# VISUAL STUDIO 2019

SUCCINCTLY

*BY* ALESSANDRO DEL SOLE

SUCCINCTLY EBOOK SERIES

Syncfusion®

# Visual Studio 2019 Succinctly

**By**

**Alessandro Del Sole**

Foreword by Daniel Jebaraj

2501 Aerial Center Parkway
Suite 200
Morrisville, NC 27560
USA


**Important licensing information. Please read.**



**Technical Reviewer:** James McCaffrey
**Copy Editor:** Courtney Wright
**Acquisitions Coordinator:** Tres Watkins, content development manager, Syncfusion, Inc.
**Proofreader:** Jacqueline Bieringer, content producer, Syncfusion, Inc.

# Table of Contents

**The Story Behind the *Succinctly* Series of Books** ..... 8

**About the Author** ..... 10

**Introduction** ..... 11

**Chapter 1 The Installation Experience** ..... 12

Installing Visual Studio 2019 ..... 12

Customizing the installation ..... 15

Installing individual components ..... 16

Installing language packs ..... 16

Customizing the installation folders ..... 17

Managing multiple versions and editions ..... 18

Creating an offline installer ..... 19

Trying out preview features ..... 20

Starting Visual Studio 2019 ..... 20

Controlling Visual Studio updates ..... 20

Chapter summary ..... 21

**Chapter 2 Layout and Project Management** ..... 22

The IDE startup ..... 22

Introducing the start window ..... 22

Working with projects and solutions ..... 23

Creating new projects ..... 23

Cloning existing code repositories ..... 25

Filtering solutions ..... 28

Graphical enhancements ..... 29

The Blue theme ..... 30

Improved multi-monitor support ..... 31

Layout and tool changes ....32
Search box and solution name....33
The Notifications hub ....33
Reporting feedback....35
The Extensions menu ....36
Chapter summary....36
**Chapter 3 Code Editor Productivity....37**
Document Health Indicator and Code Cleanup....37
Clipboard Ring....40
Classification colors....40
Quick actions and refactorings for C# and Visual Basic....41
Converting foreach loops to LINQ or lambda expressions ....41
Promoting class members to interfaces and base types ....43
Synchronizing namespace and folder names....44
Converting anonymous types to tuples ....45
IntelliCode: AI-powered word completion....46
Chapter summary....49
**Chapter 4 Collaboration ....50**
Git tooling updates ....50
Support for stashing changes ....50
Support for pull requests....51
Collaboration sessions with VS Live Share....57
Live Share configuration ....61
Chapter summary....62
**Chapter 5 What's New in the Debugger....63**
Searching local variables within debug windows ....63

Introducing data breakpoints ....64
Formatting locals with format specifiers ....65
DataSet visualizer for .NET Core ....65
Feature preview: the Time Travel Debugger ....67
Chapter summary ....67
**Chapter 6 What's New for Mobile Development ....68**
Creating mobile projects with Xamarin ....68
Platform support and code-sharing strategies ....69
Shell template for Xamarin.Forms ....70
Xamarin.Forms improvements ....72
Xamarin.Forms Previewer updates ....72
Managing objects with the Properties window ....74
Xamarin.Android improvements ....76
Chapter summary ....76
**Chapter 7 What's New for Web and Cloud Development ....77**
Creating web projects ....77
Creating ASP.NET Core projects ....78
Productivity improvements for .NET Core ....78
Supporting .NET Core 3.0 ....79
CodeLens and Find All References for Razor files ....79
Calculating code metrics ....80
Analyzing CPU performance ....82
Publish experience enhancements ....84
Chapter summary ....86
**Chapter 8 Desktop Development with .NET Core 3.0 ....87**
.NET Core 3.0 for desktop development ....87

Enabling .NET Core 3.0 previews ..... 87
Creating WPF applications ..... 88
Hints about publishing .NET Core desktop apps ..... 90
Creating Windows Forms applications ..... 90
Chapter summary ..... 91

# The Story Behind the *Succinctly* Series of Books

Daniel Jebaraj, Vice President
Syncfusion, Inc.

## Staying on the cutting edge

As many of you may know, Syncfusion is a provider of software components for the Microsoft platform. This puts us in the exciting but challenging position of always being on the cutting edge.

Whenever platforms or tools are shipping out of Microsoft, which seems to be about every other week these days, we have to educate ourselves, quickly.

## Information is plentiful but harder to digest

In reality, this translates into a lot of book orders, blog searches, and Twitter scans.

While more information is becoming available on the Internet and more and more books are being published, even on topics that are relatively new, one aspect that continues to inhibit us is the inability to find concise technology overview books.

We are usually faced with two options: read several 500+ page books or scour the web for relevant blog posts and other articles. Just as everyone else who has a job to do and customers to serve, we find this quite frustrating.

## The *Succinctly* series

This frustration translated into a deep desire to produce a series of concise technical books that would be targeted at developers working on the Microsoft platform.

We firmly believe, given the background knowledge such developers have, that most topics can be translated into books that are between 50 and 100 pages.

This is exactly what we resolved to accomplish with the *Succinctly* series. Isn't everything wonderful born out of a deep desire to change things for the better?

## The best authors, the best content

Each author was carefully chosen from a pool of talented experts who shared our vision. The book you now hold in your hands, and the others available in this series, are a result of the authors' tireless work. You will find original content that is guaranteed to get you up and running in about the time it takes to drink a few cups of coffee.

## Free forever

Syncfusion will be working to produce books on several topics. The books will always be free. Any updates we publish will also be free.

## Free? What is the catch?

There is no catch here. Syncfusion has a vested interest in this effort.

As a component vendor, our unique claim has always been that we offer deeper and broader frameworks than anyone else on the market. Developer education greatly helps us market and sell against competing vendors who promise to "enable AJAX support with one click," or "turn the moon to cheese!"

## Let us know what you think

If you have any topics of interest, thoughts, or feedback, please feel free to send them to us at succinctly-series@syncfusion.com.

We sincerely hope you enjoy reading this book and that it helps you better understand the topic of study. Thank you for reading.

Please follow us on Twitter and "Like" us on Facebook to help us spread the word about the *Succinctly* series!

# About the Author

Alessandro Del Sole has been a Microsoft Most Valuable Professional (MVP) since 2008. Awarded MVP of the Year in 2009, 2010, 2011, 2012, and 2014, he is internationally considered a Visual Studio expert and a Visual Basic and .NET authority.

Alessandro has authored many printed books and e-books on programming with Visual Studio, including *Visual Studio 2017 Succinctly*, *Visual Basic 2015 Unleashed*, *Beginning Visual Studio for Mac*, and *Visual Studio Code Succinctly*.

He has written tons of technical articles about .NET, Visual Studio, and other Microsoft technologies in Italian and English for many developer portals, including MSDN Magazine from Microsoft. He has spoken at many Italian conferences and has released a number of mobile apps. He has also produced many instructional videos in both English and Italian.

Alessandro works as a senior software engineer, focusing on mobile apps in the healthcare market. You can follow him on Twitter at @progalex.

*This book is dedicated to Angelica, the love of my life.*

# Introduction

Visual Studio 2019 is the new release of Microsoft's premiere development environment, which allows for building applications for the desktop, the web, the cloud, and mobile devices. It follows cross-platform paradigms and uses a plethora of programming languages and frameworks. Visual Studio 2019 combines the most important investments in technology into the integrated development environment (IDE) and includes new productivity features; integrated, cloud-based collaboration tools; an AI-assisted coding experience; and support for the most recent versions of .NET Core to create cross-platform applications that run across operating systems.

Visual Studio 2019 also improves performance in many ways. Not only you will notice how the new IDE is much faster at startup, but also how solution-loading performance has been optimized, especially for very large solutions. Like the previous version, it ships with the Enterprise, Professional, and free Community editions, which you can download from the official product page.

This book describes what's new in Visual Studio 2019 from the point of view of the IDE. It covers the improved installation experience, new productivity features for managing projects, and new features in the code editor. I'll also discuss new team collaboration and debugging features, as well as updated and improved support for mobile, web, and desktop development. Most of the topics discussed here apply to all the editions, except where expressly specified.

The only prerequisite is installing Visual Studio 2019. If you haven't already installed Visual Studio 2019, do it while reading Chapter 1 to learn about the new benefits of the Visual Studio Installer. After reading this book, you will be able to use all the new features to maximize your productivity.

# Chapter 1 The Installation Experience

Visual Studio 2019 adds new, interesting features to the installation experience. This chapter describes what's new about the Visual Studio Installer tool and how you can customize the installation process.

## Installing Visual Studio 2019

In order to install Visual Studio 2019, you will run the appropriate web installer file, which you download from the source you selected, such as your MSDN subscription for the Professional and Enterprise editions, or the Microsoft website for the Community edition. The installer file name varies depending on the selected edition:

- vs_community.exe
- vs_professional.exe
- vs_enterprise.exe

This will download and launch the installation program, called Visual Studio Installer. The Visual Studio Installer in Visual Studio 2019 uses the same design logic as its predecessor and allows for installing sets of components, each targeting a specific development scenario. Each set of components is referred to as a *workload*.

Workloads make installation and maintenance easier and allow developers to install what they actually need without unnecessary components, software development kits (SDKs), and tools. This way, you can save a lot of space on disk. You could even decide to install only the Visual Studio core editor without any additional workloads in order to get the basic coding environment. At startup, you will be able to select one or more workloads of your interest, as shown in Figure 1.

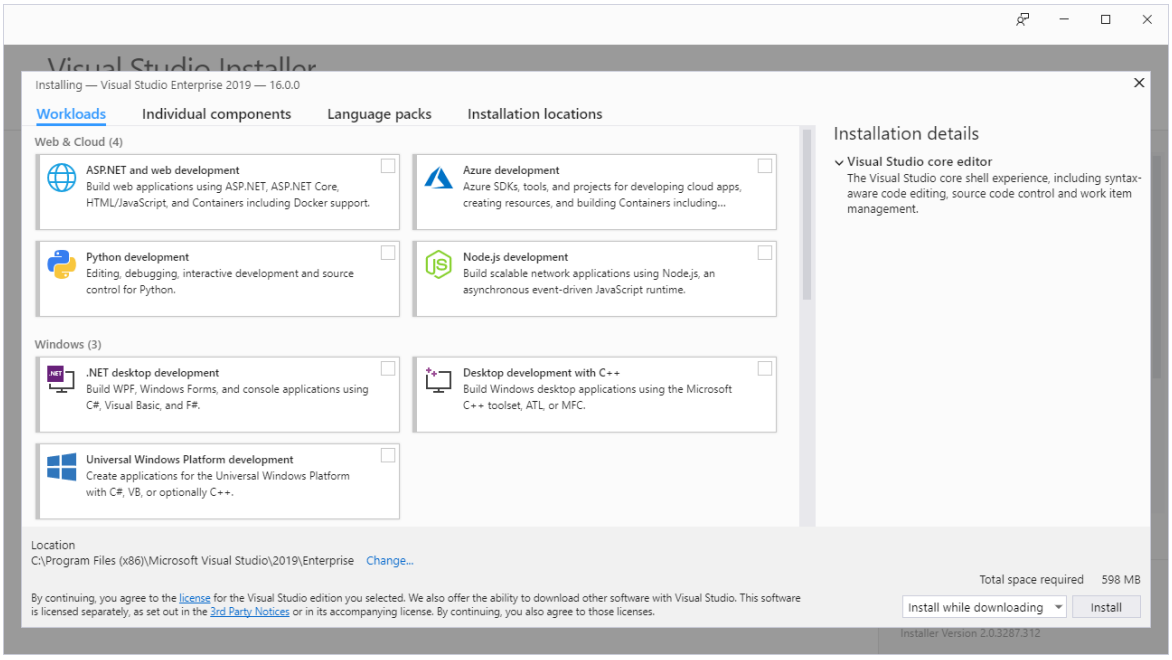


*Figure 1: Selecting workloads in the Visual Studio Installer*

Table 1 provides a description of each available workload.

*Table 1: Available workloads in Visual Studio 2019*

| **Available workloads in Visual Studio 2019** | |
|---|---|
| ASP.NET and web development | Select this workload to develop web applications using ASP.NET and standards-based technologies like HTML, JavaScript, CSS, and JSON. It also enables you to quickly deploy your app to a web server or to Azure. |
| Azure development | This workload installs the latest Azure SDK for .NET and tools for Visual Studio 2019. This allows you to view resources in Cloud Explorer, create resources using Azure Resource Manager tools, and build applications and services ready to be hosted in Azure. |
| Python development | Select this workload to enable full Python support within Visual Studio, including editing, debugging, interactive windows, profiling, package management, and source control. It works with your existing Python install to help you develop cross-platform scripts, web apps, native extension modules, and IoT applications. |

| Available workloads in Visual Studio 2019 | |
|---|---|
| Node.js development | This workload adds everything you need to build apps for Node.js, including IntelliSense, local and remote debugging, profiling, npm integration, an interactive window, test runners, and Azure integration. |
| Universal Windows Platform Development | Select this workload if you want to write universal applications for Windows 10, including PC, tablet, smartphone, the HoloLens, Xbox, and IoT devices. |
| .NET desktop development | Select this workload if you want to build classic Windows desktop applications with WPF, Windows Forms, and console apps using .NET Framework. |
| Desktop development with C++ | Select this workload if you wish to create, build, and debug native, classic desktop applications that run on versions ranging from Windows XP to the latest Windows 10 release, using the C++ language and environment. |
| Mobile development with .NET | This workload installs Xamarin, the technology that allows you to create native iOS, Android, and Universal Windows Platform apps using a shared C# codebase. |
| Game development with Unity | Select this workload if you want to develop cross-platform 2D and 3D games using the Unity framework and integrated tools for Visual Studio 2019. |
| Mobile development with C++ | Select this workload if you want to create cross-platform mobile apps using C++. |
| Game development with C++ | Select this workload if you want to create games using C++. |
| Data storage and processing | This workload provides tools for accessing on-premises SQL Server databases, SQL databases on Azure, and Azure Data Lakes resources. It also provides support for U-SQL, Hive, and Big Data on Azure. |

| Available workloads in Visual Studio 2019 | |
|---|---|
| Data science and analytical applications | This workload installs languages such as Python, R, and F#, which are tied to building applications for data analysis. |
| Office/SharePoint development | This workload provides the Office developer tools, which allow for creating Office and SharePoint add-ins and solutions. |
| Visual Studio extension development | This workload installs the Visual Studio SDK and allows you to write extensions such as new commands, tool windows, and templates. |
| Linux development with C++ | This workload enables you to author C++ code for Linux servers, desktops, and devices from within Visual Studio 2019. |
| .NET Core cross-platform development | This workload installs all the tools you need to write cross-platform web applications with .NET Core, and includes support for Docker. |

For the instructional purposes of this e-book, the following workloads are required:

- ASP.NET and web development
- .NET desktop development
- Universal Windows Platform development
- Mobile development with .NET
- .NET Core cross-platform development

You are not required to do the same—feel free to select only those you need. You can later install additional workloads as required.

***Note: Visual Studio 2019 has fewer workloads than Visual Studio 2017. These have been reorganized so that the installation is more efficient and allows for saving more space on disk, especially for mobile development with .NET.***

## Customizing the installation

As you would expect from an advanced setup tool, you can customize the Visual Studio installation in several ways. You can select additional individual components, and you can also select language packs and target folders on disk. This section will describe these customizations.

## Installing individual components

More often than not, selecting workloads is not enough to get the tools you actually need and you will need to install additional individual components. As an example, the GitHub extension for Visual Studio 2019 is not installed by default, which means you might want to select this component if you plan to work with Git repositories on that popular service. You can click the **Individual components** tab to see the full list of available individual components. Figure 2 shows an example.

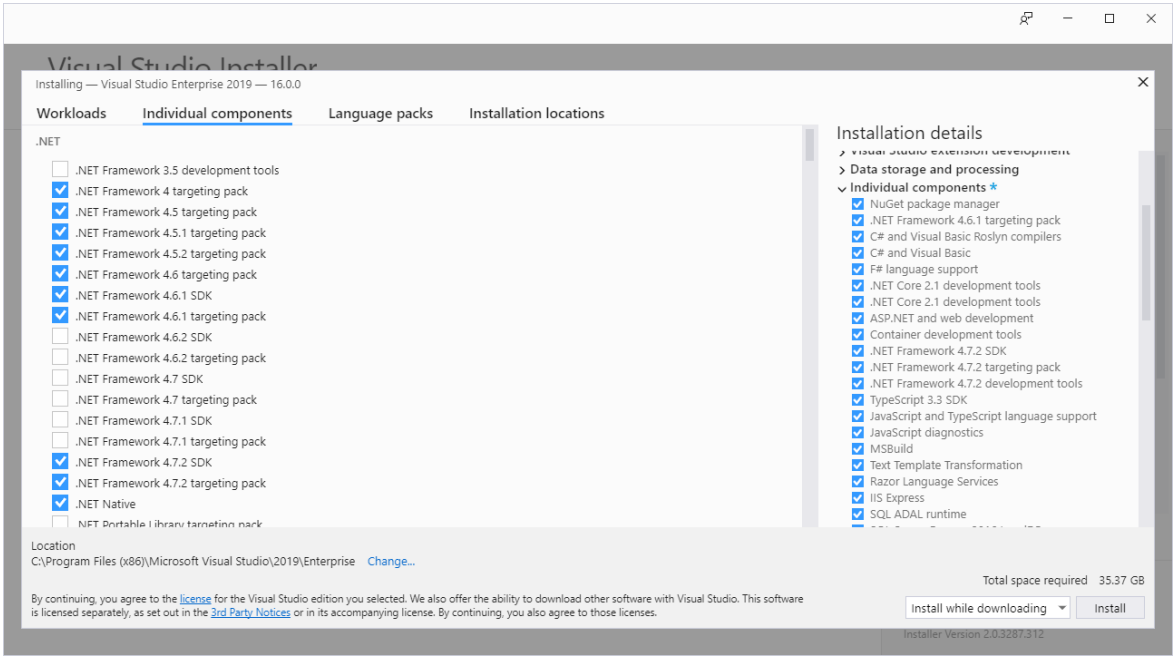


*Figure 2: Selecting individual components*

Individual components are grouped by development areas. On the right side of the installer, you can see a summary of all the individual components currently included in the installation.

## Installing language packs

A new feature in the Visual Studio Installer allows for installing language packs, so that you can use the IDE in the language of your choice. Simply click the **Language packs** tab and select one or more languages (see Figure 3).

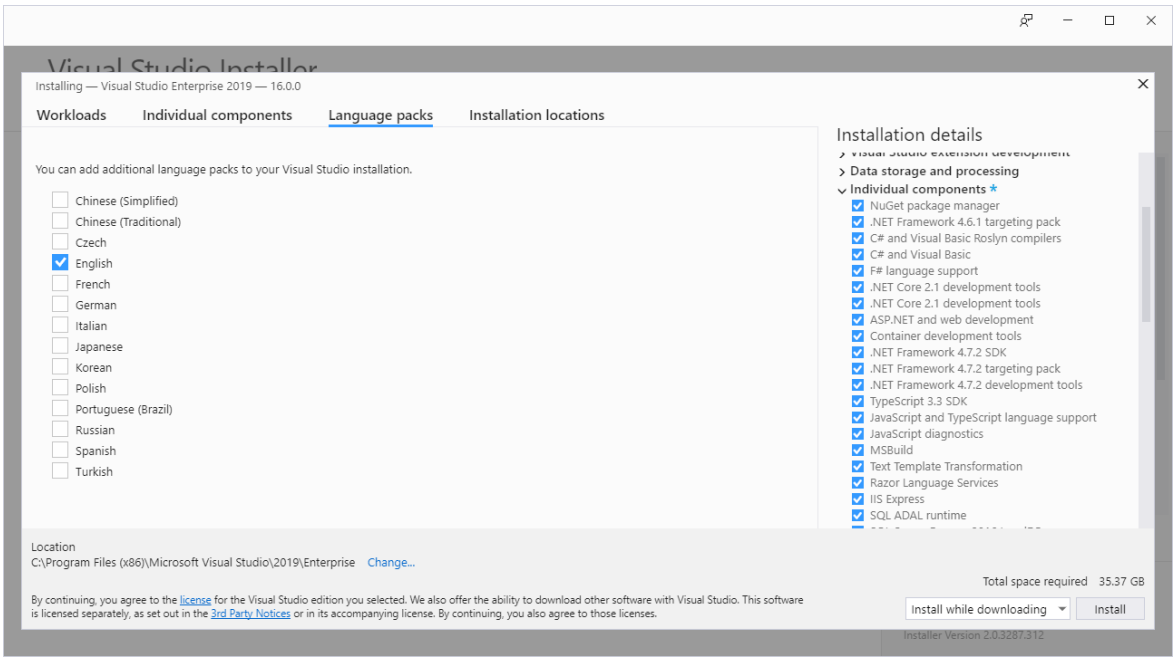


*Figure 3: Adding language packs*

Once Visual Studio 2019 is installed, you will be able to select a different language by selecting **Tools** > **Options** > **Environment** > **International Settings**.

***Tip: Changing the language requires restarting Visual Studio.***

## Customizing the installation folders

The Visual Studio Installer now allows for customizing the destination folders on disk. If you select the **Installation locations** tab (see Figure 4), you will be able to change the following target folders:

- **Visual Studio IDE**: By default, the target path is C:\Program Files (x86)\Microsoft Visual Studio\2019\{*editionName*}.
- **Download cache**: By default, the target path is C:\ProgramData\Microsoft\VisualStudio\Packages.
- **Shared components, tools, and SDKs**: The default target path is C:\Program Files (x86)\Microsoft Visual Studio\Shared.

***Note: The target folders for the download cache and for shared components can only be changed if no other versions of Visual Studio are installed on your machine. For example, if you also have Visual Studio 2017, the Installer will use the same target folders as for Visual Studio 2017, and you won't be able to select different destinations.***

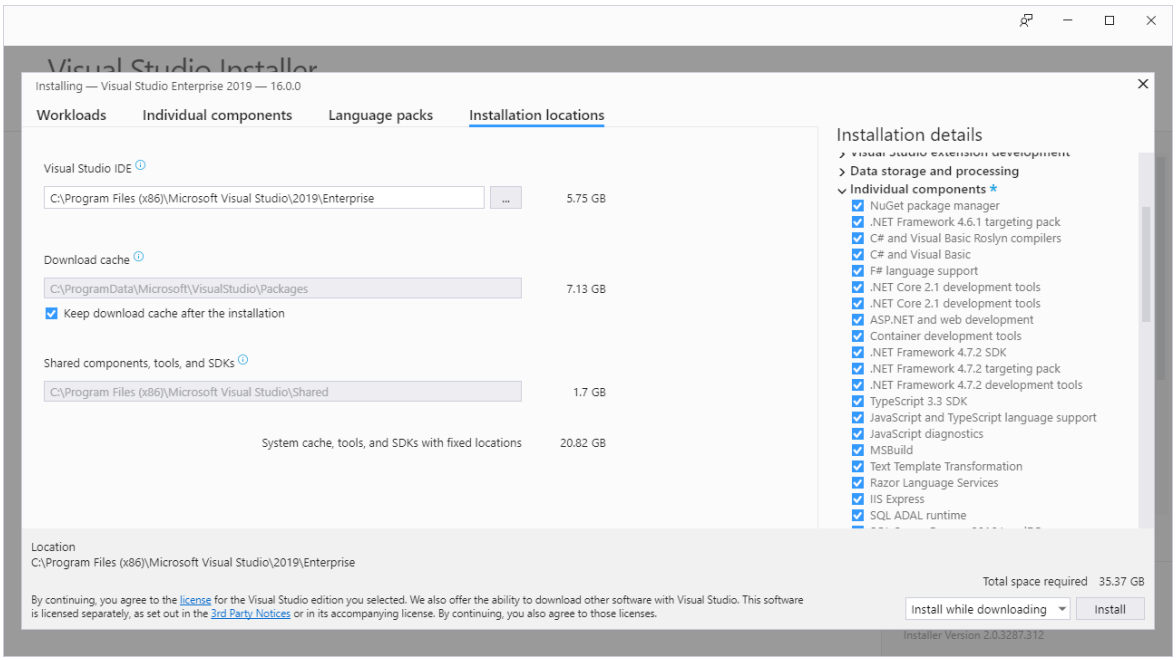


*Figure 4: Changing the installation folders*

## Managing multiple versions and editions

Like in Visual Studio 2017, you can install multiple editions side by side in Visual Studio 2019. For example, you might need to install both Professional and Enterprise editions on the same machine. Not limited to this, the Visual Studio Installer adds support for managing the installation of both Visual Studio 2017 and Visual Studio 2019. Figure 5 shows how the Installer recognizes the availability of Visual Studio 2017 on the machine while installing Visual Studio 2019.

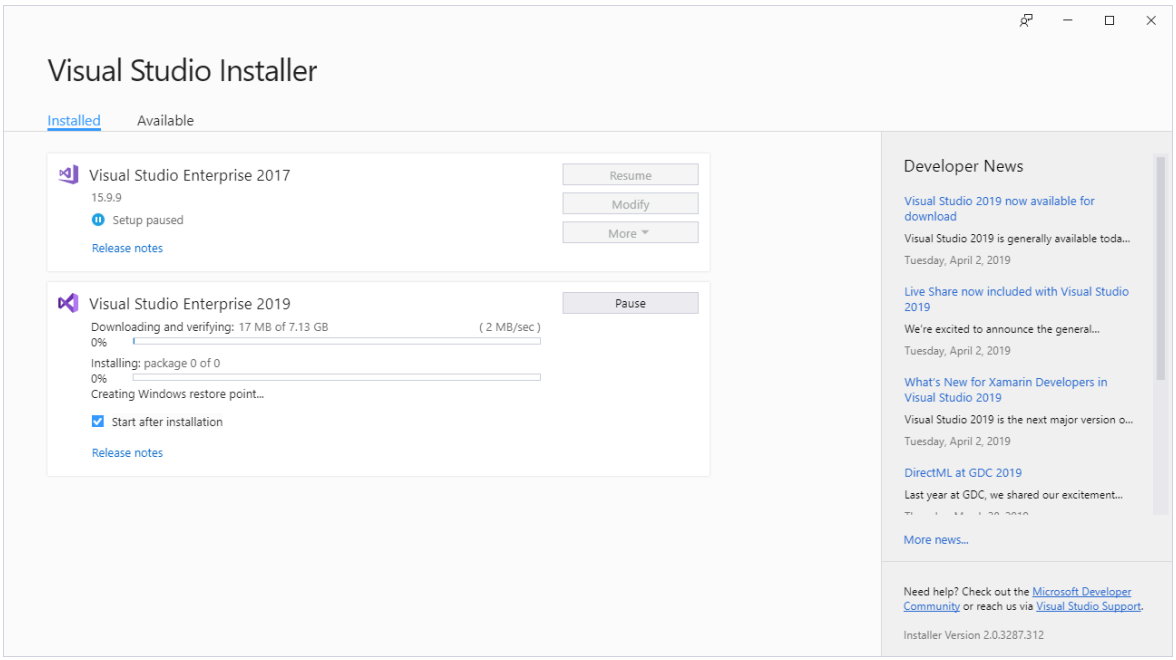


*Figure 5: The Visual Studio Installer can manage multiple versions*

The list of installed Visual Studio versions will be displayed when you launch the Visual Studio Installer.

