CHAPTER 58

Advanced Analysis Tools

Writing code is just one part of the developer life. There are so many other aspects to consider in producing highquality applications. For example, if you produce reusable class libraries, you must ensure that your code is compliant with Common Language Specification, and this requires deep analysis. Another key aspect is performance. If you produce a great application with the most desired functionalities but with poor performance, perhaps your customer will prefer a faster and less-consuming application even if it has a minor number of features. Continuing from its predecessors, Visual Studio 2010 offers a number of integrated tools for analyzing code and performance to improve your applications' quality. It is worth mentioning that most of the previously existing tools have been significatly enhanced due to the WPF-based architecture of the IDE. In this chapter you learn how to take advantage of Visual Studio 2010's integrated analysis tools for writing better applications.

VISUAL STUDIO SUPPORTED EDITIONS

Analysis tools are available only in some Visual Studio 2010 editions. To complete tasks explained in this chapter, you need at least Visual Studio 2010 Premium or the Visual Studio 2010 Ultimate that is required for IntelliTrace.

IN THIS CHAPTER

- Introducing Analysis Tools
- Performing Code Analysis
- Calculating Code Metrics
- Profiling Applications
- IntelliTrace, the Historical Debugger
- Generating Dependency Graphs

Introducing Analysis Tools

Visual Studio 2010 offers the following analysis tools, which can help you produce highquality applications:

- ▶ Code Analysis, which analyzes code for compliance with Microsoft coding rules
- Code Metrics, which returns statistic results and analyzes code maintainability according to specific indexes;
- Profiler, which analyzes application performance and suggests solutions for solving problems
- ► IntelliTrace, formerly known as Historical Debugger, which allows keeping track of every single event and exceptions happening during the entire application lifetime

In this chapter you learn to take advantage of the listed tools for improving quality in your code.

Performing Code Analysis

Earlier in this book you learned about Common Language Specification, learning that it is a set of common rules and guidelines about writing code that can be shared across different .NET languages favoring interoperability and that is considered well designed for the .NET Framework. In some cases it can be hard ensuring that all your code is CLS-compliant, especially when you have large projects with tons of lines of code. To help you write better and CLS-compliant code, Microsoft produced a code analysis tool named FxCop that analyzes compiled assemblies for non-CLS-compliant code and that reports suggestions and solutions for solving errors. Although free, FxCop is an external tool and is bound to developers using Visual Studio editions such as Express or Professional.

DOWNLOADING FXCOP

If you do not have Visual Studio Ultimate but want to try the code analysis features, you can check out FxCop, which is available on the MSDN Code Gallery at http://code. msdn.microsoft.com/codeanalysis. Generally all concepts described in this section are available in FxCop, too.

Fortunately, the Ultimate edition offers an integrated version of the code analysis tool that you can invoke on your project or solution directly within the IDE; moreover you can customize the code analysis process by setting specific rules. Before examining available rules, it is a good idea to create a simple project for demonstrating how code analysis works, so create a new class library project in Visual Basic 2010. When the code editor is ready, rename the Class1 file to **HelperClass** and write the code shown in Listing 58.1, which attempts defining a CLS-compliant class but that makes several violations to the Microsoft rules, for which I give an explanation for solving later in this section.

```
LISTING 58.1 Writing a Simple non-CLS-Compliant Class Library
```

```
<CLSCompliant(True)>
Public Class HelperClass
Private CustomField As String
Public Property customResult As String
'Just a demo function
Public Function doubleSum(ByVal FirstValue As Double,
ByVal SecondValue As Double) As Double
Return FirstValue + SecondValue * 2
End Function
End Class
```

For rules, it is worth mentioning that Microsoft divides guidelines in writing code into the rules summarized in Table 58.1.

Rule name	Description
Microsoft.Design	Determines if assemblies contain well-designed objects or if the assembly definition is CLS-compliant
Microsoft.Globalization	Determines if globalization techniques are well implemented
Microsoft.Interoperability	Determines if the code makes correct usage of COM interoperability
Microsoft.Maintainability	Checks for code maintainability according to Microsoft rules
Microsoft.Mobility	Checks for timer and processes correct implementation
Microsoft.Naming	Determines if all identifiers match the CLS rules (such as public/private members, method parameters and so on)
Microsoft.Performance	Checks for unused or inappropriate code for compile time and runtime performances from the CLR perspective
Microsoft.Portability	Determines if the code is portable for invoked API functions
Microsoft.Reliability	Provides rules for a better interaction with the Garbage Collector
Microsoft.Security	Provides security-related rules sending error messages if types and members are not considered secure
Microsoft.Usage	Determines if a code block correctly invokes other code

TABLE 58.1 Microsoft Code Analysis Rules

Performing code analysis does not require all the mentioned rules to be checked. You can specify only a subset of preferred rules or specify the complete set. To specify the rules sets involved in the code analysis, follow these steps:

1. Open **My Project** and click on the **Code Analysis** tab. Figure 58.1 shows how the designer looks

compile □ Enable Code Analysis on Build (defines CODE_ANALYSIS constant) Image: Suppress results from generated code Image: Suppress results from generated code Resources Rule Set Resources Microsoft All Rules Description: This rule set contains all rules. Running this rule set to get a Igning Open Ay Extensions Open Scode Analysis Path: Col+Program Files\Microsoft Visual Studio 10.0\Team Tools\Static Image: Col+Static Learn more about rule sets Learn more about rule sets	Compile Debug	Enable Code Analysis on Build (defines CODE_ANALYSIS constant)
Debug □ Enable Code Analysis on Build (defines CODE_ANALYSIS constant) References Suppress results from generated code Resources Rule Set Services Bun this rule set: Microsoft All Rules Open Description: This rule set contains all rules. Running this rule set to get a large number of warnings being reported. Use this rule set to get a large number of warnings being reported. Use this rule set to get a propriate to run for your projects. Code Analysis Path: C:\Program Files\Microsoft Visual Studio 10.0\Team Tools\Static Learn more about rule sets	Debug	Enable Code Analysis on Build (defines CODE_ANALYSIS constant)
References Resources Resources Services Settings Signing Wy Extensions Code Analysis Path: C\Program Files\Microsoft Visual Studio 10.0\Team Tools\Static ↓ Leam more about rule sets		
Resources Resources Resources Resources Resources Resources Resources Run this rule set: Microsoft All Rules Qpen Description: This rule set contains all rules. Running this rule set may result in a large number of warnings being reported. Use this rule set to get a comprehensive picture of all issues in your code. This can help you decide which of the more focused rule sets are most appropriate to run for your projects. Path: C:\Program Files\Microsoft Visual Studio 10.0\Team Tools\Static Leam more about rule sets	lererences	☑ Suppress results from generated code
Services Microsoft All Rules Open Settings Description: This rule set contains all rules. Running this rule set may result in a large number of warnings being reported. Use this rule set to get a comprehensive picture of all issues in your code. This can help you decide which of the more focused rule sets are most appropriate to run for your projects. Code Analysis Path: C:\Program Files\Microsoft Visual Studio 10.0\Team Tools\Static Learn more about rule sets	Resources	Rule Set
Settings Description: This rule set contains all rules. Running this rule set may result in a large number of warnings being reported. Use this rule set to get a comprehensive picture of all issues in your code. This can help you decide which of the more focused rule sets are most appropriate to run for your projects. Code Analysis Path: C:\Program Files\Microsoft Visual Studio 10.0\Team Tools\Static Learn more about rule sets	ervices	Microsoft All Rules
Signing This rule set contains all rules. Kunning this rule set may result in a large number of warmings being reported. Use this rule set to get a comprehensive picture of all issues in your code. This can help you decide which of the more focused rule sets are most appropriate to run for your projects. Code Analysis Path: C:\Program Files\Microsoft Visual Studio 10.0\Team Tools\Static Learn more about rule sets	Settings	Description:
My Extensions Scode Analysis Path: C:\Program Files\Microsoft Visual Studio 10.0\Team Tools\Static Learn more about rule sets	igning	large number of warnings being reported. Use this rule set to get a
Code Analysis Path: C:\Program Files\Microsoft Visual Studio 10.0\Team Tools\Static + Learn more about rule sets	My Extensions	you decide which of the more focused rule sets are most
C:\Program Files\Microsoft Visual Studio 10.0\Team Tools\Static +	Code Analysis	appropriate to run for your projects.
Learn more about rule sets		C:\Program Files\Microsoft Visual Studio 10.0\Team Tools\Static +
		Learn more about rule sets

FIGURE 58.1 The Code Analysis designer.

- 2. Expand the Run This Rule Set combo box. You get a list of available rule sets with an accurate description for each set. By default, the offered set of rules is Microsoft Minimum Recommended Rules. Replace it by selecting Microsoft All Rules that includes all sets listed in Table 58.1 and that is the most accurate. To get detailed information on each rule set, simply click Open. Figure 58.2 shows how you can browse rules available in the selected set, getting summary information for each rule, and specifying how you want to get help on violations (for example, online or offline)
- **3.** Select the **Analyze**, **Run Code Analysis** command and wait for a few seconds until the building and code analysis process is completed. When ready, Visual Studio shows a report listing all violations to coding rules encountered in the project. The report is shown in Figure 58.3.

Each violation message includes the violation ID and a description that can help you fix the error. In most cases violations are interpreted by the IDE as warnings, but in the code

Image: Second	AssemblyInfo.vb HelperClass.vb*	AllRules.ruleset		
Writes at Microsoft rules at and cannot be modified. To create your own rule sat based on this rule sat, make your changes below. Then save your changes to a new file to create there is at. Image: This is a Microsoft Rules at and cannot be modified. To create your own rule sat based on this rule sat, make your changes below. Then save your changes to a new file to create there is at. Image: This is a Microsoft Rules at and the same at the same	(+) (-) Group by: Category	• X 30 A • 5 m	Search	Q
We rule set. Image: Set Reliable for the rules of the	This is a Microsoft rule set and cann	the modified. To create your own rule set based on this rule s	t make your changes below. Then save your changes to a new file to	create
D ▲ Name Action ↓ Warning D ✓ Microsoft.Bediatation ↓ Warning D ✓ Microsoft.Bediatation ↓ Warning D Ø Microsoft.Maintahability ↓ Warning D Ø Microsoft.Performance ↓ Warning D Ø Microsoft.Usage ↓ Warning	the rule set.	a de modifical no create your omnale set based on this rate s	in intervention of the second s	create
Visual Studio can display help for these rules. Do you want to connect to the internet and download help?	ID .	Name	Action	Ţ
Visual Studio can display help for these rules. Do you want to connect to the internet and download help? Visual Studio can display help for tules. OKer out in help for rules. OKer out in help for rules.	Microsoft.Design		🗟 Warning	
 Microsoft.Interopresibility Microsoft.Macbility Microsoft.Macbility Microsoft.Macbility Microsoft.Performance Microsoft.Performanc	▷ ☑ Microsoft.Globalization		🔍 Warning	
Visual Studio can display help for these rules. Do you want to connect to the internet and download help? Visual Studio can display help for rules.	Microsoft.Interoperability		🗟 Warning	
Visual Studio can display help for these rules. Do you want to connect to the internet and download help?	Microsoft.Maintainability		🖳 Warning	
Witcrooft.Naming By Warning Witcrooft.Portability Warning CA1900 Value type fields should be portable Warning CA1901 P/Inroke declarations should be portable Warning CA1901 P/Inroke declarations should be portable Warning Witcrooft.Reliability Warning Warning Witcrooft.Reliability Warning Warning Mitcrooft.Reliability Warning Warning Witcrooft.Liage Warning Warning Visual Studio can display help for these rules. Do you want to connect to the internet and download help? Warning Visual Studio can display help for these rules. Do you want to connect to the internet and download help? Warning Warning the provides Warning Warning Visual Studio can display help for rules. Warning Warning Warning the provides Warning Warning Warning Warning thelp for ru	Microsoft.Mobility		🖳 Warning	
	Microsoft.Naming		🗟 Warning	
Image: Second	Microsoft.Performance		🖳 Warning	
Image: CA1900 Value type Fields should be portable ▲ Warning Image: CA1903 P. Finnote decisations should be portable ▲ Warning Image: CA1903 Use only API from targeted framework ▲ Warning Image: CA1903 Use only API from targeted framework ▲ Warning Image: CA1903 Use only API from targeted framework ▲ Warning Image: CA1903 Use only API from targeted framework ▲ Warning Image: CA1903 Warning Image: CA1903 Image: CA1903 Warning Image: CA1903 <td>✓ Microsoft.Portability</td> <td></td> <td>🏹 Warning</td> <td>•</td>	✓ Microsoft.Portability		🏹 Warning	•
Image: CA1901 P/Imode declarations should be portable ▲ Warning Image: CA1903 Use only API from tageted framework: ▲ Warning Image: CA1903 Warning Image: CA1903 Image: CA1903 Warning Image: CA1903 Image: CA1903 Image: CA1903 Image: CA1903 <td>V CA1900</td> <td>Value type fields should be portable</td> <td>🔺 Warning</td> <td></td>	V CA1900	Value type fields should be portable	🔺 Warning	
QA1303 Ube only API from targeted framework: QA1403 QA1403 QA1403 QA1404 QA140 QA140 QA140 QA140 QA140 QA14 QA14	V CA1901	P/Invoke declarations should be portable	Å Warning	
	V CA1903	Use only API from targeted framework	🔺 Warning	
Image: Second Secon	Microsoft.Reliability		🗟 Warning	
	Microsoft.Security		🗟 Warning	
Visual Studio can display help for these rules. Do you want to connect to the internet and download help? Always use online help for rules. Never use online help for rules. Use online help for rules one time. OK	Microsoft.Usage		🗟 Warning	
Visual Studio can display help for these rules. Always use online help for rules. Visua online help for rules. Use online help for rules one time. OK				
	Visual Studio can display help for these Always use online help for rules. Never use online help for rules. Use online help for rules one time. OK	rules. Do you want to connect to the internet and download he	fg7	

FIGURE 58.2 Browsing rule sets.

