CollectionAssert

<https://msdn.microsoft.com/en-us/library/microsoft.visualstudio.testtools.unittesting.collectionassert.aspx>

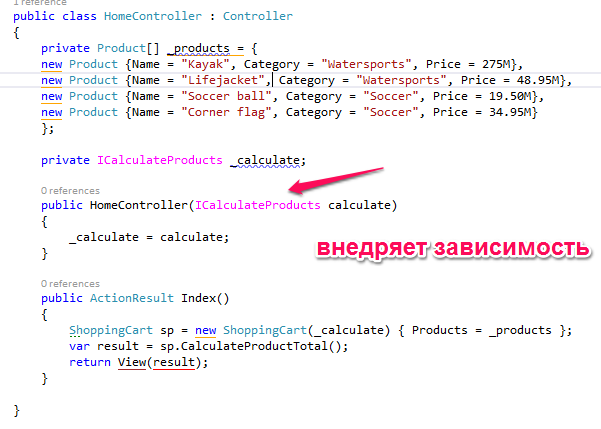
замена свойства класса

Bind<IMailMessageTo>().To<MailMessageTo>();

Bind<IMailSender>().To<MailSender>().

WithPropertyValue("message",Kernel.GetService(typeof(IMailMessageTo)));

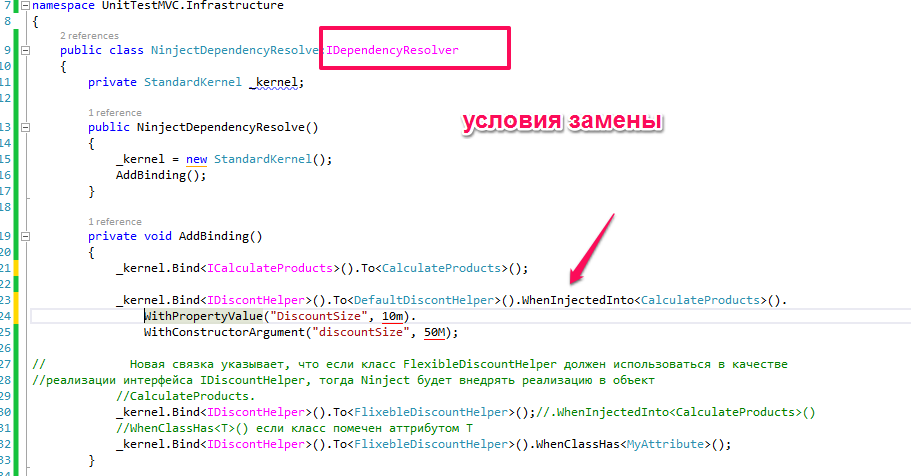
Контроллер

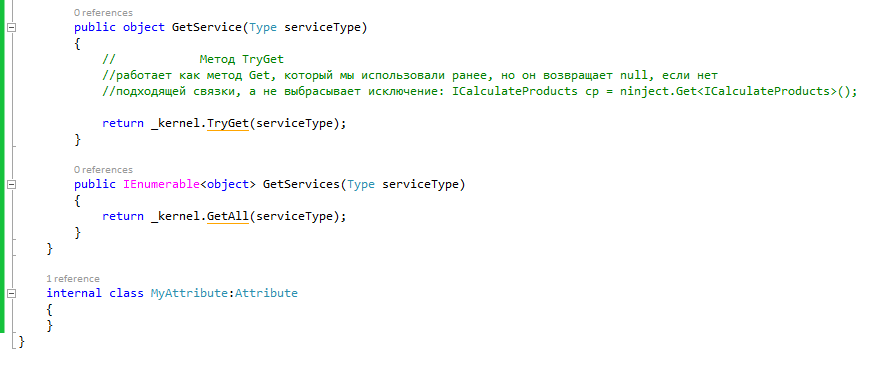


В файле Global.asax

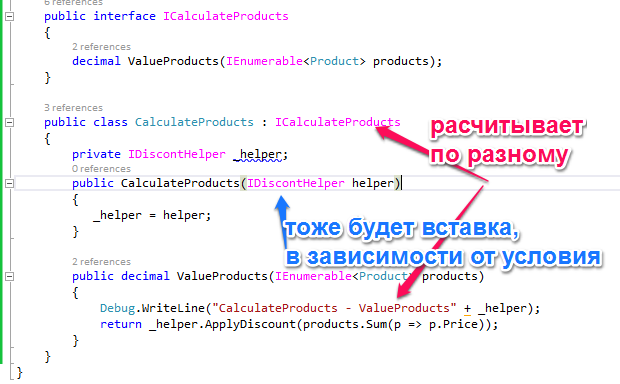


Ninject

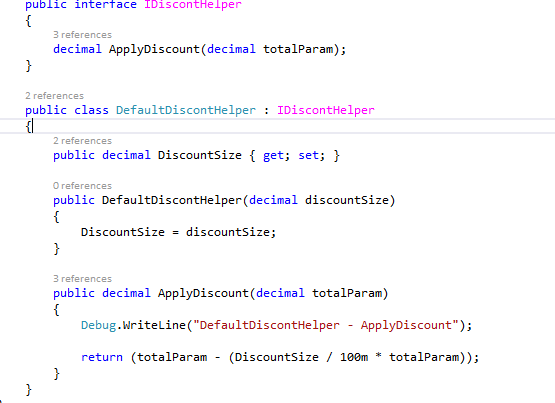




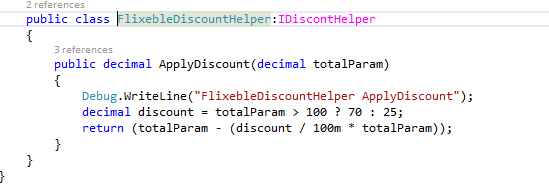
Калькулятор



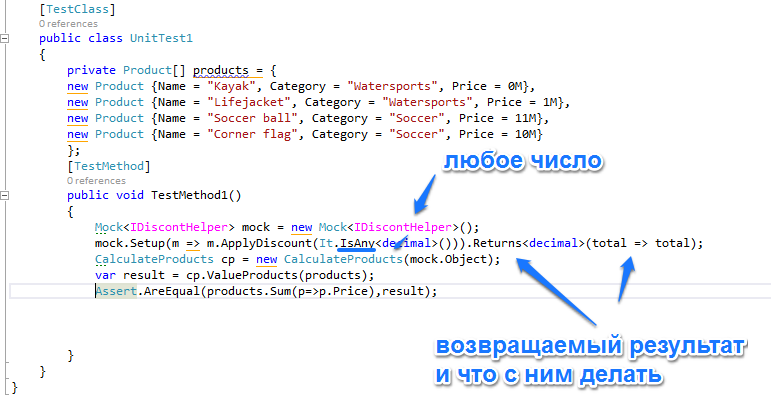
Разные реализации

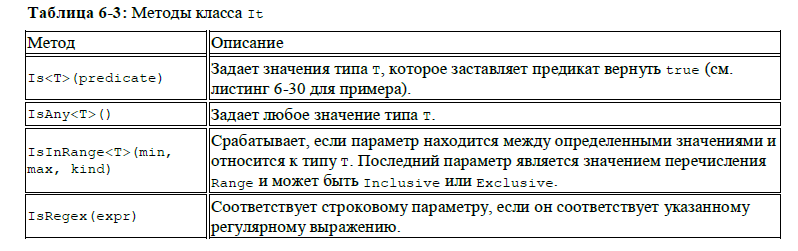


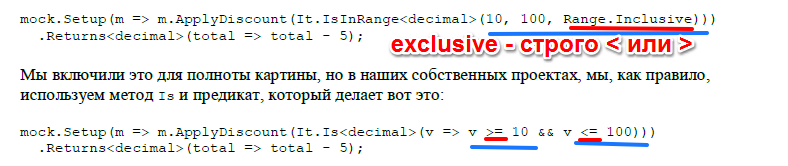
Или

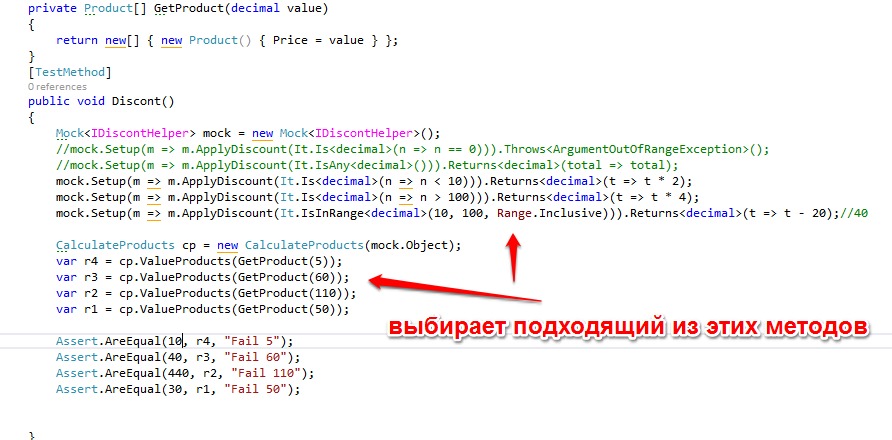


Moq

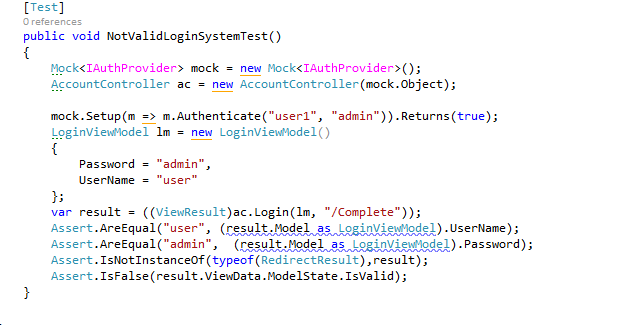








AccountController



AdminTest

[TestFixture]

public class AdminTest

{

private Mock<IProductRepository> \_mock;

[TestFixtureSetUp]

public void Initialize()

{

\_mock = new Mock<IProductRepository>();

\_mock.Setup(m => m.Products).Returns(new List<Product>

{

