ReSharper

Build Your Own Extension

ReSharper Extensibility

Matt Ellis

JetBRAINS

Why?

JetBRAINS

How?

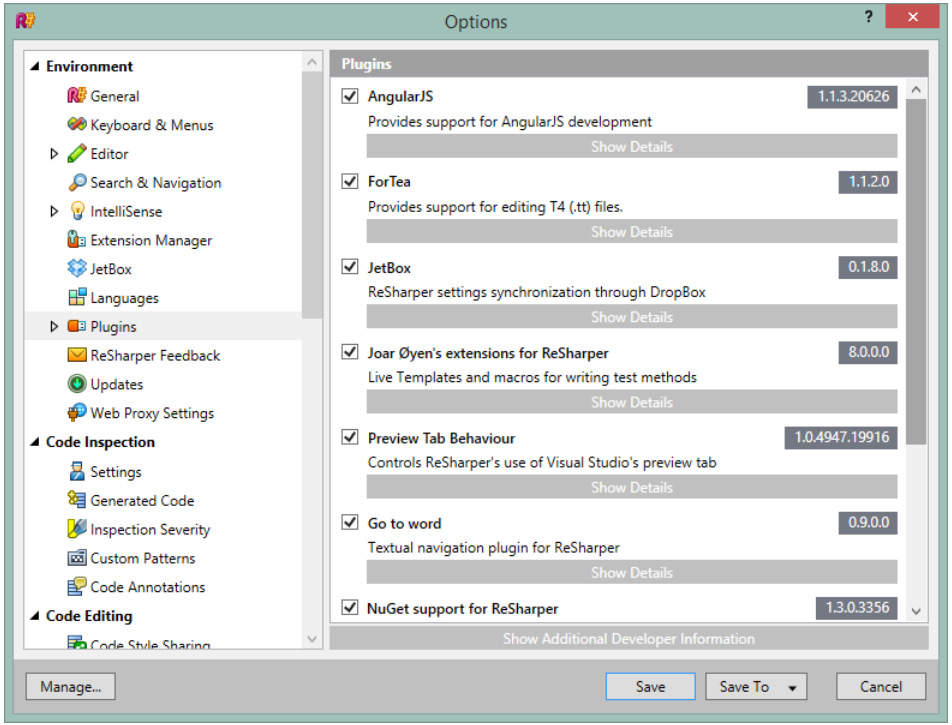
Plugins – full access to ReSharper's API

Live Templates – Generate code snippets

Structural Search and Replace – Declarative. As you type. Alt+Enter replace

External Annotations – hints for ReSharper's code analysis

JetBRAINS



Plugins

Compiled assembly (.net 3.5)

No plugin API

“OpenAPI” – same API as ReSharper devs

Very extensible

Sensitive to product evolution
(i.e. breaking changes across versions)

API is binary compatible between maintenance releases

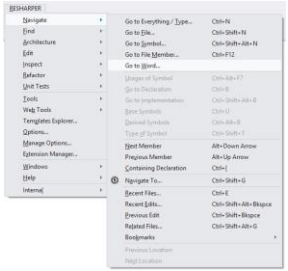
Usually (but not always) source compatible between minor releases

Rule of thumb: Support last two major versions (i.e. 7.1 and 8.0, or 7.1 and 8.1)

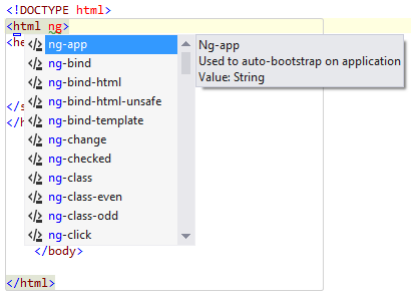
Use Anti-Corruption Layer pattern for compatibility

JetBRAINS

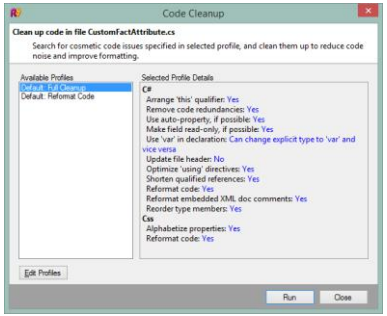
Actions and menus



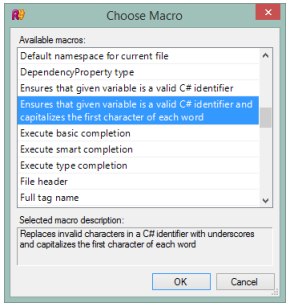
Code completion



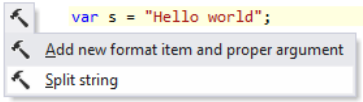
Code cleanup module



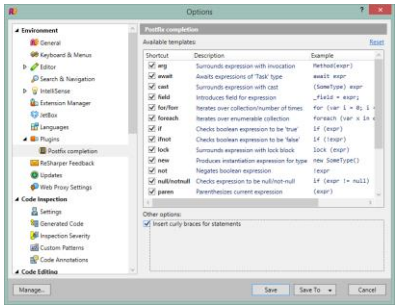
Live template macro



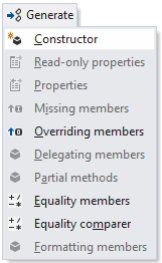
Context actions



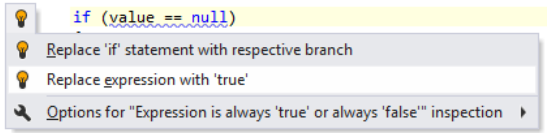
Options page



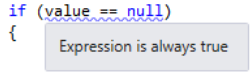
Generator



Quick fixes



Analysers



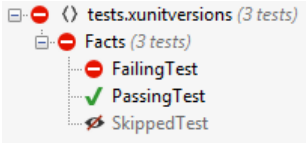
What can plugins do?

