LẬP TRÌNH WEB NÂNG CAO

Thái Thanh Tùng
FITHOU

BÀI 7: Web Security

- ✓Cookies và Session
- ✓Authentication và Authorization
- ✓ASP.NET Identity
- ✓Web Security và Lỗi an ninh

GIỚI THIỆU VỀ COOKIES

7.1 Giới thiệu Cookies

- Giao tiếp Web giữa máy chủ và máy khách sử dụng giao thức HTTP, còn được gọi là giao thức không trạng thái (State-less protocol, Request-Response Protocol). Điều này có nghĩa là thông tin của trình duyệt giữa hai request không được chia sẻ và gửi cho Server.

- Cookie là một tệp văn bản nhỏ mà trang web lưu trữ trên máy tính hoặc thiết bị di động của người dùng khi họ truy cập trang web. Tệp này chứa thông tin có thể được sử dụng để xác định người dùng, chẳng hạn như trạng thái đăng nhập hoặc tùy chọn của họ và được gửi lại trang web với mọi yêu cầu tiếp theo. Thông thường, nó được sử dụng để cho máy chủ web biết nếu các yêu cầu đến từ cùng một trình duyệt web.

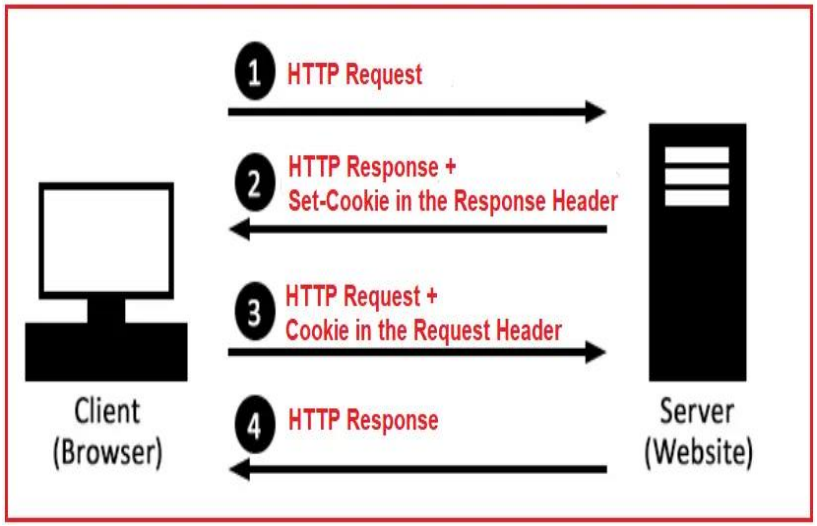
- Cookie được lưu trữ dưới dạng chuỗi cặp khóa-giá trị (NameValueCollection) và ta có thể tận dụng các khóa để đọc, ghi hoặc xóa cookie. Để hiểu rõ hơn về cách thức hoạt động của cookie. Cookie được phân loại là **Persistent cookie, Session cookie**

Cookies Trong ASP.NET Core MVC

7.1.1 Mô hình hoạt động

Mô hình hoạt động của Cookie:

- Client gửi Request **đầu tiên** từ (Web Browser) thông qua HTTPRequest tới Server.
- Server nhận và xử lý request và khởi tạo chuỗi HttpResponse. Khi tạo chuỗi response, server sẽ tạo chuỗi cookie và tích hợp trong phần header của response.
- Khi client nhận response, cookies được **lưu trữ trong bộ nhớ cache trên Client**. Với mỗi request tiếp theo được gửi từ client đó (nếu cookies chưa hết hạn hoặc bị xóa), trình duyệt sẽ **tự động** chèn cookie vào header của HTTPRequest.
- Server nhận request, và sẽ kiểm tra và nhận các dữ liệu trong cookies



Hình 3.1. Mô hình hoạt động của Cookie

Cookies Trong ASP.NET Core MVC

7.1.2. Làm việc với cookie trong ASP.net core MVC

Cookie trong ASP.NET Core MVC là một cơ chế được sử dụng để duy trì trạng thái, theo dõi tương tác của người dùng và cung cấp trải nghiệm được cá nhân hóa trong các ứng dụng web. ASP.NET Core MVC cung cấp hỗ trợ tích hợp để làm việc với cookie để kích hoạt các tình huống khác nhau, chẳng hạn như xác thực, quản lý phiên và theo dõi sở thích của người dùng ...

