FREE eBook

LEARNING Visual Studio

Free unaffiliated eBook created from
Stack Overflow contributors.

#visual-studio

Table of Contents

About

You can share this PDF with anyone you feel could benefit from it, downloaded the latest version from: visual-studio

It is an unofficial and free Visual Studio ebook created for educational purposes. All the content is extracted from Stack Overflow Documentation, which is written by many hardworking individuals at Stack Overflow. It is neither affiliated with Stack Overflow nor official Visual Studio.

Use the content presented in this book at your own risk; it is not guaranteed to be correct nor accurate, please send your feedback and corrections to info@zzzprojects.com

Chapter 1: Getting started with Visual Studio

Remarks

Visual Studio is an Integrated Development Environment (IDE) from Microsoft. It enables the developer to work project orientated with various types of projects, including Windows Forms, Console Applications, Office Plug-Ins, and Windows Universal Apps.

The IDE supports various programming languages, the most common being Visual C#, Visual Basic, Visual F#, and Visual C++.

There are several editions of Visual Studio: Community (free), Express (free), Professional, Enterprise, and Ultimate (However, not all are available for all versions).

Versions

Version	Codename	Version number	Supported .NET Framework versions	Date
97	Boston	5.0	N/A	1997-02-01
6.0	Aspen	6.0	N/A	1998-06-01
.NET 2002	Rainier	7.0	1.0	2002-02-13
.NET 2003	Everett	7.1	1.1	2003-04-24
2005	Whidbey	8.0	2.0, 3.0	2005-11-07
2008	Orcas	9.0	2.0, 3.0, 3.5	2007-11-19
2010	Dev10/Rosario	10.0	2.0 - 4.0	2010-04-12
2012	Dev11	11.0	2.0 - 4.5.2	2012-09-12
2013	Dev12	12.0	2.0 - 4.5.2	2013-10-17
2013.1 (Update				2014-01-

Version	Codename	Version number	Supported .NET Framework versions	Date
1)				20
2013.2 (Update 2)				2014-05-12
2013.3 (Update 3)				2014-08-04
2013.4 (Update 4)				2014-11-12
2013.5 (Update 5)				2015-07-20
2015	Dev14	14.0	2.0 - 4.6	2015-07-20
2015.1 (Update 1)				2015-11-30
2015.2 (Update 2)				2016-03-30
2015.3 (Update 3)				2016-06-27
"15" Preview	Dev15	15.0	2.0 - 4.6.2; Core 1.0	2016-03-30
"15" Preview 2				2016-05-10
"15" Preview 3				2016-07-07
"15" Preview 4				2016-08-22
"15" Preview 5				2016-10-05
2017	Dev15	15.0 - 15.2	3.5 - 4.7; Core 1.0 - 1.1	2017-03-07

Examples

Installation or Setup

Visual Studio can be downloaded and installed for free in *Comunity edition* from the Microsoft site and can be also found in different versions. Just click on the Download button and run the executable, then follow the instructions.

Read Getting started with Visual Studio online: https://riptutorial.com/visual-studio/topic/972/getting-started-with-visual-studio

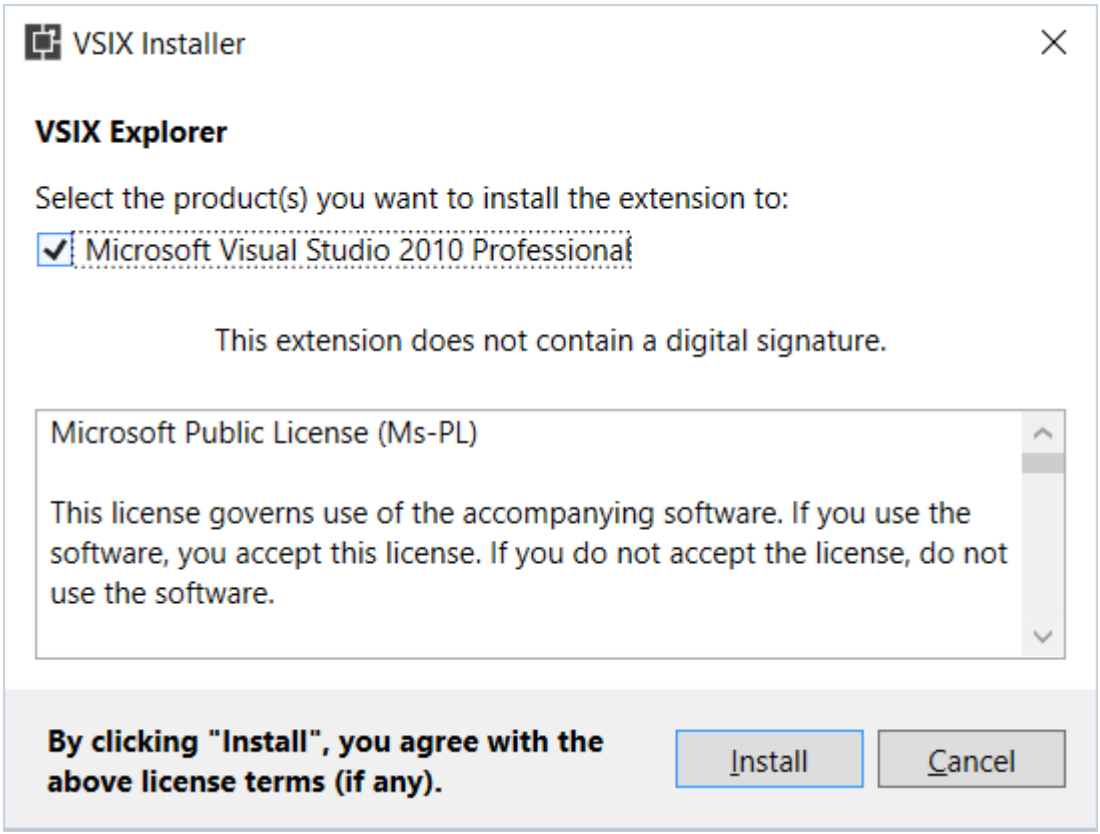
Chapter 2: Adding an extension

Examples

Adding an extension to visual studio using a `VSIX` file

If you have a `vsix` file, you can install it by running the file.

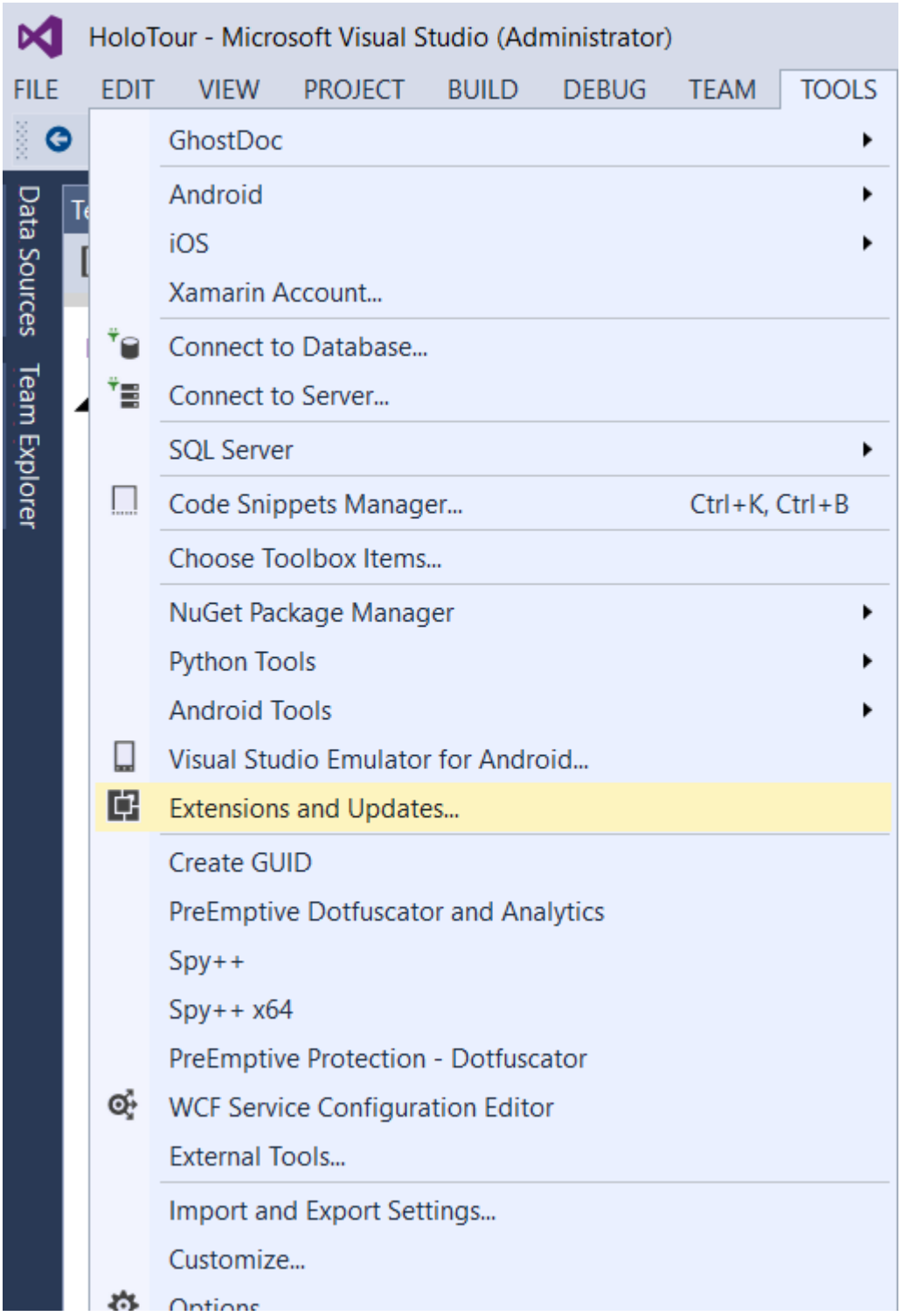
1. Get the `vsix` file (this is the extension installer)
2. Run the file.
3. In the window that opens, confirm the installation.

