ADO.NET Entity framework Interview questions with answers

By

http://www.questpond.com

(Get our 600 videos on Design pattern , EF , ADO.NET , WCF , SQL, SSIS , SSAS , SSRS , Fresher videos , Sharepoint , Biztalk and lot more…)

Contents

What is Entity framework?

ADO.NET entity is an ORM (object relational mapping) which creates a higher abstract object model over ADO.NET components.

So rather than getting in to dataset , datatables , command and connection objects as shown in the below code , you work on higher level domain objects like customers , suppliers etc.

```
DataTable table = adoDs.Tables[0];
for (int j = 0; j < table.Rows.Count; j++)
{
            DataRow row = table.Rows[j];

            // Get the values of the fields
            string CustomerName =
                (string)row["Customername"];
            string CustomerCode =
                (string)row["CustomerCode"];

}
```

Below is the code of entity framework in which we are working on the higher level domain objects like customer rather than working with base level ADO.NET components(like dataset , datareader , command , connection objects etc).

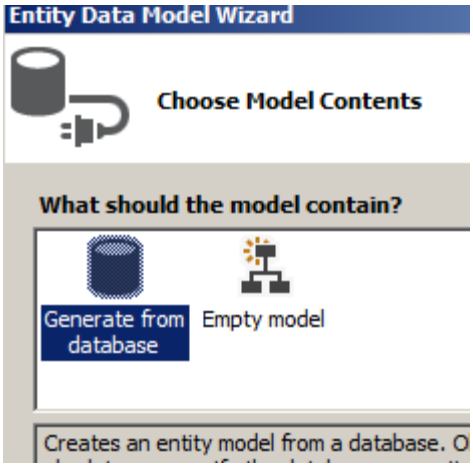
```
foreach (Customer objCust in obj.Customers)
{}
```

So what are benefits of using EF?

The main and the only benefit of EF is it auto-generates code for Model (Middle layer), Data access layer and mapping code thus reducing lot of development time.

So what are the different ways of creating these domain / entity objects?

Entity objects can be created by two ways from a database structure or by starting from scratch by creating a model.

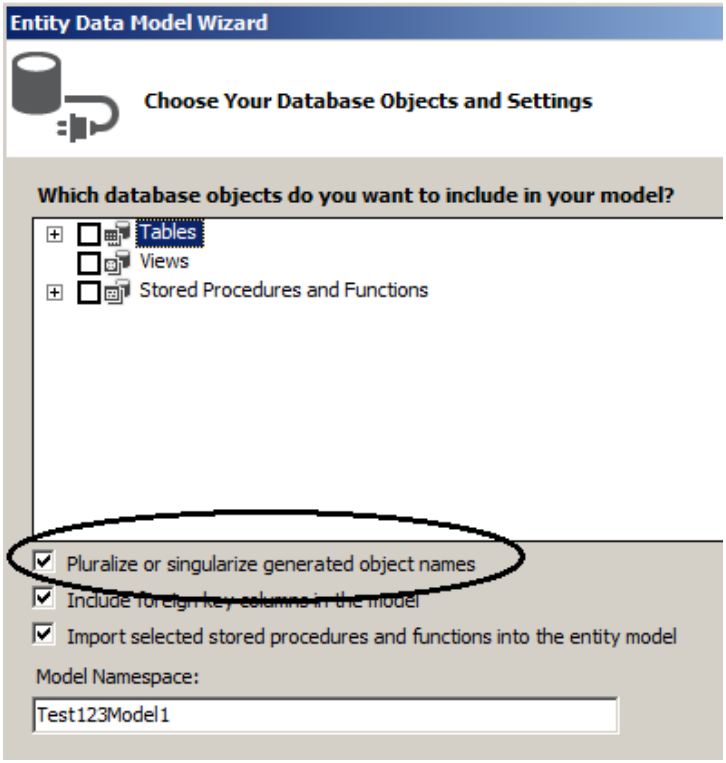


What is pluralize and singularize in Entity framework dialog box?