_	Description	File	Line	Column	Project
1	CA1020 : Microsoft.Design : Consider merging the types defined in 'CodeAnalysisDemo' with another namespace.				CodeAnalysisDem
2	CA2210 : Microsoft.Design : Sign 'CodeAnalysisDemo.dll' with a strong name key.				CodeAnalysisDem
3	CA1014 : Microsoft.Design : Mark 'CodeAnalysisDemo.dll' with CLSCompliant(true) because it exposes externally visible types.				CodeAnalysisDem
4	CA1824 : Microsoft.Performance : Because assembly 'CodeAnalysisDemo.dll' contains a Resi-based resource file, mark it with the NeutralResourcesLanguage attribute, specifying the language of the resources within the assembly. This could improve lookup performance the first time a resource is retrieved.				CodeAnalysisDem
5	CA1823 : Microsoft.Performance : It appears that field 'HelperClass.CustomField' is never used or is only ever assigned to. Use this field or remove it.	HelperClass.vb	5		CodeAnalysisDem
6	CA1709 : Microsoft.Naming : Correct the casing of 'custom' in member name 'HelperClass.customResult()' by changing it to 'Custom'.	HelperClass.vb	7		CodeAnalysisDem
7	CA1709 : Microsoft.Naming : In member 'HelperClass.doubleSum(Double, Double)', correct the casing of 'First' in parameter name 'FirstValue' by changing it to 'first'.	HelperClass.vb	10		CodeAnalysisDem
8	CA1709 : Microsoft:Naming : In member 'HelperClass.doubleSum(Double, Double)', correct the casing of 'Second' in parameter name 'Second'Value' by changing it to 'second'.	HelperClass.vb	10		CodeAnalysisDem
9	CA1709 : Microsoft.Naming : Correct the casing of 'double' in member name 'HelperClass.doubleSum(Double, Double)' by changing it to 'Double'.	HelperClass.vb	10		CodeAnalysisDem
10	CA1822 : Microsoft.Performance : The 'this' parameter (or 'Me' in Visual Basic) of 'HelperClass.doubleSum(Double, Double)' is never used. Mark the member as static (or Shared in Visual Basic) or use 'this'/Me' in the method body or at least one properly accessor, if appropriate.	HelperClass.vb	10		CodeAnalysisDem

FIGURE 58.3 Report from the code analysis tool.

analysis designer you can modify this behavior by setting what violations must be notified as breaking errors. Also notice how assembly-level violations do not include the line of code to be fixed, whereas code-level violations do. In this case you simply double-click the error message to be immediately redirected to the line of code to be fixed.

GETTING ERROR HELP

There are hundreds of Microsoft rules, so summarizing all of them in this book is not possible. You can get detailed information on each violation (and on how you solve the violation for each) by simply right-clicking the error message and selecting **Show Error Help** from the pop-up menu.

At this point we can begin fixing violations. It is worth mentioning that there can be situations in which violations cannot be fixed due to particular scenarios. For example, the first violation in our example (CA1020) indicates that we should merge the current type into an existing namespace, because a well-formed namespace contains at least five members. Due to our demo scenario, we can ignore this violation that is nonbreaking. The next error message (CA2210) indicates that the assembly must be signed with a strong name. I described strong names in Chapter 53, "Understanding the Global Assembly Cache," so follow those instructions to add a strong name file to the library. I named the strong name file as TestCode.pfx providing the **TestCode** password. The next violation (CA1014) requires the assembly to be marked as CLS-compliant. To accomplish this, click the **Show All Files** button in Solution Explorer, expand **My Project** and add the following line to the AssemblyInfo.vb file:

<Assembly: CLSCompliant(True)>

The next violation (CA1824) indicates that a neutral-language resource should be supplied to the assembly. Because you have the AssemblyInfo.vb file already open, write the following line:

```
<Assembly: NeutralResourcesLanguageAttribute("en-US")>
```

You could also set this property via the Assembly Information dialog from the Application tab in My Project. With this step, all assembly level violations were fixed. Now it's time to solve code-level violations. The CA1823 violation suggests that there is a field named CustomField that is never used and that, because of this, should be removed to improve performance. Now, remove the following line of code:

Private CustomField As String

The next step is to solve three CA1709 violations that are all about members naming. We need to replace the first letter of the doubleSum method with the uppercase and the first letter of both arguments with lowercase. This is translated in code as follows:

```
Public Function DoubleSum(ByVal firstValue As Double,
ByVal secondValue As Double) As Double
```

NAMING CONVENTIONS

We discussed naming conventions in Chapter 7, "Class Fundamentals," with regard to methods, method arguments, and properties, so refer to that chapter for details.

There is also another naming convention violation to be fixed on the customResult property that must be replaced with CustomResult. The last required fix is on performance (CA1822 violation). The code analysis tool determines that the DoubleSum method never invokes the class constructor; therefore, it suggests to mark the method as Shared or to invoke the constructor. In this particular situation the method can be marked as Shared like this:

```
Public Shared Function DoubleSum(ByVal firstValue As Double,
ByVal secondValue As Double) As Double
```

In this particular case we do not need an instance method. For your convenience all edits are available in Listing 58.2 (except for assembly level edits).

LISTING 58.2 Fixing Errors Reported by the Code Analysis Tool

```
<CLSCompliant(True)>
Public Class HelperClass
'Private CustomField As String
Public Property CustomResult As String
Public Shared Function Sum(ByVal firstValue As Double,
ByVal secondValue As Double) As Double
Return firstValue + secondValue
End Function
End Class
```

If you now run the code analysis tool again, you notice that only the CA1020 design rule is still reported (the one on merging types into an existing namespace that we decided not to fix due to our particular scenario). The code analysis tool is a helpful instrument, especially if you are a libraries developer. Microsoft has a Code Analysis Team with a blog where you can find interesting information: http://blogs.msdn.com/fxcop. Also remember that you can easily retrieve detailed information on violation errors and fixes directly from within Visual Studio.

Calculating Code Metrics

Code metrics is an interesting tool that analyzes a project or a solution providing results about the ease of maintainability according to specific indexes. You can invoke the tool from the Analyze menu by selecting the **Calculate Code Metrics** command or by right-clicking the project name in Solution Explorer and then selecting the same-named command. The tool calculates code metrics according to the indexes summarized in Table 58.2.

Index	Description
Maintainability index	A percentage value indicating ease of maintainability for the selected project or solution. A higher value indicates that the project is well structured and easily maintainable.
Cyclomatic complexity	A percentage value indicating complexity of loops, nested loops, and nested conditional blocks, such as nested ForNext loops, DoLoop loops, or IfEnd If nested blocks. A higher value indicates that you should consider refactoring your code to decrease loop complexity because this leads to difficult maintainability.
Depth of inheri- tance	Indicates the inheritance level for classes in the project. The result shows the report for the class with the highest inheritance level. A higher value indi- cates that it might be difficult finding problems in a complex inheritance hier- archy.
Class coupling	Calculates how many references to classes there are from method parame- ters and return values, local variables, and other implementations. A higher value indicates that code is difficult to reuse, and you should consider revisit- ing your code for better maintainability.
Lines of code	Just a statistic value. It returns the number of IL code affected by the analysis.

TABLE 58.2 Code Metrics Analyzed Indexes

To understand how it works, in Visual Studio 2010 open the **SnippetCompilerVSPackage** sample project described in the previous chapter; then run the Code Metrics tool by selecting the **Analyze**, **Calculate Code Metrics for Solution** command. After a few seconds you get the report shown in Figure 58.4.

As you can see from the report, the project has a maintainability index of 88 that is quite good. Generally values from 80 to 100 are the best range for maintainability; Visual Studio shows a green symbol if the index is good or a red one if the maintainability index is too poor. You can expand the **SnippetCompilerVsPackage** item to see how the global result is subdivided for each class and also for each class member. The global Cyclomatic Complexity index is 11, which is a small number for our kind of project. Depth of Inheritance index is 3, which is a small value, meaning that there is one or more class inheriting from another class that inherits from another one (the third one is

Filter: None	 Min: 		~ Мах		- 🔁 💽 🕻	a 137
Hierarchy	Maint	ainability I	Cyclomatic Com	Depth of Inherita	Class Coupling	Lines of Code
4 🛐 SnippetCompilerVSPackage (Debug)		88	11	3	28	25
4 {} AlessandroDelSole.SnippetCompilerVSPackage		88	11	3	28	25
▷ 🏂 GuidList		96	2	1	1	2
🔺 🔩 MyToolWindow		83	3	3	6	8
Content.Get() As Object		86	1		1	2
Content.Set(Object)		94	1		2	1
• New()		73	1		3	5
RegCmdIDList		100	1	1	0	1
SninnetCompilerVSPackagePackage		71	5	2	24	14

FIGURE 58.4 Calculating code metrics for the specified project.

System.Object); this is an absolutely acceptable value in this particular scenario. The Class Coupling index is a little too high. It is determined by the

SnippetCompilerVsPackagePackage class, meaning that this class has a lot of references to other classes. Particularly, if you expand the class you notice that the problem is the Initialize method that makes calls to a lot of objects. Obviously, a high index doesn't necessarily indicate problems. In this code example a high value is acceptable, because all invocations are required to make the Visual Studio package work, but in a reusable class library a high value needs attention and code refactoring.

EXPORTING TO EXCEL

If you need to elaborate the code metrics results, you can export the analysis report to Microsoft Excel. This can be accomplished with the Open List in Excel button on the Code Metrics Result tool window.

Profiling Applications

Performance is a fundamental aspect in development. An application with slow performance can discourage customers, even if it has the coolest user interface or features. Visual Studio offers an integrated profiling tool that has been deeply enhanced in the 2010 edition. The new profiler takes advantage of the WPF-based user interface of Visual Studio also offering better organization for the collected information layout. To understand how the profiler works, we need a test project. Now, create a new console application in Visual Basic. The best way for understanding how the profiler can improve your productivity is considering the simple example of strings concatenation both using String and StringBuilder objects. In the main module of the console application type this simple code:

```
Module Module1
```

Sub Main()
ConcatenationDemo()

```
Console.ReadLine()
End Sub
Sub ConcatenationDemo()
Dim testString As String = String.Empty
For i = 0 To 10000
testString += "I love VB 2010"
Next
End Sub
```

End Module

The preceding code simply creates a big concatenation of String objects. To analyze performance, we now need to start the profiler. Select the **Analyze**, **Launch Performance Wizard** command. This launches the step-by-step procedure for setting up the profiler running against your project. Figure 58.5 shows the first dialog in the wizard, where you can select the most appropriate profiling technique according to your needs.

Performance Wizard Page 1 of 3	? X
Specify the profiling method	
Profiling your application can help diagnose performance problems and identify the most common methods in your application. To begin, choose a profiling method from the options below.	expensive
What method of profiling would you like to use?	
CPU Sampling (recommended) Monitor CPU-bound applications with low overhead	
 Instrumentation Measure function call counts and timing 	
 .NET Memory Allocation (Sampling) Track managed memory allocation 	
<u>Concurrency</u> Detect threads waiting for other threads	
Collect resource contention data	
Visualize the behavior of a multithreaded application	
Read more about profiling methods	
< <u>P</u> revious <u>N</u> ext > <u>Finish</u>	Cancel

FIGURE 58.5 Selecting the profiling technique.

There are several available modes, all described in Table 58.3.

Mode	Description
CPU Sampling	Analyzes performances at predetermined intervals for monitoring CPU usage. This is the recommended mode for applications that use few resources; it collects less information but it consumes less system resources.
Instrumentation	Collects complete information on the application performance injecting specific testing code. It is suggested for long-running applications and processes and consumes more system resources.
.NET Memory Allocation	Analyzes memory allocation performance.
Concurrency	Analyzes how multithreaded application consume resources and their performance.

TABLE 58	.3 Av	ailable	Profiling	Modes
----------	-------	---------	-----------	-------

Leave unchanged the default CPU-sampling option and click **Next**. The second dialog allows selecting the executable to be analyzed. As you can see from Figure 58.6, you can choose one of the executables resulting from the current solution or an external executable. Select the current executable (the default option) and continue.



FIGURE 58.6 Selecting the executable to be analyzed.

The last dialog in this wizard is just a summary. Uncheck the one check box available so that the Profiler will not be launched when you click **Finish** (see Figure 58.7).



FIGURE 58.7 Completed set up of the Profiler.

