8 C# Development Tools

Introduction

This chapter will introduce the reader to several development tools that support the development of large scale C# systems. We will also consider the importance of documentation and show tools can be used to generate documentation for systems you create (almost automatically).

Objectives

By the end of this chapter you will be able to….

- Find details of several professional and free interactive development environments
- Understand the importance of the software documentation tools and the value of embedding XML comments within your code.
- Write XML comments and generate automatic documentation for your programs.

This chapter consists of eight sections :-

1) Tools for Writing C# Programs….
2) Microsoft Visual Studio
3) SharpDevelop
4) Automatic Documentation
5) Sandcastle Help File Builder
6) GhostDoc
7) Adding Namespace Comments
8) Summary

8.1 Tools for Writing C# Programs

Whatever mode of execution is employed (see section 1.7), programmers can work with a variety of tools to create source code. It is possible to write C# programs using simple discrete tools such as a plain text editor (e.g. Notepad) and a separate compiler invoked manually as required. However virtually all programmers would use a powerful Integrated Development Environment (IDE) which use compilers and other standard tools behind a seamless interface.

Even more sophisticated tools Computer Aided Software Engineering (CASE) tools exist which integrate the implementation process with other phases of the software development lifecycle. CASE tools could take UML class diagrams, generated as part of the software analysis and design phase, and generate classes and method stubs automatically saving some of the effort required to write the C# code (ie. the implementation phase.

CASE tools could also help by automating the software testing phase.

Overtime, as more powerful tools are being created, IDEs have begun to incorporate some of the features previously found only in CASE tools i.e. they do much more than allow programs to be written, edited and compiled. One such tool is Microsoft Visual Studio.

8.2 Microsoft Visual Studio

Moving to an 'industrial strength' IDE is an important stepping stone in your progress as a software developer, like riding a bicycle without stabilisers for the first time. With some practice you will soon find it offers lots of helpful and time-saving facilities that you will not want to work without again.

Microsoft Visual Studio is a powerful IDE platform that supports the development of .NET programs written in a range of languages not just C#.

Details of this can be found via http://www.microsoft.com/visualstudio/.

Visual Studio comes in many varieties but perhaps the two most common varieties are Visual Studio Professional edition and Visual Studio Express edition. The professional was used to create all of the code in this book and for the case study application developed in Chapter 11. However the express edition is free and still contains all of the features required to create and run the applications in the book.

For our purposes there is only one thing missing from the Express edition is the unit testing tools explained in section 10.5 and used in section 11.12. In all other respects it is a very powerful IDE and fully supports a student wanting to learn and develop C# programs.

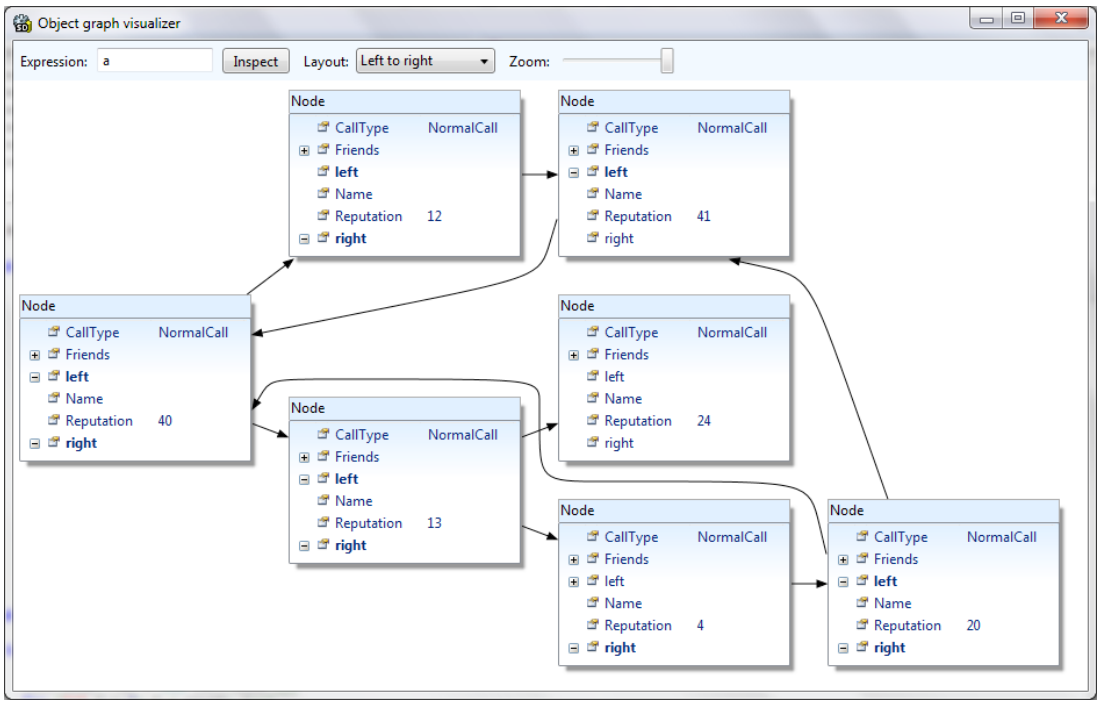
Visual Studio Express edition can be downloaded from the following website http://www.microsoft.com/express/Downloads/

