BASTA! ON TOUR

Rainer Stropek | software architects gmbh

Im Sandkasten

Professionelle Codedokumentation mit Sandcastle & Co

Abstract (German)

In Zeiten von erhöhtem Kostendruck ist Wiederverwendung ein aktuelleres Thema denn je. Egal, ob einmal entwickelte Komponenten intern wiederverwendet oder ob Bibliotheken verkauft werden sollen, Dokumentation ist wichtig. Während in früheren Zeiten Dokumentation getrennt vom Sourcecode gepflegt wurde, geht man heute dazu über, Code und Dokumentation miteinander zu verknüpfen und dadurch immer aktuell zu halten. Rund um Visual Studio gibt es mit Sandcastle eine Sammlung von Tools, die es ermöglicht, eine professionelle Dokumentation für Programmbibliotheken zu erstellen. Im Workshop gibt Rainer Stropek einen Überblick über die Werkzeuge und zeigt an einem praxisbezogenen Beispiel, wie sie eingesetzt werden. Er verwendet dabei ausschließlich kostenfreie Produkte rund um Sandcastle. Der Workshop ist ideal für Entwickler, die für die Erstellung wiederverwendbarer Bibliotheken verantwortlich sind.

Introduction

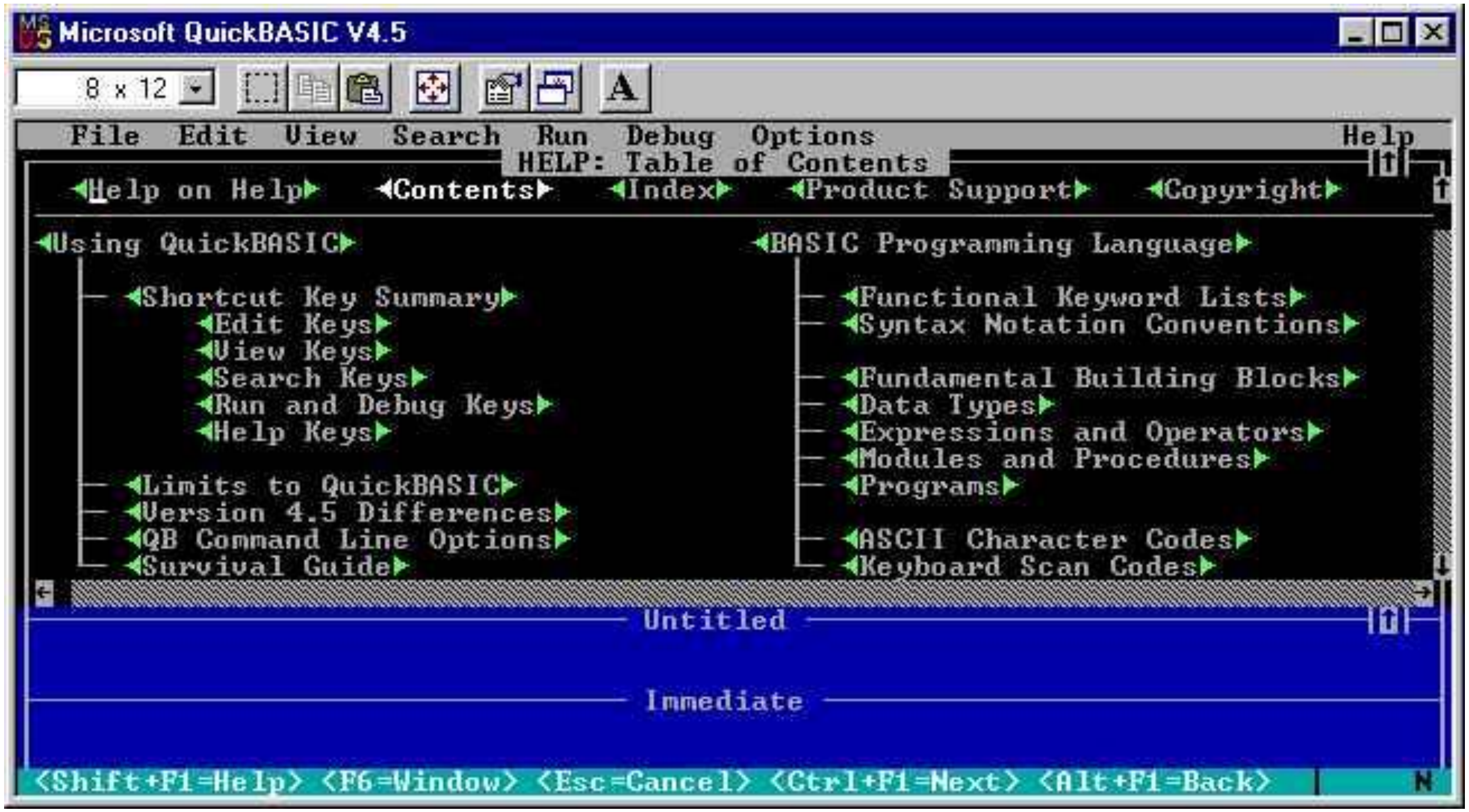
- software architects gmbh
- Rainer Stropek
- Developer, Speaker, Trainer
- MVP for Windows Azure
- rainer@timecockpit.com
- twitter FOLLOW ME @rstropek

http://www.timecockpit.com

http://www.software-architects.com

The Windows Help System

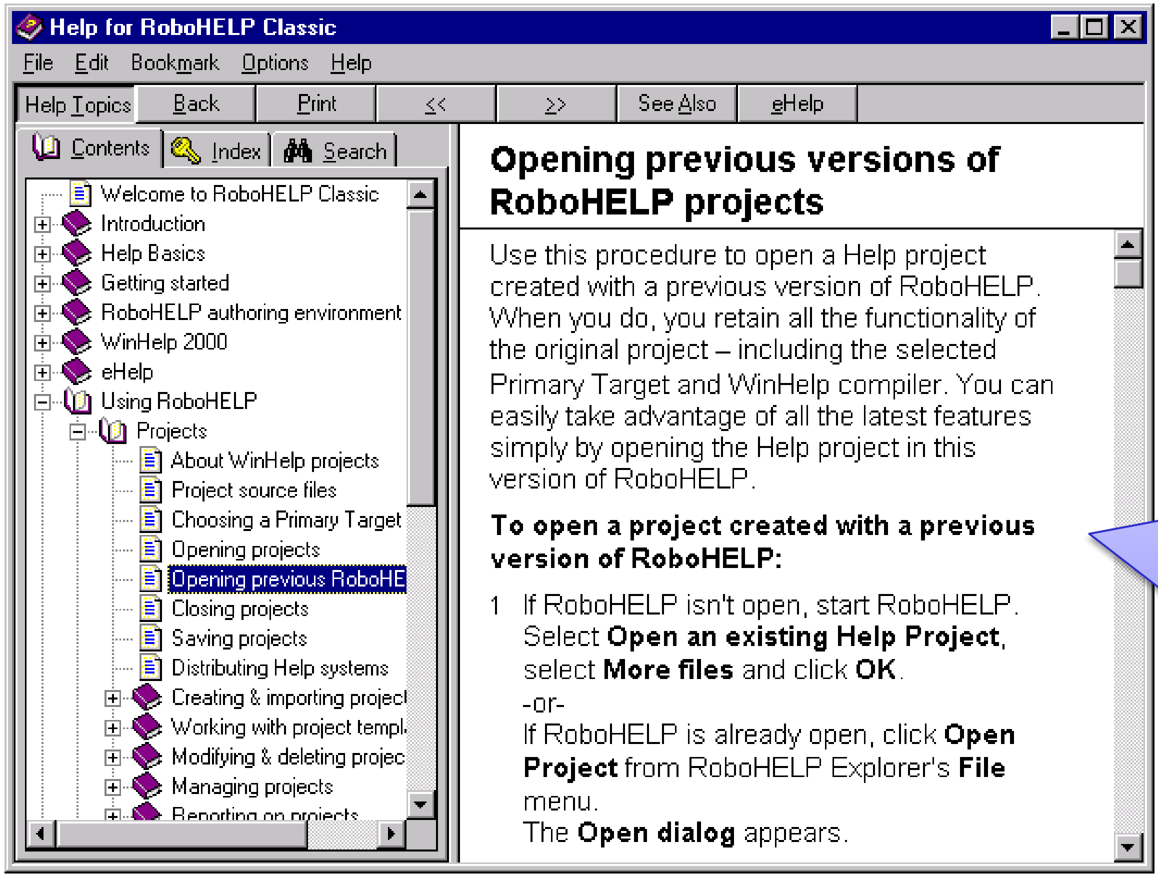
THE HISTORY



```
Microsoft QuickBASIC V4.5
8 x 12
 File  Edit  View  Search  Run  Debug  Options                          Help
                          HELP: Table of Contents
  ◄Help on Help►  ◄Contents►  ◄Index►  ◄Product Support►  ◄Copyright►

◄Using QuickBASIC►                             ◄BASIC Programming Language►
  │                                             │
  ├─ ◄Shortcut Key Summary►                     ├─ ◄Functional Keyword Lists►
  │      ◄Edit Keys►                            ├─ ◄Syntax Notation Conventions►
  │      ◄View Keys►                            │
  │      ◄Search Keys►                          ├─ ◄Fundamental Building Blocks►
  │      ◄Run and Debug Keys►                   ├─ ◄Data Types►
  │      ◄Help Keys►                            ├─ ◄Expressions and Operators►
  │                                             ├─ ◄Modules and Procedures►
  ├─ ◄Limits to QuickBASIC►                     ├─ ◄Programs►
  ├─ ◄Version 4.5 Differences►                  │
  ├─ ◄QB Command Line Options►                  ├─ ◄ASCII Character Codes►
  └─ ◄Survival Guide►                           └─ ◄Keyboard Scan Codes►
──────────────────────────────── Untitled ────────────────────────────────

──────────────────────────────── Immediate ───────────────────────────────

<Shift+F1=Help> <F6=Window> <Esc=Cancel> <Ctrl+F1=Next> <Alt+F1=Back>     N
```