Một số đặc điểm chính của cookie:

- **Lưu trữ trên Client:** Cookie là dữ liệu văn bản nhỏ được gửi từ máy chủ và được lưu trữ ở phía máy khách (trong trình duyệt của người dùng). Mỗi cookie được liên kết với một tên miền và đường dẫn cụ thể trên máy chủ.
- **Duy trì trạng thái:** Cookie thường được sử dụng để duy trì trạng thái của user. Ví dụ: họ có thể theo dõi xem người dùng có được xác thực hay lưu trữ tùy chọn của người dùng qua các phiên.
- **Xác thực và phân quyền:** Cookie thường được sử dụng cho mục đích xác thựcđịnh danh người dùng đăng nhập, máy chủ có thể tạo cookie chứa mã định danh cho người dùng được xác thực. Các yêu cầu tiếp theo có thể bao gồm cookie để xác nhận danh tính và quyền truy cập của người dùng
- **Quản lý phiên làm việc:** Cookie có thể được sử dụng để duy trì thông tin phiên ở phía người dùng. ASP.NET Core cung cấp các tính năng quản lý phiên sử dụng cookie để lưu trữ mã định danh phiên.
- **Cá nhân hóa:** Cookie có thể được sử dụng để lưu trữ tùy chọn hoặc cài đặt của người dùng. Ví dụ: một trang web có thể nhớ tùy chọn ngôn ngữ của người dùng hoặc tùy chọn hiển thị bằng cookie.
- **Theo dõi và quảng cáo:** Cookie được sử dụng để theo dõi các tương tác và hành vi của người dùng trên các trang web. Dữ liệu này có thể được sử dụng để phân tích, theo dõi hành vi của người dùng và cung cấp quảng cáo định hướng mục tiêu
- **Thời hạn lưu trữ:** Cookie có thể được thiết lập ngày hết hạn hoặc được đặt thành liên tục. Cookie phiên hết hạn khi người dùng đóng trình duyệt, trong khi cookie liên tục vẫn còn trên hệ thống của người dùng trong một khoảng thời gian xác định
- **Bảo mật và HttpOnly:** Cookie có thể được định cấu hình an toàn (chỉ được gửi qua kết nối HTTPS) và HttpOnly (không thể truy cập qua JavaScript, cải thiện bảo mật chống lại các tấn công cross-site).

Cookies Trong ASP.NET Core MVC

7.1.2. Làm việc với cookie trong ASP.net core MVC

Tạo và ghi cookie:

Cookie trong ASP.NET Core MVC, được tạo bởi lớp CookieOptions, thiết lập các thuộc tính, sau đó thêm cookie tập hợp Response.Cookies bằng phương thức Append()

```
CookieOptions options = new CookieOptions();
options.Expires = DateTime.Now.AddDays(7);
Response.Cookies.Append("UserId", "123456", options);
Response.Cookies.Append("UserName", "ttt@hou.edu.vn", options);
```

