# TOOLBOX

## Microsoft Chart Controls, Visual Studio Automatic Code Snippets, And More

Scott Mitchell

All prices confirmed at press time and are subject to change. The opinions expressed in this column are solely those of the author and do not necessarily reflect the opinions at Microsoft.

Contents



### A Chart Is Worth 1,000 Words

A well-designed chart offers a quick and clear synopsis of data and can reveal cycles, trends, or other patterns that are more difficult to discern from tables of numerical data alone. Until recently, adding charts to a Web page or a Windows client application required using an open-source or third-party charting package. (One such third-party charting component is Dundas Charts, which was reviewed in the January 2006 issue at msdn.microsoft.com/magazine/cc163668). Starting with Visual Studio 2008 and the Microsoft .NET Framework 3.5 SP1, developers can use Microsoft Chart Controls to design and display charts in both ASP.NET and Windows Forms applications.

To get started, download and install the Microsoft Chart Controls and the Chart Controls Add-On for Visual Studio. The first download installs the assemblies that contain the charting library and controls; the second adds Visual Studio Toolbox integration and IntelliSense support. (Note: The Microsoft Chart Controls require the .NET Framework 3.5 SP1; the Visual Studio Add-In is for the non-Express versions of Visual Studio 2008.)

Next, launch Visual Studio and open a Windows Forms or ASP.NET application and drag the Chart control from the Toolbox onto the Design surface. You can specify the chart's size, type, colors, series, legends, and other display-related criteria through its properties. The chart data can be specified statically or can come from a dynamic source such as an array, a database, an XML file, a CSV file, or a Microsoft Excel spreadsheet.

What impressed me most about the Microsoft Chart Controls are the breadth of its features. There are many different chart types available, from the standard line, bar, and pie charts to more exotic ones like radar, histogram, pyramid, Pareto, and bubble charts. The chart data can be grouped, sorted, filtered, and searched. With a single line of code, you can export the chart data to an XML file, or enhance your charts to include moving averages, range indicators, and trend lines. It's similarly easy to add markers, annotations, titles, legends, and other visuals to the chart surface. Furthermore, the chart's output can be customized by creating event handlers for Chart Controls events such as PrePaint and PostPaint.

The Microsoft Chart Controls also include a number of features for interactive charts. For example, when a user clicks on a data point, you can display detailed information about that data point in the same chart area or in a separate chart or grid. The Chart Controls for Windows Forms applications can be configured to allow the user to zoom or scroll through the chart data. And both versions of the Chart Controls support real-time charts, which are charts whose data is continually being generated and whose appearance is continually updated to display the new data. The Chart Controls for ASP.NET applications use the ASP.NET AJAX framework to automatically perform most of its interactive features, meaning that you don't have to be a JavaScript expert. However, you can execute custom JavaScript in response to a user's click, if needed.

There are some shortcomings of the Microsoft Chart Controls, which is to be expected with the first release of any software library. As noted earlier, there are separate downloads for the Chart Controls and the Visual Studio integration instead of a single, consolidated package. Getting the library's documentation and samples environment involves two additional downloads. And at this time, there is no support for Silverlight or Windows Presentation Foundation applications (although the Silverlight Toolkit includes its own Chart

control). Finally, there is no wizard to assist with creating and customizing the appearance of the chart. Granted, there are plenty of code samples available and the chart's properties are straightforward and easy to configure, but because there are so many appearance-related properties, it can take a while to unearth the property to change to, say, adjust the font size of the primary X-axis label. These shortcomings will hopefully be addressed in future releases.

