CS 240: Unit Testing Database Code Transcript

This video shows a split screen of Professor Wilkerson on the right and a PowerPoint screen on the left. The left screen will switch to IntelliJ. Visual descriptions are not needed.

[00:00:00] **JEROD WILKERSON:** The last thing I want to teach you about unit testing is how to unit test database code.

[00:00:05] There are a few additional things you have to think about when you're testing database code.

[00:00:10] Remember, our tests should be independent of each other, so that means that one test shouldn't leave anything behind that affects the next test.

[00:00:17] If you think about that, that can be a little bit tricky with databases because if we're testing database code, for example, we're testing DAOs, then that code will be interacting with the database.

[00:00:31] It's easy for a test to leave something in the database, make some kind of a change that the next test will see and be affected by.

[00:00:37] That's what we don't want.

[00:00:40] What we need to do then is first of all, we'll have to have our database driver on the class path in order to run database tests.

[00:00:48] Then there really are two primary ways that we can write database tests that don't affect each other.

[00:00:55] One is to recreate the tables before each test. That can be an option.

[00:01:00] We can have a BeforeEach method that will run that will clear out our database, maybe drop all the tables and create new ones, or do whatever it needs to do to clean up the database so we have a pristine clean database for every test.

[00:01:14] The other option. Remember, we have the ability to do things within a transaction.

[00:01:19] The other option is to create a transaction inside of a BeforeEach method and rollback the transaction in an AfterEach.

[00:01:28] That will allow a test to do whatever it wants to the database and then the AfterEach will clean up the database before the next test runs.

[00:01:36] I'm going to show you an example of the second option.

[00:01:40] Before I do that though, let's think about when we might use each one.

[00:01:44] We probably would use the second option if we have a fairly complex database with a lot of setup.

[00:01:51] If we have to do a lot of setup for every test, we will create a lot of tables or something like that.

[00:01:57] It's going to make the test run really slow if we have to keep doing that and so that would be a case where we would prefer the second option.

[00:02:08] Here we have a class called DatabaseManager.

[00:02:13] This is the class that we want to test, and I can use that to show you how to test database code.

[00:02:19] This method has an openConnection, or this class has an openConnection method that will allow us to get a connection to some database.

[00:02:28] It has a closeConnection method. It has a method for creating tables, and really, the only table is a Dictionary table so it also has a method for populating that table, for putting some values in it.

[00:02:43] It has a method for getting all the words from it.

[00:02:47] It's called loadDictionary.

[00:02:49] We would use that if we were loading a dictionary from the words in a database.

[00:02:53] Then it just has a main method that we're not really too interested in for this example.

[00:02:57] Here is our DatabaseManagerTest, and what we have here is we're using both a BeforeAll method and a BeforeEach method.

[00:03:08] In our BeforeAll method, we are getting a reference to the DatabaseManager instance.

[00:03:13] We're creating an instance of DatabaseManager.

[00:03:16] We are calling openConnection on it.

[00:03:18] Now we have an opened connection.

[00:03:19] One of the things that I should have pointed out in here is when we open the connection, we setAutoCommit to false.

[00:03:25] We start a transaction.

[00:03:27] We have started the transaction.

[00:03:29] Now if this were testing an existing application, the tables would probably already exist, but since this is a small example, not against a big working piece of software, we're going to need to create the tables when we start up the test.

[00:03:44] Within our BeforeAll, we create all the tables.

[00:03:48] Now our test is ready to go. W have really only one test method here, but you can see from this how it would work if I had multiple test methods.

[00:03:59] My one test method will be executed, but before it is, my BeforeEach method is executed, so we call the fillDictionary method.

[00:04:07] We'll populate some values.

[00:04:09] Now we have those values available to this test.

[00:04:12] This test will test that the words in the dictionary are correct, and then after that, we will call a closeConnection in the AfterEach.

[00:04:23] If we look at closeConnection, we are passing it false.

[00:04:29] False is the indicator of whether we should commit or not.

[00:04:32] After each test, we close connection passing false, which means we're going to rollback.

[00:04:37] It says if commit's true, commit. Otherwise, rollback.

[00:04:41] In the test, we're going to be rolling back, and that makes it so this test won't leave anything behind and so if we had another test, it would start again after setup is called, so after the database has refilled.

[00:04:55] That's the thing you have to think about when you're testing database code.

[00:04:59] You need to write your code in a way that each test is cleaned up before or after or both so you don't end up in a situation where one test leaves things behind in the database that can infect the next test.