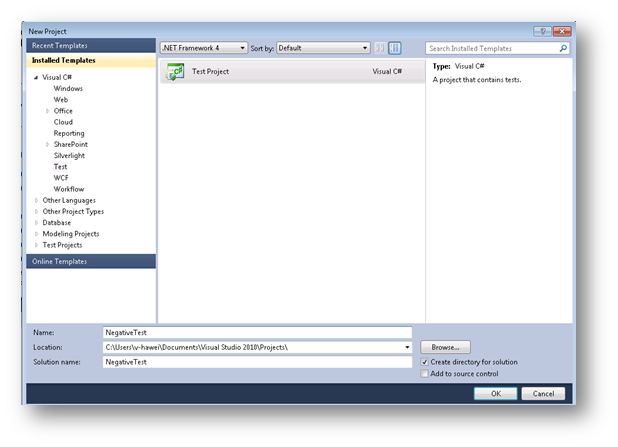
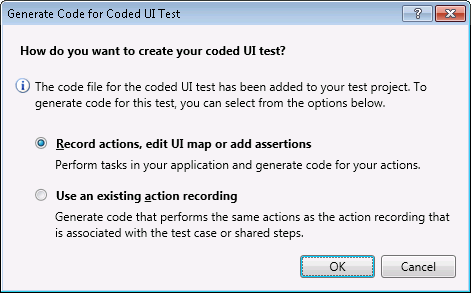
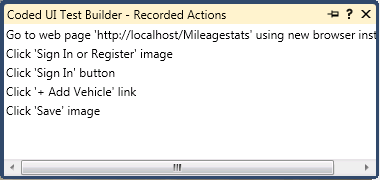
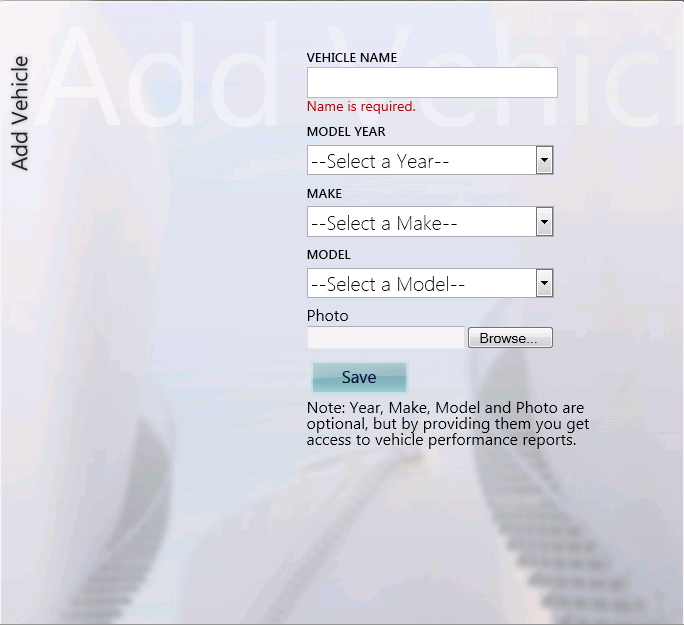
How to: Create Automation Negative Case with Coded UI Test

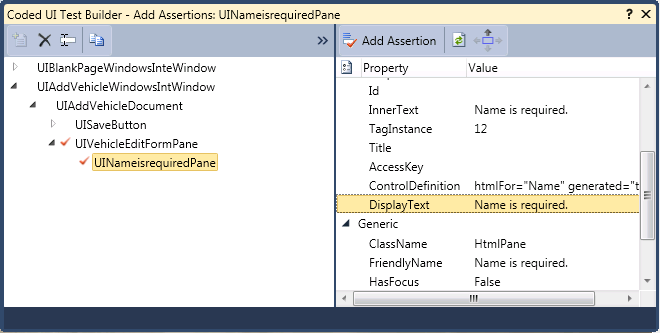
* 1. The following How-to topic will walk you through the creation of an automated test for your web application by building a coded UI test using Visual Studio Premium or Visual Studio Ultimate. The coded UI test performs actions on the user interface (UI) controls and verifies that the UI controls display the expected 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 assert a negative test case scenario. These tests will help to test the error handling capability of the application at the user level.

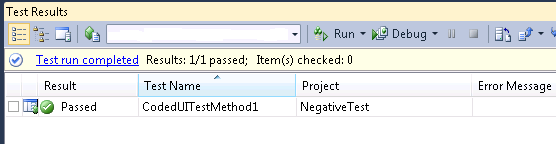
# Prerequisites

* 1. This topic requires you to have the same prerequisites required by 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/)
  + [Internet Explorer 9](http://windows.microsoft.com/en-US/windows/downloads/internet-explorer)
  1. It is assumed that the Mileage Stats web application has been deployed to a server running Microsoft Internet Information Services (IIS) in debug mode, and that the test site is <http://localhost/mileagestats>.

