Practical 5: Authentication with Entity Framework Core (Identity)

In this practical, you will

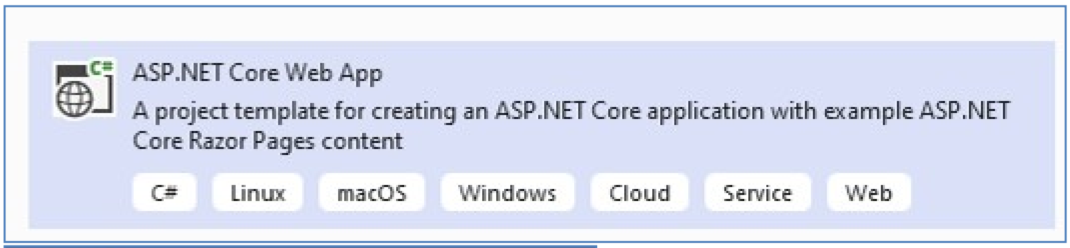
- Set up the environment for Entity Framework Core
- Create database using DB context and models
- Implement Register page with build-in membership system
- Implement Login Page and redirect

One feature that a lot of web applications will need to include is the ability restrict access to certain resources within the application to authorised users only. To do this, we need to be able to authenticate users by letting them register and log in. Once authenticated, the server will be able to determine which resources the user should have access to.

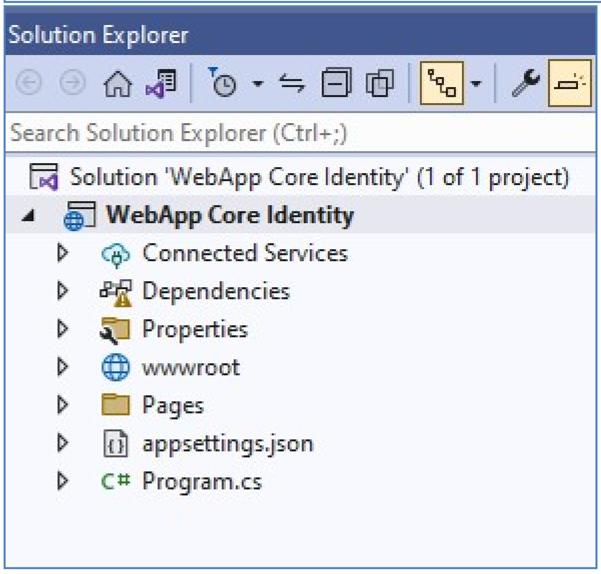
In the next few practicals, we will go through the steps needed to add authentication and authorisation using ASP.NET Core Identity as well as ASP.NET Core's built-in membership system. We will implement user authentication and authorisation using ASP.NET Core Identity.

Activity 1: Create New Core WebApp

1. Create a new project ➔ "ASP.NET Core Web App.
2. The default files will be generated in the Solution Explorer.

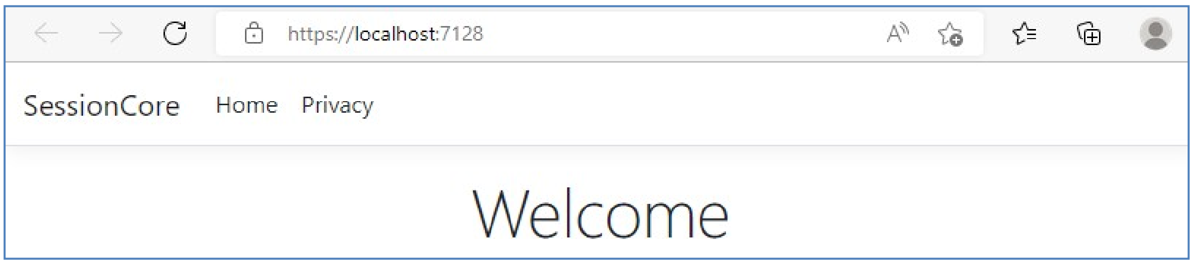


ASP.NET Core Web App
A project template for creating an ASP.NET Core application with example ASP.NET Core Razor Pages content



Solution 'WebApp Core Identity' (1 of 1 project)
WebApp Core Identity
Connected Services
Dependencies
Properties
wwwroot
Pages
appsettings.json
Program.cs

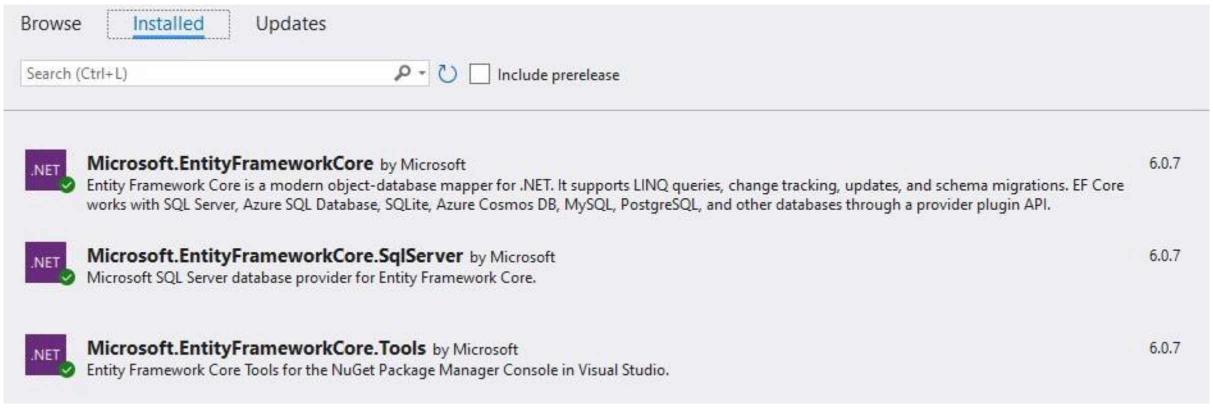
3. Run the app and check the default behaviour.



