A little while ago, when I was designing this course, I sent out a survey to my students, and told them to ask me any questions about unit testing, the areas that they find challenging. One of the questions that came up a few times was how to test void methods. So far all the examples

we have seen, were around methods that returned some value.

So in this lecture, we're going to look at void methods. Now just to refresh your memory, earlier I told you about two types of functions

in programming. We have query functions that return some value and command functions which are supposed to perform an action. A void function by definition is a command function. These command functions often change some kind of state, that means they change the state of an object in memory, they may change the value of one or more properties, and additionally, they may persist a state, they may score this object in a database, they may call a web service, they may call a message que (?).

Now, for this section, we're not going to deal with those resources. So let's take a look at an example of a void method that changes the state of an object in memory.

So here we have this ErrorLogger class, look at this method, Log. It takes an error message it does some basic validation to make sure that

the error IsNullOrWhiteSpace, otherwise it's going to throw an ArgumentNullException. Now, we're going to look at writing tests for methods that throw exceptions later in this section, so don't worry about the first part, then it sets the last error property, here it's supposed to write this error to a log, actually there is a problem in this comment.

So let's ignore that for now, and finally, it raises an event. Again, we're going to look at how to test methods that raise an event later in this section. But for now we just want to focus on testing this part. So when we call the log method, we should verify that the last arrow property is set properly. This is the outcome of this method. So it's not returning a value, it's changing the state of an object in memory. So, back to our unit testing project, another unit testing class, ErrorLogger, Tests.

First we apply the TestFixture attribute.

Now, a test method public void Log When Called should set the last

ErrorProperty. Of course there are other test cases but we are not going to worry about them yet. So let's create an instance of error logger, a new ErrorLogger, now we call logger.Log, through the error message again we use simple values. A simple string like a. This method is void so we're not going to get a result here. Doesn't make sense, right? Instead we wanna assert that the last arrow property set accordingly. So Assert That, log

logger.LastError, IsEqualTo a. Let's run this test.

Okay, it passed, beautiful, now as I told you before, I want to make

sure that this is a trust worthy test. So, I'm going to go back in production code, this is the line that is responsible for making our test pass.

So if I comment this out, and our test still passes, that means that test is testing the wrong thing, it's not a trust for the test, and if it fails, we know that's a good test. So, save, back here, I'm going to run this test again, this time, it failed. We expected A, but we got Null. Because the default value of that property is null.

So, back to our ErrorLogger class, let's bring this back in. In the next lecture, I'll show you how to test methods that throw an exception.