**Database First Approach in Asp.Net Core MVC with EF Core**



[By Gowtham](https://www.techmeet360.com/blog/author/gowtham-sundarapandiankovai-co/)

|  Posted: August 10, 2022  |  Categories: [.NET](https://www.techmeet360.com/blog/category/net/) [C#](https://www.techmeet360.com/blog/category/c/) [CSharp](https://www.techmeet360.com/blog/category/csharp/) [Microsoft](https://www.techmeet360.com/blog/category/microsoft/)

Tags: [.NET Core](https://www.techmeet360.com/blog/tag/net-core/) [Entity Framework](https://www.techmeet360.com/blog/tag/entity-framework/) [SQL Server](https://www.techmeet360.com/blog/tag/sql-server/)

A database is an organized collection of structured information, or data, typically stored electronically in a computer system. There are several types of databases that can be used in real world applications. Let’s talk about relational databases (representing data in the form of table). While creating the database, developers adapt various approach as per their requirements. Let’s talk about database first approach.

A blue rectangular sign with white text

Description automatically generated

In database first approach, the database and tables are created first. Then the entity data model is created using the created database. It is easier to map and create the keys and the relationships as the developer does not have to write them in code. Therefore, this approach is suitable for large applications that rely highly on data.

**Other approaches**

**Code First Approach:**

A blue rectangle with white text

Description automatically generated

In the code first approach, the developer first creates the entity classes with the required properties. Then the entity framework creates database and tables according to those defined classes. This approach is more suitable for small scale applications.

**Model First Approach:**

A diagram of a framework

Description automatically generated

In the model first approach you can create the Entity Data Model (EDM) first, then generate the database from it. Entities definition and their relationships are created using designer in Visual Studio. If neither the domain classes nor the database is ready, then model first approach is the go-to solution.

**Comparison:**

A group of blue rectangular boxes with white text

Description automatically generated

|  |  |  |
| --- | --- | --- |
| Database First | Code First | Model First |
| Preferred for intense and large applications. | Good for small applications. | Preferred when the domain classes nor the database is ready. |
| The database is created with raw queries. | Database is created using scaffolding. | Database is created using a GUI tool. |

**Entity Framework:**

A diagram of a data base

Description automatically generated

Entity Framework (EF) is an object-relational mapper that enables .NET developers to work with relational data using domain-specific objects. It eliminates the need for most of the data-access code that developers usually need to write. It is an enhancement to ADO.NET that gives developers an automated mechanism for accessing & storing the data in the database.

**What is DbContext?**

The DBContext class is basically used by our application to interact with the underlying database. That means this class is used to manage the database connection as well as also used to perform CRUD operations with the underlying database.

In order to perform any useful task by the DbContext class, we need an instance of the DbContextOptions class. The instance of the DbContextOptions carries all the required configuration information such as the connection string, database provider, etc.

**What is DbSet?**

A DbSet represents the collection of all entities in the context, or that can be queried from the database, of a given type.

DBSet is an abstraction which is literally a collection of objects (usually lazily loaded) stored in a table. Operations done on DBSet are actually performed on the actual database records via SQL queries.

**Implementation:**

Let’s create a ***university*** database with database first approach

To create a database –

|  |  |
| --- | --- |
|  | -- Create a new database called 'University' |
|  | -- Connect to the 'master' database to run this snippet |
|  |  |
|  | USE master |
|  | GO |
|  |  |
|  | -- Create the new database if it does not exist already |
|  | IF NOT EXISTS ( |
|  | SELECT [name] |
|  | FROM sys.databases |
|  | WHERE [name] = N'University' |
|  | ) |
|  | CREATE DATABASE University |
|  | GO |

[view raw](https://gist.github.com/gowtham758550/c0c1914ed9fc77ae2ea554ef7f2f88d3/raw/fb51078038df3cf424f68610f655164296e9126e/create-database.sql)[create-database.sql](https://gist.github.com/gowtham758550/c0c1914ed9fc77ae2ea554ef7f2f88d3#file-create-database-sql)hosted with  by [GitHub](https://github.com/)

To create tables,

|  |  |
| --- | --- |
|  | --student table |
|  | CREATE TABLE [dbo].[Student] ( |
|  | [StudentID] INT IDENTITY (1, 1) NOT NULL, |
|  | [LastName] NVARCHAR (50) NULL, |
|  | [FirstName] NVARCHAR (50) NULL, |
|  | [EnrollmentDate] DATETIME NULL, |
|  | PRIMARY KEY CLUSTERED ([StudentID] ASC) |
|  | ) |
|  |  |
|  | --course table |
|  | CREATE TABLE [dbo].[Course] ( |
|  | [CourseID] INT IDENTITY (1, 1) NOT NULL, |
|  | [Title] NVARCHAR (50) NULL, |
|  | [Credits] INT NULL, |
|  | PRIMARY KEY CLUSTERED ([CourseID] ASC) |
|  | ) |
|  |  |
|  | --enrollment table |
|  | CREATE TABLE [dbo].[Enrollment] ( |
|  | [EnrollmentID] INT IDENTITY (1, 1) NOT NULL, |
|  | [Grade] DECIMAL(3, 2) NULL, |
|  | [CourseID] INT NOT NULL, |
|  | [StudentID] INT NOT NULL, |
|  | PRIMARY KEY CLUSTERED ([EnrollmentID] ASC), |
|  | CONSTRAINT [FK\_dbo.Enrollment\_dbo.Course\_CourseID] FOREIGN KEY ([CourseID]) |
|  | REFERENCES [dbo].[Course] ([CourseID]) ON DELETE CASCADE, |
|  | CONSTRAINT [FK\_dbo.Enrollment\_dbo.Student\_StudentID] FOREIGN KEY ([StudentID]) |
|  | REFERENCES [dbo].[Student] ([StudentID]) ON DELETE CASCADE |
|  | ) |

