Two-Way Data Binding with WinJS

By Marcin Kawalerowicz and Craig Berntson, authors of *Continuous Integration in .NET*

One of the keys to improving applications and productivity is to automate some of the work. Continuous integration (CI) is one of the best ways to do this. In this article, based on chapter 1 of Continuous Integration in .NET, the authors introduce you to the tools that do the real work in continuous integration.

You may also be interested in…

A complete CI process consists of several tools. You can buy expensive CI systems that are feature rich and often easy to set up and maintain; or you can use tools that aren't as feature rich and often require some work to set up but are either free or low cost. Either way, no one tool does everything you need in a complete CI system. In this book, we'll work with free or low-cost tools and show you how they work and how to integrate them into a fully functional CI process. In this section, we'll give a brief introduction to several tools, starting with those that you must have.

Essential tools

Five tools are required to get started with CI. At a minimum, you should have these tools as part of your initial CI setup.

Source code control

The first essential tool is source control. Source control systems are most often used to store each revision of the source code so that you can go back to any previous version at any time. But you should also use the source control system to store customer notes, development documentation, developer and customer help files, test scripts, unit tests, install scripts, build scripts, and so on. In fact, every file used to develop, test, and deploy an application should be saved into source control. There's a debate in the developer community about whether this should include binaries that you can build; that decision is up to you and should be based on the needs of your team.

You have many source control options, ranging from high-end enterprise tools from IBM Telelogic that integrate with requirements and bug-reporting systems, to Visual SourceSafe (VSS) from Microsoft, which has been around for years. You can spend thousands of dollars on these tools or find ones like Subversion and Git that are open source and free. Even if you don't use CI, you should have source control, no matter the size of your team.

> **NOTE** Microsoft discontinued the aging and not well-respected VSS in early 2010 and replaced it with Team Foundation Server Basic. But many teams continue to use VSS and have no plans to change in the near future.

This book looks at mostly free tools from the Subversion family and mostly paid tools related to Microsoft Team Foundation Server (TFS). If you choose Subversion, make sure you also install another tool such as AnkhSVN (http://ankhsvn.open.collab.net/), VisualSVN (www.visualsvn.com/visualsvn/), or TortoiseSVN (http://tortoisesvn.tigris.org/) that integrates into Windows Explorer or Visual Studio and makes it easy to work

with Subversion. TortoiseSVN (see figure 1) seems to be the most popular (according to StackOverflow[1] and SuperUser[2]), so that's what we'll use for our examples.

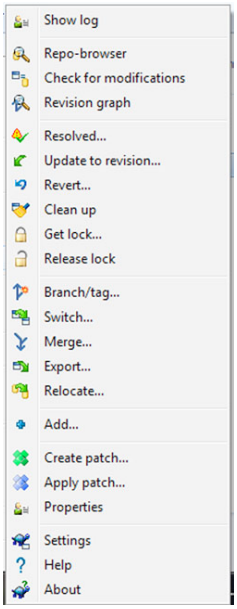


Figure 1 TortoiseSVN integrates into Windows Explorer to make it easy to manage your Subversion

If you're using TFS and have Visual Studio 2010 installed, you're ready to go.

Continuous integration server

The second and most important tool you need is one to drive your CI process. This sits on the CI server, watches for changes in the source code repository, and coordinates the other steps in the CI process. It also allows on-demand builds to be made. Essentially, this application is the traffic cop of any CI system. The CI server software typically has its own feedback mechanism that's used to aggregate feedback from the other tools and provide it to the feedback mechanism.

The most common CI tools for .NET development are Team Foundation Server from Microsoft and open source tools such as CruiseControl.NET and Hudson. TeamCity is another application that sits between these two options, because it's free for small teams but requires licensing fees as the size of the team or number of projects increase. Most CI tools are driven by a configuration file (see figure 2) that specifies when a build should take place and what specific steps are taken during the build or integration process.