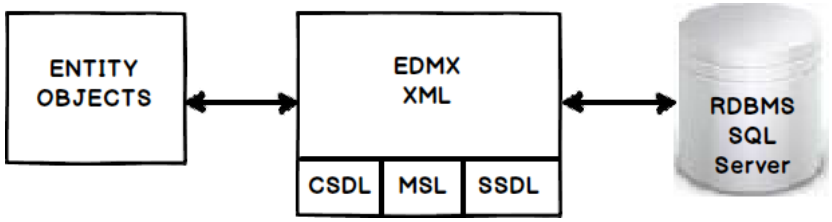
"Pluralize" and "Singularize" give meaningful naming conventions to the objects. In simple word it says do you want to represent your objects with the below naming convention:-

- One Customer record means "Customer" (singular).
- Lot of customer records means "Customer's" (plural , watch the "S")

If you select the below check box entity framework generates naming convention which adheres to plural and singular coding convention.



What is the importance of EDMX file in entity framework?



EDMX (Entity Data Model XML) is a XML file which contains all mapping details of how your objects map with SQL tables. The EDMX file is further divided in to three section CSDL , SSDL and MSL.

Can you explain CSDL, SSDL and MSL section in EDMX file?

CSDL (Conceptual Schema definition language) is the conceptual abstraction which is exposed to the application.

SSDL (Storage schema definition language) defines the mapping with your RDBMS data structure.

MSL (Mapping Schema language) connects the CSDL and SSDL.

CSDL, SSDL and MSL are actually XML files.

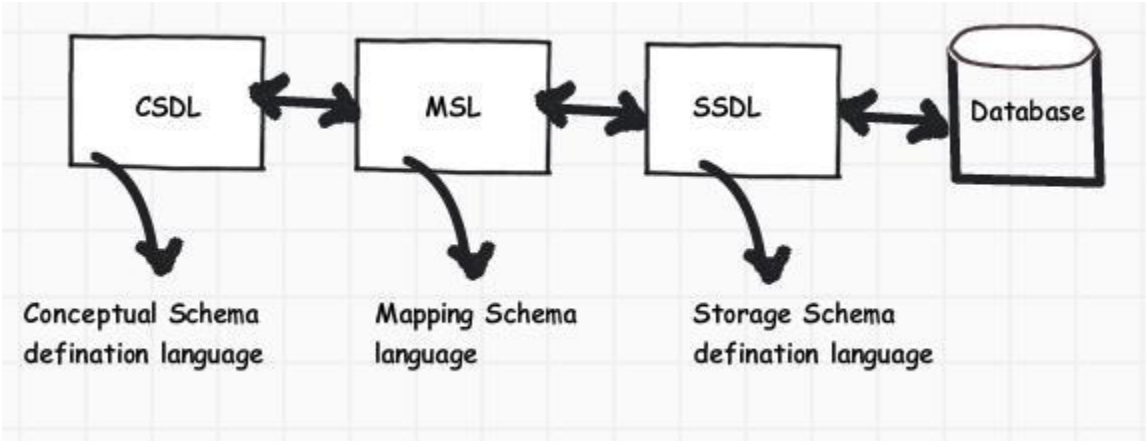


Figure 9.7: -CSDL, MSL and SSDL

What are T4 templates?

T4 (Text template transformation toolkit) is template based code generation engine. So you can go and write C# code in T4 templates (.tt is the extension) files and those c# codes execute to generate the file as per the written C# logic.

For instance the below T4 c# code:-

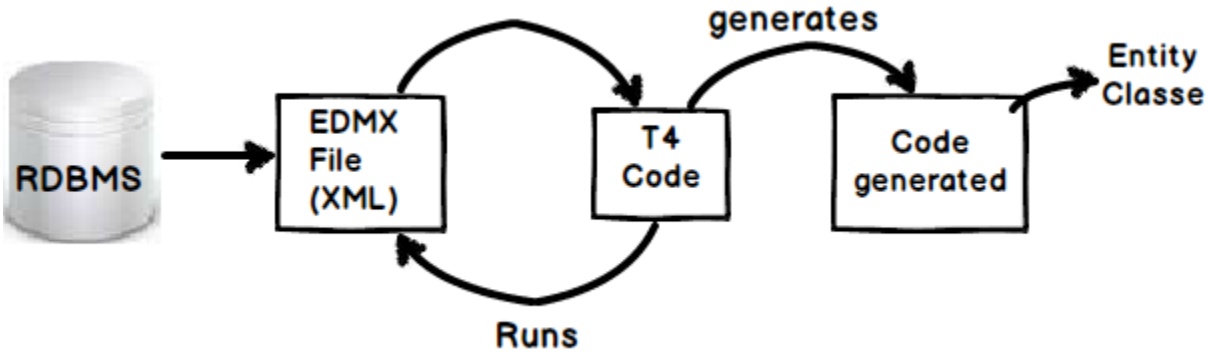
```
<#@ template language="C#" #>
Hello <# Write("World!") #>
```

