Developer Tools for .NET

Lesson 2:

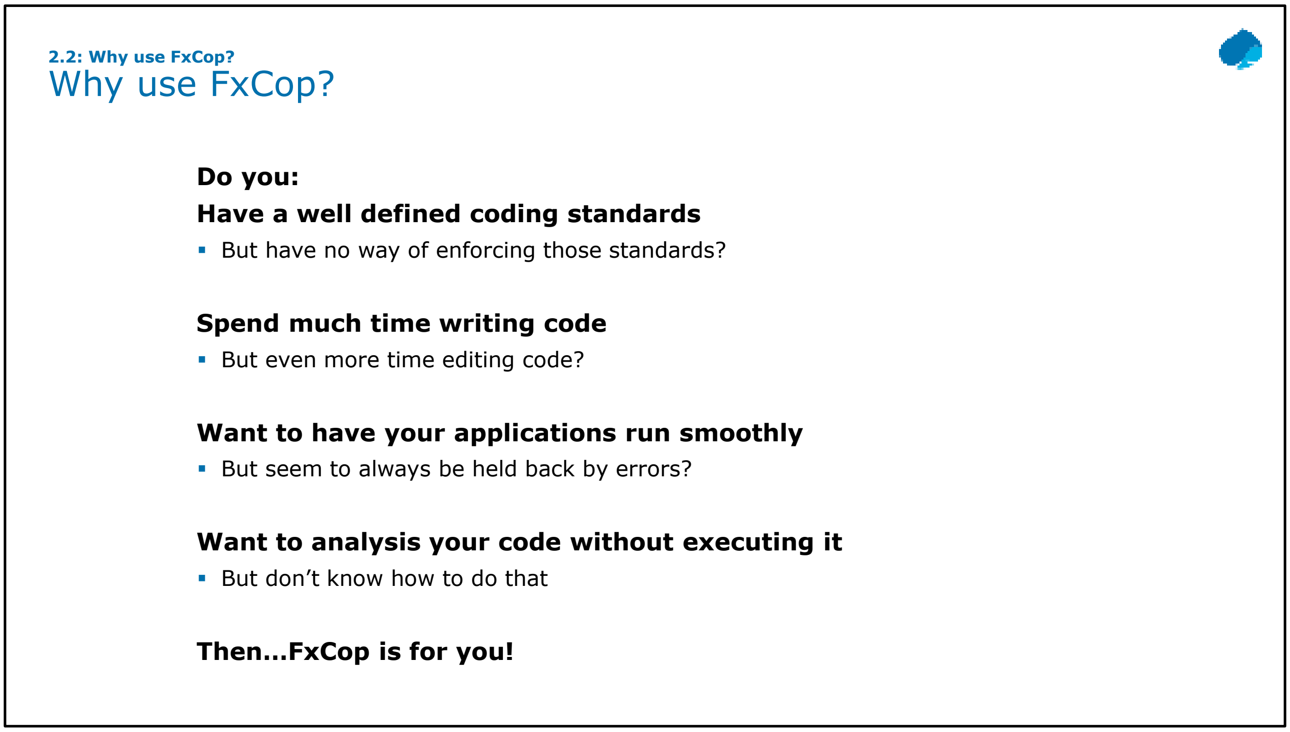
Overview of FxCOP, StyleCop & Log4NET

Static Code Analysis

From Wikipedia: “The analysis of computer software that is performed without actually executing programs built from that software”

- In most cases the analysis is performed on some version of the source code and in the other cases some form of the object code

In .NET, this means analyzing managed compiled code (IL)



2.2: Why use FxCop?

Why use FxCop?

Do you:

Have a well defined coding standards

- But have no way of enforcing those standards?

Spend much time writing code

- But even more time editing code?

Want to have your applications run smoothly

- But seem to always be held back by errors?

Want to analysis your code without executing it

- But don't know how to do that

Then…FxCop is for you!

The nuances of writing code present many problems, and perhaps the most annoying are those that involve the tendency for small errors to plague your code.

FxCop is designed to solved these problems.

2.3: What is FxCOP?

