

ADO.NET Entity Framework Interview Questions



Shivprasad koirala, 11 Dec 2013 [CPOL](#)




4.92 (97 votes)

Rate this: [vote 1](#)[vote 2](#)[vote 3](#)[vote 4](#)[vote 5](#)

Quick revision of 25 important ADO.NET Entity Framework interview questions with answers.

• [Download in PDF format - 738.5 KB](#)

A yellow rectangular sticky note with a red tab at the top, placed on a white grid background. The text on the note is in a large, bold, black sans-serif font.

**Entity framework
Interview Questions
and
Answers**

Contents

- [What is Entity Framework?](#)
- [What are the benefits of using EF?](#)
- [What are the different ways of creating these domain / entity objects?](#)
- [What is pluralize and singularize in the Entity Framework dialog box?](#)
- [What is the importance of EDMX file in Entity Framework?](#)
- [Can you explain CSDL, SSDL, and MSL sections in an EDMX file?](#)
- [What are T4 templates?](#)
- [What is the importance of T4 in Entity Framework?](#)
- [How can we read records using Entity Framework classes?](#)
- [How can we add, update, and delete using EF?](#)
- [People say Entity Framework runs slow](#)
- [Can you explain lazy loading in a detailed manner?](#)
- [How can we turn off lazy loading?](#)
- [How can we use Stored Procedures in Entity Framework?](#)
- [What are POCO classes in Entity Framework?](#)
- [How do we implement POCO in Entity Framework?](#)
- [In POCO classes do we need EDMX files?](#)
- [What is Code First approach in Entity Framework?](#)
- [What is the difference between POCO, Code First, and the simple EF approach?](#)
- [How can we handle concurrency in Entity Framework?](#)
- [How can we do pessimistic locking in Entity Framework?](#)
- [What are client wins and store wins mode in Entity Framework concurrency?](#)
- [What are scalar and navigation properties in Entity Framework?](#)
- [What are complex types in Entity Framework?](#)
- [What's the difference between LINQ to SQL and Entity Framework?](#)
- [What is the difference between DbContext andObjectContext?](#)

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

Hide Copy Code

```
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 for Entity Framework in which we are working on higher level domain objects like customer rather than with base level ADO.NET components (like dataset, datareader, command, connection objects, etc.).

[Hide](#) [Copy Code](#)

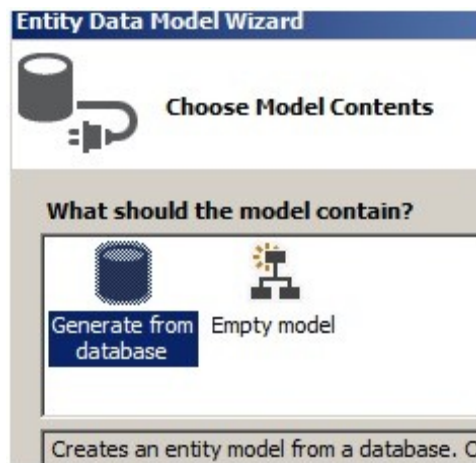
```
foreach (Customer objCust in obj.Customers)
{ }
```

What are the benefits of using EF?

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

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

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

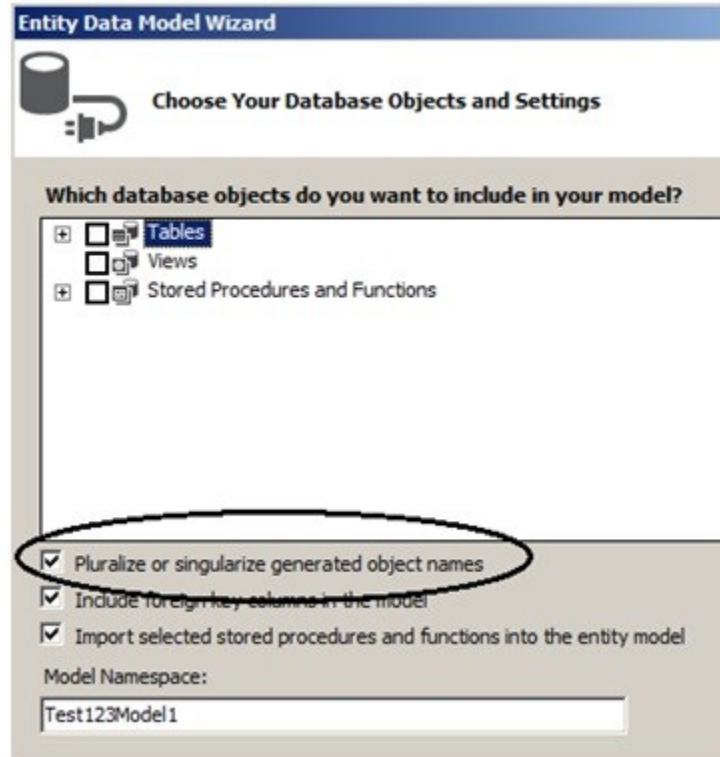


What is pluralize and singularize in the Entity Framework dialog box?

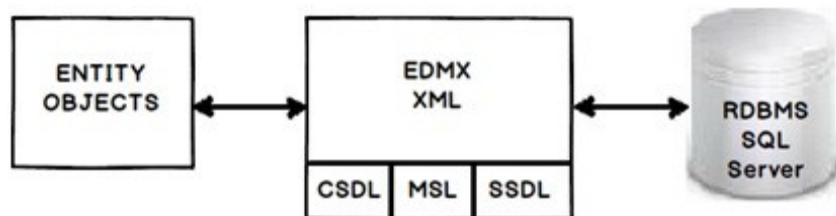
“Pluralize” and “Singularize” give meaningful naming conventions to objects. In simple words it says do you want to represent your objects with the below naming convention:

- One Customer record means “Customer” (singular).
- Lot of customer records means “Customer’s” (plural, watch the “s”)

If you select the below checkbox, Entity Framework generates a naming convention which adheres to plural and singular coding conventions.



What is the importance of EDMX file in Entity Framework?



EDMX (Entity Data Model XML) is an XML file which contains all the mapping details of how your objects map with SQL tables. The EDMX file is further divided into three sections: CSDL, SSDL, and MSL.

Can you explain CSDL, SSDL and MSL sections in an 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.

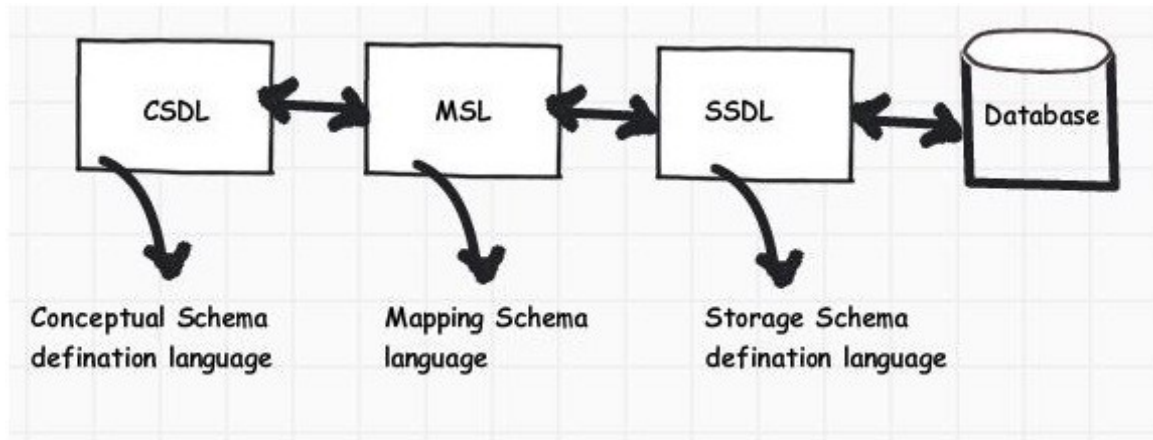


Figure: CSDL, MSL, and SSDL

What are T4 templates?

T4 (Text Template Transformation Toolkit) is a template based code generation engine. 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:

[Hide](#) [Copy Code](#)

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

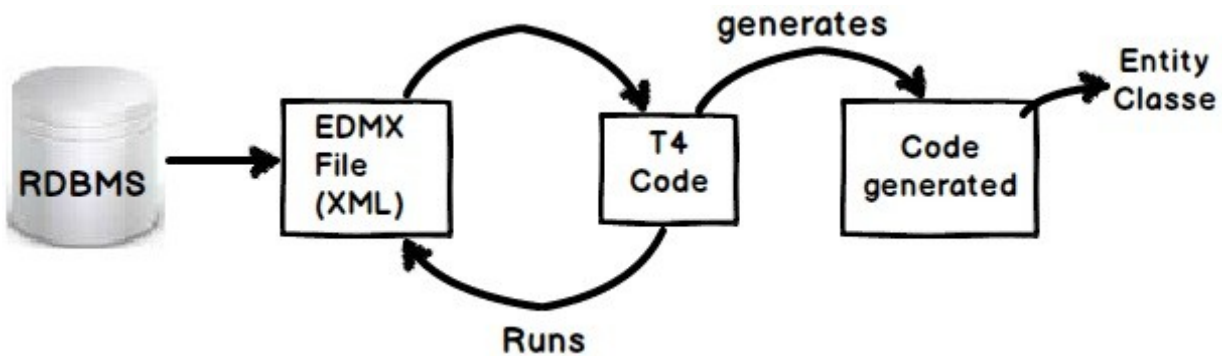
Will generate the following C# output:

[Hide](#) [Copy Code](#)

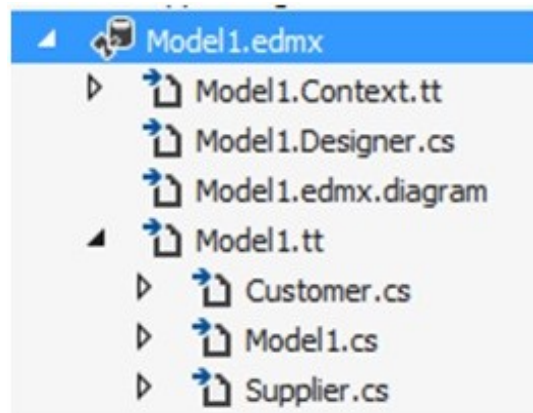
```
Hello
World !
```

What is the importance of T4 in Entity Framework?

T4 files are the heart of EF code generation. 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 a customer object collection. This customer collection is the output given by the context class **CustomermytestEntities**.

Hide Copy Code

```
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 **AddObject** method, and then call the **SaveChanges** method.

Hide Copy Code

```
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 changes to the object, and call **AcceptAllChanges**.

Hide Copy Code

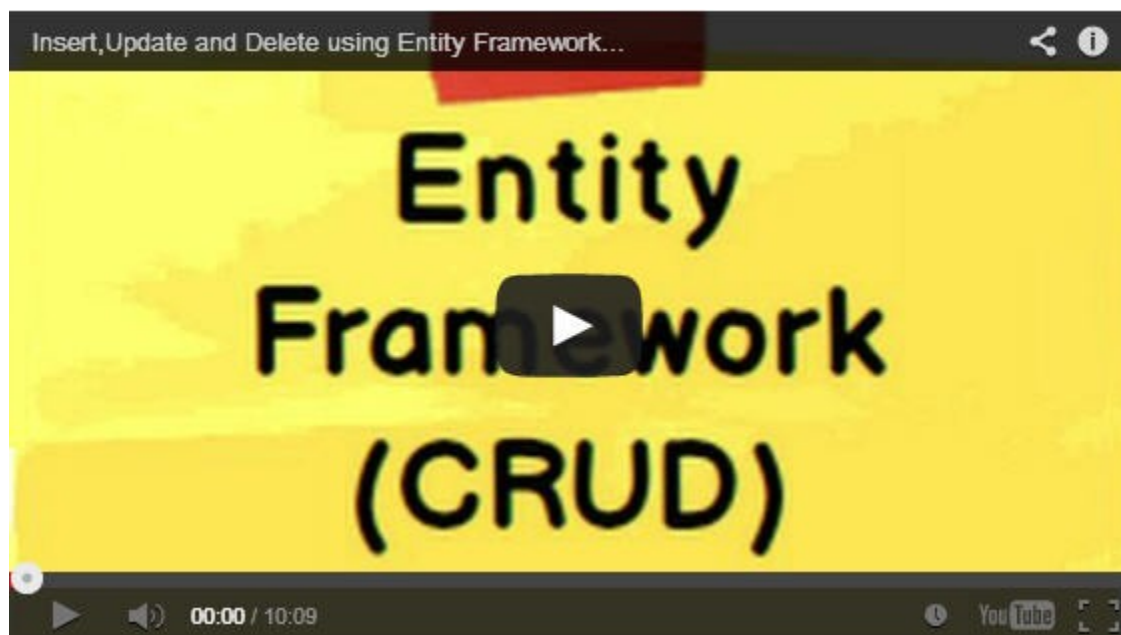
```
CustomermytestEntities objContext = new CustomermytestEntities();
Customer objCustomer = (Customer)objContext.Customers.FirstOrDefault();
objCustomer.CountryCode = "NEP";
objContext.AcceptAllChanges();
```

If you want to delete, call the **DeleteObject** method as shown in the below code snippet:

Hide Copy Code

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

You can see the following YouTube video which shows a simple insert, update, and delete example using Entity Framework:



People say Entity Framework runs slow

By default EF has lazy loading behavior. Due to this default behavior if you are loading a large number of records and especially if they have foreign key relationships, you can have performance issues. So you need to be cautious if you really need lazy loading behavior for all scenarios. For better performance, disable lazy loading when you are loading a 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 the Customer and Address objects. Now let's say you are browsing the customer data but you do not want address data to be loaded at that moment. But the time you start accessing the address object you would like to load address data from the database.

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

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

[Hide](#) [Copy Code](#)

```
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 is 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**.

[Hide](#) [Copy Code](#)

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


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.

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

[Hide](#) [Copy Code](#)

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

How can we use stored procedures in Entity Framework?