***Note: It's not possible to modify the installation of another edition while a setup operation is in progress.***

## Creating an offline installer

When you download the installation bootstrapper from the source of your choice, you'll need an internet connection to download all the components you select. However, it is possible to create an offline installer, which can include all the workloads for full installation, or only the ones you need. The MSDN library offers a specific article about creating offline installers, explaining all the possible options and settings. The full offline installer size is about 35 GB. Obviously, creating an offline installer for the first time requires an internet connection.

***Tip: The Visual Studio Installer is also the tool through which you update Visual Studio 2019. When new updates are available, the Visual Studio Installer will make the update process straightforward. Later, in Chapter 2 Layout and Project Management, you will see how you get notifications of product updates from within the IDE.***

## Trying out preview features

Together with the stable release, Microsoft offers preview builds of Visual Studio 2019 that you can use to have an early look at what will be included with the next updates. In fact, it is possible to download and install the so-called Visual Studio 2019 Preview from a dedicated download page. You can install Visual Studio 2019 Preview side by side with the stable release, which enables you to test not only upcoming features in the IDE, but also to preview builds of the various SDKs (such as .NET Core 3.0 Preview).

Remember that this is pre-release software, so it is not supported, not to be used in production, and intended only for testing purposes.

## Starting Visual Studio 2019

As with its predecessor, Visual Studio 2019 is launched by using the same-named shortcut in the All Programs menu. When it starts for the first time, Visual Studio will ask for your Microsoft account credentials to log in (optional). As you might know, entering a Microsoft account will allow for synchronizing settings across machines. This will also automatically restore customized settings you might have on an existing VS 2017 or VS 2019 installation.

## Controlling Visual Studio updates

In Visual Studio 2019, you have an option to control how product updates are downloaded and installed. If you select **Tools** > **Options** > **General** > **Product Updates** (see Figure 6), you will be able to:

- Download updates automatically over nonmetered connections, and when the machine is idle.
- Decide whether to download updates and then install, or install while downloading.

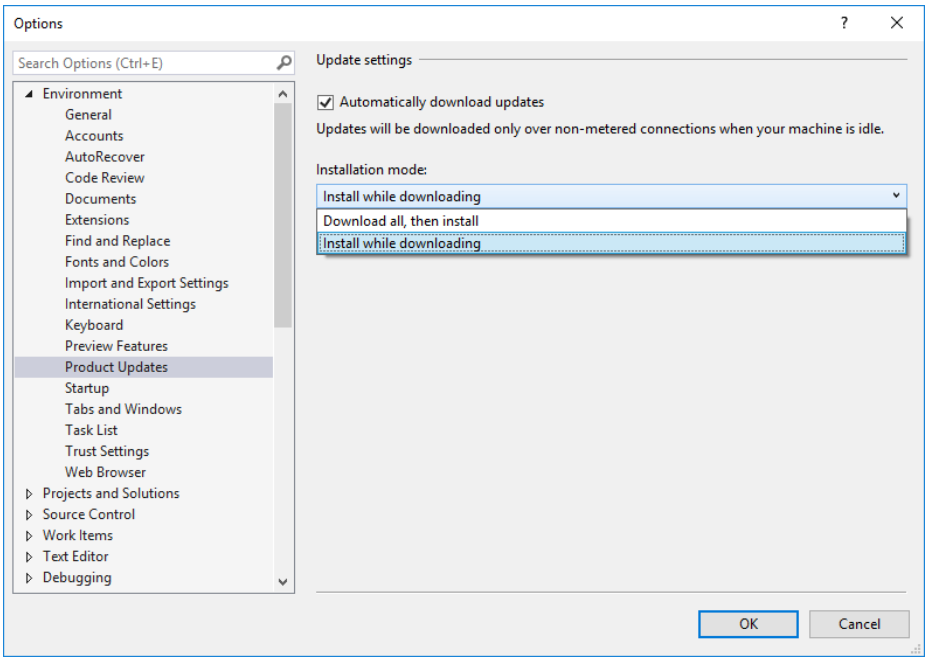


*Figure 6: Controlling product updates*

These settings are sent to the Visual Studio Installer engine, which will manage product updates based on your decision. Figure 6 also shows the default settings: automatic download and installation while downloading.

## Chapter summary

The installation experience in Visual Studio 2019 is based on the Visual Studio Installer tool, introduced with Visual Studio 2017. In the new version, not only can you still select only the workloads you actually need for easier setup and maintenance, but you can now select language packs and change the installation folders. You can also manage multiple editions and versions in one place.

# Chapter 2  Layout and Project Management

When you run Visual Studio 2019 for the first time, you will immediately notice some differences with Visual Studio 2017. Additionally, many tools have been moved to different places, some changes have been done to the IDE layout, and the way you create new projects has completely changed. This chapter describes what's new in the IDE layout and project management tools, so that you will quickly become more familiar with VS 2019.

## The IDE startup

Microsoft is always committed to improving its products' performance, and Visual Studio 2019 is no exception. When you launch VS 2019, you will see that startup is much faster than with Visual Studio 2017. You will also notice a new splash screen and a new product icon. You will no longer see the start page like in the previous versions, as it has been discontinued in favor of a new dialog called the start window.

## Introducing the start window

When Visual Studio 2019 launches, the first visual element you see is the new start window. Figure 7 shows an example.

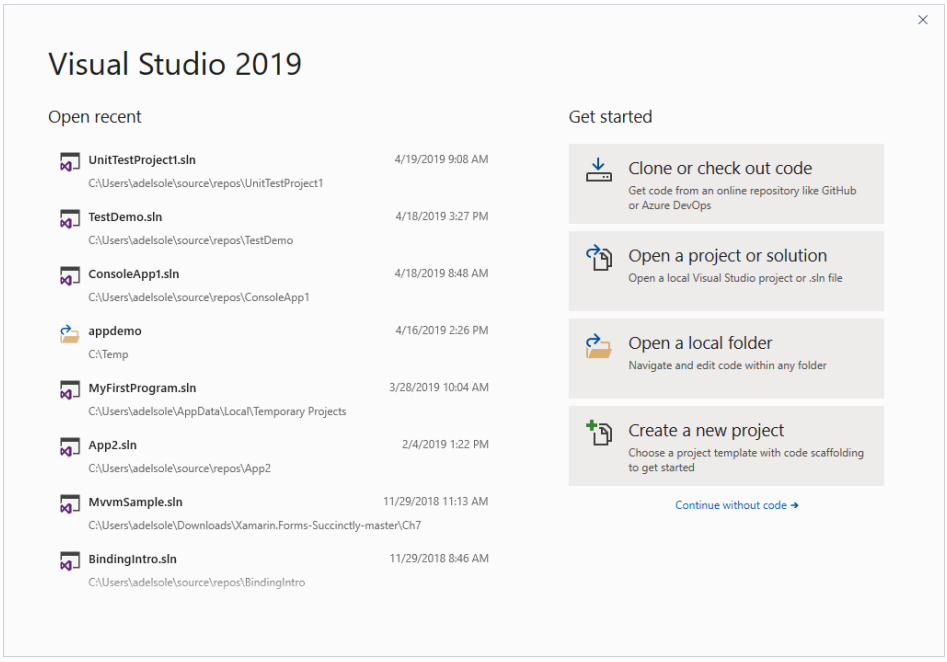


*Figure 7: The start window*

The start window replaces the start page you were used to seeing in the previous versions of Visual Studio. Here you can see the list of recently used projects on the left. You can remove items from the list by right-clicking them and selecting **Remove From List**. On the right, you get shortcuts to common actions:

- **Clone or check out code**: This shortcut allows you to quickly connect to a Git repository on Azure DevOps or GitHub, so that you can easily clone the repository or simply check out the code. I'll discuss this feature in more detail in the next section.
- **Open a project or solution**: This shortcut allows you to open an existing solution or project by using a simple, common dialog.
- **Open a local folder**: This shortcut allows you to open an existing folder. Microsoft introduced this feature in Visual Studio 2017.
- **Create a new project**: This shortcut will launch the user interface for creating a new project. This feature has been completely redesigned in Visual Studio 2019 and will be discussed thoroughly in the next section.

You are not obliged to select one of these actions: you can click the **Continue without code** hyperlink to simply open the Visual Studio IDE without working on code. Connecting to a Git repository and creating a new project require more detailed explanations, so the next section walks through both features.

***Tip: You can customize the behavior of Visual Studio at startup by selecting Tools > Options > Environment > Startup. Possible options include displaying the start window, the most recent solution, or the empty environment.***

# Working with projects and solutions

The start window simplifies working with projects and solutions by providing useful shortcuts. You can quickly open an existing solution via the **Open a project or solution** shortcut, and you can also open a folder via the **Open a local folder** shortcut. If you instead do not wish to open a project or solution at startup, you can simply click the **Continue without code** hyperlink in order to open the IDE with no code. You will then be able to open a solution later, via the usual commands in the File menu. In the next two paragraphs, I will cover two features that present a new user interface and experience in Visual Studio 2019: creating new projects and cloning existing code repositories.

## Creating new projects

Visual Studio 2019 provides a completely new user interface for creating new projects. In the start window, click **Create a new project**. The same-named dialog appears at this point (see Figure 8).

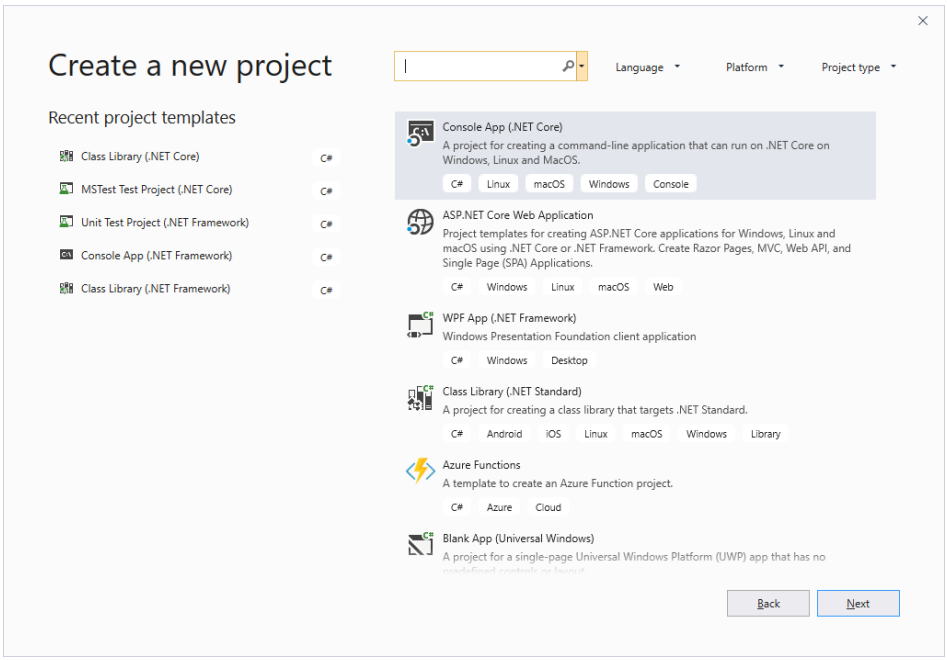


*Figure 8: The Create a new project dialog*

On the left side of the dialog, you will see a list of recently used project templates. You can select a template from the list, scroll the list for additional templates, or you can search for a specific template by typing in the search box. You can also quickly filter the list by language, platform, and project type, as well as combine multiple filters. For example, Figure 9 shows the list of templates filtered by language (Visual Basic), platform (Windows) and project type (Library).

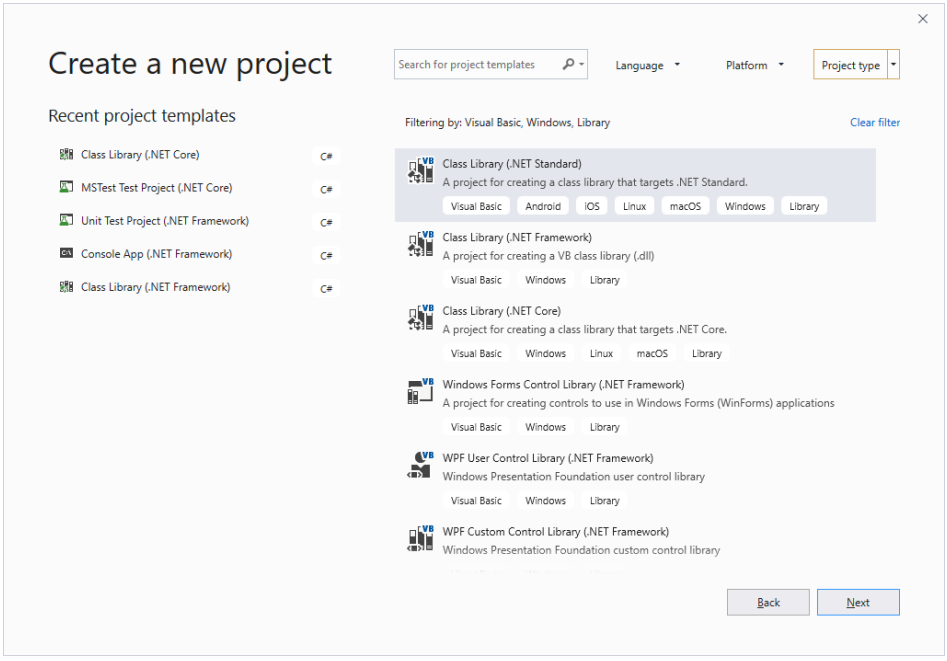


*Figure 9: Filtering the list of project templates*

The dialog shows the filters currently applied, if any, and provides the Clear filter shortcut, which restores the original view. When you have selected the project template of your choice, click **Next**. The “Configure your new project” dialog appears (see Figure 10) and allows for entering a project name, and for specifying the target folder. Note that, unlike its predecessors, in VS 2019 the dialog asks if you want to place the solution and project in the same directory, rather than asking you if you want to create a separate directory for the solution.

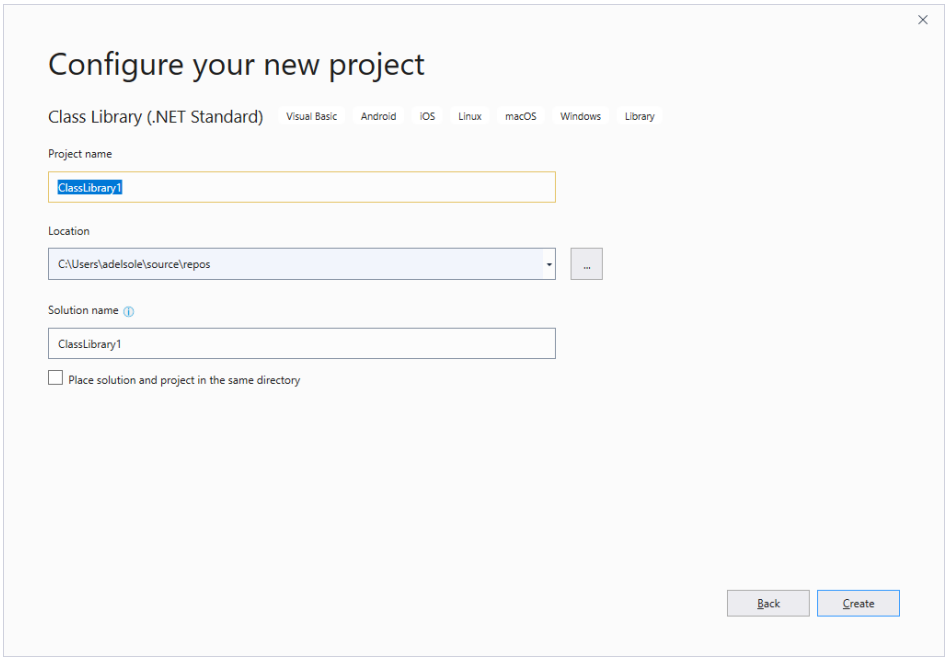


*Figure 10: Configuring project name and folder*

When you’re ready, click **Create**. Visual Studio 2019 will generate a new solution and project the usual way, and you will quickly get to the code editor and to all the other well-known tools.

## Cloning existing code repositories

Though you can still connect to an online code repository via the Manage Connections button of the Team Explorer window, you can quickly connect to an Azure DevOps or GitHub repository at startup by clicking the **Clone or checkout code** shortcut in the start window. When you click this shortcut, the same-named dialog appears (see Figure 11).

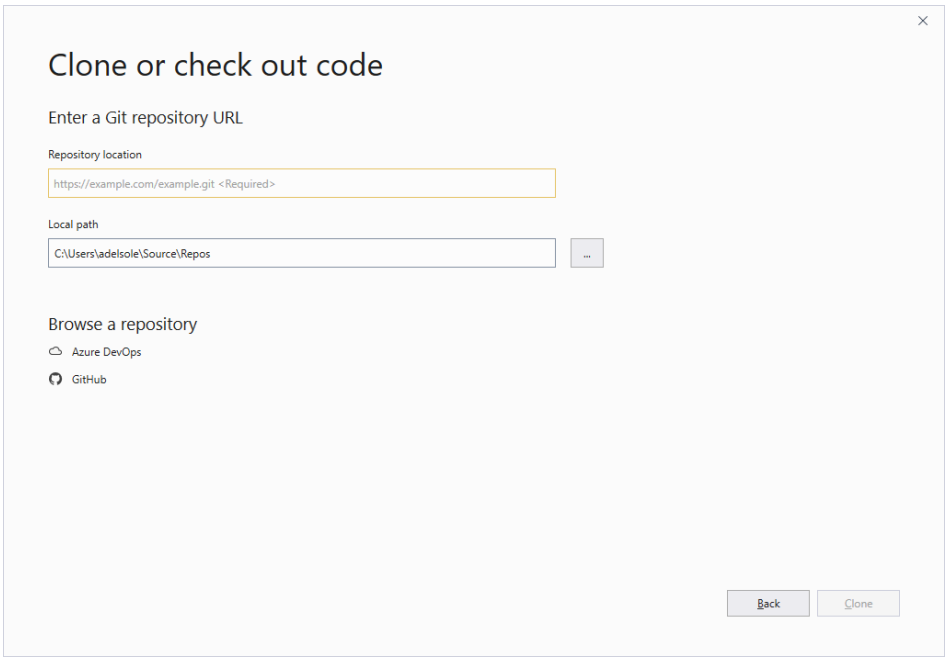


*Figure 11: Configuring the connection to a Git repository*

The first option you have is to provide the address of a Git repository in the **Repository location** box. The repository can be hosted on any Git provider. You can then specify a local path to which the repository will be cloned and click **Clone**. For repositories hosted on Azure DevOps or GitHub, you can quickly browse the list of projects by clicking either link. In the case of Azure DevOps, you will be prompted with the list of servers and of team projects that your current Microsoft Account has access to, as demonstrated in Figure 12.

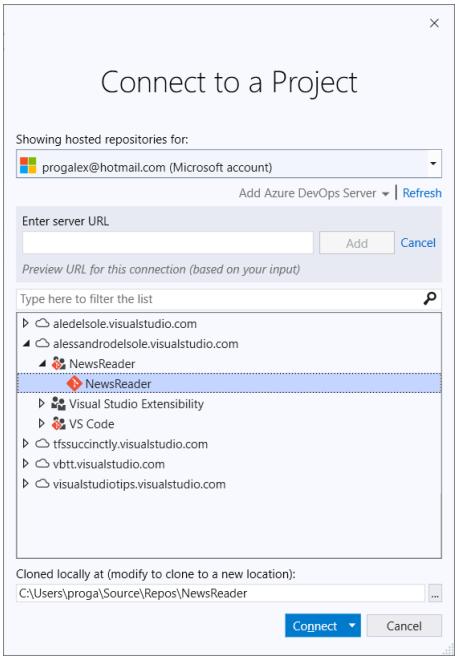


*Figure 12: Connecting to a Git repository hosted on Azure DevOps*

If a specific server is not listed, you can click the **Add Azure DevOps Server** hyperlink and enter the server address manually. You can then simply expand a server name, list all the team projects, and select the one you want to clone. When you're ready, make sure the local target path is appropriate for you, and then click **Connect** (or double-click the project name).

***Tip: To browse other servers, you can click the Microsoft Account combo box and provide a different email address.***

At this point Visual Studio 2019 starts cloning the repository to the local machine. When finished, it will open up the Team Explorer window, showing a success message and the address of the repository (see Figure 13).

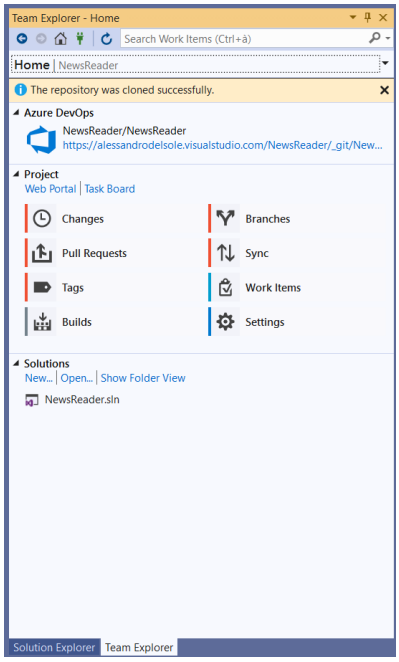


*Figure 13: The completion message after cloning a repository*

Visual Studio 2019 automatically opens the repository as a folder. If the repository contains a solution, you can open Solution Explorer and double-click the solution file of your interest and move from the folder view to the solution view. Cloning a repository hosted on GitHub works similarly. In fact, if you select GitHub in the "Clone or checkout code" dialog, you will be first asked to enter your GitHub credentials, and then you will be able to browse all your repositories, as shown in Figure 14.

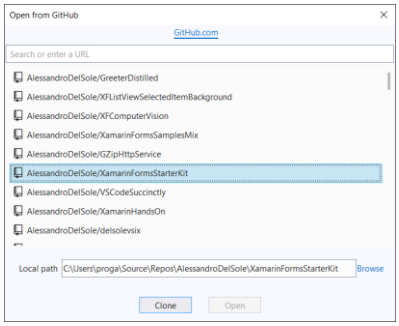


*Figure 14: Browsing GitHub repositories*

Simply select the repository of your interest, check out the local target path, and then click **Clone**. Visual Studio will display the Team Explorer window like you have seen for Azure DevOps, with the same default behavior of opening the code as a folder.

## Filtering solutions

With solutions containing many projects, you can now reduce the time required for opening the solution by loading only a subset of projects. To accomplish this, in the **Open Project/Solution** dialog, make sure you select the **Do not load projects** check box (see Figure 15).

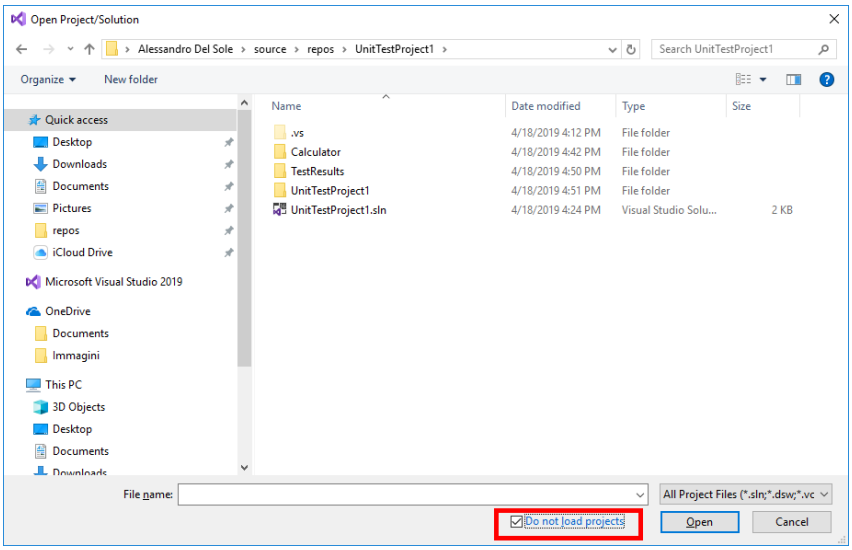


*Figure 15: Opening a solution without loading projects*

At this point, the solution will be visible in Solution Explorer with all the projects in the *unloaded* state (see Figure 16).

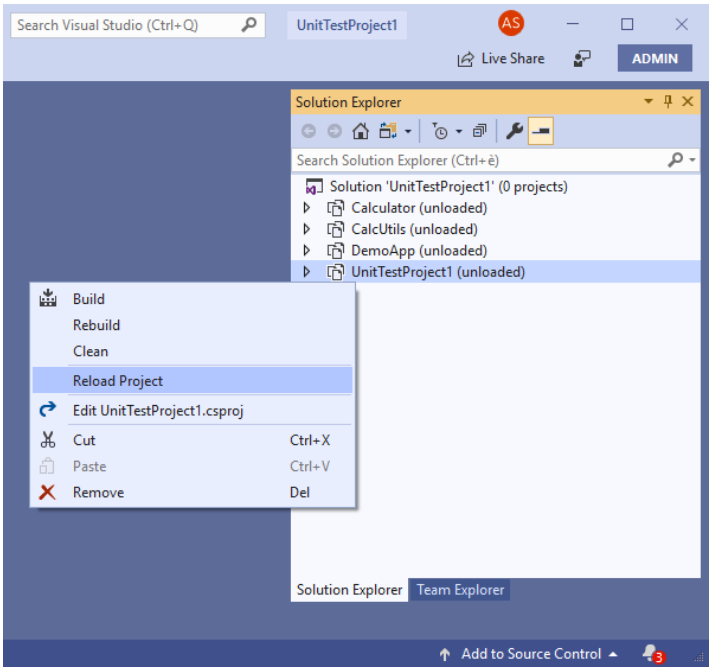


*Figure 16: Loading only the desired projects*

Now, right-click the projects you want to load and select **Reload Project** for each one. Visual Studio will remember your selection and, the next time you open the solution, it will load only the projects that you have selected. If your filter should be shared with other developers, for example via source control, you can create a solution filter file. To accomplish this, right-click the solution name in Solution Explorer and then select **Save As Solution Filter**. This will create a .slnf file that is part of the solution, and which can be committed to your code repository.

