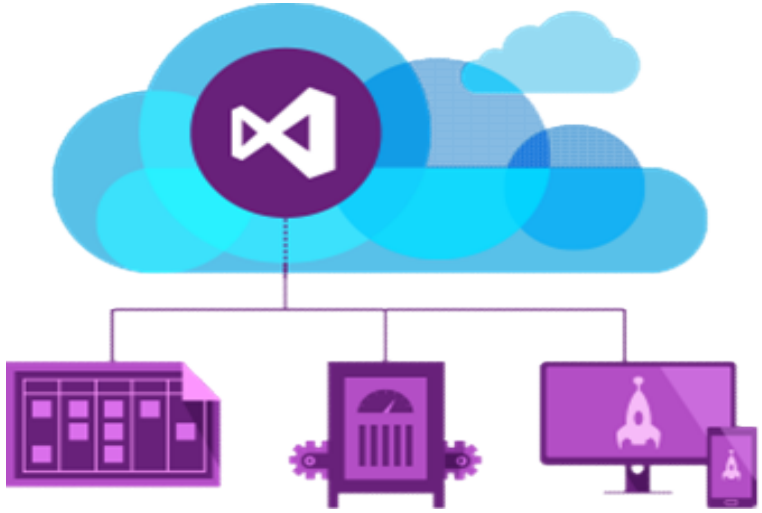


.NET Core

Mohamed ELshafei

ASP.NET Core Identity

- **Is an API that supports user interface (UI) login functionality.**
- **Manages users, passwords, profile data, roles, claims, tokens, email confirmation, and more.**

- ASP.NET Core Identity is a membership system that adds login functionality to an ASP.NET Core application.

- Identity comprises a framework for managing user authentication, authorization and other related capabilities..

What Is ASP.NET Core Identity and What Are Its Advantages?

- **User authentication and authorization:**
  - Identity provides a robust user authentication and authorization system in ASP.NET Core applications. It supports multiple authentication methods, including cookies, tokens and external login providers (like Google or Facebook).
- **User and role management:**
  - Identity allows you to manage user accounts and roles in your application. You can easily create, update, delete and recover user information. Additionally, you can assign users to roles and manage role-based authorizations.

What Is ASP.NET Core Identity and What Are Its Advantages?

- **Password hashing:**
  - Security is a top priority, and Identity uses secure password hashing techniques to store user passwords. It employs industry-standard algorithms like PBKDF2 with HMAC-SHA256.
- **Two-Factor Authentication (2FA):**
  - Identity supports two-factor authentication, providing an extra layer of security for user accounts. Users can receive codes via email, SMS or authenticator apps.
- **Token-based authentication:**
  - Identity supports token-based authentication, allowing you to generate and validate tokens for secure communication between applications.

What Is ASP.NET Core Identity and What Are Its Advantages?

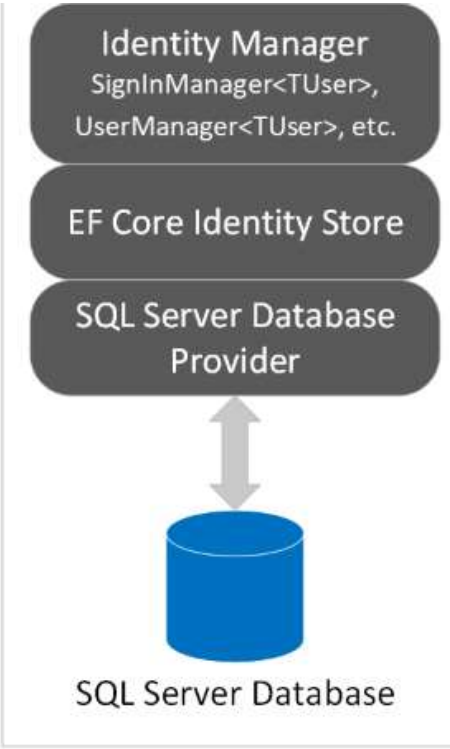
- **Integration with ASP.NET Core:**
  - ASP.NET Core Identity is fully integrated into the ASP.NET Core framework, making it easy to incorporate into your web applications. It leverages the built-in dependency injection system and works seamlessly with other ASP.NET Core components.
- **Customization and extensibility:**
  - Identity is designed to be customizable and extensible. You can modify the default data model, implement custom validation and extend the system to meet your application’s requirements.

What Is ASP.NET Core Identity and What Are Its Advantages?

