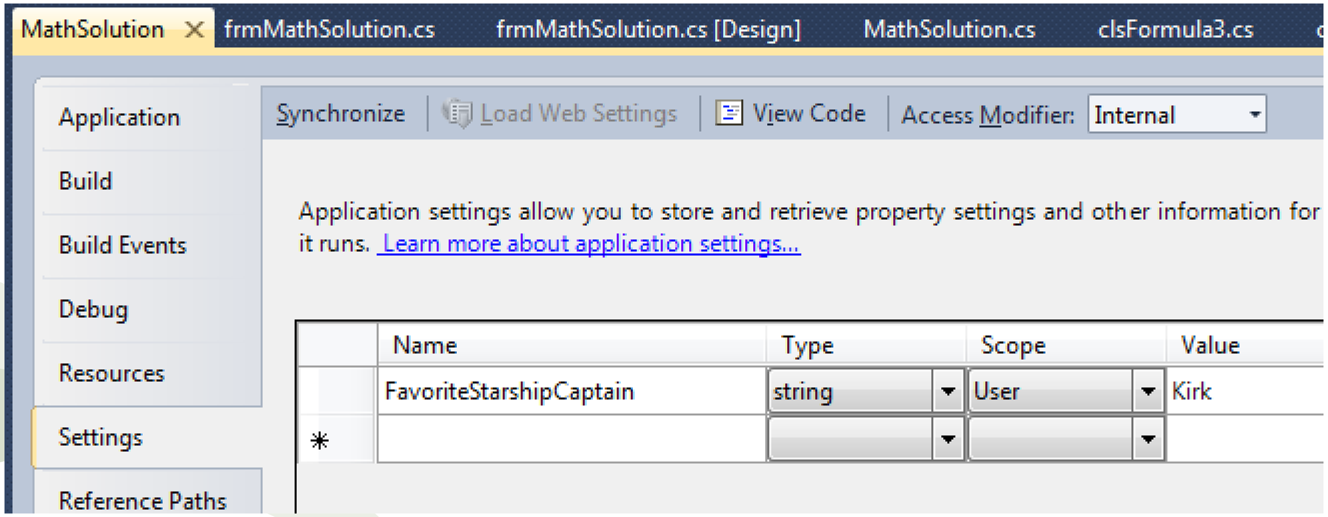
*If you use Visual Studio, you can define settings within the Windows Forms Designer using the (ApplicationSettings) property in the Properties window. When you define settings this way, Visual Studio automatically creates a custom managed wrapper class which associates each setting with a class property. Visual Studio also takes care of binding the setting to a property on a form or control so that the control's settings are restored automatically when its form is displayed, and saved automatically when the form is closed. For details, see How to: Create Application Settings Using the Designer.*

*If you want more detailed control over your settings, you can define your own custom applications settings wrapper class. This is accomplished by deriving a class from ApplicationSettingsBase, adding a property that corresponds to each setting, and applying special attributes to these properties. For details about creating wrapper classes, see Application Settings Architecture.*

*You can also use the Binding class to bind settings programmatically to properties on forms and controls. For more information about creating wrapper classes, see How to: Create Application Settings Using the Designer.*

# Using Application Settings

Now, let's add a User-scoped setting. In the Settings tab, add the name `FavoriteStarshipCaptain` and set its Type to string, its Scope to User, and set its initial value to Kirk. Your Setting screen should look like this:



Application settings allow you to store and retrieve property settings and other information for
it runs. Learn more about application settings...

| | Name | Type | Scope | Value |
|---|---|---|---|---|
| | FavoriteStarshipCaptain | string | User | Kirk |
| * | | | | |

If you look in the Solution Explorer pane, you'll notice that the file app.config has been added. This contains the following XML code:

```
<?xml version="1.0" encoding="utf-8" ?>
<configuration>
    <configSections>
        <sectionGroup name="userSettings" type="System.Configuration.UserSettingsGroup, System, Version=4.0.0.0, Culture=neutral, 
PublicKeyToken=b77a5c561934e089" >
            <section name="MathSolution.Properties.Settings" type="System.Configuration.ClientSettingsSection, System, Version=4.0.0.0, 
Culture=neutral, PublicKeyToken=b77a5c561934e089" allowExeDefinition="MachineToLocalUser" requirePermission="false" />
        </sectionGroup>
    </configSections>
    <userSettings>
        <MathSolution.Properties.Settings>
            <setting name="FavoriteStarshipCaptain" serializeAs="String">
                <value>Kirk</value>
            </setting>
        </MathSolution.Properties.Settings>
    </userSettings>
</configuration>
```