**Price:** Free

go.microsoft.com/fwlink/?LinkId=142580

go.microsoft.com/fwlink/?LinkId=142581

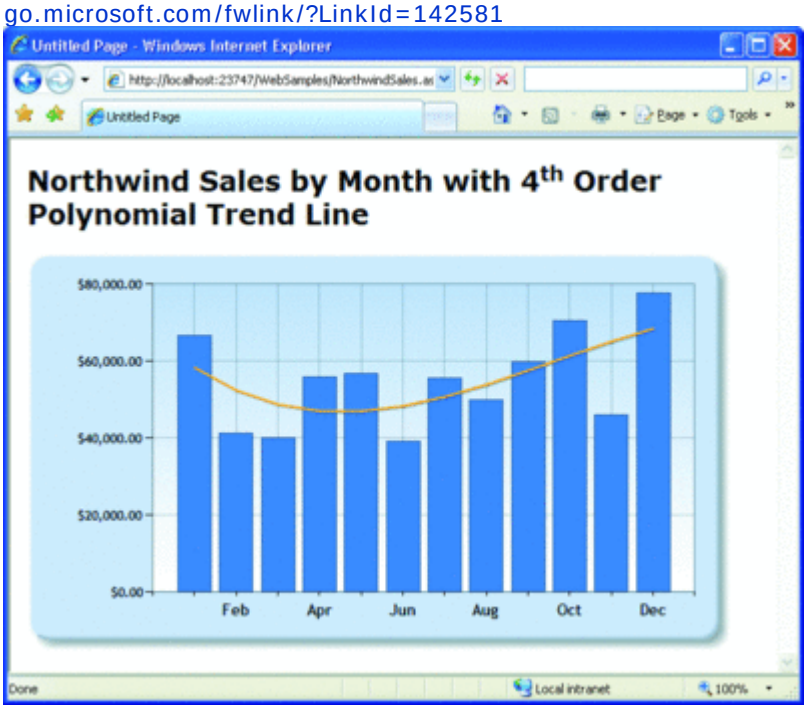



**Chart from Microsoft Chart Controls**(Click the image for a larger view)

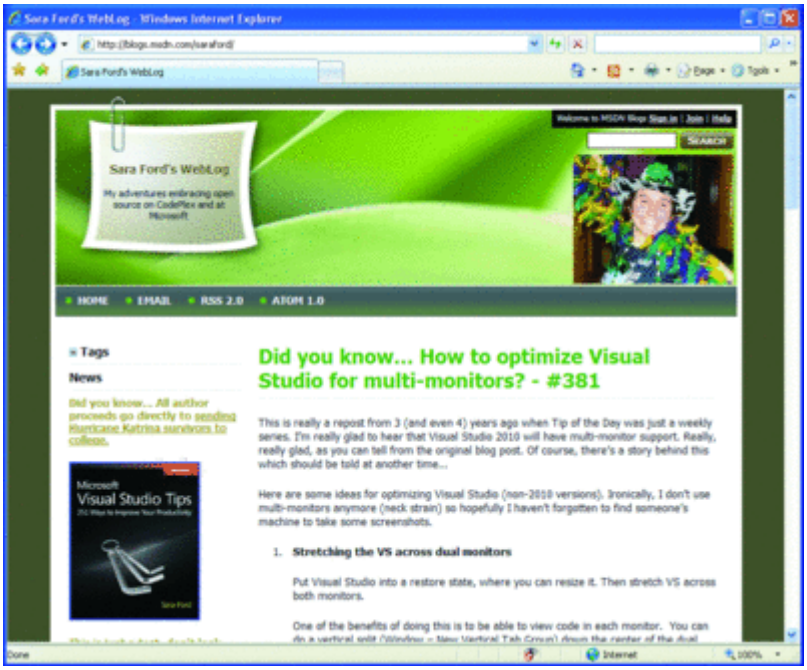
## Blogs of Note

Like most .NET developers, I spend most of my day in Visual Studio. Despite my familiarity with the tool, I am constantly learning about features, options, windows, and wizards that I had no idea existed. I discover some of these previously unknown features through trial and error, but much of my Visual Studio knowledge comes from blogs like the one by Sara Ford, a Program Manager for CodePlex.com. Over the years, she's compiled hundreds of Visual Studio tips and published them on her blog.

