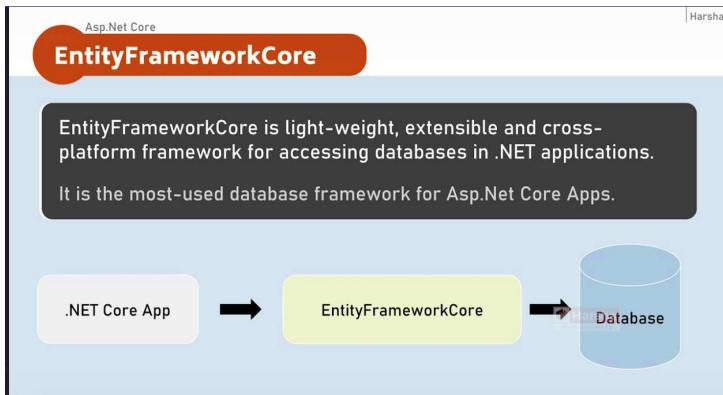


Entity Framework Core

It is one of the most commonly used database frameworks with ASP.NET Core applications.
It is a lightweight, extensible, cross-platform framework for connecting to databases as a part of .NET Core applications.
So why do you require database connections in your projects?



See, what about the information that you are maintaining currently? For example, a person's information or countries—all of that is temporary.

Persons

Person Name	Email	Date of Birth	Age	Gender	Country	Address	Receive News Letters	Options
Aguste	aleddy0@booking.com	02 Jan 1993	29	Male	USA	0858 Novick Terrace	False	Edit Delete
Corabelle	cadams5@t-online.de	23 Oct 1993	29	Female	India	4489 Hazelcrest Place	False	Edit Delete
Dulcinea	dbus4@pbs.org	02 Sep 1996	26	Female	UK	56 Sundown Point	False	Edit Delete
Faydra	fbischof6@boston.com	14 Feb 1996	26	Female	Australia	2010 Farragut Pass	True	Edit Delete
Freemon	faugustin9@vimeo.com	27 Apr 1996	26	Male	Australia	8754 Becker Street	False	Edit Delete
Jasmina	jsyddie1@mailbeian.gov.cn	24 Jun 1991	31	Female	Canada	0742 Fieldstone Lane	True	Edit Delete
Kendall	khaquard2@arstechnica.com	13 Aug 1993	29	Male	Canada	7050 Pawling Alley	False	Edit Delete

For example, if you make any changes here—let's say I add the digit "1" after the name—it seems to be updated.
But next time, when you close and restart the application, the changes are gone.

Why? Because the changes you make are stored in a temporary in-memory collection, which gets destroyed as soon as the application closes.

To overcome this problem, we need to store data permanently, and databases are the solution.
Every real-world application requires a database, and it also needs a database framework to connect with the database from the code.

Entity Framework Core is the tool or framework that helps you connect with databases efficiently.

Particularly in Entity Framework Core, you need to keep your model classes in sync with your actual database tables. In general, a table contains columns, and the model class is coupled with the table, where each property name is mapped to the corresponding column.

By convention, property names should be the same as column names, but it is not mandatory.

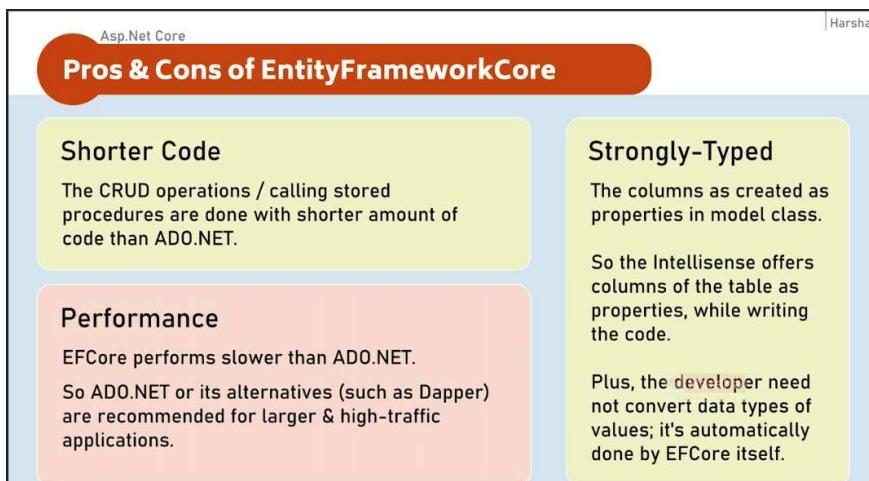
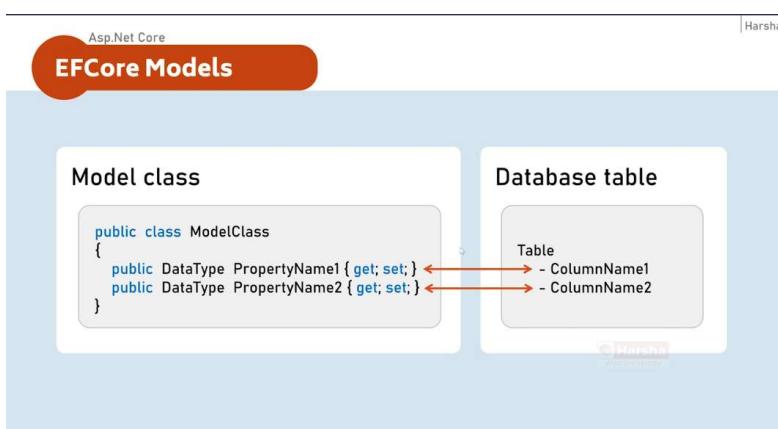
You may use different names if required and map them to specific columns using data annotations.

However, following the practice of keeping property names the same as column names is a common convention. so you have to create one model class for one table

in your database if you have 30 tables

for example you have to create 30 model

classes



Benefits of Using Entity Framework Core Compared to Traditional ADO.NET

When comparing Entity Framework Core with traditional database connections such as ADO.NET, there are several benefits:

1. Less Code, Faster Development

- With Entity Framework Core, you write a shorter amount of code.
- You can quickly create a DbContext object and write LINQ queries against it to perform query operations.
- Calling SaveChanges() automatically handles updates, insertions, or deletions based on changes made to entities.

2. Simplified Code & Object Management

- In ADO.NET, you need to create multiple objects like SqlConnection, SqlDataAdapter, SqlCommand, SqlDataReader, and more.
- Entity Framework Core abstracts this complexity, allowing you to focus on business logic rather than database interactions.

3. Strongly Typed Data

- In ADO.NET, column names are written as string literals (e.g., dataReader["PersonName"].ToString()), which is error-prone and requires typecasting.
- In Entity Framework Core, properties are strongly typed, meaning values are automatically typecasted based on the declared type (e.g., Person.Name as a string).

- This reduces errors

Benefits of Using Entity Framework Core Compared to Traditional ADO.NET

When comparing Entity Framework Core with traditional database connections such as ADO.NET, there are several benefits:

- 1. Less Code, Faster Development**
 - With Entity Framework Core, you write a shorter amount of code.
 - You can quickly create a DbContext object and write LINQ queries against it to perform query operations.
 - Calling SaveChanges() automatically handles updates, insertions, or deletions based on changes made to entities.
- 2. Simplified Code & Object Management**
 - In ADO.NET, you need to create multiple objects like SqlConnection, SqlDataAdapter, SqlCommand, SqlDataReader, and more.
 - Entity Framework Core abstracts this complexity, allowing you to focus on business logic rather than database interactions.
- 3. Strongly Typed Data**
 - In ADO.NET, column names are written as string literals (e.g., dataReader["PersonName"].ToString()), which is error-prone and requires typecasting.
 - In Entity Framework Core, properties are strongly typed, meaning values are automatically typecasted based on the declared type (e.g., Person.Name as a string).
 - This reduces errors and eliminates the need for manual type conversion.
- 4. Automatic Mapping of Columns to Properties**
 - Instead of manually handling column-to-property mapping, Entity Framework Core automatically maps database columns to class properties.
 - If you have 20 columns in a table, you define 20 properties in your entity, and values are directly available in those properties.

Drawback of Entity Framework Core

- Slower Performance Compared to ADO.NET**
 - Entity Framework Core is built on top of ADO.NET, meaning it introduces an abstraction layer.
 - Even though it is highly optimized, it cannot outperform raw ADO.NET queries.
 - For **data-intensive applications** with high traffic, direct ADO.NET queries or alternatives like **Dapper** may be preferable.

When to Use Entity Framework Core?

- Recommended for small to medium applications** due to ease of development.
- Good for prototyping** in the early stages of development.
- Not ideal for high-performance applications** handling large amounts of data.

What's Next?

In this section, we will explore Entity Framework Core and perform CRUD operations using LINQ queries as well as stored procedures.

- and eliminates the need for manual type conversion.

- Automatic Mapping of Columns to Properties**
 - Instead of manually handling column-to-property mapping, Entity Framework Core automatically maps database columns to class properties.
 - If you have 20 columns in a table, you define 20 properties in your entity, and values are directly available in those properties.

Drawback of Entity Framework Core

- Slower Performance Compared to ADO.NET**
 - Entity Framework Core is built on top of ADO.NET, meaning it introduces an abstraction layer.
 - Even though it is highly optimized, it cannot outperform raw ADO.NET queries.
 - For **data-intensive applications** with high traffic, direct ADO.NET queries or alternatives like **Dapper** may be preferable.

When to Use Entity Framework Core?

- Recommended for small to medium applications** due to ease of development.
- Good for prototyping** in the early stages of development.
- Not ideal for high-performance applications** handling large amounts of data.

What's Next?

In this section, we will explore Entity Framework Core and perform CRUD operations using LINQ queries as well as stored procedures.



Two Approaches in Entity Framework Core: Database First vs. Code First

Entity Framework Core can be used in **two approaches**, also known as **two ways**:

- 1. Database First Approach**
- 2. Code First Approach**

Difference Between Database First & Code First Approach

- Database First Approach**
 - You **generate entity model classes** based on an **existing database**.
 - The database structure is already defined, and Entity Framework extracts models from it.
- Code First Approach**
 - You **create entity model classes first**, and the **database is generated automatically** using migrations.
 - You apply migration commands (dotnet ef migrations add and dotnet ef database update) to reflect changes in the database.

Handling Database Changes

Aspect	Database First Approach	Code First Approach
Modifying database schema	Changes must be made directly in the database using SQL commands.	Changes are made in entity model classes and applied using migrations.
Syncing models with the database	After modifying the database, models must be manually updated to match.	Entity Framework automatically syncs models with the database via migrations.
Control over database	More control over database design, making it suitable for	Easier for new projects where database structure

structure

legacy or large-scale databases.

evolves with development.

Which One to Use?

Choose Database First If:

- You are working with an **existing database**.
- You need **precise control** over the database schema.
- The database is **managed by DBAs** separately from application code.

Choose Code First If:

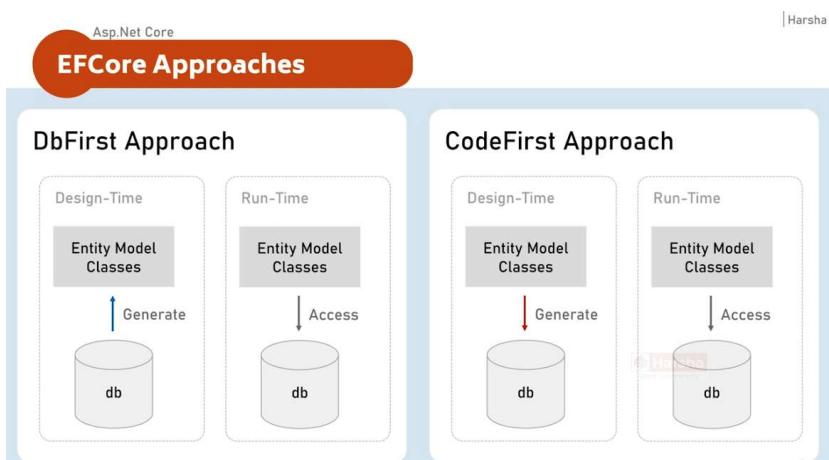
- You are **starting a new project** from scratch.
- You prefer to manage the database **directly from code**.
- You want **automatic database migrations** instead of manual SQL changes.

Runtime Behavior (Same for Both Approaches)

- Queries written using **LINQ** will retrieve data from the database.
- Calling **SaveChanges()** will apply **insert, update, and delete** operations to the database.
- The runtime model remains the same regardless of the approach used.

Conclusion

- **Database First** is better for existing databases and projects with strict DB control.
- **Code First** is better for new projects where database structure evolves with development.
- **Both approaches work the same at runtime**, so the choice depends on **how you prefer to manage database schema changes**.



Which Approach is Better? When to Prefer Which One?

Both **Code First** and **Database First** approaches have their own advantages and use cases. The choice depends on factors like project requirements, database management preferences, and long-term scalability.

Code First Approach: Best for New Applications

When to Use Code First:

- When developing a **new application** from scratch.
- When you **don't have an existing database**.
- When you prefer to manage the database schema **directly from C# models**.
- When you want to use **automatic database migrations** to evolve the schema.
- When working in **small teams** where the database structure changes frequently.
- For **prototyping and proof-of-concept applications** before client approval.

Limitations of Code First:

- **Limited database control**—DBA teams may not prefer this approach.
- Direct SQL modifications are **discouraged**, as migrations should handle schema changes.
- **Less flexibility** when dealing with database-specific optimizations like indexes, stored procedures, triggers, and complex queries.
- Risk of **losing manual database changes** when migrating to another environment.

Database First Approach: Best for Existing Databases

When to Use Database First:

- When **working with an existing database** (migrating an old application).
- When the **DBA team manages the database independently**.
- When the database requires **manual optimizations** like indexes, stored procedures, and triggers.
- When performance is a **high priority** and you need **precise control** over database schema.
- When dealing with **large-scale, data-intensive applications** with high traffic.

Limitations of Database First:

- Requires **manual synchronization** between database schema and entity models.
- **More effort** when making schema changes (developers must update both the database and code manually).
- **Less flexibility for rapid development**, as database modifications require SQL changes first.

Which Approach to Choose Based on Application Size?

Application Size	Recommended Approach
Small Applications / Prototypes	Code First
New Mid-Sized Applications	Code First or Database First
Large Applications with Heavy Data Usage	Database First
High-Traffic Enterprise Systems	Database First or Custom ORM (e.g., Dapper)

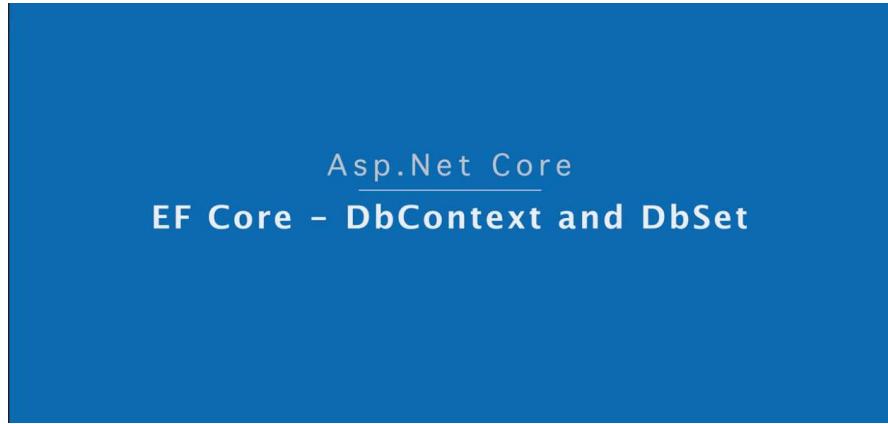
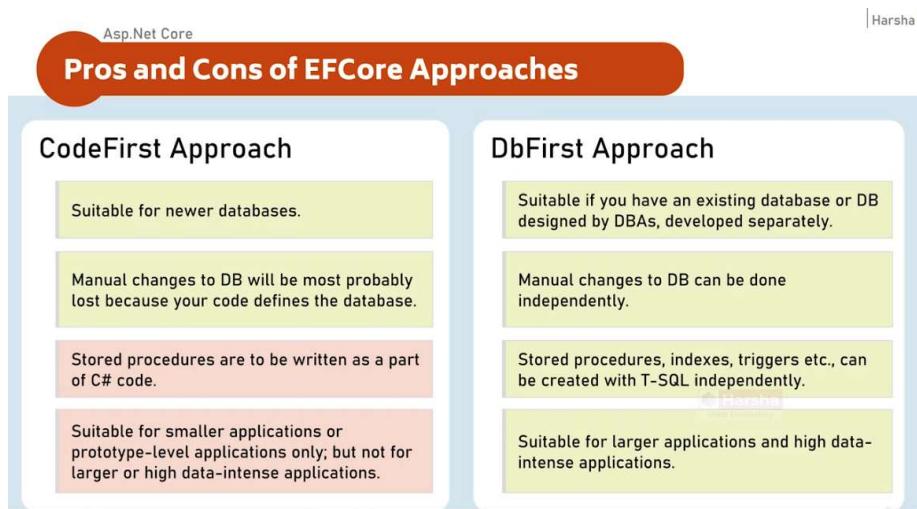
Can You Switch Between Code First and Database First?

Yes! Many developers start with **Code First** for prototyping and then **migrate to Database First** as the project grows and requires more database control.

Final Recommendation

- **For learning purposes, start with Code First** since it simplifies development.
- **For real-world projects, consider Database First** if the database needs independent management.
- **For high-performance applications, consider Database First or an alternative like Dapper.**

In the next sections, we will explore **Entity Framework Core** using the **Code First approach**, as it is the best starting point for learning EF Core. ↗



In order to begin with Entity Framework Core, you require to know **DbContext** and **DbSet**. The **DbContext** is bound to a specific SQL database, and **DbSet** is bound to a specific table.

For example, in your SQL Server, you have a database with two tables. So, for the database, you will be creating one **DbContext**, and for each table, you will be creating one **DbSet**.

The **DbSet** is a generic class, so you require to pass the generic parameter, and that is your model class.