new Product(){Price=12,ProductID = 1,Category= "Football"},

new Product(){Price=10,ProductID = 2,Category= "Tennis",Name = "Rocket"},

new Product(){Price=13,ProductID = 3,Category= "Basketball"},

new Product(){Price=19,ProductID = 4,Category= "Football"},

new Product(){Price=18,ProductID = 5,Category= "Hokey"},

new Product(){Price=15,ProductID = 6,Category= "Basseball"},

new Product(){Price=10,ProductID = 7,Category= "Wrestling"},

new Product(){Price=12,ProductID = 8,Category= "Hokey"},

new Product(){Price=9,ProductID = 9,Category= "Football"},

new Product(){Price=4,ProductID = 10,Category = "Football"}

}.AsQueryable());

}

[Test]

public void GetAllProducts()

{

AdminController ac = new AdminController(\_mock.Object);

var result = ((ViewResult)ac.Index()).Model as IQueryable<Product>;

Assert.AreEqual(10, result.Count());

Assert.AreEqual(1, result.First().ProductID);

}

[Test]

public void EditProducts()

{

AdminController ac = new AdminController(\_mock.Object);

int id = 7;

var result = ac.Edit(id).Model as Product;

Assert.AreEqual(id, result.ProductID);

Assert.AreEqual("Wrestling", result.Category);

ac = new AdminController(\_mock.Object);

id = 11;

//если Model уже содержит в себе значение и мы пытаемся присвоит null то результат останется

//прежний, отличный от null

var result1 = ac.Edit(id).Model as Product;

Assert.IsNull(result1);

}

[Test]

public void EditForSave()

{

AdminController ac = new AdminController(\_mock.Object);

Product product = new Product() { Name = "Ball" };

var view = (RedirectToRouteResult)ac.Edit(product,null);

\_mock.Verify(p => p.SaveProduct(product));

Assert.That(view.RouteValues["action"], Is.EqualTo("Index"));

Assert.AreEqual(string.Format("{0} has been saved", product.Name), ac.TempData["message"]);

}

[Test]

public void EditForSaveNotValid()

{

AdminController ac = new AdminController(\_mock.Object);

ac.ModelState.AddModelError("Error", "Error");

Product product = new Product() { Name = "Ball" };

var model = ((ViewResult)ac.Edit(product,null)).Model as Product;

\_mock.Verify(p => p.SaveProduct(product), **Times**.Never);

Assert.That(model.Name, Is.EqualTo("Ball"));

}

[Test]

public void DeleteProduct()

{

AdminController ac = new AdminController(\_mock.Object);

var startCount = \_mock.Object.Products.Count();

