The NUnit-Testlink Adapter

Author: Stephan Meyn
Date: March 25th, 2009

Copyright Notice

About

The NUnit Testlink Adapter (NTA) allows the automated exporting of test run results into Testlink.

NUnit is a unit-testing framework for all .Net languages. Initially ported from JUnit, the current production release, version 2.4, is the fifth major release of this xUnit based unit testing tool for Microsoft .NET. It is written entirely in C# and has been completely redesigned to take advantage of many .NET language features, for example custom attributes and other reflection related capabilities. NUnit brings xUnit to all .NET languages.

NUnit also includes its own command-line runner, Echo, and a Windows GUI.

To find out more about NUnit go to http://www.NUnit.org

TestLink enables easily to create and manage Test cases as well as organize them into Test plans. These Test plans allow team members to execute Test cases and track test results dynamically, generate reports, trace software requirements, prioritize and assign tasks.

To find out more about Testlink go to www.teamst.org

A Quick Run Through

(this assumes you are comfortable with unit test harnesses).

To make use of the NTA, you must first add a new attribute to each TestFixture. This attribute, called **TestLinkFixture** takes parameters that tell the adapter where to find the test cases to record results to.

Typically the tests are run as part of a build script or a batch file. The GTA is an extension to the command line runner.

When the tests have been run, the extension is called with the test results.

The extension parses the results, loads the test assemblies that have been run and extracts the test link fixture. It then contacts Testlink to look for test cases with the same name as they test methods in the unit test.

If the test case does not exist, GTA will create them and assign them to the Testplan.

It then records the test result against that test case.

Installation

Run Time Files

Copy all the DLLs of the ZIP files into the **NUnit addins** folder:

- CookComputing.XmlRpcV2.dll
- Log4Net.dll
- NunitTestLinkExporter.dll
- TestLinkAPI.dll
- TestLinkFixture.DLL
- TlNUnitAddOn.dll

Invocation

In this example the console runner is used:

NUnit-console.exe *[unitTestAssembly]*

TestLink Setup

In order for this to work you need to have in testlink:

a) Enabled the API (see installation manual)
b) Created an account for testing
c) Created an ApiKey for this account
d) Setup a test project
e) Set up test suite(s) (top level test suite) for the test cases
f) In the test project set up a test plan
g) Optionally created test cases in the test suite and assigned to the test plan

h) Have an active build defined in the testplan

Step g really is optionally because:

a) You must make sure you name the test cases exactly as they are defined in your unit tests
b) If you use advanced features such as contract verifiers and row tests in MBUnit you have to guess what these test cases will be named because they are dynamically generated
c) If you don't create test cases, ATG will do it for you – much easier

Unit Test setup

Write your test cases as you are used to.

Then have the project reference TestLinkFixture.dll and add a Using Statement for it:

```
using Meyn.TestLink;
```

Add the **TestLinkFixture** to your testfixture.

The parameters are:

- URL of the TestLink Api
- UserName under which the test cases are authored
- An ApiKey (provided by the testlink application)
- The test project name
- The test plan in the test project
- The test suite name that will contain the test cases

Example

```
using MbUnit.Framework;
using Meyn.TestLink;

namespace NUnitAddOnSampleTests
{
    [TestFixture]
    [TestLinkFixture(
        Url = "http://localhost/testlink/lib/api/xmlrpc.php",
        ProjectName = "TestLinkApi",
        UserId = "admin",
        TestPlan = "Automatic Testing",
        TestSuite = "NUnitAddOnSampleTests",
        DevKey = "ae28ffa45712a041fa0b31dfacb75e29")]
    public class SampleFixture
    {
        [Test]
        public void Test2Succeed()
        {
            Assert.IsTrue(true);
        }
    }
}
```

Voila.

This will create a test case named Test2Succeed and record test case result against it.

Limitations

This has been tested with NUnit V2.4.8, C# and Testlink V 1.8RC5.

It should work with VB.NET.

You use this code at your own risk.