### Project visibility and dependencies

You can hide unloaded projects from the Solution Explorer view by right-clicking the solution name and selecting **Hide Unloaded Projects**. To restore the view, right-click again and select **Show Unloaded Projects**. Loaded projects might have dependencies on unloaded projects and their dependencies, so Visual Studio offers an option called Load Project Dependencies, which allows for loading dependencies that are necessary to compile and run the loaded projects. This option is available when you right-click a loaded project in Solution Explorer.

## Graphical enhancements

Visual Studio 2019 introduces a couple of graphical improvements: an updated Blue theme and enhanced multi-monitor support. This section covers both.

## The Blue theme

In Visual Studio 2019, the popular Blue theme has been revised for the first time since Visual Studio 2012. In the new version, the Blue theme has sharper overall contrast, which helps improve legibility and accessibility. It also makes it easier to distinguish between different versions of Visual Studio running on the same machine. The brightness of the base color has been reduced, and a set of complementary and analogous accent colors has been added. Actually, Visual Studio 2019 offers the Blue theme and the Blue (Extra Contrast) theme. Both have lighter colors than their counterpart in the previous versions of Visual Studio, and the Blue theme offers lighter colors for the code editor, as you can see in Figure 17.

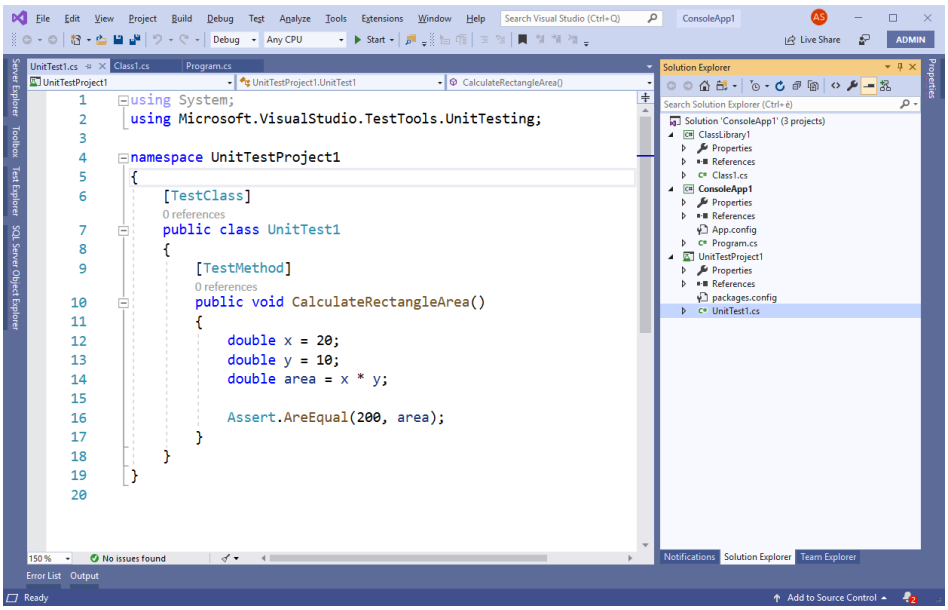


*Figure 17: The updated Blue theme*

The Blue (Extra Contrast) theme basically includes the same colors, but with much higher contrast, as you can see in the code editor in Figure 18.

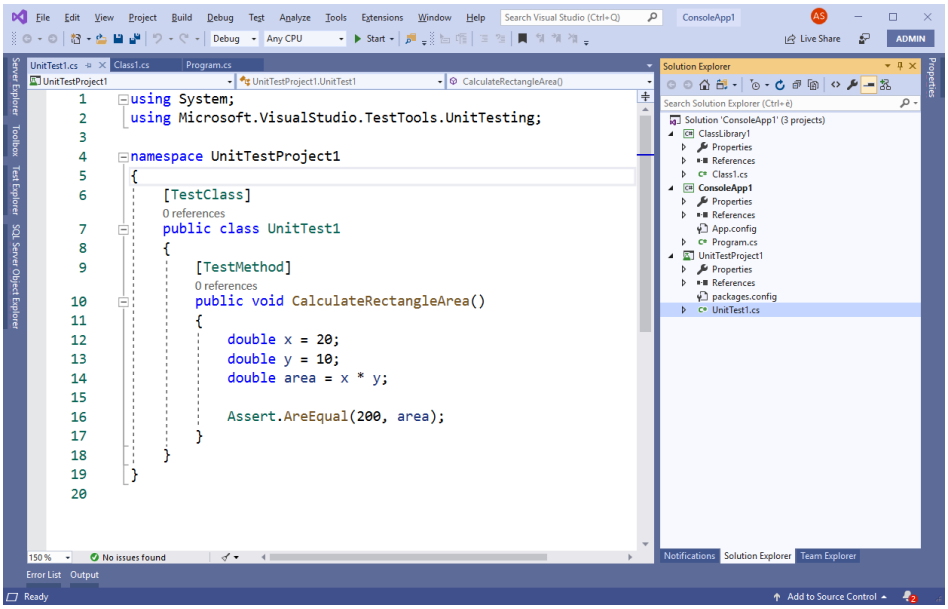


*Figure 18: The Blue (Extra Contrast) theme*

Like in the previous versions, you change the theme via **Tools** > **Options** > **Environment** > **General** > **Color Theme**. The Dark and Light themes are obviously still available.

## Improved multi-monitor support

In the past, if you worked with Visual Studio across multiple monitors with different scale factors, you might have noticed that the code could be rendered incorrectly and, in some cases, it could also look blurry. This happened because previous versions of Visual Studio were set to render as a system scaled application.

Visual Studio 2019 introduces support for the per-monitor DPI awareness (PMA). PMA means that the IDE, and more importantly, the code you work on, appear crisp in any monitor display scale factor and DPI configuration, including across multiple monitors. PMA is not enabled by default. You must select **Tools** > **Options** > **Environment** > **General**, and then select the **Optimize rendering for screens with different pixel densities** check box (see Figure 19).

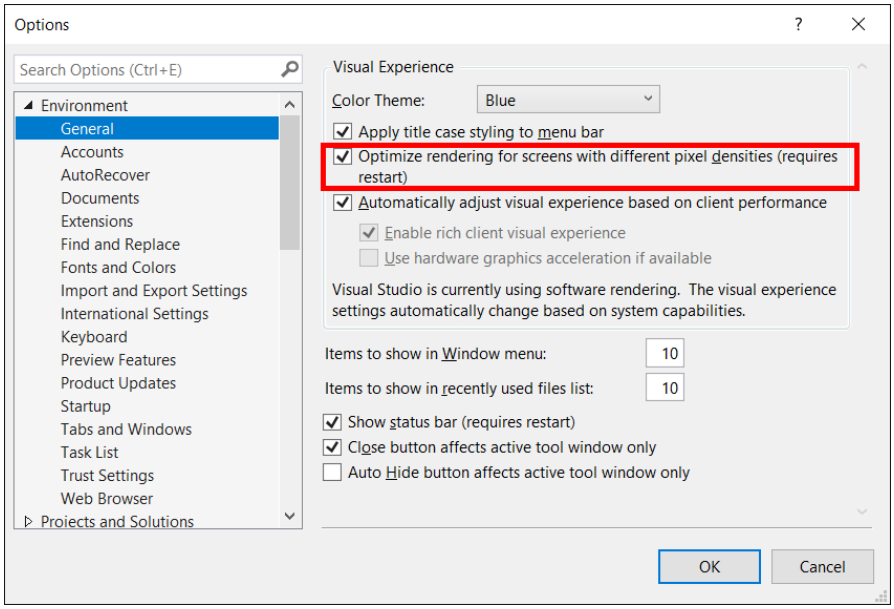


*Figure 19: Enabling PMA*

***Note: This feature requires Windows 10 version 1803 and the .NET Framework 4.8. If your system doesn't meet these requirements, it will be disabled in the Options window.***

According to Microsoft, these are the areas where PMA brings noticeable improvements:

- Core Shell
- Menus and context menus
- Most code editors
- Solution Explorer
- Team Explorer
- Toolbox
- Breakpoints
- Watch
- Locals
- Autos
- Call stack

Since most developers work with multiple monitors, this is a very important feature that will also decrease the fatigue of your eyes after many hours of coding.

## Layout and tool changes

In Visual Studio 2019, some tools have been repositioned in the IDE, and others have a slightly different layout. This section describes the most relevant changes in the development environment, so that you will not feel confused.

## Search box and solution name

In Visual Studio 2019, the search tool known as Quick Launch has been renamed Visual Studio Search and moved next to the Help menu, as you can see in Figure 20, where you can also see an example of a search.

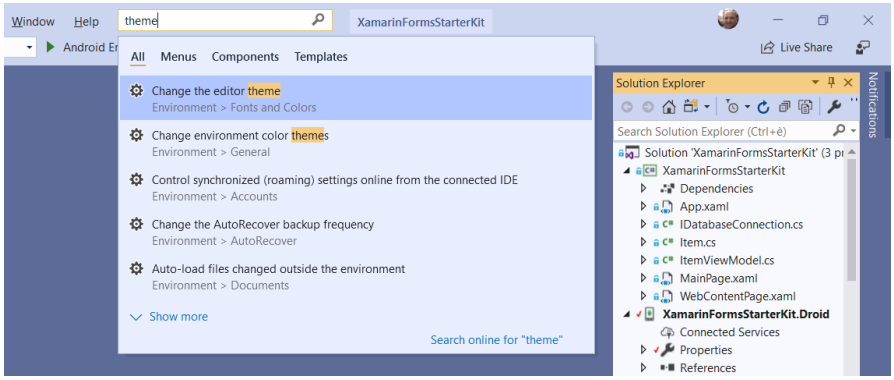


*Figure 20: Searching across Visual Studio*

The new search tool has been improved with filters by all items, menus, components, and templates. The solution name has also been moved next to the search box (see Figure 20), and the background color for the name is lighter so that it can be recognized more easily. Additionally, when you click the search box, the last three most-recently used actions will be displayed.

## The Notifications hub

The Notifications hub is still offered as a tool window. However, the bell icon is now slightly different than in Visual Studio 2017 and shows the number of notifications available. Figure 21 shows both the bell icon at the bottom right of Visual Studio 2019, and a sample list of notifications.

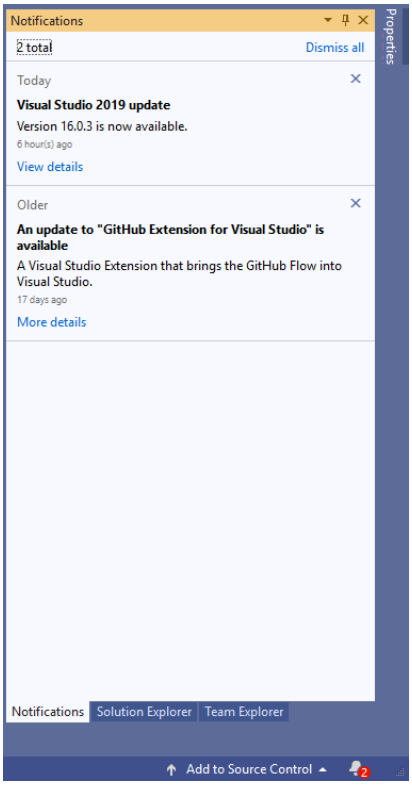


*Figure 21: The Notifications hub*

Depending on the notification, you will see either the View details or the More details hyperlinks, which you can click to discover more about a specific notification. In the case of Visual Studio updates, by clicking the notification you will be redirected to the Visual Studio Installer, which will show an Update button (see Figure 22).

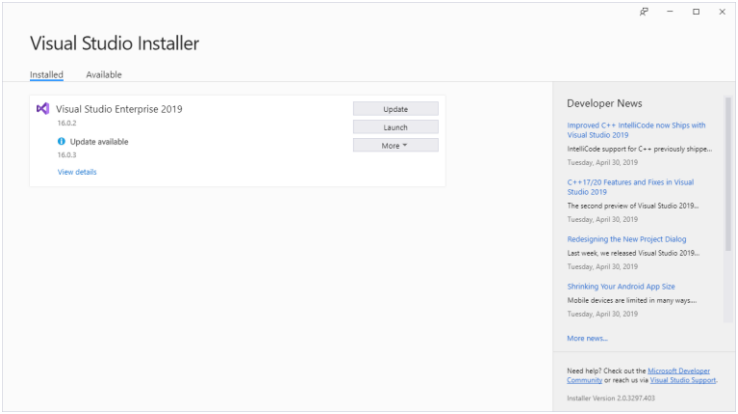


*Figure 22: Updating Visual Studio 2019*

In case of extension updates, the process is the same as in Visual Studio 2017, so updates are actually scheduled to be installed once you close Visual Studio.

## Reporting feedback

The Feedback menu is now placed below the closing icon. You can use it to report a problem, and Visual Studio 2019 will show the built-in feedback tool. It automatically connects to your Microsoft account and will display any previous feedback you submitted. Figure 23 shows an example based on my activity.

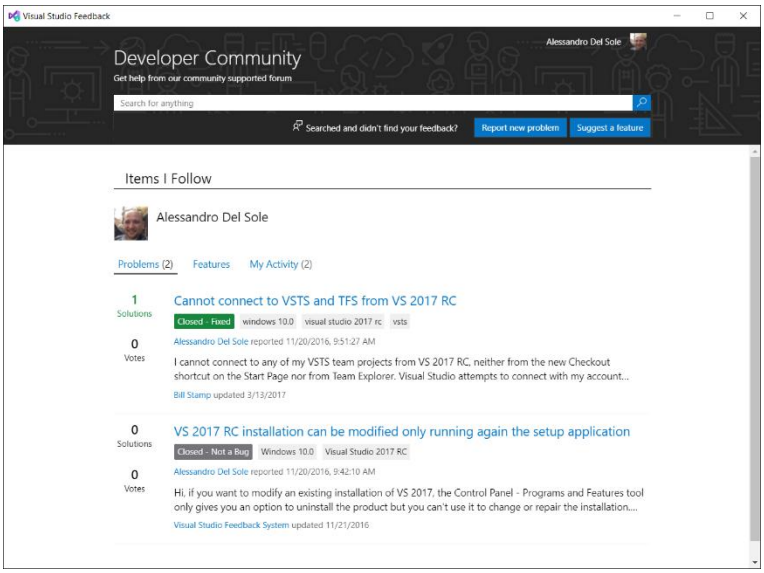


*Figure 23: Latest developer activities from the feedback tool*

If you click **Report new problem**, you will be able to fill in a form (see Figure 24) to send your feedback.

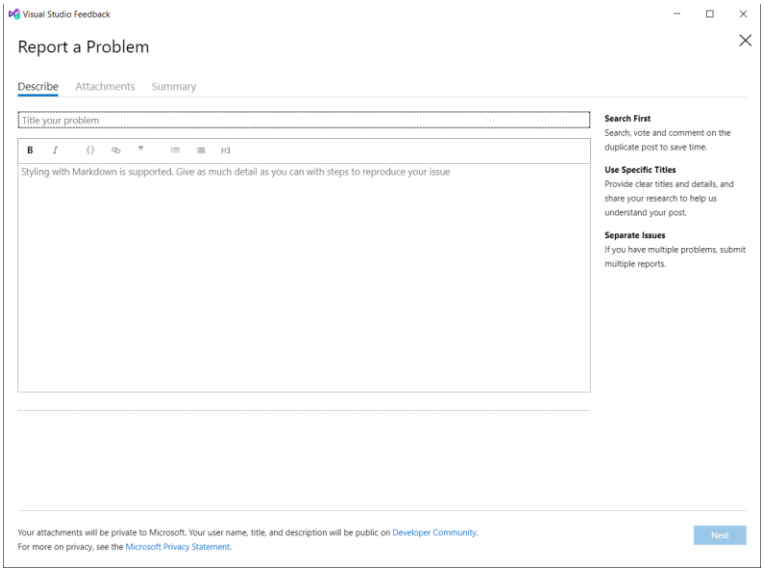


*Figure 24: The form offered to report a problem*

You will get a similar form to send a suggestion if you click **Suggest a feature** from the screen seen in Figure 23. However, if you click the **Suggest a feature** item under the feedback menu, you will be redirected to the Developer Community website, rather than to a built-in form.

### The Extensions menu

Visual Studio 2019 introduced a new menu called Extensions. This menu contains the Manage Extensions menu item, previously available under the Tools menu. It also contains submenus that contain shortcuts to items installed by third-party providers. In Figure 25, you can see the Extensions menu and a submenu pointing to installed extensions by Syncfusion.

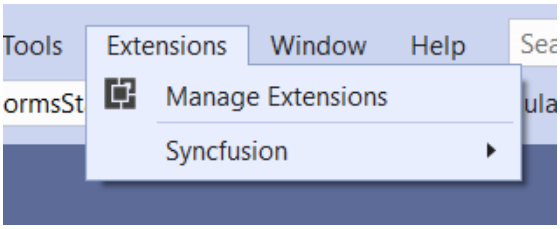


*Figure 25: The Extensions menu*

## Chapter summary

Visual Studio 2019 introduces new ways to work with projects and solutions after startup. With the start window, you have shortcuts to common tasks, such as creating new projects and cloning Git repositories from Azure DevOps and GitHub. There are also graphical enhancements, such as high-contrast and low-contrast Blue themes, as well as improved multi-monitor support with PMA. Finally, some tools have been repositioned on the screen, such as the search box, the solution name, and the feedback tool. In the next chapter, you will discover productivity improvements for the code editor.

# Chapter 3  Code Editor Productivity

With every major release, Microsoft introduces new productivity features to the code editor in Visual Studio, because it is the place where you spend most of your development time. VS 2019 improves the way you write high-quality code with lots of productivity enhancements. This chapter is all about the code editor, including new features and updated tools.

## Document Health Indicator and Code Cleanup

Visual Studio 2019 includes a new feature called Document Health Indicator with the code editor. As the name suggests, the Document Health Indicator detects code issues in a single document, showing icons and issue descriptions. In the VS terminology, a document is an individual code file opened in the IDE. For example, Figure 26 shows the Document Health Indicator with a document that has no code issues.

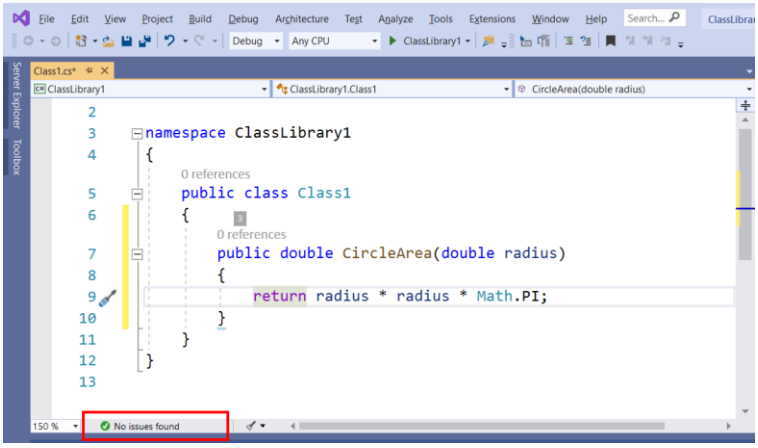


*Figure 26: The Document Health Indicator*

If the code file contains errors, the Document Health Indicator shows the number of errors and warnings represented by appropriate icons. You can click on the error or warning icon to quickly open the Error List window, and you can quickly navigate among code issues by using the two arrows near the indicator. Figure 27 shows an example of code file that contains intentional errors and shows how the Document Health Indicator appears at this point.

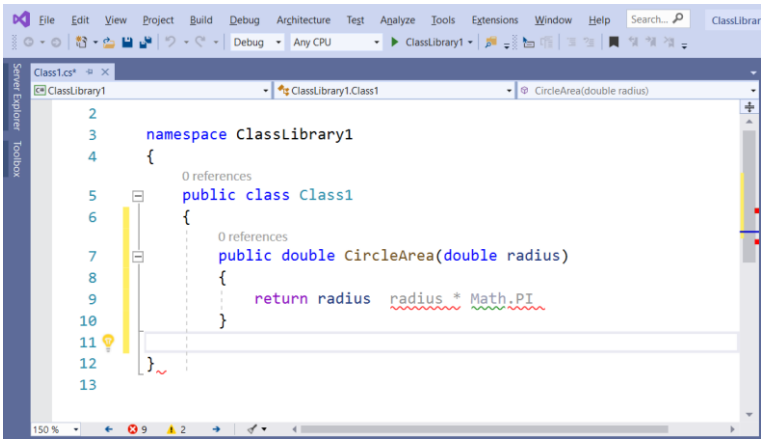


*Figure 27: The Document Health Indicator shows the number of errors and warnings*

The Document Health Indicator is very useful, especially if you don't keep the Error List window open, and gives you an updated report of the number of code issues in your document.

***Note: The Document Health Indicator is available not only for C# and Visual Basic, but also for other languages, including C++ and Python.***

Another very useful addition to the code editor is Code Cleanup. Code Cleanup is a customizable tool that automates the maintenance of your code by applying more rules. In the code editor, it is represented by the sweep/broom icon on the right side of the Document Health Indicator (see Figures 26 and 27). You can click the **Code Cleanup** icon to run the tool with the built-in rules, but you might want to customize it first.

***Note: Code Cleanup is only available in C# and Visual Basic.***

If you click the arrow at the right side of the icon, you will see the following commands:

- Run Code Cleanup (Profile 1)
- Run Code Cleanup (Profile 2)
- Configure Code Cleanup

Code Cleanup ships with two profiles, each customizable with different rules. By default, clicking the **Code Cleanup** icon runs Profile 1. You instead customize each profile by clicking **Configure Code Cleanup**. The same-named dialog that appears is shown in Figure 28.

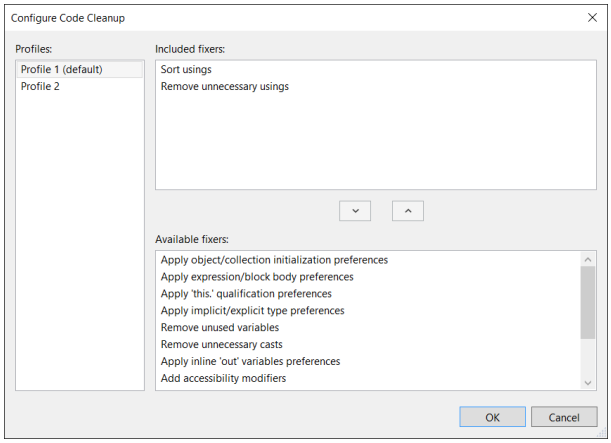


*Figure 28: Configuring Code Cleanup*

Each profile includes many *fixers*. A fixer is a code-refactoring action, such as Sort usings and Remove unnecessary usings. These are the fixers included by default. You can include more fixers, selecting them from the **Available fixers** list and then clicking the up-arrow button. For example, you could select the **Remove unused variables** fixer. You can remove a fixer from the Included fixers list by clicking the down-arrow button. When you have customized your profiles, click **OK**.

Now let's consider the code shown in Figure 29, which intentionally includes unnecessary **`using`** directives and an unused integer variable.

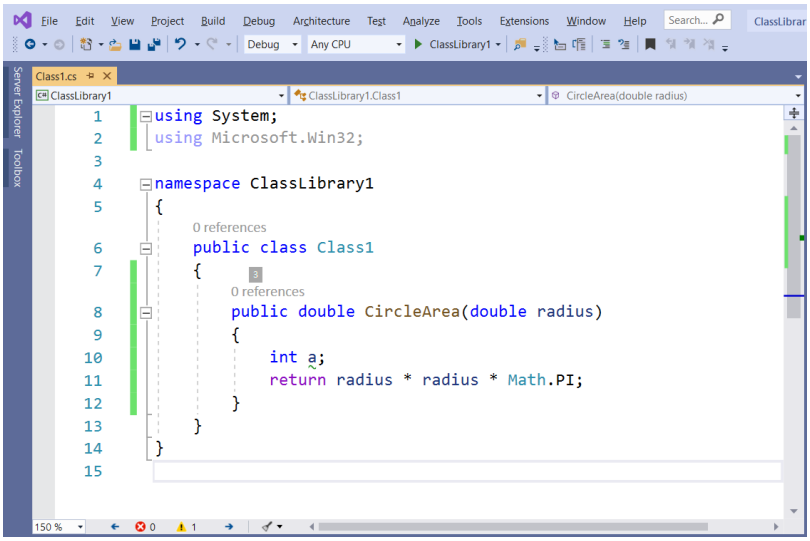


*Figure 29: Code with issues ready for Code Cleanup*

Clicking the Code Cleanup icon will result in the removal of both the unnecessary **`using`** directive and the unused integer variable. Note that some of the Code Cleanup fixers are strictly related to the code style preferences that you can set up in the language options. Code Style is a feature that was introduced in Visual Studio 2017, and you can access these preferences via **Tools** > **Options** > **Text Editor** > **C#** (or **Visual Basic**) > **Code Style**. Figure 30 shows how the Code Style options look.

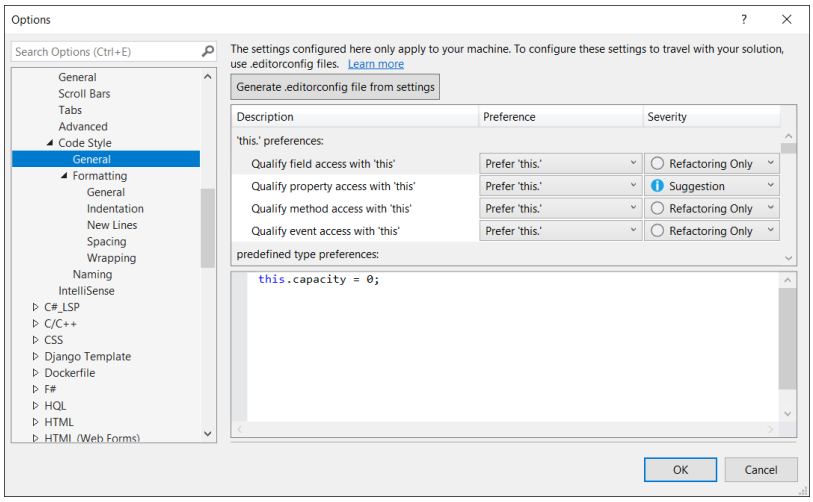


*Figure 30: Code Style options for C#*

All the Code Cleanup fixers whose names contain the word *preferences* are applied based on the Code Style preferences. For example, with the fixer called **Apply 'this.'**, qualification preferences will be applied during Code Cleanup only if your code style preferences for the usage of the **`this`** keyword have a severity of at least Warning, and if your code contains qualifications of fields, properties, methods, or events that match the specified preference. Configuring Code Style might take some time, but the combination of this feature with Code Cleanup gives you a powerful set of tools that helps you write code based on your team's requirements, and that makes maintenance much easier and faster.

