



Using Identity In ASP.NET Core MVC Authentication



Ramasagar Pulidindi

Updated date Aug 12, 2020

71.2k

1

9

[Download Free .NET & JAVA Files API](#)

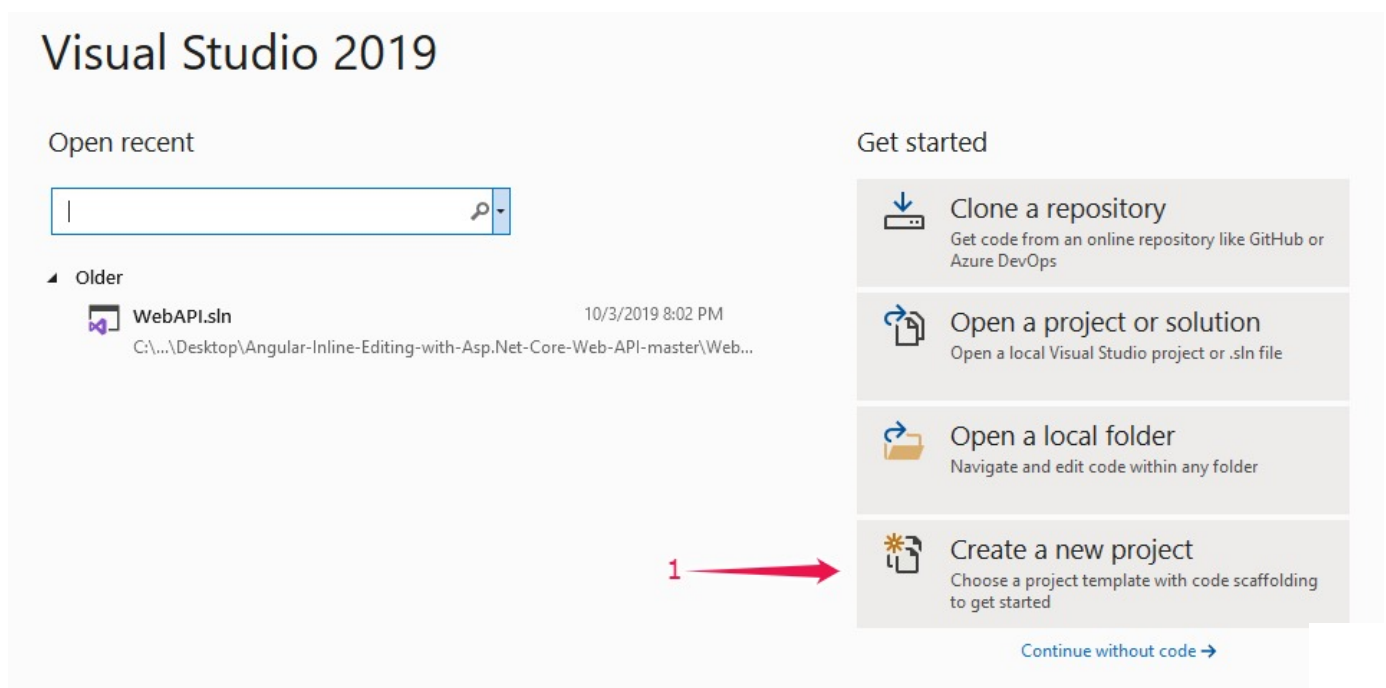
Introduction

This article explains how to get started with Identity.UI in ASP.Net Core MVC user authentication and registration.

Link to download the project source code [here](#).

Step 1

Let's create an ASP.NET Core web application. Open Visual Studio 2019 and click on Create a new project.




Create a new project

Search for templates (Alt+S) [Clear all](#)

Recent project templates


A list of your recently accessed templates will be displayed here.

2 


C#

All platforms


Web




ASP.NET Core Web Application
Project templates for creating ASP.NET Core web apps and web APIs for Windows, Linux and macOS using .NET Core or .NET Framework. Create web apps with Razor Pages, MVC, or Single Page Apps (SPA) using Angular, React, or React + Redux.
C# Linux macOS Windows Cloud Service Web




Blazor App
Project templates for creating Blazor apps that run on the server in an ASP.NET Core app or in the browser on WebAssembly (wasm). These templates can be used to build web apps with rich dynamic user interfaces (UIs).
C# Linux macOS Windows Cloud Web




gRPC Service
A project template for creating a gRPC ASP.NET Core service using .NET Core.
C# Linux macOS Windows Cloud Service Web



Razor Class Library
A project template for creating a Razor class library.
C# Linux macOS Windows Library Web



JUnit Test Project (.NET Core)
A project that contains NUnit tests that can run on .NET Core on Windows, Linux and MacOS.
C# Linux macOS Windows Desktop Test Web

3 

Back Next

Provide the project name of your choice, select the preferred location & click on Create

Configure your new project

ASP.NET Core Web Application C# Linux macOS Windows Cloud Service Web

Project name

Location
 ...

Solution name ⓘ

☐ Place solution and project in the same directory

Back Create

Select MVC Template and click on create, as shown below:

×

Create a new ASP.NET Core web application

.NET Core

Empty
An empty project template for creating an ASP.NET Core application. This template does not have any content in it.