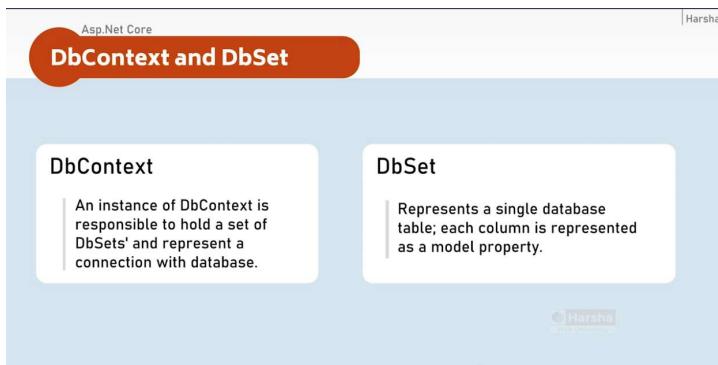
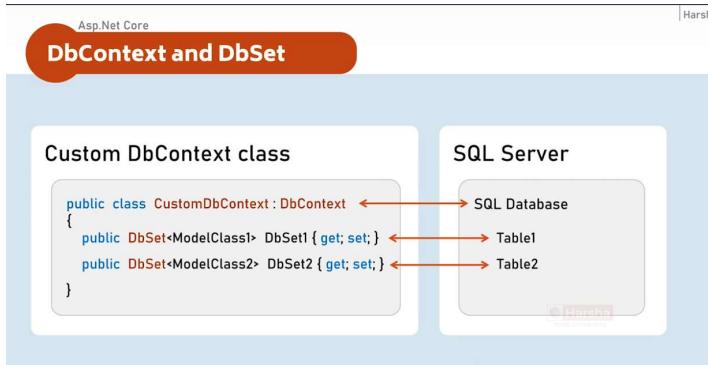
For each table, you will be creating a model class, right? That model class will be mentioned here.

For example, we want two tables like Countries and Persons, and the model class names will be singular, that is **Country** and **Person**, respectively. So, you will be creating **DbSet<Country>** and **DbSet<Person>**.

It is a convention to name the **DbSet** as plural, meaning the **DbSet** names will be **Countries** and **Persons**, respectively.

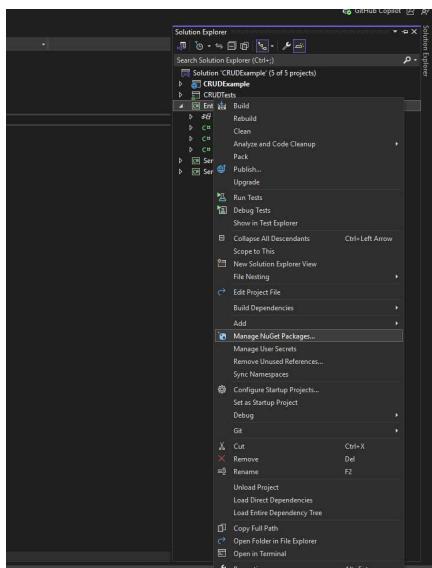
Here, your **DbContext** class name can be anything, but it must be inherited from a predefined class called **DbContext** from the **Microsoft.EntityFrameworkCore** namespace.

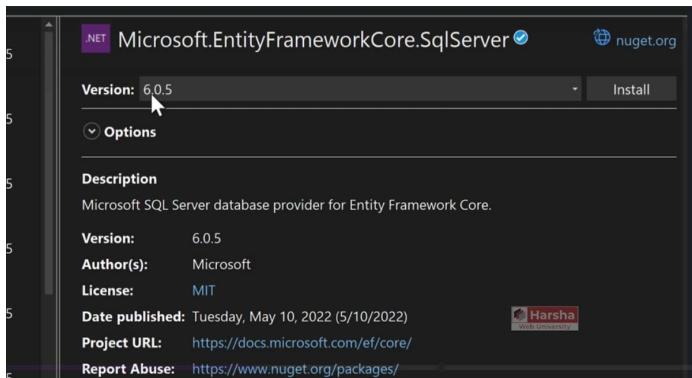
In the same way, in the same namespace, you have the DbSet class as well.
So, in short, the DbContext represents the entire database, but DbSet represents a single table or maybe a database view.
OK, let me demonstrate the same practically.



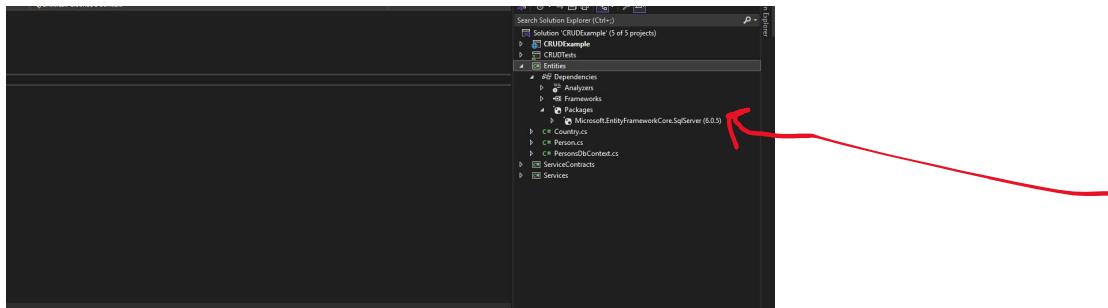
But it is a normal class, we have to convert it into a DbContext class, To do that, we need to download a nuget package.

```
1  using System;
2  using System.Collections.Generic;
3  ...
4  namespace Entities
5  {
6      public class PersonsDbContext
7      {
8      }
9  }
```





To ensure that package is installed or not.



Our Dbset is ready now.

```

    PersonDbContext.cs
    Person.cs
    Country.cs
    Entities
    1  using Microsoft.EntityFrameworkCore;
    2  using System;
    3  using System.Collections.Generic;
    4
    5  namespace Entities
    6  {
    7      public class PersonDbContext : DbContext
    8      {
    9          public DbSet<Country> Countries { get; set; }
    10         public DbSet<Person> Persons { get; set; }
    11
    12         //we have to bind these DbSets to corresponding table
    13         protected override void OnModelCreating(ModelBuilder modelBuilder)
    14         {
    15             base.OnModelCreating(modelBuilder);
    16
    17             //modelBuilder.Entity<Country>() => Hey modelBuilder, I am trying to talk about an entity of the 'Country' type.
    18             //This method will get the table name of dbSet that is 'Countries' (line no 9)
    19             // and that is mapped to the table called 'Countries'
    20
    21             modelBuilder.Entity<Country>().ToTable("Countries"); //I want the table name to be 'Countries'. you can define any
    22             modelBuilder.Entity<Person>().ToTable("Persons");
    23
    24         }
    25     }
    
```

After that, we have to add our db set as a service in your program.cs file. Because EF core by default uses 'Dependency Injection'

Add DbContext as Service

in Program.cs:

```

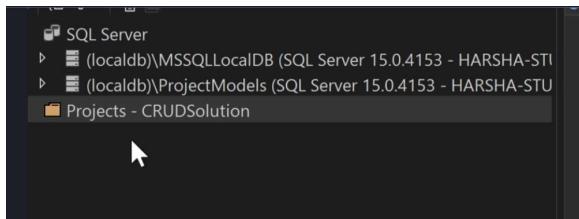
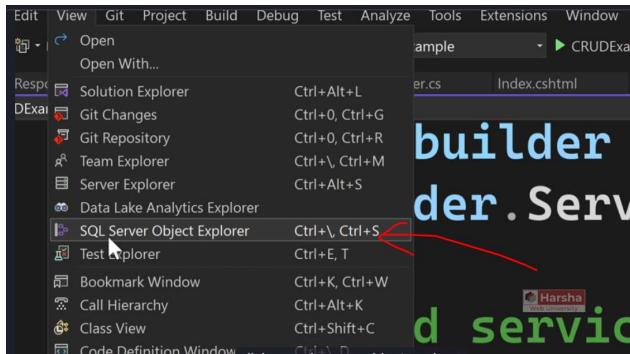
builder.Services.AddDbContext<DbContextClassName>(
    options => {
        options.UseSqlServer();
    });
    
```

Asp.Net Core

EF Core - Connection String

A connection string is a string value that specifies the details of the database you want to connect to. In the initial stages of development, it is recommended to use the local database. This means your SQL Server database file will be created as part of the same developer machine. If there are multiple developers, each developer would have their own database on their own machine, ensuring no conflicts between databases of multiple developers.

In order to create a new database go to the view menu, click on sql server object explorer.



Here you can see your local DB instances.

SQL Server mainly supports two types of database instances:

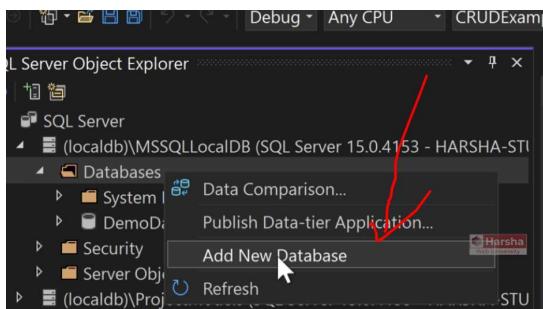
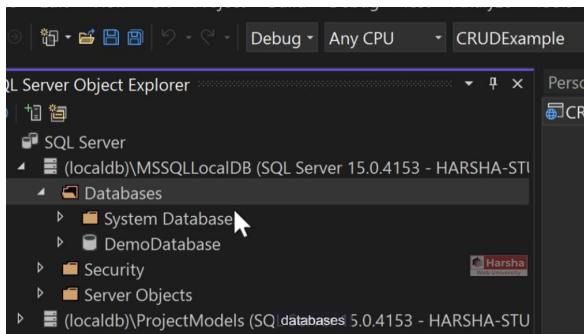
1. **Standard SQL Server database instance** – Connected based on the server name or IP address.
2. **LocalDB** – This LocalDB database file is created as part of the same developer machine.

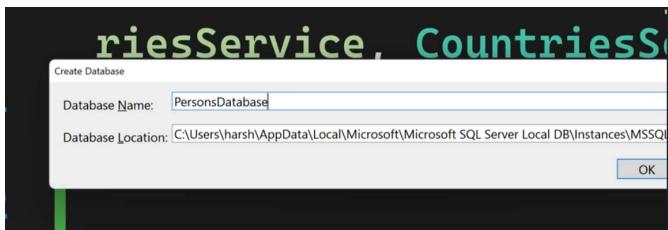
For staging and production environments, maintaining a dedicated centralized database server is recommended. However, for development scenarios, it is recommended to use LocalDB.

When you install Visual Studio, it installs LocalDB by default, named **MSSQLLocalDB**, which is the name of the LocalDB server instance.

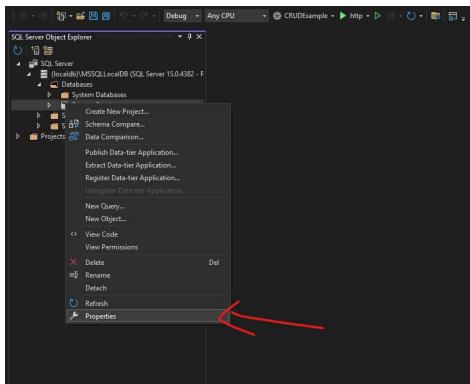
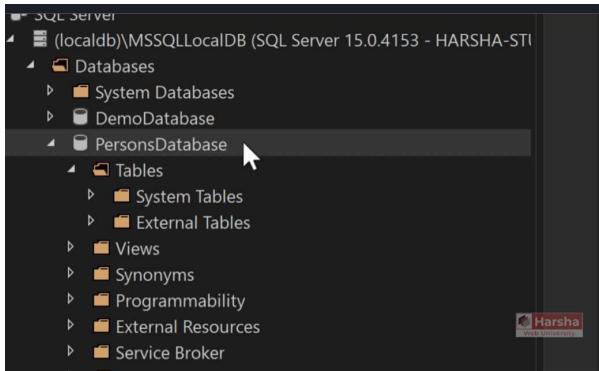
In the SQL Server Object Explorer, expand LocalDB and click on **Databases** to see the list of existing databases.

Now, I would like to create a new one.

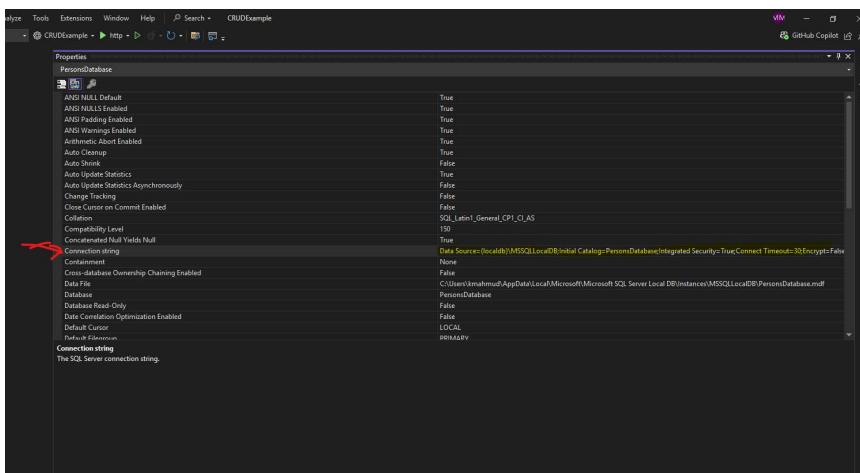




There are no tables by default. We would like to connect to this database. But we have to mention the connection details as connection string.



Copy this connection string.



Now paste the connection string into program.cs file.

Open your Configuration file appSettings.json file and paste here.

```
appsettings.json
{
  "Logging": {
    "LogLevel": {
      "Default": "Information",
      "Microsoft.AspNetCore": "Warning"
    }
  },
  "AllowedHosts": "*",
  "ConnectionStrings": {
    "DefaultConnection": "Data Source=(localdb)\\MSSQLLocalDB;Initial Catalog=PersonsDatabase;Integrated Security=True;Connect Timeout=30;Encrypt=False;TrustServerCertificate=False"
  }
}
```



```
public class Person
{
    [Key]
    public Guid PersonID { get; set; }

    [StringLength(40)] //nvarchar(40)
    public string? PersonName { get; set; }

    [StringLength(40)] //nvarchar(40)
    public string? Email { get; set; }
}
```

Okay, assume that we have created the Persons and Countries table, and by default, those tables will be empty, right? Assume there is a Persons table with all these columns, but by default, it has no rows. But generally, we would like to add some initial data into the table, and that is called seed data here.

Asp .Net Core

Seed Data

in DbContext:

```
modelBuilder.Entity<ModelClass>().HasData(entityObject);
```

Seed Data

It adds initial data (initial rows) in tables, when the database is newly created.

How to add seed data in Entity Framework?

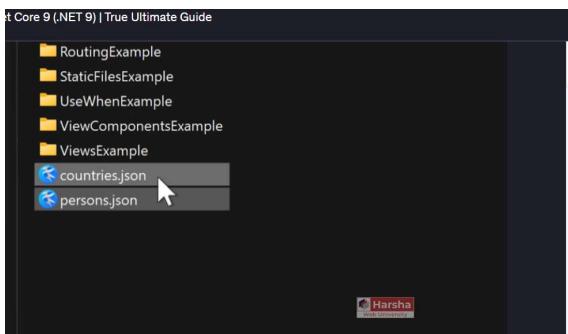
We have been provided with a method called `HasData`. If you supply the entity object to it, that object (i.e., that row) will be inserted into the database table as soon as the database is created initially. However, it will not be reinserted if you delete it, meaning this data will not be maintained in the table if you make any changes to the table. These are the initial rows that get added only when the table is created, not after that. So, let me add the seed data.

```

1  using Microsoft.EntityFrameworkCore;
2  using System;
3  using System.Collections.Generic;
4
5  namespace Entities
6  {
7      public class PersonsDbContext : DbContext
8      {
9          public DbSet<Country> Countries { get; set; }
10         public DbSet<Person> Persons { get; set; }
11
12         //we have to bind these Obsets to corresponding table
13         protected override void OnModelCreating(ModelBuilder modelBuilder)
14         {
15             base.OnModelCreating();
16
17             //modelBuilder.Entity<Country>().ToTable("Countries"); // Hey modelBuilder, I am trying to talk about an entity of the 'Country' type.
18             //This method will get the country type of dbSet that is 'Countries' (line no 9)
19             // and that is mapped to the table called 'Countries'
20
21             modelBuilder.Entity<Country>().ToTable("Countries"); //I want the table name to be 'Countries'. you can define any other name.
22             modelBuilder.Entity<Person>().ToTable("Persons");
23
24             //To add the seed data we have to use this on model creating method in the dbcontext class.
25             //so after you map your model class to the table. here is the place for that.
26
27             //Seed to Countries
28             modelBuilder.Entity<Country>().HasData(new Country() { CountryID = Guid.NewGuid(), CountryName = "Bangladesh" });
29
30             //but there is a more shortcut way that this.
31
32         }
33     }
34 }

```

Paste the json file into your Project folder.



Countries.json file and Pesron.json file

```

1  [
2      {"CountryID": "114629847-905a-4a0e-9abe-88b61655c5cb", "CountryName": "Philippines", "Address": "Park Point", "Gender": "Male", "ReceiveNewsLetters": false},
3      {"CountryID": "56bf46a4-02b8-4693-a0f5-0a5e2218bdc", "CountryName": "Thailand", "Address": "Rama 9", "Gender": "Female", "ReceiveNewsLetters": true},
4      {"CountryID": "12e15727-d369-49a9-8b13-bc22e9362179", "CountryName": "China", "Address": "Beijing", "Gender": "Male", "ReceiveNewsLetters": false},
5      {"CountryID": "8f30bedc-47dd-4286-8950-73d8a68e5d41", "CountryName": "Palestinian Territory", "Address": "Jerusalem", "Gender": "Female", "ReceiveNewsLetters": true},
6      {"CountryID": "501c6d33-1bbe-45f1-8fb0-2275913c6218", "CountryName": "China", "Address": "Shanghai", "Gender": "Male", "ReceiveNewsLetters": false}
]

```

CountryId of the Person should map to Countries.json. CountryId is a foreign key to the Person table. Assume that.

