Entity Framework Core Interview Questions & Answers

By Monalisa Das

Author and Full stack developer

DotNetTricks

Entity Framework Core Interview Questions and 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 - 7th Mar 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

DotNetTricks

Dedication

My mother Mrs. Anita Das and my father Mr. Shib Narayan Das deserve to have their name on the cover for their support to reach to the position I am now. I would also like to thank www.dotnettricks.com to encourage and believe on me to write this book.

-Monalisa Das

Introduction

What Where Author Qualification to Write This Book

With a mixed experience of being a lecturer and now as a core .Net Developer, **Monalisa Das** holds a strong learning skill over keeping herself updated with the changing technologies in her area as well as other programming languages like Python. Being a full stack developer, Monalisa has hands on over Asp.Net MVC, C#, Vb.Net, Web API, .Net Core, Angular 5, Entity Framework (EF), EF Core, Bootstrap, SQL, Postgres to name a few.

Enormous feedback and support from the previous articles which appreciated by all users inspired me to write interview question and answer for EF Core.

What This Book Is

This book basically will focus on general topics which you can encounter while interviewing over an open source and cross platform ORM tool, Entity Framework Core. This eBook has been written by referring to Microsoft official website at http://www.entityframeworktutorial.net/efcore

What You'll Learn

Being into the software industry, we all are aware of data storage methodologies used by almost all the applications/websites like SQL, Oracle. To enable applications to access the data from data storage, we make use of ORM (Object Relational Mapping) like EF (Entity Framework). This book will help you get an overview of an ORM known as Entity Framework (EF) Core.

Share to Help Others

Hope you will enjoy this book and find it valuable for yourself. The authors are truly delighted about this if you will share this book among others because they want it to reach as many techy people as possible.

Keep Connected with Us

We always post about latest technologies updates on our website so that techy people keep themselves up to date. That's why; we please to suggest you subscribe yourself on www.dotnettricks.com

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

DotNetTricks

About the Author

Monalisa Das - A Full Stack Developer

With a Bachelor's degree in Information Technology, Master's degree in Computer Science, and hands-on experience using .Net languages to create and implement software applications, She is confident and an asset to the organization.

She enjoy being challenged and working on projects that require her to work outside herself comfort and knowledge set, as continuing to learn new languages and developing new techniques are important to herself.

Hand on experience on ASP.NET, MVC, C#, SQL, jQuery, JavaScript, Bootstrap, Entity Framework 6.x, Entity Framework Core, HTML5, CSS3 and few months of hands-on over WebAPI, PostgreSQL, Angular 5, Kendo UI.

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 are 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

DotNetTricks

DotNetTricks

DotNetTricks

1

Introducing EF Core

Q1. What is ADO.Net?

Ans. ADO stands for ActiveX Data Objects. ADO.Net is a database technology within .Net framework which is used to access data or services from the backend (database) from the front-end (application).

Q2. What is ADO.Net Entity Framework?

Ans. Entity Framework, also known as EF, is one of the strong ORM (Object Relational Mapping) framework used to connect the database server aka data access within .Net application to exchange or store data from the database. There are predefined functions to perform the CRUD operation automatically without the need of writing the queries within the application.

Q3. What is Entity Framework 6?

Ans. Entity Framework 6 or EF6 is a tried and tested data access technology which runs on .Net Framework 4.x. It's available as Entity Framework (https://www.nuget.org/packages/EntityFramework/) in NuGet manager.

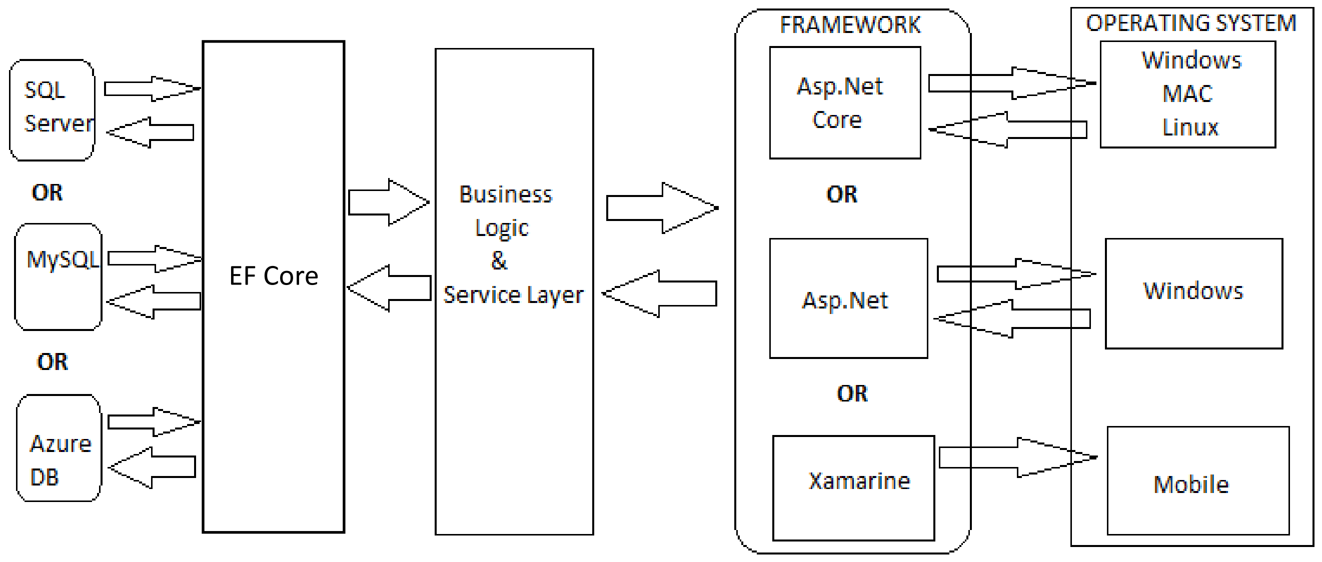
Q4. What is LINQ to SQL?