Will generate the following C# output:-

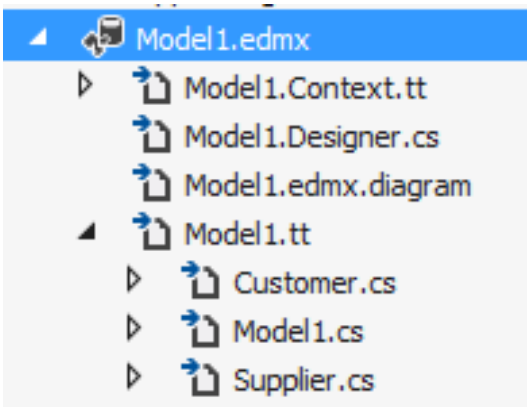
```
Hello
World !
```

So what is the importance of T4 in Entity framework?

T4 files are heart in EF code generation. So the T4 code templates read the EDMX XML file and generate C# behind code. This C# behind code is nothing but your entity and context classes.



If you create a project using VS 2012, you will see the following hierarchy. At the top we have the EDMX file, followed by the TT or T4 file and then the .CS code file.



How can we read records using entity framework classes?

In order to browse through records you can create the object of the context class and inside the context class you will get the records.

For instance in the below code snippet we are looping through list of customer object collection. This customer collection is a output given by the context class "CustomermytextEntities".

```
CustomermytestEntities obj = new CustomermytestEntities();
foreach (Customer objCust in obj.Customers)
{}
```

How can we add, update and delete using EF?

Create the object of your entity class, add it to the data context using add object method and then call save changes method.

```
CustomermytestEntities obj = new CustomermytestEntities();
Customer objCust = new Customer();
objCust.CustomerCode = "1001";
obj.Customers.AddObject(objCust);
obj.SaveChanges();
```

If you want to update , select the object , make change to the object and call accept changes.

```
CustomermytestEntities objContext = new CustomermytestEntities();
Customer objCustomer = (Customer)objContext.Customers.FirstOrDefault();
objCustomer.CountryCode = "NEP";
objContext.AcceptAllChanges();
```

If you want to delete call the delete object method as shown in the below code snippet.

```
CustomermytestEntities objContext = new CustomermytestEntities();
Customer objCustomer = (Customer)objContext.Customers.FirstOrDefault();
objContext.DeleteObject(objCustomer);
```

You can see this youtube.com video which shows a simple insert,update and delete example using entity framework http://www.youtube.com/watch?v=b6vTIiBNcJ0

People say entity framework runs slow?

By default EF has lazy loading behavior. Due to this default behavior if you are loading large number of records and especially if they have foreign key relationship, you can have performance issues.

So you need to be cautious if you really need lazy loading behavior for all scenarios.

So for better performance disable lazy loading when you are loading large number of records or use stored procedures.

Can you explain lazy loading in a detailed manner ?

Lazy Loading is a concept where we load objects on demand rather than loading everything in one go.

Consider a situation where you have 1 to many relationships between "customer" and "address" object.

Now let's say you are browsing the customer data but you do not want address data to be loaded that moment. But the time you start accessing the address object you would like to load address data from database.

Entity framework has the lazy loading behavior by default enabled. For instance consider the below code. When we are doing "foreach" on the customer object address object is not loaded. But the time you start doing "foreach" on the address collection address object is loaded from SQL Server by firing SQL queries.

So in simple words he will fire separate query for each address record of the customer which is definitely not good for large number of records.

```
MyEntities context = new MyEntities();

var Customers = context.Customers.ToList();

foreach (Customer cust in Customers) // In this line no address object loaded
{
     foreach(Address add in cust.Addresses){}// Address object is loaded here
}
```

How can we turn off lazy loading?

The opposite of lazy loading eager loading. In eager loading we load the objects beforehand. So the first thing is we need to disable lazy loading by setting "LazyLoadingEnabled" to false.

```
context.ContextOptions.LazyLoadingEnabled = false;
```

So now we have to explicitly tell EF what objects we want to load by using the "include" function. Below is a simple sample code where we tell EF to load customer as well as address objects by using the "include" function.

So now the customer object and the related address objects will be loaded in one query rather than multiple queries.

```
var employees = context.Customers.Include("Addresses").Take(5);
```

How can we use stored procedures in entity frame work?

