Acumatica
The Cloud ERP

Acuminator, Visual Studio and other developer tools

Developer productivity and solution quality validation

Sergey Nikomarov
Senior System Developer
Acumatica

Agenda

Acuminator

- Acuminator 3.0.0 – new features

Visual Studio

- Overview
- Code Editor Appearance
- Writing Code
- Code Analysis
- Integration with external tools

Other Developer Tools

- Visual Studio alternatives
- Examples of useful Visual Studio plugins

Acuminator

New Features Overview

- Support of Visual Studio 2022
- Code analysis enhancements
- Code Map displays more information
- Suppression mechanism improvements
- Bugfixes

Acuminator: Static Code Analysis

Overview

- Throwing exception in row persisted graph event is now allowed if one of the following statements is true:
  - The graph is a processing graph
  - The exception type is one of the allowed exception types or their descendant:
    - `PX.Data.PXRowPersistedException`
    - `PX.Data.PXLockViolationException`
    - .NET exceptions from the `System` namespace: `NotImplementedException`, `NotSupportedException`, `ArgumentException`
- Underscores are now allowed in DAC extension names
- PX1008 diagnostic is significantly extended
- PX1088 diagnostic is entirely removed

Common Mistake

Capturing reference to the screen graph in the long run delegate is a common error

```
// Reference to the graph is captured by the delegate because
// GraphInstanceMethod is an instance method of the graph
PXLongOperation.StartOperation(this, delegate() { GraphInstanceMethod(myDac); });
```

Capturing reference to the screen graph will cause synchronous delegate execution and different tricky issues including data consistency problems.

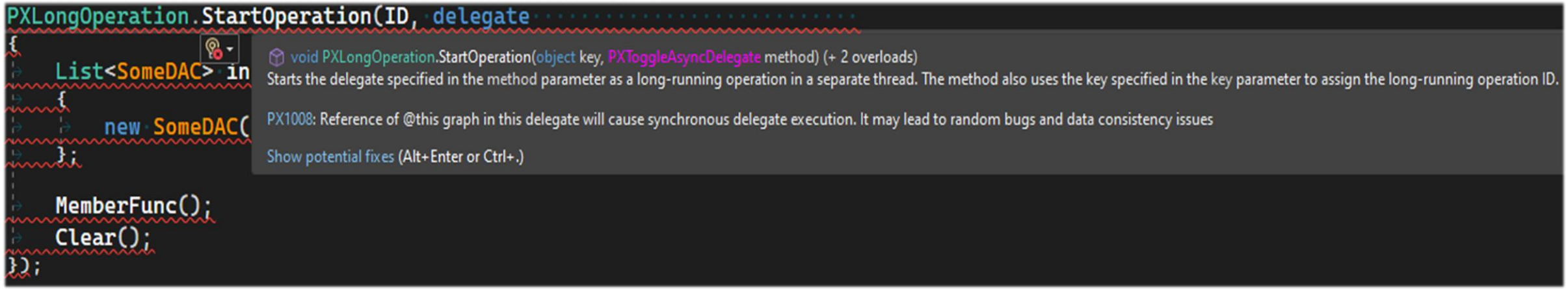
Here is a correct way to use graph instance methods in long run delegates

```
PXLongOperation.StartOperation(this, delegate()
{
    MyGraph graph = PXGraph.CreateInstance<MyGraph>();
    graph.GraphInstanceMethod(myDac);
});
```

PX1008 Diagnostic

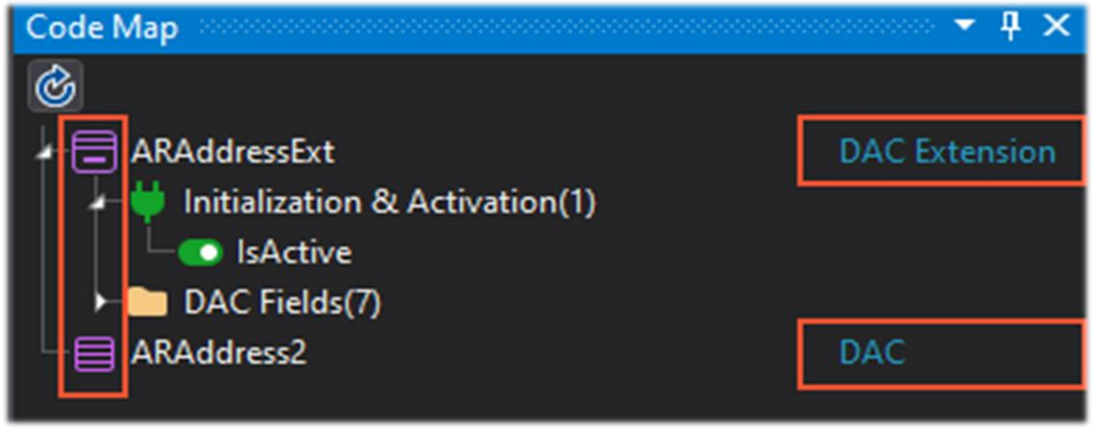
- In previous versions of Acuminator performed search for captured graph reference in processing delegates

- Now this diagnostic also validates long run delegates

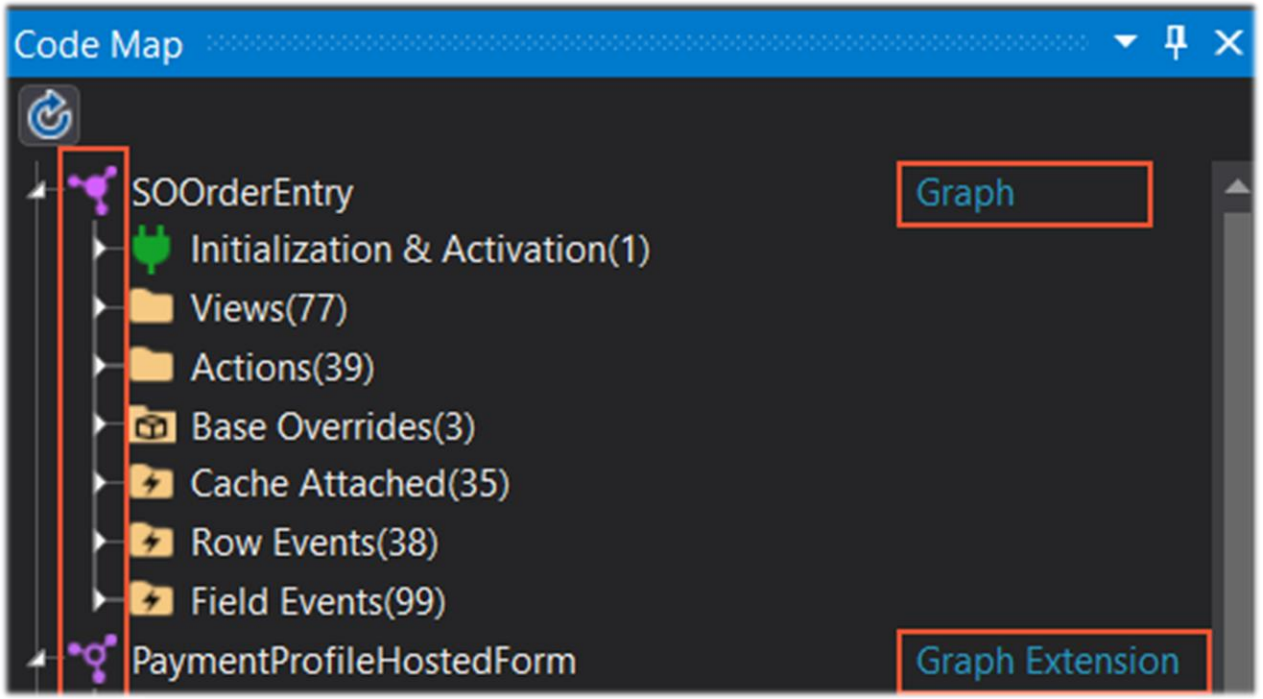


Acuminator: Code Map

Code Map Improvements



DAC nodes now indicate whether the node represents a DAC or a DAC extension

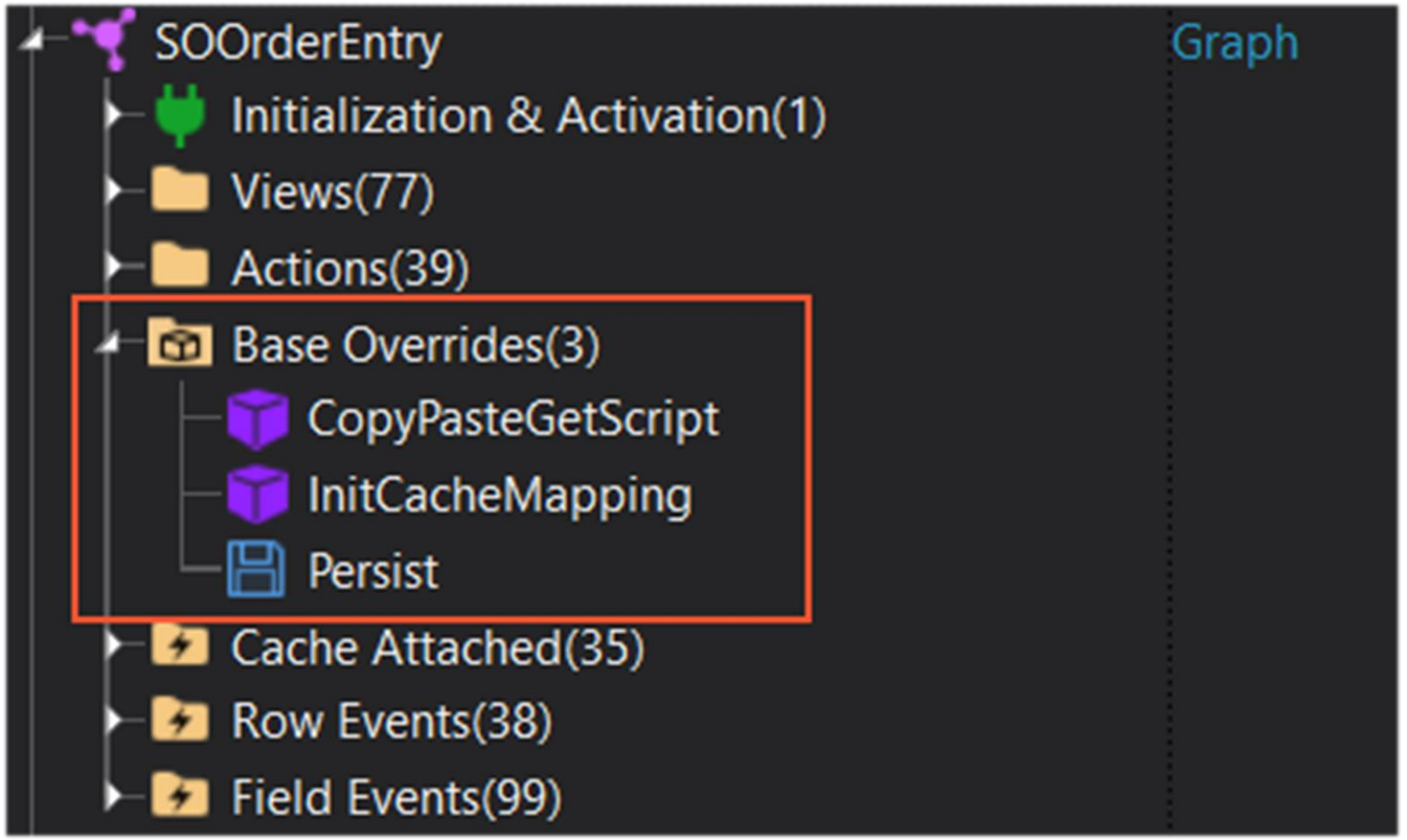


Different icons for DACs, DAC extensions, graphs, and graph extensions

The Code Map window layout now is remembered by Visual Studio

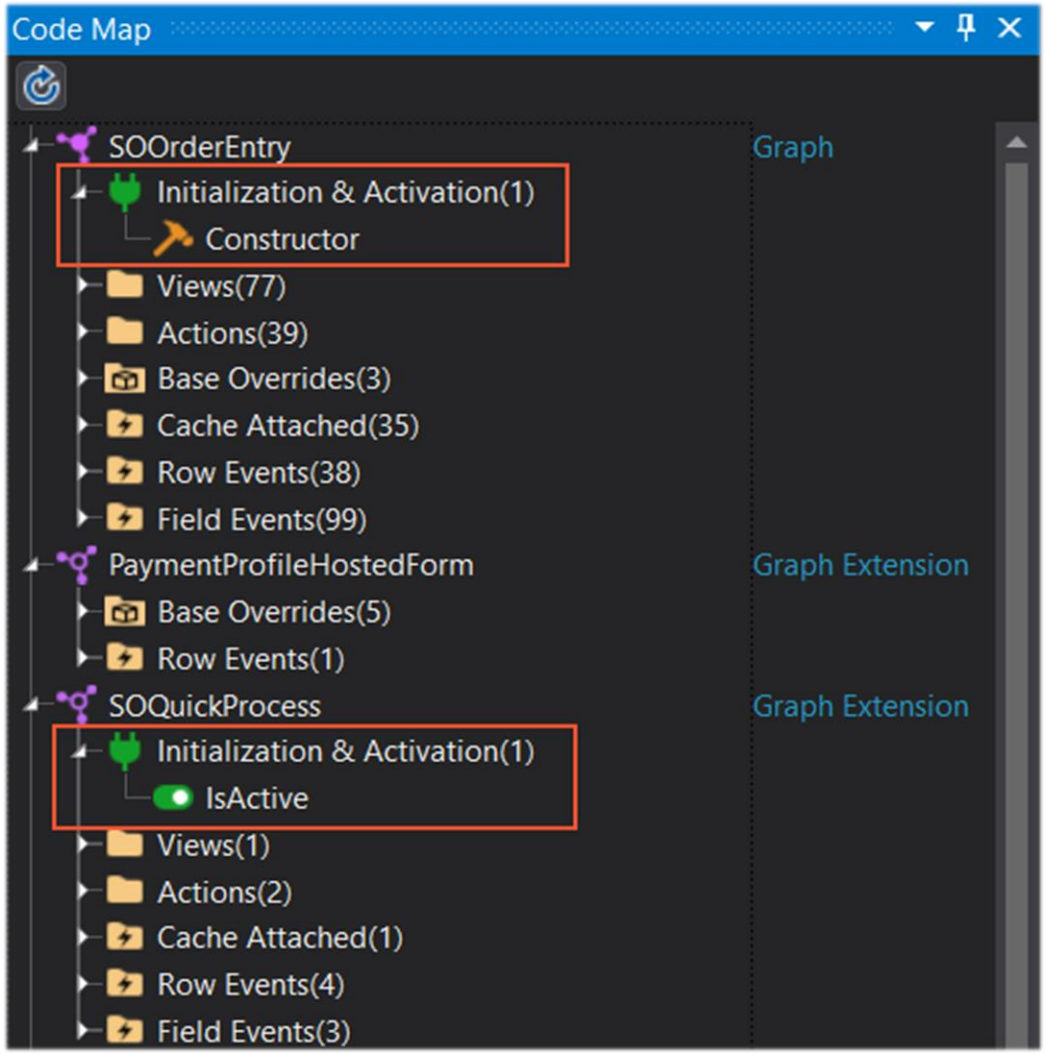
Base Overrides Category

- For graphs and graph extensions overrides of virtual type members are now displayed under the new **Base Overrides** node.
- The `Persist` method overrides are indicated with a special diskette icon