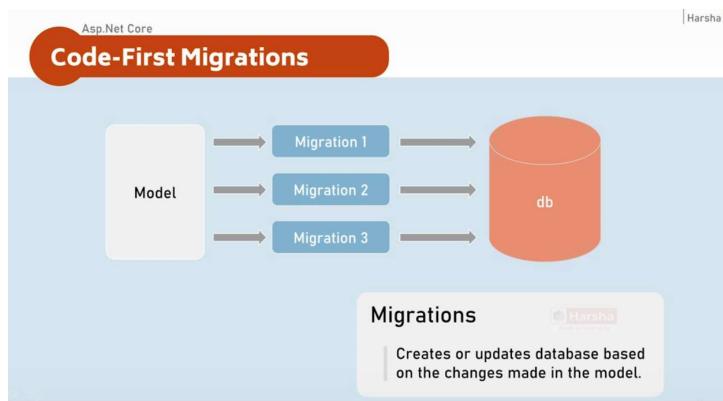
```

1  public class Person
2  {
3      public string PersonID { get; set; }
4      public string PersonName { get; set; }
5      public string Email { get; set; }
6      public DateTime DateOfBirth { get; set; }
7      public string Gender { get; set; }
8      public string CountryID { get; set; }
9      public string Address { get; set; }
10     public bool ReceiveNewsLetters { get; set; }
11
12     public virtual Country Country { get; set; }
13
14     public Person()
15     {
16         PersonID = "c3abddbd-cf80-1ld2-be4-cc7ds758928";
17         PersonName = "Harpreet";
18         Email = "mehbsdale@people.com.cn";
19         DateOfBirth = "1989-08-28";
20         Gender = "Male";
21         CountryID = "114629847-905a-4a0e-9abe-88b61655c5cb";
22         Address = "4 Parkside Point";
23         ReceiveNewsLetters = false;
24     }
25
26     public Person(string personName, string email, string address)
27     {
28         PersonID = "c3abddbd-cf80-1ld2-be4-cc7ds758928";
29         PersonName = "Harpreet";
30         Email = "mehbsdale@people.com.cn";
31         DateOfBirth = "1989-08-28";
32         Gender = "Male";
33         CountryID = "114629847-905a-4a0e-9abe-88b61655c5cb";
34         Address = "4 Parkside Point";
35         ReceiveNewsLetters = false;
36     }
37
38     public Person(string personName, string email, string address, string countryID)
39     {
40         PersonID = "c3abddbd-cf80-1ld2-be4-cc7ds758928";
41         PersonName = "Harpreet";
42         Email = "mehbsdale@people.com.cn";
43         DateOfBirth = "1989-08-28";
44         Gender = "Male";
45         CountryID = "114629847-905a-4a0e-9abe-88b61655c5cb";
46         Address = "4 Parkside Point";
47         ReceiveNewsLetters = false;
48     }
49
50     public Person(string personName, string email, string address, string countryID, bool receiveNewsLetters)
51     {
52         PersonID = "c3abddbd-cf80-1ld2-be4-cc7ds758928";
53         PersonName = "Harpreet";
54         Email = "mehbsdale@people.com.cn";
55         DateOfBirth = "1989-08-28";
56         Gender = "Male";
57         CountryID = "114629847-905a-4a0e-9abe-88b61655c5cb";
58         Address = "4 Parkside Point";
59         ReceiveNewsLetters = receiveNewsLetters;
60     }
61
62     public Person(string personName, string email, string address, string countryID, DateTime dateOfBirth, string gender)
63     {
64         PersonID = "c3abddbd-cf80-1ld2-be4-cc7ds758928";
65         PersonName = "Harpreet";
66         Email = "mehbsdale@people.com.cn";
67         DateOfBirth = dateOfBirth;
68         Gender = gender;
69         CountryID = "114629847-905a-4a0e-9abe-88b61655c5cb";
70         Address = "4 Parkside Point";
71         ReceiveNewsLetters = false;
72     }
73
74     public Person(string personName, string email, string address, string countryID, DateTime dateOfBirth, string gender, bool receiveNewsLetters)
75     {
76         PersonID = "c3abddbd-cf80-1ld2-be4-cc7ds758928";
77         PersonName = "Harpreet";
78         Email = "mehbsdale@people.com.cn";
79         DateOfBirth = dateOfBirth;
80         Gender = gender;
81         CountryID = "114629847-905a-4a0e-9abe-88b61655c5cb";
82         Address = "4 Parkside Point";
83         ReceiveNewsLetters = receiveNewsLetters;
84     }
85
86     public Person(string personName, string email, string address, string countryID, DateTime dateOfBirth, string gender, bool receiveNewsLetters, string address2)
87     {
88         PersonID = "c3abddbd-cf80-1ld2-be4-cc7ds758928";
89         PersonName = "Harpreet";
90         Email = "mehbsdale@people.com.cn";
91         DateOfBirth = dateOfBirth;
92         Gender = gender;
93         CountryID = "114629847-905a-4a0e-9abe-88b61655c5cb";
94         Address = "4 Parkside Point";
95         Address2 = address2;
96         ReceiveNewsLetters = receiveNewsLetters;
97     }
98
99     public Person(string personName, string email, string address, string countryID, DateTime dateOfBirth, string gender, bool receiveNewsLetters, string address2, string address3)
100    {
101        PersonID = "c3abddbd-cf80-1ld2-be4-cc7ds758928";
102        PersonName = "Harpreet";
103        Email = "mehbsdale@people.com.cn";
104        DateOfBirth = dateOfBirth;
105        Gender = gender;
106        CountryID = "114629847-905a-4a0e-9abe-88b61655c5cb";
107        Address = "4 Parkside Point";
108        Address2 = address2;
109        Address3 = address3;
110        ReceiveNewsLetters = receiveNewsLetters;
111    }
112
113    public Person(string personName, string email, string address, string countryID, DateTime dateOfBirth, string gender, bool receiveNewsLetters, string address2, string address3, string address4)
114    {
115        PersonID = "c3abddbd-cf80-1ld2-be4-cc7ds758928";
116        PersonName = "Harpreet";
117        Email = "mehbsdale@people.com.cn";
118        DateOfBirth = dateOfBirth;
119        Gender = gender;
120        CountryID = "114629847-905a-4a0e-9abe-88b61655c5cb";
121        Address = "4 Parkside Point";
122        Address2 = address2;
123        Address3 = address3;
124        Address4 = address4;
125        ReceiveNewsLetters = receiveNewsLetters;
126    }
127
128    public Person(string personName, string email, string address, string countryID, DateTime dateOfBirth, string gender, bool receiveNewsLetters, string address2, string address3, string address4, string address5)
129    {
130        PersonID = "c3abddbd-cf80-1ld2-be4-cc7ds758928";
131        PersonName = "Harpreet";
132        Email = "mehbsdale@people.com.cn";
133        DateOfBirth = dateOfBirth;
134        Gender = gender;
135        CountryID = "114629847-905a-4a0e-9abe-88b61655c5cb";
136        Address = "4 Parkside Point";
137        Address2 = address2;
138        Address3 = address3;
139        Address4 = address4;
140        Address5 = address5;
141        ReceiveNewsLetters = receiveNewsLetters;
142    }
143
144    public Person(string personName, string email, string address, string countryID, DateTime dateOfBirth, string gender, bool receiveNewsLetters, string address2, string address3, string address4, string address5, string address6)
145    {
146        PersonID = "c3abddbd-cf80-1ld2-be4-cc7ds758928";
147        PersonName = "Harpreet";
148        Email = "mehbsdale@people.com.cn";
149        DateOfBirth = dateOfBirth;
150        Gender = gender;
151        CountryID = "114629847-905a-4a0e-9abe-88b61655c5cb";
152        Address = "4 Parkside Point";
153        Address2 = address2;
154        Address3 = address3;
155        Address4 = address4;
156        Address5 = address5;
157        Address6 = address6;
158        ReceiveNewsLetters = receiveNewsLetters;
159    }
160
161    public Person(string personName, string email, string address, string countryID, DateTime dateOfBirth, string gender, bool receiveNewsLetters, string address2, string address3, string address4, string address5, string address6, string address7)
162    {
163        PersonID = "c3abddbd-cf80-1ld2-be4-cc7ds758928";
164        PersonName = "Harpreet";
165        Email = "mehbsdale@people.com.cn";
166        DateOfBirth = dateOfBirth;
167        Gender = gender;
168        CountryID = "114629847-905a-4a0e-9abe-88b61655c5cb";
169        Address = "4 Parkside Point";
170        Address2 = address2;
171        Address3 = address3;
172        Address4 = address4;
173        Address5 = address5;
174        Address6 = address6;
175        Address7 = address7;
176        ReceiveNewsLetters = receiveNewsLetters;
177    }
178
179    public Person(string personName, string email, string address, string countryID, DateTime dateOfBirth, string gender, bool receiveNewsLetters, string address2, string address3, string address4, string address5, string address6, string address7, string address8)
180    {
181        PersonID = "c3abddbd-cf80-1ld2-be4-cc7ds758928";
182        PersonName = "Harpreet";
183        Email = "mehbsdale@people.com.cn";
184        DateOfBirth = dateOfBirth;
185        Gender = gender;
186        CountryID = "114629847-905a-4a0e-9abe-88b61655c5cb";
187        Address = "4 Parkside Point";
188        Address2 = address2;
189        Address3 = address3;
190        Address4 = address4;
191        Address5 = address5;
192        Address6 = address6;
193        Address7 = address7;
194        Address8 = address8;
195        ReceiveNewsLetters = receiveNewsLetters;
196    }
197
198    public Person(string personName, string email, string address, string countryID, DateTime dateOfBirth, string gender, bool receiveNewsLetters, string address2, string address3, string address4, string address5, string address6, string address7, string address8, string address9)
199    {
200        PersonID = "c3abddbd-cf80-1ld2-be4-cc7ds758928";
201        PersonName = "Harpreet";
202        Email = "mehbsdale@people.com.cn";
203        DateOfBirth = dateOfBirth;
204        Gender = gender;
205        CountryID = "114629847-905a-4a0e-9abe-88b61655c5cb";
206        Address = "4 Parkside Point";
207        Address2 = address2;
208        Address3 = address3;
209        Address4 = address4;
210        Address5 = address5;
211        Address6 = address6;
212        Address7 = address7;
213        Address8 = address8;
214        Address9 = address9;
215        ReceiveNewsLetters = receiveNewsLetters;
216    }
217
218    public Person(string personName, string email, string address, string countryID, DateTime dateOfBirth, string gender, bool receiveNewsLetters, string address2, string address3, string address4, string address5, string address6, string address7, string address8, string address9, string address10)
219    {
220        PersonID = "c3abddbd-cf80-1ld2-be4-cc7ds758928";
221        PersonName = "Harpreet";
222        Email = "mehbsdale@people.com.cn";
223        DateOfBirth = dateOfBirth;
224        Gender = gender;
225        CountryID = "114629847-905a-4a0e-9abe-88b61655c5cb";
226        Address = "4 Parkside Point";
227        Address2 = address2;
228        Address3 = address3;
229        Address4 = address4;
230        Address5 = address5;
231        Address6 = address6;
232        Address7 = address7;
233        Address8 = address8;
234        Address9 = address9;
235        Address10 = address10;
236        ReceiveNewsLetters = receiveNewsLetters;
237    }
238
239    public Person(string personName, string email, string address, string countryID, DateTime dateOfBirth, string gender, bool receiveNewsLetters, string address2, string address3, string address4, string address5, string address6, string address7, string address8, string address9, string address10, string address11)
240    {
241        PersonID = "c3abddbd-cf80-1ld2-be4-cc7ds758928";
242        PersonName = "Harpreet";
243        Email = "mehbsdale@people.com.cn";
244        DateOfBirth = dateOfBirth;
245        Gender = gender;
246        CountryID = "114629847-905a-4a0e-9abe-88b61655c5cb";
247        Address = "4 Parkside Point";
248        Address2 = address2;
249        Address3 = address3;
250        Address4 = address4;
251        Address5 = address5;
252        Address6 = address6;
253        Address7 = address7;
254        Address8 = address8;
255        Address9 = address9;
256        Address10 = address10;
257        Address11 = address11;
258        ReceiveNewsLetters = receiveNewsLetters;
259    }
260
261    public Person(string personName, string email, string address, string countryID, DateTime dateOfBirth, string gender, bool receiveNewsLetters, string address2, string address3, string address4, string address5, string address6, string address7, string address8, string address9, string address10, string address11, string address12)
262    {
263        PersonID = "c3abddbd-cf80-1ld2-be4-cc7ds758928";
264        PersonName = "Harpreet";
265        Email = "mehbsdale@people.com.cn";
266        DateOfBirth = dateOfBirth;
267        Gender = gender;
268        CountryID = "114629847-905a-4a0e-9abe-88b61655c5cb";
269        Address = "4 Parkside Point";
270        Address2 = address2;
271        Address3 = address3;
272        Address4 = address4;
273        Address5 = address5;
274        Address6 = address6;
275        Address7 = address7;
276        Address8 = address8;
277        Address9 = address9;
278        Address10 = address10;
279        Address11 = address11;
280        Address12 = address12;
281        ReceiveNewsLetters = receiveNewsLetters;
282    }
283
284    public Person(string personName, string email, string address, string countryID, DateTime dateOfBirth, string gender, bool receiveNewsLetters, string address2, string address3, string address4, string address5, string address6, string address7, string address8, string address9, string address10, string address11, string address12, string address13)
285    {
286        PersonID = "c3abddbd-cf80-1ld2-be4-cc7ds758928";
287        PersonName = "Harpreet";
288        Email = "mehbsdale@people.com.cn";
289        DateOfBirth = dateOfBirth;
290        Gender = gender;
291        CountryID = "114629847-905a-4a0e-9abe-88b61655c5cb";
292        Address = "4 Parkside Point";
293        Address2 = address2;
294        Address3 = address3;
295        Address4 = address4;
296        Address5 = address5;
297        Address6 = address6;
298        Address7 = address7;
299        Address8 = address8;
300        Address9 = address9;
301        Address10 = address10;
302        Address11 = address11;
303        Address12 = address12;
304        Address13 = address13;
305        ReceiveNewsLetters = receiveNewsLetters;
306    }
307
308    public Person(string personName, string email, string address, string countryID, DateTime dateOfBirth, string gender, bool receiveNewsLetters, string address2, string address3, string address4, string address5, string address6, string address7, string address8, string address9, string address10, string address11, string address12, string address13, string address14)
309    {
310        PersonID = "c3abddbd-cf80-1ld2-be4-cc7ds758928";
311        PersonName = "Harpreet";
312        Email = "mehbsdale@people.com.cn";
313        DateOfBirth = dateOfBirth;
314        Gender = gender;
315        CountryID = "114629847-905a-4a0e-9abe-88b61655c5cb";
316        Address = "4 Parkside Point";
317        Address2 = address2;
318        Address3 = address3;
319        Address4 = address4;
320        Address5 = address5;
321        Address6 = address6;
322        Address7 = address7;
323        Address8 = address8;
324        Address9 = address9;
325        Address10 = address10;
326        Address11 = address11;
327        Address12 = address12;
328        Address13 = address13;
329        Address14 = address14;
330        ReceiveNewsLetters = receiveNewsLetters;
331    }
332
333    public Person(string personName, string email, string address, string countryID, DateTime dateOfBirth, string gender, bool receiveNewsLetters, string address2, string address3, string address4, string address5, string address6, string address7, string address8, string address9, string address10, string address11, string address12, string address13, string address14, string address15)
334    {
335        PersonID = "c3abddbd-cf80-1ld2-be4-cc7ds758928";
336        PersonName = "Harpreet";
337        Email = "mehbsdale@people.com.cn";
338        DateOfBirth = dateOfBirth;
339        Gender = gender;
340        CountryID = "114629847-905a-4a0e-9abe-88b61655c5cb";
341        Address = "4 Parkside Point";
342        Address2 = address2;
343        Address3 = address3;
344        Address4 = address4;
345        Address5 = address5;
346        Address6 = address6;
347        Address7 = address7;
348        Address8 = address8;
349        Address9 = address9;
350        Address10 = address10;
351        Address11 = address11;
352        Address12 = address12;
353        Address13 = address13;
354        Address14 = address14;
355        Address15 = address15;
356        ReceiveNewsLetters = receiveNewsLetters;
357    }
358
359    public Person(string personName, string email, string address, string countryID, DateTime dateOfBirth, string gender, bool receiveNewsLetters, string address2, string address3, string address4, string address5, string address6, string address7, string address8, string address9, string address10, string address11, string address12, string address13, string address14, string address15, string address16)
360    {
361        PersonID = "c3abddbd-cf80-1ld2-be4-cc7ds758928";
362        PersonName = "Harpreet";
363        Email = "mehbsdale@people.com.cn";
364        DateOfBirth = dateOfBirth;
365        Gender = gender;
366        CountryID = "114629847-905a-4a0e-9abe-88b61655c5cb";
367        Address = "4 Parkside Point";
368        Address2 = address2;
369        Address3 = address3;
370        Address4 = address4;
371        Address5 = address5;
372        Address6 = address6;
373        Address7 = address7;
374        Address8 = address8;
375        Address9 = address9;
376        Address10 = address10;
377        Address11 = address11;
378        Address12 = address12;
379        Address13 = address13;
380        Address14 = address14;
381        Address15 = address15;
382        Address16 = address16;
383        ReceiveNewsLetters = receiveNewsLetters;
384    }
385
386    public Person(string personName, string email, string address, string countryID, DateTime dateOfBirth, string gender, bool receiveNewsLetters, string address2, string address3, string address4, string address5, string address6, string address7, string address8, string address9, string address10, string address11, string address12, string address13, string address14, string address15, string address16, string address17)
387    {
388        PersonID = "c3abddbd-cf80-1ld2-be4-cc7ds758928";
389        PersonName = "Harpreet";
390        Email = "mehbsdale@people.com.cn";
391        DateOfBirth = dateOfBirth;
392        Gender = gender;
393        CountryID = "114629847-905a-4a0e-9abe-88b61655c5cb";
394        Address = "4 Parkside Point";
395        Address2 = address2;
396        Address3 = address3;
397        Address4 = address4;
398        Address5 = address5;
399        Address6 = address6;
400        Address7 = address7;
401        Address8 = address8;
402        Address9 = address9;
403        Address10 = address10;
404        Address11 = address11;
405        Address12 = address12;
406        Address13 = address13;
407        Address14 = address14;
408        Address15 = address15;
409        Address16 = address16;
410        Address17 = address17;
411        ReceiveNewsLetters = receiveNewsLetters;
412    }
413
414    public Person(string personName, string email, string address, string countryID, DateTime dateOfBirth, string gender, bool receiveNewsLetters, string address2, string address3, string address4, string address5, string address6, string address7, string address8, string address9, string address10, string address11, string address12, string address13, string address14, string address15, string address16, string address17, string address18)
415    {
416        PersonID = "c3abddbd-cf80-1ld2-be4-cc7ds758928";
417        PersonName = "Harpreet";
418        Email = "mehbsdale@people.com.cn";
419        DateOfBirth = dateOfBirth;
420        Gender = gender;
421        CountryID = "114629847-905a-4a0e-9abe-88b61655c5cb";
422        Address = "4 Parkside Point";
423        Address2 = address2;
424        Address3 = address3;
425        Address4 = address4;
426        Address5 = address5;
427        Address6 = address6;
428        Address7 = address7;
429        Address8 = address8;
430        Address9 = address9;
431        Address10 = address10;
432        Address11 = address11;
433        Address12 = address12;
434        Address13 = address13;
435        Address14 = address14;
436        Address15 = address15;
437        Address16 = address16;
438        Address17 = address17;
439        Address18 = address18;
440        ReceiveNewsLetters = receiveNewsLetters;
441    }
442
443    public Person(string personName, string email, string address, string countryID, DateTime dateOfBirth, string gender, bool receiveNewsLetters, string address2, string address3, string address4, string address5, string address6, string address7, string address8, string address9, string address10, string address11, string address12, string address13, string address14, string address15, string address16, string address17, string address18, string address19)
444    {
445        PersonID = "c3abddbd-cf80-1ld2-be4-cc7ds758928";
446        PersonName = "Harpreet";
447        Email = "mehbsdale@people.com.cn";
448        DateOfBirth = dateOfBirth;
449        Gender = gender;
450        CountryID = "114629847-905a-4a0e-9abe-88b61655c5cb";
451        Address = "4 Parkside Point";
452        Address2 = address2;
453        Address3 = address3;
454        Address4 = address4;
455        Address5 = address5;
456        Address6 = address6;
457        Address7 = address7;
458        Address8 = address8;
459        Address9 = address9;
460        Address10 = address10;
461        Address11 = address11;
462        Address12 = address12;
463        Address13 = address13;
464        Address14 = address14;
465        Address15 = address15;
466        Address16 = address16;
467        Address17 = address17;
468        Address18 = address18;
469        Address19 = address19;
470        ReceiveNewsLetters = receiveNewsLetters;
471    }
472
473    public Person(string personName, string email, string address, string countryID, DateTime dateOfBirth, string gender, bool receiveNewsLetters, string address2, string address3, string address4, string address5, string address6, string address7, string address8, string address9, string address10, string address11, string address12, string address13, string address14, string address15, string address16, string address17, string address18, string address19, string address20)
474    {
475        PersonID = "c3abddbd-cf80-1ld2-be4-cc7ds758928";
476        PersonName = "Harpreet";
477        Email = "mehbsdale@people.com.cn";
478        DateOfBirth = dateOfBirth;
479        Gender = gender;
480        CountryID = "114629847-905a-4a0e-9abe-88b61655c5cb";
481        Address = "4 Parkside Point";
482        Address2 = address2;
483        Address3 = address3;
484        Address4 = address4;
485        Address5 = address5;
486        Address6 = address6;
487        Address7 = address7;
488        Address8 = address8;
489        Address9 = address9;
490        Address10 = address10;
491        Address11 = address11;
492        Address12 = address12;
493        Address13 = address13;
494        Address14 = address14;
495        Address15 = address15;
496        Address16 = address16;
497        Address17 = address17;
498        Address18 = address18;
499        Address19 = address19;
500        Address20 = address20;
501        ReceiveNewsLetters = receiveNewsLetters;
502    }
503
504    public Person(string personName, string email, string address, string countryID, DateTime dateOfBirth, string gender, bool receiveNewsLetters, string address2, string address3, string address4, string address5, string address6, string address7, string address8, string address9, string address10, string address11, string address12, string address13, string address14, string address15, string address16, string address17, string address18, string address19, string address20, string address21)
505    {
506        PersonID = "c3abddbd-cf80-1ld2-be4-cc7ds758928";
507        PersonName = "Harpreet";
508        Email = "mehbsdale@people.com.cn";
509        DateOfBirth = dateOfBirth;
510        Gender = gender;
511        CountryID = "114629847-905a-4a0e-9abe-88b61655c5cb";
512        Address = "4 Parkside Point";
513        Address2 = address2;
514        Address3 = address3;
515        Address4 = address4;
516        Address5 = address5;
517        Address6 = address6;
518        Address7 = address7;
519        Address8 = address8;
520        Address9 = address9;
521        Address10 = address10;
522        Address11 = address11;
523        Address12 = address12;
524        Address13 = address13;
525        Address14 = address14;
526        Address15 = address15;
527        Address16 = address16;
528        Address17 = address17;
529        Address18 = address18;
530        Address19 = address19;
531        Address20 = address20;
532        Address21 = address21;
533        ReceiveNewsLetters = receiveNewsLetters;
534    }
535
536    public Person(string personName, string email, string address, string countryID, DateTime dateOfBirth, string gender, bool receiveNewsLetters, string address2, string address3, string address4, string address5, string address6, string address7, string address8, string address9, string address10, string address11, string address12, string address13, string address14, string address15, string address16, string address17, string address18, string address19, string address20, string address21, string address22)
537    {
538        PersonID = "c3abddbd-cf80-1ld2-be4-cc7ds758928";
539        PersonName = "Harpreet";
540        Email = "mehbsdale@people.com.cn";
541        DateOfBirth = dateOfBirth;
542        Gender = gender;
543        CountryID = "114629847-905a-4a0e-9abe-88b61655c5cb";
544        Address = "4 Parkside Point";
545        Address2 = address2;
546        Address3 = address3;
547        Address4 = address4;
548        Address5 = address5;
549        Address6 = address6;
550        Address7 = address7;
551        Address8 = address8;
552        Address9 = address9;
553        Address10 = address10;
554        Address11 = address11;
555        Address12 = address12;
556        Address13 = address13;
557        Address14 = address14;
558        Address15 = address15;
559        Address16 = address16;
560        Address17 = address17;
561        Address18 = address18;
562        Address19 = address19;
563        Address20 = address20;
564        Address21 = address21;
565        Address22 = address22;
566        ReceiveNewsLetters = receiveNewsLetters;
567    }
568
569    public Person(string personName, string email, string address, string countryID, DateTime dateOfBirth, string gender, bool receiveNewsLetters, string address2, string address3, string address4, string address5, string address6, string address7, string address8, string address9, string address10, string address11, string address12, string address13, string address14, string address15, string address16, string address17, string address18, string address19, string address20, string address21, string address22, string address23)
570    {
571        PersonID = "c3abddbd-cf80-1ld2-be4-cc7ds758928";
572        PersonName = "Harpreet";
573        Email = "mehbsdale@people.com.cn";
574        DateOfBirth = dateOfBirth;
575        Gender = gender;
576        CountryID = "114629847-905a-4a0e-9abe-88b61655c5cb";
577        Address = "4 Parkside Point";
578        Address2 = address2;
579        Address3 = address3;
580        Address4 = address4;
581        Address5 = address5;
582        Address6 = address6;
583        Address7 = address7;
584        Address8 = address8;
585        Address9 = address9;
586        Address10 = address10;
587        Address11 = address11;
588        Address12 = address12;
589        Address13 = address13;
590        Address14 = address14;
591        Address15 = address15;
592        Address16 = address16;
593        Address17 = address17;
594        Address18 = address18;
595        Address19 = address19;
596        Address20 = address20;
597        Address21 = address21;
598        Address22 = address22;
599        Address23 = address23;
600        ReceiveNewsLetters = receiveNewsLetters;
601    }
602
603    public Person(string personName, string email, string address, string countryID, DateTime dateOfBirth, string gender, bool receiveNewsLetters, string address2, string address3, string address4, string address5, string address6, string address7, string address8, string address9, string address10, string address11, string address12, string address13, string address14, string address15, string address16, string address17, string address18, string address19, string address20, string address21, string address22, string address23, string address24)
604    {
605        PersonID = "c3abddbd-cf80-1ld2-be4-cc7ds758928";
606        PersonName = "Harpreet";
607        Email = "mehbsdale@people.com.cn";
608        DateOfBirth = dateOfBirth;
609        Gender = gender;
610        CountryID = "114629847-905a-4a0e-9abe-88b61655c5cb";
611        Address = "4 Parkside Point";
612        Address2 = address2;
613        Address3 = address3;
614        Address4 = address4;
615        Address5 = address5;
616        Address6 = address6;
617        Address7 = address7;
618        Address8 = address8;
619        Address9 = address9;
620        Address10 = address10;
621        Address11 = address11;
622        Address12 = address12;
623        Address13 = address13;
624        Address14 = address14;
625        Address15 = address15;
626        Address16 = address16;
627        Address17 = address17;
628        Address18 = address18;
629        Address19 = address19;
630        Address20 = address20;
631        Address21 = address21;
632        Address22 = address22;
633        Address23 = address23;
634        Address24 = address24;
635        ReceiveNewsLetters = receiveNewsLetters;
636    }
637
638    public Person(string personName, string email, string address, string countryID, DateTime dateOfBirth, string gender, bool receiveNewsLetters, string address2, string address3, string address4, string address5, string address6, string address7, string address8, string address9, string address10, string address11, string address12, string address13, string address14, string address15, string address16, string address17, string address18, string address19, string address20, string address21, string address22, string address23, string address24, string address25)
639    {
640        PersonID = "c3abddbd-cf80-1ld2-be4-cc7ds758928";
641        PersonName = "Harpreet";
642        Email = "mehbsdale@people.com.cn";
643        DateOfBirth = dateOfBirth;
644        Gender = gender;
645        CountryID = "114629847-905a-4a0e-9abe-88b61655c5cb";
646        Address = "4 Parkside Point";
647        Address2 = address2;
648        Address3 = address3;
649        Address4 = address4;
650        Address5 = address5;
651        Address6 = address6;
652        Address7
```

```

13     protected override void OnModelCreating(ModelBuilder modelBuilder)
14     {
15         base.OnModelCreating(modelBuilder);
16
17         //modelBuilder.Entity<Country>() >> Hey modelBuilder, I am trying to talk about an entity of the 'Country' type.
18         //This method will get the country type of dbSet that is 'Countries' (line no 9)
19         // and that is mapped to the table called 'Countries'
20
21         modelBuilder.Entity<Country>().ToTable("Countries"); //I want the table name to be 'Countries', you can define any other name
22         modelBuilder.Entity<Person>().ToTable("Persons");
23
24         //To add the seed data we have to use this on model creating method in the dbcontext class.
25         //so after you map your model class to the table. here is the place for that.
26
27         //Seed to Countries
28         modelBuilder.Entity<Country>().HasData(new Country() { CountryID = Guid.NewGuid(), CountryName = "Bangladesh" });
29
30         //but there is a more shortcut way that this.
31
32         string countriesJson = System.IO.File.ReadAllText("countries.json");
33         List<Country> countries = System.Text.Json.JsonSerializer.Deserialize<List<Country>>(countriesJson);
34
35         foreach (Country country in countries)
36         {
37             modelBuilder.Entity<Country>().HasData(country);
38         }
39
40         //Seed Persons
41         string personsJson = System.IO.File.ReadAllText("persons.json");
42         List<Person> persons = System.Text.Json.JsonSerializer.Deserialize<List<Person>>(personsJson);
43
44         foreach (Person person in persons)
45         {
46             modelBuilder.Entity<Person>().HasData(person);
47         }
48
49
50     }
51 }