Where everything started: *QuickHelp*

End of 1980's


Help for RoboHELP Classic
File Edit Bookmark Options Help
Help Topics Back Print << >> See Also eHelp
Contents Index Search
Welcome to RoboHELP Classic
Introduction
Help Basics
Getting started
RoboHELP authoring environment
WinHelp 2000
eHelp
Using RoboHELP
Projects
About WinHelp projects
Project source files
Choosing a Primary Target
Opening projects
Opening previous RoboHE
Closing projects
Saving projects
Distributing Help systems
Creating & importing project
Working with project templ
Modifying & deleting projec
Managing projects
Reporting on projects

Opening previous versions of RoboHELP projects

Use this procedure to open a Help project created with a previous version of RoboHELP. When you do, you retain all the functionality of the original project – including the selected Primary Target and WinHelp compiler. You can easily take advantage of all the latest features simply by opening the Help project in this version of RoboHELP.

To open a project created with a previous version of RoboHELP:

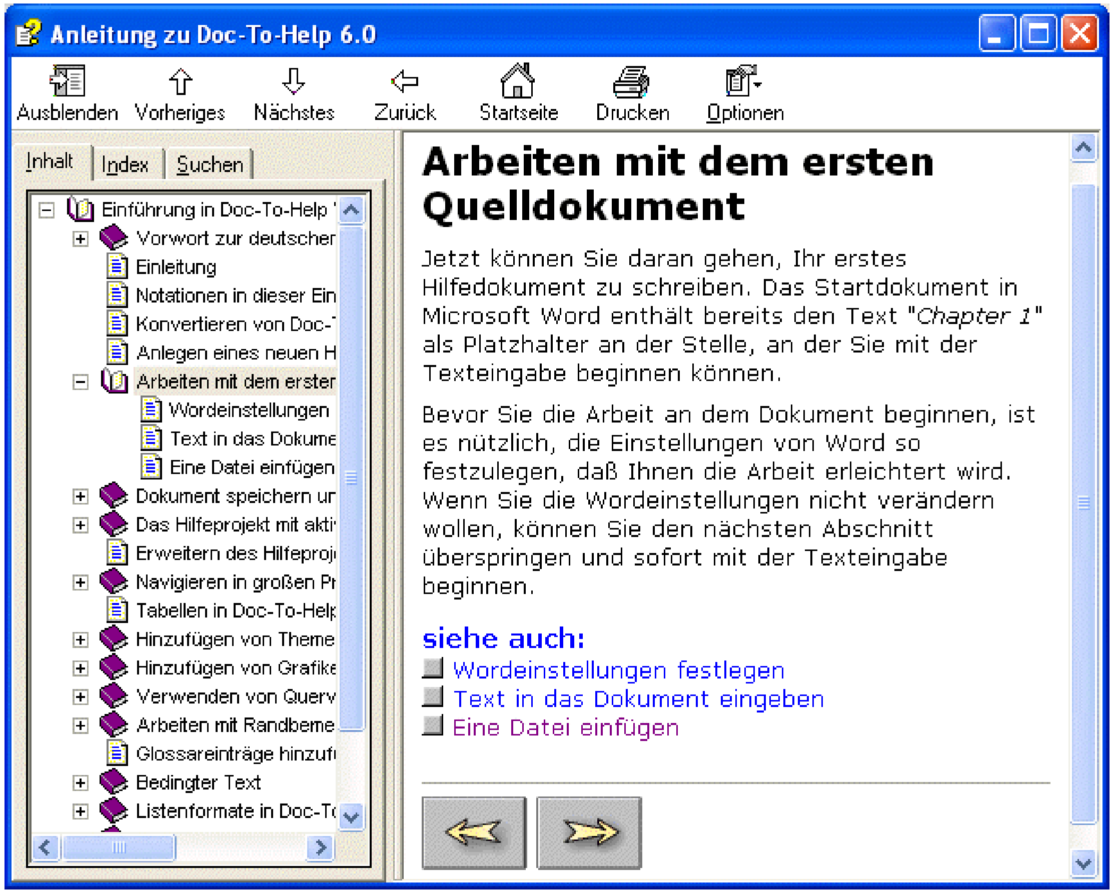
1 If RoboHELP isn't open, start RoboHELP. Select **Open an existing Help Project**, select **More files** and click **OK**.
-or-
If RoboHELP is already open, click **Open Project** from RoboHELP Explorer's **File** menu.
The **Open dialog** appears.

*"Microsoft strongly recommends that software developers **discontinue** using the Windows Help application."*

Source: MSDN

Next Evolution: *WinHelp*

Combines RTF + Images + VBA Macros
No out-of-the-box support starting with Windows Vista



Arbeiten mit dem ersten Quelldokument

Jetzt können Sie daran gehen, Ihr erstes Hilfedokument zu schreiben. Das Startdokument in Microsoft Word enthält bereits den Text "*Chapter 1*" als Platzhalter an der Stelle, an der Sie mit der Texteingabe beginnen können.

Bevor Sie die Arbeit an dem Dokument beginnen, ist es nützlich, die Einstellungen von Word so festzulegen, daß Ihnen die Arbeit erleichtert wird. Wenn Sie die Wordeinstellungen nicht verändern wollen, können Sie den nächsten Abschnitt überspringen und sofort mit der Texteingabe beginnen.

siehe auch:
- Wordeinstellungen festlegen
- Text in das Dokument eingeben
- Eine Datei einfügen

HTML Reaches The World Of Help: *HTMLHelp*

HTML + Images + JavaScript compiled into a single file

Detailed history of HTMLHelp see Wikipedia article on
Microsoft Compiled HTML Help