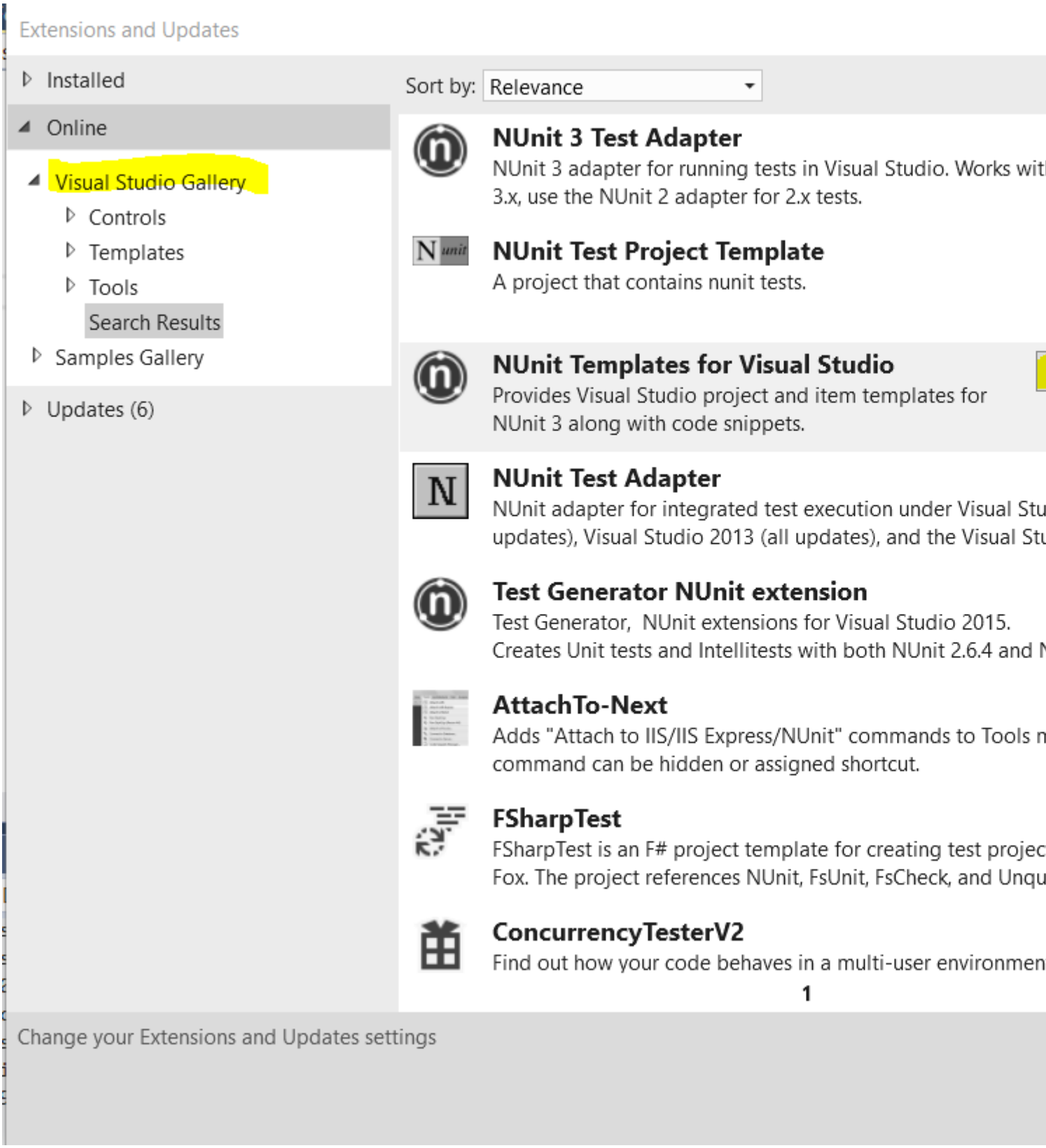


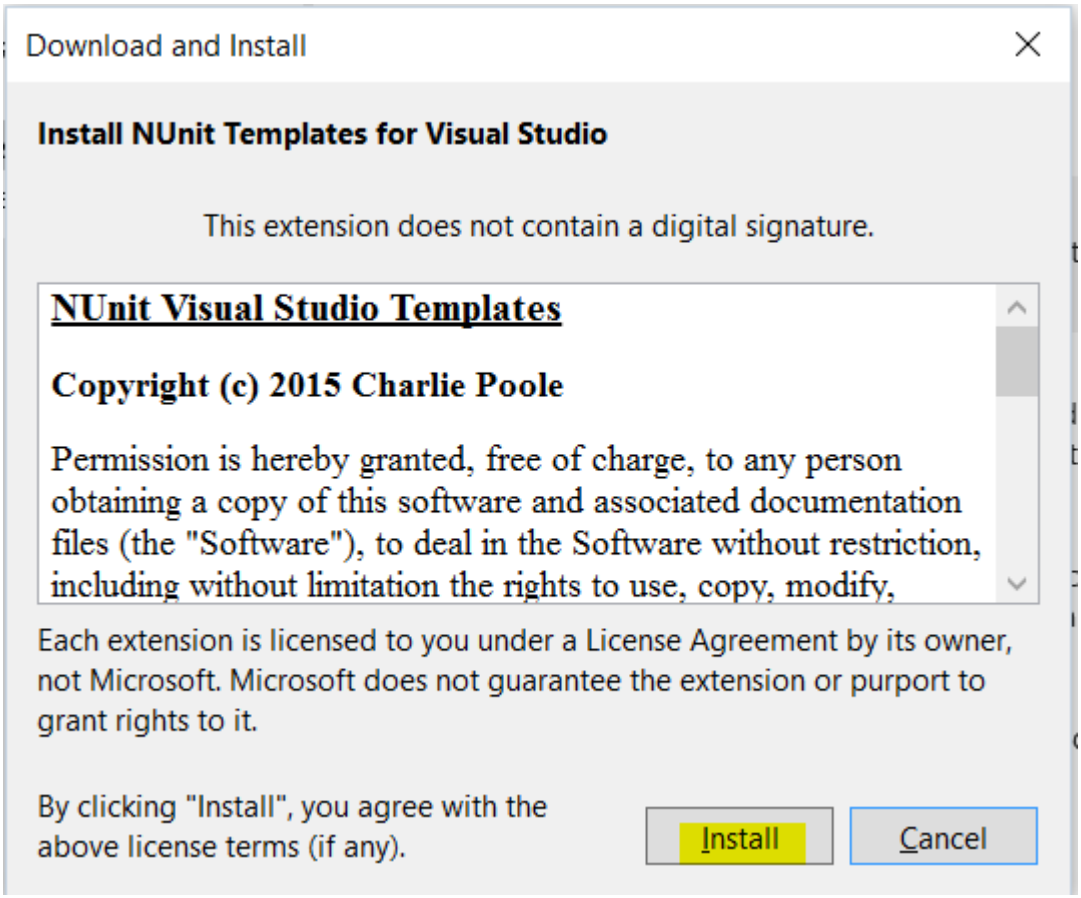
Adding an extension to visual studio from Visual Studio Gallery