Initialization and Activation Category

- New types of nodes represent:
  - Graph instance and static constructors for graphs and graph extensions
  - The `IsActive` method for graph and DAC extensions
- **Initialization and Activation** category node will contain these new nodes



Visual Studio

Visual Studio in comparison with other tools

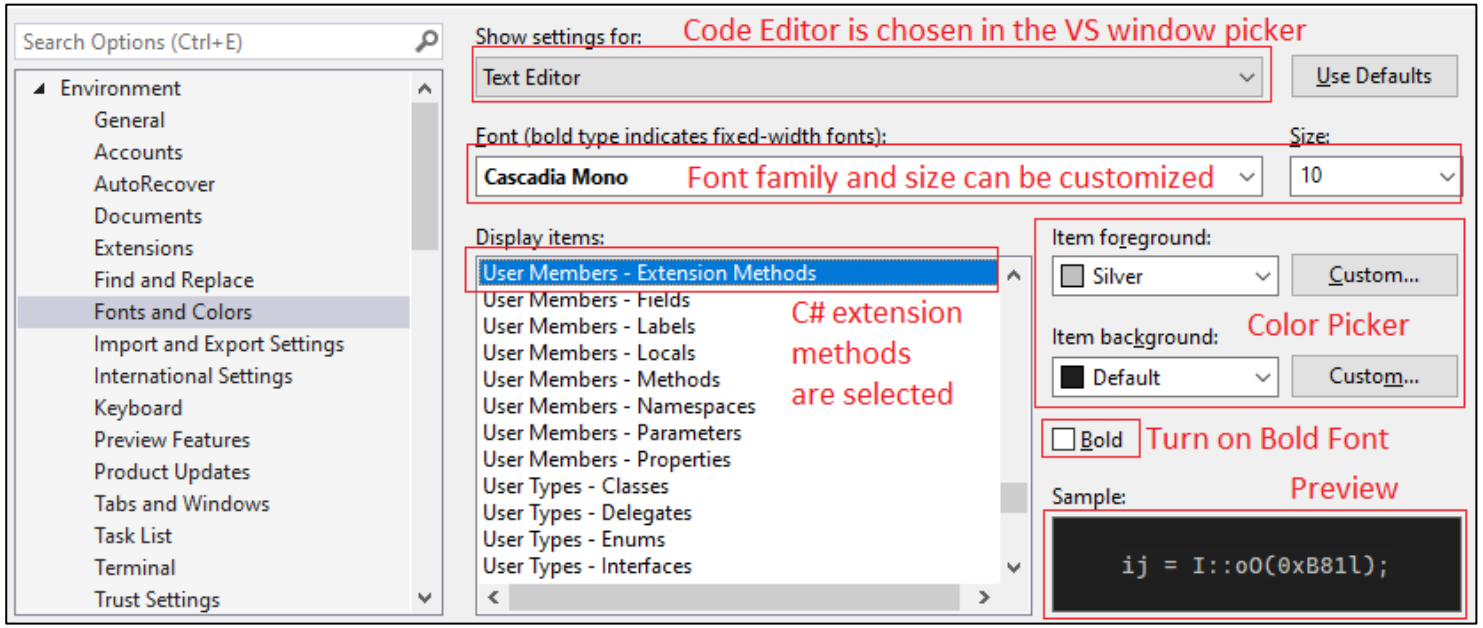
Visual Studio features

- Read code with syntax highlighting and code regions folding
- Write code with intelligent code completion, code snippets, multiple carets, keyboard shortcuts
- Search your code with different search mechanisms
- Check your code with code analysis diagnostics
- Debug your code step by step, check objects in memory and modify their state
- Collaborate with your teammates with “Live Share” feature
- Modify your code during debug with “Hot reload” feature
- Profile your code with different diagnostic tools
- Integration with Version Control Systems like Git and other external tools
- Support for unit tests
- Highly configurable and extendable with many available plugins
- Many other things – C# interpreter, navigation to decompiled sources, etc.

Visual Studio: Code Editor Appearance

Syntax Highlighting

```
[PXNote(ShowInReferenceSelector = true, Selector = typeof(
    Search2<
        ARInvoice.refNbr,
    InnerJoinSingleTable<ARRegister, On<ARInvoice.docType, Equal<ARRegister.docType>,
        And<ARInvoice.refNbr, Equal<ARRegister.refNbr>>>,
    InnerJoinSingleTable<Customer, On<ARRegister.customerID, Equal<Customer.bAccountID>>>>,
    Where2<
        Where<ARRegister.origModule, Equal<BatchModule.moduleAR>,
            Or<ARRegister.origModule, Equal<BatchModule.moduleEP>,
            Or<ARRegister.released, Equal<True>>>>,
        And<Match<Customer, Current<AccessInfo.userName>>>>,
    OrderBy<
        Desc<ARRegister.refNbr>>>))]
66 references | 0 changes | 0 authors, 0 changes
public override Guid? NoteID...
```



Visual Studio highlights common C# syntax elements

It supports custom syntax highlighting

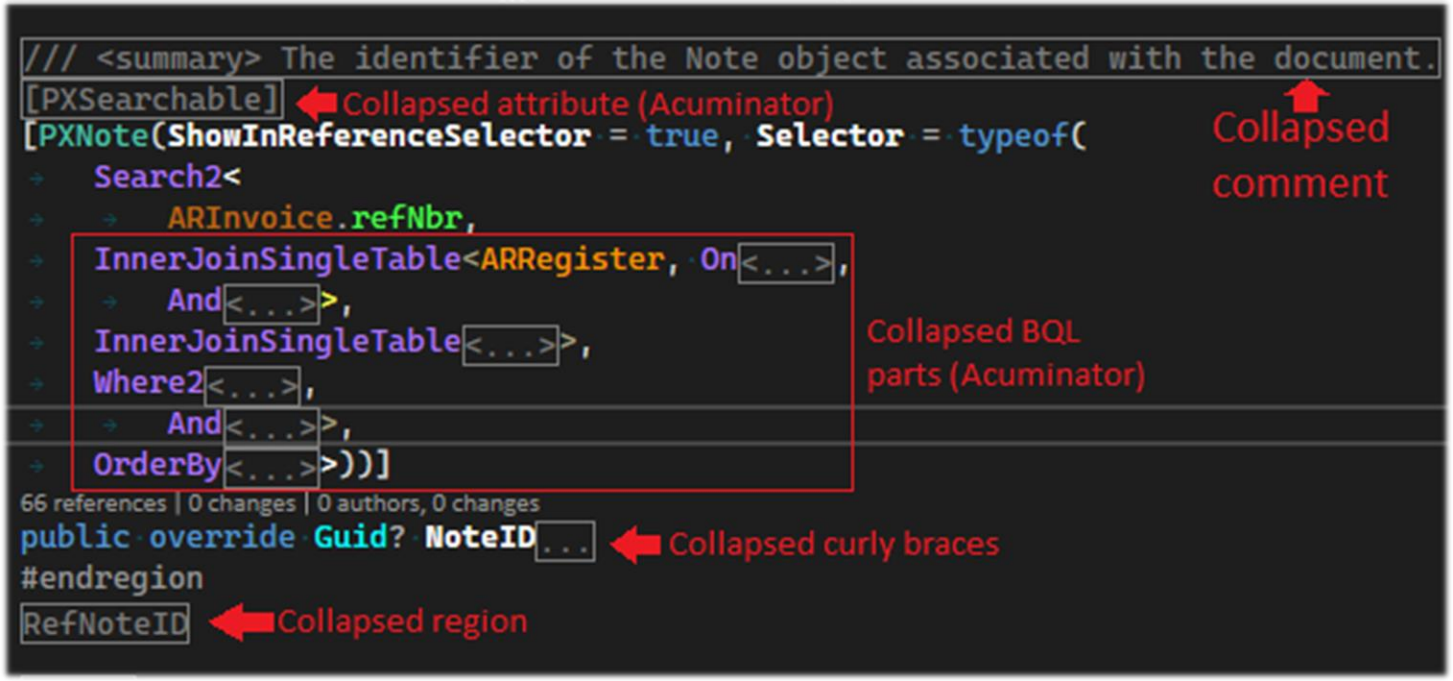
Visual Studio allows you to override default colors, backgrounds and fonts for any colorable code element

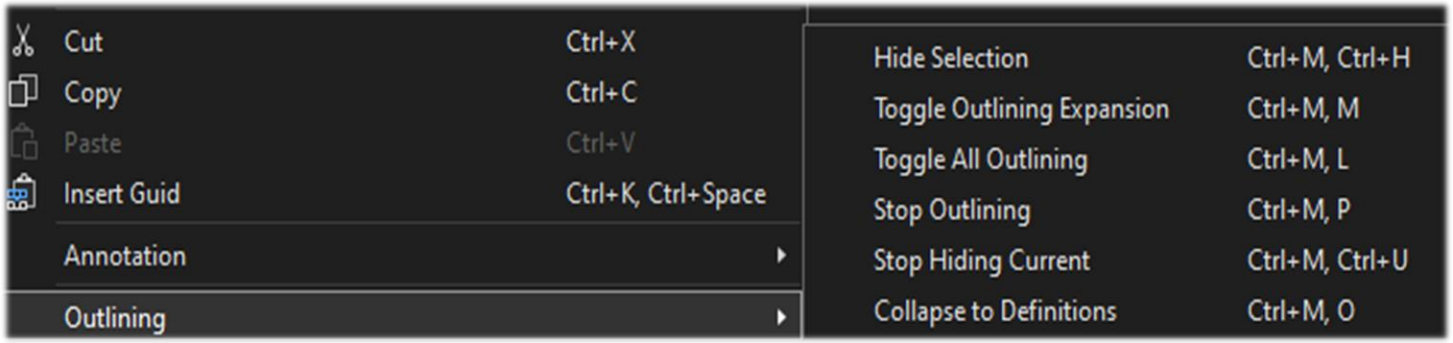
You can access syntax highlighting settings via “Tools” → “Options” menu. Here look for the “Environment” → “Fonts and Colors” section

Overrides are stored separately for each Visual Studio Theme

You can always reset color to default one

Code Outlining





Visual Studio supports **code outlining** (or **code folding**)

You can collapse some code fragments to get a better overview of the overall structure:

- curly braces
- #regions
- XML doc comments

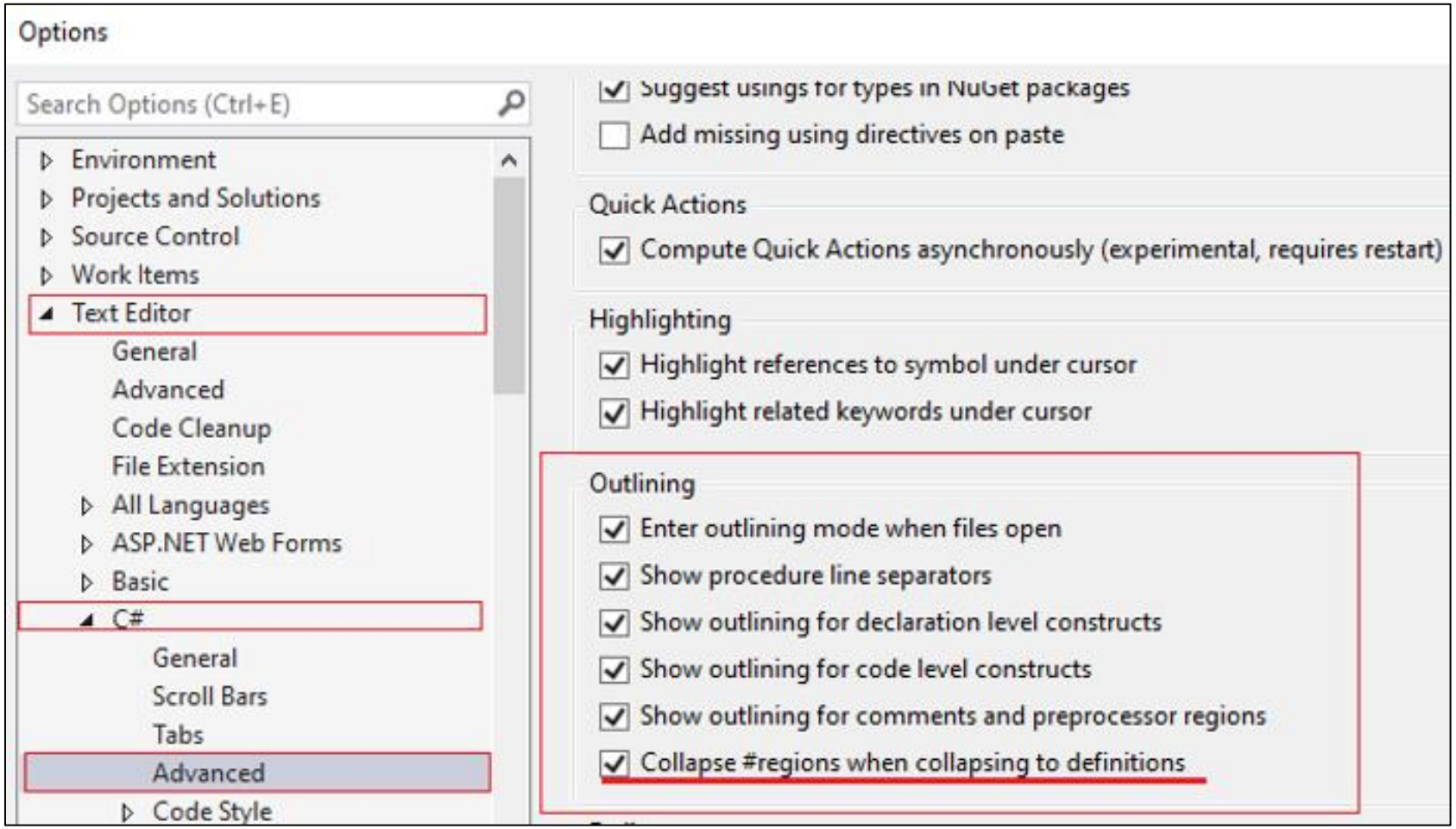
Acuminator adds outlining for attributes and BQL parts

