So, once again, in the Mocking folder, let's take a look at this

order service. This is the order service that you saw in the last lecture.

So we're going to write the test for this method, add Assert .storage.Store is called with the same object that we passed to this PlaceOrder method, okay?

So, in our unit testing project, inside the Mocking folder, let's add a new class, OrderServiceTests.

So first we apply the TestFixture attribute, and then create a Test method. So public void, PlaceOrder\_WhenCalled should StoreTheOrder.

Now back to our order service, look at the constructor of this class. We are injecting this IStorage object here. So in our test, we need to create a mock object for this interface. So, let's create a new Mock of IStorage interface. Now, we create an order service so new OrderService and we pass storage.object as an argument.

So this is the arrange part. Now, we need to act. Service.PlaceOrder.

We give it an order object, very simple, now finally we need to assert that storage.store is called. To do that we call storage. so far you have seen the set up method, right?

This is how we program a mock object. We also have a Verify method. And with this we can verify any given method is called with the right arguments are not. So again, here we pass a Lambda expression, storage goes to storage.Store, and as an argument we need to pass the same order object that is passed to the PlaceOrder method. So I'm going to extract this into a local variable, so var order pass an object here, and also, verify that the same object is passed to the store method of the storage.

So, we add order here. Now, let's run this test. Okay, here's our new test. It's passing, beautiful, let's just make sure that we're passing the right thing, so I'm going to go back to order service, if we comment out this line, that test is going to fail, so, we can either comment this out or we can return something earlier that has the same affect.

So, back here, let's run the test one more time. Okay, our test failed. Look at the error message. Expected in location on the mock at least once, but was never performed.

So this verifies that we're testing the right thing. Now back to our order service, let's revert this back. Okay, so to recap, in order to test the interaction between two objects, you can use the verify method of mock objects.