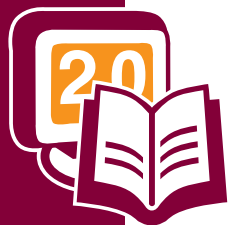


JetBRAINS

R# *ReSharper*



JetBrains ReSharper 2.0 Overview

Introduction

ReSharper is undoubtedly the most intelligent add-in to Visual Studio.NET 2003 and 2005. It greatly increases the productivity of C# and ASP.NET developers, and provides quick and easy access to all of its advanced features right from Visual Studio.

ReSharper 2.0 not only assists with actual coding and project navigation, but also helps with application testing.



All ReSharper features are available from the keyboard.

This guide will walk you through the installation stage and introduce you to key ReSharper features that include:

- Error Highlighting and Quick Fixes
- Advanced Code Editing
- Numerous Refactorings
- Code Generation
- Code Templates
- Navigation and Search
- Unit Testing
- ASP.NET Editing
- NAnt and MS Build Scripts Editing
- Open API

Installation

You will find the ReSharper installer at <http://www.jetbrains.com/resharper/download/>. ReSharper 2.0 supports Visual Studio .NET 2003 and Visual Studio .NET 2005. There are separate installation packages for both supported versions of Visual Studio.

● Important notice for VS .NET 2003 users

Before installing ReSharper, please make a copy of your default keyboard scheme (this is required since VS.NET does not allow one to modify the default scheme). To do this open VS.NET and go to **Tools > Options > Environment > Keyboard** and click the **Save As** button. In the opened dialog enter some name for the new scheme (e.g. *ReSharper*).

To install ReSharper, exit all Visual Studio instances and start the ReSharper installation wizard.

After installing ReSharper, just open Visual Studio .NET and you can immediately start to develop with pleasure.

System Requirements

Before installing ReSharper, please make sure that the following requirements have been met:

Processor

Minimum: Pentium III 850 Mhz

Memory

Minimum: 512 Mb

Disk space

100 Mb

Operating System

Microsoft Windows 2000/XP

Software installed

Microsoft Visual Studio 2003 or 2005

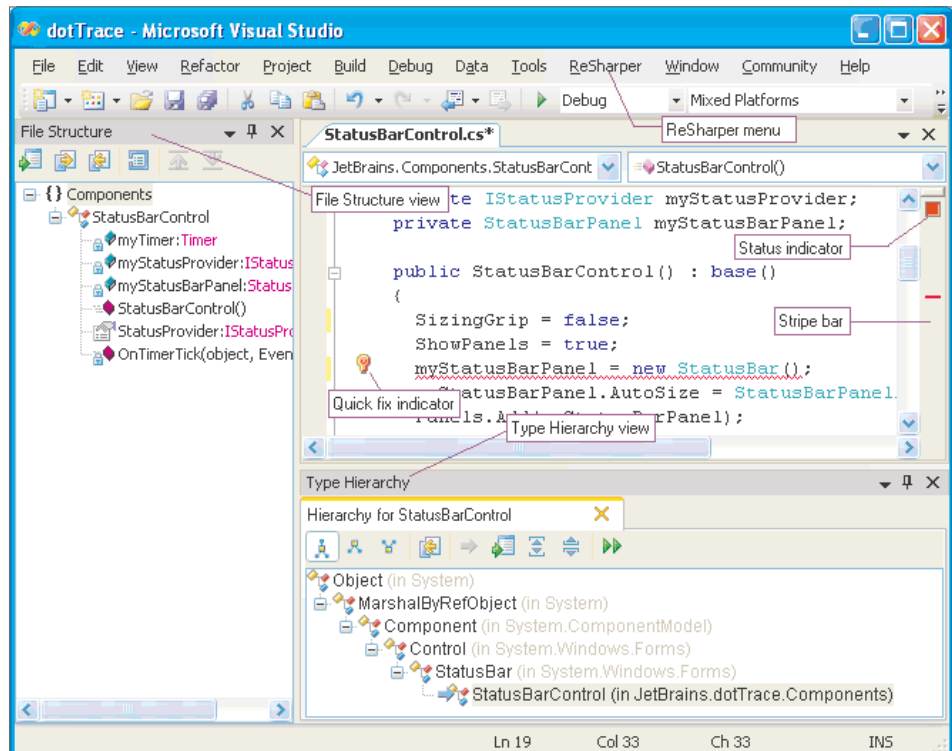
Getting Started

Once installed, you will notice that ReSharper slightly extends the Visual Studio.NET user interface. For starters, it adds its own menu – *ReSharper* – to the main menu bar. From this menu you can access most ReSharper features.

