Introducing JetBrains dotPeek

Redirection Notice
This page will redirect to NETCOM:Introducing JetBrains dotPeek.

What is dotPeek

dotPeek is a free .NET decompiler and assembly browser from JetBrains, the makers of ReSharper, dotTrace, and dotCover for .NET developers, as well as a family of IDEs for Java, Ruby, Python, PHP, and other languages, plus team development tools: TeamCity for continuous integration and build management and YouTrack for issue tracking.

The main idea behind dotPeek is to make high-quality decompiling coupled with powerful ReSharper-like navigation and search features available to everyone in the .NET community, free of charge.

dotPeek has gone public for the first time in May 2011 as JetBrains opened an Early Access Program (EAP) that implied regular publishing of pre-release builds. dotPeek 1.0 has been officially released on May 10, 2012. The latest official version of the decompiler and assembly browser is always available at http://www.jetbrains.com/decompiler

JetBrains is also providing decompiling functionality as part of ReSharper, the renowned productivity tool for Visual Studio.

Distribution Options

dotPeek is available for download in two distributions: an .msi installer and a .zip archive.

The installer-based distribution is a safe bet if you want to use dotPeek on a single PC and take advantage of automatic version management.

The .zip distribution is a better fit if you prefer to share dotPeek between multiple machines - for example, using a Dropbox folder.

Both distributions are functionally equivalent except for the way you configure Windows Explorer integration.

General Features

For all features described below, default keyboard shortcuts are specified. By default, dotPeek uses the Visual Studio shortcut scheme derived from ReSharper. dotPeek also provides another shortcut scheme familiar to ReSharper users - that is, the IntelliJ IDEA scheme. You can switch between those schemes under Tools > Options > Environment > General.

All shortcuts defined in both Visual Studio and IntelliJ IDEA shortcut schemes are listed in a document called dotPeek Keyboard Shortcuts that is available from dotPeek via Help > Keyboard Shortcuts.

Opening and Browsing Files

dotPeek decompiles any .NET assemblies and presents them as C# code.

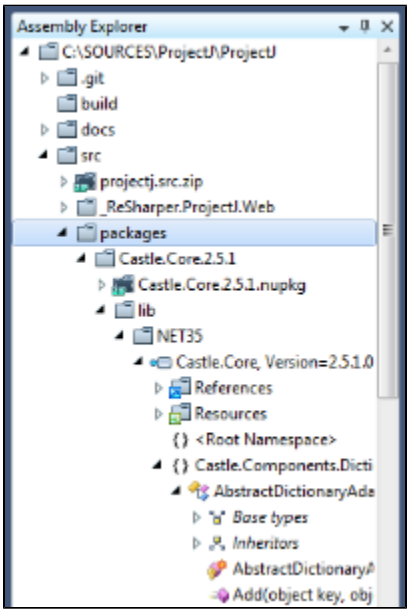
Supported file types include:

- Libraries (.dll)
- Executable files (.exe)
- Windows 8 metadata files (.winmd)
- Archives (.zip)
- NuGet packages (.nupkg)
- Microsoft Visual Studio Extensions packages (.vsix)

Files can be opened in one of the following ways:

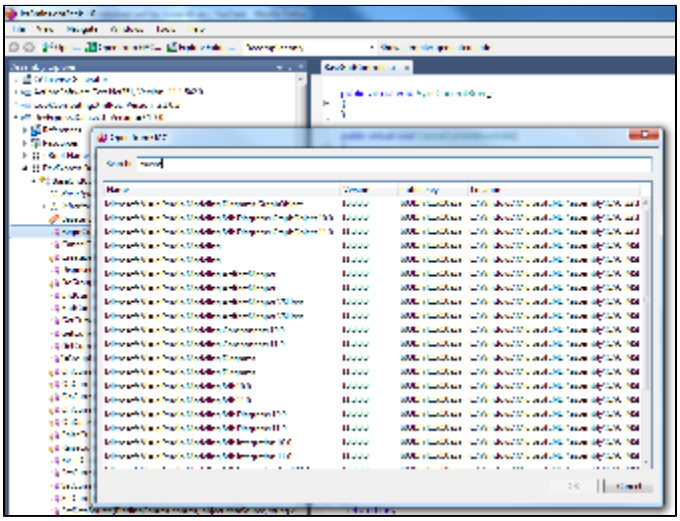
- Using the File > Open command.
- Via drag-and-drop of a file (or a selection of files) to dotPeek window.
- For .dll files: By double-clicking a file in Windows Explorer (provided that Windows Explorer integration is configured.)
- For .dll and .exe files: By right-clicking a file in Windows Explorer and selecting Browse with JetBrains dotPeek.