Ans. LINQ to SQL is an ORM in .Net Framework 3.5 which creates strongly types classes/entity components which represents the tables in your database. These classes are then used to exchange data between the application and the database.

Q5. What is Entity Framework Core?

Ans. Just like Entity Framework 6, Entity Framework Core is an ORM framework used to connect the database server aka data access with .Net Core or .Net 4.6+ applications. Below is a pictorial description for integrating EF Core in your application.

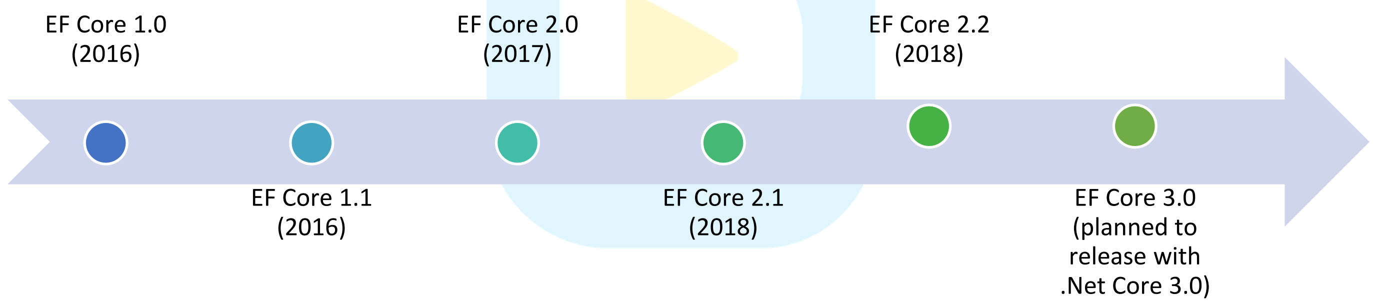
DotNetTricks



The Entity Framework Core block would contain all the entity classes along with EF Core DBContext details.

Q6. Explain the evolution history of EF Core?

Ans. The evolution history of EF Core is given below,



Refer EF Core Roadmap: docs.microsoft.com/en-us/ef/core/what-is-new/roadmap

Q7. Does EF6 work on Linux or MAC?

Ans. Microsoft basically wanted to introduce a database framework which would be open-source along with feasible enough to support all the platforms like Windows, MAC and Linux; unlike Entity Framework which only works with windows only.

Q8. What is cross-platform?

Ans. In the software world, we always want our application/website to work on multiple platforms/OS and devices. To achieve the same, we had to build our application using different technologies like .Net, Android, iOS. Hence sooner there was need of having a technology which can be used across multiple platforms/OS and devices using the same chunk of code. This result in the evolution of .Net Core framework and to support such a framework, EF Core was developed as the ORM.

Q9. Where does EF Core work on?

Ans. Being written from scratch, EF Core supports

1. All operating system: Windows, Linux and MAC.
2. Platforms like Windows phone, Windows store.

Q10. What is the EDMX file? Does Entity Framework Core have an EDMX file?

Ans. EDMX (Entity Data Model XML) file is a graphical representation of the database at the application level. The tables in the database are represented as classes which are used by ADO.Net for data exchange. Entity Framework is an ORM which is represented using EDMX file. Whereas EF Core being developed from scratch, it's built using multiple packages which can be installed as per our application requirement. Hence instead of EDMX file, for EF Core, entity class files are used to represent each table in the database.

Q11. Can you extend EF Core?

Ans. EF Core is open source which code can be found out on GitHub. Now as per your requirement you can customize it and extends its features. Even an open source library or project, you can customize as per your requirement.

Q12. What new features does EF Core support?

Ans. Following are the list of features supported by EF Core but not by EF 6:

1. Relationship configuration
2. CUD (Create, Update, Delete) operations in BATCH mode
3. In-memory provider for testing
4. IoC (Inversion of Control) support
5. Unique constraints
6. Shadow properties
7. Alternate keys
8. Global query filter
9. Field mapping
10. DbContext pooling

Q13. What is In-memory provider?

Ans. When we want to make use of in-memory databases, i.e. IMDB or MMDB, to test a chunk of code without actually connecting to the actual database, EF Core makes use of In-Memory Providers.

Note:
In-memory databases are not relational databases or neither they are mimic of the actual relational database

Q14. What is the NuGet package used for In-memory provider?

Ans. Microsoft.EntityFrameworkCore.InMemory

Q15. What database providers supported by EF Core?

Ans. Following is a list of commonly used database provider.

S. No.	DATABASE PROVIDER	NUGET PACKAGE NAME
1	SQL Server 2008 onwards	Microsoft.EntityFrameworkCore.SqlServer
2	MySQL	MySql.Data.EntityFrameworkCore
3	PostgreSQL	Npgsql.EntityFrameworkCore.PostgreSQL

For more details, you can visit https://docs.microsoft.com/en-us/ef/core/providers/

Q16. What are the advantages EF Core?

Ans. There are following advantages of EF Core:

- Just like .Net Core, EF Core also supports cross-platform. i.e. It can support Microsoft SQL Server, Microsoft SQL Server Compact, PostgreSQL, Oracle to name a few. You can find the entire list in https://docs.microsoft.com/en-us/ef/core/providers/
- It is light-weight and extensible. As EF core has been written from scratch, so it a combination of several small packages. Hence one can use only the packages required in the project and in future can add more packages as per project need.
- No EDMX file is generated. Basically, it follows the Code First Approach which creates database and tables based on migration configuration written in the domain classes or Database First Approach if you already have a database present.
- Supports non-relational databases like Azure table storage and NoSQL.

Q17. What features EF Core does not support as compared to EF6.x?

Ans. Following are the list of features which are not supported by EF Core 2.1.

1. EDMX file or Graphical Visualization of Model/Entity
2. For DB first approach, there is no entity model wizard
3. ObjectContext
4. Migration is not being automated
5. Inheritance for Table per type (TPT) and Table per concrete class (TPC)
6. Many-to-Many without join entity
7. Entity Splitting
8. Spatial Data
9. Stored procedure mapping with DbContext for CUD operation
10. Seed data

For more details over the features, visit https://docs.microsoft.com/en-us/ef/efcore-and-ef6/index

DotNetTricks

Q18. What are the differences between EF 6 and EF Core?

Ans. The differences between EF6 and EF Core are given below:

S.No.	Entity Framework 6	Entity Framework Core
1	Does not support Cross-platform	Supports Cross-platform
2	Contains EDMX file (pictorial presentation of the tables and relationship)	Doesn't contain EDMX file
3	Is available as a package in .Net Framework	Is not available as a package in .Net 4.5+ or .Net Core framework. Hence need to install the packages from NuGet
4	Has a wizard to create and use in application	Need to execute commands in NuGet Package Manager
5	Is not lightweight	Is lightweight and extendable because the user can install packages as per their application need as and when required.
6	Cannot call C# or Vb.Net functions in LINQ	In LINQ queries, we can call C# or Vb.Net functions
7	Doesn't support batch updates	Supports batch update
8	Lazy Loading supported in EF 6	Lazy Loading not supported in EF Core 2.0

DotNetTricks

2

EF Core and Database

Q1. What is ObjectContext?

Ans. It is a part of the CORE Entity Framework which allows us to connect to database, query and track the changes by using strongly typed entity classes

Q2. What is difference ObjectContext and DBContext?

Ans. DbContext is just a wrapper around ObjectContext. The DbContext API consists of methods which are easier to use as compared to that of ObjectContext.

Q3. What is Table Per Type (TPT) inheritance?

Ans. Consider two entity classes, Students and Professor having the following properties in their entity classes.

STUDENT			**PROFESSOR**	
ID	**Int**		**ID**	**int**
FirstName	**String**		**FirstName**	**String**
MiddleName	**String**		**MiddleName**	**String**
LastName	**String**		**LastName**	**String**
DOJ	Datetime		DOJ	Datetime
Course	string		Experience	double

The fields marked in orange, are common to both. So, what we can do is, we can create a separate entity class, say, Person, consisting of the above properties and then Student & Professor inheriting from the Person Class. Such an approach is known as Table Per Type (TPT).

Q4. What is Table Per concrete (TPC) inheritance?

Ans. In TPT, the 'ID' column is also being considered in the base class 'Person'. In TPC, each entity class will have their own table and own primary 'ID' key.

Q5. What is Entity Splitting?

Ans. Suppose there are two tables, Employee and EmployeeAddress. Both the tables contain a primary key 'EmployeeId'. In such cases, the entity framework clubs the columns of both the tables and creates a single entity name 'Employee'. As a result, whenever we query the Employee entity, EF automatically creates a join between Employee and EmployeeAddress tables giving you back the desired data. Such a scenario is known as Entity Splitting.

Q6. What is Spatial Data?

Ans. In SQL Server 2008, there were two types of data-types been introduced: Geography and Geometry. The geography type represents data in a round-earth coordinate system whereas geometry represents data in a Euclidean (flat) coordinate system. We can use this data-types if we want to specify any 'Location' property in an entity. Geography and Geometry were introduced in Entity Framework 5 version.

Q7. What is Lazy Loading?

Ans. When you query an entity for the first time, you must have observed that it takes some time to bring data from the database as compared to when you query for second or more times. Basically, what happens is the related data is not loaded until the time it has not been requested at least once. Such kind of loading is known as lazy loading.

Q8. What is Eager Loading?

Ans. Now consider an Entity Student which has a property List<Courses>. So, whenever you query the Student entity, it also brings the Courses details. Such kind of loading is known as Eager Loading.

Q9. What is Seed Data?

Ans. While creating entities for Migration, we need to insert default data from the application into a database for development/testing purpose. This kind of data is known as Seed data.

Q10. What are the packages required to install EF Core in an application?

Ans. As EF Core is not a part of .Net Core or .Net 4.5+ framework installation, we are supposed to manually install packages available in NuGet.

1. EF Core DB provider – This package depends on the database server you are using. For e.g., if you using SQL Server, you need to install `Microsoft.EntityFrameworkCore.SqlServer`

   You can find the entire list in https://docs.microsoft.com/en-us/ef/core/providers/

2. EF Core Tool (Microsoft.EntityFrameworkCore.Tools) – Used to execute EF Core Commands. For instance, while creating a page using scaffolding.

3. Addition to this, if you want to execute EF Core commands in .Net CLI then also install Microsoft.EntityFrameworkCore.Tools.DotNet. Once the installation is finished, you need to edit the .csproj file and add <DotNetCliToolReference> node.

Q11. What is the use of EF Core DB Provider package?

Ans. To access the required database by the application.

Q12. What is the use of EF Core Tool?

Ans. Used during design-time development tasks. They are basically used to manage Migrations and to scaffold a DbContext and entity types by reverse engineering the schema of a database.

3

Database Modelling

Q1. What type of approaches does EF Core support?

Ans. Two types of approaches EF Core support:

1. Code First Approach
2. Database First Approach (Reverse Engineering)

Q2. What is Code First Approach?

Ans. If you want to create a database schema through the Entity classes defined in your Application, such kind of approach is known as Code First Approach.

Q3. How to implement the Code First Approach?

Ans. To Implement the code first approach you have to follow the given steps.

Step 1: Create an Entity.cs. This file should represent the tables as class name and columns as properties in each table. Please make sure the properties should be properly designed with attributes you wish to see in the table. For e.g. primary key, foreign key, required, max length, etc. In other words, properties should have proper data annotations

Step 2: Add connection string in app.config

Step 3: Create DBContextClass.cs file which contains the DBSet of the classes you created in Step 1 and an overridden OnConfiguration method which contains a reference to the connection string created in Step 2.

Step 4: Execute Add-Migration and Update-Migration commands in the Nuget Package Manager.

Q4. What is Database First Approach?

Ans. If you have an existing database and want to create entity classes based on them, such a scenario is termed as Database First Approach.

Q5. How to implement Database First Approach?

Ans. In the NuGet Package manager you need to run a Scaffold-DbContext command having the below syntax:

```
Scaffold-DbContext [-Connection] [-Provider] [-OutputDir] [-Context] [-Schemas>] [-Tables>]
             [-DataAnnotations] [-Force] [-Project] [-StartupProject] [<CommonParameters>]
```

DotNetTricks

For Example, If I am using SQL Server with database name as EFCoreExample and server name MonalisaSQLServer2014, then Scaffold-DbContext command will look like

```
Scaffold-DbContext "Server= MonalisaSQLServer2014; Database= EFCoreExample;
Trusted_Connection=True;" Microsoft.EntityFrameworkCore.SqlServer -OutputDir Models
```

The "–OutputDir Models" creates a Models folder in the application which consists of different classes each representing each table in your database.

Q6. What are DBContext and DBSet in EF Core?

Ans. DBContext represents a session of database connection which is used to perform CRUD operations. Whereas DBSet is used to create a mapping with the tables in the database. It basically converts the LINQ queries to database queries which are evaluated In-Memory and as a result returns the set of result.

Q7. How can you define the model mapping in EF Core?

Ans. There are two types of approaches,

1. Data Annotations
2. Fluent API

Q8. How to configure model using Fluent API?

Ans. The ModelBuilder class in EF Core is responsible for Fluent API. You can override the OnModelCreating method in your context class and use the ModelBuilder to configure your model. Fluent API configures 3 aspects of a model, viz. Model Configuration, Entity Configuration and Property Configuration

Q9. Explain Fluent API with an example?

Ans. Consider an entity Student with a required property Name. This can be achieved in the following way,

```
class DatabaseContextClass : DbContext
{
    public DbSet<Student> Students { get; set; }

    protected override void OnModelCreating(ModelBuilder modelBuilder)
    {
        modelBuilder.Entity<Student>()
            .Property(b => b.Name)
            .IsRequired();
    }

}
```

Q10. How to exclude property in Fluent API?

Ans. To exclude property in Fluent API, we use 'Ignore' method.

```
class YourContextClass : DbContext
{
    public DbSet<Student> Students { get; set; }

    protected override void OnModelCreating(DbModelBuilder modelBuilder)
    {
        modelBuilder.Entity<Student>()
                .Ignore(b => b.Course);
    }
}
```

Q11. How to configure the model using Data Annotation?

Ans. Decorating the model/entity properties with the attributes like required, max length, foreign key and so on. Please Note. If a property has a name ID or prefixed with ID, then EF Core considers it as a primary key.

Q12. Explain Data Annotation with an example?

Ans. Consider an entity Student with a required property Name. This can be achieved in the following way,

```
public class Student
{
    public int StudentId { get; set; }
    [Required]
    public string Name { get; set; }
    public Course CourseList { get; set; }
}
```

Q13. How to exclude property in Data Annotation?

Ans. To exclude property in Data Annotation, 'NotMapped' attribute is used.

```
public class Student
{
    public int StudentId { get; set; }
    [Required]
    public string Name { get; set; }

    [NotMapped]
    public Course CourseList { get; set; }
}
```

Q14. How are relationships defined in EF Core?

Ans. Relationships in EF Core are usually defined as follows,

1. **Dependent entity:** Contains the foreign key property, sometimes referred to as the 'child' of the relationship.
2. **Principal entity:** Contains the primary/alternate key property, sometimes referred to as the 'parent' of the relationship.
3. **Foreign Key:** Contains the value of the entity key it is dependent on.
4. **Principal Key:** Contains the primary/alternate key property that uniquely identifies the data.
5. **Navigation property:** A property defined on the principal and/or dependent entity that contains a reference(s) to the related entity(s).
    - **Collection navigation property:** A navigation property that contains references to many related entities.
    - **Reference navigation property:** A navigation property that holds a reference to a single related entity.
    - **Inverse navigation property:** When discussing a particular navigation property, this term refers to the navigation property on the other end of the relationship.

For more details, you can visit https://docs.microsoft.com/en-us/ef/core/modeling/relationships

Q15. What are the unique constraints?

Ans. To identify data in an entity different from each other, we make use of unique constraints. This can be uniquely identified using 2 factors,

1. Primary Key
2. Alternate Key

Q16. What is a Primary Key?

Ans. A primary key is a key defined within the relational database which can uniquely identify each record. For Eg, in the Student table, 'StudentId' is used to identify each student record.

Q17. How to define primary key using Data Annotation?

Ans. By Convention, if a property has a name ID or prefixed with ID, then EF Core considers it as a primary key. In the case of Data Annotation, we can decorate the property by 'Key' attribute.

```
public class Student
{
    [Key]
    public int StudentId { get; set; }

    [Required]
    public string Name { get; set; }
    [NotMapped]
    public Course CourseList { get; set; }
}
```

Q18. How to define primary key using Fluent API?

Ans. By Convention, if a property has a name ID or prefixed with ID, then EF Core considers it as a primary key. For Fluent API, we use 'HasKey' keyword.

```
class YourContextClass : DbContext
{
    public DbSet<Student> Students { get; set; }

    protected override void OnModelCreating(ModelBuilder modelBuilder)
    {
        modelBuilder.Entity<Student>()
 .HasKey(c => c.StudentId)
            .Property(b => b.Name)
            .IsRequired();
    }
}
```

Q19. What is an Alternate Key?

Ans. A non-primary key which can identify a set of data uniquely is known as an alternate key. For Eg, in the Student table, 'RollNo' can also be used to identify each student record apart from 'StudentId'.

Q20. How to implement Alternate Key using Data Annotation?

Ans. An alternate key cannot be implemented using Data Annotation.

DotNetTricks

Q21. How to implement Alternate Key using Fluent API?

Ans. It can be implemented by using 'HasAlternateKey' method as given below:

```
class YourContextClass : DbContext
{
    public DbSet<Student> Students { get; set; }

    protected override void OnModelCreating(ModelBuilder modelBuilder)
    {
        modelBuilder.Entity<Student>()
            .HasKey(c => c.StudentId)
            .HasAlternateKey(d => d.RollNo)
            .Property(b => b.FirstName)
            .IsRequired()
            .Property(x => x.LastName)
            .IsConcurrencyToken()
            .Property<DateTime>("LastUpdated");
    }
}
```

Q22. What are the types of relationships in EF Core?

Ans. Relationships are distinguished in three types,

1. One to Many
   - ➔ Consider 'Student' and 'SpecialSubject' entity as follows,

```
public class Student
{
    public int StudentId { get; set; }
    public string Name { get; set; }
    public int currentSpecialSubjectId { get; set; }
    public SpecialSubject spl_Subject { get; set; }
}

public class SpecialSubject
{
    public int splSubjectId { get; set; }
    public string SubjectName { get; set; }
    public list<Student> Students { get; set; }
}
```

In the above scenario, one 'SpecialSubject' entity can have multiple 'Students' whereas one 'Student' can be mapped to only one special subject. This makes the 'Student' entity as Principle entity, ie. the 'StudentId' becomes the principle/primary key. In another hand, as the 'Student' entity has a map to 'SpecialSubject' entity via the 'currentSpecialSubjectId' property, thus making 'currentSpecialSubjectId' as the dependant/foreign key.

2. One to One
   - ➔ Consider the 'Student' and 'Address ' entity as follows,

```
public class Student
{
    public int studentId { get; set; }
    public string Name { get; set; }
    public int studentAddressId { get; set; }
    public Address studentAddress { get; set; }
}
```

DotNetTricks

```
public class Address
{
    public int addressId { get; set; }
    public string street { get; set; }
    public string City { get; set; }
    public string State { get; set; }
    public Student student { get; set; }
}
```

As you can see, unlike one-to-many, each 'Address' entity can have only one 'Student' entity and vice-versa, making them a one-to-one relationship.

3. Many to Many
   ➔ Consider 'Subject', 'Book' and 'Author ' entity as follows,

```
public class Subject
{
    public int BId { get; set; }
    public Book book { get; set; }

    public int AId { get; set; }
    public Author author { get; set; }
}

public class Book
{
    public int bookId { get; set; }
    public string Name { get; set; }
    public List<Subject> subjectDetail { get; set; }
}

public class Author
{
    public int authorId { get; set; }
    public string authorName { get; set; }
    public List<Subject> subjectDetail { get; set; }
}
```

So here one Author can write for multiple books/subjects and on the other hand, one book/subject can be written by multiple authors, thus making it a many-to-many relationship.

Q23. How can we implement a one-to-many relationship using Fluent API?

Ans. This can be achieved using HasOne(), WithMany() and HasForeignKey() methods in the OnModelCreating method as follows,

```
protected override void OnModelCreating(ModelBuilder modelBuilder)
{
    modelBuilder.Entity<Student>()
        .HasOne<SpecialSubject>(s => s.spl_Subject)
        .WithMany(g => g.Students)
        .HasForeignKey(s => s.currentSpecialSubjectId);
}
```

So here we start building up the principal entity 'Student' where '.HasOne' would tell the ModelCreating method that the student entity has a dependant entity 'SpecialSubject', which in return has many students using the

‘WithMany’ property and ‘.HasForeginKey’ defines the dependant/foreign key. In our case, it’s the ‘currentSpecialSubjectId’.

Q24. How can we implement a one-to-one relationship using Fluent API?

Ans. This can be achieved using ‘HasForeignKey’ property in the OnModelCreating method as follows,

```
protected override void OnModelCreating(ModelBuilder modelBuilder)
{
    modelBuilder.Entity<Student>()
        .HasOne<Address>(s => s.studentAddress)
        .WithOne(ad => ad.student)
        .HasForeignKey<Address>(ad => ad.studentAddressId);
}
```

So here we start building up the principal entity ‘Student’ where ‘.HasOne’ would tell the ModelCreating method that the student entity has a dependant entity ‘Address’, which in return is mapped to only one student using the ‘WithOne’ methods and ‘.HasForeginKey’ defines the dependant/foreign key. In our case, it’s the ‘studentAddressId’.

Q25. How can we implement a many-to-many relationship using Fluent API?

Ans. Firstly, create a one-to-many relationship for Book and Author with Subject and then create a composite key wrt BookId and AuthorId as below

```
protected override void OnModelCreating(ModelBuilder modelBuilder)
{
    modelBuilder.Entity<Subject>().HasKey(sc => new { sc.BId, sc.AId });

    modelBuilder.Entity<Subject>()
        .HasOne<Book>(sc => sc.book)
        .WithMany(s => s.subjectDetail)
        .HasForeignKey(sc => sc.BId);


    modelBuilder.Entity<Subject>()
        .HasOne<Author>(sc => sc.author)
        .WithMany(s => s.subjectDetail)
        .HasForeignKey(sc => sc.AId);
}
```

Q26. What concurrency token in EF Core?

Ans. If a property in the entity is declared as concurrency then it means that the property is not allowed to be changed while saving the data. If it has been, then EF Core throws an error.

Q27. How to implement concurrency token in Data Annotation?

Ans. Decorate property with ‘ConcurrencyCheck’.

```
public class Student
{
    [Key]
    public int StudentId { get; set; }
    [Required]
    public string FirstName { get; set; }

    [ConcurrencyCheck]
    public string LastName { get; set; }

    [NotMapped]
    public Course CourseList { get; set; }
}
```

Q28. How to implement concurrency token in Fluent API?

Ans. Use 'IsConcurrencyToken' method

```
class YourContextClass : DbContext
{
    public DbSet<Student> Students { get; set; }

    protected override void OnModelCreating(ModelBuilder modelBuilder)
    {
        modelBuilder.Entity<Student>()
            .HasKey(c => c.StudentId)
            .Property(b => b.FirstName)
            .IsRequired()
            .Property(x => x.LastName)
            .IsConcurrencyToken();
    }
}
```

Q29. What is Shadow property?

Ans. The properties which are not a part of entity classes but can be included in the model and also a part of database columns is known as shadow properties.

Q30. How to declare shadow property using Data Annotation?

Ans. We cannot declare shadow property using data annotation

Q31. How to declare shadow property using Fluent API?

Ans. Declare datatype and column name in Property method

```
class YourContextClass : DbContext
{
    public DbSet<Student> Students { get; set; }

    protected override void OnModelCreating(ModelBuilder modelBuilder)
    {
        modelBuilder.Entity<Student>()
            .HasKey(c => c.StudentId)
            .Property(b => b.FirstName)
            .IsRequired()
            .Property(x => x.LastName)
            .IsConcurrencyToken()
            .Property<DateTime>("LastUpdated");
    }
}
```

DotNetTricks

Q32. What is ChangeTracker in EF Core?

Ans. The DBContext in EF Core includes a ChangeTracker class under the namespace Microsoft.EntityFrameworkCore.ChangeTracking. This class is responsible to keep a track over the state of each entity been called from the DBContext instance.

Q33. What are backing fields?

Ans. Instead of writing/reading data from a property, EF Core allows us to write/read data from fields. i.e. a variable is assigned to get/set of the entity property and data can be written/read using the variable for the property. This is known as backing fields.

Q34. How are backing fields identified?

Ans. For a variable to be identified as backing fields, following convention are followed,

- _<camel-cased property name>
- _<property name>
- m_<camel-cased property name>
- m_<property name>

Q35. Is it possible to declare backing fields in Data Annotations?

Ans. No, Data Annotations doesn't support backing fields declaration.

Q36. How can we use Fluent API for using backing fields?

Ans. By using 'HasField' extension method as follows,

```
class MyContext : DbContext
{
    public DbSet<Student> Students { get; set; }

    protected override void OnModelCreating(ModelBuilder modelBuilder)
    {
        modelBuilder.Entity<Student>()
            .Property(b => b.FirstName)
            .HasField("_StudentFirstName");
    }
}
```

DotNetTricks

4
Migrations

Q1. What is EF core Migrations?

Ans. When developing an application, there are chances that the requirement may change. This indirectly may result in making changes in the application entity/database to build in the new requirement. Using Migration, one can change the models at the application level and sync the database for the same.

In EF Core, Migration is enabled by default and can be executed using Package Manager Console or EF Core Command Lines.

Q2. What task can be performed in EF Core migrations?

Ans. EF Core migration contains the following list of task,

1. Create Migration
   - ➔ Generates code to sync the database with the model classes
2. Update Migration
   - ➔ Use to run any pending migration
3. Customize migrated code
   - ➔ Sometimes there is a need to customize or modify the generated code as per our requirement. In that case, we can use customization of the migrated code.
4. Remove Migration
   - ➔ To remove the generated code

6. Revert Migration
   - ➔ Revert the database to its previous state.
7. Generate SQL Scripts
   - ➔ This task is used if we want to update the production database or maybe we want to troubleshoot the migrated code
8. Applying migration at run-time
   - ➔ Used if we want to execute the migration code while the application executes instead of manually executing them through CLI or package manager console.

Q3. How to create a migration in EF Core?