HTMLHelp

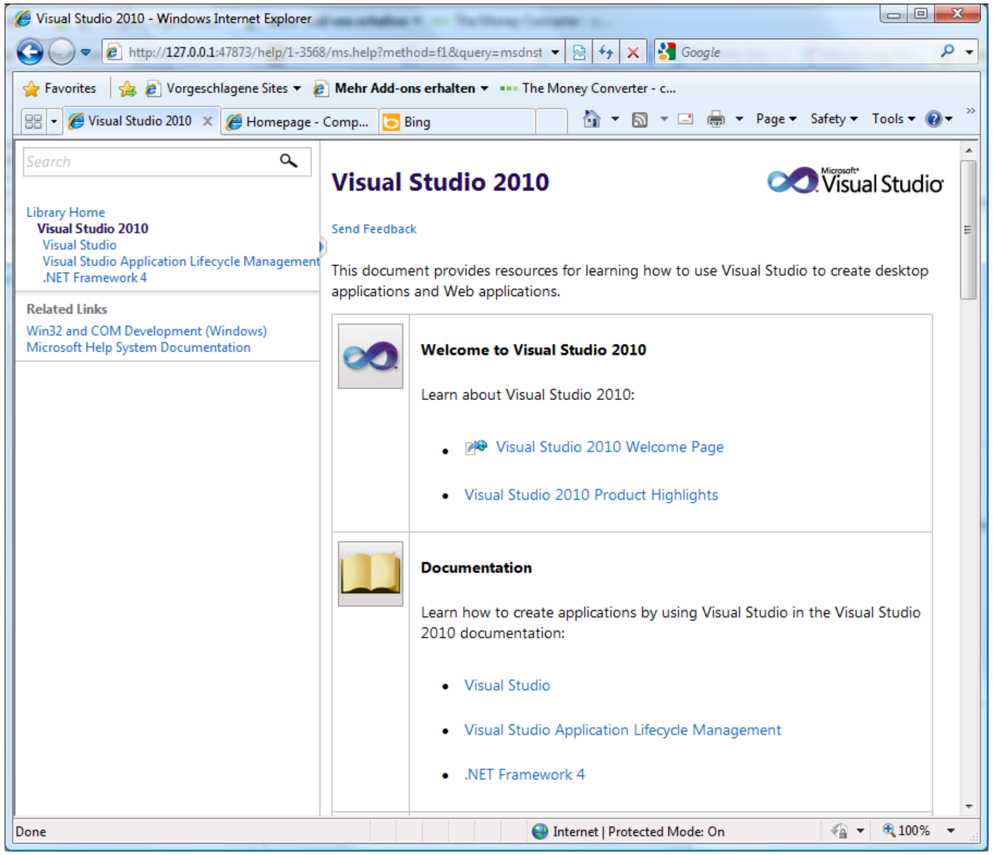
- Embedded Internet Explorer is used to display help content
  - Problem: Untrusted network locations
- Current Version is 1.4
  - Included in Windows 7

Assistance Platform 1.0 Client SDK

- Formerly *Windows Vista Help SDK*
- “Can only be used to display the Windows Help content set.”
- “[…] cannot be used by third-party programs”
- Source: MSDN

HTMLHelp 2

- Designed for the VS .NET environment only
- Not released as a general Help platform (Source: MS press announcements)
- Need more details? MS Help 2 FAQ


Visual Studio 2010 - Windows Internet Explorer

http://127.0.0.1:47873/help/1-3568/ms.help?method=f1&query=msdnst

Favorites | Vorgeschlagene Sites | Mehr Add-ons erhalten | The Money Converter - c...

Visual Studio 2010 | Homepage - Comp... | Bing

Page | Safety | Tools

Search

Library Home
Visual Studio 2010
Visual Studio
Visual Studio Application Lifecycle Management
.NET Framework 4

Related Links

Win32 and COM Development (Windows)
Microsoft Help System Documentation

Visual Studio 2010

Send Feedback

This document provides resources for learning how to use Visual Studio to create desktop applications and Web applications.

Welcome to Visual Studio 2010

Learn about Visual Studio 2010:

- Visual Studio 2010 Welcome Page
- Visual Studio 2010 Product Highlights

Documentation

Learn how to create applications by using Visual Studio in the Visual Studio 2010 documentation:

- Visual Studio
- Visual Studio Application Lifecycle Management
- .NET Framework 4

Done | Internet | Protected Mode: On | 100%

MS Help Viewer 1.0

Making everything better…

MS Help Viewer 1.0

- Called *Help3* during development
- Shipped with Visual Studio 2010
  - Replaced MS Help 2.0
  - Details see series of blog posts by Jeff Braaten (Microsoft)
  - See also video
- Future: Help for VS 2010 SP1
  - Offline Viewer
  - TOC Tree
  - Support for tabs
  - Traditional index tab
  - History and favorites
  - See also preview video

So What???

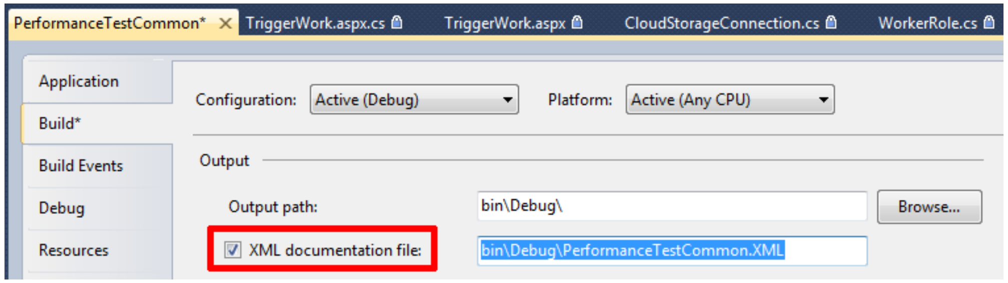
- General Windows application
  - Use HTML Help 1.x (.CHM files)
  - HTML Help ships as part of the Windows OS.
- OEM with the need to produce documentation to integrate with Windows Help
  - Assistance Platform 1.0 (.H1S files).
- Extending Visual Studio
  - Visual Studio 2010
    - Help Viewer 1.0 (.MSHC files)
    - Help Viewer 1.0 ships with Visual Studio 2010.
  - Earlier versions (2002 – 2008)
    - Microsoft Help 2.x (.HxS files)
    - Microsoft Help 2.x ships with Visual Studio 2002 – 2008.
- Assistance Platform 1.0, Microsoft Help 2.x and Help Viewer 1.0 are not available for general 3rd party use or redistribution
- Source: Paul O'Rear (Program manager MS help team)

Introduction

SANDCASTLE

The Problem

- .NET introduced XML-based code comments
  - See MSDN for details
  - We will go into more details later
- C# compiler can extract XML-based code documentation
  - `csc /doc:<targetFile>.xml`
  - See MSDN for details



PerformanceTestCommon* | TriggerWork.aspx.cs | TriggerWork.aspx | CloudStorageConnection.cs | WorkerRole.cs

Application
Build*
Build Events
Debug
Resources

Configuration: Active (Debug) Platform: Active (Any CPU)

Output

Output path: bin\Debug\ Browse...

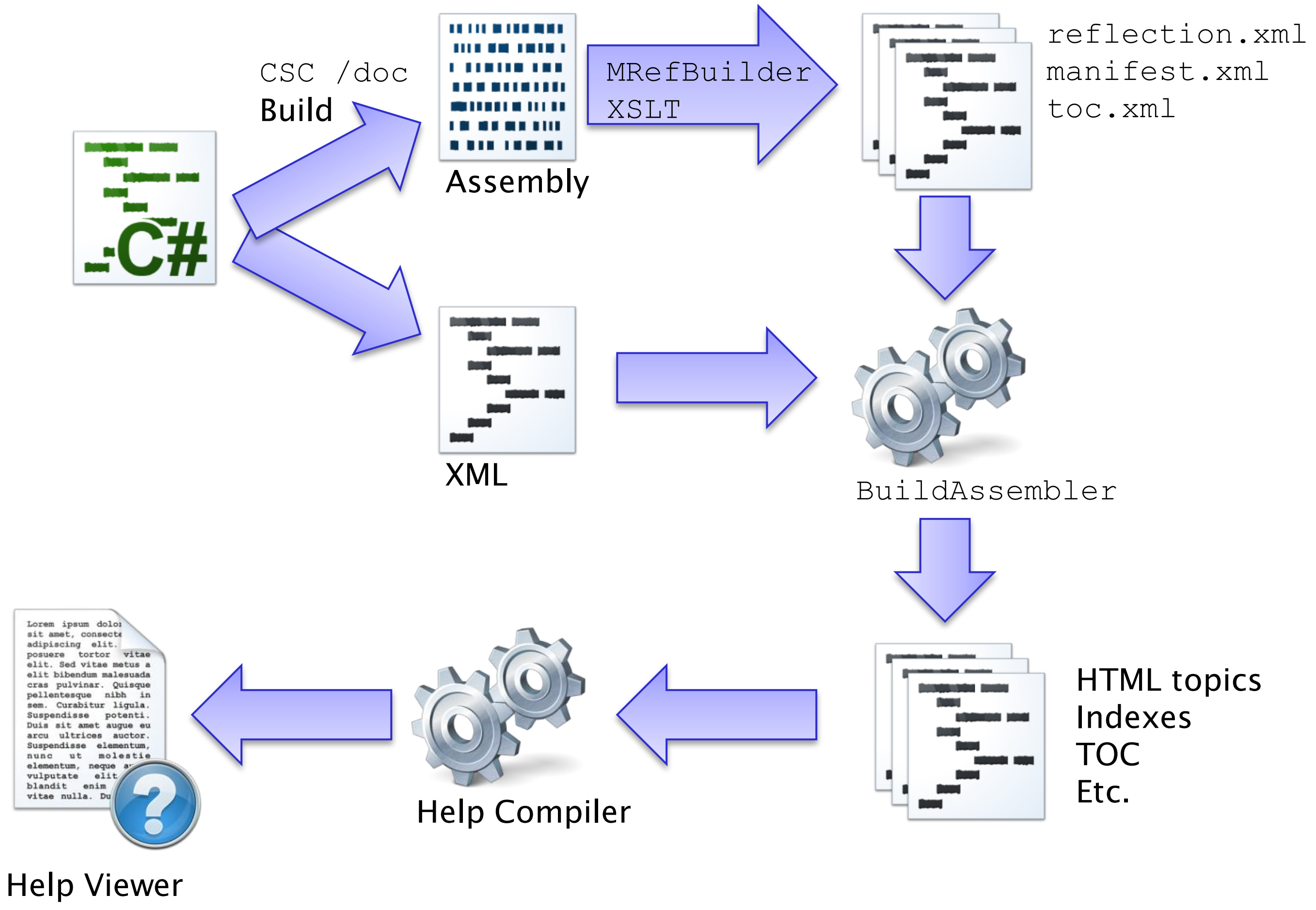
[✓] XML documentation file: bin\Debug\PerformanceTestCommon.XML

- What to do with this XML documentation??