You might want to leave the flag on the checkbox to automatically launch the Profiler if the default settings are okay for you, but in this particular scenario we need to make a couple of manual adjustments. When the wizard shuts down, the Performance Explorer tool window appears in the IDE. Because the sample application does not actually stress the CPU very intensively, we need to set a smaller value for the CPU Sampling intervals. To accomplish this, follow these steps:

- 1. In the Performance Explorer window, right-click the ProfilerDemo_Before item.
- 2. Select the **Properties** command from the popup menu.
- **3.** In the ProfilerDemo_Before Property Pages dialog, select the Sampling node on the left and then replace the default value in the Sampling Interval text box with 50000. This will let the profiler collect data at smaller intervals of CPU clock cycles (see Figure 58.8 for details). This small value is appropriate for small pieces of code like the current example, but you could leave unchanged the default value in real-world applications.

ProfilerDemo_Before Property Pages			? ×
General Launch Sampling Binaries Tier Interactions Instrumentation CPU Counters Windows Events Windows Events Windows Counters Advanced	Sample gvent: Clock cycles ▼ Available gerformance counters: Portable Events Platform Events	Sampling interval:	50000
	Summary Sample once every 50.000 Clock cycles		
		OK Cancel	Apply

FIGURE 58.8 Setting CPU Sampling interval.

4. Click **OK** to close the dialog and then right-click the **ProfilerDemo_Before** item in Performance Explorer, finally click Start Profiling from the popup menu. This will launch the Profiler.

If it is the first time you have run the Profiler, Visual Studio requires your permission for upgrading driver credentials to let the Profiler access system resources for monitoring purposes (see Figure 58.9).

Microsoft	Visual Studio
4	Unable to open profiler driver, would you like to upgrade credentials of DELSOLEWINT\Alessandro?
	Yes No

FIGURE 58.9 Upgrading driver profiler credentials.

Visual Studio now runs your application with an instance of the Profiler attached. During all the application lifetime Visual Studio will look like Figure 58.10, showing a work-in-progress message.



FIGURE 58.10 The profiler is currently running.

The application will be analyzed unless you terminate it. The profiler monitors performances at specific intervals to collect detailed information. When ready, simply close the application. This also detaches the profiler instance. At this point the profiler generates a report about the analysis. Figure 58.11 shows how the report appears.

If you are familiar with the profiler in Visual Studio 2008, you notice some differences but also some improvements in the new reporting system. On the top of the report there is a graph showing the CPU usage at monitored intervals. In the center of the screen there is the Hot Path summary, which shows the most resources consuming function calls and their hierarchy. The analyzed executable has a global impact of 100%, which is a high value and that can be negative for performances. Notice how the ConcatenationDemo method makes invocations to String.Concat. This is the function doing most individual work, as evidenced in the Hot Path view and in the Functions Doing Most Individual Work graph. It has a 98.65% impact meaning that there is some work to do to improve performance. By default, the profiler report shows results for your code but not for code invoked at a lower level by and within the CLR. Fortunately you can also analyze deeper function calls by clicking the **Show All Code** link in the upper right. Figure 58.12 shows how behind the scenes the sample code invokes the System.String.wstrcpy system function that also has a high impact.

Profession Control Vision Control V) ProfilerDemo - Microsoft Visual Studio ile Edit View Project Build Dahun Team Data Toole Architectura Tect Anabra	Window Help	-		
Profestionen, Sedorade 128.kmg Modulation Vertrait free: Sample Profiling Report Vertrait free: Sample Report	in gut get pojet guid geographicam oga geos sjentetale reg sjanget [] • □ • [] • [] • [] a a b b b o • ○ • [] • [] • [] • [] • [] • [] • []	- (2	2	- 🤜 🕾 🖬 🖄 🗶 🖩	1 🖻 🗉 -
Current View: Sumple Profiling Report 220 MBW samples calcuded The Current View: Sumple Profiling Report 220 MBW samples calcuded The Current View: Sum of the Sum	ProfilerDemo Before091203.vsp × Module1.vb				-
Spectrum with a construction of the function of	Current View: Summary	144			
<figure><figure></figure></figure>	Convelle Des filling Desent				1000
With the set of the set	222 total samples collected				\square
 Image: a constraint of the state o			CPU (% Usage)	Notifications	^
With the most exclusive simple scales Prefixe/Demus places active scales Prefixe/De	80 - /	٨	Zoom by selection	Show All Code View Guidance	
Image: Specific strains (Specific strains) Specific strains) Specific strains) Image: Specific strains) Specific strains) Specif strains) <t< td=""><td></td><td>ΛA</td><td>🔯 Zoom out</td><td>Report</td><td>^</td></t<>		ΛA	🔯 Zoom out	Report	^
Image: Second Secon	20-	\sqrt{V}		Egg Show Trimmed Call Tree	
0 3 6 9 12 13 12 24 27 30 Wall Clock Time (Eccond) Wall Clock Time (Eccond) 10 <td></td> <td>V ·</td> <td></td> <td>F Show Hot Lines</td> <td></td>		V ·		F Show Hot Lines	
Wall Clock Time (Excend) If Expert Report Data. Hot Path Image: Stand Angling Report. The most repensive call path based on sample counts Image: Stand Angling Report. Name Includive IS PrefineDemo, Before.ace 100,00 PrefineDemo, Before.Moduld.ConcetensionDemo() 90,55 PrefineDemo, Before.Moduld.ConcetensionDemo() 10,00 PrefineDemo, Before.Moduld.ConcetensionDemo() 10,00 PrefineDemo, Before.Moduld.ConcetensionDemo() 10,00 PrefineDemo, Before.Moduld.ConcetensionDemo() 10,00 PrefineDemo, Before.Moduld.LoncetensionDemo()	0 3 6 9 12 15 18 21 24	27 30		🚰 Compare Reports	
Hot Pathi Image: Second State	Wall Clock Time (Seconds)			Steport Report Data	
The most sepensive call path based on sample counts Image in the sepond base Name Inclusive % Exclusive % ProfiledDemo, Before Moduld. Admin) 100,00 0,00 ProfiledDemo, Before Moduld. ConcetantisticationDemo() 98,65 98,65 ProfiledDemo, Before Moduld. ConcetantisticationDemo() 98,65 98,65 Related Views: Call Tree Functions Sectore % 98,65 Functions with the most exclusive samples taken Name Exclusive Samples % System. String, ConcertationDemo() 9,65 System. School (Section Module) 10,65 ProfineDemo, Before, Moduld. (AncentationDemo() 10,65 ProfineDemo, Before, Module) 10,00	Hot Path			ave Analyzed Report	
Name Declare % Exclusive % Profile/Demo, Bidrac Advalda Main() 100,00 0,00 Profile/Demo, Bidrac Modula ConcentrationDemo() 98,35 98,65 Profile/Demo, Bidrac Modula ConcentrationDemo() 98,65 98,65 Retard View: Call The Functions Functions Doing Most Individual Work: Functions With the most activity samples taken Name Exclusive Samples % System, String, Concellstring, Demo 98,65 System String, Concellstring, Demo 98,65 Profile/Demo, Bidrac Modula ConstantionDemo() 98,65 System, Concellstring, Demo 98,65 Profile/Demo, Bidrac Modula ConstantionDemo() 98,65 System Concellstring, Demo 98,65 Profile/Demo, Bidrac Modula Locations and bene 98,65 Profile/Demo, Bidrac Modula Locations and bene 98,65 System, Console Readling() 1 Profile/Demo, Bidrac Modula Locations and bene 1 Distring Locations and bene <td< td=""><td>The most expensive call path based on sample counts</td><td></td><td></td><td>Filter Report Data</td><td></td></td<>	The most expensive call path based on sample counts			Filter Report Data	
ProfileDome,Before.ModuleLAnin() ProfileDome,Before.ModuleLAnin() ProfileDome,Before.ModuleLAnin() DotO ProfileDome,Before.ModuleLAnin() System.Concidential samples taken Name System.Concidential samples taken System.Concidential Anin()	Name	Inclusive %	Exclusive %	Toggle Full Screen	
• A indication between status used updet • A indication between status 0.00 0.00 • System String Concentration Demo() 98.55 98.65 Packed Views: Call Tree Functions 98.65 98.65 Functions using Most Individual Work Functions with the most exclusive samples taken Name Exclusive Samples % System String ConcentrationDemo() 10,00 System Console Reading() 10,00 System Console Reading() 10,00	ProfilerDemo_Before.exe DesfilerDemo Refere Maskelal Main()	100,00	0,00		
• Orace and exception of the second of the secon	Profile/Demo Before Module1 ConcatenationDemo()	100,00	0,00		
Related Views: Call Tree Functions Functions: Doing Most Individual Work Functions with the most acclusive samples taken Name System: Conscituting_string) ProfileDome, Before: ModuledConcetention:Domo() 1 0,03 System: Conscituting.taking) 1 ProfileDome, Before:Moduled.Main() 1 Vitror List (moduled.Main()) 1	System.String.Concat(string,string)	98,65	98,65		
Functions Doing Most Individual Work Functions with the most exclusive samples taken Name Exclusive Samples % System: String Concel[string,string) @88,65 Profile/Dome, Before: Module(I.concetestationDemo) i 0,99 System: Concele Read(Inc) i 0,45 Profile/Dome, Before: Module(I.Mainc) i 0,00	Related Views: Call Tree Functions				
Functions with the most exclusive samples taken Name System String Concet(string string) Performants System Console.ReadLine() Performants	Functions Doing Most Individual Work				
Name Exclusive Samples % System.String.Concet(stringstring) 98,65 Profiler/Demo,Before.ModuleL.Concetenation/Demo() 1 0,90 System.Console.Read(inet) 1 0,45 Profiler/Demo,Before.ModuleL.Main() 1 0,00	Functions with the most exclusive samples taken				
ayotemiaumayotematukang ang ang ang ang ang ang ang ang ang	Name Distant String Constitution string)		Exclusive Samples %		
System-Console.ReadLine() 1 0,45 Profile/Demo_Bidere_ModuleI.Main() 1 0,00	System.string.concatisting.string) ProfilerDemo.Before.Module1.ConcatenationDemo0		0.00		
ProfilerDano_Before.Modulel.Maint) 1 0,00	System.Console.ReadLine()		0,50		
🗱 Enror Est. 🖬 Output	ProfilerDemo_Before.Module1.Main()		0,00		
🙀 Error List 🖬 Output					
	🎼 Error List 📰 Output				

FIGURE 58.11 The analysis report produced by the profiler.

Contract Views Sammary Views Note: Not	an a se se ar a an a			
Image: Normal Views (Summary) Image: Normal Views (Summary) Sample Profiling Report Sample Profiling Report Views (Summary) Image: Normal Views (Summary) Image: Normal Views (Summary) Image: Normal V	rofilerDemo_Before091203.vsp × Modulc1.vb			
Sample Profiling Report 45 total surgets callactions	🗢 🗢 Current View: Summary 🔹 🗐 🔅	ା ମ ାନ୍ୟ ୟ		
445 total samples collected 1000000000000000000000000000000000000	Sample Profiling Report			
Image: Second				
Image: Constraint of the second of the se				
Image: State Stat	100		CPU (% Usage)	Notifications ^
Image: Contract of the section of t			Filter by selection	📴 Show Just My Code
Apple of a g a g a g a g a g a g a g a g a g a	60 - A	A a	Zoom reset Zoom out	(i) View Guidance
adjuine bit	40 -			Report ^
1 1 2 3 6 9 21 23 3 24 24 27 30 Will Clock Time (Second) PoliciteDong Sefore.ce PoliciteDong Sefore.ce O 0,000 O 1,0000 O 1,0000 </td <td></td> <td></td> <td></td> <td>Show Trimmed Call Tree</td>				Show Trimmed Call Tree
0 6 9 12 15 10 21 24 27 30 Wall Clock Time (2scend) 2 2 30 21 24 27 30 22 25 26		\vee \vee		E Cherry Met Lines
Wall Clock Time (Second) Image: Compare RegionL. Wall Clock Time (Second) Image: Compare RegionL. Hot Path Image: Compare RegionL. The most spensive call path based on sample counts Image: Compare RegionL. ProfileDemo_Before.se Image: Compare RegionL. ProfileDemo_Before.se Image: Compare RegionL. Montecontrained Image: Compare RegionL. ProfileDemo_Before.se Image: Compare RegionL. Montecontrained Image: Compare RegionL. Related View: Call Tree Functions Image: Compare RegionL. Montecontrained Compare RegionL. Image: Compare RegionL.	0 3 6 9 12 15 19 21 24	1 27 20		3- anow not blies
A transmission Constrained on transmission Constrained	Wall Clock Time (Seconds)			2 Compare Reports
Hot Path Issue any encounts Issue any encount				Export Report Data
The mass expensive call path based on sample counts QR These Report Data New Inclusive % Exclusive % ProfileDemo, Before.ose 100,00 0,00 MisCOREE.0L1 100,00 0,00 MisCOREE.0L1 50,79 0,00 MisCOREE.0L1 50,79 0,00 Related Verse: Call Tree Functions 49,21 0,00 Related Verse: Call Tree Functions Exclusive Samples 5,80 5,90 System: Sing workproj(char', char', inst2) Exclusive Samples 5,80 6,80 (riddil) 1,122 45,09 1,122 (ChritPs dil) 1,122 1,122 1,122 (ProfileDemo, Before.Moduld, ConcentrationDemo) 1,122 1,122 1,122	Hot Path			ave Analyzed Report
Name Inclusive % Exclusive % Exclusive % Exclusive % Inclusive % <th< td=""><td>The most expensive call path based on sample counts</td><td></td><td></td><td>Filter Report Data</td></th<>	The most expensive call path based on sample counts			Filter Report Data
> ProfileBrow_Before.ose 100,00 0,00 > IMSCOREE.0L1 100,00 0,00 > UMSCORE.0L1 50,79 0,00 > UMSCORE.0L1 50,79 0,00 > UMSCORE.0L1 50,79 0,00 > UMSCORE.0L1 49,21 0,00 Related Views: Call Tree Functions	Name	Inclusive %	Exclusive %	Toggle Full Screen
	ProfilerDemo_Before.exe	100,00	0,00	
OpenActional Lott I S0L 79 Ux000 OpenActions Transition 49.21 0.000 Related Views: Call Tree Functions Super-Super	[MSCOREE.DLL]	100,00	0,00	
Control material 49,21 0,00 Related Views: Call Tree Functions Related Views: Call Tree Functions Functions Units the most exclusive samples taken Exclusive Samples % (cf.rdf) 46,54 Systems.String.systerpcy(char", char", int22) 46,09 (relative) 1,122 (CRYP159-dt) 1,122 (CRYP159-dt) 1,025	(MSCOREE.DLL)	50,79	0,00	
Reactions Functions System Simily we colspan="2">System Simil	Onknown Frame(s)	49,21	0,00	
Functions Doing Most Individual Work Functions with the most exclusive samples taken Sciculare Samples % Name Exclusive Samples % [dxdll] 40,54 [metaldl] 41,22 [crdit/ling] 1,122 [CrivPTSrdl] 0,45	Related Views: Call Tree Functions			
Functions with the most exclusive samples taken Exclusive Samples % [cf.rdl] 40,54 System.String waterpoycharé,charé,int32) 40,09 [rdll.dl] 1,12 [CRPPTSP.dl] 1,12 [CRPPTSP.dl] 1,12	Functions Doing Most Individual Work			
Name Exclusive Samples % [cl.dl] 45,5 system.String,witrepy(char*,char*,int22) 40,0 [ndll.dl] 1,12 [CRYP159.dl] 1,12 [CRYP159.dl] 1,12 [CRYP159.dl] 1,12	Functions with the most exclusive samples taken			
[cf.dl] 454 SystemSuring workcpy(char*,char*,int32) 459 [cf.dl] 1.12 [CKVPTSHd] 1.12 [CKVPTSHd] 1.12	Name		Exclusive Samples %	
System:Sting.wstrcp/char/shar/in32) 48,99 [redlat0] 1,12 [CKVP159:d0] 1,12 PolisiDemo (Sefore:Moduld.ConcentrationDemo) 0,45	[cir.dil]		48,54	
[ndtlid] 1.12 [CRVP153-dD] 1.12 PoliciDemo_Before.Moduld.ConcatentianDemo() 0,45	System.String.wstrcpy(char*,char*,int32)		48,09	
[CRYPTSP.dll] I 1,12 Profile/Demo, Before.Module1.ConcatenationDemo() I 0,45	[ntdll.dll]		1,12	
ProfilerDemo_Before.Module1.ConcatenationDemo() I 0,45	[CRYPTSP.dll]		1,12	
	ProfilerDemo_Before.Module1.ConcatenationDemo()		1 0,45	