[1] http://stackoverflow.com/questions/108/best-subversion-clients-for-windows-vista-64bit
[2] http://superuser.com/questions/33513/which-subversion-client-should-i-use

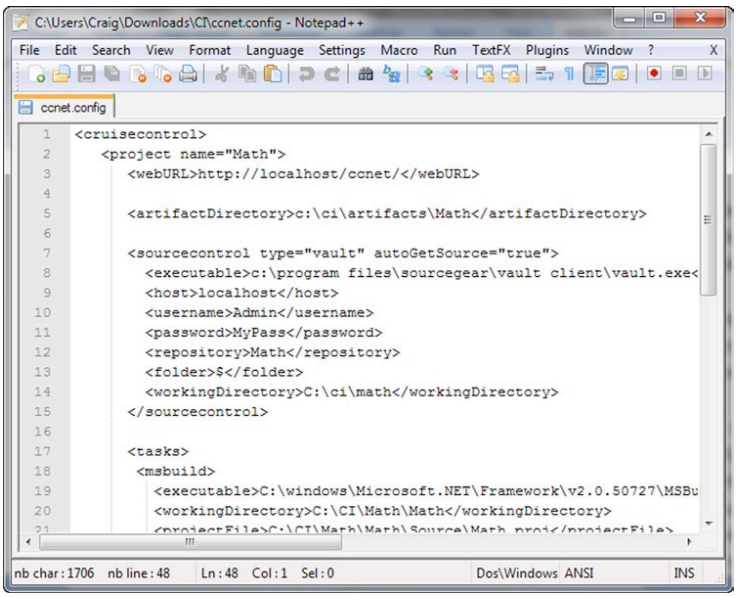


Figure 2 Part of the XML configuration file for CruiseControl.NET

Feedback mechanism

The feedback mechanism is another essential part of the CI process. Your team needs to know the status of any build at any time, especially when the build fails. There are many ways to provide feedback to the team. But the most common method is through a website.

Build manager

Next, you need something to do the actual build. The two most common options are MSBuild and NAnt. MSBuild is part of the .NET Framework, so it's free and most closely matches what happens when you click Build from the Visual Studio menu. NAnt is designed after the Java tool Ant. It's an open source solution, but it has received few updates in the past couple of years. Both applications are controlled by XML configuration files, but you can find GUI tools such as MSBuild Sidekick (see figure 3) to make the configuration files easier to maintain.

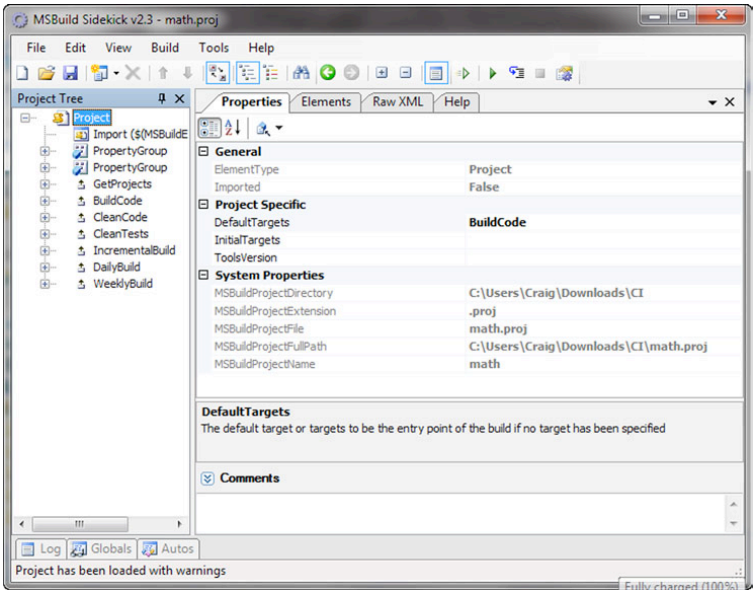


Figure 3 MSBuild Sidekick from Attrice makes it easy to develop and maintain MSBuild scripts.

The build-manager application takes a Visual Studio solution or individual project files and calls the correct compiler, generally C# or VB.NET. The compilers come free as part of the .NET Framework. Some shops use MSBuild for the actual compilation of the source and then use NAnt for the remaining steps, such as running unit tests.

Unit test framework

The last essential tool you need is a unit testing tool. The two most common options are MSTest and NUnit (see figure 4), but there are others such as MbUnit and xUnit.net. These tools run the unit tests that you write for your application and then generate the results into a text file. The text file can be picked up by your CI server software; a red/green condition for fail/succeed is reported to the team through the feedback mechanism.