In Visual studio

- go to **Tools** > **Extensions and updates...**
- In the window that opens go to online
- Select Visual Studio Gallery
- You can search for an extension on the search box at the upper right corner
- Select the extension you want to add
- Click on download.
- Once download is complete, click on the **Install** button on the window that opened.
- In order to use the extension, you might be requested to restart visual studio


HoloTour - Microsoft Visual Studio (Administrator)
FILE
EDIT
VIEW
PROJECT
BUILD
DEBUG
TEAM
TOOLS
Data Sources
Team Explorer
GhostDoc
Android
iOS
Xamarin Account...
Connect to Database...
Connect to Server...
SQL Server
Code Snippets Manager... Ctrl+K, Ctrl+B
Choose Toolbox Items...
NuGet Package Manager
Python Tools
Android Tools
Visual Studio Emulator for Android...
Extensions and Updates...
Create GUID
PreEmptive Dotfuscator and Analytics
Spy++
Spy++ x64
PreEmptive Protection - Dotfuscator
WCF Service Configuration Editor
External Tools...
Import and Export Settings...
Customize...
Options


Extensions and Updates
Installed
Online
Visual Studio Gallery
Controls
Templates
Tools
Search Results
Samples Gallery
Updates (6)
Sort by: Relevance
NUnit 3 Test Adapter
NUnit 3 adapter for running tests in Visual Studio. Works wit
3.x, use the NUnit 2 adapter for 2.x tests.
NUnit Test Project Template
A project that contains nunit tests.
NUnit Templates for Visual Studio
Provides Visual Studio project and item templates for
NUnit 3 along with code snippets.
NUnit Test Adapter
NUnit adapter for integrated test execution under Visual Stu
updates), Visual Studio 2013 (all updates), and the Visual Stu
Test Generator NUnit extension
Test Generator, NUnit extensions for Visual Studio 2015.
Creates Unit tests and Intellitests with both NUnit 2.6.4 and
AttachTo-Next
Adds "Attach to IIS/IIS Express/NUnit" commands to Tools m
command can be hidden or assigned shortcut.
FSharpTest
FSharpTest is an F# project template for creating test projec
Fox. The project references NUnit, FsUnit, FsCheck, and Unqu
ConcurrencyTesterV2
Find out how your code behaves in a multi-user environmen
1
Change your Extensions and Updates settings



Read Adding an extension online: https://riptutorial.com/visual-studio/topic/2257/adding-an-extension

Chapter 3: Code Contracts

Remarks

In order to fully benefit from Code Contracts you need to install the extension for Visual Studio. There's also a Code Contracts User Manual.

Examples

Standard precondition

```
using System.Diagnostics.Contracts;

public int DivideNumbers(int numerator, int denominator)
{
    Contract.Requires(denominator != 0);

    return numerator / denominator;
}
```

Precondition that throws a specific Exception

```
using System.Diagnostics.Contracts;

public int DivideNumbers(int numerator, int denominator)
{
    Contract.Requires<ArgumentOutOfRangeException>(denominator != 0);

    return numerator / denominator;
}
```

Pre and postconditions

```
using System.Diagnostics.Contracts;

public int IncrementByRandomAmount(int input)
{
    Contract.Requires<ArgumentNullException>(input != null); // Don't allow null parameter.
    Contract.Requires<ArgumentOutOfRangeException>(input < int.MaxValue); // We can't do
anything if we're given int.MaxValue.
    Contract.Ensures(Contract.Result<int>() > input); // Return value will be greater than
input value.

    Random rnd = new Random();
    input += rnd.Next(1, 13); // Creates a number between 1 and 12 and adds it to input.

    return input;
}
```

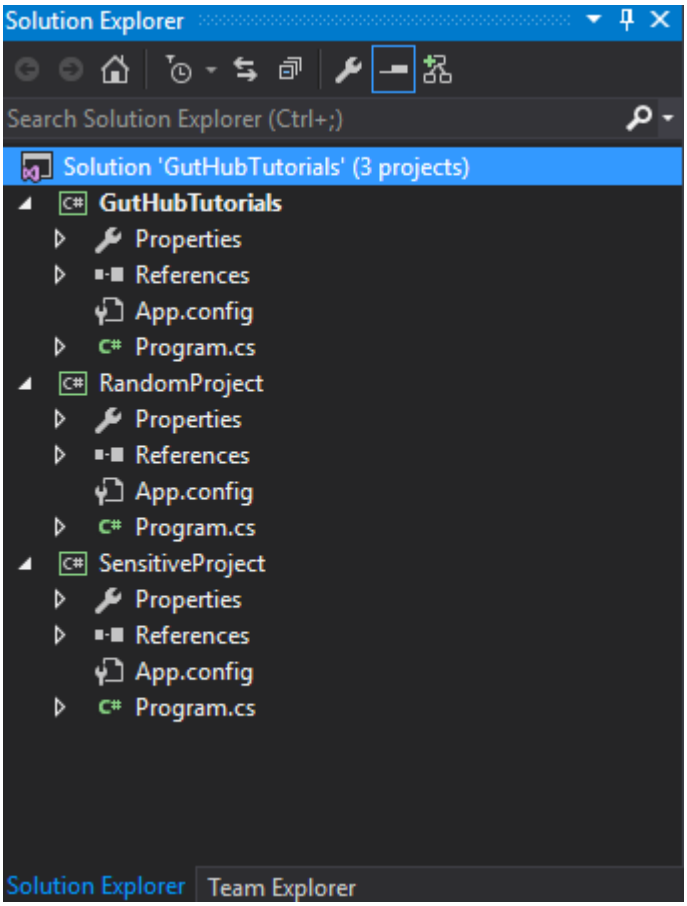
Read Code Contracts online: https://riptutorial.com/visual-studio/topic/6311/code-contracts

