Sql server, .net and c# video tutorial

Free C#, .Net and Sql server video tutorial for beginners and intermediate programmers.

ASP.NET Core Identity tutorial from scratch

Suggested Videos

Part 62 - Logging exceptions in ASP.NET Core | Text | Slides
Part 63 - Logging to file in asp.net core using nlog | Text | Slides
Part 64 - ASP.NET Core LogLevel configuration | Text | Slides

In this video and in our upcoming videos in this series we will discuss everything you need to effectively use **ASP.NET Core Identity** to implement security related features in your asp.net core application.

ASP.NET Core Identity

ASP.NET Core Identity is a membership system. It allows us to create, read, update and delete user accounts. Supports account confirmation, authentication, authorisation, password recovery, two-factor authentication with SMS. It also supports external login providers like Microsoft, Facebook, Google etc. We will discuss implementing these features in our upcoming videos in this series.

Adding ASP.NET Core Identity Support in ASP.NET Core

Complete Tutorials

JavaScript tutorial

Bootstrap tutorial

Angular tutorial for beginners

Angular 5 Tutorial for beginners

Important Videos

The Gift of Education

Web application for your business

Application

The following are the steps to add and configure ASP.NET Core Identity

Step 1 : Inherit from IdentityDbContext class

```
public class AppDbContext : IdentityDbContext
{
    // Rest of the code
}
```

Your application DbContext class must inherit from IdentityDbContext class instead of DbContext class. This is required because IdentityDbContext provides all the DbSet properties needed to manage the identity tables in SQL Server. We will see all the tables that the asp.net core identity framework generates in just a bit. If you go through the hierarchy chain of IdentityDbContext class, you will see it inherits from DbContext class. So this is the reason you do not have to explicitly inherit from DbContext class if your class is inheriting from IdentityDbContext class.

Step 2 : Configure ASP.NET Core Identity Services

In ConfigureServices() method of the Startup class, include the following line of code.

```
services.AddIdentity<IdentityUser, IdentityRole>()
        .AddEntityFrameworkStores<AppDbContext>();
```

- AddIdentity() method adds the default identity system configuration for the specified user and role types.
- IdentityUser class is provided by ASP.NET core and contains properties for UserName, PasswordHash, Email etc. This is the class that is used by default by the ASP.NET Core Identity framework to manage registered users of your application.
- If you want store additional information about the registered users like their Gender, City etc. Create a custom class that derives from IdentityUser. In this custom class add the additional properties you need and then plug-in

How to become .NET developer

Resources available to help you

Dot Net Video Tutorials

ASP.NET Core Tutorial

Angular 6 Tutorial

Angular CRUD Tutorial

Angular CLI Tutorial

Angular 2 Tutorial

Design Patterns

SOLID Principles

ASP.NET Web API

Bootstrap

AngularJS Tutorial

jQuery Tutorial

JavaScript with ASP.NET Tutorial

JavaScript Tutorial

Charts Tutorial

LINQ

LINQ to SQL

LINQ to XML

Entity Framework

this class instead of the built-in IdentityUser class. We will discuss how to do this in our upcoming videos.

- Similarly, IdentityRole is also a builtin class provided by ASP.NET Core Identity and contains Role information.
- We want to store and retrieve User and Role information of the registered users using EntityFrameWork Core from the underlying SQL Server database. We specify this using AddEntityFrameworkStores<AppDbContext>() passing our application DbContext class as the generic argument.

Step 3 : Add Authentication middleware to the request pipeline

```
public void Configure(IApplicationBuilder app, IHostingEnvironment env)
{
    if (env.IsDevelopment())
    {
        app.UseDeveloperExceptionPage();
    }
    else
    {
        app.UseExceptionHandler("/Error");
        app.UseStatusCodePagesWithReExecute("/Error/{0}");
    }

    app.UseStaticFiles();
    app.UseAuthentication();
    app.UseMvc(routes =>
    {
        routes.MapRoute("default", "{controller=Home}/{action=Index}/{id?}");
    });
}
```

In the Configure() method of the Startup class, call UseAuthentication() method to add the Authentication middleware to the application's request processing pipeline. We want to be able to authenticate users before the request reaches the MVC middleware. So it's important we add authentication middleware before the MVC middleware in the request processing pipeline.

Step 4 : Add Identity Migration

WCF

ASP.NET Web Services

Dot Net Basics

C#

SQL Server

ADO.NET

ASP.NET

GridView

ASP.NET MVC

Visual Studio Tips and Tricks

Dot Net Interview Questions

Slides

Entity Framework

WCF

ASP.NET Web Services

Dot Net Basics

C#

SQL Server

ADO.NET

ASP.NET

In Visual Studio, from the Package Manager Console window execute the following command to add a new migration

```
Add-Migration AddingIdentity
```

This migration contains code that creates the tables required by the ASP.NET Core Identity system.

Error : The entity type 'IdentityUserLogin<string>' requires a primary key to be defined

If you get this error, the most likely cause is that you are overriding OnModelCreating() method in your application DbContext class but not calling the base IdentityDbContext class OnModelCreating() method.

Keys of Identity tables are mapped in OnModelCreating method of IdentityDbContext class. So, to fix this error, all you need to do is, call the base class OnModelCreating() method using the base keyword as shown below.

```
public class AppDbContext : IdentityDbContext
{
    public AppDbContext(DbContextOptions<AppDbContext> options)
        : base(options)
    {
    }

    public DbSet<Employee> Employees { get; set; }

    protected override void OnModelCreating(ModelBuilder modelBuilder)
    {
        base.OnModelCreating(modelBuilder);
        modelBuilder.Seed();
    }
}
```

Step 4 : Generate ASP.NET Core Identity Tables

Finally execute Update-Database command to apply the identity migration and have the required identity tables created.

GridView

ASP.NET MVC

Visual Studio Tips and Tricks

Java Video Tutorials

Part 1 : Video | Text | Slides

Part 2 : Video | Text | Slides

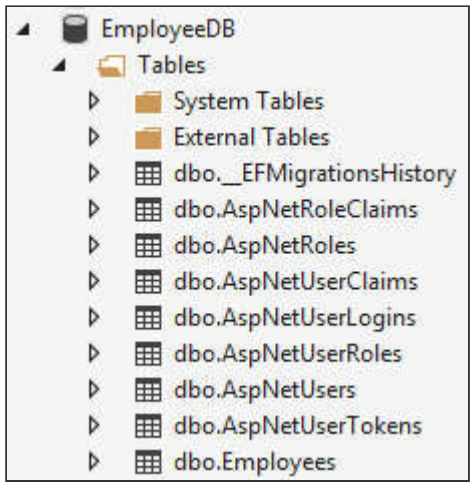
Part 3 : Video | Text | Slides

Interview Questions

C#

SQL Server

Written Test



Next Video : **How to register a new user using asp.net core identity**

No comments:

Post a Comment

If you like this website, please share with your friends on facebook and Google+ and recommend us on google using the g+1 button on the top right hand corner.

Enter your comment...

Comment as: toilati123vn@g

Publish Preview

Links to this post

Create a Link

Newer Post Home Older Post

Subscribe to: Post Comments (Atom)