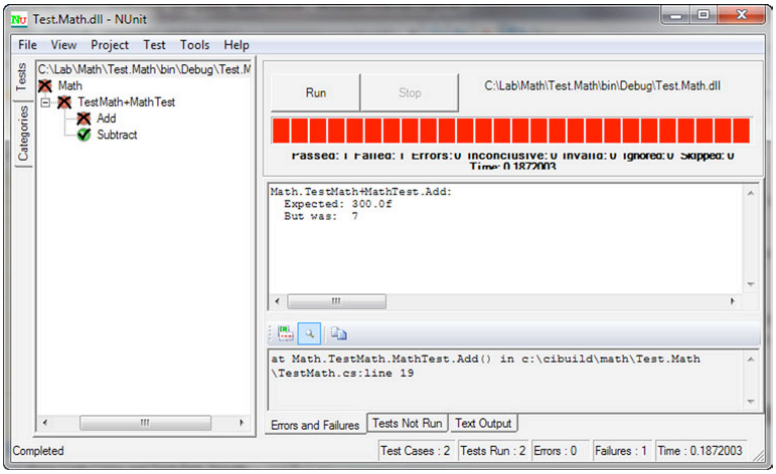


Figure 4 NUnit runs unit tests on your code and reports the results as red/green for failure or success.

Although NUnit has a GUI tool, it can also be run as a console application as part of your CI process. Many of the tools we'll discuss in this book have both a GUI and a command-line version. The command-line tools provide results as text or XML files that can be processed by your CI server software; the results are displayed using the feedback mechanism.

Now that you know the required tools, let's turn our attention to other tools that will help you write better code: code-analysis tools.

Code-analysis tools

Code analysis plays an important part in the development process. Code from multiple team members should use the same naming conventions. And the code should follow best practices so that it's robust, performant, extensible, and maintainable.

Several code-analysis tools can assist in the process. The first, FxCop (see figure 5), a free tool from Microsoft, analyzes code and reports possible issues with localization, security, design, and performance. The tool is targeted at developers creating components for other developers to use, but application teams are finding FxCop a useful part of their CI process.

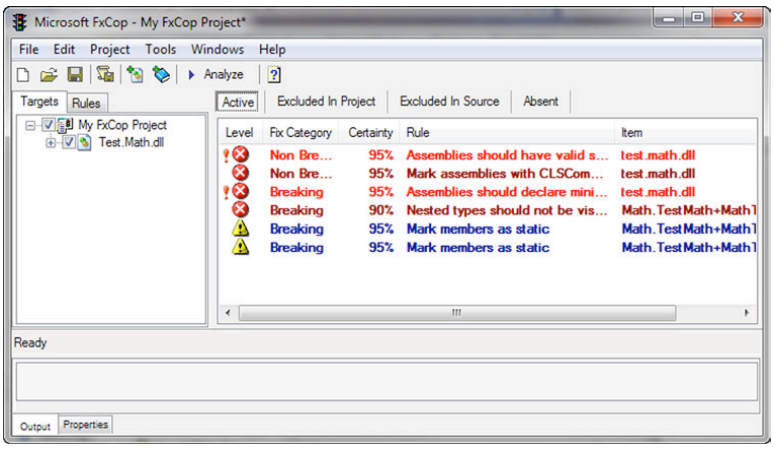


Figure 5 FxCop reports problems in code that can be issues with design, performance, security, or localization.

Another free Microsoft tool is StyleCop (see figure 6). It comes with Visual Studio and is delivered with a set of MSBuild plug-ins for standalone usage. This tool checks your code against best-practice coding standards. It compares your code to recommended coding styles in several areas including maintainability, readability, spacing, naming, layout, documentation, and ordering.

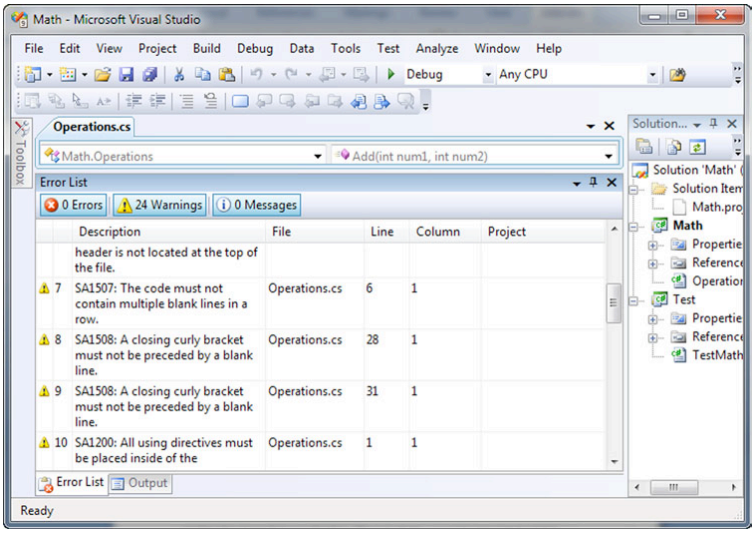


Figure 6 The StyleCop GUI integrates with Visual Studio and reports issues with coding style.