# Using Application Settings

Next, add a ComboBox to the form and in the ComboBox's Properties, make the following changes:

1. Change the DropDownStyle to DropDownList. This will prevent a user from entering their own choice.
2. In the Items property, add the following captains making sure that each one appears on a separate line: Archer, Bateson, Decker, Janeway, Kirk, Picard, Wesley

Next, in the Load event of the form, add the following code (some of which was added before):

```
//Update the picture box to include the image stored as a resource.
pbImage.Image = Properties.Resources.CommandModule_SplashDown;

//Based on the Application setting for the user, select the appropriate captain.
String sSelectedCaptain = Properties.Settings.Default.FavoriteStarshipCaptain;
Int32 iSelectedCaptain = cbCaptains.FindString(sSelectedCaptain);
cbCaptains.SelectedIndex = iSelectedCaptain;
```

The code in red above pulls the FavoriteStarshipCaptain into the variable sSelectedCaptain. The code then finds the index of that captain and then sets the ComboBox so that captain is chosen.

Next, double-click on the ComboBox and you'll be taken to the skeleton code for the SelectedIndexChanged event. Change the code to look like this:

## Using Application Settings

```
private void cbCaptains_SelectedIndexChanged(object sender, EventArgs e)
{
    cbCaptains.Enabled = false;
    Properties.Settings.Default.FavoriteStarshipCaptain = cbCaptains.Text;
    Properties.Settings.Default.Save();
    cbCaptains.Enabled = true;
}
```

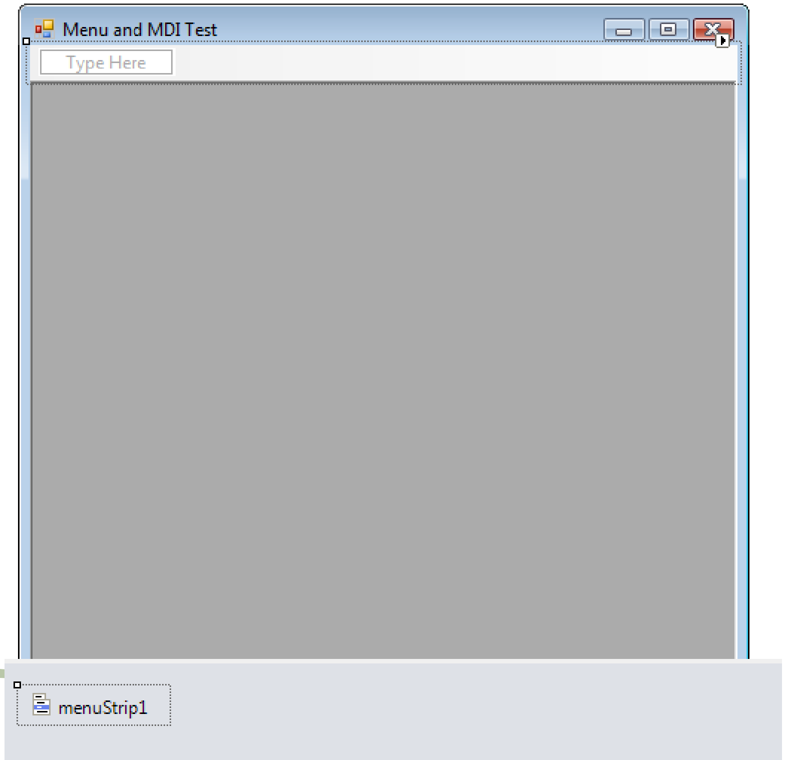
You note that we disable the `ComboBox` at the beginning and the re-enable it at the end. In the middle, we set the `FavoriteStarshipCaptain` property to the text that appears within the `ComboBox` at the time, and then we issue a Save() to persist the change.

