Q u i c k   a n s w e r s   t o   c o m m o n   p r o b l e m s

Visual Studio 2012 and .NET 4.5 Expert Development Cookbook

Over 40 recipes for successfully mixing the powerful capabilities of .NET 4.5 and Visual Studio 2012

Abhishek Sur

[PACKT] enterprise
PUBLISHING professional expertise distilled

Visual Studio 2012 and .NET 4.5 Expert Development Cookbook

Over 40 recipes for successfully mixing the powerful capabilities of .NET 4.5 and Visual Studio 2012

Abhishek Sur

[PACKT] PUBLISHING enterprise
professional expertise distilled

BIRMINGHAM - MUMBAI

Visual Studio 2012 and .NET 4.5 Expert Development Cookbook

First published: April 2013

Production Reference: 1050413

Published by Packt Publishing Ltd.
Livery Place
35 Livery Street
Birmingham B3 2PB, UK.
ISBN 978-1-84968-670-9

`www.packtpub.com`

Cover Image by Siddhart Ravishankar (`sidd.ravishankar@gmail.com`)

Credits

Author
Abhishek Sur

Reviewers
Carlos Hulot
Ahmed Ilyas
Sergiy Suchok
Ken Tucker

Acquisition Editor
Kartikey Pandey

Lead Technical Editor
Susmita Panda

Technical Editors
Jalasha D'costa
Amit Ramadas

Project Coordinator
Anish Ramchandani

Proofreader
Claire Cresswell-Lane

Indexer
Tejal Soni

Graphics
Aditi Gajjar

Production Coordinator
Aparna Bhagat

Cover Work
Aparna Bhagat

About the Authors

Abhishek Sur is a Microsoft MVP in Client App Dev since 2011. He is an architect in the .NET platform. He has profound theoretical insight and years of hands on experience in different .NET products and languages. Over the years, he has helped developers throughout the world with his experience and knowledge. He owns a Microsoft User Group in Kolkata named **KolkataGeeks**, and regularly organizes events and seminars in various places for spreading .NET awareness. A renowned public speaker, voracious reader, and technology buff, his main interest lies in exploring the new realms of .NET technology and coming up with priceless write-ups on the unexplored domains of .NET. He is associated with the Microsoft Insider list on WPF and C#, and is in constant touch with product group teams. He holds a Masters degree in Computers along with various other certificates to his credit.

On the web, Abhishek is a freelance content producer, developer, and a site administrator. His website `abhisheksur.com` guides both budding and experienced developers to understand the details of languages and latest technology. He enjoys a huge fan following on social networks. You can reach him at `books@abhisheksur.com`, or get online updates from Facebook or Twitter `@abhi2434`.

Acknowledgement

Writing this book would not have been possible without the help of many people. Knowledge is a priceless alternative of time. While writing this book, I have spent a considerable amount of time in reading a lot of online journals, documentations, and blogs to explain and present concepts and ideas in a more lucid way.

First of all, I am extremely thankful to the entire Packt Publishing team for their continuous help. I would like to thank Anish Ramchandani and Susmita Panda, who have been very supportive and helped me to meet the deadlines. My most sincere thanks to my reviewers Carlos Hulot, Ahmed Illyas, Sergiy Suchok, and Ken Tucker for reviewing every line I wrote for this book, which resulted in bringing so many positive changes in this book. Last but not the least, a lot of thanks to Amit Ramadas and Jalasha D'costa for editing the chapters of this book.

I would also like to thank my colleague cum friends Shibatosh, Ranjit, Pallab, Ayan, Malini, and other members of our group for their continuous support and motivation to write this book. I am also thankful for being in touch with Mr. Raj Goswami, CEO and Mr. D. K Goswami, Director of BuildFusion, India. I would also like to extend my sincere thanks to Anoop Madhusudan, Abhijit Jana, Kunal Chowdhury, Dhananjay Kumar, Karthikayan Anabarasan, Lohith, Abhishek Kant, and Shivprasad Koirala for their continuous help and motivation.

I would also like to acknowledge my fiancée Riya for helping me and motivating me to write better. I want to dedicate this book to her.

Being a community lover I used to write a lot of blogs and articles, but this is the first time I am putting my experience and efforts in the form of a book. I am really thankful to all my readers and followers all over the world who loved my blogs and hopefully they will love this book too.

About the Reviewers

Carlos Hulot has been working in the IT area for more than 20 years in different capabilities, from software development, project management to IT marketing product development and management. Carlos has worked for multinational companies like Royal Philips Electronics, PricewaterhouseCoopers, and Microsoft. Currently, Carlos is working as an independent IT consultant. Carlos is a Computer Science lecturer at two Brazilian universities. Carlos holds a Ph.D in Computer Science and Electronics from the University of Southampton, UK, and a B.Sc. in Physics from the University of São Paulo, Brazil.

Ahmed Ilyas has a bachelor's degree in engineering from Napier University in Edinburgh, Scotland, and has majored in software development. He has 15 years of professional experience in software development.

After leaving Microsoft, he ventured into setting up his consultancy company offering the best possible solutions for a magnitude of industries and providing real-world answers to problems. They only use the Microsoft stack to build these technologies, to be able to bring in best practice, patterns, and software to their client base. Thus, enabling long term stability and compliance in the ever changing software industry and also improving software developers around the globe—pushing the limits in technology as well as develop themselves to become better.

This went on to being awarded the MVP in C# three times by Microsoft for providing excellence and independent real-world solutions to problems that developers face.

With the breadth and depth of knowledge he as obtained not only from his research but also with the valuable wealth of information and research at Microsoft, the motivation and inspirations come from this, with 90 percent of the world using at least one form of Microsoft technology.

Ahmed Ilyas has worked for a number of clients and employers. With the great reputation that he has, this has resulted in having a large client base for his consultancy company, Sandler Ltd (UK) which includes clients from different industries from, media to medical and beyond. Some clients have included him on their "approved contractors/consultants" list, which include ICS Solution Ltd and he has been placed on their "DreamTeam" portal and also CODE Consulting/EPS Software (`www.codemag.com`, based in the US).

Ahmed Ilyas has also been involved, in the past, in reviewing books for Packt Publishing and wishes to thank them for the opportunity once again.

I would like to thank the author/publisher of this book for giving me the great honor and privilege in reviewing the book. I would also like to thank my client base and especially Microsoft Corporation and my colleagues over there for enabling me to become a reputable leader as a software developer in the industry, which is my passion.

Sergiy Suchok graduated in 2004 with honors from the Faculty of Cybernetics, Taras Shevchenko National University of Kyiv (Ukraine), and has since then been keen on information technology. He currently works in the banking area and has a PhD in Economics. Sergiy is the coauthor of more than 45 articles and has participated in more than 20 scientific and practical conferences devoted to economic and mathematical modeling. He is a member of the New Atlantis Youth Public Organization (`newatlantida.org.ua`) and devotes his leisure time to environmental protection issues, historical, and patriotic development and popularization of a grateful attitude towards the Earth. He also writes poetry and short stories and makes macramé.

I would like to thank my wife and my young daughter for their patience and understanding while reviewing.

Ken Tucker is a web developer for Sea World, and has been a Microsoft MVP since October 2003. In his spare time he enjoys writing Windows Phone and Windows Store apps.

Table of Contents

Preface

Moving through the last few decades, from being the new kid on the block, .NET has turned itself into the most talked about young thing on the IT horizon. More and more people are getting inclined to use .NET, to make a mark in their career and also, more and more, clients are coming to .NET to accomplish their dreams. The recent buzz about C# being the best programming language has now made the world of developers very eager to build .NET applications. As the crowd is growing in the developers arena, there is always a need to have a clear perception on how things work and ways to make things better.

Visual Studio 2012 and .NET 4.5 Expert Development Cookbook mainly focuses on things that you need to know in those crunch situations as a developer. It also tries to feed you with as much details as it can, and covers as much technological domain in the world of .NET. The book is written in the form of recipes with step-by-step tutorials on every topic where the developer accompanies the author in this wonderful journey into the known and hitherto unknown realms of .NET. The recipes in the book, which are mostly sought out by developers on the Internet, are chosen in such a way so as to practically demonstrate them and not restrict developers only to them, but to also expose the other unexplored domains on the same topic to give them a clear view of the whole picture. There is a special section for each recipes bearing the heading *There's more...*, which always focuses on giving you extra knowledge on things that you might have missed without it. By the time you come to the end of this journey, you will feel the comfort and enjoy the confidence that a clear understanding of the insight of .NET gives you.