JetBRAINS

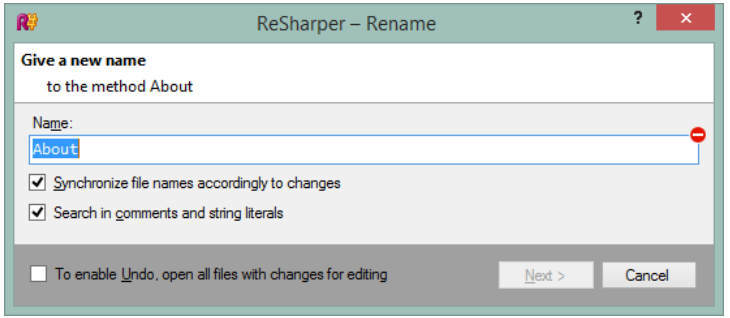
Language support

```
<#
var timeout = TimeSpan.FromMilliseconds(2400);
#
    fromMilliseconds
    milliseconds
    span
    timeSpan
<#+
public string FormatTimeSpan(TimeSpan timeSpan)
{
    return timeSpan.ToString(@"d\.hh\.mm\.FFFF");
}
#>
```

Unit testing



Refactoring



Reference provider

```
[Theory]
[PropertyData("TheoryDataEnumerator")]
public void
{
    Console.WriteLine("DataFromProperty({0})", value);
}
```

DerivedReturnTypeTheoryDataEnumerator IList<object[]>
TheoryDataEnumerator IEnumerable<object[]>

What else can plugins do?

JetBRAINS

Read/write ASTs

Full fidelity abstract syntax tree

Strongly typed navigation

Modifying the tree updates the document text

References to semantic model

FILE FORMATS

C#, VB

HTML, CSS, Javascript

ASPX (ashx, ascx, asmx, asax, skin)

Razor (C# and VB)

Xml, DTD

XAML

WinRT appxmanifest

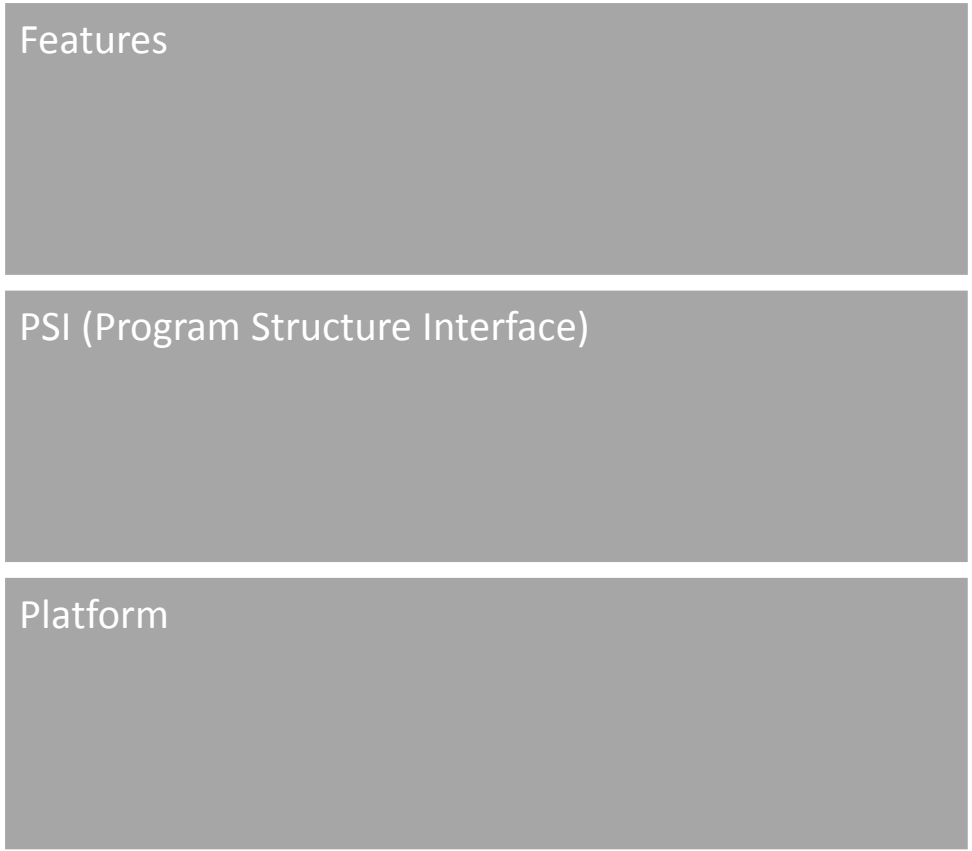
Build scripts (nant, msbuild)

