Testing using UFT Pro (LeanFT) and NUnit in C#

Overview

In this tutorial, we will create a NUnit Test Case in C#, using UFT Pro (LeanFT) for browser automation.

Learn more

UFT Pro (LeanFT) is an advanced functional test automation solution that was designed specifically for continuous integration and continuous testing. It provides tools and capabilities that easily and efficiently create robust test automation in the developer IDE and integrate it seamlessly into the CI process. LeanFT enables automating applications in a wide range of technologies, including web, mobile and desktop.

Description

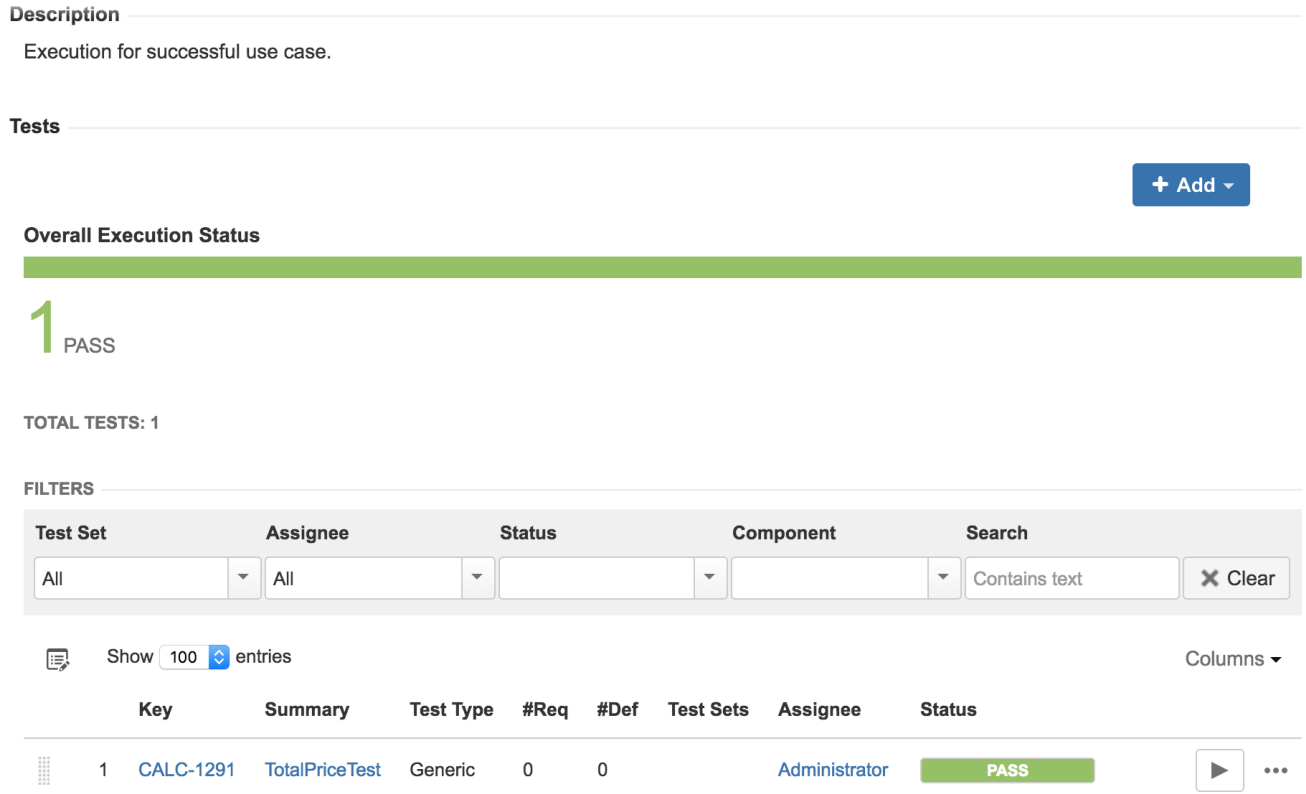
The following automated test uses LeanFT library in order to navigate through a website and validate the price shown for a product versus the one presented when it was added to the shopping cart.

