

DevSourceTM
Sponsored by **Microsoft**

Microsoft **Visual Studio**
Microsoft **Visual Studio.net**

Build
re r b st cod e

Download your Visual C++[®] Toolkit now [▶](#)

Testing Web Apps On a Shoestring Budget

May 6, 2004

By Dee-Ann LeBlanc

Not every software development company has a multi-million dollar budget with hundreds of programmers, testers, and other support staff. The majority of companies around the world are smaller shops. In North America, for instance, 52% of developers say that their company has ten or fewer programmers, according to the Evans Data Corp. North American Developer Survey. So, how can small shops manage the time and expense of developing quality products for Web delivery, when they can't afford dedicated quality assurance (QA) testers?

ADVERTISEMENT

Microsoft

Now Microsoft Visual Studio .NET makes it easier to make your big idea even bigger.

Try Visual Studio .NET for free now [▶](#)

Microsoft **Visual Studio.net**

In this article, I'll show you a number of best practices that Web developers can incorporate into their routines that will help you put out quality code. Good apps encourage people to buy products, after all, and thus you may grow the company to the point where professional QA testers can join the staff. Implementing some of these techniques will involve learning a bit about the QA field, but other suggestions just come down to making smart use of resources, and employing programming tactics commonly used to produce fast but solid code.

Smart Resource Allocation

To start with, one (half joking) suggestion from the QA community was to lay off a programmer and hire a QA tester, but this

may not be an option (especially if that programmer would be you). Another possibility is to hire a high school or college student "on the cheap" to run through the manual parts of the tests. Again, you may not have the budget to do this. If you have a technical writer doing documentation, this person could be a good choice to handle some of the QA load. It's amazing how many bugs tech writers can discover when writing up step-by-step instructions.

But you might have to stick with the programming staff you have on hand. For many smaller firms, if your programmers are not all assigned to the same project, one workable method is to assign programmers who are not coding a particular project to its testing. If your firm has all its programmers working on the same project, you can assign people to test components that they aren't working on.

The key here is that if you wrote the code, you know what it's *supposed* to do. This knowledge, on a gut level, makes it more difficult for you to properly test the code. No doubt, there are things that you didn't take into account, and it's these very things that people will immediately *try* to do with your Web application. Notice that this issue doesn't just focus on coding mistakes. For example, in Web applications, it's a common problem for the programmers to forget that ZIP and postal codes come in formats not used in their own country. If the person who forgot this "minor" issue is the one testing the application, he won't typically think to try a Canadian postal code. He might not even think to try the longer form American postal codes when entering test data.

Ultimately, at least one person at the company is going to have to learn some QA testing basics; there's no escaping it. However, these basics should be able to tide your team over until you can afford someone to do a more thorough job.

Programming Tactics

Before getting into QA techniques, it's worth taking a moment to talk about programming tactics. This might sound like we're shying away from the testing issue, but well-designed and -implemented apps are easier to test and to use. Do it right in the first place, and you won't have to do it over. Two popular methods are [eXtreme Programming \(XP\)](#) and [Test Driven Development](#). Both approaches emphasize building testing into the entire programming process, and so can cut down on the amount of testing that must be done at major milestones.

The XP process begins with collecting brief "user stories," which are basically short descriptions of tasks that the customers need the program to accomplish. From there, the developers get more information from the users, determine how long each story will take to develop as a concentrated effort, and then develop a release plan that includes iterations where the developers give the customers fully completed pieces of functionality for testing and other feedback. This cycle of constant give-and-take can result in well-tested software that highly meets the needs of your customers—especially since automated tests are built for each set of functionality along the way.

The Test Driven Development methodology is in some ways a refinement of eXtreme Programming, except that before you write any code, you write an automated test. Essentially, you create simple tests, one by one, and write code that will pass each of the tests. Once you can pass the test with whatever code you hacked together, you then turn around and refine the code to make it more efficient and better commented. As you go, you re-test, making sure you haven't broken anything in the process.

One advantage to Test Driven Development is that your tests are cumulative. As the program grows, you continue running every test that you created since the beginning of the project. Since making changes to anything computer-related tends to break something else, this is great way of making sure that you don't break previously good code when adding new features.

Both eXtreme Programming and Test Driven Development obviously need to be adopted at the beginning of a project in order to bring the greatest benefits. They also both require a serious change in how you approach your work. Your entire programming team, plus management, plus (at least some of) your customers—in the case of XP—will have to really buy in to this approach to make it work.

QA Techniques

Another aspect to putting out solid code is becoming familiar with the world of the Quality Assurance tester. While you don't need to earn a Ph.D. in this discipline, even small Web development teams can use key QA techniques to be absolutely sure that their code is tested consistently and thoroughly.

There are two major approaches to testing, each with its own advantages and drawbacks: using humans equipped with checklists, and using automated programs. A mix of both techniques is likely best in most situations, since a program can easily check some things (do all images have ALT tags assigned?) and other items are best suited for humans to pick out (is the background so "busy" that it's hard to read the text?).

The difficulty with automation is that someone has to write the tools in the first place, and many commercial offerings are expensive. Sites such as [Open Source Testing](#) are a budget-constricted developer's dream come true, offering everything from the [Web Worm Flooder](#) for rapid-fire crawling your site and filling out every form imaginable, and seeing what breaks, to the [Brute Force Binary Tester](#), which whacks at your Web application to see if can succeed in creating buffer overflows and other security problems.

When the humans enter the fray, you'll need to use a kind of automation to make sure that every test that needs to be run is done. This "automation" is managed in the form of checklists. Lots and lots of checklists. While it is important to include tests in those checklists specific to your application, you'll also find several generic ones freely available online for a base to build on.

Let's start with Web app usability. Many countries have a version of the United States' [Rehabilitation Act](#) which lays out the accessibility rules to ensure everyone can use various resources. WebAIM kindly offers a [checklist](#) to help you comply with these rules.

InfoDesign offers a set of [usability resources](#) for the look and feel part of your testing, along with [evaluation checklists](#) you can have people run through.

To learn more about testing, you'd do well to check out the [ApTest resources page](#), along with watching this space for more coverage of how you can make sure your Web applications are solid, dependable, and secure by integrating testing into your team's workflow, even if you're on one serious budget.

Copyright (c) 2004 Ziff Davis Media Inc. All Rights Reserved.