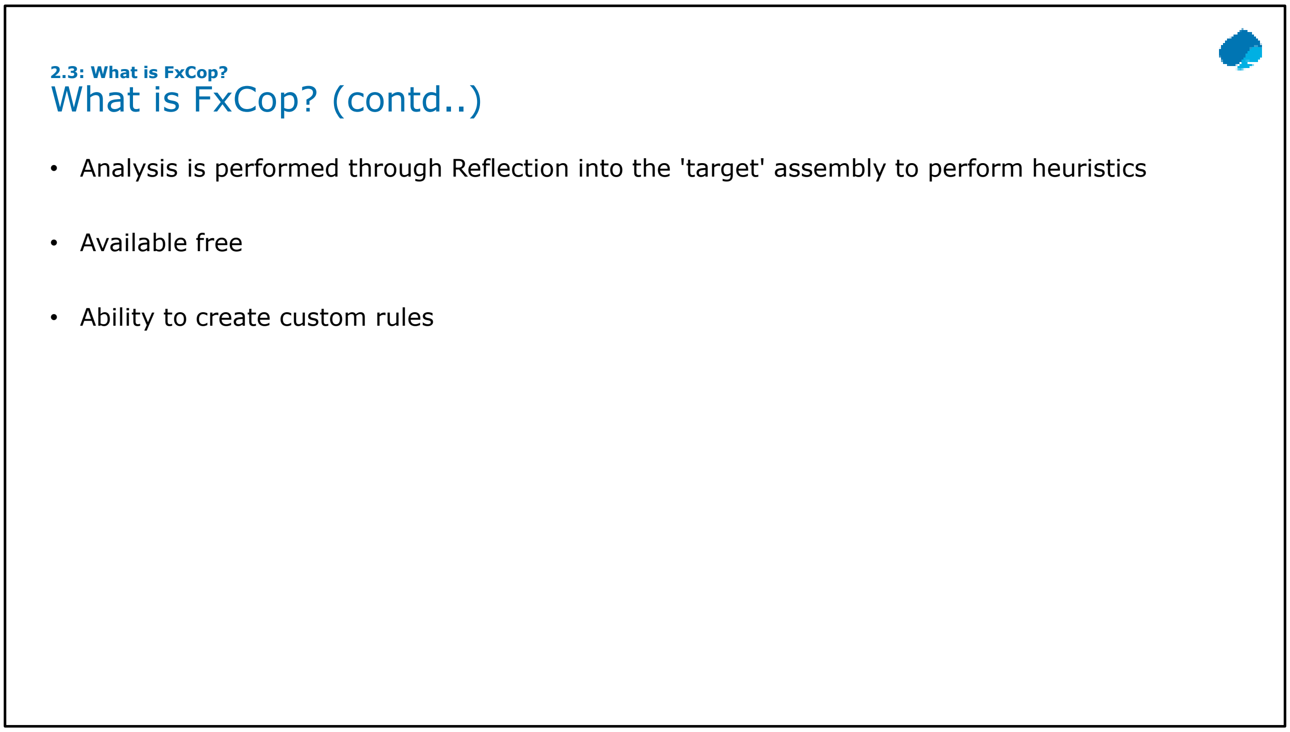
What is FxCop?

- FxCop is an application that analyzes managed code assemblies and reports information about the assemblies, such as possible design, localization, performance, and security improvements
- Enforces adherence to .NET Framework Design Guidelines
- Internal tool developed by Microsoft to fix common design errors during .NET 1.0 development

Many of the issues concern violations of the programming and design rules set forth in the Microsoft .NET Framework Design Guidelines, which are Microsoft's guidelines for writing robust and easily maintainable code using the .NET Framework.
FxCop is intended for class library developers; however, anyone creating applications that should conform to .NET Framework best practices will benefit.

What is FxCop? (contd..)

- Contains a set of rules that match the Microsoft .NET Framework Design Guidelines
- Encourages learning good design by writing code rather than reading guidelines
- FxCop is also useful as an educational tool for those who are new to the .NET Framework or are unfamiliar with the .NET Framework Design Guidelines



2.3: What is FxCop?

What is FxCop? (contd..)

- Analysis is performed through Reflection into the 'target' assembly to perform heuristics
- Available free
- Ability to create custom rules

Microsoft has put a lot of effort into disseminating best practices and guidelines for writing code to be used for the .NET framework. The FxCop project began as an internal one, aimed at ensuring that Microsoft developers followed their own rules.

FxCop is now available from http://www.gotdotnet.com/team/fxcop.

Introspection is the new element of FxCop, contained in version 1.3. The old Reflection version required FxCop to shutdown to complete a fix, then run again to recompile. Now, since the assemblies are not locked, there is no need to shut down. Those who have used the old Reflection engine will be happy to hear that rules can be updated by just changing a few base class names.

The rules for FxCop include everything from ensuring that you use COM interop correctly, to ensuring proper globalization, to enforcing rules for writing high-performance code

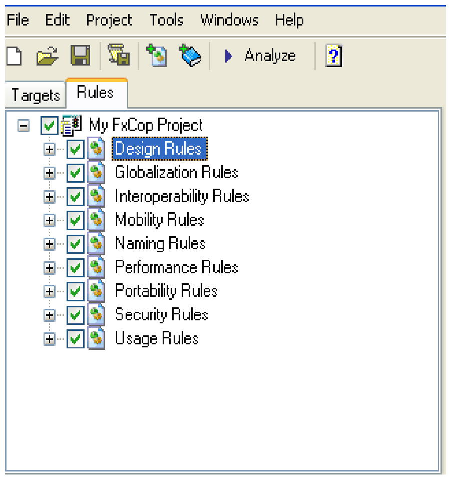
FxCop Rules

It Contains over 200 rules in the following areas:

- Library design
- Localization
- Naming conventions
- Performance
- Security

Defined in logical groups

Extensible – you can write your own rules!



My FxCop Project
- Design Rules
- Globalization Rules
- Interoperability Rules
- Mobility Rules
- Naming Rules
- Performance Rules
- Portability Rules
- Security Rules
- Usage Rules

FxCop Rules (contd..)