The Idea

- Take XML documentation,
- extract type and member information from assemblies into XML-files,
- apply a set of XSL templates
- and finally build help files from that.

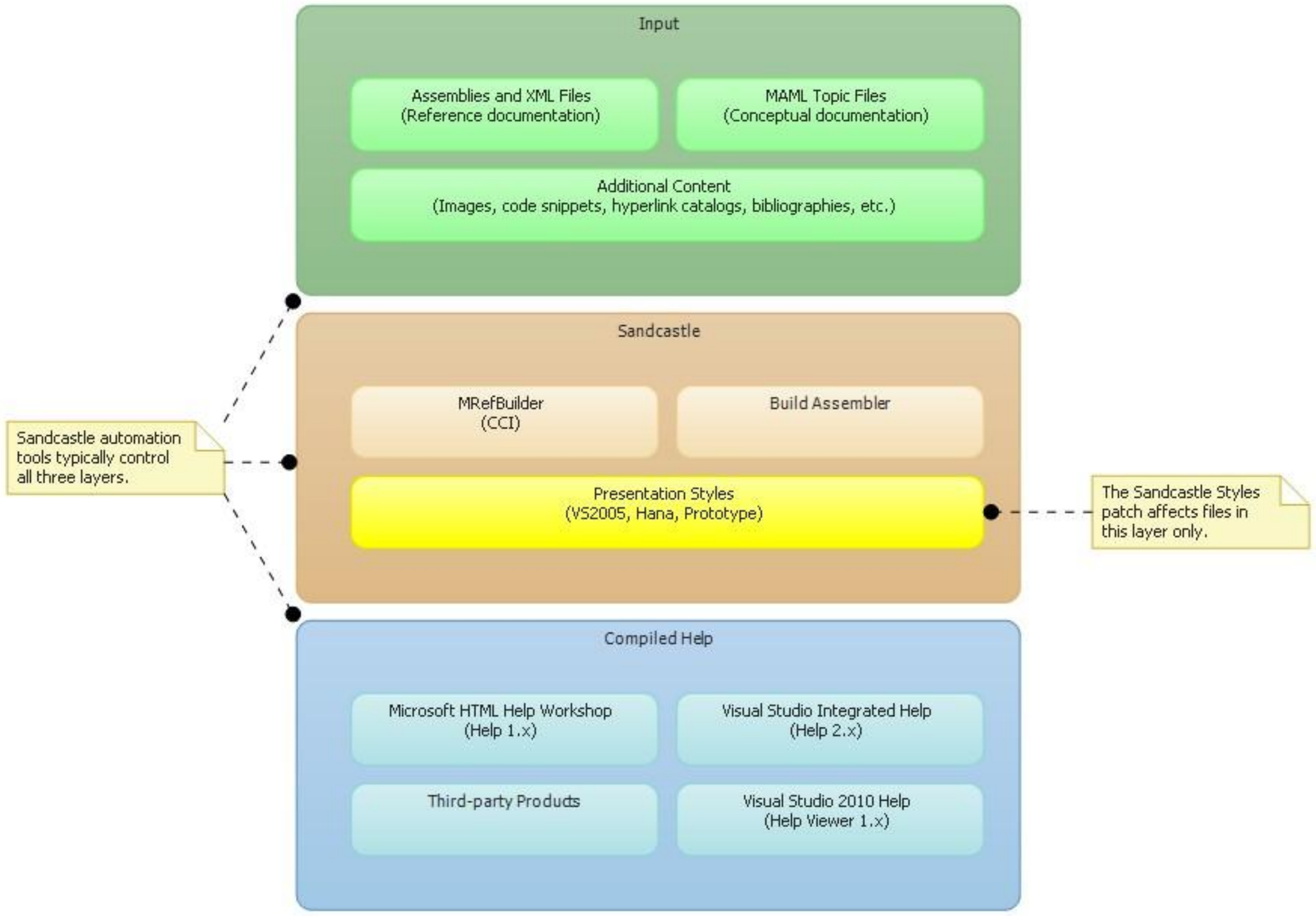
Sandcastle Process (CHM File)



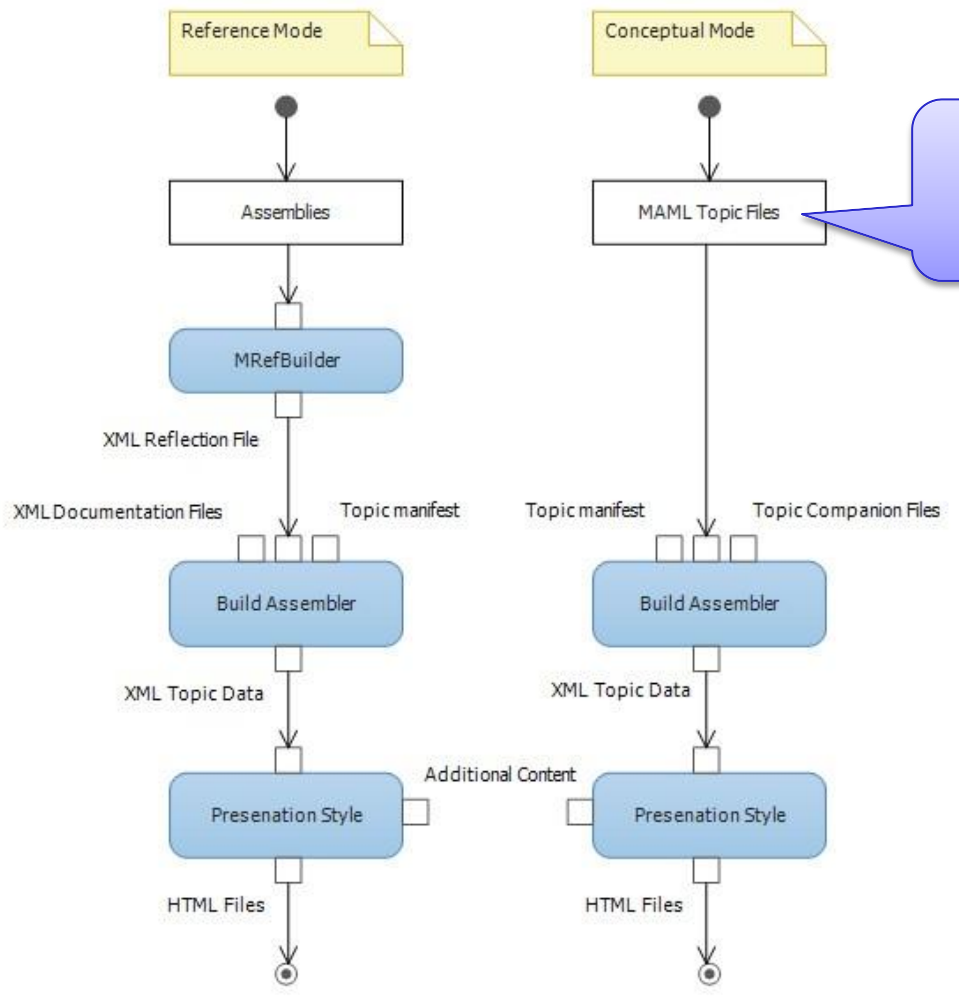
MRefBuilder

- Analyze the physical properties of the assemblies
- Generates XML reflection file
  - Contains all of the information required to document the physical properties of the input assemblies
- XML reflection file is mapped to XML documentation file from C# compiler

Role of Sandcastle Styles


Input
Assemblies and XML Files (Reference documentation)
MAML Topic Files (Conceptual documentation)
Additional Content (Images, code snippets, hyperlink catalogs, bibliographies, etc.)
Sandcastle
MRefBuilder (CCI)
Build Assembler
Presentation Styles (VS2005, Hana, Prototype)
Compiled Help
Microsoft HTML Help Workshop (Help 1.x)
Visual Studio Integrated Help (Help 2.x)
Third-party Products
Visual Studio 2010 Help (Help Viewer 1.x)
Sandcastle automation tools typically control all three layers.
The Sandcastle Styles patch affects files in this layer only.

