

A Look at Canoo WebTest

by Lisa Crispin

Agile teams need lightweight tools that allow them to deliver high-quality software and keep up with the pace of frequent code deliveries. One key Agile practice is to have the entire team, not just the testers, responsible for test automation. As a tester on an Agile team, I've found it challenging to find automated test tools for implementing "customer-facing" tests that our programmers would be willing and happy to use. This was especially tricky when we needed some way to test via our Web application user interface.

My team decided to try Canoo WebTest, an open source test tool designed to support the development of Web applications through test automation. It does this from a user's perspective, simulating a user's actions through its user interface. Test cases are specified as XML files, using "steps" with names that make obvious the function, such as invoke, clickLink, setCheckbox, verifyText, and verifyXPath.

Executing WebTest scripts produces easy-to-read HTML reports showing the success or failure of each test script in the suite, with a play-by-play description of each test execution. Based on Ant, a Java-based build tool, it is easily integrated into a build process as well as build automation tools such as CruiseControl. (See the StickyNotes for more on Ant and CruiseControl.)

We looked at several tools, both commercial capture/playback-type GUI test tools and open source tools. We didn't mind spending money on a tool. Developing and maintaining test scripts is what really costs money, not the initial purchase of the tool. The open source tools we considered were lighter weight than the commercial tools and seemed more suited to an Agile environment.

One reason we decided to try WebTest was its active developer and user community. With an open source tool, you can't call up Tech Support, so you need somewhere to turn for help. WebTest's users and developers post to

Test Summary for "GoogleTest"

Test started at: Thu Sep 29 16:37:50 MDT 2005
(C:\jane\test\src\web_test\SmokeTest\GoogleTest.xml:10.)
Test was successful!

Test Parameters

Executed Test Steps

No	Name	Parameter	Duration	Result
1	invoke	description "Go to main page" url "http://www.google.com"	2719	Passed
2	verifyText	description "Make sure we got there" text "I'm Feeling Lucky"	16	Passed
3	verifyTitle	description "check title" text "Google"	15	Passed
4	setInputField	description "set query" name "q" value "Lisa Crispin"	0	Passed
5	clickButton	description "submit query" label "Google Search"	344	Passed
6	verifyText	description "check for result" text "Agile Tester"	0	Passed

The inset image shows a Google search page with the search term "Lisa Crispin" and results for "Lisa Crispin, Agile Tester".

its mailing list frequently, and questions and problems are quickly addressed. The WebTest developer community even maintains a bug tracking system where users can report bugs—and they actually get fixed.

Canoo WebTest's Origins

Back in 2001, a team developing a Web-based application tested a tool for early acceptance testing from a user perspective—a tool that could keep pace with the speed of Agile development. After failed attempts using both a commercial tool and a homegrown tool, the team identified the concepts that led to the successful creation of the test tool:

- Tests should be specified rather than programmed. The specification should be simple to read yet formal enough for automation.
- It should be easy to run the same test against different environments (i.e., have some means to set and use parameters).
- Specifying tests shouldn't require inventing a new language.

Lots of the execution work should be able to be delegated. (HttpUnit was newly

available at that time.)

Because the team was using Ant for automating the project's continuous builds, they decided to specify tests as Ant tasks. Developing WebTest on their own time, they used Extreme Programming practices such as test-first development with both unit tests and acceptance tests. For the automated acceptance tests, of course, they used Canoo WebTest! Team members pair-programmed as much as possible, traveling to each others's locations after hours and on weekends.

Why the name "Canoo"? Several of the developers worked—or work—for Canoo Engineering AG in Basel, Switzerland. Canoo provides infrastructure and hosting services for Canoo WebTest and supports development of this open source tool in other ways.

The busy WebTest programmer community, which shares knowledge in the community forum <http://webtest-community.canoo.com>, has continued to make major changes and add important features. Some features that make WebTest a powerful test tool include support of regular expressions, XPath, dynamic properties, frame handling, HTTPS, and client certificate authentication. In 2004, WebTest was

converted to use HtmlUnit instead of HttpUnit, providing better support of client-side code such as Javascript. PDF support is another recent feature (and one we use a lot!). In addition to the bug tracking system, the Canoo WebTest mailing list offers a good source of ideas and troubleshooting techniques. Many developers contribute custom steps. The developers are even writing a book to help WebTest users.