# Clipboard Ring

You can quickly view the history of items copied into the clipboard by simply pressing CTRL + Shift + V in the code editor. This feature is not new, but has been redesigned in Visual Studio 2019. Figure 31 shows an example.

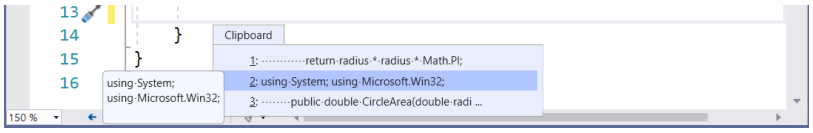


*Figure 31: Clipboard Ring in action*

If you hover over an item, a tooltip will show its content. If you click an item, its content will be pasted into the code editor.

# Classification colors

In Visual Studio 2019, colors in the code editor are now based on the so-called Roslyn classification colors. These color schemes are already part of Visual Studio Code, and they have been brought to VS 2019 as well. Now there are basically more colors in your code.

If you look at Figure 27, you can see how, in the Blue theme, a method name is represented in a tone of brown, and how the **`return`** statement is represented in violet. Obviously, color schemes differ in the various themes, but colors can be still customized via **Tools** > **Options** > **Environment** > **Fonts and Colors**. In the list of colors, scroll to the items whose names start with **User Members**. These specific items represent the Roslyn classification colors that you can customize according to your preferences.

# Quick actions and refactorings for C# and Visual Basic

Introduced with Visual Studio 2015 and dramatically improved with Visual Studio 2017, quick actions and refactorings based on the .NET Compiler Platform (Roslyn) for C# and Visual Basic make additional steps forward in VS 2019. They introduce new code fixes that will help you keep your code in good health. This section describes the most important new quick actions that will boost your productivity while coding.

***Tip: For additional quick actions and refactoring, you can read the "Visual Studio 2019 .NET Productivity" blog post.***

## Converting foreach loops to LINQ or lambda expressions

A new refactoring quickly allows you to convert a **`foreach`** loop into a LINQ or lambda expression. For example, consider the following code snippet, which includes a **`foreach`** loop over an object of type **`IEnumerable<int>`**.

```
private IEnumerable<int> IterateCollection(IEnumerable<int> collection)
{
    foreach (int number in collection)
    {
        if (number > 10)
            yield return number;
    }
    yield break;

}
```

The new quick action offers to refactor the code as a LINQ expression, as demonstrated in Figure 32.

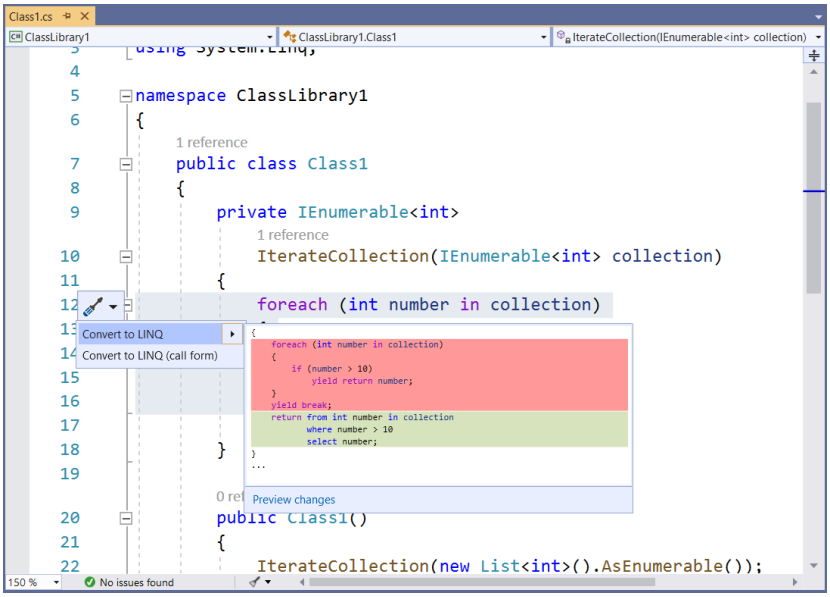


*Figure 32: Refactoring a foreach loop into a LINQ expression*

It's also possible to refactor the same code as a lambda expression based on extension methods, as shown in Figure 33.

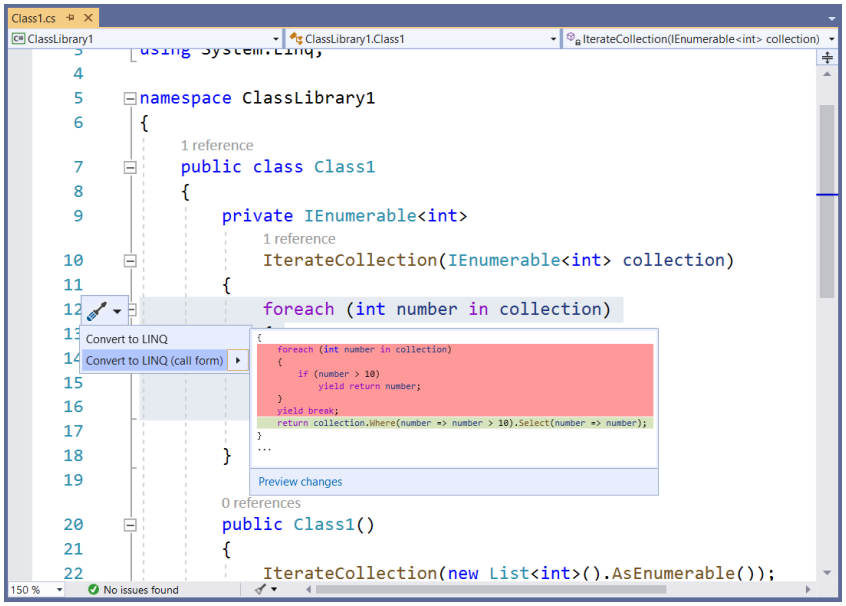


*Figure 33: Refactoring a foreach loop into a lambda expression*

This is a very nice feature, and definitely my favorite one among the new quick actions.

## Promoting class members to interfaces and base types

It is now possible to quickly promote a class member to an interface, if not done previously. For example, consider Figure 34, where an interface called **`IUtils`** and a class called **`Utils`** are defined. **`Utils`** implements **`IUtils`** and defines a method that isn't actually declared in the interface.

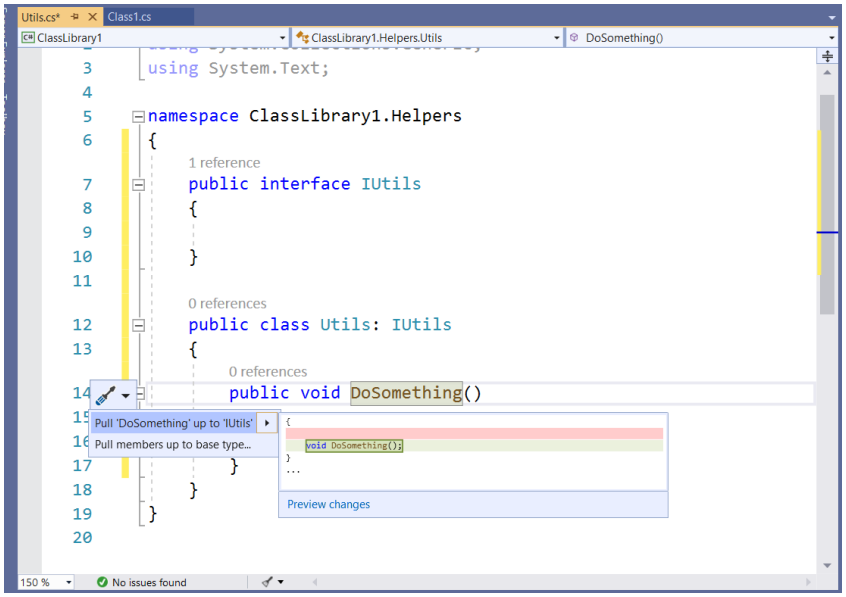


*Figure 34: Promoting class members to interfaces*

If you enable quick actions over the method, you will get an option to pull the method definition to the implemented interface. In the current example, after applying this fix, the **`IUtils`** interface will look like the following.

```
public interface IUtils
{
    void DoSomething();

}
```

This tool not only works with interfaces, but also with base classes from which your classes inherit. You will still see the option to pull the member definition to a specific class, but you will be also able to use the option called **Pull members up to base type** (see Figure 34). Suppose the **`Utils`** class is instead inheriting from a base type called **`BaseUtils`**. If you select this option, a new dialog called Pull Members Up appears, as shown in Figure 35.

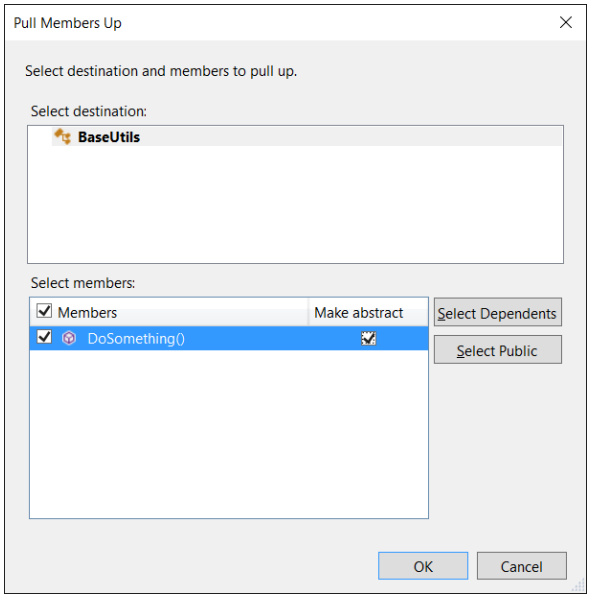


*Figure 35: The Pull Members Up dialog*

Here you can select the destination type. If your class is concurrently inheriting from a bass class and implementing one or more interfaces, you will see the list of types that can receive the member definition. Second, you can select one or more members to be pulled up. You also have the option to mark the member as abstract and to select only public members and dependent members. Simply click **OK** to apply the fix.

## Synchronizing namespace and folder names

It is a common practice to move code files in a project from the root folder to a subfolder or from one subfolder to another. In C#, creating a subfolder also means adding a child namespace to the parent one, so if you move code files between folders, there will be a namespace mismatch between the code file and the new target folder.

Visual Studio 2019 includes a new code fix that will help synchronizing the namespace with the name of the folder that contains the code file. An example will make it easier to understand this. Consider Figure 36: you can see how the active code file's namespace is **`ClassLibrary1`**, but the file is under the Helpers folder.

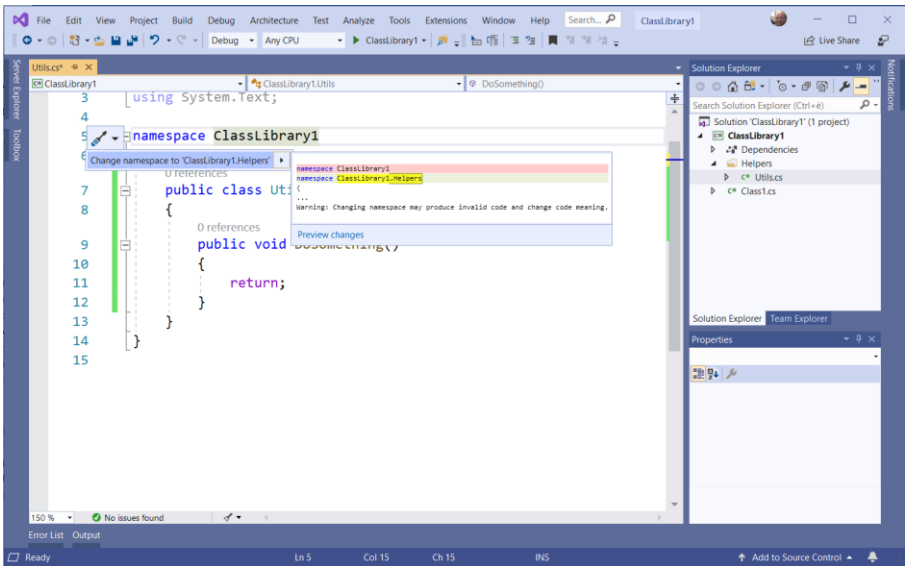


*Figure 36: Synchronizing namespace and folder names*

The Change namespace quick fix will synchronize the namespace with the folder name, changing into **`ClassLibrary1.Helpers`**. The quick fix will also add the proper **`using`** directive to all code files that have a reference to the types defined in the renamed namespace, so that your code will continue to work as expected. This quick fix should not be underestimated—it really helps keep the code in a consistent state.

## Converting anonymous types to tuples

A new refactoring makes it easy to convert an anonymous type into a tuple object. Figure 37 shows an example.

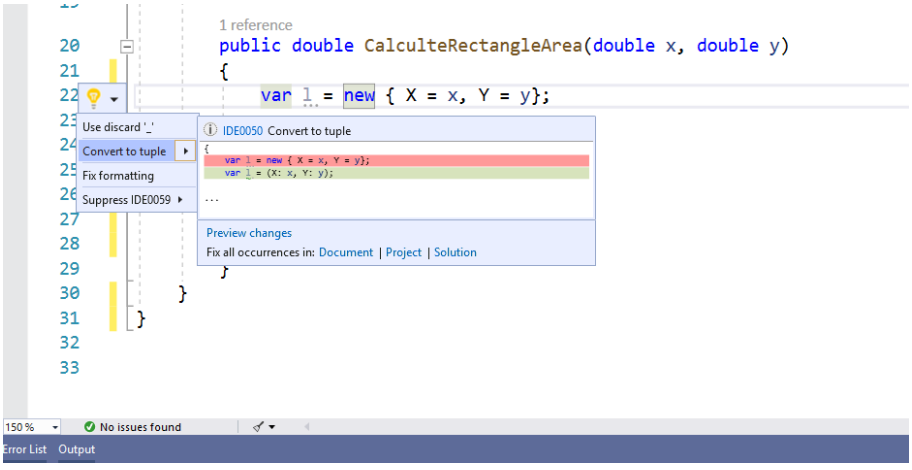


*Figure 37: Converting anonymous types into tuples*

Simply select the **Convert to tuple** suggestion to see a preview of how the anonymous type will be refactored.

***Tip: Tuple types were introduced in C# 7.0. You can read the official documentation for more information.***

## IntelliCode: AI-powered word completion

IntelliCode is an extension to IntelliSense, the well-known and powerful code completion tool available in the code editor. IntelliCode provides context-aware code completion based on artificial intelligence and is available for both Visual Studio and Visual Studio Code. Currently, IntelliCode is available for C#, XAML, TypeScript, JavaScript, and Java, with more enhanced support for C# and XAML.

With IntelliCode, you will see code completion recommendations at the top of the IntelliSense list, marked with a star. Such recommendations are based on thousands of open-source projects on GitHub, each rated with over 100 stars. When combined with the context of your code, the completion list is tailored to promote common practices, and is able to suggest the most likely and most relevant API calls. IntelliCode can also recommend the most suitable argument signature on method definitions. To enable IntelliCode, the first step is installing the IntelliCode extension.

***Tip: Installing the IntelliCode extension is not necessary if you have installed any workloads that include C#, XAML, or C++. In this case, IntelliCode is automatically installed.***

This can be accomplished by selecting **Extensions** > **Manage Extensions** and searching for IntelliCode, as shown in Figure 38.

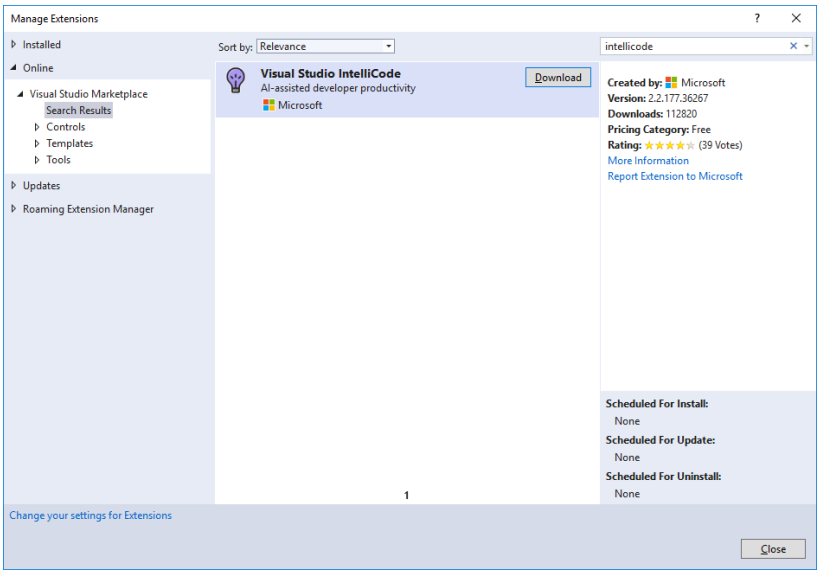


*Figure 38: Installing the IntelliCode extension*

Click **Download**, and when it's finished, click **Close** and close Visual Studio so that the extension gets installed. Restart Visual Studio and open a new or existing C# project. When you type code, IntelliSense will show IntelliCode recommendations at the top of the list, as demonstrated in Figure 39. You can recognize recommendations by a black star icon.

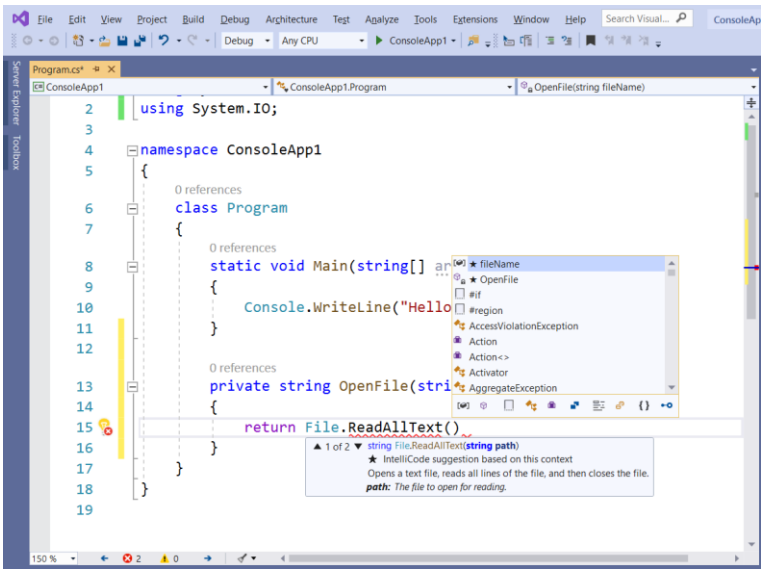


*Figure 39: IntelliSense shows IntelliCode recommendations*

***Tip: If you do not see IntelliCode suggestions, there might be one or more extensions that interact with IntelliSense and override the expected behavior. If this happens, try to disable extensions that might affect IntelliSense and retry.***

Because IntelliCode is based on artificial intelligence for searching and suggesting the most likely patterns related to the context of your code, the more it learns from your coding patterns, the more it will provide appropriate suggestions. You can help IntelliCode learn from your coding patterns by selecting **View** > **Other Windows** > **IntelliCode Model Management**. This will open the Visual Studio IntelliCode page, which you can see in Figure 40.

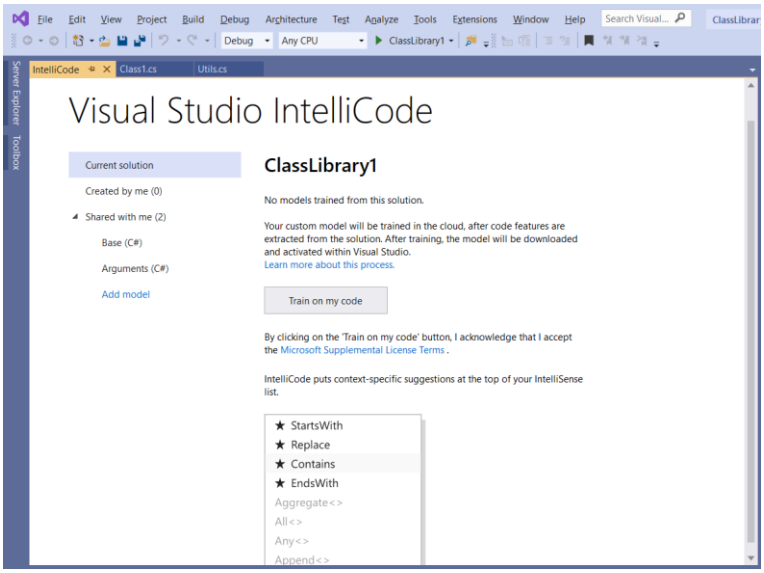


*Figure 40: The Visual Studio IntelliCode management page*

On this page, you can manage the so-called IntelliCode models. A model is a set of coding patterns and rules that can be either of general purpose or based on the context of your code. By default, IntelliCode includes two general purpose models (under the **Shared with me** node). The **Created by me** node contains models generated based on your code. In this case, there are no models yet.

If you click **Train on my code**, IntelliCode will start analyzing your solution, sending a lot of information to an online Microsoft analysis service based on artificial intelligence. This is collected into a new model, which will be available under the **Created by me** node, as you can see in Figure 41.

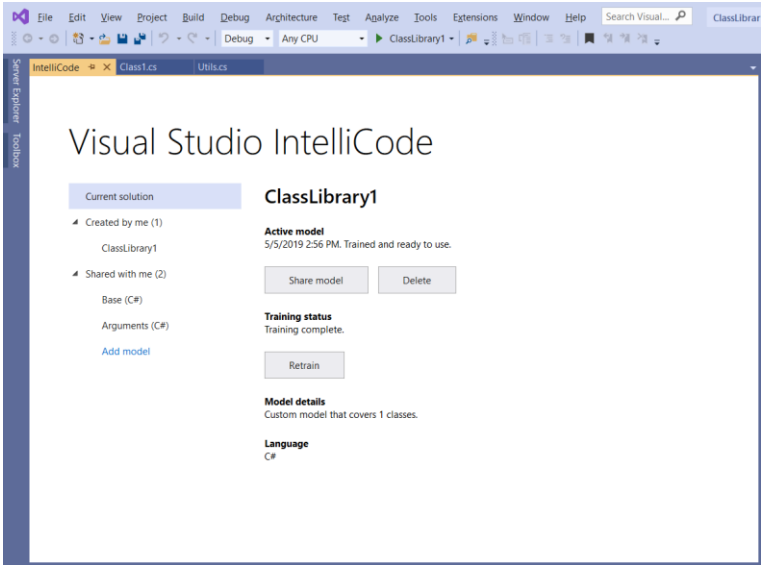


*Figure 41: Code analysis completion*

Now that a new model has been generated, IntelliCode will be able to provide much better suggestions based on your code context. A model can be also shared, deleted, or regenerated using the appropriate buttons in the IntelliCode management page. You can now go back to your code and work as you would normally do. IntelliCode will now be able to provide much better suggestions.

## Chapter summary

Visual Studio 2019 provides many improvements to the code-editing experience. The Document Health Indicator provides a quick view of the code issues in each document, and the Code Cleanup tool allows for one-click code maintenance based on customizable rules.

The Clipboard Ring makes it easy to navigate among items in your clipboard, and Roslyn classification colors improve the readability of your code with better color schemes. New quick actions and refactorings have been added to the Roslyn code analysis engine, and you can now quickly refactor **`foreach`** loops into LINQ and lambda expressions. You can easily promote classes' members into base types and interfaces, and you can quickly synchronize namespace with folder names.

Microsoft is also investing in AI-assisted coding and is working on IntelliCode to provide suggestions based on the most appropriate patterns. In the next chapter, you will read about the improvements that Microsoft has brought to Visual Studio related to team collaboration.

# Chapter 4 Collaboration

Team collaboration is another key area for developers using Visual Studio. For this reason, Microsoft has invested in adding tools that will simplify your teamwork, from source control to integrated live code-sharing. This chapter describes what's new for team collaboration in Visual Studio 2019, starting with updates to Git support, and then walking through online code sharing.

## Git tooling updates

Visual Studio 2019 introduces two important improvements to the integrated tools for Git, the popular source-control engine: stashing changes and supporting pull requests. Both are described in the next paragraphs.

### Support for stashing changes

Git allows for temporarily storing changes to your source code so that you can work on a different task before committing such changes to the repository. This operation is referred to as *stashing changes*.

***Tip: The stash feature has basically the same goal as the Shelve operation in Team Foundation Server.***

Visual Studio 2019 introduces integrated support for stashing in the Team Explorer window. For example, suppose you have some uncommitted changes in your source code. In Team Explorer, you will see a new command called Stash (see Figure 42).

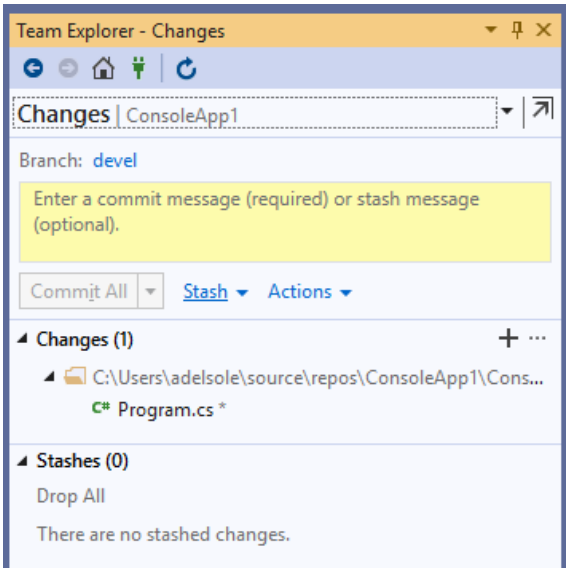


*Figure 42: Support for stashing changes in Team Explorer*

You can quickly click **Stash** > **Stash All** to temporarily store your changes, which will appear as a unique set in the Stashes group at the bottom of Team Explorer (see Figure 43).

