Unit Testing Guide

getting started

Content

Key properties of good designed unit testing

1. It should be automated and repeatable.
2. It should be easy to implement.
3. It should be relevant tomorrow.
4. Anyone should be able to run it at the push of a button.
5. It should run quickly.
6. It should be consistent in its results.
7. It should have full control of the unit under test.
8. It should be fully isolated.
9. When it fails, it should be easy to detect what was expected and determine how to pinpoint the problem.

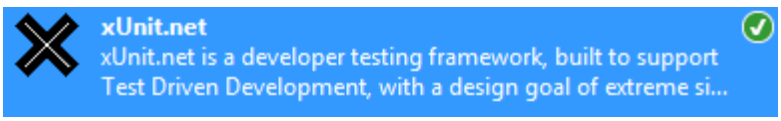
Prerequisite

Visual studio 2012+ Ultimate or Premium
NuGet Package Manager client version 2.8.1+(download)

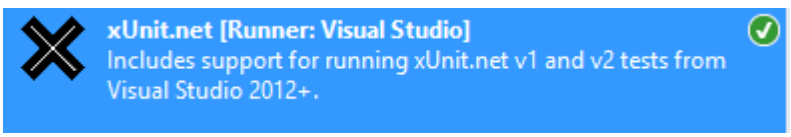
Environment setup

xUnit setup:

1. Create class library.
2. NuGet install xUnit.net



3. NuGet install xUnit.net Runner Visual Studion



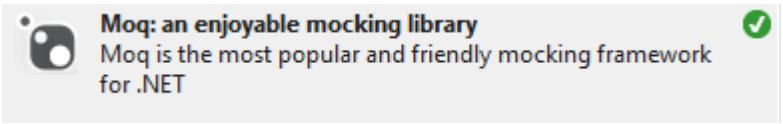
4. Add reference to xUnit (using Xunit;)
5. Add reference to target project (under test)

Fakes setup:

6. Generate fake, right click on reference to project click Add Fakes Assembly
7. Add reference to fake assembly (using api.protravel.dk.Fakes;)

Moq setup:

8. NuGet install Moq



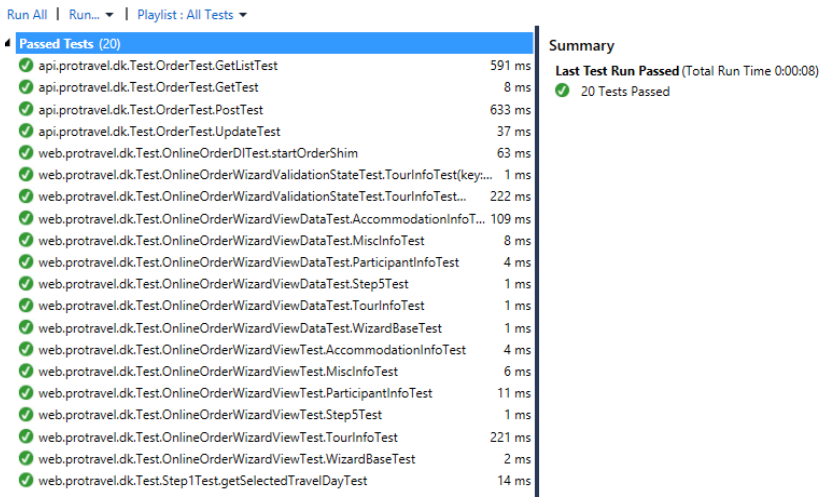
9. Add reference to Moq (using Moq;)

Writing Unit test

1. Create public class within class library
2. Create public void method
3. Decorate method with [Fact] or [Theory]
   [Fact]- normal test, [Theory] - parameterized test
4. **Arrange** method (`OnlineOrderDI order = new OnlineOrderDI();`)
5. **Act** method (`var result = order.startOrder(Guid.NewGuid());`)
6. **Assert** method(`Assert.Equal(Id, result.TourId);`)

Running test

1. In menu bar navigate to Test>Windows>Test Explorer
2. Click “Run All” or select specific option on dropdown “Run”
3. Wait for while and read result.