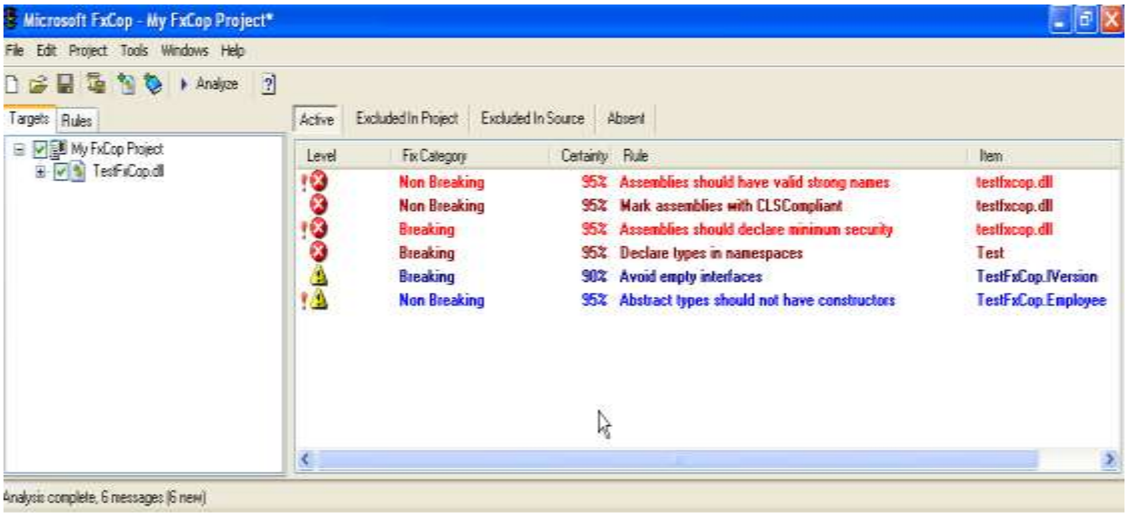
FxCop tests rules against assemblies and reports on failed rules

- FxCop can be applied to any .NET language because it works on assemblies and not code
- The rules included with FxCop are based upon "Microsoft .NET Framework Design Guidelines"

FxCOP Tool

- FxCop is designed to be fully integrated into the software development cycle
- It is distributed as a fully featured application with a graphical user interface (FxCop.exe) for interactive work
- Also available as a command-line tool (FxCopCmd.exe) suitable for use as part of automated build processes
- It can be integrated with Microsoft Visual Studio .NET as an external tool

FxCop in Action


Microsoft FxCop - My FxCop Project*
File Edit Project Tools Windows Help
Analyze
Targets Rules
My FxCop Project
TestFxCop.dll
Active Excluded In Project Excluded In Source Absent

Level	Fix Category	Certainty	Rule	Item
! ⊗	Non Breaking	95%	Assemblies should have valid strong names	testfxcop.dll
⊗	Non Breaking	95%	Mark assemblies with CLSCompliant	testfxcop.dll
! ⊗	Breaking	95%	Assemblies should declare minimum security	testfxcop.dll
⊗	Breaking	95%	Declare types in namespaces	Test
⚠	Breaking	90%	Avoid empty interfaces	TestFxCop.IVersion
! ⚠	Non Breaking	95%	Abstract types should not have constructors	TestFxCop.Employee

Analysis complete, 6 messages (6 new)

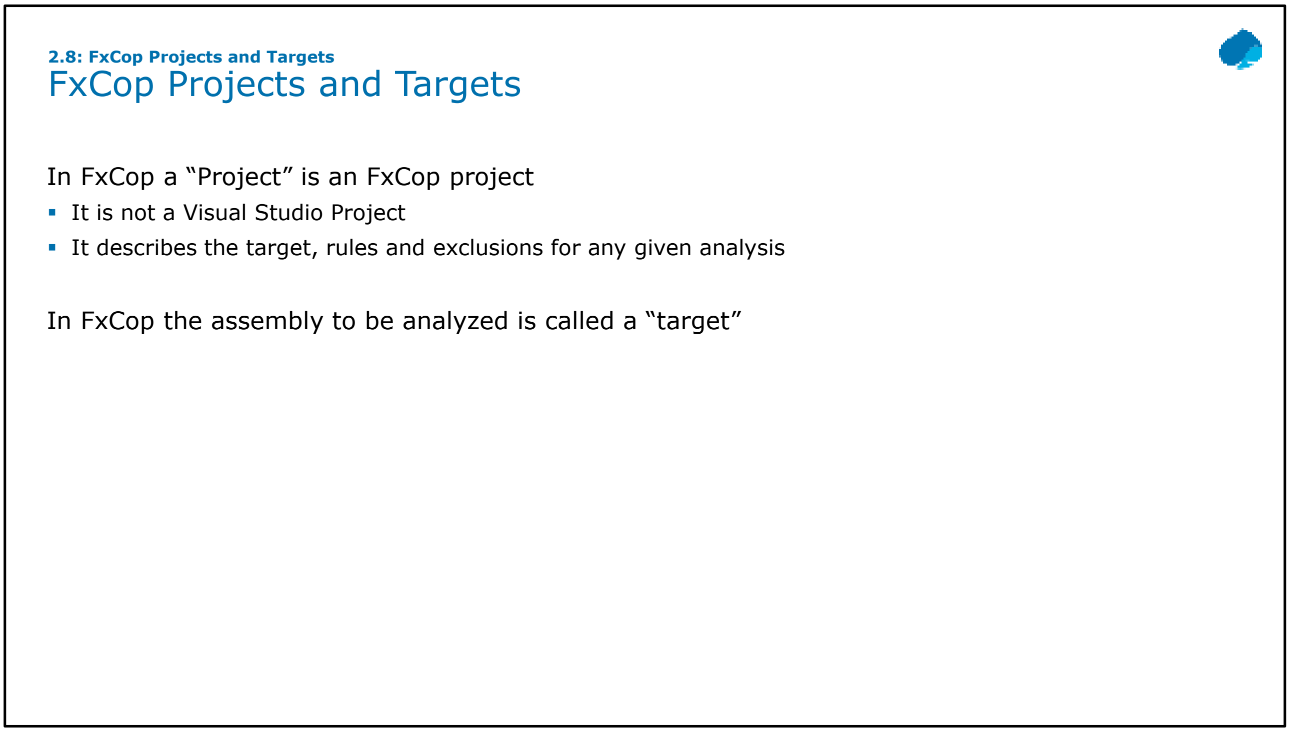
How does FxCop Work?