Each tip is a concise description of some little-known Visual Studio feature, often complete with a screen shot or two illustrating the tip in action. Some tips are simple keyboard shortcut tips, like Tip #346: You can press Ctrl+Shift+C to jump to the Class View. Others are tips on how to remove certain annoyances, like Tip #168: How to stop the Output window from showing itself during a build. And ones like Tip #239: You can set condition breakpoints, offer debugging tips and tricks.

The information in these tips is invaluable. Next time you have a free hour or two, surf over to Sara's blog and start poking through the tips. I guarantee you will learn something new. (After one and a half years of sharing her Visual Studio tips, Sara wrote her last tip—#382—in December 2008; her blog now focuses on CodePlex and the .NET community.)

blogs.msdn.com/saraford




**Sara Ford's Blog**(Click the image for a larger view)

## Create and Manage Code Snippets

Visual Studio includes a bevy of features that provide a surefire way to become a more productive programmer. One of the most powerful productivity enhancements is Code Snippets, which are task-oriented blocks of code that can include replaceable regions. For example, there are Code Snippets that generate LINQ query code, some that add the boilerplate code for a foreach loop, and others that write the code for working with text files. Adding a Code Snippet to a class file injects the Code Snippet's code and allows the developer to specify the text for any replacement regions.

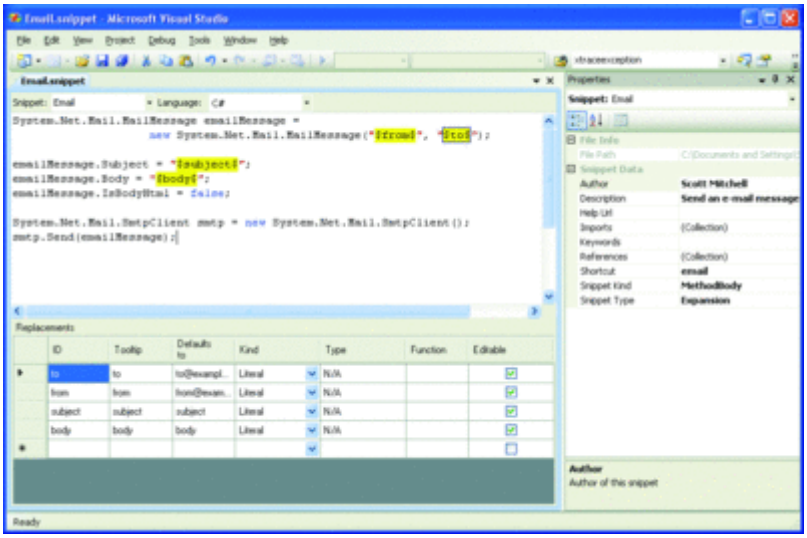
Visual Studio includes several dozen built-in Code Snippets (more for Visual Basic than for C#); you can also import Code Snippets or create your own from scratch. Unfortunately, Visual Studio does not offer any help in creating Code Snippets from scratch—you must type the Code Snippet's code and replaceable regions in a properly formatted XML file and then import the file into Visual Studio.

Creating Code Snippets is a lot easier when using Snippet Designer (version 1.1), a free, open-source Add-In for Visual Studio 2008 for creating and editing Code Snippets directly within the IDE. Once you install it, creating a new Code Snippet is as easy as going to the File menu and creating a new Code Snippet File. This opens a window in Visual Studio where you can type in the Code Snippet's code, specify any replacement regions at the click of a mouse, and customize the Code Snippet's information through the Properties window. Snippet Designer makes it easy to create a new snippet from existing code—select the code to use and then right-click and choose "Export as Snippet" from the context menu. This loads the Code Snippet interface and imports the selected code.