A tooltip with contents is displayed on mouse hover

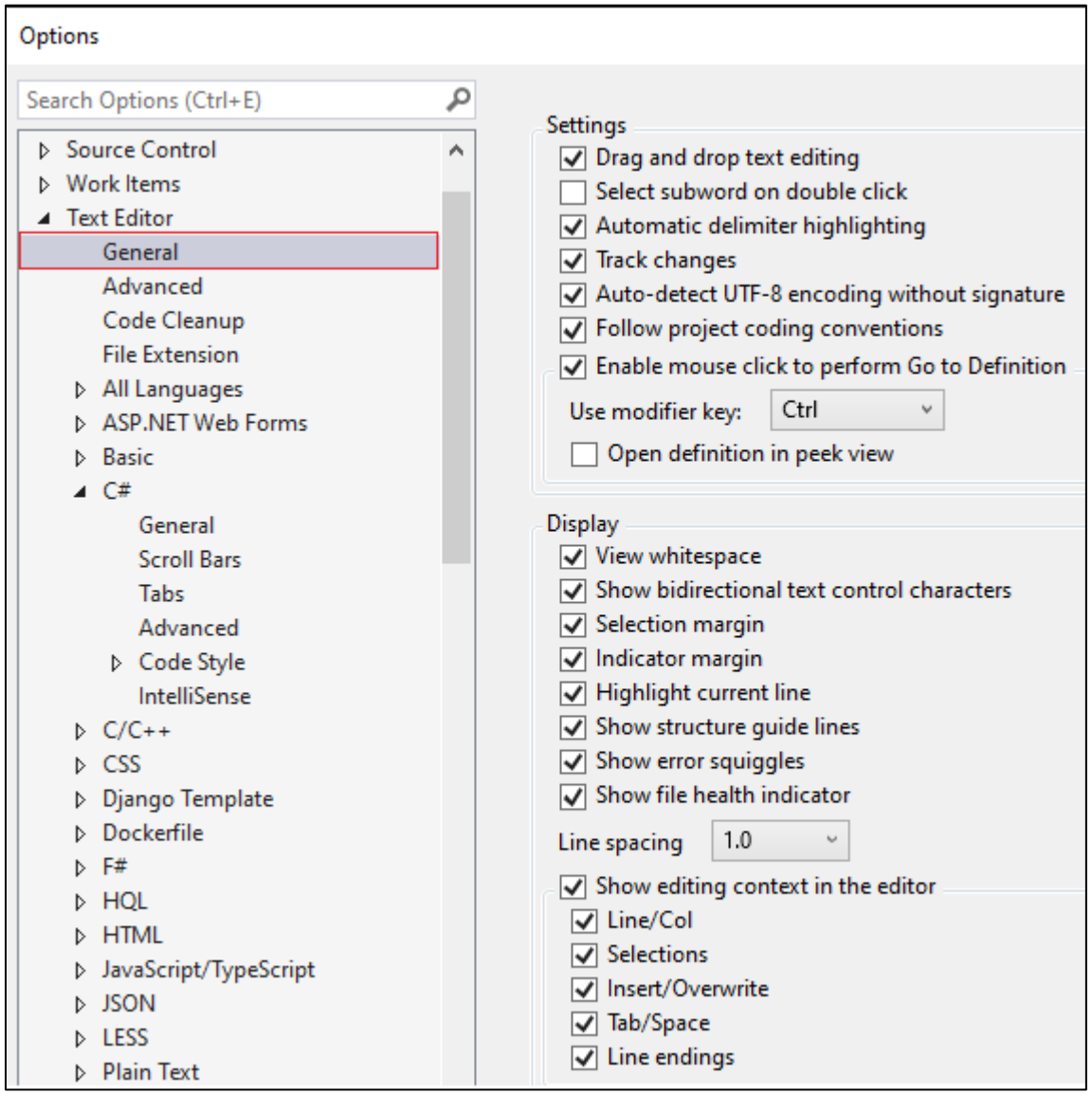
Code editor context menu contains commands that can hide the selected text or manage the outlining for the whole file

Code Outlining Settings

- C# outlining settings in Visual Studio options: “Text Editor” → “C#” → “Advanced” → “Outlining”
- Enable “Collapse #regions when collapsing to definitions” for better experience with DACs



General Settings



- ❑ Code editor is highly configurable
- ❑ Different indicators, controls, lines and margins can be hidden
- ❑ Text display can be customized
- ❑ Default values are sensible

Some interesting settings:

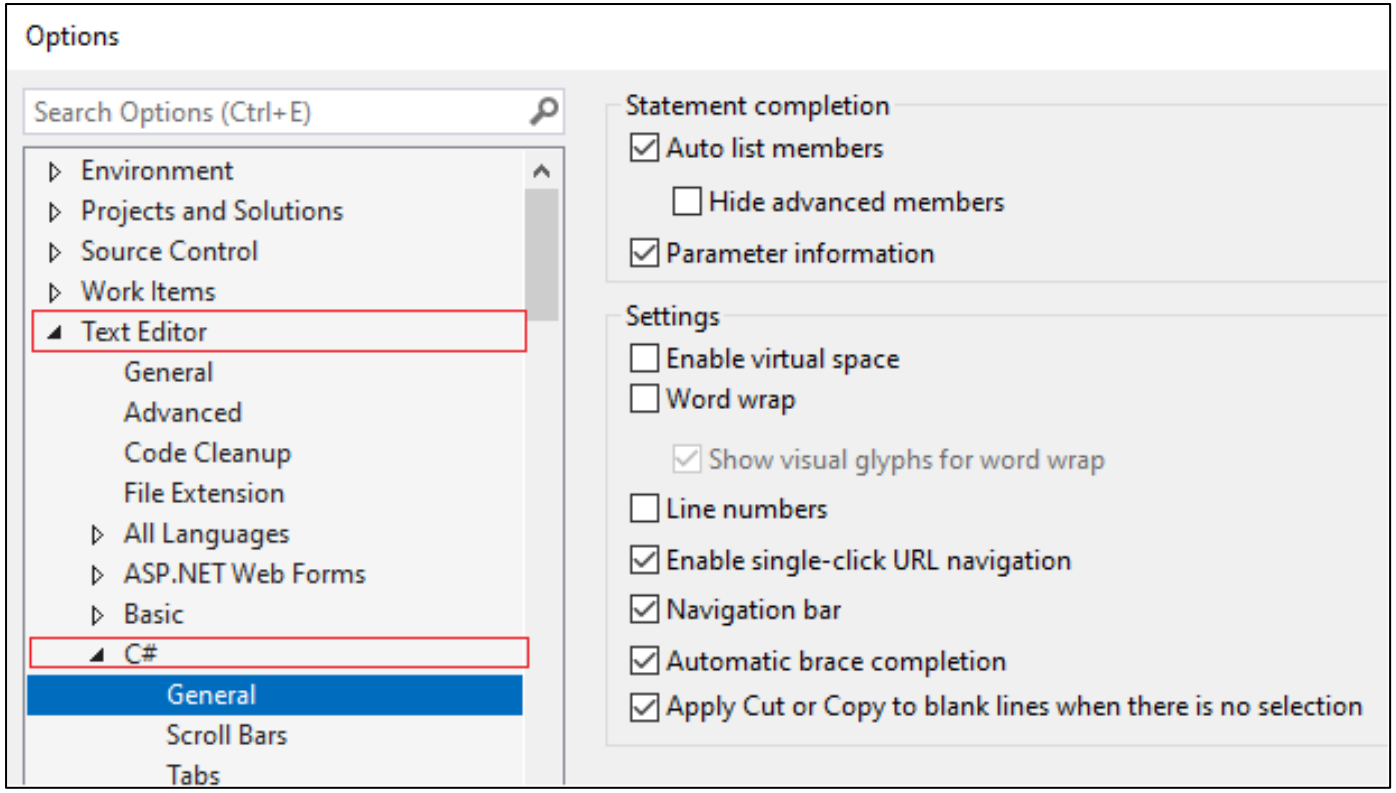
- Drag and drop of the selected text can be disabled
- Subword selection can be enabled
- Disable .**editorConfig** files support for your Visual Studio
- Configure Go to Definition with mouse click
  - Enable peek view
- Enable the display of tabs and whitespaces

C# Settings

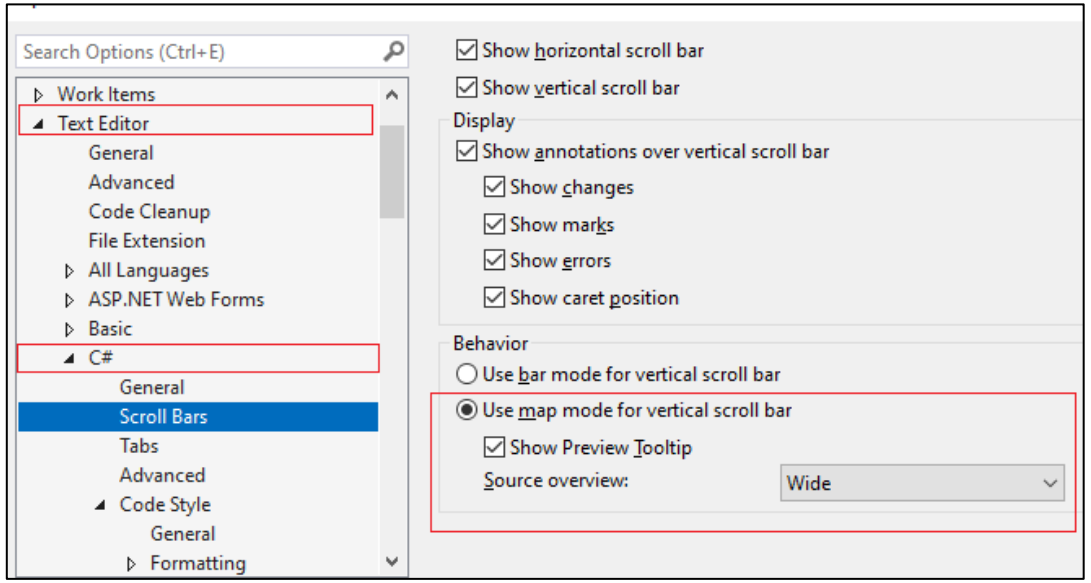
- Many languages have their own section with language specific settings
- All Languages allows to configure common settings like indentation for all languages

Interesting C# specific settings:

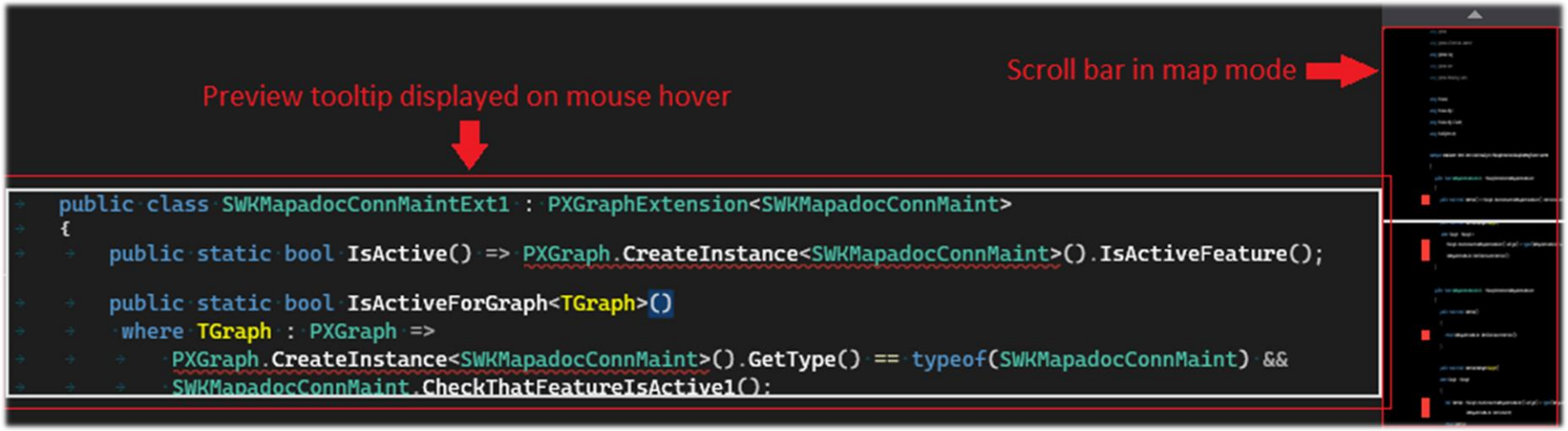
- Enable virtual space at the end of the line
- Enable word wrap
- Disable line numbers



Scroll Bar



- Scroll bars can be configured independently for any language or in a uniform way for all languages
- All Languages allows to configure common settings like indentation for all languages
- A map mode for vertical scroll bar will draw a small overview of the opened file in the vertical scroll bar
- If the preview tooltip is enabled, it will display a code fragment from the file on mouse hover



