Contents

[CHAPTER 1: READING DATA FROM appsettings.json USING STRONGLY TYPED CONFIGURATION 1](#_Toc144644419)

[DETAILS 1](#_Toc144644420)

[a. Defining object and its properties in appsettings.json 1](#_Toc144644421)

[b. Creating the C# class to mimic the properties in appsettings.json file that you want to read/access 1](#_Toc144644422)

[c. Adding appsettings.json in program.cs 1](#_Toc144644423)

[d. Configuring/Registering the class above in program.cs 1](#_Toc144644424)

[e. Using dependency injection to access the appsettings.json properties 2](#_Toc144644425)

# CHAPTER 1: READING DATA FROM appsettings.json USING STRONGLY TYPED CONFIGURATION

1. Define your object and its properties in the appsettings.json file
2. Create a C# class and specify properties that corresponds to the object properties defined in the appsettings.json file (Make sure the properties of the class mimics the object properties in the appsettings.json)
3. Add appsettings.json file in program.cs
4. Configure/Register your class in the program.cs file
5. Use dependency injection of the class you created to access the properties in appsettings.json file

## DETAILS

### Defining object and its properties in appsettings.json

"MyAppSettings": {

"email": "fostersnt@gmail.com",

"password": "mypassword"

}

### Creating the C# class to mimic the properties in appsettings.json file that you want to read/access

public class AppSettings //*You can give any name to the class*

{

public string email { get; set; } //*mimics “email” in appsetings.json*

public string password { get; set; } //*mimics “password” in appsetings.json*

}

### Adding appsettings.json in program.cs

//Adding apsettings.json file

builder.Configuration.AddJsonFile("appsettings.json");

### Configuring/Registering the class above in program.cs

// Configure the strongly typed options

builder.Services.Configure<AppSettings>(builder.Configuration.GetSection("MyAppSettings"));

### Using dependency injection to access the appsettings.json properties

Create a constructor out of your controller. In this code below, the of name my controller class is **HomeController**

private readonly AppSettings \_appSettings;

public HomeController(IOptions<AppSettings> appSettings)

{

\_appSettings = appSettings.Value;

}

# CHAPTER 2: FILE PROCESSING

## DETAILS

### Creating and storing a file

using System;

using System.IO;

using File = System.IO.File; // Alias for System.IO.File

class Program

{

static void Main()

{

string filePath = @"C:\Your\Specific\Directory\Path\example.txt";

try

{

// Check if the file exists and write content to it

if (!File.Exists(filePath))

{

File.WriteAllText(filePath, "Hello, world!");

Console.WriteLine("File created and written successfully.");

}

else

{

Console.WriteLine("File already exists.");

}

}

catch (Exception ex)

{

Console.WriteLine($"An error occurred: {ex.Message}");

}

}

}

# CHAPTER FINAL: ERRORS IN ASP.NET API/EF Core

## Migration Errors

### The entity type 'IdentityUserLogin<string>' requires a primary key to be defined. If you intended to use a keyless entity type, call 'HasNoKey' in 'OnModelCreating'. For more information on keyless entity types, see <https://go.microsoft.com/fwlink/?linkid=2141943>.

During migration, you are likely to encounter the above error if your dbContext class inherits IdentityDbContext<IdentityUser> and you fail to include base.OnModelCreating(builder) in the OnModelCreating(ModelBuilder builder) in your dbContext class. To resolve it, you just have to add that line of code or you look out for an advanced alternative.