


C# & Dotnet

| | |
|------------------|--|
| 🕒 Created | @December 28, 2023 1:50 PM |
| 📁 Class | WebDev |
| 📁 Type | Reading |
| ☑ Reviewed | <input type="checkbox"/> |
| 👤 Hudhaifa Gburi |  hudhaifa gburi |

Chapter01

Getting code solutions for this book

The GitHub repository for this book has solutions using full application projects for all code tasks and exercises, found at the following link:

<https://github.com/markjprice/cs12dotnet8>

Pros and cons of the Polyglot Notebooks extension

The Polyglot Notebooks extension for Visual Studio Code provides an easy and safe place to write simple code snippets for experimenting and learning. For example, data scientists use them to analyze and visualize data. Students use them to learn how to write small pieces of code for language constructs and to explore APIs.

Visual Studio Code for cross-platform development

- The most modern and lightweight code editor to choose from, and the only one from Microsoft that is cross-platform, is Visual Studio Code. It can run on all common operating systems, including Windows, macOS, and many varieties of Linux, including Red Hat Enterprise Linux (RHEL) and Ubuntu.
- Visual Studio Code is a good choice for modern cross-platform development because it has an extensive and growing set of extensions to support many languages beyond C#. The most important extension for C# and .NET developers is the C# Dev Kit that was released in preview in June 2023 because it turns Visual Studio Code from a general-purpose code editor into a tool optimized for C# and .NET developers.

GitHub Codespaces for development in the cloud

- GitHub Codespaces is a fully configured development environment based on Visual Studio Code that can be spun up in an environment hosted in the cloud and accessed through any web browser. It supports Git repos, extensions, and a built-in command-line interface so you can edit, run, and test from any device.

Visual Studio 2022 for general development

- Visual Studio 2022 for Windows can create most types of applications, including console apps, websites, web services, and desktop apps. Although you can use Visual Studio 2022 for Windows to write a cross-platform mobile app, you still need macOS and Xcode to compile it.
- It only runs on Windows 10 version 1909 or later, Home, Professional, Education, or Enterprise; or on Windows 11 version 21H2 or later, Home, Pro, Pro Education, Pro for Workstations, Enterprise, or Education. Windows Server 2016 and later are also supported. 32-bit operating systems and WindowsS mode are not supported.

Keyboard shortcuts for Visual Studio 2022 for Windows

<https://learn.microsoft.com/en-us/visualstudio/ide/identifying-and-customizingkeyboard-shortcuts-in-visual-studio>.

Understanding .NET

"Those who cannot remember the past are condemned to repeat it."

– George Santayana

Understanding .NET runtime and .NET SDK versions

If you have not built a standalone app, then the .NET runtime is the minimum needed to install so that an operating system can run a .NET application. The .NET SDK includes the .NET runtime as well as the compilers and other tools needed to build .NET code and apps. .NET runtime versioning follows semantic versioning, that is, a major increment indicates breaking changes, minor increments indicate new features, and patch increments indicate bug fixes. .NET SDK versioning does not follow semantic versioning. The major and minor version numbers are tied to the runtime version it is matched with. The third number follows a convention that indicates the minor and patch versions of the SDK. The minor number is multiplied by 100 and added to the patch number.

Understanding intermediate language

The benefit of this two-step compilation process is that Microsoft can create Common language runtimes (CLR) for Linux and macOS, as well as for Windows. The same IL code

runs everywhere
because of the second compilation step, which generates code for the native operating system and CPU instruction set.

Comparing .NET technologies

Comparing .NET technologies

We can summarize and compare the current .NET technologies as shown in *Table 1.4*:

| Technology | Description | Host operating systems |
|----------------|--|--|
| Modern .NET | A modern feature set, with full C# 8 to C# 12 language support. It can be used to port existing apps or create new desktop, mobile, and web apps and services. It can target older .NET platforms. | Windows, macOS, Linux, Android, iOS, tvOS, Tizen |
| .NET Framework | A legacy feature set with limited C# 8 support and no C# 9 or later support. It should be used to maintain existing applications only. | Windows only |
| Xamarin | Mobile and desktop apps only. | Android, iOS, macOS |

Table 1.4: Comparison of .NET technologies

Top-level programs in C# refer to a feature introduced in C# 9.0 that allows you to write a C# program without explicitly defining a class or a Main method. It simplifies the structure of simple programs and scripts by eliminating some of the boilerplate code.

Traditionally, a C# program starts with a `Main` method inside a class. For example:

```
using System;

class Program
{
    static void Main()
    {
        Console.WriteLine("Hello, World!");
    }
}
```

```
}  
}
```

With top-level programs, you can simplify this to:

```
using System;  
  
Console.WriteLine("Hello, World!");
```

In the top-level program, there's no need for a separate class or a `Main` method. The entry point of the program is the top-level code itself. This makes the code more concise and readable, especially for small scripts or short programs.

Here are some key points about top-level programs:

1. **No Explicit Main Method:** In a top-level program, you don't need to explicitly define a `Main` method. The top-level code is treated as the entry point.
2. **Implicit Class:** Behind the scenes, the C# compiler implicitly generates a class to wrap the top-level code. This class contains the compiled code from the top-level program.
3. **Using Directives:** You can still use `using` directives and reference external assemblies at the top level.
4. **Script-Like Syntax:** Top-level programs make C# feel more like a scripting language for short, focused tasks.

Keep in mind that while top-level programs are convenient for certain scenarios, they might not be suitable for more complex applications where a structured class-based approach is beneficial.

