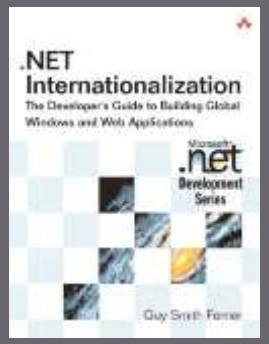
Enforcing Code 'Beauty' With StyleCop

Guy Smith-Ferrier
guy@guysmithferrier.com
Twitter: @GuySmithFerrier
Blog: http://www.guysmithferrier.com

About…

- Author of .NET Internationalization
  - Visit http://www.dotneti18n.com to download the complete source code



- The .NET Developer Network
  - http://www.dotnetdevnet.com
  - Free user group for .NET developers, architects and IT Pros based in Bristol

- DDD South West 3
  - http://www.dddsouthwest.com
  - Saturday 11th June 2011

Agenda

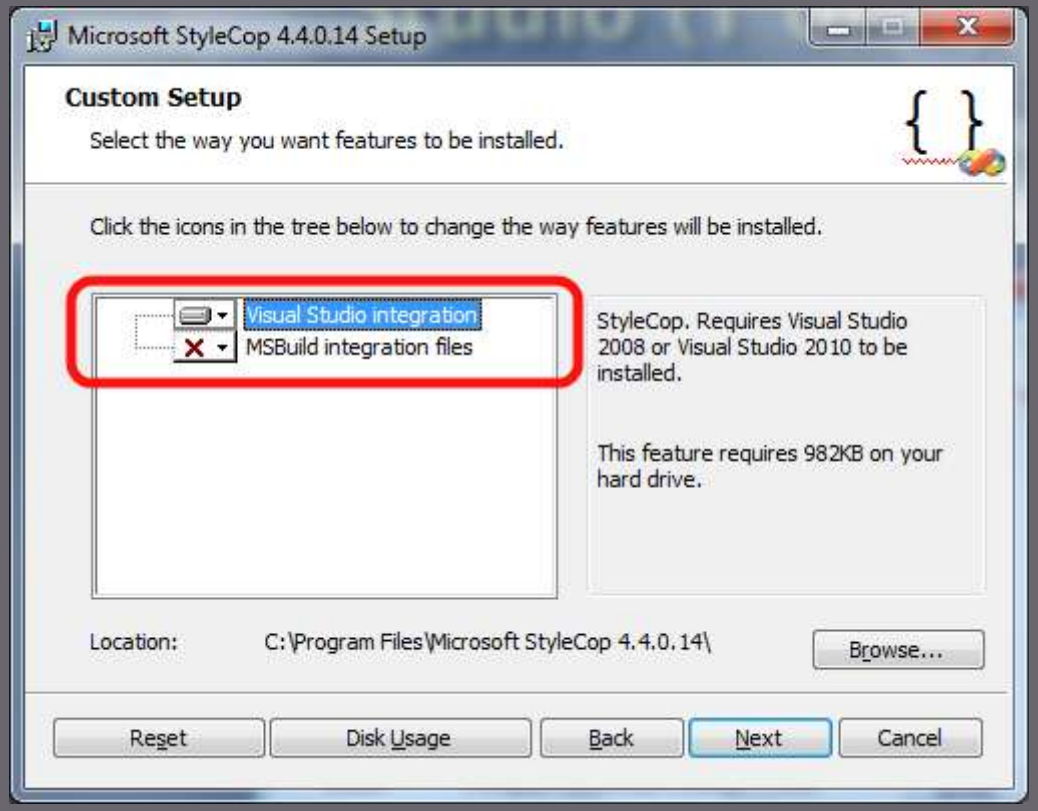
- Introduction To StyleCop
- Integration With Visual Studio
- Customizing StyleCop Project Settings
- Excluding Files From StyleCop Analysis
- 'Favourite' StyleCop Rules
- Creating Custom StyleCop Rules

StyleCop

- An open-source source code analyzer for C#
  - StyleCop is to code what FxCop is to assemblies
    - v4.2 released May 2008
    - v4.3 released August 2008
    - v4.3.3 released January 2010
    - v4.4 released July 2010
  - Includes 151 rules
  - Integrates with Visual Studio 2010 and 2008
- Download
  - http://stylecop.codeplex.com/
- Blog
  - http://blogs.msdn.com/sourceanalysis

Integration With Visual Studio (1 of 2)

- When you install StyleCop you must include msbuild support in the initial setup:-



Integration With Visual Studio (2 of 2)