Conceptual Mode



Microsoft Assistance Markup Language

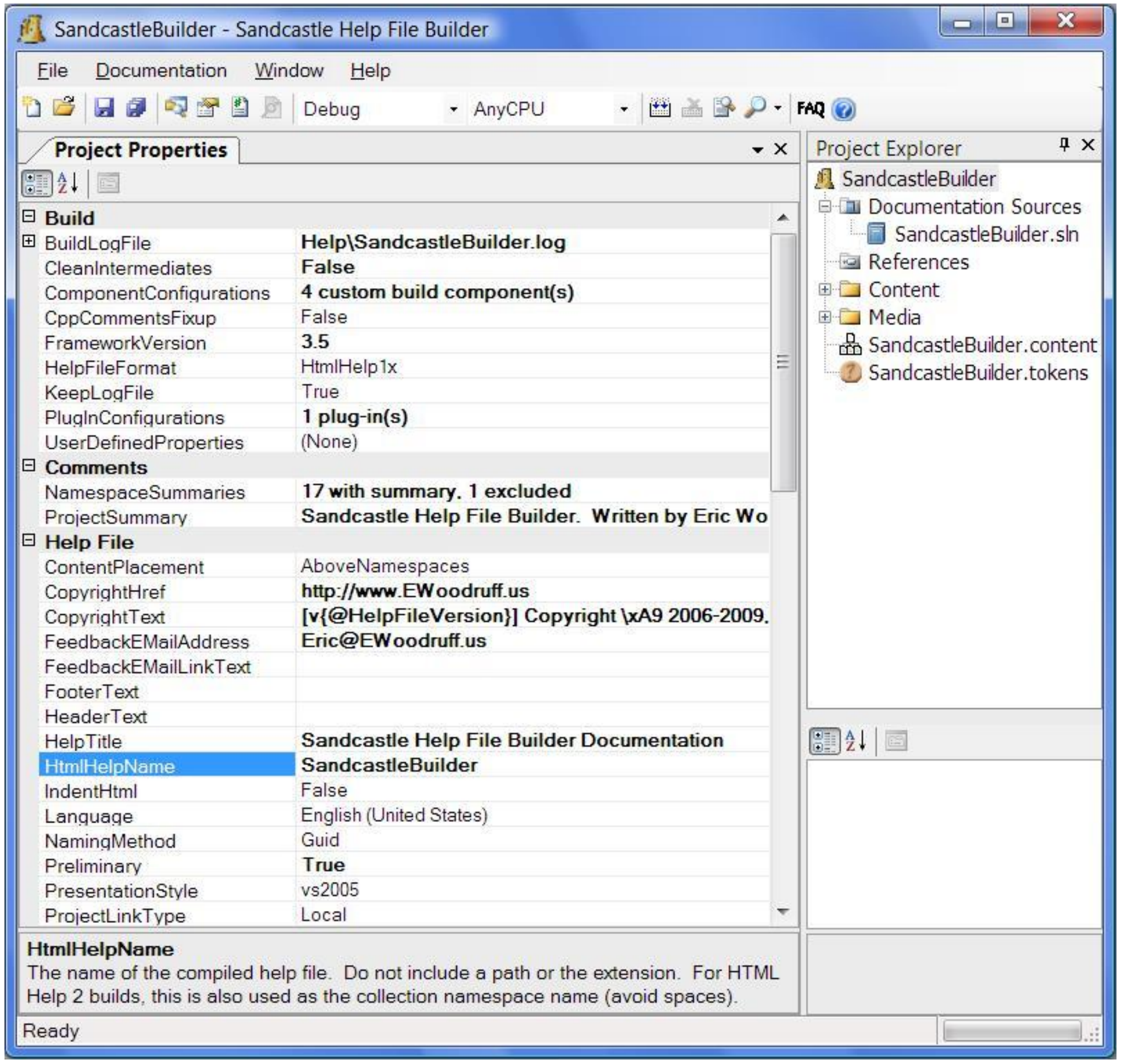
The Raw Sandcastle Process

HANDS-ON LAB 1 (10 MINUTES)

Getting rid of config files and the command line

SANDCASTLE HELPFILE BUILDER

Sandcastle Helpfile Builder



SandcastleBuilder - Sandcastle Help File Builder

File Documentation Window Help

Debug AnyCPU FAQ

Project Properties

Build	
BuildLogFile	**Help\SandcastleBuilder.log**
CleanIntermediates	**False**
ComponentConfigurations	**4 custom build component(s)**
CppCommentsFixup	False
FrameworkVersion	**3.5**
HelpFileFormat	HtmlHelp1x
KeepLogFile	True
PlugInConfigurations	**1 plug-in(s)**
UserDefinedProperties	(None)
Comments	
NamespaceSummaries	**17 with summary, 1 excluded**
ProjectSummary	**Sandcastle Help File Builder. Written by Eric Wo**
Help File	
ContentPlacement	AboveNamespaces
CopyrightHref	**http://www.EWoodruff.us**
CopyrightText	**[v{@HelpFileVersion}] Copyright \xA9 2006-2009,**
FeedbackEMailAddress	**Eric@EWoodruff.us**
FeedbackEMailLinkText	
FooterText	
HeaderText	
HelpTitle	**Sandcastle Help File Builder Documentation**
HtmlHelpName	**SandcastleBuilder**
IndentHtml	False
Language	English (United States)
NamingMethod	Guid
Preliminary	**True**
PresentationStyle	vs2005
ProjectLinkType	Local

HtmlHelpName
The name of the compiled help file. Do not include a path or the extension. For HTML Help 2 builds, this is also used as the collection namespace name (avoid spaces).

Project Explorer

- SandcastleBuilder
  - Documentation Sources
    - SandcastleBuilder.sln
  - References
  - Content
  - Media
  - SandcastleBuilder.content
  - SandcastleBuilder.tokens

Ready

SHFB Tips & Tricks

- You can add multiple documentation sources at a time
  - Use e.g. solution or wildcards
  - See SHFB Online Help for details
- Central bibliography files
  - Generate a central bibliography XML file (see SHFB Online Help)
  - Use `<cite>` element to reference bibliography
- Use token files to add a central repository of tokens
  - Enhances maintainability of your documentation

Building a simple help file using Sandcastle Help File Builder

HANDS-ON LAB 2
(10 MINUTES)

