|  |  |  |
| --- | --- | --- |
| Section 1 | Click Instructions | Talking Points |
| Beacon.pngimage.png  **Click Here** | 1. Open the PartsUnlimited.sln in Visual Studio. 2. Create a new Coded UI Test Project by right-clicking on the "tests" folder in the PartsUnlimited solution, then selecting Add -> New Project... | Coded UI tests provide a framework to create fully-automated tests that will validate the functionality and workflow of an application's user interface (UI). This is especially important because if new changes to the UI are deployed to an application, it is necessary to ensure that those changes will not break the application or cause unexpected behavior for the end-users. Since Coded UI tests rely on the validation of elements in the user interface, if the elements' identifiers are changed or removed and the tests are not updated, the tests may fail. However, the Coded UI tests will continue even if elements are moved around the view so as to remain flexible. In this demo, we will use the Coded UI Test Builder to generate test code for the Parts Unlimited Web application and modify the generated code in order to enable Cross Browser testing.  Let's start by adding a new Coded UI Test project to the solution. We'll use this to drive a browser as we search for various items in the catalog. |
| Beacon.pngimage.png  **Click Here** | 1. Name the Coded UI Test Project PartsUnlimited.CodedUI and then press Create. | As you can see, just after the project is created, a pop-up window appears and prompts us to either use an existing action recording or record actions using the Coded UI Test Builder. We'll select "Record actions using the Coded UI Test Builder" so that we can record our actions in a browser. Now, if a manual tester had already run this test manually, we'd be able to simply generate the code from their test case execution. |
| Beacon.pngimage.png  **Click Here** | 1. Click OK. 2. Troubleshooting: If the window doesn't open automatically, in the CodedUI.cs file, right-click in CodedUITestMethod1() and select "Generate code for Coded UI test." |  |
| Beacon.pngimage.png  **Click Here** | 1. Click on Start Recording (red square on the left in the Test Builder panel) and browse to the Parts Unlimited site (either https://partsunlimitede2edemo.azurewebsites.net or http://localhost depending on where you are running the demo) through Internet Explorer. | At this point, we can start recording our Coded UI test using the Coded UI Test Builder. After navigating to Parts Unlimited, we are going to search in the upper-right text field for the keyword, "battery," then press the search button. It should return two results and we will validate that those two batteries exist. |
| Beacon.pngimage.png  **Click Here** | 1. Open Internet Explorer |  |
| Beacon.pngimage.png  **Click Here** | 1. In Parts Unlimited, search for "battery" in the search text field at the top of the page, then press "Enter" or click on the search button. Then press the pause button in the Coded UI Test Builder to stop recording. | Two batteries should be returned |
| Beacon.pngimage.png  **Click Here** | 1. Click the Pause button |  |
| Beacon.pngimage.png  **Click Here** | 1. Click to show the steps | Once we've completed a set of related actions we'll want to generate code. Each time we stop and generate code a new method will be created in our coded UI test. |
| Beacon.pngimage.png  **Click Here** | 1. Click on Generate code in the Test Builder to generate the code for your recorded steps. Name the method something descriptive then click on "Add and generate." |  |
| Beacon.pngimage.png  **Click Here** | 1. Click Add and Generate |  |
| Beacon.pngimage.png  **Click Here** | 1. Click the button to Resume recording | Now we need to validate that two batteries were shown in the test results. We do that through an assertion.  To assert the result of our search, we drag the target button to the Calcium battery search result element from the Test Builder. Then, we add an assertion that the "Exists" property of the battery is equal to true. We're going to do this for the spiral coil battery as well. Once again, we'll generate code, this time for the assertions. |
| Beacon.pngimage.png  **Click Here** | 1. Drag the target button to the Calcium battery search result element from the Test Builder. |  |
| Beacon.pngimage.png  **Click Here** | 1. Ensure the first battery is highlighted with the blue box. |  |
| Beacon.pngimage.png  **Click Here** | 1. Click on the "Exists" property of the Calcium element so that it is higlighted. |  |
| Beacon.pngimage.png  **Click Here** | 1. Click the "Add assertion" button at the top of the panel for the elements to add an assertion that the "Exists" property of the battery is equal ("AreEqual") to true. |  |
| Beacon.pngimage.png  **Click Here** | 1. Click OK. |  |
| Beacon.pngimage.png  **Click Here** | 1. Repeat this for the spiral coil battery. |  |
| Beacon.pngimage.png  **Click Here** | 1. After adding both assertions, click on the generate code button. |  |
| Beacon.pngimage.png  **Click Here** | 1. Add a method named CalcumAndSpiralCoilBatteriesExist to validate these elements exist. Click "Add and Generate" |  |
| Beacon.pngimage.png  **Click Here** | 1. Click the "X" to close the recorder. |  |
| Beacon.pngimage.png  **Click Here** | 1. In the CodedUITest1.cs file in the Coded UI test project, rename the CodedUITestMethod1 method to something descriptive such as TestForSearchBattery. | It's a good idea to rename out generated method to something more descriptive. |
| Beacon.pngimage.png  **Click Here** | 1. Save the file. |  |
| Beacon.pngimage.png  **Click Here** | 1. In the same method, right click and select "Run tests" to run the Coded UI test that you just created. A window should open and walk through the steps of the test. | OK. Let's run our tests! |
| Beacon.pngimage.png  **Click Here** | 1. Note that the tests have passed. | After the test completes, the results will appear in the Test Explorer window in Visual Studio. |
| Beacon.pngimage.png  **Click Here** | 1. In the Test Explorer window, right-click on the Coded UI Test that we just ran and select Add to playlist -> New Playlist. | You can see the true value of automated UI tests by running the one that we just created. By recording this test once, we can run this same testing scenario as many times as we want without having to manually run through the test steps. What if we want to run this test as one of many of our automated regression tests? We can either add this to an existing playlist of tests or we can create a new one and add our Coded UI test to it.  Let's add our test to a regression suite of tests. |
| Beacon.pngimage.png  **Click Here** | 1. Create a new playlist file in the test folder named RegressionTests, then click on the "Save" button. |  |
| Beacon.pngimage.png  **Click Here** | 1. CS27: Click Here |  |
| Beacon.pngimage.png  **Click Here** | 1. Select the playlist in Test Explorer. |  |
| Beacon.pngimage.png  **Click Here** | 1. Re-run the test to view results of just the RegressionTests. | Coded UI tests can automate testing for web applications by recording your tests using Internet Explorer. You can then customize your test and play it back using Internet Explorer or other browsers. For instance, we can use Chrome, instead. |