Let's make some changes to upgrade bootstrap to version 5. See demo from tutor.

Activity 2: Set up the Environment for Entity Framework Core

1. In Solution Explorer, right click on project, click **Manage Nuget Packages**. Search package and install:
   - Microsoft.EntityFrameworkCore
   - Microsoft.EntityFrameworkCore.SqlServer
   - Microsoft.EntityFrameworkCore.Tools



Microsoft.EntityFrameworkCore by Microsoft 6.0.7
Entity Framework Core is a modern object-database mapper for .NET. It supports LINQ queries, change tracking, updates, and schema migrations. EF Core works with SQL Server, Azure SQL Database, SQLite, Azure Cosmos DB, MySQL, PostgreSQL, and other databases through a provider plugin API.

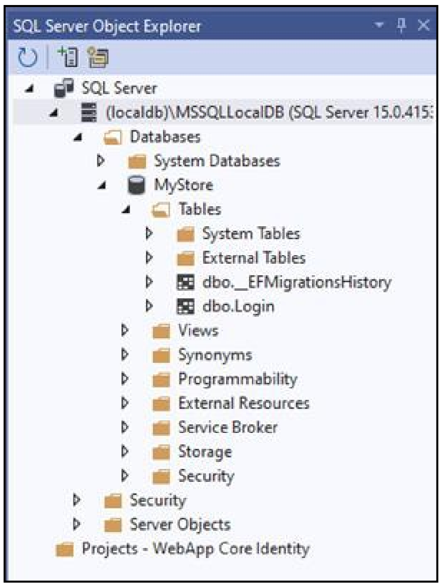
Microsoft.EntityFrameworkCore.SqlServer by Microsoft 6.0.7
Microsoft SQL Server database provider for Entity Framework Core.

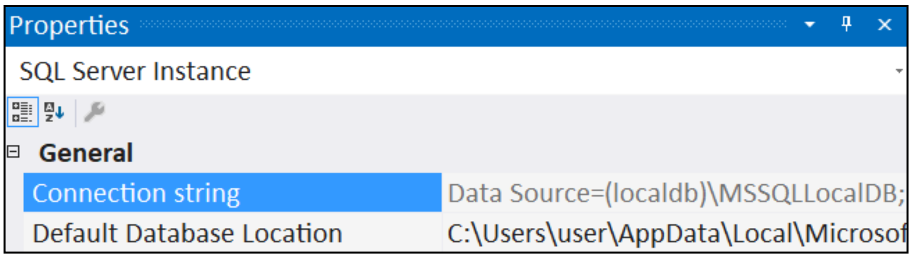
Microsoft.EntityFrameworkCore.Tools by Microsoft 6.0.7
Entity Framework Core Tools for the NuGet Package Manager Console in Visual Studio.

2. Open **appsetting.json** to define the **ConnectionStrings**. This allows us to configure the connection strings without recompile it during the application deployment. This practical only requires one database connection **MyConnection**.

```
"AllowedHosts": "*",

"ConnectionStrings": {
"AuthConnectionString": "Data Source=(LocalDB)\\MSSQLLocalDB;Initial
Catalog=AspNetAuth;Integrated Security=True; "
 }
```

3. Click Visual Studio Menu: View -> SQL Server Object Explorer. Right click on (locandb)\MSSQLLocalDB and select Properties. Copy the **Connection string** value, put inside appsetting.json, replace the DB name "master" with "**AspNetAuth**".



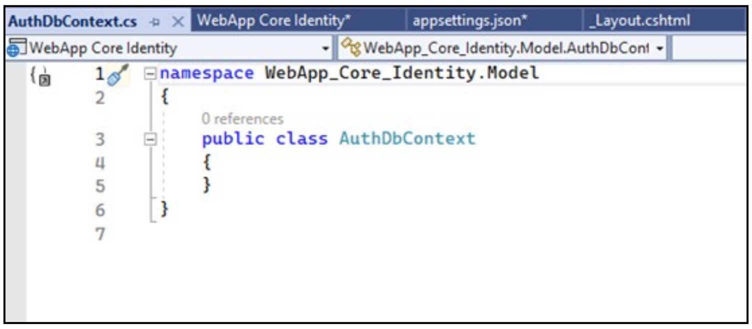


Connection string	Data Source=(localdb)\MSSQLLocalDB;
Default Database Location	C:\Users\user\AppData\Local\Microsof

Your code in appsetting.json will be like below:

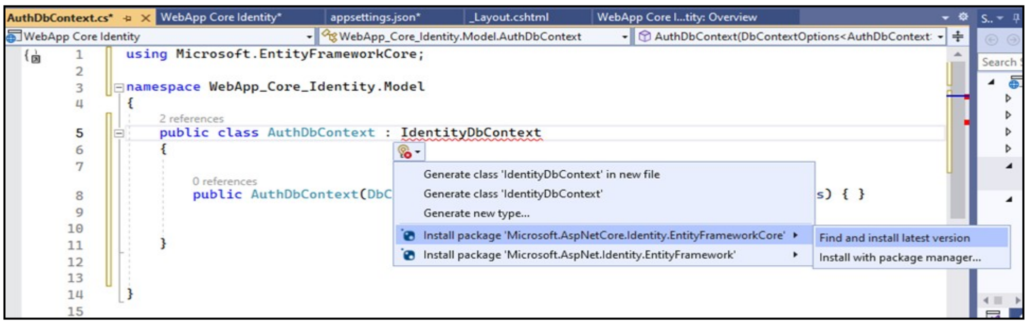
```
"AllowedHosts": "*",

"ConnectionStrings": {
"AuthConnectionString": "Data Source=(LocalDB)\\MSSQLLocalDB;Initial
Catalog=AspNetAuth;Integrated Security=True; "
 }
```

4. Under solution project, add a new Folder “Model” and add AuthDbContext class AuthDbContext.cs is shown below.



```
namespace WebApp_Core_Identity.Model
{
    0 references
    public class AuthDbContext
    {
    }
}
```

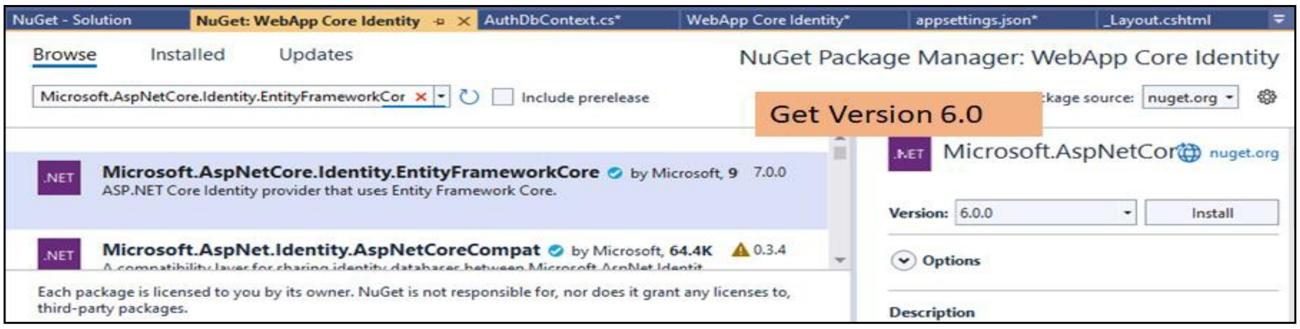
```
using Microsoft.AspNetCore.Identity.EntityFrameworkCore;
using Microsoft.EntityFrameworkCore;

namespace WebApp_Core_Identity.Model
{
   public class AuthDbContext:IdentityDbContext
   {

      private readonly IConfiguration _configuration;
      //public AuthDbContext(DbContextOptions<AuthDbContext> options):base(options){ }

       public AuthDbContext(IConfiguration configuration){
             _configuration = configuration;
       }

       protected override void OnConfiguring(DbContextOptionsBuilder optionsBuilder)
       {
       string    connectionString    =    _configuration.GetConnectionString("AuthConnectionString");optionsBuilder.UseSqlServer(connectionString);
       }
  }
}
```



```
using Microsoft.EntityFrameworkCore;

namespace WebApp_Core_Identity.Model
{
    2 references
    public class AuthDbContext : IdentityDbContext
    {

        0 references
        public AuthDbContext(DbC                                  s) { }

    }

}
```

Generate class 'IdentityDbContext' in new file
Generate class 'IdentityDbContext'
Generate new type...
Install package 'Microsoft.AspNetCore.Identity.EntityFrameworkCore'
Install package 'Microsoft.AspNet.Identity.EntityFramework'
Find and install latest version
Install with package manager...

Install Microsoft.AspNetCore.Identity.EntityFrameworkCore (use Version 6.0 which is compatible with .net 6)



NuGet Package Manager: WebApp Core Identity

Microsoft.AspNetCore.Identity.EntityFrameworkCore by Microsoft, 9 7.0.0
ASP.NET Core Identity provider that uses Entity Framework Core.