You can also load entire folders in the Assembly Explorer by choosing File > Explore Folder. When you ask dotPeek to explore a folder, it processes all its subfolders in hunt for files that it is able to decompile, and displays the folder's hierarchy in the Asse mbly Explorer.



> ⚠ Since dotPeek processes a selected folder recursively, make sure to act wise selecting one. You wouldn't really want to explore the entire Program Files folder since it would take a really long time to load all assemblies that it contains in dotPeek.

Assemblies from Global Assembly Cache can be opened via File > Open from GAC. One thing to note about the Open from GAC dialog is that you can batch-select assembly items there, and you can also filter out assemblies by entering their CamelHumps - the capitals that different parts of assembly names start with. For example, to find all assemblies with names containing Micro soft.VisualStudio.Modeling in the list of GAC assemblies, you can type mvsm:

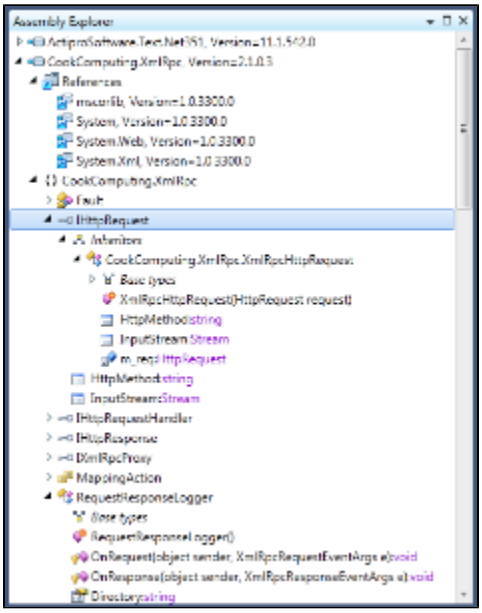


CamelHumps support is an important concept that also spans multiple navigation features of dotPeek that are highlighted below.

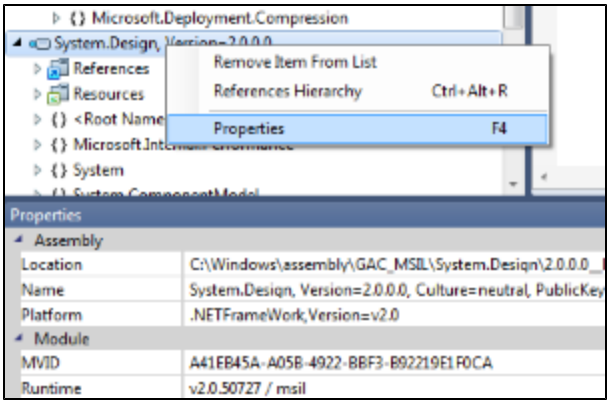
dotPeek provides the Assembly Explorer to traverse the list of opened assemblies. Expanding an assembly node lists namespaces within the assembly that can be further expanded to types and type members, as well as assembly references.

In addition, the Assembly Explorer contains nodes representing base types and inheritors of the current type. This is a way to browse type inheritance trees that developers with Reflector background usually find appealing. However, dotPeek also provides a bunch of ReSharper-inspired navigation features that work not only from the Assembly Explorer but from other tool windows and from the source code view areas as well.

Note that the Assembly Explorer uses the same set of icons that are used in Visual Studio for member identification.



Pressing F4 on an entry in the Assembly Explorer brings up the Properties tool window that shows basic information about a selected assembly or reference, such as assembly location, full name, and platform version.