8.3 SharpDevelop

Another very powerful and free tool that can be used to develop C# programs is SharpDevelop (or #develop). It is open-source so you can download both the sourcecode and executable versions.

SharpDevelop includes support for several .NET languages including C# 4.0 and VB.NET 10. It includes unit testing facilities (via optional plug ins), code completion facilities and powerful debugging facilities.

To help debugging, SharpDevelop includes an Object graph visualiser (see below). This visualiser displays a visual representation of your data structures that is dynamically updated when stepping through your code.



For more information on SharpDevelop or to download it go to http://community.sharpdevelop.net/forums/t/12513.aspx

It should be noted that other free tools exist including MonoDevelop (monodevelop.com/) an IDE which offers multiplatform support including Linux, Windows and Mac OSX).

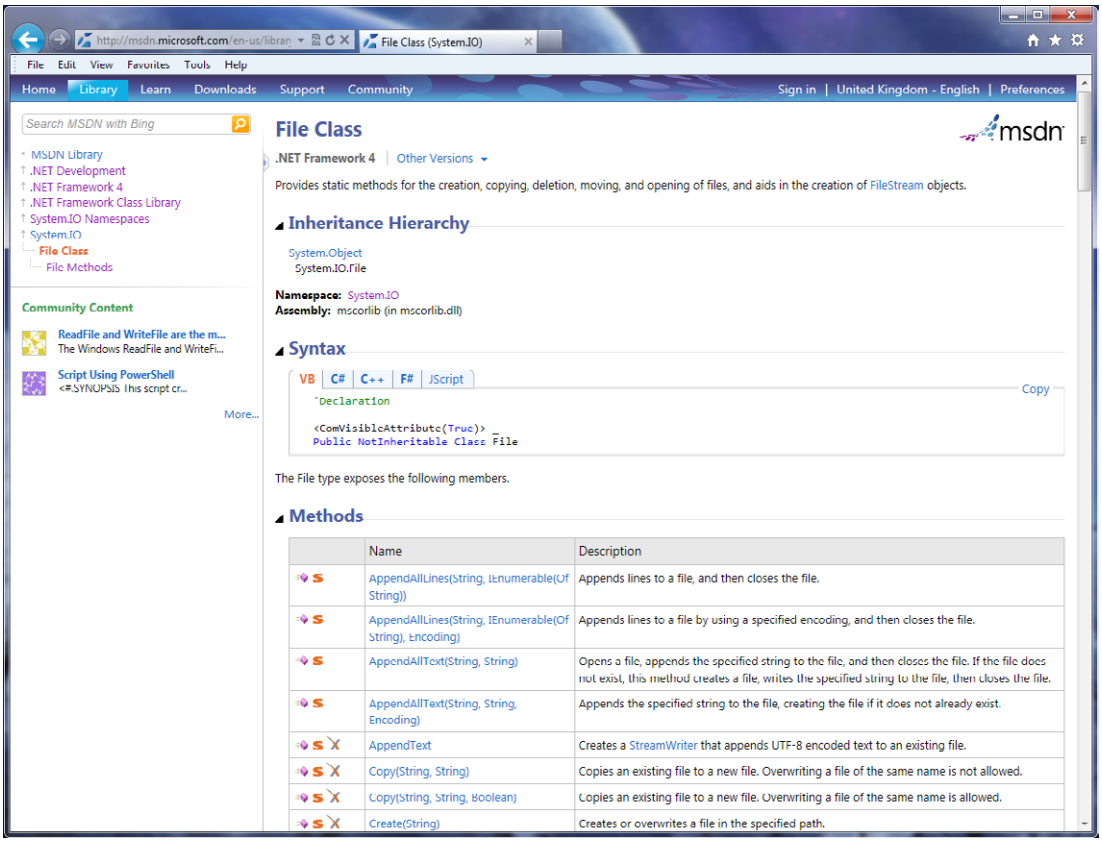
8.4 Automatic Documentation

One particularly useful feature found within some IDE's is the ability to generate automatic documentation for the programs we create.

Programmers for many years have been burdened with the tedious and time consuming task of writing documentation. Some of this documentation is intended for users and explain what a program does and how to use it. Other documentation is intended for future programmers who will need to amend and adapt the program so that its functionality will change as the needs of an organisation change. These programmers need to know what the program does and how it is structured. They need to know :-

- what packages it contains
- what classes exist in each of these packages and what these classes do,
- what methods exist in each class and for each of these methods
    - what does the method do
    - what parameters does it require
    - what, if any, value is returned.

Tools can't produce a user guide but they can provide a technical description of the program. Tools can analyse C# source files for the programs we write and produce documentation, either as a set of web pages (HTML files) or in some other format. Technical documentation should contain similar information as you would find when looking MSDN to find details of the .NET libraries.



This website provides, as a set of indexed web pages, essential details of the .NET libraries including all packages, classes, methods and method parameters and return values. This extremely useful information was not generated by hand but generated using the automatic tools.

Tools can produce similar detailed documentation for any C# program you create and this would help future programmers amend and update your programs. However this relies on properly formatted (and informative!) XML-style comments in source files, including tags such as @author, @param etc.

Because this documentation is generated automatically, at the push of a button, it saves programmers from a tedious, time consuming and error prone task. Furthermore the documentation can be updated whenever a program is changed.

