Create the database for identity

Saturday, November 25, 2023 10:46 PM

```
    public ApplicationDbContext(DbContextOptions<ApplicationDbContext> options):base(options)
    {

    }
```

Code Explanation:

1. ApplicationDbContext: This class inherits from IdentityDbContext. IdentityDbContext is a class provided by ASP.NET Core Identity, which is a framework for handling user authentication and authorization.
2. public ApplicationDbContext(DbContextOptions<ApplicationDbContext> options) : base(options): This is the constructor for the ApplicationDbContext class. It takes an instance of DbContextOptions<ApplicationDbContext> as a parameter. DbContextOptions is a generic type that is used to configure the behavior of the DbContext. The : base(options) part of the constructor is calling the constructor of the base class (IdentityDbContext) and passing the provided DbContextOptions to it.
   In simpler terms, when you create an instance of ApplicationDbContext, you need to pass in configuration options (typically related to the database connection) that will be used by Entity Framework Core.
   Here's an example of how you might use this ApplicationDbContext in a startup class to configure Entity Framework Core:

   The code (DbContextOptions<ApplicationDbContext> options) : base(options) is part of the constructor of the ApplicationDbContext class. Let's break down its purpose:
3. Constructor Parameters:
   - (DbContextOptions<ApplicationDbContext> options): This part declares a parameter for the constructor of type DbContextOptions<ApplicationDbContext>. DbContextOptions is a generic type that allows you to configure the behavior of the DbContext.
4. Base Class Constructor:
   - : base(options): This part is calling the constructor of the base class (IdentityDbContext) and passing the options parameter to it. The base keyword is used to explicitly call a constructor of the base class.
5. Purpose:
   - The purpose of this code is to initialize the ApplicationDbContext using the configuration options provided through the DbContextOptions. These options typically include information such as the database connection string, the database provider to use (e.g., SQL Server, PostgreSQL), and other settings related to how Entity Framework Core should interact with the database.
   - By passing the options to the base class constructor (IdentityDbContext), you are essentially configuring the base class (IdentityDbContext) with the database-related settings specified in DbContextOptions. This is crucial for Entity Framework Core to know how to connect to and interact with the underlying database.

   In summary, this code ensures that when you create an instance of ApplicationDbContext, you provide the necessary configuration options for it to establish a connection to the database and perform operations using Entity Framework Core. The base(options) call passes these options up to the constructor of the base class (IdentityDbContext).

```
1
```

Appsetting.Development.json

```
  "ConnectionStrings": {
    "Default": "Data Source=MTNEPAL-DEEPAK\\SQLEXPRESS03;Initial Catalog=AppUsers;Integrated Security=True;Encrypt=False;Trust Server Certificate=True"
  }
}
```

Program.cs

```
builder.Services.AddDbContext<ApplicationDbContext>(options => {
    options.UseSqlServer(builder.Configuration.GetConnectionString("Default"));
});
```

In Package Manager Console

```
Add-Migration Init
Update-Database
```