The book is a practical handbook that could give you optimal utilization of time for knowledge. It separately presents the topics very precisely and elaborates the same which you can rely on for a deeper look. It explains the recipes with proper sample code blocks that might make the usage of each topic very clear and make you utilize the final source code while writing your real-world applications. The examples taken for this book will clear your understanding of how things exactly work for that particular recipe and also adapt you as a developer to make use of the same source code in your production environment efficiently and quickly. If you want to utilize your busy schedule to explore all the necessary ongoing technology in the market, this book is best suited for you.

The book focuses on giving you:

- Maximum utilization of time for learning
- Major insights into ongoing technologies such as Visual Studio 2012, .NET 4.5, ASP.NET, Windows 8 Applications, Windows Presentation Foundation, HTML5, jQuery, memory management, and so on
- Practical examples on procedures to create real-world applications
- Step-by-step examples to create simple applications based on heads

What this book covers

Chapter 1, *Introduction to Visual Studio IDE Features*, starts with a basic introduction to Visual Studio IDE and gives the developer insights into how to increase productivity of development using a common set of tools and features present inside the IDE.

Chapter 2, *Basics of .NET Programs and Memory Management*, introduces the intersection of a .NET program and its core components. It dives deep in demonstrating the .NET infrastructure with detailed explanation of memory management and related techniques.

Chapter 3, *Asynchronous Programming in .NET*, focuses on introducing all existing techniques to deal with threading in .NET followed by the newer patterns that takes over the existing working principles with in-depth explanation on their working principles.

Chapter 4, *Enhancements to ASP.NET*, gives you an introduction to latest enhancements of ASP.NET 4.5 with HTML5 and jQuery. It also introduces some of the performance boosters available in .NET 4.5 and Visual Studio 2012 with ASP.NET.

Chapter 5, *Enhancements to WPF*, introduces the enhancements to WPF 4.5 and the major components of WPF. It gives a practical implementation of MVVM based WPF application covering all the facets required to program in WPF environment.

Chapter 6, *Building Touch-sensitive Device Applications in Windows 8*, introduces the new programming model for developing Windows 8 style tiles application. It gives a step-by-step introduction in how to program using HTML5 and JavaScript as well as WPF and C# for developing Windows 8 applications.

Chapter 7, *Communication and Sharing Using Windows 8*, focuses on how to implement network-enabled applications in Windows 8 with step by step implementation on how sharing and searching works inside the Windows 8 environment.

Appendix, *.NET languages and its Construct*, focuses on giving insights on how languages work in the .NET framework and C# with details explanation with examples of various features of C# language.

You can download this Appendix from `http://www.packtpub.com/sites/default/files/downloads/6709EN_Appendix_NET_Languages_and_its_Construct.pdf`

What you need for this book

The basic software requirements for this book are as follows:

- Microsoft .NET Framework 4.5 and higher
- Microsoft Visual Studio 2012 Express or higher editions
- Windows 8 Operating System (especially to work with Chapter 6 and 7)
- Latest web browsers

Who this book is for

The purpose of this book is to give you ready made steps in the form of recipes to develop common tasks that, as a developer, you might often be required to access. The book utilizes its chapters skillfully to provide as much information as it can and also with as much detail as necessary to kick start the subject. The book also delivers in-depth analysis of some advanced section of the subject to get you to expertise level. If you are a starter in the development environment and want to get expertise on ongoing technologies in the market, this book is ideal for you. Even for architects and project managers, the book can be a guide to enrich their existing knowledge.

The book uses C# and Visual Studio 2012 with Windows 8 (as the operating system) in the examples. Even though the book does not require any knowledge to start, it expects some basic theoretical and practical overall experience on the subjects to understand the recipes. The book bridges the gap between a normal developer to an expert architect.

Conventions

In this book, you will find a number of styles of text that distinguish between different kinds of information. Here are some examples of these styles, and an explanation of their meaning.

Code words in text are shown as follows: "The `requestValidationMode` attribute of `httpRuntime` defines how the request is validated to the server before calling `HttpPipeline`."

A block of code is set as follows:

```
(function($){
  $.fn.extend({
    Value1 : 20,
    myMethod : function(msg){
      alert(msg + "value : " + this.Value1);
    }
  });
})(JQuery);
```

Any command-line input or output is written as follows:

```
Install-Package Microsoft.Web.Optimization
```

New terms and **important words** are shown in bold. Words that you see on the screen, in menus or dialog boxes for example, appear in the text like this: "you can also open Nuget package manager by right-clicking on the references folder of the project and select **Add Library Package Reference**".

[Warnings or important notes appear in a box like this.]

[Tips and tricks appear like this.]

Reader feedback

Feedback from our readers is always welcome. Let us know what you think about this book—what you liked or may have disliked. Reader feedback is important for us to develop titles that you really get the most out of.

To send us general feedback, simply send an e-mail to `feedback@packtpub.com`, and mention the book title via the subject of your message.

If there is a topic that you have expertise in and you are interested in either writing or contributing to a book, see our author guide on `www.packtpub.com/authors`.

Customer support

Now that you are the proud owner of a Packt book, we have a number of things to help you to get the most from your purchase.

Downloading the example code

You can download the example code files for all Packt books you have purchased from your account at `http://www.packtpub.com`. If you purchased this book elsewhere, you can visit `http://www.packtpub.com/support` and register to have the files e-mailed directly to you.

Errata

Although we have taken every care to ensure the accuracy of our content, mistakes do happen. If you find a mistake in one of our books—maybe a mistake in the text or the code—we would be grateful if you would report this to us. By doing so, you can save other readers from frustration and help us improve subsequent versions of this book. If you find any errata, please report them by visiting `http://www.packtpub.com/submit-errata`, selecting your book, clicking on the **errata submission form** link, and entering the details of your errata. Once your errata are verified, your submission will be accepted and the errata will be uploaded on our website, or added to any list of existing errata, under the Errata section of that title. Any existing errata can be viewed by selecting your title from `http://www.packtpub.com/support`.

Piracy

Piracy of copyright material on the Internet is an ongoing problem across all media. At Packt, we take the protection of our copyright and licenses very seriously. If you come across any illegal copies of our works, in any form, on the Internet, please provide us with the location address or website name immediately so that we can pursue a remedy.

Please contact us at `copyright@packtpub.com` with a link to the suspected pirated material.

We appreciate your help in protecting our authors, and our ability to bring you valuable content.

Questions

You can contact us at `questions@packtpub.com` if you are having a problem with any aspect of the book, and we will do our best to address it. You can also contact the author directly at `books@abhisheksur.com` to address any technical problems while covering the recipes.

1
Introduction to Visual Studio IDE Features

In this chapter, we will start with a basic introduction to Visual Studio IDE, and understand how we can increase the productivity of our development using some of the tools and features present in the IDE. After going through the chapter, you will understand the following recipes:

- Identifying the various components of Visual Studio IDE
- Working with Solution Explorer and Class View
- Working with the main workspace area of IDE
- Navigating between code inside the IDE
- Extending Visual Studio templates
- Using Code Snippets in Visual Studio
- Using Smart Tags and Refactor in Visual Studio

Introduction

Ever since Microsoft announced .NET for the first time almost 10 years ago, there has been a lot of noise in the developer community about the way the changes are going. .NET led its way to modernize the ideas of coding with more sophisticated techniques by adopting more object-oriented paradigm in programming and also changing the style of coding altogether. The Microsoft forerunner VB was announced to be modernized in the new environment and redesigned to be named as VB.NET, and also some other languages that are totally different in syntax, such as C#, J#, and C++ have been announced. All of these languages are built on top of the .NET Runtime (known as **Common Language Runtime** or **CLR**) and produce the same intermediate output in **Microsoft Intermediate Language** (**MSIL**).

Microsoft announced .NET runtime as a separate entity by defining standardized rules and specifications that every language must follow to take advantage of CLR. The entirely new set of libraries, classes, syntaxes, or even the way of coding in Microsoft technologies, created a huge hindrance in the developer community. Many developers switched their jobs, while there are a few who really switched gears to understand how to work with the new technology that is totally different from its predecessors. The community has already started to realize that the existing set of Microsoft tools might not satisfy the needs of new evolving technology. Microsoft had to give a strong toolset to help the developers to work easier and better with the new technology.

Visual Studio is the answer to some of them. Microsoft Visual Studio is an **Integrated Development Environment** (**IDE**) to work with Microsoft languages. It is the premier tool that developers can posses to easily work with Microsoft technologies. But you should note, Visual Studio is not a new product from Microsoft. It has been around for quite sometime, but the new Visual Studio had been redesigned totally and released as Visual Studio 7.0 to support .NET languages.