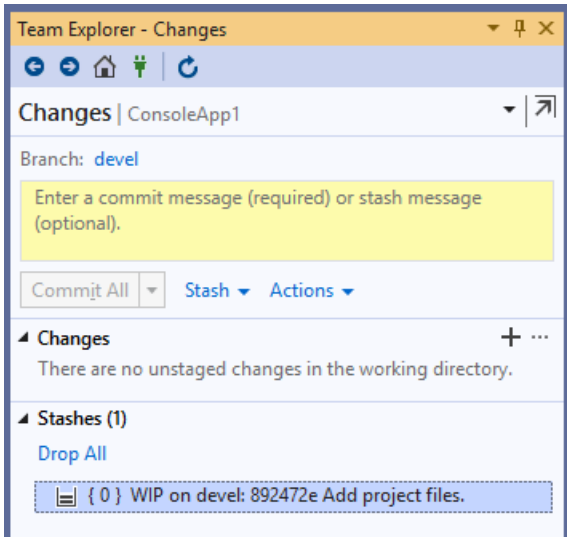


*Figure 43: A stash has been created for your changes*

If you want your stashed changes to be back in your code, you can right-click the stash and select one of the following from the submenu:

- **Apply**: This will bring your changes back to the source code and will not remove the stash, so that it can be saved for later reuse.
- **Pop**: This will bring your changes back to the source code and will also remove the stash.

You can also delete a stash by selecting **Drop** from the contextual menu. Stashing is a very useful feature because it allows for working on different tasks without the need of committing incomplete source code every time.

## Support for pull requests

With pull requests, you can let other developers know about changes to the source code you have pushed to a Git repository, and you can ask for a code review. Visual Studio 2019 is the first edition of the IDE that supports pull requests via an official Microsoft extension, called Pull Requests for Visual Studio, that you can download with the Manage Extensions tool, as shown in Figure 44. This extension also allows for code comparisons between specific branches, and not just between the current code and the latest commit.

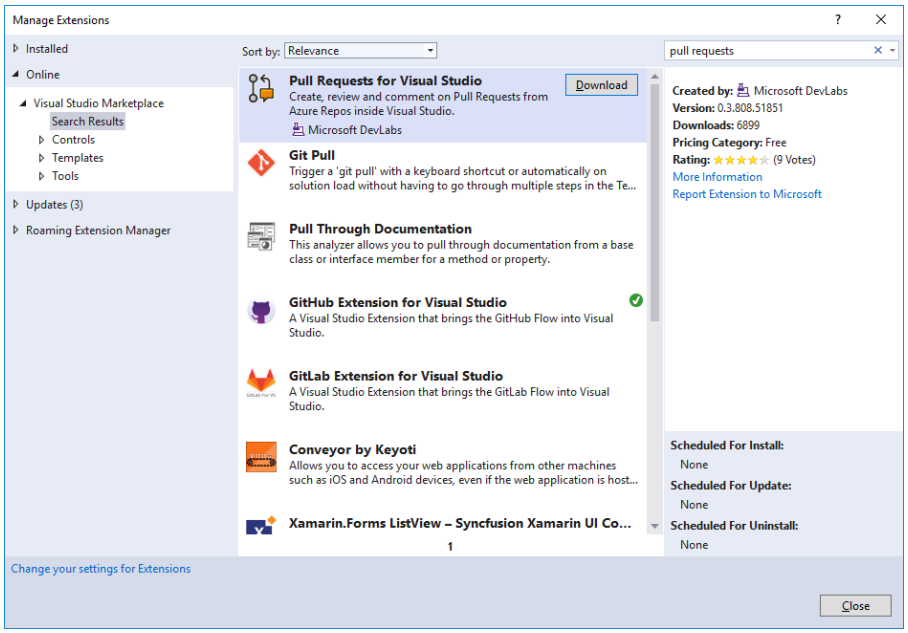


*Figure 44: Downloading the Pull Requests for Visual Studio extension*

Currently, Pull Requests for Visual Studio only supports Git repositories hosted on Azure DevOps. For a better understanding of how this extension works, create a new C# project (any template is fine) and then follow these steps:

1. Add the project to a Git repository via the **Add to Source Control** command at the bottom-right corner of Visual Studio.
2. Create a new branch called **devel**.
3. Make any changes to the code, even simply a comment.
4. Click the **Comparisons** button on the toolbar (highlighted in red in Figure 45).

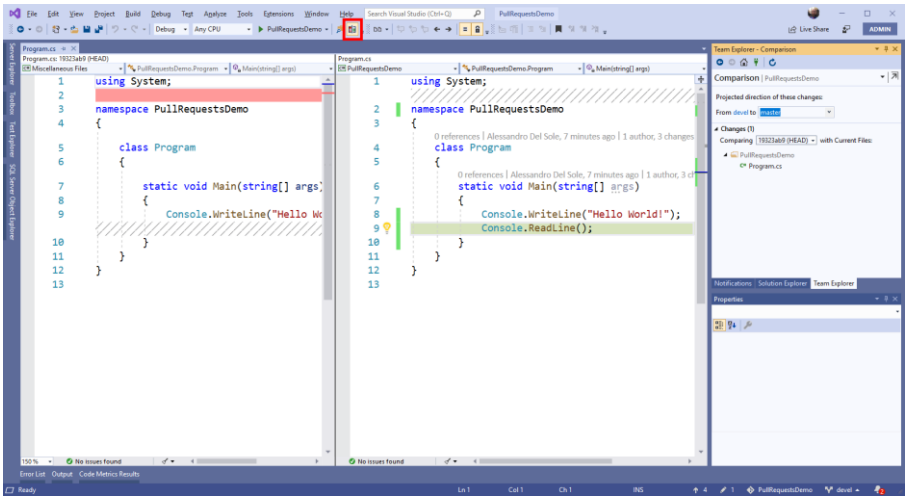


*Figure 45: The Comparisons view*

Team Explorer allows you to select the branch to which the current branch should be compared, and then the code editor can display the differences between files. Green highlights show new lines of code, and red highlights represent code that has been removed.

This view is similar to what you get when you use the Compare with Unmodified source control command, but the difference is that here you can decide which branches to compare, whereas the Compare with Unmodified command compares the current code with the code in the latest commit. This view is interactive, meaning that you can also work inside the current branch's code directly.

To create a pull request instead, commit your code both locally and remotely. To accomplish this, you need an Azure DevOps subscription. Free subscriptions are available for teams with up to five members. Assuming you already have your subscription, push your changes to a remote repository created on Azure DevOps. Then, click the branch name at the bottom-right corner of the IDE and select **New Pull Request**. Team Explorer will now show the New Pull Request user interface (see Figure 46).

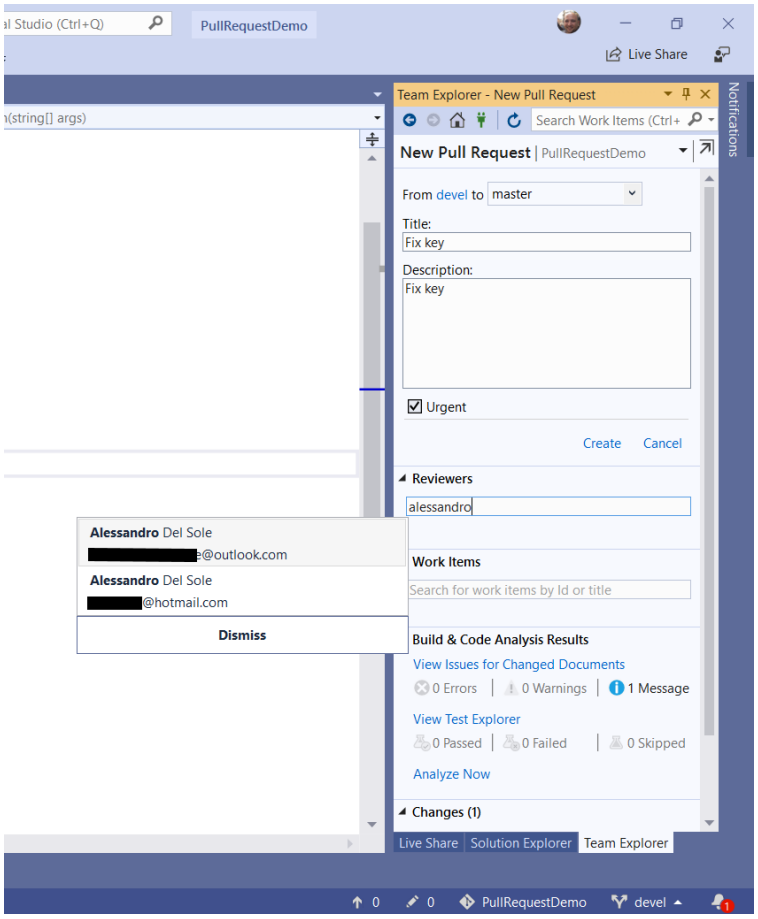


*Figure 46: The New Pull Request user interface*

From top to bottom, you can:

- Specify the pull request's target branch.
- Provide a title and detailed description for the pull request.

- Mark the pull request as urgent. This will highlight the pull request with an [Urgent] tag in the project management page on Azure DevOps.
- Select one or more code reviewers from your team (optional). In Figure 46, you can see how team member names appear as you type. This implies that you (or your project administrator) previously assigned members to the team project on Azure DevOps.
- Link the pull request to one or more work items (optional).
- View code issues and unit test results (if any).

When you're ready, click the **Create** hyperlink. After a few seconds, you will see the pull request in the Requested By Me group of Team Explorer (see Figure 47).

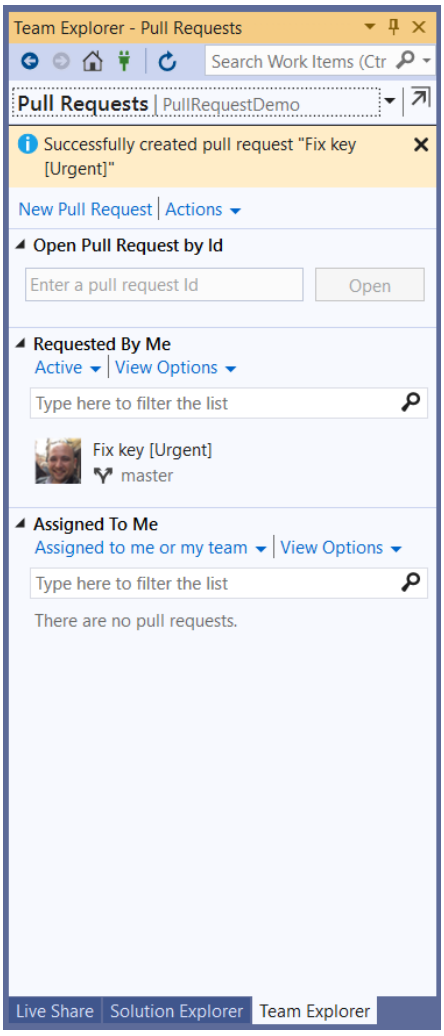


*Figure 47: The new pull request has been created*

***Tip: Assignees of the pull request will be notified via an automatic email. This is the default behavior, and you can disable automatic emails in the Azure DevOps portal.***

The new pull request will be also visible in the Azure DevOps portal. Once you enter, you just need to open the team project, and the pull request will be visible in the project summary. You will also be able to see the code that has been changed in the Files tab, as shown in Figure 48.

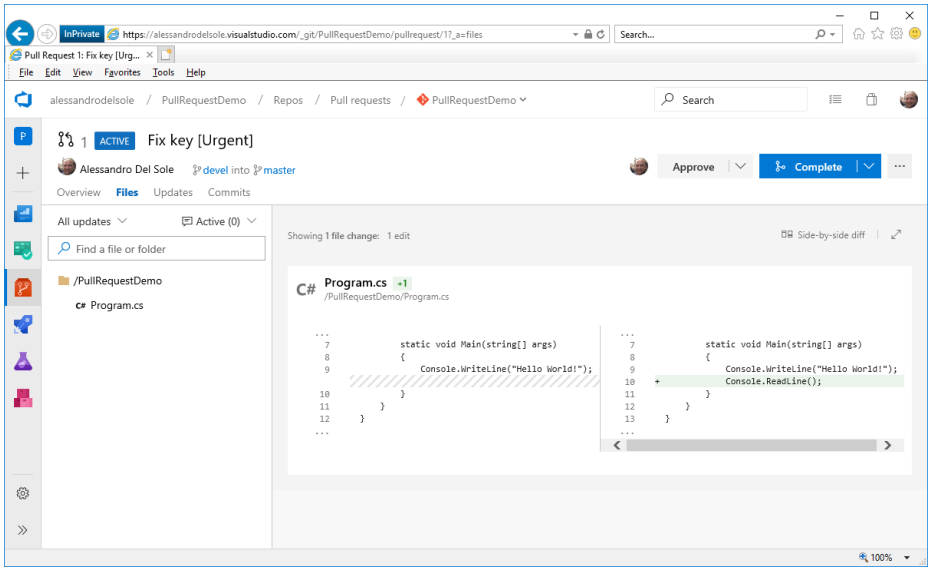


*Figure 48: The pull request details in the Azure DevOps portal*

Pull requests can certainly be approved, rejected, or commented within the portal, but we want to see how this works inside Visual Studio with the Pull Requests extension. Now, suppose you are the assignee of the pull request for code review. When you open Visual Studio, Team Explorer will display the pull request as assigned to you, also showing the name of the person who requested your code review. Figure 49 demonstrates this.

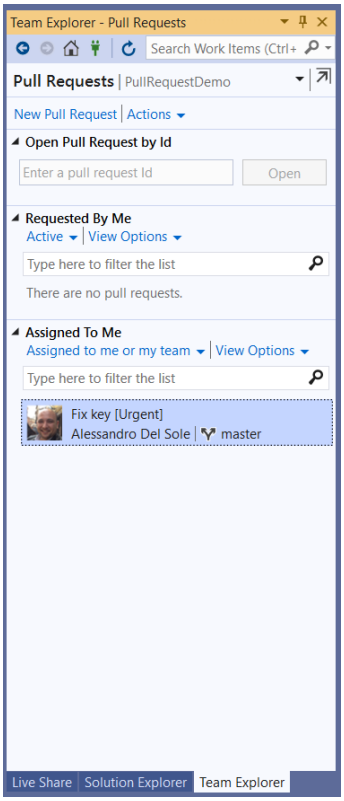


*Figure 49: The pull request assignee sees the pull request and who requested it*

You can open the details view with a simple double-click on the pull request. From the details view (see Figure 50), you will be able to see the description and discussion comments (these typically happen on Azure DevOps).

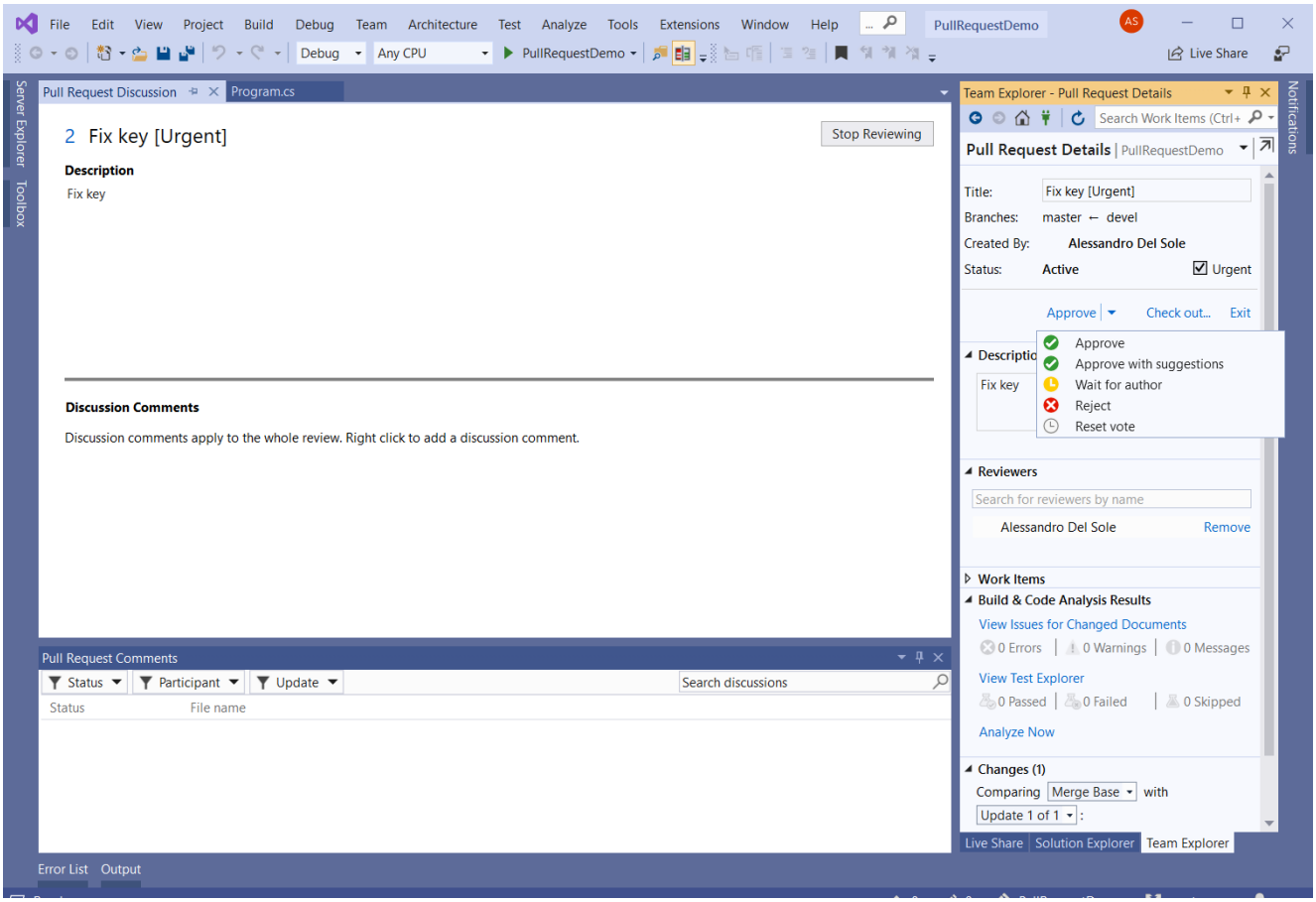


*Figure 50: The pull request details*

If you click **Check out** in Team Explorer, you will be able to download the source code changes for your review. With the Approve menu, you will be able to decide what to do with the pull request after your review, such as approving or rejecting it. The team member who has requested the code review will be notified of the decision you make as a code reviewer, and that decision will be applied to the team project on Azure DevOps.

***Note: As an extension, Pull Requests for Visual Studio is constantly updated. For this reason, it might have additional features when this e-book is released.***

## Collaboration sessions with VS Live Share

Visual Studio 2019 makes it easier to collaborate on source code across distributed and remote teams. This is possible with a new tool called Visual Studio Live Share. This tool is actually an IDE extension that is automatically installed for you at setup time, and it is also available for Visual Studio Code, which means you can work in a collaboration session via different development environments.

The workflow is very simple: you start a collaboration session, and then you send a connection link to attendees you want to invite. A maximum of five participants is currently allowed. You can start a collaboration session by clicking the **Live Share** button, located at the upper-right corner of the IDE, at the left side of the feedback button. Look at Figure 15 as a reference. Starting a collaboration session takes a few seconds. When started, Visual Studio shows an informational page with the session details and automatically copies the invitation link to the clipboard (see Figure 51).

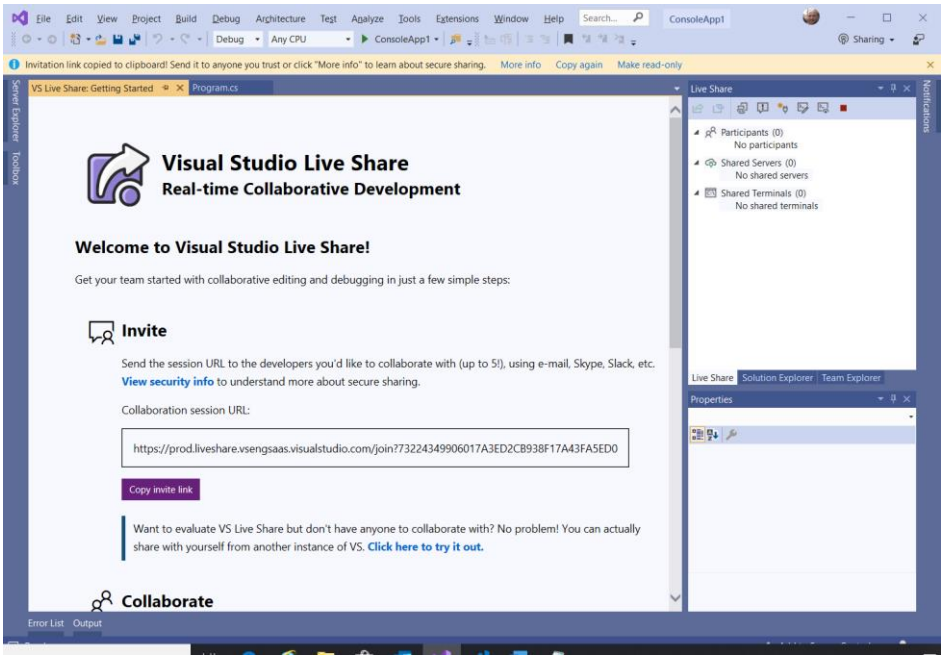


*Figure 51: Live Share's informational page*

***Tip: The informational page will be visible only the first time you create a collaboration session. Should you want to make it visible again, you will need to click the More Info hyperlink in the bar at the top.***

A new docking window called Live Share is also displayed. In Figure 46, it is visible next to Solution Explorer. This window includes some shortcuts that will be discussed in moments. Notice that Live Share does not include integrated chat or call capabilities, so you will need to use a program such as Skype or Skype for Business if you need to talk by voice with the other developers. Extensions for Visual Studio Code exist to add chat and audio capability to Live Share, but not for VS 2019 at the moment.

***Note: You might encounter some issues when running Live Share behind a proxy. If the connection doesn't work, check out the Connectivity documentation page for potential solutions, or contact your network administrator.***

Collaboration starts when other people join the Live Share session. You can send the invitation link using your channel of choice, such as email, Skype, or Slack. Participants can:

- Select **File** > **Join Live Share session** and paste the link in the connection dialog.
- Open the invitation link in their default browser. The system will recognize this as a link that should be opened in Visual Studio 2019 and will ask the user permission for doing this.

Participants will then be automatically entered into the collaboration session, and they will see the Live Share tool window in their instance of Visual Studio 2019. The presenter will see the list of participants in the Live Share tool window (see Figure 52). Attendees can move among source code files via your shared Solution Explorer, and they can directly work on the code, making their edits from their machines. Figure 52 shows how the attendee's email address is displayed in the code editor to highlight that another person is working on a specific code block.

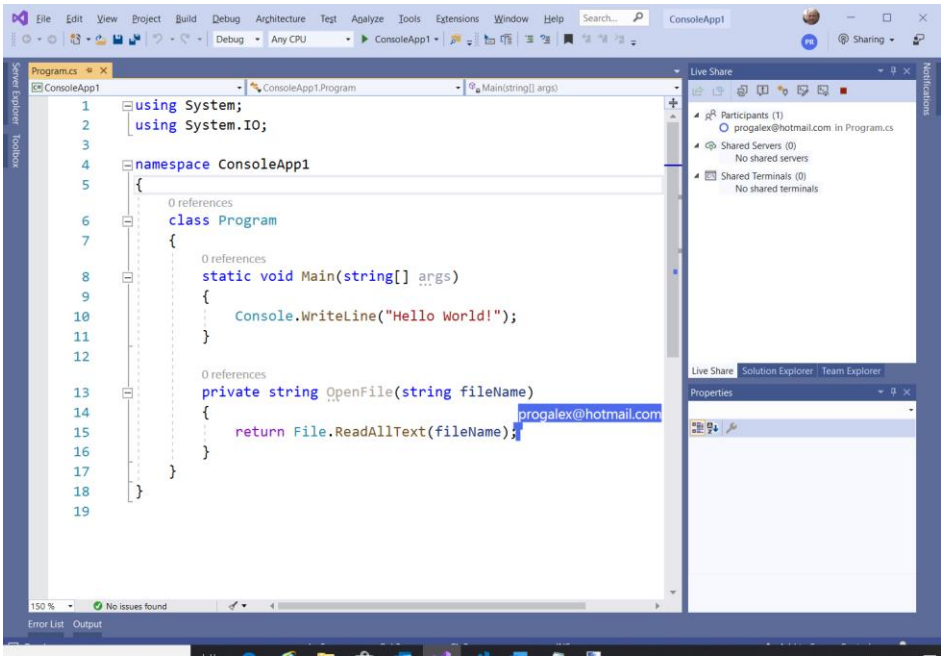


*Figure 52: Collaborating on the source code and viewing the list of participants*

Live Share is particularly useful for pair programming, a well-known approach in the Agile methodology, especially with remote teams. You are not limited to sharing source code: you can also share a terminal window and access to a web server. This can be done via the appropriate buttons on the Live Share window's toolbar. For example, Figure 53 shows an instance of a shared terminal. Notice how the title bar highlights this as a shared instance.