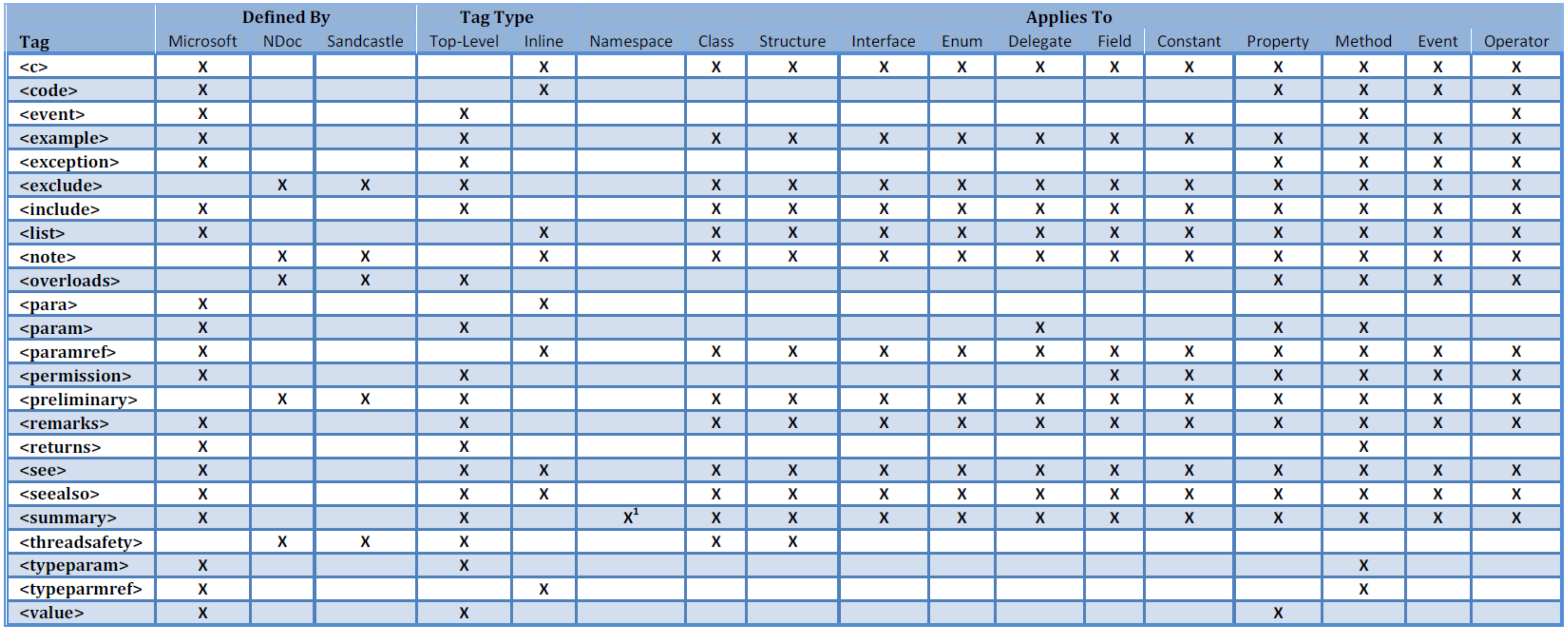
Documenting C# using XML

C# CODE DOCUMENTATION

General Tips

- All XML/HTML within an XML Documentation Comment must be well formed
- Use `&lt;` and `&gt;` for text in angle brackets
- You can use HTML markup to add additional formatting to your XML comments if needed
  - Avoid if possible

XML Documentation Comments Matrix



	Defined By			Tag Type		Applies To											
Tag	Microsoft	NDoc	Sandcastle	Top-Level	Inline	Namespace	Class	Structure	Interface	Enum	Delegate	Field	Constant	Property	Method	Event	Operator
<c>	X				X		X	X	X	X	X	X	X	X	X	X	X
<code>	X				X									X	X	X	X
<event>	X			X											X		X
<example>	X			X			X	X	X	X	X	X	X	X	X	X	X
<exception>	X			X										X	X	X	X
<exclude>		X	X	X			X	X	X	X	X	X	X	X	X	X	X
<include>	X			X			X	X	X	X	X	X	X	X	X	X	X
<list>	X				X		X	X	X	X	X	X	X	X	X	X	X
<note>		X	X		X		X	X	X	X	X	X	X	X	X	X	X
<overloads>		X	X	X										X	X	X	X
<para>	X				X												
<param>	X			X							X			X	X		
<paramref>	X				X		X	X	X	X	X	X	X	X	X	X	X
<permission>	X			X								X	X	X	X	X	X
<preliminary>		X	X	X			X	X	X	X	X	X	X	X	X	X	X
<remarks>	X			X			X	X	X	X	X	X	X	X	X	X	X
<returns>	X			X											X		
<see>	X			X	X		X	X	X	X	X	X	X	X	X	X	X
<seealso>	X			X	X		X	X	X	X	X	X	X	X	X	X	X
<summary>	X			X		X[1]	X	X	X	X	X	X	X	X	X	X	X
<threadsafety>		X	X	X			X	X									
<typeparam>	X			X											X		
<typeparmref>	X				X										X		
<value>	X			X										X			

1. NDoc and Sandcastle Help File Builder support adding namespace comments using the GUI

Source: XML Documentation Comments Guide

<summary>

- Describe a type or a type member
- Important because used in
  - IntelliSense and
  - Object Browser
- Important StyleCop rules
  - SA1642: Constructor
    - Non-private, non-static:
      Initializes a new instance of the {class name} class.
    - Static:
      Initializes static members of the {class name} class.
    - Private:
      Prevents a default instance of the {class name} class from being created.
  - SA1623: Property
    - Has to start with *Gets or sets*, *Gets* or *Sets* according to read/write status
    - For boolean properties has to start with *Gets or sets a value indicating whether*

<summary> (continued)

- Recommendation for partial classes (SA1619)
    - Provide the official SDK documentation for the class on the main part of the partial class.
        - Use the standard `<summary>` tag.
    - All other parts of the partial class should omit the `<summary>` tag completely, and replace it with a `<content>` tag.

<remarks>

- Adds information about a type, supplementing the information specified with `<summary>`
- Important because displayed in Object Browser
- Sample:

```
/// <summary>
/// You may have some primary information about this class.
/// </summary>
/// <remarks>
/// You may have some additional information about this class.
/// </remarks>
public class TestClass
{ […] }
```

<value>

- Describes the value that a property represents, supplementing the information specified with `<summary>`
- Sample:

```
/// <summary>Gets or sets a value indicating whether this instance
     is dirty.</summary>
/// <value><c>true</c> if this instance is dirty>; otherwise,
     <c>false</c></value>
public bool Dirty
{ get {…} set {…} }
```

Important Inline Elements

- `<c>`
  - Indicate that text within a description should be marked as code
  - Sample:
    ```
    /// <summary><c>DoWork</c> is a method in the
    <c>TestClass</c> class.</summary>
    ```
- `<see>`
  - Reference to a member or field
  - More about code references later
  - Sample:
    ```
    This sample shows how to call the <see cref="GetZero"/>
    method.
    ```
- `<para>`
  - Used to define a paragraph
  - Sample:
    ```
    <para>Paragraph 1</para>
    <para>Paragraph 2</para>
    ```

Important Inline Elements (continued)

- `<list>`
  - Defines a bullet list, numbered list or a table
  - Sample (bullet list):

```
/// <summary>Here is an example of a bulleted list:
/// <list type="bullet">
///   <item><description>Item 1.</description></item>
///   <item><description>Item 2.</description></item>
/// </list>
/// </summary>
```

  - Sample (table):

```
/// <summary>Here is an example of a table:
/// <list type="table">
///   <listheader>
///     <term>Col1</term>
///     <description>Column 1</description>
///   </listheader>
///   <item>
///     <term>Col1</term>
///     <description>Column 1</description>
///   </item>
/// </list>
/// </summary>
```

Important Inline Elements (continued)

- <paramref> and <typeparamref>
  - Gives you a way to indicate that a word in the code comments refers to a (type) parameter
  - Sample paramref:

```
/// <summary>DoWork is a method in the TestClass class.
/// The <paramref name="Int1"/> parameter takes a number.
/// </summary>
public static void DoWork(int Int1)
{ […] }
```

  - Sample typeparamref:

```
/// <summary>
/// Creates a new array of arbitrary type <typeparamref name="T"/>
/// </summary>
/// <typeparam name="T">The element type of the array</typeparam>
public static T[] mkArray<T>(int n)
{ […] }
```

<example>

- Specifies an example of how to use a method or other library member
- Use <code> to specify sample code
- Sample:

```
/// <summary>
/// The GetZero method.
/// </summary>
/// <example>
/// Sample of how to call the <see cref="GetZero"/> method.
/// <code>
/// class TestClass
/// {
///     static int Main()
///     {
///         return GetZero();
///     }
/// }
/// </code>
/// </example>
```

<exception>

- Specifies which exceptions can be thrown
- Sample:

```
/// <exception cref="System.Exception">Thrown when...</exception>
```

<param>

- Describes one of the parameters for the method
- Sample:

```
/// <param name="Int1">Used to indicate status.</param>
/// <param name="Float1">Used to specify context.</param>
public static void DoWork(int Int1, float Float1)
{ […] }
```

<typeparam>

- Used in the comment for a generic type or method declaration to describe a type parameter
- Sample:

```
/// <summary>
/// Creates a new array of arbitrary type <typeparamref name="T"/>
/// </summary>
/// <typeparam name="T">The element type of the array</typeparam>
public static T[] mkArray<T>(int n)
{ […] }
```

<seealso>

- Specifies the text that you might want to appear in a *See Also* section
- More about code references later
- Sample:

```
/// <summary>DoWork is a method in the TestClass class.
/// <para>Here's how you could make a second paragraph in a
description. <see cref="System.Console.WriteLine(System.String)"/>
for information about output statements.</para>
/// <seealso cref="TestClass.Main"/>
/// </summary>
```

<include>

- Refers to comments in another file that describe the types and members in your source code
- Uses the XML XPath syntax
- Problematic with StyleCop
- Sample:

```
/// <include file='xml_include_tag.doc'
       path='MyDocs/MyMembers[@name="test"]/*' />
class Test { […] }

/// <include file='xml_include_tag.doc'
       path='MyDocs/MyMembers[@name="test2"]/*' />
class Test2 { […] }
```