You can use the stored procedure mapping details in EDMX as shown in the below figure.

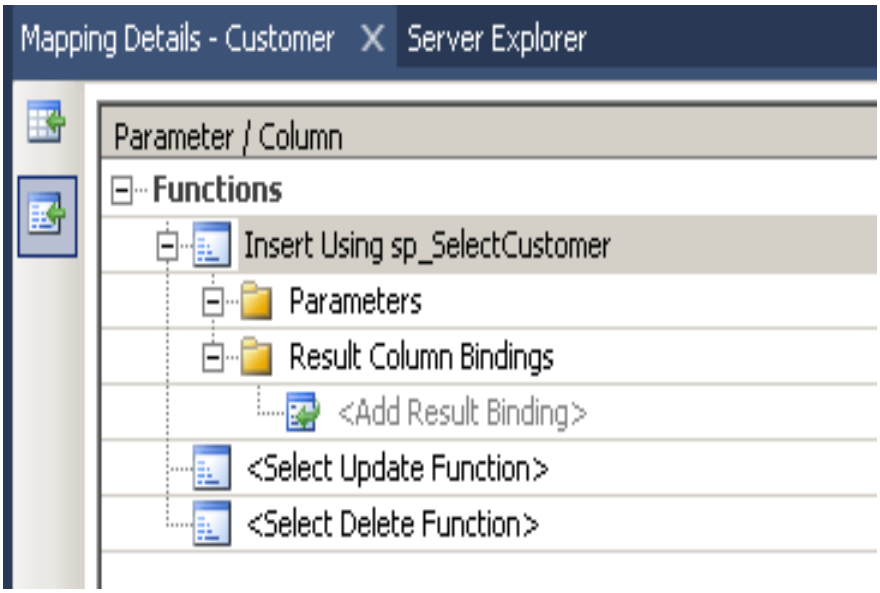


Figure 9.8: - Specify stored procedures

What are POCO classes in Entity framework?

POCO means Plain old C# object. When EDMX creates classes they are cluttered with lot of entity tags. For instance below is a simple customer class generated using entity framework. Many times we would like to use simple .NET classes and integrate the same with entity framework.

Entity framework allows the same. In other words you can create a simple .NET class and use the entity context object to load your simple .NET classes.

Below is a simple class generated by EF which is cluttered with lot of EF attributes.

```
[EdmEntityTypeAttribute(NamespaceName="CustomermytestModel",
Name="Customer")]
    [Serializable()]
    [DataContractAttribute(IsReference=true)]
    public partial class Customer : EntityObject
    {
        #region Factory Method

        /// <summary>
        /// Create a new Customer object.
        /// </summary>
```

```
        /// <param name="id">Initial value of the Id property.</param>
        /// <param name="customerCode">Initial value of the CustomerCode
property.</param>
        /// <param name="customername">Initial value of the Customername
property.</param>
        public static Customer CreateCustomer(global::System.Int32 id,
global::System.String customerCode, global::System.String customername)
        {
            Customer customer = new Customer();
            customer.Id = id;
            customer.CustomerCode = customerCode;
            customer.Customername = customername;
            return customer;
        }

        #endregion
        #region Primitive Properties
```

How to implement POCO in entity framework?

To implement POCO is a 3 step process:-

- Go to the designer and make the code generation strategy to NONE. This step means that you would be generating the classes with your own hands rather than relying on EF auto code generation.
- So now that we have stopped the auto generation of code , we need to create the domain classes manually. So add a class file and create the domain classes like the one “Customer” class we have created.

```
public class Customer
    {
        private string _customerName;

        public string CustomerName
        {
            get { return _customerName; }
            set { _customerName = value; }
        }
```

```
        private int _Customerid;

        public int Customerid
        {
            get { return _Customerid; }
            set { _Customerid = value; }
        }

    }
```

- Write your Context layer code inheriting from the "ObjectContext". This code you can copy paste from the behind code of EF also before disabling autogeneration.

```
public partial class Test123Entities : ObjectContext
    {
        public Test123Entities()
            : base("name=Test123Entities", "Test123Entities")
        {
            this.ContextOptions.LazyLoadingEnabled = true;
            OnContextCreated();
        }
        partial void OnContextCreated();
        public ObjectSet<Customer> Customers
        {
            get
            {
                if ((_Customers == null))
                {
                    _Customers = base.CreateObjectSet<Customer>("Customers");
                }
                return _Customers;
            }
        }
        private ObjectSet<Customer> _Customers;
        public void AddToCustomers(Customer customer)
```

```
        {
            base.AddObject("Customers", customer);
        }

    }
```

And finally you can use the above code in your client as you where using your EF normally.

```
Test123Entities oContext = new Test123Entities();
List<Customer> oCustomers = oContext.Customers.ToList<Customer>();
```

In POCO classes do we will need EDMX files?

Yes, you will still need EDMX files because the context object reads the EDMX files to do the mapping.

What is code first approach in entity framework?

In code first approach we avoid working with visual designer of entity framework. In other words the EDMX file is excluded from the solution. So you now have complete control over the context class as well as the entity classes.

What is the difference between POCO, code first and simple EF approach?

All these three approaches define how much control you want on your Entity frame work code. Entity framework is an OR MAPPER it generates lot of code, it creates your middle tier (Entity) and Data access layer (Context).

But lot of times you want to enjoy benefits of both the world you want the auto-generation part to minimize your development time and also you want control on the code so that you can maintain code quality.

Below is the difference table which defines each of the approaches. In Simple entity framework everything is auto generated and so you need the EDMX XML file as well. POCO is semi-automatic so you have full control on the entity classes but then the context classes are still generated by the EDMX file.

In code first you have complete control on how you can create the entity and the context classes. Because you are going to manually create these classes you do not have dependency on the EDMX XML file. Below is a simple table which shows the cross comparison.

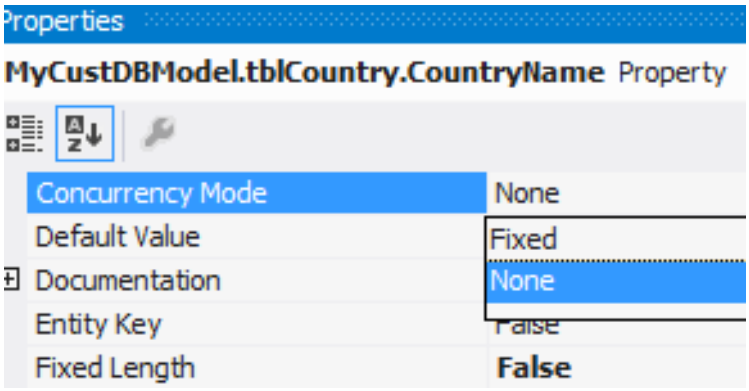
	EDMX	***Entity***	***Context***
Simple entity framework	Needed	Auto	Auto
POCO approach	Needed	Manual	Auto

Code First	Not Needed	Manual	Manual

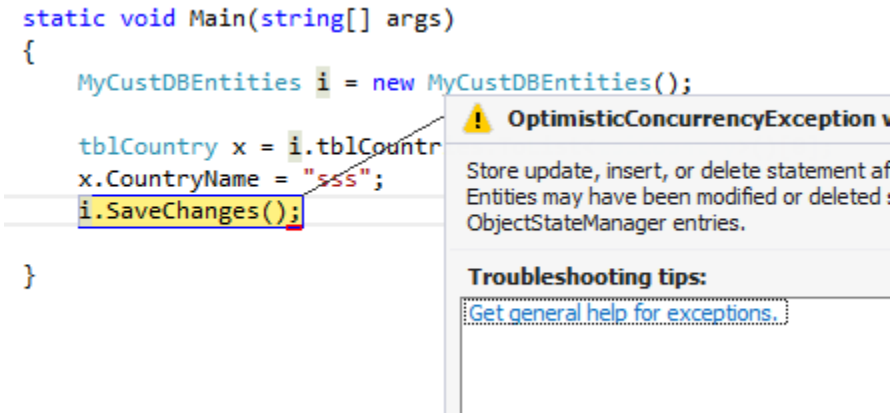
How can we handle concurrency in Entity framework?

Note :- Before this question interviewer can ask you about What is concurrency and What is pessimistic and optimistic locking. Please do refer ADO.NET chapter for the same.

In EF concurrency issue is resolved by using optimistic locking. Please refer ADO.NET chapter for what is optimistic locking and pessimistic locking?. To implement optimistic locking right click on the EDMX designer and set the concurrency mode to "Fixed" as shown in the below figure.



Now whenever we have concurrency issues you should get "OptimisticConcurrencyException" error as shown in the below figure. You can then put a try / catch to handle this situation.