# Creating a Menu

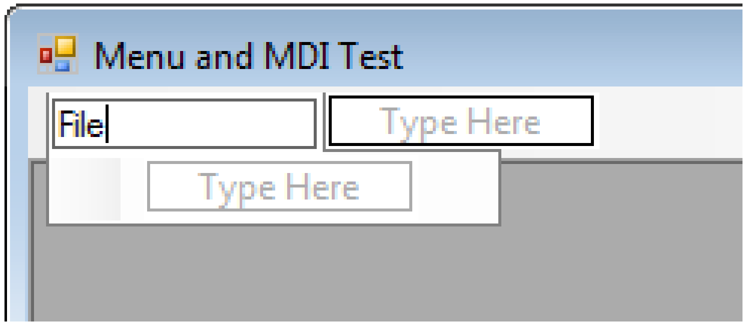
You can very easily create a menu on one of your forms by dragging the menu control from the Toolbox to the form itself. Don't worry, the menu won't be stuck in a strange place (like the middle right side of the form where you might have dropped it)…no, the menu is located where menus always are: at the top of the form, silly! ☺ Although, you can change the Dock property so that the menu can appear on the left, right, bottom, etc. of the form…but that would look silly and confuse your users!

Now, create a new Visual Studio project (Windows Forms Application, of course) and on the form itself, drag the `MenuStrip` from the Toolbox to the form. When you first add the menu to the form, you will see something like this:

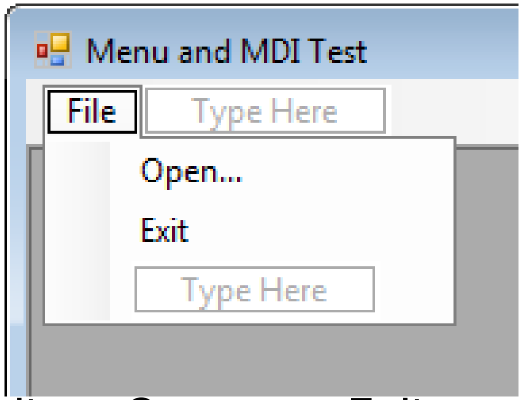
As you see to the right, you are given an indication that a menu exists by the large gray bar at the bottom…this will not appear when you run the program. Now, you start to enter your menus by typing in the gray rectangular box at the top (where the words Type Here is located). When you type in the word File, the MenuStrip will automatically show you a placeholder for the next menu, but you will also see a place for a single menu item below your first menu. See next slide for an example of this.



## Creating a Menu



Now, to create a menu item below File, type in the rectangle just below File. For example, put in the word Open… and then put in the word Exit. You should have something like this:



If you double-click the menu item Open… or Exit, you will be taken to the code window for the form and a blank event for that menu item will be created for you:

```
private void openToolStripMenuItem_Click(object sender, EventArgs e)
{

}

private void exitToolStripMenuItem_Click(object sender, EventArgs e)
{

}
```

## Creating a Menu

Next, let's fill in the two click event methods.

First, for the `exitToolStripMenuItem_Click` event, put in the familiar Close method:

```
private void exitToolStripMenuItem_Click(object sender, EventArgs e)
{
    this.Close();
}
```

Next, for the Open… menu item, we want to display the Open File Dialog Box. In the Toolbox, drag over the `OpenFileDialog` control to the form and rename it to `ofdOpen`. Next, in the `openToolStripMenuItem_Click` event, place this code in it:

```
private void openToolStripMenuItem_Click(object sender, EventArgs e)
{
     if (ofdOpen.ShowDialog() == DialogResult.OK) {
        sFileName = ofdOpen.FileName;
    }
}
```