API
A project template for creating an ASP.NET Core application with an example Controller for a RESTful HTTP service. This template can also be used for ASP.NET Core MVC Views and Controllers.

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

Web Application (Model-View-Controller)
A project template for creating an ASP.NET Core application with example ASP.NET Core MVC Views and Controllers. This template can also be used for RESTful HTTP services.

Angular
A project template for creating an ASP.NET Core application with Angular

React.js

Authentication

No Authentication
[Change](#)

Advanced

☒ Configure for HTTPS

☐ Enable Docker Support
(Requires [Docker Desktop](#))

☐ Enable Razor runtime compilation

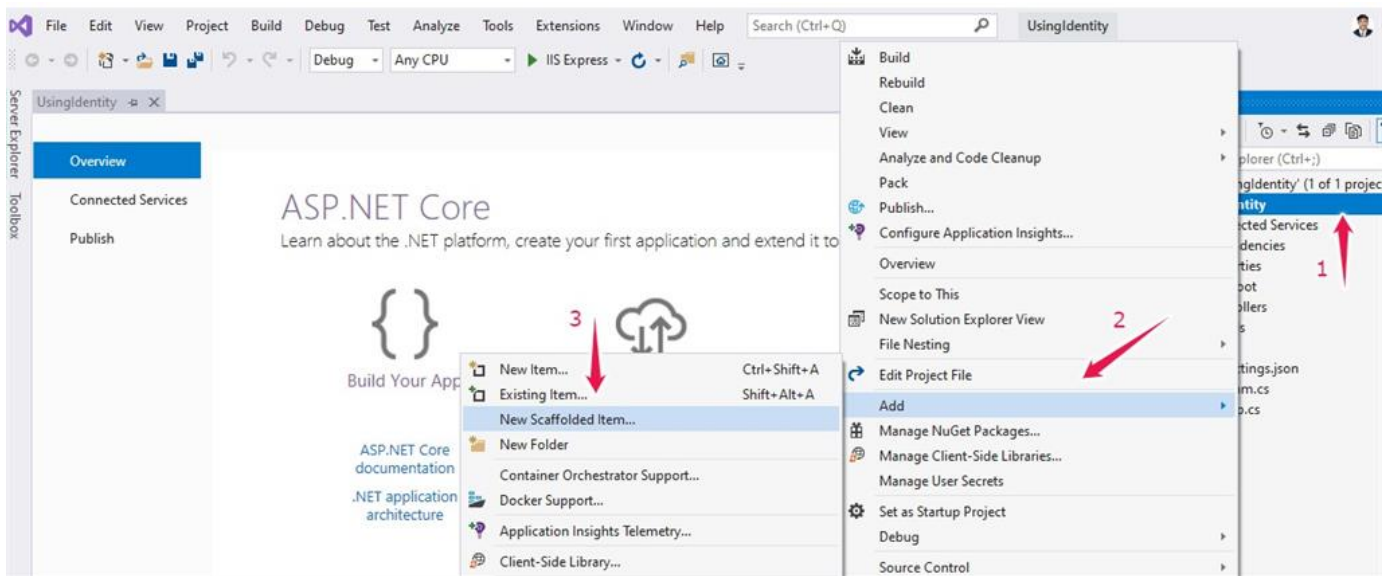
Author: Microsoft
Source: Templates 3.1.7

[Get additional project templates](#)

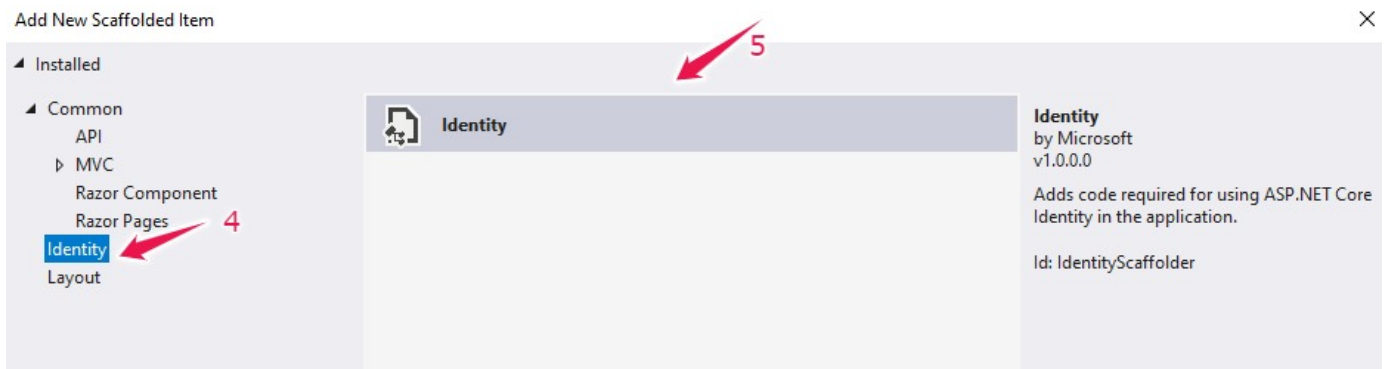
Back Create

Step 2

Scaffold item



Select Identity and click on add



Now select the layout page, as we want authentication. Let's select login and Register as shown below and provide DbContext class and user class and click on Add

Select an existing layout page, or specify a new one:

(Leave empty if it is set in a Razor _viewstart file)

☐ Override all files

Choose files to override

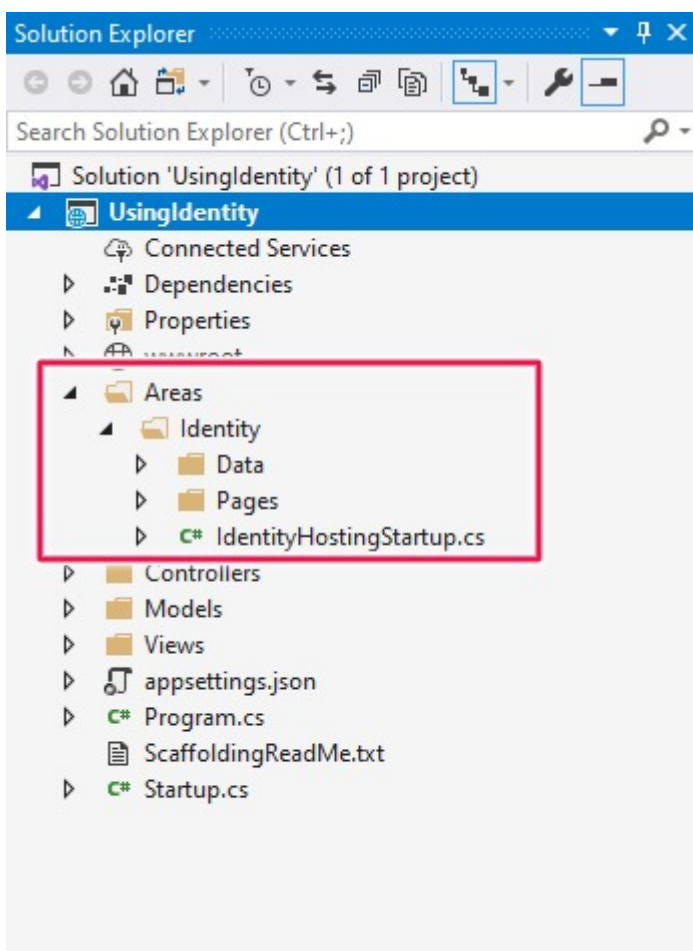
<input type="checkbox"/> Account\StatusMessage	<input type="checkbox"/> Account\AccessDenied	<input type="checkbox"/> Account\ConfirmEmail
<input type="checkbox"/> Account\ConfirmEmailChange	<input type="checkbox"/> Account\ExternalLogin	<input type="checkbox"/> Account\ForgotPassword
<input type="checkbox"/> Account\ForgotPasswordConfirmation	<input type="checkbox"/> Account\Lockout	<input type="checkbox"/> Account>Login
<input type="checkbox"/> Account>LoginWith2fa	<input type="checkbox"/> Account>LoginWithRecoveryCode	<input type="checkbox"/> Account\Logout
<input type="checkbox"/> Account\Manage\Layout	<input type="checkbox"/> Account\Manage\ManageNav	<input type="checkbox"/> Account\Manage\StatusMessage
<input type="checkbox"/> Account\Manage\ChangePassword	<input type="checkbox"/> Account\Manage\DeletePersonalData	<input type="checkbox"/> Account\Manage\Disable2fa
<input type="checkbox"/> Account\Manage\DownloadPersonalData	<input type="checkbox"/> Account\Manage\Email	<input type="checkbox"/> Account\Manage\EnableAuthenticator
<input type="checkbox"/> Account\Manage\ExternalLogins	<input type="checkbox"/> Account\Manage\GenerateRecoveryCodes	<input type="checkbox"/> Account\Manage\Index
<input type="checkbox"/> Account\Manage\PersonalData	<input type="checkbox"/> Account\Manage\ResetAuthenticator	<input type="checkbox"/> Account\Manage\SetPassword
<input type="checkbox"/> Account\Manage>ShowRecoveryCodes	<input type="checkbox"/> Account\Manage\TwoFactorAuthentication	<input type="checkbox"/> Account\Register
<input type="checkbox"/> Account\RegisterConfirmation	<input type="checkbox"/> Account\ResetPassword	<input type="checkbox"/> Account\ResetPasswordConfirmation

Data context class: 9

☐ Use SQLite instead of SQL Server

User class: 10

We can find the Areas in the application with Data & Razor pages, as shown below:



Now let's add user authentication to the application. Open the Startup class and Modify as shown below.

In order to add support to the razor pages, we have to call the function `services.AddRazorPages()` and `endpoints.MapRazorPages()`

```
01. public class Startup
02. {
03.     public Startup(IConfiguration configuration)
04.     {
05.         Configuration = configuration;
06.     }
07.
08.     public IConfiguration Configuration { get; }
09.
10.     // This method gets called by the runtime. Use this method to add
11.     public void ConfigureServices(IServiceCollection services)
12.     {
13.         services.AddControllersWithViews();
14.         services.AddRazorPages();
15.     }
16.
17.     // This method gets called by the runtime. Use this method to configure
18.     public void Configure(IApplicationBuilder app, IWebHostEnvironment env)
19.     {
20.         if (env.IsDevelopment())
21.         {
22.             app.UseDeveloperExceptionPage();
23.         }
24.         else
25.         {
26.             app.UseExceptionHandler("/Home/Error");
27.             // The default HSTS value is 30 days. You may want to change this to a shorter
28.             hsts. app.UseHsts();
29.         }
30.         app.UseHttpsRedirection();
31.         app.UseStaticFiles();
32.
33.         app.UseRouting();
34.         app.UseAuthentication();
35.
36.         app.UseAuthorization();
37.
38.         app.UseEndpoints(endpoints =>
39.         {
40.             endpoints.MapControllerRoute(
41.                 name: "default",
42.                 pattern: "{controller=Home}/{action=Index}/{id?}");
43.             endpoints.MapRazorPages();
44.         });
45.     }
```


Step 4

Now let's start with creating the database for the application.