Evolution of Visual Studio

As time progressed, Microsoft released newer versions of Visual Studio with additional benefits and enhancements. Visual Studio being a plugin host to host number of services as plugins, has evolved considerably with a lot of tools and extensions available; it has been the integral part of every developer's day-to-day activity. Visual Studio is not only a tool used by developers, but it has been identified that a large number of people who are not a part of the developer community have been loving this IDE and using it for editing/managing documents. The wide acceptance of Visual Studio to the community had made the product even better.

This year, Microsoft has released the latest version of Visual Studio. In this chapter, we will tour Visual Studio IDE features, its utilities, and mostly cover parts that can really help to make your work done more quickly.

Identifying the various components of Visual Studio IDE

Visual Studio 2012 has come up with lots of new enhancements and features. Some of these features widely enhance productivity of development. Knowing your IDE better is always an advantage to a developer. In this recipe, we will try to get our hands on to various Visual Studio IDE features to get started with using Visual Studio.

Getting ready

Before we start using Visual Studio, we need to first make a choice on which version practically suits us. Let's have a look at the features of all the versions of Visual Studio.

- **Visual Studio Express**: If you are looking to try out small applications or medium-sized applications and do not want to spend a single penny from your pocket, Visual Studio Express is the right choice for you. Microsoft has given the Express build free to everyone that is capable of doing all the basic needs of software build up.
- **Visual Studio Professional**: This edition of Visual studio is for individual development with most of the important debugging tools and all the things a developer commonly needs. So if your primary orientation of using the IDE is basic development, this would be the right choice for you. This edition is reasonable in price too.
- **Visual Studio Premium**: Visual studio Premium edition is for people who make high-quality usage of the IDE. It adds tools for testing, code analysis, debugging, profiling, discovers common coding errors, generate test data, and so on.
- **Visual Studio Ultimate**: This is the ultimate edition of the product with all the components that could exist within Visual Studio. This edition provides advanced debugging capabilities with all architecture and modeling tools with it.

You can find the entire comparison list between all the versions of Visual Studio from the link below:

`http://www.microsoft.com/visualstudio/eng/products/compare`

Once you are determined on what suits your requirement best, you can install it on your machine and we are ready to go.

[

If you are opening the IDE for the first time, the IDE will present you few options and ask you what type of development you want to take. In most cases, I would recommend you to choose **General Development**, as this is the most convenient layout of Visual Studio.

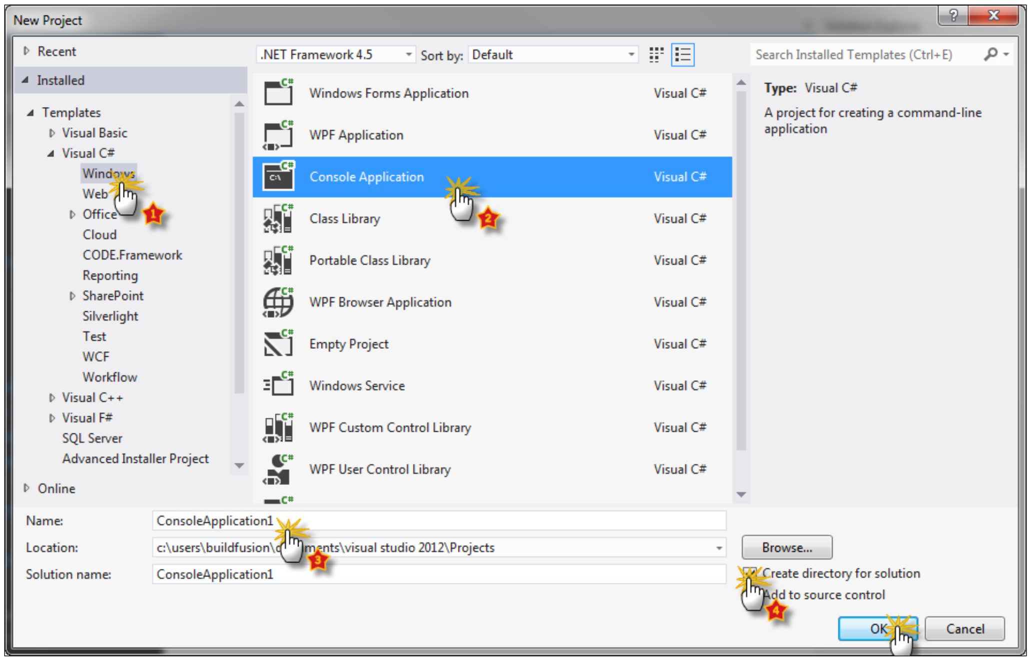
]

How to do it...

In this recipe, we will understand the different sections of the Visual Studio 2012 IDE and will show you where to start.

1. To start with the recipe, let us navigate to the **Start** menu | **All Programs**, choose the `Visual Studio 2012` folder and select **Visual Studio 2012**. This will launch the Visual Studio IDE.

2. After displaying the initial splash screen for a while when the IDE has been loaded, it presents you with a **Start** page with a three main options:
    - **Connect to Team foundation server**
    - **New Project**
    - **Open Project**
3. We can use either **New Project** here from the link, or we can navigate to **File** | **New Project** to create a new project. This pops up a **New Project** dialog box. In this dialog box, we have a number of options available based on the packages that are currently installed with the IDE. On the left-hand side of the dialog box, we see a tree of various items installed within the IDE. You can see there are a number of templates listed in the tree. When one item on the left-hand side gets selected, the corresponding templates associated with that group will be listed in the middle section of the dialog box (marked as **2** in the next screenshot).
4. Select an appropriate name for your project in the **Name:** field (marked as **3** in the next screenshot).
5. You can select the **Create directory for the solution** checkbox (marked as **4** in the next screenshot) to indicate that a new folder with the name just specified will be created inside the location you specify, which will hold the various files rather than storing them directly inside the specified location.


New Project
Recent
Installed
Templates
Visual Basic
Visual C#
Windows
Web
Office
Cloud
CODE.Framework
Reporting
SharePoint
Silverlight
Test
WCF
Workflow
Visual C++
Visual F#
SQL Server
Advanced Installer Project
Online
Search Installed Templates (Ctrl+E)
Windows Forms Application Visual C#
WPF Application Visual C#
Console Application Visual C#
Class Library Visual C#
Portable Class Library Visual C#
WPF Browser Application Visual C#
Empty Project Visual C#
Windows Service Visual C#
WPF Custom Control Library Visual C#
WPF User Control Library Visual C#
Browse...
Add to source control
OK
Cancel
1
2
3
4

.NET Framework 4.5 Sort by: Default

Type: Visual C#

A project for creating a command-line application

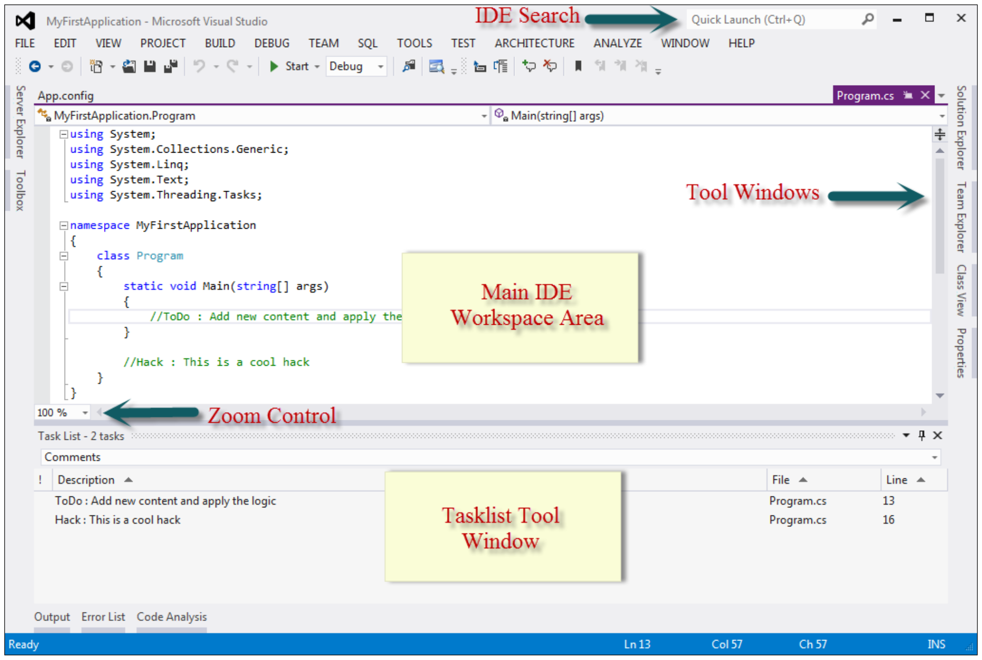
Name: ConsoleApplication1

Location: c:\users\buildfusion\documents\visual studio 2012\Projects

Solution name: ConsoleApplication1

✓ Create directory for solution

6. We choose **Visual C#** from the left-hand side pane, **Console Application** from the middle pane; keeping the default name we click on **OK** as shown in the previous screenshot. If everything is good, it opens the IDE and displays something as shown in the following screenshot:



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

namespace MyFirstApplication
{
    class Program
    {
        static void Main(string[] args)
        {
            //ToDo : Add new content and apply the
        }

        //Hack : This is a cool hack
    }
}
```

!	Description	File	Line
	ToDo : Add new content and apply the logic	Program.cs	13
	Hack : This is a cool hack	Program.cs	16

7. In the previous screenshot, we have marked a few sections of the IDE which need special attention. They are as follows:

    - The first section is **IDE Search**, which is just a blank textbox to search the IDE component.
    - **Tool Windows** are docked on the left, right, or bottom of the screen. When a tool window is open as shown in **TaskList Tool Window** at the bottom, it shows up a small dockable container and when it is collapsed, it shows a reference of it in the IDE sidebar as shown in the left-hand and right-hand side of the window.
    - The main IDE workspace area represents the main working area of the IDE. This forms the major portion of the IDE and mainly the application developer writes code here.
    - A special **Zoom Control** is also there inside the IDE, which helps to zoom in and out of the editor.

8. Finally, you can start writing your code in the main working area of the IDE or start exploring other options in the IDE yourself.

How it works...

There are a few things that need attention when a Visual Studio IDE is opened. Visual Studio is a process that is launched using an executable called `devenv` (which can be spelled as **Developer's Environment**). You can either double-click on the Visual Studio icon from the **Start** menu (which most of the people do), or go to **Start** and then search for `devenv` to run the IDE. The IDE is generally invoked in default permission mode. Sometimes, it is important to open the IDE as **Administrator** to enjoy administrative features on the environment. To change this behavior, you can right-click on the shortcut and select **Run as Administrator**. You can also permanently set the IDE to run as administrator from the **Properties** menu.

After the Visual Studio initial splash screen is displayed during the opening sequence, the first thing that you see is the **Start** page. We have navigated to **File** | **New Project** to open the **New Project** dialog box. As shown in the first screenshot, on the left-hand side of the window (marked as **1**), we see a tree of all the installed project type groups into collapsible panels.

If you do not find your template, you can also use **Search Installed Template** to search any template by its name in the right-hand corner of the dialog box.

As more than one framework can coexist in the same PC, the **New Project** dialog box is smart enough to allow you to choose the Framework that you need to use while deploying the application. By default it shows .NET 4.0 as the framework for the project, but you can change it by selecting the dropdown. The whole environment will change itself to give you only the options available for your current selection.

We choose **Visual C#** from the left tree and select **Console Application** from the middle pane as project template. Upon choosing any template, the description of the current template is loaded on the right-hand side of the screen. It gives you a brief idea on what **Console Application** is and is capable of doing.

At the bottom, we have the option to name the project and the solution, and we also have option to select the location where the project needs to be created (marked as **3**). You can select your own folder path to store the files you create inside the project by choosing the appropriate filesystem path in the box.

There are two checkboxes available as well. One of them is **Create directory for solution**, when selected (which is by default remains selected) creates a directory below the chosen path and places the files inside it. Otherwise it will create files just inside the folder chosen. To make it a habit, it is good to keep it selected.

Finally, click on **OK** to create the project with default files.

After the project is created, the basic IDE you see looks like the screenshot in step 5. We will now divide the whole IDE into those parts and explore the IDE together in the recipes that follow.

Let's paste the code inside the `Main` method that you see when you open the program class and paste the following code between the curly braces of `Main` method:

```
string content = "This is the test string to demonstrate Visual Studio
IDE features. ";
string content2 = "This is another string content";
Debug.Assert(content.Equals(content2), "The contents of the two
strings are not same");
Console.WriteLine("Thanks!");
Console.ReadKey(true);
```

Downloading the example code

You can download the example code files for all Packt books you have purchased from your account at `http://www.packtpub.com`. If you purchased this book elsewhere, you can visit `http://www.packtpub.com/support` and register to have the files e-mailed directly to you.

After you paste the code, let's press *F5* on the keyboard. You will see a **Console** window that appears inside the IDE showing a message. Press the *Enter* key to close the **Console**.

There's more...

There are lots of other options that Visual Studio comes up with. In spite of opening Visual Studio normally, you can also use some special options to handle Visual Studio better. Let's try to look into other options that can be good to eye on.

Visual Studio command switches

Visual Studio being a normal executable that runs under Windows also provides some switches that can be used when we open the IDE. To use Visual Studio with these switches, we need to either use command prompt or use **Run** to add switches to the IDE. To use command prompt, just navigate to the location `%systemdrive%\Program Files\Microsoft Visual Studio 11\VC` and type in `devenv /?` to get a list of all the command switches available for the IDE. For instance `devenv /resetsettings`.

This command switch will reset all the user settings that have been applied to the IDE. The reset settings can also be used to specify the `vssettings` file, which has been used up to override settings for the current IDE. Similarly, you can use `devenv /resetSkipPkgs`.

This command will reset loading of all user-related tags associated to the packages that need to load to make everything work smoothly with Visual Studio. Sometimes if the IDE gets corrupted or loading time increases, you can also turn on diagnostic load using `devenv /SafeMode`.

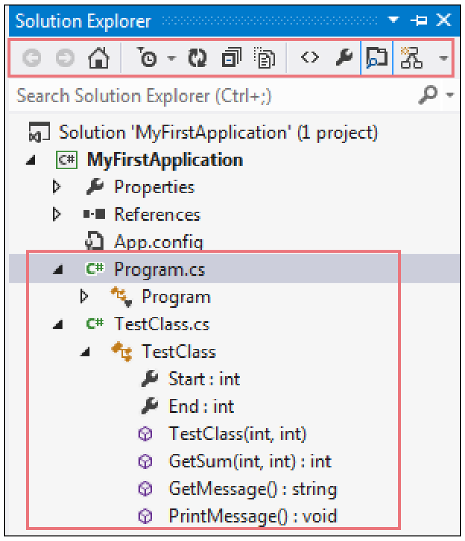
You can also try running a project without opening the IDE totally. I mean if you only need to open the IDE, run the project, and exit. The best option is to use Command switch `devenv /runexit "[solution/Project Path]"`.

You need to replace the `[solution/Project path]` with the path where you find the `Solution` file (`.sln` extension) or project files (`.csproj/.vbproj files`).

To see all the command switches supported by the executable, you can try `devenv /?` from the command prompt too.

Working with Solution Explorer and Class View

The most important part of the IDE that you need most often is your **Solution Explorer**. **Solution Explorer**, which resides on the right-hand side of the IDE, is the most widely used navigation tool between files and classes. It is shown in the following screenshot:



In the above screenshot, you see the basic structure of **Solution Explorer** when the IDE is loaded with a project. The **Solution Explorer** window starts with the `Solution` file and loads all the projects that are associated with the solution in a tree. The very next node of the `Solution` file generally is the project file. For the sake of identifying each of the files in the IDE, it provides you the proper icon and also makes the project file names bold.

How to do it...

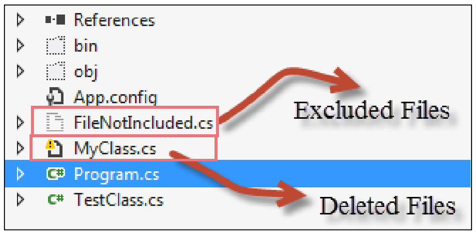
In this recipe, we are going to explore **Solution Explorer**.

1. On the right-hand side of the IDE, you will see **Solution Explorer** as shown in the preceding screenshot. This is the main screen area where you can interact with the files and folders associated with your project. If you do not see **Solution Explorer**, please navigate to **View** | **Solution Explorer** from the menu or press *Ctrl* + *W*, *S* from the keyboard.
2. Once you see the **Solution Explorer**, it should contain a tree of all the files and folders that are associated with the project. Toggle each node in the tree to see its related information. You can see in the figure, the **Solution Explorer** is capable of showing the list of members of a type that is written inside a file as well. The node **TestClass**, when opened, shows a tree of all its members in subsequent nodes.
3. The header section of **Solution Explorer** contains a number of buttons. These buttons are commands associated with the current selection of the tree node.

How it works...

Solution Explorer is the main window that lists the entire solution that is loaded to the IDE. It gives you an organized view of projects and files that are associated with the solution for easy navigation in a form of a tree. The outermost node of the **Solution Explorer** is the `Solution` itself, and below it are the projects, then files/folders. The `Solution` file also supports you to load folders directly inside the solution and even store documents in the first level. The project that is set as startup is marked in bold.

There are a number of buttons stacked at the top of the **Solution Explorer** window called toolbar buttons, and based on the type of file that is selected in the tree will be made available or disabled. Let's talk about each of them individually:



The solution tree in Visual Studio 2012 also loads the entire structure of the class into its nodes. Just expand the `.cs` file and you will see all its members and classes are listed. Visual Studio also has a class view window, but **Solution Explorer** is smart enough to list all the Class View elements inside its own hierarchy. You can open Class View by navigating to **View** | **ClassView** or pressing *Ctrl* + *W*, *C*, to see only the portion of class and its members.

Another important consideration is **Solution Explorer** as it shows the files from the Solution file, it also tracks the actual existence of the file in the physical locations too. While loading the files sometimes, it might show exclamatory signs if the file doesn't exists in physical location.

Here the MyClass file, even though it is a `.cs` file, does not show up the usual icon, but shows one exclamatory sign which indicates that the file is added to the solution, but the physical file does not exists.

On the contrary, some files are shown in the solution as blank files, (in our case `FileNotIncluded.cs` or folders like `bin/obj`). These files, even though they exist in the filesystem, are not included in the solution file.

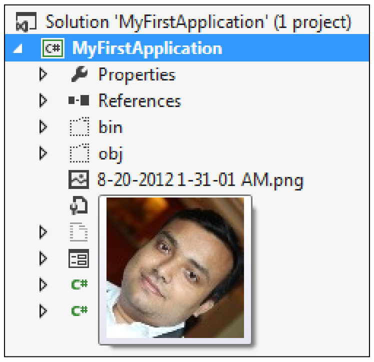
Each of the files show one **Additional Information** button on the right-hand side of the tree node in the solution. This button gives extra information associated with the file. For instance, if you click on the button corresponding to a `.cs` file it will pop up a menu with **Contains**. This will get the associated class view for the particular file in the solution. The menu can be pretty long depending on the items that cannot be shown in generalized toolbar buttons. When the solution loads additional information, there are forward/back buttons which can be used to navigate between views in the solution.

There's more...

In addition to the basic updates to **Solution Explorer**, there are lots of other enhancements that are made to the **Solution Explorer** to increase productivity and better user experience to the IDE. Let's explore them one by one.

Previewing images in Solution Explorer

Solution Explorer shows a preview of images just by hovering on the image without actually opening the image. This is a new enhancement to **Solution Explorer** and is not available with any previous version of Visual Studio IDE.

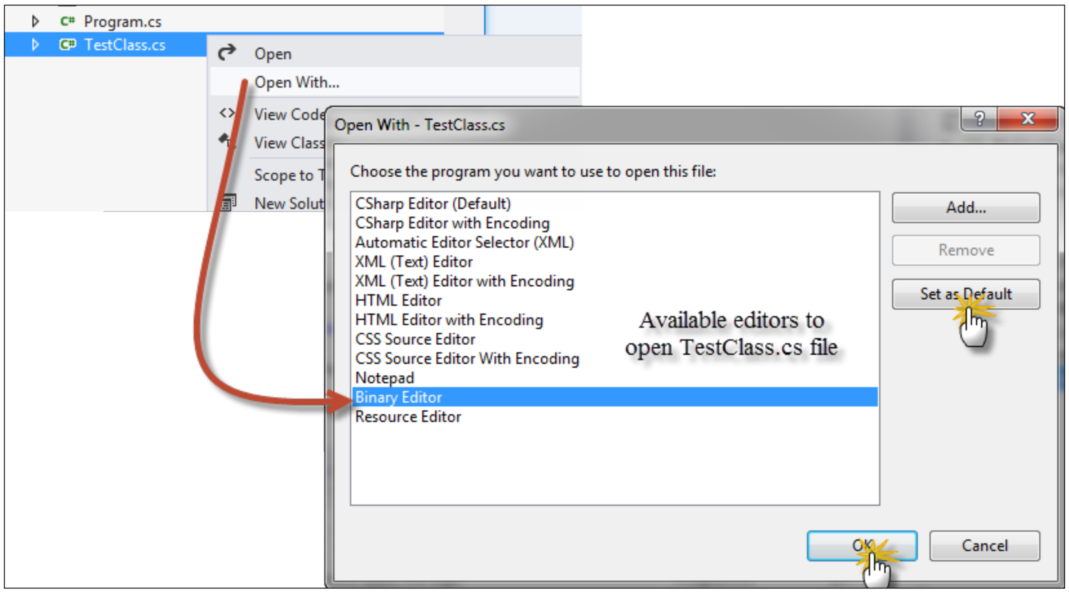


Add an image in the solution by right-clicking on the project and navigating to **Add** | **Add Existing Item** and select an image from the filesystem.

If you did it correctly, the image will be loaded in the tree as in the screenshot. Just hover the mouse pointer over the image, and you will see a small preview of the image as shown in the previous screenshot.

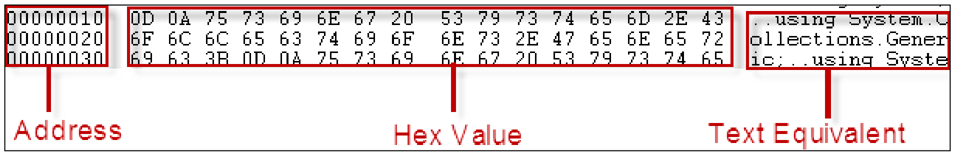
Different IDE editors

Visual Studio comes with a number of editors installed within it by default. Based on the type of the file, this editor gets loaded onto the IDE. For instance, if you double-click on an image it will open it in image editor, while when you choose a `.cs` file, it will open it in a C# editor.


Program.cs
TestClass.cs
Open
Open With...
View Code
View Class
Scope to T
New Solut
Open With - TestClass.cs
Choose the program you want to use to open this file:
CSharp Editor (Default)
CSharp Editor with Encoding
Automatic Editor Selector (XML)
XML (Text) Editor
XML (Text) Editor with Encoding
HTML Editor
HTML Editor with Encoding
CSS Source Editor
CSS Source Editor With Encoding
Notepad
Binary Editor
Resource Editor
Add...
Remove
Set as Default
Available editors to open TestClass.cs file
OK
Cancel

You can right-click on any file from the `Solution` and select **Open With...** rather than using the normal double-click to open a dialog box, which lists all the available editors that can load the selected file. Some of the default IDE editors are C# Editor, C# Editor with Encoding, Automatic Editor Selector, XML Editor, HTML Editor, Notepad, Binary Editor, Resource Editor, and so on.

To open a CS file in Binary Editor, right-click on the CS file and choose **Open With...** choose **Binary Editor** and select **OK**. You can see from the following screenshot that the code file looks like a sequence of binary characters:



```
00000010  0D 0A 75 73 69 6E 67 20  53 79 73 74 65 6D 2E 43  ..using System.C
00000020  6F 6C 6C 65 63 74 69 6F  6E 73 2E 47 65 6E 65 72  ollections.Gener
00000030  69 63 3B 0D 0A 75 73 69  6E 67 20 53 79 73 74 65  ic;..using Syste
```

The first column shows the address of the bytes in the file, the second shows the actual byte content, and the third contains the string equivalent of the same.

You can also try out other editors in the list.

Working with the main workspace area of IDE

This section represents the main workspace area of the screen. This is the most important section of the Visual Studio IDE which the developers mostly use. The workspace generally fills up the entire IDE or most of the portion of the IDE. Each of the windows that can be loaded inside the IDE has the feature to toggle hidden, can float outside the IDE, and even be snapped into different dock positions.

In the main workspace area, a file has already been loaded for us with a class named `class1`. The editor associated with `.cs` file is loaded in the screen to show the file. There are a number of editors available with Visual Studio, each of them can be loaded directly in this section. Generally, we do our main development in this section of the IDE.

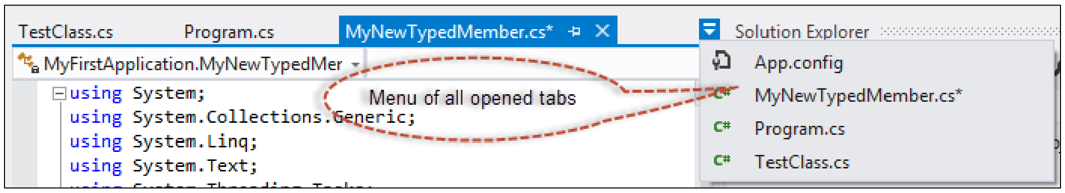
How to do it...

We will work the main workspace area of IDE by performing the following steps:

1. Close all associated windows in the IDE that you can see the portion of the IDE that still remain on the screen is the main workspace area.
2. Open **Solution Explorer**, double-click on some files. You can see each file produces a stack of tabs. Upon opening a new file, you can see the new tabs are stacked on the left-hand side of the tabs. Opening a large number of files in the IDE will produce a menu on the top-right corner of the screen.
3. Drag a tab and place it in between other tabs to reposition.
4. Change something in the file without saving the content. The tab header will indicate the update with a star sign. It will show a lock sign when you open a read-only file.
5. Use the **Toggle** button on one of the tabs to make it sticky, so that opening new files does not changes its position. If you are in the **Preview** tab, you will see a special **Promote** button, which will promote it as a new window to work on. The workspace contains the editor which forms the most of the part of IDE. This section loads the content of the actual file. The changes in the editor are tracked in yellow (when the change is not saved) and green (when the content is saved).
6. You can zoom the content of the editor using the **Zoom** dropdown in the bottom-left corner of the screen.

How it works...

The workspace loads the editors in tabs. So, when you pick two nodes from **Solution Explorer** to open the code window, Visual Studio keeps links to each of the files that are opened in separate tabs. Each tab header contains a few fixed set of items.

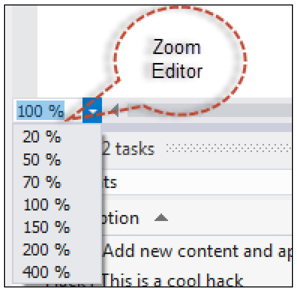


```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
```

In the previous screenshot, you can see that the tab header containing the name of the file (`MyNewTypedMember.cs`) that links to the tab, it shows a `*` when the item needs to be saved, it has a toggle pinner button (just like all other IDE tool windows), which makes the tab sticky on the left-hand side, and the close button. The title section sometimes also indicates an additional status, like when the file is locked it shows a lock icon, when the object is loaded from metadata it shows that in square braces as in the screenshot. In this section, as we keep on opening files it goes in to a stack of tab pages-one after another until it reaches the end. After the whole area is occupied, it finally creates a menu in the right most corner of the workspace title to hold a list of all the files that cannot be shown on the screen. You can select from this menu to choose which file you need to open. *Ctrl* + *Tab* can also be used to toggle between the tabs that are already loaded in the workspace.

Below the title of the tab, before the main workable area, there are two dropdowns. One has been loaded with the class that is opened in the IDE and the right one loads all the members that are created in the file. These dropdowns help easier navigation in the file by listing all the classes that are loaded in the current file on the left. On the right-hand side there is another which contextually lists all the members that are there in the class that is chosen on the right-hand side. These two dropdowns are smart enough to update automatically whenever any new code is added to the editor.

The main workspace area is bounded by two scroll bars that handle the overflow of the document. But after the vertical scroll bar, there is a special button to split the window.



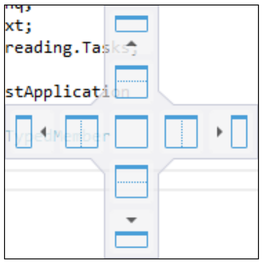
The horizontal scroll bar on the other hand holds another dropdown that shows the current zoom of the editor. Visual Studio now allows you to scale your editor to your preferred zoom level. The shortcut for the zoom is *Ctrl* + the scroll mouse wheel.

There's more...

As our basic overview of the Visual Studio 2012 IDE is over, let us have an insight on a few other features that are available inside the IDE.

Docking windows inside the IDE workspace

Let's go on opening a few of the windows in the IDE. You can start from **View** | **Windows Menu** of the IDE. After you have opened up quite a number of tool windows, you will find a requirement for easy arrangement of the windows. Visual Studio IDE is composed of a number of dock holders, each of which can freely flow both inside or outside of the main IDE. You can move a window by dragging its title bar as you do for any normal window. When you move the panel just inside Visual Studio, a set of controls appear as shown in the following screenshot, which indicates the dock positions available for the current window:



```
xt;
reading.Tasks;