ReSharper also extends the standard set of Visual Studio.Net views. For example, it can show an opened file structure and a selected type hierarchy in dedicated tool windows.

Of course, more significant changes introduced by ReSharper become apparent when you actually start coding. ReSharper adds advanced syntax and error highlighting; shows error and warning messages and provides quick navigation to them via a “stripe bar” along the right-hand edge of the editor. ReSharper also suggests quick fixes for instant correction of discovered problems, and a whole lot more as will be shown momentarily.

★ ReSharper analyzes what features you have yet to use, and on startup, displays them as **Tips of the Day** so you learn these additional functionalities as you get more comfortable with ReSharper.



ReSharper features in Visual Studio

Key Features

Error Highlighting and Quick-Fixes

- **On-the-fly error checking**

ReSharper is able to quickly detect and highlight errors in code without having to first compile. ReSharper automatically analyzes and highlights your code while you work. If you roll the mouse pointer over an error, its description is displayed in a tooltip.

```
public NamespaceTreeNode GetChild(String name)
{
    return myNameToChildNodeMap[name];
}
Cannot convert expression type 'object' to return type 'JetBrains.SnapShotTree.NamespaceTreeNode'
```

Error highlighting with tooltip error message

Besides errors and warnings detectable by compile, ReSharper can also prompt you about additional warnings that may represent real coding errors. For example, ReSharper informs you about redundant casts or incorrect format strings.

```
Console.WriteLine("Name: {0}; Address: {1}", name);
Non-existing argument in format string
```

Warning for the incorrect format string


You can instantly see whether the opened file contains any errors or warnings by looking at the stripe bar that is added by ReSharper to the right-hand side of the editor. The status indicator at the top changes its color if the file contains errors or warnings. Pointing to it will show you the total number of errors or warnings in the current file.

Each warning or error is represented by an individual stripe on the bar. Clicking on a stripe navigates the user directly to the line of code which contains the error or causes the warning.



You can navigate between errors and warnings by pressing **F12** and **Shift + F12**. The corresponding error message will be displayed in the status bar.

- **Quick-fixes**

ReSharper suggests quick-fixes for most errors, helping you solve problems instantly. Quick-fixes are represented by the red light bulb  that appears automatically to the left of the code line containing the error when you position the caret on the error. To see the list of available quick-fixes for a given error, you can either click the light bulb icon or press **Alt + Enter**, and then simply select the desired quick-fix from the list.

```
public NamespaceTreeNode GetChild(String name)
{
    return myNameToChildNodeMap[name];
}
Cast to 'NamespaceTreeNode'
Safely cast to 'NamespaceTreeNode'
Change type of 'GetChild' to 'object'
ldrenNodes()
```

List of quick-fixes for a highlighted error

- **Auto-insertion of missing using directives**

Whenever you have a type name in your code that cannot be resolved because you have forgotten to issue the corresponding using directive in the source file, a small popup appears suggesting one or more types to import. If you press **Alt + Enter**, the appropriate using directive will be inserted automatically, and you will not lose your current caret position.



A screenshot of a code editor showing a popup window. The popup contains the text "System.Collections.ArrayList?(Alt+Enter)". Below the popup, the code "public ArrayList" is visible with a red squiggly line under "ArrayList".

Auto-insertion of missing using directive



ReSharper can extend the default syntax highlighting support of Visual Studio with additional highlighting for fields, local variables, types, and more. To switch on ReSharper's highlighting, click **ReSharper>Options** menu, and then click Highlighting.

Advanced Code Editing

- **Code completion**

ReSharper extends Visual Studio's native completion (IntelliSense) with more advanced capabilities. For example, it narrows down the list of suggestions according to your typing, adds parenthesis when completing methods names, and completes variable and field names depending on their types, etc.

Smart code completion, invoked with **Ctrl + Shift + Space**, filters the list of methods and variables to match the expected type of an expression. For example, when you use Smart Completion for arguments of some method call, the list of suggested values is narrowed down only to the required parameter type.

Type Name code completion is invoked with **Ctrl + Alt + Space**. Unlike basic completion (which only completes types accessible at the current location), this one completes the names of types available in the current project, and automatically adds the appropriate using directives when necessary.