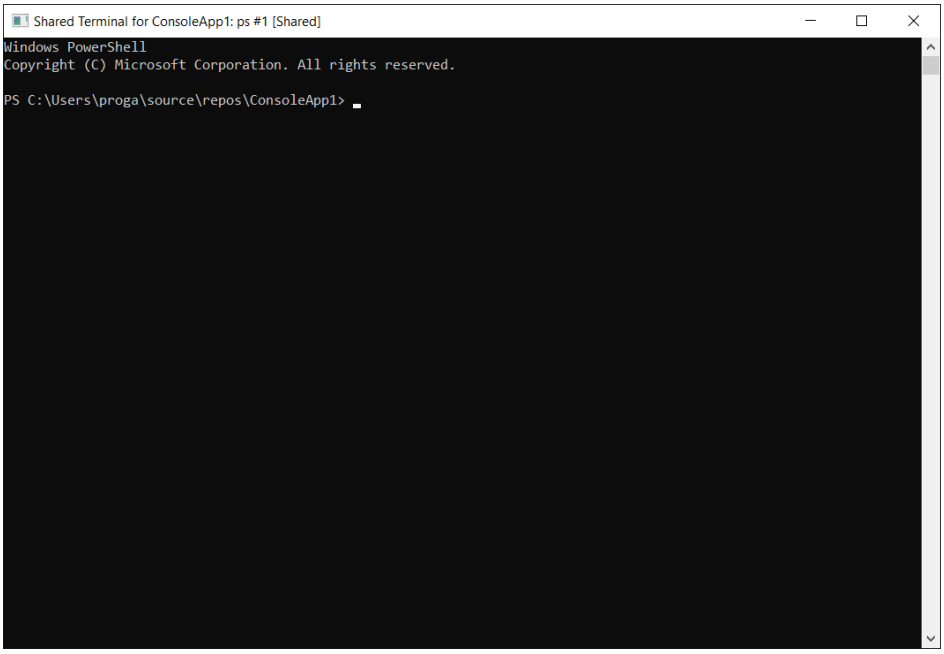


*Figure 53: A shared instance of the terminal*

A shared instance means that a participant can work directly on the terminal window. If you want to avoid this, you can share a read-only terminal instance. Similarly, you can share access to a local server. Figure 54 shows how you can configure sharing by clicking **Add** and then specifying the port number.

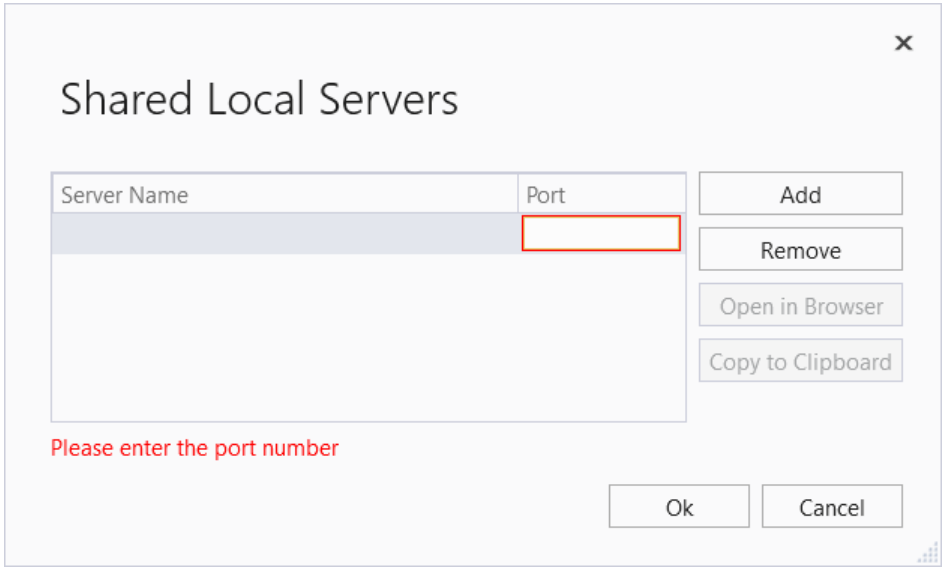


*Figure 54: Sharing a local server*

Once the port number has been specified, the local server will be shared, and participants will be able to access it via the full link that you can quickly generate by clicking **Copy to Clipboard**.

***Tip: It's important to remember that sharing terminals and local servers can be very dangerous. Make sure participants know what they are doing, and set up the proper limitations, as described in the next subsection.***

Both shared terminals and servers will be listed in the Live Share tool window for easy reference, in both the presenter and the participant views. The Stop button in the Live Share's toolbar can be used by the presenter to end the collaboration session, and by the participant to leave the session.

## Live Share configuration

VS Live Share can be configured in many ways, and the most important configuration options are about limiting direct control on your machine when working in a collaboration session. You can access Live Share options via **Tools** > **Options** > **Live Share**. Figure 55 shows the full list of options.

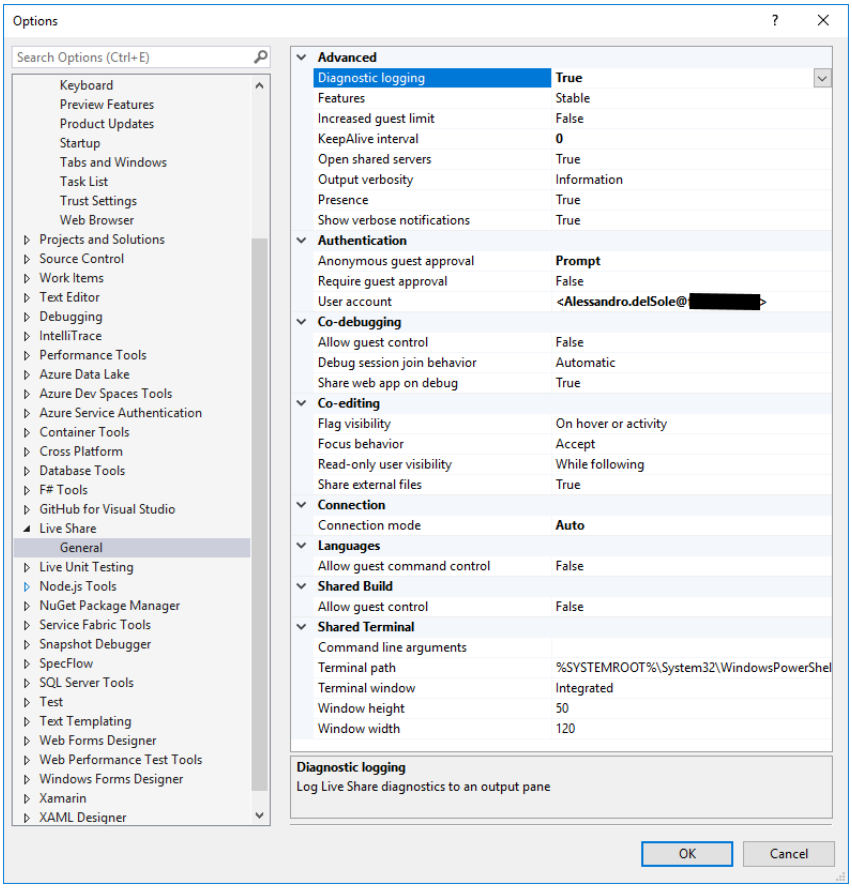


*Figure 55: Configuring VS Live Share*

These options are self-explanatory, so I will not discuss each in detail. It is worth mentioning that, by default, guests cannot control many Visual Studio features. For instance, “Allow guest control” is False for Co-debugging, and “Anonymous guest approval” requires your permission. The description of each option is available at the bottom of the Options dialog when you click one. I recommend you use the default options, because they have a good balance between team collaboration and access limitations.

## Chapter summary

Collaboration is the key to success for modern development teams, and Visual Studio 2019 introduces important new collaboration features. In terms of source control, the IDE now supports stashing changes on Git repositories, and it can also be enabled to support pull requests via the Pull Requests extension, which will allow you to manage pull requests on Git repositories hosted on Azure DevOps.

In terms of real-time collaboration, VS Live Share enables pair programming directly from within the IDE, and developers can join a collaboration session with a simple click. Another key area is crucial in terms of developer productivity: the debugging experience.

The next chapter focuses on what’s new in the debugger, completing the overview about the new productivity features.

# Chapter 5 What's New in the Debugger

Every major release of Visual Studio puts productivity at the core of its improvements, and VS 2019 is no exception. Productivity is not only about coding; it's about all the development activities, and debugging is crucial in an app development lifecycle. For this reason, new features have been introduced to improve your productivity at debugging time. This chapter describes what's new for the Visual Studio debugging experience, with a closer look at new features for .NET Core.

## Searching local variables within debug windows

***Note: This feature is available to all .NET applications.***

Inspecting local variables while debugging has always been one of the most-used features within popular debug windows such as Locals, Autos, and Watch. However, you might have hundreds of local variables, so scrolling the list of locals and focusing on their values can be difficult.

Visual Studio 2019 introduced a search box for local variables in the Locals, Autos, and Watch debug windows. Figure 56 shows an example based on the Locals window, and Figure 57 shows another example based on the Autos window.

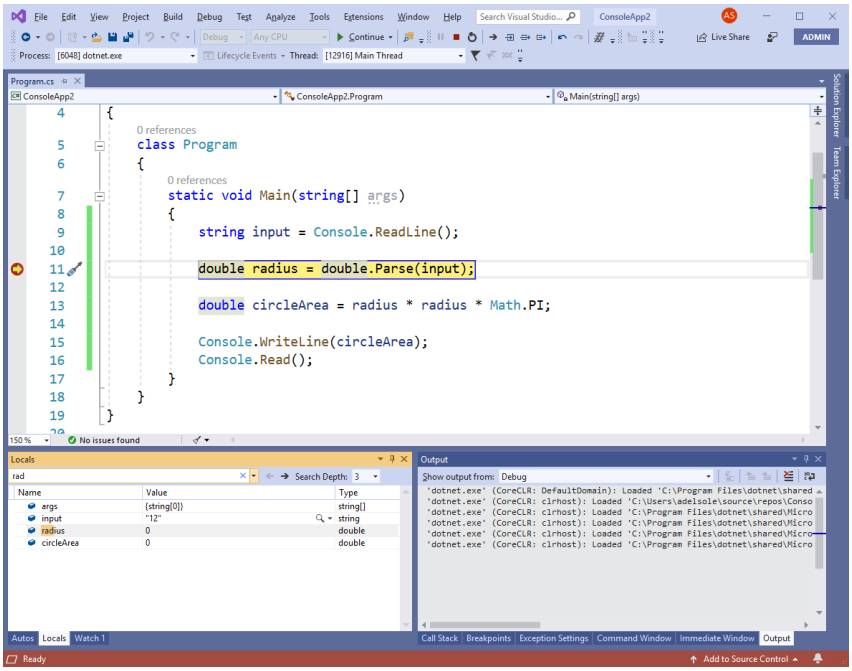


*Figure 56: Searching for local variables in the Locals window*

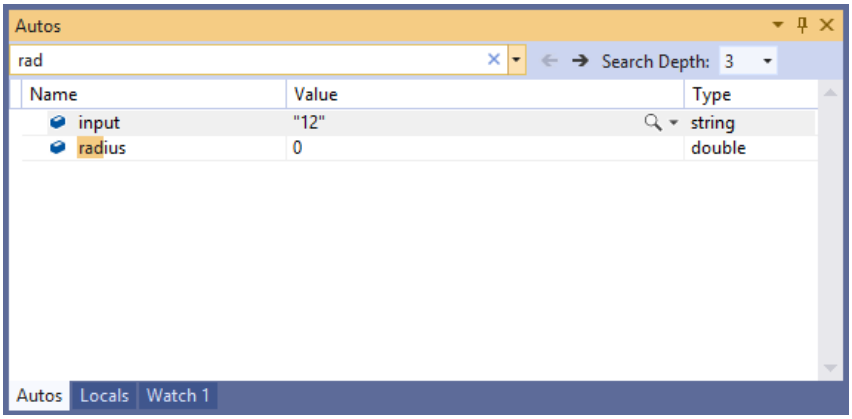


*Figure 57: Searching for local variables in the Autos window*

You can type in the search box and variable names matching your criterion will be highlighted. You can then use the **Find Next** and **Find Previous** buttons (at the right side of the search box) to quickly move between variables. The Search Depth combo box allows you to specify how many locals matching the criterion will be highlighted, with a minimum of 1 and a maximum of 10.

## Introducing data breakpoints

> ***Note: This feature is available only to projects built with .NET Core 3.0 (and later). It is new to .NET Core, but already existed for C++.***

It is very common to have code blocks that are called many times, and you might want to debug only when a certain condition happens. This has always been possible using breakpoint conditions, which you set by right-clicking a breakpoint, selecting **Conditions**, and writing the conditional expression that must be evaluated as true in order to hit the breakpoint. A very common condition is breaking the application execution when the value of a variable changes, so Visual Studio 2019 makes it easier to control this scenario without the need of breakpoint conditions. In fact, you can now right-click a variable name in the **Local** window and then select **Break When Value Changes** (see Figure 58).

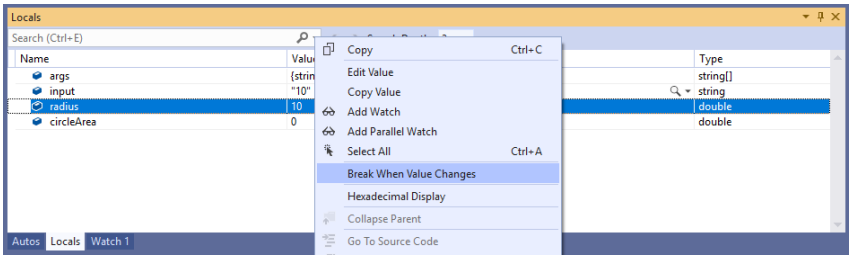


*Figure 58: Using the new Break When Value Changes feature*

Visual Studio will automatically track your variable, and it will break the application execution when the variable value changes. If the variable represents a reference type, the application execution will break when a property referencing the object changes its value. This new feature is referred to as data breakpoints and is also useful when you want to know when something is added or removed from a collection.

For example, setting a data breakpoint on the **`Count`** property of collection classes exposed by the **`System.Collections.Generic`** namespace makes it easy to detect when the collection has changed.

## Formatting locals with format specifiers

Visual Studio 2019 introduces support for the so-called format specifiers inside debug windows that display local variables. To understand how this works, add a local variable to a Watch window while in break mode. Click the variable name and enter a comma when the cursor is available (see Figure 54). You will see a drop-down that contains a list of format specifiers; you can hover over each specifier to see a description. For example, the **nq** format specifier presents a string without quotes in the Value column of the Watch window, as demonstrated in Figure 59.

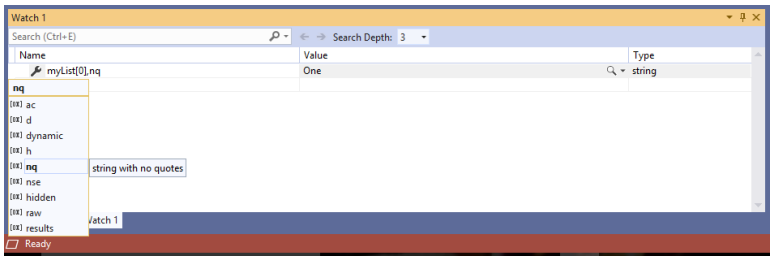


*Figure 59: The list of format specifiers*

Format specifiers are applied according to the object type. The description you get when hovering will help you understand if a specifier can be applied to a local variable. This feature is useful when you have complex data representations, and it is also available to C++.

## DataSet visualizer for .NET Core

The .NET Core API now includes the **`System.Data`** namespace and objects such as **`DataSet`** and **`DataTable`**. Visual Studio 2019 introduces a specific debugger visualizer for **`DataSet`** objects in .NET Core, called DataSet visualizer, which provides a tabular view of the data contained in a **`DataSet`** and its **`DataTable`** objects. Assuming you have a **`DataSet`** populated with some data, and that the application is in break mode, you can hover over a variable of type **`DataSet`** and enable the data tip, as shown in Figure 60.

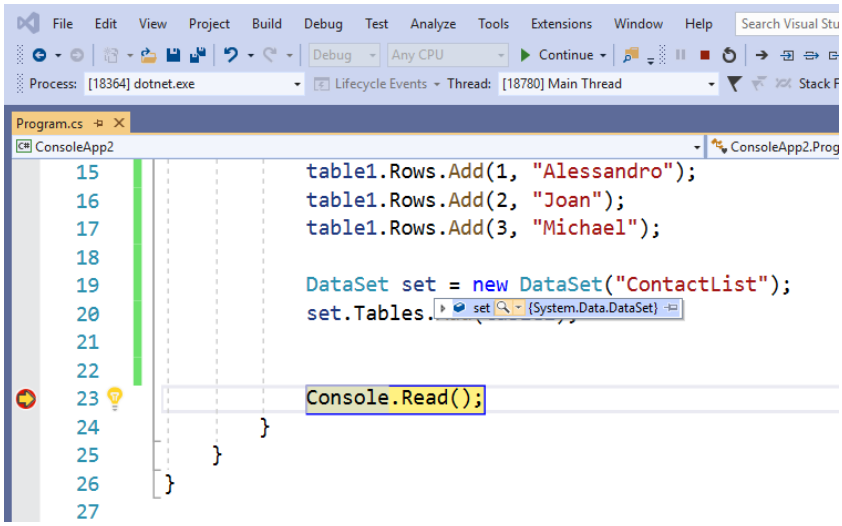


*Figure 60: Enabling the data tip on a **`DataSet`***

If you click the magnifier icon, the DataSet visualizer will appear, and will display a tabular representation with columns and rows, as you can see in Figure 61.

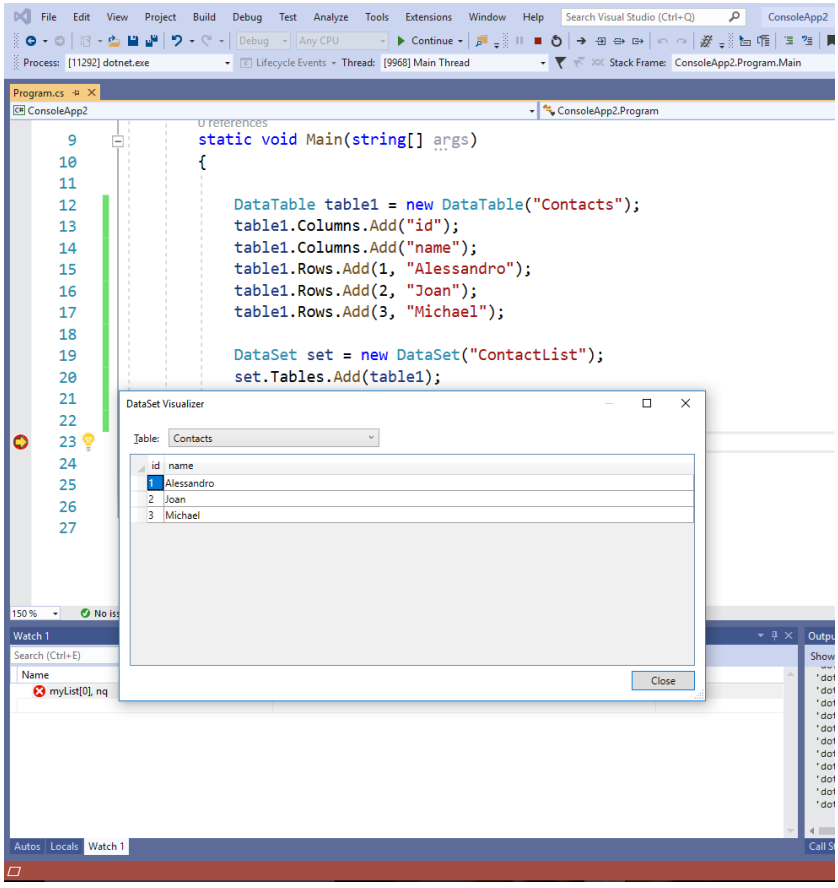


*Figure 61: The DataSet visualizer in action*

You can select a different table by expanding the Table combo box. This is a very useful feature that will help you have a precise representation of your data at debugging time.

## Feature preview: the Time Travel Debugger

Microsoft is working on a new debugging tool called Time Travel Debugger (TTD). The TTD provides the ability to record a web app running on an Azure Virtual Machine and then accurately reconstruct and replay the execution path, with the help of the integrated Snapshot Debugger. I will not cover this feature in detail for the following reasons:

- It is in a preview state.
- It is available only in Visual Studio 2019 Enterprise.
- It only works against code running on Azure virtual machines and, therefore, an Azure subscription with a proper environment setup is required.

You can read more about Time Travel Debugger in this blog post.

## Chapter summary

Powerful debugging tools are very important for developers and Visual Studio has always offered first-class instrumentation. Visual Studio 2019 improves the debugging tools in many ways.

You can now search and filter the list of local variables in the Local, Autos, and Watch windows. These debugging windows now also offer format specifiers, so that you can format a variable value the most appropriate way. .NET Core apps can leverage data breakpoints and break the application execution only when a local variable or an object property changes its value.

Another addition to .NET Core is the DataSet visualizer, which allows you to view the content of an object of type **`System.Data.DataSet`** in a tabular, human-readable way. This chapter completes the overview about IDE productivity improvements. The next two chapters cover topics that are more related to application development.

# Chapter 6 What's New for Mobile Development

Mobile development with C# is one of the areas where Microsoft has made more investments in recent years, starting with the acquisition of Xamarin and continuing with many updates and improvements to the technology. Xamarin is now a consolidated development platform and the technology of choice for thousands of developers.

Visual Studio 2019 continues the path of its predecessor, adding new tools to make the development experience more productive, including support for the next generation of mobile development tools. Additionally, the size of the Xamarin workload in the Visual Studio Installer has been dramatically reduced so that setup is faster and less space on disk is required. This chapter describes what's new for mobile development with Xamarin in Visual Studio 2019, from the IDE point of view.

***Note: Microsoft has made huge investments in improving Xamarin performance. This includes not only the development tools, but also the build processes and simulators. Specific performance improvements will not be covered in this chapter, but you will immediately notice how much faster the process is, from project creation to debugging and deployment.***

## Creating mobile projects with Xamarin

In Chapter 2, you saw the new user interfaces for creating projects. When it comes to mobile apps with Xamarin, you can filter the list of project templates by project type and select **Mobile**. Figure 62 shows how the list of Xamarin projects appears at this point.

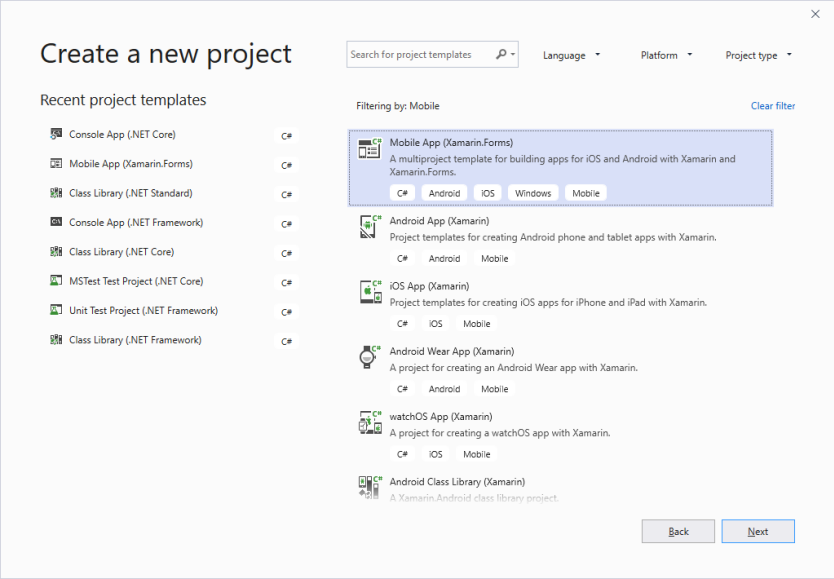


*Figure 62: Getting the list of Xamarin project templates*

The list of templates, which you will need to scroll down to see in its entirety, includes both Xamarin.Forms and Xamarin native project templates. Xamarin.Forms deserves a few more words, since Microsoft is hugely investing in this platform.

## Platform support and code-sharing strategies

If you select the Mobile App (Xamarin.Forms) project template (see Figure 57) and click **Next**, you will be asked to specify the target folder on disk, and then you will be asked to select a specific template, as you can see in Figure 63.

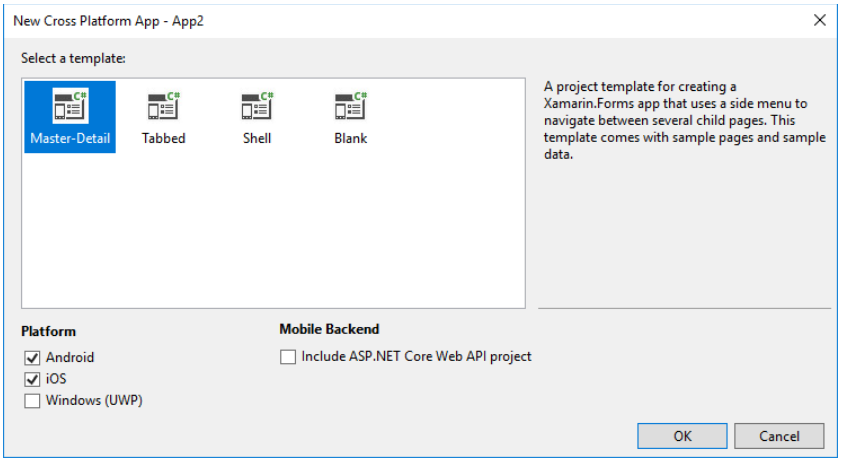


*Figure 63: Xamarin.Forms project templates*

Table 2 describes Xamarin.Forms templates in more detail.

*Table 2: Xamarin.Forms project templates*

| Xamarin.Forms project templates | |
|---|---|
| Master-Detail | Create a mobile app with a side menu with built-in navigation to several child pages. Sample pages and sample data are included. |
| Tabbed | Create a mobile app that uses tabs to navigate among several child pages. Sample pages and sample data are included. |
| Shell | Create a mobile app that uses the new Xamarin.Forms Shell to hold child pages. |
| Blank | Create an empty Xamarin.Forms project. |

Except for the Blank template, the other templates also include an option to generate an ASP.NET Core Web API project that will serve as the mobile back end. It is important to note that Visual Studio 2019 generates new Xamarin.Forms projects based only on the .NET Standard code-sharing strategy, so you will no longer choose between .NET Standard and shared projects, as you did with Visual Studio 2017. This is a very important change that will make your mobile projects easier to maintain in the future.

***Note: When you create a Xamarin.Forms project, Visual Studio 2019 will automatically use the latest stable release of the library. At this writing, the latest stable release for Xamarin.Forms is 4.0.0.48.2894.***

## Shell template for Xamarin.Forms

The Shell is a new feature in Xamarin.Forms 4.0 that reduces the complexity of a project by providing the fundamentals that most mobile applications require. The Shell includes a common navigation experience, a URI-based navigation scheme, and an integrated search handler. Apps built upon the Shell can easily include flyout menus and tabs. Writing code using the Shell is not in the scope of this e-book, so you can read all the details in the official documentation. Here, it's important to note that Visual Studio 2019 introduces a specific project template for Xamarin.Forms called Shell, shown in Figure 64.