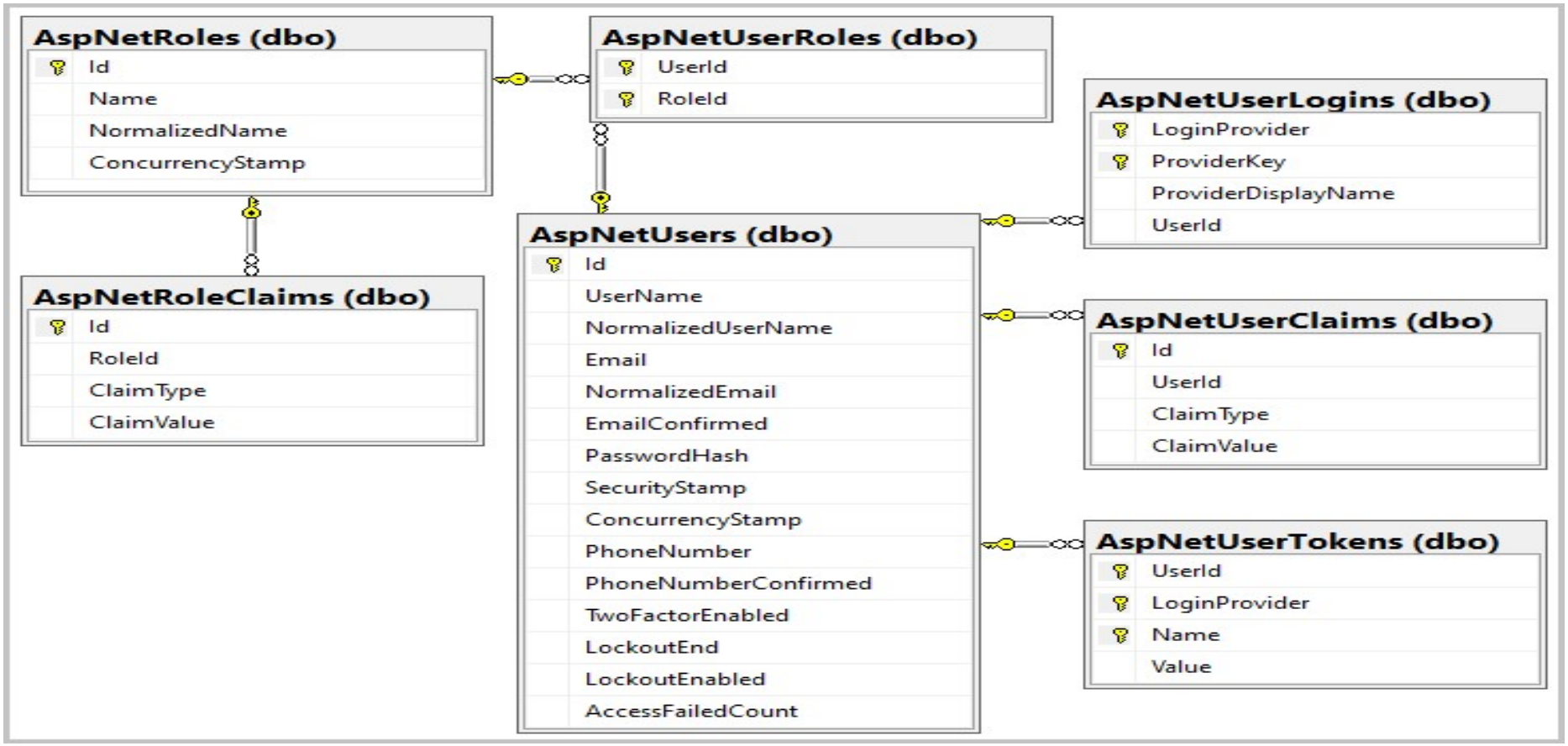
- **Persistence Providers:**
  - Identity supports multiple storage providers for user data, such as Entity Framework Core, allowing you to choose the appropriate storage solution for your application.
- **Claims-Based Authorization:**
  - Identity allows you to use claims for more granular and flexible authorization. Claims represent specific attributes about a user, and you can use them to make access control decisions.
- **ASP.NET Core Identity UI:**
  - Identity comes with pre-built UI components that you can use in your app, including login, registration, password recovery and account management views. This can save development time and ensure a consistent user experience.

ASP.NET Core Identity architecture

- The Identity Manager layer contains classes used from the Microsoft.AspNetCore.Identity namespace. Llike SignInManager<TUser> and UserManager<TUser>.
- The EF Core Identity Store layer contains classes from the Microsoft.AspNetCore.Identity.EntityFrameworkCore namespace. Like UserStore<TUser>.
- The Database Provider is a database-specific library that accepts SQL from the EF Core Provider (not pictured) and executes it.



ASP.NET Core Identity architecture



ASP.NET Core Identity architecture

- The following table explains the role of each table with their entity name

Entity	Table Name	Remarks
IdentityUser	AspNetUsers	Primary table to store user information
IdentityUserClaim	AspNetUserClaims	tables holds the claims associated with the user.
IdentityUserLogin	AspNetUserLogins	table holds the information about 3rd party/external logins
IdentityUserToken	AspNetUserTokens	is for storing tokens received from the external login providers.
IdentityUserRole	AspNetUserRoles	table contains the roles assigned to the user
IdentityRole	AspNetRoles	tables to store the roles
IdentityRoleClaim	AspNetRoleClaims	The claims that are assigned to the Role

Configure Identity support

- **Add ASP.NET Core Identity to the project**
- Install package:

```
Microsoft.AspNetCore.Identity.EntityFrameworkCore --version 8.0.*
```

- Configure Service :

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

- Add Middleware

```
app.UseAuthentication();
```

Identity customizations"Extending the Identity data model"

- By default, Identity represents a user with an IdentityUser class
- you created a class named RazorPagesPizzaUser that inherits from IdentityUser and modify the derived class to include properties to store the user's first and last name.
- Extending the data model requires changes to the underlying database. Luckily, Entity Framework Core makes this task simple with migrations.