- FxCop analyzes programming elements in managed assemblies, called targets, using rules that when violated, return informational messages about the targets
- A report containing these messages appears in the user interface, or in the output window when using the command-line tool
- Messages identify any relevant programming and design issues and, possible, supply information on how to fix the target

FxCop analyzes programming elements in managed assemblies, called targets, and provides an informational report containing messages about the targets, including suggestions on how to improve the source code used to generate them. FxCop represents the checks it performs during an analysis as rules. A rule is managed code that can analyze targets and return a message about its findings. Rule messages identify any relevant programming and design issues and, when possible, supply information on how to fix the target.

How does FxCop Work? (contd..)

- A message is associated with a specific rule and a specific target and can be excluded from further analysis
- A default set of rules is provided and additional custom rules can be created using the FxCop SDK



2.8: FxCop Projects and Targets

FxCop Projects and Targets

In FxCop a “Project” is an FxCop project

- It is not a Visual Studio Project
- It describes the target, rules and exclusions for any given analysis

In FxCop the assembly to be analyzed is called a “target”

Projects are used to specify the set of assemblies to be analyzed, the rules used to analyze the assemblies, the excluded messages, and settings to customize the saved project or report file.

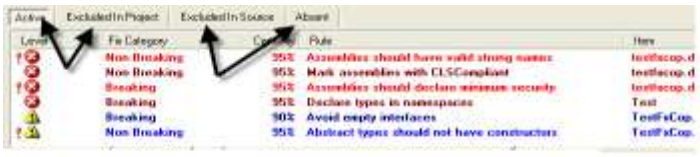
Using FxCop

Using FxCop, you can:

- Control which rules are applied to targets
- Exclude a rule message from future reports
- Apply style sheets to FxCop reports
- Filter and save messages
- Save and reuse application settings in FxCop projects

FxCop Analysis Result

- Active – Current issues
- Excluded In Project – Issue suppressed for the entire project
- Excluded In Source – Individual issue suppressed at the source level
- Absent – Issues that have been resolved

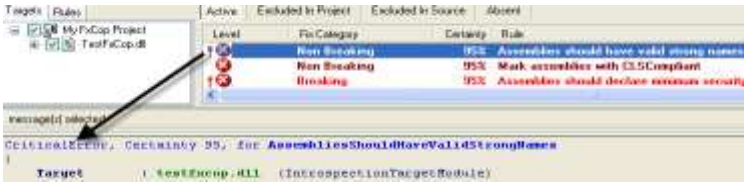


Level	Fix Category	Certainty	Rule	Item
! ❌	Non Breaking	95%	Assemblies should have valid strong names	testfxcop.d
❌	Non Breaking	95%	Mark assemblies with CLSCompliant	testfxcop.d
! ❌	Breaking	95%	Assemblies should declare minimum security	testfxcop.d
❌	Breaking	95%	Declare types in namespaces	Test
⚠	Breaking	90%	Avoid empty interfaces	TestFxCop.
! ⚠	Non Breaking	95%	Abstract types should not have constructors	TestFxCop.

FxCop Analysis Result (contd..)

Levels

- Critical Error
- Error
- Critical Warning
- Warning
- Informational



Level	Fix Category	Certainty	Rule
	Non Breaking	95%	Assemblies should have valid strong names
	Non Breaking	95%	Mark assemblies with CLSCompliant
	Breaking	95%	Assemblies should declare minimum security

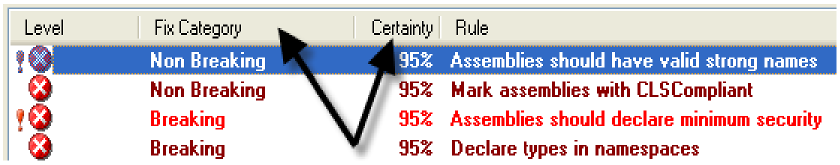
```
CriticalError, Certainty 95, for AssembliesShouldHaveValidStrongNames
{
    Target        : testFxcop.dll  (IntrospectionTargetModule)
```

FxCop Analysis Result (contd..)