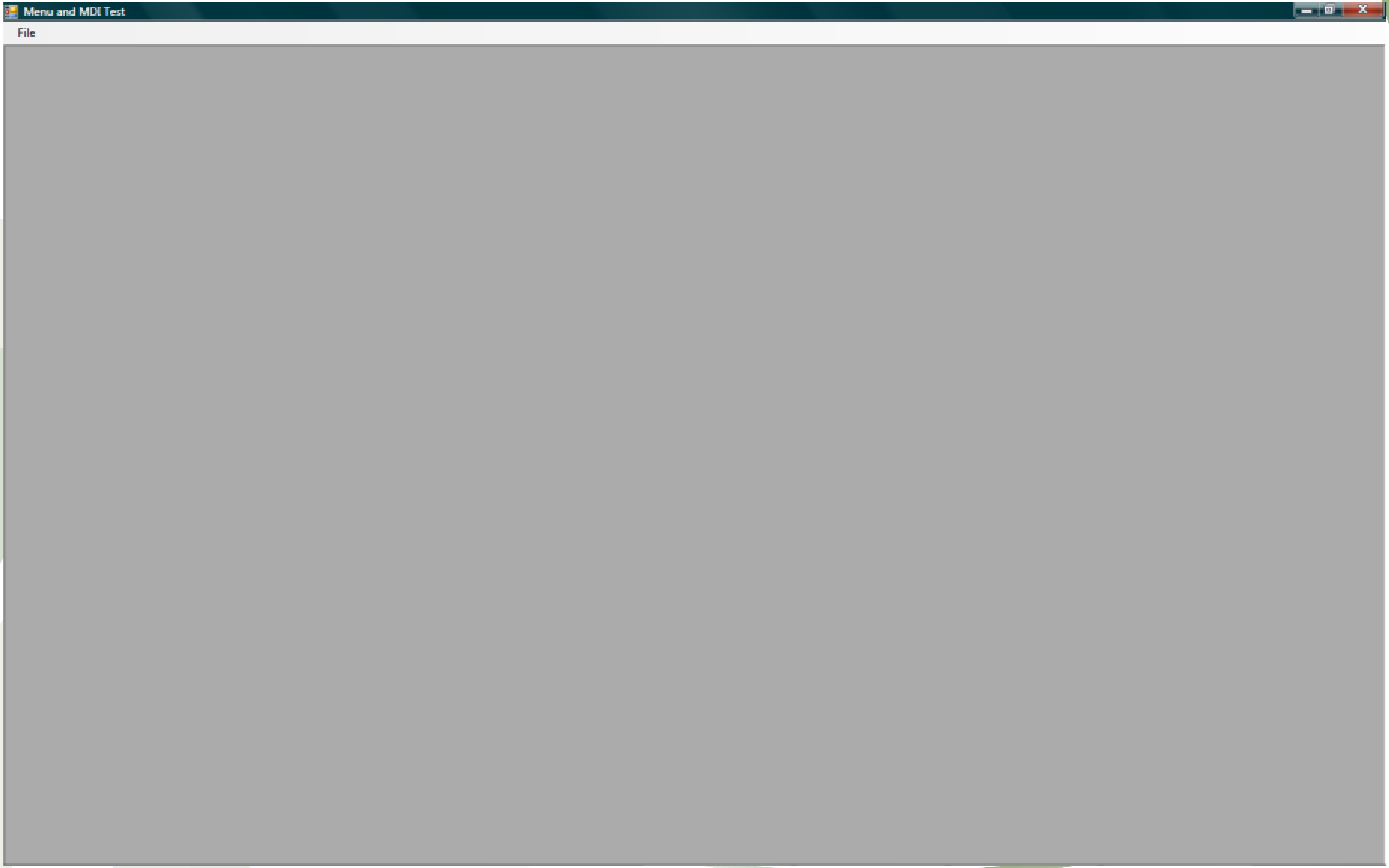
Next, add the following code to the same class:

```
public String sFileName;
```

Also, add the following code to the Filter property for the OpenFileDialog component: `Image Files(*.BMP;*.JPG;*.GIF)|*.BMP;*.JPG;*.GIF`. We'll use this later on in the presentation.