stApplication
```

When you hold against the dock position, it will show you how it appears when stacked in a certain position (shown in the previous screenshot). Visual Studio will automatically adjust the stacks of items that are stacked through the Dock Managers.

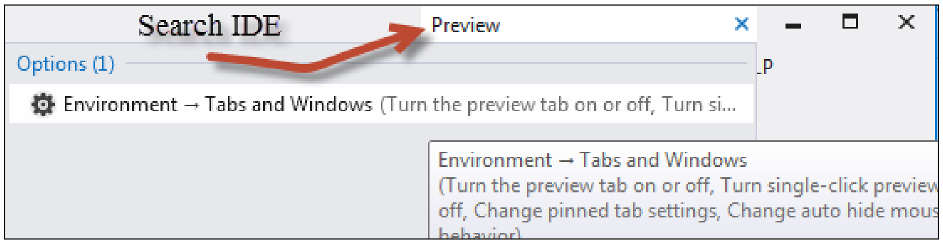
The whole IDE has a stack of tabs on each side of its workspace, which holds tabs to link each of these windows. As I have already mentioned, each of the tabbed window is capable of toggling to auto hide. When the window is hidden it shows only the tab link, while when the window is open, it associates itself into a set of tabs.

The other parts of the IDE consists of the menu bar, the standard toolbar, and the status bar. Each of them is used to give commands to the IDE.

Search IDE features

On the top-right corner of the screen, you will find a new search box. This is called the IDE search box. Visual Studio IDE is vast. There are thousands of options available inside IDE, which you can configure. Sometimes, it is hard to find a specific option that you want. The IDE search feature helps you find an option more easily.

As shown in the previous recipe, say if I forget where the option for **Preview File** tab on single click is available, I can type `Preview` in the search box.



Options (1)

Environment → Tabs and Windows (Turn the preview tab on or off, Turn si...

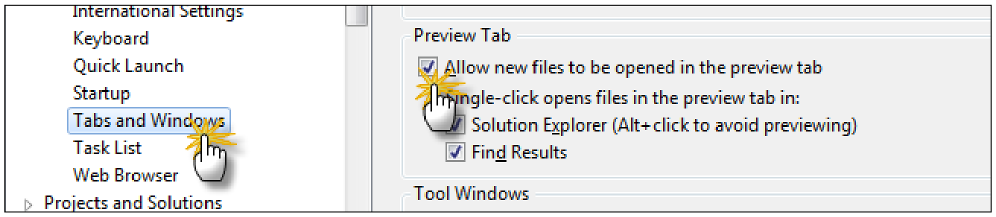
Environment → Tabs and Windows
(Turn the preview tab on or off, Turn single-click preview
off, Change pinned tab settings, Change auto hide mous
behavior)

As shown in the screenshot, Visual Studio will open a list of all the options where you can see the `Preview` text available. On choosing the first option, you will navigate to the appropriate location.

Preview files in IDE

Visual Studio IDE has a feature to preview files without opening the same in the IDE. Just with a single click on the file, it will open up in the main workspace area as preview. As we have already seen, the preview of the file is generally opened on the right-hand side of the IDE, and selecting another file replaces the previous preview, it is called as a temporary preview of file. The file can be promoted or opened either by working directly inside the preview, or by using a special promote button on the title of the preview tab.

Generally this option is disabled by default. You can navigate to **Tools** | **Options**, then in the left tree, **Environment** | **Tabs and Windows**, check the **Allow new files to be opened in preview Tab**.



Preview Tab

☑ Allow new files to be opened in the preview tab

Single-click opens files in the preview tab in:

☑ Solution Explorer (Alt+click to avoid previewing)

☑ Find Results

Tool Windows

The **Preview Tab** option is generally useful when you are working with a large number of files and loaded in the IDE.

Navigating between code inside the IDE

Visual Studio comes with a number of useful code navigators; most of them are pretty useful and handy. First of all let us try the **Code Highlighting** feature in Visual Studio.

How to do it...

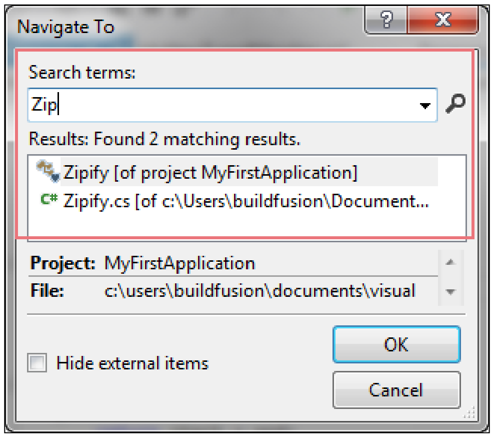
To try the Code Highlighting feature in Visual Studio, perform the following:

1. Double-click on a C# file to highlight each occurrence of the same type in the code.
2. Press *Ctrl* + *Shift* + down/up arrow to navigate between references.
3. Press *Ctrl* + ,(comma) to open the **Navigate to** dialog box to list all the navigation options available for the current selection (or under cursor position).
4. Right-click on any method or type and select **Go To Definition** to move to the definition of it.
5. Right-click on any of the type and select **Find All References** to list all the references in a new tool window.
6. You can select **View Call Hierarchy** from the right-click pop-up menu to list all the references to and from the member you have selected.
7. You can rename a type in the IDE and press *Ctrl* + .(dot) to open a menu and rename all its references.

How it works...

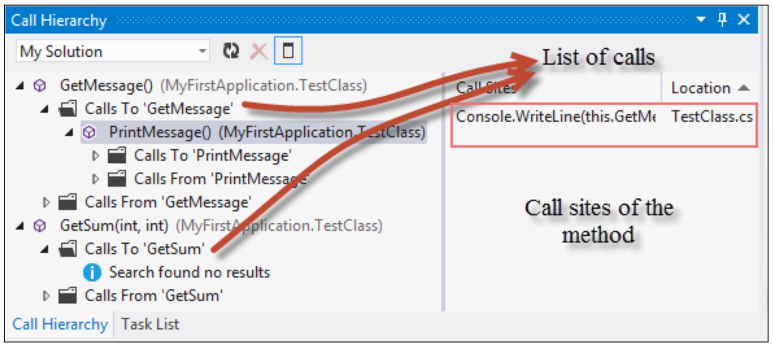
Navigating from one file to another and one type to another is an important thing while working with any project. Visual Studio IDE did a splendid job by giving us tools and features that help in navigating from one place to another easily.

Select any text in the project document by double-clicking on it, which highlights all the occurrences of the same code in the file. You can press *Ctrl* + *Shift* + down/up arrows to navigate between the references:


Navigate To
Search terms:
Zip
Results: Found 2 matching results.
Zipify [of project MyFirstApplication]
Zipify.cs [of c:\Users\buildfusion\Document...
Project: MyFirstApplication
File: c:\users\buildfusion\documents\visual
Hide external items
OK
Cancel

For advanced code navigation, you can press *Ctrl* + , to open a new window named **Navigate To** that quickly searches code members and files and list them.

The **Navigate To** dialog box also shows the file from which the window is invoked and the name of the project.



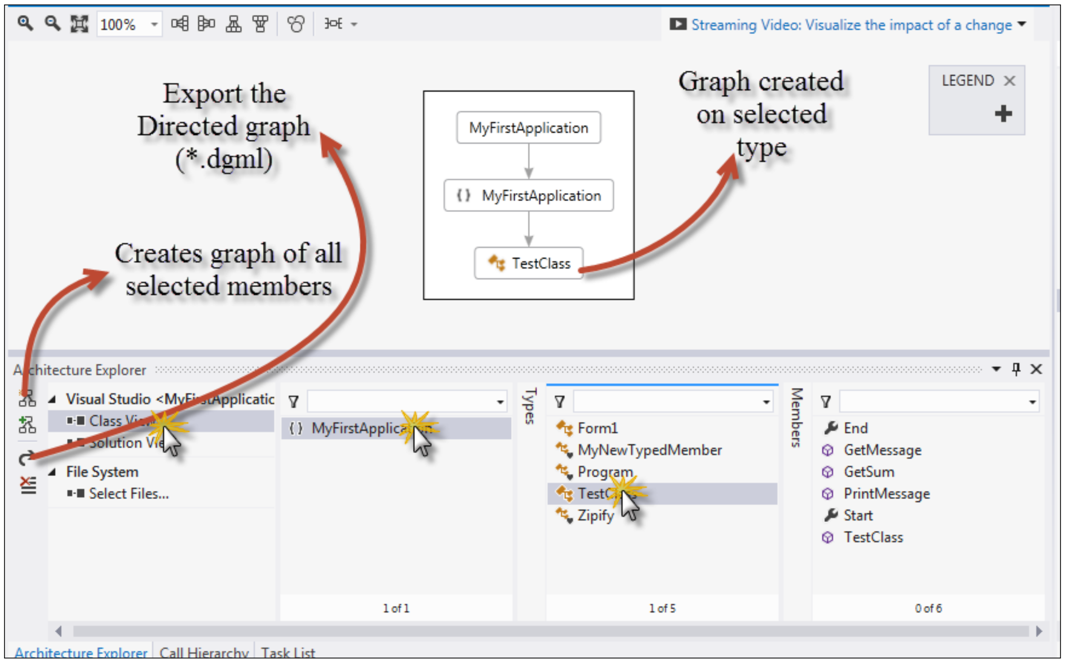
Another important code navigation tool is the **Call Hierarchy** window. You can invoke this window using right-click and selecting **View Call Hierarchy**. The **Call Hierarchy** window gives you the overview of all the references that are using the same code. You can also view the overrides of the method if you have any. The right-hand side of the window also lists all the calls by giving the exact line and the filename where the call has been made.

There's more...

In addition to the basic navigation tools inside the IDE, there are a few advanced options available to the IDE that help in navigating within the UI quickly, logically, and efficiently. Let's now take a look at a few other options available to us.

Architecture Explorer

One of the coolest additions to Visual Studio recently is **Architecture Explorer**. The tool helps you to navigate between the solutions assets very easily. Let's navigate to **View** | **Architecture Explorer** to get a window similar to the following screenshot:

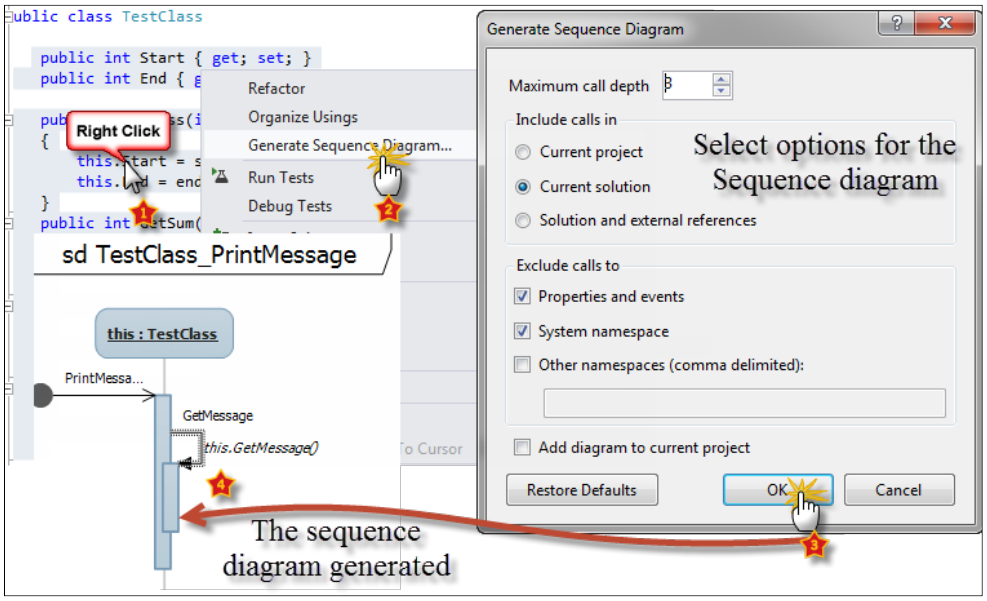


In the **Architecture Explorer** window you can view, navigate to the content of any class, namespace, or method. Visual Studio also has the facility to export the **Architecture Explorer** window in a graph document. You can select the items that you need to show in the graph and select the **Create New Graph Document** button on the top-left corner of the **Architecture Explorer** window. The graph will show up to analyze code members, circular references, unreferenced nodes, and so on and create a pictorial representation of the entire library. You can even export the document as a directed graph using the **Export Directed graph** button on the explorer.

The graph in the **Architecture Explorer** window can be exported to XPS document for future reference too.

Sequence diagrams

You can generate sequence diagrams from Visual Studio IDE by directly right-clicking on a method and by selecting the **Generate Sequence Diagram** option. This option is only available in the Ultimate version of Visual Studio. **Sequence diagram** will show a diagrammatic representation of the complete method body using a diagram.


Right Click
Refactor
Organize Usings
Generate Sequence Diagram...
Run Tests
Debug Tests
o Cursor
sd TestClass_PrintMessage
this : TestClass
PrintMessa...
GetMessage
this.GetMessage()
The sequence diagram generated
Generate Sequence Diagram
Select options for the Sequence diagram
Restore Defaults
OK
Cancel

