**Steps to Dependency Injection – Example 2  
  
Alter the Application**

using System.Web.Mvc;

using EssentialTools.Models;

namespace EssentialTools.Controllers

{

public class HomeController : Controller

{

private Product[] products = {

new Product {Name = "Kayak", Category = "Watersports", Price = 275M},

new Product {Name = "Lifejacket", Category = "Watersports", Price = 48.95M},

new Product {Name = "Soccer ball", Category = "Soccer", Price = 19.50M},

new Product {Name = "Corner flag", Category = "Soccer", Price = 34.95M}

};

* Remove the original statement that created   
  the object.
* Add a declaration for the object at the top of the class. Use an interface.
* **Add a constructor** that accepts the object as a parameter.

public ActionResult Index()

{

IValueCalculator calc = new LinqValueCalculator();

ShoppingCart cart = new ShoppingCart(calc) { Products = products };

decimal totalValue = cart.CalculateProductTotal();

return View(totalValue);

}

}

}

**Set Up Ninject**

* Install Ninject.MVC5 using NuGet. **Use version 3.2.1!**
* Add a folder called Infrastructure and create in there a class called NinjectDependencyResolver.cs. Copy code from Canvas and paste in code. Add the bindings in AddBindings().   
  kernel.Bind< >().To< >();
* In the RegisterServices method in NinjectWebCommon.cs, tell the application where to find the resolver.

System.Web.Mvc.DependencyResolver.SetResolver(new   
 *YourProjectNameHere*.Infrastructure.NinjectDependencyResolver(kernel));