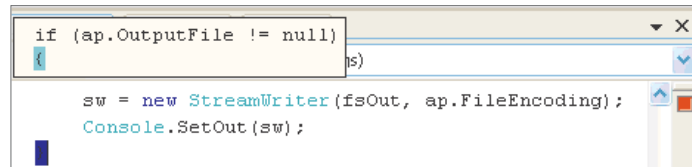
Coding assistance

ReSharper provides you with many small coding assistance features that altogether may greatly improve your productivity. For example, ReSharper automatically inserts pair brackets, parentheses, and quotation marks.



If you have a strong habit of typing closing parenthesis and quotation marks yourself, don't worry – ReSharper will recognize that the closing bracket, quotation mark, etc., already exists and will leave only one character printed.

Another example of a useful feature is the highlighting of pair parentheses and brackets. If you position the caret at the enclosing parenthesis, the opening will be highlighted. If the opening parenthesis is not currently displayed in the editor, the related code will be shown in a small popup window.




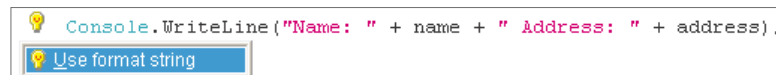
Matching parentheses

You will certainly discover more coding assistance features while actually coding, including the auto-indentation of parentheses, highlighting current row, and others.

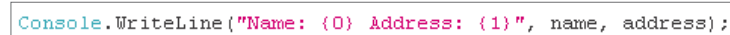
- **Context actions (new!)**

Context actions are intended for small code transformations that you may want to perform in a certain context. When you position the caret at any place in the code editor, ReSharper displays what context actions can be applied for the code under the caret.

Similar to the quick-fixes feature, available context actions are shown to the left of the code line where the caret is positioned and indicated by the yellow light bulb: . To perform an action, you can either click the bulb or press **Alt + Enter**, and then select the desired action.



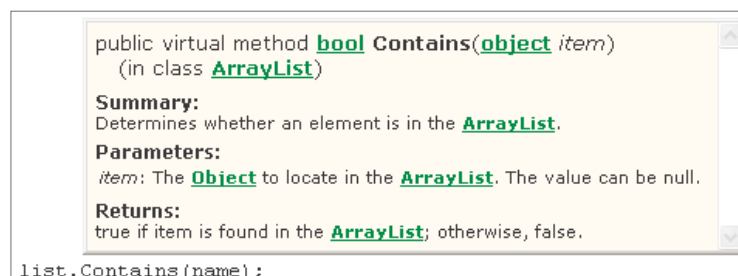
Context action for concatenated string



Generated format string

- **Quick documentation lookup (new!)**

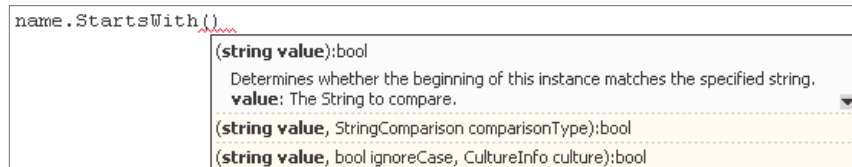
To see the documentation for a certain class, method, or other symbol right in the editor, position the caret on the symbol name and press **Ctrl + Q**. The documentation will be shown in a popup window.



Documentation shown in popup window

- **Parameter info**

This part of Visual Studio's IntelliSense is also extended in ReSharper. When you call a method, it shows you all possible method signatures and parameters with relevant documentation (the tooltip appears automatically while you type, or you can display it by pressing **Ctrl + P**). While you are adding new arguments, ReSharper grays out all signatures that are incompatible.

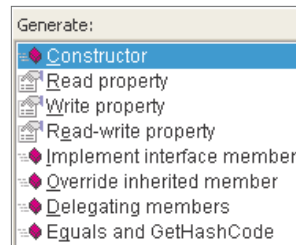


Method parameters tooltip

Code Generation

- **Generate common code constructs**

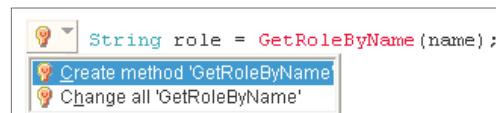
ReSharper helps developers quickly generate constructors, read and write properties, and other common code constructs. Just press **Alt + Ins** (**ReSharper>Code>Generate**), and then select the necessary item from the list.



