Entity Framework Lab

1. Create an MVC project called CarsForSale.
2. Install Ninject and EntityFramework.
3. Add a controller called CarController with an action method called InventoryList.
4. Alter the RouteConfig.cs file so that CarController will be the default controller and InventoryList will be the default action method.
5. Create a model class called Car.cs that has the following properties: CarID, CarMake, CarModel, Year, Miles, Color, Price
6. Now create a database file in App_Data that has one table called Cars. The table should have a column called CarID that is set as a primary key identity as well as six other columns corresponding to the properties in the Car.cs class. Add at least 3 records to the table.
7. Add a connection string to Web.Config.
8. Create an interface called ICarRepository.cs in the Models folder. This interface should be public and have a property called Cars that is a collection (IEnumerable) of Car objects. There should be no "setter" for this property. (Use IProductRepository in the notes for a reference.)
9. Create the EFDbContext.cs class in the Models folder that **inherits from DbContext**. This class also should have a property called Cars that is a DbSet of Car objects.

   ```
   public DbSet<Car> Cars { get; set; }
   ```

   Be sure and add a using statement for System.Data.Entity.
10. Also in the Models folder, create a class called EFCarRepository.cs that **implements the ICarRepository** interface. (You will need a *using* statement so that your interface will be found.) Like ICarRepository, this should also have a property called Cars that is a collection (IEnumerable) of Car objects. In this case, though, the "getter" will return the rows of the database by using an instance of the EFDbContext.

    ```
    private EFDbContext context = new EFDbContext();
    public IEnumerable<Car> Cars
    {
        get { return context.Cars; }
    }
    ```

11. Create a constructor for CarController that accepts an implementation of ICarRepository. Then in the InventoryList method, send the Cars property of the implementation (repository.Cars) into the View method as a parameter.
12. Now create the InventoryList view with a @model IEnumerable<Car> directive and any @using directives that are necessary and add Razor code to go through the collection of Car objects and display all of the property values of each object on one line.
13. Create a NinjectDependencyResolver and bind the EFCarRepository to ICarRepository. When you copy and paste the code for NinjectDependencyResolver.cs, don't forget to change the namespace to match the name of your project.
14. As we've done before, add a line in the RegisterServices method of NinjectWebCommon.cs to tell Ninject where to find the NinjectDependencyResolver.
15. Run the application to verify that it will list the records in your Cars table.