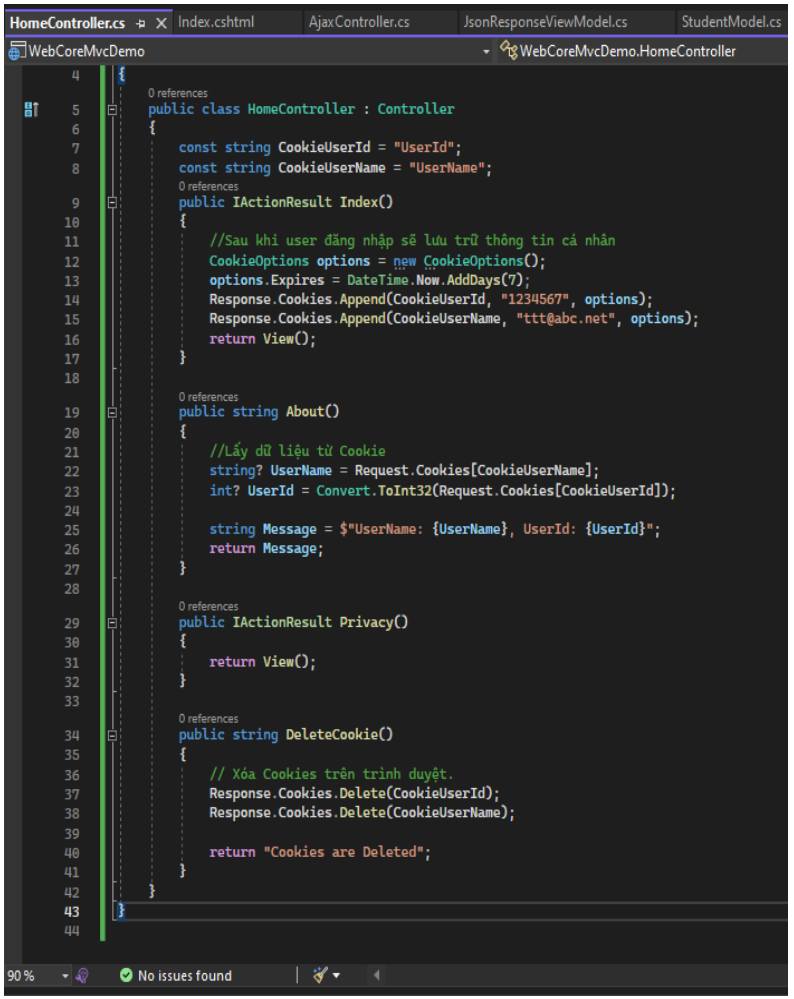
- **Domain:** Get hoặc set domain cookies được đọc ghi.
- **Expiration time:** Thời hạn của cookie được lưu trữ phía Client.
- **Path:** Đường dẫn các modules trên Server được đọc ghi cookie. Đường dẫn mặc định ứng dụng '/'.
- **Secure:** Thiết lập kênh mã hóa giá trị cookies với Secure Sockets Layer (SSL) HTTPS.
- **HttpOnly:** Cookie có thể truy cập bởi mã Client. Nếu giá trị của nó được đặt thành true, thì nó không thể được truy cập từ JavaScript.
- **MaxAge:** Thời hạn tối đa của Cookie.
- **IsEssential:** Xác định cookie là cần thiết để ứng dụng hoạt động chính xác. Nếu đúng, bạn có thể bỏ qua quy trình kiểm tra chính sách về sự đồng ý. Giá trị mặc định là false.

Cookies Trong ASP.NET Core MVC

7.1.2. Làm việc với Cookie trong ASP.net core MVC

Tạo và ghi cookie:

- Cookie được tạo và ghi trong các Action Methods của Controller, thường là HomeController
- Thiết lập các thuộc tính cho Cookies vơi đối tượng CookieOptions



Cookies Trong ASP.NET Core MVC

7.1.2. Làm việc với Cookie trong ASP.net core MVC

Đọc cookie từ View:

- @using Microsoft.AspNetCore.Http;
- @inject IHttpContextAccessor HttpContextAccessor

Đăng ký dịch vụ DI trong **program.cs**

```
builder.Services.AddSingleton<IHttpContextAccessor,
        HttpContextAccessor>();
```

Gọi HttpContextAccessor để truy cập các đối tượng HttpContext và Request, Session.

```
@HttpContextAccessor?.HttpContext?
        .ReQuest.Cookies[“Cookie_Name”]
```

Program.cs | Privacy.cshtml | HomeController.cs | Index.cshtml | AjaxController.cs | JsonResponseViewModel.cs

