

Class ACT?

WAS, Microsoft's simple and much-loved Microsoft load generator, has been replaced. Edward Garson, senior consultant for Dunstan Thomas Consulting, looks critically at the new tool



ACT ("Application Center Test"), distributed as part of Visual Studio.NET Enterprise, is a simple intuitive application which testers will learn to use very quickly – except when it comes to scripting.

Test creation

There are two ways to create ACT tests: firstly by recording a browser session and secondly by writing test code from scratch. Although recording a browser session has its merits (it can help determine the requisite code to implement some bit of functionality), the generated scripts are convoluted and poorly structured. ACT produces VBScript code in the course of recording a browser session; for users who prefer JavaScript this is a nuisance. Because of this authoring scripts from scratch is recommended.

Using ACT as a script-driven environment is tremendously flexible on one hand, but also mandates more work up-front to test anything less trivial than simple forms submission. It also requires that test authors get their hands dirty using a little VBScript or JavaScript. Thankfully, ACT comes with a number of well-documented test scripts and samples to get you started quickly. It is also reasonably easy to modify existing scripts in the course of creating new ones. The sample scripts are excellent examples of good programming principles, although they are all in VBScript which can be less elegant than JavaScript. As users become accustomed to the programming model their scripts will grow in sophistication.

Most users will almost certainly want to write scripts in an alternative environment such as Visual Studio.NET or Emacs as the ACT environment does not provide even syntax highlighting. VS.NET has a built-in interface to underlying ACT functionality which can be used to author scripts, even though it is developer-centric and primarily intended to be used in the course of building web-based applications.

Solid fundamentals

In the course of authoring scripts users should not dismiss good programming practices even though they are working in a scripting environment. Attention should be paid to basic programming principles to facilitate extending

and reusing scripts. Endeavour to keep routines short and focused on a single task and avoid repetitive code by writing subroutines which accept arguments. Where appropriate use constants and always comment your code. By sticking closely to these basic principles, code will be easier to manage in the long run.

Limitations of script

Unfortunately, one of the biggest drawbacks of VBScript and JavaScript is that there is no viable way to share code across physical files. This invalidates any effort to build reusable libraries, despite rudimentary support for classes in JavaScript. However, well-written routines can certainly be reused in terms of cut and paste, so users should strive to write generic routines that are not coupled to any given test.

Test properties dialog

After having authored a test adding it to the ACT test environment is the next step. Each test that is added to ACT appears as a node in the main tree view. Highlighting the node in the tree view causes the test script to be displayed in a pane on the right. Invoking the properties dialog for a given test brings up a window where basic test conditions may be stipulated.

The test properties dialog enables configuring basic attributes of ACT tests such as whether the test should run a preconfigured number of times or for a stipulated duration. This is useful in the context of employing multiple clients on a network to run tests. The number of simultaneously spawned connections may also be specified which should be configured to a reasonable value for the machine running the tests. User groups may also be set which ACT uses to perform authentication. The users available are drawn from those configured within the ACT environment. Finally performance counters may be added to the tests to provide additional information about how the test target is affected in particular ways. This latter feature draws on the Windows

Management Instrumentation (WMI) subsystem and provides a wealth of information from many different performance categories. Given that it is possible to write custom WMI code that emits instrumentation-related data from applications for interested listeners the sky is the limit.

Performance counters may be added to the machine performing the tests or the target machine. They can help determine the best settings to use to put load on the test target (eg by selectively adjusting the number of connections to create). They can also be added to the test target (if it is a Windows box). This may yield interesting information as to how best to optimize the machine under load. Note that ACT may be used to test any HTTP-enabled device, which includes configurations such as Apache running on a Linux box.

Test results

After running the tests, right-click the test node and select 'View Latest Results'. This displays a graphical representation of the load placed on the test target, with the number of requests served per second on the Y-axis across time along the X-axis. Information below the graph contains focused information organised coherently and is not over cluttered with extraneous information. The information displayed includes average requests per second, time to last byte, errors counts and network statistics. In short, all the information required to keep performance testing simple and intuitive. **PT**



Figure 1: ACT's test summary display