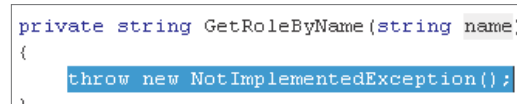
List of code generation options

- **Create from usage**

You can start using a method, field, local variable or even a class before it was declared. ReSharper will suggest a quick fix for generating a corresponding symbol based on the usage and smartly adjust the declaration according to the usage context. For example, if you create a method from usage, ReSharper will not only create a method, but also detect its return type as well as the types of its parameters.



“Create method from usage” quick-fix



Generated method

Code Templates

- **Live Templates**

ReSharper’s analogous feature to Visual Studio 2005’s code snippets is the so-called “Live Templates”. As with code snippets, you need to type a template abbreviation, press Tab to expand the template, and then use **Enter** or **Tab** to navigate through template variables. But Live Templates are far more intelligent than code snippets. ReSharper can guess from the context what variables, fields, and types can be used in corresponding templates. So, in most cases you don’t need to type anything yourself, but only choose from a list of suggested values.

```

ArrayList stringList = new ArrayList(strings);
foreach (string s in stringList)
{
    charList
    chars
    stringList
    strings
}
    
```

Live Template example

- **“Surround with” templates**

Surround With functionality is quite similar to Live Templates and is used to quickly enclose an expression, a single statement, or an arbitrary block of code with if/else, try/catch or other code constructs. ReSharper intelligently reformats the code, adjusts the selection, and/or repositions the caret as specified by the template.

To surround some code, select the code block, and then press **Ctrl + Alt + J** or click the **ReSharper>Code>Surround with** menu.

```

FileStream outputStream = new FileStream(filePath, FileMode.Create);
outputStream.Write(bytes, 0, bytes.Length);
outputStream.Close();
    
```

Code block before using “Surround with”

```

try
{
    FileStream outputStream = new FileStream(filePath, FileMode.Create);
    outputStream.Write(bytes, 0, bytes.Length);
    outputStream.Close();
}
catch (Exception e)
{
    Console.WriteLine(e);
}
    
```

Code block after using “Surround with”

- **File templates (new!)**

ReSharper lets you add new files to your project with predefined code fragments already generated in them. For example, you can create a new file with a class declaration, an interface, struct declaration, and so on. To create a file from a template, click the **ReSharper>New** from template menu, and then select the necessary file type.



You can easily create your own Live Templates as well as “Surround with” or file templates. No matter what kind of template you create, you will be provided with a convenient user interface.

- **Templates Configuration**

You can modify any ReSharper templates as well as create new ones. To configure existing templates or write your own, click the **ReSharper>Options** menu, and then in the **ReSharper Options** dialog box, select the necessary template group under the **Templates** node.



Once defined, any template can be easily shared among team members.

Refactoring

ReSharper provides a larger set of automated code refactorings than Visual Studio 2005. In total, 21 different refactorings are provided that allow you to rename, move, and safely delete symbols; introduce and inline fields, variables, and parameters; and more.

To invoke a specific refactoring, use the corresponding menu item in the **ReSharper>Refactor** menu. In addition, most refactorings have their own keyboard shortcuts.

★ To see what refactorings are available at a current caret position, press **Ctrl + Shift + R**.

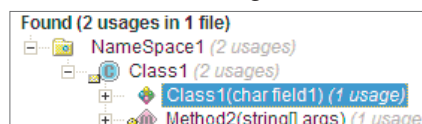
Navigation and Search

- Search for a type or file by name**
 Press **Ctrl + N** to navigate to a class, or **Ctrl + Shift + N** to navigate to a file. A small search window will pop up. As you start typing in the text field, a lookup list appears which contains names matching the entered substring. Use the wildcard character * (asterisk) in the search string to represent zero or more characters, or + (plus) to represent one or more characters.

★ Instead of typing the full name of a class, you can type only capitalized symbols. For example, instead of *AbstractTreeBuilder* you can simply type *ATB*.

- Go to declaration from reference**
 To navigate to the declaration of the symbol, position the caret at any symbol usage and press **Ctrl + B** or press **Ctrl** and click the symbol. The caret will position itself at the declaration of the corresponding type, method, field, or local variable in the relevant source file (which opens automatically, if necessary). For library symbols, the corresponding entry will be displayed in Visual Studio's Object Browser.
- Find usages**
 The Find Usages feature helps you locate usages of any symbol (e.g., type, method, field, etc.) in your code. Simply position the caret on the symbol for which you want to find usages and press **Alt + F7**. Search results are displayed in the **Find Results** window organized in a hierarchy. From this window you can directly navigate to any usage with either keyboard or mouse. Also when this window is open, you can navigate between usages by pressing **Ctrl + Alt + Up/Down** (even from the editor).

o specify the search scope and the type of usages to be found, use the Advanced Find Usages feature invoked by pressing **Shift + Alt + F7**.