Open UsingIdentityUser class and add the properties and decorate with the attribute PersonalData.

```
01. public class UsingIdentityUser : IdentityUser
02. {
03.     [PersonalData]
04.     [Column(TypeName = "nvarchar(100)")]
05.     public string Firstname { get; set; }
06.     [PersonalData]
07.     [Column(TypeName = "nvarchar(100)")]
08.     public string LastName { get; set; }
09. }
```

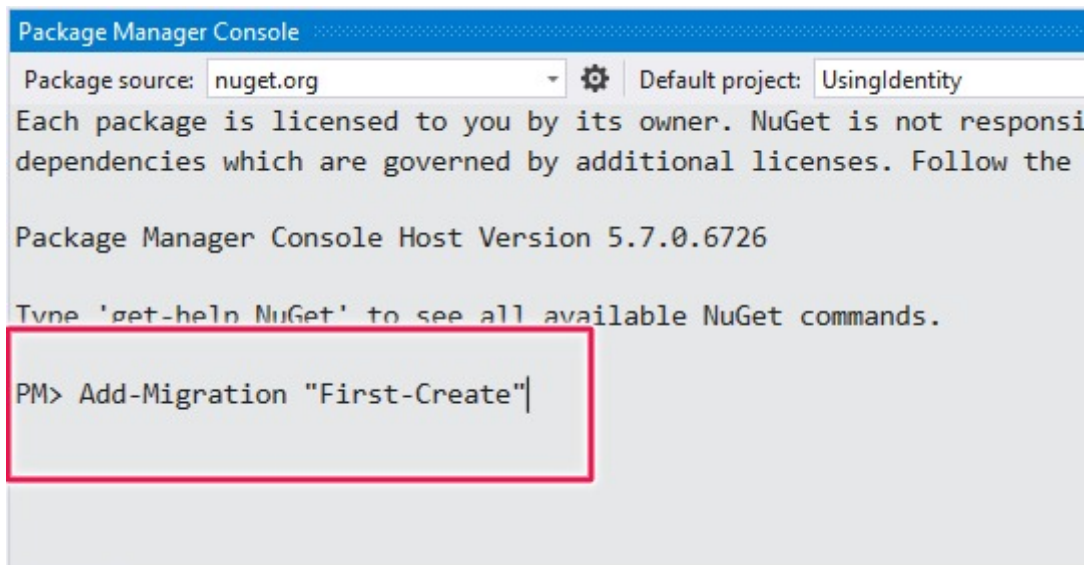
We have Dbcontext, which is also inherited from the parent class IdentityDB context. This identity db context is injected inside this identityhostingstartup class.

```
01. public class IdentityHostingStartup : IHostingStartup
02. {
03.     public void Configure(IWebHostBuilder builder)
04.     {
05.         builder.ConfigureServices((context, services) => {
06.             services.AddDbContext<UsingIdentityContext>
07.             (options =>
08.                 options.UseSqlServer(
09.                     context.Configuration.GetConnectionString("Using
10.                 services.AddDefaultIdentity<UsingIdentityUser>
11.             (options => options.SignIn.RequireConfirmedAccount = true)
12.                 .AddEntityFrameworkStores<UsingIdentityContext>
13.             ());
14.         }));
15.     }
16. }
```

Now open appsettings.json. We can find the connection string with the name UsingIdentityContextConnection. By default, it will connect to the local DB.