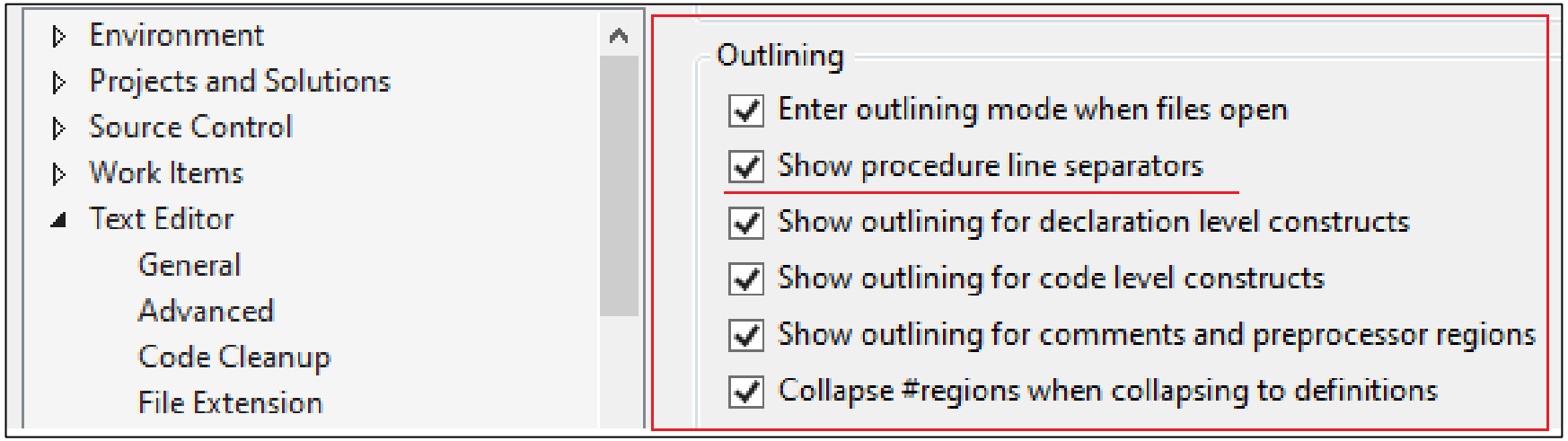
Line Separators

Line Separators are gray horizontal lines that separate different types and type members

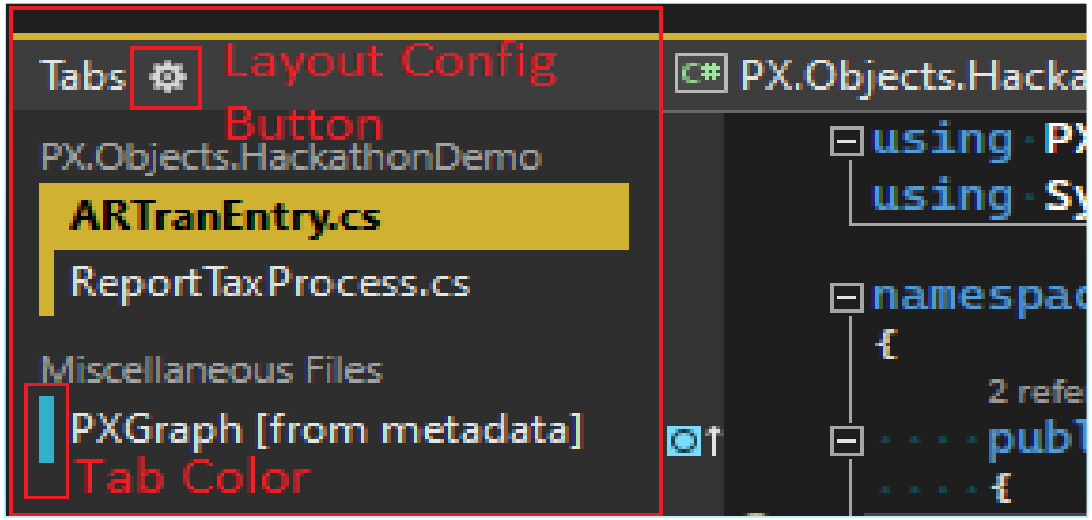
There must be at least one empty lines between adjacent code elements.

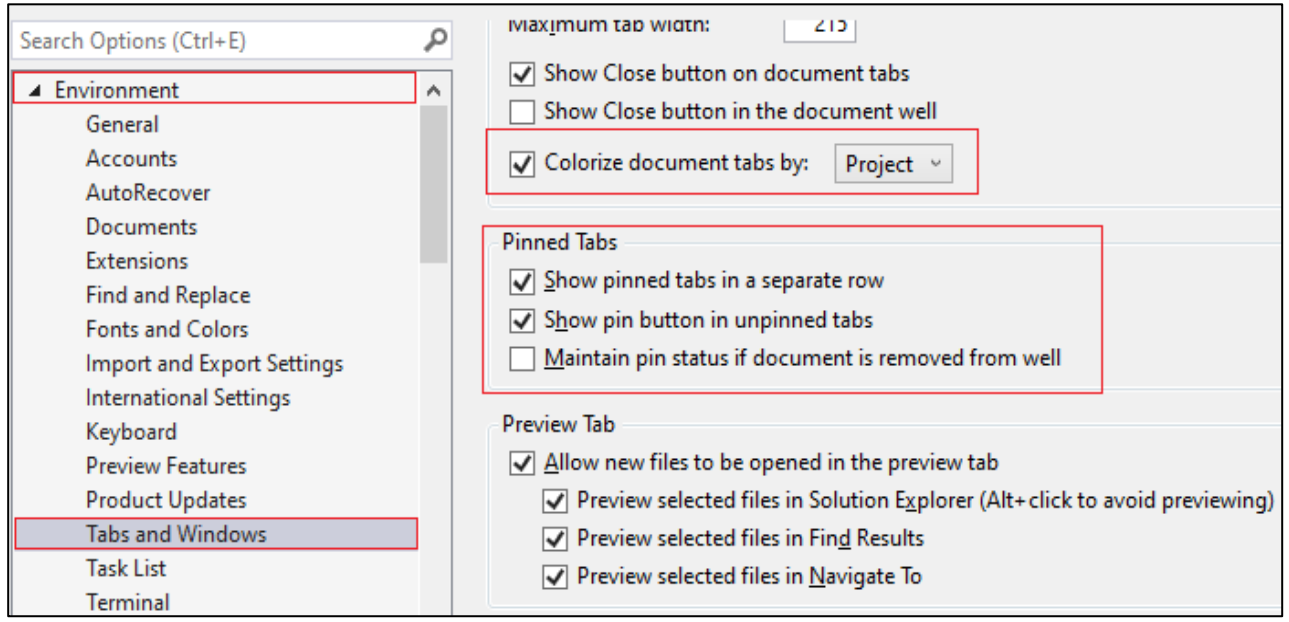
```
}

0 references | 0 changes | 0 authors, 0 changes
protected virtual void _(Events.FieldSelecting<APPayment, APPayment.curyID> e)
{
    e.ReturnValue = CuryIDFieldSelecting<APRegister.curyInfoID>(e.Cache, e.Row);
}
```

Switch to enable Line Separators is located in the Outlining settings



Document Tabs





- **Document Tabs** – tabs with opened documents that allow you to switch between them
- Document Tabs support behavior of any Visual Studio tool window (drag and drop, docking, etc.)
- Document Tabs layout can be changed. You can place them on top of the editor, to the left or to the right
- Settings are located in "Options" → "Environment" → "Tabs and Windows"
- Document Tabs can be colorized based on document's project of file extension
- Document Tabs can be pinned. When Document Tabs are located on top of the code editor Pinned Tabs can be placed in a separate row

Visual Studio: Writing Code

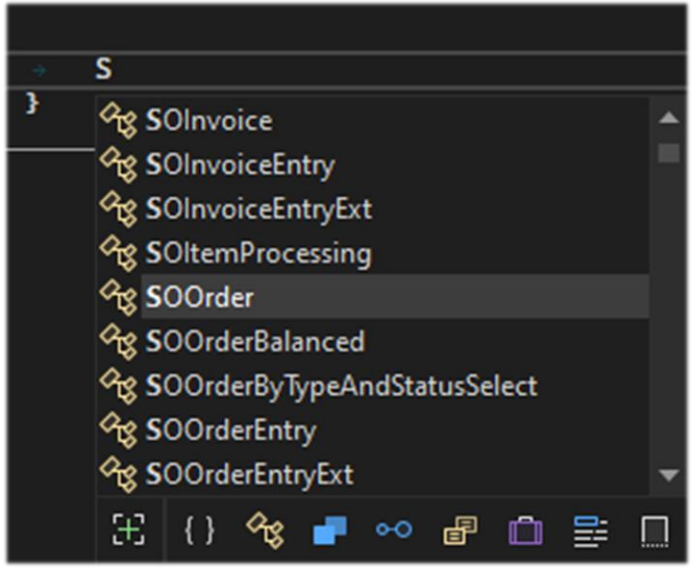
Overview

Different ways to boost our code writing skills:

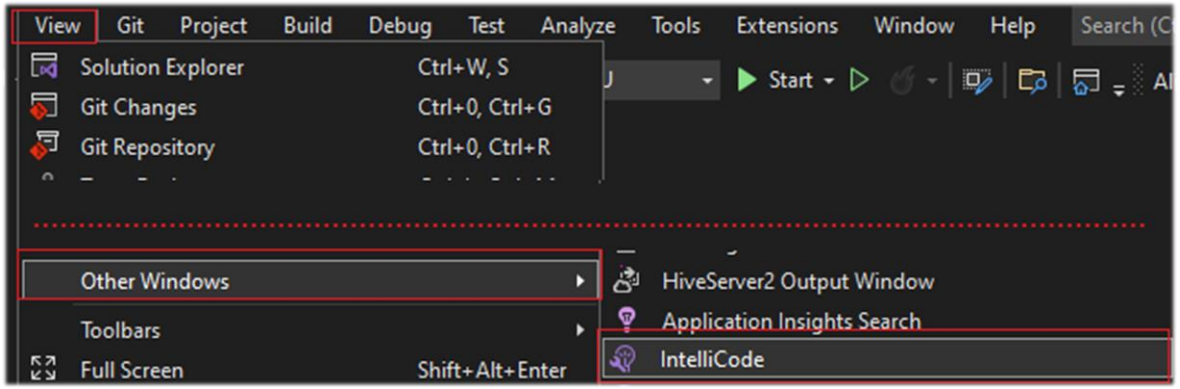
- ❑ Code Completion
- ❑ Clever suggestions made with ML models
- ❑ Refactorings (renaming) and Code Fixes
- ❑ Code Snippets
- ❑ Keyboard Shortcuts
- ❑ Multiple Carets
- ❑ Code generation
- ❑ Automatic Formatting and Code Cleanup
- ❑ Other things

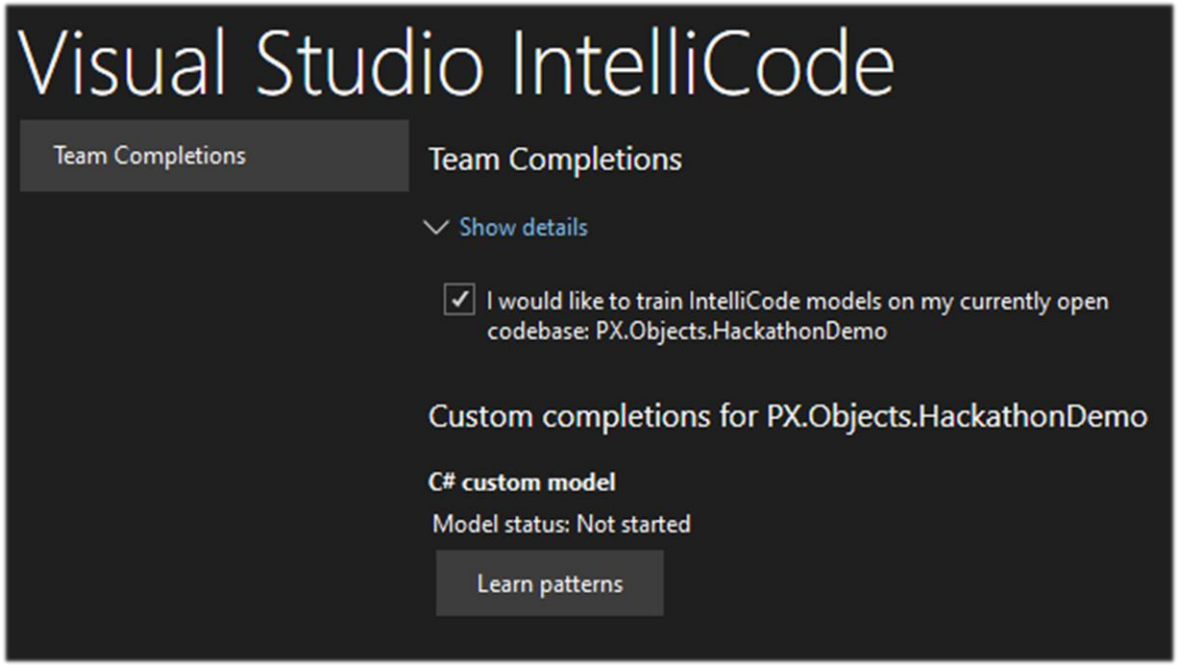
IntelliSense Code Completion

- **IntelliSense –** advanced code completion VS functionality
- IntelliSense suggests possible options while you type
- Suggestions may include types, local variables, method parameters, type members, extensions methods and code snippets
- The list of variants depends on the context. It is affected by the programming language, framework, type of the document, caret location and some other things.
- Settings for C# are stored in Visual Studio options:
  - “Text Editor” → “C#” → “General”
  - “Text Editor” → “C#” → “IntelliSense”
- IntelliSense commands can be accessed in “Edit” → “IntelliSense” top level menu



