How to: Check UIElement Properties with Coded UI Test

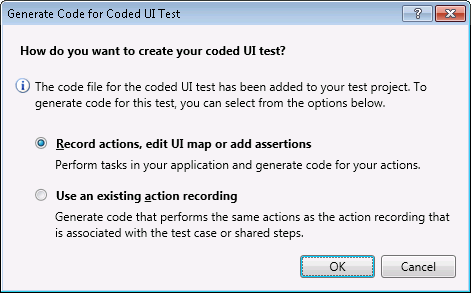
* 1. The following How-to topic will walk you through creating an automated test that checks for **UIElement** properties in a web application by using Visual Studio 2010 Coded UI Test. The Coded UI test performs actions on the user interface (UI) controls and verifies that the **UIElement** properties are displayed with the correct values. For this topic, the Mileage Stats Reference Implementation (Mileage Stats) will be the targeted application used for testing.
  2. The automated test created in this topic will navigate to the Mileage Stats Home page and verify that the images contain an **Alt** property with the expected values. This is important to ensure that the site is accessible and usable to all visitors.

# Prerequisites

* 1. This topic requires you to have the same prerequisites as Mileage Stats:
  + Microsoft Visual Studio 2010 Professional, Premium, or Ultimate edition
  + [Microsoft Visual Studio 2010 SP1](http://www.microsoft.com/downloads/en/details.aspx?FamilyID=75568aa6-8107-475d-948a-ef22627e57a5&displaylang=en)
  + Microsoft .NET Framework 4.0 (installed with Visual Studio 2010)
  + [ASP.NET MVC 3](http://www.asp.net/mvc/mvc3)
  + [Microsoft SQL Server Compact 4.0](http://www.microsoft.com/downloads/en/details.aspx?FamilyID=033cfb76-5382-44fb-bc7e-b3c8174832e2)
  + [ADO.NET Entity Framework 4.1](http://www.microsoft.com/downloads/en/details.aspx?FamilyID=b41c728e-9b4f-4331-a1a8-537d16c6acdf&displaylang=en)
  + [NuGet](http://nuget.org/)
  1. It is assumed that the Mileage Stats debug model web application has been deployed to an Internet Information Services (IIS) server, and that the test site is <http://localhost/mileagestats>.

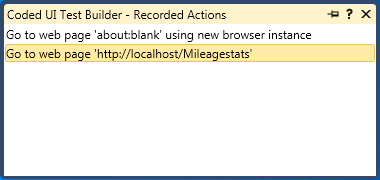
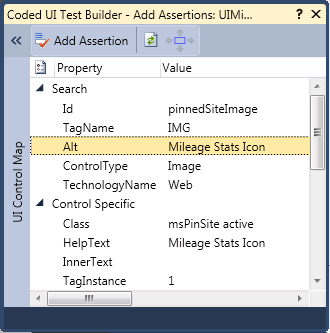
# Steps

* 1. In Visual Studio, create a new Test Project, named **CheckUIElementProperty**. To do this, point to **New** on the File menu, and then click **Project**. In the New Project dialog, select **Test Documents** under **Test Projects**. Set the project's name to **CheckUIElementProperty**, specify a valid location, and then click **OK**.
  2. Add a Coded UI Test. To do this, in Solution Explorer, right-click the **CheckUIElementProperty** project, point to **Add**, and select **New Test**. In the **Add New Test** dialog, select the **Coded UI Test**. Name the Coded UI Test **VerifyImageAltProperty** and click **OK**. In the **Generate Code for Coded UI Test** dialog as shown in the screenshot below, select **Record actions, edit UI map or add assertions**, and click **OK**.



* 1. Record the UI Test as follows:
     1. Click the **Record** button to start recording.