Chapter 4: Connecting your visual studio project to Github

Examples

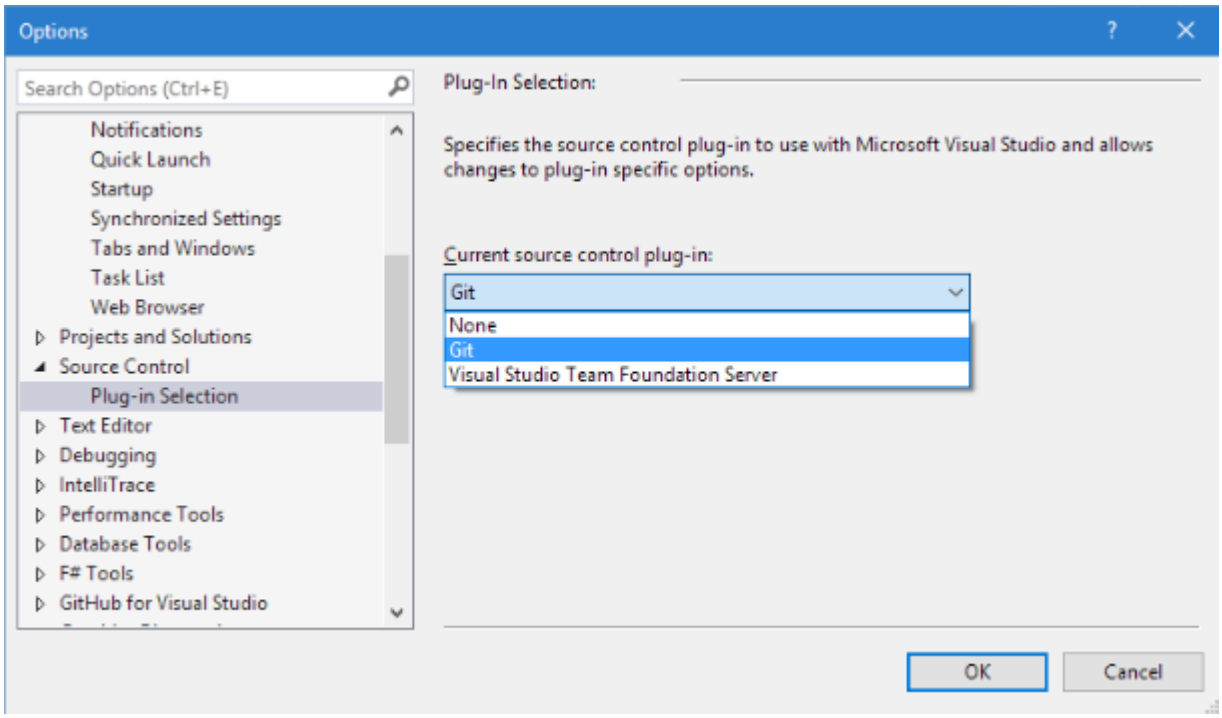
Publishing your project to a github repository removing sensitive data

the steps in this example will use the following project structure as a demonstration

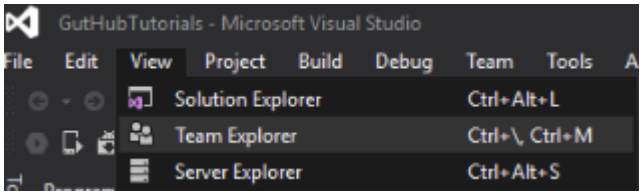


and we intend to export it to the "GHTuts" Repository [Note that the Repo doesn't exist yet on github] but leave the "SensitiveProject" without publish as it contains some passwords, keys, etc..

1. First of all we make sure the source control plug in is set to "Git" in "Tools > Options > Plug-in Selection"



2. If you can't see the "Team Explorer" tab, view it in visual studio like this



---

1. Go to your local solution folder and create a new file called ".gitignore.txt" [Note] this step is only important if you have some sensitive information in your project,otherwise, let visual studio create it for you

2. Now open the ".gitignore.txt" file and paste this in it, this is a template for ignoring common visual studio files (check the links below)

```
## Ignore Visual Studio temporary files, build results, and
## files generated by popular Visual Studio add-ons.

# User-specific files
*.suo
*.user
*.userosscache
*.sln.docstates

# User-specific files (MonoDevelop/Xamarin Studio)
*.userprefs

# Build results
[Dd]ebug/
[Dd]ebugPublic/
[Rr]elease/
[Rr]eleases/
x64/
x86/
bld/
[Bb]in/
```

```
[Oo]bj/
[Ll]og/

# Visual Studio 2015 cache/options directory
.vs/
# Uncomment if you have tasks that create the project's static files in wwwroot
#wwwroot/

# MSTest test Results
[Tt]est[Rr]esult*/
[Bb]uild[Ll]og.*

# NUNIT
*.VisualState.xml
TestResult.xml

# Build Results of an ATL Project
[Dd]ebugPS/
[Rr]eleasePS/
dlldata.c

# DNX
project.lock.json
project.fragment.lock.json
artifacts/

*_i.c
*_p.c
*_i.h
*.ilk
*.meta
*.obj
*.pch
*.pdb
*.pgc
*.pgd
*.rsp
*.sbr
*.tlb
*.tli
*.tlh
*.tmp
*.tmp_proj
*.log
*.vspscc
*.vssscc
.builds
*.pidb
*.svclog
*.scc

# Chutzpah Test files
_Chutzpah*

# Visual C++ cache files
ipch/
*.aps
*.ncb
*.opendb
*.opensdf
*.sdf
```

```
*.cachefile
*.VC.db
*.VC.VC.opendb

# Visual Studio profiler
*.psess
*.vsp
*.vspx
*.sap

# TFS 2012 Local Workspace
$tf/

# Guidance Automation Toolkit
*.gpState

# ReSharper is a .NET coding add-in
_ReSharper*/
*.[Rr]e[Ss]harper
*.DotSettings.user

# JustCode is a .NET coding add-in
.JustCode

# TeamCity is a build add-in
_TeamCity*

# DotCover is a Code Coverage Tool
*.dotCover

# NCrunch
_NCrunch_*
.*crunch*.local.xml
nCrunchTemp_*

# MightyMoose
*.mm.*
AutoTest.Net/

# Web workbench (sass)
.sass-cache/

# Installshield output folder
[Ee]xpress/

# DocProject is a documentation generator add-in
DocProject/buildhelp/
DocProject/Help/*.HxT
DocProject/Help/*.HxC
DocProject/Help/*.hhc
DocProject/Help/*.hhk
DocProject/Help/*.hhp
DocProject/Help/Html2
DocProject/Help/html

# Click-Once directory
publish/

# Publish Web Output
*.[Pp]ublish.xml
*.azurePubxml
```

