Developer Tips: C# Events

By Andrew Hinkle

We've all used events whether we know it or not, such as WebForms click events. Usually, you use the UI properties to click on the event you want, populate the method, and you're done. For the most part of my career, that's been the extent of my usage.

Recently I ran into some legacy code that raised events when an address was changed. Yes, events have been around for a long while now. It took me a little bit to pick up what was going on (an order of events issue, pun intended!), but it emphasized my lack of understanding.

I recently watched an excellent PluralSight video titled ["C# Events, Delegates and Lambdas" by Dan Wahlin](https://app.pluralsight.com/library/courses/csharp-events-delegates/table-of-contents). It's definitely worth your 3h 11m.

In the following example, I wanted to illustrate how to use events with a fairly simple example. The shopping cart contains a billing and shipping address, and a Boolean flag if the shipping address is the same as the billing address. Declare the event handler with a custom parameter of Address. The event handler will default the sender as the current object, ShoppingCart in this case.

In the shopping cart constructor:

1. Register a logger method to add the city info to a string list before any changes were applied
2. Register a method to update the shipping address
3. Register a logger method to add the city info to a string list after the changes were applied.

Keep in mind that you don't have to implement the event methods (delegates) in the class. In this case the Logger is also used. I could have decoupled the logger completely from the ShoppingCart by only passing in the delegate, so the ShoppingCart wouldn't have any knowledge of the logger or where the delegates came from.

When the billing address is changed, update the shipping address if it should be changed to match the billing address. Whenever possible add proper unit tests to test the functionality (not the integration) to verify your code works as expected. Code samples are provided below and in my GitHub Tips repository (Tips.Events) <https://github.com/penblade/Tips>.

# Address



# Logger

The logger simply adds messages to a string list, though you could have performed any action. I added the before and after scenarios to clearly illustrate that the order you add the events to the handler are honored when called.



# ShoppingCart

Declare the EventHandler on the object where the event will be triggered. I injected the addresses and other settings to simplify the unit tests. This approach also makes the class honest in that this information is expected to be provided before any actions may be performed on the object. That's what is expected for this scenario, however, real world implementation may vary.





# Unit Tests

The tests are a little verbose, but get the point across. Usually, I separate the test methods so that the main setup (Arrange) and process (Act) are handled in the constructor or [TestInitialize], and each test (Assert) is a separate [TestMethod]. For the sake of brevity and a lot of dead whitespace, the assert statements have been grouped into two unit test methods.