* + 1. Open Internet Explorer.
    2. Navigate to the Mileage Stats Home page.
    3. Click the **Record** button to stop recording. Click the **Show Recorded Steps** button to check if the steps were recorded correctly. The coded UI test builder – Recorded Actions should look like the following screenshot.
       1. 
    4. If there are unexpected steps, you can remove them by right-clicking the step you want to delete and then select **Delete**.
    5. Click the **Generate Code** button. Name the method **GoToHomePage**. Click the **Add and Generate** button. The code will be generated in a **UIMap.Designer.cs** file. This code can be modified/customized according to your need.
       1. **Note:** Each time you generate the code from a recorded method, the code in the UIMap.Designer.cs file will be overridden.
  1. Use the **Coded UI Test Builder** to create a validation method to validate properties of the target UI control. For this example, you will verify that the property of each image on the Home page is set to expected values as follows:
     1. Add an assertion to the UI control. To do this, drag the crosshair onto the UI control in your application that you want to test. When the box outlines your control, release the mouse. For example drag the crosshairs to the mileage status icon on the Home page.
        1. **Note:** It will be easier to select the UI elements if the browser is maximized to 100%.
           1. 
     2. The properties for this control are now listed in the **Coded UI Test Builder - Add Assertions** dialog box.
     3. Right-click the **Alt** property, and select the **Add Assertion**. Keep all values as default and click **OK**.
        1. 
     4. Repeat the above three steps for all images on the Home page, such as MyOpenId, Yahoo, and the HTML5 icons. For this test, collect all multi-assertions into one **Assert** method.
     5. Click the **Generate Code** button. Name the method name as **AssertImageAltProperty**. The following code snippet will be auto generated in **VerifyImageAltProperty.cs** file
  2. C#
     + 1. [TestMethod]
       2. public void CodedUITestMethod1()
       3. {
       4. // To generate code for this test, select "Generate Code for   
           // Coded UI Test" from the shortcut menu and select one of the menu items.
       5. // For more information on generated code, see   
           // http://go.microsoft.com/fwlink/?LinkId=179463
       6. this.UIMap.VerifyAltPropertyMethod();
       7. this.UIMap.AssertImageAltProperty();
       8. }
     1. Verify the assertion method generated in **UIMap.Designer.cs**
  3. C#
     1. // <summary>
     2. // AssertImageAltProperty - Use 'AssertImageAltPropertyExpectedValues'  
         // to pass parameters into this method.
     3. // </summary>
     4. public void AssertImageAltProperty()
     5. {
     6. #region Variable Declarations
     7. HtmlImage uIMileageStatsIconImage = this.UIBlankPageWindowsInteWindow.UIMileageStatsKnowwherDocument.UIMileageStatsIconImage;
     8. HtmlImage uIMyOpenIDImage = this.UIBlankPageWindowsInteWindow.UIMileageStatsKnowwherDocument.UILoginPane.UIMyOpenIDImage;
     9. HtmlImage uIYahooImage = this.UIBlankPageWindowsInteWindow.UIMileageStatsKnowwherDocument.UILoginPane.UIYahooImage;
     10. HtmlImage uISignInorRegisterImage = this.UIBlankPageWindowsInteWindow.UIMileageStatsKnowwherDocument.UILoginPane.UISignInorRegisterImage;
     11. #endregion
     12. // Verify that 'Mileage Stats Icon' image's property 'Alt'   
         // equals 'Mileage Stats Icon'
     13. **Assert.AreEqual(this.AssertImageAltPropertyExpectedValues.UIMileageStatsIconImageAlt, uIMileageStatsIconImage.Alt);**
     14. // Verify that 'My OpenID' image's property 'Alt' equals 'My OpenID'
     15. **Assert.AreEqual(this.AssertImageAltPropertyExpectedValues.UIMyOpenIDImageAlt, uIMyOpenIDImage.Alt);**
     16. // Verify that 'Yahoo' image's property 'Alt' equals 'Yahoo'
     17. **Assert.AreEqual(this.AssertImageAltPropertyExpectedValues.UIYahooImageAlt, uIYahooImage.Alt);**
     18. // Verify that 'Sign In or Register' image's property 'Alt' equals 'Sign In or Register'
     19. **Assert.AreEqual(this.AssertImageAltPropertyExpectedValues.UISignInorRegisterImageAlt, uISignInorRegisterImage.Alt);**
     20. }
  4. Modify the generated code as follows:
     1. Copy the code in UIMap.Designer.cs and paste it in UIMap.cs.
     2. In UIMap.cs, if not already present add the following **using** statement:
        + 1. C#
          2. using Microsoft.VisualStudio.TestTools.UITesting.HtmlControls;
     3. If we want to close the browser window automatically after each test case runs, add a **CloseBrowserWindow** function in the UIMap.cs partial class as follows:
        1. C#
        2. public partial class UIMap
        3. {
        4. ...
        5. ...
        6. public void CloseBrowserWindow()
        7. {
        8. #region Variable Declarations
        9. BrowserWindow currentBrowserWindow = this.mUIBlankPageWindowsInteWindow;
        10. #endregion
        11. currentBrowserWindow.Close();
        12. }
        14. ...
        16. }
     4. Add the following code snippet to the **VerifyImageAltProperty** Class in **VerifyImageAltProperty.cs** file. The **TestCleanup** attribute in this method marks this method to be executed every time a test method completes its run.
        1. C#
        2. //Use TestCleanup to run code after each test has run
        3. [TestCleanup()]
        4. public void MyTestCleanup()
        5. {
        6. // To generate code for this test, select "Generate Code for   
            // Coded UI Test" from the shortcut menu and select one of   
            // the menu items.
        7. // For more information on generated code, see   
            // http://go.microsoft.com/fwlink/?LinkId=179463
        8. this.UIMap.CloseBrowserWindow();
        9. }
  5. To run the test, Close all browser windows. Right-click inside the **VerifyImageAltProperty.cs** file and click **Run Tests**. The CodedUI Test begins to execute, this will open a browser and will run the application programmatically based on the recorded steps and will assert if the conditions are met. If so the test will Fail or Pass. Once the test completes the results are shown in the Test Results window

# Outcome

* 1. The Automation test project is created which can be used to automate testing of the UIElement **Alt** property of your web application.

# Further Reading

* + [Testing the User Interface with Automated UI Tests](http://msdn.microsoft.com/en-us/library/dd286726.aspx) on MSDN
  + [How to: Create a Coded UI Test](http://msdn.microsoft.com/en-us/library/dd286681.aspx) on MSDN