Web.config, App.config

Xml documentation

C++ and TypeScript (coming soon)

JetBRAINS



Architecture

Simplified view – 3 layers

JetBRAINS

Platform

Core .net platform. Shared between ReSharper, dotCover, dotTrace, dotPeek

Shell provides app services – change propagation, threading, dispatcher, read/write locks, logging, extensions

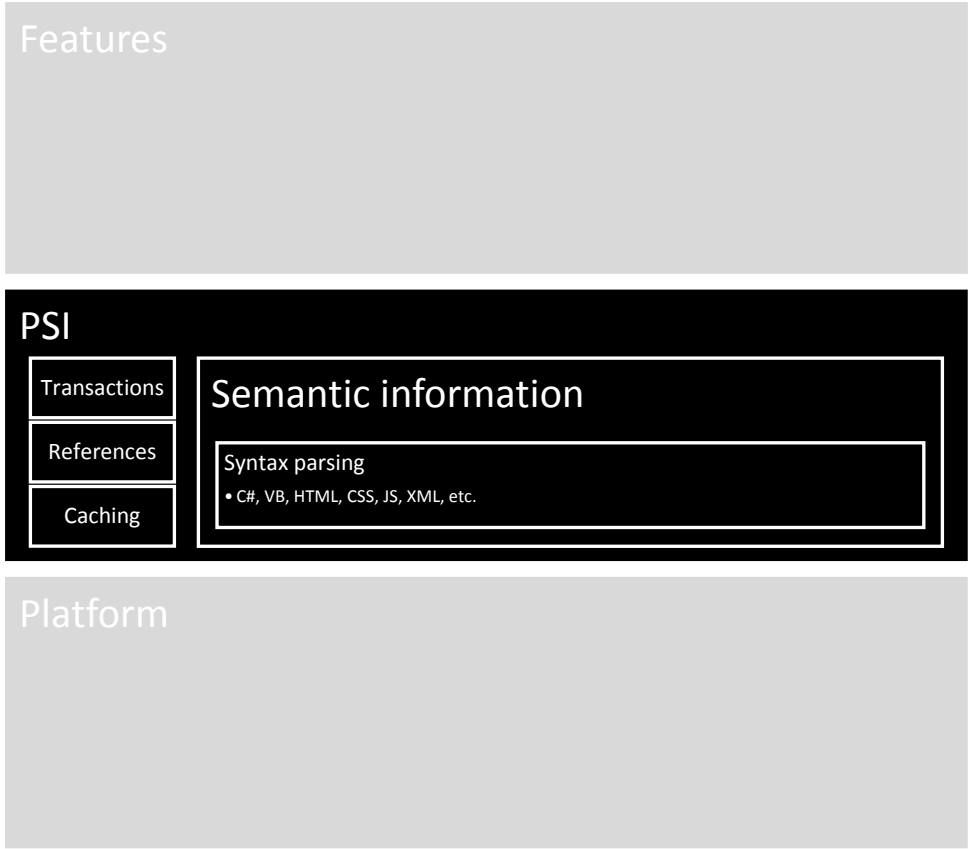
Component Model – ReSharper's IOC container

Project Model works with projects, files and references

Document Model and **Text Control** manipulate files

Util – useful base functionality, e.g. xml reading/writing, tuples, registry, file system paths, rich text, etc.

JetBRAINS



Program Structure Interface (PSI)

File parsers

Builds Abstract Syntax Trees (AST)

Semantic representation from ASTs (e.g. type resolution)