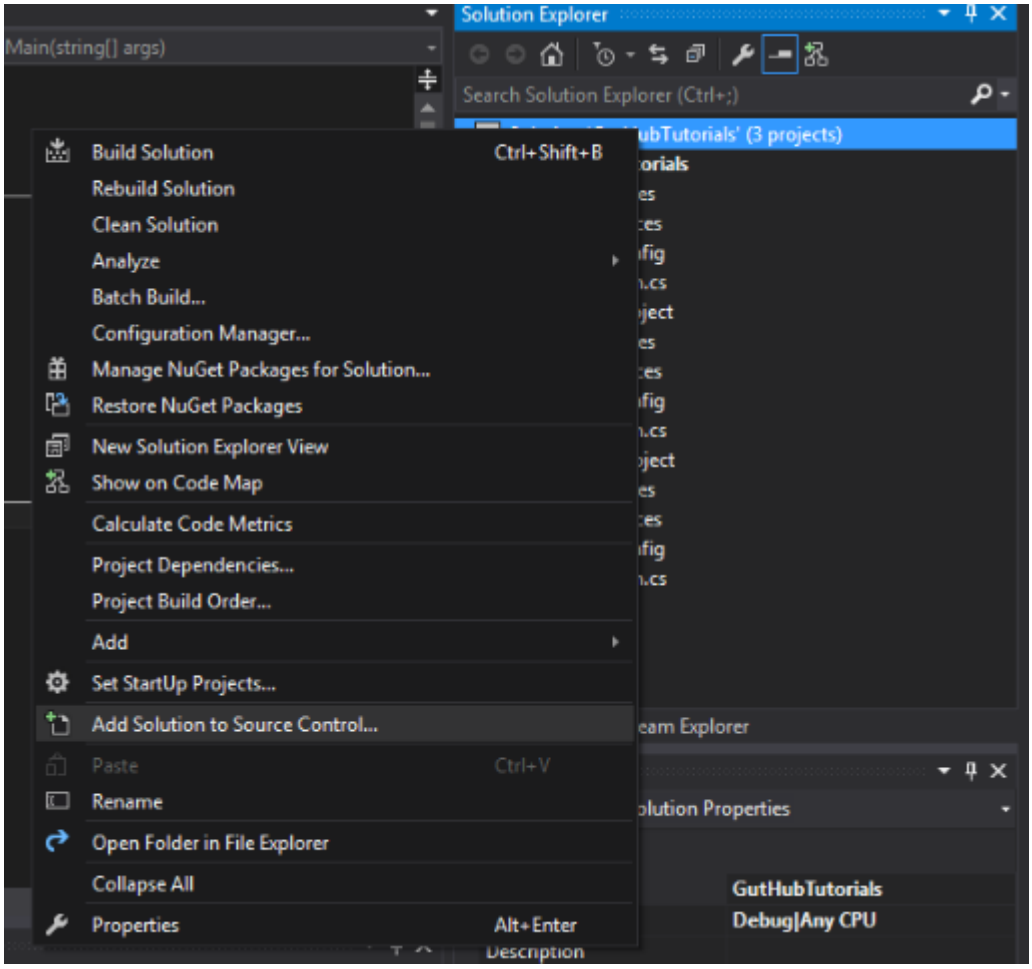
```
# TODO: Comment the next line if you want to checkin your web deploy settings
# but database connection strings (with potential passwords) will be unencrypted
*.pubxml
*.publishproj

# Microsoft Azure Web App publish settings. Comment the next line if you want to
# checkin your Azure Web App publish settings, but sensitive information contained
# in these scripts will be unencrypted
PublishScripts/

# NuGet Packages
*.nupkg
# The packages folder can be ignored because of Package Restore
**/packages/*
# except build/, which is used as an MSBuild target.
!**/packages/build/
# Uncomment if necessary however generally it will be regenerated when needed
#!**/packages/repositories.config
# NuGet v3's project.json files produces more ignoreable files
*.nuget.props
*.nuget.targets

# Microsoft Azure Build Output
csx/
*.build.csdef

# Microsoft Azure Emulator
ecf/
rcf/

# Windows Store app package directories and files
AppPackages/
BundleArtifacts/
Package.StoreAssociation.xml
_pkginfo.txt

# Visual Studio cache files
# files ending in .cache can be ignored
*.[Cc]ache
# but keep track of directories ending in .cache
!*.[Cc]ache/

# Others
ClientBin/
~$*
*~
*.dbmdl
*.dbproj.schemaview
*.pfx
*.publishsettings
node_modules/
orleans.codegen.cs

# Since there are multiple workflows, uncomment next line to ignore bower_components
# (https://github.com/github/gitignore/pull/1529#issuecomment-104372622)
#bower_components/

# RIA/Silverlight projects
Generated_Code/

# Backup & report files from converting an old project file
```

```
# to a newer Visual Studio version. Backup files are not needed,
# because we have git ;-)
_UpgradeReport_Files/
Backup*/
UpgradeLog*.XML
UpgradeLog*.htm

# SQL Server files
*.mdf
*.ldf

# Business Intelligence projects
*.rdl.data
*.bim.layout
*.bim_*.settings

# Microsoft Fakes
FakesAssemblies/

# GhostDoc plugin setting file
*.GhostDoc.xml

# Node.js Tools for Visual Studio
.ntvs_analysis.dat

# Visual Studio 6 build log
*.plg

# Visual Studio 6 workspace options file
*.opt

# Visual Studio LightSwitch build output
**/*.HTMLClient/GeneratedArtifacts
**/*.DesktopClient/GeneratedArtifacts
**/*.DesktopClient/ModelManifest.xml
**/*.Server/GeneratedArtifacts
**/*.Server/ModelManifest.xml
_Pvt_Extensions

# Paket dependency manager
.paket/paket.exe
paket-files/

# FAKE - F# Make
.fake/

# JetBrains Rider
.idea/
*.sln.iml
```

3. Now add your sensitive project folder to the ".gitignore.txt" file at any line that doesn't contain `#`, so just add it at the very end, and it should look something like this

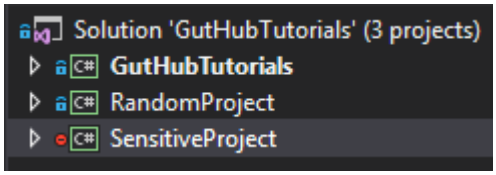
```
# Paket dependency manager
.paket/paket.exe

# FAKE - F# Make
.fake/

# User-Ignored projects
SensitiveProject/
```

4. Right click on the solution and choose "Add Solution to Source Control..."



[Note] it might ask you to save the solution before you continue

5. Now you have a "LOCAL" git Repo on your pc , which VS will read from,but without a github Repo, and you will see a small blue lock icon next to each file in the solution that was added to git and a red circle at the ignored project

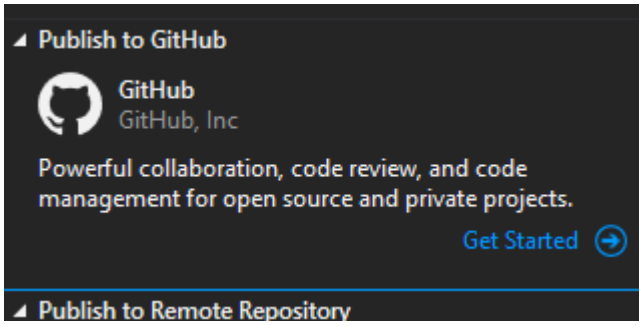


[Note]
for more information about .gitignore file, check these links

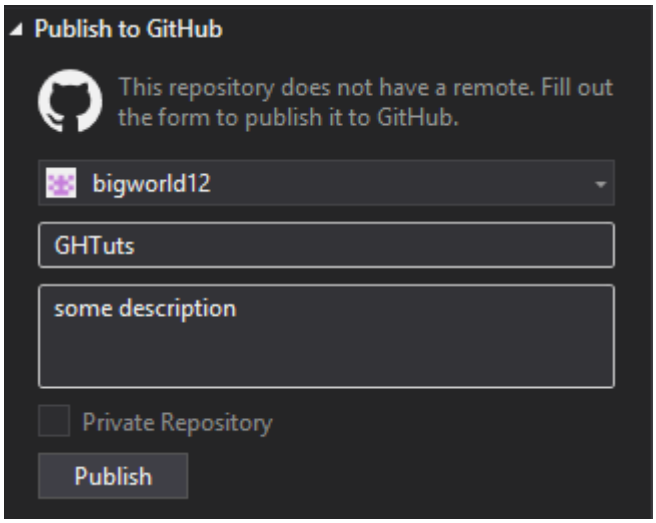
- https://help.github.com/articles/ignoring-files/
- https://github.com/github/gitignore

7. Go to the "Team Explorer" tab and then "Sync"

8. Now we create a repo from vs to github like this, press the "Get Started" button



9. Now fill in your information in github for the new Repo, then click "Publish"



10. Now when we go to github we see our local repo got published to github without our sensitive project [Note]
    the url of the repo will look something like this
    `https://github.com/<user name>/<repo name>`

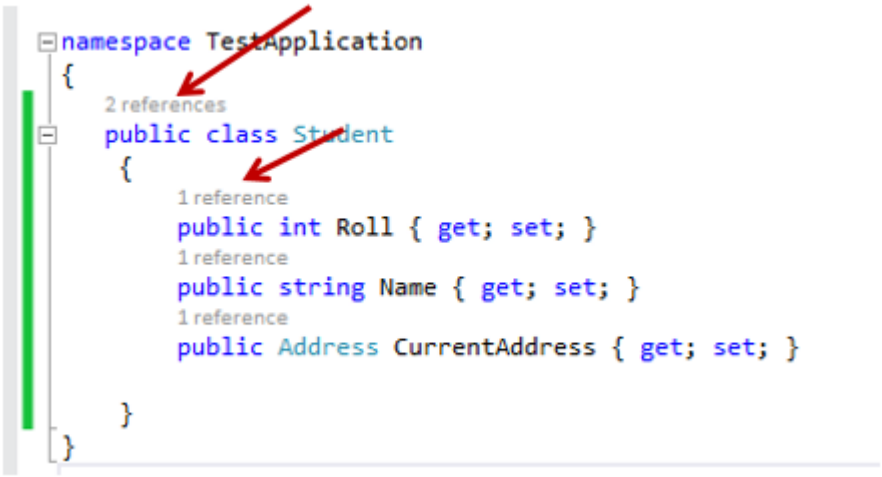
Read Connecting your visual studio project to Github online: https://riptutorial.com/visual-studio/topic/3826/connecting-your-visual-studio-project-to-github

Chapter 5: Visual Studio tools

Examples

Code Lens

Code lens is a simple way to know what happens with the code. Here you could find an image with the number of references of a method or class.



If you can't see the code lens please see this question: Missing CodeLens references count in VS 2015 Community edition

Snippets

Intoduction

Since Visual Studio 2005 can you make Intellisense Code Snippets. This allow you to generate some code just by typing one keyword and press two times the `tab` key.

