|  |  |  |
| --- | --- | --- |
| Section 1 | Click Instructions | Talking Points |
| Beacon.pngimage.png  **Click Here** | 1. Open the PartsUnlimited.sln solution in Visual Studio 2017. 2. Open the Test Explorer window. (You should see 14 unit tests that already exist.) 3. Open the StringContainsProductSearch.cs file 4. Scroll down to the ProductSearch | Depluralize method file. | Let's take a quick look at the Depluralize method. This method is responsible to return singular form of a string value. For example: if you search using a keyword Batteries, it returns Battery. This is a naive version of 'stemming' which can assist in searching.  By looking at the Code Lens line above it, we can see that there are currently no Unit Tests testing this method. That's not good. However, as a good developer, I want to be sure to leave the code better than I found it, so I'm going to cover it with unit tests.  Rather than write my own, I'm going to use a feature called IntelliTest to help me generate a suite of appropriate tests. |
| Beacon.pngimage.png  **Click Here** | 1. Right Click within the method Depluralize and then select IntelliTest | Run IntelliTest |  |
| Beacon.pngimage.png  **Click Here** | 1. Highlight the first failing test. | After running IntelliTest, I now have 8 unit tests with a pretty decent amount of code coverage (14/17 blocks). Of those, 2 are failing and 6 are passing. Looks like we may have some work to do!  We see that IntelliTest passed in a null, and a null reference exception is thrown. This makes sense. |
| Beacon.pngimage.png  **Click Here** | 1. Click on the second failing test. | The second failing test is an argument out of range exception. |
| Beacon.pngimage.png  **Click Here** | 1. Highlight all the text cases (Ctrl-Click top and bottom to select all). 2. Click the Save icon. | When we save the tests, IntelliTest will create a Test Project with a suite of new unit tests. |
| Beacon.pngimage.png  **Click Here** | 1. Add the following code to the top of the method, and rerun the IntelliTest. 2. if (string.IsNullOrEmpty(query)) return query; 3. else | Let's put a quick null guard in our code and rerun IntelliTest. |
| Beacon.pngimage.png  **Click Here** | 1. Highlight the formerly failing null test. | As you can see our test is now passing. |
| Beacon.pngimage.png  **Click Here** | 1. To fix the remaining error, change the code to: 2. query = query.Substring(0, query.Length -1). | It looks like we have an off by one error. Lets fix the code. |
| Beacon.pngimage.png  **Click Here** | 1. Rerun IntelliTest. |  |
| Beacon.pngimage.png  **Click Here** | 1. Highlight the formerly failing test. | It looks like all our tests are now passing. |
| Beacon.pngimage.png  **Click Here** | 1. Show the Test Explorer, and click Analyze Code Coverage for All Tests. | Let's run all our tests in Test Explorer. |
| Beacon.pngimage.png  **Click Here** | 1. Open the new tests. | Note that all the tests are now passing. |
| Beacon.pngimage.png  **Click Here** | 1. Click to see the Code Coverage Results. |  |
| Beacon.pngimage.png  **Click Here** | 1. Click the button to Show Code Coverage Coloring |  |
| Beacon.pngimage.png  **Click Here** | 1. Highlight the method. | Note that all the code is currently covered. IntelliTest covered all our code without us having to write a single unit test. |