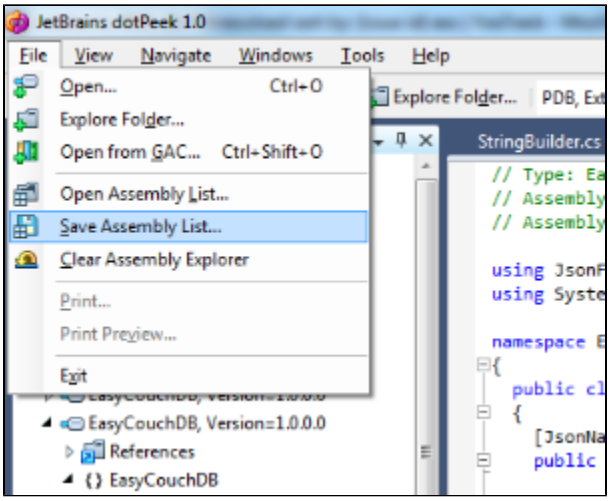


Double-clicking a reference loads the referenced assembly, if immediately available. Double-clicking a type or type member displays decompiled code in the source code view area.

Double-clicking an entry under the Resources folder opens the source representation of the corresponding resource in the source code view area (this, for example, applies to .resources files that are converted back to human-readable .resx form, as well as .png and .bmp image formats), or passes it to an external application.

Assembly Lists and Folders

To apply order to the way you work with multiple assemblies, and also to support switching between different sets of assemblies, dotPeek provides assembly lists. You can now work with different assembly lists depending on your context. You can save and reopen assembly lists, or clear the current list if you no longer need it.



Assembly lists are not limited to .dll and .exe files: they can contain all supported file types (including archives) and folders.

Windows Explorer Integration

If you want to be able to open assemblies in dotPeek by double-clicking them or using the Windows Explorer context menu, you need to enable Windows Explorer integration.

This is accomplished in two different ways depending on the type of dotPeek distribution that you're using.

If you're deploying dotPeek using the installer, Windows Explorer integration is set using the installation wizard.

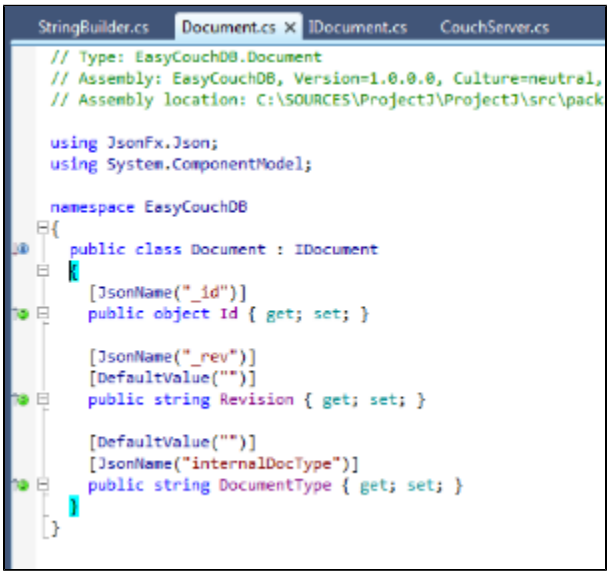
However, if you're using the zip distribution, you need to choose Tools > Options, and select Integrate with Windows Explorer u nder Environment > General to enable Windows Explorer integration.

Viewing the Source Code

Source code that dotPeek decompiles is presented as C#. The source code view area has the look-and-feel of editor tabs in Visual Studio, with line numbers, options for word wrap and outlining, white space marks, and tabs to open different types in. To set outlining options, set of switch off word wrap and tweak other source code view area options to your liking, consider playing with menu items available through View > Outlining and View > Options. You might also want to glance through Tools > Options.

Code syntax is highlighted ReSharper-style, with distinctive colors for properties, types, accessors, and methods.