## Creating a Menu

If you compile and execute the application, you should see something like this:



## Creating a Menu

Now, click on the Open… menu item. You should see something like this:

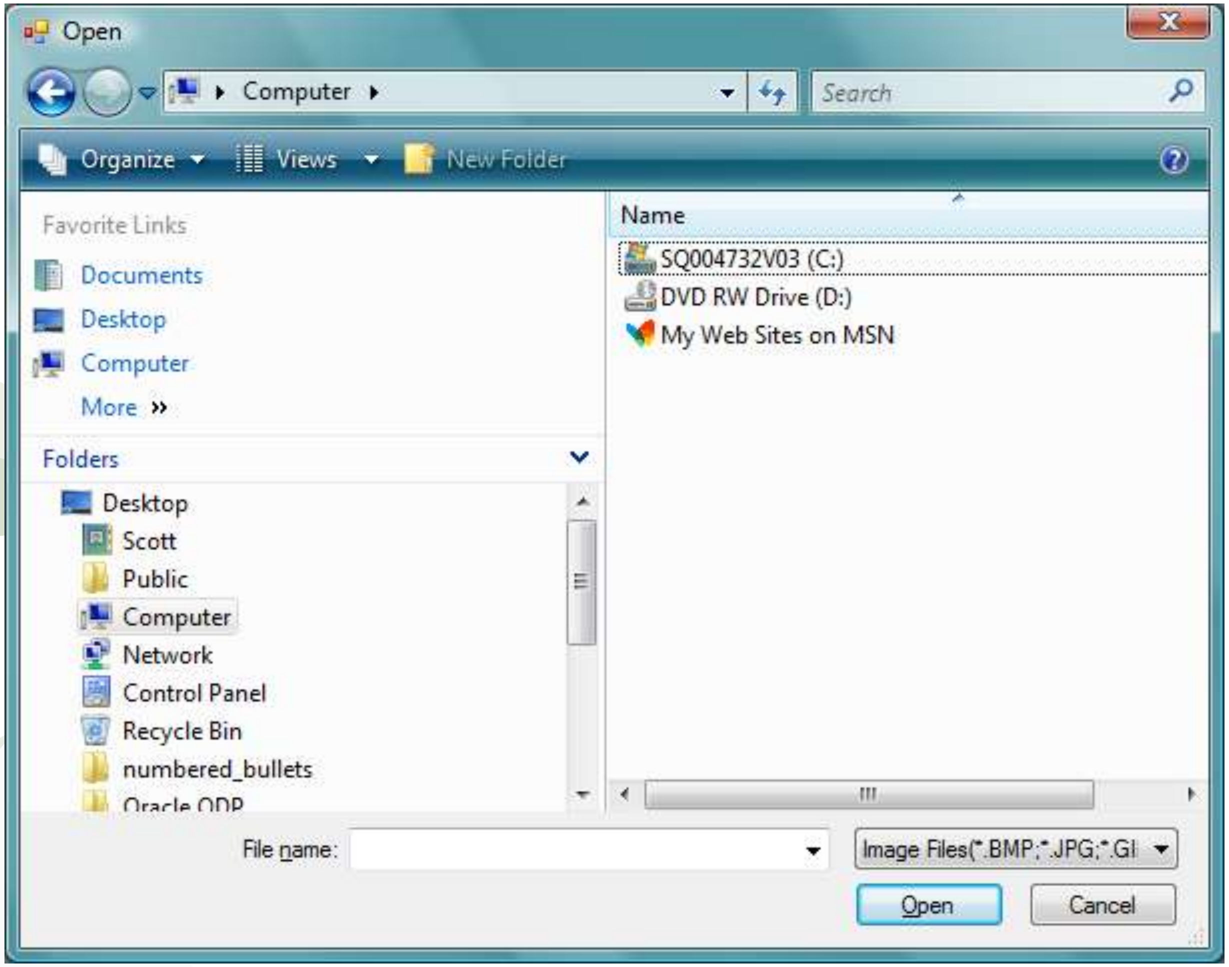
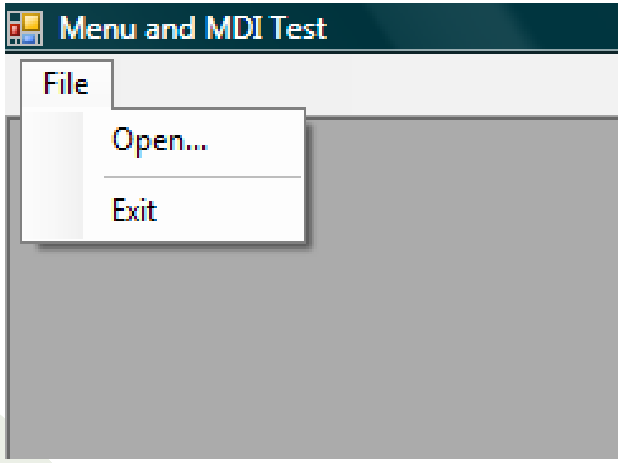


Image Files(*.BMP;*.JPG;*.GI

Take note that the Filter we specified shows up in the drop-down box on the bottom right of the Open File Dialog Box.