In addition to the integrated Code Snippet editor, Snippet Designer also includes a Snippet Explorer window that is more functional than Visual Studio's built-in Code Snippets Manager window. The Snippet Explorer window allows you to list all snippets in one or more specified languages and includes a textbox for filtering the results. Selecting a Code Snippet from the search results displays the Code Snippet's code. Right-click a Code Snippet and choose Open to edit the Code Snippet in the integrated Snippet Designer editor.

**Price:** Free

www.codeplex.com/SnippetDesigner



**Managing Code Snippets in Visual Studio**(Click the image for a larger view)

## The Bookshelf

Each time a new version of C# is released, so are a flurry of books that showcase the language's new features. These books usually do a good job of explaining why each new feature was added and offer examples on how to use it, but often the reader is left with some sinking questions: when does it make sense to use this new feature? How can I best use these new features to implement best practices? Are there hidden pitfalls when using these new features?

A great resource for getting to the bottom of these types of questions is the Effective Software Development series by Addison-Wesley. I was first introduced to this series by Bill Wagner's book, Effective C#. Written in 2004, Effective C# offers 50 detailed guidelines for making the most of C# version 1.0. Bill recently wrote a brand-new book, More Effective C#, which includes another 50 guidelines for writing better C# code using the language features added to C# versions 2.0 and 3.0. For example, there are 10 guidelines on using generics, eight on C# 3.0 language enhancements, and another nine on LINQ.

More Effective C# assumes the reader is already familiar with C# and language features like generics, LINQ, multithreading, and so on. You won't find any discussions on what generics are or the syntax for lambda expressions. Instead, the book states a specific guideline, such as "Distinguish between IEnumerable and IQueryable data sources," and then provides a deep and detailed discussion on the why of the guideline with illuminating prose and helpful code snippets. And each guideline stands on its own and can be read and digested without having first read other guidelines, meaning that you can jump around to what guidelines interest you the most or are the most relevant for your current projects.

The amount of information per page in this book is astounding. There are no callouts, no diagrams, no pictures, no superfluous anecdotes or humorous side stories, and little whitespace to break up the book's text. It's a lot of great information in a very terse and compact format.

More Effective C# is a must-read for experienced C# developers who are already familiar with C# language features such as generics, extension methods, lambda expressions, anonymous types, and LINQ, but who want guidance on how best to use these features.

**Price:** $44.99

www.informit.com/esds

Improving the responsiveness of a data-driven application often involves using SQL Profiler or other tools to determine which queries take the longest to execute, and then fixing those problem queries by adding new indexes or modifying existing ones. But sometimes indexes only provide a partial solution.

There may still be ways to squeeze out more performance by rewriting the SQL query itself, but where do you start?

There already exists a proven methodology for making small, incremental changes to the source code of an application—refactoring. Refactoring is the process of incrementally improving existing source code without changing the program's overall output or functionality and is a core practice of agile software development. While refactoring has traditionally been used with object-based languages such as C# and Visual Basic, its tenets can also apply to SQL. In Refactoring SQL Applications (O'Reilly, 2008), authors Stéphane Faroult and Pascal L'Hermite offer guidance on how to improve performance by retooling a data-driven application's existing SQL queries, views, user-defined functions (UDFs), and data-access logic.

Refactoring SQL Applications begins with a discussion on analyzing the impact of refactoring to determine whether it is worth doing. Such an analysis will answer questions like, "Are there multiple SQL queries executed because of a loop construct in the application?" and "Could multiple queries be packaged into a single stored procedure?" Unearthing the answers to these questions involves a deep understanding of the data-driven application that is gained, in part, by logging and scrutinizing the database activity.

There are five chapters with refactoring advice. The first one examines refactoring UDFs and views. Another chapter looks at refactoring SQL statements and provides guidance on how to streamline the FROM clause, tips on when to use subqueries, ways to simplify filtering conditions, and other optimizations. There's also a chapter on refactoring tasks that explores how to improve the way the data-driven applications interact with the database.