Fix Category

- Breaking – resolving this issue will cause dependent code to break; recompilation needed
- Non Breaking – issue can be resolved without breaking dependent code

Certainty – How sure it is an issue

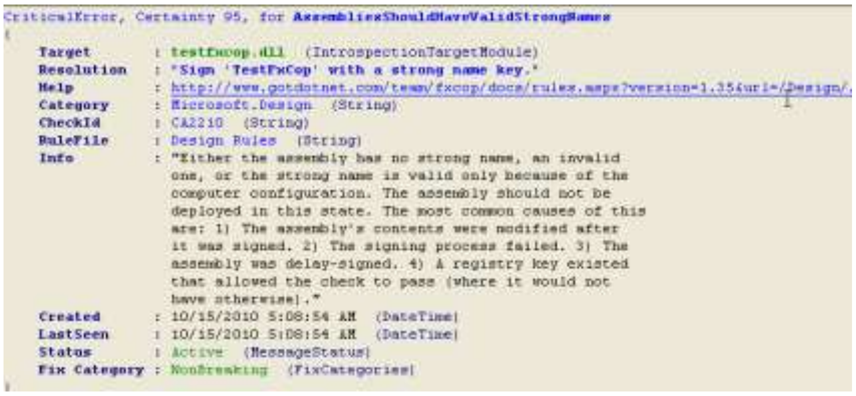


Level	Fix Category	Certainty	Rule
	Non Breaking	95%	Assemblies should have valid strong names
	Non Breaking	95%	Mark assemblies with CLSCompliant
	Breaking	95%	Assemblies should declare minimum security
	Breaking	95%	Declare types in namespaces

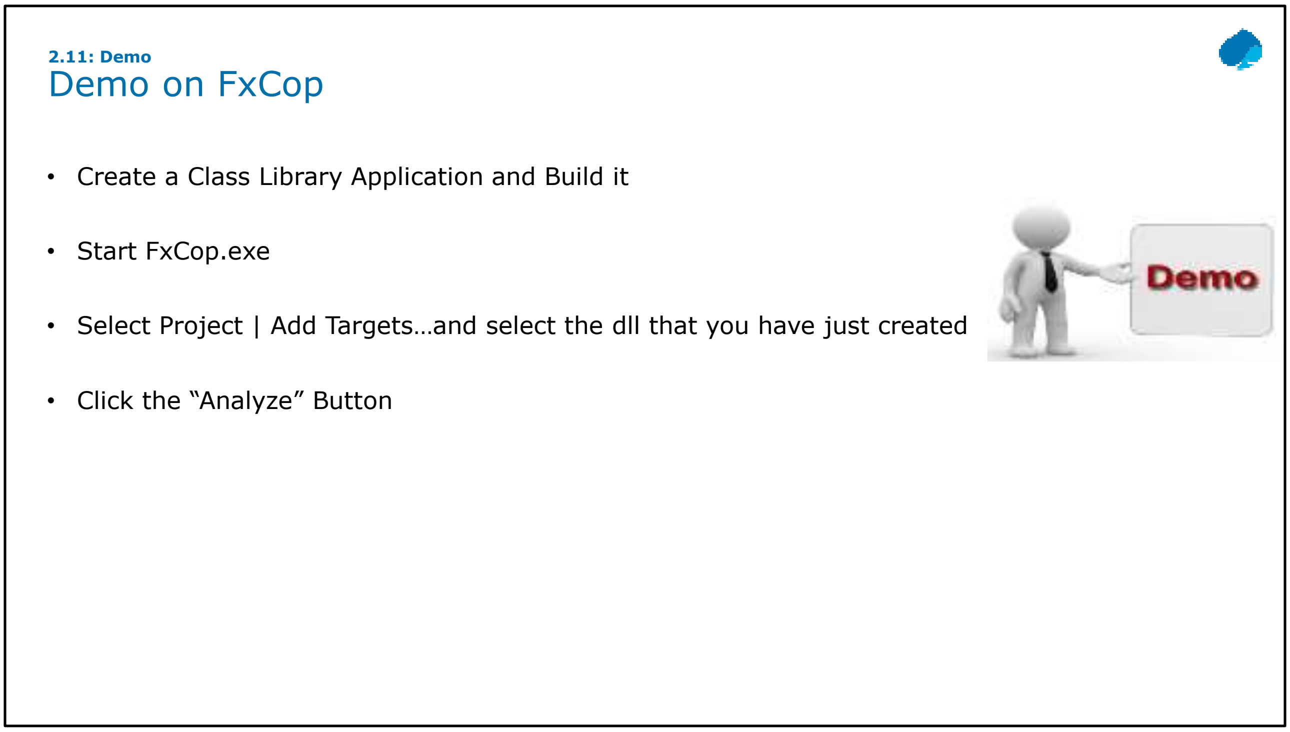
FxCop Analysis Result (contd..)

Details pane for a selected issue

- Where the issue is located, brief description, link to more info, etc



```
CriticalError, Certainty 95, for AssembliesShouldHaveValidStrongNames
{
    Target       : testfxcop.dll  (IntrospectionTargetModule)
    Resolution   : "Sign 'TestFxCop' with a strong name key."
    Help         : http://www.gotdotnet.com/team/fxcop/docs/rules.aspx?version=1.35&url=/Design/
    Category     : Microsoft.Design  (String)
    CheckId      : CA2210  (String)
    RuleFile     : Design Rules  (String)
    Info         : "Either the assembly has no strong name, an invalid
                   one, or the strong name is valid only because of the
                   computer configuration. The assembly should not be
                   deployed in this state. The most common causes of this
                   are: 1) The assembly's contents were modified after
                   it was signed. 2) The signing process failed. 3) The
                   assembly was delay-signed. 4) A registry key existed
                   that allowed the check to pass (where it would not
                   have otherwise)."
    Created      : 10/15/2010 5:08:54 AM  (DateTime)
    LastSeen     : 10/15/2010 5:08:54 AM  (DateTime)
    Status       : Active  (MessageStatus)
    Fix Category : NonBreaking  (FixCategories)
}
```