FIGURE 58.12 Showing all code function calls.

The **Current View** combo box enables viewing lots of other useful information, such as processes involved in profiling, function details, lines of code most consuming, and marks. For marks, you can check out intervals and related CPU usage percentage, as shown in Figure 58.13.

8 - 🗳 🖬 🖉 😭 👗 🖧 🖄 🖄	(* - 💭 - 🖏 🕨 De	bug • x86	• 🚳	- 🖓 🐨 🎲 🖄 💥 🔛
ilerDemo_Before091203.vsp × Module1.vt	, ,			
⇒ Current View: Marks		12 3 5 1 h d 4 4		
Mark ID Mark Name	Timestamp	\Processor(_Total)\% Pr		
0 Start of Program	0,00	0		
1 AutoMark	67,09	0		
2 AutoMark	583,60	72		
3 AutoMark	1.106,25	91		
4 AutoMark	1.823,32	92		
5 AutoMark	2.339,78	86		
6 AutoMark	2.887,36	98		
7 AutoMark	3.403,48	86		
8 AutoMark	3.915,26	51		
9 AutoMark	4.440,31	22		
10 AutoMark	5.440,35	20		
11 AutoMark	5.940,37	15		
12 AutoMark	6.940,59	10		
13 AutoMark	7.940,45	15		
14 AutoMark	8.440,48	23		
15 AutoMark	9,444,51	24		
16 AutoMark	10.444,58	18		
17 AutoMark	11.444,60	18		
18 AutoMark	11.944,63	18		
19 AutoMark	12.944,89	63		
20 AutoMark	13.947,07	24		
21 AutoMark	14.447,78	15		
22 AutoMark	15.447,84	23		
23 AutoMark	15.949,81	26		
24 AutoMark	16.950,87	68		
25 AutoMark	17.950,91	20		
26 AutoMark	18.972,89	35		
27 AutoMark	19.483,97	43		
28 AutoMark	19.984,01	14		
29 AutoMark	20.984,09	24		
30 AutoMark	21.984,11	32		
31 AutoMark	22.485,10	34		
32 AutoMark	22.985,49	7		
33 AutoMark	23.485,21	40		
34 AutoMark	24.491,60	25		
35 AutoMark	24.997,22	35		
36 AutoMark	25.997,25	8		
37 AutoMark	26.998,78	27		
38 AutoMark	27.515,71	51		

FIGURE 58.13 Checking intervals and CPU usage.

The report is saved as a .vsp file that can be useful for later comparisons. You can also export the report to Microsoft Excel or as Xml data (Export Report Data command). There is other good news. If you check out the Error List window, you notice a warning message saying that the invocation to String.Concat has an impact of 100 and that you should consider using a StringBuilder for string concatenations. This means that in most cases Visual Studio can detect the problem and provide the appropriate suggestions to fix it. Following this suggestion, replace the application code as follows:

```
Module Module1
Sub Main()
ConcatenationDemo()
Console.ReadLine()
End Sub
```

```
Sub ConcatenationDemo()
Dim testString As New Text.StringBuilder
For i = 0 To 10000
        testString.Append("I love VB 2010")
        Next
End Sub
End Module
```

Now follow the same steps described before about setting the value for the CPU Sampling clock cycles to 50000 for the current project. Then run the Profiler again via the **Analyze**, **Profiler, Start Profiling** command. After a few seconds, Visual Studio 2010 generates a new report, represented in Figure 58.14.



FIGURE 58.14 The new report after code improvements.

Notice how the invocations to strings concatenations method completely disappeared from the performance negative impacts. Also notice how the CPU usage is better than before. This means that our edits improved the application performance, as also demonstrated by the Error List window that is now empty. The most resource-consuming function is now Console.ReadLine, which just waits for you to press a key and therefore can be completely ignored. To get a better idea of what actually happened you can compare the two generated reports. Right-click one of the available reports in the Performance Explorer tool window and select Compare Performance Report. In the dialog browse for the second report and click OK. Figure 58.15 shows the result of the comparison.

90 P	rofilerDemo - Microse	oft Visual Studio	_				
Eile	<u>E</u> dit <u>V</u> iew <u>P</u> roje	ct <u>B</u> uild <u>D</u> ebug	Tea <u>m</u> D <u>a</u> ta	<u>I</u> ools A <u>r</u> chite	cture Te <u>s</u> t	A <u>n</u> alyze <u>W</u> indow <u>H</u> elp)]]
] • 💷 • 🚰 🗐 🔐		=) = (= = 4	3 • 🖏 🕨 🗖	ebug *	x86	• 🙆 ÷
	w work item * La	- 18 -					
	Comparison Report	× ProfilerDemo	After091203(1).vsp	o Module1	.vb Modu	le1.vb	
ata S	Comparison Files				Comparison	Options	olutio
onice	Baseline File:	ProfilerDemo_B	fore091203.vsp		Table:	Functions	• <u></u>
и - 111	Comparison File:	ProfilerDemo A	ter091203(1).vsp		Column:	Exclusive Samples %	↓ plore
Ser					Threshold:	1	
ver E						Apply	Prop
cplon						Арріу	oertie
							5
Per							
forma							n Ext
ance	(1) Comparison con	malata					
Explo	Comparison Colum	npiete.	· ·	Delta F	aseline Value	Comparison Value	
ē,	System.Console.Read	dLine()	1	99,55	0,45	100,00	Class
7	System.String.Conca	t(string,string)	1	-98,65	98,65	0,00	Viev
polbc							<
× N							
Dat							
0							
lectic							
on Ce							
Introl							
	eta						
Par	🙀 Error List 🔳 Out	tput					
L	y						.:

FIGURE 58.15 Comparing two performance reports.

The Delta column indicates the difference between the old value and the new one. The Baseline column indicates the value in the old analysis, whereas Comparison indicates the value in the new analysis. Referring to this specific example, it means that the String.Concat function passed from a bad state to a good one. The profiler is a good friend in your developer life, and the suggestion is to profile applications every time you can. The Visual Studio 2010 profiler also offers improvements for profiling multithreaded applications, which is useful against parallel programming techniques.

Profiling External Executables

The Visual Studio profiler is not limited to solutions opened in the IDE, but it can be successfully used for profiling standalone executables. Basically you are allowed profiling also Win32 executables over managed ones. To demonstrate how this works, close the solution opened in Visual Studio (if any) ensuring that nothing is available in Solution Explorer. Now start the Performance Wizard following the steps shown in the previous section until you get the result shown in Figure 58.5. The next step is selecting the **An Executable** (.EXE file) option. When selected this, you need to specify an executable to analyze. Just for demo purposes, select the executable generated in the previous example. Figure 58.16 shows how to accomplish this.

After completing this wizard, Visual Studio launches the specified executable with an instance of the profiler attached. When completed, Visual Studio also generates a report, as shown in the previous section. In this particular case you get nothing but the same profiling results, being substantially the same application. In all other cases you get specific analysis results for the selected executable. The Error List continues to show warning or error messages related to the application performance providing the appropriate suggestions when available. Of course, improving performance requires having the source code for the executable analyzed.

Performance Wizard Page 3 of 4	x
Select an executable to profile and specify start options if needed	
The Performance Explorer will launch the executable that you have specified and gather performance data as it executes. Once the executable terminates, the performance results will be presented for analysis.	
What is the full path to the executable?	
C:\VB2010Unleashed\Code\Ch58\ProfilerDemo\ProfilerDemo_Before\bin\Debug\ProfilerDemo_Before.exi	
Start options	
Command-line arguments:	
Working directory:	
< Previous Next > Einish Cancel	

FIGURE 58.16 Selecting a standalone executable.

IntelliTrace, the Historical Debugger

EDITION NOTE

The IntelliTrace debugger is available only with the Visual Studio 2010 Ultimate edition.

One of the most important new tools in Visual Studio 2010 is *IntelliTrace*, formerly known as the historical debugger. This tool can improve your debugging experience because it can record and navigate every event occurring during the application lifetime, such as events and failures including information on specific threads. IntelliTrace is fully integrated with the code editor and with the rest of the IDE functionalities, such as Call Stack and Locals tool windows so that it can provide a complete debugging environment. The tool is capable of recording (to a file, too) and debugging the following happenings:

- ▶ Application events, such as user interface events or application exceptions
- Playback debugging, which allows deep debugging over specific events occurred before and after a particular code block
- Unit test failures
- Load test failures and build acceptances test
- Manual tests

In this section you learn how to use IntelliTrace to debug application events, exceptions, and unit test failures. Before showing IntelliTrace in action, it is a good idea to set its options in Visual Studio.

IntelliTrace Options

There are a lot of available options for customizing IntelliTrace's behavior. You set its options via the usual **Tools**, **Options** command and then selecting the IntelliTrace item. Figure 58.17 demonstrates this.

IntelliTrace is enabled by default. The default behavior is that IntelliTrace will collect events only, but for this discussion select **IntelliTrace Events and Call Information** to get complete information on the code calls, too. This kind of monitoring is the most expensive in terms of system resources and should be used only when necessary but it offers a high-level view of what happens during the application lifetime. You have deep control over application events that IntelliTrace can keep track of. Select the **IntelliTrace Events** item. You see a list of .NET elements, such as ADO.NET, Data Binding, and Threading; for each of them you can specify particular events you want to be traced. For example the Gesture events collection enables specifying user interface events such as button clicks, as demonstrated in Figure 58.18.



FIGURE 58.17 Setting general options for IntelliTrace.



FIGURE 58.18 Selecting events to be tracked by IntelliTrace.

The Advanced and Modules items enable respectively specifying locations and size for logs and which .NET modules must be tracked other than the current application. For this demonstration leave the default settings unchanged. At this point we need a sample application to see IntelliTrace in action.

