Demo Steps

1. Start debugging MyShuttle.Client.Web

Talking points: I’m expecting a driver to appear, but something obviously isn’t right. Click the “Select” button a different car, still fails so obviously something isn’t right.

1. Go back to Visual Studio and look at the Diagnostic Tools window

Talking points: Since I’m using enterprise I’m going to start by looking at the Diagnostic Tools window, I can see that I see that exceptions happened.

1. Click on the exception

Talking points: The great thing about IntelliTrace is it will take me to the location in code where this occurred. I can either click “Activate historical debugging” or just double click

1. Double click the exception event in the IntelliTrace window

Talking points: I can see the exact line of code this occurred on, and that it’s a null reference exception. Now I know exactly where to put my breakpoint and debug, IntelliTrace took any guess work out of.  
However if I I’m not using the Enterprise SKU so don’t have IntelliTrace I can still figure out what is going on.

1. Unpin the Diagnostic Tools window, and Open the Output window

Talking points: Looking through the output window I can see that there were some System.NullReferenceExceptions thrown. The next thing I want to do is configure the debugger to break when this is thrown since the application is clearly handling it since the debugger would have stopped if there had been any unhandled exceptions

1. Open the Exception Settings window from Debug -> Windows

Talking points: To do that, I’ll open the improved exception settings window. It now opens as non-modal tool window.

1. Type NullRef into the search box in the Exception Settings window.

Talking points: Notice it now supports incremental search filtering

1. Click the “System.NullReferenceException” node
2. Press F5 twice to return to live debugging
3. Click “Select” in Internet Explorer, the debugger will break on the NullReferenceException  
   Talking points: The debugger now stopped on the exception just like I had a breakpoint set for the Exception event. See I can even step
4. Step Over (F10), inspect the driver parameter (I use a DataTip in the editor)
5. In the Immediate window, type the .Where() lambda into the watch window replacing the condition with “driver.id == d.DriverId” so the whole lambda appears as drivers.Where(d => driver.id == d.DriverId).FirstOrDefault()
6. Right click on line 68 (opening brace of the try statement) and choose “Set Next Statement”
7. Replace the lambda with the one from the Immediate window (look at the file tab, if there is a lock icon on it, do a quick step before making the change. Sometimes the file gets marked as read only—haven’t figured out why but a step fixes it)
8. Step over the execution of the lambda, inspect the “selectedDriver” variable and see that it correctly populated
9. Resume debugging (F5)
10. Click the “Select” button on another driver to confirm it’s fixed
11. Set a breakpoint on line 52 on HomeController.cs, then quickly hit F10 to step again
12. Step over the function call to get the driver
13. Set a breakpoint on the closing brace of the current method (line 62), drag the yellow instruction pointer back up to line 52 and start debugging
14. Change the GetIndividualDriver method call to GetIndividualDriverCached, then move the instruction pointed back to line 53 (the method call)
15. Step Over, move the instruction pointer back again, Step Over
16. Delete breakpoints, resume debugging. Click “Select” a few times to prove its fast now
17. In Visual Studio pin the Diagnostic Tools window again
18. Zoom out on the graphs
19. Navigate to the memory tab and click “Take Snapshot”
20. Click “Select” ~5 more times in Internet Explorer
21. Set a breakpoint on ~line 60 in HomeController.cs (exact line doesn’t matter)
22. Click “Select” again, take another snapshot when the breakpoint is hit
23. Click on one of the + numbers to the left of the red up arrow
24. Use View->Full Screen
25. Click on the “MyShuttle.Client.Core.DocumentResponse.Driver” row (should be the top one)
26. Hover over the \*.Driver row and click the icon that appears on the right side of the first column
27. Open DriverCache.cs, set a breakpoint on line 19, click the “Settings” icon on the toolbar, click the “Actions” checkbox and enter “Adding driver {key} $FUNCTION”
28. Set a breakpoint on line 28 of DriverCache.cs, open the settings window, click “Actions” and enter “Getting driver {key}”. Close the window (Enter 2x from the message dialog will do it)
29. Continue debugging, click “Select” several times in Internet Explorer
30. Go to Visual Studio, open the Output window (you might need to scroll down if you scrolled up to find the exceptions)
31. Set a breakpoint on line 23 of DriverCache.cs (beginning of GetDriverFromCache function), click “Select” again, and uncomment the commented code at the beginning of both functions in the file
32. Delete the breakpoint and resume debugging
33. Click “Select” several more times in the web browser, look at the Output window and verify that you are now only getting the driver, not adding it to the cache multiple times
34. Go to the “Events” tab in the Diagnostic Tools window and notice the Tracepoints appear there as well, double click and it navigates you to where the Tracepoint was hit (stop debugging or switch your screen back quickly at this point, there is a bug because you used EnC IntelliTrace will put the historical instruction pointer on the wrong line of code)