- To incorporate StyleCop into Visual Studio's build process edit the .csproj and add the following line:-

```
<Import Project="$(ProgramFiles)\MSBuild\Microsoft
\StyleCop\v4.4\Microsoft.StyleCop.targets" />
```

- To treat StyleCop Warnings as errors add the following line to a PropertyGroup:-

```
<StyleCopTreatErrorsAsWarnings>false</StyleCopTreatErrorsAsWarnings>
```

MSBuild StyleCop Task

- To run StyleCop without building the project use the StyleCop task:-

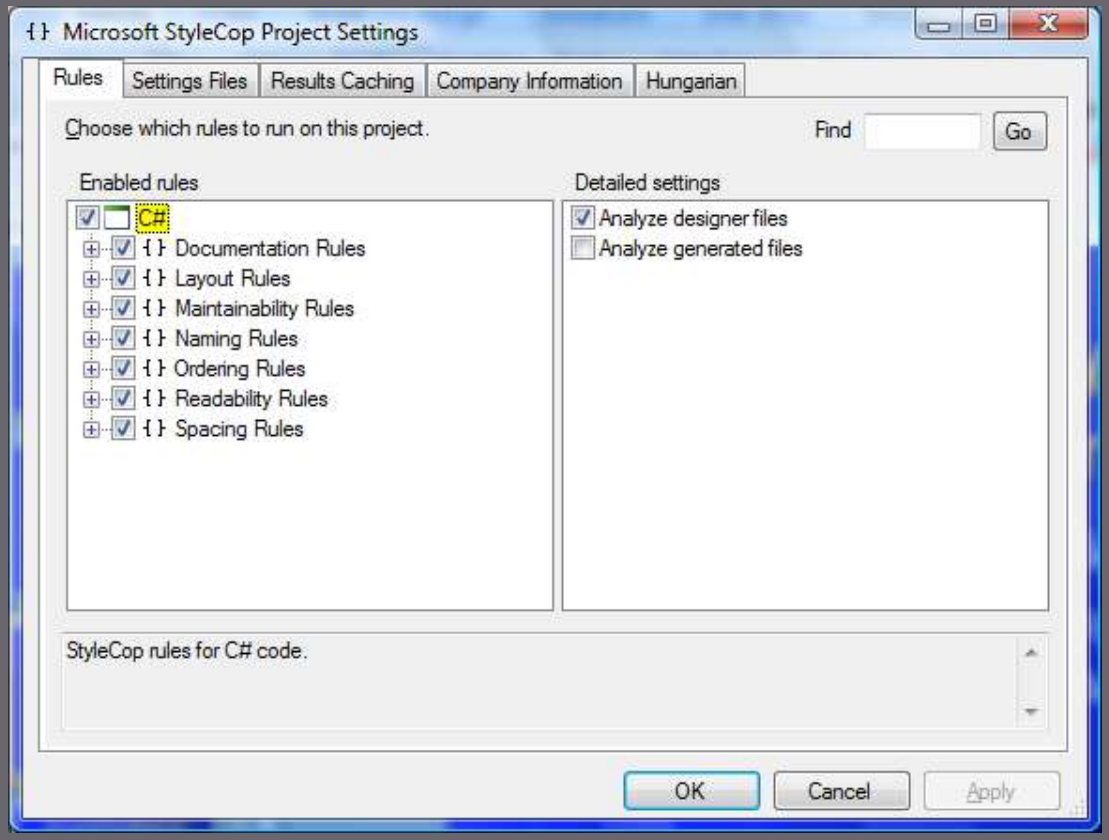
```
msbuild ConsoleApplication1.csproj /t:StyleCop
```

StyleCopCmd And NAnt Support

- StyleCopCmd is a command line interface for StyleCop
- Download StyleCopCmd and NAnt task from:-
  - http://stylecopcmd.wiki.sourceforge.net/
- StyleCopCmd has options to:-
  - Analyze multiple solution files
  - Analyze folders and sub-folders
  - Output results as XML

Customizing StyleCop Project Settings

- You can enable, disable and configure StyleCop project settings using the StyleCop Settings Editor:-



StyleCop And Generated Code

- The StyleCop Settings Editor has two settings to configure analysis of generated code:-
  - Analyze Designer Files (i.e. .Designer.cs)
  - Analyze Generated Files (i.e. .g.cs)
- StyleCop always ignores regions with 'generated code' in the name

```
#region This is generated code (StyleCop will ignore this region)
```