Creating a Sample Application

The goal of this section is to illustrate how IntelliTrace can save your time by tracking events and exceptions with detailed information that can help you to understand what the problem is. According to this, creating a client WPF application can be a good example. To demonstrate both application events and exceptions, we can place a button

that invokes a method attempting to open a file that does not exist. After creating a new WPF project named (IntelliTraceDemoApp) with Visual Basic, in the XAML code editor, write the code shown in Listing 58.3.

```
LISTING 58.3 Setting Up the User Interface for the Sample Application
```

```
<Window x:Class="MainWindow"
xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
Title="MainWindow" Height="350" Width="525">
<Grid>
<Grid>
<Grid.RowDefinitions>
<RowDefinition />
<RowDefinition />
<Grid.RowDefinitions>
<Chrid.RowDefinitions>
<Chrid.RowDefinitions>
<Chrid.RowDefinitions>
<Grid.RowDefinitions>
<Grid.Row="1"/>
</Grid>
</Window>
```

On the Visual Basic code-behind side, write the code shown in Listing 58.4, which simply handles the Button.Click event and tries to open a text file.

```
LISTING 58.4 Writing the Failing Visual Basic Code Before Running IntelliTrace
```

Class MainWindow

```
Private Sub OpenFileButton_Click(ByVal sender As System.Object,
ByVal e As System.Windows.RoutedEventArgs) _
Handles OpenFileButton.Click
Me.FileTextBox.Text = OpenFile()
End Sub
'Just for demo purposes
'Consider dialogs implementation in real life apps
Private Function OpenFile() As String
'Attempting to open a fake file
Return My.Computer.FileSystem.
```

```
ReadAllText("C:\Alessandro.txt")
```

End Function End Class

Now that we have a sample project, which will voluntarily fail at runtime, it is time to see how to catch problems via IntelliTrace.

Tracking Application Events and Exceptions with IntelliTrace

Run the demo application and click the button to cause an exception. The application breaks because the specified file is not found. At this point the IntelliTrace tool window appears inside Visual Studio 2010, showing a list of events that you can see in Figure 58.19.

IntelliTrace									• 4 ×
10 C									
All Categories			- AI	Threads					-
Search									Q
Debugger: Beginn	ing of Application: M	ain, Application.g.vb line 61							
# Environment Variation	ables: Retrieving "win	dir"							
🎸 Environment Vari	ables: Evaluated to 'C:	\Windows'							
Gesture: Clicked The user clicked the Thread: Main Thr Related views: L	Open file" (System.Wi ne OpenFileButton cor ead [3320] ocals, Call Stack	ndows.Controls.Button) trol of type System.Windov	vs.Controls.Butt	on with content	"Open file".				
Exception: Throw	n: "Could not find file	'C:\\Alessandro.txt'." ("Syste	em.IO.FileNotFo	undException")					
Debugger: Stoppe	ed at Exception								
Live Event: Except	tion Intercepted: Open	File, MainWindow.xaml.vb	line 16						
🗟 IntelliTr Soluti	on Ex 📷 Team E	Properties 🛛 😹 Call Stack	c 👼 Breakp	🚬 Comm	📻 Immedi	E Output	🗮 Autos	🐺 Locals	👼 Watch 1

FIGURE 58.19 IntelliTrace in action, showing occurred events.

FILTERING RESULTS

You can filter information by category and by thread. Use the upper combo box to respectively select which kind of events category you want to check out (for example, Console, ASP.NET, ADO.NET, and so on) and which specific thread you want to get tracking information for.

As you can see, IntelliTrace is not just a debugger showing errors or places where errors occurred. It is a powerful tool capable of tracking every single event occurring during the entire application lifetime—meaning that you could even use IntelliTrace to just keep track of events without errors. In the current example notice how the first tracked event is the application startup. There are other events tracked: Click the **Gesture** event to get information on such a happening. IntelliTrace informs you that the user clicked the

OpenFileButton control, which is of type System.Windows.Controls.Button and whose content is Open File, also showing the thread description and ID. This is a predictable event, but as you can imagine you can keep track of all events, for example when databinding is performed on a data-bound control or when the user selects an item from a ListBox or when the application attempts to make a particular action. This can produce a huge amount of information that you can analyze later. To get information on the exception thrown, simply click the exception item. Visual Studio shows the full error message and the line of code that caused the exception; when you click such an item, the line of code will be automatically highlighted in the code editor for your convenience. IntelliTrace also offers the ability of checking the entire tree of function calls (just in case you enable call information). In a few words, this feature keeps track of every single function call made by the runtime during the application lifetime. To show this, click the **Show Calls View** button in the IntelliTrace toolbar. Figure 58.20 shows an example, pointing to the place where the user clicked the button.



FIGURE 58.20 Showing functions call tree with IntelliTrace.

The sample figure shows the first function call that is a call for starting the main thread. The next functions are calls that the CLR makes to run the application and initialize it; the list is complete, also showing calls made during the rest of the application lifetime until it broke because of the exception. You can easily browse calls using the vertical scroll bar on the right.

Analyzing IntelliTrace Logs

When the application is running with the debugger attached, IntelliTrace records everything happening. (This is the reason why you might notice a small performance decrease when you select the events and call recording option.) Such recordings are saved to log files available in the C:\ProgramData\Microsoft Visual Studio\10.0\TraceDebugging folder and can be analyzed directly from within Visual Studio. To accomplish this, follow these steps:

- In Windows Explorer, open the C:\ProgramData\Microsoft Visual Studio\10.0\TraceDebugging folder.
- **2.** Double-click the last log file related to your application. Notice that log file names all begin with the application name but you can open the most recent according to the date modified value.

Visual Studio opens the log file showing a result similar to what you see in Figure 58.21.

Logs contain lots of information. For example, you can select a particular thread in the upper graph and check for the related threads list below. This is useful for understanding at what time a specific thread was tracked. Moreover information on exceptions will also be shown. You can also check about system information and modules involved in the tracking process (such information is placed at the bottom of the page).