\_mock.Setup(m => m.DeleteProduct(It.IsAny<int>())).Returns((int i) =>\_mock.Object.Products.First(n=>n.ProductID==i));

var view = (RedirectToRouteResult)ac.Delete(2);

\_mock.Verify(p => p.DeleteProduct(2), **Times**.Once);

Assert.That(view.RouteValues["action"], Is.EqualTo("Index"));

Assert.AreEqual(string.Format("{0} товар был удален", \_mock.Object.Products.First(n => n.ProductID == 2).Name),

ac.TempData["message"]);

//Assert.That(model.Count(), Is.EqualTo(startCount - 1));

//Assert.That(model.Select(n=>n.ProductID), Is.Not.Contains(2));

}

}

ImageTest

[TestFixture]

public class ImageTest

{

private Mock<IProductRepository> \_mock;

[TestFixtureSetUp]

public void Initialize()

{

\_mock = new Mock<IProductRepository>();

\_mock.Setup(m => m.Products).Returns(()=> new List<Product>

{

new Product(){ ProductID = 1,ImageData = new byte[]{},ImageMimeType = "image/png"},

new Product(){ ProductID = 2,ImageData = new byte[]{},ImageMimeType = "image/jpeg"},

new Product(){ ProductID = 3,ImageData = new byte[]{},ImageMimeType = "image/bin"}

}.AsQueryable());

}

[Test]

public void GetImageTest()

{

int id = 2;

ProductController pc = new ProductController(\_mock.Object);

ActionResult result = pc.GetImage(id);

Assert.IsNotNull(result);

Assert.IsNotInstanceOf(result.GetType(),typeof(FileResult));

Assert.AreSame(\_mock.Object.Products.First(n=>n.ProductID==id).ImageMimeType,((FileContentResult)result).ContentType);

}

[Test]

public void GetNonImageTest()

{

int id = 10;

ProductController pc = new ProductController(\_mock.Object);

ActionResult result = pc.GetImage(id);

Assert.IsNull(result);

}

}

CartTest

[TestFixture]

public class CartTest

{

private Mock<IProductRepository> \_mock;

[TestFixtureSetUp]

public void Initialize()

{

\_mock=new Mock<IProductRepository>();

\_mock.Setup(m => m.Products).Returns(new List<Product>

{

new Product(){Price=12,ProductID = 1,Category= "Football"},

new Product(){Price=10,ProductID = 2,Category= "Tennis"},

new Product(){Price=13,ProductID = 3,Category= "Basketball"},

new Product(){Price=19,ProductID = 4,Category= "Football"},

new Product(){Price=18,ProductID = 5,Category= "Hokey"},

new Product(){Price=15,ProductID = 6,Category= "Basseball"},

new Product(){Price=10,ProductID = 7,Category= "Wrestling"},

new Product(){Price=12,ProductID = 8,Category= "Hokey"},

new Product(){Price=9,ProductID = 9,Category= "Football"},

new Product(){Price=4,ProductID = 10,Category = "Football"}

}.AsQueryable());

}

#region Controller checkout

[Test]

public void CheckOutInController()

{

Mock<IOrderProcess> mock = new Mock<IOrderProcess>();

Cart cart = new Cart();

CartController cc = new CartController(null, mock.Object);

ViewResult result = (ViewResult)cc.CheckOut(cart, new ShippingDetails() { Name = "Vitek" });

//Проверка того, что метод ILogWriter.Write вызвался в точности один раз (ни больше, ни меньше):

//mock.Verify(lw => lw.Write(It.IsAny<string>()),Times.Once());

//if (!cart.Lines.Any())//добавили ошибку если корзина пуста

//{

// ModelState.AddModelError("", "Ваша корзина пуста");

//}

mock.Verify(m => m.ProcessOrder(It.IsAny<Cart>(), It.IsAny<ShippingDetails>()), **Times**.Never);

var details = (ShippingDetails) result.Model;

Assert.AreEqual("", result.ViewName);

Assert.AreEqual("Vitek", details.Name);

Assert.AreEqual(false.ToString(), result.ViewData.ModelState.IsValid.ToString());

}

[Test]

public void CheckOutIfCountCartAny()

{

Mock<IOrderProcess> mock = new Mock<IOrderProcess>();

Cart cart = new Cart();

cart.AddItem(\_mock.Object.Products.First(), 1);