Code References (`cref`)

- Can reference a type, method, or property
- Use {} to reference generics
- Sample:

```
<see cref="GetGenericValue"/>
<see cref="GenericClass{T}"/>
<see cref="List{T}.RemoveAll(Predicate{T})"/>
```

Using C# code documentation

HANDS-ON LAB 3 (15 MINUTES)

Documenting C# using XML (the SHFB way)

C# CODE DOCUMENTATION ADVANCED TOPICS

External Code Samples

- Sandcastle supports enhanced code block
- Syntax:

```
<code [source="SourceFile"
  region="Region Name"
  lang="langId"
  numberLines="true|false"
  outlining="true|false"
  tabSize="###"
  title="Code Block Title"]>content</code>
```

- For details see Code Block Component in SHFB online help

API Filter

- Exclude certain parts of the API from the generated documentation
- Not defined by Microsoft, extension in *NDoc* and *SHFB*
- Sample:

```
/// <summary>
/// Gets the size of the file.
/// </summary>
/// <value>Size of the file.</value>
/// <exclude />
public int Size { get; private set; }
```

<overloads>

- Provides documentation that applies to all the overloads of a member.
- Not defined by Microsoft, extension in *NDoc* and *SHFB*
- Sample:

```
/// <summary>Decrements the number by 1.</summary>
/// <overloads>This method has two overloads.</overloads>
public void Dec()
{}

/// <summary>Decrements the number by amount.</summary>
/// <param name="amount">The amount to decrement it
by.</param>
public void Dec(int amount)
{}
```

<see>

- NDoc and Sandcastle extend `<see>` element
- Improve readability with keyword expansion
- `<see langword=„…“/>`
  - `null`
    → `“null reference (Nothing in Visual Basic)”`
  - `sealed`
  - `static`
  - `abstract`
  - `virtual`

`<threadsafety>`

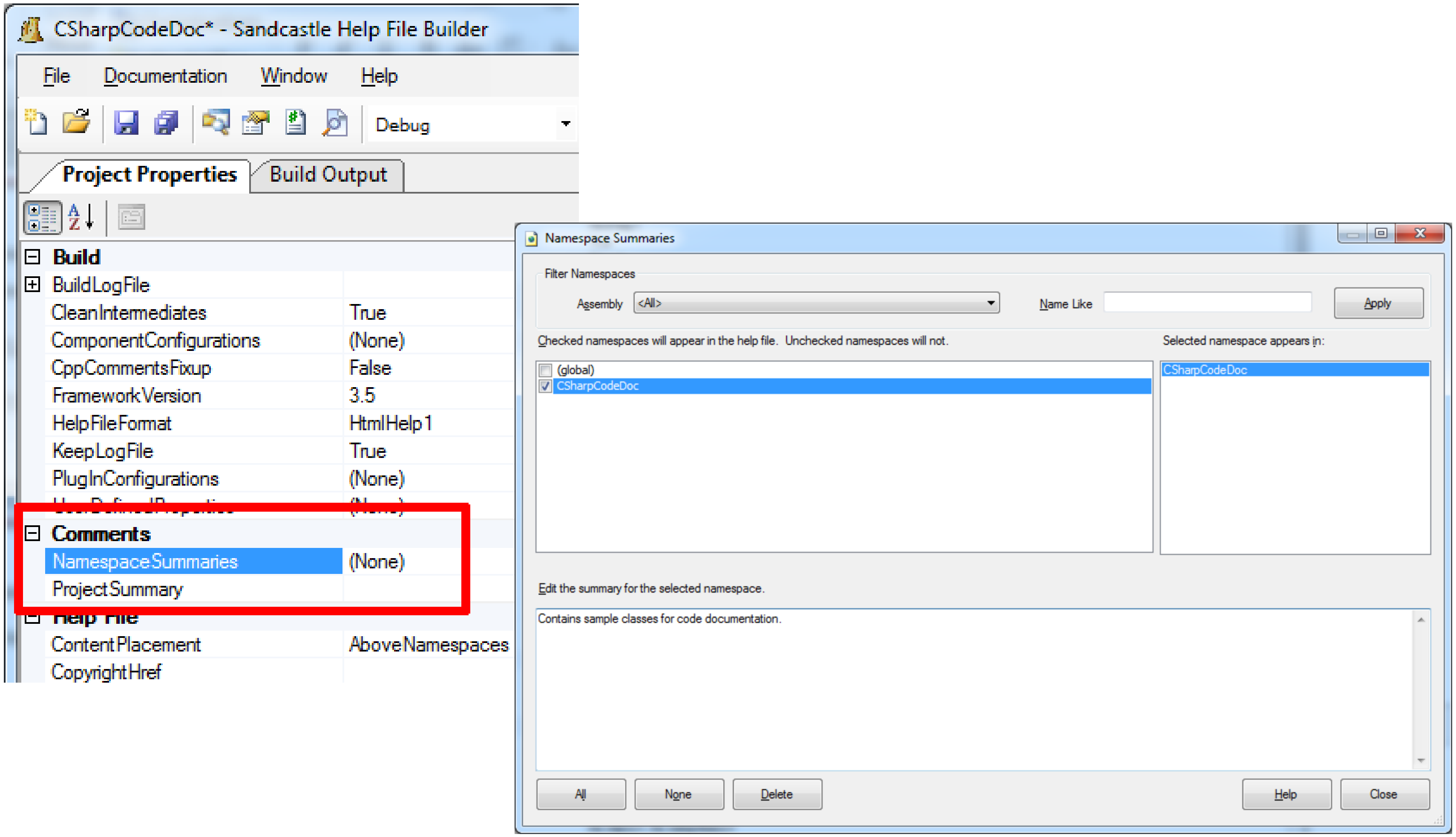
- Describes how a class or structure behaves in multi-threaded scenarios.
- Not defined by Microsoft, extension in *NDoc* and *SHFB*
- Syntax:

```
<threadsafety
  static="true|false"
  instance="true|false"/>
```

Namespace And Project Documentation

- Not defined by Microsoft, extension in SHFB
- Two options:
  - In project properties (*NamespaceSummaries* and *ProjectSummaries*)
  - Using a `NamespaceDoc` class

Namespace And Project Documentation (continued)



Build	
BuildLogFile	
CleanIntermediates	True
ComponentConfigurations	(None)
CppCommentsFixup	False
FrameworkVersion	3.5
HelpFileFormat	HtmlHelp1
KeepLogFile	True
PlugInConfigurations	(None)
UserDefinedProperties	(None)
Comments	
NamespaceSummaries	(None)
ProjectSummary	
Help File	
ContentPlacement	AboveNamespaces
CopyrightHref	

Checked namespaces will appear in the help file. Unchecked namespaces will not.

- [ ] (global)
- [x] CSharpCodeDoc

Selected namespace appears in: CSharpCodeDoc

Edit the summary for the selected namespace.

Contains sample classes for code documentation.

Namespace And Project Documentation (continued)

```
namespace CSharpCodeDoc
{
    using System.Runtime.CompilerServices;

    /// <summary>
    /// Contains sample classes for code documentation.
    /// </summary>
    [CompilerGenerated]
    internal class NamespaceDoc
    {
    }
}
```

Using C# code documentation

HANDS-ON LAB 4 (10 MINUTES)

Enhancing your documentation with conceptual content

CONCEPTUAL DOCUMENTATION

Introduction

- Add non-reference content to the help file
- Appears in the table of contents
- Written in *Microsoft Assistance Markup Language* (MAML)
  - Layout- and style-agnostic
  - XSL Transformations are used to generate the target format
- Currently no WYSIWYG editor in SHFB available
  - Use built-in editor or your favorite XML editor (VS, notepad++, etc.)
  - Use SHFB's "Debug" feature (F5) to preview result

Introduction (continued)

- Conceptual content file types
  - Content layout files
    - Defines TOC
    - Edit with content layout file editor
  - Topic files
    - Identified using a GUID
  - Image files
  - Token files
  - Code snippets files
- Tip: You can create your own template files (see SHFB Online Help)

Topic File Structure

```
<!-- The topic element contains the unique ID and revision number -->
<topic id="303c996a-2911-4c08-b492-6496c82b3edb" revisionNumber="1">

  <!-- This element name will change based on the document type -->
  <developerConceptualDocument
    xmlns="http://ddue.schemas.microsoft.com/authoring/2003/5"
    xmlns:xlink="http://www.w3.org/1999/xlink">


  <!-- The content goes here -->


  </developerConceptualDocument>

</topic>
```