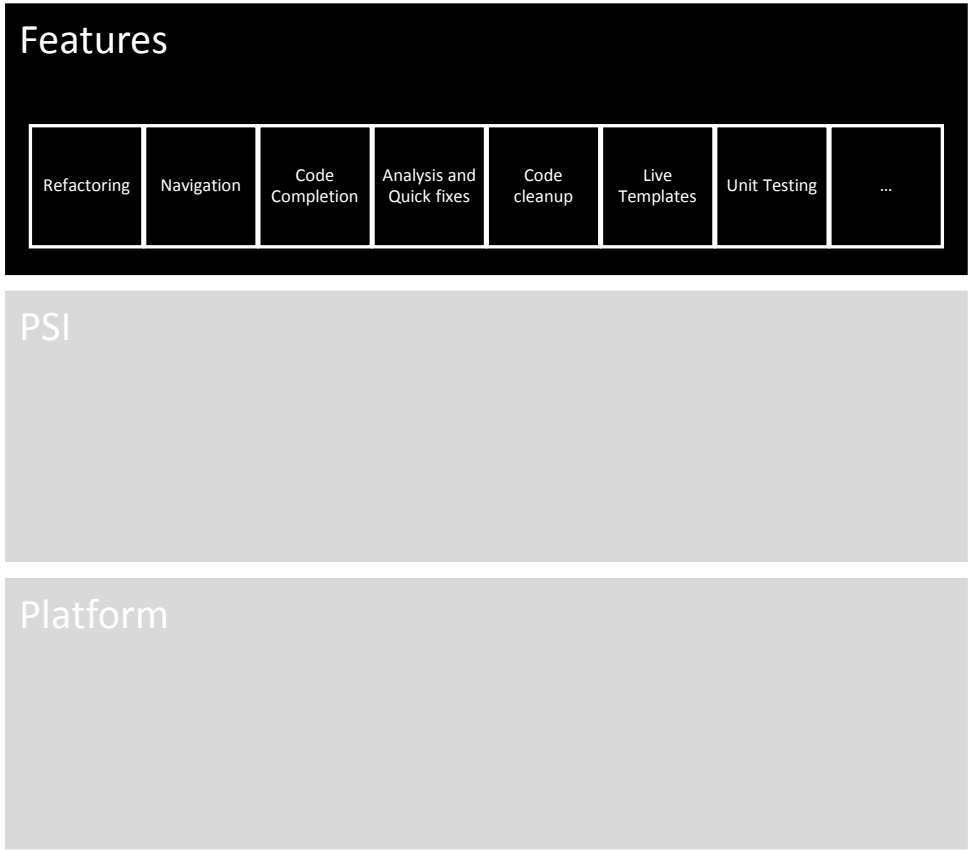
Transactions for modification

Modifying PSI updates document text

References between nodes in ASTs

Caching

JetBRAINS



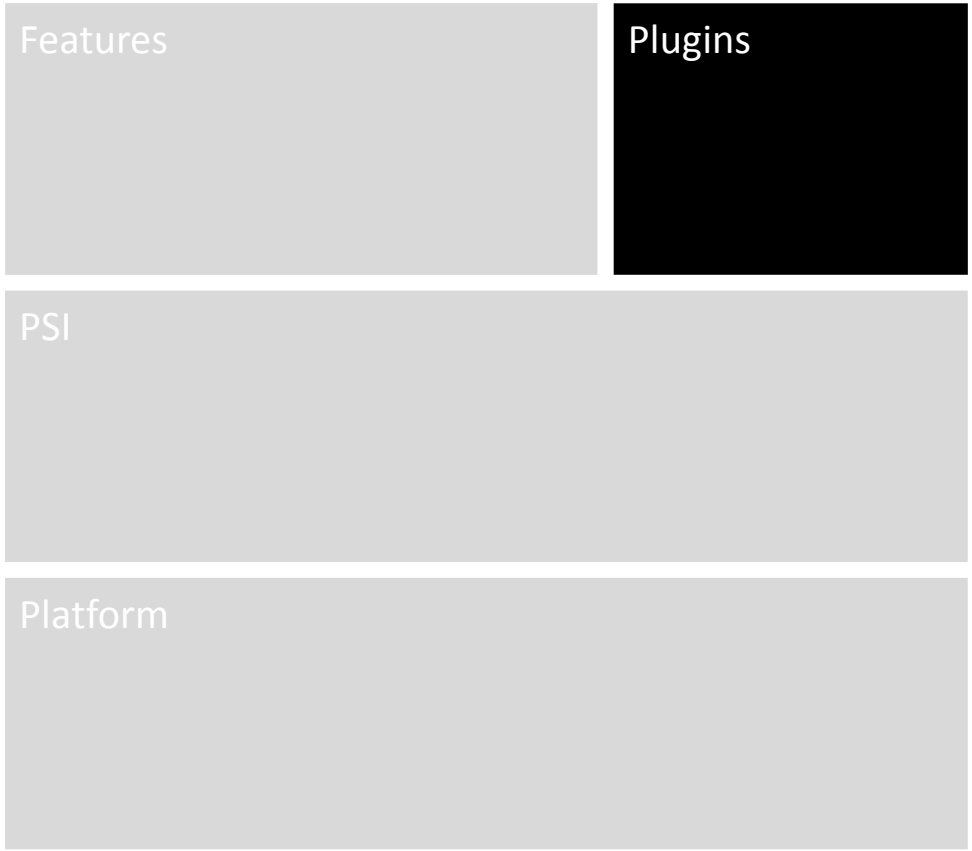
Features

User facing features

Interrogates and manipulates PSI

Provides UI – interacts with documents, dialogs for refactoring parameters, etc.