Summary of steps for Visual Studio Code

Follow these steps to create a solution and projects using Visual Studio Code:

1. Create a folder for the solution, for example, Chapter01
2. Create a solution file in the folder: `dotnet new sln`
3. Create a folder and project using a template: `dotnet new console -o HelloCS`
4. Add the folder and its project to the solution: `dotnet sln add HelloCS`
5. Repeat steps 3 and 4 to create and add any other projects.
6. Open the folder containing the solution using Visual Studio Code: `code .`

How to build VSCode Project

Getting definitions of types and their members

1. In your preferred code editor, open the solution/folder named Chapter01.

If you are using Visual Studio 2022:

- Navigate to **Tools | Options**.
- In the search box, enter `navigation to source`.
- Select **Text Editor | C# | Advanced**.
- Clear the **Enable navigation to Source Link and Embedded sources** check box, and then click **OK**, as shown in *Figure 1.16*:

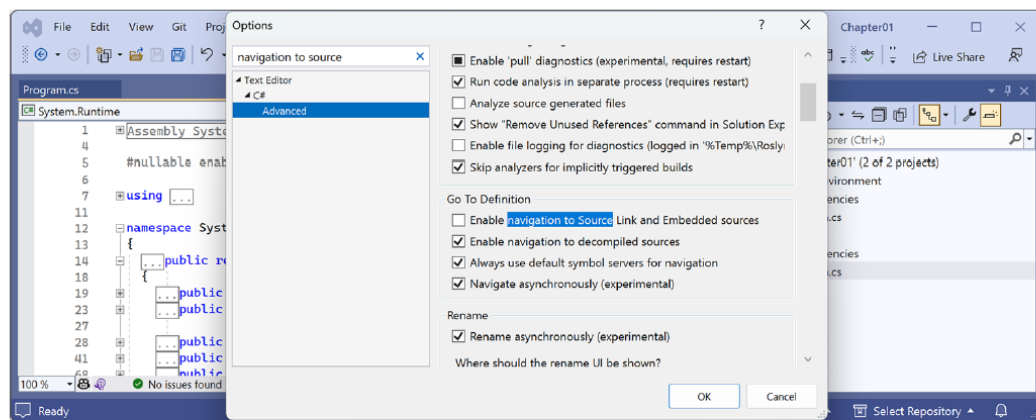


Figure 1.16: Disabling Source Link for the Go To Definition feature

Inline hints, aka **inlay hints**, show the names of parameters without you having to type them, as shown in *Figure 1.19*:

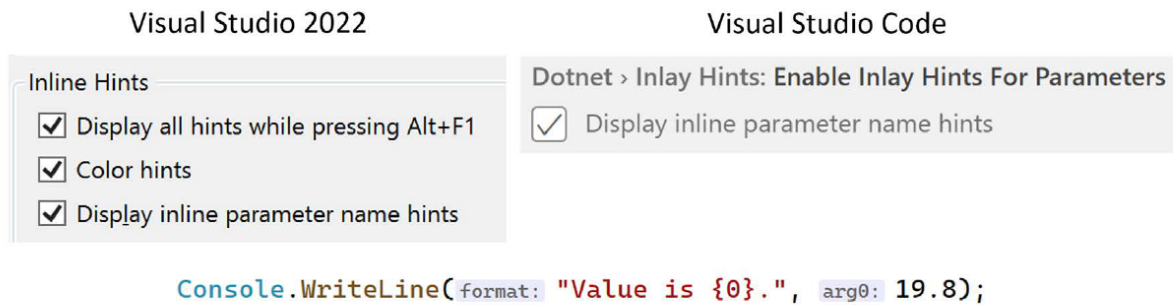


Figure 1.19: Configuring inline hints, aka inlay hints

Most code editors have this feature that you can enable permanently or only when a key combination like *Alt + F1* or *Ctrl* is held down:

- In Visual Studio 2022, navigate to **Tools | Options**, navigate to **Text Editor | C# | Advanced**, scroll down to the **Inline Hints** section, select the **Display inline parameter hint names** check box, and then click **OK**.
- In Visual Studio Code, navigate to **File | Preferences | Settings**, search for **inlay**, select the **C#** filter, and then select the **Display inline parameter name hints** check box.
- In JetBrains Rider, in **Settings**, navigate to **Editor | Inlay Hints | C# | Parameter Name Hints**.

Searching for answers using Google

You can search Google with advanced search options to increase the likelihood of finding what you need:

1. Navigate to Google at the following link: <https://www.google.com/>.
2. Search for information about **garbage collection** using a simple Google query and note that you will probably see a lot of ads for garbage collection services in your local area before you see the Wikipedia definition of garbage collection in computer science.
3. Improve the search by restricting it to a useful site such as Stack Overflow, and by removing languages that we might not care about, such as C++, Rust, and Python, or by adding C# and .NET explicitly, as shown in the following search query:

```
garbage collection site:stackoverflow.com +C# -Java
```

Searching the .NET source code

Sometimes you can learn a lot from seeing how the Microsoft teams have implemented .NET. The source for the entire code base for .NET is available in public GitHub repositories. For example, you might know that there is a built-in attribute to validate an email address. Let's search the repositories for the word "email" and see if we can find out how it works:

1. Use your preferred web browser to navigate to <https://github.com/search>.
2. Click **advanced search**.
3. In the search box, type **email**.

4. In the **In these repositories** box, type **dotnet/runtime**. (Other repositories you might want to search include **dotnet/core**, **dotnet/aspnetcore**, **dotnet/wpf**, and **dotnet/winforms**).
5. In the **Written in this language** box, select **C#**.
6. At the top right of the page, note how the advanced query has been written for you. Click **Search**, and then click the **Code** filter and note the results include **EmailAddressAttribute**, as shown in *Figure 1.21*:

