

Entity Framework Interview Questions & Answers

EF 6.x



By Shailendra Chauhan

Microsoft MVP, Founder & CEO - Dot Net Tricks

 **DotNetTricks**

Entity Framework Interview Questions & Answers

All rights reserved. No part of this book can be reproduced or stored in any retrieval system or transmitted in any form or by any means, electronic, mechanical, photocopying, recording, uploading on server and scanning without the prior written permission of the Dot Net Tricks Innovation Pvt. Ltd.

The author of this book has tried their best to ensure the accuracy of the information described in this book. However, the author cannot guarantee the accuracy of the information contained in this book. The author or Dot Net Tricks Innovation Pvt. Ltd. will not be liable for any damages, incidental or consequential caused directly or indirectly by this book.

Further, readers should be aware that the websites or reference links listed in this book may have changed or disappeared between when this book was written and when it is read.

All other trademarks referred to in this book are the property of their respective owners.

Release History

- Initial Release 1.0.0 - 18th Jan 2019



About Dot Net Tricks

Dot Net Tricks is founded by Shailendra Chauhan (Microsoft MVP), in Jan 2010. Dot Net Tricks came into existence in form of a blog post over various technologies including .NET, C#, SQL Server, ASP.NET, ASP.NET MVC, JavaScript, Angular, Node.js and Visual Studio etc.

The company which is currently registered by a name of Dot Net Tricks Innovation Pvt. Ltd. came into the shape in 2015. Dot Net Tricks website has an average footfall on the tune of 300k+ per month. The site has become a cornerstone when it comes to getting skilled-up on .NET technologies and we want to gain the same level of trust in other technologies. This is what we are striving for.

We have a very large number of trainees who have received training from our platforms and immediately got placement in some of the reputed firms testifying our claims of providing quality training. The website offers you a variety of free study material in form of articles.

Dot Net Tricks Training Solutions

Dot Net Tricks provide you training in traditional as well as new age technologies, via various formats.

Master Courses (Instructor-led)

For a beginner who needs regular guidance, we have a fully packed Master Courses. They are almost equal to semester courses taught in engineering colleges when it comes to length, breadth of content delivery, the only difference instead of 5-6 months, they take approx. 16-weekend classes (2 months).

The detail about Master courses can be found here: <https://www.dotnettricks.com/instructor-led-courses>

Hands-On Learning (Learn to Code)

Dot Net Tricks offers hands-on learning courses, which give you the confidence to code and equally helpful to work in real-life scenarios. This course is composed with hands-on exercise using IDE or cloud labs so that you can do hands-on practice on everything learned on this platform. While creating these courses we have ensured that quality of courses doesn't get compromise at any parameter, and they also will be able to produce the same results as our other course formats, given the fact you will be able to put your own honest effort.

The detail about Hands-On Learning courses can be found here: <https://www.scholarhat.com>

Skill Bootcamps (Instructor-led)

Professionals who don't have two months' time and want to get skilled up in least possible time due to some new project that their company has to take in very near future, we have designed Skill Bootcamps Concept, where you will get trained on consecutive days in fast-paced manner, where our full focus is going to be on hands-on delivery of technological exercises. Where you will be going to gain confidence to handle any particular technology on an enterprise scale. In short, if I say, Skill Bootcamps are actually our Master's program minus theory.

The detail about Skill Bootcamps can be found here: <https://www.dotnettricks.com/skill-bootcamp>

Self-paced Courses (Video Courses)

Dot Net Tricks offers the Self-paced courses, which give you the liberty to study at your own pace, time and place. We understand everyone has their own comfort zone, some of you can afford to dedicate 2 hours a day, some of you not. Keeping this thing in mind, we created these self-paced courses. While creating these courses we have ensured that quality of courses doesn't get compromise at any parameter, and they also will be able to produce the same results as our other course formats, given the fact you will be able to put your own honest effort.

The detail about Self-paced courses can be found here: <https://www.dotnettricks.com/self-paced-courses>

Corporate Training (Online and Classroom)

Dot Net Tricks having a pool of mentors who help the corporate to enhance their employment skills as per changing technology landscape. Dot Net Tricks offers customized training programs for new hires and experienced employees through online and classroom mode. As a trusted and resourceful training partner, Dot Net Tricks helps the corporate to achieve success with its industry-leading instructional design and customer training initiatives.

Apart from these, we also provide on-demand boot camps and personalized project consultation.

The detail about Corporate Training can be found here: <https://www.dotnettricks.com/corporate-training>

Learning Platforms

We have very robust technology platforms to answer the needs of all our trainees, no matter in which program they have enrolled. We offer two self-intuitive Learning Management Systems (LMS), which help to our learners to learn code by doing and evaluates their learning.

We offer the following two Learning Platforms for learning technologies:

1. Dot Net Tricks: <https://www.dotnettricks.com>
2. Scholar Hat: <https://www.scholarhat.com>

Dedication

My mother Mrs Vriksha Devi and my wife Reshu Chauhan deserve to have their name on the cover as much as I do for all their support made this possible. I would like to say thanks to all my family members Virendra Singh(father), Jaishree and Jyoti(sisters), Saksham and Pranay(sons), friends, to you and to readers or followers of my articles at <https://www.dotnettricks.com/mentors/shailendra-chauhan> to encourage me to write this book.

-Shailendra Chauhan



Introduction

Writing a book has never been an easy task. It takes a great effort, patience and consistency with strong determination to complete it. Also, one should have a depth knowledge over the subject is going to write.

So, what where my qualification to write this book? My qualification and inspiration come from my enthusiasm for and the experience with the technology and from my analytic and initiative nature. Being a trainer, analyst, consultant and blogger, I have thorough knowledge and understandings of .NET technologies. My inspiration and knowledge have also come from many years of my working experience and research over it.

What This Book Is

EF is an O/RM framework to query the database in an object-oriented fashion. This book will teach you Entity Framework concepts from scratch to advance with the help of Interview Questions & Answers. Here, you will about the EF fundamentals, data modelling approaches, relationship, database migrations and querying database. This book covers Entity Framework version 1.x, 4.x, 5.x and 6.x.

What You'll Learn

This book is for .NET developers who are looking for a change or want to make a bright future in Entity Framework. This book covers the interview questions on the following topics:

- Introduction to O/RM
- Entity Framework version history.
- EF core features.
- Entity Framework Architecture.
- Data Modelling approaches: Database First, Model First and Code First.
- Defining Relationship among Entities using the Code First Approach.
- Entity Framework Inheritance.
- Entity Life Cycle and its methods.
- Defining Database Relationships like as 1:1, 1:M, M:M
- Database Migrations for DB migration and rollback.
- Entity loading: Eager, Lazy and Explicit Loading.
- Executing Plain SQL commands.
- Using Transactions with Entity Framework.

Our best wishes always with you for your learning and growth!

About the Author

Shailendra Chauhan - An Entrepreneur, Author, Architect, Corporate Trainer, and Microsoft MVP



He is the **Founder and CEO** of Dot Net Tricks which is a brand when it comes to e-Learning. Dot Net Tricks provides training and consultation over an array of technologies like **Cloud, .NET, Angular, React, Node and Mobile Apps development**. He has been awarded as **Microsoft MVP** three times in a row (2016-2018).

He has changed many lives from his writings and unique training programs. He has a number of most sought-after books to his name which have helped job aspirants in **cracking tough interviews** with ease.

Moreover, and to his credit, he has delivered **1000+ training sessions** to professionals worldwide in Microsoft .NET technologies and other technologies including JavaScript, AngularJS, Node.js, React and NoSQL Databases. In addition, he provides **Instructor-led online training, hands-on workshop** and **corporate training** programs.

Shailendra has a strong combination of **technical skills and solution development for complex application architecture with proven leadership and motivational skills** have elevated him to world-renowned status, placing him at the top of the list of most sought-after trainers.

"I always keep up with new technologies and learning new skills to deliver the best to my students," says Shailendra Chauhan, he goes on to acknowledge that the betterment of his followers and enabling his students to realize their goals are his prime objective and a great source of motivation and satisfaction.

Shailendra Chauhan - **"Follow me and you too will have the key that opens the door to success"**

How to Contact Us

Although the author of this book has tried to make this book as accurate as it possible but if there is something strikes you as odd, or you find an error in the book please drop a line via e-mail.

The e-mail addresses are listed as follows:

- mentor@dotnettricks.com
- info@dotnettricks.com

We always happy to hear from our readers. Please provide your valuable feedback and comments!

You can follow us on [YouTube](#), [Facebook](#), [Twitter](#), [LinkedIn](#) and [Google Plus](#) or subscribe to [RSS feed](#).



Table of Contents

| | |
|--|-----------|
| Entity Framework Interview Questions & Answers | 1 |
| About Dot Net Tricks | 2 |
| Dot Net Tricks Training Solutions..... | 2 |
| Dedication | 4 |
| Introduction | 5 |
| About the Author | 6 |
| How to Contact Us..... | 7 |
| Introducing Entity Framework | 12 |
| Q1. What is ADO.NET Entity Framework? | 12 |
| Q2. Explain the evolution history of ADO.NET Entity Framework? | 12 |
| Q3. What is O/RM? | 13 |
| Q4. What are the main components of an O/RM?..... | 13 |
| Q5. Why use O/RM?..... | 13 |
| Q6. What is the O/RMs you can use with .NET based applications? | 13 |
| Q7. What is Dapper? | 13 |
| Q8. What is Micro O/RMs? | 13 |
| Q9. What are the advantages or benefits of Entity Framework? | 13 |
| Q10. Explain Entity Framework Architecture?..... | 14 |
| Q11. How to handle SQL injection attack it in Entity Framework?..... | 15 |
| Q12. What are the differences between ADO.NET and Entity Framework? | 15 |
| Q13. What are the differences between LINQ to SQL and Entity Framework?..... | 16 |
| Database Modelling..... | 17 |
| Q1. What are various approaches to domain modelling in Entity Framework?..... | 17 |
| Q2. What is Code First approach?..... | 17 |
| Q3. What are the advantages of Code First? | 17 |
| Q4. What is the Model First approach? | 18 |
| Q5. What are the advantages of Model First?..... | 18 |
| Q6. What is Database First? | 18 |
| Q7. What are the advantages of the Database First approach? | 18 |
| Q8. What are the types of entities in EF? | 19 |

| | | |
|----------------------------|--|-----------|
| Q9. | What are POCO classes? | 19 |
| Q10. | What is the proxy object?..... | 19 |
| Q11. | How to disable dynamic proxy in EF? | 19 |
| Q12. | How to decide which Database modelling approach will suit you? | 20 |
| Q13. | What is the difference between POCO and Code First/Code Only? | 20 |
| Q14. | What is EDM or Entity Data Model?..... | 20 |
| Q15. | What is the role of EDM Wizard or Entity Data Model Wizard? | 21 |
| Q16. | What is the purpose of EDM or .edmx file? | 21 |
| Q17. | Can we edit/modify EDM or .edmx file? | 21 |
| Q18. | What is pluralize or singularize object names option in Entity Data Model? | 21 |
| Q19. | What are T4 templates? | 22 |
| Q20. | What is the role of T4 templates in Entity framework? | 23 |
| Q21. | Why Entity Framework is slow? | 24 |
| Q22. | How to improve EF performance? | 24 |
| Q23. | Who is responsible for tracking changes in an entity?..... | 25 |
| Q24. | What is the role of ObjectStateManager? | 25 |
| Relationships | | 26 |
| Q1. | What is EntityClient? | 26 |
| Q2. | What is Entity Type?..... | 26 |
| Q3. | What is Entity Container?..... | 27 |
| Q4. | What is Entity Set?..... | 27 |
| Q5. | What is Association Set? | 27 |
| Q6. | What is Entity Key?..... | 27 |
| Q7. | What is Association?..... | 28 |
| Q8. | What Association does EF support? | 28 |
| Q9. | What is foreign key association? | 28 |
| Q10. | What is Independent association? | 29 |
| Q11. | What is Complex Type? | 29 |
| Q12. | What is scalar property?..... | 31 |
| Q13. | What is navigation property? | 31 |
| Q14. | How to define a navigation property using EF Code First? | 31 |

| | | |
|-------------------------|---|-----------|
| Q15. | What are the various Entity States in EF? | 32 |
| Q16. | Explain Entity Life Cycle? | 32 |
| Q17. | What are different types of inheritance in Entity Framework? | 33 |
| Q18. | What is TPH inheritance in EF?..... | 33 |
| Q19. | What is TPT inheritance in EF? | 34 |
| Q20. | What is TPC inheritance in EF? | 36 |
| Q21. | What is DbContext API and how it works?..... | 37 |
| Q22. | What is DLL for DbContext API? | 37 |
| Q23. | What isObjectContext API and how it works?..... | 38 |
| Q24. | What are the differences between ObjectContext API and DbContext API?..... | 38 |
| Q25. | What are various approaches in Code First for model designing?..... | 38 |
| Q26. | What is fluent API and how it is different from data annotations? | 39 |
| Q27. | Which method is required to configure fluent API?..... | 39 |
| Q28. | What C# Datatype is mapped with which Datatype in SQL Server? | 39 |
| Q29. | Give some example to define mapping using fluent API?..... | 39 |
| Q30. | What default relationship convention EF6 supports?..... | 40 |
| Q31. | How to define mapping using data annotation in Entity Framework? | 40 |
| Q32. | How to define One-to-Zero-or-One (1:0 or 1:1) mapping in Entity Framework? | 40 |
| Q33. | How to define One-to-Many (1:m) mapping in Entity Framework? | 41 |
| Q34. | How to define Many-to-Many (m:m) mapping in Entity Framework?..... | 42 |
| Q35. | How to define CUD Operations using Stored Procedures in EF 6? | 43 |
| Migrations | | 44 |
| Q1. | What is Code First Migrations in Entity Framework?..... | 44 |
| Q2. | What are the various Code First Database initializers?..... | 44 |
| Q3. | Why use Database Migrations?..... | 45 |
| Q4. | Can you create custom database Initializers in Entity Framework Code First? | 45 |
| Q5. | Explain step by step Code First Migrations in Entity Framework?..... | 45 |
| Q6. | How to update existing Database using Code First Migrations?..... | 48 |
| Q7. | What is Migrations History Table? | 49 |
| Q8. | How to Undo/Rollback a Migrations? | 49 |
| Q9. | What is automatic migration? | 50 |

| | |
|--|-----------|
| Q10. How to disable automatic migration? | 50 |
| Q11. What is Seed Data in EF Code First? | 50 |
| Querying Database | 51 |
| Q1. What is DbSet? | 51 |
| Q2. What is ObjectSet? | 51 |
| Q3. What is EntitySQL? | 51 |
| Q4. How EntitySQL is different from T-SQL? | 51 |
| Q5. What is SQL injection attack? | 51 |
| Q6. What is Lazy/Deferred Loading? | 52 |
| Q7. How to turn off Lazy/Deferred Loading? | 53 |
| Q8. What is Eager Loading? | 53 |
| Q9. How to execute plain SQL in EF6? | 54 |
| Q10. How to use DbSet.SqlQuery() method? | 54 |
| Q11. How to use DbContext.Database.SqlQuery() method? | 54 |
| Q12. How to use DbSet.Database.ExecuteSqlCommand () method? | 54 |
| Q13. How to load an Entity explicitly in EF? | 54 |
| Q14. How to query and save your data asynchronously in EF 6? | 55 |
| Q15. Write an asynchronous query in EF 6? | 55 |
| Q16. Write a query to save data in async way in EF 6? | 55 |
| Q17. How does EF support Transaction? | 55 |
| Q18. How to use a transaction explicitly in EF? | 55 |
| References | 56 |

Introducing Entity Framework

Q1. What is ADO.NET Entity Framework?

Ans. ADO.NET Entity Framework is an ORM framework that empowers developers to work with various relational databases like SQL Server, Oracle, DB2, MYSQL etc. It allows developers to deal with data as objects or entities. Using the Entity Framework, developers issue queries using LINQ, then retrieve and manipulate data as strongly typed objects using C# or VB.NET Framework.

Q2. Explain the evolution history of ADO.NET Entity Framework?

Ans. The Evolution history of ADO.NET Entity Framework is given below:

| Version | Features Details | Supported Framework & IDE |
|---------|---|---|
| 6.0 | <ul style="list-style-type: none"> • Async Query and Save • Code-Based Configuration • Dependency Resolution • Interception/SQL logging • Improved Connection Management • Improved Transaction Support | .NET 4.5.1 and Visual Studio 2013 |
| 5.0 | <ul style="list-style-type: none"> • Enum Support in Code First and EF Designer • Spatial Data Types in Code First and EF Designer • Table-Valued Functions • Multiple Diagrams per Model | .NET 4.5 and Visual Studio 2012 |
| 4.3 | <ul style="list-style-type: none"> • Code First Migrations • Automatic Migrations | .NET 4.0 and Visual Studio 2010 |
| 4.2 | The EF 4.2 release included the bug fixes to EF 4.1 | .NET 4.0 and Visual Studio 2010 |
| 4.1 | <ul style="list-style-type: none"> • Code First development • Introduced DbContext API • Data Annotations and Fluent API Validation | .NET 4.0 and Visual Studio 2010 |
| 4.0 | <ul style="list-style-type: none"> • Model-first development • POCO support • Lazy Loading • T4 Code Generation | .NET 4.0 and Visual Studio 2010 |
| 3.5 | This release provided basic O/RM support using the Database first development. | .NET 3.5 SP1 and Visual Studio 2008 SP1 |

Q3. What is O/RM?

Ans. O/RM stands for object-relational mapping. In fact, it is a tool which is used to query and manipulate data stored in the relational databases (like SQL Server, Oracle, DB2 etc.) by using model objects and with minimum code.

Q4. What are the main components of an O/RM?

Ans. There are following main components of an O/RM:

1. **Model objects:** - These are business class objects mapped to relational database objects.
2. **Relational Database objects:** - These are relational database Tables, Views, Functions & Stored Procedures.
3. **Mapping:** - It describes how model objects are mapped to relational database objects (tables, views & stored procedures). By mapping, it is defined how a class and its properties are mapped to a database object.

The mapping information is generally expressed as an XML file but some O/RM tools use attributes on the classes and their properties to maintain mapping data

Q5. Why use O/RM?

Ans. OR/M enables us to keep our database design separate from our model classes design. This makes our application more maintainable and extendable. OR/M also automates standard CRUD operations (Create, Read, Update and Delete). So, we don't need to implement these manually.

Q6. What is the O/RMs you can use with .NET based applications?

Ans. The following O/RMs, you can use with .NET based applications:

- Entity Framework 6.x
- Entity Framework Core
- Dapper
- N Hibernate

Q7. What is Dapper?

Dapper is simple/ micro ORM for the .NET world. Dapper was created by StackOverflow team to address their issues and open source it. It's a NuGet library that can be added to any .NET project for database operations.

Q8. What is Micro O/RMs?

Ans. A Micro ORM is architected to focus on the most important task of working with database tables instead of creating, modifying the database schema, tracking changes etc. EF 6.x and EF Core both are O/RMs since they provide a full set of features.

Q9. What are the advantages or benefits of Entity Framework?

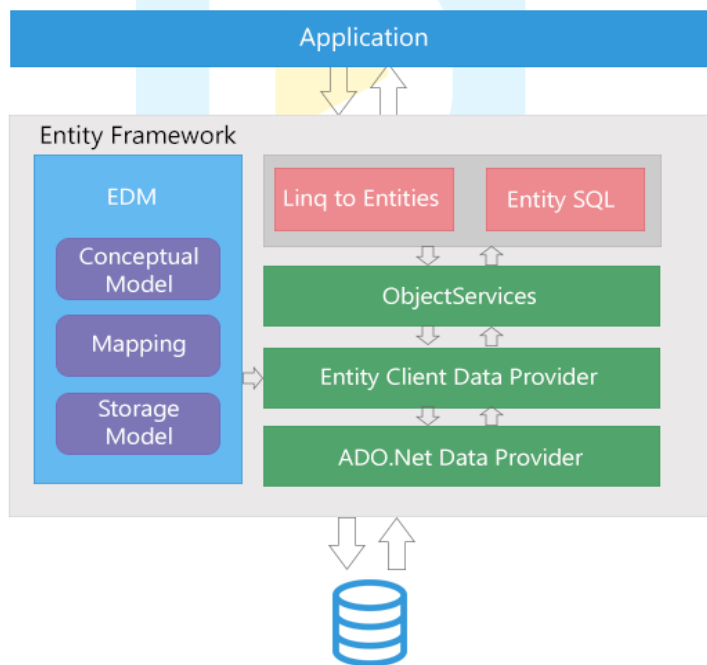
Ans. There are following advantages of using entity framework

1. **Productivity:** - Entity Framework makes the developer's life easier than ever since it automates standard CRUD operations (Create, Read, Update & Delete). Moreover, Entity Framework designer integrated into Visual Studio also simplifies the mapping process. All these things cause more productivity.
2. **Maintainability:** - Entity Framework keeps our database design separate from our model classes design. This makes our application more maintainable and extendable.
3. **Performance:** - Entity Framework is a wrapper around traditional ADO.NET to make the database manipulation simple and easy. This complexity of Entity Framework makes it slow in performance as compared to ADO.NET.

Q10. Explain Entity Framework Architecture?

Ans. Let's understand the components of the EF architecture.

- **EDM (Entity Data Model):** EDM has three parts: Conceptual model, Mapping and Storage model.
 1. **Conceptual Model:** The conceptual model contains the classes with their relationships in your code. These classes are mapped with the Database objects like tables, views, result sets etc.
 2. **Storage Model:** The storage model contains information about the database objects like tables, views, stored procedures, functions and their relationships.
 3. **Mapping:** The mapping contains information about how the conceptual model classes are mapped to the storage model database objects.



- **LINQ to Entities:** It follows LINQ syntax to query and manipulate the database. It returns conceptual model classes as result sets. This is the widely used way to query in EF.
- **Entity SQL:** It is a query language which is used to query the database. This syntax is not so much popular and used to query the database.

- **Object Service:** This is the entry point to access data from the database and return it back. This is responsible for converting data returned from an entity client data provider to an entity object structure.
- **Entity Client Data Provider:** This is responsible for converting LINQ-to-Entities or Entity SQL queries into plain T-SQL queries which are further executed at the database side. It communicates with the ADO.Net data provider to execute the generated plain SQL.
- **ADO.Net Data Provider:** This layer is used to communicate with the database using standard ADO.Net providers for each RDBMS like Oracle, SQL Server, MySQL etc.

Q11. How to handle SQL injection attack in Entity Framework?

Ans. Entity Framework is **injection safe** since it always generates parameterized SQL commands which help to protect our database against SQL Injection.

A SQL injection attack can be made in **Entity SQL** syntax by providing some malicious inputs that are used in a query and in parameter names. To avoid this one, you should never combine user inputs with Entity SQL command text.

Hence, **avoid returning IQueryable** results when exposing methods to potentially untrusted callers since it could call methods on the result that expose secure data or increase the size of the result set like as:

```
public IQueryable<Customer> GetCustomer(int customerId)
```

A consumer of this query could call **Include("Orders")** on the returned **IQueryable<Customer>** to retrieve data that the query did not intend to expose. This can be avoided by changing the return type of the method to **IEnumerable** and calling a method (such as **.ToList()**) that materializes the results.

Q12. What are the differences between ADO.NET and Entity Framework?

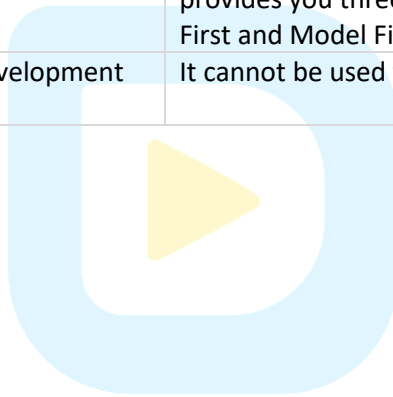
Ans. There are following differences between ADO.NET and Entity Framework:

| ADO.NET | Entity Framework |
|--|--|
| It is a part of the .NET Framework framework since .NET Framework 1.0 | It is a part of the .NET Framework framework since .NET Framework 3.5 |
| SqlConnection/OleDbConnection are used for database connectivity. | We can use EntitySQL,ObjectContext, DbContext for database connectivity. |
| Difficult to debug and cause syntax errors at run-time. | Easy to debug and cause syntax errors at compile-time. |
| It has full type checking at compile-time and Intellisense support in Visual Studio since it used the T-SQL to query the database. | It has full type checking at compile-time and Intellisense support in Visual Studio since it used the .NET Framework framework languages like C# and VB. |
| It used T-SQL to query the data to query the database and some other syntax for querying the other data source. | It used LINQ to query the data which provides the uniform programming model (means common query syntax) to query the various data sources. |
| Easier syntax and coding. | Syntax and coding is somewhat complex. |

Q13. What are the differences between LINQ to SQL and Entity Framework?

Ans. There are following differences between LINQ to SQL and Entity Framework:

| LINQ to SQL | Entity Framework |
|--|---|
| It only works with SQL Server Database. | It can work with various databases like Oracle, DB2, MYSQL, SQL Server etc. |
| It generates a .dbml to maintain the relation | It generates a .edmx files initially. The relation is maintained using 3 different files .csdl, .msl and .ssdl |
| It has no support for the complex type. | It has support for the complex type. |
| It cannot generate a database from the model. | It can generate the database from the model. |
| It allows only one to one mapping between the entity classes and the relational tables /views. | It allows one-to-one, one-to-many & many-to-many mappings between the Entity classes and the relational tables /views |
| It allows you to query data using DataContext. | It allows you to query data using EntitySQL,ObjectContext, DbContext. |
| It provides a tightly coupled approach. | It provides a loosely coupled approach. Since it provides you three approaches: Code First, Database First and Model First. |
| It can be used for rapid application development with SQL Server. | It cannot be used for rapid application development. |



2

Database Modelling

Q1. What are various approaches to domain modelling in Entity Framework?

Ans. There are three approaches to domain modelling which was introduced with Entity Framework 4.1:

Database First (VS 2008 and EF 1)



Model First (VS 2010 and EF 4.0)



Code First (VS 2010 and EF 4.1)



Q2. What is Code First approach?

Ans. Code first is one of the **domain modelling** approach in Entity Framework to design database. It enables developers to describe a model by using C# or VB.NET classes and then create database objects from these classes. Such classes are called POCO classes.

This approach enables developers to work entirely in an object-oriented fashion, and not worry about the structure of the database. This abstraction allows developers to make a more logically and flexible application that focuses on the behaviour of the application rather than the database generated by it.

Q3. What are the advantages of Code First?

Ans. There are following advantages of Code First approach:

- It is a very popular approach since it allows you to make a **more logically and flexible** application.
- It provides full control over the code since there is no auto-generated code which is difficult to modify.

- In this approach, your code **defines only the database mappings** and EF will take care of the creation of a database with its relations.
- In this approach, manual changes to database schema are not recommended because your model classes define the database schema.
- This approach can also be used with your **existing database**.

Q4. What is the Model First approach?

Ans. The model first is one of the **domain modelling** approach in Entity Framework to design database. It allows developers to define Entities and relationships on the design surface of the empty **model** (.edmx file) using Visual Studio toolbox. In this approach after designing the entities and their relationships, developers create SQL scripts and run it SQL side to create a database from it. This approach is generally adopted by the architect and solution lead developers in a company.

In this approach, you must select the option “Empty Model” in place of “**Generate from database**” option while creating the Entity Data Model using Visual Studio.

Q5. What are the advantages of Model First?

Ans. There are following advantages of Model First Approach:

- This approach is useful when you like to visualize the structure and relations of the database objects.
- This is good when you don't prefer to write code or SQL for generating the database since it generates both the things for us.
- This approach is not good for big projects but for **smaller** projects, this approach will be very productive.
- To add extra properties to a POCO class, you need to use partial classes.
- In this approach, manual changes to database schema are not recommended since your model entities define the database schema.

Q6. What is Database First?

Ans. Database first is one of the most popular approaches to domain modelling in Entity Framework. This approach enables developers to create a model from an existing database (like SQL Server, MySQL, Oracle, DB2 etc.). In this approach, the code is generated automatically by the Visual Studio. But this approach also restricts developers to work with the structure of the generated code by Visual Studio.

Q7. What are the advantages of the Database First approach?

Ans. There are following advantages of Code First Approach:

- This is the most popular approach used with EF. It is useful when you have a Database designed by DBAs, or if you have an existing Database.
- In this approach, the Visual Studio wizard creates entities and relationships for you. Even, when you do any modification at database side then you can re-generate or update the entities and their relationships.
- To add extra properties or methods into generated entities you need to use partial classes.
- In this approach, manual changes to the database are accepted since the database defines your entities. Here, you have to always update model from the database.

Q8. What are the types of entities in EF?

Ans. There are two types of entities are supported in EF:

1. POCO Class/Entity (Plain Old CLR Object)
2. Dynamic Proxy Object/Entity (POCO Proxy)

Q9. What are POCO classes?

Ans. The term POCO does not mean to imply that your classes are either plain or old. The term POCO simply specifies that the POCO classes don't contain any reference that is specific to the entity framework or .NET framework.

Basically, POCO (Plain Old CLR Object) entities are existing domain objects within your application that you use with Entity Framework.

Q10. What is the proxy object?

Ans. An object that is created from a POCO class or entities generated by the Entity Framework to support change tracking and lazy loading, is known as a proxy object.

There are some rules for creating a proxy class object:

1. The class must be public and not sealed.
2. Each property must be marked as virtual.
3. Each property must have a public getter and setter.
4. Any collection navigation properties must be typed as `ICollection<T>`.

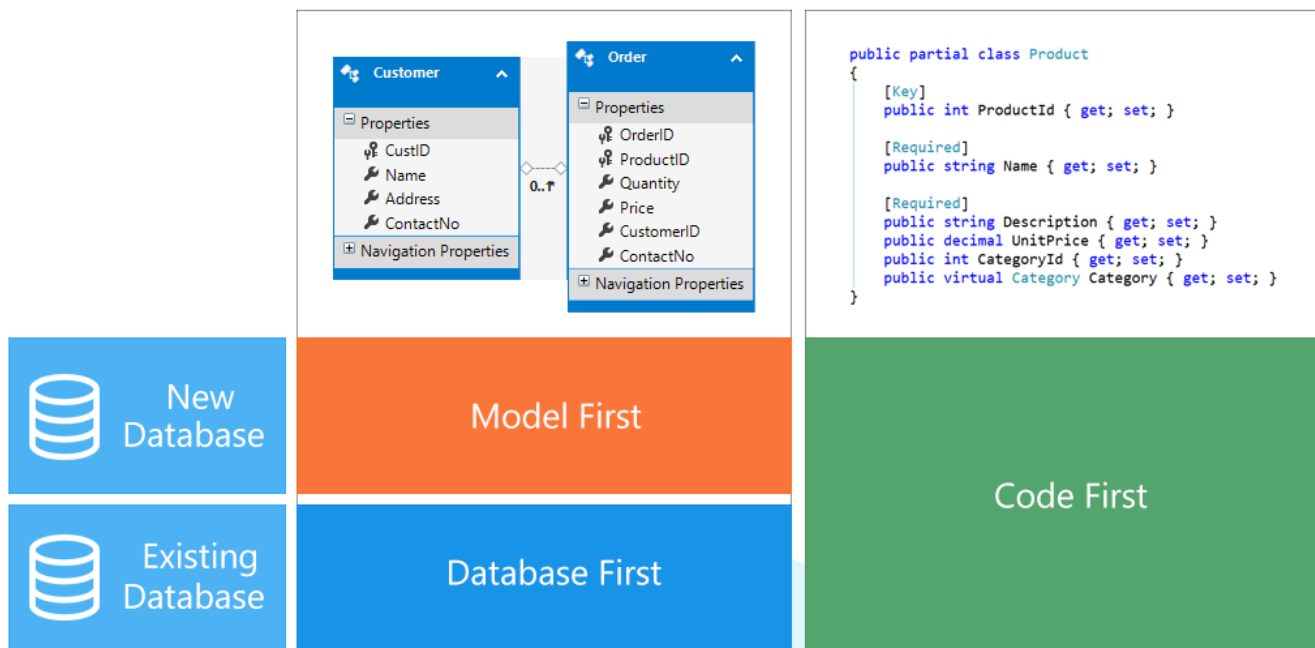
Q11. How to disable dynamic proxy in EF?

Ans. By default, the dynamic proxy is enabled for every entity in EF. A dynamic proxy can be disabled by setting the `context.Configuration.ProxyCreationEnabled = false;` in the context class constructor as given below:

```
public class DatabaseContext : DbContext
{
    public DatabaseContext() : base("DefaultConnection")
    {
        Configuration.ProxyCreationEnabled = false;
    }
}
```

Q12. How to decide which Database modelling approach will suit you?

Ans. This can be understood by the following diagram:



Q13. What is the difference between POCO and Code First/Code Only?

Ans. Entity Framework with POCO, allows you to map your own POCO entities to the mapping defines in XML form within the EDMX XML file. This is not the best approach to manage extremely large models with XML mappings since you need to manually write the code for mapping POCO entities.

Code first is better than POCO based approach. Since, this approach enables developers to work entirely in an object-oriented fashion, and not worry about the structure of the database. This abstraction allows developers to make a more logically and flexible application that focuses on the behaviour of the application rather than the database generated by it.

Q14. What is EDM or Entity Data Model?

Ans. An Entity Data Model (EDM) acts as a bridge between application and database. It enables us to work with the conceptual view of data rather than the actual database schema. It has three parts:

1. **Conceptual Model (Conceptual schema definition language (CSDL)):** It describes the model classes and their relationships.
2. **Storage Model (Store schema definition language (SSDL)):** It describes the database schema which includes tables, views, stored procedures and their relationships and keys.
3. **Mapping (Mapping specification language (MSL)):** It describes how the conceptual model is mapped to the storage model.

These three parts of the model are generated automatically by the Entity Data Model Wizard.

```

<?xml version="1.0" encoding="utf-8"?>
<edmx:Edmx Version="3.0" xmlns:edmx="http://schemas.microsoft.com/ado/2009/11/edmx">
  <!-- EF Runtime content -->
  <edmx:Runtime>
    <!-- SSDL content -->
    <edmx:StorageModels>...</edmx:StorageModels>
    <!-- CSDL content -->
    <edmx:ConceptualModels>...</edmx:ConceptualModels>
    <!-- C-S mapping content -->
    <edmx:Mappings>...</edmx:Mappings>
  </edmx:Runtime>
  <!-- EF Designer content (DO NOT EDIT MANUALLY BELOW HERE) -->
  <Designer xmlns="http://schemas.">...</Designer>
</edmx:Edmx>

```

Q15. What is the role of EDM Wizard or Entity Data Model Wizard?

Ans. It is used to create an entity data model from the relational database.

Q16. What is the purpose of EDM or .edmx file?

Ans. A .edmx file is an XML based file that includes information about your database schema. This file contains mainly three parts: a **conceptual model**, a **storage model**, and the **mapping** between these two models. A .edmx file also contains information that is used by the Entity Framework Designer to render a model graphically.

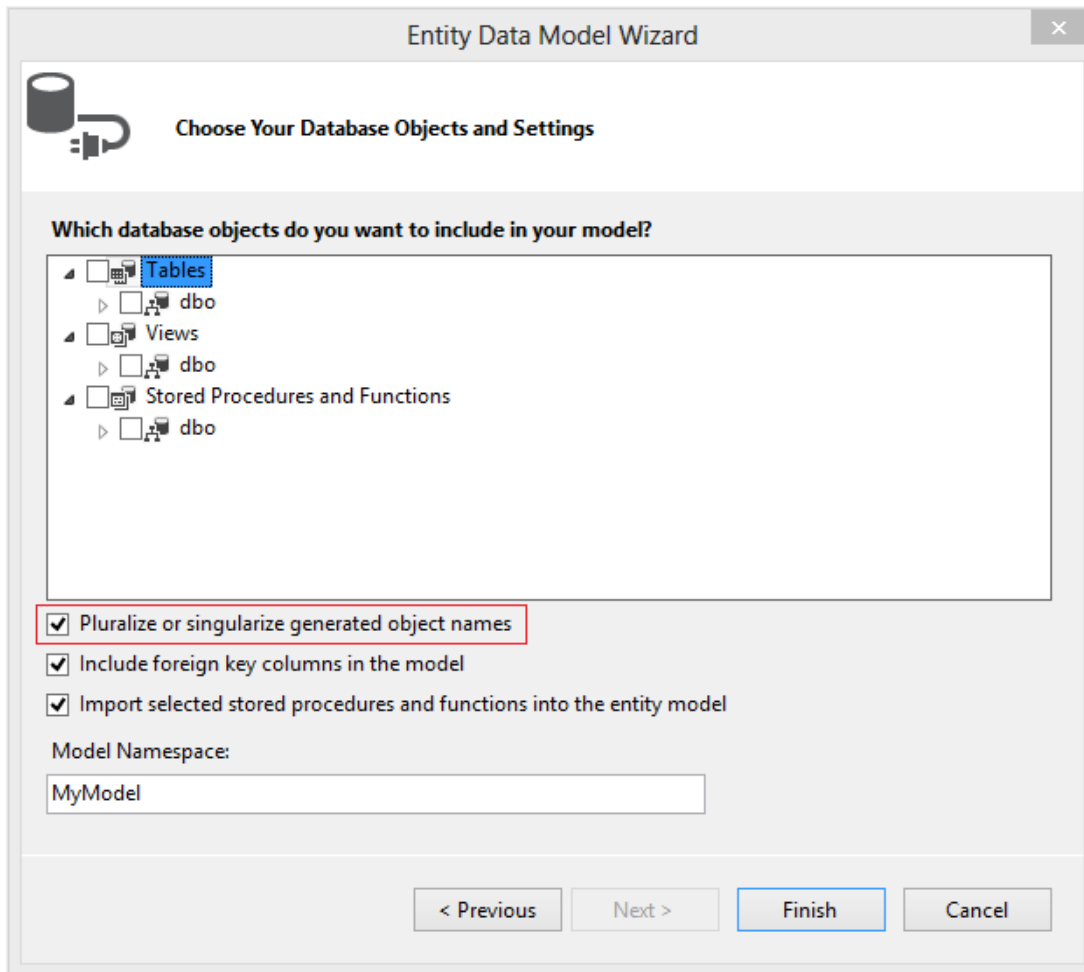
Q17. Can we edit/modify EDM or .edmx file?

Ans. Yes, we can do. By default, a .edmx file opens with the Entity Framework Designer included in the Visual Studio. However, we can open a .edmx file with the XML Editor for inspecting or editing.

The best practice for creating and editing a .edmx file is to use the EF Designer but there might be some scenarios when you required editing a .edmx file manually.

Q18. What is pluralize or singularize object names option in Entity Data Model?

Ans. In Entity Data Model Wizard, pluralize or singularize option is used to make generated objects names plural or singular which is a coding convention. Now, it depends on you which naming convention you would like to follow.



1. When you check this option, generated objects from the database would have plural name means employee object name would become employees.
2. When you uncheck this option, generated objects from the database would have singular name means employee object name remains the same.

Q19. What are T4 templates?

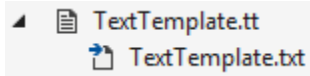
Ans. T4 stands for Text template transformation toolkit which is a template-based code generation engine built into Visual Studio. It is available in Visual Studio from Visual Studio 2008 and higher version of Visual Studio. T4 engine allows you to generate C#, T-SQL, XML or any other text files by using ASP.NET – ASPX template like syntax. T4 template has .tt extension.

For example, TextTemplate.tt T4 template has the following code:

```
<#@ template language="C#" #>
<#@ assembly name="System.Core" #>
<#@ import namespace="System.Text" #>
<#@ output extension=".txt" #>
<# Write("Hello Dot Net Tricks!"); #>
```

The T4 template processor will transform above T4 template code into a text file having extension .txt by executing the code and processing directives inside.

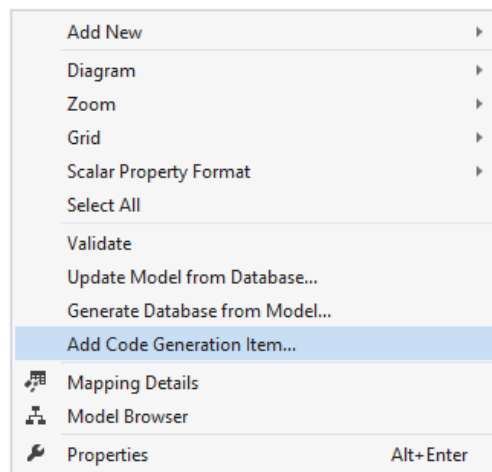
```
Hello Dot Net Tricks!
```



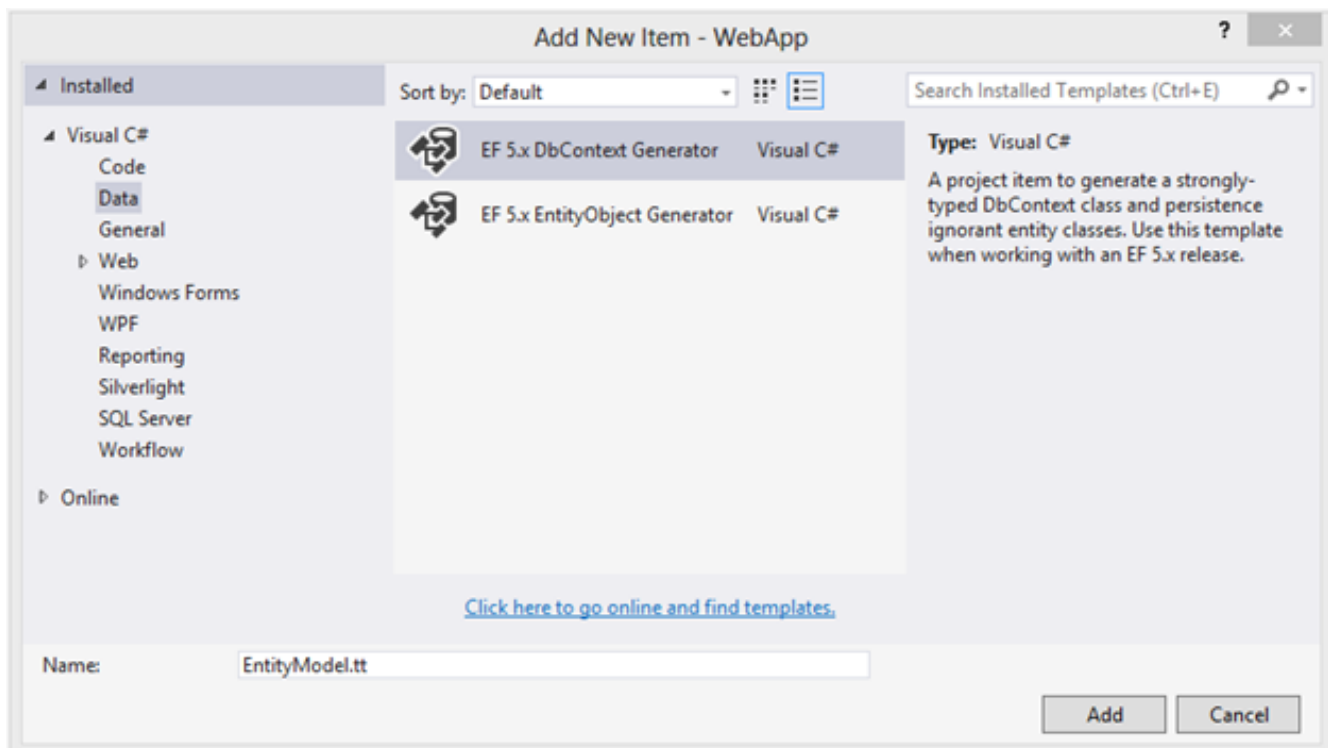
Q20. What is the role of T4 templates in Entity framework?

Ans. T4 templates in entity framework are used to generate C# or VB entity classes from EDMX files. Visual Studio 2012 provides two templates- EntityObject Generator and DbContext Generator for creating C# or VB entity classes. The additional templates are also available for download.

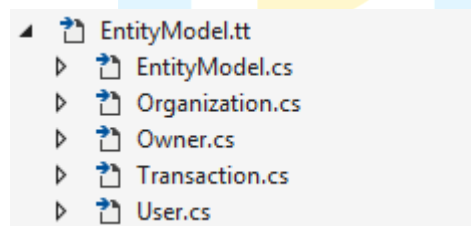
To use one of these templates, right-click on the design surface of a .edmx file and select the “Add Code Generation Item” command as given below-



Selecting the command will launch a dialogue box allowing you to select one of the installed code-generation items, or to search for new items online.



Selected template will generate the C# or VB code files i.e. entity and context classes as given below.



Q21. Why Entity Framework is slow?

Ans. Entity Framework is a great ORM for querying and managing databases. It provides an abstraction for data access in an object-oriented fashion which impact performance. There are several other EF features which also impact performance like Lazy Loading, Entity Self Tracking etc.

Q22. How to improve EF performance?

Ans. For improving EF performance disable Lazy Loading and Entity Self Tracking behaviours when you are loading or modifying a large number of records. Also, for manipulating complex operations use stored procedures instead of LINQ to Entity queries and avoid using views.

The latest version of Entity Framework 5.0 and 6.0 provides better performance by some 67 percent over EF 4.0.

Q23. Who is responsible for tracking changes in an entity?

Ans. ObjectStateManager

Q24. What is the role of ObjectStateManager?

Ans. It maintains object state and tracks modifications to the object when a user performs insert, delete, or update operation on that object. It provides the change set for updates which are used by the change processor to persist modifications.

When we add, attach, or delete an entity from the context, we actually do that against the state manager.



3

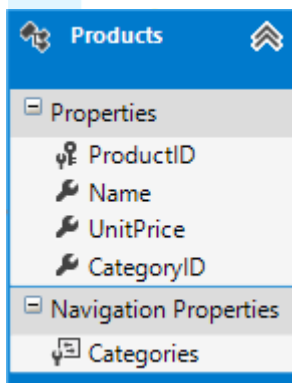
Relationships

Q1. What is EntityClient?

Ans. It is a data provider used by the Entity Framework to access data. It contains classes such as EntityConnection, EntityCommand, and EntityDataReader which are used to work with **Entity SQL** to **query** the database. For this, the namespace is **System.Data.EntityClient**.

Q2. What is Entity Type?

Ans. Entity Type specifies the structure of data within the EDM conceptual model. It includes a key, a set of properties to define data and navigation properties to define a relationship between two entities. Moreover, it must have a **unique name**.



```
<EntityType Name="Products">
  <Key>
    <PropertyRef Name="ProductID" />
  </Key>
  <Property Name="ProductID" Type="Int32" Nullable="false"
p1:StoreGeneratedPattern="Identity" />
  <Property Name="Name" Type="String" Nullable="false" MaxLength="50" Unicode="true"
FixedLength="false" />
  <Property Name="UnitPrice" Type="Double" Nullable="false" />
  <Property Name="CategoryID" Type="Int32" Nullable="false" />
  <NavigationProperty Name="Categories"
Relationship="Model.FK_dbo_Products_dbo_Categories_CategoryID" FromRole="Products"
ToRole="Categories" />
</EntityType>
```

Q3. What is Entity Container?

Ans. An entity container is a logical grouping of entity sets, association sets, and function imports. There should be at least one entity container within each conceptual model and must have a unique name.

```
<EntityContainer Name="ModelStoreContainer">
  <EntitySet Name="Categories" EntityType="Model.Store.Categories" store:Type="Tables"
Schema="dbo" />
  <EntitySet Name="Products" EntityType="Model.Store.Products" store:Type="Tables"
Schema="dbo" />
  <AssociationSet Name="FK_dbo_Products_dbo_Categories_CategoryID"
Association="Model.Store.FK_dbo_Products_dbo_Categories_CategoryID">
    <End Role="Categories" EntitySet="Categories" />
    <End Role="Products" EntitySet="Products" />
  </AssociationSet>
</EntityContainer>
```

Q4. What is Entity Set?

Ans. An entity set is defined inside an entity container. It is a logical container for instances of an entity type or instances of any type derived from that entity type.

The relationship between an entity type and an entity set looks like to the relationship between a row and a table. Like a row, an entity type describes data structure and like a table, an entity set contains instances of a given structure.

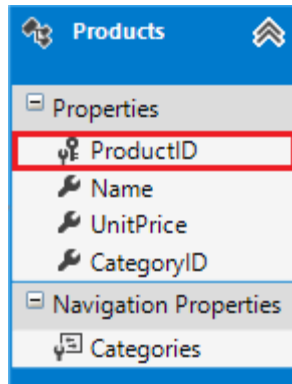
```
<EntityContainer Name="ModelStoreContainer">
  <EntitySet Name="Categories" EntityType="Model.Store.Categories" store:Type="Tables"
Schema="dbo" />
  <EntitySet Name="Products" EntityType="Model.Store.Products" store:Type="Tables"
Schema="dbo" />
  <AssociationSet Name="FK_dbo_Products_dbo_Categories_CategoryID"
Association="Model.Store.FK_dbo_Products_dbo_Categories_CategoryID">
    <End Role="Categories" EntitySet="Categories" />
    <End Role="Products" EntitySet="Products" />
  </AssociationSet>
</EntityContainer>
```

Q5. What is Association Set?

Ans. An association set is a logical container for association instances of the same type. It is defined inside an entity container as shown in the above example.

Q6. What is Entity Key?

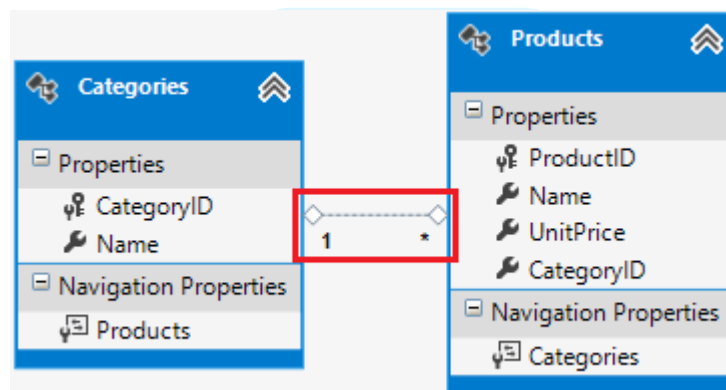
Ans. An entity key is a property or a set of properties of an entity type that are used to verify the uniqueness of a given entity. Normally, an entity key is mapped to a primary Key or unique Key field in the database.



Q7. What is Association?

Ans. Association specifies a relationship between two entities within the EDM conceptual model by the "Association" Element. In association, each relationship contains two ends, one specify the entity type and other specify the multiplicity having a value of one (1), zero or one (0..1), or many (*).

For example, an association between Category and Product entities as given below:

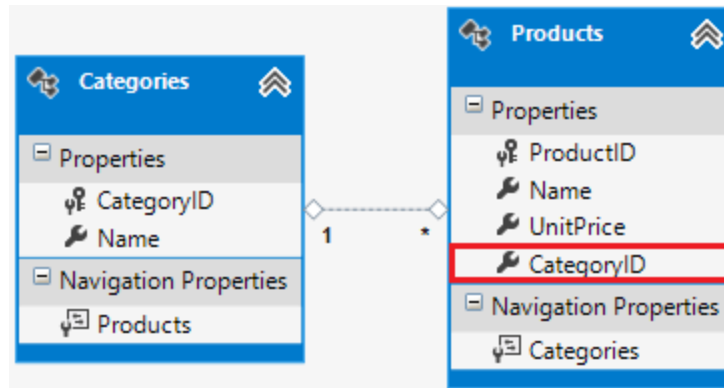


Q8. What Association does EF support?

Ans. Entity Framework supports two types of association - **Foreign key association** and **Independent association**.

Q9. What is foreign key association?

Ans. When the relationship between two entities is defined by the foreign key property, this association is called a foreign key association. Foreign key property resembles a foreign key column in a relational database. A referential integrity constraint is used to define the foreign key association between two entity types as shown below.



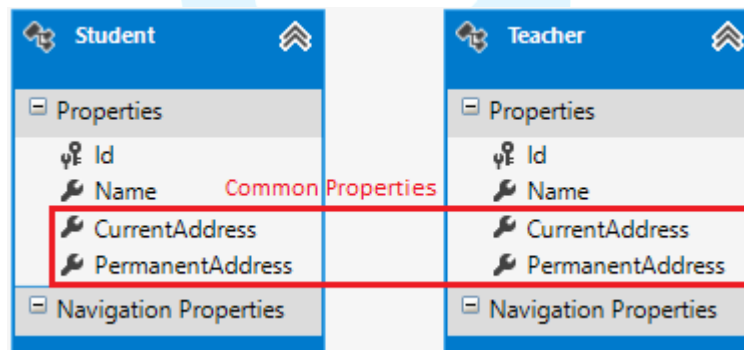
Q10. What is Independent association?

Ans. In independent association, there is no foreign key property defines within storage model means there is no physical foreign key in the database. The relationship between two entities is defined by an independent object and handle by the object state manager. It has its own object state that must be handled correctly to avoid disconnected/detached entities issues.

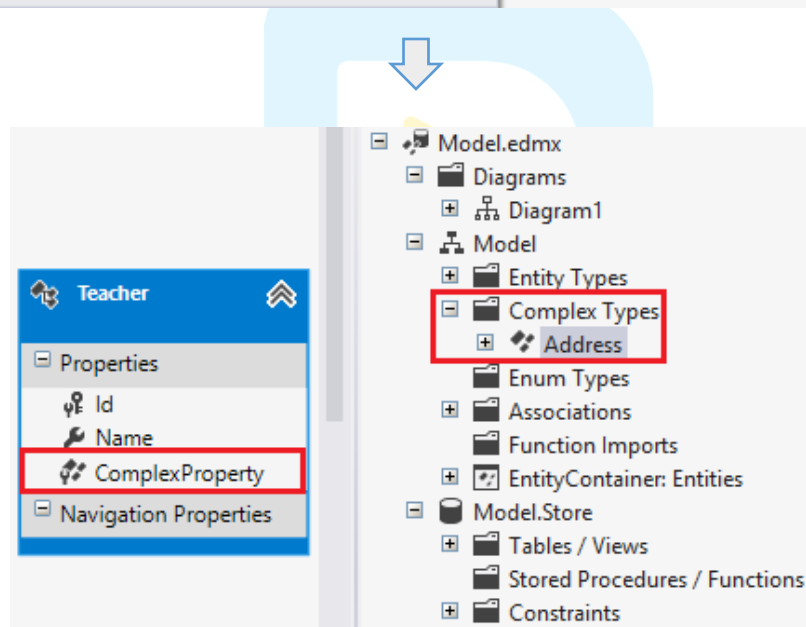
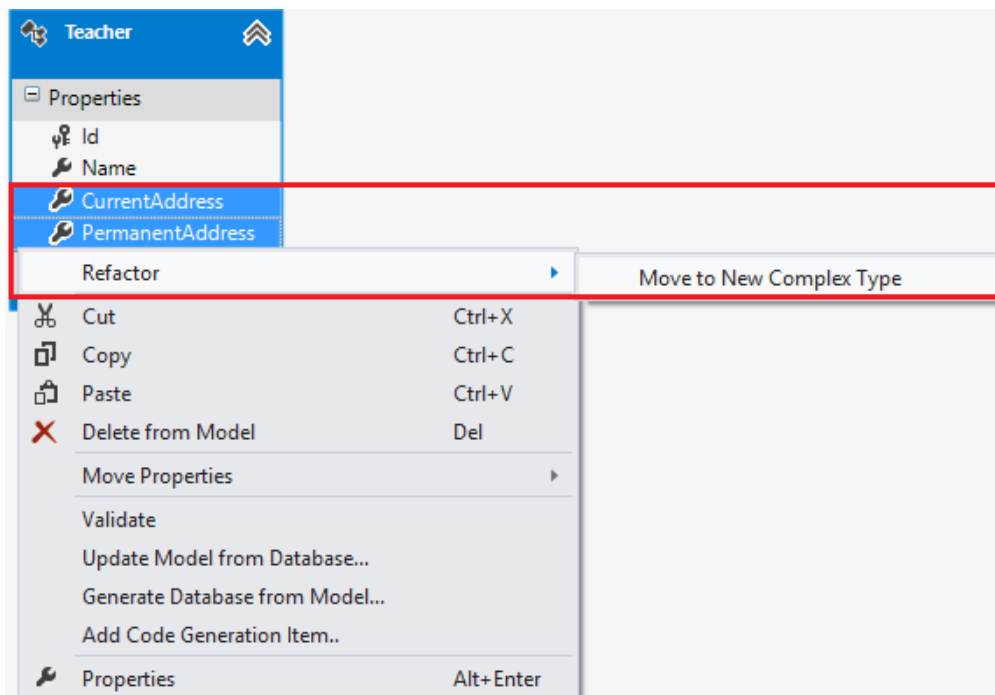
Note - One-to-one relation in EF4+, always uses Foreign Key association and many-to-many relation always uses independent association.

Q11. What is Complex Type?

Ans. Complex type specifies a data type that represents a set of properties which are common between more than two entities.



For example, in the above diagram both the entities- Student and Teacher have some common properties (CurrentAddress and Permanent Address). Hence these properties can be refactored into a complex type *Address* as given below.



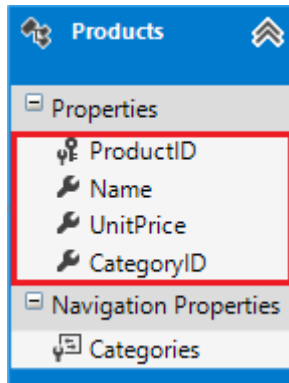
Key points about Complex Type

1. The name should be unique.
2. Complex types cannot exist without parent entities or other complex types.
3. Can't implement inheritance with complex types.
4. A property of a complex type can be another complex type.
5. Can't have association and navigation properties.

Note - Complex types can also be mapped to the database function or stored procedure which return complex data from multiple database tables.

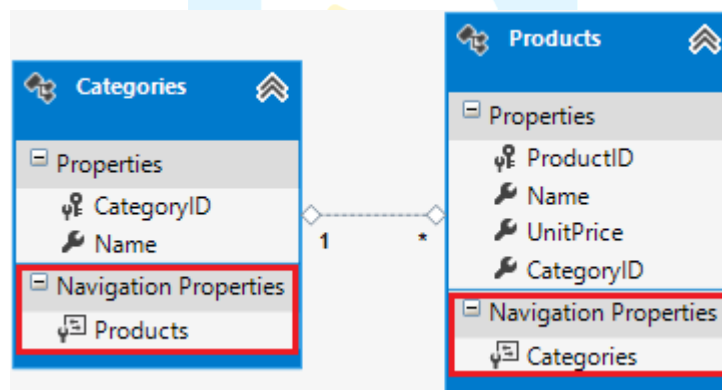
Q12. What is scalar property?

Ans. A scalar property is a property of an entity type that contains the value. Normally, a scalar property is mapped to a database field. For example, Products entity has three scalar properties – ProductID, Name, UnitPrice and CategoryID. Primary Key and Foreign Key are also scalar properties.



Q13. What is navigation property?

Ans. A navigation property is the property of an entity type that allows navigating from one entity to the other entity. This property doesn't carry data. The navigation properties are automatically created from the primary and foreign key references.



The above model contains two entities- Categories and Products that participate in a one-to-many relationship. Both entities have navigation properties. Products are depend on the entity and have the CategoryID foreign key property defined.

Q14. How to define a navigation property using EF Code First?

Ans. The following code shows the way how to create a navigation property using Code First.

```
public partial class Categories
{
    public Categories()
    {
        this.Products = new HashSet<Products>();
    }
    public int CategoryID { get; set; } //Primary Key
}
```



```

    public string Name { get; set; }
    public virtual ICollection<Products> Products { get; set; } //Navigation Property
}
public partial class Products
{
    public int ProductID { get; set; } //Primary Key
    public string Name { get; set; }
    public double UnitPrice { get; set; }
    public int CategoryID { get; set; } //Foreign Key
    public virtual Categories Categories { get; set; } //Navigation Property
}

```

Q15. What are the various Entity States in EF?

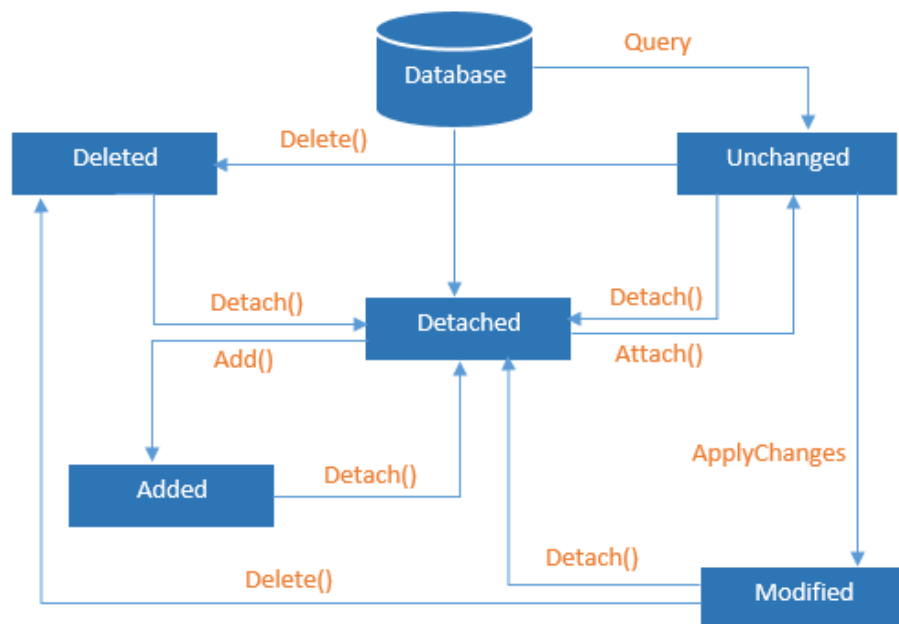
Ans. Each and every entity has a state during its lifecycle which is defined by an enum (EntityType) that have the following values:

1. Added
2. Modified
3. Deleted
4. Unchanged
5. Detached

Q16. Explain Entity Life Cycle?

Ans. An entity has a state during its lifecycle which is managed by the EF itself. An entity life cycle can be understanding with the help of the following diagram:

1. **Detached** – In this state an entity exists but it's not being tracked. An entity lives in this state when it has been created or before adding to the object context. An entity can be in this state when you remove it from the object context by calling the detach() method.



2. **Unchanged** – An entity in this state means that entity is unmodified since it was attached to the context or when the last *SaveChanges()* method was called.
3. **Added** — The entity in this state means that the entity is ready to add but the *SaveChanges()* method has not been called yet. After the changes are saved, the object state changes to unchanged.
4. **Deleted**— The entity in this state means that entity is marked for deletion. After the changes are saved, the object state changes to Detached.
5. **Modified** — The entity in this state means that the entity is ready for the modification but the *SaveChanges()* method has not been called yet. After the changes are saved, the object state changes to unchanged.

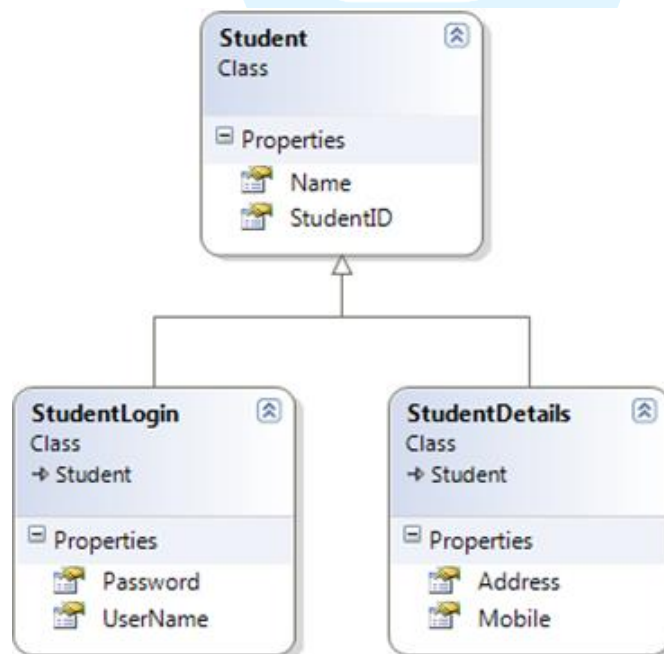
Q17. What are different types of inheritance in Entity Framework?

Ans. Inheritance in the Entity Framework is similar to inheritance for classes in C#. In Entity Framework, you can map an inheritance hierarchy to single or multiple database tables based on your requirements. EF supports three types of inheritances:


1. Table-per-Hierarchy (TPH)
2. Table-per-Type (TPT)
3. Table-per-Concrete-Type (TPC)


Q18. What is TPH inheritance in EF?

Ans. The TPH inheritance states that all entities, in a hierarchy of entities, are mapped to a single table in storage schema. It means, there is only one table in the database and different Entity types in the Entity model that inherits from a base Entity are mapped to that table.



Entity Framework: TPH Inheritance



| | Column Name | Data Type | Nullable |
|---|-------------|---------------|----------|
|  | StudentID | int | No |
| | Name | nvarchar(MAX) | No |
| | Address | nvarchar(MAX) | Yes |
| | Mobile | nvarchar(MAX) | Yes |
| | UserName | nvarchar(MAX) | Yes |
| | Password | nvarchar(MAX) | Yes |
| | Type | nvarchar(128) | Yes |

C# Implementation Code for TPH

```
public class Student
{
    public int StudentID { get; set; }
    public string Name { get; set; }
}

public class StudentDetails: Student
{
    public string Address { get; set; }
    public string Mobile { get; set; }
}

public class StudentLogin : Student
{
    public string UserName { get; set; }
    public string Password { get; set; }
}
```

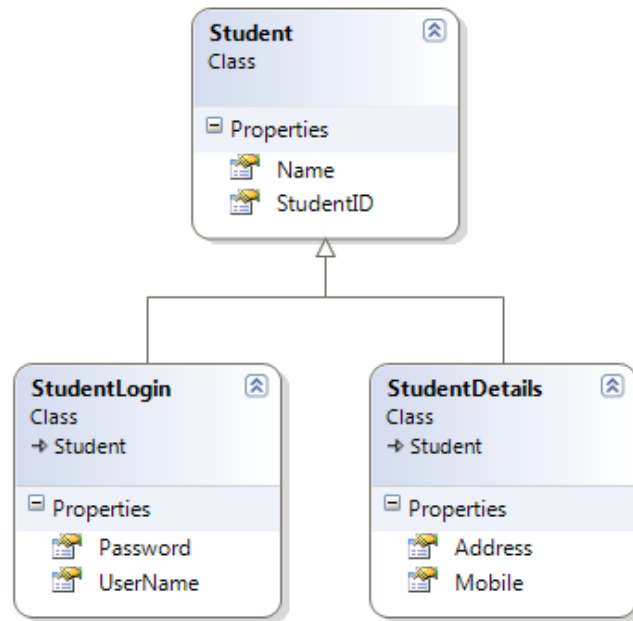
Entity Framework Code First Mapping for TPH:

```
modelBuilder.Entity<Student>()
    .Map<StudentDetails>(m =>
m.Requires("Type").HasValue("StudentDetails"))
    .Map<StudentLogin>(m => m.Requires("Type").HasValue("StudentLogin"));
```

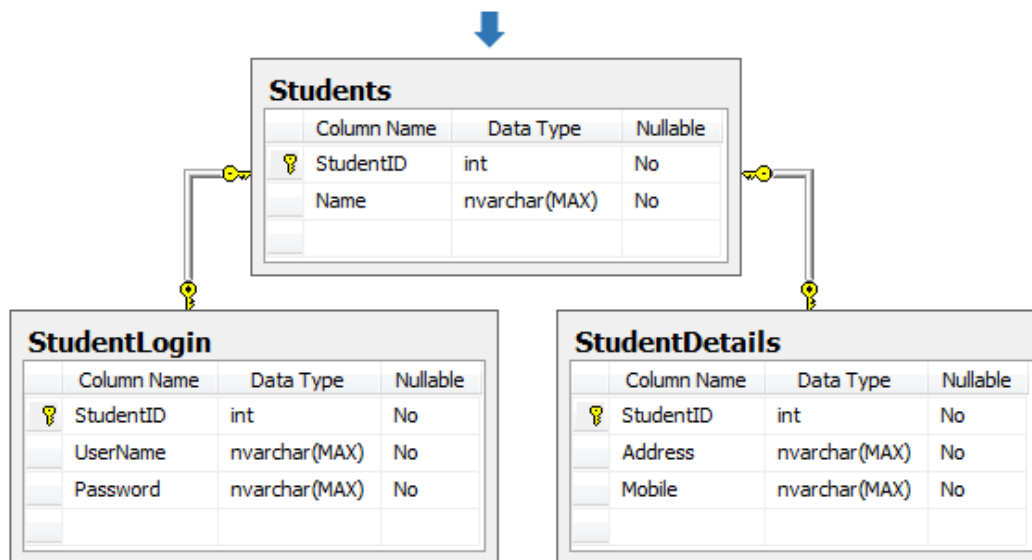
Note - By default, Entity Framework supports TPH inheritance, if you don't define any mapping details for your inheritance hierarchy.

Q19. What is TPT inheritance in EF?

Ans. The TPT inheritance states that each entity in the hierarchy of entities is mapped to a separate table in storage schema. It means, there is a separate table in the database to maintain data for each Entity Type.



Entity Framework: TPT Inheritance



C# Implementation Code for TPT

```

public class Student
{
    public int StudentID { get; set; }
    public string Name { get; set; }
}

public class StudentDetails: Student
{
    public string Address { get; set; }
    public string Mobile { get; set; }
}
  
```

```
public class StudentLogin : Student
{
    public string UserName { get; set; }
    public string Password { get; set; }
}
```

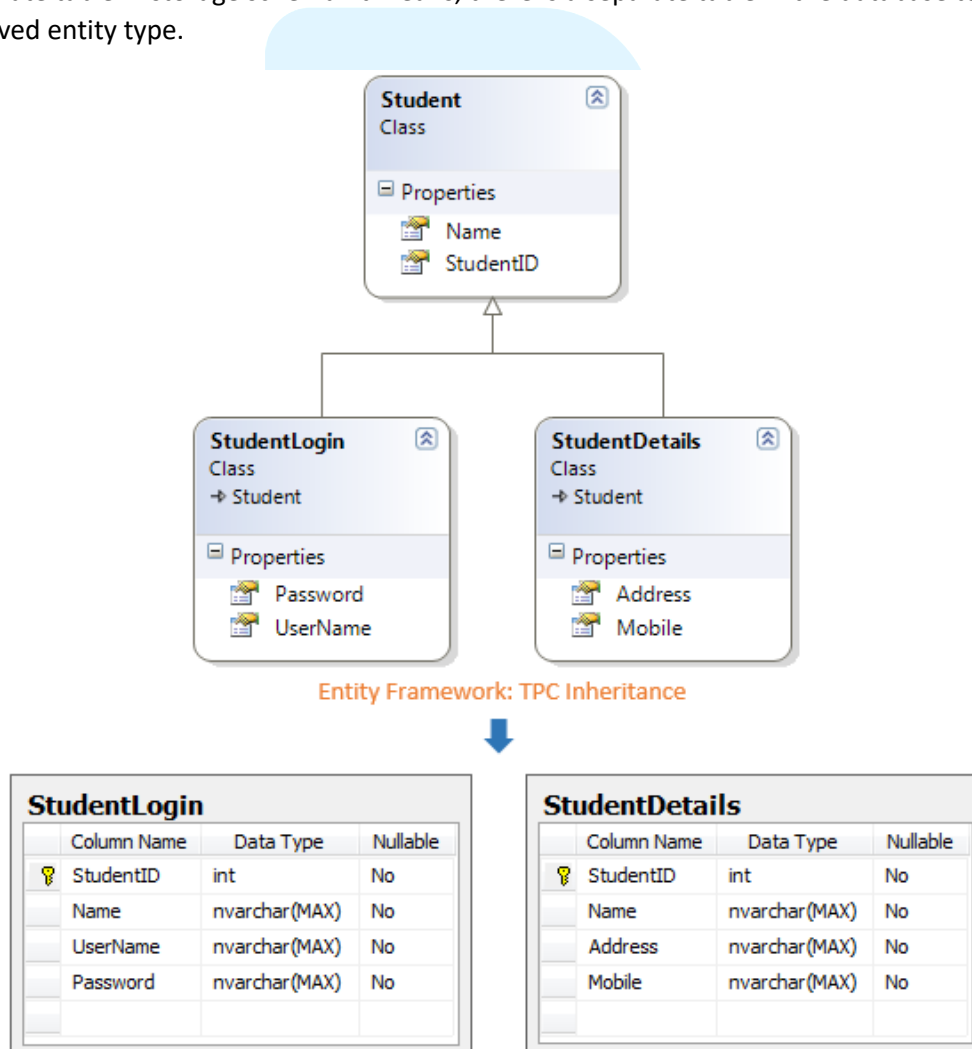
Entity Framework Code First Mapping for TPT:

```
modelBuilder.Entity<Student>().ToTable("Student");
modelBuilder.Entity<StudentLogin>().ToTable("StudentLogin");
modelBuilder.Entity<StudentDetails>().ToTable("StudentDetails");
```

Note- TPH inheritance patterns generally deliver better performance in the Entity Framework than TPT inheritance patterns, because TPT patterns can result in complex join queries.

Q20. What is TPC inheritance in EF?

Ans. The TPC inheritance states that each derived entity (not base entity) in the hierarchy of entities is mapped to a separate table in storage schema. It means, there is a separate table in the database to maintain data for each derived entity type.



This inheritance occurs when we have two tables with overlapping fields in the database like as a table and its history table that has all historical data which is transferred from the main table to it.

C# Implementation Code for TPC:

```
public abstract class Student
{
    public int StudentID { get; set; }
    public string Name { get; set; }
}

public class StudentDetails: Student
{
    public string Address { get; set; }
    public string Mobile { get; set; }
}

public class StudentLogin : Student
{
    public string UserName { get; set; }
    public string Password { get; set; }
}
```

Entity Framework Code First Mapping for TPC:

```
modelBuilder.Entity<StudentDetails>().Map(m =>
{
    m.MapInheritedProperties();
    m.ToTable("StudentDetails");
});

modelBuilder.Entity<StudentLogin>().Map(m =>
{
    m.MapInheritedProperties();
    m.ToTable("StudentLogin");
});
```

Note - The TPC inheritance is very rare but you should be aware of it and when it is needed.

Q21. What is DbContext API and how it works?

Ans. DbContext API is a new API introduced with EF4.1 which provides a more productive and flexible way of working with the Entity Framework. This API can be used with the EF Code First, Database First, and Model First approaches.

The DbContext API is responsible for populating objects with data from a database, change tracking and persisting data to the database.

Q22. What is DLL for DbContext API?

Ans. EntityFramework.dll

Q23. What isObjectContext API and how it works?

Ans. ObjectContext API was used with a previous version of Entity Framework (before EF4.1) which provides facilities for querying and working with entity data as objects. This API can be used with the EF Code First, Database First, and Model First approaches.

The class ObjectContext is responsible for populating objects with data from a database, change tracking, and persisting data to the database.

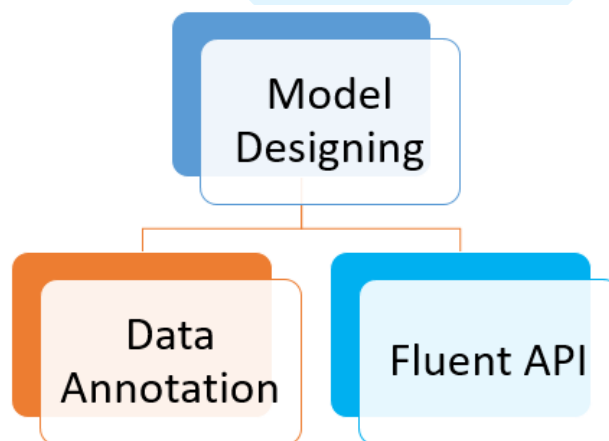
Q24. What are the differences between ObjectContext API and DbContext API?

Ans. DbContext is a lightweight version of the ObjectContext API which builds on the top of ObjectContext API. It also provides all of the same POCO support as the ObjectContext. DbContext also wraps some of the more complex logic of ObjectContext into simpler methods & properties and makes it easier to execute the most common coding tasks in the EF.

With EF5.0 and EF5+, EF team recommend to use DbContext with Code First, Database First, and Model First approach. Before EF5.0, DbContext is the default context to use with Code First classes, but the ObjectContext is the default with **Database First** and **Model First**.

Q25. What are various approaches in Code First for model designing?

Ans. In Entity Framework **Code First** approach, our POCO classes are mapped to the database objects using a set of conventions defined in Entity Framework. If you do not want to follow these conventions while defining your POCO classes, or you **want to change** the way the **conventions** work then you can use the **fluent API** or **data annotations** to configure and to map your POCO classes to the database tables. There are two approaches, which you can use to define the model in EF Code First:



```
[Key]
0 references
public int EmployeeID { get; set; }
[Required]
[Column(TypeName = "varchar")]
[StringLength(200)]
0 references
public string Name { get; set; }
```

```
builder.Entity<Employee>().HasKey(e => e.EmployeeId);
builder.Entity<Employee>().Property(e => e.Name)
    .IsRequired()
    .IsUnicode(false)
    .HasMaxLength(200);
```

2

Q26. What is fluent API and how it is different from data annotations?

Ans. The fluent API is a more **advanced** way of specifying the **model configuration** that covers everything that data annotations can do in addition to some more advanced configuration **not possible** with data annotations. **Data annotations** and the **fluent API** can be used **together**.

Fluent API that is **associated** with the DbContext and can take advantage of features that already exist in .NET 4.0 and the **IValidatableObject**. This integration is a great benefit to developers.

Q27. Which method is required to configure fluent API?

Ans. To configure the fluent API, you need to override the **OnModelCreating** method in DbContext.

Q28. What C# Datatype is mapped with which Datatype in SQL Server?

Ans. The following table having the list of C# Datatype mapping to the corresponding SQL Server Datatype:

| C# Data Type | Default Mapped SQL Server Data Type |
|--------------|-------------------------------------|
| int | int |
| string | nvarchar(Max) |
| decimal | decimal(18,2) |
| float | real |
| byte[] | varbinary(Max) |
| datetime | datetime |
| bool | bit |
| byte | tinyint |
| short | smallint |
| long | bigint |
| double | float |
| char | No mapping |
| sbyte | No mapping |
| object | No mapping |

Q29. Give some example to define mapping using fluent API?

Ans. There are some examples for configuring fluent API-

1. Configuring a primary key.

```
modelBuilder.Entity<Employee>().HasKey(t => t.EmployeeId);
```

2. Specifying the maximum length on a property

```
modelBuilder.Entity<Employee>().Property(t => t.Name).HasMaxLength(50);
```


3. Configuring the property to be required

```
modelBuilder.Entity<Employee>().Property(t => t.Name).IsRequired();
```

4. Mapping a CLR property to a specific column in the Database

```
modelBuilder.Entity<Employee>().Property(t => t.Name).HasColumnName("Name");
```

Q30. What default relationship convention EF6 supports?

Ans. By default, EF6 supports the One-to-Many (1:m) relationship by using the navigation property. There is no default support for One-to-One and Many-to-Many relationships. But You can configure them by using Fluent API or DataAnnotation syntax.

Q31. How to define mapping using data annotation in Entity Framework?

Ans. Below is the example for configuring data annotation in entity framework code first-

```
[Table("Employees")]
public class Employee
{
    [key]
    public int EmployeeID { get; set; }

    [Required]
    [MaxLength(200)]
    public string Name { get; set; }

    public int Country { get; set; }

    [MaxLength(500)]
    public string Address { get; set; }
}
```

Q32. How to define One-to-Zero-or-One (1:0 or 1:1) mapping in Entity Framework?

Ans. To define 1:0 or 1:1 mapping, let's take the following example:

```
public class Employee
{
    public int EmployeeId { get; set; }
    public string Name { get; set; }
}

public class Address
{
    public int AddressId { get; set; }
    public string Address1 { get; set; }
    public string Address2 { get; set; }
    public string City { get; set; }
    public int Zipcode { get; set; }
    public string State { get; set; }
}
```

```

    public string Country { get; set; }
}

```

Let's define mapping using Data annotation approach:

```

public class Employee
{
    public int EmployeeId { get; set; }
    public string Name { get; set; }

    public virtual Address Address { get; set; } //navigation property
}

public class Address
{
    [ForeignKey("Employee")] //foreign key
    public int AddressId { get; set; }
    public string Address1 { get; set; }
    public string Address2 { get; set; }
    public string City { get; set; }
    public int Zipcode { get; set; }
    public string State { get; set; }
    public string Country { get; set; }

    public virtual Employee Employee { get; set; } //navigation property
}

```

Let's define mapping using Fluent API approach:

```

modelBuilder.Entity<Employee>()
    .HasOptional(e => e.Address) // Mark Address property optional in Employee entity
    .WithRequired(ad => ad.Employee); // mark Employee property as required in Address
entity. Cannot save Address without Employee

```

Q33. How to define One-to-Many (1:m) mapping in Entity Framework?

Ans. To define 1:m mapping, let's take the following example:

```

public class Category
{
    public int CategoryId { get; set; }
    public string Name { get; set; }
}

public class Product
{
    public int ProductId { get; set; }
    public string Name { get; set; }
    public string UnitPrice { get; set; }
}

```

Let's define mapping using Data annotation approach:

```

public class Category
{

```

```

    public int CategoryId { get; set; }
    public string Name { get; set; }

    public virtual ICollection<Product> Products { get; set; }
}

public class Product
{
    public int ProductId { get; set; }
    public string Name { get; set; }
    public string UnitPrice { get; set; }
    public virtual Category Category { get; set; }
}

```

Let's define mapping using Fluent API approach:

```

modelBuilder.Entity<Product>()
    .HasRequired(p => p.Category)
    .WithMany(c => c.Products);

```

Q34. How to define Many-to-Many (m:m) mapping in Entity Framework?

Ans. To define m:m mapping, let's take the following example:

```

public class Category
{
    public int CategoryId { get; set; }
    public string Name { get; set; }
}

public class Product
{
    public int ProductId { get; set; }
    public string Name { get; set; }
    public string UnitPrice { get; set; }
}

```

Let's define mapping using Data annotation approach:

```

public class Category
{
    public int CategoryId { get; set; }
    public string Name { get; set; }

    public virtual ICollection<Product> Products { get; set; }
}

public class Product
{
    public int ProductId { get; set; }
    public string Name { get; set; }
    public string UnitPrice { get; set; }
    public virtual ICollection<Category> Categories { get; set; }
}

```

Let's define mapping using Fluent API approach:

```
modelBuilder.Entity<Product>()
    .HasRequired(p => p.Categories)
    .WithMany(c => c.Products)
    .Map(pc =>
    {
        pc.MapLeftKey("ProductId");
        pc.MapRightKey("CategoryId");
        pc.ToTable("ProductCategories");
    });
```

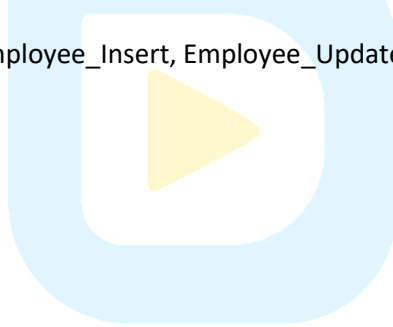
Q35. How to define CUD Operations using Stored Procedures in EF 6?

Ans. From EF 6, you can create and use stored procedures for insert, update, and delete operations on a single database table. EF6 provides `MapToStoredProcedures()` method for creating procedures to perform insert, update, and delete operations on an entity.

The given example maps an Employee entity with the default stored procedures in the database for CUD operations:

```
modelBuilder.Entity<Employee>()
    .MapToStoredProcedures();
```

EF6 API will create three procedures `Employee_Insert`, `Employee_Update` and `Employee_Delete` for the above Employee entity.



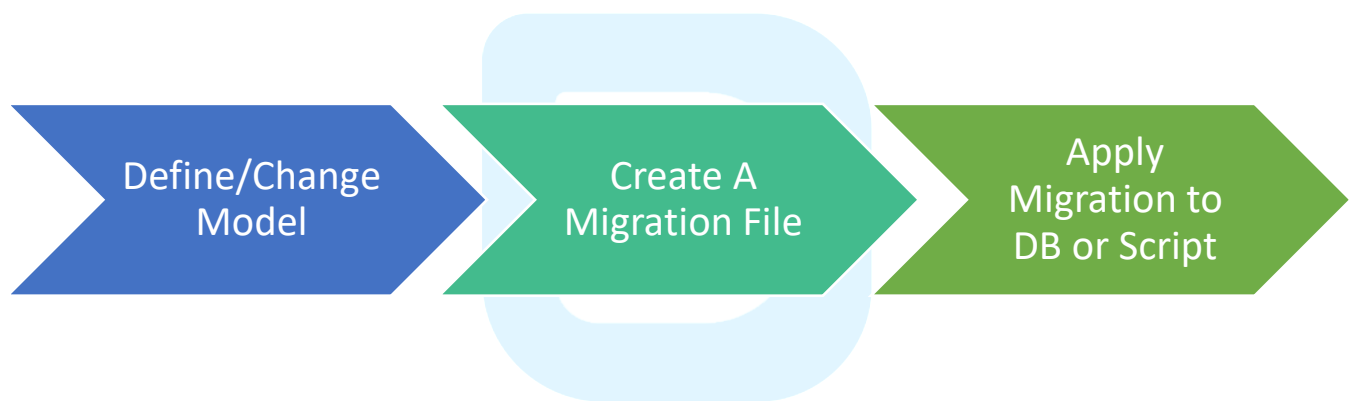
4

Migrations

Q1. What is Code First Migrations in Entity Framework?

Ans. Code-First approach allows you to define model classes as per the Domain requirements via POCOs. Hence, you have complete control over the classes being written or Implemented.

Code First Migrations allow you to create a new database or to update the existing database based on your model classes by using Package Manager Console exist within Visual Studio.



Q2. What are the various Code First Database initializers?

Ans. When you follow the EF code first approach, then you have three options for initializing database:

1. **CreateDatabaseIfNotExists**- This is the default database initializer class used by Code First. This class creates a database only if it doesn't exist. This initializer helps you to avoid any accidental deletion of the database.
2. **DropCreateDatabaseWhenModelChanges**- This database initializer class creates a new database if it doesn't exist already or if a database is already present but there is a mismatch between the model classes and table schema then it drops the database and re-creates it. This initializer is useful during the starting phase of development and testing when models are changing often and there is no concern about the existing database records.
3. **DropCreateDatabaseAlways**- This database initializer class always drop and creates a new database, whether it is already present or not with every run of the application. This initializer is useful during testing when you want to run the application with a fresh set of data.

Q3. Why use Database Migrations?

Ans. The above three databases initializing approach become fails when you add new the model classes and you want to update the existing database without any deletion or change. To achieve this, you need to use EF code first migrations which allow you to update the existing database with new model classes changes and your existing database remains same with your database records.

Q4. Can you create custom database Initializers in Entity Framework Code First?

Ans. Yes, you can also create custom database initializers by implementing the `IDatabaseInitializer` interface.

Q5. Explain step by step Code First Migrations in Entity Framework?

Ans. Suppose you have the following two entities within your application.

```
public class Category
{
    public int CategoryID { get; set; }
    public string Name { get; set; }
}

public class Product
{
    public int ProductID { get; set; }
    public string Name { get; set; }
    public decimal Price { get; set; }
    public int CatID { get; set; }

    public virtual Category Category { get; set; }
}
```

DataContext class and Fluent API mapping details for above two entities are given below –

```
public class DataContext : DbContext
{
    public DataContext():base("DefaultConnection"){ }

    protected override void OnModelCreating(DbModelBuilder modelBuilder)
    {
        modelBuilder.Entity<Category>().HasKey(c => c.CategoryID);
        modelBuilder.Entity<Category>().Property(p =>
p.Name).HasColumnType("VARCHAR").IsRequired().HasMaxLength(50);

        modelBuilder.Entity<Product>().HasKey(p => p.ProductID);
        modelBuilder.Entity<Product>().Property(p =>
p.Name).HasColumnType("VARCHAR").IsRequired().HasMaxLength(50);
        modelBuilder.Entity<Product>().Property(p =>
p.Price).HasColumnType("DECIMAL").IsRequired();

        modelBuilder.Entity<Product>().HasRequired(m =>
m.Category).WithMany().HasForeignKey(c => c.CatID);
    }
}
```

```

    }
    public DbSet<Product> Product { get; set; }
    public DbSet<Category> Category { get; set; }
}

```

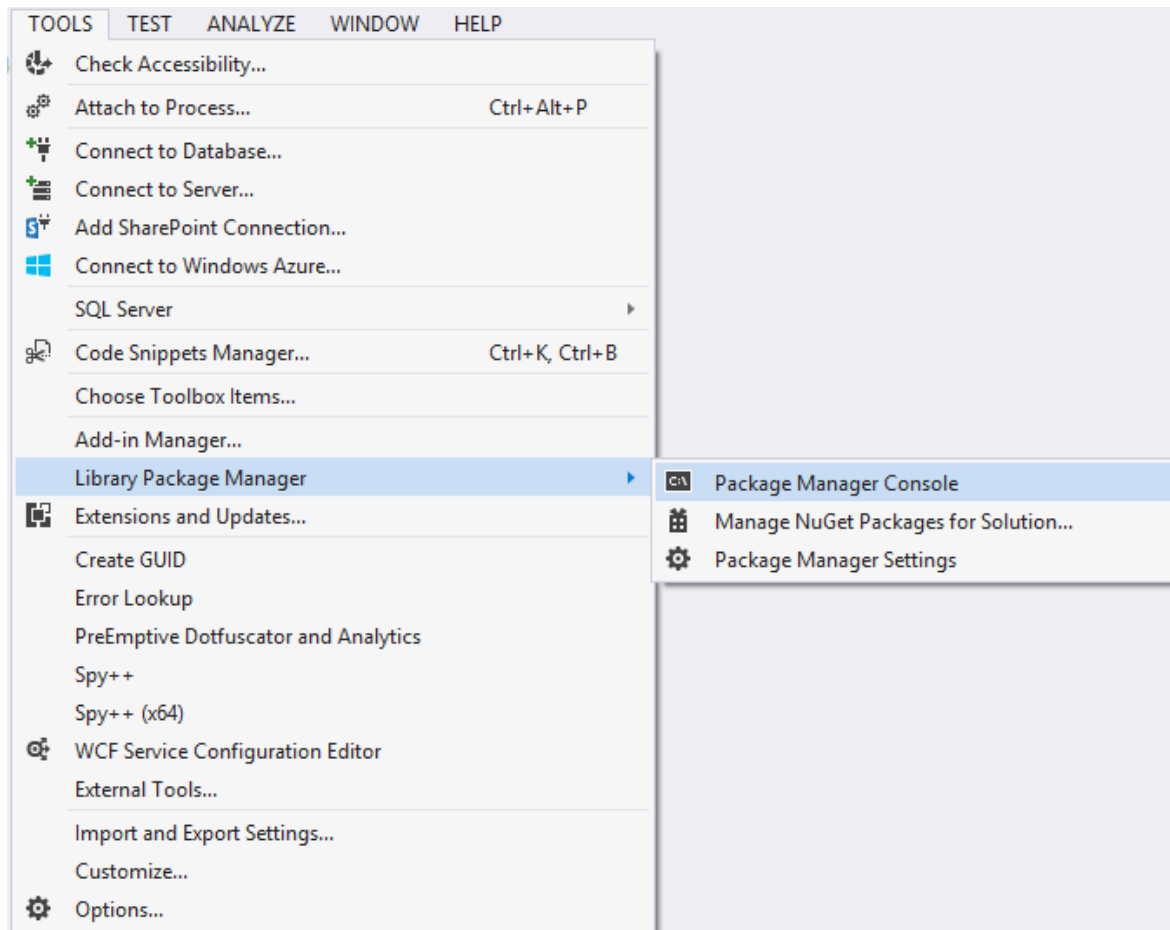
And, your database connection string is given below-

```

<connectionStrings>
  <add name="DefaultConnection" providerName="System.Data.SqlClient"
  connectionString="data source=SHAIENDRA\SQLEXPRESS;initial catalog=EFCodeFirst;persist
  security info=True;user id=sa;password=mypassword; App=EntityFramework" />
</connectionStrings>

```

To create the database for these two entities within your application, go to the Package Manager Console option as shown below:

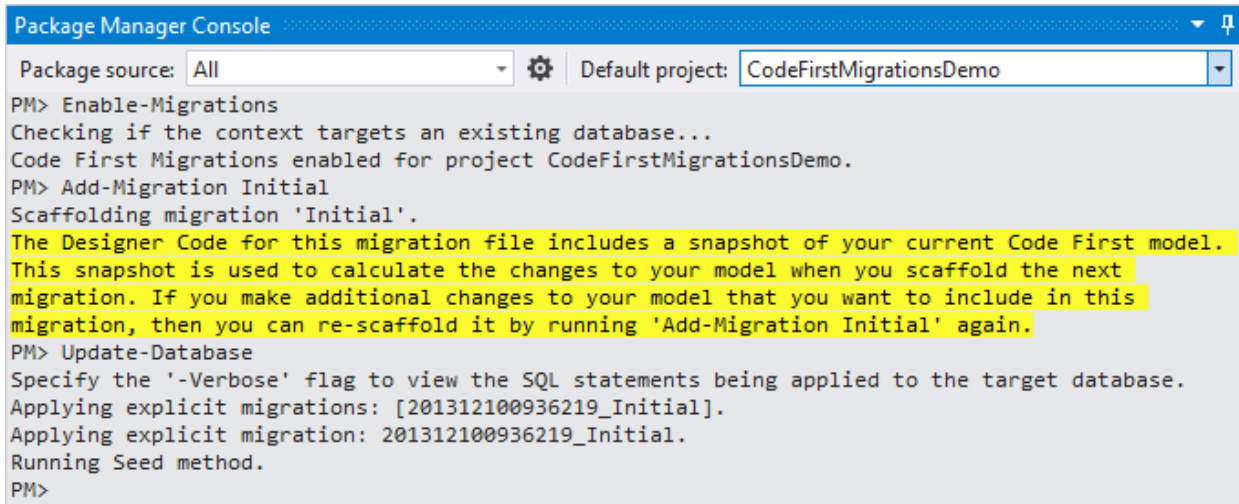


Creating New Database:

Run the following command to configure migrations within your project and for creating a new database.

1. Enable migrations: *Enable-Migrations*
2. Create migration: *Add-Migration "Migrations Name"*

3. Create upgrade/downgrade script: *Update-Database*

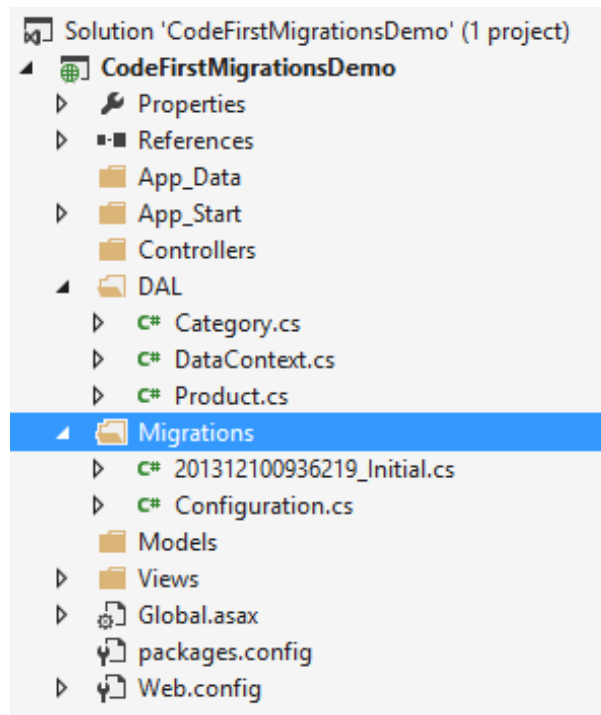


```
Package Manager Console
Package source: All
Default project: CodeFirstMigrationsDemo

PM> Enable-Migrations
Checking if the context targets an existing database...
Code First Migrations enabled for project CodeFirstMigrationsDemo.
PM> Add-Migration Initial
Scaffolding migration 'Initial'.
The Designer Code for this migration file includes a snapshot of your current Code First model.
This snapshot is used to calculate the changes to your model when you scaffold the next
migration. If you make additional changes to your model that you want to include in this
migration, then you can re-scaffold it by running 'Add-Migration Initial' again.
PM> Update-Database
Specify the '-Verbose' flag to view the SQL statements being applied to the target database.
Applying explicit migrations: [201312100936219_Initial].
Applying explicit migration: 201312100936219_Initial.
Running Seed method.
PM>
```

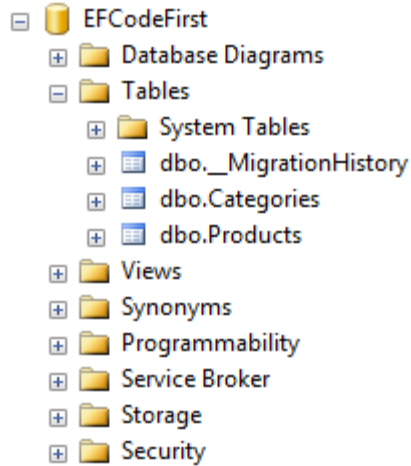
When you run an above-listed command on the Package Manager Console windows one by one then:

1. Firstly, a Migrations folder will be added into your applications having Configuration.cs file for Migrations configuration setting.



2. Secondly, a class file will be created having a name suffix as MigrationsName followed by an underscore and a uniquely generated number. This file will have all the entities to be created in the database.

3. Thirdly, a new database will be created having initial catalog name (initial catalog = YourDBName) as given in your connections string.



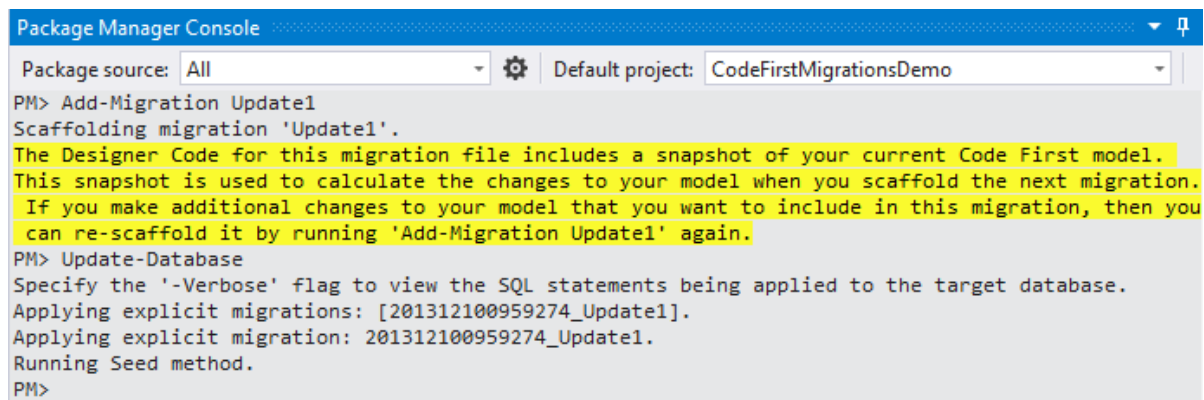
Q6. How to update existing Database using Code First Migrations?

Ans. Suppose you also added one more class named as *Customer* into your data model classes as :

```
public class Customer
{
    public int CustomerID { get; set; }
    public string Name { get; set; }
    public string Address { get; set; }
}
```

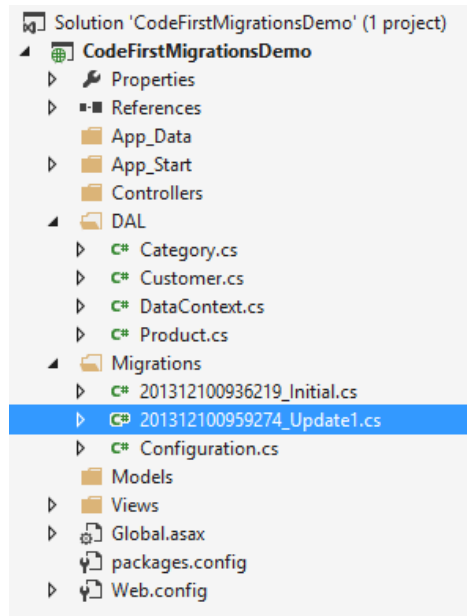
Now for updating the database with new changes run the following commands as given below-

1. Create migration: *Add-Migration Update1*
2. Create upgrade/downgrade script: *Update-Database*

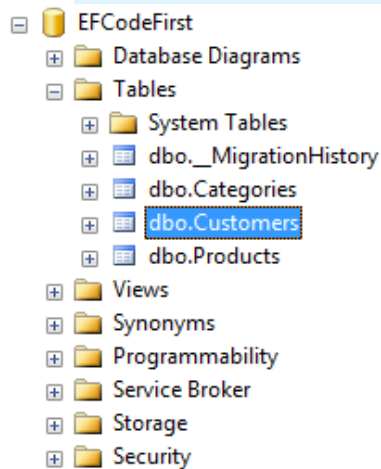


When you run above-listed command on the Package Manager Console windows one by one then

1. Firstly, a class file will be created having a name suffix as MigrationsName followed by an underscore and a uniquely generated number. This file will have all the entities to be created or updated in the database.



2. Secondly, the existing database will be updated with new changes.



Q7. What is Migrations History Table?

Ans. In EF6, Migrations history table (`__MigrationHistory`) is a part of the application database and used by Code First Migrations to store details about migrations applied to a database. This table is created when you apply the first migration to the database. This table stores metadata describing the schema version history of one or more EF Code First models within a given database.

In EF 5, this table was a system table when you use the Microsoft SQL Server database.

Q8. How to Undo/Rollback a Migrations?

Ans. You can Undo/Rollback a specific migrations by using following commands-

- Rollback to a specific migrations: `Update-Database -TargetMigration:"MigrationName"`
- Rollback all migrations: `Update-Database -TargetMigration:0`

Q9. What is automatic migration?

Ans. Entity Framework supports automatic migration so that you don't need to migrate model changes manually. So, when you will run the application, it will be handled by the EF.

Q10. How to disable automatic migration?

Ans. An automatic migration can be disabled by setting the *AutomaticMigrationsEnabled* property false in the configurations file as give below:

```
internal sealed class Configuration : DbMigrationsConfiguration<DAL.DatabaseContext>
{
    public Configuration()
    {
        AutomaticMigrationsEnabled = false;
    }
}
```

Q11. What is Seed Data in EF Code First?

Ans. The Seed() methods are used to insert data into database tables during the database initialization process. This is helpful when you want to insert some test data for your application or some default data into your database master tables.

To seed data into your database, you have to create either a custom DB initializer or you can define it within configuration.cs file as given below:

```
internal sealed class Configuration : DbMigrationsConfiguration<DAL.DatabaseContext>
{
    public Configuration()
    {
        AutomaticMigrationsEnabled = false;
    }

    protected override void Seed(ApplicationCore.DatabaseContext context)
    {
        Role r1 = new Role { Name = "Admin", Description = "Administrator" };
        Role r2 = new Role { Name = "User", Description = "End User" };

        User u1 = new User { Name = "Admin", Username = "admin", Password = "123456",
ContactNo = "9876543210", Address = "Noida" };
        User u2 = new User { Name = "User", Username = "user", Password = "123456",
ContactNo = "9876543210", Address = "Noida" };

        u1.Roles.Add(r1);
        u2.Roles.Add(r2);

        context.Users.Add(u1);
        context.Users.Add(u2);
    }
}
```

5

Querying Database

Q1. What is DbSet?

Ans. DbSet is a typed entity set which is used to perform create, read, update, and delete operations on a particular entity. DbSet is can only be created from a **DbContext** instance. DbSet does not support the Entity SQL methods.

Q2. What is ObjectSet?

Ans. ObjectSet a typed entity set which is used to perform create, read, update, and delete operations on a particular entity. ObjectSet is can only be created from an **ObjectContext** instance. ObjectSet does not support the Entity SQL methods.

Q3. What is EntitySQL?

Ans. Entity SQL is an alternate way to query your database using EF. It is a storage-independent query language and looks similar to SQL.

The Entity SQL is helpful in the following cases:

1. When a query is constructed at runtime.
2. When a query needs to define as a part of the model definition.
3. When you need to return read-only entity data as row sets using an EntityDataReader.

Q4. How EntitySQL is different from T-SQL?

Ans. EntitySQL syntax looks similar to T-SQL but it is more composable than Transact-SQL since every expression can be used anywhere. Query expressions always result in collections of the projected types and can be used anywhere.

Q5. What is SQL injection attack?

Ans. A SQL injection attack is an attack mechanism used by hackers to steal sensitive information from the database of an organization. It is the application layer (means front-end) attack which takes benefit of inappropriate coding of our applications that allows a hacker to insert SQL commands into your code that is using SQL statement.

SQL Injection arises since the fields available for user input allow SQL statements to pass through and query the database directly. SQL Injection issue is a common issue with an ADO.NET Data Services query.

Q6. What is Lazy/Deferred Loading?

Ans. By default, EF supports lazy loading. In lazy loading, related entities (child entities) data is not loaded automatically with its parent entity until it is requested.

For Example:

```
var query = context.Categories.Take(2); // Lazy Loading
foreach (var Category in query)
{
    Console.WriteLine(Category.Name);
    foreach (var Product in Category.Products)
    {
        Console.WriteLine(Product.ProductID);
    }
}
```

The Generated SQL Query will be:

```
SELECT TOP (2)
[c].[CatID] AS [CatID],
[c].[Name] AS [Name],
[c].[CreatedDate] AS [CreatedDate]
FROM [dbo].[Category] AS [c]
GO
-- Region Parameters
DECLARE @EntityKeyValue1 Int = 1
-- EndRegion
SELECT
[Extent1].[ProductID] AS [ProductID],
[Extent1].[Name] AS [Name],
[Extent1].[UnitPrice] AS [UnitPrice],
[Extent1].[CatID] AS [CatID],
[Extent1].[EntryDate] AS [EntryDate],
[Extent1].[ExpiryDate] AS [ExpiryDate]
FROM [dbo].[Product] AS [Extent1]
WHERE [Extent1].[CatID] = @EntityKeyValue1
GO
-- Region Parameters
DECLARE @EntityKeyValue1 Int = 2
-- EndRegion
SELECT
[Extent1].[ProductID] AS [ProductID],
[Extent1].[Name] AS [Name],
[Extent1].[UnitPrice] AS [UnitPrice],
[Extent1].[CatID] AS [CatID],
[Extent1].[EntryDate] AS [EntryDate],
[Extent1].[ExpiryDate] AS [ExpiryDate]
FROM [dbo].[Product] AS [Extent1]
WHERE [Extent1].[CatID] = @EntityKeyValue1
```

In above example, internally 3 SQL queries have been generated it means it calling the database 3 times. First time its loading data from the Categories table and next two times its loading the data from Products table. In this way, the child entity is populated when it is requested.

Q7. How to turn off Lazy/Deferred Loading?

Ans. The lazy loading behavior can be turned off by setting `LazyLoadingEnabled` property of the `ContextOptions` to false. In this case you can fetch only the related entity data with its parent entity in one query.

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

Q8. What is Eager Loading?

Ans. In eager loading, the related entity (child entity) data is loaded automatically with its parent entity. For using Eager loading, you need to mention the property name of entity in the `Include()` method as given below:

```
var query = context.Categories.Include("Products").Take(2); // Eager Loading

foreach (var Category in query)
{
    Console.WriteLine(Category.Name);
    foreach (var Product in Category.Products)
    {
        Console.WriteLine(Product.ProductID);
    }
}
```

The generated SQL would be:

```
SELECT [Project1].[CatID] AS [CatID],
[Project1].[Name] AS [Name],
[Project1].[CreatedDate] AS [CreatedDate],
[Project1].[C1] AS [C1],
[Project1].[ProductID] AS [ProductID],
[Project1].[Name1] AS [Name1],
[Project1].[UnitPrice] AS [UnitPrice],
[Project1].[CatID1] AS [CatID1],
[Project1].[EntryDate] AS [EntryDate],
[Project1].[ExpiryDate] AS [ExpiryDate]
FROM (SELECT
[Limit1].[CatID] AS [CatID],
[Limit1].[Name] AS [Name],
[Limit1].[CreatedDate] AS [CreatedDate],
[Extent2].[ProductID] AS [ProductID],
[Extent2].[Name] AS [Name1],
[Extent2].[UnitPrice] AS [UnitPrice],
[Extent2].[CatID] AS [CatID1],
[Extent2].[EntryDate] AS [EntryDate],
[Extent2].[ExpiryDate] AS [ExpiryDate],
CASE WHEN ([Extent2].[ProductID] IS NULL) THEN CAST(NULL AS int)
ELSE 1 END AS [C1]
FROM (SELECT TOP (2) [c].[CatID] AS [CatID], [c].[Name] AS [Name], [c].[CreatedDate] AS
[CreatedDate]
FROM [dbo].[Category] AS [c] )
AS [Limit1]
LEFT OUTER JOIN [dbo].[Product] AS [Extent2]
ON [Limit1].[CatID] = [Extent2].[CatID]) AS [Project1]
ORDER BY [Project1].[CatID] ASC, [Project1].[C1] ASC
```

In above example, you can see internally only one SQL query has been generated which means it calling the database only one time. So, in one query its loading the database from the Categories and its related table Products.

Q9. How to execute plain SQL in EF6?

Ans. EF6 allows us to execute raw SQL queries to query the database. The following methods are used to execute raw SQL queries:

- DbSet.SqlQuery()
- DbContext.Database.SqlQuery()
- DbContext.Database.ExecuteSqlCommand()

Q10. How to use DbSet.SqlQuery() method?

Ans. This method is used to query a single database table and return an entity instance. Also, the returned entity will be tracked by the context.

```
var user = context.Users
    .SqlQuery("Select * from Users where UserId=@id", new SqlParameter("@id", 1))
    .FirstOrDefault();
```

Q11. How to use DbContext.Database.SqlQuery() method?

Ans. Unlike DbSet.SqlQuery() method, this method can be used to return any value.

```
String userName = context.Users
    .SqlQuery<string>("Select Name from Users where UserId=@id", new
    SqlParameter("@id", 1)).FirstOrDefault();
```

Q12. How to use DbSet.Database.ExecuteSqlCommand () method?

Ans. This method is used to execute database commands: Insert, Update and Delete.

```
using (var context = new DatabaseContext())
{
    int updatedRows = context.Database.ExecuteSqlCommand("Update User set Name = 'Mohan'
    where UserId = @id", new SqlParameter("@id", 1));
}
```

Q13. How to load an Entity explicitly in EF?

Ans. If you have disabled the dynamic proxy or lazy loading, an entity can be loaded explicitly by using the Load() method as given below :

```
context.Entry(product).Reference(p => p.Category).Load(); // loads category for a product
context.Entry(category).Collection(c => c.Products).Load(); // loads product collection
```

Q14. How to query and save your data asynchronously in EF 6?

Ans. Asynchronous code execution has been introduced in .NET 4.5. This feature can be used with EF 6 to execute a query or save database changes asynchronously using *async* and *await* keywords.

Q15. Write an asynchronous query in EF 6?

Ans. A query can be executed in EF6 in asynchronous way as given below:

```
public async Task<User> GetUser()
{
    User user = null;
    using (var context = new DatabaseContext())
    {
        user = await (context.Users.Where(s => s.UserId == 1).FirstOrDefaultAsync<User>());
    }
    return user;
}
```

Q16. Write a query to save data in async way in EF 6?

Ans. An entity can be saved in async way by calling the SaveChangesAsync() method as given below:

```
public async Task<int> SaveUser(User user)
{
    using (var context = new DatabaseContext())
    {
        context.Entry(user).State = EntityState.Modified;
        return await (context.SaveChangesAsync());
    }
}
```

Q17. How does EF support Transaction?

Ans. In EF, whenever you execute SaveChanges() to insert, update or delete data into the database, it wraps that operation in a transaction. So, you need to open a transaction scope explicitly.

Q18. How to use a transaction explicitly in EF?

Ans. EF allows you to open a transaction explicitly as given below:

```
using (var context = new DatabaseContext()){
    using (DbContextTransaction transaction = context.Database.BeginTransaction()) {
        try{
            context.Users.Add(user);
            context.SaveChanges();
            context.Roles.Add(role);
            context.SaveChanges();
            transaction.Commit();
        }
        catch (Exception ex){
            transaction.Rollback();
        }
    }
}
```


References

This book has been written by referring to the following sites:

1. <https://docs.microsoft.com/en-us/ef/ef6/> - Microsoft Docs – EF6
2. <https://stackoverflow.com/questions/tagged/entity-framework> – Stack Overflow - Entity Framework
3. <https://www.dotnettricks.com/learn/entityframework> - Dot Net Tricks – Entity Framework