Using the code

The XML code you need for make an Intellisense Code Snippet stands below:

```
<?xml version="1.0" encoding="utf-8"?>

<CodeSnippets xmlns="http://schemas.microsoft.com/VisualStudio/CodeSnippet">
  <CodeSnippet Format="1.0.0"> <!-- format attribute is required -->

    <Header> <!-- 1 -->
      <Title></Title>
      <Author></Author>
      <Shortcut></Shortcut>
```

```
      <Description></Description>
      <Keywords>
        <Keyword>abc<Keyword>
        <Keyword>def<Keyword>
      </keywords>
    </Header>

    <Snippet> <!-- 2 -->

      <Imports> <!-- 2.1 -->
        <Import>
          <Namespace>System</Namespace>
        </Import>
      </Imports>

      <Declarations> <!-- 2.2 -->
        <Literal Editable="true/false"> <!-- 2.2.1 -->
          <ID>example</ID>
          <Type>System.String</Type>
          <ToolTip>A tip you can show</ToolTip>
          <Default>default value</Default>
          <Function></Function> <!-- 2.2.2 -->
        </Literal>

        <Object> <!-- 2.2.1 -->
          <ID>example</ID>
          <Type>System.String</Type>
          <ToolTip>A tip you can show</ToolTip>
          <Default>default value</Default>
          <Function></Function> <!-- 2.2.2 -->
        </Object>
      </Declarations>

      <References> <!-- 2.3 -->
        <Reference>
          <Assembly>System.Data.dll</Assembly>
        </Reference>
      </References>

      <Code Language=""> <!-- 2.4 -->
        <![CDATA[
            <!-- your code here if you use literals use dollar chars  -->
        ]]>
      </Code>

    </Snippet>

  </CodeSnippet>
</CodeSnippets>
```

In the snippet tag, you have two required tags named Header and Snippet. You can find more information in next headings. The number near the name are correspondents with the numbers in the code above.

There can be zero or more CodeSnippet elements added into the CodeSnippets element.

1. Header

In the Header-tag, you can place some specific information about the snippet and what he does. The important tags you can use inside this tag are:

Element	Description
Title	The title of the snippet. This attribute is required.
Author	The author of the snippet.
Shortcut	Is the shortcut, you can use for generating the code. Note that this can only contain letters and numbers and must begin with a letter. **Note:** Remember also to give the snippet a good and unique name and shortcut. Otherwise, it will give problems when you import the snippet into Visual Studio.
Description	Gives more information about the snippet if you need that.
HelpUrl	A url for a help page on the internet.
Keywords	Groups one or more keyword elements.
SnippetTypes	Groups `SnippetType` elements. This element contain a text value and must be one of the following values. Snippet types are merged with a forward slash. • `SurroundsWith`: Allows the code snippet to be placed around a selected piece of code. • `Expansion`: Allows the code snippet to be inserted at the cursor. • `Refactoring`: Specifies that the code snippet is used during Visual C# refactoring. Refactoring cannot be used in custom code snippets. Source list: msdn.microsoft.com

Source table (but edits): msdn.microsoft.com

2. Snippet

In the snippet tag, you can use three different tags. This can be:

- Imports
- Declarations
- Code (required)
- References

These are explained below.

2.1 Imports

`Imports` contain the needed namespaces you need for the code. Use the import-tag inside this tag and here you can place the needed namespaces each with the `Namespace`-tag.

2.2 Declarations

`Declarations` can be used for declaring some literals or objects into your code in the `Code`-tag. The children are literals and objects.

2.2.1 Literals and objects

Literals and objects define the literals and objects of the code snippet that you can edit. Functionality are literals and objects are the same, but it has an additional type constraint.

The Literal and object-tag can contain next children:

- **`ID`:** The ID of the literal (required)
- **`Type`:** The type of that object including namespace and class (required by objects)
- **`ToolTip`:** Gives a tip
- **`Default`:** A default value of that object (required)
- **`Functions`**

In the snippets, there are some predefined literals. They are listed below:

Literal	Details
`$end$`	Marks the location to place the cursor after the code snippet is inserted.

`$selected$`	Represents text selected in the document that is to be inserted into the snippet when it is invoked. Example, If you have:
	`A $selected$ is an object that I like.`
	and the word was car selected when you invoked the template, you would get:
	`A car is an object that I like.`

2.2.2 Functions

Functions in the Literal- or Object-tag means that you can use a function for generating code depending on another element. There are three functions that I know:

Function	Description	Language
`GenerateSwitchCases (EnumerationLiteral)`	Generates a switch statement and a set of case statements for the members of the enumeration specified by the EnumerationLiteral parameter. The EnumerationLiteral parameter must be either	Visual C# and Visual J#[1]

Function	Description	Language
	a reference to an enumeration literal or an enumeration type.	
`ClassName()`	Returns the name of the class that contains the inserted snippet.	Visual C# and Visual J#[1]
`SimpleTypeName(TypeName)`	Reduces the TypeName parameter to its simplest form in the context in which the snippet was invoked.	Visual C#

[1] only available in Visual Studio 2005.

Source table: msdn.microsoft.com

Attributes for the Literal and Object Elements

The Literal and Object tags can have some optional attributes.

Attribute	Description	Type
Editable	Specifies whether or not you can edit the literal after the code snippet is inserted. The default value of this attribute is true.	Boolean

Source table: msdn.microsoft.com

2.3 References

Groups reference elements that contains information about assembly references for the code snippet. This can contain next elements:

- **Assembly:** Contains the name of the assembly by the code snippet (required)
- **Url:** Contains a website that gives more information about the assembly

2.4 Code

Code is the code you will generate between `<![CDATA[` and `]]>`. Place the `ID` of your literal between dollar chars and Visual Studio will ask you for change these default value if the declarations are filled in. Here, you've an example for C# and VB for the shortcut propfull.

```
<!-- ... Other code ... -->
<Declarations>
  <Literal>
    <Id>variablename</Id>
    <Default>_myproperty</Default>
  </Literal>
```

```
  <Literal>
    <Id>propertytype</Id>
    <Default>int</Default>
  </Literal>

  <Literal>
    <Id>propertyname</Id>
    <Default>myproperty</Default>
  </Literal>
</Declarations>

<Code Language="CSharp">
  <![CDATA[
    private $propertyvalue$ $variablename$;

    public $propertyvalue$ $propertyname$
    {
        get { return $variablename$; }
        set { $Variablename$ = Value; }
    }
  ]]>
</Code>

<!-- ... Other code ... -->

<Declarations>
  <Literal>
    <Id>variablename</Id>
    <Default>_myproperty</Default>
  </Literal>

  <Literal>
    <Id>propertytype</Id>
    <Default>int</Default>
  </Literal>

  <Literal>
    <Id>propertyname</Id>
    <Default>myproperty</Default>
  </Literal>
</Declarations>

<Code Language="VB">
  <![CDATA[
    Private $variablename$ As $propertyvalue$

    Public Property $propertyname$ As $propertyvalue$
        Get
            Return $variablename$
        End Get

        Set (ByVal value As $propertyvalue$)
            $variablename$ = value
        End Set
    End Property
  ]]>
</Code>

<!-- ... Other code ... -->
```

In the required Language attribute, you can define your language where you are making the

snippet. You can find the languages you can use in the next table.

Language	Keyword	Available in next versions
Visual C#	CSharp	2005, 2010, 2012 and later
Visual Basic	VB	2005, 2010, 2012 and later
XML	XML	2005, 2010, 2012 and later
Visual J#	VJSharp	2005, 2012 and later
C++	CPP	2012 and later
JavaScript	JavaScript	2012 and later
JScript	JScript	2012 and later
SQL	SQL	2012 and later
HTML	HTML	2012 and later
CSS	CSS	2012 and later
XAML	XAML	2012 and later

Other optional attributes are:

Attribute	Description
Delimiter	Specifies the delimiter used to describe literals and objects in the code. By default, the delimiter is `$`.
Kind	Specifies the kind of code that the snippet contains and, therefore, the location at which a code snippet must be inserted for the code snippet to compile.