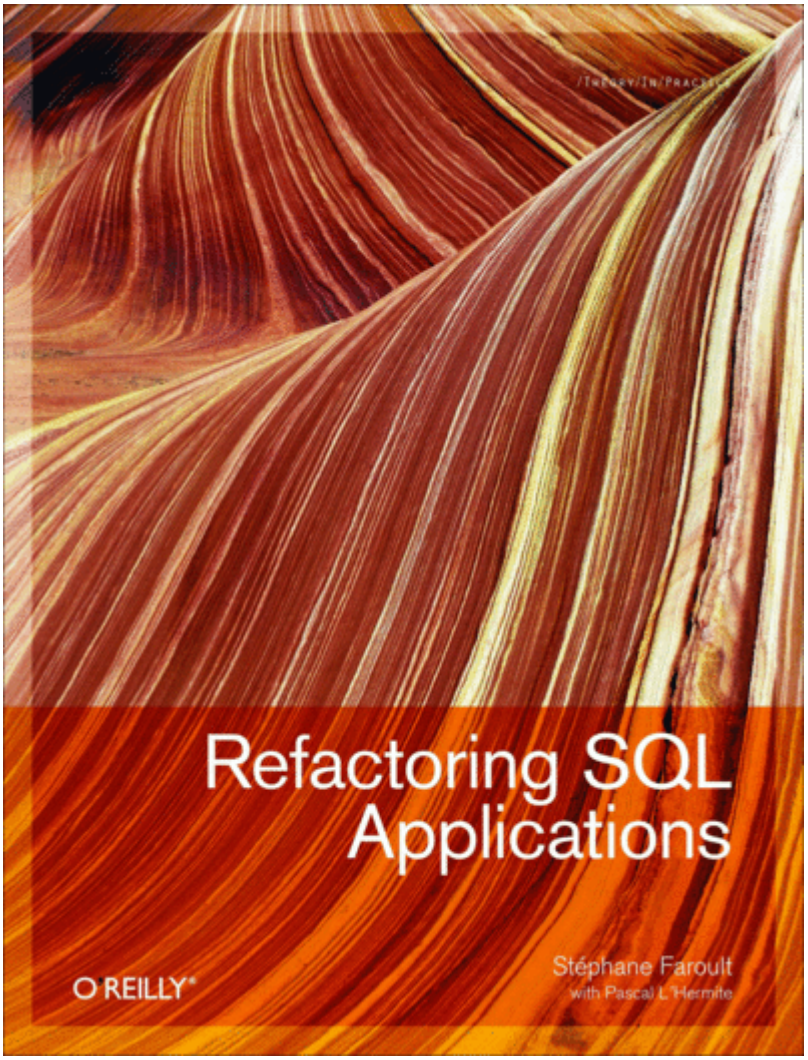
Large performance gains are possible by moving data access code outside of loops, using the database's aggregate functions when possible, and fetching all the data you need at once. There's another chapter on how to refactor SQL queries to best take advantage of parallelism. The book concludes with a chapter titled "Refactoring in Practice," which poses a number of questions to ask and items to investigate when tasked with refactoring a SQL application.

When refactoring code, it is imperative to ensure that the changes made while refactoring have not affected the program's existing functionality. Unit tests are commonly used to verify that refactored code still generates the expected output, but they are not always ideal for comparing database results.

The authors offer advice on how to create and use a custom testing framework when refactoring SQL queries. This discussion includes a look at how to generate large amounts of realistic test data and different techniques for verifying that the resultsets returned by the original query and the refactored query are equivalent.

**Price:** $44.99

oreilly.com




**Refactoring SQL Applications**

Send your questions and comments for Scott to toolsmm@microsoft.com.

**Scott Mitchell**, author of numerous books and founder of 4GuysFromRolla.com, is an MVP who has been working with Microsoft Web technologies since 1998. Scott is an independent consultant, trainer, and writer. Reach him at Mitchell@4guysfromrolla.com or via his blog at ScottOnWriting.NET