CHAPTER 5

Creating Test Utilities

On any automated testing project, if you have to write every line of code from scratch every time, you'll never finish. If you start early in the project to identify tasks that can be written and stored as utilities and used repeatedly, it will save you time and headaches. An additional benefit is that some of these utilities are usable on other projects. Remember from Chapter 1 that one of the goals of good software is reuse. Actually, the whole reason we are automating testing in the first place is to increase our testing capabilities with code and to avoid doing certain tests over and over manually, right? So, it makes sense to create pieces of code generic enough to be used more than once.

To create and use these utilities, you will need to learn to use a common programming structure called the procedure. You will also need to use a Visual Basic object called a Standard module in order to store your utilities. In this chapter, you will use these procedures and modules to begin a utility library. I will also introduce some ideas for practical utilities, for example, logging routines and an application-startup routine. Along the way, you will learn some additional techniques to round out your test, such as adding timing and using the SendKeys statement to send keystrokes to an application. After that, the sky is the limit; you can create many different kinds of utilities with the techniques you will learn in this chapter.

Objectives

By the end of this chapter, you will be able to:

• Explain why the creation of utilities is important on a testing project.

• Define the difference between a subroutine and function.

• Build simple test utilities using subroutines and functions.

• Describe the purpose and scope of a Visual Basic Standard module.

• Create and use a Visual Basic Standard module.

• Describe the difference between a Standard module and a Form module.
• Add a new Form module to a Visual Basic application.

• Explain the use of the Show and Hide methods for forms.

• Explain the use of the Load and Unload statements.

• Read from and write to a text file using the Microsoft Scripting Library.

• Explain the importance of logging on an automated testing project.

• Add code for simple timing to your test script.

• Use the SendKeys command to simulate user input.

Creating and Using Procedures

Many lines of code are useful enough to be used more than once, not just in one project but in others as well. Rather than cutting and pasting, it makes sense to write the steps down and put them in a library where they can be accessed or called repeatedly. After all, the idea behind automated testing is to avoid duplication of labor. This same philosophy can be applied to reusing code, why reinvent the wheel?

We call each discrete set of steps a procedure. There are two kinds of procedures: subroutine procedures and function procedures. We have already worked with a special-purpose subroutine procedure, the event procedure. The event procedures we have worked with have the keyword Sub in their first line. Sub is short for subroutine. The procedures we have worked with have been attached to a control or form in our Visual Basic project, for example, Public Sub Form_Load. This procedure runs just before the form displays so whether or not it runs depends on the form. Now, you will learn how to create your own generic procedures that are not attached to any specific object. These generic procedures will prove to be a very powerful way to expand the usefulness of your code. Before you do, though, you need to engage in some planning to make sure you are headed in the right direction.

Planning Your Procedures

Before actually sitting down and writing a bunch of routines, you need to do some planning. What exactly is the task you are trying to accomplish? How can you organize this task into manageable parts and write the code to implement it? Your initial task definition will come from the Test Plan and your Test Lead in the