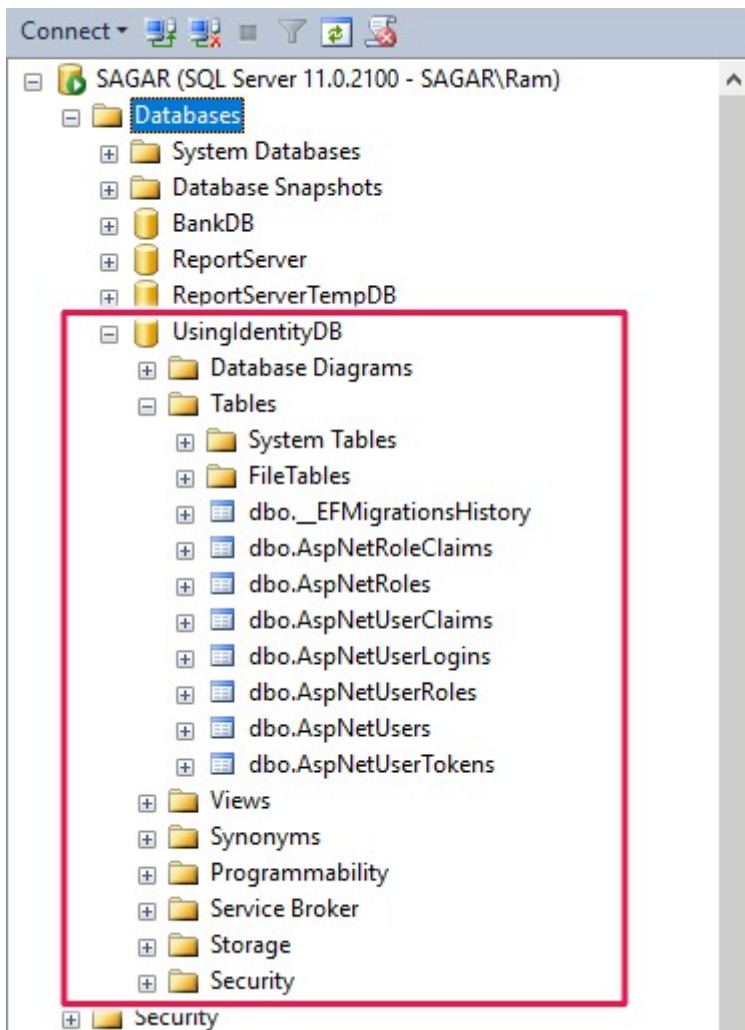
```
01. {
02.     "Logging": {
03.         "LogLevel": {
04.             "Default": "Information",
05.             "Microsoft": "Warning",
06.             "Microsoft.Hosting.Lifetime": "Information"
07.         }
08.     },
09.     "AllowedHosts": "*",
10.     "ConnectionStrings": {
11.         "UsingIdentityContextConnection": "Server=SAAD;Database=UsingIdent
```

Now open the Package manager console and Execute the command Add-Migration "First-Create" to generate the actual physical DB



```
Package Manager Console
Package source: nuget.org
Default project: UsingIdentity
Each package is licensed to you by its owner. NuGet is not responsible for the
dependencies which are governed by additional licenses. Follow the
Package Manager Console Host Version 5.7.0.6726
Type 'get-help NuGet' to see all available NuGet commands.
PM> Add-Migration "First-Create"
```

Finally, execute the command Update-Database and you can find the new database as shown below.



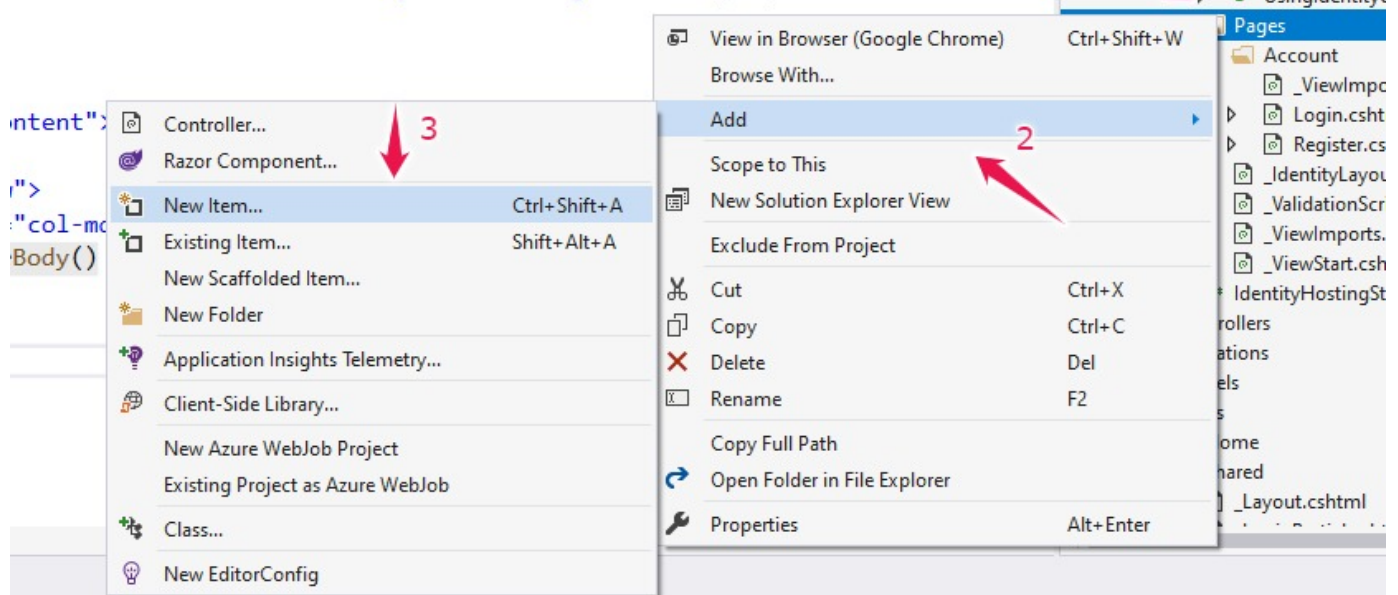
	Column Name	Data Type	Allow Nulls
▶	Id	nvarchar(450)	<input type="checkbox"/>
	UserName	nvarchar(256)	<input checked="" type="checkbox"/>
	NormalizedUserName	nvarchar(256)	<input checked="" type="checkbox"/>
	Email	nvarchar(256)	<input checked="" type="checkbox"/>
	NormalizedEmail	nvarchar(256)	<input checked="" type="checkbox"/>
	EmailConfirmed	bit	<input type="checkbox"/>
	PasswordHash	nvarchar(MAX)	<input checked="" type="checkbox"/>
	SecurityStamp	nvarchar(MAX)	<input checked="" type="checkbox"/>
	ConcurrencyStamp	nvarchar(MAX)	<input checked="" type="checkbox"/>
	PhoneNumber	nvarchar(MAX)	<input checked="" type="checkbox"/>
	PhoneNumberConfirmed	bit	<input type="checkbox"/>
	TwoFactorEnabled	bit	<input type="checkbox"/>
	LockoutEnd	datetimeoffset(7)	<input checked="" type="checkbox"/>
	LockoutEnabled	bit	<input type="checkbox"/>
	AccessFailedCount	int	<input type="checkbox"/>
	Firstname	nvarchar(100)	<input checked="" type="checkbox"/>
	LastName	nvarchar(100)	<input checked="" type="checkbox"/>
			<input type="checkbox"/>