How can we do pessimistic locking in Entity framework?

We cannot do pessimistic locking using entity framework. You can invoke a stored procedure from entity framework and do pessimistic locking by setting isolation level in the stored procedure. But directly entity framework does not support pessimistic locking.

What is client wins and store wins mode in entity framework concurrency?

Client wins and store wins are actions which you would like to take when concurrency happens. In store wins / database wins the data from the server is loaded in to your entity objects. Client wins is opposite to stored wins, data from the entity object is saved to the database.

We need to use the "Refresh" method of the entity framework context and provide the "RefreshMode" enum values. Below is a simple code snippet which executes "ClientWins".

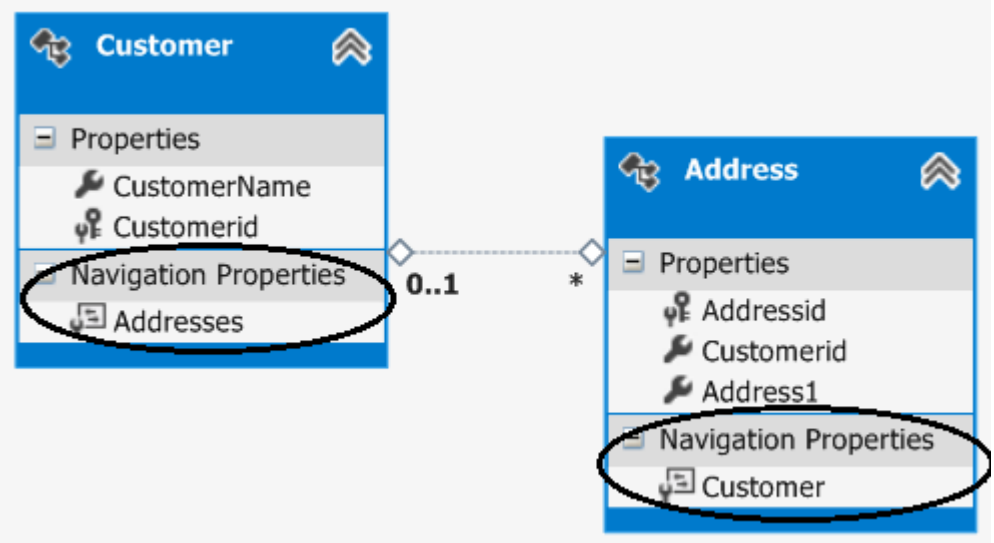
```
Context.Refresh(System.Data.Objects.RefreshMode.ClientWins,Obj);
```

What are scalar and navigation properties in Entity framework?

Navigation properties help to navigate from one entity to the other entity. For instance consider the below example in which we have two entities customer and address and one customer has multiple address objects.

Now we would like to have a facility where at any given moment we would like to browse from a given customer object to the addresses collection and from address object to the customer.

If you open the entity designer you would notice "Navigation" properties as shown below. The navigation properties are automatically created from the primary and foreign key references.



So now because of those navigation properties we can browse from Customer to addresses object , look at the below code.

```
Customer Cust =  oContext.Customers.ToList<Customer>()[0];


// From customer are browsing addresses
List<Address> Addresses = Cust.Addresses.ToList<Address>();
```

You can also go vice versa. In other words from the address object you can reference the customer object as shown in the below code.

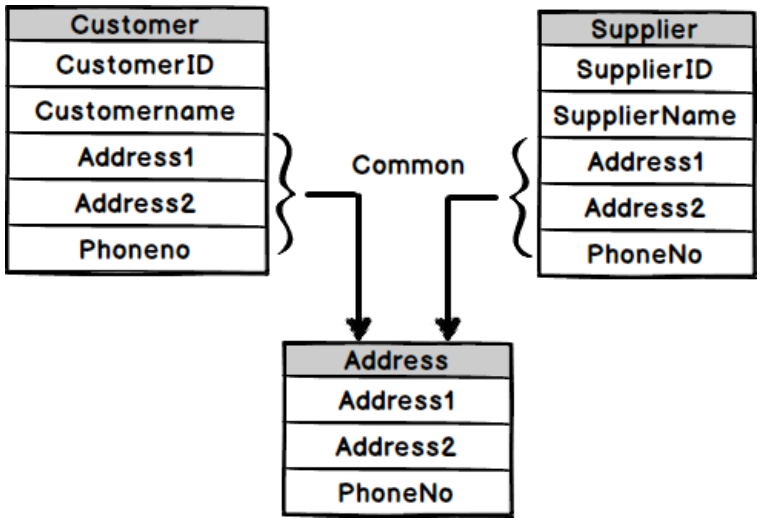
```
Address myAddress = Addresses[0];


// From address we can browse customer
Customer cus = myAddress.Customer;
```