```



Migrations, also called code-first migrations, are commands used to manage database schema changes in Entity Framework. They allow you to:

- Create tables initially.
- Modify existing table structures, such as adding a new column or changing a data type.

Each time you need to make structural changes to the database, you run a migration.

Fortunately, you don't need to write code manually for migrations. Instead, you run a simple command in the Package Manager Console, which generates the necessary C# migration code.

To apply the migration, you execute another command.

Let me demonstrate this practically.

```

protected override void OnModelCreating(ModelBuilder modelBuilder)
{
    base.OnModelCreating(modelBuilder);
    //This method will be called after OnModelCreating()
    //so after you define your entities
    modelBuilder.Entity<Country>()
        .HasData(new Country() { CountryID = Guid.NewGuid(), CountryName = "Bangladesh" });
    //but there is a more shortcut way that this.

    string countriesJson = System.IO.File.ReadAllText("countries.json");
    List<Country> countries = System.Text.Json.JsonSerializer.Deserialize<List<Country>>(countriesJson);

    foreach (Country country in countries)
    {
        modelBuilder.Entity<Country>().HasData(country);
    }

    //Seed Persons
    string personsJson = System.IO.File.ReadAllText("persons.json");
}

```

Make sure the default project is set to 'Entities' where the dbSets are present.

```

PM> Add-Migration Initial

```

```

PM> Add-Migration Initial

```

You can set any name other than 'Initial'

But we are getting error here, because in order to work with migration, we have to install another package.

```

PM> Add-Migration Initial
Add-Migration: The term 'Add-Migration' is not recognized as the name of a cmdlet, function, script file, or operable program. Check the spelling of the name, or if a path was included, verify that the path is correct and try again.
At line:1 char:1
+ Add-Migration Initial
+ ~~~~~
+ CategoryInfo          : ObjectNotFound: (Add-Migration:String) [], CommandNotFoundException
+ FullyQualifiedErrorId : CommandNotFoundException

```

Install-Package Microsoft.EntityFrameworkCore.Tools

```

PM> Install-Package Microsoft.EntityFrameworkCore.Tools
Restoring packages for D:\ASSIGNMENT\Practice\DOT NET CORE\Asp.Net Core - .NET 9. True Ultimate Guide\Section 19 - EntityFrameworkCore [ MVC and Web API]\Entities\Entity
GET https://pkgs.dev.azure.com/CsfeCloud/d1bc3433-4b39-472c-a998-d6b66227b272/_packaging/76aa53e0-7c36-4fed-984a-ba506cad1cb/nuget/v3/flat2/microsoft.entityframeworkcore.tools/index.json
GET https://api.nuget.org/v3-flatcontainer/microsoft.entityframeworkcore.tools/index.json
OK https://api.nuget.org/v3-flatcontainer/microsoft.entityframeworkcore.tools/index.json 26ms
GET https://api.nuget.org/v3-flatcontainer/microsoft.entityframeworkcore.tools/9.0.1/microsoft.entityframeworkcore.tools.9.0.1.nupkg
OK https://api.nuget.org/v3-flatcontainer/microsoft.entityframeworkcore.tools/9.0.1/microsoft.entityframeworkcore.tools.9.0.1.nupkg 60ms
GET https://api.nuget.org/v3-flatcontainer/microsoft.entityframeworkcore.tools/9.0.1/microsoft.entityframeworkcore.tools.9.0.1.nupkg 110ms
GET https://pkgs.dev.azure.com/CsfeCloud/d1bc3433-4b39-472c-a998-d6b66227b272/_packaging/76aa53e0-7c36-4fed-984a-ba506cad1cb/nuget/v3/flat2/microsoft.entityframeworkcore.tools/index.json
OK https://api.nuget.org/v3-flatcontainer/microsoft.entityframeworkcore.design/index.json 26ms
GET https://api.nuget.org/v3-flatcontainer/microsoft.entityframeworkcore.design/9.0.1/microsoft.entityframeworkcore.design.9.0.1.nupkg
OK https://api.nuget.org/v3-flatcontainer/microsoft.entityframeworkcore.design/9.0.1/microsoft.entityframeworkcore.design.9.0.1.nupkg 60ms
NotFound https://pkgs.dev.azure.com/CsfeCloud/d1bc3433-4b39-472c-a998-d6b66227b272/_packaging/76aa53e0-7c36-4fed-984a-ba506cad1cb/nuget/v3/flat2/microsoft.entityframeworkcore.design/index.json
GET https://api.nuget.org/v3-flatcontainer/microsoft.extensions.dependencymodel/index.json
GET https://pkgs.dev.azure.com/CsfeCloud/d1bc3433-4b39-472c-a998-d6b66227b272/_packaging/76aa53e0-7c36-4fed-984a-ba506cad1cb/nuget/v3/flat2/microsoft.entityframeworkcore.design/index.json
GET https://api.nuget.org/v3-flatcontainer/microsoft.extensions.dependencymodel/index.json

```

```

Package Manager Console
Package source: All Default project: CRUDExample
C:\Program Files (x86)\Microsoft SDKs\NuGetPackages\ Executing nuget actions took 1.52 sec
Time Elapsed: 00:00:44.4003468
PM> Add-Migration Initial
Build started...
Build failed.
PM> Install-Package Microsoft.EntityFrameworkCore.Design

```

Select your default project and write this command

Now run this command in your Entities Project.

```

Package Manager Console
Package source: All Default project: Entities
Successfully installed 'Microsoft.Extensions.Caching.Memory 9.0.1' to CRUDExample
Successfully installed 'Microsoft.Extensions.Configuration.Abstractions 9.0.1' to CRUDExample
Successfully installed 'Microsoft.Extensions.DependencyInjection 9.0.1' to CRUDExample
Successfully installed 'Microsoft.Extensions.DependencyInjection.Abstractions 9.0.1' to CRUDExample
Successfully installed 'Microsoft.Extensions.DependencyModel 9.0.1' to CRUDExample
Successfully installed 'Microsoft.Extensions.Logging 9.0.1' to CRUDExample
Successfully installed 'Microsoft.Extensions.Logging.Abstractions 9.0.1' to CRUDExample
Successfully installed 'Microsoft.Extensions.Options 9.0.1' to CRUDExample
Successfully installed 'Microsoft.Extensions.Primitives 9.0.1' to CRUDExample
Successfully installed 'Mono.TextTemplating 3.0.0' to CRUDExample
Successfully installed 'System.CodeDom 6.0.0' to CRUDExample
Successfully installed 'System.Collections.Immutable 7.0.0' to CRUDExample
Successfully installed 'System.Composition 7.0.0' to CRUDExample
Successfully installed 'System.Composition.AttributedModel 7.0.0' to CRUDExample
Successfully installed 'System.Composition.Convention 7.0.0' to CRUDExample
Successfully installed 'System.Composition.Hosting 7.0.0' to CRUDExample
Successfully installed 'System.Composition.Runtime 7.0.0' to CRUDExample
Successfully installed 'System.Composition.TypedParts 7.0.0' to CRUDExample
Successfully installed 'System.Diagnostics.DiagnosticSource 9.0.1' to CRUDExample
Successfully installed 'System.IO.Pipelines 9.0.1' to CRUDExample
Successfully installed 'System.Reflection.Metadata 7.0.0' to CRUDExample
Successfully installed 'System.Text.Encoding.Web 9.0.1' to CRUDExample
Successfully installed 'System.Text.Json 9.0.1' to CRUDExample
Successfully installed 'System.Threading.Channels 7.0.0' to CRUDExample
Executing nuget actions took 177 ms
Time Elapsed: 00:00:02.8273528
PM> Add-Migration Initial

```

We are getting another error. It says no database provider has been configured for this dbContext. You can configure the database provider either sql server or mysql or something else by calling the add dbContext while adding the dbContext class to the services collection.

