# Working with data part 2

In this lesson we will look at defining contracts using [Interfaces](https://docs.microsoft.com/en-us/dotnet/csharp/language-reference/keywords/interface) that we create along with a built-in interface [INotifyPropertyChanged](https://docs.microsoft.com/en-us/dotnet/api/system.componentmodel.inotifypropertychanged?view=net-5.0) which provides notification of client changes to properties in a backend data class.

The objective is to return a single customer (we defined in part 1) using projections (as shown in part 1)

1. From solution Explorer, right click on the project NorthWindCoreLibrary -> select Manage NuGet Packages.
   1. If the browse tab is not selected, select the browse tab
   2. Single click on Newtonsoft.Json
   3. Click install, follow prompts then close the window
2. From solution Explorer, right click on the project NorthWindCoreLibrary -> select Classes folder.
3. Add a new class named BaseEntity.
4. Replace the contents in BaseEntity with the following code

|  |
| --- |
| using System;  using Newtonsoft.Json;    namespace NorthWindCoreLibrary.Classes  {      public class BaseEntity      {          [JsonIgnore]          public DateTime? CreatedAt { get; set; }          [JsonIgnore]          public string CreatedBy { get; set; }          public DateTime? LastUpdated { get; set; }          [JsonIgnore]          public string LastUser { get; set; }          [JsonIgnore]          public bool? IsDeleted { get; set; }      }  } |

What is the attribute JsonIgnore? Let’s look at the [definition](https://www.newtonsoft.com/json/help/html/ReducingSerializedJSONSize.htm).

BaseEntity will be implemented in a new class CustomerEntity.

1. Create a new class under the Classes folder named CustomerEntity
2. Copy the following code into CustomerEntity, replacing the current content

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.ComponentModel.DataAnnotations;

using System.Linq;

using System.Runtime.CompilerServices;

using System.Text;

using System.Threading.Tasks;

using Newtonsoft.Json;

using NorthWindCoreLibrary.Models;

namespace NorthWindCoreLibrary.Classes

{

    namespace North.Classes

    {

        public class CustomerEntity : BaseEntity, INotifyPropertyChanged

        {

            private int \_customerIdentifier;

            private string \_companyName;

            private int? \_contactIdentifier;

            private string \_firstName;

            private string \_lastName;

            private int \_contactTypeIdentifier;

            private string \_contactTitle;

            private string \_address;

            private string \_city;

            private string \_postalCode;

            private int? \_countryIdentifier;

            private string \_countyName;

            private Contacts \_contacts;

            public int CustomerIdentifier

            {

                get => \_customerIdentifier;

                set

                {

                    \_customerIdentifier = value;

                    OnPropertyChanged();

                }

            }

            [Required]

            public string CompanyName

            {

                get => \_companyName;

                set

                {

                    \_companyName = value;

                    OnPropertyChanged();

                }

            }

            public int? ContactIdentifier

            {

                get => \_contactIdentifier;

                set

                {

                    \_contactIdentifier = value;

                    OnPropertyChanged();

                }

            }

            public Contacts Contacts

            {

                get => \_contacts;

                set

                {

                    \_contacts = value;

                    OnPropertyChanged();

                }

            }

            [JsonIgnore]

            public string FirstName

            {

                get => \_firstName;

                set

                {

                    \_firstName = value;

                    OnPropertyChanged();

                }

            }

            [JsonIgnore]

            public string LastName

            {

                get => \_lastName;

                set

                {

                    \_lastName = value;

                    OnPropertyChanged();

                }

            }

            public string ContactName => $"{FirstName} {LastName}";

            public int? ContactTypeIdentifier

            {

                get => \_contactTypeIdentifier;

                set

                {

                    \_contactTypeIdentifier = (int) value;

                    OnPropertyChanged();

                }

            }

            public string ContactTitle

            {

                get => \_contactTitle;

                set

                {

                    \_contactTitle = value;

                    OnPropertyChanged();

                }

            }

            public string Street

            {

                get => \_address;

                set

                {

                    \_address = value;

                    OnPropertyChanged();

                }

            }

            public string City

            {

                get => \_city;

                set

                {

                    \_city = value;

                    OnPropertyChanged();

                }

            }

            public string PostalCode

            {

                get => \_postalCode;

                set

                {

                    \_postalCode = value;

                    OnPropertyChanged();

                }

            }

            public int? CountryIdentifier

            {

                get => \_countryIdentifier;

                set

                {

                    \_countryIdentifier = value;

                    OnPropertyChanged();

                }

            }

            public string CountryName

            {

                get => \_countyName;

                set

                {

                    \_countyName = value;

                    OnPropertyChanged();

                }

            }

            public int? ContactId { get; set; }

            public Countries CountryNavigation { get; set; }

            public override string ToString()

            {

                return CustomerIdentifier.ToString();

            }

            public event PropertyChangedEventHandler PropertyChanged;

            protected virtual void OnPropertyChanged([CallerMemberName] string propertyName = null)

            {

                PropertyChanged?.Invoke(this, new PropertyChangedEventArgs(propertyName));