IntelliCode Model Training

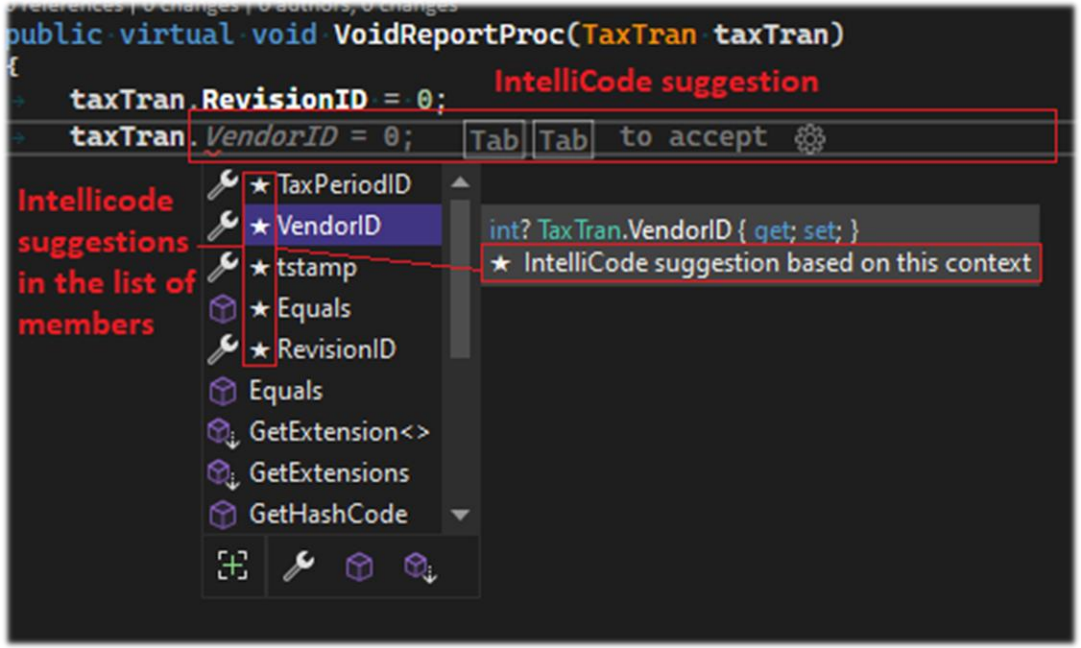


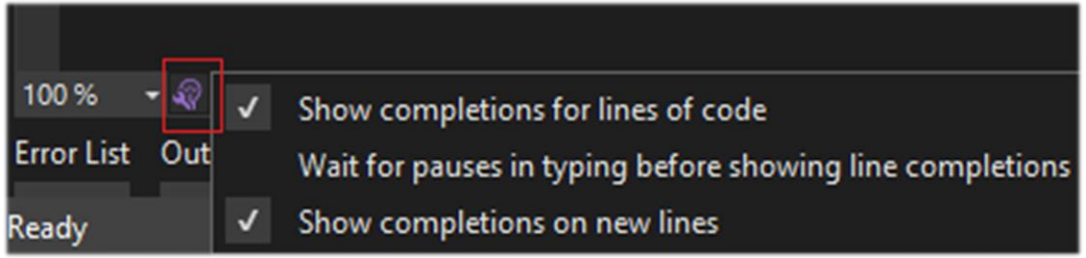


- **IntelliCode –** machine learning Visual Studio functionality that relies on predictive model trained on GitHub open-source repos
- IntelliCode trains ML models on your code base to provide context-specific suggestions
- IntelliCode model training functionality is located in:
  "View" → "Other Windows" → "IntelliCode"
- Trained models can be shared with your team
- Retraining can be done as a step in the CI pipeline

IntelliCode ML Features

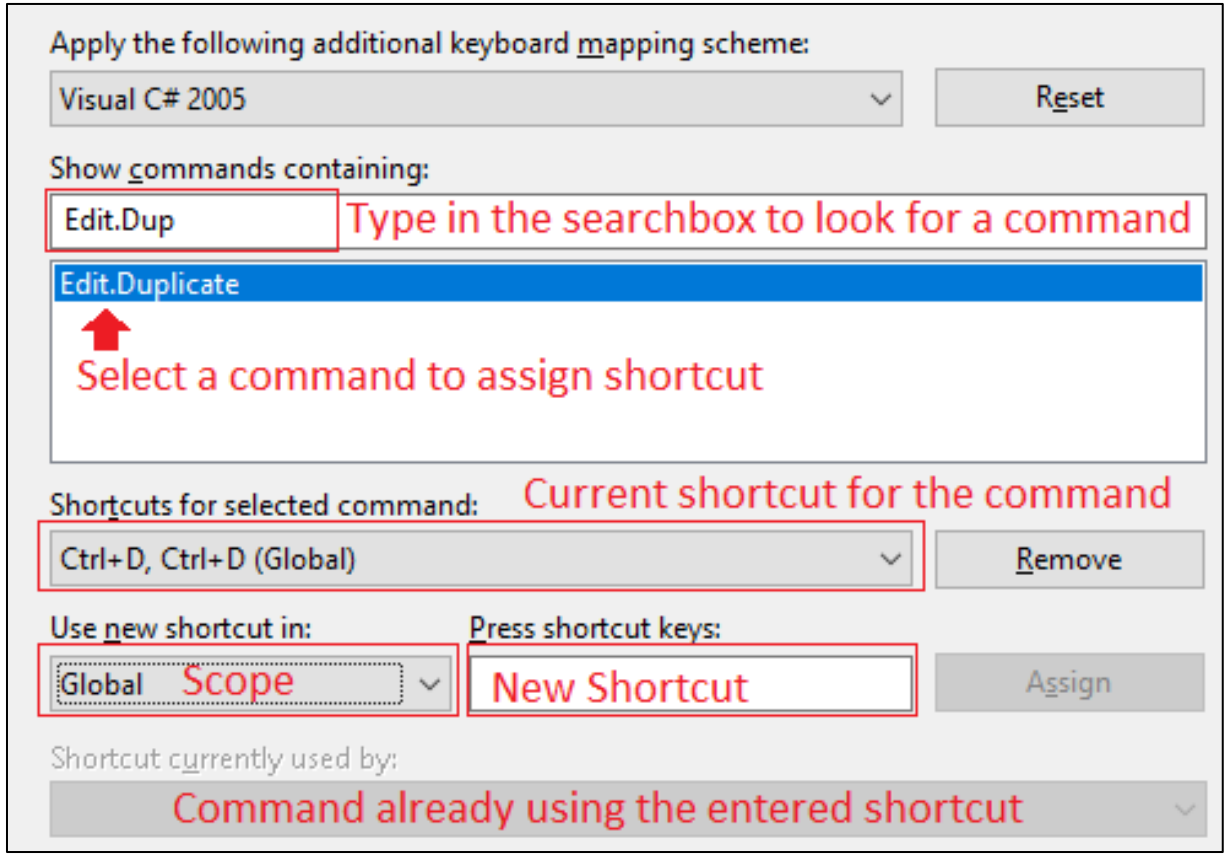
- **AI-assisted IntelliSense** – IntelliSense code completion extension that predicts API that are most likely to be used
  - Suggestions are marked with a star and placed first in the list of available variants
  - Argument completion for C#
  - Suggestions based on your own code for C#
- **Refactoring suggestions** for C# – track your edits, detect repetitive actions and offer to apply these actions in similar places
- **Whole line completions** – predict the next chunk of your code and present it as a gray text inline prediction
  - Available only in Visual Studio 2022
  - Can be configured by pressing lightbulb at the bottom right of the editor
- **.editorconfig file generation** with code style and formatting conventions inferred from the C# code base





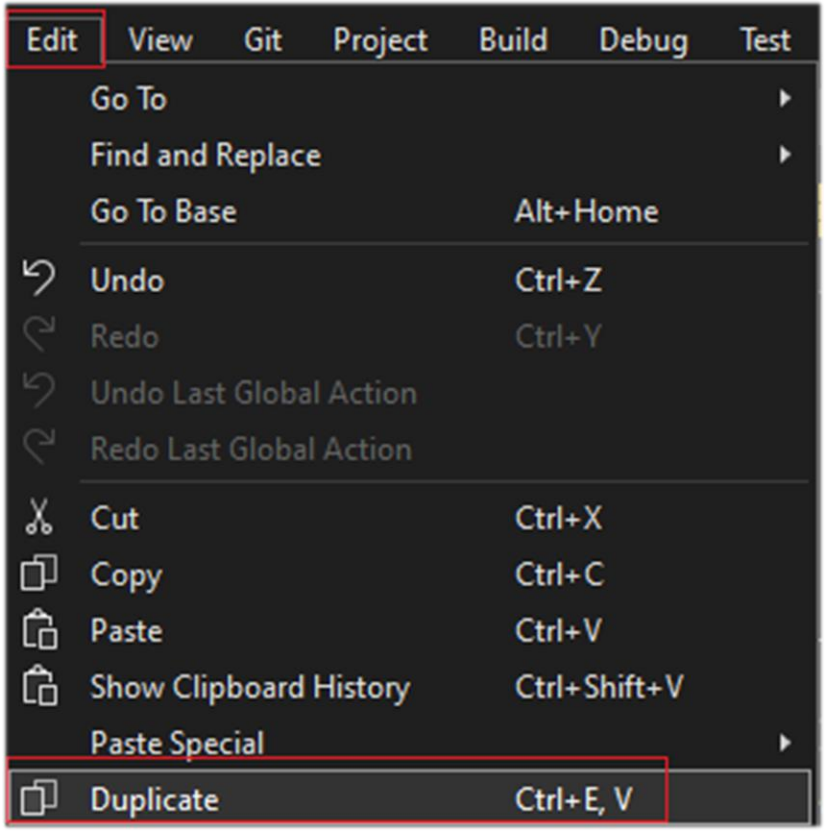
Keyboard Shortcuts

- **Keyboard shortcuts** allow user to quickly perform some action by pressing a combination of keys on keyboard
- Well-known examples:
  - Copy / Paste with Ctrl + C / Ctrl +V
  - Cancel with Ctrl + Z
  - Search / Replace with Ctrl + F / Ctrl + H
- Most Visual Studio commands support shortcuts and have a default shortcut assigned
- Plugins may provide their own shortcuts
- Custom shortcuts can be assigned to Visual Studio command even if it does not have a default one
- Shortcut settings can be accessed in Visual Studio options:
  "Environment" → "Keyboard"



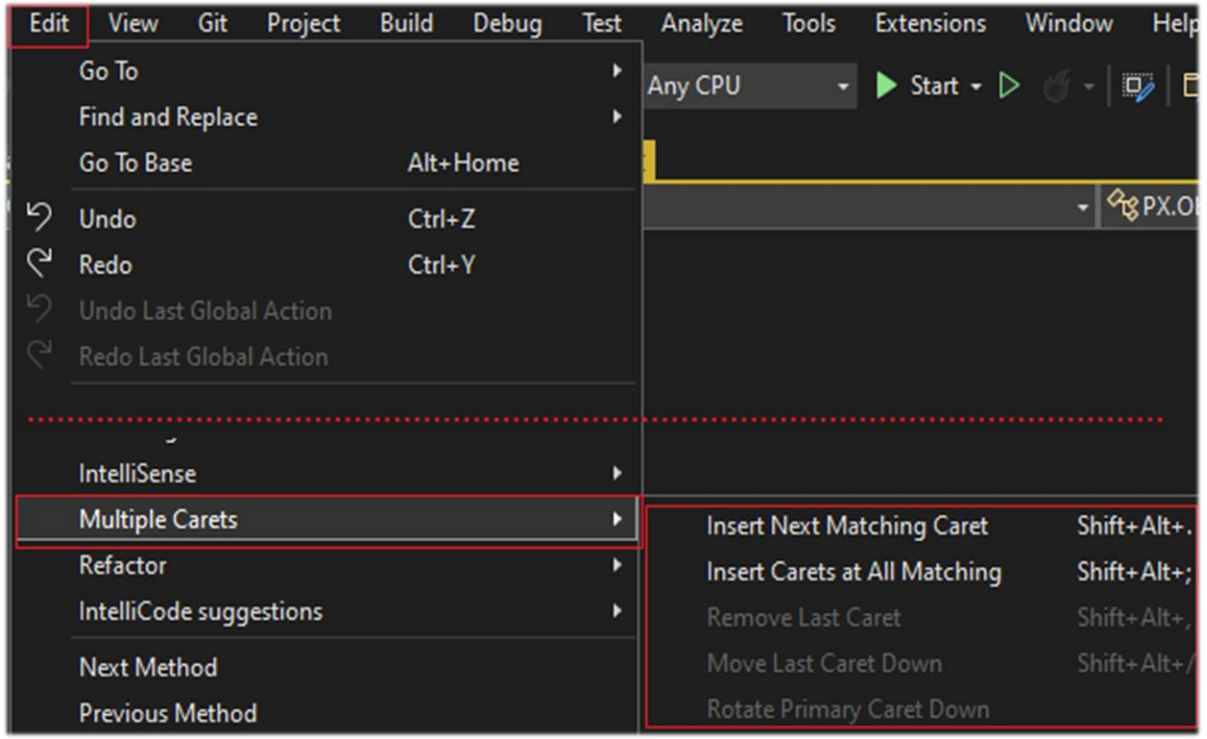
Useful Keyboard Shortcuts

- Duplicate selected line with **Duplicate** command
  - Can be accessed with “Edit” → “Duplicate”
  - Look for **Edit.Duplicate** command in shortcut settings to assign a shortcut
- Change indent of selected line
  - Increase indent with Tab
  - Decrease indent with Shift + Tab
- Call IntelliSense code completion suggestions with Ctrl + Space
- Move selected lines up and down
  - Move up with Alt + Up arrow
  - Move down with Alt + Down arrow
- Other scenarios
  - Custom shortcuts for Visual Studio extensions
  - Text selection
  - Multiple Carets
  - Other