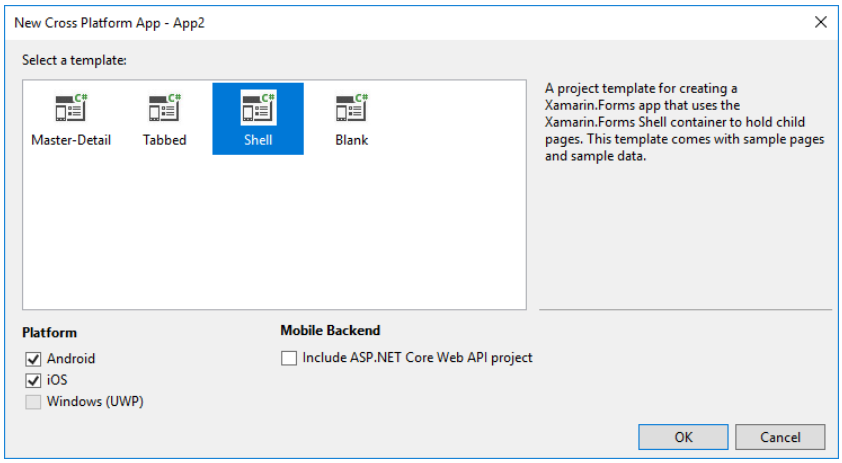


*Figure 64: The Shell project template*

This template generates a project based on the Model-View-ViewModel (MVVM) architecture, with data objects, pages, and view models. The core of the generated code is in the AppShell.xaml file, whose content is included in Code Listing 1 for your convenience (comments in the original code have not been included).

*Code Listing 1*

```
<?xml version="1.0" encoding="UTF-8"?>
<Shell xmlns="http://xamarin.com/schemas/2014/forms"
       xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
       xmlns:local="clr-namespace:App1.Views"
       RouteHost="companyname.com"
       RouteScheme="app"
       Route="App1"
       FlyoutBehavior="Disabled"
       Title="App1"
       x:Class="App1.AppShell">

    <Shell.Resources>
        <ResourceDictionary>
            <Color x:Key="NavigationPrimary">#2196F3</Color>
            <Style x:Key="BaseStyle" TargetType="Element">
                <Setter Property="Shell.ShellBackgroundColor"
                        Value="{StaticResource NavigationPrimary}" />
                <Setter Property="Shell.ShellForegroundColor"
Value="White" />
                <Setter Property="Shell.ShellTitleColor" Value="White" />
                <Setter Property="Shell.ShellDisabledColor"
Value="#B4FFFFFF" />
                <Setter Property="Shell.ShellUnselectedColor"
Value="#95FFFFFF" />
                <Setter Property="Shell.ShellTabBarBackgroundColor"
                        Value="{StaticResource NavigationPrimary}" />
```

```
                <Setter Property="Shell.ShellTabBarForegroundColor"
Value="White"/>
                <Setter Property="Shell.ShellTabBarUnselectedColor"
                        Value="#95FFFFFF"/>
                <Setter Property="Shell.ShellTabBarTitleColor"
Value="White"/>
            </Style>
            <Style TargetType="ShellItem" BasedOn="{StaticResource
BaseStyle}" />
        </ResourceDictionary>
    </Shell.Resources>

    <!-- Your Pages -->
    <ShellItem>
        <ShellSection Title="Browse" Icon="tab_feed.png">
            <ShellContent ContentTemplate="{DataTemplate
local:ItemsPage}" />
        </ShellSection>
        <ShellSection Title="About" Icon="tab_about.png">
            <ShellContent ContentTemplate="{DataTemplate
local:AboutPage}" />
        </ShellSection>
    </ShellItem>
</Shell>
```

This XAML file contains the basic structure for an app built upon the Shell and demonstrates how easy it is to use this feature. You will work on your content pages as you are already used to doing, but the Shell will simplify the project structure by doing most of the work.

## Xamarin.Forms improvements

Visual Studio 2019 introduces some improvements to the existing development tools for Xamarin.Forms and, since version 16.1, it has supported Xamarin.Forms 4.0. More specifically, you can now select a device factor when previewing the result of your XAML, and you can now edit property values via the Properties window. The next paragraphs describe these additions in detail.

***Tip: Remember that the XAML code editor can also benefit from IntelliCode.***

### Xamarin.Forms Previewer updates

Visual Studio 2017 introduced the Xamarin.Forms Previewer, an integrated tool that allows you to see the result of your XAML code without deploying the app to a device, for both iOS and Android form factors. Visual Studio 2019 brings two interesting updates to the Previewer:

- You no longer need to build the project to see the preview. Simply enable the Previewer to see your changes.
- You now have the option to display the preview based on different devices.

***Tip: You can show the Xamarin.Forms Previewer by clicking the Design icon at the upper-right corner of the XAML code editor. Additionally, you can control the appearance of both the Previewer and the XAML code editor via Tools > Options > Xamarin > Xamarin.Forms Previewer.***

To demonstrate these two interesting updates, simply create a new Xamarin.Forms project. The next figures are based on the Shell template. Double-click a XAML page in Solution Explorer, and when the code editor is displayed, click the **Design** icon at the top-right corner. You will immediately notice that the Xamarin.Forms previewer will be able to render the preview without prebuilding the project. You will also see a combo box that shows a list of devices. Figure 65 shows how this appears for Android, and Figure 66 shows how this appears for iOS.

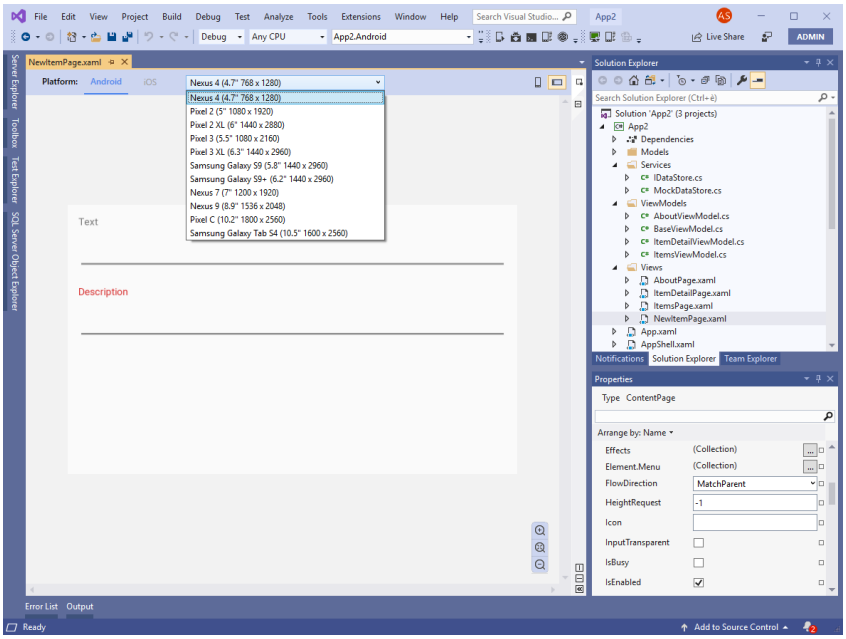


*Figure 65: Selecting a device factor on Android*

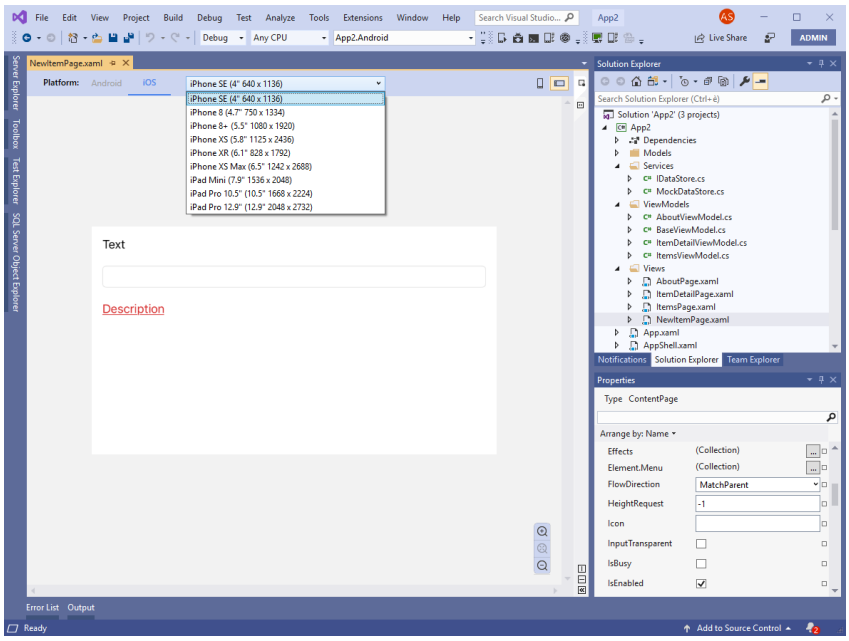


*Figure 66: Selecting a device factor on iOS*

These are very nice additions because they will help you understand how your code is rendered on both platforms at design time without the need to rebuild and re-deploy the app every time.

> ***Note: You might still need to rebuild the project before you see your changes on the Previewer. This might happen if you have custom renderers or third-party controls. The IDE will tell you when the project needs to be rebuilt.***

## Managing objects with the Properties window

In Xamarin.Forms, you can now quickly edit object properties via the Properties tool window. This has always been possible with other development technologies, and now it is for Xamarin.Forms, too. All you have to do is click inside an object declaration in the XAML code editor, and the Properties tool window will allow you to view current object properties and assign their values, as shown in Figure 67, which provides an example based on the **`Label`** control.

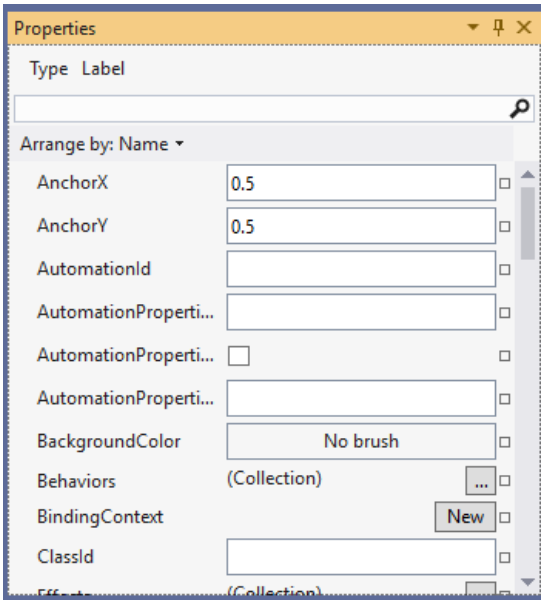


*Figure 67: Editing object properties via the Properties window*

For some specific types, the Properties window provides specialized editors. For example, you can edit properties of type **`Color`** with a convenient color picker, as shown in Figure 68.

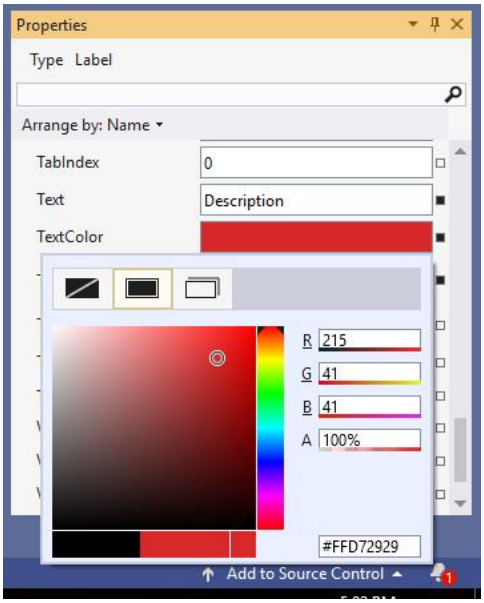


*Figure 68: Editing object properties via the Properties window*

Currently, you can only enable editing properties in the Properties window by selecting objects in the XAML code editor. Clicking objects in the Xamarin.Forms Previewer does not enable selection, and therefore, the Properties window is not activated.

## Xamarin.Android improvements

Specific investments have been made in the Xamarin.Android development experience. Here is a summary of what's new:

- There is a simplified way to create emulator images via the new Create Android Emulator button directly in the Run drop-down.
- Visual Studio can now determine the best emulator image based on your system configuration (for example, if the Intel HAXM acceleration drivers are installed).
- Fast Deployment, a mode used to package assemblies for debugging, is now even faster.
- When working with the designer on .axml (Android XML) files, you can now drag views from the toolbox onto the code editor and generate the related code.
- The XML editor for the user interface files now displays an integrated preview for attributes that contain color values (or references to color values), as you can see in Figure 69. Additionally, you can hover over the attribute and a tooltip will display the color information.


*Figure 69: Color preview in the XML editor for Android*

***Note: There are no significant updates for Xamarin.iOS in Visual Studio 2019 at the moment. Xamarin.iOS updates depend on Apple SDKs updates, so make sure to check the release notes of future VS releases to see if new features are available.***

## Chapter summary

Mobile app development is crucial to Microsoft, and Visual Studio 2019 makes several steps forward to support C# developers who use Xamarin for their business. The new version of the IDE has new and updated, yet simplified, project templates. It brings improvements to the Xamarin.Forms experience with support for multiple form factors in the Previewer, and with the option to edit property values in the Properties window.

For Xamarin.Android, several improvements are also available for the designer, emulators, and the overall experience. Mobile is certainly important, but so is web: in the next chapter, you'll see what's new for web and cloud development in VS 2019.

# Chapter 7 What's New for Web and Cloud Development

Web and cloud development go hand in hand, and Visual Studio 2019 enhances support for cross-platform web development with .NET Core and integrates new scenarios and tools based on Microsoft Azure. This chapter provides guidance on how to leverage the new tools available for web development with Visual Studio 2019, with a strong focus on ASP.NET Core and Azure integration.

## Creating web projects

The new user interface for creating projects in Visual Studio 2019 makes it easier to discover templates for web applications. You can simply filter the list by selecting **Web** in the **Project Type** combo box (see Figure 70).

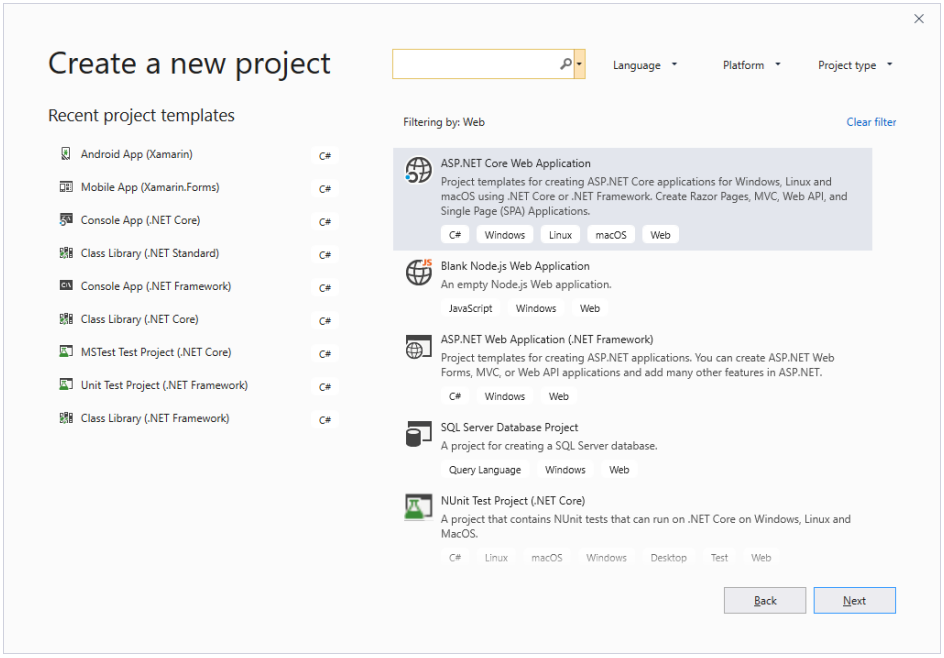


*Figure 70: Creating web projects*

You can refine filtering by selecting a specific language or platform, but this is the way you will find the list of project templates for the web, including ASP.NET, Node.js, and ASP.NET Core. ASP.NET and ASP.NET Core deserve some additional explanation, expressed in the next paragraph.

## Creating ASP.NET Core projects

If you select the **ASP.NET Core Web Application** project, you will be asked to select a more specific template. Figure 71 demonstrates this.

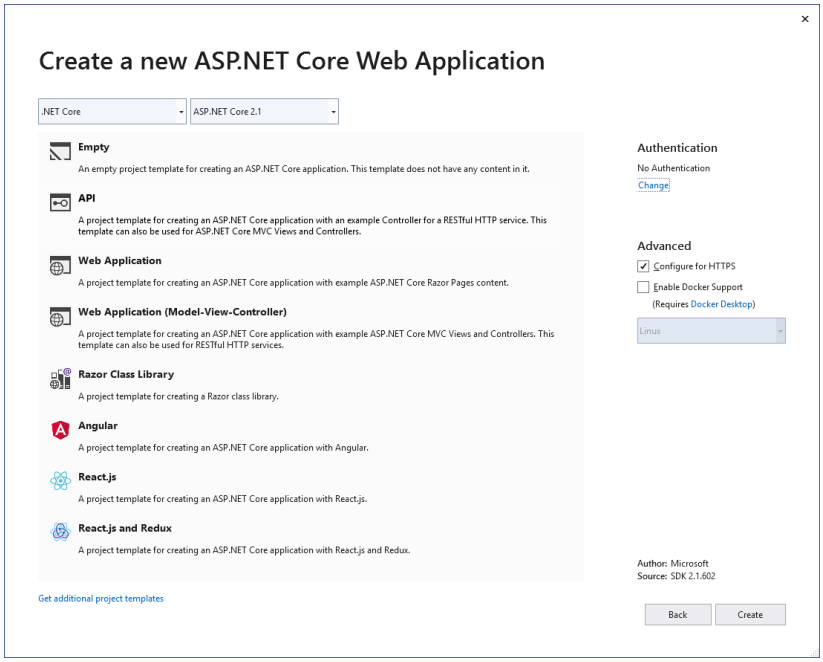


*Figure 71: Creating an ASP.NET Core web project*

While the list of project templates is self-explanatory, for .NET Core, it's worth mentioning that:

- You can quickly enable Docker support (this must be installed on your machine).
- For the API and all the web application projects, you can easily implement authentication. Supported providers are Microsoft Azure, Office 365, and Windows authentication.
- You are not limited to pure C# projects, since you can use different front-end technologies such as React.js and Angular.

Notice how you can select a different version of ASP.NET Core at the top of the dialog (which will also be useful if you install the .NET Core 3.0 SDK). It is important to note that you can also switch between .NET Core and .NET Framework web projects by simply selecting the desired technology from the combo box at the top-left corner of the dialog. For .NET Core, Visual Studio 2019 introduces several productivity improvements, described in the next section.

# Productivity improvements for .NET Core

Most of the improvements that Visual Studio 2019 brings to web development are related to .NET Core. Since the focus on cross-platform development is at the heart of Microsoft's vision, this totally makes sense. This section describes the most relevant productivity improvements that Visual Studio 2019 introduces to .NET Core development.

## Supporting .NET Core 3.0

Microsoft is working hard on .NET Core 3.0, the next major version of the popular cross-platform technology. At this writing, .NET Core 3.0 is available in preview, and the SDK can be downloaded from the .NET Core download page. If you wish to give it a try or to work with .NET Core 3.0 in Visual Studio, version 2019 is a requirement because VS 2017 will not support .NET Core 3.0. Once you download and install the SDK, you will see .NET Core 3.0 listed as an option in the project creation user interface.

## CodeLens and Find All References for Razor files

A specific code editor improvement allows CodeLens and Find All References to include any references to objects declared inside Razor (.cshtml) files. To understand how this works, simply create a new ASP.NET Core project using the Web Application (Model-View-ViewController) template. Locate and open the **ErrorViewModel.cs** file (under the **Models** subfolder). If you click the number of references for the **`RequestId`** property (see Figure 72), not only will you see references to this property in other C# code files, but also one reference in the Error.cshtml file.

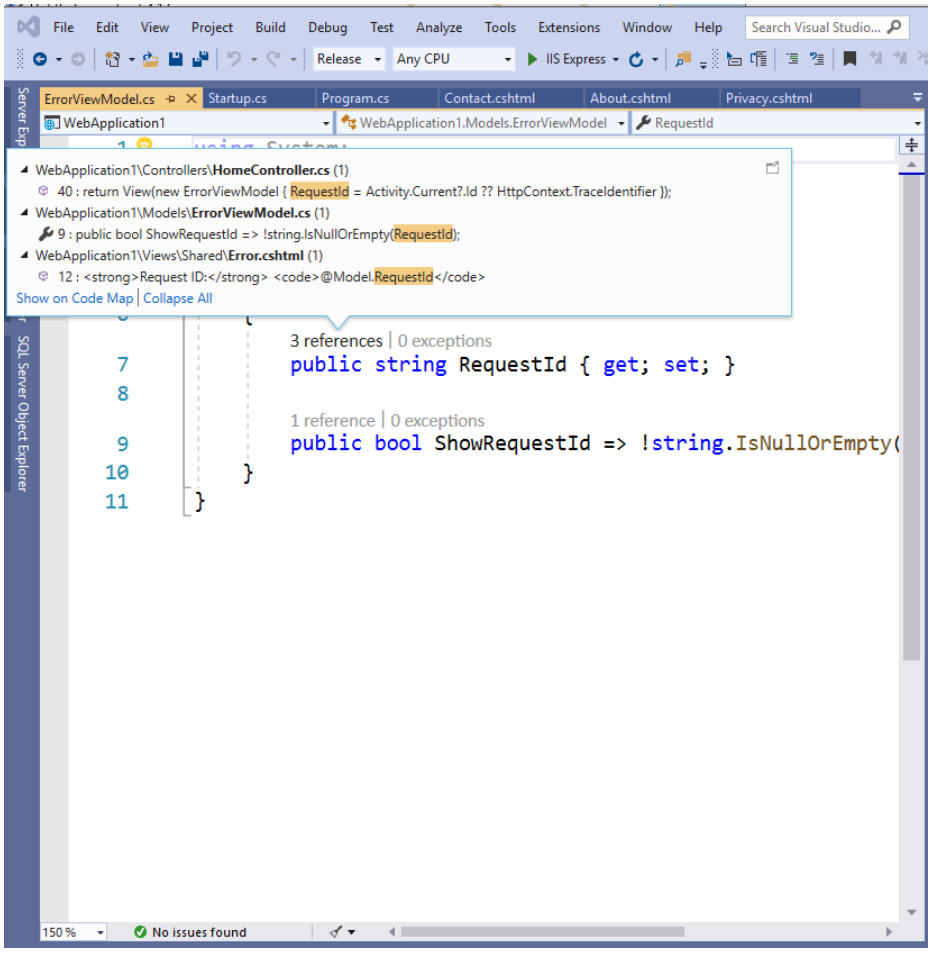


*Figure 72: CodeLens displaying references inside Razor files*

You will also be able to see references from Razor files inside the results view of the Find All References window. This is an extremely useful addition, especially with Razor files that use model binding.

## Calculating code metrics

Code metrics are a set of software measures that provide developers better insight into the code they are developing. With code metrics, developers can understand how reliable and maintainable the code is. Code metrics have always been part of the analysis tools in Visual Studio, but now they are also available to .NET Core applications (and not just ASP.NET Core).

With a .NET Core solution opened, select **Analyze** > **Calculate Code Metrics**. Then you can decide to analyze the entire solution or an individual project. Once the analysis is complete, you will see the results in the Code Metrics Result tool window (see Figure 73).

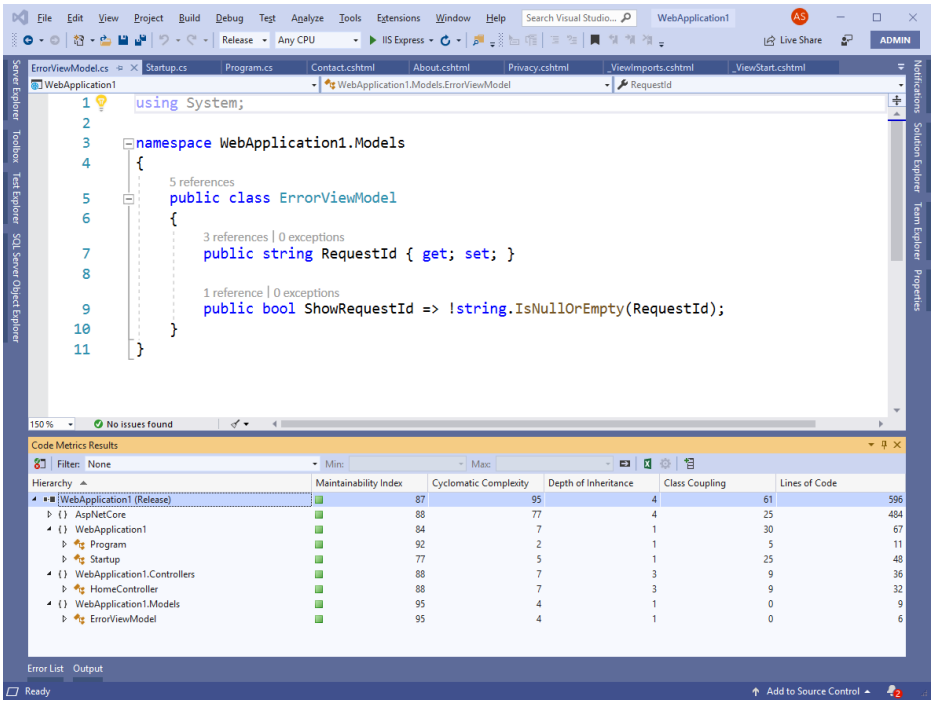


*Figure 73: Code Metrics results*

Table 3 describes code metrics in more detail.