            }

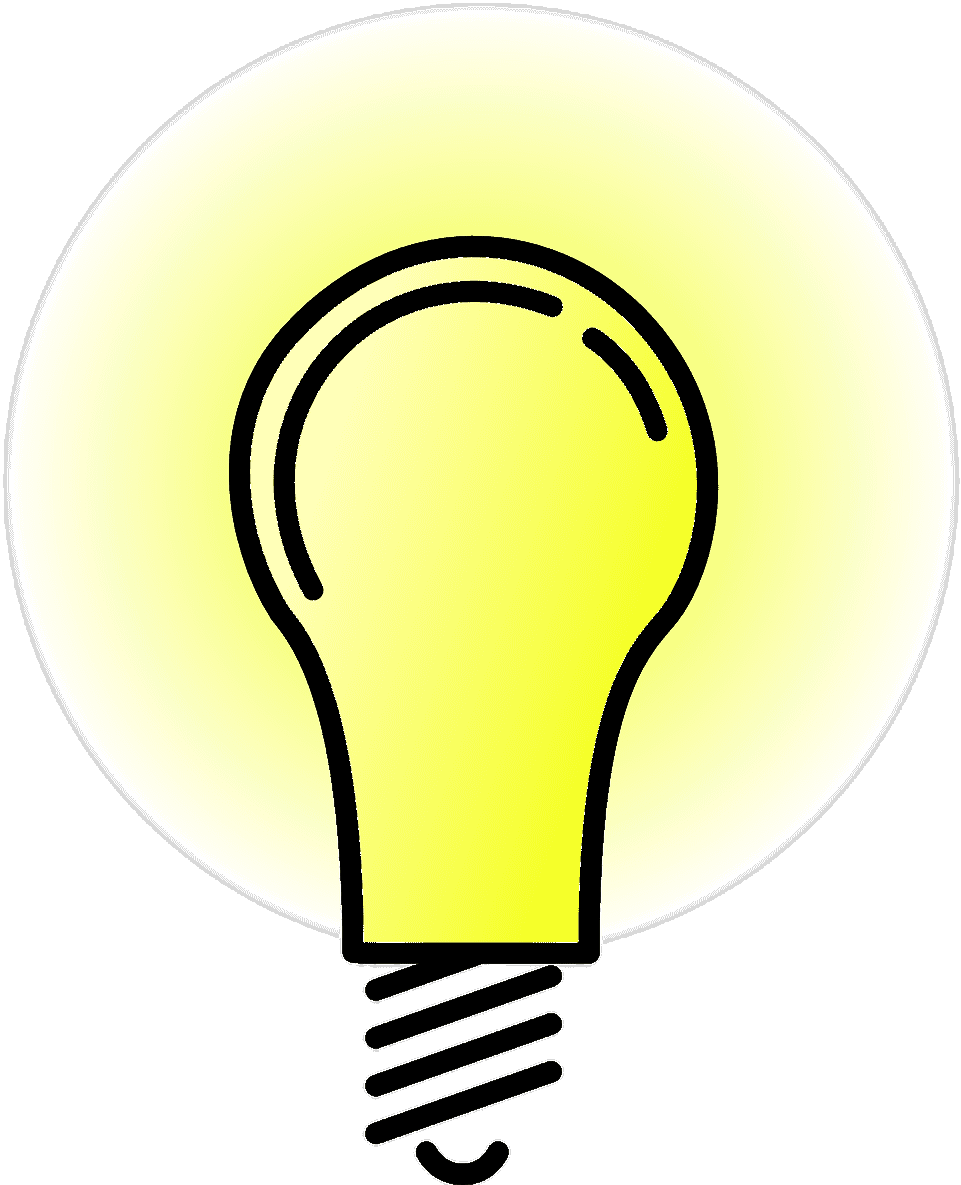
        }

    }

}

Dissecting CustomerEntity class

* By implementing BaseEntity we now have those properties exposed.
* [INotifyPropertyChanged](https://docs.microsoft.com/en-us/dotnet/api/system.componentmodel.inotifypropertychanged?view=net-5.0) provides notification of backend code to client code changes e.g. change CompanyName property in the frontend and the backend is aware of the change to CompanyName while without implementing INotifyPropertyChanged the backend is unaware of any client side changes to CompanyName.
* [JsonIgnore] indicated to ignore a property from EF Core operations and also when serializing/desterilizing data.
* Properties, each property has a
  + get to return the current value of the property
  + set which when a property is changed sets the value for reads followed by a call to OnPropertyChanged();
* OnPropertyChanged notifies backend that a change has been make (there are permutations on this, ignore them)
  + [[CallerMemberName](https://docs.microsoft.com/en-us/dotnet/api/system.runtime.compilerservices.callermembernameattribute?view=net-5.0)] string propertyName = null - Allows you to obtain the method or property name of the caller to the method.
  + PropertyChanged?.Invoke(this, new PropertyChangedEventArgs(propertyName)); does the actual notification.