Multiple Carets

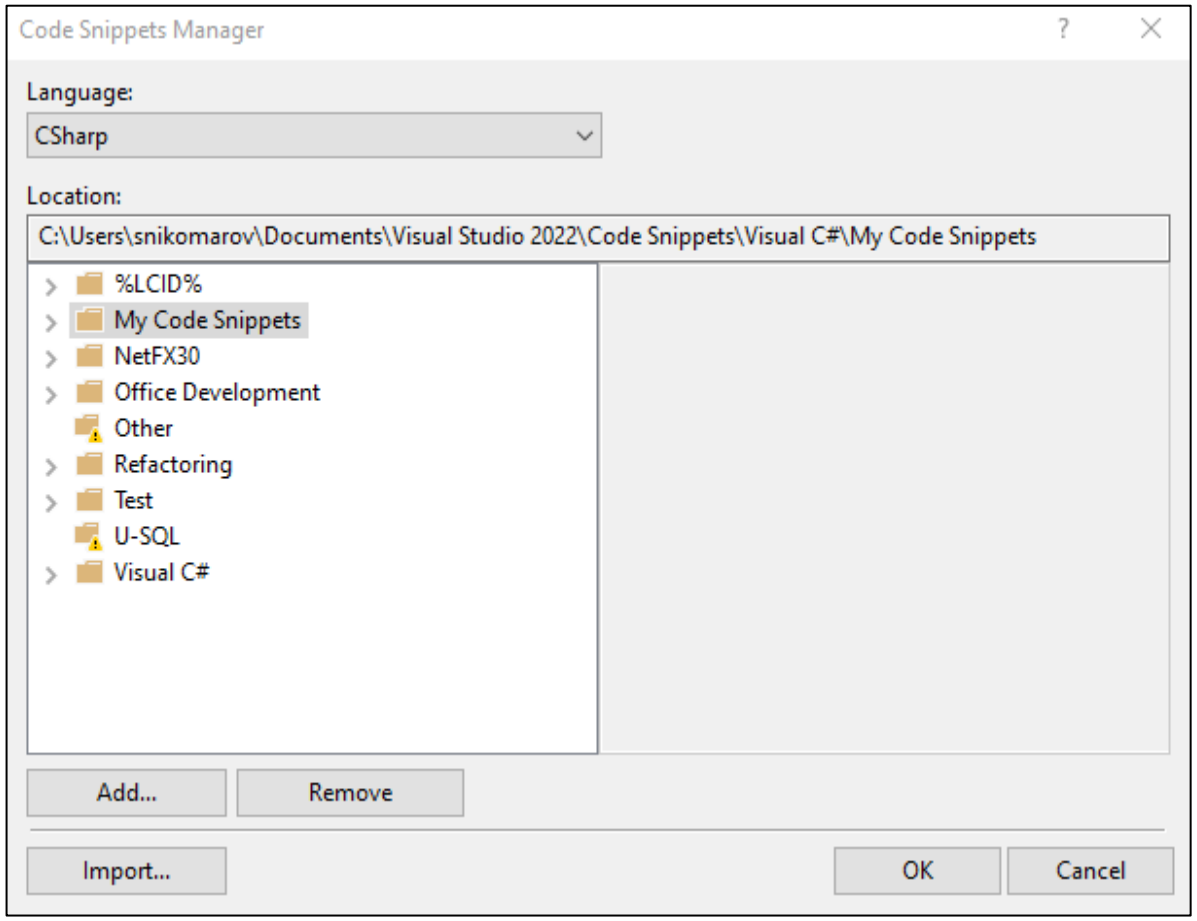
- **Multiple Carets** allows you to edit multiple places of your code by creating multiple carets and applying changes to all of them simultaneously
- Multiple Carets works with the text selected in the code editor. It finds occurrences of the same text in the document and creates extra carets for them
- Carets can be created for all occurrences of the selected text or only for some of them
- Multiple Carets can be found in "Edit" → "Multiple Carets"
- Shortcuts are much more convenient
  - Insert Carets at All Matching **Shift + Alt + ;**
  - Insert Next Matching Caret **Shift + Alt + .**



Code Snippets Overview

- **Code Snippets** – small blocks of reusable code that can be quickly inserted into your document
- Snippets can contain placeholders with different names that you replace with your code.
- The replacement will be done for every instance of the same placeholder in the inserted snippet
- Two types of code snippets
  - **Expansion** snippets inserted at the cursor position
  - **Surround With** snippets that surround selected code
- Extra features:
  - Generate cases for every member of enum in a code snippet for the **switch** statement
  - Add missing using for namespaces
  - Add references to assemblies from Global Assembly Cache (GAC)

Managing Code Snippets

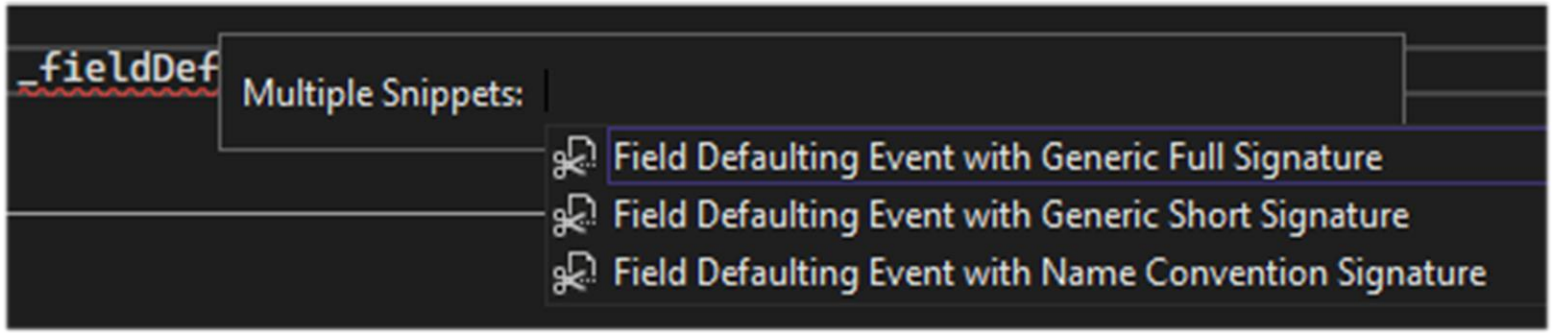


- Visual Studio comes with a lot of code snippets for different programming languages and frameworks
- You can write your own code snippets and import them in Visual Studio
- Code Snippets are XML files with **.snippet** extension
- Custom code snippets are stored in Visual Studio subfolder in current user documents:
  <Current User Folder>\Documents\<Visual Studio version>

  There should be a dedicated subfolder:
  \Code Snippets\Visual C#\My Code Snippets
- You can manage code snippets in the “Code Snippets Manager” “Tools” → “Code Snippets Manager”

Acumatica Code Snippets

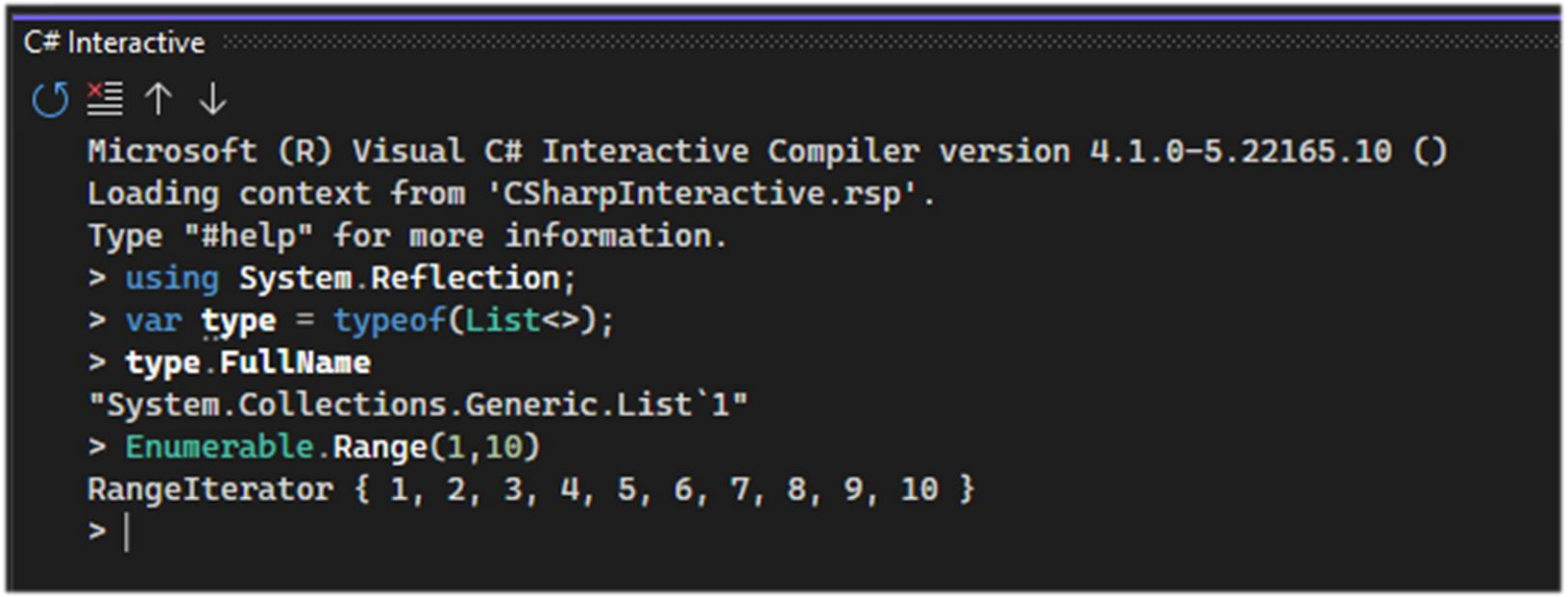
- Acumatica Code Snippets – code snippets developed for Acumatica Framework
- Acumatica Code Snippets include templates for
  - DAC
  - DAC fields
  - Graph field and row events
    - Name Convention Signature
    - Generic Signature
    - Short Generic Signature for field events



- Available on GitHub: https://github.com/Acumatica/CodeSnippets
- Acumatica Code Snippets can be imported into Visual Studio manually
- Acuminator next version will include Acumatica Code Snippets collection

C# Interactive

- **C# interactive** is a useful tool for quickly running a piece of code
- C# Interactive is based on C# read-evaluate-print-loop (REPL) functionality
- Located in “View” → “Other Windows” → “C# Interactive”
- You can run it either line by line or from a script



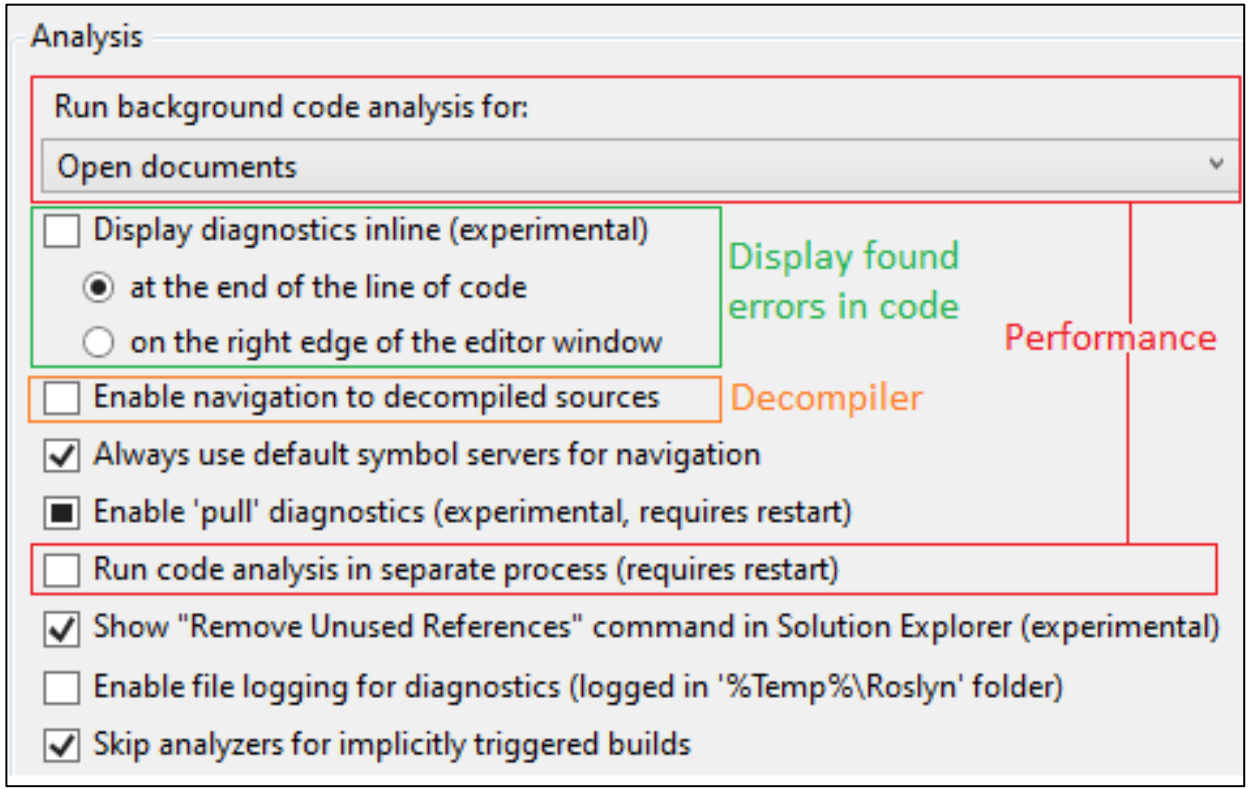
Visual Studio: Code Analysis

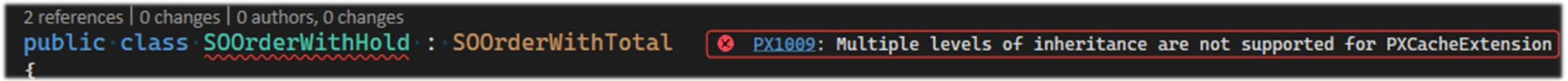
Overview

- Code analysis for C# and VB in modern versions of Visual Studio is based on Roslyn framework
- Library authors can provide custom diagnostics verifying that their API is used correctly
  - xUnit
  - Acuminator
- Advanced general-purpose diagnostics developed by community as open-source projects
- Many ways to suppress a particular error – pragma directives, special attributes, project settings, .editorconfig and .ruleset files
- Two ways to add custom diagnostics to the solution
  - Install Visual Studio extension to use diagnostics only for yourself
  - Add a Nuget package with diagnostics to the project to enforce rules for everyone in the team