The project for which Canoo WebTest was developed was successfully delivered and is still in production. Its suite of WebTests issues about 100,000 server roundtrips. That is, it invokes the Web application via a button, link, invoke, or other means and receives a response it can verify 100,000 times. More typical projects using Canoo WebTest have a size of about three to five person-years and 5,000 to 10,000 server roundtrips in the average test suite, which generally takes fewer than ten minutes to complete. Our own project is about this size, and our tests take barely more than ten minutes, which makes it possible to run them multiple times a day if needed.

Our WebTest Experience

Implementing Canoo WebTest in our environment was straightforward but not without bumps in the road. Our legacy system contained some poorly coded HTML and Javascript. Since WebTest is based on HtmlUnit, it likes to have clean HTML and Javascript code with which to work. This was painful with our old code, but it's yet another incentive to follow good practices in coding HTML and Javascript as we develop our new software.

Because WebTest scripts run as Ant targets, it was fairly easy to integrate WebTest into our CruiseControl nightly build process. Our tests have caught many regression failures. This was especially important when we were still building up our unit test suite. Most tools that test through the GUI are, by definition, brittle, but we've found that the WebTest XML files are fairly easy to maintain. Framework or code changes such as adding new fields will cause them to break, but when we plan changes that we think might break the tests, we plan for extra time to fix them. The formatted

results viewer, which presents the results in HTML format and allows you to drill down into the detailed test results, makes it easy to detect problems—whether actual bugs or script breakages.

My programmer co-workers help me debug WebTest scripts and support me by writing code that makes WebTest scripting easier. For example, they avoid Javascript except when needed for usability, and they provide a unique ID for each object in the HTML. They are even willing to occasionally write test scripts in WebTest themselves. If there's a test failure and I'm not around, it's easy for them to understand what broke, and they can make the necessary code changes.

How Does WebTest Work?

WebTests are specified in XML files, using a number of steps provided by WebTest. Typically, a script invokes a site via its URL and begins by clicking links and buttons and verifying the resulting pages. Many teams already use Ant to compile, build, and deploy their applications. Continuous builds are a key practice for successful development teams, especially Agile teams. WebTest scripts are initiated using Ant tasks, so they are easy to incorporate into a continuous build process. There are also Ant tasks for formatting the results in an easy-to-read HTML format. (See the StickyNotes to see what a simplistic test of Google.com might look like in a WebTest XML file.)

The WebTest Developer Community

The Canoo WebTest developer community continually impresses me. I've used quite a few commercial test tools in the past, but I can't say that I've experienced any better support than I get from these amazing developers, often working on their own time. Many of the developers are in Europe, so they often answer my questions or look at my bugs in what is their evening—still the workday for me. (I sometimes wonder when they sleep!) Besides, how often do you get any personal involvement from the developers of your test tool?

Two of the Canoo WebTest developers with whom I'm most familiar are Dierk

Koenig, one of the original WebTest creators, and Marc Guillemot, who recently converted WebTest from HttpUnit to HtmlUnit. I asked Marc and Dierk about WebTest users and what they foresee for the future of WebTest. (See the Sticky Notes for their responses.)

Should You Try Canoo WebTest?

If you're testing a Web application and must do some testing through the user interface, Canoo WebTest is worth a look. If you're a tester with programming experience and a fair amount of technical experience, and/or you've used tools such as Ant, you'll be able to try it out quickly and see if it meets your needs. If you're already using Ant and a tool such as CruiseControl for continuous builds, and/or your team is using Agile development practices, WebTest deserves a good look. If you aren't confident of your technical skills, you'll probably need some help from a programmer or system administrator teammate to give WebTest a try. **{end}**

Since 2000, Lisa Crispin has been a tester on Agile teams that develop Web-based applications. You can often find Lisa at Agile- and testing-related conferences, user group meetings, and seminars in the US and Europe, helping people discover good ways for Agile teams to do testing, and for testers to add value to Agile teams. She co-authored Testing Extreme Programming (Addison-Wesley, 2002) with Tip House. She contributes Agile testing articles to magazines and newsletters such as Better Software, Methods and Tools, Agile Times, and Novatica. Find out more about Lisa's work (and see pictures of her miniature donkey friends) at <http://lisa.crispin.home.att.net>.

Sticky Notes

For more on the following topics, go to www.StickyMinds.com/bettersoftware

- More on Ant and CruiseControl
- Test example
- Koenig and Guillemot on WebTest users and the future