Found usages – click to navigate to source

When you know there are not many usages of a symbol, and you just need to quickly navigate to it, you may want to use the **Ctrl + Alt + F7** combination to open usages in a popup, instead of an entire tool window.

All these search commands are grouped under the **ReSharper>Search** menu.



You can highlight all the usages of a symbol within the current file. Just position the caret on any symbol usage and press **Ctrl + Shift + F7**.

- **Navigate through type and method hierarchies**

You can easily see that a given method overrides, implements, or hides another method by the presence of special icons that appear at the method declaration on the left gutter of the editor window. Click this icon to navigate up the methods hierarchy.

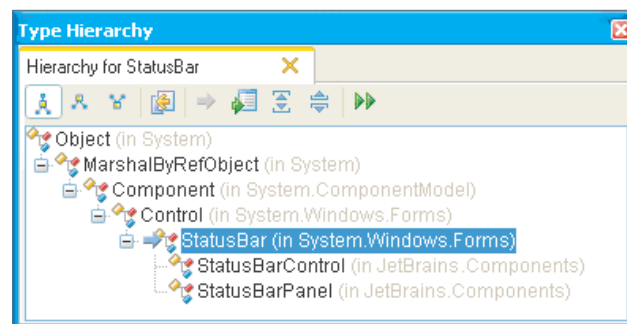
```
public override void OnUsage (String errorInfo)
    Overrides method from class 'MyNameSpace.Samples.ArgumentParser'(click to navigate)
```

Overriding method

You can also navigate through inheritance hierarchies using keyboard shortcuts. Place the caret on the desired type or method in the editor and press **Ctrl + U** to navigate up the hierarchy, or press **Ctrl + Alt + B** to navigate down.

- **Type Hierarchy view (new!)**

With ReSharper, you can view the inheritance hierarchy of a certain type in a dedicated window. The window shows both base types and inheritors of the selected type and allows you to navigate to any of them with a single click. To open the window for a type, position the caret at the type name, and click **Ctrl + Alt + H** or click the **ReSharper>View->Type Hierarchy** menu.

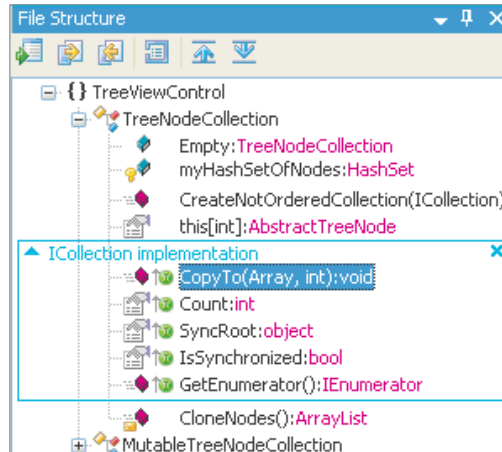


Type hierarchy window

- **File Structure view (new!)**

With the **File Structure** window, you can see what methods, fields, classes, and regions your current file contains, as well as navigate directly to their declarations.

The window also displays regions defined in the current file and allows you to arrange declarations within classes and regions according to your needs. Just drag the node to the new location.



File Structure window with “ICollection implementation” region



The File Structure window is fully synchronized with the editor. All changes made to a file are immediately reflected in the File Structure and vice-versa.

- **Quick navigation through opened file**

To quickly navigate to a particular method or field of a given type, use the File Structure Popup (**Ctrl + F12**) that shows the members of the current class. Simply click the required symbol and its declaration will be opened in the editor.

The File Structure Popup also provides “quick search” ability: just start typing the name you are looking for, and the selection will navigate to the first matching occurrence in the list.

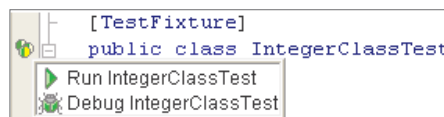
- **Stack Trace Explorer**

When you receive an external stack trace (for example, from a bug report), you can copy-paste it into the Stack Trace Explorer to navigate to an exception’s origin. The lines within the stack trace will be represented as hyperlinks. To open the Stack Trace Explorer, click **ReSharper | Stack Trace Explorer** or press **Ctrl + Shift + E**.

