In the last section, you learned the fundamentalsof unit testing. However, in that section, our focus was purely on testing utility classes that didn't have any external dependencies like files, databases, web servers, and so on. But in the real world, our applications almost always use one or more of these external resources.

So, you might be wondering how we can unit test a class that depends on an external resource. Well, as I told you before. Your unit test should not touch external resources. A test that touches an external resource is classified as an integration test, which is a separate topic.

But you can still unit test the logic in your classes, while excluding their external dependencies. And, that's what you're going to learn in this section. So here's a basic principle. Imagine you have a class called video service. This class uses the file class to read the content of a file. In your application, this video service may use the db context class to edit a (?) framework to read a record from the database, so it has a dependency to an external resource. In order to unit test the video service class, you should do some surgery in your code and decouple the video service from the file class or db context or whatever that external dependency is. This way when unit testing this class, you can replace the file class with another class that looks like the file class from the outside, but doesn't talk to the file system. In fact, it doesn't do anything, its like a class

with a bunch of methods that have no implementation. We call this class a fake, or a test double, just like a stunt double in movies. In action scenes, a stunt double who replaces the hero of the movie, right? We have the same concept here.

When unit testing classes with external dependencies, we replace

a production object with a test double, or fake object. Okay, but now you might be asking, how should we do that, that requires surgery to decouple our classes from their external resources.

That's the topic for the next lecture.