```

Package Manager Console
Default project: Entities
Package source: All
Successfully installed 'System.CodeDom 6.0.0' to CRUDExample
Successfully installed 'System.Collections.Immutable 7.0.0' to CRUDExample
Successfully installed 'System.Composition 7.0.0' to CRUDExample
Successfully installed 'System.Composition.AttributedModel 7.0.0' to CRUDExample
Successfully installed 'System.Composition.Convention 7.0.0' to CRUDExample
Successfully installed 'System.Composition.Hosting 7.0.0' to CRUDExample
Successfully installed 'System.Composition.Runtime 7.0.0' to CRUDExample
Successfully installed 'System.Composition.TypedParts 7.0.0' to CRUDExample
Successfully installed 'System.Diagnostics.DiagnosticSource 9.0.1' to CRUDExample
Successfully installed 'System.IO.Pipelines 9.0.1' to CRUDExample
Successfully installed 'System.Reflection.Metadata 7.0.0' to CRUDExample
Successfully installed 'System.Text.Encodings.Web 9.0.1' to CRUDExample
Successfully installed 'System.Text.Json 9.0.1' to CRUDExample
Successfully installed 'System.Threading.Channels 7.0.0' to CRUDExample
Executing nugget actions took 177 ms
Time Elapsed: 00:00:02.8273528
PM> Add-Migration Initial
Build started...
Build succeeded.
An error occurred while accessing the Microsoft.Extensions.Hosting services. Continuing without the application service provider. Error: 'AddDbContext' was called with configuration, but the context type 'PersonsDbContext' only declares a parameterless constructor. This means that the configuration passed to 'AddDbContext' will never be used. If configuration is passed to 'AddDbContext', then 'PersonsDbContext' should declare a constructor that accepts a DbContextOptions<PersonsDbContext> and must pass it to the base constructor for DbContext.
Unable to create a 'DbContext' of type 'PersonsDbContext'. The exception 'No database provider has been configured for this DbContext. A provider can be configured by overriding the 'DbContext.OnConfiguring' method or by using 'AddDbContext' on the application service provider. If 'AddDbContext' is used, then also ensure that your DbContext type accepts a DbContextOptions<Context> object in its constructor and passes it to the base constructor for DbContext.' was thrown while attempting to create an instance. For the different patterns supported at design time, see https://go.microsoft.com/fwlink/?linkid=851728
PM>

```

But we have already set the sql server (database provider) and connection string. These options will be supplied to the constructor of the db context class as options parameter.

```

11    builder.Services.AddDbContext<PersonsDbContext>(options =>
12    {
13        //means, hey asp dot net core, we are trying to use asp dot net core for database connection
14        options.UseSqlServer(builder.Configuration.GetConnectionString("DefaultConnection"));
15    });
16}
17
18

```

```

1  using System;
2  using System.Collections.Generic;
3  using Microsoft.EntityFrameworkCore;
4
5  namespace Entities
6  {
7      public class PersonsDbContext : DbContext
8      {
9          public PersonsDbContext(DbContextOptions options) : base(
10             options)
11         {
12         }
13     }
14 }
15
16
17
18

```

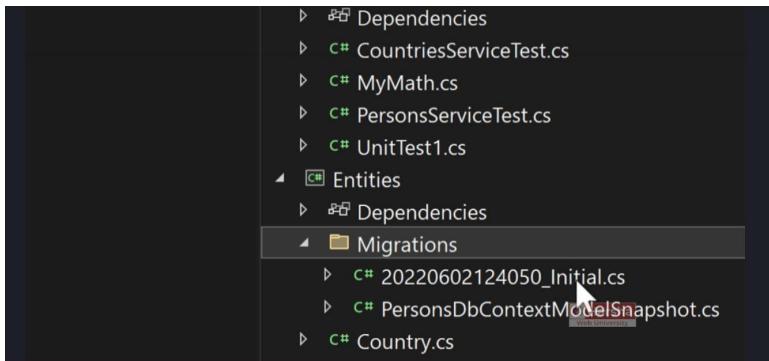
Now build your solution and run this same command.

```

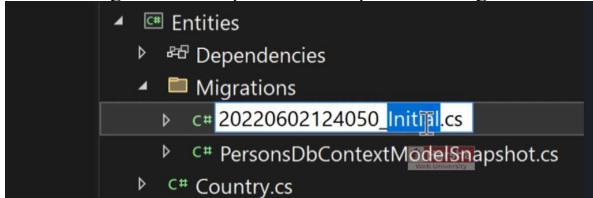
Package Manager Console
Default project: Entities
PM> Add-Migration Initial
An error occurred while accessing the Microsoft.Extensions.Hosting services. Continuing without the application service provider.
'PersonsDbContext' only declares a parameterless constructor. This means that the configuration passed to 'AddDbContext' will never be used.
Should declare a constructor that accepts a DbContextOptions<PersonsDbContext> and must pass it to the base constructor for DbContext.
Unable to create a 'DbContext' of type 'PersonsDbContext'. The exception 'No database provider has been configured for this DbContext. A provider can be configured by overriding the 'DbContext.OnConfiguring' method or by using 'AddDbContext' on the application service provider. If 'AddDbContext' is used, then also ensure that your DbContext type accepts a DbContextOptions<Context> object in its constructor and passes it to the base constructor for DbContext.' was thrown while attempting to create an instance. For the different patterns supported at design time, see https://go.microsoft.com/fwlink/?linkid=851728
PM>

```

Now build is successful. Under Migrations folder, we have got 2 files.



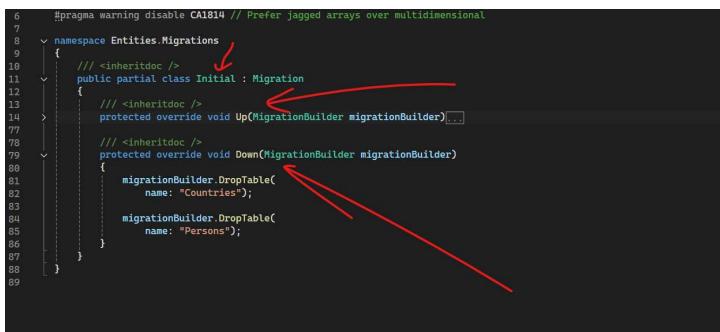
First Porting is timestamp and second portion is migration name that we have given.



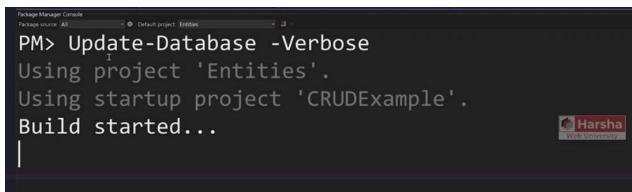
Class name is name as your migration name.

`Up()` method gets executed while creating the database or while making the changes into the existing.

But in case if you rollback the migrations, the `Down()` method gets execute



Now run this command,



To see the table,

```

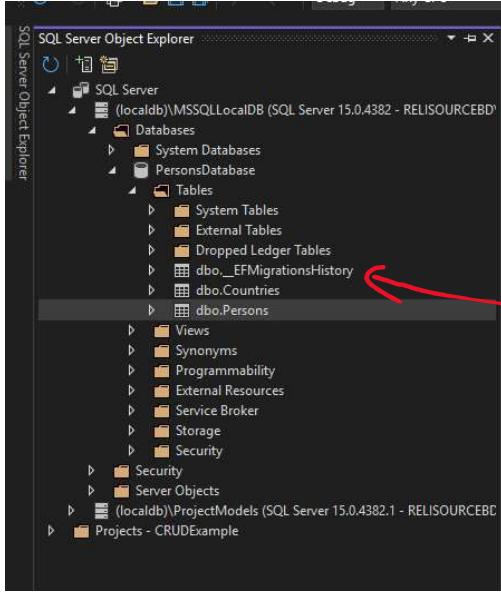
    L = table.Column<string>(type: "nvarchar(40)", maxLength: 40, nullable: true);
    L.Birth = table.Column<DateTime>(type: "datetime2", nullable: true);
    varID = table.Column<string>(type: "uniqueidentifier", nullable: true);
    var = table.Column<string>(type: "nvarchar(200)", maxLength: 200, nullable: true);

    tryID = table.Column<Guid>(type: "uniqueidentifier", nullable: true);

    </NewMigrations>
    <table>>
        <Primarykey>PK_Persons</Primarykey>
        <InsertData>
            <Countries>
                <new>[] { "CountryID", "CountryName" },</new>
                new object[] {
                    Guid("12e15727-d169-4f16-ba94-e5c16886f805"), "China",
                    Guid("28d11936-9466-4a4b-b9c5-2f8a8e0cbde9"), "Philippines",
                    Guid("14629947-905a-4a0e-9ab0-80b165c5d9"),
                    Guid("581c6d33-1bbe-4f51-87fd-2275913c6219"), "China",
                    Guid("56bf4faa-02b3-4693-a0f5-8a95e2218bdcc"), "Thailand",
                    Guid("8f39bedc-47dd-4286-8958-73d8a68e5d41"), "Palestinian Territory"
                }
            </Countries>
            <persons>
                <new>[] { "PersonID", "Address", "CountryID", "DateOfBirth", "Email", "Gender", "PersonName", "ReceiveNewsLetters" },</new>
                new object[] {
                    Guid("912107df-862f-4f16-ba94-e5c16886f805"), "413 Sachtjen Way", new Guid("12e15727-d169-4f16-ba94-e5c16886f805"),
                    Guid("28d11936-9466-4a4b-b9c5-2f8a8e0cbde9"), "2 Warrior Avenue", new Guid("581c6d33-1bbe-4f51-87fd-2275913c6219"),
                    Guid("2933209-63ef-492f-8059-754943c74abf"), "5749 Brown Way", new Guid("12e15727-d169-4f16-ba94-8b13-4693-a0f5-8a95e2218bdcc"),
                    Guid("2a6d3738-9def-43ac-9279-0318edc7ceca"), "97570 Raven Circle", new Guid("8f39bedc-47dd-4286-8958-73d8a68e5d41"),
                    Guid("89e5f445-d89f-4e12-94e0-5ad5b235d704"), "58467 Holy Cross Crossing", new Guid("56bf4faa-02b3-4693-a0f5-8a95e2218bdcc"),
                    Guid("a3b0933b-8a4d-43e9-8699-61e08d8f1a9a"), "9334 Fremont Street", new Guid("581c6d33-1bbe-4f51-87fd-2275913c6219"),
                    Guid("ac660a73-b0b7-4340-abc1-a914257a6189"), "4 Stuart Drive", new Guid("12e15727-d169-4f16-ba94-e5c16886f805"),
                    Guid("c035b4d5-9aeb-4d24-99e6-474304ffce98"), "4 Parkside Point", new Guid("56bf4faa-02b3-4693-a0f5-8a95e2218bdcc")
                }
            </persons>
        </InsertData>
    </table>

```

It stores what migrations are stored so far.



It stores what migrations are stored so far.



EF - Query**SELECT - SQL**

```
SELECT Column1, Column2 FROM TableName
WHERE Column = value
ORDER BY Column
```

LINQ Query:

```
_dbContext.DbSetName
    .Where(item => item.Property == value) //Specifies condition for where clause
    .OrderBy(item => item.Property) //Specifies condition for 'order by' clause
    .Select(item => item); //Expression to be executed for each row
```

EF - Insert**INSERT - SQL**

```
INSERT INTO TableName(Column1, Column2)
VALUES (Value1, Value2)
```

Add:

```
_dbContext.DbSetName.Add(entityObject); //Adds the given model
object (entity object) to the
DbSet.
```

SaveChanges()

```
_dbContext.SaveChanges(); //Generates the SQL INSERT statement
based on the model object data and
executes the same at database server.
```

EF - Update**UPDATE - SQL**

```
UPDATE TableName SET Column1 = Value1,
Column2 = Value2 WHERE PrimaryKey = Value
```

Update:

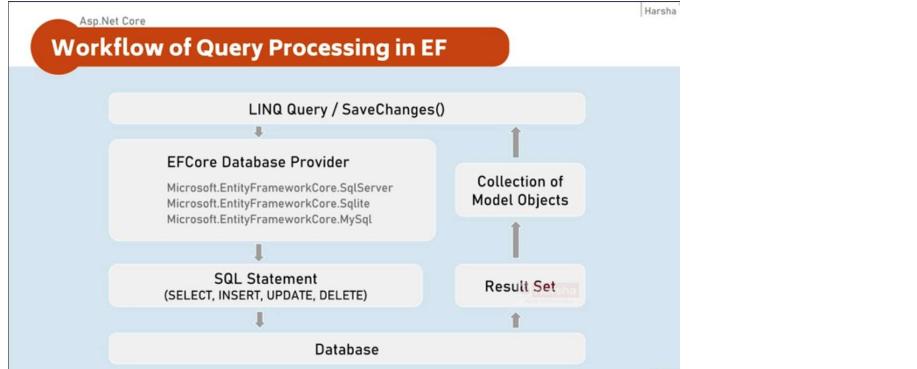
```
entityObject.Property = value; //Updates the specified value in
the specific property of the model
object (entity object) to the DbSet.
```

SaveChanges()

```
_dbContext.SaveChanges(); //Generates the SQL UPDATE statement
based on the model object data and
executes the same at database server.
```

Asp.Net Core

How EF Core Query Works



Asp.Net Core

EF Core - Stored Procedures

If you want to perform multiple or complex database operations, meaning you would like to execute multiple SQL statements at a time with a single database call, then creating a stored procedure in the database is the best choice. With stored procedures, you will **group multiple SQL statements** as one stored unit, and you can call the same from your application source code with a **single database call**.

EF - Calling Stored Procedures

Stored Procedure for CUD (INSERT | UPDATE | DELETE):

```
int DbContext.Database.ExecuteSqlRaw(
    string sql, //Eg: "EXECUTE [dbo].[StoredProcedure] @Param1 @Param2"
    params object[] parameters) //A list of objects of SqlParameter type
```

Stored Procedure for Retrieving (Select):

```
IQueryable<Model> DbSetname.FromSqlRaw(
    string sql, //Eg: "EXECUTE [dbo].[StoredProcedure] @Param1 @Param2"
    params object[] parameters) //A list of objects of SqlParameter type
```

licenses. Follow the package source (feed) URL to determine any dependencies.

Package Manager Console Host Version 6.12.2.1

Type 'get-help NuGet' to see all available NuGet commands.

```
PM> Add-Migration GetPersons_StoredProcedure
Build started...
Build succeeded.
To undo this action, use Remove-Migration.
PM>
```

```
using Microsoft.EntityFrameworkCore.Migrations;
#nullable disable

namespace Entities.Migrations
{
    /// <inheritdoc />
    public partial class GetPersonsStoredProcedure : Migration
    {
        /// <inheritdoc />
        protected override void Up(MigrationBuilder migrationBuilder)
        {

        }

        /// <inheritdoc />
        protected override void Down(MigrationBuilder migrationBuilder)
        {

        }
    }
}
```

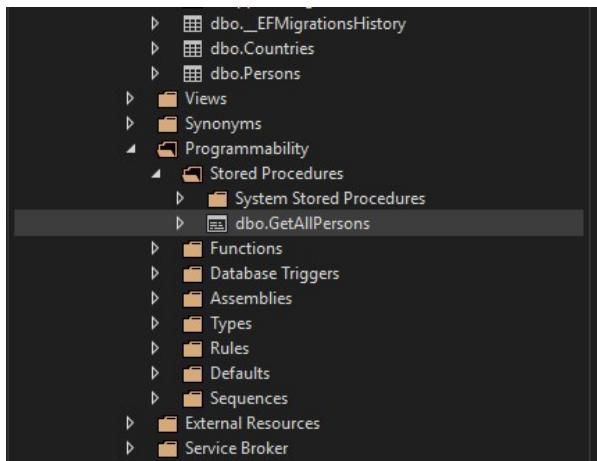
```
    /// <inheritdoc />
    protected override void Up(MigrationBuilder migrationBuilder)
    {
        string sp_GetAllPersons = @"CREATE PROCEDURE [dbo].[GetAllPersons]
AS BEGIN
    SELECT PersonID, PersonName, Email, DateOfBirth, Gender, CountryID, Address, ReceiveNewsLetter
    FROM [dbo].[Persons]
END";
        migrationBuilder.Sql(sp_GetAllPersons);
    }

    /// <inheritdoc />
    protected override void Down(MigrationBuilder migrationBuilder)
    {
        string sp_GetAllPersons = @"DROP PROCEDURE [dbo].[GetAllPersons]";
        migrationBuilder.Sql(sp_GetAllPersons);
    }
}
```

Now run this command.

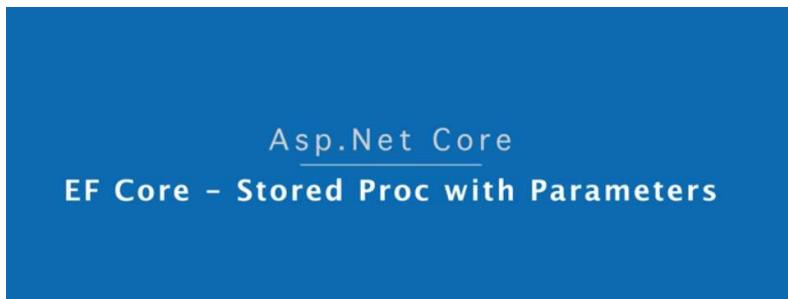
```
To undo this action, use Remove-Migration.
PM> Update-Database
Build started...
Build succeeded.
Microsoft.EntityFrameworkCore.Database.Command[20101]
    Executed DbCommand (10ms) [Parameters=[], CommandType='Text', CommandTimeout='30']
SELECT 1
```

Here you can see your stored procedure.



Now come back here and create your stored procedure method.