*Table 3: Code metrics in Visual Studio*

| **Code metrics in Visual Studio** | |
|---|---|
| Maintainability Index | Measures the ease of maintainability of the code with an index value. If such index is between 20 and 100, maintainability is good and a green indicator is displayed. If it is between 10 and 19, maintainability is moderate and an orange indicator is |

| Code metrics in Visual Studio | |
|---|---|
| | displayed. If it is between 0 and 9, maintainability is poor and a red indicator is displayed. |
| Cyclomatic Complexity | Measures the code complexity by calculating different code paths in the program flow. For example, several nested **`for`** loops or nested **`if`** blocks might increase the cyclomatic complexity. |
| Depth of Inheritance | Determines the number of class definitions that extend the root of a class hierarchy. A high value might make it difficult to understand the inheritance level and might make the code more difficult to maintain. |
| Class Coupling | Measures interdependencies of a class on other types. For this measurement, Visual Studio considers parameters, local variables, return types, method calls, base classes, interface implementations, fields defined on external types, and even attribute decoration. |
| Lines of Code | Measures the approximate number of lines of IL code. A high value might indicate that a method is doing too much work. Notice that this index is not related to C# or Visual Basic source code, but the Intermediate Language code that .NET generates at compile time. |

You can find more detailed information about code metrics in the official documentation. Remember that code metrics were already available to all .NET solutions, and that support for .NET Core is an addition.

***Tip: Code metrics results can be exported to Excel with the Open List in Microsoft Excel button on the Code Metrics Result window's toolbar.***

## Analyzing CPU performance

Another addition to the .NET Core tooling is the possibility of executing the CPU Usage analysis tool included in the Performance Profiler hub. This tool is useful for understanding where the CPU is spending more time in executing your code. Like code metrics, CPU Usage has been available for several years to other .NET applications, and now is finally available in .NET Core. Its usage is simple. First, make sure the build configuration is **Release** and select **Analyze** > **Performance Profiler**. When the user interface for the Performance Profiler appears (see Figure 74), select the **CPU Usage** option.

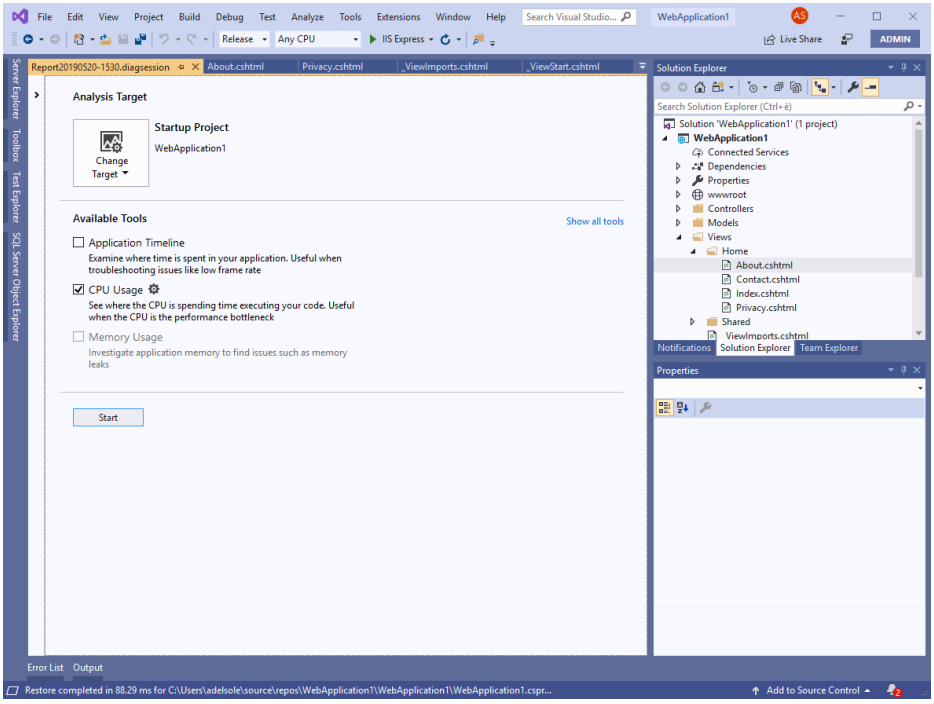


*Figure 74: Enabling CPU Usage as an analysis tool*

When you're ready, click **Start**. The application will start (without an instance of the debugger), and VS will profile it during the execution. You can simply close the application to stop profiling, or you can click the **Stop collection to view CPU usage data** hyperlink in Visual Studio. After a few seconds, Visual Studio will show the analysis results in a visual report (see Figure 75).

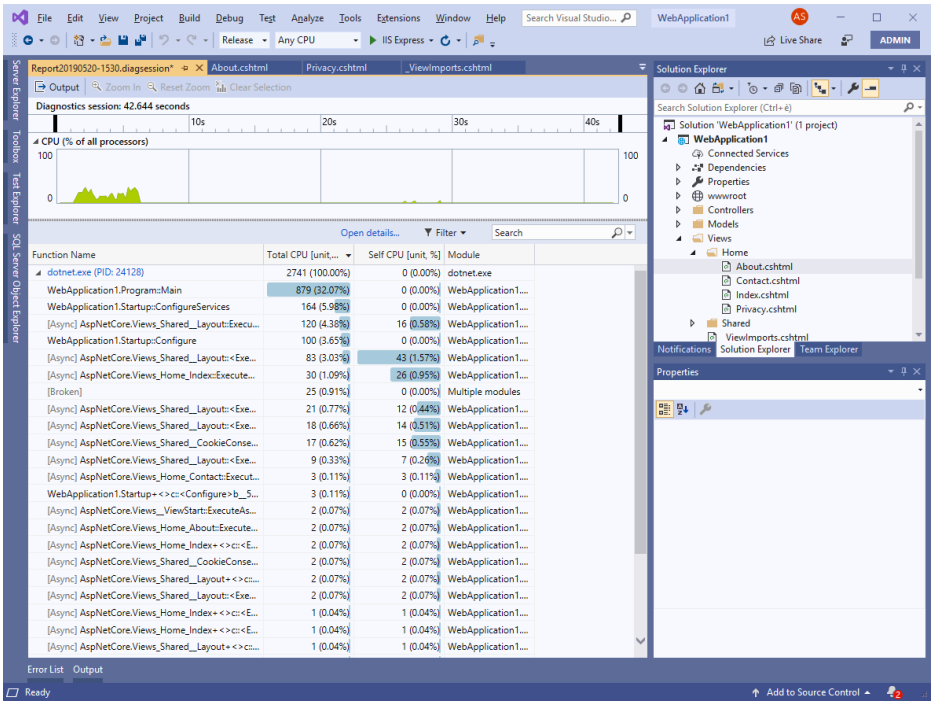


*Figure 75: The report of the CPU Usage analysis*

Notice how, for each method (Function Name column) you can see the percentage of CPU usage. If you double-click a method name, Visual Studio will display a detail window where you can see the current method, caller methods, and called methods. Figure 76 shows an example where the calling and called methods are inside external code.

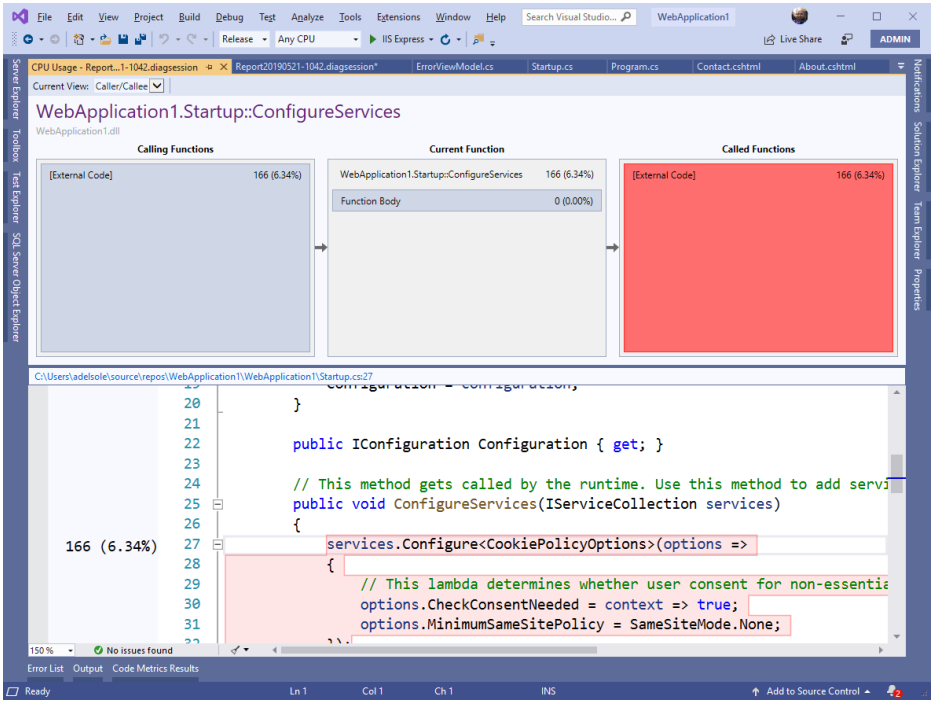


*Figure 76: CPU Usage across method calls*

Notice how Visual Studio highlights the code that had an impact on the CPU usage and displays the percentage of usage next to the code. You can also click inside the Calling Function and Called Function areas to see additional details on caller and called methods, including their source code, when available.

## Publish experience enhancements

Visual Studio 2019 also provides a redesigned experience for publishing a web application. When you right-click the project name in Solution Explorer and click **Publish**, you will see the Publish tab of the project's welcome page. Click **Start** to start configuring the app deployment. You will be first asked to specify a publish target, as you can see in Figure 77.

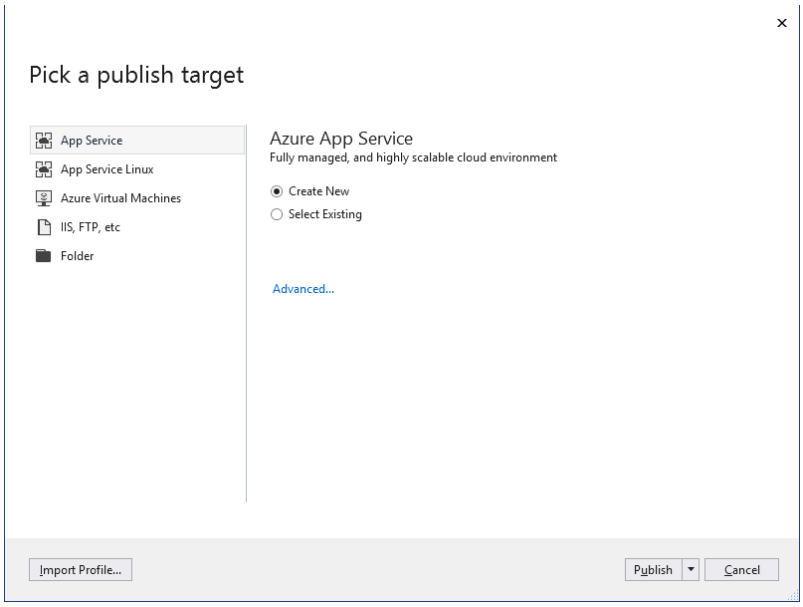


*Figure 77: Selecting a publish target*

For both the App Service and App Service Linux targets, you will be able to publish your web app directly to your Azure subscription.

> ***Note: The App Service, App Service Linux, and Azure Virtual Machines options require an active Microsoft Azure subscription. If you want to try out these features, you can request a free trial.***

The only difference is that for the App Service option, you can create a new app service or select an existing one, whereas for the App Service Linux option, you can only create a new app service on Azure. In both cases, if you click **Publish**, you will be able to configure your app service, as demonstrated in Figure 78.

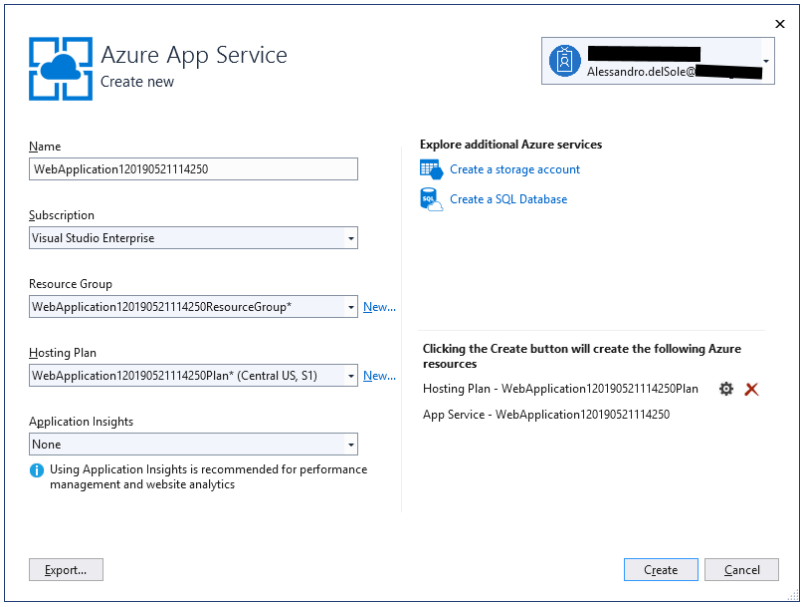


*Figure 78: Azure App Service configuration*

Visual Studio will automatically search for an Azure subscription associated with the Microsoft Account you used to sign into the IDE. If one is found, Visual Studio will also automatically prefill the configuration form with a service name, a resource group name, and a hosting plan. Replace the app service name with a more meaningful name and make sure you select the appropriate hosting plan, as these are paid services.

> ***Note: App services, resource groups, hosting plans, and subscription types are not covered in this e-book. If you are new to Azure, refer to the official Microsoft Azure portal for all the information you need before publishing any resource to your full or trial subscription.***

In a similar way, you will be able to select an existing Azure Virtual Machine as a publish target. The IIS and FTP publish option will open the well-known dialog for configuring the deployment to an IIS or FTP server, so there's nothing new here. The Folder option allows you to publish the web application to a local folder for manual deployment (see Figure 79) with a slightly different user interface, due to the whole experience redesign.

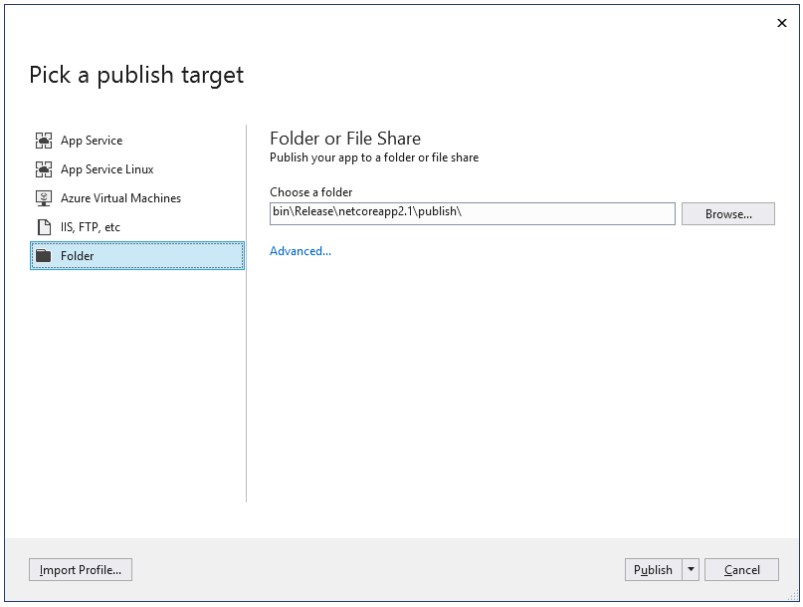


*Figure 79: Publishing to a local folder*

This redesigned publish experience not only makes the user interface consistent with VS 2019's look and feel, but it also provides a much better integration with the Azure services.

***Note: To stay up to date on new or updated features (especially with .NET Core 3.0 around the corner), don't forget to look at the official ASP.NET blog and read the release notes when Visual Studio 2019 releases updates.***

## Chapter summary

The focus of the improvements for web development in Visual Studio 2019 is on extending support for .NET Core and its ASP.NET Core flavor. Creating an ASP.NET Core project is now a better-organized experience.

Analysis tools such as Code Metrics and CPU Usage are now available to .NET Core apps. CodeLens and Find All References in .NET Core can now show references from Razor files. Publishing a web application is still as easy as it was in the past, but with better integration with several Azure services.

Actually, for .NET Core development there is something very interesting that is almost ready to see the light: desktop development with Windows Presentation Foundation and Windows Forms, as you will see in the next chapter.

# Chapter 8 Desktop Development with .NET Core 3.0

With cross-platform development, it is common to think about writing code that runs on different mobile and web systems. However, desktop development still has the same importance, especially for line-of-business and multimedia applications. With the upcoming release of .NET Core 3.0, Microsoft will bring desktop development with .NET to macOS and Linux systems. This chapter provides a quick overview of what you will be able to achieve soon with Visual Studio 2019.

## .NET Core 3.0 for desktop development

The .NET Core 3.0 framework, currently available as a preview release, will include support for developing desktop applications with popular frameworks, such as Windows Presentation Foundation (WPF) and Windows Forms. This is a tremendous benefit because these applications will run not only on Windows, but also on macOS and Linux, in the pure spirit of .NET Core.

At this writing, the latest preview version of the .NET Core runtime is called v3.0.0-preview5, and the build number of the SDK is v3.0.100-preview5.011568. Don't forget to check out the official download page for more recent updates. Support for developing desktop applications is offered through new .NET Core project templates and is based on its project system. I will now guide you through setting up the development environment before discussing how to create applications.

### Enabling .NET Core 3.0 previews

The first step is downloading .NET Core 3.0 Preview from the official download page. You will have different options, but make sure you download the most recent installer from the Build apps – SDK column.

***Tip: I recommend downloading the Installer file instead of the Binaries package. The Installer will automatically set up and configure .NET Core files properly, simplifying the whole process.***

Once .NET Core 3.0 Preview has been installed, the next step is enabling Visual Studio 2019 to use it. This step will not be necessary once .NET Core 3.0 is fully released.

***Note: Though you can use the .NET Core 3.0 Preview with the stable release of Visual Studio 2019, this is not recommended. You can instead use Visual Studio 2019 previews to test .NET Core early releases. Early builds of VS 2019 include snapshots***

***of new features that are under development. The Visual Studio 2019 Preview has a dedicated download page, and can be installed side by side with the main release.***

To accomplish this, select **Tools** > **Options** > **Environment** > **Preview Features** and select the **Use previews of the .NET Core SDK** check box, as shown in Figure 80.

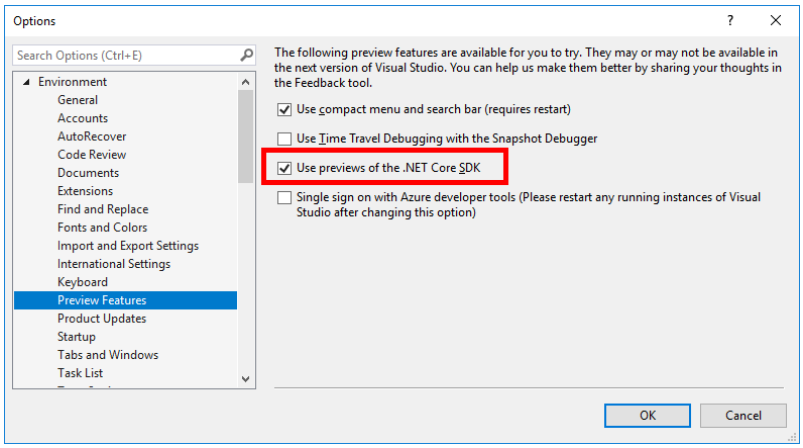


*Figure 80: Enabling .NET Core SDK previews*

Click **OK**. Now everything is set up, and you can start creating cross-platform desktop apps.

# Creating WPF applications

Windows Presentation Foundation (WPF) has been Microsoft's premiere technology for building Windows desktop applications for many years. Based on XAML for the user interface definition, and on C#/Visual Basic for the imperative code, WPF provides a robust and powerful infrastructure that covers all desktop development needs, from business applications to multimedia to document management.

The good news is that now (most of) WPF is open source and part of the .NET Core code base. With the upcoming .NET Core 3.0 and Visual Studio 2019, you will be able to build WPF applications that run on other desktop systems, such as macOS and Linux. To accomplish this, you simply create a new WPF App (.NET Core) project, as shown in Figure 81.

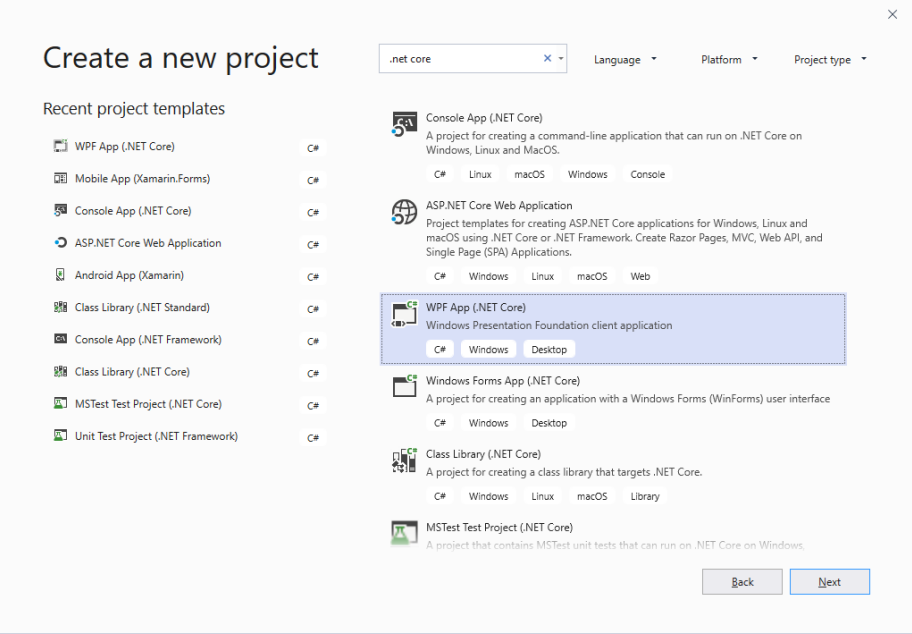


*Figure 81: Creating a new WPF app on top of .NET Core*

Click **Next** and provide a project name. When the project is created, you will see the usual WPF development environment that you would expect. This includes a fully working designer and the XAML code editor that already has the same functionalities as the classic WPF editor. Figure 82 shows an example with a few lines of code, where you can also see how IntelliSense is displaying IntelliCode suggestions.

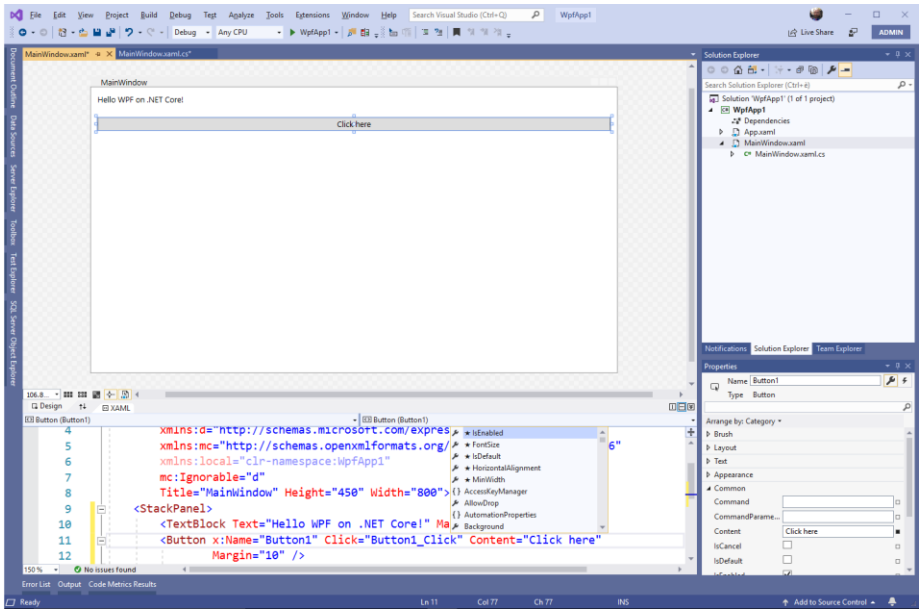


*Figure 82: The WPF development environment with .NET Core*

As you can see, the whole development experience is the same as for the classic WPF environment. You will get the same tools, and you will be able to write the same code in a cross-platform way. Keep in mind that, at this stage, WPF on .NET Core 3.0 is also in a preview state. This means that it can be subject to changes, improvements, and updates until it reaches the RTM milestone.

### Hints about publishing .NET Core desktop apps

Unlike Windows applications built on top of the .NET Framework, desktop apps built with .NET Core need a new way to be packaged and deployed. This is mainly for two reasons:

- Classic desktop apps expect to find the .NET Framework on the target machine as a system component and will install the full .NET if it's not found. Instead, .NET Core apps need to include libraries and runtime components in the deployment package.
- .NET Core apps will work across operating systems, so a unified installation process cannot be implemented.

To solve this problem, you can prepare a .NET Core desktop app for deployment using the **`dotnet publish`** command, which works cross-platform. Additionally, for apps targeting Windows 10, Microsoft has also created a new packaging format called MSIX, which is designed specifically for Win32, WPF, and Windows Forms apps. The official documentation includes a page about MSIX, which I recommend you read.

***Note: The concepts described in the previous paragraph apply to both WPF and Windows Forms in .NET Core 3.0. Windows Forms is discussed in the next section.***

## Creating Windows Forms applications

Windows Forms was the very first .NET technology that developers could use to build desktop applications. Not only has it always been very popular over time, despite the power and flexibility of WPF, but there are still likely many hundreds of thousands, or possibly even millions, of Windows Forms applications in the world that need to be maintained and updated.

If you are interested in both Windows Forms and cross-platform development, there is good news: Windows Forms is now open source and runs on top of .NET Core 3.0. The project template you will use is called Windows Forms (.NET Core App), which you can see in Figure 81. At this writing, the development experience for Windows Forms in Visual Studio 2019 still has limited support. You can create a solution, but the designer is not ready, and there are other IDE limitations.

If you want to try Windows Forms and you can't wait for the final release, my suggestion is to use Visual Studio 2019 Preview, which includes more updates, even if it cannot be used for production. Apart from this, having all the most popular desktop development frameworks running on a cross-platform technology will help you bring your software to macOS and Linux markets very much faster and more efficiently.

## Chapter summary

Microsoft is working hard on the next major release of .NET Core, version 3.0. This release, which you can currently test in a preview stage, will include support for developing WPF and Windows Forms desktop applications on a cross-platform technology. At the moment, these can be used by enabling Visual Studio 2019 to use .NET Core previews.

Remember that preview releases cannot be used for production, and that they are subject to change. However, you can get an idea of how beautiful the future is going to be for you as a developer, especially if you have existing Windows desktop apps that you would like to quickly bring to other markets.