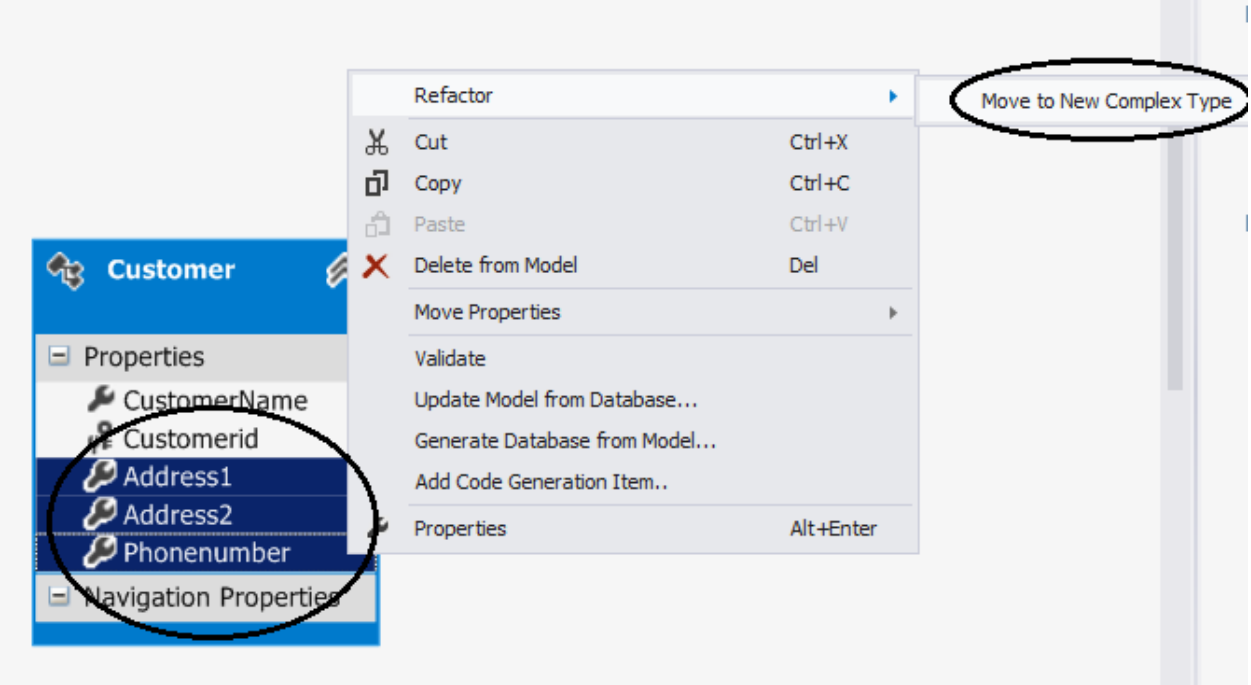
What are complex types in Entity framework?

There can be situations where you have common properties across entities. For example consider the below figure where we have "Customer" and "Supplier" entities. They have three fields in common "Address1" , "Address2" and "PhoneNo". These fields have been duplicated both in "Customer" as well as "Supplier" entities.

So to remove these duplicate and redundant fields we can move them to a common complex type called "Address". So complex types group common fields so that they can be reused across entities.



So to create a complex type, select the fields which you want to group in a complex type, click on refactor and create the complex type. Below is the figure which shows the same. Once the complex type is created you can then reuse the complex type with other entities as well.



What's the difference between LINQ to SQL and Entity framework?

- LINQ to SQL is good for rapid development with SQL Server. EF is for enterprise scenarios and works with SQL server as well as other databases.
- LINQ maps directly to tables. One LINQ entity class maps to one table. EF has a conceptual model and that conceptual model map to storage model via mappings. So one EF class can map to multiple tables or one table can map to multiple classes.
- LINQ is more targeted towards rapid development while EF is for enterprise level where the need is to develop loosely coupled framework.

Please do visit my site www.questpond.com which has lot of videos on C#,.NET,EF,MVC,Design pattern etc.