## Creating a Menu

Finally, you can add a horizontal bar to break up the menu items on a menu into logical groups by placing a single dash (-) where the text Type Here is located. You can then use your mouse pointer to move the menu items around until you are pleased with the results.



## Creating a Multiple Document Interface (MDI)

In this section, we will learn how to create a Multiple Document Interface (MDI) so that your "main" form can contain several child forms.

Now, create a new Visual C# Windows Forms Application project. If you followed the last section on creating a menu, you can use that project.

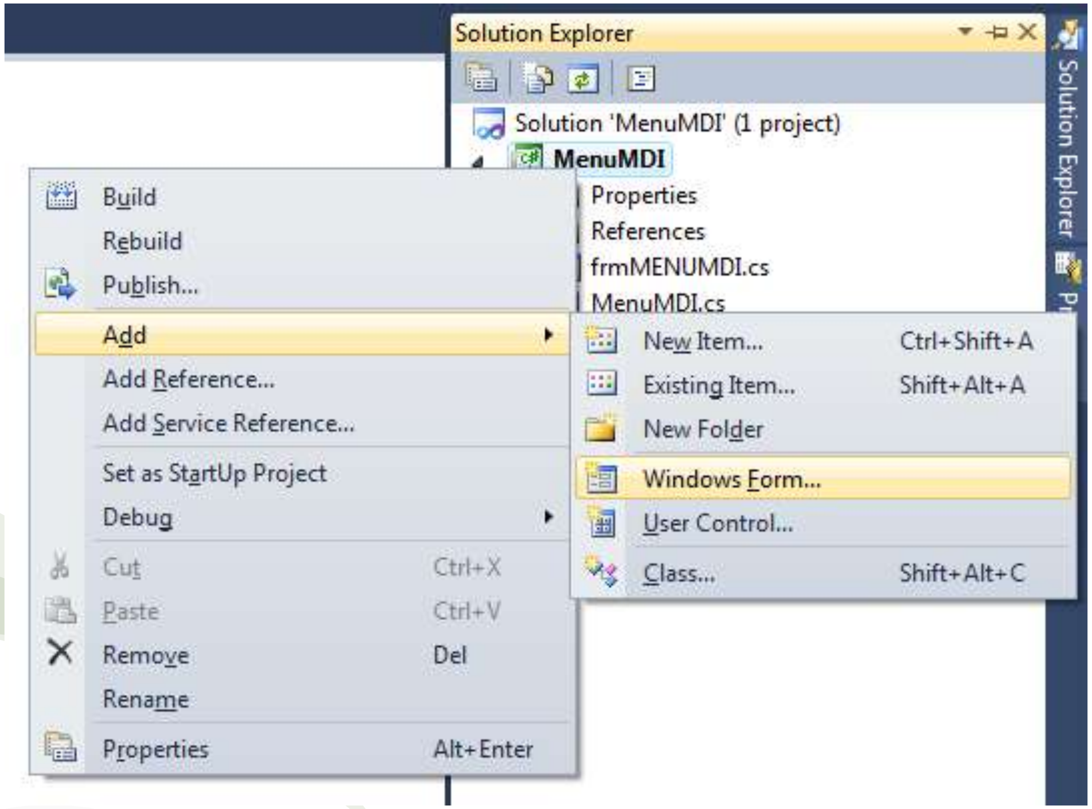
In the Properties pane for the form, change the `IsMdiContainer` from false to true. This will enable this form to contain one or more child forms.

Another property I like to change is the `WindowState` property. Change this to `Maximized` so that when the application starts, it takes up the entire window.

Next, you have to be able to add child forms to the parent form (the MDI container). What I want my application to do is to open up an image (.bmp, .jpg or .gif) and show it as a child form within the containing MDI parent. In order to do this, we need to add another form to the project. To do this, you right-click on the project name in the Solution Explorer pane and click on Add → Windows Form… This will bring up the Add New Item dialog box. Ensure that Windows Form is highlighted, change the name from `Form1.cs` to `frmImage.cs` and then click the Add button.