CartController cc = new CartController(null, mock.Object);

ViewResult result = (ViewResult) cc.CheckOut(cart, new ShippingDetails());

// Проверка того, что метод ILogWriter.Write вызвался в точности один раз (ни больше, ни меньше):

//mock.Verify(lw => lw.Write(It.IsAny<string>()),Times.Once());

mock.Verify(m => m.ProcessOrder(It.IsAny<Cart>(), It.IsAny<ShippingDetails>()));

Assert.AreEqual("Completed", result.ViewName);

Assert.AreEqual(false, cart.Lines.Any());

Assert.AreEqual(true.ToString(), result.ViewData.ModelState.IsValid.ToString());

}

[Test]

public void CheckOutIsNoValidModel()

{

Mock<IOrderProcess> mock = new Mock<IOrderProcess>();

Cart cart = new Cart();

cart.AddItem(\_mock.Object.Products.First(), 1);

CartController cc = new CartController(null, mock.Object);

cc.ModelState.AddModelError("Error","No Name");

ViewResult result = (ViewResult)cc.CheckOut(cart, new ShippingDetails());

// Проверка того, что метод ILogWriter.Write вызвался в точности один раз (ни больше, ни меньше):

//mock.Verify(lw => lw.Write(It.IsAny<string>()),Times.Once());

mock.Verify(m => m.ProcessOrder(It.IsAny<Cart>(), It.IsAny<ShippingDetails>()),**Times**.Never);

Assert.AreEqual("", result.ViewName);

Assert.AreEqual(true, result.ViewData.ModelState.ContainsKey("Error"));

Assert.AreEqual("No Name", result.ViewData.ModelState["Error"].Errors.First().ErrorMessage);

Assert.AreEqual(false.ToString(), result.ViewData.ModelState.IsValid.ToString());

}

#endregion

#region Add and Remove product

[Test]

public void AddProductToCartInConroller()

{

Cart cart = new Cart();

CartController cc = new CartController(\_mock.Object, null);

var router = cc.AddToCart(cart, 2, "Tennis/Page");

Assert.AreEqual(1, cart.Lines.Count());

Assert.AreEqual("Tennis", cart.Lines.ElementAt(0).Product.Category);

Assert.AreEqual("Index", router.RouteValues["action"]);

Assert.AreEqual("Tennis/Page", router.RouteValues["returnUrl"]);

}

[Test]

public void RemoveProductFromCartInConroller()

{

Cart cart = new Cart();

cart.AddItem(\_mock.Object.Products.First(), 2);

CartController cc = new CartController(\_mock.Object, null);

var router = cc.RemoveFromCart(cart, 2, "Football/Page");

Assert.AreEqual(1, cart.Lines.Count());

Assert.AreEqual("Index", router.RouteValues["action"]);

Assert.AreEqual("Football/Page", router.RouteValues["returnUrl"]);

}

#endregion

#region Test class Cart

[Test]

public void AddCartTest()

{

Cart cart = new Cart();

var collection = \_mock.Object.Products.ToArray();

cart.AddItem(collection[0], 2);

cart.AddItem(collection[0], 4);

cart.AddItem(collection[5], 1);

cart.AddItem(collection[5], 3);

Assert.AreEqual(2, cart.Lines.Count());

}

[Test]

public void CartTotalValueTest()

{

Cart cart = new Cart();

var collection = \_mock.Object.Products.ToArray();

cart.AddItem(collection[0], 2);

cart.AddItem(collection[0], 4);

cart.AddItem(collection[5], 1);

cart.AddItem(collection[5], 3);

Assert.AreEqual(6\*collection[0] + 4\*collection[5], cart.ComputeTotalValue());

Assert.AreEqual(2 + 4 + 1 + 3, cart.Lines.Sum(n => n.Quantity));

}

[Test]

public void RemoveItemTest()

{

Cart cart = new Cart();

var collection = \_mock.Object.Products.ToArray();

cart.AddItem(collection[0], 2);

cart.AddItem(collection[0], 4);

cart.AddItem(collection[5], 1);

cart.AddItem(collection[5], 3);

cart.RemoveLine(collection[0]);

Assert.AreEqual(1, cart.Lines.Count());

Assert.AreEqual(collection[5].ProductID, cart.Lines.First().Product.ProductID);

}

#endregion

}