LeanFtTest.cs

```
using System;
using NUnit.Framework;
using HP.LFT.SDK;
using HP.LFT.SDK.Web;
using HP.LFT.Verifications;

namespace LeanFT_Demo
{
    [TestFixture]
    public class LeanFtTest : UnitTestClassBase
    {
        [TestFixtureSetUp]
        public void TestFixtureSetUp()
        {
            // Setup once per fixture
        }

        [SetUp]
        public void SetUp()
        {
            // Before each test
        }

        [Test]
        public void TotalPriceTest()
        {
            //Launch Chrome and navigate to the online store application
                IBrowser browser = BrowserFactory.Launch(BrowserType.Chrome);
                browser.Navigate("http://www.advantageonlineshopping.com");
```

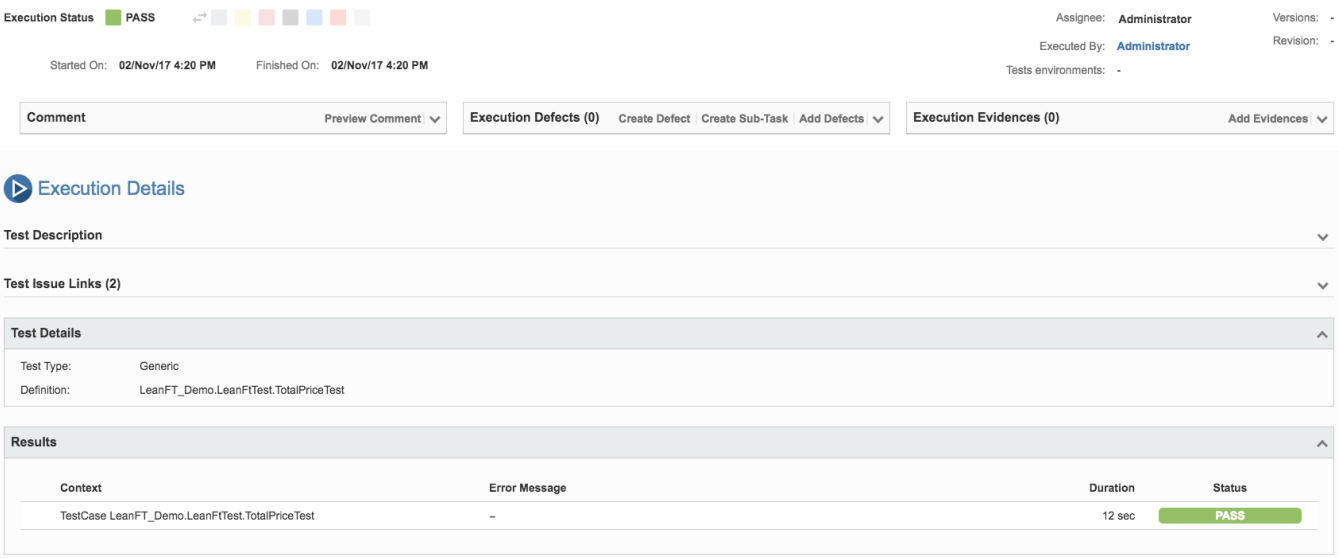
```
            //Click the "Tablets" category
            ILink tabletsLink = browser.Describe<ILink>(new LinkDescription { Id = @"TabletsImg" });
            tabletsLink.DisplayName = "Tablets";
            tabletsLink.Click();

            //Click a specific tablet
            IImage tabletElitePad = browser.Describe<IImage>(new ImageDescription
            {
                Src = @"http://www.advantageonlineshopping.com/catalog/fetchImage?image_id=3100",
                ClassName = @"imgProduct"
            });
            tabletElitePad.DisplayName = "Tablet ElitePad";
            tabletElitePad.Click();

                //Add it to the cart
            browser.Describe<IButton>(new ButtonDescription { Name = @"ADD TO CART"        }).Click();

                //Store its price
            String tabletPrice = browser.Describe<IWebElement>(new WebElementDescription { ClassName = @"roboto-
medium cart-total ng-binding"}).InnerText;

                //Check out
                browser.Describe<IButton>(new ButtonDescription { ClassName = @"roboto-medium ng-binding"}).
Click();

            //Verify that the total price presented in the purchase summary page, is exactly the price of the
selected tablet
            String totalPrice = browser.Describe<IWebElement>(new WebElementDescription {
                                    ClassName = @"roboto-medium totalValue ng-binding",
                                    InnerText = As.RegExp(@"\$.*")
                                }).InnerText;

                Verify.AreEqual(tabletPrice, totalPrice, "Verify total price");
        }

        [TearDown]
        public void TearDown()
        {
            // Clean up after each test
        }

        [TestFixtureTearDown]
        public void TestFixtureTearDown()
        {
            // Clean up once per fixture
        }
    }
}
```

After successfully running the Test Case and generating the NUnit XML report (e.g., results.xml), it can be imported to Xray via the REST API or the **Import Execution Results** action within the Test Execution.