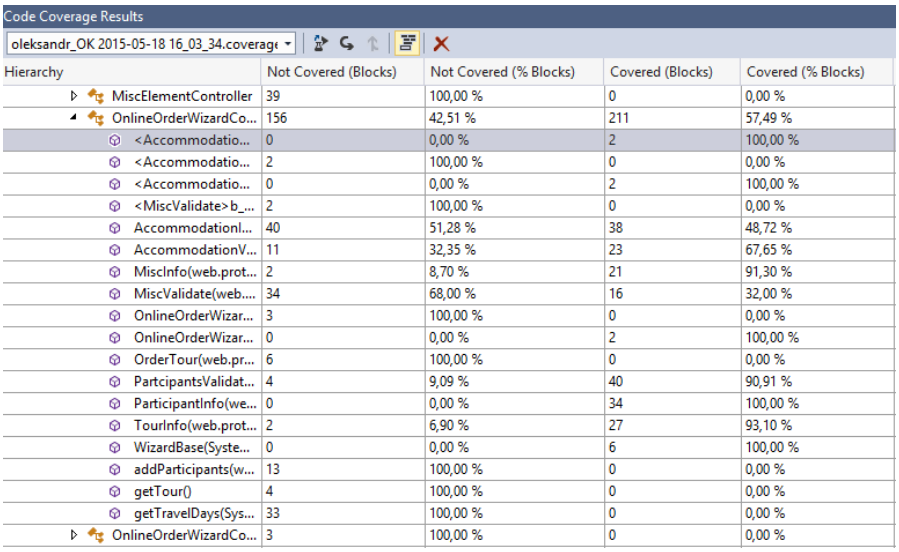


Debugging of test

1. In test explorer select failed test
2. Right click on test and select “Debug selected test”

Analyzing of code coverage

1. In menu bar navigate to Test>Analyze code coverage
2. Click “All Tests” or “Selected Tests”
3. Wait for while and read result in result explorer



Hierarchy	Not Covered (Blocks)	Not Covered (% Blocks)	Covered (Blocks)	Covered (% Blocks)
MiscElementController	39	100,00 %	0	0,00 %
OnlineOrderWizardCo...	156	42,51 %	211	57,49 %
<Accommodatio...	0	0,00 %	2	100,00 %
<Accommodatio...	2	100,00 %	0	0,00 %
<Accommodatio...	0	0,00 %	2	100,00 %
<MiscValidate>b_...	2	100,00 %	0	0,00 %
AccommodationI...	40	51,28 %	38	48,72 %
AccommodationV...	11	32,35 %	23	67,65 %
MiscInfo(web.prot...	2	8,70 %	21	91,30 %
MiscValidate(web....	34	68,00 %	16	32,00 %
OnlineOrderWizar...	3	100,00 %	0	0,00 %
OnlineOrderWizar...	0	0,00 %	2	100,00 %
OrderTour(web.pr...	6	100,00 %	0	0,00 %
PartcipantsValidat...	4	9,09 %	40	90,91 %
ParticipantInfo(we...	0	0,00 %	34	100,00 %
TourInfo(web.prot...	2	6,90 %	27	93,10 %
WizardBase(Syste...	0	0,00 %	6	100,00 %
addParticipants(w...	13	100,00 %	0	0,00 %
getTour()	4	100,00 %	0	0,00 %
getTravelDays(Sys...	33	100,00 %	0	0,00 %
OnlineOrderWizardCo...	3	100,00 %	0	0,00 %

4. Go to source code and find the untested code using “source code coverage highlighting”.

```
public  void AccommodationValidate(Step3 step3)
{
    ModelState.Clear();
    List<OrderAccommodationType> accTypes = step3.Order.OrderAccommodations.SelectMany(x => x.Or
    if (accTypes.Count().Equals(0) && !accTypes.Any(x => x.isSelected))
    {
        ModelState.AddModelError("AccommodationType", "Valg indkvarteringstype");
    }
    List<Participant> accparticipants = accTypes.SelectMany(x => x.Participants).ToList();
    if (accparticipants.Count().Equals(0))
    {
        ModelState.AddModelError("AccParticipants", "Placere deltagerne i værelserne");
    }
```

Color Coding	Description
Light Blue	Indicates that the entire line of code was exercised in the test run.
Beige	Indicates that only a portion of the code blocks within the line of code were exercised in the test run.
Reddish Brown	Indicates that the line was not exercised in the test run.