When you put the caret on a delimiter, be it a brace or, say, parenthesis, it gets highlighted along with its counterpart, bringing focus to the scope of the particular code block you're in:



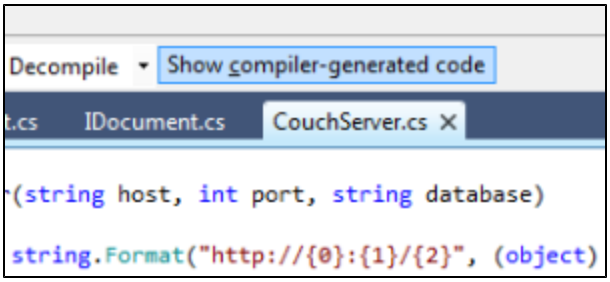
Similar to ReSharper, you can choose to highlight matching delimiters with a color. This and other ReSharper-style source code view area options can be set via Tools > Options > Environment Editor.

Another noticeable ReSharper-like feature gets handy when you want to select a part of decompiled code, and is called Extend/Shrink Selection. Using a dedicated shortcut (by default, Ctrl+Alt+Right) lets you successively select expanding blocks of code, starting from a substring of a symbol, on to a statement, line, code block, and all the way to the entire code file. A pair shortcut (Ctrl+Alt+Left) works the opposite way, successively narrowing a selection. Read more about this functionality as it is implemented in ReSharper.

When you explore decompiled code, you may be willing to learn more about referenced types and method calls without opening their declarations. Two coding assistance features from ReSharper will help you with that:

- Edit > Parameter Information (Ctrl+P) on a method call will display all available signatures of a given method.
- Edit > Show Quick Documentation (Ctrl+Shift+F1) on a usage of a type, method, property, or another type member will display an overview of its documentation comments:

Using the Show compiler-generated code switch on the dotPeek toolbar, you can choose to turn off certain compiler transformations, thus making code structure that dotPeek displays very similar to what the compiler turns it to. This helps see how compiler deals with lambdas, closures, and auto-properties, among other things.



Navigation and Search

The primary idea behind dotPeek is to bring ReSharper experience to browsing external assemblies and make this available to everyone. The main thing that distinguishes dotPeek from other decompilers around is that the majority of ReSharper navigation features are available in dotPeek as well. Let's take a closer look at those:

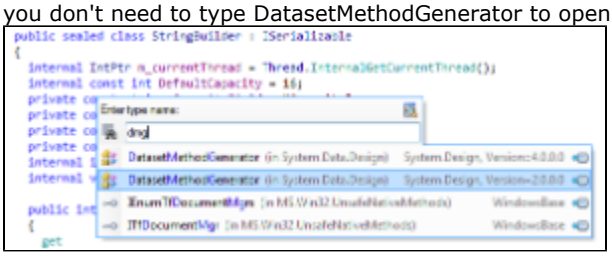
Context-Insensitive Navigation in Decompiled Code

When you're loading an assembly and you don't know what you're looking for and you want to find out how things are organized within the assembly, you start off with the Assembly Explorer and you probably proceed by navigating between symbol declarations in the code view area.

However, you go a different path if you know exactly (or even approximately) which part of the assembly you need to look at - in this case, you can use one of ReSharper's "go to" context-insensitive navigation features:

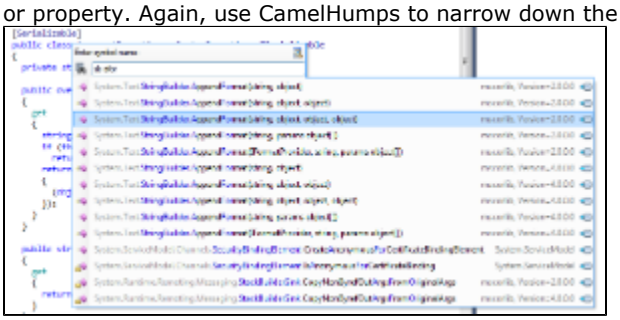
Navigating to a Type

Use Navigate > Go to Type (Ctrl+T) to navigate to a specific class or interface. You type in the name of the type you want to find, and dotPeek searches for a match within all loaded assemblies. Here again, the concept of CamelHumps is applicable - you don't need to type DatasetMethodGenerator to open this class - typing dmg is enough:

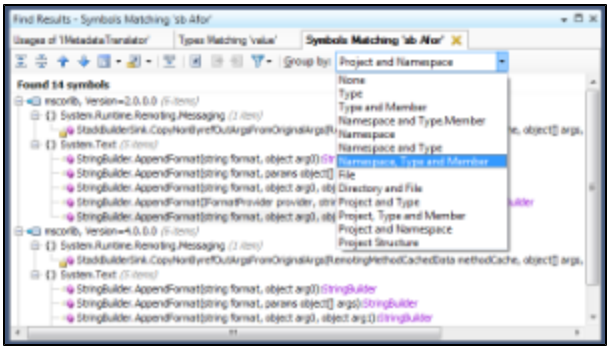


Navigating to a Specific Symbol

Use Navigate > Go to Symbol (Shift+Alt+T) to navigate to a specific symbol declaration, which could be a type, method, field, or property. Again, use CamelHumps to narrow down the list of symbols that dotPeek presents:

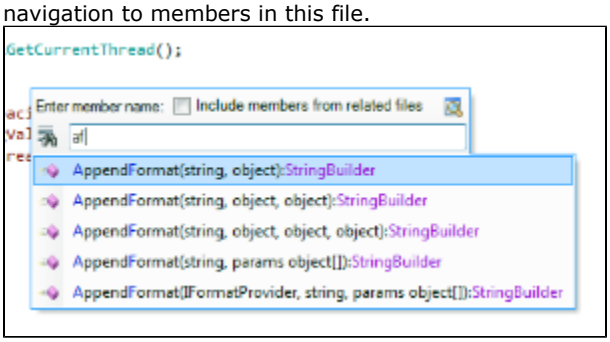


Since the number of symbols is greatly higher than that of types, and there can possibly be multiple symbols with the same name, you may want to spend a little more time investigating the list of results. One way to do that is press the plus sign (Sho w in Find Results) while dotPeek shows symbols in the Go to Symbol drop-down list - this will allow you to flush all found results to the Find Results tool window where you can take your time to investigate the results, group based on different criteria, copy to clipboard or export to a file.

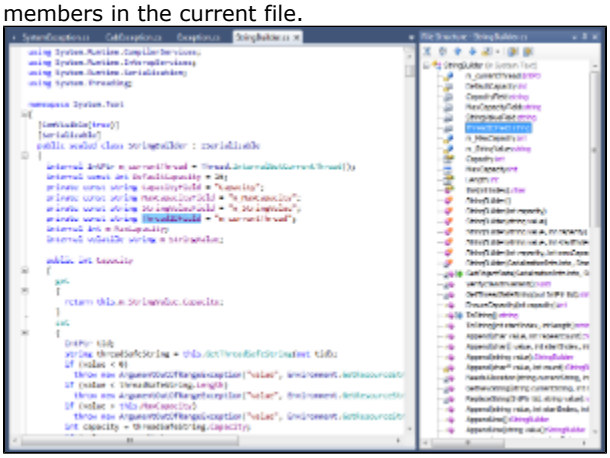


Navigating Between File Members

As soon as you've opened a specific type, use Navigate > Go to File Member (Alt+Backslash) for a quick overview of and navigation to members in this file.



Alternatively, you can open the File Structure tool window (Windows > File Structure, or Ctrl+Alt+F) for a static display of members in the current file.



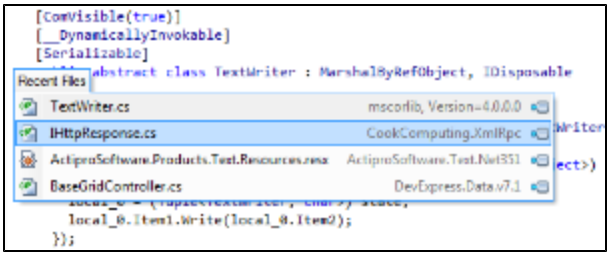
File Structure provides additional file browsing capabilities: for example, if you set Automatically scroll to source in File Structure toolbar, every time you select a member in File Structure, the code view area scrolls to the declaration of this member.
The Track caret in editor option works the opposite way: as you move the caret within the code view area, the corresponding member is highlighted in File Structure.
You can learn more about File Structure options in ReSharper web help.