Ans. After creating the initial models, now we need to create the database. This can be done by using the following commands,

```
PACKAGE MANAGER CONSOLE
Add-Migration InitialCreate
```

DotNetTricks

CLI
```
dotnet ef migrations add InitialCreate
```

Once you execute the above line, three files are generated within the project under the Migrations directory:

- 00000000000000_InitialCreate.cs -> This is the main migrations file which Contains the operations necessary to apply the migration (in Up()) and to revert it (in Down()).
- 00000000000000_InitialCreate.Designer.cs -> It's the migrations metadata file which contains information used by EF.
- MyContextModelSnapshot.cs -> This file contains the snapshot of the current model. This helps to determine what changed when adding the next migration.

The timestamp in the filename helps keep them ordered chronologically so you can see the progression of changes.

Q4. How to update a migration in EF Core?

Ans. After create, now we need to apply the migration to the database to generate the schema. It's done as follows,

PACKAGE MANAGER CONSOLE
```
Update-Database
```

CLI
```
dotnet ef database update
```

Q5. How to customize a migrated code in EF Core?

Ans. This is a three-step procedure.

1. Create the migration
2. Make changes in the generated code files
3. Update the database

Q6. In what cases do we need to remove a migration in EF Core?

Ans. Suppose there is a need or we forgot to make changes in the migrated file or model & we need to make the changes before the database schema gets updated. For this purpose, we execute remove migration. This will remove the lastly created migrated files.

PACKAGE MANAGER CONSOLE
```
Remove-Migration
```

CLI
```
dotnet ef migrations remove
```

After executing the command, you can make the changes and create – update the migrations.

Q7. Can we remove a specific migration in EF Core?

Ans. No. Only the lastly created migration can be removed

DotNetTricks

Q8. How to revert a migration in EF Core?

OR

Q9. Is remove migration same as that of revert migration in EF Core?

Ans. Suppose, you have to sync or update the database with one/several migrations but due to some requirement changes you need to roll back the migration(s). In such cases, revert migration is used. For the same, we use the update command and specify the name of the migration until where we want to rollback.

```
PACKAGE MANAGER CONSOLE
Update-Database nameofMigration
```

```
CLI
dotnet ef database update nameofMigration
```

Q10. How to apply migration to a remote database?

Ans. There are situations where we have to apply the migrations to some other databases. For instance, a test database or production database. In such cases, we need to script the migration so that migration is up to the mark.

```
PACKAGE MANAGER CONSOLE
Script-Migration
```

```
CLI
dotnet ef migrations script
```

By default, all the migration listed in the migration folder will be executed but then if we want only specific migration(s) to be applied, we can do it by using '-to' and '-from' properties of the command.

Q11. How to target multiple providers in EF Core migration?

Ans. There are cases where we wish to apply a few initial steps before the migration happens with respect to the database providers. To achieve this, we can override the 'Up' method in the CreateDatabase class as follows,

```
public partial class CreateDatabase: Migration
{
    protected override void Up(MigrationBuilder migrationBuilder)
    {
        if (ActiveProvider == "Microsoft.EntityFrameworkCore.SqlServer")
        {
            // do something SQL Server - specific
        }
        if (ActiveProvider == "Microsoft.EntityFrameworkCore.Sqlite")
        {
            // do something SqLite - specific
        }
    }
}
```

Q12. How to apply migration during runtime?

Ans. If you want the migration to execute during the start of the application, you can use :

```
myDbContext.Database.Migrate();
```

DotNetTricks

5

Querying Database

Q1. How to perform eager loading in EF Core?

Ans. Using 'Include' keyword. The following code will fetch a list of students along with the course they are enrolled in.

```
var students = context.Students
                .Include(x => x.Course)
                .Include(y => y.Addresses)
                .ToList();
```

Q2. What is the difference between Include and ThenInclude?

OR

Q3. What is multiple Level relationship?

Ans. 'Include' is used to fetch related data of an entity whereas 'ThenInclude' is used to fetch related data of the data received from 'Include'. Below code, fetches related courses of a student and then related time of each course.

```
var students = context.Students
                .Include(x => x.Course)
               .ThenInclude(y => y.Time)
                .ToList();
```

This kind of relationship is known as a multiple level relationship.

Q4. What is Tracking in EF Core?

Ans. The queries which return a set of entity data which can be changed within the current entity and save in database is called Tracking. In other words, the entity instances whose changes are tracked in the ChangeTracker is called Tracking.

```
using (var context = new BloggingContext())
{
    var blog = context.Blogs.SingleOrDefault(b => b.BlogId == 1);
    blog.Rating = 5;
    context.SaveChanges();
}
```

DotNetTricks

Q5. What is NoTracking in EF Core?

Ans. The queries which return a set of entity data for the read-only purpose.

```
using (var context = new BloggingContext())
{
    var blogs = context.Blogs
                            .AsNoTracking()
                            .ToList();
}
```

Q6. Can we use Raw SQL Queries in EF Core?

Ans. Yes

Q7. Limitations to using Raw SQL Queries?

Ans. While using Raw SQL Queries in EF Core we need to keep in mind the following points,

1. The SQL query must return data for all properties of the entity or query type.
2. The column names in the result set must match the column names that properties are mapped to.
3. SELECT statements passed to this method should generally be composable: If EF Core needs to evaluate additional query operators on the server (for example, to translate LINQ operators applied after FromSql), the supplied SQL will be treated as a subquery. This means that the SQL passed should not contain any characters or options that are not valid on a subquery, such as:
    a. a trailing semicolon
    b. On SQL Server, a trailing query-level hint (for example, OPTION (HASH JOIN))
    c. On SQL Server, an ORDER BY clause that is not accompanied by TOP 100 PERCENT in the SELECT clause
4. SQL statements other than SELECT are recognized automatically as non-composable. As a consequence, the full results of stored procedures are always returned to the client and any LINQ operators applied after FromSql are evaluated in-memory.