XML (Extensible Markup Language) is a way of specifying and sharing structured information.

By adding meaningful XML comments to the code we write tools can produce automatic documentation for us. Poor attention to commenting of source code will result in poor documentation. On the other hand, if the commenting is done properly then the reference documentation is produced for virtually no effort at all.

XML comments should therefore be added at the start of every class using the standard tags <summary> and <remarks> to give an overview of that class in the following format..

/// <summary>
/// Provide a short description of the lass and its role here ….
/// </summary>
/// <remarks>Author Simon Kendal
/// Version 1.0 (5th May 2011)</remarks>

Similar comments should be provided for every method using the /// <param name=”name of paramter”>, and /// <returns> tags to describe each parameter and to describe the value returned. The details of each parameter, starting with the name of the parameter, should be provided on separate lines as shown below.

/// <summary>
/// A description of the method
/// </summary>
/// <param name=”name of first parameter”>A description of that parameter </param>
/// <param name=”name of 2nd parameter”>A description of that parameter </param>
/// <returns> A description of the value returned by a method. /// </returns>

Activity 1

The method below takes two integer numbers and adds them together. This value is returned by the method. Write an XML comment, using appropriate tags, to describe this method.

```
public int add(int number1, int number2)
{
return (number1 + number2);
}
```

Feedback 1

```
/// <summary>
/// This method adds two integer numbers.
/// </summary>
/// <param name=”number1”>The first number.</param>
/// <param name=”number2”>The second number.</param>
 /// <returns>The sum of the two numbers. </returns>
```

Automatic tools cannot analyse comments to determine if these provide an accurate description of the methods, however tools can do far more than cut and paste a programmers comments onto a web document. One thing tools do is analyse method signatures and compare this with the tags in the comments to check for logical errors. If a method requires three

parameters but the comment only describes two then this error will be detected and reported to the programmer.

```
"Parameter 'name' has no matching param tag in the XML comment for 'Name of class here' (but
other parameters do)".
```

By analysing the code the tools can also provide additional information about the class, the methods that have been inherited and the methods that have been overridden.

By using automatic tools reference documentation can be produced for programmers using the class(es) concerned, it does not include comments within methods intended for programmers editing the class source code in maintenance work.