JetBRAINS



Plugins

Plugins can span whole architecture stack

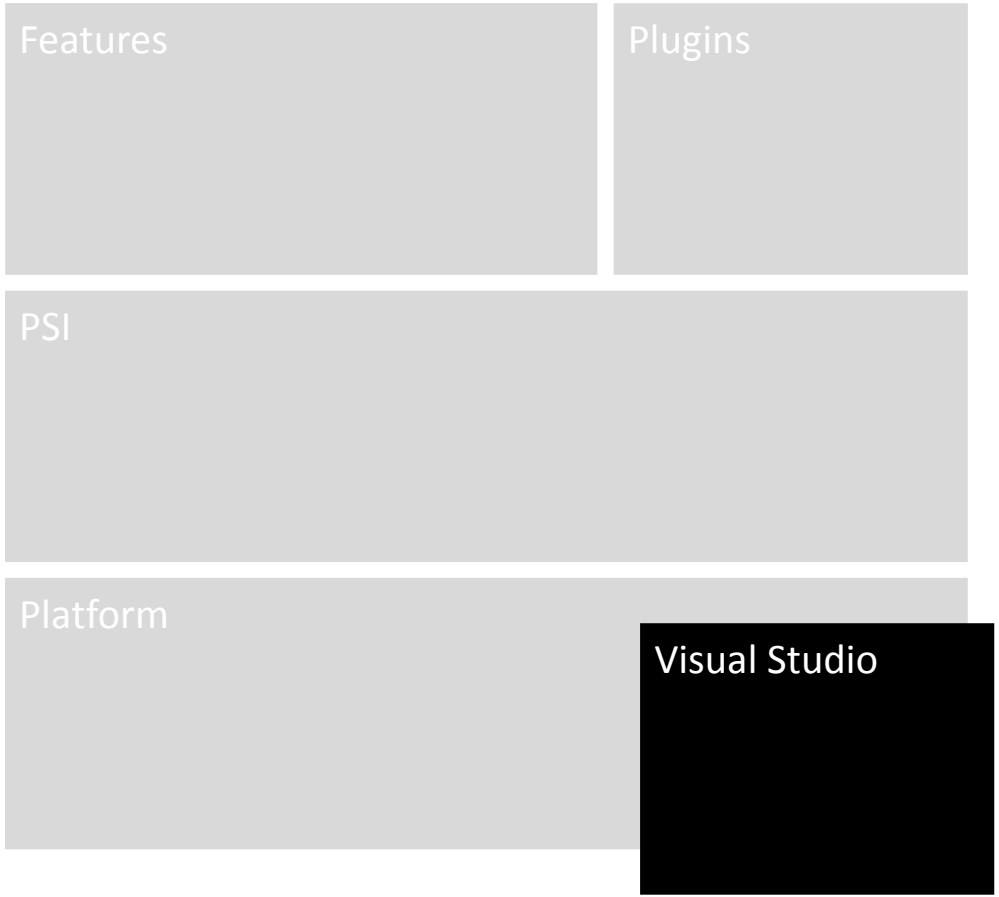
Usually live at Features level

Implement Feature interfaces to integrate with e.g. code completion, unit testing, etc.

Implement PSI interfaces for e.g. caching, language support

Less likely to implement something in Platform

JetBRAINS



Visual Studio

Platform abstracts and interfaces with VS

Plugins are usually VS version agnostic

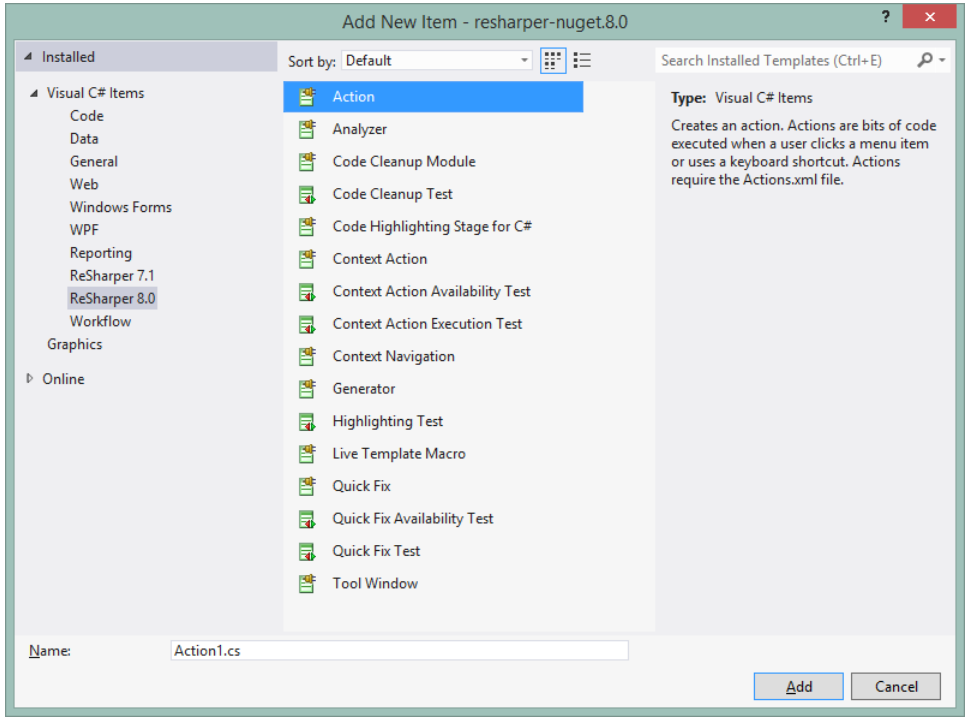
Can consume VS interfaces, but becomes VS version specific

Cannot export interfaces via MEF

VSIX integration
(github.com/JetBrains/resharper-vsix)

- resharper-vsix – install VSIX extensions bundled in a ReSharper extension
- vsix2resharper – load ReSharper extensions bundled in a VSIX

JetBRAINS



Building

SDK
- Reference assemblies, pdb & xml doc
- Build targets & tools
- Visual Studio templates
- Testing
- Samples

Checked build of ReSharper
- Exceptions reported
- More details in exceptions (data dictionary)
- Asserts, consistency checks, tracing

dotPeek – point at SDK bin dir

NuGet – packaging and distribution

JetBRAINS

```
devenv.exe /ReSharper.LogLevel Verbose
           /ReSharper.LogFile "C:\...\rs.log"
           /ReSharper.Plugin "C:\...\plugin.dll"
           /ReSharper.Internal
```