Navigating to a Code File or Assembly

dotPeek also provides Go to File (Navigate > Go to File or Ctrl+Shift+T) to quickly open files and folders, and assemblies. There are two use cases for this feature:

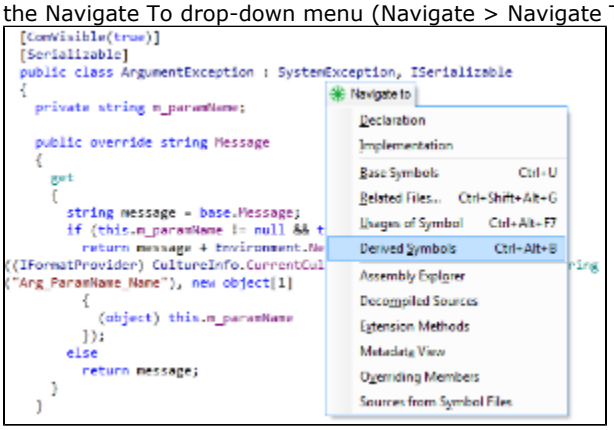
- You can navigate to a specific loaded assembly that gets highlighted in the Assembly Explorer.
- You can use it as an extended tab switcher that not only works with currently opened code files but also takes into account any tabs that you've opened during the current dotPeek session.

Speaking of previously opened files, if you've recently closed a code file but you need to have it open again, there's an easy way to have it back: just choose Navigate > Recent Files (Ctrl+,) and in the resulting drop-down list, pick the file you're looking for:



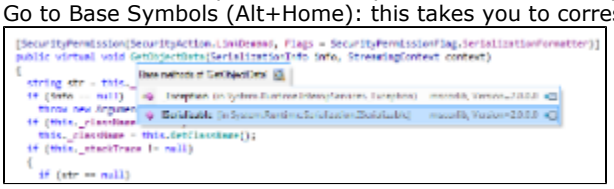
Context-Sensitive Navigation Between Symbols in Decompiled Code

dotPeek provides the same level of insight in context-sensitive navigation between decompiled code symbols as ReSharper does for source code. You navigate to symbol declarations, implementations, derived and base symbols, and any other applicable destinations just like you would in Visual Studio with ReSharper enabled.
Specifically, when you've landed the caret on a symbol, you can always get an overview of all possible navigation targets using the Navigate To drop-down menu (Navigate > Navigate To or Alt+`):

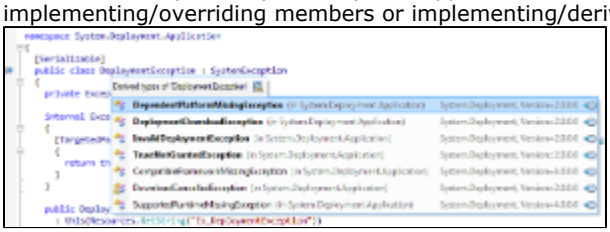


The majority of navigation destinations presented in the Navigate To drop-down menu are also available directly through the top-level Navigate menu.
The following context-sensitive navigation commands can be available depending on context:

- Go to Declaration (F12): this takes you from a usage of any symbol to its declaration. Should the symbol be dependent on another assembly, the assembly will be loaded silently, if available.
- Go to Base Symbols (Alt+Home): this takes you to corresponding symbols upwards the inheritance hierarchy:



- Go to Derived Symbols (Alt+End): the opposite of base symbols, this command lets you go to any of implementing/overriding members or implementing/derived types:



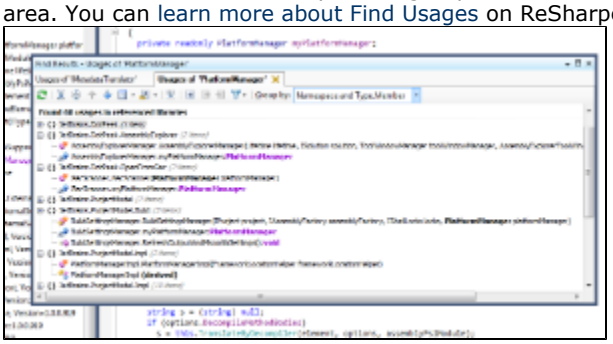
- Go to Implementation (Ctrl+F12): allowing navigating to end implementations of types and type members, bypassing intermediate inheritance steps in the inheritance chain, such as abstract classes.
- Go to Extension Methods (only available through Navigate To): shows all extension methods for a certain type.
- Go to Sources from Symbol Files (only available through Navigate To): download symbol information from a source server and recreate source code. (See below for more information.)
- Go to Assembly Explorer (only available through Navigate To): navigates from a type or type member in source code view to the corresponding node in the Assembly Explorer. Interestingly, this command is duplicated by another shortcut, Shift+Alt+L, that, when applied in ReSharper, locates the currently open file in the Solution Explorer.