Architecture pattern guidelines

- Implement depended objects through interface.
- Use Dependency injection for implementation of Inversion of control.
- Implement IDependencyResolver interface
- Ninject framework can be used to inject dependency into objects.
- Ninject available in NuGet
- Install Ninject.Mvc5,Ninject.Web.Common
- Register external services in app_start of web common.cs (see “Dependency injection with Ninject").

Unit testing examples

Note: After the code was extended it is necessary to rebuild unit test project in order to update fakes assembly.

Theory example

The [Theory] uses for data-driven test with several type of data source such as (MemberData, InlineData, ExcelData, SqlServerData, and OleDbData) for test inputs. The tests within the theory is true for particular set of data. The test below demonstrates theory with member data. The “ParticipantInfoStep” is a name of method that return data set.

```
[Theory, MemberData("ParticipantInfoStep")]
0 references | oleksandr klymchuk, 2 days ago | 2 changes
public void ParticipantInfoTest(StubStep2 step2, string nextBtn, string prevBtn,string viewDataType )
{
    //Arrange
    stubionlineorderDI = new StubIOnlineOrderDI();
    stubionlineorderDI.GetOrderGuid = value => { return new Order(); };
    stubionlineorderDI.UpdateParticipantsInAccTypeOrder = value => { return new Order(); };
    controller = new OnlineOrderWizardController(stubionlineorderDI);
    //Act
    var result = controller.ParticipantInfo(step2, nextBtn, prevBtn) as PartialViewResult;
    //Assert
    Assert.Equal(viewDataType, result.ViewData.Model.GetType().ToString());
}
```

Fact example

The test within the fact is always true.

```
[Fact]
0 references | oleksandr klymchuk, 2 days ago | 2 changes
public void TourInfoTest()
{
    //Arrange
    stubionlineorderDI = new StubIOnlineOrderDI();
    stubionlineorderDI.CreateOrderOnTourOrder = value => { return new Order(); };
    StubStep1 step1 = new StubStep1();
    step1.Order = new Order();
    controller = new OnlineOrderWizardController(stubionlineorderDI);
    //Act
    var result = controller.TourInfo(step1, "") as PartialViewResult;
    //Assert
    Assert.IsType<Step2>(result.ViewData.Model);
}
```

Inline data

The InlineData uses within a theory and it accepts any primitive data type. The below example demonstrates theory method with InlineData source

```
[Theory]
[InlineData("key", "value", "key", "_Step2ParticipantsInfo")]
[InlineData("key", "value", "", "_Step1TravelInfo")]
0 references | oleksandr klymchuk, 7 days ago | 1 change
public void TourInfoTest(string key,string value,string remove,string viewname)
{
    //Arrange
    stubionlineorderDI = new StubIOnlineOrderDI();
    stubionlineorderDI.CreateOrderOnTourOrder = par => { return new Order(); };
    stubionlineorderDI.StartOrderGuid = par => { return new Order(); };
    StubStep1 step1 = new StubStep1();
    step1.Order = new Order();
    controller = new OnlineOrderWizardController(stubionlineorderDI);
    controller.ModelState.AddModelError(key, value);
    controller.ModelState.Remove(remove);
    //Act
    var result = controller.TourInfo(step1, "") as PartialViewResult;
    //Assert
    Assert.Equal(viewname, result.ViewName);
}
```

Generic method in test

In order to isolate one part of application from another the stub could be used. The below method demonstrates how to remove dependency using stub and delegate type to assign fake implementation.

```
[Fact]
0 references | Oleksandr Klymchuk, 4 hours ago | 2 changes
public void TourInfoTest()
{
    //Arrange
    stubionlineorderDI = new StubIOnlineOrderDI();
    stubionlineorderDI.CreateOrderOnTourOrder = value => { return new Order(); };
    stubionlineorderDI.StepOf1Order<Step2>(delegate
    {
        return new Step2() { Order = new Order() };
    });
    StubStep1 step1 = new StubStep1();
    step1.Order = new Order();
    step1.Order.TravelDays = new List<TravelDay>() { new TravelDay()
    { Id = Guid.NewGuid(), StartDate = DateTime.Now, Departure = new Departure() { Name = "Billund" } } };
    controller = new OnlineOrderWizardController(stubionlineorderDI);
    //Act
    var result = controller.TourInfo(step1, "") as PartialViewResult;
    //Assert
    Assert.Equal("_Step2ParticipantsInfo", result.ViewName);

}
```