```
ublic class TestClass

  public int Start { get; set; }
  public int End { g

  pub              ss(i
  {
     this.Start = s
     this.End = end
  }
  public int GetSum(
```

Maximum call depth: 3

Include calls in:
- ○ Current project
- ● Current solution
- ○ Solution and external references

Exclude calls to:
- ☑ Properties and events
- ☑ System namespace
- ☐ Other namespaces (comma delimited):

☐ Add diagram to current project

You can see in this method that I have used a few classes and objects, which are shown in the diagram based on their usage.

Task List

Visual Studio can be used to list task information on the project. You can open **Task List** from the **View** menu, which lists all the tasks that are outstanding in the project.

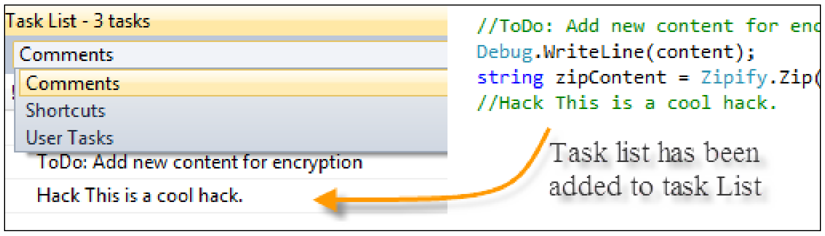
There are a number of options that you can use to create tasks in the project.

- When you comment something in the code with **Task List** tokens, your task will get listed in the **Comments** section. For example:

  ```
  // ToDo This is a task that is outstanding or
  // Hack This hack need to be changed
  ```

The **ToDo** task will be listed on the **Tasklist Tool** window.

- You can also create tasks directly inside the task window. Choose **User Tasks** and click on the button just beside the combo. You can also specify the priority of a task.



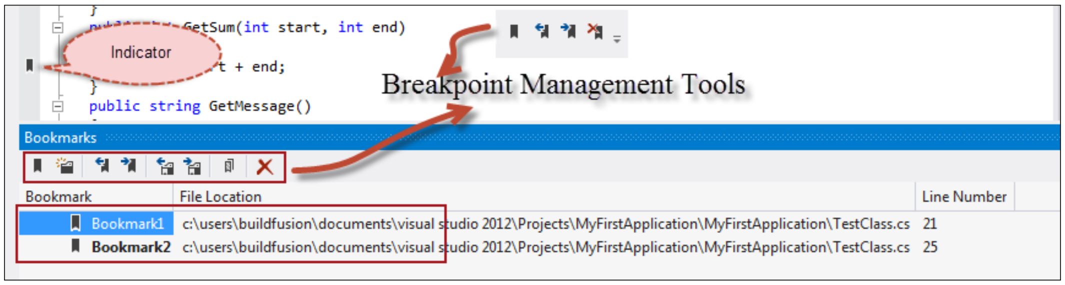
```
//ToDo: Add new content for enc
Debug.WriteLine(content);
string zipContent = Zipify.Zip(
//Hack This is a cool hack.
```

Task List - 3 tasks
Comments
Comments
Shortcuts
User Tasks
ToDo: Add new content for encryption
Hack This is a cool hack.
Task list has been added to task List

The commented tasks can be double-clicked to navigate to the appropriate line where the comment is written.

Bookmark menu

Another important window to manage code is **Bookmarks**. You can also use Bookmarks to navigate between code. To add a bookmark, go to the line where you want to apply a bookmark to, and select the **Toggle Bookmark** option in **Edit** | **Bookmarks** ,or press *Ctrl + B, T*. A white box will appear against the line. Once the bookmark is set, you can move to next and previous bookmarks using *Ctrl + B, N* and *Ctrl + B, P* respectively. You can clear all breakpoints either from the menu or simply choosing *Ctrl + B, C* from the keyboard. You can also open the **Bookmark** tool window to navigate between bookmarks more easily.



```
}
GetSum(int start, int end)
t + end;
}
public string GetMessage()
```

Bookmark	File Location	Line Number
Bookmark1	c:\users\buildfusion\documents\visual studio 2012\Projects\MyFirstApplication\MyFirstApplication\TestClass.cs	21
Bookmark2	c:\users\buildfusion\documents\visual studio 2012\Projects\MyFirstApplication\MyFirstApplication\TestClass.cs	25

The **Bookmark** tool window can be opened using *Ctrl + W, B* or by navigating to **View** | **Other Window** | **Bookmarks**. You can manipulate bookmarks from this window.

The Code Definition window

This is a read-only editor present inside the IDE which displays the definition of types and methods while the user navigates on the code in the editor. As you move the cursor over the IDE or change the selection on Class View, Object Browser, or Call Browser, the content of the **Code Definition** window automatically gets updated with either the actual code from within the application, or it displays metadata content of a selected type.

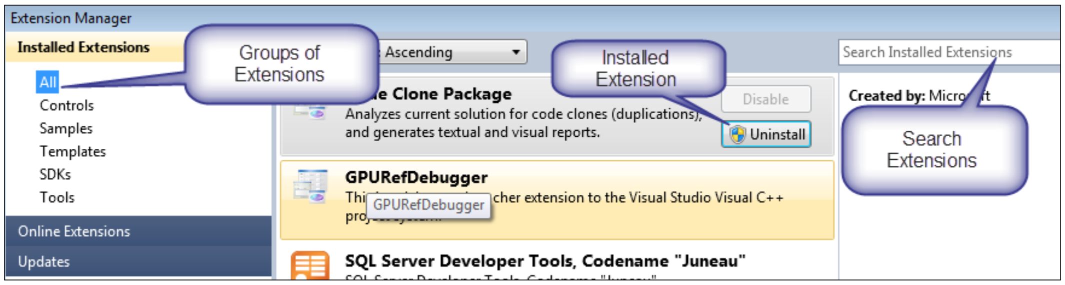
To open the **Code Definition** window, navigate to **View** | **Code Definition** window.

The **Code Definition** window not only displays the definition of the code in the navigation, but it also allows you to copy code, use **Edit Definition** to edit the definition, put breakpoints, and so on. The **Code Definition** window is very useful in certain cases while working with large applications.

Extension Manager

Visual Studio supports extensibility. A large number of Visual Studio IDE components are now extensible. **Extension Manager** is a special section which allows you to view, control, or uninstall any extension associated with Visual Studio.

Navigate to **Tools** | **Extension Manager** to open the window. If you choose **Online Extension** from the left-hand side, it will connect to the online extension gallery. You can select an extension from the list and download the extension and install.



e Clone Package
Analyzes current solution for code clones (duplications), and generates textual and visual reports.

GPURefDebugger
Thi cher extension to the Visual Studio Visual C++ pro

SQL Server Developer Tools, Codename "Juneau"

Once the extension is installed, it will show you the option to either disable or uninstall the extension.

What is MSBuild and how can I use it?

The Microsoft Build Engine is a platform to simplify the build process when there are a large number of files that need to be compiled together. Visual Studio build process uses the MSBuild environment to provide transparent build experience on all the files and folder structures together in one library. The entire build process needs a project file, which is an XML-based file that provides the basic structure of the library.

Visual studio has a project file to maintain all the items that are included in the project and this file is later passed on to the MSBuild interface to invoke build process. In case of advanced scenarios when you don't have Visual Studio available, but you need to build a hierarchy of project structure, MSBuild can be invoked manually too by writing the project file manually.