Now let's start customizing the application. Let's add a Nested layout for tab control headers.

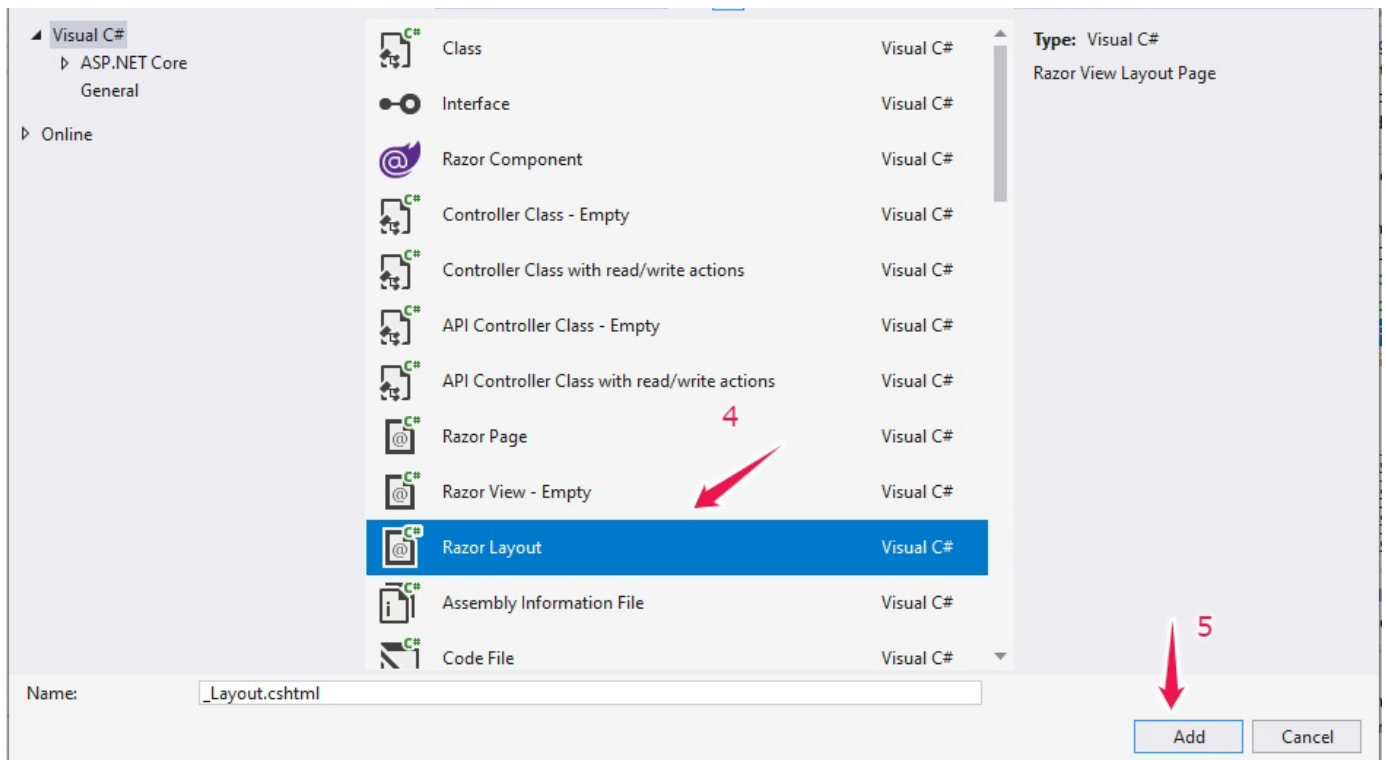
```

<div class="col-md-3">
  <div class="logout-tab">
    <div class="card-header">
      <div class="nav-tabs card-header-tabs">
        <div class="nav-item">
          <a class="nav-link" href="/Identity/Account/Login">Log In</a>
        </div>
        <div class="nav-item">
          <a class="nav-link" href="/Identity/Account/Register">Sign Up </a>
        </div>
      </div>
    </div>
  </div>
</div>