IntelliTrace Summary 12242000 503:09 PM Image: State St	itelli i race	eSamplTrace Log Do	cument × Application.g.vb	MainWindow.xaml.vb			
Threads bit Double clock on a thread in the lat below to stat a kitell Tace Debugging session for that thread. Threads List Double clock on a thread in the lat below to stat a kitell Tace Debugging session for that thread. State 0 No Name 04/12/2000 173:025 7:474511 State 0/12/2000 173:025 7:4	IntolliT						·
* Threads List Double click on a fired in the list below to star a fired Taxe Debugging session for that thread. * Threads List Double click on a fired in the list below to star a fired Taxe Debugging session for that thread. * Threads List 0 10 Name > [5124] 17:05:00 17:25:00 17:25:00 17:05:00 17:25:00 17:30:00 Threads List Threads List 10 10 Name > [5124] 04/12/2000 17:00:53:24511 10 Name > [6142000 17:00:00 53:24511 04/12/2000 17:00:53:24511 10 Name > [6142000 17:00:00 17:02:00 17:02:57:44:511 104/12/2000 17:00:53:4511 10 Name > [6142000 17:00:00 17:02:00 17:02:57:44:511 104/12/2000 17:03:51:24:1511 10 Name > [6142000 17:00:00 17:02:57:44:511 104/12/2000 17:03:51:24:1511 10 Name > [6142000 17:00:00 17:00:00 17:32:57:44:511 104/12/2000 17:00:17:32:57:44:511 10 Name > [6142000 17:00:00 17:00:00 17:32:57:44:511 104/12/2000 17:00:17:32:57:44:511 10 Name > [6142000 17:00:00 17:00:00 17:32:57:44:511 104/12/2000 17:00:17:34:57:44:511 10 Name > [6142000 17:0	12/4/200	1ace Summary					^
Threads List Dable dok on a threads in the list below to stat a Intel ^T Tace Debugging session for that thread. Search 10 Name Stat Time Stat 10 Name Stat 10 Name Stat 116 cHo Name Stat 116 cHo Name Stat 116 cHo Name Stat 117:05:00 17:20:00 17:05:00 17:20:00 17:05:00 17:20:00 17:05:00 17:20:00 17:05:00 17:20:00 17:05:00 17:20:00 17:05:00 17:20:00 17:05:00 17:20:00 17:05:00 17:20:00 17:05:00 17:20:00 17:05:00 17:20:00 17:05:00 17:20:00 17:05:00 17:20:00 18:00 17:20:00 19:00 Name Stat 10 Name Stat <							
- (Thread Ended> (4136) - (No Name> [016]) - (No Name> [016]) - (No Name> [016]) - (No Name> [017:20:00]) - (No Name> [017:20:00])	1.	1 1		1 1			
All marks 1 (dd 2) No Name > [016] <t< td=""><td>^</td><td></td><td></td><td></td><td></td><td></td><td></td></t<>	^						
No Name> [6116] (No Name> [6176])	-			<thread ended=""></thread>	4836]		
Image: second	-			<no name=""> [61</no>	16]		
Image: Signed in the lat below to stat a kreal Trace Debugging session for that threads Image: Signed in the lat below to stat a kreal Trace Debugging session for that threads Image: Signed in the lat below to stat a kreal Trace Debugging session for that threads Image: Signed in the lat below to stat a kreal Trace Debugging session for that threads Image: Signed in the lat below to stat a kreal Trace Debugging session for that threads Image: Signed in the lat below to stat a kreal Trace Debugging session for that threads Search Image: Signed in the lat below to stat a kreal Trace Debugging session for that threads Search Image: Signed in the lat below to stat a kreal Trace Debugging session for that threads Search Image: Signed in the lat below to stat a kreal Trace Debugging session for that threads Search Image: Signed in the lat below to stat a kreal Trace Debugging session for that threads Search Image: Signed in the lat below to stat a kreal Trace Debugging session for that threads Search Image: Signed in the lat below to stat a kreal Trace Debugging session for that threads Search Image: Signed in the lat below to stat a kreal Trace Debugging session for that threads Search Image: Signed in the lat below to stat a kreal Trace Debugging session for that threads Search Image: Signed in the lat below to stat a kreal Trace Debugging session for that threads Sear			531				
Cho Name [476] vhorkLoad68ference [016] Threads List Double clock on a thread in the lat below to stat a kitell Trace Debugging session for that thread. Search 0 Name 9404 Amme > 04/12/2009 170 00.08 5324511 6116 cho Name > 04/12/2009 170 00.08 5324511 6116 cho Name > 04/12/2009 170 00.08 5324511 6116 vho Name > 04/12/2009 170 00.08 5344511 6117 04/12/2009 170.00.0	read	<no name="">[3</no>	52]				
Threads List Double click on a thread in the list below to start a Intell Tace Debugging session for that thread. Search ID Name Start Time End Time 4356 <17:05:00 17:32:00 17:32:02 17:32:02 17:32:00 17:32:02 17:32:00 17:32:02 17:32:00 17:32:02 17:32:02 17:03:00 17:32:02 17:32:02 17:03:00 17:32:02 17:03:00 17:02:00:00 17:02:00:00 17:02:00:00 17:02:00:00 17:02:00:00 17:02:00:00	Ē.			<no name=""> [47</no>	64]		
Unionit Loaderference [14] Instance • Threads List Instance Instance Instance Instance Dable clock on a thread in the list below to stat a kreal Trace Debugging session for that thread Instance Instance Instance Start In Instance Instance Instance Instance ID Name Start Time End Time Instance Instance IB ID Name Start Time End Time Instance Start ID Instance Instance Instance Instance Start Instance Instance Instance Instance Instance Start Insthe Instance Instan		· · · ·	1 10 0 10 10 10	[
Image: Size of the last below to stat a Intell Table Debugging session for that three many Intracting in the last below to stat a Intell Table Debugging session for that three many Image: Intell table of the last below to stat a Intell Table Debugging session for that three many East Table Debugging session for that three many Image: Intell table of	-	vsho	ist.LoadKererence [816]				
Image: Non-Name > 13384 17:10:00 17:15:00 17:20:00 17:20:00 17:20:00 Threads List Timestamp Timestamp Timestamp Timestamp Timestamp * Threads List Double click on a thread in the list below to start a Intell Times Debugging session for that thread. Search Search <td>-</td> <td>vshost.NotifyLoad [5</td> <td>i832]</td> <td></td> <td></td> <td></td> <td></td>	-	vshost.NotifyLoad [5	i832]				
Image: Provide (Name) 17:10:00 17:15:00 17:20:00 17:20:00 17:20:00 Timestamp Timestamp <td< td=""><td>v</td><td></td><td></td><td></td><td></td><td></td><td></td></td<>	v						
Timestamp * Threads List Double click on a thread in the lit below to star a Intell Trace Debugging session for that thread. Search 10 Name Start Time End Time End Time 4836 <11mead Ended> 04/12/2009 173:039 5324511 04/12/2009 173:039 5324511 04/12/2009 173:039 173:425 747511 516 child Name> 04/12/2009 173:039 534511 04/12/2009 173:039 173:425 747511 516 vhort Laad/Reference 04/12/2009 173:011 10:04511 04/12/2009 173:0557:0414511 518 vhort Laad/Reference 04/12/2009 173:04511 04/12/2009 173:04511 5384 child Name> 04/12/2009 173:04511 04/12/2009 173:04511 5384 child Name> 04/12/2009 173:04511 04/12/2009 173:04511 5384 child Name> 04/12/2009 173:01511 04/12/2009 173:01511 552 Mani Tread 04/12/2009 173:01511 04/12/2009 173:01511 5534 Alle RameA 04/12/2009 173:01511 04/12/2009 173:01511 5534 NET SystemErveria 04/12/2009 173:0151 04/12/2009 173:0151 5534 Alle RameA		<no name=""> [5384] 17:05:00</no>	17:10:00	17:15:00 17:20	17:25:00	17:30:00	
				Timestamp			
Image: Second				······			
Characteristic End Time Double clok on a thread in the list below to start a Intell Trace Debugging session for that thread. Search Search ID Name Start Time End Time 4836 c/Thread Ended> 04/12/2009 17/3039 5324511 04/12/2009 17/3425 7474511 5116 c/Ne Name 04/12/2009 17/3039 5324511 04/12/2009 17/3425 7474511 522 c/Ne Name 04/12/2009 17/3012000 17/3425 7474511 5816 whole Name 04/12/2009 17/3012000 17/3425 7474511 582 valout Loderflerence 04/12/2009 17/301120000 17/3425 7474511 5832 valout NethfyLoad 04/12/2009 17/3012000 17/3425 7474511 5834 clo Name 04/12/2009 17/3012000 17/3425 7474511 5834 valout NethfyLoad 04/12/2009 17/3012000 17/3425 7474511 5834 valout NethfyLoad 04/12/2009 17/3120101 1/3425 1/7342511 5934 MET SystemEnverts 04/12/2009 17/312/325 1/7342511 5932 Man Thread 04/12/2009 17/3425 7474511 5932 Man Thread 04/12/2009 17/3425 7474511							=
Incelos List Double clok on a thread in the list below to stat a total Trace Debugging session for that thread. Search End Time ID Name Stat Time End Time 4436 Chrome Search Out/12009 170 30.95324511 GA1/22009 173.92.57444511 516 cHear Reference 04/12/2009 170.311.0504511 GA/122009 173.92.57444511 518 whose Laad Reference 04/12/2009 170.32.104064511 GA1/22009 173.92.57444511 5384 cHear Reference 04/12/2009 173.57104611 GA1/22009 173.92.20145131 5394 WhET Reference 04/12/2009 173.57104511 GA1/22009 173.42.57474511 5552 Main Thread 04/12/2009 173.03.57104511 GA1/22009 173.92.2714511 5564 MET SystemEverts 04/12/2009 173.03.57104151 GA1/22009 173.42.57474511 5552 Main Thread 04/12/2009 173.03.57104151 GA1/22009 173.42.57474511 <td></td> <td></td> <td></td> <td></td> <td></td> <td></td> <td></td>							
Booke clack on a thread in the list below to stat a Intell Trace Debugging session for that thread. Search ID Name Stat Time End Time 4336 <thread ended=""> 0.4/12/2009 173.09.5324511 0.4/12/2009 173.42.57.474511 6116 OH Rame> 0.4/12/2009 173.09.5324511 0.4/12/2009 173.42.57.474511 532 oH Rame> 0.4/12/2009 173.10.37511 0.4/12/2009 173.42.57.474511 516 vhoit LoadFeferrine 0.4/12/2009 173.11.0374511 0.4/12/2009 173.42.57.474511 518 vhoit AbdifyLand 0.4/12/2009 173.42.21.174511 0.4/12/2009 173.42.57.474511 5384 vhoit AbdifyLand 0.4/12/2009 173.42.2009 173.42.2009 173.42.57.474511 5384 vhoit AbdifyLand 0.4/12/2009 173.42.2009 173.42.2009 173.42.57.474511 5384 vhoit AbdifyLand 0.4/12/2009 173.42.2000 173.42.57.474511 5632 Mait Thread 0.4/12/2009 173.61.32.611</thread>	^ 1 hr	eads List					
Boarch IV ID Name Start Time End Time 4836 <thread ended=""> 04/12/2009 17/30.95 5324511 04/12/2009 17/3425 7474511 6116 <hr/><hr/><hr/><hr/><hr/><hr/><hr/><hr <="" td=""/><td></td><td></td><td></td><td></td><td></td><td></td><td></td></thread>							
ID Name Start Time End Time 4036 Chroad Exded> Oth/22009 1730.09.5524511 Adv.2000 1734.25.744511 6116 cline Name> Out/22009 1730.09.5524511 Adv.2009 1734.25.744511 352 cline Name> Out/22009 1730.11.074511 Out/22009 1734.25.744511 4764 cline Name> Out/22009 1730.11.074511 Out/22009 1734.25.744511 518 vinos Laad Riference Out/22009 1730.21.0064511 Out/22009 173.57.5444511 582 vinos Handraketer Out/22009 173.02.1046511 Out/22009 173.02.14511 584 cline NameNetWidsad Out/22009 173.57.5104651 Out/22009 173.02.21.04511 584 vinos Landriference Out/22009 173.02.01.0001 13.42.57.474511 5934 vinos Landriference Out/22009 173.57.5104611 Out/22009 173.42.57.474511 5954 NET SystemEversis Out/22009 173.03.57.510461 Out/22009 173.42.57.474511 5954 MET SystemEversis Out/22009 173.03.59.51041 Out/22009 173.42.57.474511	Doub	ole click on a thread in th	e list below to start a IntelliTrace I	Debugging session for that thread			
436 0.4/12/2000 17/0.309 532451 0.4/12/2000 17/0.428 27/47511 6116 clo Name> 0.4/12/2000 17/0.309 534451 0.4/12/2000 17/0.428 27/47511 352 clo Name> 0.4/12/2000 17/0.459 534451 0.4/12/2000 17/0.428 27/47511 352 clo Name> 0.4/12/2000 17/0.411 0.04/12/2000 17/0.428 57/47511 4764 clo Name> 0.4/12/2000 17/0.111.05/0.611 0.4/12/2000 17/0.001 71.557 57/47511 5832 vrhont.Net/j.cod 0.4/12/2000 17/0.122000 17/0.12200 17/0.512 0.174511 0.4/12/2000 17/0.1200 17/0.123/174511 5384 clo Name> 0.4/12/2000 17/0.122000 17/0.12200 17/0.123/174511 0.4/12/2000 17/0.1200 17/0.123/174511 5384 clo Name> 0.4/12/2000 17/0.122000 17/0.12200 17/0.123/174511 0.4/12/2000 17/0.1200 17/0.123/174511 562 Wahot.RunFlaxing/Window 0.4/12/2000 17/0.12200 17/0.1200 17/0.425 7/47511 0.4/12/2000 17/0.123/174511 562 Man 0.4/12/2000 17/0.1384511 0.4/12/2000 17/0.425 7/47511 562 Man 0.4/12/2000 17/0.1384511 0.4/12/2000 17/0.425 7/47511	Doub	ole click on a thread in th arch	e list below to start a IntelliTrace I ×	Debugging session for that thread			
6116 cHo Name> 04/12/2009 17/0.39 53/4511 04/12/2009 17/0.32 57/4511 552 cHo Name> 04/12/2009 17/0.31 20/3511 04/12/2009 17/0.31 20/3511 04/12/2009 17/0.31 20/3511 4764 cHo Name> 04/12/2009 17/0.31 10/35451 04/12/2009 17/0.31 20/3511 04/12/2009 17/0.31 20/3511 516 vrbot LodeReirenco 04/12/2009 17/0.31 10/5514511 04/12/2009 17/0.31 20/35111 5384 cHo Name> 04/12/2009 17/0.31 20/45111 04/12/2009 17/0.31 20/15511 5384 vrbot Andreidrenco 04/12/2009 17/0.31 20/15511 04/12/2009 17/0.31 20/1511 548 vrbote Andreidrenco 04/12/2009 17/0.51 20/35111 04/12/2009 17/0.31 20/1511 5594 vrBragkenette 04/12/2009 17/0.51 20/3511 04/12/2009 17/3.42 57/4511 5632 Main Tread 04/12/2009 17/0.51 20/1511 04/12/2009 17/3.42 57/4511	Sea	ole click on a thread in th arch ID	e list below to start a IntelliTrace I × Name	Debugging session for that thread Start Time	End Time		
352 He Name> 04/12/2009 17:03:11:0374511 04/12/2009 17:03:11:0374511 4764 He Name> 04/12/2009 17:03:11:036411 04/12/2009 17:03:1575414511 816 vrhon: Load/Reference 04/12/2009 17:03:11:1764511 04/12/2009 17:03:1575414511 5382 vrhon: Neft/Load 04/12/2009 17:03:12:04611 04/12/2009 17:03:22:0147511 5384 He Name> 04/12/2009 17:03:12:014511 04/12/2009 17:03:22:0145411 4168 vrhon: RunParkerg/Wrhon 04/12/2009 17:03:12:014511 04/12/2009 17:03:22:0145411 5504 .NET SystemEverts 04/12/2009 17:10:13:384511 04/12/2009 17:13:22:747511 5522 Main Tread 04/12/2009 17:10:13:384511 04/12/2009 17:14:25:7474511	Doub Sea	ole click on a thread in th arch ID 36	e list below to start a IntelliTrace I X Name <thread ended=""></thread>	Debugging session for that thread Start Time 04/12/2009 17:03:09.5324511	End Time 04/12/2009 17:34:25.7474511		
4764 clo Hame> 04/12/2009 17:03:11.0504511 04/12/2009 17:03:225 7474511 816 vrbnct.Lod3Fdremce 04/12/2009 17:03:11.724511 04/12/2009 17:03:22009 17:155:75414511 5832 vrbnot.NtelfyLoad 04/12/2009 17:03:11.724511 04/12/2009 17:03:120174511 5834 clo Hame> 04/12/2009 17:03:12.014511 04/12/2009 17:03:12.014511 4168 vrbnot.RunFarking/Window 04/12/2009 17:03:12.014511 04/12/2009 17:03:25.7474511 5054 .HET SystemEverts 04/12/2009 17:01:334511 04/12/2009 17:03:25.7474511 5652 Main Treed 04/12/2009 17:03:2851 04/12/2009 17:03:27.747511	Doub Sea 483 61	ID I	e list below to start a IntelliTrace I Name <thread ended=""> <no name=""></no></thread>	Start Time 04/12/2009 17:03:09.5324511 04/12/2009 17:03:09.5344511	End Time 04/12/2009 17:34:25.7474511 04/12/2009 17:34:25.7474511		
B16 vnhor. LaddFederence D4/12/2009 17:03:11.1724511 O4/12/2009 17:03:57.8414511 5832 vnhor. Norfl, Laad D4/12/2009 17:03:12.064511 04/12/2009 17:03:22.0145711 584 vlant. RuhPeterg/Whorks D4/12/2009 17:03:12.014511 O4/12/2009 17:03:22.0145711 584 vlant. RuhPeterg/Whorks D4/12/2009 17:03:57.014511 04/12/2009 17:03:22.0145711 4168 vnhort. RuhPeterg/Whorks D4/12/2009 17:10:57.014511 04/12/2009 17:10:42:57.474511 5632 Main Treesd 04/12/2009 17:10:13.984511 04/12/2009 17:14:22:7474511	Doub Sec 483 611 352	ID I	e list below to start a Intelli Trace I Name <thread ended=""> <thread ended=""> <thread name=""> <thread name=""> <thread name=""> <thread name=""></thread></thread></thread></thread></thread></thread>	Start Time 04/12/2009 17:03:09.5324511 04/12/2009 17:03:09.5344511 04/12/2009 17:03:09.5344511 04/12/2009 17:03:11.0374511	End Time 04/12/2009 17:34:25.7474511 04/12/2009 17:34:25.7474511 04/12/2009 17:04:18.0484511		
5832 vehonk. NetifyLaad 0.6/1/2/000 17:03:12.0064511 0.6/1/2/000 17:03:12.0064511 5834	Doub Sec 483 611 352 476	ID I	e list below to start a Intelli Trace I Name <thread ended=""> <no name=""> <no name=""> <no name=""></no></no></no></thread>	Debugging session for that thread Start Time 04/12/2009 17:03:09.5324511 04/12/2009 17:03:09.5334511 04/12/2009 17:03:11.0504511	End Time 04/12/2009 17:34:25.7474511 04/12/2009 17:34:25.7474511 04/12/2009 17:34:25.7474511 04/12/2009 17:34:25.7474511		
534 <hb 17.03.12.0114511<="" logo="" td=""> 04/12/2008 17.03.32.0164511 4168 vuback_norbinkright/mode 04/12/2008 17.15.357.5104511 04/12/2008 17.16.23.22.0164511 3504 .NET SystemEverts 04/12/2008 17.16.13.34511 04/12/2008 17.16.22.57474511 5522 Main Treed 04/12/2008 17.16.13.34511 04/12/2008 17.14.25.7474511</hb>	Doub Sec 483 611 355 476 816	ID I	In the line of the line o	Start Time 04/12/2009 17:03:05 5324511 04/12/2009 17:03:05 5344511 04/12/2009 17:03:11:0374511 04/12/2009 17:03:11:0374511 04/12/2009 17:03:11:0374511 04/12/2009 17:03:11:0374511 04/12/2009 17:03:11:0374511	End Time 04/12/2009 17:34/25 7474511 04/12/2009 17:34/25 7474511 04/12/2009 17:04/18 0484511 04/12/2009 17:34/25 7474511 04/12/2009 17:15:57.8414511		
4168 vahote.RunFakring/Whodow 04/12/0009 17:3657 3104511 04/12/2009 17:3425 27:47511 3504 .NET SystemEverts 04/12/2009 17:1601 1334511 04/12/2009 17:3425 7:47511 5632 Main Threed 04/12/2009 17:1601 1334511 04/12/2009 17:425 7:474511	Doub Sec 483 611 353 476 816 583	ID ID 36 16 2 64 6 32	e lat below to start a Intell Trace I Name Traced Ended> Traced Ended> CNo Name> CNo Name> CNo Name> vehost.Load/Reference vehost.NotifyLoad	Start Time 04/12/2009 17:03:09.5324511 04/12/2009 17:03:09.5324511 04/12/2009 17:03:10.374511 04/12/2009 17:03:11.0374511 04/12/2009 17:03:11.0374511 04/12/2009 17:03:11.2324511 04/12/2009 17:03:11.2324511 04/12/2009 17:03:11.2324511 04/12/2009 17:03:11.2324511 04/12/2009 17:03:11.2324511 04/12/2009 17:03:11.2324511 04/12/2009 17:03:12.0064511	End Time 04/12/2009 17:34:25.7474511 04/12/2009 17:34:25.7474511 04/12/2009 17:34:25.7474511 04/12/2009 17:34:25.7474511 04/12/2009 17:03:12.3174511		
