[2- LinkedIn article about RESTful API.](https://www.linkedin.com/posts/ahmed-b-ramzy_aspnetcore-restapi-dotnet-activity-7338870738306699264-sn-n?utm_source=share&utm_medium=member_android&rcm=ACoAADuIxXkBdVPXITPJTt0yTVu1y8HgMAABtYE)

[3- LinkedIn article about Onion Architecture.](https://www.linkedin.com/posts/ahmed-b-ramzy_onionarchitecture-dotnetcore-cleanarchitecture-activity-7338873537824264194-fPQA?utm_source=share&utm_medium=member_android&rcm=ACoAADuIxXkBdVPXITPJTt0yTVu1y8HgMAABtYE)

4- What is Data Seeding?

**Data Seeding** is the process of **pre-populating a database with initial data** when the application starts or during the database creation/migration process. It's commonly used in application development to ensure that essential or default data exists in the database, especially during development, testing, or when deploying to new environments.

**Why is Data Seeding Important?**

Data Seeding is useful in multiple scenarios:

* Populating default user roles (e.g., Admin, User, Moderator)
* Adding lookup data (e.g., list of countries, currencies, or categories)
* Creating default settings or configuration values
* Populating sample data for demos or development

Without seeding, developers would have to manually insert this data each time the database is recreated — which is repetitive and error-prone.

**How Data Seeding Works in EF Core (ASP.NET Core)**

In Entity Framework Core, you typically seed data inside the OnModelCreating method of your DbContext. Here's an example:

protected override void OnModelCreating(ModelBuilder modelBuilder)

{

base.OnModelCreating(modelBuilder);

modelBuilder.Entity<Role>().HasData(

new Role { Id = 1, Name = "Admin" },

new Role { Id = 2, Name = "User" }

);

}

Once this is added, run:

dotnet ef migrations add SeedInitialData

dotnet ef database update

EF Core will insert this data during the database update process.

**Seeding Related Entities**

When seeding entities that have relationships, you need to make sure the dependent data is seeded in the right order and with correct foreign key values. Example:

modelBuilder.Entity<Category>().HasData(

new Category { Id = 1, Name = "Books" }

);

modelBuilder.Entity<Product>().HasData(

new Product { Id = 1, Name = "C# Basics", CategoryId = 1 }

);

EF Core doesn’t automatically resolve foreign key relationships — you must provide explicit values (like CategoryId = 1).

**Seeding in Development vs Production**

It's common to separate **development-only** seeding (e.g., sample users or mock data) from **production-safe** seeding (e.g., system roles, config values). You can use custom logic to detect the environment and seed accordingly.

Example:

if (env.IsDevelopment())

{

// Seed test users, dummy data

}

else

{

// Seed only critical, safe data

}

**Alternative: Manual Seeding with Services**

Instead of using HasData, you can create custom seeding services and call them during application startup. This gives you more flexibility:

public static class SeedData

{

public static void Initialize(IServiceProvider serviceProvider)

{

using var context = new AppDbContext(

serviceProvider.GetRequiredService<DbContextOptions<AppDbContext>>());

if (!context.Roles.Any())

{

context.Roles.Add(new Role { Name = "Admin" });

context.SaveChanges();

}

}

}

Then call it from Program.cs:

SeedData.Initialize(app.Services);