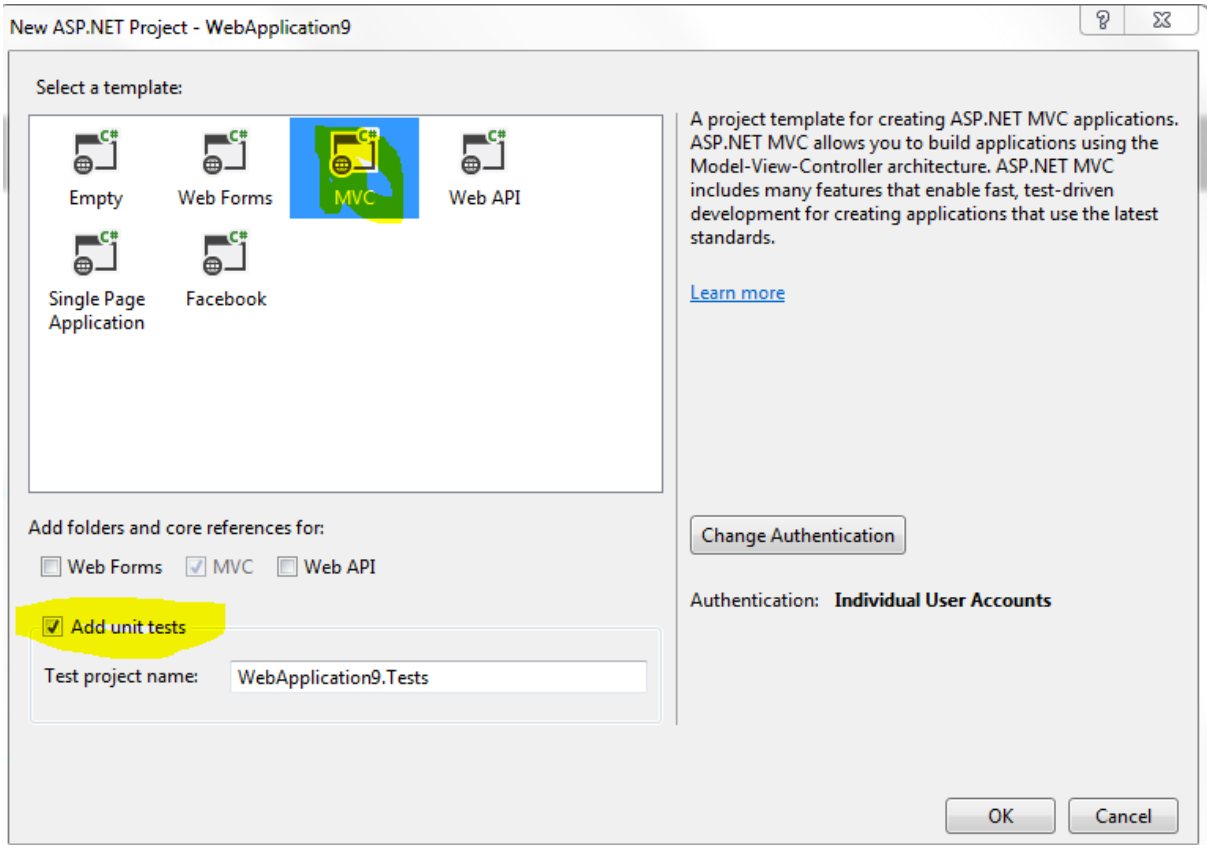
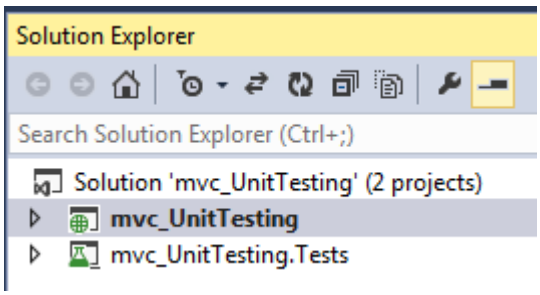
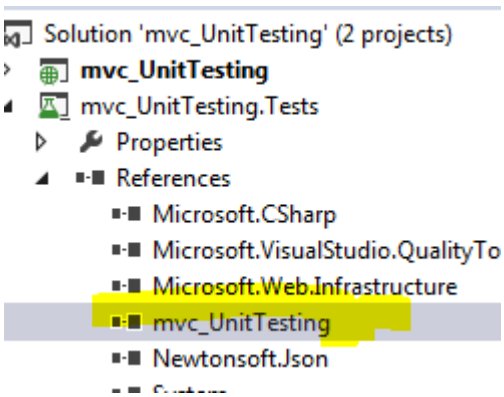
Add a Unit Test Project while creating mvc Project