3504 .NET SystemEvents 04/12/2009 17:16:01.1334511 04/12/2009 17:34:25.7474511 5632 Main Thread 04/12/2009 17:16:01.3384511 04/12/2009 17:34:25.7474511	Doub Sez 483 611 355 476 816 583 538	ID 36 15 2 64 6 32 84	e lat below to star a Intell Trace I Name < Threed Ended> <no name=""> <no name=""> vehost LoadReference vehost NotfyLoad <no name=""></no></no></no>	Start Time 04/12/2009 17:03:09.5324511 04/12/2009 17:03:09.5324511 04/12/2009 17:03:09.5324511 04/12/2009 17:03:11.0374511 04/12/2009 17:03:11.0374511 04/12/2009 17:03:11.0374511 04/12/2009 17:03:11.0374511 04/12/2009 17:03:11.0374511 04/12/2009 17:03:11.0374511 04/12/2009 17:03:12.0064511 04/12/2009 17:03:21.0114511	End Time 04/12/2009 17:34:25.7474511 04/12/2009 17:34:25.7474511 04/12/2009 17:34:25.7474511 04/12/2009 17:05:27.841511 04/12/2009 17:155:78.414511 04/12/2009 17:03:22.0145111 04/12/2009 17:03:22.0145111		
5632 Main Thread 04/12/2009 17:16:01.3984511 04/12/2009 17:34:25.7474511	2000 2000 2000 2000 2000 2000 2000 200	ele click on a thread in th arch 1D 36 52 54 53 54 58	e lat below to stat a Intell Trace I	Debugging session for that thread Start Time 04/12/2009 17:03:09 5324511 04/12/2009 17:03:09 53244511 04/12/2009 17:03:10 304511 04/12/2009 17:03:11 034511 04/12/2009 17:03:11 034511 04/12/2009 17:03:12 004511 04/12/2009 17:03:12 004511 04/12/2009 17:03:12 004511 04/12/2009 17:03:12 004511 04/12/2009 17:03:12 0114511 04/12/2009 17:05:75 0104511	End Time 04/12/2009 17:34:25:7474511 04/12/2009 17:34:25:7474511 04/12/2009 17:04:25:0474511 04/12/2009 17:03:12:3174511 04/12/2009 17:03:12:3174511 04/12/2009 17:03:12:3174511 04/12/2009 17:03:12:3174511		
	2000 2000 2000 2000 2000 2000 2000 200	ele click on a thread in th arch ID 36 16 2 6 6 6 6 32 2 84 88 04	e lat below to start a Intell Trace I Konger Stread Erded> Chread Erded> cho Name> cho Name> cho Name> cho Name> whont. LoadFeference whont. Notf, Load cho Name> whont. RuhPackingWindow NET SystemEverts	Start Time 04/12/2006 17/03/09 5324511 04/12/2006 17/03/09 5324511 04/12/2006 17/03/09 5324511 04/12/2006 17/03/10 30/54511 04/12/2006 17/03/11 03/54511 04/12/2006 17/03/11 03/54511 04/12/2006 17/03/11 03/54511 04/12/2006 17/03/11 03/54511 04/12/2006 17/03/11 03/54511 04/12/2006 17/03/11 03/54511 04/12/2006 17/03/11 03/54511 04/12/2006 17/03/12 00/54511 04/12/2006 17/03/12 00/54511 04/12/2006 17/03/12 00/54511 04/12/2006 17/03/12 00/54511 04/12/2006 17/03/12 00/54511 04/12/2006 17/03/12 00/54511 04/12/2006 17/03/12 00/54511 04/12/2006 17/03/12 00/54511 04/12/2006 17/03/12 00/54511 04/12/2006 17/03/12 00/54511 04/12/2006 17/03/12 00/54511 04/12/2006 17/03/14 01/54511 04/12/2006 17/03/14 01/54511 04/12/2006 17/03/14 01/54511 04/12/2006 17/03/14 01/54511 04/12/2006 17/03/14 01/54511 04/12/2006 17/03/14 01/54511 04/12/2006 17/03/14 01/54511 04/12/2006 17/03/14 01/54511 04/12/2006 17/03/14 01/54511<	End Time 04/12/2009 173-425 7474511 04/12/2009 173-425 7474511 04/12/2009 1770-118 0448511 04/12/2009 173-425 7474511 04/12/2009 173-125 774511 04/12/2009 173-425 7474511 04/12/2009 173-425 7474511 04/12/2009 173-425 7474511		
	483 611 355 476 816 583 416 350 563	Ue click on a thread in th arch 1D 35 16 2 2 44 65 82 84 86 84 86 84 86 84 86 84 84 85 84 85 84 85 84 85 85 85 85 85 85 85 85 85 85	e lat below to start a intelliTrace I Name <hr/> <hr/> <hr/> <hr> <hr/> <hr <="" td=""/><td>Start Time 04/12/2000 17:00:06 53:24511 04/12/2000 17:00:06 53:24511 04/12/2000 17:00:06 53:44511 04/12/2000 17:00:01 10:054511 04/12/2000 17:00:01 10:054511 04/12/2000 17:00:01 10:054511 04/12/2000 17:00:01 10:054511 04/12/2000 17:01:02:00:05411 04/12/2000 17:01:02:00:05411 04/12/2000 17:01:02:00:05411 04/12/2000 17:05:75 10:05411 04/12/2000 17:05:75 10:05411 04/12/2000 17:05:75 10:05411 04/12/2000 17:05:70 10:05411 04/12/2000 17:05:70 10:05411 04/12/2000 17:05:70 10:05411 04/12/2000 17:05:70 10:05411 04/12/2000 17:05:70 10:05411 04/12/2000 17:05:70 10:05411 04/12/2000 17:05:70 10:05411 04/12/2000 17:05:70 10:05411 04/12/2000 17:05:70 10:05411 04/12/2000 17:05:70 10:05411</td><td>End Time 04/12/2003 173-825 747511 04/12/2003 173-825 747511 04/12/2003 172-825 747511 04/12/2003 172-825 747511 04/12/2003 173-825 747511 04/12/2003 173-825 747511 04/12/2003 173-825 747511 04/12/2003 173-825 747511 04/12/2003 173-825 747511</td><td></td><td></td></hr>	Start Time 04/12/2000 17:00:06 53:24511 04/12/2000 17:00:06 53:24511 04/12/2000 17:00:06 53:44511 04/12/2000 17:00:01 10:054511 04/12/2000 17:00:01 10:054511 04/12/2000 17:00:01 10:054511 04/12/2000 17:00:01 10:054511 04/12/2000 17:01:02:00:05411 04/12/2000 17:01:02:00:05411 04/12/2000 17:01:02:00:05411 04/12/2000 17:05:75 10:05411 04/12/2000 17:05:75 10:05411 04/12/2000 17:05:75 10:05411 04/12/2000 17:05:70 10:05411 04/12/2000 17:05:70 10:05411 04/12/2000 17:05:70 10:05411 04/12/2000 17:05:70 10:05411 04/12/2000 17:05:70 10:05411 04/12/2000 17:05:70 10:05411 04/12/2000 17:05:70 10:05411 04/12/2000 17:05:70 10:05411 04/12/2000 17:05:70 10:05411 04/12/2000 17:05:70 10:05411	End Time 04/12/2003 173-825 747511 04/12/2003 173-825 747511 04/12/2003 172-825 747511 04/12/2003 172-825 747511 04/12/2003 173-825 747511 04/12/2003 173-825 747511 04/12/2003 173-825 747511 04/12/2003 173-825 747511 04/12/2003 173-825 747511		
 Exception Data 	A83 483 477 816 583 416 350 563 416 350 563	le click on a thread in th arch 36 16 2 64 64 63 22 84 84 84 84 84 84 84 82 82 84 84 85 85 84 84 85 85 86 86 86 86 86 86 86 86 86 86 86 86 86	e lat below to start a Intell Trace I Name Name	Debugging session for that thread Start Time 04/12/2009 17:00:09 534511 04/12/2009 17:00:09 5345451 04/12/2009 17:00:11:054511 04/12/2009 17:00:11:054511 04/12/2009 17:00:11:204511 04/12/2009 17:01:20114511 04/12/2009 17:15:57 5104511 04/12/2009 17:16:01 1334511 04/12/2009 17:16:01 1334511	End Time 04/12/2009 17:34:25:747511 04/12/2009 17:34:25:747511 04/12/2009 17:34:25:747511 04/12/2009 17:34:25:747511 04/12/2009 17:34:25:747511 04/12/2009 17:33:21:04511 04/12/2009 17:34:25:747511 04/12/2009 17:34:25:747511 04/12/2009 17:34:25:747511		
Exception Data	Doub Sez 483 611 350 476 816 583 416 350 563 563	le click on a thread in th arch 10 36 5 5 5 6 6 6 6 6 8 4 6 6 8 4 6 6 8 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2	e lat below to stat a intell'Trace I Name Chroad Ended> cho Name> cho Name> cho Name> cho Name> cho Name> cho Name> vahot LoadRerence vahot LoadRerence vahot Ref/chad cho Name> vahot Ref/chad vahot Ref/	Start Time 04/12/2000 17/0.306 5324511 04/12/2000 17/0.306 5324511 04/12/2000 17/0.306 5324511 04/12/2000 17/0.311 6304511 04/12/2000 17/0.311 6304511 04/12/2000 17/0.311 6304511 04/12/2000 17/0.311 6304511 04/12/2000 17/0.311 6304511 04/12/2000 17/0.312 0045411 04/12/2000 17/0.312 0045411 04/12/2000 17/0.513 1334511 04/12/2000 17/1.6101 1334511 04/12/2000 17/1.6101 3334511	End Time 04/12/2003 173-825 747511 04/12/2003 173-825 747511 04/12/2003 173-825 747511 04/12/2003 173-825 747511 04/12/2003 173-825 7474511 04/12/2003 173-825 7474511 04/12/2003 173-825 7474511 04/12/2003 173-425 7474511		
Exception Data Double click on an exception in the list below to start an Intell Trace debugging session from this exception.	A83 483 611 352 476 816 583 416 356 563 563 563 563 563 563	ID 36 15 2 24 2 25 4 6 2 24 32 25 4 32 32 32 32 32 32 Seption Data 32	In the list below to start an intellifyace I Name <the dended-<br="">Chio Name> chio Name> chio Name> chio Name> vahoat.LoadReference vahoat.LoadReference vahoat.NotfyLoad chio Name> vahoat.NotfyLoad chio Name> chio Name> vahoat.NotfyLoad chio Name> chio Name></the>	Debugging session for that thread Start Time 04/12/2008 17:03:06 55:324511 04/12/2008 17:03:06 55:324511 04/12/2008 17:03:11:0374511 04/12/2008 17:03:11:0374511 04/12/2008 17:03:11:0374511 04/12/2008 17:03:11:0374511 04/12/2008 17:03:12:00:4511 04/12/2008 17:03:12:00:4511 04/12/2008 17:05:71:06:511 04/12/2008 17:05:71:06:511 04/12/2008 17:06:13:334511 04/12/2009 17:06:13:334511 04/12/2009 17:06:13:334511 04/12/2009 17:06:13:334511 04/12/2009 17:06:13:334511	End Time 04/12/2003 173-825.747511 04/12/2003 173-825.747511 04/12/2003 173-825.747511 04/12/2003 173-825.747511 04/12/2003 173-825.747511 04/12/2003 173-825.747511 04/12/2003 173-825.747511 04/12/2003 173-825.747511 04/12/2003 173-825.747511 04/12/2003 173-825.747511		
Exception Data Double clok on an exception the list below to start an Intell Trace debugging session from this exception. Search X	Doub Sec 48: 61' 355 476 58: 538 416 356 56: 56: 56: 56: 56: 56: 56: 56: 56: 5	le click on a thread in th arch 10 36 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5	e lat below to stat a intell Trace I × Name Chread Ended> cho Name> cho Name> c	Debugging session for that thread Start Time 04/12/2009 17:03:09 53:24511 04/12/2009 17:03:09 53:44511 04/12/2009 17:03:11:03:4511 04/12/2009 17:03:11:05:4511 04/12/2009 17:03:11:05:4511 04/12/2009 17:03:11:05:4511 04/12/2009 17:03:12:01:4511 04/12/2009 17:03:12:01:4511 04/12/2009 17:05:12:01:4511 04/12/2009 17:05:12:01:4511 04/12/2009 17:05:12:01:4511 04/12/2009 17:05:13:05:4511 04/12/2009 17:06:13:13:4511 04/12/2009 17:06:13:15:57:10:6411 04/12/2009 17:06:13:15:57:10:6411 04/12/2009 17:16:01:13:4511 04/12/2009 17:16:01:13:4511 04/12/2009 17:16:01:13:4511 04/12/2009 17:16:01:13:4511 04/12/2009 17:16:01:13:4511 04/12/2009 17:16:01:13:4511 04/12/2009 17:16:01:13:4511 04/12/2009 17:16:01:13:4511	End Time 04/12/2009 17:34:25,747511 04/12/2009 17:34:25,747511 04/12/2009 17:04:18.0448511 04/12/2009 17:04:18.0448511 04/12/2009 17:15:57.8414511 04/12/2009 17:03:24:0547511 04/12/2009 17:03:24:0547511 04/12/2009 17:34:25,747511 04/12/2009 17:34:25,747511 04/12/2009 17:34:25,747511		
Exception Data Double click on an exception in the list below to start an intel® Trace debugging session from this exception. Search Type Message Thread Id HResult	Doub Sec 48: 61' 355 476 58: 538 416 356 56: 56: 56: 56: 56: 56: 56: 56: 56: 5	ID 10 36 10 36 2 4 2 5 32 24 32 25 2 26 32 29 20 29 20 20 22 20 24 21 24 22 25 20 24 21 24 22 25 24 26 25 26 26 26 27 27 28 29 29 20 201 20 202 20 203 20 204 20 205 20 205 20 205 20 205 20 205 20 205 20 205 20 205	e lat below to start a Intell Trace I	Start Time 04/12/2000 17:00:06 53:24511 04/12/2000 17:00:06 53:24511 04/12/2000 17:00:06 53:24511 04/12/2000 17:00:11:05:04511 04/12/2000 17:00:11:05:04511 04/12/2000 17:00:11:05:04511 04/12/2000 17:00:11:05:04511 04/12/2000 17:00:12:00:14511 04/12/2000 17:00:12:01:14511 04/12/2000 17:00:12:01:14511 04/12/2000 17:00:12:01:13:04511 04/12/2000 17:00:12:01:13:04511 04/12/2000 17:00:12:01:13:04511 04/12/2000 17:00:13:045451 04/12/2000 17:00:13:045451 04/12/2000 17:00:13:045451 04/12/2000 17:00:13:045451 04/12/2000 17:00:13:045451 04/12/2000 17:00:13:04551 04/12/2000 17:00:13:04551 04/12/2000 17:00:13:04551 04/12/2000 17:00:13:04551 04/12/2000 17:00:13:04511 04/12/2000 17:00:13:04511 04/12/2000 17:00:13:04511 04/12/2000 17:00:13:04511 04/12/2000 17:00:13:04511 04/12/2000 17:00:13:04511 04/12/2000 17:00:13:04511 04/12/2000 17:00:13:04511 04/12/2000 17:00:13:04511 0	End Time 04/12/2009 173-8425 747511 04/12/2009 173-8425 747511 04/12/2009 173-8425 747511 04/12/2009 173-845 747511 04/12/2009 173-845 7474511 04/12/2009 173-8425 747451 04/12/2009 173-8425 7475 04/12/2009 173-8425 7475 04/12/2009 1745 8425 7475 04/12/2009 1745 8425 8425 8425 8475 8425 8425		
Exception Data Double clok on an exception in the list below to start an Intell Trace debugging session from this exception. Search Type Message Thread M HRewit System UD Field/sTrundException Could not thin file "C-Valesandro bd", 552 0	Doub See 483 617 353 476 816 583 476 583 533 416 356 563 563 563 563 563 563 563 563 56	le cick on a thread in th arch ID ID S6 S6 S6	e lat below to start a intell Trace I	Start Time 04/12/2009 17:03:06 55:324511 04/12/2009 17:03:06 55:345511 04/12/2009 17:03:11:03:54511 04/12/2009 17:03:11:03:54511 04/12/2009 17:03:11:03:54511 04/12/2009 17:03:11:03:54511 04/12/2009 17:03:12:01:4511 04/12/2009 17:03:12:01:4511 04/12/2009 17:03:12:01:4511 04/12/2009 17:03:12:01:4511 04/12/2009 17:01:01:33:4511 04/12/2009 17:01:01:33:4511 04/12/2009 17:01:01:33:4511 04/12/2009 17:01:01:33:4511 04/12/2009 17:01:01:33:4511 04/12/2009 17:01:01:33:4511 04/12/2009 17:01:01:33:4511 04/12/2009 17:01:01:33:4511 04/12/2009 17:01:01:33:4511 04/12/2009 17:01:01:33:4511 04/12/2009 17:01:01:33:4511 04/12/2009 17:01:01:33:4511 04/12/2009 17:01:01:33:4511 04/12/2009 17:01:01:33:4511 04/12/2009 17:01:01:33:4511 04/12/2009 17:01:01:33:4511 04/12/2009 17:01:01:33:4511 04/12/2009 17:01:01:33:4511 04/12/2009 17:01:01:33:4511 04/12/2009 17:01:01:34:511 04/12/2009 17:01:01:34:511 <tr< td=""><td>End Time 04/12/2009 173-425.724511 04/12/2009 173-425.724511 04/12/2009 173-425.724511 04/12/2009 174-12009 173-425.72451 04/12/2009 173-425.72451 04/12/2009 173-32.0146511 04/12/2009 173-32.0146511 04/12/2009 173-425.724511 04/12/2009 173-425.724511 04/12/2009 173-425.724511 0</td><td></td><td></td></tr<>	End Time 04/12/2009 173-425.724511 04/12/2009 173-425.724511 04/12/2009 173-425.724511 04/12/2009 174-12009 173-425.72451 04/12/2009 173-425.72451 04/12/2009 173-32.0146511 04/12/2009 173-32.0146511 04/12/2009 173-425.724511 04/12/2009 173-425.724511 04/12/2009 173-425.724511 0		
	Doub Sec 483 611 352 476 816 583 533 416 353 563 563 563 563 563 563 563 563 56	ID and in the sarch in the sarc	e lat below to start a intell Trace I	Start Time 04/12/2000 17:00:06 53:24511 04/12/2000 17:00:06 53:24511 04/12/2000 17:00:06 53:24511 04/12/2000 17:00:05 10:05:0511 04/12/2000 17:00:05 10:05:0511 04/12/2000 17:00:05 11:05:0511 04/12/2000 17:00:05 11:05:0511 04/12/2000 17:05:12:00:05:11 04/12/2000 17:05:12:00:05:11 04/12/2000 17:05:12:00:05:11 04/12/2000 17:05:12:00:05:11 04/12/2000 17:05:12:00:05:11 04/12/2000 17:05:12:00:05:11 04/12/2000 17:05:13:00:05:11 04/12/2000 17:05:13:00:05:11 04/12/2000 17:05:13:00:05:11 04/12/2000 17:05:13:00:05:11 04/12/2000 17:05:13:00:05:11 04/12/2000 17:05:01:34:05:11 04/12/2000 17:05:01:34:05:11 04/12/2000 17:05:01:34:05:11 04/12/2000 17:05:01:34:01:10 04/12/2000 17:05:01:34:01:10 04/12/2000 17:05:01:34:01:10 04/12/2000 17:05:01:34:01:10 04/12/2000 17:05:01:34:01:10 04/12/2000 17:05:01:34:01:10 04/12/2000 17:05:01:34:01:10 04/12/2000 17:05:01:34:01:10 04/12/2000 17:05:01:34:01:10 04/12/2000 17:05:0	End Time 04/12/2003 173-425 7474511 04/12/2003 173-425 7474511 04/12/2003 173-425 7474511 04/12/2003 174-18 048511 04/12/2003 173-1557 8414511 04/12/2003 173-1557 8414511 04/12/2003 173-425 747451 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0		
Exception Data Double click on an exception in the list below to start an intell Trace debugging session from this exception. Search Type Type Message Thread Id HResult System:ID.Fielket/coundException Could not find file "C-Wessandro tot" Soack Frame:	Doub Sec 483 617 355 476 816 583 416 356 563 416 356 563 416 356 563 563 563 563 563 563 56	le cick on a thread in th arch ID ID S6 S6 S6	e lat below to start a intelliTrace I Name Trace I Mame The Mame Alto Name Alto Nat	Debugging session for that thread 04/12/2009 17:03:06 55:324511 04/12/2009 17:03:06 55:345411 04/12/2009 17:03:11:03:4511 04/12/2009 17:03:11:03:4511 04/12/2009 17:03:11:03:4511 04/12/2009 17:03:11:03:4511 04/12/2009 17:03:12:01:4511 04/12/2009 17:03:12:01:4511 04/12/2009 17:03:12:01:4511 04/12/2009 17:03:12:01:4511 04/12/2009 17:01:01:33:4511 04/12/12/100:01:01:01:01:01:0	End Time 04/12/2009 173-425.747511 04/12/2009 173-425.747511 04/12/2009 173-425.747511 04/12/2009 173-425.747511 04/12/2009 173-425.7474511 04/12/2009 173-425.7474511 04/12/2009 173-425.7474511 04/12/2009 173-425.7474511 04/12/2009 173-425.7474511 04/12/2009 173-425.7474511 0		
Cxception Data Double dok on an exception in the list below to data an itelia Trace debugging session from this exception. Soarch Type Type Message Thread M HReuit System ID FielderfoundException Could not find the C-Messando bit 5532	Doub Sec 48: 61: 53: 477 816 53: 476 53: 53: 416 350 55: 55: 50: 50: 50: 50: 50: 50: 50: 5	ID I	e lat below to stat a intell Trace I	Start Time 04/12/2000 17:03:06 53:24511 04/12/2000 17:03:06 53:24511 04/12/2000 17:03:06 53:24511 04/12/2000 17:03:01 65:04511 04/12/2000 17:03:11 65:04511 04/12/2000 17:03:11 65:04511 04/12/2000 17:03:12 00:04511 04/12/2000 17:03:12 00:04511 04/12/2000 17:03:12 00:04511 04/12/2000 17:05:12 01:04511 04/12/2000 17:16:01:13:34511 04/12/2000 17:16:01:13:34511 04/12/2000 17:16:01:33:4511 04/12/2000 17:16:01:33:4511 04/12/2000 17:16:01:33:4511 04/12/2000 17:16:01:33:4511 04/12/2000 17:16:01:33:4511 04/12/2000 17:16:01:33:4511 04/12/2000 17:16:01:33:4511 04/12/2000 17:16:01:33:4511 04/12/2000 17:16:01:33:4511 04/12/2000 17:16:01:33:4511 04/12/2000 17:16:01:33:4511 04/12/2000 17:16:01:33:4511 04/12/2000 17:16:01:33:4511 04/12/2000 17:16:01:33:4511 04/12/2000 17:16:01:33:4511 04/12/2000 17:16:01:33:4511 04/12/2000 17:16:01:33:4511 04/12/2000 17:16:01:33:4511 04/12/2000 17:16:01:33:4511	End Time 04/12/2003 173-8425 7474511 04/12/2003 173-8425 747451 0 0		