Command line switches

JetBRAINS

Test.cs:

```
public class CSharpHighlightingTest : CSharpHighlightingTestNet4Base
{
  protected override string RelativeTestDataPath
  {
    get { return @"Daemon\CSharp"; }
  }

  [Test] public void testAccessorMissing()
  {
    DoNamedTest2();
  }
}
```

AccessorMissing.cs:

```
class Foo {
  public int Bar { }
  public int this[int i] { }
}
```

AccessorMissing.gold.cs:

```
class Foo {
  public int Bar |{ }|(0)
  public int this[int i] |{ }|(1)
}

-------------------------------------------------------
(0): ReSharper Underlined Error Highlighting: Accessor declaration is missing
(1): ReSharper Underlined Error Highlighting: Accessor declaration is missing
```

Testing

Functional testing, not unit testing

Instantiates a ReSharper environment, with solution and project(s)

Base classes provided, e.g. `CSharpHighlightingTest`, `CSharpContextActionExecuteTestBase`, `BaseTestWithSingleProject`

Processes source file, compares output to “gold” file

Gold file contains annotations for warnings and highlights

JetBRAINS

```
public int Thing()
{
  var value = GetValue();
  if (value != null)
    re
  // ...
  return 42;
}

[NotNull] public string GetValue()
{
  return "Hello";
}
```

Expression is always true

```
public void Foo(Action<int> action1, Action<int> action2)
{
  int c = GetValue();
  RegisterHandler(() => action1(c));
  RegisterHandler(() => action2(c));
}
```

Implicitly captured closure: action1

```
public void RegisterHandler(Action action)
{
  // ...
}
```

```
public void Foo(Action<int> action1, Action<int> action2)
{
  int c = GetValue();
  RegisterHandler(() => action1(c));
  RegisterHandler(() => action2(c));
}


public void RegisterHandler([InstantHandle] Action action)
{
  // ...
}
```

Annotations

"Hints" to ReSharper's analysis

Add binary reference to JetBrains.Annotations.dll

Or paste directly into your own code
ReSharper » Options » Code Annotations

`CanBeNull, NotNull`

`StringFormatMethod`

`InvokerParameterName`

`NotifyPropertyChangedInvocator`

`ContractAnnotation`

`UsedImplicitly, MeansImplicitUse, PublicAPI`

`InstantHandle, Pure`

`AspMvcController, AspMvcModelType, AspMvcView`

Etc.

JetBRAINS

```
<assembly name="mscorlib, Version=4.0.0.0">    <!-- Annotations for mscorlib.dll 4.0 -->

  <!-- Same as String.IndexOf([NotNull] string value, int startIndex, StringComparison type) -->
  <member name="M:System.String.IndexOf(System.String,System.Int32,System.Int32,System.StringComparison)">
    <parameter name="value">
      <attribute ctor="M:JetBrains.Annotations.NotNullAttribute.#ctor" />
    </parameter>
  </member>
</assembly>
```

```
<assembly name="Catel.Core">    <!-- Annotations for any version of Catel.Core.dll -->

  <!-- Same as [NotifyPropertyChangedInvocator("propertyName")] on RaisePropertyChanged(string propertyName) -->
  <member name="M:Catel.Data.ObservableObject.RaisePropertyChanged(System.String,System.Object)">
    <attribute ctor="M:JetBrains.Annotations.NotifyPropertyChangedInvocatorAttribute.#ctor(System.String)">
      <argument>propertyName</argument>
    </attribute>
  </member>
</assembly>
```

External Annotations

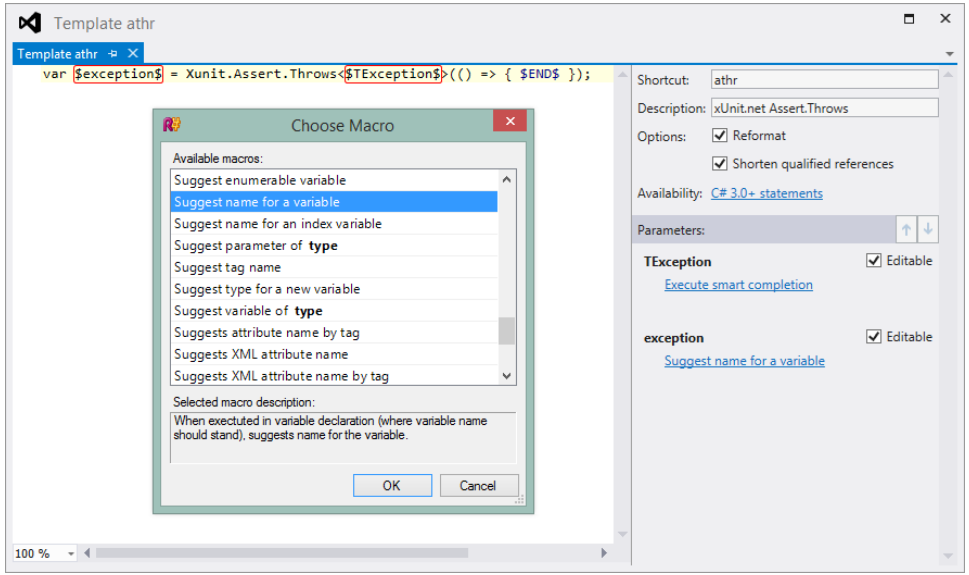
JetBrains.Annotations in an .xml file

Applies annotation attributes to type members

Ship side-by-side with dll or in product dir

Ship as extension in ReSharper 8

JetBRAINS



Live Templates

Expand snippets

Surround code (e.g. try/catch)