MemberData

The “MemberData” provides data source for theory. The member data could be retrieved from static field, static method or static property. The below examples demonstrates implementation of theory method that use MemberData as datasource that use static property.

```
[Theory,MemberData("ParticipantInfoStep")]
0 references | oleksandr klymchuk, 2 days ago | 2 changes
public void ParticipantInfoTest(StubStep2 step2, string nextBtn, string prevBtn, string viewName)
{
    //Arrange
    stubionlineorderDI = new StubIOnlineOrderDI();
    stubionlineorderDI.GetOrderGuid = value => { return new Order(); };
    controller = new OnlineOrderWizardController(stubionlineorderDI);
    //Act
    var result = controller.ParticipantInfo(step2,nextBtn,prevBtn) as PartialViewResult;
    //Assert
    Assert.Equal(viewName, result.ViewName);
}

0 references | oleksandr klymchuk, 2 days ago | 2 changes
public static IEnumerable<object[]> ParticipantInfoStep()
{
    StubStep2 step2 = new StubStep2();
    step2.Order = new Order();
    step2.Order.Participants = new List<Participant>()
    {new Participant(){FirstName="o",LastName="k",Gender=true,Birthday=DateTime.Now} };
    return new[]{
        new  object[]{step2,"",null,"_Step3AccommodationsInfo"},
        //validation modelstate false
        new  object[]{new StubStep2(){Order=new Order()},"",null,"_Step2ParticipantsInfo"},
        new  object[]{step2,null,"","_Step1TravelInfo"},
        new  object[]{step2,null,null,"_errorPage"}
   };
}
```

Web api test

Get

In the below method demonstrates testing of web api get method. Since the get method is static method it is necessary to use shim in order to remove dependency.

```
[Fact]
0 references | oleksandr klymchuk, 2 days ago | 1 change
public void GetTest()
{
    using (ShimsContext.Create())
    {
        //Arrange
        OrderController controller = new OrderController();
        ShimOrder.GetGuid = value =>
        { return orders.Where(x => x.Id == value).FirstOrDefault(); };
        //Act
        var result = controller.Get(orderId1);
        //Assert
        Assert.Equal(orderId1,result.Id);
    }

}
```

Post

The unit testing could be performed to test exception as well. In the example below demonstrates testing of returned message that returned by exception.

```
[Fact]
0 references | oleksandr klymchuk, 2 days ago | 1 change
public void PostTest()
{
  OrderController controller = new OrderController();
  var result= Assert.Throws<System.ArgumentException>(delegate { controller.Post(new Order()); });
  Assert.Equal("Both parameters 'connectionString' and 'cacheKey' cannot be empty at the same time.", result.Message);
}
```

Put

This examples demonstrates testing of exception type.

```
 [Fact]
0 references | oleksandr klymchuk, 2 days ago | 1 change
public void UpdateTest()
{
    OrderController controller = new OrderController();
    Assert.Throws<System.ArgumentException>(delegate{
        controller.Put(orders.FirstOrDefault(), orders.FirstOrDefault().Id);
    });
}
```

Mvc 5 test

ViewData test

In this example demonstrates testing of data type returned by controller action.

```
[Fact]
 | 0 references | oleksandr klymchuk, 2 days ago | 2 changes
public void WizardBaseTest()
{
    //Arrange
    stubionlineorderDI = new StubIOnlineOrderDI();
    stubionlineorderDI.StartOrderGuid = value => { return new Order(); };
    controller = new OnlineOrderWizardController(stubionlineorderDI);
    //Act
    var result = controller.WizardBase(Guid.NewGuid()) as ViewResult;
    //Assert
    Assert.IsType<Step1>(result.ViewData.Model);
}
```