FIGURE 58.21 Analyzing IntelliTrace logs.

Using IntelliTrace for Unit Tests

IntelliTrace is not limited to tracking the application lifetime but can also be used for unit tests for understanding what is behind failures. This particular scenario is discussed in Chapter 59, "Testing Code with Unit Tests, Test-Driven Development, and Code Contracts," which provides an overview of unit testing and test-driven development.

Generating Dependency Graphs

Another interesting addition, new to VS 2010, is the Dependency Graph generation. Basically this feature enables generating a WPF-based, graphical, browsable view of dependencies between objects in your projects. Dependency graphs can be generated at assembly level (including all types), namespace level (including only types from a given namespace), or at class level. To demonstrate this feature, create a new console project and add the following items:

- ▶ An Entity Data Model mapping the Customers, Orders and Order_Details tables from the Northwind database (see Chapter 27, "Introducing the ADO.NET Entity Framework," for a recap).
- ▶ A new class named Helper, whose purpose is just offering a simple method returning a collection of order details based on the order identifier. The code for this class is shown in Listing 58.5.

LISTING 58.5 Demo Class to Be Mapped into the Graph

Now select the **Architecture**, **Generate Dependency Graph**, **By Assembly** command. After a few seconds the new graph will be available inside Visual Studio 2010. You can then expand items and check for their dependencies with other objects, as represented in Figure 58.22.

To understand items mapping you can check out the legend. To show complete dependencies information, right-click the graph and select the Show Advanced Selection command. This launches the Selection tool window where you can select one or more objects to be investigated and additional objects to be analyzed such as public or private properties. Figure 58.21 shows public properties dependencies from all classes in the project. For example, the NorthwindEntities class has dependencies with the Helper.GetOrderDetails method because this one receives an argument of type NorthwindEntities. The dependency graphs are flexible and can be exported to XPS documents or to images (right-click the graph for additional options).



FIGURE 58.22 The newly generated assembly-level dependency graph.

Summary

This chapter covered some important analysis tools available in Visual Studio 2010 Premium and Ultimate that are necessary for improving applications' quality. You discovered how to analyze code for compliance with Microsoft coding rules, required especially when you produce reusable class libraries. Then you saw how to check for code maintainability using the Code Metrics tool. In the second part of the chapter you got information on the integrated Profiler that simplifies checking for performance issues through IDE suggestions; you also saw different usages of the Profiler, including standalone executables. The chapter also focused on the most interesting addition in Visual Studio 2010, the IntelliTrace debugger. Explanations provided information on using this tool for understanding how to keep track of all application events and exceptions during the entire application lifetime, showing also how to analyze summary logs within Visual Studio. Finally, you got an overview of another new addition, the Dependency Graph generation that provides a graphical view of dependencies at assembly, namespace, and class levels.