```



Select Razor layout and click on Add



Customize the layout as below

```

01. @{
02.     Layout = "/Views/Shared/_Layout.cshtml";
03. }
04. <div class="row">
05.     <div class="col-md-6 offset-md-3">
06.         <div class="card login-logout-tab">
07.             <div class="card-header">
08.                 <ul class="nav nav-tabs card-header-tabs">
09.                     <li class="nav-item">
10.
11.                         <a class="nav-
link" href="/Identity/Account/Login">Log In</a>
12.                     </li>
13.                     <li class="nav-item">
14.
15.                         <a class="nav-
link" href="/Identity/Account/Register">Sign Up </a>
16.                     </li>
17.                 </ul>
18.             </div>
19.             <div class="card-content">
20.
21.                 <div class="row">
22.                     <div class="col-md-12 ">
23.                         @RenderBody()
24.                     </div>
25.                 </div>
26.             </div>
27.         </div>
28.     </div>

```

```

31. @section Scripts{
32.
33.     @RenderSection("Scripts", required: false)
34.     <script>
35.         $(function () {
36.
37.             var current = location.pathname;
38.             $('.nav-tabs li a').each(function () {
39.                 var $this = $(this);
40.                 if (current.indexOf($this.attr('href')) !== -1) {
41.                     $this.addClass('active');
42.                 }
43.             })
44.         })
45.     </script>
46. }

```

Open site.css and use the below styles

```

01. /*for Tab control*/
02. div.login-logout-tab div.card.header{
03.     padding: 0px 0px 12px 0px;
04. }
05. div.login-logout-tab li.nav-tabs{
06.
07.     margin: 0px 0px -12px 0px;
08. }
09. div.login-logout-tab li.nav-item{
10.
11.     width:50%
12. }
13. div.login-logout-tab a.nav-link{
14.     font-size:25px;
15.     color:#495057;
16.     text-align:center;
17. }

```

Modify the Login page and register the HTML pages with our new layout.

```

01. @page
02. @model LoginModel
03.
04. @{
05.     ViewData["Title"] = "Log in";
06.     Layout = "~/Areas/Identity/Pages/_IdentityLayout.cshtml";
07. }
08.
09. <div class="col-md-10 offset-md-1">
10.     <section>
11.         @*<div class="login-form-icon">
12.             <i class="fas fa-user-circle fa-5x text-secondary">
13.
14.         </div>*@
15.         @*fas fa-user-circle fa-9x text-secondary*@
16.         <form id="account" method="post">

```

```

17.         <div class="form-group">
18.             <label asp-for="Input.Email"></label>
19.             <input asp-for="Input.Email" class="form-
control" />
20.             <span asp-validation-for="Input.Email" class="text-
danger"></span>
21.         </div>
22.         <div class="form-group">
23.             <label asp-for="Input.Password"></label>
24.             <input asp-for="Input.Password" class="form-
control" />
25.             <span asp-validation-for="Input.Password" class="text-
danger"></span>
26.         </div>
27.         <div class="form-group">
28.             <div class="checkbox">
29.                 <label asp-for="Input.RememberMe">
30.                     <input asp-for="Input.RememberMe" />
31.                     @Html.DisplayNameFor(m => m.Input.RememberMe)
32.                 </label>
33.             </div>
34.         </div>
35.         <div class="form-group">
36.             <button type="submit" class="btn btn-primary btn-
block">Log in</button>
37.         </div>
38.     </form>
39.
40. </section>
41. </div>
42.
43. @section Scripts {
44.     <partial name="_ValidationScriptsPartial" />
45. }
46.
47. @page
48. @model RegisterModel
49. @{
50.     ViewData["Title"] = "Register";
51.     Layout = "~/Areas/Identity/Pages/_IdentityLayout.cshtml";
52. }
53.
54. <h1>@ViewData["Title"]</h1>
55.
56.
57. <form asp-route-returnUrl="@Model.ReturnUrl" method="post">
58.     <div asp-validation-summary="All" class="text-danger"></div>
59.     <div class="row">
60.         <div class="col-md-6">
61.             <div class="form-group">
62.                 <label asp-for="Input.FirstName"></label>
63.                 <input asp-for="Input.FirstName" class="form-
control" />
64.                 <span asp-validation-
for="Input.FirstName" class="text-danger"></span>
65.             </div>

```

```

68.         <div class="col-md-6">
69.             <div class="form-group">
70.                 <label asp-for="Input.LastName"></label>
71.                 <input asp-for="Input.LastName" class="form-
control" />
72.                 <span asp-validation-for="Input.LastName" class="text-
danger"></span>
73.             </div>
74.         </div>
75.     </div>
76.     <div class="row">
77.         <div class="col-md-6">
78.             <div class="form-group">
79.                 <label asp-for="Input.Email"></label>
80.                 <input asp-for="Input.Email" class="form-
control" />
81.                 <span asp-validation-for="Input.Email" class="text-
danger"></span>
82.             </div>
83.         </div>
84.         <div class="col-md-6">
85.             <div class="form-group">
86.                 <label asp-for="Input.PhoneNumber"></label>
87.                 <input asp-for="Input.PhoneNumber" class="form-
control" />
88.                 <span asp-validation-
for="Input.PhoneNumber" class="text-danger"></span>
89.             </div>
90.         </div>
91.     </div>
92.
93.     <div class="row">
94.         <div class="col-md-6">
95.             <div class="form-group">
96.                 <label asp-for="Input.Password"></label>
97.                 <input asp-for="Input.Password" class="form-
control" />
98.                 <span asp-validation-for="Input.Password" class="text-
danger"></span>
99.             </div>
100.        </div>
101.        <div class="col-md-6">
102.            <div class="form-group">
103.                <label asp-for="Input.ConfirmPassword"></label>
104.                <input asp-for="Input.ConfirmPassword" class="form-
control" />
105.                <span asp-validation-
for="Input.ConfirmPassword" class="text-danger"></span>
106.            </div>
107.        </div>
108.    </div>
109.
110.    <button type="submit" class="btn btn-
primary">Register</button>
111. </form>
112.

