Lessons learned in Unit Testing

We all have to start somewhere

Assumptions:

You are using:
A Unit testing framework
IoC container framework
CI

Some code:

An OrderService Example;

-Saves Order

- when order is saved, sends email

Doubles
Fakes
Stubs
Mocks

Testing State

Assert on the object

Testing Behaviour

Verify on the Mock

Mocking frameworks

Integration Tests vs Unit Test

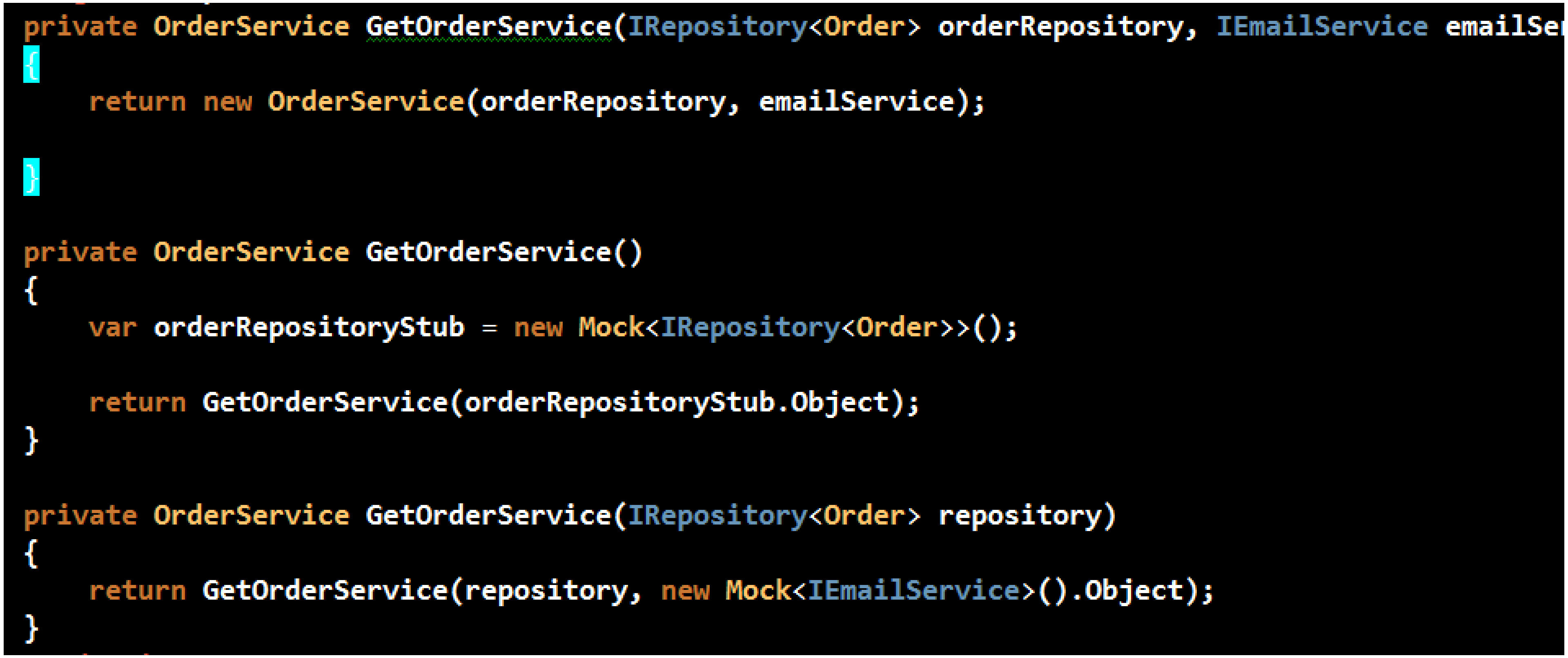
Use it, Don't abuse it
Setup | TestFixture
TearDown

Encapsulation.. and testing

Exceptions

Creation Pattern

-Simple test helper
- Factory
- Object Mother



```
private OrderService GetOrderService(IRepository<Order> orderRepository, IEmailService emailSe
{
    return new OrderService(orderRepository, emailService);

}

private OrderService GetOrderService()
{
    var orderRepositoryStub = new Mock<IRepository<Order>>();

    return GetOrderService(orderRepositoryStub.Object);
}

private OrderService GetOrderService(IRepository<Order> repository)
{
    return GetOrderService(repository, new Mock<IEmailService>().Object);
}
```

AAA

Naming Convention

Test API

Required and Optional Dependencies

Test Smells:

-Conditional Logic in tests

- Test Duplication

- Test depends on other tests

-Setup abuse

- Too many expectations

=

Hard to test code

Conditional Logic in Tests
=
Remove indentation

Test Duplication

Test Interdependency (also called shared state)

Multiple Asserts

Setup Abuse

Good Tests:
Run in Isolation
Are Trustworthy
Are Readable

=> Maintanable

Isolation

Trustworthy

Readable

Be Paranoid
Verify before action and after action

Andrea Magnorsky
Twitter: @silverspoon
www.roundcrisis.com