The valid values for the kind variable are:

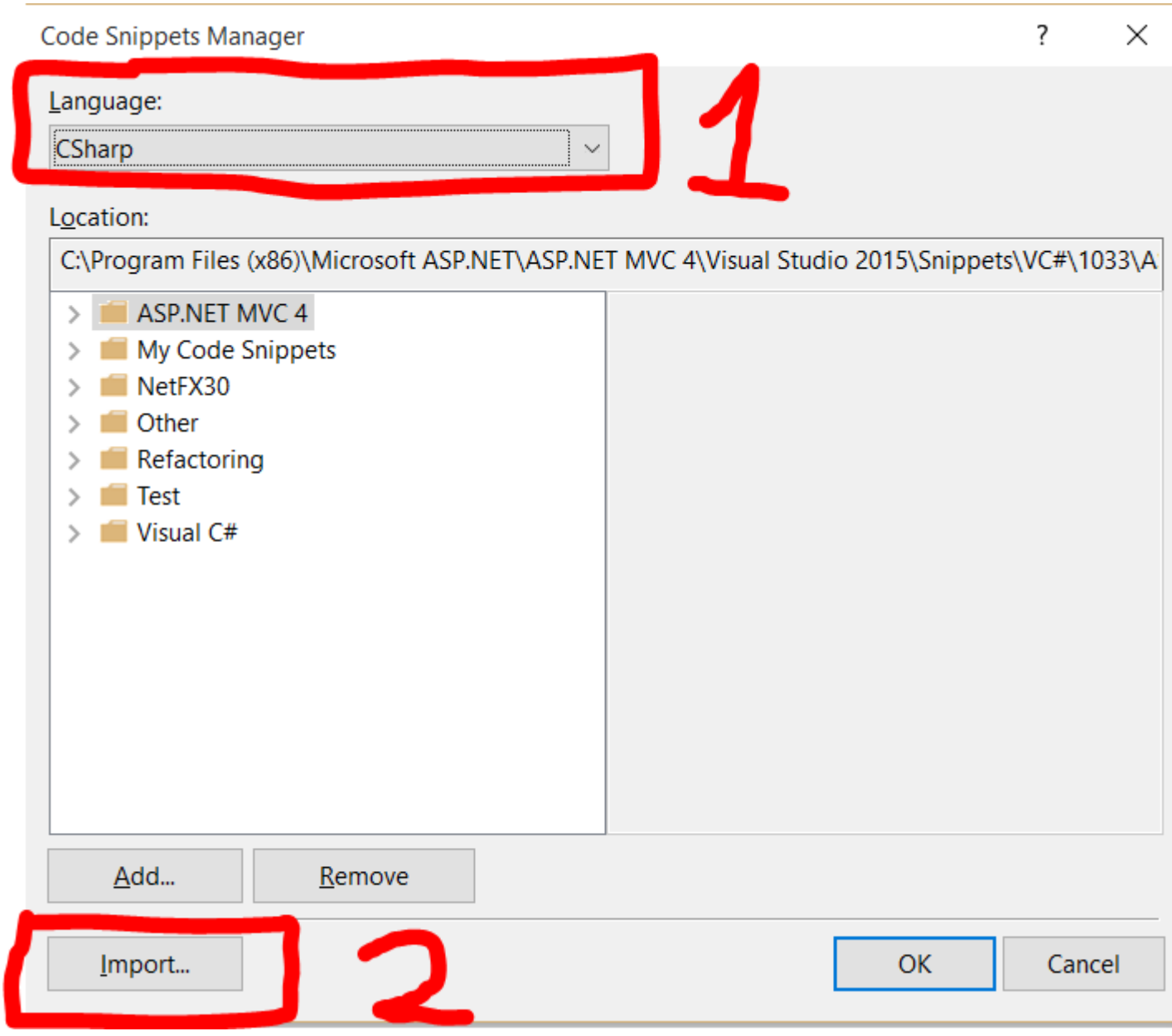
Value	Description
method body	Specifies that the code snippet is a method body, and therefore, must be inserted inside a method declaration.
method decl	Specifies that the code snippet is a method, and therefore, must be inserted inside a class or module.
type decl	Specifies that the code snippet is a type, and therefore, must be inserted inside a class, module, or namespace.

Value	Description
file	Specifies that the snippet is a full code file. These code snippets can be inserted alone into a code file, or inside a namespace.
any	Specifies that the snippet can be inserted anywhere. This tag is used for code snippets that are context-independent, such as comments.

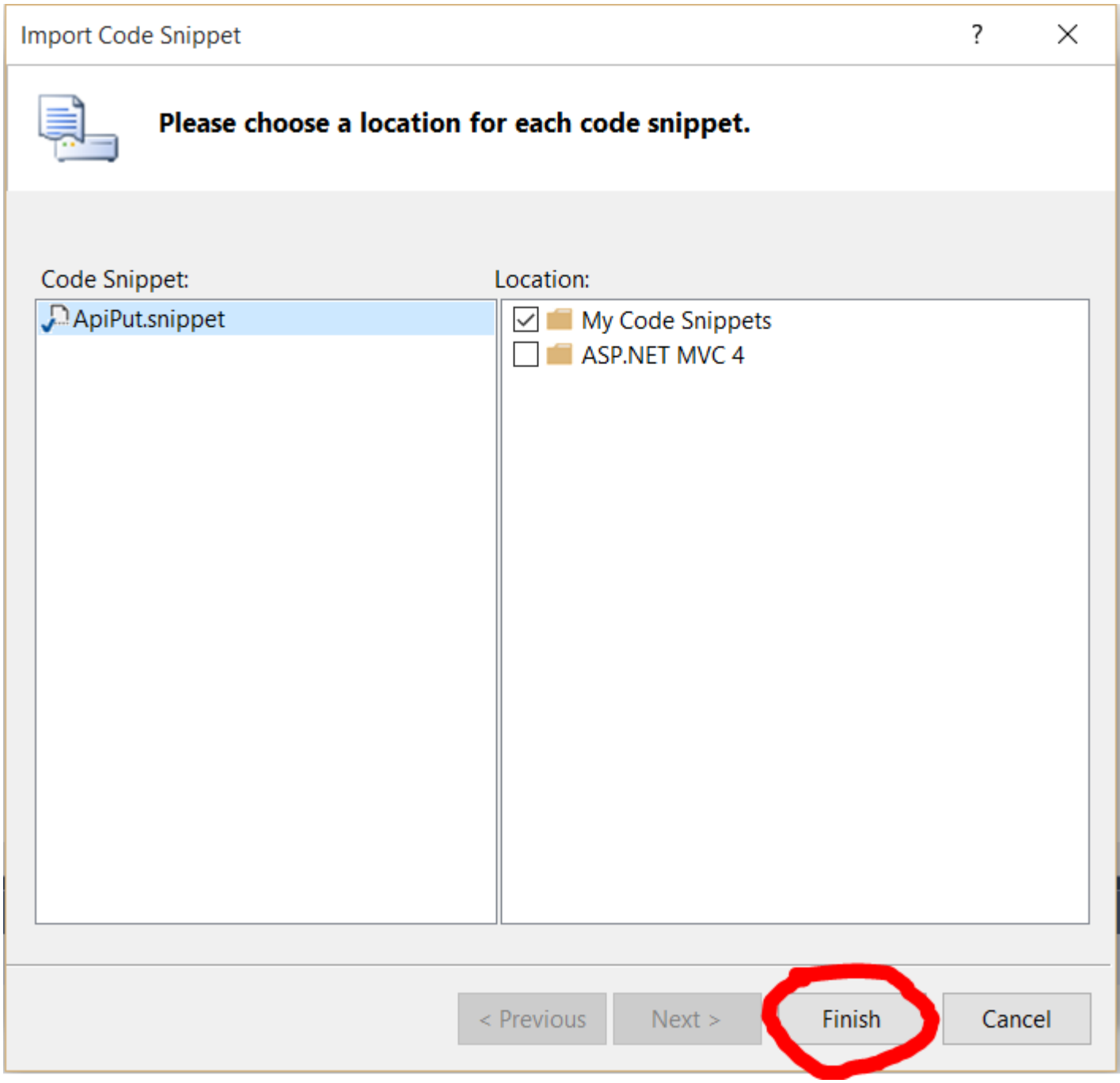
Source tables: msdn.microsoft.com

Import Snippet into Visual Studio

1. Save the XML code and give it the extension `.snippet`.
2. You can add the new made snippet into Visual Studio by pressing `Control` + `K`, `Control` + `B` or go to *"Tools"* → *"Code Snippets Manager..."*. This open next window:



3. Choose the language into the combo box for which language you've made the snippet. click on *"Import..."* and choose the file you've made.