Automatic documentation tools do not exist as standard within either Microsoft Visual Studio or SharpDevelop. However both of these IDE's can be integrated with Sandcastle Help File Builder… this then enables automatic documentation to be generated. Better still another tool exists called GhostDoc which further helps by generating the necessary XML comments.

8.5 Sandcastle Help File Builder

To be precise Sandcastle Help File Builder (SHFB) does not produce automatic documentation for your code… but it adds a graphical front end to 'Sandcastle'. 'Sandcastle' is a difficult to use tool that lacks a GUI, and has a complex installation routine but will generate automatic documentation.

Sandcastle Help File builder is a GUI that will drive Sandcastle and will also automate the installation of Sandcastle. Sandcastle Help File Builder can be downloaded from http://shfb.codeplex.com/.

Running this software will install

a) Sandcastle,
b) updates patches and additional files
c) compilers for different help file formats – only Help 1 HTML compiler required.
d) Sandcastle Help File Software itself.

When running SHFB you need to specify an XML file with comments generated from your code and the C# code itself i.e. a .sln file

To generate the XML file from within Visual Studio select the project Properties / Build / XML comments. Note Visual Studio will then complain if comments are missing from your code.

8.6 GhostDoc

SHFB is very useful as it generates automatic documentation for your programs assuming of course appropriate XML comments exist in the body of the code.

GhostDoc is one piece of software that will automatically add XML comments to your C# code. Though the comments should be edited to ensure they are meaningful.

GhostDoc can be downloaded from http://submain.com/products/ghostdoc.aspx

8.6 Adding Namespace Comments

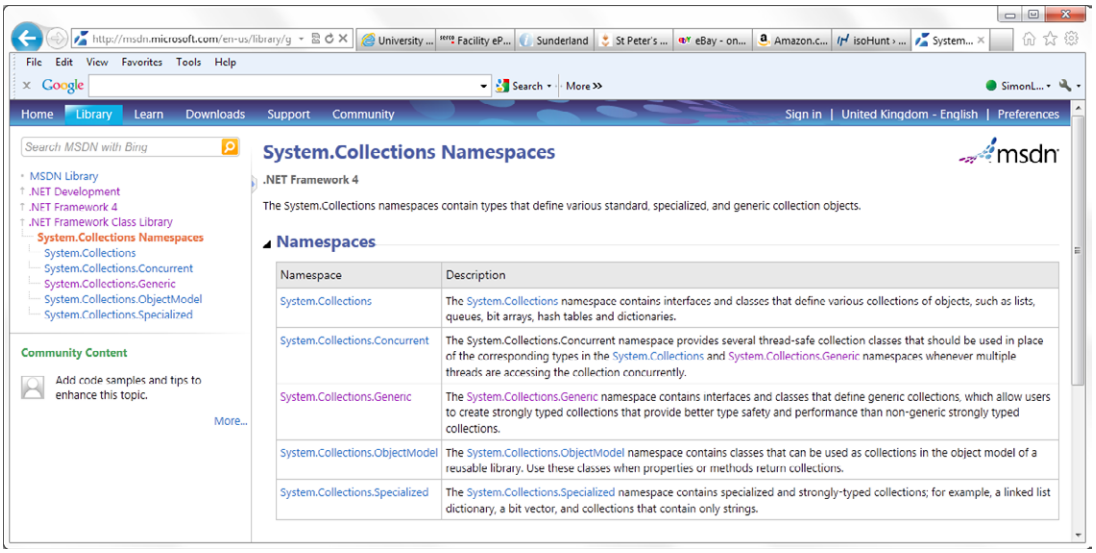
One final complication exist regarding namespace comments.

A room exists and it is possible to go outside of that room and put a sign on the door. A building exists and we can do the same i.e. go outside and put a sign on the building but while the Universe exists we cannot go outside and put a sign on it … because outside does not exist!

A similar problem exists with namespace comments.