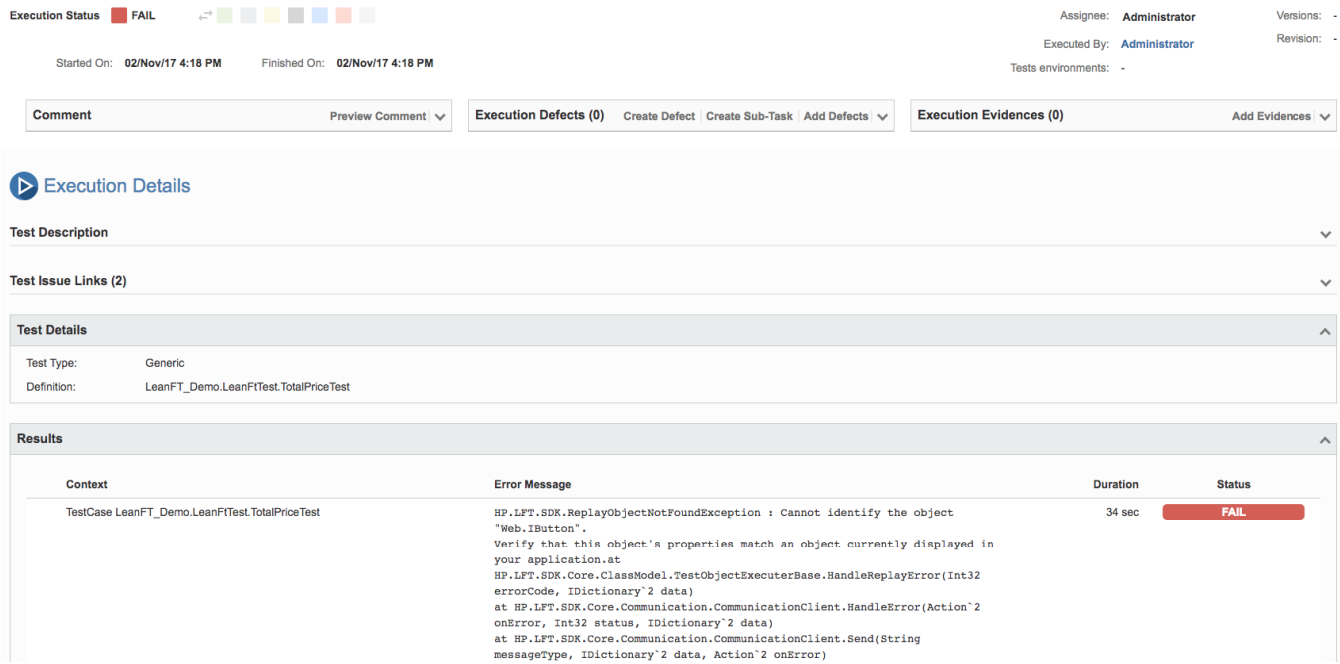


NUnit's Test Case is mapped to a Generic Test in Jira, and the **Generic Test Definition** field contains the name of the namespace, class, and the method name that implements the Test case.

The Execution Details of the Generic Test contains information about the context, which in this case corresponds to "TestCase" followed by the name of the namespace, class, and the method name that implements the Test case.



If the Test fails, for example, due to a missing web element (e.g., results.xml), then you will see the overall Test Run being marked as FAIL as well as the detailed information on the exception that was raised during the execution of the automated test.



Note that if you're using LeanFT's "Verify" method, that verification won't raise an exception by itself. The Test will appear as passed (if it didn't fail until then) even if the verification itself failed.

The "Verify" class' method returns a Boolean value reflecting the verification result. If it is “false”, it is possible to manually throw an exception to make the test status reflect the actual verification result.

References

- https://software.microfocus.com/ja-jp/software/leanft