Document Types

- Conceptual
- Error Message
- Glossary
- How-To
- Orientation
- Reference
- Reference With Syntax
- Reference Without Syntax
- Sample
- SDK Technology Architecture
- SDK Technology Code Directory
- SDK Technology Orientation
- SDK Technology Scenarios
- SDK Technology Summary
- Troubleshooting
- User Interface Reference
- Walkthrough
- Whitepaper
- XML Reference

Typical Conceptual MAML File

```
<?xml version="1.0" encoding="utf-8"?>
<topic id="00000000-0000-0000-0000-000000000000" revisionNumber="1">
  <developerConceptualDocument xmlns="…" xmlns:xlink="…">
    <summary>
      <para>Optional summary abstract</para>
    </summary>

    <introduction>
      <para>Required introduction</para>
      <autoOutline />
    </introduction>

    <section address="Section1">
      <title>Optional section title</title>
      <content>
        <autoOutline />
        <para>Add one or more sections with content</para>
      </content>
      <!-- optional sub sections -->
    </section>

    <relatedTopics>
      <link xlink:href="Other Topic's ID">Link inner text</link>
    </relatedTopics>
  </developerConceptualDocument>
</topic>
```

BASTA! ON TOUR

Important Block Elements

- `alert`
    - Creates a note-like section to draw attention to some important information
    - General notes, cautionary, security, language-specific notes
    - Sample:
      ```
      <alert class="note">
        <para>This is a note</para>
      </alert>
      ```
- `code`
    - Like `<code>` element in C# code documentation
    - Tip: Use `<![CDATA[…]]>` to avoid encoding of sample code
- `codeReference`
    - Inserts a code snippet from an external code snippets file
    - Sample:
      ```
      <codeReference>ClassDefinition#Define</codeReference>
      ```

Typical Snippets File

```
<?xml version="1.0" encoding="utf-8" ?>
<examples>
  <item id="ClassDefinition#Define">
    <sampleCode language="CSharp">
      public class CSharpClass()
      {
          // Members go here
      }
    </sampleCode>
    <sampleCode language="VisualBasic">
      Public Class VBClass
          ' Members go here
      End Class
    </sampleCode>
  </item>

  […]

  </item>
</examples>
```

Important Block Elements (continued)

- `definitionTable, definedTerm, definition`
  - Defines a list of terms and their definitions (two-column table without a header or footer)
  - Sample:
    ```
    <definitionTable>
      <definedTerm>Term 1</definedTerm>
      <definition>Definition 1</definition>
      …
    </definitionTable>
    ```
- `list, listItem`
  - Bullet, ordered or nobullet
  - Sample:
    ```
    <list class="bullet">
      <listItem>Item 1</listItem>
      <listItem>Item 2</listItem>
      …
    </list>
    ```

Important Block Elements (continued)

- `relatedTopics`
  - List of links to other topics that may be of interest to the reader
  - Link types (details see later)
    - `codeEntityReference`
    - `externalLink`
    - `link`
  - Link grouping
    - Secify target group using optional `topicType_id` attribute to elements
    - Group GUID values see MAML guide
  - Sample:

```
<relatedTopics>
  <link topicType_id="1FE70836-AA7D-4515-B54B-E10C4B516E50"
    xlink:href="b32a73b8-fc26-4c98-912c-d595fc1a17c2" />
  <externalLink>
    <linkText>Sandcastle Styles</linkText>
    <linkUri>http://www.codeplex.com/SandcastleStyles</linkUri>
    <linkTarget>_blank</linkTarget>
  </externalLink>
  <codeEntityReference qualifyHint="true">
    T:System.IO.FileStream</codeEntityReference>
</relatedTopics>
```

Important Block Elements (continued)

- `table, tableHeader, row, entry`
  - Arrange data in a table format with rows and columns
  - Sample:

```
<table>
  <title>A Simple Table with Title and Headers</title>
  <tableHeader>
      <row>
        <entry>Header 1</entry>
        <entry>Header 2</entry>
        <entry>Header 3</entry>
      </row>
  </tableHeader>
  <row>
        <entry>Row 1, Cell 1</entry>
        <entry>Row 1, Cell 2</entry>
        <entry>Row 1, Cell 3</entry>
  </row>
</table>
```

Important Inline Elements

- For a complete list see MAML Guide
- `codeInline`
  - String of code or a single keyword
- `command`
  - Name of an executable or other software application than can be run
- `literal`
  - Literal value
- `ui`
  - Describes a user interface element
- `token`
  - References a token in the token file

Links To Media

- Insert an image within the text of a conceptual topic
- Tip: Use `medialLinkInline` (details see MAML Guide) to display an image inline with text.
- Sample:

```
<!-- No caption, default image placement -->
<mediaLink>
<image xlink:href="6be7079d-a9d8-4189-9021-0f72d1642beb"/>
</mediaLink>

<!-- Caption before, centered image -->
<mediaLink>
<caption>Caption Before</caption>
<image placement="center" xlink:href="6be7079d-a9d8-4189-9021-0f72d1642beb"/>
</mediaLink>

<!-- Caption after with lead-in text, far image alignment -->
<mediaLink>
<caption placement="after" lead="Figure 1">
  Caption after with lead-in</caption>
<image placement="far" xlink:href="6be7079d-a9d8-4189-9021-0f72d1642beb"/>
</mediaLink>
```

Links

- `link`
  - Link to another conceptual topic using its ID value
  - Links to elements within the same page that have an `address` attribute (e.g. `section`)
  - Sample:

```
<link xlink:href="cfd9dabf-22f3-4742-8b54-d84404610db1" />
<link xlink:href="cfd9dabf-22f3-4742-8b54-d84404610db1">
companion file</link>
<link xlink:href="#intro">Back to
Intro</link>
<link xlink:href="dc4fcc96-283e-4202-9ecc-08a65e0c9313#BuildComps" />
```

Links (continued)

- `externalLink`
  - Link to an external URL
  - Attributes
    - `linkText`
    - `linkAlternateText` (for mouse hovering over it)
    - `linkUri`
    - `linkTarget` (default: `_blank`)
  - Sample:

```
<externalLink>
    <linkText>Sandcastle Styles</linkText>
    <linkAlternateText>
      Visit Sandcastle Styles</linkAlternateText>
    <linkUri>http://www.codeplex.com/SandcastleStyles</linkUri>
</externalLink>
```

Links (continued)

- `codeEntityReference`
  - Reference to a code entity such as a type, method, property, event, field, etc.
  - Can be a reference to a member in one of your classes or one in the .NET Framework.
  - Sample:

```
<codeEntityReference qualifyHint="true">
  T:System.IO.FileStream</codeEntityReference>
<codeEntityReference qualifyHint="true" autoUpgrade="true">
  M:System.IO.FileStream.#ctor(
    System.String,System.IO.FileMode)
</codeEntityReference>
<codeEntityReference qualifyHint="true" autoUpgrade="true">
  M:System.IO.FileStream.Write(
    System.Byte[],System.Int32,System.Int32)
</codeEntityReference>
<codeEntityReference qualifyHint="false">
  P:System.IO.FileStream.Length
</codeEntityReference>
<codeEntityReference qualifyHint="false" autoUpgrade="true">
  M:System.IO.FileStream.Flush
</codeEntityReference>
```

Conceptual documentation

HANDS-ON LAB 5 (15 MINUTES)

Read more about help, find the right tools

RESOURCES

Resources About Help

- Collection of help links by The Helpware Group
- Windows Help on MSDN
  - Covers MS Help 1.4 and Assisted Platform 1.0 Client
- Sandcastle Blog on MSDN
- Wikipedia articles on Online help and MS Compiled HTML Help
- The Story of Help in Visual Studio 2010
- MSHelpWiki.com

Resources About Help (continued)

- XML Documentation Comments on MSDN
- Sandcastle Help File Builder installation instructions
- Online Help on Sandcastle Help File Builder

Tool Reference

- Sandcastle
  - Documentation Compiler for Managed Class Libraries
- MshcMigrate.exe
  - Convert older projects over to MS Help Viewer
- H3Viewer.exe
  - Alternative VS 2010 help viewer
- Sandcastle Styles Project
  - Style Patches and resources about MAML

Tool Reference (continued)

- GhostDoc
  - Generates documentation based on naming conventions
- StyleCop
  - Analyzes C# source code to enforce a set of style and consistency rules
- Sandcastle Help File Builder
  - Provides graphical and command line based tools to build a help file in an automated fashion

Tool Reference (continued)

- Sandcastle MAML Guide
  - Help about Microsoft Assistance Markup Language (MAML)