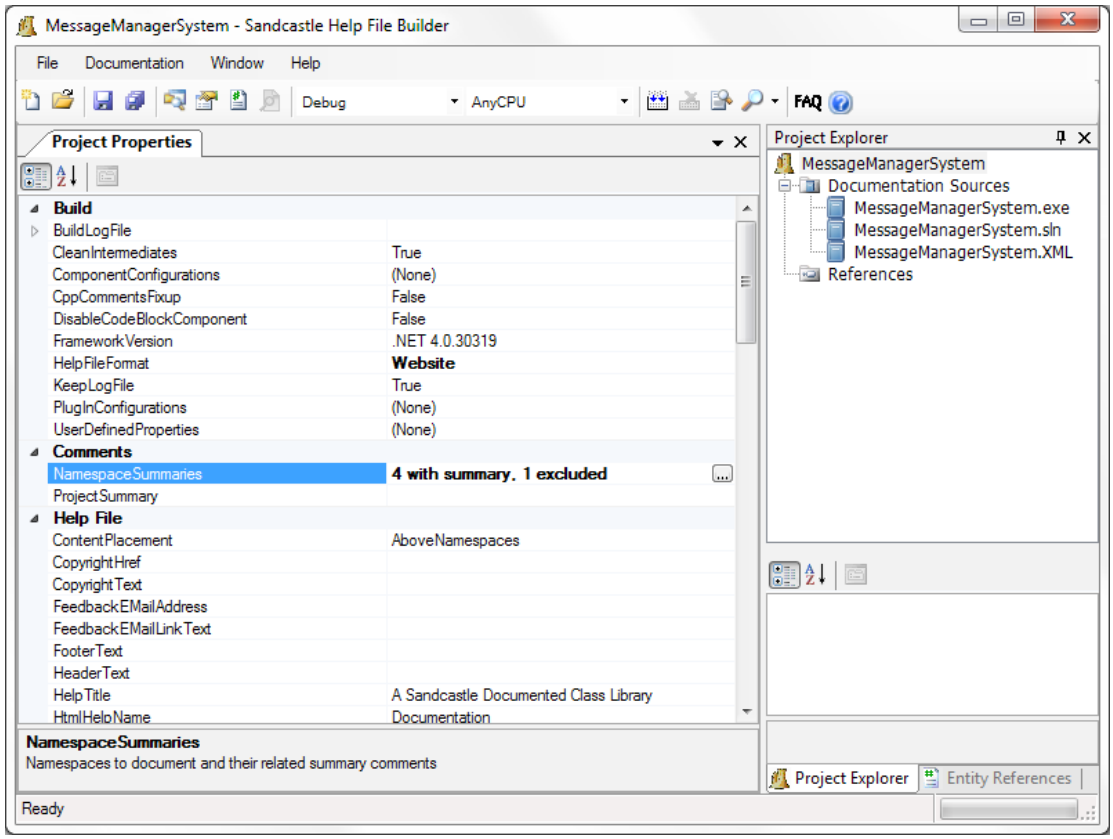
We know where a class starts in our code so we can put an XML comment at the start to describe that class. Similarly we can put comments at the start of methods to describe those methods… but while namespaces exist we cannot define the start of these. So if we segment our larger systems into namespaces, as we should, we cannot put comments at the start of a namespace to describe that namespace.

If we look at part of the MSDN documentation (see figure below) we see that it describes name spaces and we should do the same for our programs.



But if we cannot define the start of a namespace we must put those comments elsewhere.

SHFB provides a solution to this problem by allowing us to enter the namespace comments for our system directly into SHFB.



While it may involve several pieces of software and take a while to set up the ability to generate automatic documentation can save hours and hours of effort. What is more each time we edit and amend our programs we can then generate automatic documentation within a few button pushes.

Other programmers may need to amend and update our the programs we create. As professional programmers we have a duty to the to provide documentation to make this task as easy as possible. Automatic tools such as SHFB and GhostDoc can help us do this.

8.8 Summary

We can go 'back to basics' creating C# programs using a text editor and standalone compilers (freely available) but the use of a professional IDE can offer additional facilities to support programmers.

Specialist tools are available for aspects of development such as GUI design, diagramming and documentation but some IDEs go beyond the basics and provide some of these facilities in built into the IDE.

Visual Studio Express and SharpDevelop are two free powerful IDEs which support the development of C# and other .NET programs.

These tools can initially seem daunting as they provides extensive support for professional development including code formatting and refactoring though online help does exist and both tools provide some level of automatic code generate, e.g. the main method, to make the job of the programmer easier.

Professional programmers have a duty to create up to date documentation. The SHFB software is an extremely useful and timesaving by helping to create this documentation but it does require the programmer to inserting meaningful XML comments into their code (for all classes and all public methods).

GhostDoc be help the programmer by adding XML comments though they will need some manual editing.