Unit Testing

- **Running and debugging tests**

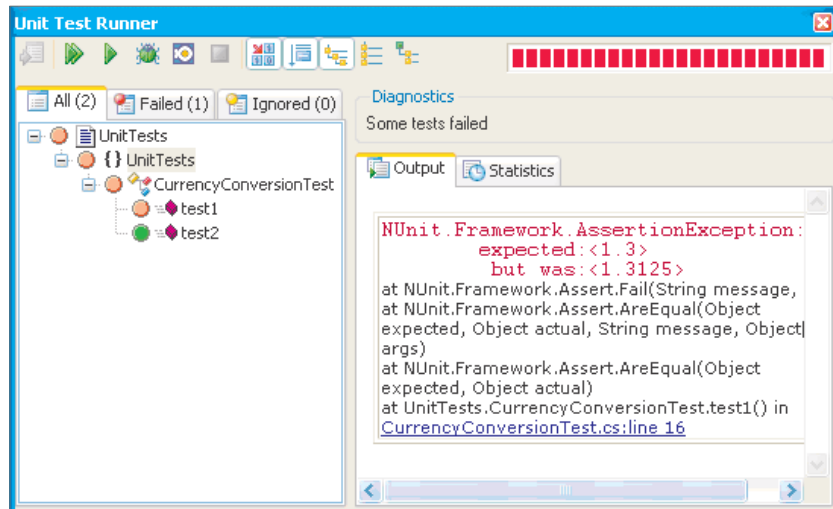
ReSharper automatically detects whether your project contains unit tests of supported test frameworks (JUnit and NUnit). Next to declarations of test classes and single tests, ReSharper adds special icons on the left gutter of the editor window. Click these icons to run or debug tests.



Unit test icons

You can also run tests for a whole solution or project from the Visual Studio’s Solution Explorer. Just right-click the project or solution and select run or debug tests.

When you start running unit tests, ReSharper opens the **Unit Test Runner** window that is intended to help you with analyzing test results. Using this window, you can also run/re-run any tests.



Unit Test Runner window

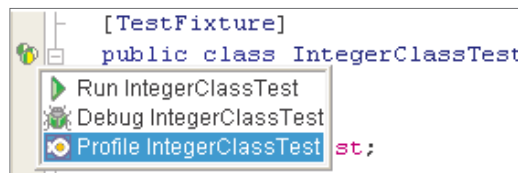


When using the Unit Test Runner, you can navigate from a failed test's output to the lines that originated the exception, all with a single click.

- **Profiling unit tests with dotTrace Profiler™**

You can also quickly profile the performance of unit tests from Visual Studio via JetBrains dotTrace Profiler™, a powerful and easy-to-use .NET profiling tool.

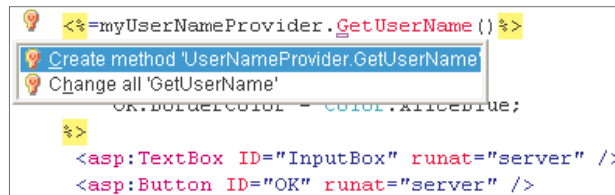
To profile tests, you will need to install dotTrace Profiler (see details at <http://www.jetbrains.com/profiler>). You will then be able to start profiling directly from the editor using the sidebar marks that ReSharper adds for test classes and individual tests.



Option for profiling test

ASP.NET Editing

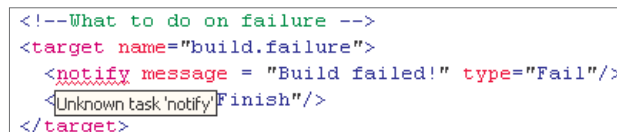
ReSharper provides most of its C# editing features for ASP.NET files, including error highlighting, quick-fixes and context actions, coding assistance, navigation, refactoring, and more. Now you can edit and refactor C# code within ASP.NET without hassle.



Code generation in ASP.NET

NAnt and MSBuild Scripts Editing

Advanced editing capabilities, formerly available only for C#, are extended to NAnt and MSBuild scripts. Features like code completion, error highlighting and quick-fixes, rename refactoring, quick navigation to declarations, and more, are now at your disposal. ReSharper recognizes a build script file automatically as soon as you open it.



Error highlighting for build script

Open API

ReSharper's functionality can be significantly extended by means of its open API. In fact, most of ReSharper's features are implemented using this same API that is available to ReSharper plugins. For example, one can add new quick-fixes, tool windows, and even refactorings.