Both of these tools generate analysis reports that can be used by your CI server software and integrated into the build report available via the feedback mechanism. NCover (see figure 7) is a coverage tool that checks that all code paths are being tested. So is NCover an analysis tool or a testing tool? The truth is, it's a little of both.

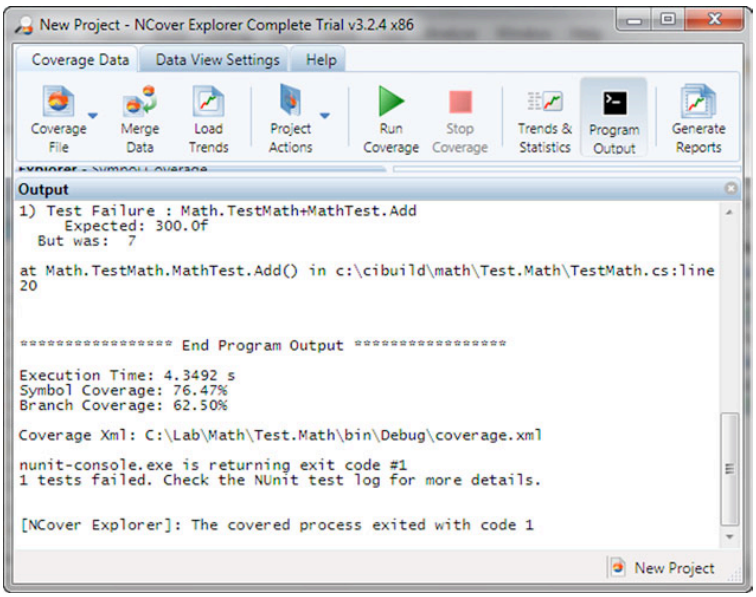


Figure 7 NCover reports the results of testing code paths through the application.

NCover uses either MSTest or NUnit to run the tests and can integrate with several CI server applications. But there are additional test tools, and they're the subject of the next section.

Testing tools

Unit testing tools are an essential part of the CI process. But other test tools can be run against your code to help ensure that the application functions correctly. One such tool is Selenium, an open source tool developed by ThoughtWorks. Selenium has record and playback capabilities for authoring tests that check web applications.

If you're creating WinForms, Windows Presentation Foundation (WPF) or Silverlight applications, you may be interested in White: it allows testing of your UI classes. Finally, there's FitNesse. This testing tool allows you to specify the functionality of the application; then tests are run against the code to ensure that it works as specified.

There are also several other tools you can add to your CI system.

Other tools

Have you ever put XML comments into your code? You can, and then extract them and compile them into useful documentation. That's the purpose of Sandcastle. These comments are most often added by component and framework vendors for building help files. But they can also be useful for other team members or even for yourself when you have to make changes to a module a month from now.

You can also automate the building of your deployment. It doesn't matter if you use ClickOnce, Visual Studio Installer, WiX, Inno Setup, or something else. Having your CI process automatically build the application, create the install set, and then test the install are big steps to ensuring a good, solid application.

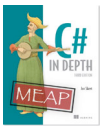
Summary

The tools presented here are by no means an exhaustive list. You can find many tools for things like code visualization, performance testing, static analysis, and more through a web search. Some of the tools cost several thousand dollars, and others are free. In this book, we take the low-cost approach and discuss tools that are free or available at a minimal cost. Tools like this emerge continuously in the community. To keep track of what's new and hot, you can check community sites like StackOverflow and ALT.NET (http://altdotnet.org/).

Here are some other Manning titles you might be interested in:

Dependency Injection in.NET
Mark Seemann

Windows Phone 7 in Action
Timothy Binkley-Jones, Massimo Perga and Michael Sync



C# in Depth, Third Edition
Jon Skeet

Last updated: 8/2/13