ViewName test

In this example demonstrates testing of view name returned by controller action

```
[Fact]
 | 0 references | oleksandr klymchuk, 7 days ago | 1 change
public void WizardBaseTest()
{
    //Arrange
    stubionlineorderDI = new StubIOnlineOrderDI();
    stubionlineorderDI.StartOrderGuid = value => { return new Order(); };
    controller = new OnlineOrderWizardController(stubionlineorderDI);
    //Act
    var result = controller.WizardBase(Guid.NewGuid()) as ViewResult;
    //Assert
    Assert.Equal("WizardBase", result.ViewName);

}
```

ViewModel test

In the below method demonstrates testing of “getSelectedTravelDay” method of viewmodel.

```
[Theory,MemberData("traveldayInfo")]
0 references | oleksandr klymchuk, 2 days ago | 1 change
public void getSelectedTravelDayTest(Guid Id, Order Order)
{
    //Arange
    Step1 step1 = new Step1();
    step1.Order = Order;
    //Act
    var result = step1.getSelectedTravelDay();
    //Assert
    Assert.Equal(Id,result);

}
0 references | oleksandr klymchuk, 2 days ago | 1 change
public static IEnumerable<object[]> traveldayInfo()
{
    Guid Id = Guid.NewGuid();
    Order orderWithSelectedTravelDay = new Order() { TravelDays = new List<TravelDay>()
    {
        new TravelDay() { Selected = true, Departure = new Departure() { Id = Id } } }
    };
    Order orderWithoutSelectedTravelDay = new Order() { TravelDays = new List<TravelDay>() };
    return new[] {
        new object[]{Id, orderWithSelectedTravelDay},
        new object[]{Guid.Empty, orderWithoutSelectedTravelDay}
   };
}
```

DI test

In the below method demonstrates testing of the method that coupled by another static method. In this case it used shim to isolate unit from external dependency.

```
[Fact]
0 references | oleksandr klymchuk, 2 days ago | 1 change
public void startOrderShim()
{
    using (ShimsContext.Create())
    {
        //Arange
        OnlineOrderDI order = new OnlineOrderDI();
        Guid Id = Guid.NewGuid();
        ShimApi<Tour>.GetGuid = value =>
        { return new Tour() {Id=Id, TravelDays=new List<TravelDay>()}; };
        //Act
        var result = order.startOrder(Guid.NewGuid());
        Assert.Equal(Id, result.TourId);
    }

}
```

Dependency injection with Ninject

File generated by NuGet during installing Ninject.Mvc5

```
public static class NinjectWebCommon
{
    private static readonly Bootstrapper bootstrapper = new Bootstrapper();

    /// <summary>
    /// Starts the application
    /// </summary>
    0 references
    public static void Start() ...
    /// <summary>
    /// Stops the application.
    /// </summary>
    0 references
    public static void Stop()...
    /// <summary>
    /// Creates the kernel that will manage your application.
    /// </summary>
    /// <returns>The created kernel.</returns>
    1 reference
    private static IKernel CreateKernel()...

    /// <summary> ...
    1 reference
    private static void RegisterServices(IKernel kernel)
    {
        DependencyResolver.SetResolver(new NinjectDependencyResolver(kernel));
    }
}
```

Interface of mvc framework for service location and dependency resolution

```
public class NinjectDependencyResolver:IDependencyResolver
{
    private IKernel kernel;
    1 reference | 0 authors | 0 changes
    public NinjectDependencyResolver(IKernel kernel)
    {
        this.kernel = kernel;
        AddBindings();
    }
    0 references | 0 authors | 0 changes
    public   object GetService(Type serviceType)
    {
        return kernel.TryGet(serviceType);
    }
    0 references | 0 authors | 0 changes
    public IEnumerable<object> GetServices(Type serviceType)
    {
        return kernel.GetAll(serviceType);
    }
    1 reference | 0 authors | 0 changes
    internal void AddBindings()
    {
        kernel.Bind<IOnlineOrderService>().To<OnlineOrderService>();
    }
}
```

Inject dependency through contractor

```
public class OnlineOrderWizardController : Controller
{
    private IOnlineOrderService ionlineorderService;
    11 references | 2/10 passing | 0 authors | 0 changes
    public OnlineOrderWizardController(IOnlineOrderService IOnlineOrderService)
    {
        ionlineorderService = IOnlineOrderService;
    }
```

Note:without above DI the following declaration of controller will be through exception "**No parameterless constructor defined for this object**"