Q8. How can we incorporate Raw SQL query in EF Core?

OR

Q9. How to call a database view in EF Core?

Ans. Using the 'FromSql' extension method.

```
var students = context.Students
                        .FromSql("SELECT * FROM ViewORTableName")
                        . ToList();
```

Q10. Can we execute a stored procedure using FromSql?

Ans. You can execute stored procedure just like the following code:

```
        context.Students
          .FromSql("EXECUTE StoreProcedureName").ToList();
```

Q11. How to call functions in EF Core?

Ans. You can execute function in EF core just like the following code:

```
context.Students
  .FromSql("Select FunctionName AS Value")
  .ToList();
```

Q12. What is the Global Query Filter?

Ans. The filter which is applied to all the LINQ queries related to an entity by default is known as Global Query Filter.

Q13. How to define Global Query Filter?

Ans. Using the ‘HasQueryFilter’ method in the ‘OnModelCreating’.

```
class YourContextClass : DbContext
{
    public DbSet<Student> Students { get; set; }

    protected override void OnModelCreating(ModelBuilder modelBuilder)
    {
        modelBuilder.Entity<Student>().HasQueryFilter(p => !p.IsDeleted);
    }

}
```

Whenever Student entity is queried, it will bring the data which is not deleted.

Q14. Does EF Core support bulk insert/update/delete?

Ans. Yes

Q15. What do you mean by batching of statement?

Ans. When multiple statements, i.e. multiple insert or multiple updates or multiple delete, are executed in one round trip that methodology is known as batching of statements. Consider the below example,

```
using (var c= new SampleDBContext())
{
    Category cat = dataContext.Categories.Where(c => c.CategoryID == 3).First();
    cat.CategoryName = "Accessory";
    c.Categories.Add(new Category() { CategoryID = 4, CategoryName = "Fragnance" });
    c.Categories.Add(new Category() { CategoryID = 5, CategoryName = "Sports" });
    Category catToDelete = dataContext.Categories.Where(c => c.CategoryID == 2).First();
    c.Entry(catToDelete).State = EntityState.Deleted;
    c.SaveChanges();
}
```

You can see we are having three category values. So whenever ‘SaveChanges’ occurs, SQL will generate three different insert queries but executing at a time

Q16. Can we perform insert, update and delete in one batch?

Ans. Yes

Q17. How can you disable batching?

Ans. By limiting 'MaxBatchSize' in OnConfiguring method.

```
protected override void OnConfiguring(DbContextOptionsBuilder optionbuilder)
{
    string conn =
@"Server=YourServerLocation;Database=YourDatabaseName;Trusted_Connection=true;";
    optionbuilder.UseSqlServer(conn, b => b.MaxBatchSize(1));
}
```

So here '1' in the MaxBatchSize tells EF core that it can generate only 1 SQL statement at a time.

Q18. What are the functions used for batch insert, update and delete?

Ans. Following are the functions which are used for batch insert, update and delete:

For Insert	AddRange
For Update	UpdateRange
For Delete	RemoveRange

Q19. How can we enable batching?

Ans. To enable batching in EF Core, we need to install 'EFCore.BulkExtensions' from NuGet package manager.

```
PM > Install-Package EFCore.BulkExtensions -Version 2.3.4
```

Q20. What are the states of an entity in EF Core?

Ans. Every entity always is in either of the one below state,

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

Q21. Which enum is used to call the states of an entity in EF Core?

Ans. Microsoft.EntityFrameworkCore.EntityState

Q22. How can we execute store procedure in EF Core?

Ans. There are 2 ways,

1. DBSet<TEntity>.FromSql()
2. DBContext.Database.ExecuteSqlCommand()

Q23. What are Asynchronous queries in EF Core?

Ans. Usually, when one query executes, the other queries go on hold till the time the current query doesn't finish off executing. To avoid this, we make use of asynchronous queries.

DotNetTricks

Q24. How to save data asynchronously in EF Core?

Ans. The procedure is the same as that of normal 'SaveChanges' method. The only difference is instead of calling 'SaveChanges', we call the 'SaveChangesAsync' method. See the below example.

```
using (var c= new SampleDBContext())
{
    Category cat = dataContext.Categories.Where(c => c.CategoryID == 3).First();
    cat.CategoryName = "Accessory";
    c.Categories.Add(new Category() { CategoryID = 4, CategoryName = "Fragnance" });
    c.Categories.Add(new Category() { CategoryID = 5, CategoryName = "Sports" });
    Category catToDelete = dataContext.Categories.Where(c => c.CategoryID == 2).First();
    c.Entry(catToDelete).State = EntityState.Deleted;
    c.SaveChangesAsync();
}
```

Q25. How to call stored procedure asynchronously?

Ans. Instead of 'ToList', use 'ToListAsync' method

```
context.Students
 .FromSql("EXECUTE StoreProcedureName")
 .ToListAsync();
```

DotNetTricks

References

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

1. https://docs.microsoft.com/en-us/ef/core/ - Microsoft Docs - EF Core
2. https://stackoverflow.com/questions/tagged/entity-framework-core - Stack Overflow - EF Core
3. https://www.dotnettricks.com/learn/ef- Dot Net Tricks - EF