2 projects will be created in Solution Explorer



The unit test project contains the reference of Main Project



Code Demonstration

Asp.Net MVC User interaction testing or Controller testing consist of

- Testing for ViewResult
- Testing of ViewData/ ViewBag
- Testing For RedirectResult


```
Add a controller to Main Project ProductController
```

```
namespace mvc_UnitTesting.Controllers
{
    public class ProductController : Controller
    {
        //
        // GET: /Product/
        public ActionResult Index()
        {
            // Add action logic here
            throw new NotImplementedException();
        }

        public ActionResult Details(int Id)
        {

            return View("Details");
        }
       }
}
```

Imagine that we want to test whether or not the `ProductController` returns the right view. We want to make sure that when the`ProductController.Details()` action is invoked, the Details view is returned. The test class in Listing 2 contains a unit test for testing the view returned by the `ProductController.Details()` action.

Add a claas "**ProductControllerTest**" to unit test Project to test

```
using Microsoft.VisualStudio.TestTools.UnitTesting;
using mvc_UnitTesting.Controllers;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Web.Mvc;
```

```
namespace mvc_UnitTesting.Tests.Controllers
{
        [TestClass]
    class ProductControllerTest
    {
        [TestMethod]
        public void TestDetailsView()
        {
            //Arrange
            var controller = new ProductController();

            //Act
            var result = controller.Details(2) as ViewResult;

            //Asert
            Assert.AreEqual("Details", result.ViewName);

        }
    }
}
```

The class in Listing 2 includes a test method named `TestDetailsView()`. This method contains three lines of code.
The first line of code creates a new instance of the `ProductController` class.
The **second line** of code invokes the controller's `Details()` action method.
Finally, **the last line** of code checks whether or not the view returned by the `Details()` action is the Details view.

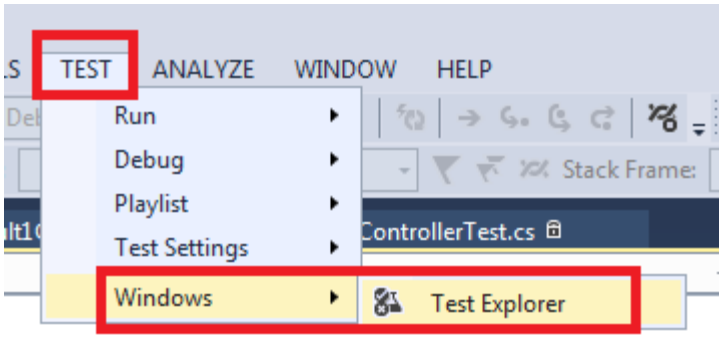
The `ViewResult.ViewName` property represents the name of the view returned by a controller. One big warning about testing this property.
There are two ways that a controller can return a view. A controller can explicitly return a view like this:

```
public ActionResult Details(int Id)
{
     return View("Details");
}
```

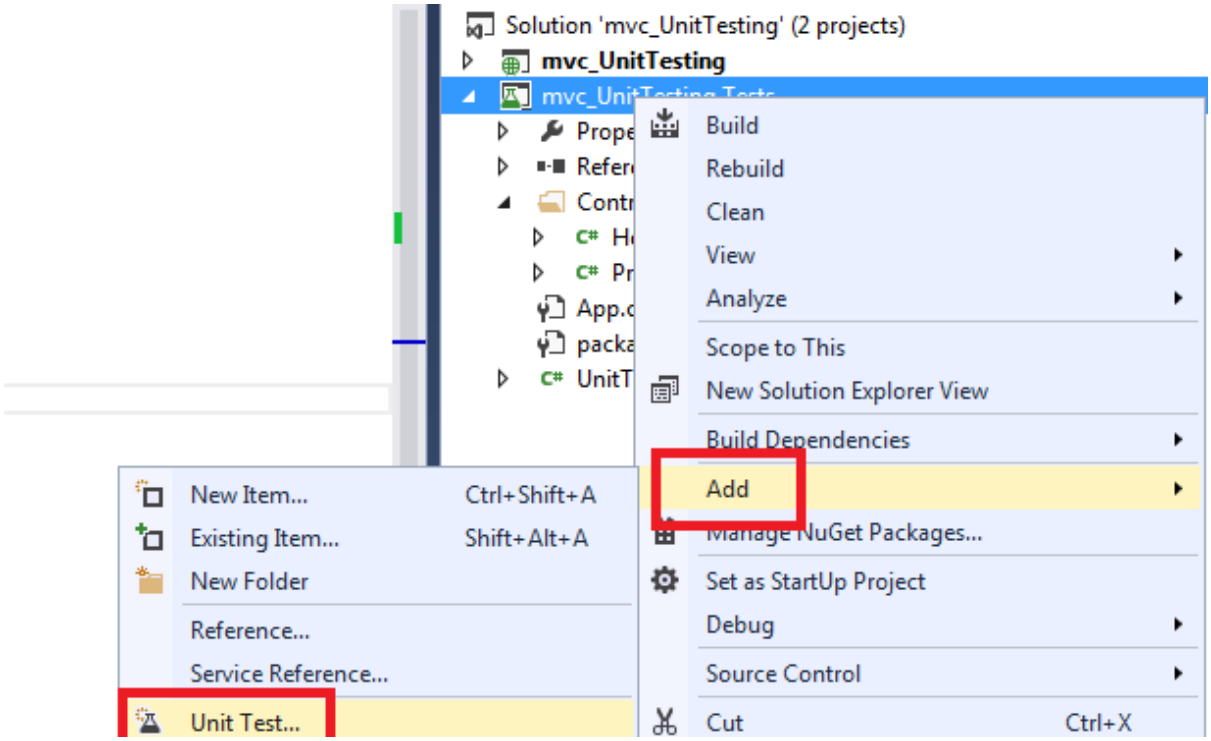
Alternatively, the name of the view can be inferred from the name of the controller action like this:

```
public ActionResult Details(int Id)
{
     return View();
}
```