2.11: Demo

Demo on FxCop

- Create a Class Library Application and Build it
- Start FxCop.exe
- Select Project | Add Targets…and select the dll that you have just created
- Click the "Analyze" Button

<u>Demonstration: HelloWorld</u>:
Let us take a look at the first ASP.NET application. To create this application follow the steps given below:

Step 1: Open Visual Studio 2013. Select **File → New-WebSite**. It opens the **New WebSite** dialog box.
With ASP.NET 4.5, using Visual Studio 2013 you have an option to create an application with virtual directory mapped to IIS or a standalone application outside the confines of IIS.
Built-in Web Server: As mentioned earlier ASP.Net 4.5 comes with a built-in Web Server called as **Cassini**. By default, VS2013 builds applications without the use of IIS. The location provided by default for your application is 'C:\Documents and Settings\username\My Documents\Visual Studio 2013\WebSites'. Optionally you can also create your own folder and allow your WebSites to be created in that location. If you use the Built-in Web Server, then you are not locked into the Websites folder or in the 'C:\Inetpub\wwwroot' folder. You can also run your application completely from this location. Hence with this new way of creating ASP.NET applications you are not dependent on having access to any IIS server and you can go ahead and develop applications on any Windows OS machine.

The figure on the following page shows the creation of website on the FileSystem, that is in the Built-in Web Server.

What is StyleCop?

- StyleCop is a source analysis tool for C#
- It can be used for analyzing source code as opposed to compiled assemblies which is the area for FxCop
- StyleCop is currently in version 4.7 and can be downloaded from http://stylecop.codeplex.com
- It has been used by Microsoft to help teams enforce a common set of best practices for layout, readability, maintainability, and documentation of C# source code

What is StyleCop? (contd..)

- StyleCop is similar in many ways to Microsoft Code Analysis (specifically FxCop), but there are some important distinctions
- FxCop performs its analysis on compiled binaries, while StyleCop analyzes the source code directly
- FxCop focuses more on the design of the code, while StyleCop focuses on layout, readability and documentation

StyleCop Rules

Specifically, these rules cover the following, in no particular order:

- Layout of elements, statements, expressions, and query clauses
- Placement of curly brackets, parenthesis, square brackets, etc
- Spacing around keywords and operator symbols
- Line spacing
- Placement of method parameters within method declarations or method calls
- Standard ordering of elements within a class

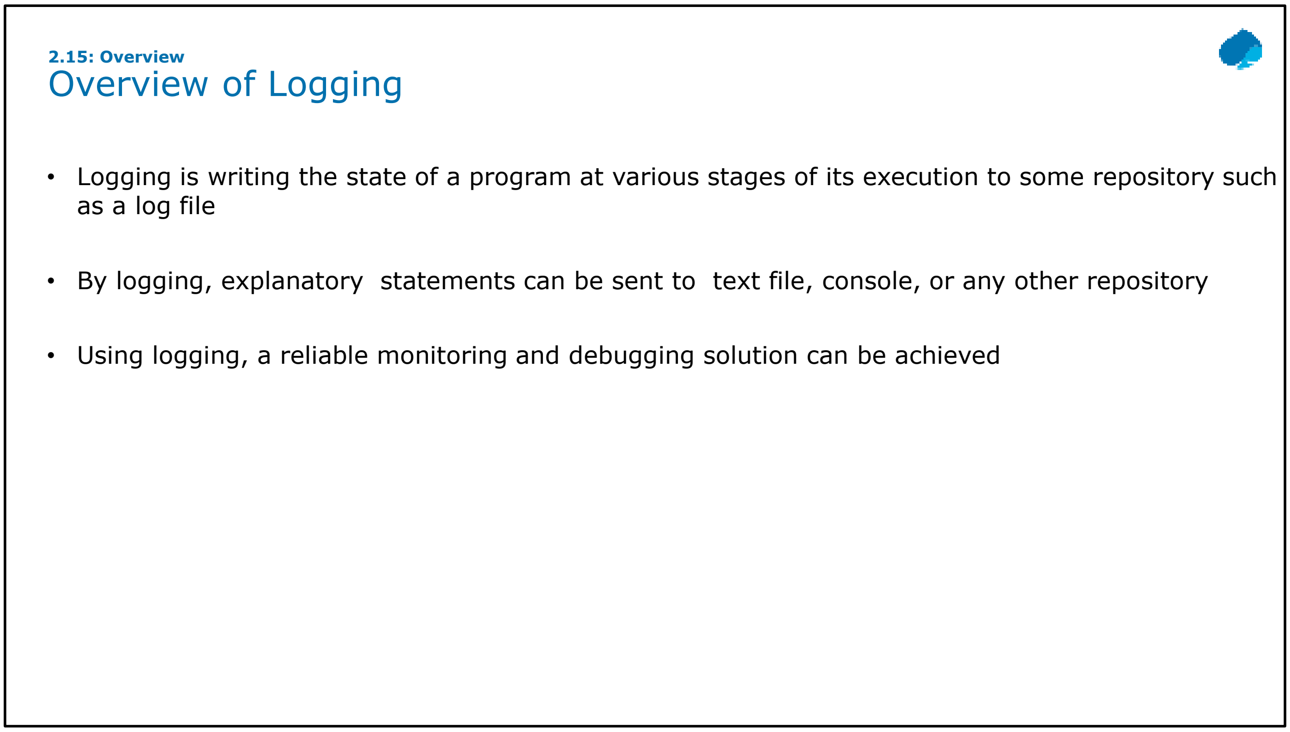
StyleCop Rules (contd..)