Code Analysis Settings

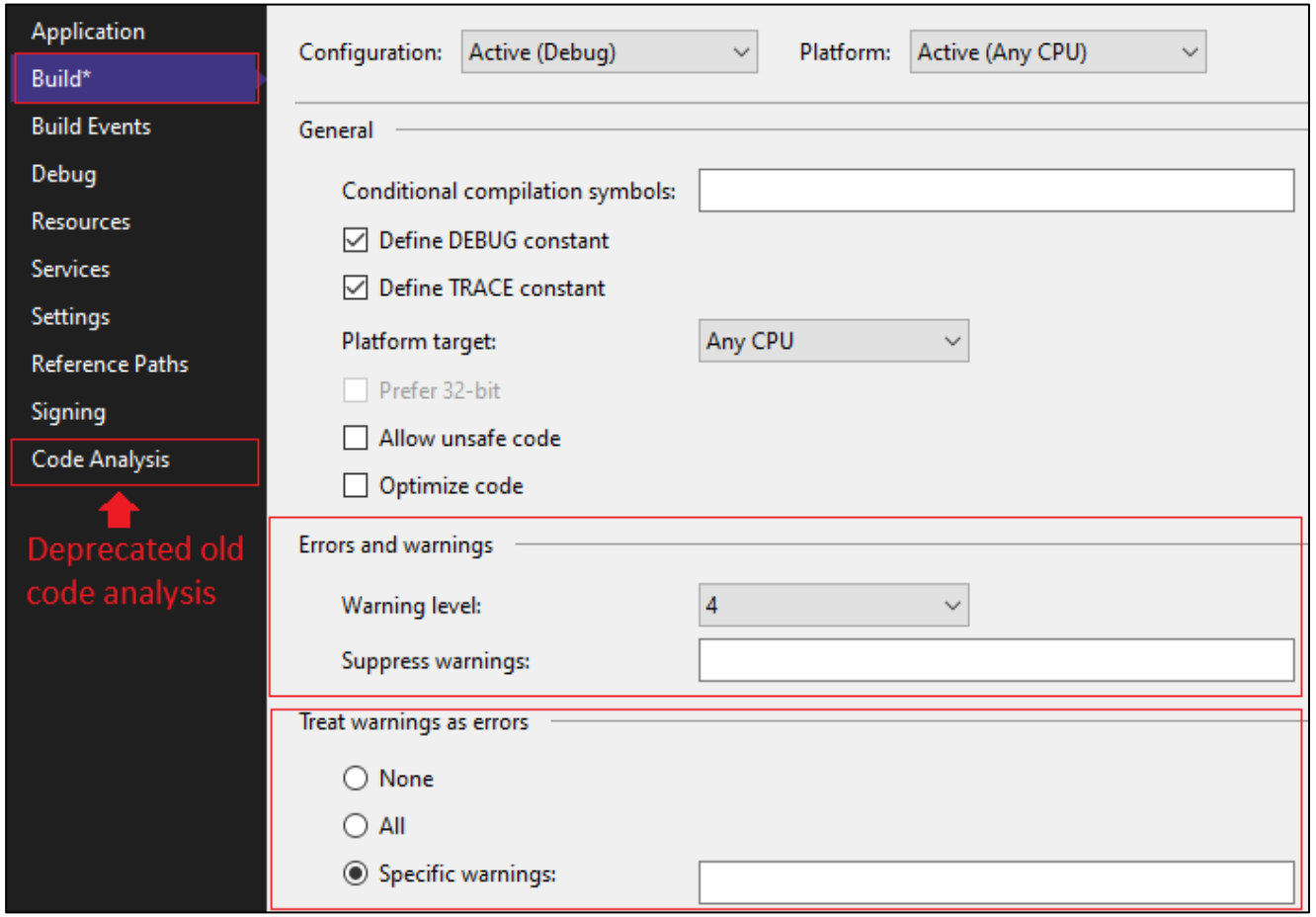
- Each language has its own code analysis settings
  - C# settings can be found in
    “Text Editor” → “C#” → “Advanced” → “Analysis”
- Some settings affect Visual Studio performance
  - Background analysis scope
  - Run code analysis in a separate process in VS 2022
  - Use 64-bit process for code analysis in VS 2019
- Enable built-in decompiler with “Enable navigation to decompiled sources”
- Try experimental “Display diagnostics inline” feature





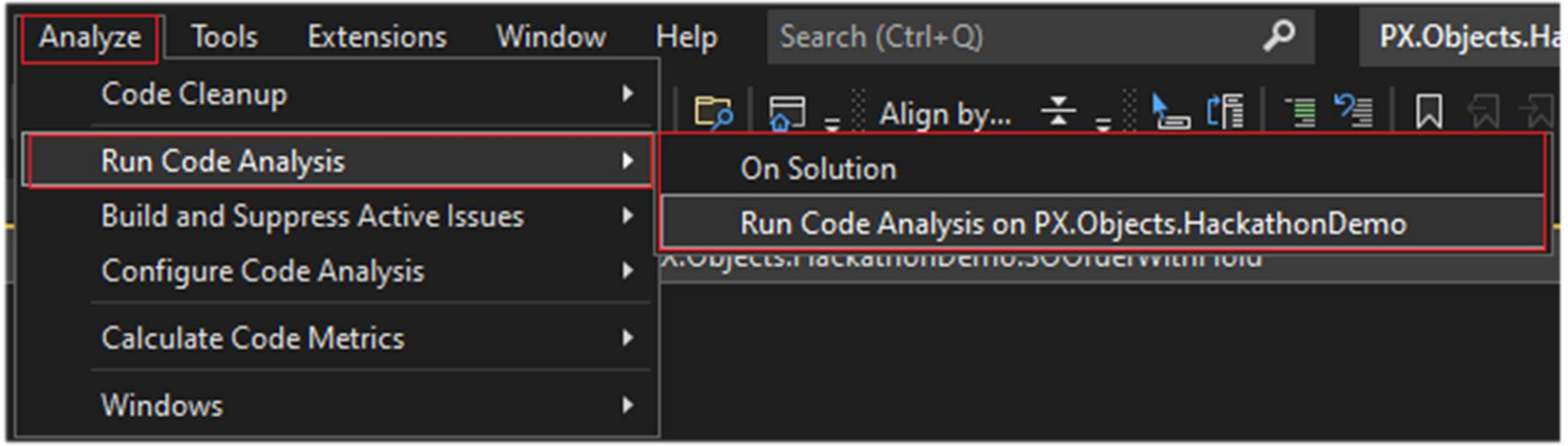
Project specific settings

- .Net projects have project specific code analysis settings
- Some code analysis settings can be accessed in project “Properties” window
- Code Analysis section is related to a legacy FxCop code analysis and is deprecated
- Build section contains analysis settings
  - Warning level. For higher levels compiler provides more warnings
  - Disabled warnings
  - Treat warnings as errors
- More settings can be specified directly in the project file



Manually Run Code Analysis on Solution

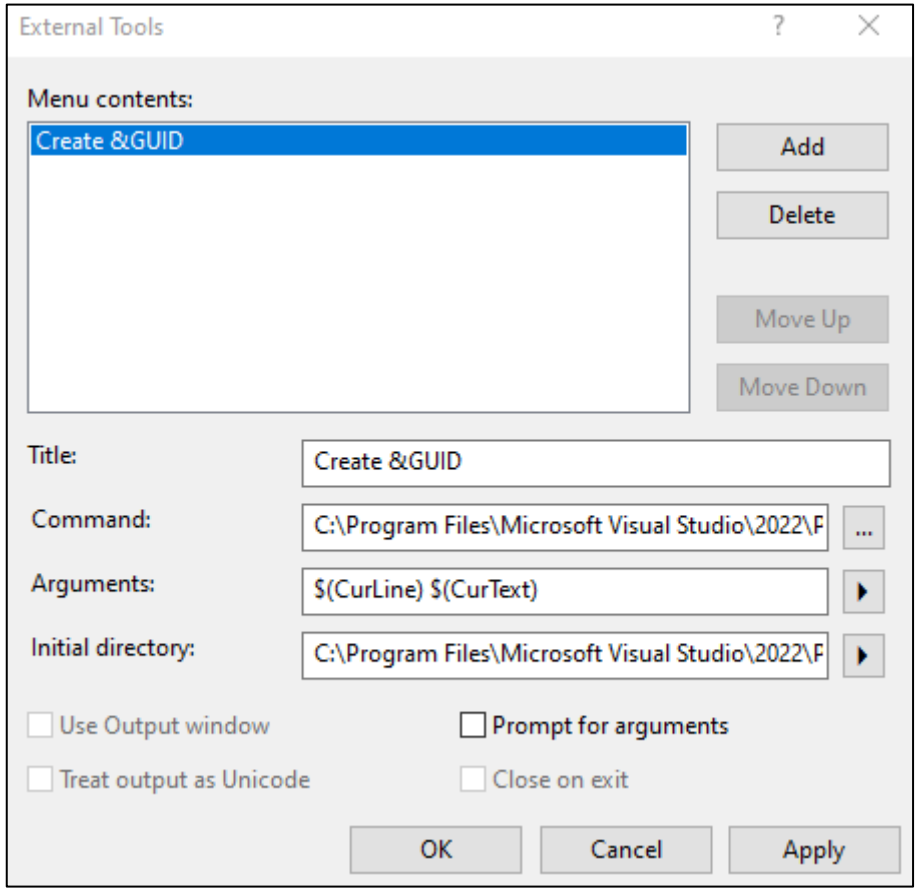
- Commands to run code analysis manually are located in "Analyze" → "Run Code Analysis"
- You can run code analysis for the current project of for the entire solution
- All errors are listed in the "Error List" window
- Works in latest versions of Visual Studio 2019 and 2022



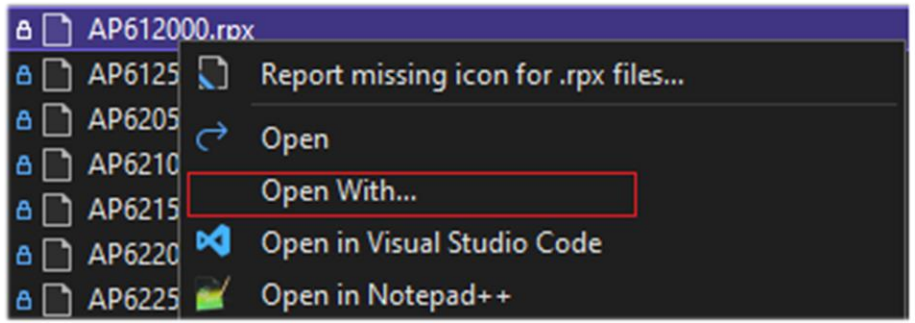
Integration with Other Tools

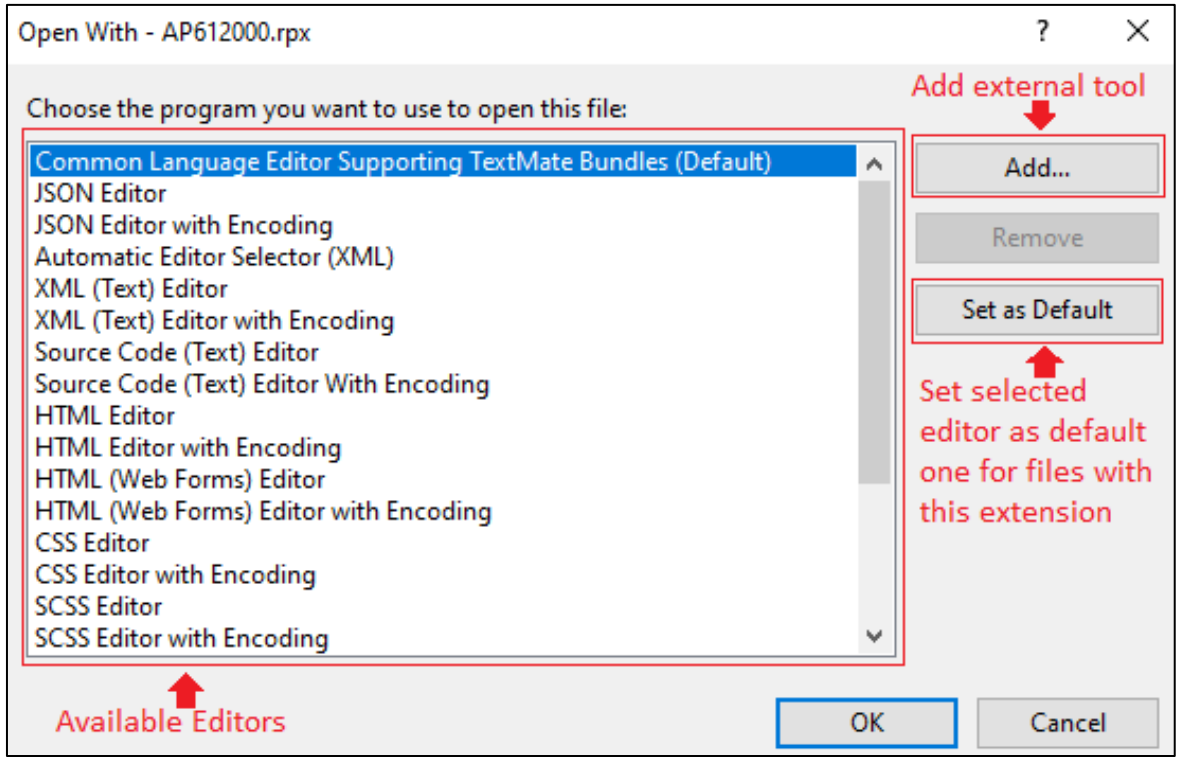
Different Integration Approaches

- Integrations built-in into Visual Studio
  - Git and other Version Control Systems
  - SQL Server
  - Unity and other game engines
  - Unit test frameworks
- Integrations provided by plugins
  - “Open with XXX” plugins that allow you to open file in a different tool
  - Support for different frameworks and tools
- Add tool to Visual Studio External Tools list located in “Tools” → “External Tools”
- Specify external tool as a designer for files with specific extension