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

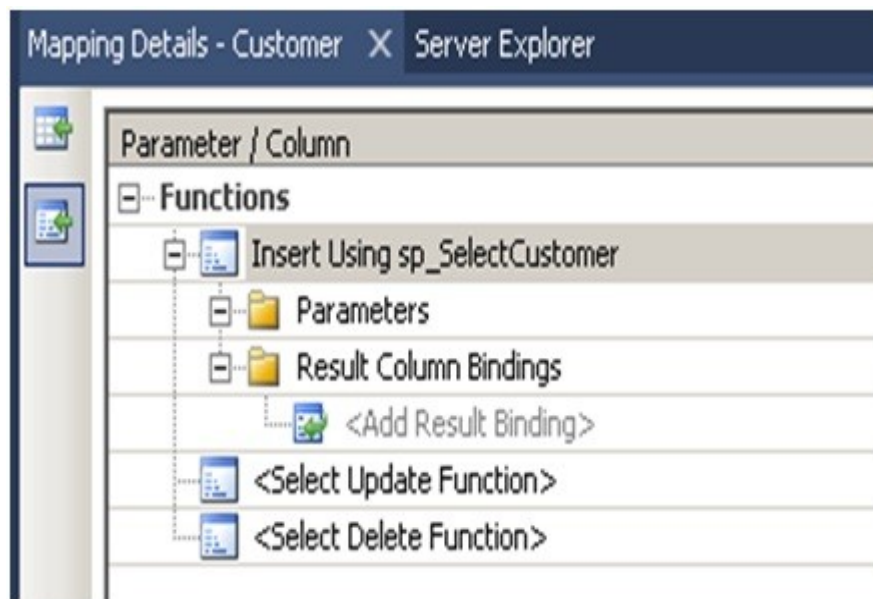


Figure: Specify stored procedures

What are POCO classes in Entity Framework?

POCO means Plain Old C# Object. When EDMX creates classes, they are cluttered with a 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 them with Entity Framework.

Entity Framework allows this. 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 a 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 name="customerCode" />Initial value of the CustomerCode property.
    /// <param name="customername" />Initial value of the Customername property.
    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 do we implement POCO in Entity Framework?

To implement POCO is a three step process:

- Go to the designer and set the code generation strategy to **NONE**. This step means that you would be generating the classes on your own rather than relying on EF auto code generation.
- Now that we have stopped the auto generation of code, we need to create the domain classes manually. Add a class file and create the domain classes like the **Customer** class we 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 **ObjectContext**. This code you can copy paste from the behind code of EF, also before disabling auto-generation.

[Hide](#) [Copy Code](#)

```
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 if you were using EF normally.

[Hide](#) [Copy Code](#)

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

In POCO classes do we 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 the 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 Framework code. Entity Framework is an OR mapper, it generates a lot of code, it creates your middle tier (Entity), and Data Access layer (Context).

But a lot of times you want to enjoy the benefits of both worlds, you want the auto-generation part to minimize your development time and 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.

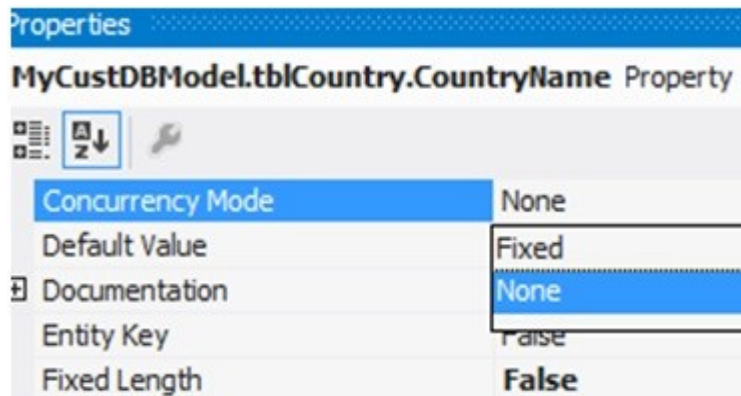
In Code First, you have complete control on how you can create the entity and context classes. Because you are going to manually create these classes, you do not have dependency on the EDMX XML file. Below is a simple table which shows the cross comparison.

| | 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, the interviewer can ask you about concurrency and what is pessimistic and optimistic locking. Please do refer to the ADO.NET chapter for those.

In EF, concurrency issue is resolved by using optimistic locking. Please refer to the 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 an **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 the 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?

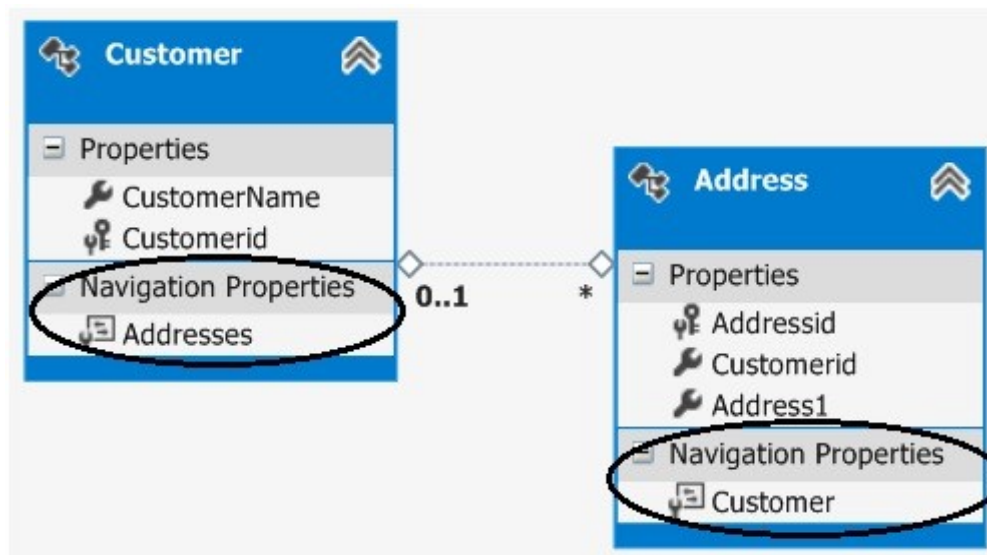
Client wins and store wins are actions which you would like to take when concurrency happens. In store wins / database wins, the data from the server is loaded into your entity objects. Client wins is opposite to stored wins, data from the entity object is saved to the database.

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**.

[Hide](#) [Copy Code](#)

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

What are scalar and navigation properties in Entity Framework?



Scalar properties are those where actual values are contained in the entities. For example, in the above customer entity, **customername** and **customerid** are scalar properties. Normally a scalar property will map to a database field.

Navigation properties help to navigate from one entity to another entity. For instance, consider the below example in which we have two entities: Customer and Address, and a 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 the 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 the **Customer** to the **Addresses** object, look at the below code:

[Hide](#) [Copy Code](#)