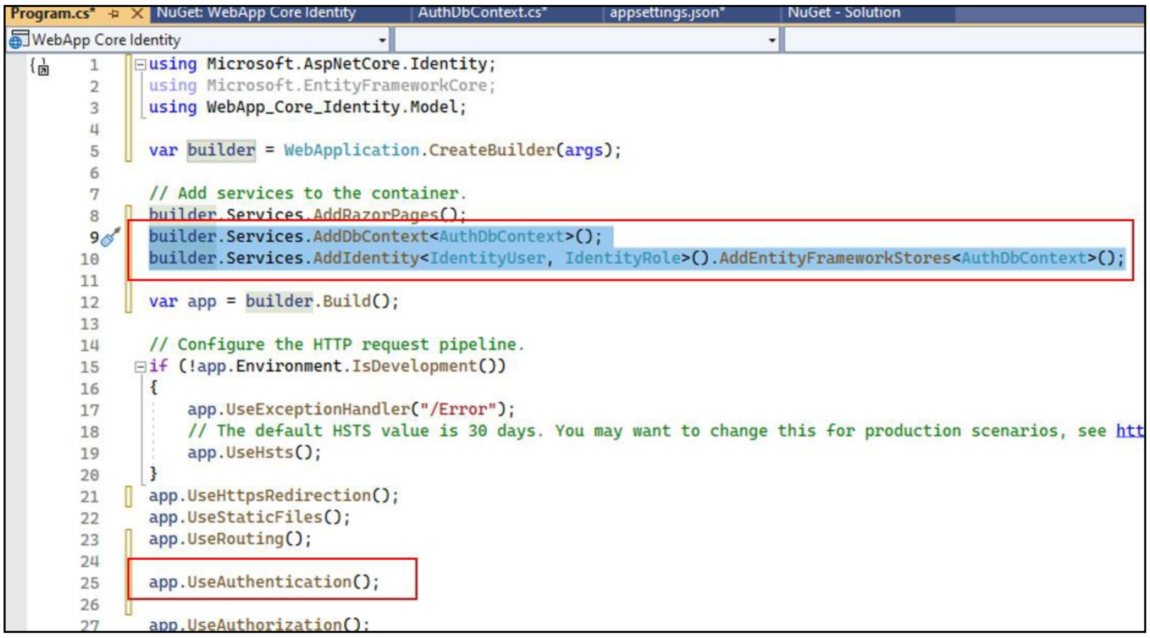
Microsoft.AspNet.Identity.AspNetCoreCompat by Microsoft, 64.4K 0.3.4

Each package is licensed to you by its owner. NuGet is not responsible for, nor does it grant any licenses to, third-party packages.

Get Version 6.0

Version: 6.0.0

5. Update code in **Program**.cs to use the DB context.

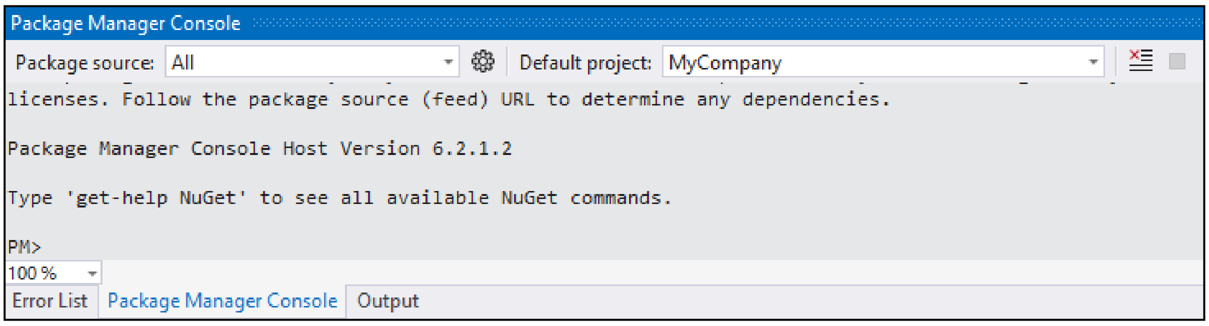


```
using Microsoft.AspNetCore.Identity;
using Microsoft.EntityFrameworkCore;
using WebApp_Core_Identity.Model;

var builder = WebApplication.CreateBuilder(args);

// Add services to the container.
builder.Services.AddRazorPages();
builder.Services.AddDbContext<AuthDbContext>();
builder.Services.AddIdentity<IdentityUser, IdentityRole>().AddEntityFrameworkStores<AuthDbContext>();

var app = builder.Build();

// Configure the HTTP request pipeline.
if (!app.Environment.IsDevelopment())
{
    app.UseExceptionHandler("/Error");
    // The default HSTS value is 30 days. You may want to change this for production scenarios, see htt
    app.UseHsts();
}
app.UseHttpsRedirection();
app.UseStaticFiles();
app.UseRouting();

app.UseAuthentication();

app.UseAuthorization();
```

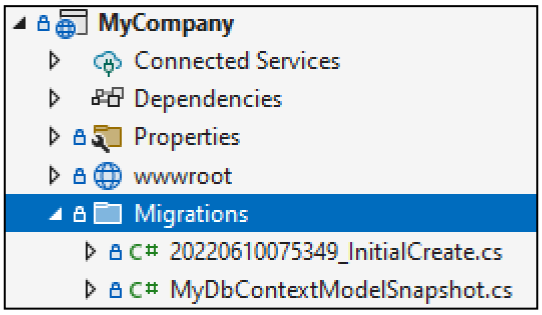
builder.Services.AddDbContext<AuthDbContext>();
builder.Services.AddIdentity<IdentityUser, IdentityRole>().AddEntityFrameworkStores<AuthDbContext>();

App.UseAuthentication();

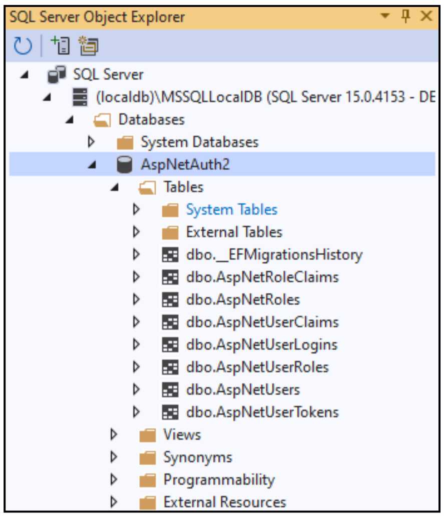
6. In package manager console, run command: **Add-Migration InitialCreate**



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

Package Manager Console Host Version 6.2.1.2

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

PM>
```

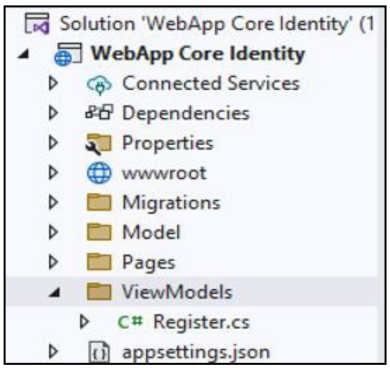


7. The folder Migrations is auto generated with 2 files inside.
8. In package manager console, run command: **Update-Database**
   Note: If you update your model class code later, then need to run commands to add new migration and update database.
9. Click Visual Studio Menu: View -> SQL Server Object Explorer. Observe that the database AspNetAuth is created under the Local DB.

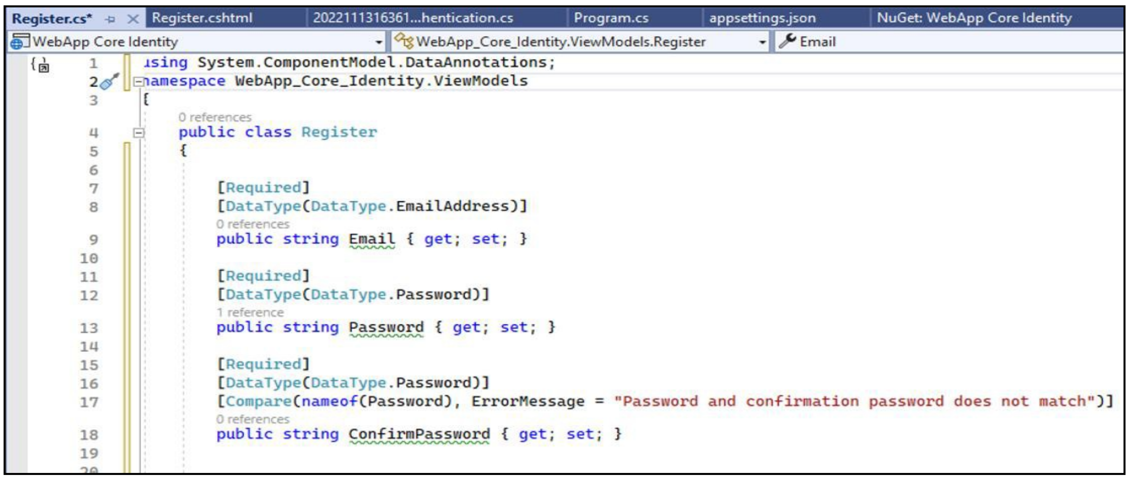


Activity 3: Create Registration Page (View and Model)

1. Create a folder in solution explorer "ViewModels"



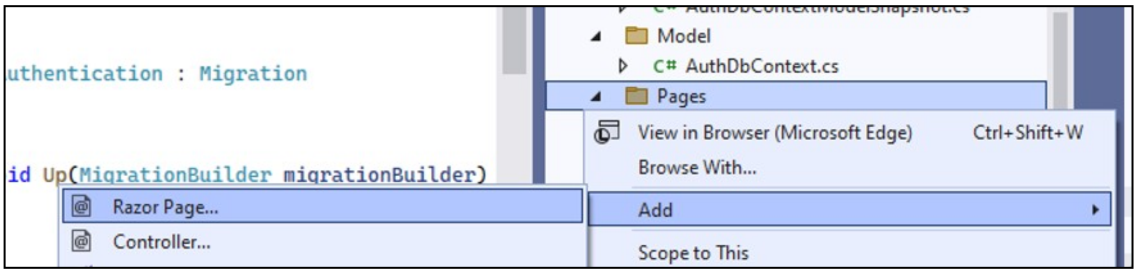
2. Inside the folder create a class Register.cs

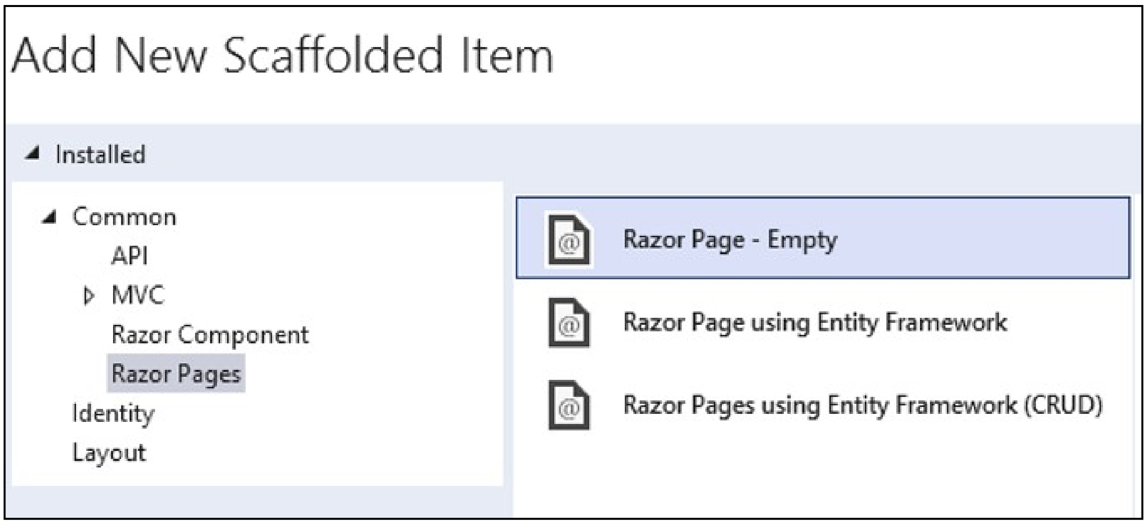

Register.cs*
Register.cshtml
2022111316361...hentication.cs
Program.cs
appsettings.json
NuGet: WebApp Core Identity
WebApp Core Identity
WebApp_Core_Identity.ViewModels.Register
Email

```
using System.ComponentModel.DataAnnotations;
namespace WebApp_Core_Identity.ViewModels
{
    public class Register
    {

        [Required]
        [DataType(DataType.EmailAddress)]
        public string Email { get; set; }

        [Required]
        [DataType(DataType.Password)]
        public string Password { get; set; }

        [Required]
        [DataType(DataType.Password)]
        [Compare(nameof(Password), ErrorMessage = "Password and confirmation password does not match")]
        public string ConfirmPassword { get; set; }

```

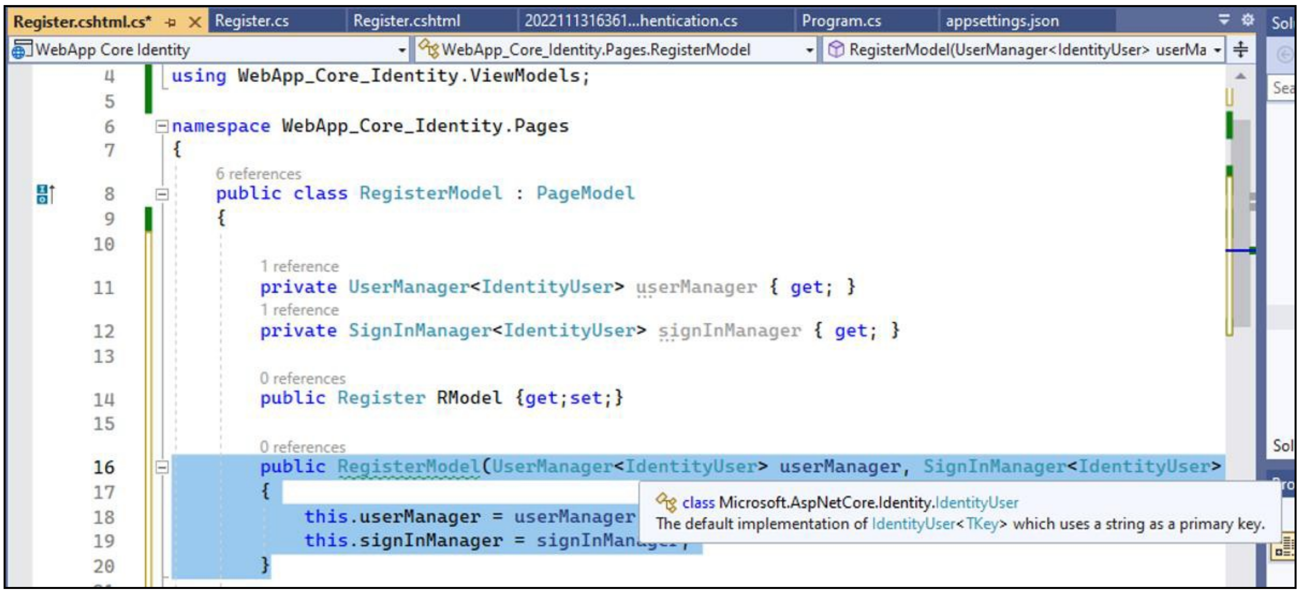
```
public class Register
  {
    [Required]
    [DataType(DataType.EmailAddress)]
    public string Email { get; set; }

    [Required]
    [DataType(DataType.Password)]
    public string Password { get; set; }