WebCoreMvcDemo

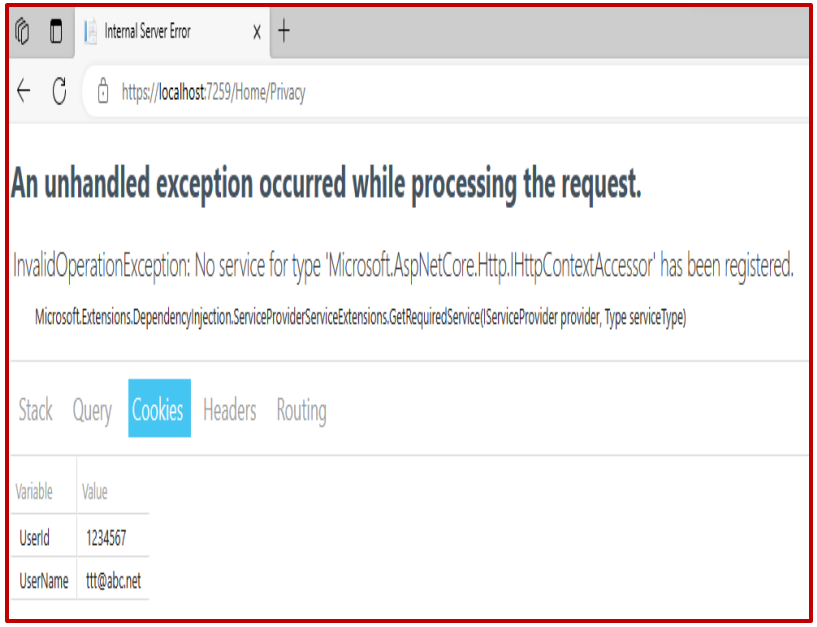
```
using Microsoft.EntityFrameworkCore;
using WebMvcDatabase.DBContext;
using WebMvcDatabase.Models;

var builder = WebApplication.CreateBuilder(args);

// Add services to the container.
builder.Services.AddControllersWithViews();
builder.Services.AddDbContext<StudentDbContext>(options =>
    options.UseSqlServer(builder.Configuration.GetConnectionString("StudentEFcoreDB")
    ?? throw new InvalidOperationException("Connection string 'StudentEFcoreDB' not found.")));

builder.Services.AddSingleton<IHttpContextAccessor, HttpContextAccessor>();

var app = builder.Build();

using (var scope = app.Services.CreateScope())
{
    var services = scope.ServiceProvider;

    ImportData.Initialize(services);
}
```

Privacy.cshtml | HomeController.cs | Index.cshtml | AjaxController.cs | JsonResponseViewModel.cs

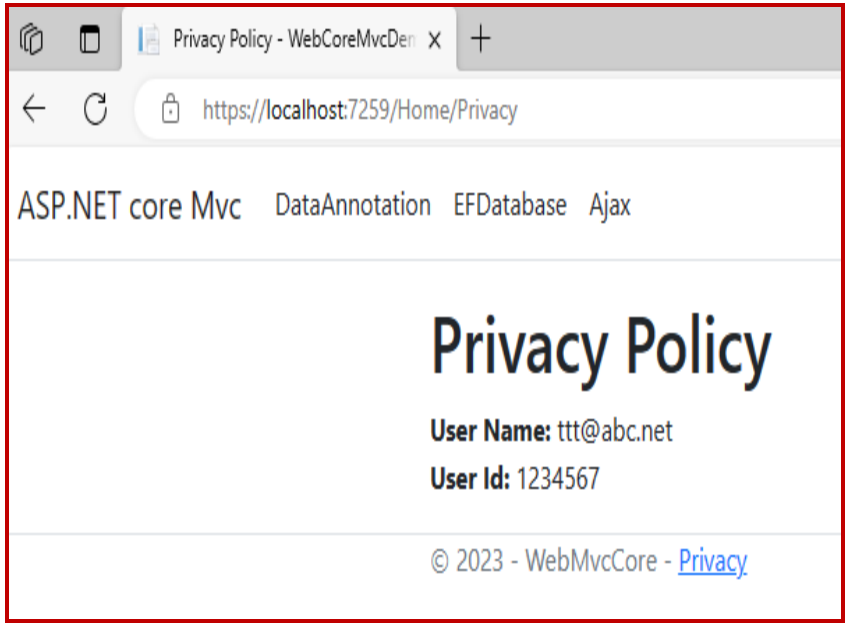
```
@using Microsoft.AspNetCore.Http;
@inject IHttpContextAccessor HttpContextAccessor

@{
    ViewData["Title"] = "Privacy Policy";
}
<h1>@ViewData["Title"]</h1>

<div class="text-left">
    <b>User Name:</b> @HttpContextAccessor?.HttpContext?.Request.Cookies["UserName"]
    <br />
    <b>User Id:</b> @HttpContextAccessor?.HttpContext?.Request.Cookies["UserId"]
</div>
```

Cookies Trong ASP.NET Core MVC

7.1.2. Làm việc với Cookie trong ASP.net core MVC



Lỗi Service: Dependency Injection



Kết quả: Đọc dữ liệu Cookie

Session Trong ASP.NET Core MVC

7.2. Phiên làm việc (Session)

- Phiên làm việc (Session) trong ASP.NET core MVC là một trong nhiều cách để quản lý trạng thái trong ứng dụng Web. Giao thức HTTP, theo mặc định, là giao thức không có trạng thái. Nhưng đôi khi, điều cần thiết là phải lưu trữ và truyền dữ liệu người dùng trong khi người dùng đang duyệt các trang web. Dữ liệu này có thể giúp ghi nhớ các thông tin gần đây của họ tại các trang trong ứng dụng. Ví dụ: trên trang web thương mại điện tử, dữ liệu này có thể giúp lưu trữ sản phẩm trong giỏ hàng.

- Trong ASP.NET Core MVC, các phiên làm việc là một cơ chế để lưu trữ và quản lý dữ liệu dành riêng cho người dùng ở phía máy chủ qua nhiều HTTPRequest trong một phiên truy cập. Session cung cấp một cách để duy trì trạng thái cho người dùng trong suốt quá trình tương tác của họ với ứng dụng web, cho phép ta lưu trữ và truy xuất dữ liệu được liên kết với phiên của từng người dùng cụ thể.

- Phiên đặc biệt hữu ích khi ta cần lưu trữ dữ liệu có sẵn trên các trang hoặc yêu cầu khác nhau cho một người dùng cụ thể mà không cần dựa vào cookie hoặc thông số truy vấn. Không giống như cookie, được lưu trữ ở phía máy khách, dữ liệu phiên được lưu trữ ở phía máy chủ, tăng cường bảo mật và giảm nguy cơ lộ thông tin nhạy cảm.

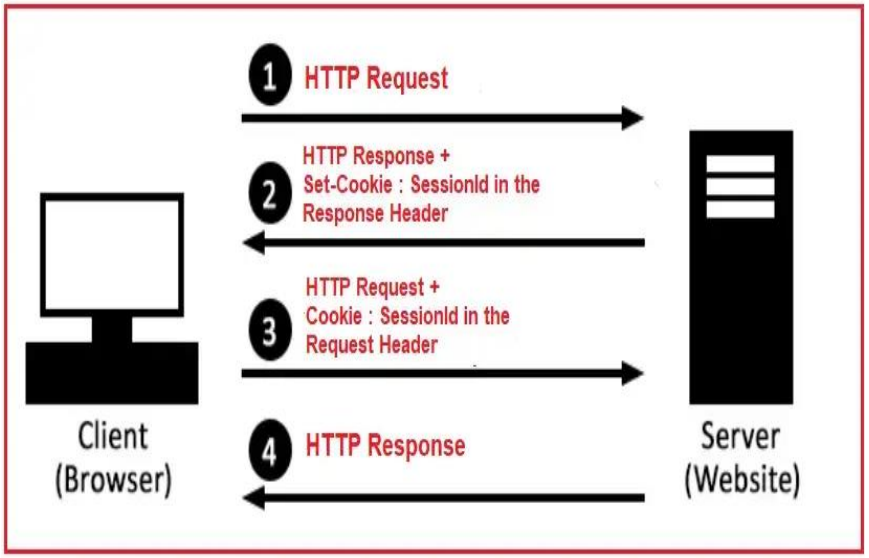
Session Trong ASP.NET Core MVC

7.2.1 Mô hình hoạt động

Mô hình hoạt động của Session:

- Đầu tiên, Client (Web Browser) gửi HTTPRequest đến Server.
- Khi Server nhận yêu cầu, nó sẽ xử lý và trả lại HttpResponse, một SessionId được tạo và gửi trả qua headers bằng cookie. Đồng thời, **SessionId cũng sẽ lưu trữ trên bộ đệm của Server** dưới dạng khóa với các giá trị bắt buộc.
- Từ những request tiếp theo trở đi (nếu session chưa hết hạn), cCient sẽ gửi HTTPRequest cùng với cookie chứa SessionId) trong request header cho Server
- Khi Server nhận Request, nó sẽ kiểm tra cookie, lấy SessionId từ Cookie, lấy dữ liệu từ bộ đệm của Server dựa trên SessionId và xử lý yêu cầu theo đó.

Chú ý: Mỗi trình duyệt có một SessionId duy nhất bị xóa vào cuối phiên và không được chia sẻ với các trình duyệt khác. Thời gian chờ phiên mặc định là 20 phút nhưng có thể được cấu hình lại



Hình 3.1. Mô hình hoạt động của Cookie

Session Trong ASP.NET Core MVC

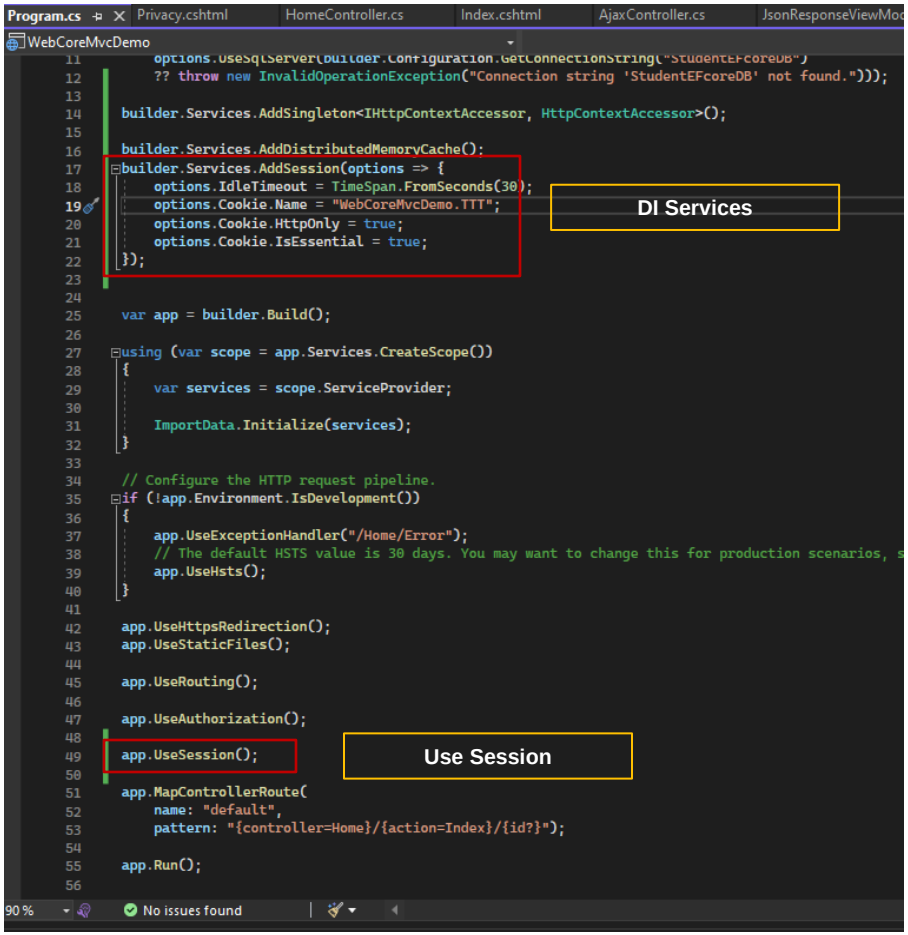
7.2.2. Làm việc với Session trong ASP.net core MVC

