How do I unit test a WCF service?

Asked 16 years, 3 months ago Modified 9 years, 1 month ago

Viewed 15k times

18

We have a whole bunch of DLLs that give us access to our database and other applications and services.

We've wrapped these DLLs with a thin WCF service layer which our clients then consume.

I'm a little unsure on how to write unit tests that only test the WCF service layer. Should I just write unit tests for the DLLs, and integration tests for the WCF services? I'd appreciate any wisdom... I know that if my unit tests actually go to the database they won't actually be true unit tests. I also understand that I don't really need to test the WCF service host in a unit test.

So, I'm confused about exactly what to test and how.

wcf unit-testing

Share

Improve this question

Follow

edited Nov 20, 2015 at 11:46

Yeldar Kurmangaliyev

34.2k ● 13 ● 63 ● 103

asked Sep 1, 2008 at 2:13

Esteban Araya

29.6k ● 25 ● 111 ● 141

1 There is no need to UNIT tests your WCF services, it's perfectly OK to write INTEGRATION tests.
– Michael Freidgeim Jul 23, 2013 at 12:32

Avoid using mocks, a unit is not a class without any outside dependency. A unit is an end-to-end slice of business logic, even if it involves database. You can always use an in-memory or even better use an embedded database such as BerkeleyDB to test. You don't need an actual database residing on other machine. That way you can test your unit effectively without huge amount of mocking.
– Narendra Pathai Nov 16, 2014 at 8:31

3 Answers

Sorted by: Highest score (default)

7

If you want to unit test your WCF service classes make sure you design them with loose coupling in mind so you can mock out each dependancy as you only want to test the logic inside the service class itself.

For example, in the below service I break out my data access repository using "Poor Man's Dependency Injection".

```
Public Class ProductService
    Implements IProductService

    Private mRepository As IProductRepository

    Public Sub New()
        mRepository = New ProductRepository()
    End Sub
    Public Sub New(ByVal repository As IProductReposit
        mRepository = repository
    End Sub

    Public Function GetProducts() As System.Collection
Product) Implements IProductService.GetProducts
        Return mRepository.GetProducts()
    End Function
End Class
```

On the client you can mock the WCF service itself using the interface of the service contract.

```
<TestMethod()> _
Public Sub ShouldPopulateProductsListOnViewLoadWhenPos
    mMockery = New MockRepository()
    mView = DirectCast(mMockery.Stub(Of IProductView)(
    mProductService = DirectCast(mMockery.DynamicMock(
IProductService)
    mPresenter = New ProductPresenter(mView, mProductS
    Dim ProductList As New List(Of Product)()
    ProductList.Add(New Product)
    Using mMockery.Record()
        SetupResult.For(mView.PageIsPostBack).Return(F

Expect.Call(mProductService.GetProducts()).Return(Prod
    End Using
    Using mMockery.Playback()
        mPresenter.OnViewLoad()
    End Using
    'Verify that we hit the service dependency during
is false
    Assert.AreEqual(1, mView.Products.Count)
    mMockery.VerifyAll()
End Sub
```

Share Improve this answer Follow

edited Sep 1, 2008 at 16:31

answered Sep 1, 2008 at 2:26

Toran Billups
27.4k ● 41 ● 158 ● 272

From martinfowler.com/bliki/InterfaceImplementationPair.html Using interfaces when you aren't going to have multiple implementations is extra effort to keep everything in sync.Furthermore it hides the cases where you actually do provide multiple implementations. See more examples in stackoverflow.com/questions/90851/… – Michael Freidgeim Jul 23, 2013 at 11:37

7

It depends on what the thin WCF service does. If it's really thin and there's no interesting code there, don't bother unit testing it. Don't be afraid to not unit test something if there's no real code there. If the test cannot be at least one level simpler then the code under the test, don't bother. If the code is dumb, the test will also be dumb. You don't want to have more dumb code to maintain.

If you can have tests that go all the way to the db then great! It's even better. It's not a "true unit test?" Not a problem at all.

Share Improve this answer Follow

answered Sep 1, 2008 at 19:19

Jan Soltis
4,419 ● 2 ● 27 ● 24

4

The consumer of your service doesn't care what's underneath your service. To really test your service layer, I think your layer needs to go down to DLLs and the database and write at least CRUD test.

Share Improve this answer

Follow

answered Sep 1, 2008 at 2:27

Eugene Yokota

95.5k ● 45 ● 217 ● 320