```
1  using Microsoft.EntityFrameworkCore;
2  using System;
3  using System.Collections.Generic;
4
5  namespace Entities
6  {
7      public class PersonsDbContext : DbContext
8      {
9          public DbSet<Country> Countries { get; set; }
10         public DbSet<Person> Persons { get; set; }
11
12         public PersonsDbContext(DbContextOptions options) : base(options)
13         {
14         }
15
16         //we have to bind these Obsets to corresponding table
17         protected override void OnModelCreating(ModelBuilder modelBuilder)
18         {
19             base.OnModelCreating(modelBuilder);
20
21             //modelBuilder.Entity<Country>() => Hey modelBuilder, I am trying to talk about an entity of the 'Country' type.
22             //This method will get the country type of dBSet that is 'Countries' (line no 9)
23             // and that is mapped to the table called 'Countries'
24
25             modelBuilder.Entity<Country>().ToTable("Countries"); //I want the table name to be 'Countries'. you can define any other name.
26             modelBuilder.Entity<Person>().ToTable("Persons");
27
28             //To add the seed data we have to use this on model creating method in the dbContext class.
29             //so after you map your model class to the table. here is the place for that.
30
31             //Seed to Countries
32             modelBuilder.Entity<Country>().HasData(new Country() { CountryID = Guid.NewGuid(), CountryName = "Bangladesh" });
33
34             //but there is a more shortcut way that this.
35
36             string countriesJson = System.IO.File.ReadAllText("countries.json");
37             List<Country> countries = System.Text.Json.JsonSerializer.Deserialize<List<Country>>(countriesJson);
38             modelBuilder.Entity<Country>().HasData(countries);
39         }
40     }
41 }
```



How can you pass parameter to the stored procedure?

Asp.Net Core | Harsha

EF - Calling Stored Procedures

Stored Procedure for CUD (INSERT | UPDATE | DELETE):

```
int DbContext.Database.ExecuteSqlRaw(
    string sql, //Eg: "EXECUTE [dbo].[StoredProcedure] @Param1 @Parm2"
    params object[] parameters) //A list of objects of SqlParameter type
```

Stored Procedure for Retrieving (Select):

```
IQueryable<Model> DbSetname.FromSqlRaw(
    string sql, //Eg: "EXECUTE [dbo].[StoredProcedure] @Param1 @Parm2"
    params object[] parameters) //A list of objects of SqlParameter type
```

Instead of adding like this, let's use stored procedure to insert data through stored procedure.

```
}
//Model validation
ValidationHelper.ModelValidation(personAddRequest);

//convert personAddRequest into Person type
Person person = personAddRequest.ToPerson();

//generate PersonID
person.PersonID = Guid.NewGuid();

//add person object to persons list
_db.Pessoas.Add(person);
_db.SaveChanges();

//convert the Person object into PersonResponse type
return ConvertPersonToPersonResponse(person);
```

Create a migration file

```
33     ("persons.json");
34     List<Person> persons =
35         System.Text.Json.JsonSerializer.Deserialize<List<Person>>(
36             personsJson);
37
38 }
```

PM> Add-Migration InsertPerson_StoredProcedure
Build started...

Asp.Net Core | Harsha

Creating Stored Procedure (SQL Server)

```
CREATE PROCEDURE [schema].[procedure_name]
(@parameter_name data_type, @parameter_name data_type)
AS BEGIN
    statements
END
```

Use the same data type

```
System databases
  > PersonsDatabase
    > Tables
      > System Tables
      > External Tables
    > Dropped Ledger Tables
    > Dropped Tables
    > Drop Table History
    > db_Countries
  > dbo_Pessoas
    > Columns
      > PersonID PK unique
        > PersonName (nvarchar)
        > Email (nvarchar(40))
        > DateOfBirth (date)
        > Gender (nchar(1))
        > CountryID (uniqueid)
        > CountryID (uniqueid)
        > Address (nvarchar(50))
        > ReceiveNewsLetters (bit)
    > Constraints
    > Triggers
  > Initial doc
    > InsertPersonStoredProcedure : Migration
      > inheritdoc
      > ted override void Up(MigrationBuilder migrationBuilder)
        > ring sp_GetAllPersons = @"
          > CREATE PROCEDURE [dbo].[InsertPerson]
            > (@Person)
            > AS BEGIN
              >     SELECT PersonID, PersonName, Email, DateOfBirth, Gender, CountryID, Address, ReceiveNewsLetters
              >     FROM [dbo].[Persons]
            > END
          >     &gt; class System.String
            >     Represents text as a sequence of UTF-16 code units.
      >     &gt; migrationBuilder.Sql(sp_GetAllPersons);
```

```

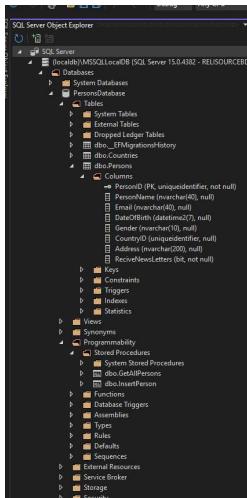
using Microsoft.EntityFrameworkCore.Migrations;
namespace Entities.Migrations
{
    /// <inheritdoc />
    public partial class InsertPersonStoredProc : Migration
    {
        /// <inheritdoc />
        protected override void Up(MigrationBuilder migrationBuilder)
        {
            string sp_InsertPerson = @"
                CREATE PROCEDURE [dbo].[InsertPerson]
                (@PersonID uniqueidentifier, @PersonName nvarchar(40),
                @Email nvarchar(40), @DateOfBirth datetime2(7), @Gender nvarchar(10),
                @CountryID uniqueidentifier, @Address nvarchar(200), @RecieveNewsLetters bit)
                AS BEGIN
                    INSERT INTO [dbo].[Persons](PersonID,PersonName,Email,DateOfBirth,Gender,CountryID,Address,RecieveNewsLetters) VALUES (
                    @PersonID,@PersonName,@Email,@DateOfBirth,@Gender,@CountryID,@Address,@RecieveNewsLetters)
                END
            ";
            migrationBuilder.Sql(sp_InsertPerson);
        }

        /// <inheritdoc />
        protected override void Down(MigrationBuilder migrationBuilder)
        {
            string sp_InsertPerson = @"
                DROP PROCEDURE [dbo].[InsertPerson];
            ";
            migrationBuilder.Sql(sp_InsertPerson);
        }
    }
}

```

Now go to package manager console and run this command.

Update-Database



At your practice time, you can try update and delete procedure.

Asp Net Core | Harsha

Advantages of Stored Procedure

Single database call

You can execute multiple / complex SQL statements with a single database call.

As a result, you'll get:

- Better performance (as you reduce the number of database calls)
- Complex database operations such as using temporary tables / cursors becomes easier.

Maintainability

The SQL statements can be changed easily WITHOUT modifying anything in the application source code (as long as inputs and outputs doesn't change)

Asp.Net Core

Changes in Table Structure

Once the table has been created, how do you make changes to the existing database table?
How do you change the table structure?

For example, you may want to rename a column, change its data type, or add some constraints.
This is exactly what we are going to perform in the upcoming lectures.

In this lecture, let us try to learn how to add or remove columns.
For example, in the **Person** table, these are all the existing columns.

Now, I would like to add an additional column.
In the **Person** model class, we are going to add a new column, for example, **TIN** (Tax Identification Number).

Assume it is an alphanumerical value, so we are using the **string** data type, and of course, it is nullable, meaning the values can be **null** in the database table.
Since we have added this new column directly in the model class, these changes should be reflected in the actual database table.

But it's not magic—it doesn't happen automatically.
See, in the **Persons** table, if we refresh, there are no changes.
So, we have to apply **migrations** in order to create that column in the table.
But here, instead of writing the actual SQL script to add the column, we are letting the **migrations system** generate it for us.

```
1  using System.ComponentModel.DataAnnotations;
2
3  namespace Entities
4  {
5      /// <summary>
6      /// Domain Model for Country
7      /// </summary>
8      public class Country
9      {
10         [Key]
11         public Guid CountryID { get; set; }
12
13         public string? CountryName { get; set; }
14     }
15 }
16
```

```
5  {
6      /// <summary>
7      /// Person domain model class
8      /// </summary>
9      public class Person
10     {
11         [Key]
12         public Guid PersonID { get; set; }
13
14         [StringLength(40)] //nvarchar(40)
15         public string? PersonName { get; set; }
16
17         [StringLength(40)] //nvarchar(40)
18         public string? Email { get; set; }

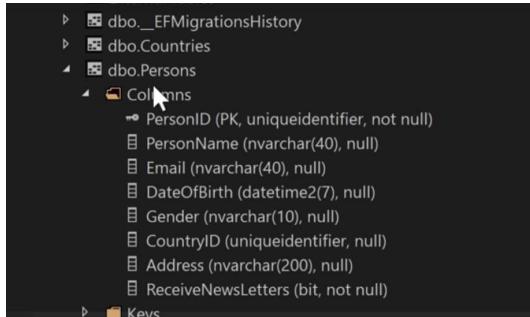
```

```

23     public string? Gender { get; set; }
24
25     //uniqueidentifier
26     public Guid? CountryID { get; set; }
27
28     [StringLength(200)] //nvarchar(200)
29     public string? Address { get; set; }
30
31     //bit
32     public bool ReceiveNewsLetters { get; set; }
33
34     public string? TIN { get; set; }
35
36 }
37

```

Newly Added



Now go to your package manager console.

```

    INSERT INTO [__EFMigrationsHistory]
    [ProductVersion])
    VALUES (N'20220611123233_Inse
N'6.0.5');
Done.
PM> Add-Migration TINColumn

```

```

using Microsoft.EntityFrameworkCore.Migrations;
#nullable disable
namespace Entities.Migrations
{
    /// <inheritdoc />
    public partial class TINColumn : Migration
    {
        /// <inheritdoc />
        protected override void Up(MigrationBuilder migrationBuilder)
        {
            migrationBuilder.AddColumn<string>(
                name: "TIN",
                table: "Persons",
                type: "nvarchar(max)",
                nullable: true);

            migrationBuilder.UpdateData(
                table: "Persons",
                keyColumn: "PersonID",
                keyValue: new Guid("012107df-862f-4f16-ba94-e5c16886f005"),
                column: "TIN",
                value: null);

            migrationBuilder.UpdateData(
                table: "Persons",
                keyColumn: "PersonID",
                keyValue: new Guid("28d11936-9466-4a4b-b9c5-2f0a8e0cbde9"),
                column: "TIN",
                value: null);

            migrationBuilder.UpdateData(
                table: "Persons",
                keyColumn: "PersonID",
                keyValue: new Guid("29339209-63f5-492f-8459-754943c74abf"),
                column: "TIN",
                value: null);
        }

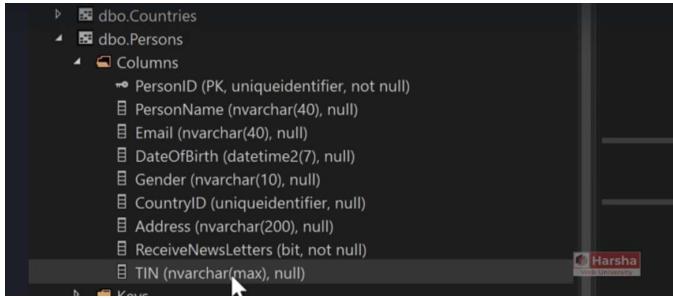
        protected override void Down(MigrationBuilder migrationBuilder)
        {
            migrationBuilder.DropColumn("TIN");
        }
    }
}

```

```

To undo this action, use Remove-Migration.
PM> Update-Database
Build started...
Build succeeded.

```

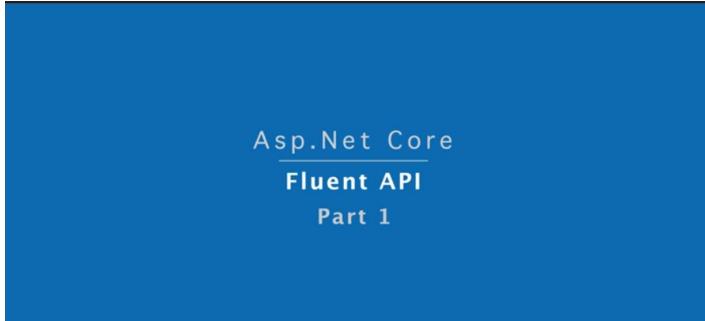


Now, your assignment is to create a new migration to set default value for TIN

CountryID	Address	ReceiveNe...	TIN
8f30bedc-4...	97570 Rave...	False	NULL
501c6d33-...	2 Warrior A...	True	NULL
56bf146a4-0...	4 Parkside ...	False	NULL
14629847-...	73 Heath A...	True	NULL
56bf146a4-0...	50467 Holy...	False	NULL
501c6d33-...	9334 Frem...	True	NULL
12e15727-...	57449 Bro...	True	NULL
8f30bedc-4...	484 Claren...	False	NULL
12e15727-...	4 Stuart Dri...	True	NULL
14629847-...	6 Mornings...	False	NULL
14629847-...	dfs	True	NULL
12e15727-...	413 Sachtje...	True	NULL
12e15727-...	83187 Merr...	True	NULL
NULL	NULL	NULL	NULL

Now run our application, we have got an error. Because in previous SP, we have not set it.

```
3  #nullable disable
4
5  namespace Entities.Migrations
6  {
7      public partial class GetPersons_StoredProcedure : Migration
8      {
9          protected override void Up(MigrationBuilder migrationBuilder)
10         {
11             string sp_GetAllPersons = @""
12             CREATE PROCEDURE [dbo].[GetAllPersons]
13             AS BEGIN
14                 SELECT PersonID, PersonName, Email, DateOfBirth, Gender,
15                     CountryID, Address, ReceiveNewsLetters FROM [dbo].[Persons]
16             END
17         }
18     }
19 }
```



```

7   ///<summary>
8   ///</summary>
9   public class Person
10  {
11      [Key]
12      public Guid PersonID { get; set; }
13
14      //default data type in database is : nvarchar(max)
15
16      [StringLength(40)] //nvarchar(40)
17      public string? PersonName { get; set; }
18
19      [StringLength(40)]
20      public string? Email { get; set; }
21
22      public DateTime? DateOfBirth { get; set; }
23
24      [StringLength(10)]
25      public string? Gender { get; set; }
26
27      //data type: uniqueidentifier
28      //It's a Foreign key, that means it refer to the Country Table.
29      public Guid? CountryID { get; set; }
30
31      [StringLength(200)]
32      public string? Address { get; set; }
33
34      //data type: bit (no need to mention any other data type)
35      public bool ReceiveNewsLetters { get; set; }
36
37      public string? TIN { get; set; }
38
39  }
40

```

The screenshot shows the 'dbo.Persons' table in the 'dbo' schema. The 'Columns' section lists several columns: PersonID (PK, uniqueidentifier, not null), PersonName (nvarchar(40), null), Email (nvarchar(40), null), DateOfBirth (datetime2(7), null), Gender (nvarchar(10), null), CountryID (uniqueidentifier, null), Address (nvarchar(200), null), ReceiveNewsLetters (bit, not null), and TIN (nvarchar(max), null). A red arrow points to the 'TIN' column.

Okay, in the last lecture, we added a new column called **TIN** in the **Persons** table.
First, we added a property called **TIN** in the **Person** class.

Then, we ran the **Add-Migration** and **Update-Database** commands in the **Package Manager Console**.
This generated the migration and added that column to the **Persons** table.

Now, if you check **SQL Server Object Explorer**, expand **Databases**, and navigate to the **Persons** table, you will see that it has the **TIN** column.

However, we have a big complaint about its **data type**.
It is **nvarchar(max)**, which means it supports the maximum number of characters allowed—up to **2 billion characters** in a single string.

Moreover, **nvarchar** is a Unicode data type, meaning it supports storing **alphabetical, numerical, and special characters** from any language supported by Unicode.

While this sounds good, it has a **negative impact on database performance** because it dramatically **increases the database size**.
Each character in **nvarchar** occupies **two bytes** instead of one byte (as in ASCII).

Therefore, you should **not** use the **nvarchar** data type unless it's absolutely necessary.
For example, in the case of a **person's name**, you might require **nvarchar** because names can contain **non-numerical characters**, such as those from different languages.

But in this case, the **TIN number** consists only of **alphabets and digits**, so it is **not advisable** to use **nvarchar**.
Now, the question is:

How can we control the SQL Server data type from Entity Framework?
How do we specify the exact data type for a specific column in the database?
This is where the **Fluent API** comes into the picture.

The **Fluent API** is a set of predefined methods in **Entity Framework** that allows you to specify more details about **column data types** and table structures.
This includes:

- **Table name**
- **Column names**
- **Data types**
- **Constraints**

You can use the **Fluent API** inside the **OnModelCreating** method of the **DbContext** class.

In fact, we are already using it with the **ToTable** method to specify the table name that is mapped to a specific **model class**.

Asp.Net Core | Harsha

DbContext class

```
public class CustomDbContext : DbContext
{
    protected override void OnModelCreating(ModelBuilder modelBuilder)
    {
        //Specify table name (and schema name optionally) to be mapped to the model class
        modelBuilder.Entity<ModelClass>().ToTable("table_name", schema: "schema_name");

        //Specify view name (and schema name optionally) to be mapped to the model class
        modelBuilder.Entity<ModelClass>().ToView("view_name", schema: "schema_name");

        //Specify default schema name applicable for all tables in the DbContext
        modelBuilder.HasDefaultSchema("schema_name");
    }
}
```

Asp.Net Core | Harsha

DbContext class

```
public class CustomDbContext : DbContext
{
    protected override void OnModelCreating(ModelBuilder modelBuilder)
    {
        modelBuilder.Entity<ModelClass>()
            .Property(temp => temp.PropertyName)
            .HasColumnName("column_name") //Specifies column name in table
            .HasColumnType("data_type") //Specifies column data type in table
            .HasDefaultValue("default_value") //Specifies default value of the column
    }
}
```

Go to your PersonsDbContext class.

```
35
36     foreach (Person person in persons)
37         modelBuilder.Entity<Person>().HasData(person);
38
39
40     //Fluent API
41     modelBuilder.Entity<Person>()
42         .Property(temp => temp.TIN)
43             .HasColumnName("TaxIdentificationNumber")
44             .HasColumnType("varchar(8)")
45             .HasDefaultValue("ABC12345");
46
```

Let's add migration.

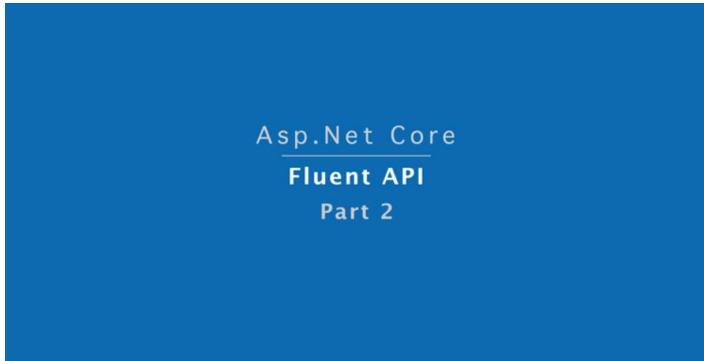
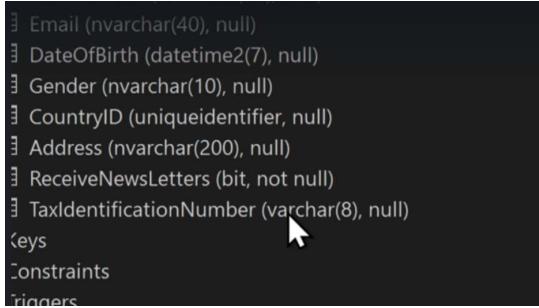
```
42     .HasColumnName("TaxIdentificationNumber")
43     .HasColumnType("varchar(8)")

available NuGet commands.