Go to test window



Note We may face some problem while adding class and convering it test class. So we should add Unit Test as below and modify the logic in it.



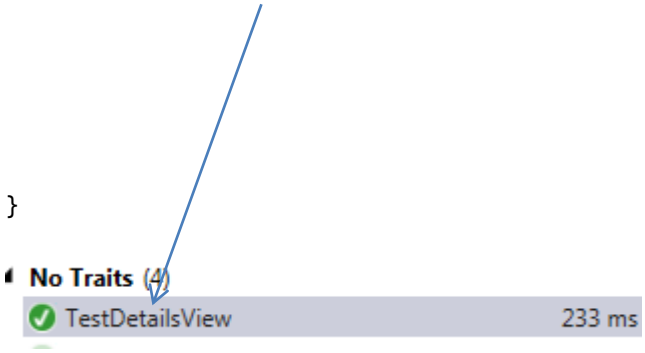

```
namespace mvc_UnitTesting.Tests
{
    [TestClass]
    public class UnitTest1
    {
        [TestMethod]
        public void TestDetailsView()
        {
            //Arrange
            var controller = new ProductController();

            //Act
            var result = controller.Details(2) as ViewResult;

            //Asert
            Assert.AreEqual("Details", result.ViewName);

        }
    }
```

```
}
```



- Testing the View Data returned by a Controller(Testing of ViewData/ ViewBag)

Add model class

```
public class Product
    {
        public int Id { get; set; }
        public string Name { get; set; }
    }
```

Add a new Controller “ProductNewWithParameter”

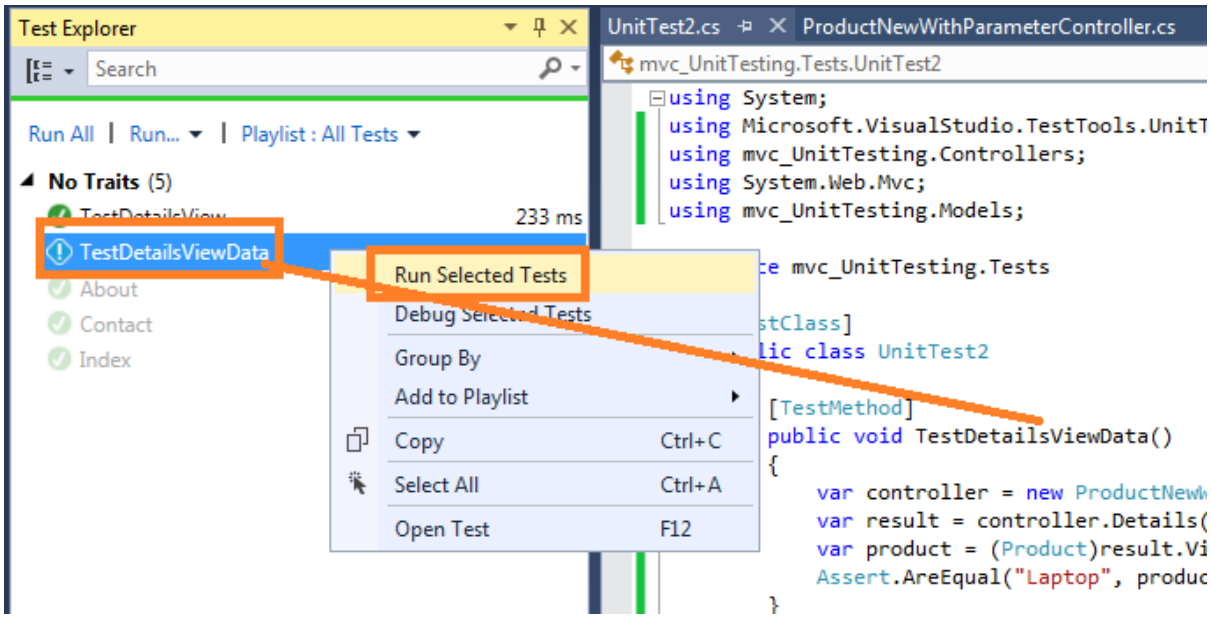
```
namespace mvc_UnitTesting.Controllers
{
    public class ProductNewWithParameterController : Controller
    {
        //
        // GET: /ProductNewWithParameter/
        public ActionResult Index()
        {
            // Add action logic here
            throw new NotImplementedException();
        }

        public ActionResult Details(int Id)
        {
            var product = new Product();
            product.Id = Id;
            product.Name = "Laptop";
            return View("Details", product);
        }
       }
}
```