Suppressing Rules In Code

```
[SuppressMessage("Microsoft.StyleCop.CSharp.DocumentationRules",
    "SA1600:ElementsMustBeDocumented")]
internal static class Program
```

```
[SuppressMessage("Microsoft.StyleCop.CSharp.DocumentationRules",
    "SA1600:ElementsMustBeDocumented",
    Justification = "Legacy code will be documented later")]
internal static class Program
```

```
[SuppressMessage("Microsoft.StyleCop.CSharp.DocumentationRules",
    "*",
    Justification = "Legacy code will be documented later")]
internal static class Program
```

<ExcludeFromStyleCop>

- You can exclude any file from StyleCop analysis by changing its file reference in the .csproj file from this:-

```
<Compile Include="Program.cs" />
```

- to this:-

```
<Compile Include="Program.cs">
    <ExcludeFromStyleCop>true</ExcludeFromStyleCop>
</Compile>
```

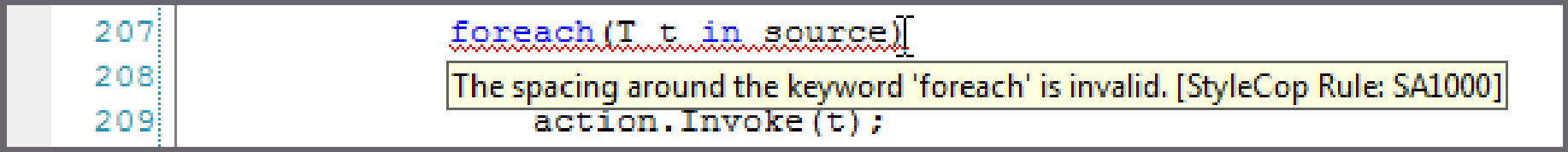
- This affects msbuild only

ExcludeStyleCop.exe

- ExcludeStyleCop is a separate download from:-
  - http://code.msdn.microsoft.com/sourceanalysis
- It is C# source that you build to create ExcludeStyleCop.exe
- ExcludeStyleCop.exe changes **all** of the compile references in **all** project files in the current directory to include <ExcludeFromStyleCop>

StyleCop For ReSharper

- StyleCop for ReSharper is a ReSharper plugin
  - allows Microsoft StyleCop to be run as you type
  - generates real-time syntax highlighting of violations



  - automatically fixes 58 StyleCop issues during ReSharper Code CleanUp
    - ReSharper itself fixes another 52 StyleCop issues
  - Open source, written by Howard van Rooijen (@HowardvRooijen)
- Download from:-
  - http://stylecop.forresharper.org

Style Ninja

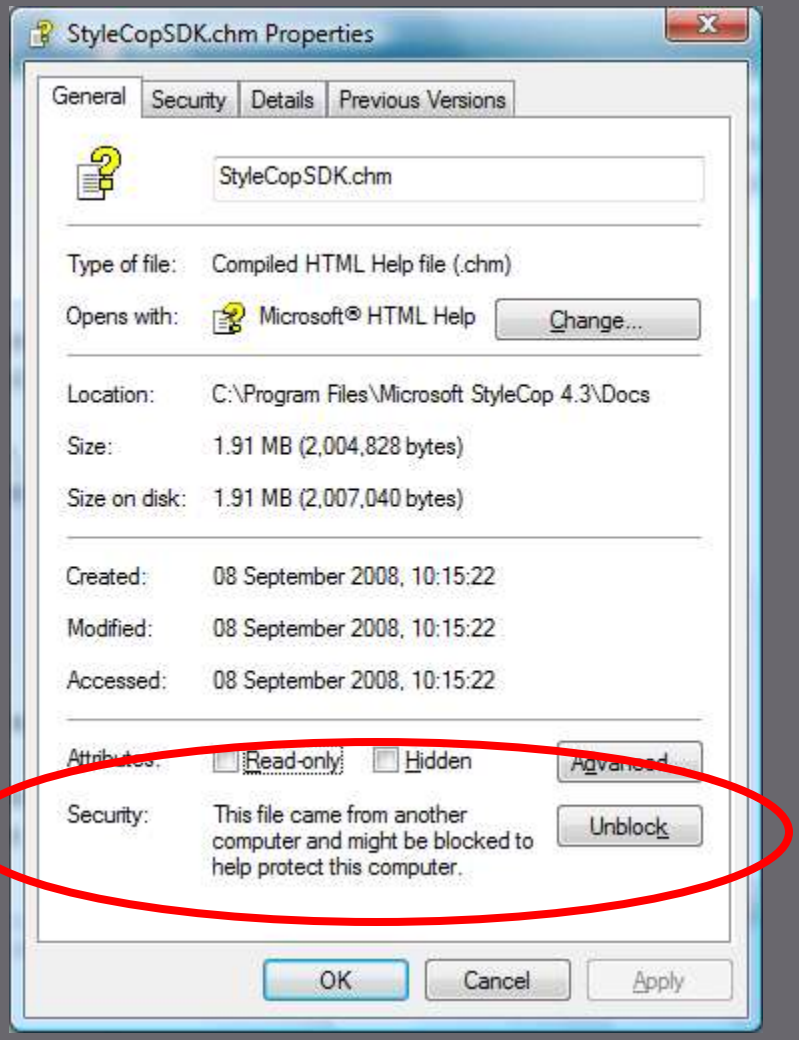
- Style Ninja is a CodeRush add-on
  - http://code.google.com/p/dxcorecommunityplugins/wiki/CR_StyleNinja
  - Open Source, written by Rory Becker (@RoryBecker)
  - Fixes 20 StyleCop issues (plus 5 others)
  - Works with C#, Visual Basic.NET and other languages

