Alright, here's an exercise for you. So, in the Mocking folder, look at EmployeeController. This is an example of code that you see in a lot of ASP.NET MVC applications. Now this is not a fully working example, it's purely simulation, because here we don't have an ASP.NET MVC project. We have a simple class library.

But that aside, let's see what's happening here. So in this class, we have this private field of type EmployeeContext. This is part of entity framework.

So if you look down here, this EmployeeContext looks like a db context in entity framework, so it has a property of type, DbSet of Employee. Here we should actually derive this from DB context, I forgot to do that, but it doesn't really matter, because we don't want to run this code. So, we have this EmployeeContext and we have initialized this in the constructor. We have this method, deleteEmployee, takes an integer, that's the ID of the employee, now here over these three lines, we try to delete this employee from the database. So first, we try to find that employee, then remove it from the context and at this point, that employee is marked for deletion.

But changes are not persisted to the database, yet. When we call DB.save changes, entity framework generates the right query, and sends it to the database.

And at this point, the employee record (?) gets deleted. And finally, we direct the user to the list of employees.

So, if you want to unit test this method, you need to write, two tests, one is to ensure that this method returns the right result. And this is what we call state based testing. Another test is to ensure that this method deletes the given employee from the database.

So first of all, you need to extract these few lines into a separate class, because these lines touch an external resource in this case, a database, so you want to extract this into a separate class like EmployeeStorage. Then, extract the interface, injected in the constructor here, and finally, Assert that this method calls the right method in that storage object. So, go ahead and do this exercise, in the next lecture, we'll look at my solution.