Add a Unit Test to Test this and modify the Logic to test

```
namespace mvc_UnitTesting.Tests
{
    [TestClass]
    public class UnitTest2
    {
        [TestMethod]
        public void TestDetailsViewData()
        {
            var controller = new ProductNewWithParameterController();
            var result = controller.Details(2) as ViewResult;
            var product = (Product)result.ViewData.Model;
            Assert.AreEqual("Laptop", product.Name);
        }
    }
}
```

the `TestDetailsView()` method tests the View Data returned by invoking the `Details()` method. The `ViewData` is exposed as a property on the `ViewResult` returned by invoking the `Details()` method. The `ViewData.Model` property contains the product passed to the view. The test simply verifies that the product contained in the View Data has the name Laptop.

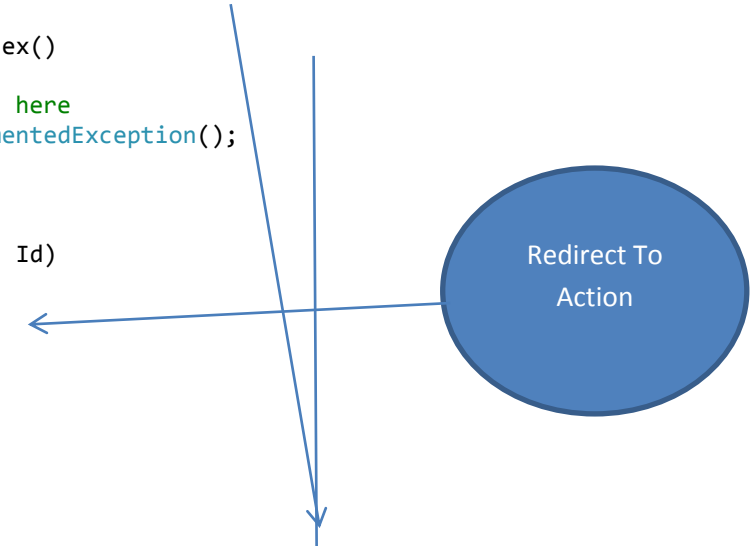


Testing the Action Result returned by a Controller(**RedirectResult**)

Add a controller to test this

```
namespace mvc_UnitTesting.Controllers
{
    public class ProductWithRedirectResultController : Controller
    {
        public ActionResult Index()
        {
            // Add action logic here
            throw new NotImplementedException();
        }


public ActionResult Details(int Id)
        {
            if (Id < 1)
```



```
            return RedirectToAction("Index");
        var product = new Product();
        product.Id = Id;
        product.Name = "Laptop";
        return View("Details", product);
    }
}
```

Add a Unit Test to Test this and modify the Logic to test

```
using System;
using Microsoft.VisualStudio.TestTools.UnitTesting;
using mvc_UnitTesting.Controllers;
using System.Web.Mvc;

namespace mvc_UnitTesting.Tests
{
    [TestClass]
    public class UnitTest3
    {
        [TestMethod]
        public void TestDetailsRedirect()
        {
            var controller = new ProductWithRedirectResultController();
            var result = (RedirectToRouteResult)controller.Details(-1);
            Assert.AreEqual("Index", result.RouteValues["action"]);

        }
    }
}
```