 [ObservableCollection](https://docs.microsoft.com/en-us/dotnet/api/system.collections.objectmodel.observablecollection-1?view=net-5.0) is a class that can do what INotifyPropertyChanged does but this class may become obsolete.

* Open Customers class under Models folder, replace the current content with the following which is adding a Projection which we created one in part 1 lesson.

// <auto-generated> This file has been auto generated by EF Core Power Tools. </auto-generated>

using System;

using System.Collections.Generic;

using System.Linq.Expressions;

using NorthWindCoreLibrary.Classes.North.Classes;

#nullable disable

namespace NorthWindCoreLibrary.Models

{

    public partial class Customers

    {

        public Customers()

        {

            Orders = new HashSet<Orders>();

        }

        /// <summary>

        /// Id

        /// </summary>

        public int CustomerIdentifier { get; set; }

        /// <summary>

        /// Company

        /// </summary>

        public string CompanyName { get; set; }

        /// <summary>

        /// ContactId

        /// </summary>

        public int? ContactId { get; set; }

        /// <summary>

        /// Street

        /// </summary>

        public string Street { get; set; }

        /// <summary>

        /// City

        /// </summary>

        public string City { get; set; }

        /// <summary>

        /// Region

        /// </summary>

        public string Region { get; set; }

        /// <summary>

        /// Postal Code

        /// </summary>

        public string PostalCode { get; set; }

        /// <summary>

        /// CountryIdentifier

        /// </summary>

        public int? CountryIdentifier { get; set; }

        /// <summary>

        /// Phone

        /// </summary>

        public string Phone { get; set; }

        /// <summary>

        /// Fax

        /// </summary>

        public string Fax { get; set; }

        /// <summary>

        /// ContactTypeIdentifier

        /// </summary>

        public int? ContactTypeIdentifier { get; set; }

        /// <summary>

        /// Modified Date

        /// </summary>

        public DateTime? ModifiedDate { get; set; }

        public virtual Contacts Contact { get; set; }

        public virtual ContactType ContactTypeIdentifierNavigation { get; set; }

        public virtual Countries CountryIdentifierNavigation { get; set; }

        public virtual ICollection<Orders> Orders { get; set; }

        public static Expression<Func<Customers, CustomerEntity>> Projection

        {

            get

            {

                return (customer) => new CustomerEntity()

                {

                    CustomerIdentifier = customer.CustomerIdentifier,

                    CompanyName = customer.CompanyName,

                    Street = customer.Street,

                    City = customer.City,

                    PostalCode = customer.PostalCode,

                    ContactTypeIdentifier = customer.ContactTypeIdentifier.Value,

                    ContactTitle = customer.ContactTypeIdentifierNavigation.ContactTitle,

                    ContactId = customer.ContactId,

                    CountryName = customer.CountryIdentifierNavigation.Name,

                    FirstName = customer.Contact.FirstName,

                    LastName = customer.Contact.LastName,

                    ContactIdentifier = Convert.ToInt32(customer.ContactId),

                    Contacts = customer.Contact,

                    CountryIdentifier = customer.CountryIdentifier,

                    CountryNavigation = customer.CountryIdentifierNavigation,

                    LastUpdated = customer.ModifiedDate

                };

            }

        }

    }

}

* Open CustomerOperations under the Classes folder. Replace it’s content with the following.

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using Microsoft.EntityFrameworkCore;

using NorthWindCoreLibrary.Classes.North.Classes;

using NorthWindCoreLibrary.Data;

using NorthWindCoreLibrary.Models;

using NorthWindCoreLibrary.Projections;

namespace NorthWindCoreLibrary.Classes

{

    public class CustomersOperations

    {

        public static async Task<List<CustomerItem>> GetCustomersWithProjectionAsync()

        {

            return await Task.Run(async () =>

            {

                await using var context = new NorthwindContext();

                return await context.Customers

                    .Select(CustomerItem.Projection)

                    .ToListAsync();

            });

        }

        public static CustomerEntity CustomerByIdentifier(int identifier)

        {

            using var context = new NorthwindContext();

            return context.Customers.Select(Customers.Projection)

                .FirstOrDefault(custEntity => custEntity.CustomerIdentifier == identifier);

        }

    }

}

Back in the unit test project, class CustomersTest and a two new test method. The first test to verify the code above works as expected while the second test verifies the first test works.

#region Positive and negative test

[TestMethod]

[TestTraits(Trait.EfCoreCustomersSelect)]

public void SingleCustomerByIdentifierGood()

{

    int customerIdentifier = 1;

    CustomerEntity customer = CustomersOperations.CustomerByIdentifier(customerIdentifier);

    Assert.IsNotNull(customer);

    Assert.IsTrue(customer.CompanyName == "Alfreds Futterkiste");

}

[TestMethod]

[TestTraits(Trait.EfCoreCustomersSelect)]

public void SingleCustomerByIdentifierBad()

{

    int customerIdentifier = 134;

    CustomerEntity customer = CustomersOperations.CustomerByIdentifier(customerIdentifier);

    Assert.IsNull(customer);

}

#endregion