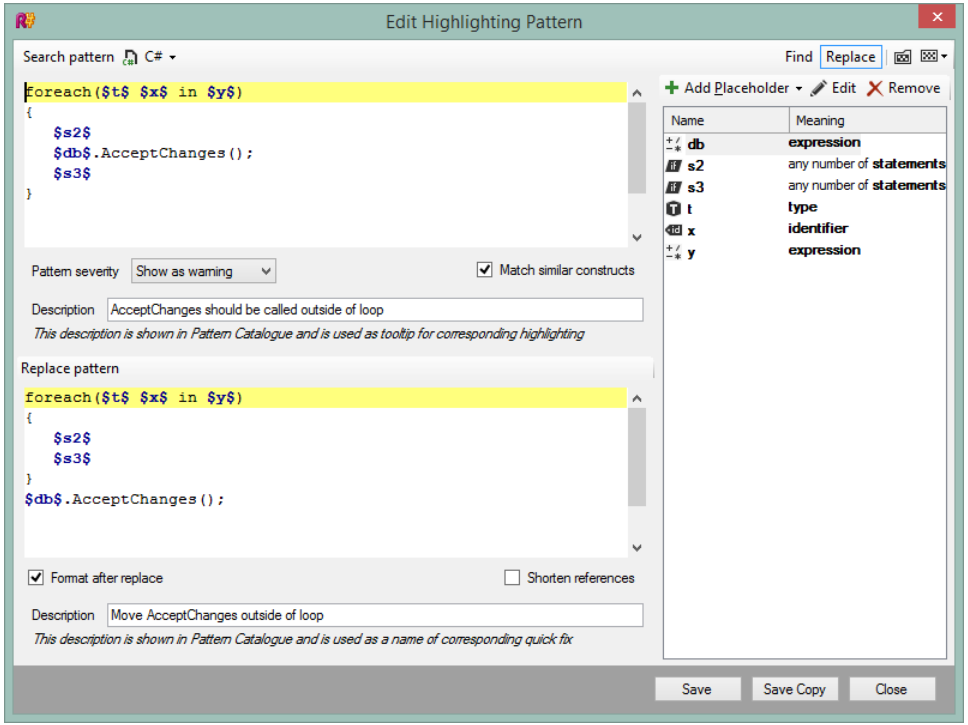
Create files (multiple files in R# 8)

Parameters and macros
  Macros extensible via plugins

Special macros:
  END for final caret position
  $SELECTION$ for surround templates

Set availability scope

JetBRAINS



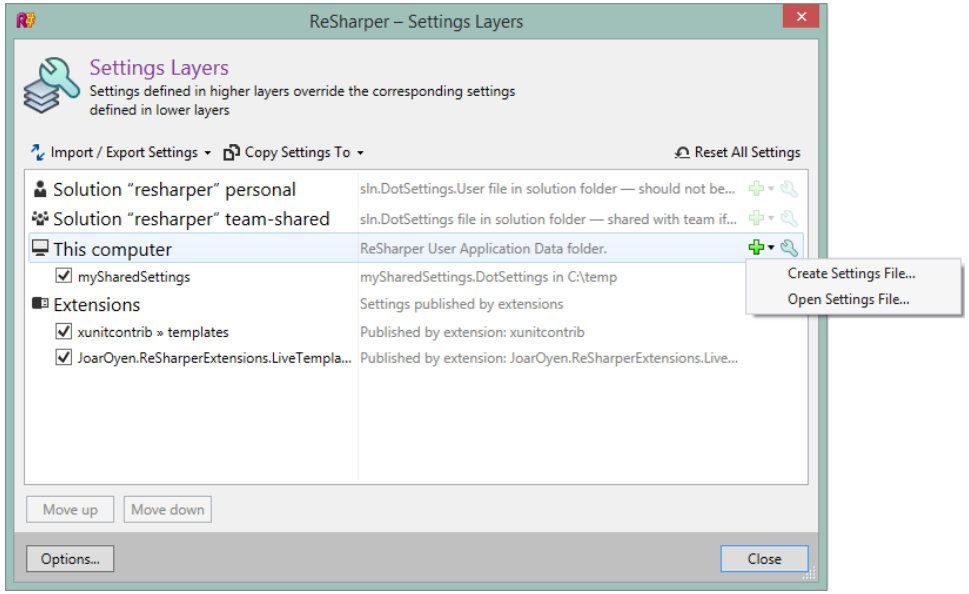
Structural Search and Replace

Declarative code patterns

Find and replace on demand
(via ReSharper » Find menu)

