Walkthrough: Creating and Running a Generic Test

Visual Studio 2015

This walkthrough will step you through the process of wrapping an executable file as a generic test, then running the test. You will start by creating an executable file using existing sample code. Next, you will create a new generic test and add the executable file to the generic test. Finally, you will run the test.

Note
For information about generic tests, see Creating an Automated Test That Runs an Executable Using Generic Tests.

In this walkthrough, you will complete the following procedures:

- Create a program, that is, an executable file, to wrap as a generic test. This program represents a third-party testing tool that produces pass or fail results for its exit or return value at program completion. See **Prepare the Walkthrough**.
- Create the generic test.
- Run the generic test.
- Pass command-line arguments when you run the generic test. For more information, see Run the generic test.
- Deploy a File When You Run the Generic Test.

Prerequisites

- Visual Studio Ultimate, Visual Studio Premium
- You must create the executable file EvenOdd.exe. To do this, use the source code in **Generic Test Sample** and follow the steps in the next procedure, "Prepare the Walkthrough."

Prepare the Walkthrough

To prepare the walkthrough

1. Create a project for a new Visual C# console application. In Name field of the New Project dialog box, type EvenOdd and then choose OK.

   The EvenOdd solution is displayed in Solution Explorer. It contains a project named EvenOdd.

2. Within the EvenOdd project, open the source-code file Program.cs.
3. Replace the code of the Program.cs file with the code in Generic Test Sample.
4. Build the solution.

This creates the program that you will wrap to create a generic test.

Create a Generic Test

To create a generic test

1. Right-click the EvenOdd solution, point to Add, and then choose New Project.

   The Add New Project dialog box is displayed.

2. In the Add New Project dialog box, expand Visual C# and then choose Test.
3. In the Templates pane, choose Unit Test Project and then choose OK. Accept the default name, for example, TestProject1.

Note
Coded UI test projects can also include generic tests.

4. Right-click the unit test project, point to Add, and then choose Generic Test.

   The template for a generic test is added to your test project and appears in the main editing window. The new generic test is given a default name, such as GenericTest1.GenericTest, and is displayed in Solution Explorer.

5. Under Specify an existing program (a test, test harness, or test adapter) to wrap as a generic test, indicate the path and file name of the EvenOdd.exe file.

Note
To determine this path, choose Options on the TOOLS menu and then choose Projects and Solutions. The path of the EvenOdd solution is displayed under Visual Studio projects location. The EvenOdd solution contains a folder for the EvenOdd project. Under the EvenOdd project folder, EvenOdd.exe resides under bin\Debug\.

6. This designation will look similar to the following sample:
7. C:\Documents and Settings\<your user name>\My Documents\Visual Studio 2015\Projects\EvenOdd\EvenOdd\bin\Debug\EvenOdd.exe.
8. Save the generic test.

   You have created a generic test that wraps EvenOdd.exe. This test has the following characteristics:

   - You can run the test from a command line.
   - The test returns a value of 0, for Passed, or 1, for Failed.
   - You can now run the generic test from Test Explorer. To run the test now, see Run the Generic Test.

Run the Generic Test

To run the generic test that you created

1. On the TEST menu, point to Windows and then choose Test Explorer.

   The Test Explorer is displayed.

2. On the BUILD menu, choose Build Solution.
3. In Test Explorer, select the generic test and choose Run.

   The EvenOdd executable file returns a value of 0 or 1 at random. Accordingly, when the generic test that wraps EvenOdd runs, it passes or fails depending on the number generated by EvenOdd. The Test Results window displays the result, either Passed or Failed.

Note
In general, a generic test passes when the executable it wraps returns a value of **0**, and fails if any other value is returned.

   You can also pass arguments to the EvenOdd executable file. A test based on EvenOdd.exe passes or fails depending on the supplied arguments. For a description of the values that EvenOdd.exe returns, see Generic Test Sample.

4. To pass an argument when you run the generic test, on the GenericTest1.generic page, type 12 on the line Command line arguments to pass to the generic test and run the test again.

   Because you passed an even number, the test passes. You can confirm this result in the Test Results window.

5. (Optional) Run the test additional times, passing in different values.

   Some existing tests or executable programs require additional files when they run. You can specify files to deploy along with a generic test. For more information, see Deploy a File When You Run the Generic Test.

Deploy a File When You Run the Generic Test

To deploy an additional file when you run the generic test

1. Create and save a file that is named mydeployedfile.txt. The file can be empty. Note the folder in which you created it.
2. Under Additional files to deploy with this generic test, choose Add.

The Add Deployment Files dialog box is displayed.

3. In the Add Deployment Files dialog box, under Files of type, choose All files(*.*).
4. Navigate to the folder that contains mydeployedfile.txt, select the file, and then choose Open.

   You have specified mydeployedfile.txt to be deployed when you run the generic test.

   In the following steps, you can verify that the file is being deployed. You do this by using a specific feature of EvenOdd.exe that produces a Passed or Failed result depending on the presence or absence of the file that you specified.

5. In the generic test, change the value of the Command-line arguments to pass to the generic test run setting to: 12 "%TestDeploymentDir%\mydeployedfile.txt"
6. Save the generic test.

   Two command-line arguments are now passed to the generic test. This change causes EvenOdd.exe to use a different criterion to produce a result of Pass or Fail. When you pass two arguments, the first argument is ignored. If the file specified by the second argument exists in the same directory as the test, the test passes. However, if the file was not deployed or if the name of the file specified in the command-line argument does not match the name of the deployed file, the test fails.

   To run the test now, see Run the Generic Test.

Sample Code

```
using System;
using System.Globalization;
using System.IO;

namespace EvenOdd
{
    class TestSecondsOrNumbersOrFiles
    {
    /* Purpose: Wrap this sample app to create a generic test that passes or fails.

    When you run the EvenOdd app, it exhibits the following Pass/Fail behavior:
    * Pass zero arguments: EvenOdd randomly returns 1 (Fail) or 0 (Pass).
    * Pass one (integer) argument: EvenOdd returns 1 if the argument is odd, 0 if even.
    * Pass 2 arguments: EvenOdd ignores the first argument and uses only the second one, a string.
    If the file named by that string has been deployed, EvenOdd returns 0(Pass); otherwise 1(Fail).
        */

        [STAThread]
        public static int Main(string[] args)
        {
            // If no argument was supplied, test whether the value of Second is even.
            if (args.Length == 0)
                return TestNumber(DateTime.Now.Second);

            // If only a single numeric (integer) argument was supplied,
            // test whether the argument is even.
            if (args.Length == 1)
            {
                try
                {
                    int num = Int32.Parse(args[0], CultureInfo.InvariantCulture);
                    return TestNumber(num);
                }
                // catch non-integer argument for args[0]
                catch (FormatException)
                {
                    Console.WriteLine("Please type an integer.");
                    return 1;
                }
                // catch too-large integer argument for args[0]
                catch (OverflowException)
                {
                    Console.WriteLine("Type an integer whose value is between {0} and {1}.",
int.MinValue, int.MaxValue);
                    return 1;
                }
            }
            // If two arguments are supplied, the test passes if the second
            // argument is the name of a file that has been deployed.
            if (args.Length == 2)
            {
                if (File.Exists(args[1]))
                    return 0;
            }
            // Test fails for all other cases
            return 1;
        }

        public static int TestNumber(int arg)
        {
            return arg % 2;
        }
    }
}
```