- Formatting of documentation within element headers and file headers
- Naming of elements, fields and variables
- Use of the built-in types
- Use of access modifiers
- Allowed contents of files
- Debugging text

2.14: Demo

Demo on StyleCop

Create a Class Library Application and Build it
Go To Tools – Run StyleCop
That's it…



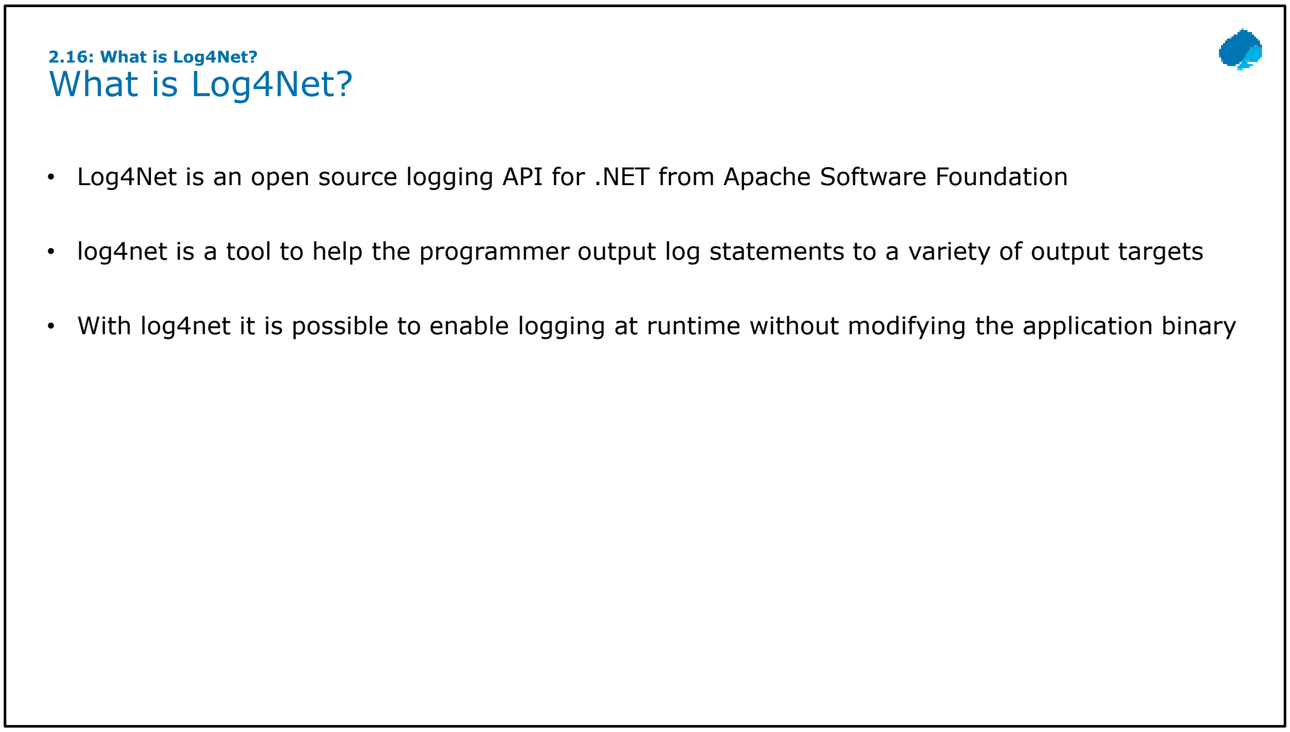
2.15: Overview

Overview of Logging

- Logging is writing the state of a program at various stages of its execution to some repository such as a log file
- By logging, explanatory statements can be sent to text file, console, or any other repository
- Using logging, a reliable monitoring and debugging solution can be achieved

Logging is used due to the following reasons:

It can be used for debugging.
It is cost effective than including some debug flag.
There is no need to recompile the program to enable debugging.
It does not leave your code messy.
Priority levels can be set.
Log statements can be appended to various destinations such as file, console, socket, database, and so on.
Logs are needed to quickly target an issue that occurred in service.