```
Customer Cust = oContext.Customers.ToList<Customer>()[0];  
// From customer are browsing addresses  
List<Address> Addresses = Cust.Addresses.ToList<Address>();
```

You can also do vice versa. In other words, from the **Address** object, you can reference the **Customer** object, as shown in the below code.

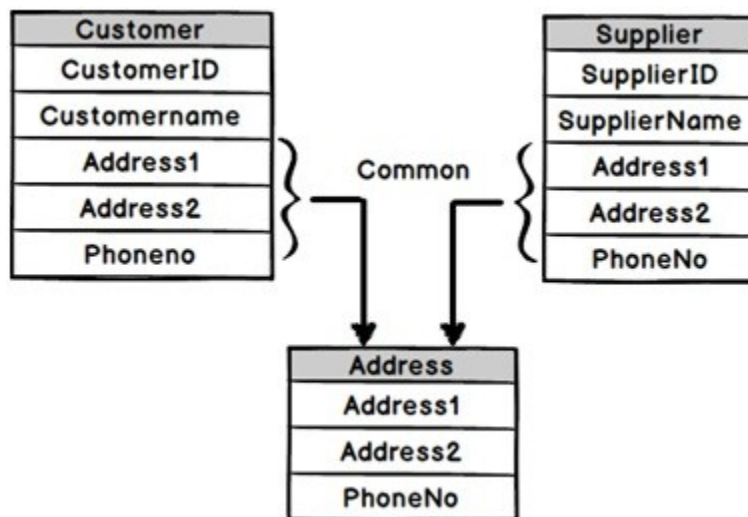
[Hide](#) [Copy Code](#)

```
Address myAddress = Addresses[0];  
  
// From address we can browse customer  
Customer cus = myAddress.Customer;
```

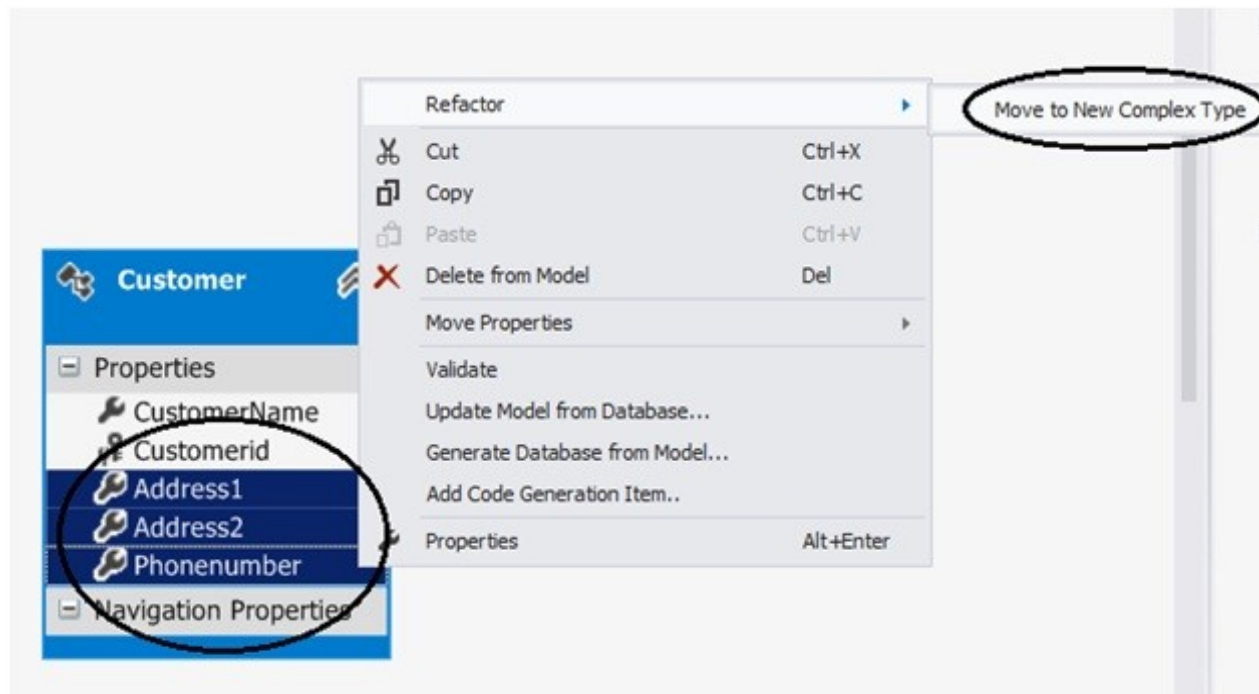
What are complex types in Entity Framework?

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 the **Customer** and **Supplier** entities.

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



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 a figure which shows this. Once the complex type is created, you can then reuse the complex type with other entities as well.

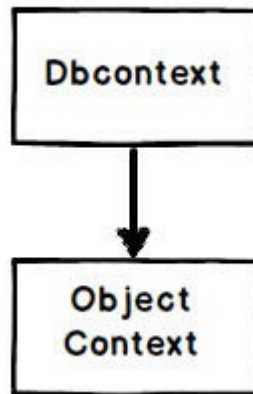


What's the difference between LINQ to SQL and Entity Framework?

- 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 maps to the 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 a loosely coupled framework.

What is the difference between DbContext andObjectContext?

DbContext is a wrapper around **ObjectContext**, it's a simplified version of **ObjectContext**.



As a developer you can start with **DbContext** as it's simple to use. When you feel that some of the operations cannot be achieved by **DbContext**, you can then access **ObjectContext** from **DbContext**, as shown in the below code:

[Hide](#) [Copy Code](#)