Edit saved patterns via Options dialog

JetBRAINS



Settings

Live Templates + SSR stored in **.dotSettings** files

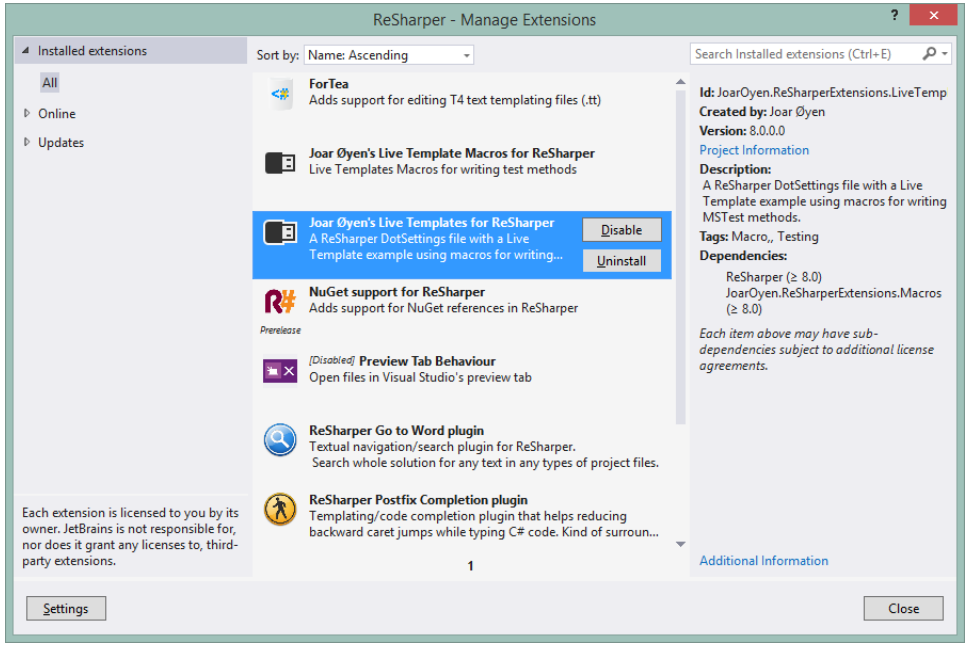
Add settings files in **Manage Options** dialog

Files become "layers" override settings in lower layers

Create file, then use "**Save To**" in options, or the drop down in Templates Explorer

.dotSettings file stores **all** settings (e.g. code style, inspection severity, etc.)

JetBRAINS



Distribution

Extensions distributed as NuGet packages

Browse new, see updates, view installed

Search

Diagnostic information for installed extensions

Dependencies on other extensions

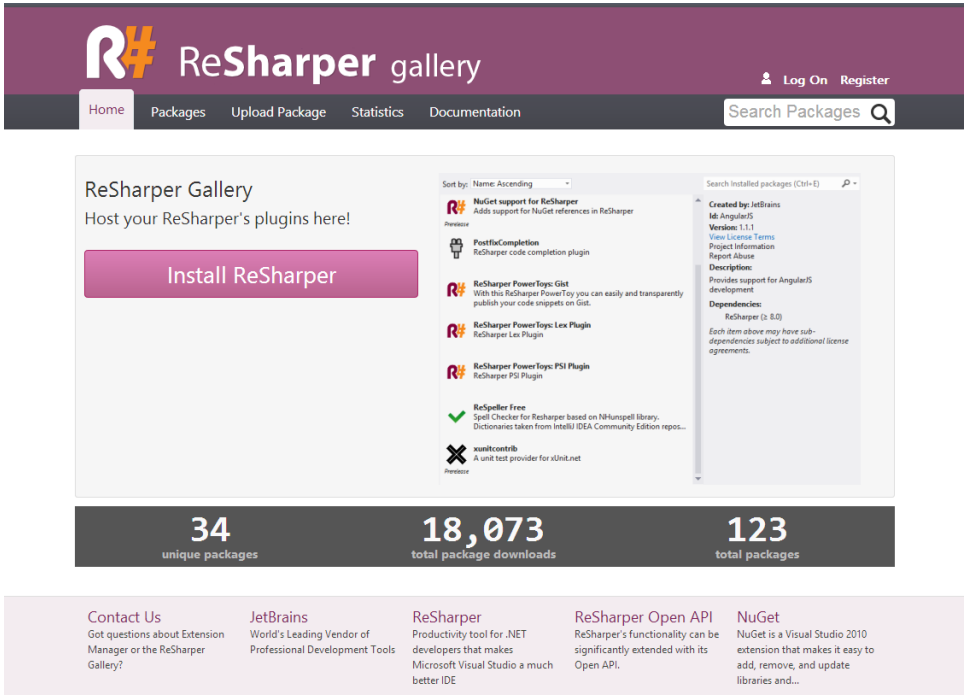
Enable/disable

Pre-release

Settings allows for extra sources (e.g. TeamCity, myget.org, UNC path)

VS2010+

JetBRAINS



Hosting packages

NuGet.exe pack
(ignore warnings about missing lib folders)

NuGet.exe push

Host on default gallery or custom
e.g. TeamCity, myget.org or filesystem

https://resharper-plugins.jetbrains.com/

JetBRAINS


JetBRAINS
Develop with pleasure!