PM> Add-Migration TIN_Updated
```

Now run this command.

```
igration.
M> Update-Database
```



A screenshot of a code editor showing a `DbContext` class named `CustomDbContext`:public class CustomDbContext : DbContext
{
 protected override void OnModelCreating(ModelBuilder modelBuilder)
 {
 //Adds database index for the specified column for faster searches
 modelBuilder.Entity<ModelClass>().HasIndex("column_name").IsUnique();

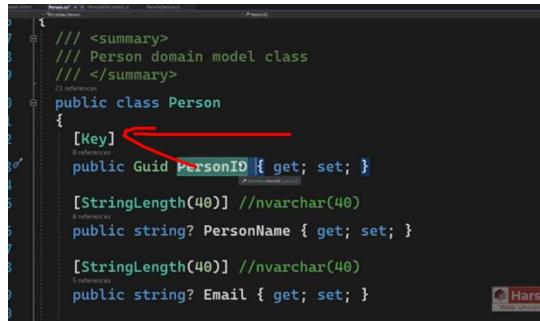
 //Adds check constraint for the specified column - that executes for insert & update
 modelBuilder.Entity<ModelClass>().HasCheckConstraint("constraint_name", "condition");
 }
}

To add **constraints** to a database table using **Entity Framework Fluent API**, you can define them inside the **OnModelCreating** method of the **DbContext** class.

Common Constraints in Fluent API:

1. Primary Key
2. Foreign Key
3. Unique Constraint
4. Required (NOT NULL)
5. Default Values
6. Column Length and Type

See, the first constraint is the primary key constraint, but we have already mentioned the same by using the key attribute for the person ID, so this is your primary key. That means it is not **NULL + UNIQUE**



But to add the not null constraint, we will add the required attribute. For example, if you add a required attribute for a particular column, that means it has a not null constraint. Okay, we don't want to add it right now.

Asp.Net Core 9 (NET 9) | True Ultimate Guide

```

11 {
12     [Key]
13     public Guid PersonID { get; set; }
14
15     [StringLength(40)] //nvarchar(40)
16     [Required] ←
17     public string? PersonName { get; set; }
18
19     [StringLength(40)] //nvarchar(40)
20     public string? Email { get; set; }
21
22     [Required]
23     public DateTime? DateOfBirth { get; set; }
24
25     [StringLength(10)] //nvarchar(10)
26     public string? Gender { get; set; }

```

But how to add the unique constraint and check constraint we can do that through Fluent API.

EF - Fluent API

DbContext class

```

public class CustomDbContext : DbContext
{
    protected override void OnModelCreating(ModelBuilder modelBuilder)
    {
        //Adds database index for the specified column for faster searches
        modelBuilder.Entity<ModelClass>().HasIndex("column_name").IsUnique();

        //Adds check constraint for the specified column - that executes for insert & update
        modelBuilder.Entity<ModelClass>().HasCheckConstraint("constraint_name", "condition");
    }
}

```

In the "PersonsDbContext"

```

//Seed to Countries
modelBuilder.Entity<Country>().HasData(new Country() { CountryID = Guid.NewGuid(), CountryName = "Bangladesh" });

//but there is a more shortcut way that this.

string countriesJson = System.IO.File.ReadAllText("countries.json");
List<Country> countries = System.Text.Json.JsonSerializer.Deserialize<List<Country>>(countriesJson);

foreach (Country country in countries)
{
    modelBuilder.Entity<Country>().HasData(country);
}

//Seed Persons
string personsJson = System.IO.File.ReadAllText("persons.json");
List<Person> persons = System.Text.Json.JsonSerializer.Deserialize<List<Person>>(personsJson);

foreach (Person person in persons)
{
    modelBuilder.Entity<Person>().HasData(person);
}

//Fluent API
modelBuilder.Entity<Person>().Property(temp => temp.TIN)
    .HasColumnName("TaxIdentificationNumber") //instead of TIN, this will be the name of the database table column
    .HasColumnType("varchar(8)")
    .HasDefaultValue("ABC12345");

//modelBuilder.Entity<Person>().HasIndex(temp => temp.TIN).IsUnique(); //no duplicates values allowed on TIN column
modelBuilder.Entity<Person>()
    .HasCheckConstraint("CHK_TIN", "[TIN] = 8"); // this condition will be checked for insertion as well as update
}

```

Now run this command in the package manager console.

```

Type 'get-help NuGet' to see all available NuGet commands.

PM> Add-Migration TIN_Updated_CHK
Build started...
Build succeeded.
Your target project 'CRUDExample' doesn't match your migrations assembly 'Entities'. Either change your targ
Change your migrations assembly by using DbContextOptionsBuilder. E.g. options.UseSqlServer(connection, b =>
the DbContext.
Change your target project to the migrations project by using the Package Manager Console's Default project.
PM> Add-Migration TIN_Updated_CHK
Build started...
Build succeeded.
To undo this action, use Remove-Migration.
PM> Update-Database

```

Asp .Net Core

Table Relations with EF

Asp .Net Core | Harsha

EF - Table Relations with Fluent API

Primary / Master / Parent Table

CategoryID	CategoryName
1	Category1
2	Category2

Related / Detailed / Child Table

ProductID	ProductName	CategoryID
1	Product1	1
2	Product2	1
3	Product3	2

Diagram illustrating the many-to-many relationship between the Primary Table (Categories) and the Related Table (Products). CategoryID 1 is associated with ProductID 1 and 2. CategoryID 2 is associated with ProductID 3.

Asp .Net Core | Harsha

EF - Table Relations with Navigation Properties

Master Model class

```
public class MasterModel
{
    public data_type PropertyName { get; set; }
    public virtual ICollection<ChildModel> ChildPropertyName { get; set; }
}
```

Child Model class

```
public class ChildModel
{
    public data_type PropertyName { get; set; }
    public virtual ParentModel ParentPropertyName { get; set; }
}
```

Diagram illustrating navigation properties in Entity Framework. A dashed arrow points from the Child Model class to the Master Model class, indicating a one-to-many relationship.

SQL Server referential integrity specifies how you separate the data between the primary table and detailed table. In the primary table or master table, you will have the primary key, which contains all available values of the particular column. For example, there are two tables like Categories and Products. The category_id column of the Categories table contains all available categories, so these values are referenced in the foreign key property of the Products table.

For example, for this product, category_id equals 1. That means, to get the category name of this, you have to search for the matching category_id value in the Categories table. There, you can find the name.

In that way, you have to match the foreign key value with the primary key value.

Okay, this is the general database-specific rule.

But how do you configure the primary key and foreign key?

In the same way, how can you access the related table data in Entity Framework?

That's what we are trying to learn in this lecture.

In Entity Framework, you can make use of navigation properties instead of joins.

For example, in the child model class, you can create a parent property that refers to the parent record.

For example, this is the Person model class.

So, it can have a property called Country, which refers to the corresponding country object—that is, an object of the Country model class.

In the same way, every country can refer to a set of persons. So, all the persons that come under a particular country can be referred to by this collection-type property, which is of ICollection type.

Here, ICollection is the parent interface for List.

Optionally, you can make it virtual so that you can override it in the child classes.
Let me demonstrate the same.

```

31 >     public PersonResponse AddPerson(PersonAddRequest? personAddRequest) ...
58
59     public List<PersonResponse> GetAllPersons()
60     {
61         //it is not possible to call your own method in the link to entities expression
62         //return _db.People.Select(person => ConvertPersonToPersonResponse(person)).ToList();
63
64         //return _db.People.ToList() //SELECT * from Persons
65         //    .Select(person => ConvertPersonToPersonResponse(person)).ToList();
66
67         var persons = _db.People.Include("Country").ToList();
68
69         return _db.sp_GetAllPersons() < 1.542ms elapsed
70             .Select(person => ConvertPersonToPersonResponse(person)).ToList();
71     }
72
73     public PersonResponse? GetPersonByPersonId(Guid? personID)

```

//SELECT * from Persons
var persons = _db.People

```

Project Build Debug Test Analyze Tools Extensions Window Help Search CRUDExample
Debug Any CPU CRUDExample http D C S W P M F G R L I O
Person.cs Person.cs
//default data type in database is : nvarchar(max)
[StringLength(40)] //nvarchar(40)
public string? PersonName { get; set; }

[StringLength(40)]
public string? Email { get; set; }

public DateTime? DateOfBirth { get; set; }

[StringLength(10)]
public string? Gender { get; set; }

//data type: uniqueidentifier
//It's a foreign key, that means it refer to the Country Table.
public Guid? CountryID { get; set; }

[StringLength(200)]
public string? Address { get; set; }

//data type: bit (no need to mention any other data type)
public bool ReceiveNewsLetters { get; set; }

public string? TIN { get; set; }

[ForeignKey("CountryID")]
public virtual Country? Country { get; set; } //virtual so child class can override that, not mandatory.
}

```

There's another way to configure the table relations through fluent API.

Asp.Net Core | Harsha

EF - Table Relations with Fluent API

DbContext class

```

public class CustomDbContext : DbContext
{
    protected override void OnModelCreating(ModelBuilder modelBuilder)
    {
        //Specifies relation between primary key and foreign key among two tables
        modelBuilder.Entity<ChildModel>()
            .HasOne<ParentModel>(parent => parent.ParentReferencePropertyInChildModel)
            .WithMany(child => child.ChildReferencePropertyInParentModel) //optional
            .HasForeignKey(child => child.ForeignKeyPropertyInChildModel)
    }
}

```

Asp.Net Core Async EF Core Operations

The asynchronous method execution can be paused by using the await keyword based on either I/O-bound code or CPU-bound code. That means, as soon as the particular task has been completed, it can resume the execution of the asynchronous method.

Meanwhile, the same thread can execute some other method or handle another request.

In that way, you can ensure the maximum usage of CPU capacity.

First of all, you will create an asynchronous method by using a single keyword, and it can use one or more await keywords wherever you would like to wait for a response from a particular method.

For example, reading data from a file or database, or executing a large task.

Wherever you would like to wait for a particular method execution, you can use the await keyword.

Upon using the await keyword, the current method execution can be paused until we get the proper response from the task.

This is how asynchronous methods work in general in C#.

We have already used async and await keywords several times.

For example, we have used them in middleware.

Particularly in this lecture, we are going to use the async and await keywords in service methods as well as controller action methods.

In real-world projects, it is always advisable to maintain service methods and controller action methods as asynchronous methods.

Because most I/O operations, particularly database interactions, should be done with asynchronous operations.

This means you should always use async and await while interacting with files, databases, or any other external resources such as communicating with email servers, etc.

EF - Async Operations

| Harsha

async

- The method is awaitable.
- Can execute **I/O bound** code or **CPU-bound code**

await

- Waits for the I/O bound or CPU-bound code execution gets completed.
- After completion, it returns the return value.

Methods

Asp.Net Core Async Controller Action Methods

Asp.Net Core Async Unit Test Methods

Asp.Net Core Generate PDF Files

Suppose one of the client requirements is that you have to generate a PDF file programmatically.

For example, the client asks for a report or the data in the form of a PDF file.

For a real-world example, let's say there is a ticket booking application, and the ticket should be generated as a PDF for download. Such requirements can be satisfied using a NuGet package called **Rotativa**.

It is one of the third-party packages but is pretty useful and commonly used for generating PDF files in ASP.NET Core.

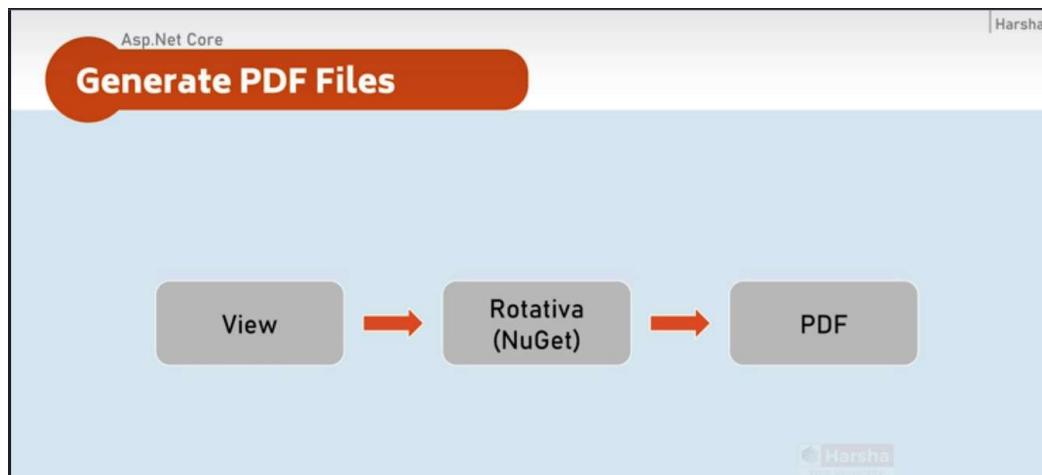
All you need to do is create a view that contains the content.

That view should include all the content that needs to be in the PDF file.

Whatever information should be in the PDF file, you can write it in the view.

Then, you will use this NuGet package called **Rotativa** to convert the view into a PDF.

Let me demonstrate the same.

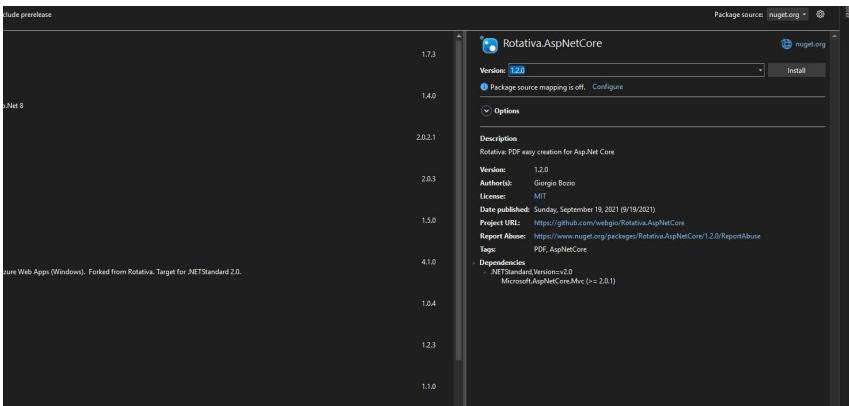


Person Name	Email	Date of Birth	Age	Gender	Country	Address	Receive News Letters	Options
Angie	asavar3@dropbox.com	09 Jan 1987	35	Male	China	83187 Merry Drive	True	Edit Delete
Franchot	fbowsher2@howstuffworks.com	10 Feb 1995	27	Male	Philippines	73 Heath Avenue	True	Edit Delete
Hanslain	hmoscoba@tripod.com	20 Sep 1990	32	Male	China	413 Sachetjen Way	True	Edit Delete
Lombard	lwoodwing9@wix.com	25 Sep 1997	25	Male	Palestinian Territory	484 Clarendon Court	False	Edit Delete
Maddy	mjarelli@wisc.edu	16 Feb 1983	39	Male	China	57449 Brown Way	True	Edit Delete
Marguerite	mwebsdale0@people.com.cn	28 Aug 1989	33	Female	Thailand	4 Parkside Point	False	Edit Delete
Minty	mconachya@va.gov	24 May 1990	32	Female	China	2 Warrior Avenue	True	Edit Delete
Mitchael	mingfoot5@netvibes.com	04 Jan 1988	35	Male	Palestinian Territory	97570 Raven Circle	False	Edit Delete

```

1  @model IEnumerable<PersonResponse>
2
3  @{
4      Layout = null;
5  }
6
7  /* we have to explicitly import the css file, because we are not using any pre-defined layout file
8
9  <h1>Persons</h1>
10 <link href="@("http://" + Context.Request.Host.ToString() + "/StyleSheet.css") rel="stylesheet" />
11
12 <table class="table w-100 mt">
13     <thead>
14         <tr>
15             <th>Person Name</th>
16             <th>Email</th>
17             <th>Date of Birth</th>
18             <th>Age</th>
19             <th>Gender</th>
20             <th>Country</th>
21             <th>Address</th>
22             <th>Receive News Letters</th>
23         </tr>
24     </thead>
25     <tbody>
26         @foreach (PersonResponse person in Model)
27         {
28             <tr>
29                 <td style="width:15%">@person.PersonName</td>
30                 <td style="width:20%">@person.Email</td>
31                 <td style="width:15%">@person.DateOfBirth</td>
32                 <td style="width:15%">@person.Age</td>
33                 <td style="width:15%">@person.Gender</td>
34                 <td style="width:15%">@person.Country</td>
35                 <td style="width:15%">@person.Address</td>
36                 <td style="width:15%">@person.ReceiveNewsLetters</td>
37             </tr>
38         }
39     </tbody>
40 </table>

```



```

[Route("PersonsPDF")]
public async Task<IActionResult> PersonsPDF()
{
    //Get list of persons
    List<PersonResponse> persons = await _personsService.GetAllPersons();

    //Return view as pdf

    //by default, every controller action method contains some 'ViewData'. We have to supply the same.
    return new ViewAsPdf("PersonsPDF", persons, ViewData)
    {
        PageMargins = new Rotativa.AspNetCore.Options.Margins()
        {
            Top = 20,
            Right = 20,
            Bottom = 20,
            Left = 20
        },
        PageOrientation = Rotativa.AspNetCore.Options.Orientation.Landscape
    };
}

```

```

    er.Configuration.GetConnectionString("DefaultConnection");

localDB;Initial Catalog=PersonsDatabase;Integrated Security=True;Connect Timeout=30;
ans, we are using windows authentication for sql server. that means you do