[view raw](https://gist.github.com/gowtham758550/fc5555bfdbeebf4b6de90367457c92f7/raw/1c90430115c70abba0b1c0555a2f06eae8aa1eed/create-tables.sql)[create-tables.sql](https://gist.github.com/gowtham758550/fc5555bfdbeebf4b6de90367457c92f7#file-create-tables-sql)hosted with  by [GitHub](https://github.com/)

As database creation is completed, let’s create our mvc app to scaffold models.

To create a new mvc app with dotnet cli

*dotnet new mvc –o DbFirst*

For scaffolding and integrating with the database we need some nuget packages to be installed. Let’s install those packages using dotnet cli.

*dotnet add package Microsoft.EntityFrameworkCore*

*dotnet add package Microsoft.EntityFrameworkCore.Design*

*dotnet add package Microsoft.EntityFrameworkCore.SqlServer*

Once the packages are installed, create dbcontext for our application.

A DbContext instance represents a combination of the Unit Of Work and Repository patterns such that it can be used to query from a database and group together changes that will then be written back to the store as a unit.

|  |  |
| --- | --- |
|  | public partial class DbFirstContext : DbContext |
|  | { |
|  | public DbFirstContext() |
|  | { |
|  | } |
|  |  |
|  | public DbFirstContext(DbContextOptions<DbFirstContext> options) |
|  | : base(options) |
|  | { |
|  | } |
|  | } |

[view raw](https://gist.github.com/gowtham758550/a799958f8b17f89dc6f944cefc955d31/raw/c50f541b8b6654c7447b563e941f2113b1277978/DbFirstContext.cs)[DbFirstContext.cs](https://gist.github.com/gowtham758550/a799958f8b17f89dc6f944cefc955d31#file-dbfirstcontext-cs)hosted with  by [GitHub](https://github.com/)

Once the dbcontext is created, then create a model with scaffolding. Enter the following command for scaffolding.

*dotnet ef dbcontext scaffold "Server=VIVOBOOK\SQLEXPRESS;Database=University;Trusted\_Connection=True;" Microsoft.EntityFrameworkCore.SqlServer -o Models*

This command will add fluent API in the context file and create the model files for the tables as shown in screenshot.

Created model files are

A screenshot of a computer

Description automatically generated

***Course.cs***

|  |  |
| --- | --- |
|  | public partial class Course |
|  | { |
|  | public Course() |
|  | { |
|  | Enrollments = new HashSet<Enrollment>(); |
|  | } |
|  |  |
|  | public int CourseId { get; set; } |
|  | public string? Title { get; set; } |
|  | public int? Credits { get; set; } |
|  |  |
|  | public virtual ICollection<Enrollment> Enrollments { get; set; } |
|  | } |

[view raw](https://gist.github.com/gowtham758550/1d210c00ffa6941ced6aad0f934c52d4/raw/4952bb1d19f8137f39be6e7903f0b8b6aa96c50d/Course.cs)[Course.cs](https://gist.github.com/gowtham758550/1d210c00ffa6941ced6aad0f934c52d4#file-course-cs)hosted with  by [GitHub](https://github.com/)

***Enrollment.cs***

|  |  |
| --- | --- |
|  | public partial class Enrollment |
|  | { |
|  | public int EnrollmentId { get; set; } |
|  | public decimal? Grade { get; set; } |
|  | public int CourseId { get; set; } |
|  | public int StudentId { get; set; } |
|  |  |
|  | public virtual Course Course { get; set; } = null!; |
|  | public virtual Student Student { get; set; } = null!; |
|  | } |

[view raw](https://gist.github.com/gowtham758550/a368c821d0b6dc9aa33f1ba41b2d224a/raw/b301caf5ce6ea5c247783b81780f5e2e8b72759f/Enrollment.cs)[Enrollment.cs](https://gist.github.com/gowtham758550/a368c821d0b6dc9aa33f1ba41b2d224a#file-enrollment-cs)hosted with  by [GitHub](https://github.com/)

***Student.cs***

|  |  |
| --- | --- |
|  | public partial class Student |
|  | { |
|  | public Student() |
|  | { |
|  | Enrollments = new HashSet<Enrollment>(); |
|  | } |
|  |  |
|  | public int StudentId { get; set; } |
|  | public string? LastName { get; set; } |
|  | public string? FirstName { get; set; } |
|  | public DateTime? EnrollmentDate { get; set; } |
|  |  |
|  | public virtual ICollection<Enrollment> Enrollments { get; set; } |
|  | } |

[view raw](https://gist.github.com/gowtham758550/18f3276bbe49df6926774021158ad2ec/raw/a72e0bf08a26ce2b694970d48ffe3fdbcd5ffcd5/Student.cs)[Student.cs](https://gist.github.com/gowtham758550/18f3276bbe49df6926774021158ad2ec#file-student-cs)hosted with  by [GitHub](https://github.com/)

**Finally, which approach to use?**

These days, it’s a fairly simple question.

* If you are starting a new project with a new database, use EF core code first approach.
* If you are starting a new project, with an existing database you want to use EF core database first approach.