External Tool Editor

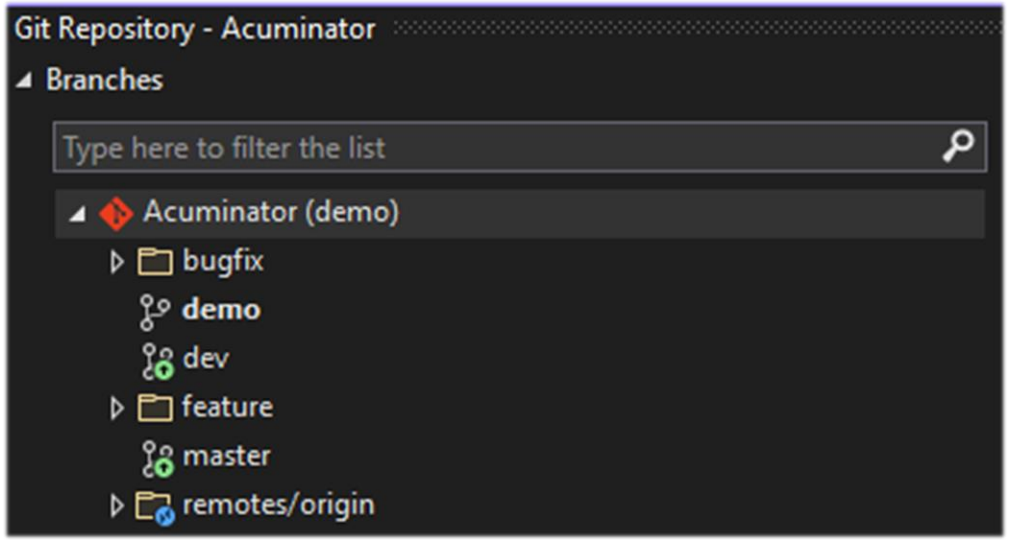


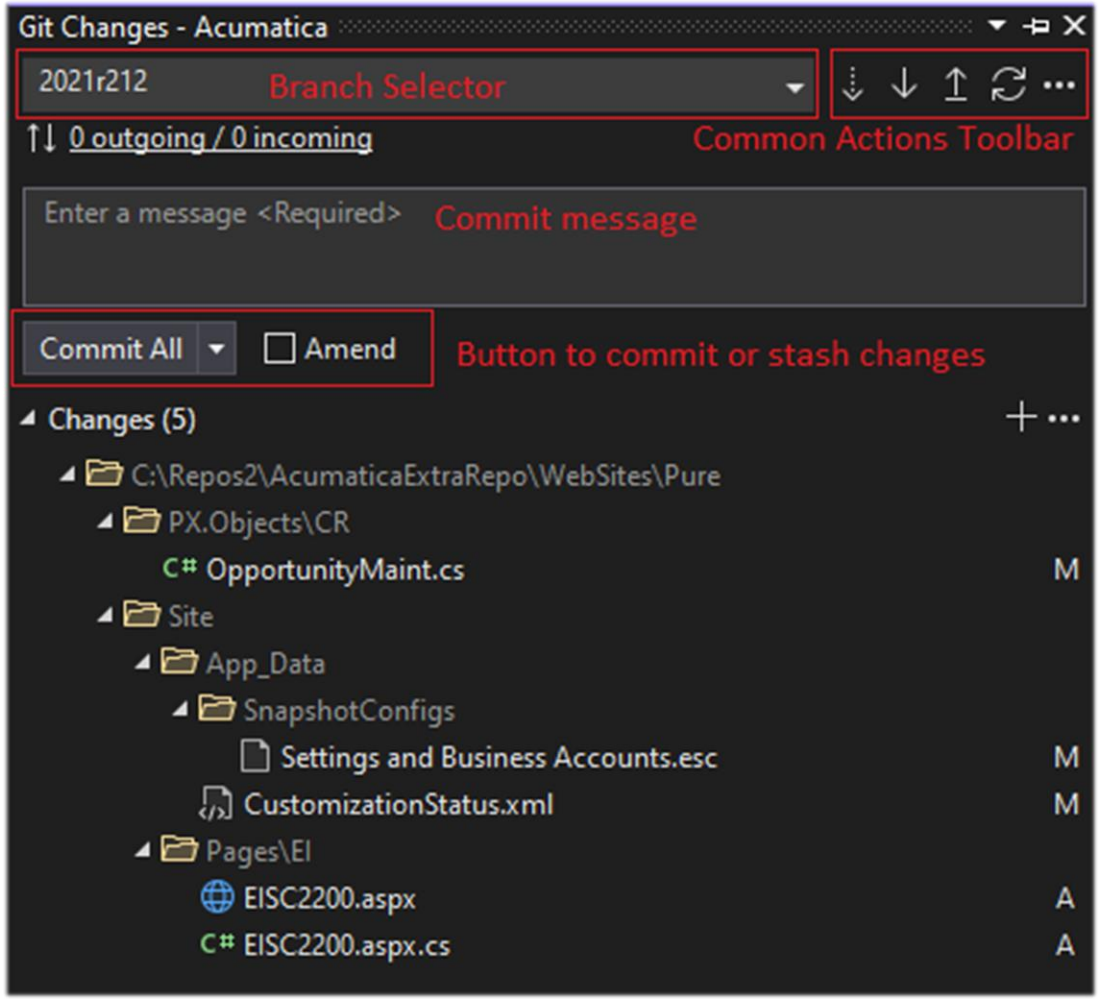


- Visual Studio allows you to choose which tool should be used to open files with specific extension
- The editor selection dialog is located in “Solution Explorer” window in the context menu
- Select required file, make right mouse button click on it and select “**Open With…**” command
- There are many available editors shipped with Visual Studio that can be used to open the file
- External tools can be added to available editors
- One of the editors can be selected as a default editor for files with the same extension
- Report Designer can be set as a default editor for Acumatica .rpx reports

Visual Studio Git Integration

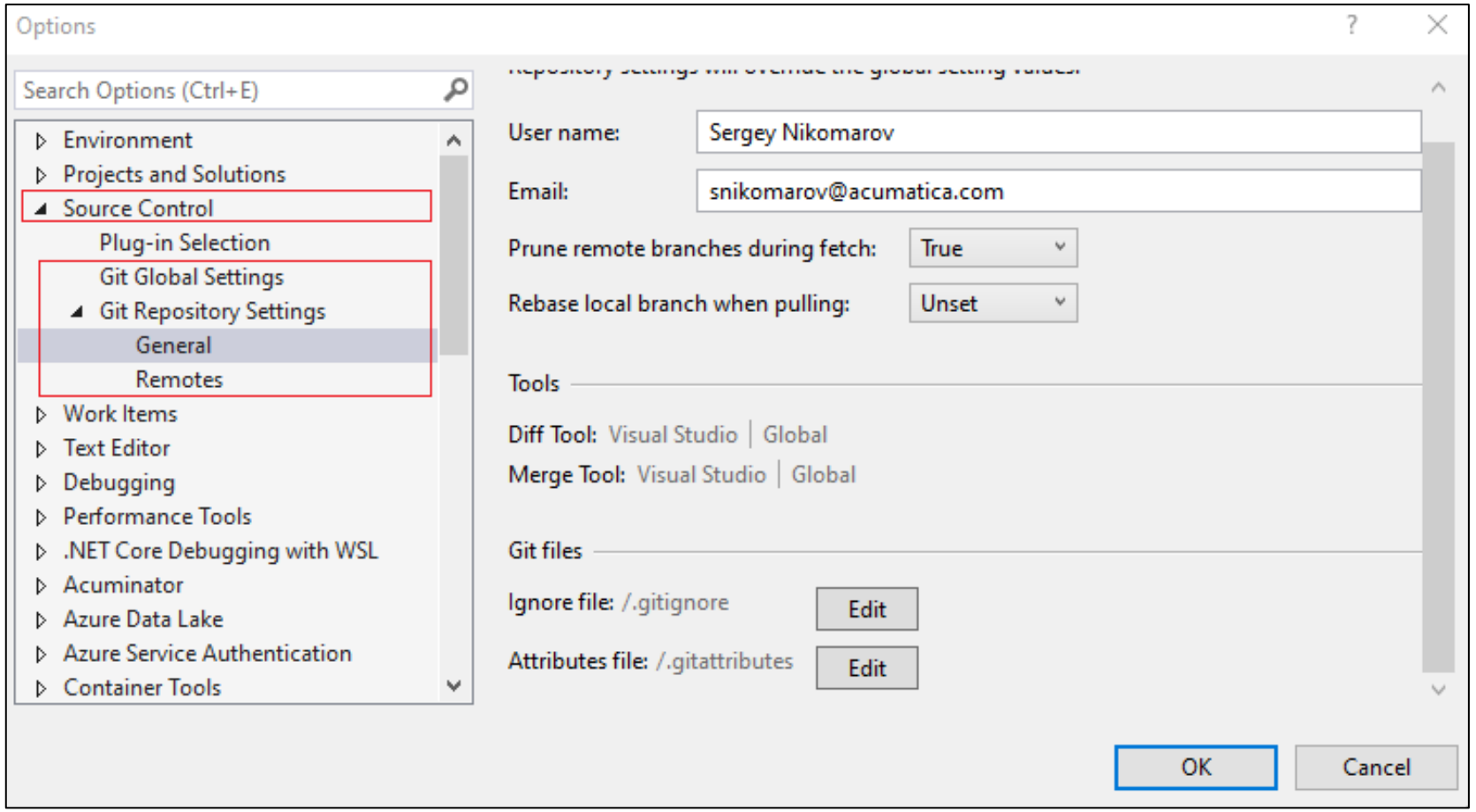
- **Git** – the most popular version control system used by many projects for collaborative development
- Git is a console application but there are many UI clients
- Visual Studio Git Integration is another UI client for Git
- Repository settings, branches and changes are displayed
- Simple Git workflow actions can be used from Visual Studio





Git Settings

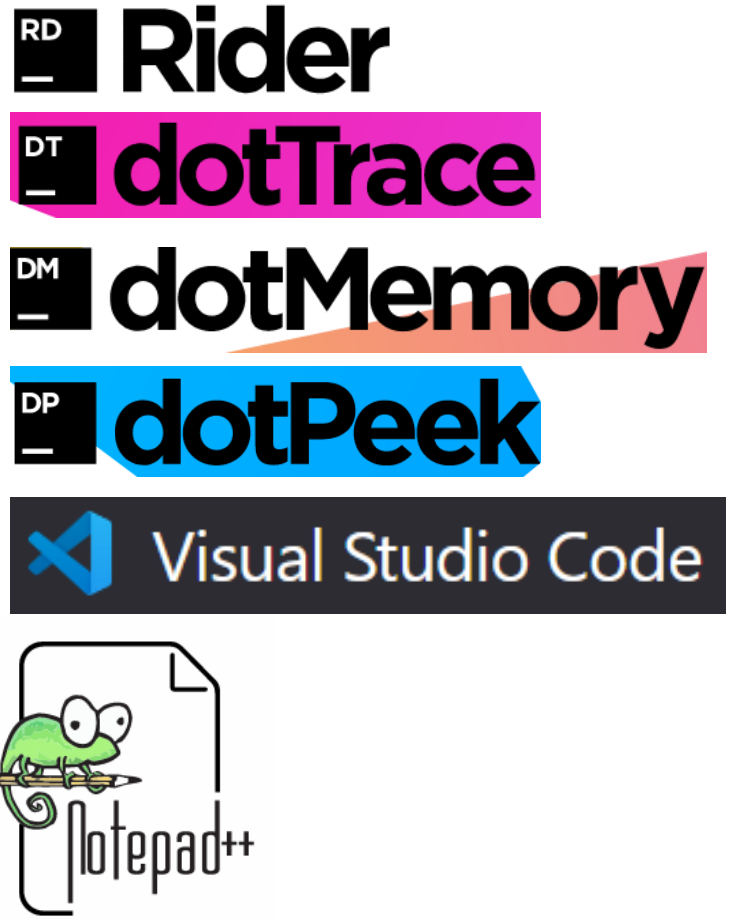
- Git settings are located in Visual Studio options in “Source Control” section
- Two sets of settings – global settings for all repositories and local repository settings



Other Developer Tools

Alternatives to Visual Studio

- Jet Brains tools
  - **Rider** IDE
  - **dotTrace** and **dotMemory** for profiling
  - **dotPeek** decompiler
- **Visual Studio Code**
- **Notepad++**



Useful Visual Studio Extension

- **ReSharper** and **OzCode**
- Code Analysis extensions
  - **Roslynator** – 500+ analyzers, refactorings and code fixes for C#
  - **StyleCop** – rules to enforce style and consistency rules for source code
- **Visual Studio Spell Checker** and other spell-checking extensions
- **Debug Attach Manager** for easy attaching to IIS processes
  - **ReAttach** as alternative
- Syntax highlight extensions – **Viasfora**, **Codinion** and many others
- Productivity extensions
  - Open in XXX extensions – **Open in Visual Studio Code, Open in Notepad++**
  - **Match Margin / Match Margin 2022** to display occurrences of the selected word in scroll bar
  - **Code alignment** aligns your code vertically to improve readability
- **Atomineer Pro Documentation** for clever generation of XML doc comments
  - **GhostDoc** as alternative

Summary

Tools that boost developer productivity, ensure product quality and take care of routine tasks

- Acuminator – new version with support of Visual Studio 2022, improved code checks and new data displayed on Code Map
- Visual Studio – know your code editor!
  - Customize appearance
  - Enable advanced features and optimize performance
  - Write code faster
  - Configure code analysis
  - Integrate with other tools
- Other Tools – useful plugins and alternatives to Visual Studio

Resources – Acuminator

- Acuminator Release Notes:
  https://github.com/Acumatica/Acuminator/blob/dev/docs/ReleaseNotes.md

- Overview of all diagnostics:
  https://github.com/Acumatica/Acuminator/blob/dev/docs/Summary.md

- Acuminator is an open-source project. The list of issues:
  https://github.com/Acumatica/Acuminator/issues

Resources – Visual Studio

- Visual Studio Blog
  https://devblogs.microsoft.com/visualstudio/
- Editor features overview
  https://docs.microsoft.com/en-us/visualstudio/ide/writing-code-in-the-code-and-text-editor?view=vs-2022
- Keyboard shortcuts
  https://docs.microsoft.com/en-us/visualstudio/ide/identifying-and-customizing-keyboard-shortcuts-in-visual-studio?view=vs-2022
- Tips for C# developers
  https://docs.microsoft.com/en-us/visualstudio/ide/csharp-developer-productivity?view=vs-2022
- IntelliCode
  https://docs.microsoft.com/en-us/visualstudio/intellicode/intellicode-visual-studio
- Code Snippets
  - C# Code Snippets
    https://docs.microsoft.com/en-us/visualstudio/ide/visual-csharp-code-snippets?view=vs-2022
  - Snippets for Acumatica
    https://github.com/Acumatica/CodeSnippets

Acumatica
The Cloud ERP

Thank You!

www.acumatica.com/developers