For more details on these and other special-purpose commands available through the Navigate To drop-down menu, see ReSharper web help.

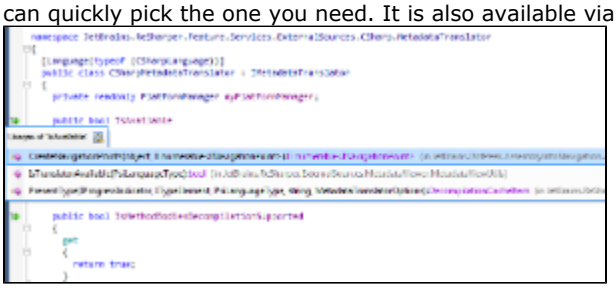
Searching in Decompiled Code

dotPeek offers the same capabilities of searching for items in decompiled code as ReSharper offers in source code inside Visual Studio. Here's the list of features serving to find all references to a certain symbol:

- Navigate > Find Usages (Shift+F12): This finds all usages of a symbol (method, property, local variable etc.) from its any occurrence, be it a declaration or one of its usages. You can invoke this command from the code view area, from the Assembly Explorer, or any other tool window. If more than a single usage is found, all usages are fetched to the Fin d Results tool window where you can group them in different ways, navigate between them, and open in the code view area. You can learn more about Find Usages on ReSharper web site.



- Navigate > Usages of Symbol (Shift+Alt+F12): This is a modification on Find Usages that shows a pop-up with all found usages instead of flushing them to Find Results. This is handy when you have a limited set of usages from which you can quickly pick the one you need. It is also available via the Navigate To drop-down menu.

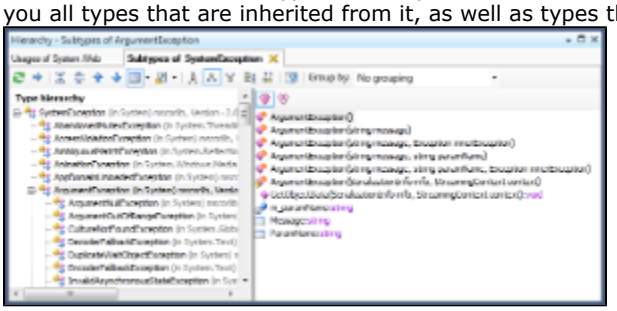


- Navigate > Find Usages Advanced (Ctrl+Shift+Alt+F12): This is a zoomed-in version of Find Usages that allows you to f ine-tune search criteria by limiting the scope of search and other characteristics.

Other Features

Type Hierarchy

We've already covered Go to Derived Symbols and Go to Base Symbols above but these two features are useful when you want to go to an inheritor or a base symbol right away. What if you're looking to plainly get an overview of a certain inheritance chain? That's where the Type Hierarchy view comes handy: press Ctrl+Alt+H on a usage of any type, and dotPeek will show you all types that are inherited from it, as well as types that it inherits itself - as a tree view, in a separate tool window.