    [Required]
    [DataType(DataType.Password)]
    [Compare(nameof(Password), ErrorMessage = "Password and confirmation password
     does not match")]
    public string ConfirmPassword { get; set; }
  }
```

3. Under Pages ➔ create new Razor page ➔ empty razor page
4. Filename : Register.cshtml (Front/backend codes)


uthentication : Migration
id Up(MigrationBuilder migrationBuilder)
Razor Page...
Controller...
Model
AuthDbContext.cs
Pages
View in Browser (Microsoft Edge) Ctrl+Shift+W
Browse With...
Add
Scope to This



5. Edit backend codes : Register.cshtml.cs

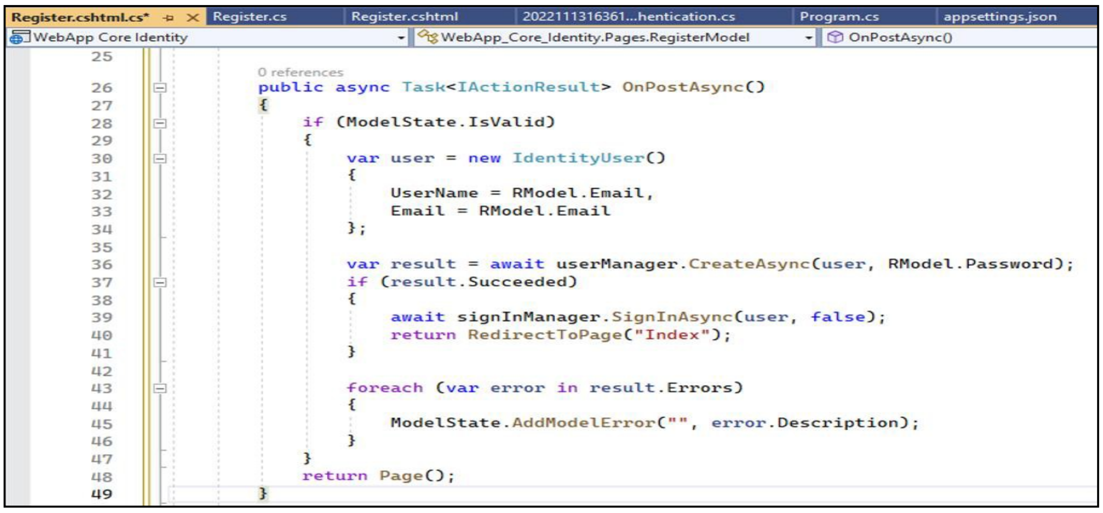

Register.cshtml.cs*
Register.cs
Register.cshtml
2022111316361...hentication.cs
Program.cs
appsettings.json
WebApp Core Identity
WebApp_Core_Identity.Pages.RegisterModel
RegisterModel(UserManager<IdentityUser> userMa

```
using WebApp_Core_Identity.ViewModels;

namespace WebApp_Core_Identity.Pages
{
    public class RegisterModel : PageModel
    {

        private UserManager<IdentityUser> userManager { get; }
        private SignInManager<IdentityUser> signInManager { get; }

        public Register RModel {get;set;}

        public RegisterModel(UserManager<IdentityUser> userManager, SignInManager<IdentityUser>
        {
            this.userManager = userManager
            this.signInManager = signInMan
        }
```

class Microsoft.AspNetCore.Identity.IdentityUser
The default implementation of IdentityUser<TKey> which uses a string as a primary key.

See codes below

```
//Initialize the build-in ASP.NET Identity
public class RegisterModel : PageModel
  {
        private UserManager<IdentityUser> userManager { get; }
        private SignInManager<IdentityUser> signInManager { get; }


        [BindProperty]
        public Register RModel {get;set;}


        public RegisterModel(UserManager<IdentityUser> userManager,
        SignInManager<IdentityUser> signInManager)
        {
              this.userManager = userManager;
              this.signInManager = signInManager;
        }
              public void OnGet() { … … …
```

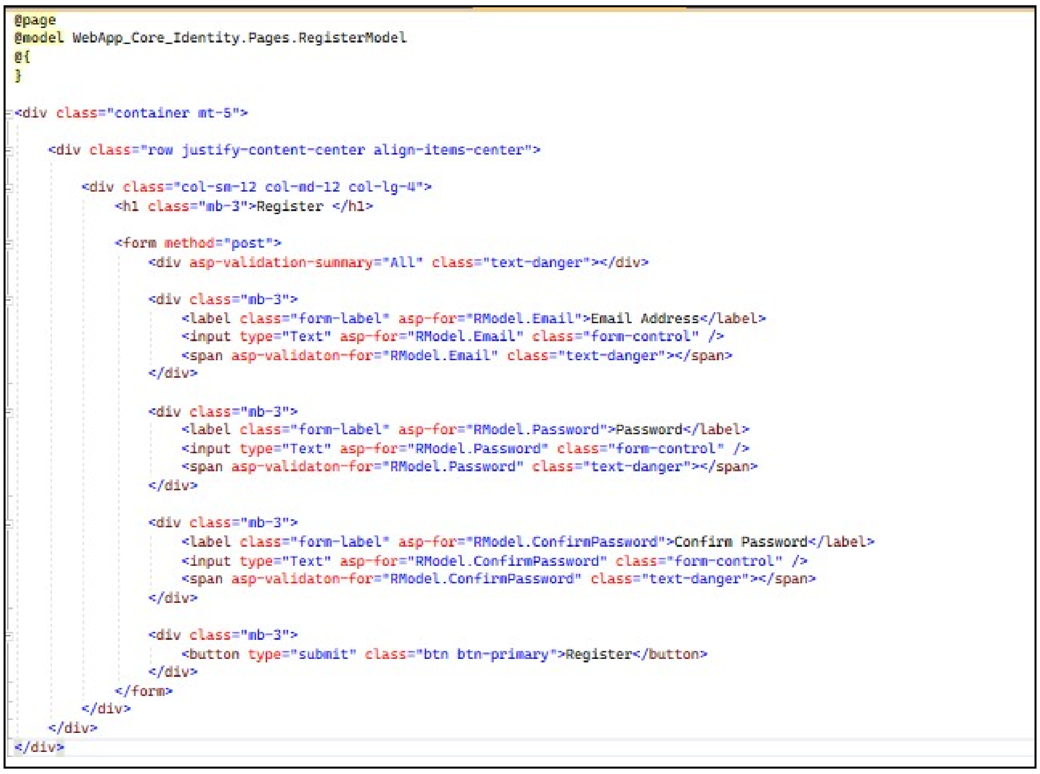


