Instructor example

Web University by Harsha Vardhan

What is Entity Framework?

Working with databases can often be rather complicated. You have to manage database connections, convert data from your application to a format the database can understand, and handle many other subtle issues.

The .NET ecosystem has libraries you can use for this, such as ADO.NET. However, it can still be complicated to manually build SQL queries and convert the data from the database into C# classes back and forth.

EF, which stands for Entity Framework, is a library that provides an object-oriented way to access a database. It acts as an object-relational mapper, communicates with the database, and maps database responses to .NET classes and objects.

Entity Framework (EF) Core is a lightweight, open-source, and cross-platform version of the Entity Framework.

Here are the essential differences between the two:

- **Cross-platform:** We can use EF Core in cross-platform apps that target .NET Core. EF 6.x targets .NET Framework, so you're limited to Windows.
- **Performance:** EF Core is fast and lightweight. It significantly outperforms EF 6.x.

"What other libraries or frameworks might you use with ASP.NET Core to build your application, and for what purposes?"

Since most applications need to store data, I'd likely reach for Entity Framework Core or Dapper for data access.

In addition, I'm also a big fan of FluentValidation to make validating user input easier to understand.

I'm pretty comfortable in writing unit tests, integration tests with xUnit, FluentAssertions and Moq. xUnit is quite advanced and extensible. FluentAssertions makes it easy to write human-readable & close-to-natural-language assert statements. Moq helps me to mock services.

What is SQL injection attack?

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.

How to handle SQL injection attacks in Entity Framework?

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.

What are POCO classes?

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.

Eg:

```
public class Model
{
  public type PropertyName1 { get; set; }
  public type PropertyName2 { get; set; }
}
```

What is the proxy object?

An object that is created from a POCO class (model class) to support change tracking and lazy loading, is known as a proxy object.

There are some rules for creating a proxy class object:

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

What are the various Entity States in EF?

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

- Added
- Modified

- Deleted
- Unchanged
- Detached

What are various approaches in Code First for model designing?

- POCO model classes with data annotations
- POCO model classes with FluentAPI in DbContext

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:

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

C# Data Type	**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

What is Code First Migrations in Entity Framework?

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.

What is Migrations History Table?

In EF Core, 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.

How you apply code first migrations through code in EF Core?

Entity Framework Core supports “Migrate()” method, which can be called anywhere either in controller or in any other middleware when you want to perform code first migrations programmatically instead of applying migrations using “Update-Database” command.

Eg:

```
using (var context = new MyDbContext(...))
{
    context.Database.Migrate();
}
```