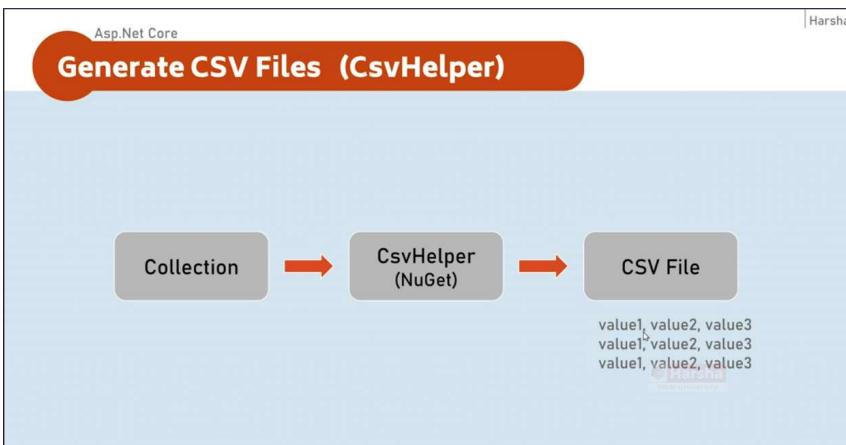
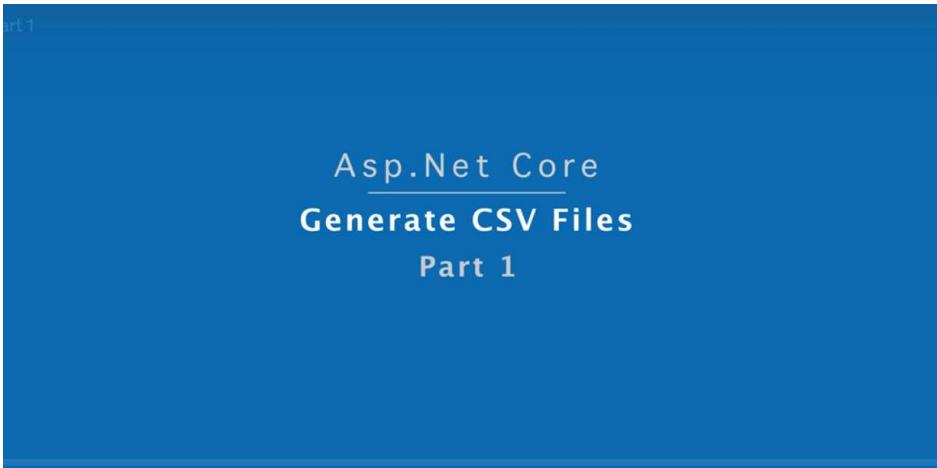
lagement()
age();

```

```

29
30     app.UseStaticFiles();
31     app.UseRouting();
32     app.MapControllers();
33
34     Rotativa.AspNetCore.RotativaConfiguration.Setup("wwwroot", wkhtmltopdfRelativePath:"Rotativa");
35
36     app.Run();
37

```



Now our next goal is to generate the data in the form of a CSV file (comma-separated values).

In the CSV file, each record is generated as one line, where a comma is the separator between the values.

For example, this is the person ID of the first person, then person name, and person email of the same person. Then, on the next row, the next record, and so on.

So, generally, for importing and exporting the data, you have to use the CSV file based on the client requirements.

For example, you can download Google contacts as a CSV file for migrating the same into another email provider.

So, in a similar way, assume our client requirement is that he is asking for a CSV file download for the persons.

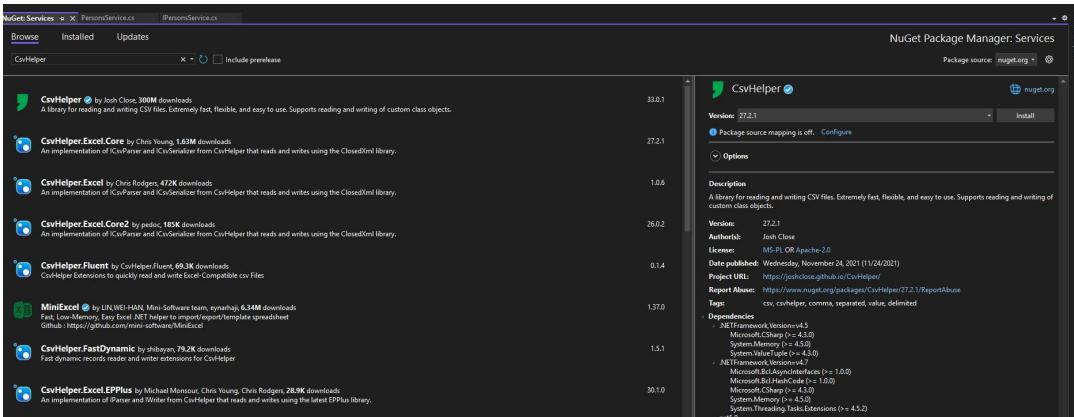
So, we have to convert the person's collection into a CSV file by using a third-party package called **CsvHelper**.

Let me demonstrate the same.

```

43     /// <param name="allPersons">Represents list of persons to sort</param>
44     /// <param name="sortBy">Name of the property (key), based on which the persons should be sorted</param>
45     /// <returns>Returns sorted persons as PersonResponse List</returns>
46     Task<List<PersonResponse>> GetSortedPersons(List<PersonResponse> allPersons, string sortBy,
47
48     /// <summary>
49     /// Updates the specified person details based on the given person ID
50     /// </summary>
51     /// <param name="personUpdateRequest">Person details to update, including person id</param>
52     /// <returns>Returns the person response object after update</returns>
53     Task<PersonResponse> UpdatePerson(PersonUpdateRequest? personUpdateRequest);
54
55     /// <summary>
56     /// Deletes a person based on the given person id
57     /// </summary>
58     /// <param name="personID">PersonID to delete</param>
59     /// <returns>Returns true, if the deletion is successful, otherwise false</returns>
60     Task<bool> DeletePerson(Guid? personID);
61
62     /// <summary>
63     /// Returns persons as CSV
64     /// </summary>
65     /// <returns>Returns the memory stream with CSV data</returns>
66     Task<MemoryStream> GetPersonsCSV();
67
68 }
69 }
70

```

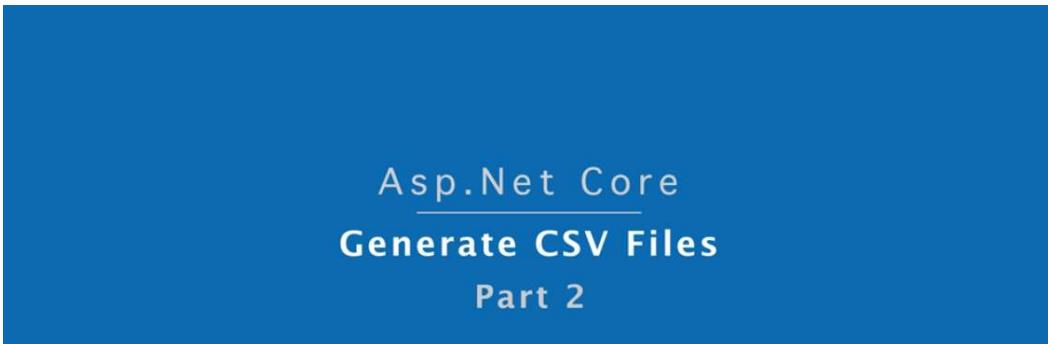


Asp.Net Core Harsha

Generate CSV Files (CsvHelper)

CsvWriter:

- WriteRecords(records)** Writes all objects in the given collection.
Eg:
Id, Name
1, abc
2, def
- WriteHeader<ModelClass>()** Writes all property names as headings.
Eg:
Id, Name



Asp.Net Core

Generate CSV Files

Part 2

Asp.Net Core

Generate CSV Files (CsvHelper)

CsvWriter:

`WriteRecords(records)`

Writes all objects in the given collection.

Eg:
1, abc
2, def

`WriteHeader<ModelClass>()`

Writes all property names as headings.

Eg:
Id, Name

Asp.Net Core

Generate CSV Files (CsvHelper)

CsvWriter:

`WriteRecord(record)`

Writes the given object as a row.

Eg:
1, abc

The problem with the **WriteRecords** and **WriteHeader** methods is that they don't let you choose which property values should be written to the stream. By default, all the property values are written through this method. And by default, all the property names (i.e., header names) are written through this method. But if you want to **customize the values** or **change the order of values**, then it's better to use the **WriteField** method. Actually, there is one more method called **WriteRecord**, which writes an entire single record, but we don't want to use that. Instead, we prefer **WriteField**.

Generate CSV Files (CsvHelper)

CsvWriter:

`WriteField(value)`

Writes given value.

`NextRecord()`

Moves to the next line.

`Flush()`

Writes the current data to the stream.



It writes a single value, no matter whether it is a value or a field heading.

When you want to move the cursor to the next line, you can use `NextRecord`.

For each record, we have to call the `Flush` method in order to write the data to the stream. Otherwise, the data will not be written to the stream.

Let me show that here.

Asp.Net Core

Generate Excel Files

Sometimes the client requirement will be to generate the data into an **Excel file**.

For example, there is a **Person's** table, and the client wants to download the person's information into an Excel file, just like a report or something.

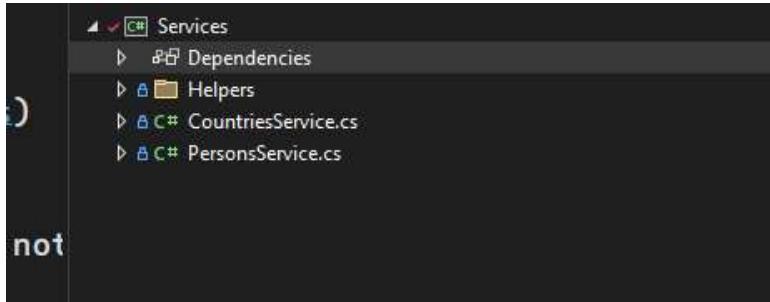
For that case, you can use the third-party package called **EPPlus**, which helps convert the existing collection of data into an **Excel file** with a **.XLS** extension.

Let me show that.

Generate Excel Files (EPPlus)

Collection → EPPlus (NuGet) → Excel File

(.xlsx)



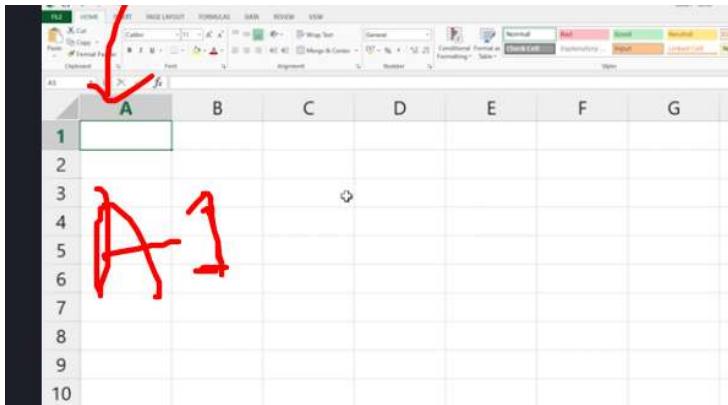
not

Screenshot of the NuGet Package Manager showing the search results for 'EPPlus'. The top result is 'EPPlus 6.0.6' by EPPlusSoftware, with 22.5M downloads. Below it is 'EPPlusSoftware 22.5M downloads' and 'EPPlus 1.3.4' by EPPlusSoftware, also with 22.5M downloads. Other packages listed include 'EPPlus 2.4.0' by magicode, 'EPPlus 2.7.3.2' by magicode, 'EPPlus 4.5.3.8' by magicode, 'EPPlus 2.2.0' by gravende, and 'EPPlus 2.0.1' by magicode.



It automatically reads this.

```
6     },
7     "AllowedHosts": "*",
8     "ConnectionStrings": {
9       "DefaultConnection": "Data Source=(localdb)\\MSSQLLocalDB;Initial Catalog=PersonsDatabase;Integrated Security=True;Connect Timeout=30;Encrypt=False;TrustServerCertificate=False;MultipleActiveResultSets=True;Pooling=True;Connection Timeout=30;MultipleSubnetFailover=False"
10    },
11    "EPPlus": {
12      "ExcelPackage": {
13        "LicenseContext": "NonCommercial"
14      }
15    }
16  }
17}
18}
19}
20}
21}
```



```
File Edit View Git Project Build Debug Test Analyze Tools Extensions Window Help Search - CRUDExample
PersonController.cs PersonService.cs IPersonService.cs appsettings.json Services\PersonsService.cs
281     memoryStream.Position = 0; //setting the cursor to the beginning
282     return memoryStream;
283   }
284 
285   public async Task<MemoryStream> GetPersonsExcel()
286   {
287     MemoryStream memoryStream = new MemoryStream();
288 
289     using (ExcelPackage excelPackage = new ExcelPackage())
290     {
291       ExcelWorksheet worksheet = excelPackage.Workbook.Worksheets.Add("PersonsSheet");
292       worksheet.Cells["A1"].Value = "Person Name";
293       worksheet.Cells["B1"].Value = "Email";
294       worksheet.Cells["C1"].Value = "Date of Birth";
295       worksheet.Cells["D1"].Value = "Age";
296       worksheet.Cells["E1"].Value = "Gender";
297       worksheet.Cells["F1"].Value = "Country";
298       worksheet.Cells["G1"].Value = "Address";
299       worksheet.Cells["H1"].Value = "Receive News Letters";
300 
301       int row = 2;
302       List<PersonResponse> persons = _db.Persons.Include("Country").Select(temp => temp.ToPersonResponse()).ToList();
303 
304       foreach (PersonResponse person in persons)
305       {
306         worksheet.Cells[row, 1].Value = person.PersonName;
307         worksheet.Cells[row, 2].Value = person.Email;
308 
309         if (person.DateOfBirth.HasValue)
310         {
311           worksheet.Cells[row, 3].Value = person.DateOfBirth.Value.ToString("yyyy-MM-dd");
312         }
313       }
314     }
315   }
316 }
```

```
199
200
201
202 [Route("PersonsExcel")]
203 public async Task<IActionResult> PersonsExcel()
204 {
205   MemoryStream memoryStream = await _personsService.GetPersonsExcel();
206 
207   return File(memoryStream, "application/vnd.openxmlformats-officedocument.spreadsheetml.sheet", "persons.xlsx");
208 }
209 }
210 }
```

```
1 @model IEnumerable<PersonResponse>
2
3 <@{
4     ViewBag.Title = "Persons";
5 }
6 <form asp-controller="Persons" asp-action="Index" method="get">
7     <h1>Persons</h1>
8
9     <a asp-controller="Persons" asp-action="Create">Create Person</a>
10    <a asp-controller="Persons" asp-action="PersonsPDF">Download as PDF</a>
11    <a asp-controller="Persons" asp-action="PersonsCSV">Download as CSV</a>
12    <a asp-controller="Persons" asp-action="PersonsExcel">Download as Excel</a>
13
14 <div class="box flex">
15     <div class="flex-1">
```

Official DOCS

The screenshot shows a browser window displaying the EPPlus software documentation. The URL is https://eplussoftware.com/docs/5.3/api/OfficeOpenXml.ExcelPackage. The page has a dark header with the EPPlus logo, 'Articles', and 'Api Documentation'. Below the header, the breadcrumb navigation shows 'Api Documentation / OfficeOpenXml /'. The main content area has a blue background with white text, titled 'Asp.Net Core Excel to Database Upload Part 1'.

Now, I would like to enable the **upload functionality** for the database table.
Here, you will learn the concept of **file uploading** as well as **migrating data from an Excel file into the database**.

The requirement is:

For example, on the **Create Persons** page, you have a list of countries, right?

So, if the user wants to add more countries, instead of adding them **one by one** through an entry form, they can enter the list of countries in an **Excel file** and upload it to the database.

That's the requirement.

So, let's see how to solve it.

localhost:5000/persons/entry

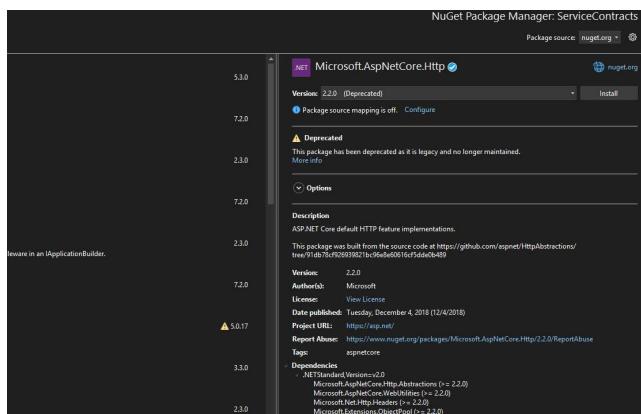
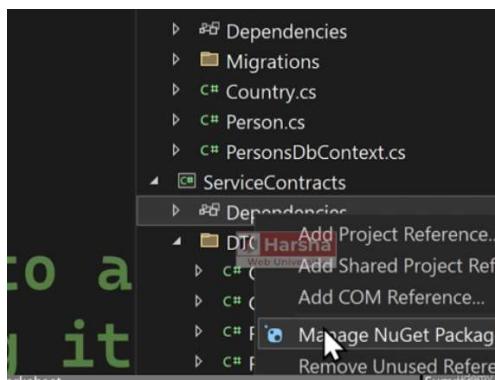
Persons

[Back to Persons List](#)

Create Person

Person Name	<input type="text"/>
Email	<input type="text"/>
Date of Birth	<input type="text"/> dd/mm/yyyy <input type="button" value=""/>
Gender	<input type="radio"/> Male <input type="radio"/> Female <input type="radio"/> Other
Country	<input type="text" value="Please Select"/> <ul style="list-style-type: none">Please SelectThailandChinaPalestinian TerritoryPhilippinesChina
Address	<input type="text"/>
<input type="checkbox"/> Receive News Letters	
<input type="button" value="Create"/>	

Harsha
Software Developer



```
    /// <summary>
    /// <returns>All countries from the list as List of CountryResponse</returns>
    Task<List<CountryResponse>> GetAllCountries();

    /// <summary>
    /// Returns a country object based on the given country id
    /// </summary>
    /// <param name="countryID">countryID (guid) to search</param>
    /// <returns>Matching country as CountryResponse object</returns>
    Task<CountryResponse?> GetCountryByCountryID(Guid? countryID);

    /// <summary>
    /// Upload countries from excel file into Database
    /// </summary>
    /// <param name="formFile"></param>
    /// <returns></returns>
    /// If formFile is a file that is submitted from the web browser
    Task<int> UploadCountriesFromExcelFile(IFormFile formFile);
}
```

We have to create a template for the end user

1	CountryName										
2											
3											
4											
5											
6											
7											
8											
9											
10											
11											
12											
13											
14											
15											
16											
17											
18											