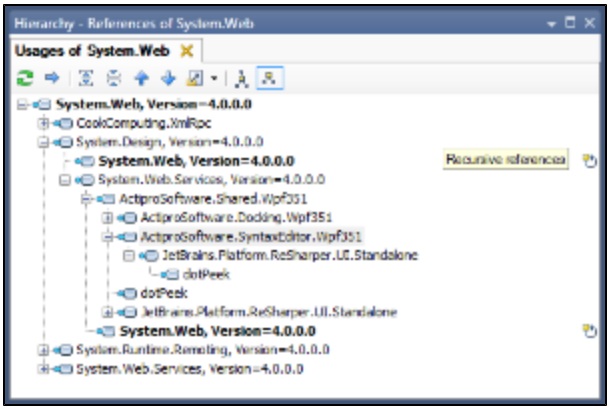


Goodness doesn't end here, though: you can select nodes in the tool window, rebase hierarchies on them; show or hide previews of type members; and switch between several hierarchy views: for example, you can opt to only show subtypes or supertypes of a given type.

Reference Hierarchy

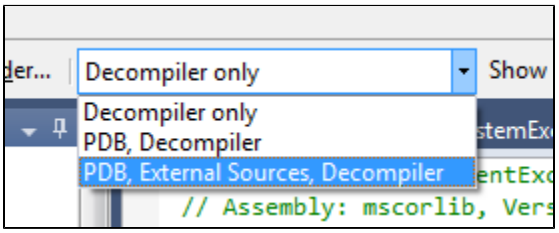
A derivative of Type Hierarchy introduced above, the Reference Hierarchy tool window shows which references the current assembly has, allowing you to track down all assembly dependencies, and additionally showing recursive dependencies with a glyph to the right of a reference entry.

If you click Referencing projects in the tool window’s toolbar, you can see which of the assemblies in your current assembly list reference the current selected assembly.



Downloading Source Code from Source Servers

Decompiled code is better than nothing (especially if it's decompiled with dotPeek) but sometimes you want to explore an assembly exactly the way it had been originally written, and be able to read comments its developers have made. Sometimes dotPeek can help with that: it is able to get symbol information and recreate source code from Microsoft Reference Source Center and SymbolSource.org. You can try calling Navigate > Navigate To > Sources from Symbol Files on a type or member and see if symbol information for this particular assembly is available. You can even go as far as make downloading symbol files the default action for Go to declaration - to do that go to Tools > Options > Decompiler and select Allow downloading from remote locations.



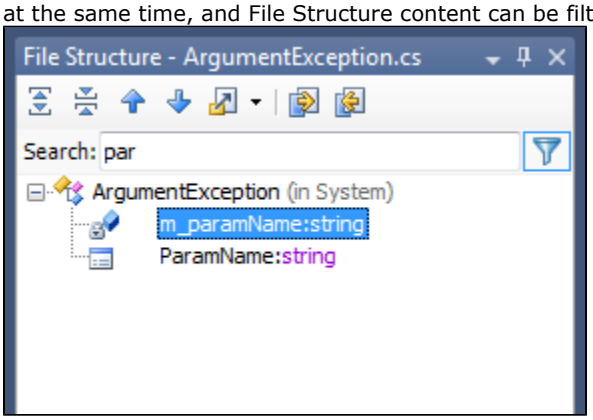
Use the navigation mode drop-down in the menu bar to choose whether you only want dotPeek to decompile assemblies, or try find source code if possible.

Keyboard-Driven Navigation Between Tool Windows

In the best traditions of JetBrains tools, you rarely need to use a mouse when working with dotPeek: switching between the As sembly Explorer and the code view area; opening File Structure, Find Results, Type Hierarchy and other tool windows doesn't require mouse clicks: every tool window is assigned a shortcut of its own (see the Windows menu for shortcut hints), and getting back to the code view area is as easy as pressing Esc.

Customizable, Rich Tool Windows

When it comes to customizing your workspace within dotPeek, you do need the mouse but otherwise, it's plain easy and familiar. Tool windows behave the same way they do in Visual Studio: they can be left floating or docked in multiple positions. Find Results and Type Hierarchy support multiple tabs, allowing you to have several sets of search results or hierarchies open at the same time, and File Structure content can be filtered to only display results that match a search string:



What's Next

As dotPeek 1.0 has been successfully released in May 2012, we're now looking to come up with a bug fix release in coming months, and then proceed to implementing your feature requests.

Thanks for reading up until this point! We hope you've had much fun playing with dotPeek.

In case you were just looking around trying to find out if the tool was worth it, now it's probably the right time to download the latest and greatest dotPeek and see with your own eyes.