```
((ObjectContextAdapter)dbContext).ObjectContext
```

Note: If the interviewer asks what kind of operations are not supported in **DbContext**, you can excuse by saying you do not remember them up front. Wonder why sometimes interviewers ask API level questions?

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

<https://www.codeproject.com/Articles/676309/ADO-NET-Entity-Framework-Interview-Questions>

ADO.NET Entity Framework is an advance framework developed by Microsoft Corporation. This article describes some *common important interview questions with answers in ADO.NET Entity Framework*. Hope it will help you to build successful carrier.

What is ADO.NET Entity Framework or EF?

ADO.NET **Entity Framework** is an ORM Framework developed by Microsoft. It is an enhancement of ADO.NET that gives us an automated mechanism to access and store data in the database. We can access database without much more code or programming. ORM is a tool to store data from domain object to relational database like MS SQL Server without too much coding. It has three parts: Domain Class Object, Relational Database Object and Mapping information on how domain object maps with relational database objects

History of ADO.NET Entity Framework

- The first version of Entity Framework released on 11 August, 2008 with .NET Framework 3.5 Service Pack 1 and Visual Studio 2008 Service Pack 1
- The second version of Entity Framework was released on 12 April 2010 with .NET Framework 4.0

- A third version of Entity Framework was released on April 12, 2011. Its name was Entity Framework 4.1
- A refresh of version of Entity Framework 4.1 was released on July 25, 2011
- The version Entity Framework 4.3.1 was released on February 29, 2012
- The latest version is 5.0.0 and is targeted at .NET framework 4.5. But it is also available for .Net framework 4
- The Entity Framework 6.0 is an open source project. Its source code is hosted at CodePlex using Git and licensed under Apache License v2. Like ASP.NET MVC Framework

What is minimum requirement for Entity Framework applications to run?

The Entity Framework is a component of the .NET Framework. Any application developed by Entity Framework, can run on any computer which has .NET Framework 3.5 SP or greater version.

What are the benefits of using Entity Framework or EF?

The main and the only benefit of EF is auto-generates code for the Model (middle layer), Data Access Layer, and mapping code. It reduces a lot of development time.

What is Entity Data Model (EDM)?

Entity Data Model or EDM refers to a set of concepts that describe data structure, regardless of its stored form. It is a bridge between application and data storage and helps us to work at a conceptual level rather than the actual schema. It uses three key concepts to describe data structure (entity type, association type and property).

What is .edmx file and what it contains?

An .edmx file (Entity Data Model XML) is an XML file. It defines a conceptual model, a storage model, and the mapping between these models. This file also contains the information that is used by the ADO.NET Entity Data Model Designer to render a model graphically. It contains all the mapping details of how objects map with SQL tables. It is divided into three sections: CSDL, SSDL, and MSL.

What are the main parts in EDM?

In entity framework, the EDM contains three main components.

- *Conceptual* *Model*
The conceptual model contains the model classes and their relationships. This will be independent from the database table design.
- *Storage* *Model*
The storage model is the database design model which includes tables, views, stored procedures, relationships and keys.
- *Mapping*
The mapping contains the information about how the conceptual model is mapped to storage model.

What are CSDL, SSDL and MSL sections in an EDMX file?

CSDL, SSDL and MSL are actually XML files.

CSDL (Conceptual Schema Definition Language) -is the conceptual abstraction which is exposed to the application.

SSDL (Storage Schema Definition Language) –defines the mapping with our RDBMS data structure.

MSL (Mapping Schema Language) -connects the CSDL and SSDL.

What are the languages used in entity framework?

In entity framework basically two languages are used.

- LINQ to Entities
- Entity SQL

- Native SQL

What is Entity SQL?

Entity SQL is a SQL-like language that is used to query in the conceptual models in the Entity Framework. The conceptual models represent data as entities and relationships. Entity SQL allows querying those entities and relationships in a format that is familiar to those who have used SQL. But it is more difficult than LINQ to Entities.

What is LINQ to Entities?

LINQ to Entities is a query language used to write queries against the object model. It returns entities, which are defined in the conceptual model. LINQ to Entities provides developers to write LINQ queries.

What is Native SQL?

In Native SQL, we can write native SQL queries for relational database. A sample example is given below:

```
using (var ctx = new TestDBEntities())

{

    var studentName = ctx.Students.SqlQuery("Select StudentID, StudentName
from Students where StudentName='AAA'").FirstOrDefault<Student>();

}
```

What are the types of entities in entity framework?

In Entity Framework 5.0/6.0 there are two types of entities:

- *POCO entity (Plain Old CLR Object)*

It is like our normal .Net classes.

- *Dynamic proxy entity*

What is the difference between DbContext andObjectContext?

DbContext is a lightweight version of the ObjectContext class. DbContext requires a lot less code for the same kind of functionality. ObjectContext EF V4.0 and DbContext EF V4.1

What are the main drawbacks of Entity Framework?

The main drawbacks of EF are lazy loading. It is EF default setting but we can disable it. Due to this behavior if we are loading large number of records (especially if they have foreign key relations) then it may take long time. So for better performance disable lazy loading for large number of records.

How to load related Entities in EF

In Entity Framework we can load related data in three ways

1. Eager loading
2. Lazy loading
3. Explicit loading

What is Lazy Loading?

Lazy loading is the process to delay the loading of related objects until we need it. Other hand, on demand objects loading rather than loading objects unnecessarily. When objects are returned by a query related objects are not loaded at the same time.

[First query will load main object and the related objects will be loaded in second queries.]

What is Eager Loading?

The opposite of Lazy Loading is eager loading. Eager loading is the process of loading related entities/ objects.

What is Explicit Loading?

If lazy loading disabled it is still possible to lazily load related entities. For this we need to call the Load method on the related entities.

How can we turn off lazy loading?

We can turn off lazy loading by setting LazyLoadingEnabled to false.
`context.ContextOptions.LazyLoadingEnabled = false;`

When we will use lazy Loading?

When we know that only main object will be used and related objects will not be used.

What are the types of development approaches in EF?

The Entity Framework supports three different development approaches to use entity framework in our applications.

- Code First
- Model First
- Database First

What is Code First approach in Entity Framework?

In Code First approach we avoid working with the Visual Designer of Entity Framework. We can say, the EDMX file is excluded from the solution. So now, we have complete control over the context class as well as the entity classes. Here the developers writes POCO classes first and then generates the database from these POCO classes. The developers, who follow Domain-Driven Design (DDD) principles, prefer to use code first model.

What is Model First Approach in Entity Framework?

In Model First approach, we can create Entities, relationships directly on the design surface of EDMX. So in this approach, when we want to add ADO.NET Entity Data Model in our application, we should select "Empty Model" instead of "Generate from database" in the entity data model wizard. By right clicking on the designer surface we can create entity, association, inheritance, database etc.

What is Model First Approach in Entity Framework?

In model first approach, we create EDMX from an existing database. We can update EDM any time based on database schema changes. Its support stored procedure, view, etc.

Which development approach is suitable for which conditions?

- *Code First*
If we have existing application with domain classes
Want to create database from your existing domain classes
- *Model First*
If we have existing database
- *Database First*
If we don't have existing database or domain classes, and we prefer to design our db model on the visual designer

Entity Framework vs LINQ-to-SQL

Entity Framework and LINQ to SQL are not same. There is some difference. The difference between Entity Framework and LINQ to SQL:

- EF has a full provider model. It supports SQL Server, Oracle, DB2, MySQL etc
- Most of the time L2S classes must be one-to-one with database objects (Customer class can be mapped only with Customer table). Where in EF we can

map our domain class with multiple tables using various inheritance strategies like table per type (class) or table per hierarchy of classes etc

- EF supports multiple modeling techniques (code first, model first or database first)
- Microsoft Corporation has long term strategy to support and integrate Entity Framework with multiple Microsoft products

What do you mean by Navigation Property?

A navigation property is an optional property on an entity/object that allows for navigation from one end of an association to the other end. Unlike other properties, it does not carry data.

What are POCO classes in Entity Framework?

POCO means Plain Old C# Object.

In POCO classes do we need EDMX files?

Yes we need. Because the context objects reads the EDMX files to do the mapping.

What are T4 templates?

Text Template Transformation Toolkit or T4 is a template based code generation engine. We 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. T4 C# code:

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

What is the importance of T4 in Entity Framework?

T4 files are the heart of Entity Framework code generation. It read the EDMX XML file and generates C# behind code. This C# behind code is our entity and context classes. If we developed our project in VS 2012 then we can found .tt files.

Give an example of EF select query

An example of EF query that returns CustomerID, CustomerName, Address from Customers table with gender male is given below:

```
TestDBEntities db = new TestDBEntities();

var result = from c in db.Customers

              where c.Gender == 'M'

              orderby c.CustomerName ascending

              select new { c.CustomerID, c.CustomerName, c.Address };
```

Give an example of EF join query

An example of EF query that returns CustomerID, CustomerName, OrderDate, Order Amount from Customer and Orders table is given below:

```
TestDBEntities db = new TestDBEntities();
```

```

var result = from c in db.Customers

                join o in db.Orders on c.CustomerID equals o.CustomerID

                where c.CustomerID == 1

                select new { c.CustomerID, c.CustomerName, o.OrderDate,
o.Amount };

```

Write the query to return a generic list in EF

An example of EF query that creates a generic list of Customer table is given below:

```

TestDBEntities db = new TestDBEntities();

var result = from c in db.Customers

                select new { c.CustomerID, c.CustomerName };

IList<object> items = new List<object>();

foreach (var r in result)

{

    items.Add(r);

}

```

Give an example of EF insert query

An example of EF query that insert data into Students table is given below:

```

TestDBEntities db = new TestDBEntities();

Student oStudent = new Student();

oStudent.StudentName    = "Mr. KK Modi";

oStudent.Address        = "Mumbai";

oStudent.Phone          = "9912548";

db.Students.AddObject(oStudent);

```



```
db.SaveChanges();
```

Give an example of EF update query

A sample EF query to update Students tables is given bellow:

```
TestDBEntities db = new TestDBEntities();
```

```
var result = (from S in db.Students  
  
              where S.StudentID == 2  
  
              select S).Single();
```

```
result.StudentName = "Mr. John Abrar";
```

```
result.Address      = "6/2 Park Road";
```

```
result.Phone        = "9658745";
```

```
db.Students.ApplyCurrentValues(result);
```

```
db.SaveChanges();
```

Give an example of EF delete query

A sample EF query to delete data from Students table is given bellow:

```
TestDBEntities db = new TestDBEntities();
```

```
var result = (from S in db.Students  
  
              where S.StudentID == 3  
  
              select S).Single();
```

```
db.Students.DeleteObject(result);
```

```
db.SaveChanges();
```

How to call stored procedure in Entity Framework?

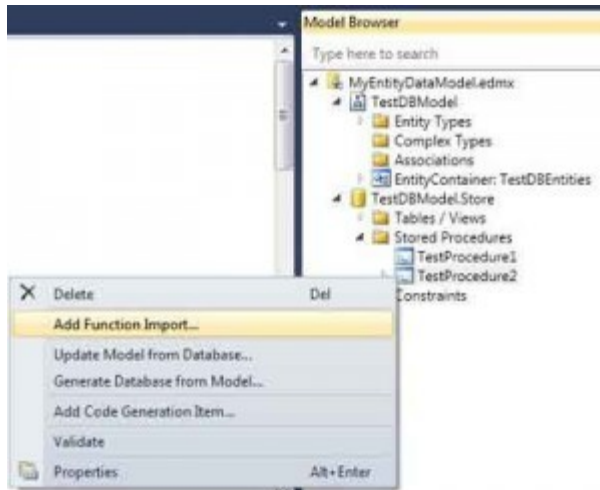
In order to use stored procedure (SP) in EF we need to follow two steps.

- *Add the stored procedure in EDM*

We can add stored procedure in EDM as like table insertion process. But if need to add SP in existing EDM, then right click on designer and click on “Update model from database...”. A popup update wizard will be open where we can select stored procedures and add it. Now we have to map this stored procedure to the conceptual model. For that we have to follow second step.

- *Add function import*

Right click on the designer surface and select “Model Browser”. From the Stored Procedures node, right click on the stored procedure and select “Add Function Import...”. Select four types of return values and click ok. Now we can write codes in to access our SP.



- Consider we have two stored procedure TestProcedure1, and TestProcedure2. The procedure TestProcedure1 has no parameter, but TestProcedure2 has two integer type parameters. Sample EF query that call a stored procedure (SP) is given bellow:

```
TestDBEntities db = new TestDBEntities();
```

```
GridView1.DataSource = db.TestProcedure1();
```

```
GridView1.DataBind();
```

```
GridView2.DataSource = db.TestProcedure2(1, 5);
```

```
GridView2.DataBind();
```

What are the return types of stored procedure in entity framework?

When we include a stored procedure in a EDM then, we can select four types of return values. A small descriptions is given bellow.

- *None*

None means stored procedure will not return any value.

- *Scalars*
Scalars means stored procedure will return single value of selected type like Boolean, binary etc.
- *Complex*
Complex means stored procedure will return complex type which is only on conceptual model but not in database table.
- *Entities*
Entities means stored procedure will return collection of selected entities as like table.

<http://cybarlab.com/ef-interview-questions-with-answers>