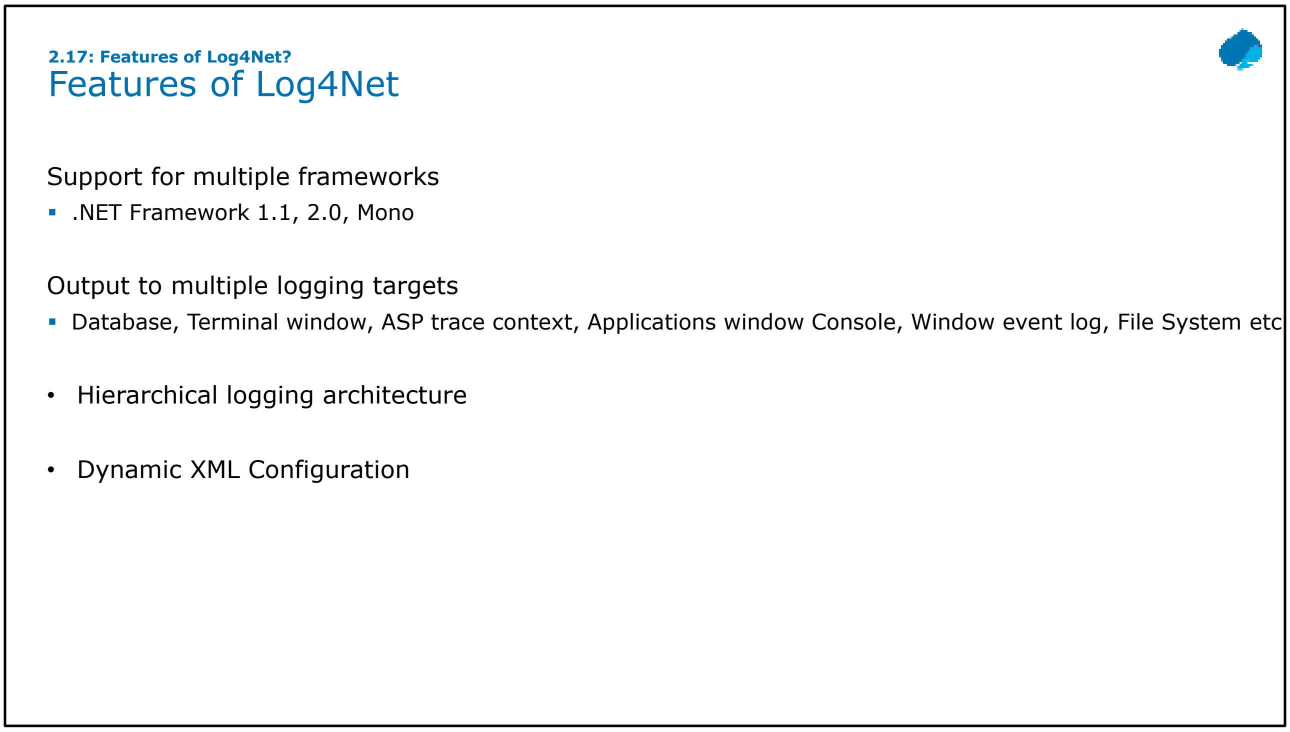


2.16: What is Log4Net?

What is Log4Net?

- Log4Net is an open source logging API for .NET from Apache Software Foundation
- log4net is a tool to help the programmer output log statements to a variety of output targets
- With log4net it is possible to enable logging at runtime without modifying the application binary

log4net provides many advantages over other logging systems, which makes it a perfect choice for use in any type of application, from a simple single-user application to a complex multiple-threaded distributed application using remoting



2.17: Features of Log4Net?

Features of Log4Net

Support for multiple frameworks

- .NET Framework 1.1, 2.0, Mono

Output to multiple logging targets

- Database, Terminal window, ASP trace context, Applications window Console, Window event log, File System etc

- Hierarchical logging architecture

- Dynamic XML Configuration

Hierarchical logging is an ideal fit with component based development. Each component has its own of logger. When individually tested, the properties of these loggers may be set as the developer requires. When combined with other components, the loggers inherit the properties determined by the integrator of the components. One can selectively elevate logging priorities on one component without affecting the other components. This is useful when you need a detailed trace from just a single component without crowding the trace file with messages from other components. All this can be done through configuration files; no code changes are required.

log4net is configured using an XML configuration file. The configuration information can be embedded within other XML configuration files (such as the application's .config file) or in a separate file. The configuration is easily readable and updateable while retaining the flexibility to express all configurations. Alternatively log4net can be configured programmatically.

Components of Log4Net

- Log4net is built using the layered approach, with the following components inside of the framework
  - Logger, Appender, & Layout
- Users can extend these basic classes to create their own loggers, appenders, and layouts

The Logger is the main component with which your application interacts. It is also the component that generates the log messages.
Generating a log message is different than actually showing the final output. The output is showed by the Layout component.
The logger provides you with different methods to log any message. You can create multiple loggers inside of your application.
Any good logging framework should be able to generate output for multiple destinations, such as outputting the trace statements to the console or serializing it into a log file. log4net is a perfect match for this requirement. It uses a component called *Appender* to define this output medium. As the name suggests, these components append themselves to the Logger component and relay the output to an output stream. You can append multiple appenders to a single logger.
The Layout component is used to display the final formatted output to the user. The output can be shown in multiple formats, depending upon the layout we are using. It can be linear or an XML file. The layout component works with an appender. There is a list of different layouts in the API documentation. You cannot use multiple layouts with an appender. To create your own layout, you need to inherit the log4net.Layout.LayoutSkeleton class, which implements the ILayout interface.

2.19: Demo

Demo on Log4Net

Summary

In this lesson, you have learnt:

- What is the use of FxCop as a Code Analysis Tool
- How to use the FxCop tool to detects errors in Managed Code
- StyleCop as a Source Code Analysis Tool
- How to use StyleCop
- Log4Net and its Components

People matter, results count.

About Capgemini

With more than 190,000 people, Capgemini is present in over 40 countries and celebrates its 50th Anniversary year in 2017. A global leader in consulting, technology and outsourcing services, the Group reported 2016 global revenues of EUR 12.5 billion. Together with its clients, Capgemini creates and delivers business, technology and digital solutions that fit their needs, enabling them to achieve innovation and competitiveness. A deeply multicultural organization, Capgemini has developed its own way of working, the Collaborative Business Experience™, and draws on Rightshore®, its worldwide delivery model.

Learn more about us at

www.capgemini.com