'Favourite' StyleCop Rules

- SA1027 Tabs must not be used. Use spaces instead.
- SA1106 The code contains an extra semi-colon.
- SA1623 The property's summary text must begin with: Gets or sets
- SA1309 Fieldnames must not start with an underscore
- SA1308 Variable names must not start with m_
- SA1117 All parameters must be placed on the same line or on separate lines
- SA1122 Use string.Empty instead of ""
- SA1124 Do not use regions (disabled)

StyleCop SDK

- The StyleCop SDK provides documentation on how to:-
  - write StyleCop rules
  - integrate with custom build environments
- It is a separate download
- You need to Unblock it before you can see the content



Custom StyleCop Rules

- Follow these broad steps to create a rule:-
  - Create a new Class Library
  - Add a new rule class
  - Add an XML file describing the rule
  - Copy the assembly to the StyleCop folder

Create A StyleCop Rule Class (1 of 2)

- Add a reference to Microsoft.StyleCop and Microsoft.StyleCop.CSharp
- Add a rule class:-

```
[SourceAnalyzer(typeof(CsParser))]
public class CodeMustNotContainHardcodedStringsRule : SourceAnalyzer
{
    public override void AnalyzeDocument(CodeDocument document)
    {
        // add code here
    }
}
```

Create A StyleCop Rule Class (2 of 2)

```
public override void AnalyzeDocument(CodeDocument document)
{
    CsDocument csharpDocument = (CsDocument)document;
    if (csharpDocument.RootElement != null &&
      !csharpDocument.RootElement.Generated)
    {
        for (Node<CsToken> tokenNode = csharpDocument.Tokens.First;
          tokenNode != null; tokenNode = tokenNode.Next)
        {
            if (tokenNode.Value.CsTokenType == CsTokenType.String)
            {
                this.AddViolation(csharpDocument.RootElement,
                tokenNode.Value.LineNumber,
                "CodeMustNotContainHardcodedStrings");
            }
        }
    }
}
```

Describe The Rule

- Add an XML file to the project
  - It must have the same name as the rule class
  - Set the Build Action to Embedded Resource

```
<SourceAnalyzer Name="StyleCop Custom Rules">
  <Description>
    Custom rules
  </Description>
  <Rules>
    <RuleGroup Name="String Rules">
      <Rule Name="CodeMustNotContainHardcodedStrings" CheckId="GS1001">
        <Context>The code should not contain any hard-coded strings.
        </Context>
        <Description>Validates that the code does not contain any hard-
coded strings.</Description>
      </Rule>
    </RuleGroup>
  </Rules>
</SourceAnalyzer>
```

JSL StyleCop

- New custom rules:-
  - Members Must be in Alphabetical Order
  - Don't Compare to Boolean Constants
  - Local Variable Assignment Unused
  - Local Variables Should be Set Where They are Declared
  - Move Variable Declaration to Usage Level
- Download from:-
  - http://jslstylecop.codeplex.com/

Future Releases Of StyleCop

- The following features may appear in a future release:-
  - StyleCop For Visual Basic.NET (and others ?)
  - Automatically fix style errors

Summary

- StyleCop applies style rules to source code
- StyleCop can be integrated into the build process
  - StyleCop violations can be treated as errors
- ExcludeStyleCop can be used to integrate StyleCop into existing projects on an iterative basis
- You can write your own custom rules to enforce your own standards