```
public async Task<IActionResult> OnPostAsync()
{
    if (ModelState.IsValid)
    {
        var user = new IdentityUser()
        {
            UserName = RModel.Email,
            Email = RModel.Email
        };

        var result = await userManager.CreateAsync(user, RModel.Password);
        if (result.Succeeded)
        {
            await signInManager.SignInAsync(user, false);
            return RedirectToPage("Index");
        }

        foreach (var error in result.Errors)
        {
            ModelState.AddModelError("", error.Description);
        }
    }
    return Page();
}
```

```
//Save data into the database
public async Task<IActionResult> OnPostAsync(){
        if (ModelState.IsValid){
              var user=new IdentityUser(){
                     UserName = RModel.Email,
                     Email = RModel.Email
              };
              var result = await userManager.CreateAsync(user,RModel.Password);
              if (result.Succeeded){
                     await signInManager.SignInAsync(user, false);
                     return RedirectToPage("Index");
              }
              foreach (var error in result.Errors){
                     ModelState.AddModelError("", error.Description);
              }
```

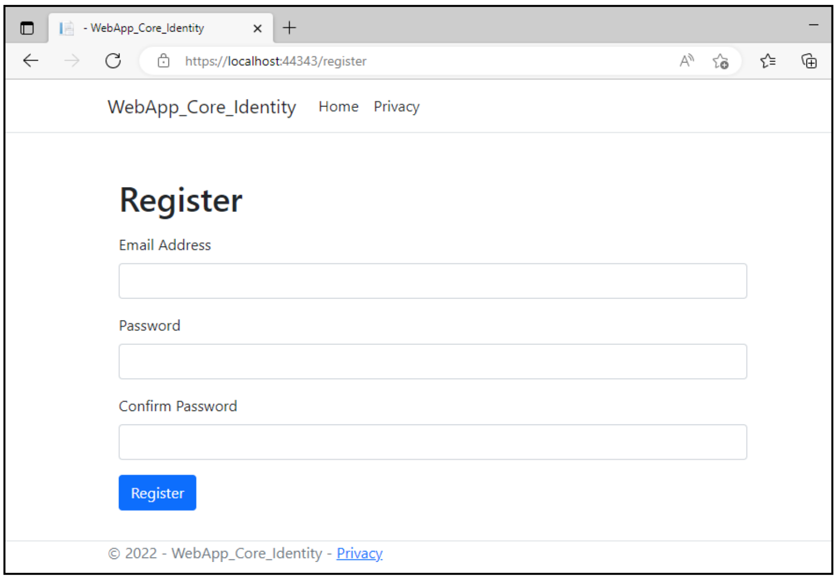
```
            }
            return Page();
    }
```

6. Edit the Register page (Register.cshtml). Use the sample codes provided in LMS.



```
@page
@model WebApp_Core_Identity.Pages.RegisterModel
@{
}

<div class="container mt-5">

    <div class="row justify-content-center align-items-center">

        <div class="col-sm-12 col-md-12 col-lg-4">
            <h1 class="mb-3">Register </h1>

            <form method="post">
                <div asp-validation-summary="All" class="text-danger"></div>

                <div class="mb-3">
                    <label class="form-label" asp-for="RModel.Email">Email Address</label>
                    <input type="Text" asp-for="RModel.Email" class="form-control" />
                    <span asp-validaton-for="RModel.Email" class="text-danger"></span>
                </div>

                <div class="mb-3">
                    <label class="form-label" asp-for="RModel.Password">Password</label>
                    <input type="Text" asp-for="RModel.Password" class="form-control" />
                    <span asp-validaton-for="RModel.Password" class="text-danger"></span>
                </div>

                <div class="mb-3">
                    <label class="form-label" asp-for="RModel.ConfirmPassword">Confirm Password</label>
                    <input type="Text" asp-for="RModel.ConfirmPassword" class="form-control" />
                    <span asp-validaton-for="RModel.ConfirmPassword" class="text-danger"></span>
                </div>

                <div class="mb-3">
                    <button type="submit" class="btn btn-primary">Register</button>
                </div>
            </form>
        </div>
    </div>
</div>
```

7. Test if your Registration is working properly.


- WebApp_Core_Identity
https://localhost:44343/register
WebApp_Core_Identity
Home
Privacy
Register
Email Address
Password
Confirm Password
Register
© 2022 - WebApp_Core_Identity - Privacy