4. Click on `"Finish"`. If the file name already has been used, Visual Studio go ask to override the existing file. You've three options:

    - **Overwrite:** Overwrites the file. You can use this option if you will edit an old snippet.
    - **Rename:** Goes to rename the file to an unique name.
    - **Skip:** Cancels the import. Renames the file to a unique name.

You could also add a new location with all the snippets you've made by clicking on the *"Add..."* button on the first window and select the folder in the *"select folder window"*. The advantage is now when a new valid snippet is added in that folder, you can use this directly in Visual Studio.

Note: Test after importing your snippet for errors, so you don't have any problems when you use

the snippet. You can always remove or overwrite the snippet if there is an error.

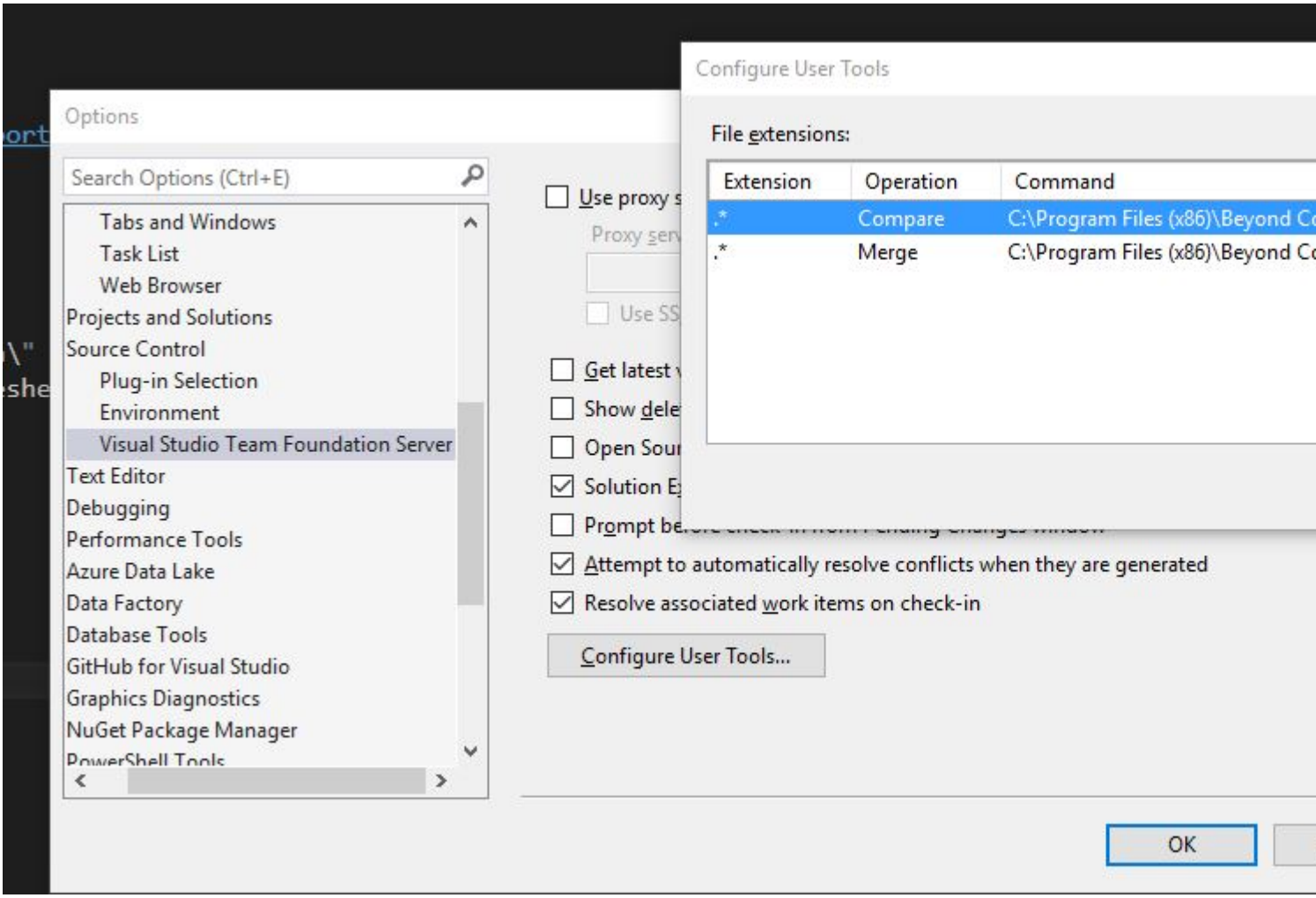
Point of intrest

You can also see the documentation on MSDN for more information.

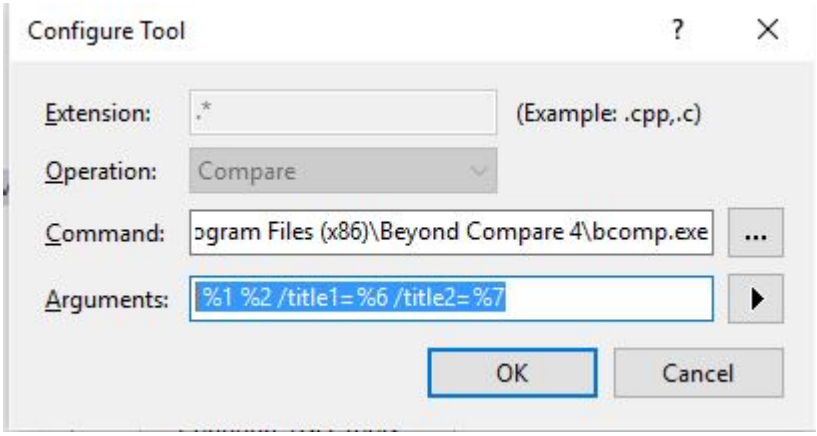
Override merge/compare tools

Got to Tools | Options | Source Control | Visual Studio Team Foundation Server

click on the Configure User Tools:



You can add separate overrides for 'Compare' and 'Merge' operations. Click on Add and select the operation you want to override. You'd need to type the path to the tool you use, and the exact arguments your tool expects. For example to use BeyondCompare, add the following Arguments " %1 %2 /title1=%6 /title2=%7":



To Merge with BeyondCompare use the Arguments "%1 %2 %3 %4 /title1=%6 /title2=%7 /title3=%8 /title4=%9"

In a 2006 blog post MS employee James Manning surveyed the arguments as expected by various tools: WinDiff, DiffDoc, WinMerge, Beyond Compare, KDiff3, Araxis, Compare It!, SourceGear DiffMerge, TortoiseMerge and Visual SlickEdit. The post is a good starting point, but be sure to check the up to date documentation of your tool.

It is highly recommended *not* to use for merge tools that are incapable of 3-way merges (e.g., WinMerge 2.x).

Entity Framework

> Entity Framework (EF) is an object-relational mapper that enables .NET developers to work with relational data using domain-specific objects. It eliminates the need for most of the data-access code that developers usually need to write.
>
> Entity Framework allows you to create a model by writing code or using boxes and lines in the EF Designer. Both of these approaches can be used to target an existing database or create a new database.

Source and more information: Entity Framework documentation

Read Visual Studio tools online: https://riptutorial.com/visual-studio/topic/2398/visual-studio-tools

Credits

S. No	Chapters	Contributors
1	Getting started with Visual Studio	Almir Vuk, Avi Turner, Cody Gray, Community, Derpcode, Fruchtzwerg, H. Pauwelyn, meJustAndrew, Misaz, Nikita, Richard Banks
2	Adding an extension	Avi Turner
3	Code Contracts	Disk Crasher
4	Connecting your visual studio project to Github	bigworld12
5	Visual Studio tools	dove, H. Pauwelyn, Ofek Shilon