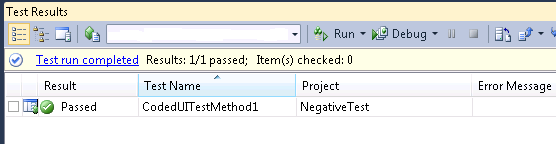
# Steps

* 1. In Visual Studio, create a new Test Project named **NegativeTest**. 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 **NegativeTest**, specify a valid location, and then click **OK**.
     1. 
  2. Add a coded UI test. To do this, in Solution Explorer, right-click the **NegativeTest** project, point to **Add**, and select **New Test**. In the **Add New Test** dialog, select **Coded UI Test**. Enter the Test name as **NegativeCodedUITest** and click **OK**. In the **Generate Code for Coded UI Test** dialog, select **Record actions, edit UI map or add assertions**, and click **OK**.
     1. 
  3. Record the UI Test as follows:
     1. Click the **Start Recording** button to start recording.
     2. Open Internet Explorer.
     3. Navigate to the Mileage Stats home page.
     4. Click the **Sign in or Register** button.
     5. On the Mock Authentication page, click the **Sign In** button.
     6. On the Dashboard, click **Add Vehicle** button.
     7. Click the **Save** button on the **Add Vehicle** form without entering any data.
  4. 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 dialog should look like the following screenshot.
     1. 
  5. If there are unexpected steps, you can remove them by right-clicking the step you want to delete and selecting **Delete**.
  6. Click the **Generate Code** button. Name the method **AddVehiclewithNullData**. Click the **Add and Generate** button. The code will be generated in a file called **UIMap.Designer.cs**. This code can be customized according to your needs.
     1. **Note:** Each time you generate the code from a recorded method, the code in the UIMap.Designer.cs file will be overwritten.
  7. 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 whether the error message is displayed after the **Save** button is clicked by following these steps:
     1. Add an assertion to a UI control. To do this, drag the crosshairs onto the UI control in your application that you want to test. When the box outlines your control, release the mouse. For this example, drag the crosshairs to the **UINameisrequiredPane** UI element, which displays the validation message "Name is required," and release the mouse.
        1. 
     2. The properties for this control are now listed in the **Coded UI Test Builder - Add Assertions** dialog box.



* + 1. Right-click the **Display Text** property, and select the **Add Assertion** command. Keep the other default values and click **OK**.
    2. Click the **Generate Code** button. Name the method **VerifyErrorMessage**. The **VerifyErrorMessage** method will be auto-generated and added to the test method in the NegativeCodedUITest.cs file.
       1. C#
          1. [CodedUITest]
          2. public class NegativeCodedUITest
          3. {
          4. public NegativeCodedUITest()
          5. {
          6. }
          7. [TestMethod]
          8. public void CodedUITestMethod1()
          9. {
          10. // To generate code for this test, select "Generate Code for   
               // Coded UI Test" from the shortcut menu and select one of   
               // the menu items.
          11. // For more information on generated code, see   
               // http://go.microsoft.com/fwlink/?LinkId=179463
          12. this.UIMap.AddVehiclewithNullData();
          13. this.UIMap.VerifyErrorMessage();
          14. }
          15. ....
          16. ....
          17. }
       2. 
  1. Modify the generated code as follows:
     1. Copy the code in UIMap.Designer.cs and paste it into 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 you 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. {
        5. ...
        6. public void CloseBrowserWindow()
        7. {
        8. #region Variable Declarations
        9. BrowserWindow currentBrowserWindow = this.mUIBlankPageWindowsInteWindow;
        10. #endregion
        11. currentBrowserWindow.Close();
        12. }
        13. ...
        14. }
     4. Add the following code snippet to the **NegativeCodedUITest** class in **NegativeCodedUITest.cs**. 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
        8. // http://go.microsoft.com/fwlink/?LinkId=179463
        9. this.UIMap.CloseBrowserWindow();
        10. }

9. Run the test method as follows:

* + 1. Close all browser windows. Right-click inside the NegativeCodedUITest.cs file and click **Run Tests**. The coded UI 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 they are met, the test will pass. Otherwise, they will fail. In the test below, it passes.
    2. Once the test completes, the results are shown in the Test Results window.
       1. 

# Outcome

* 1. Here we created the Automation test project, which can be used to automate UI testing of your web application for negative test cases.

# 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