```

```
115. | }
```

In order to display the active tab, we need to add an active class, as shown below.

```
01. | @section Scripts{
02. |
03. |     @RenderSection("Scripts", required: false)
04. |     <script>
05. |         $(function () {
06. |
07. |             var current = location.pathname;
08. |             $('.nav-tabs li a').each(function () {
09. |                 var $this = $(this);
10. |                 if (current.indexOf($this.attr('href')) !== -1) {
11. |                     $this.addClass('active');
12. |                 }
13. |             })
14. |         })
15. |     </script>
16. | }
```

Let's see the output and register a user.

Log In

Sign Up

Register

First Name

Ram

Last Name

Sagar

Email

sagar@gmail.com

Phone Number

9848032919

Password

.....

Confirm password

.....

Register

Results		Messages					
	Id	UserName	NormalizedUserName	Email	NormalizedEmail	EmailConfirmed	PasswordHash
1	74c386c-095b-4054-a618-c92b26c6573d	sagar@gmail.com	SAGAR@GMAIL.COM	sagar@gmail.com	SAGAR@GMAIL.COM	1	AQAAAAEAACcQAAAAEKwSg+hU6wqDqIPa49vQOP f

Now let's login with the registered user.

Log In

Sign Up

Email

sagar@gmail.com

Password

.....|

☐ Remember me?

Log in

Hello sagar@gmail.com! Logout

Welcome

Learn about [building Web apps with ASP.NET Core](#).

Conclusion

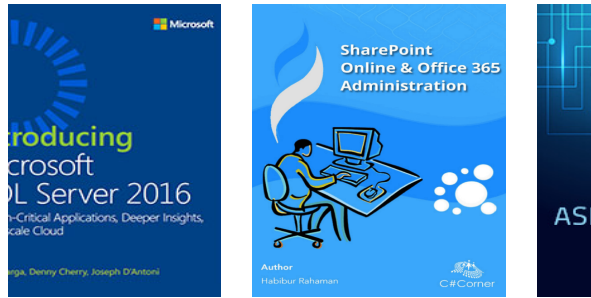
In this article, we discussed how to use Identity UI in ASP.NET Core MVC application by creating a database using the package manager console & commands. I hope you all enjoyed reading this and learned from it.

[.Net core](#)[ASP.NET.CORE](#)[Authentication using Identity](#)[Identity](#)

Next Recommended Reading

[ASP.NET Core 2.0 JWT Authentication Example](#)

OUR BOOKS



Ramasagar Pulidindi **TOP 500**

"A Journey of Thousand miles must start with a single Step" I started my programming career with C/C++, Java in my B. Tech days. Later, I got a chance to develop Windows Forms applications using C# and Web appli... [Read more](#)

<https://www.c-sharpcorner.com/members/ramasagar-pulidindi>

198

1.4m

3

9

1



Type your comment here and press Enter Key (Minimum 10 characters)



After Login, the Hello @Username and logout doesn't appear

FPL Pramit

NA 71 0

Mar 22, 2021

0 0 Reply

FEATURED ARTICLES

Test Coverage For PowerShell Scripts With Pester

Dynamic CSS Values In Blazor

Migrating a SQL IaaS Database To Azure SQL Using DMA (Data Migration Assistant)

Create a Redis Cache with Express Node JS

Symmetrical Repository Pattern - Data Access Made Easy In .NET

TRENDING UP

01 Azure Duration Functions - How To Use And Implement It

- 03 Log Correlation In Microservices
- 04 Create A PowerApps Component Framework (PCFx) Using Custom Code In PowerAPPS
- 05 How To Handle Nullable Reference In .NET 6
- 06 Import PowerApps Component Framework (PCFx) Into Model Driven PowerApps
- 07 Growth Mindset Show Ep. 11 - 2022
- 08 Safest Way To Convert String To Int In C#
- 09 Creating Various Layouts For Different Razor Pages In Blazor
- 10 Top Three VS Code Extensions Worth The Money



Learn JavaScript

CHALLENGE YOURSELF



Blockchain Basics Skill

GET CERTIFIED



HTML5 Developer

[About Us](#) [Contact Us](#) [Privacy Policy](#) [Terms](#) [Media Kit](#) [Sitemap](#) [Report a Bug](#) [FAQ](#) [Partners](#)

[C# Tutorials](#) [Common Interview Questions](#) [Stories](#) [Consultants](#) [Ideas](#) [Certifications](#)

©2022 C# Corner. All contents are copyright of their authors.