Next, add a `PictureBox` to the new form (call it `pbImage`). Click on the control arrow at the top of the `PictureBox` control itself, and ensure that `Size Mode` is set to `StretchImage` (so that our image will be resized to fit). Also, click on Dock in Parent Container so that the entire `PictureBox` takes up the entire form.

## Creating a Multiple Document Interface (MDI)



Next, we have to modify the Open… menu item's click event so that, if the user has clicked on a file, the image is displayed in the `PictureBox` and that the filename is displayed in the title bar of the child form. The code to do all of this is on the next slide.

## Creating a Multiple Document Interface (MDI)

```
private void openToolStripMenuItem_Click(object sender, EventArgs e)
{
    if (ofdOpen.ShowDialog() == DialogResult.OK) {

        //Update the public string sFileName.
        sFileName = ofdOpen.FileName;

        //Create an instance of frmImage and ensure that it is told
        //to behave like a child form within the parent container.
        frmImage frmChild1 = new frmImage();
        frmChild1.MdiParent = this;

        //Update the PictureBox for this instance with the selected image.
        frmChild1.pbImage.ImageLocation = sFileName;

        //Update this instance's title bar with the filename.
        frmChild1.Text = sFileName;

        //Show the child form.
        frmChild1.Show();

    }
}
```

Note that based on the form `frmImage`, which is the name of the form containing the `PictureBox`, we create a new instance of it and set its `MdiParent` property to this, which indicates that the parent form is the container for the this child form.  Also, note that we set the `ImageLocation` property of the `PictureBox` to the file requested via the Open File Dialog Box.  We set the title of the child form, which is the `Text` property (there is no Title property).  Finally, we show the form.

Now, at this point, the code won't compile because the `PictureBox`, `pbImage`, is inaccessible to the parent form.  So, go to the `frmImage.Designer.cs` C# code and change the following line…

# Creating a Multiple Document Interface (MDI)

```
private System.Windows.Forms.PictureBox pbImage;
```

to this…

```
public System.Windows.Forms.PictureBox pbImage;
```

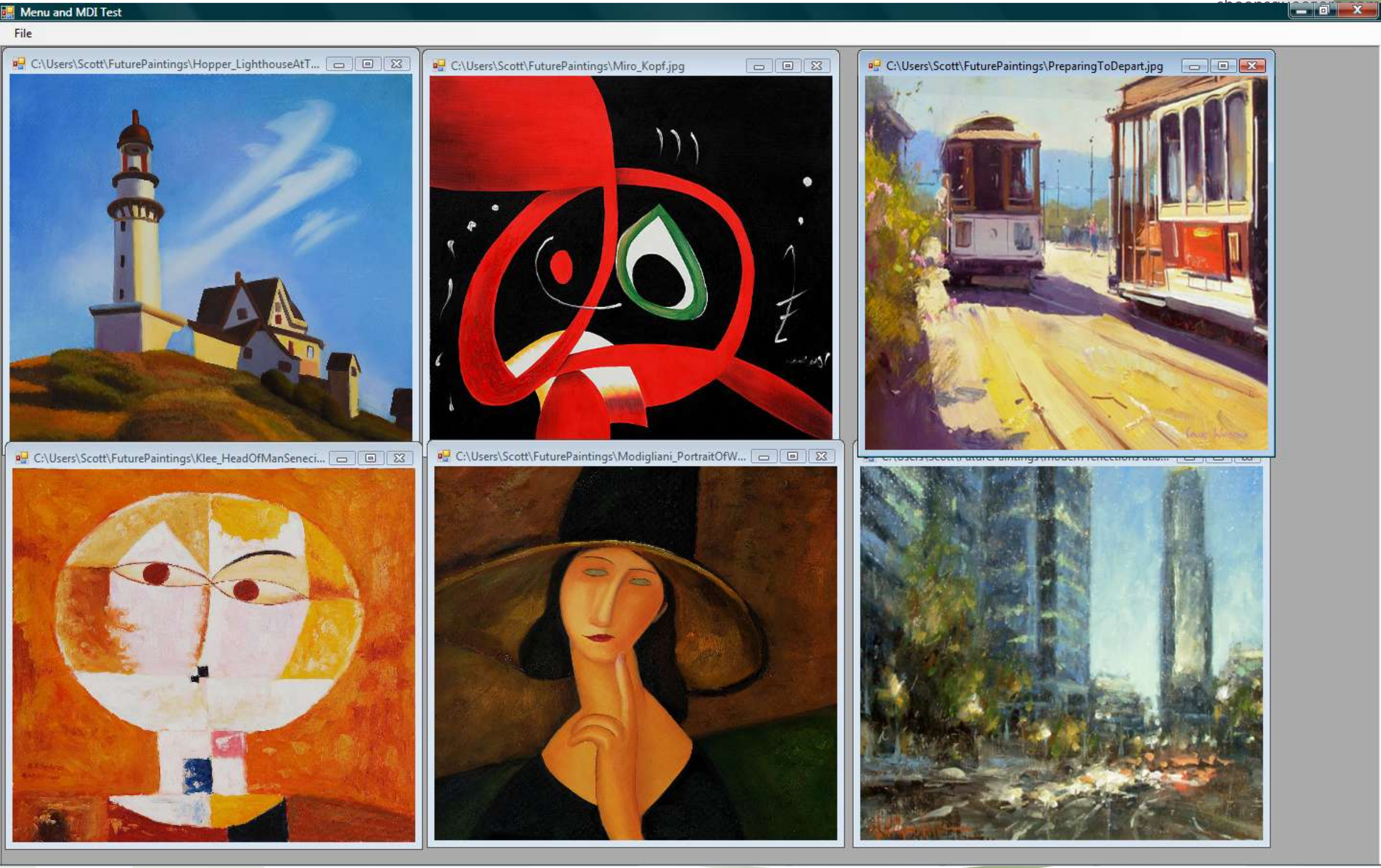
Now, compile and run the application. Using the Open… menu item, find an image to open, repeat as often as you like. On the next page, is my example.

Clearly, there is a lot more to this than what I've shown, but I hope it was a good start.

Note: The modification to the code as shown at the top of this slide is probably NOT the best way to go. You may want to create a public property that allows you to set the `ImageLocation` of the `PictureBox` rather than making `pbImage` `public`. Just my two-cents! ☺

You can find out more on creating MDI Forms at http://msdn.microsoft.com/en-us/library/aa984329(v=vs.71).aspx.

# Creating a Multiple Document Interface (MDI)


Menu and MDI Test
File
C:\Users\Scott\FuturePaintings\Hopper_LighthouseAtT...
C:\Users\Scott\FuturePaintings\Miro_Kopf.jpg
C:\Users\Scott\FuturePaintings\PreparingToDepart.jpg
C:\Users\Scott\FuturePaintings\Klee_HeadOfManSeneci...
C:\Users\Scott\FuturePaintings\Modigliani_PortraitOfW...

## References

*Click the book titles below to read more about these books on Amazon.com's website.*

- Introducing Microsoft LINQ, Paolo Pialorsi and Marco Russo, Microsoft Press, ISBN:9780735623910
- LINQ Pocket Reference, Joseph Albahari and Ben Albahari, O'Reilly Press, ISBN:9780596519247
- Inside C#, Tom Archer and Andrew Whitechapel, Microsoft Press, ISBN:0735616485
- C# 4.0 In a Nutshell, O'Reilly Press, Joseph Albahari and Ben Albahari, ISBN:9780596800956
- The Object Primer, Scott W. Ambler, Cambridge Press, ISBN:0521540186
- CLR via C#, Jeffrey Richter, Microsoft Press, ISBN:9780735621633

## Support sheepsqueezers.com

If you found this information helpful, please consider supporting sheepsqueezers.com. There are several ways to support our site:

- ☐ Buy me a cup of coffee by clicking on the following link and donate to my PayPal account: Buy Me A Cup Of Coffee?.
- ☐ Visit my Amazon.com Wish list at the following link and purchase an item: http://amzn.com/w/3OBK1K4EIWIR6

Please let me know if this document was useful by e-mailing me at `comments@sheepsqueezers.com`.