Package làm việc với Session:

- @using Microsoft.AspNetCore.Session;
- @IDistributedCache

Đăng ký dịch vụ DI trong **program.cs**

```
builder.Services.AddDistributedMemoryCache();

builder.Services.AddSession(options => {
    options.Properties = value;
});

app.UseSession();
```



Session Trong ASP.NET Core MVC

7.2.2. Làm việc với Session trong ASP.net core MVC

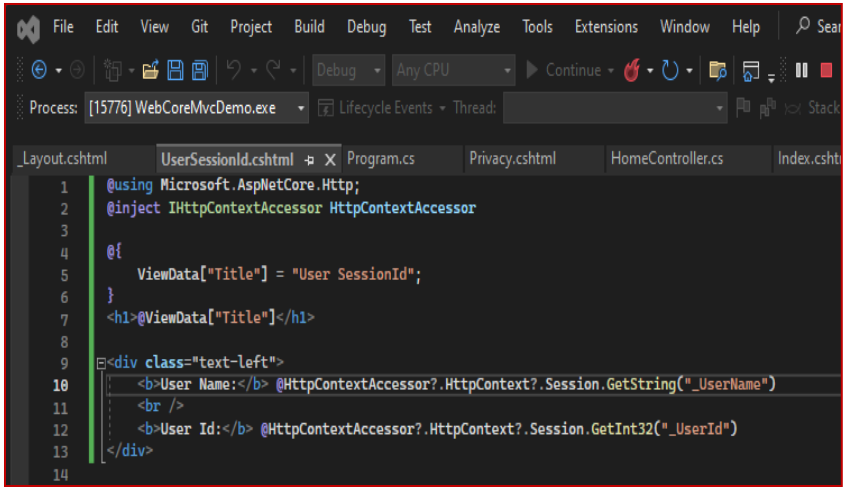
Đọc Session từ View:

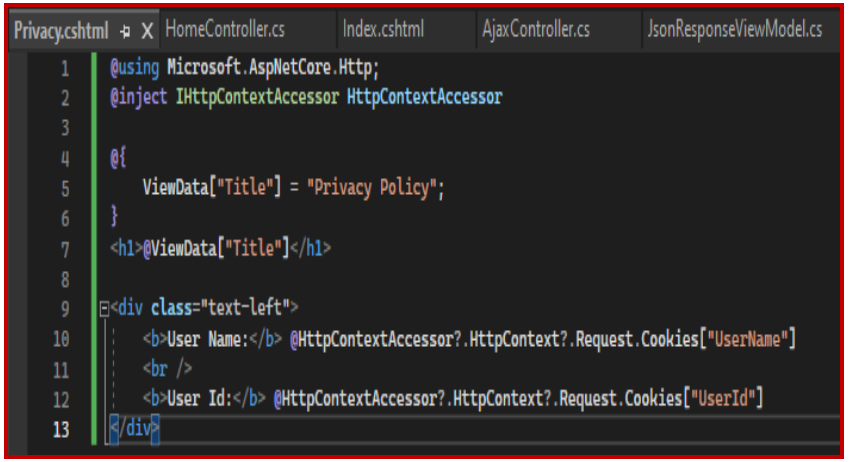
- @using Microsoft.AspNetCore.Http;
- @inject IHttpContextAccessor HttpContextAccessor

Gọi HttpContextAccessor để truy cập các đối tượng HttpContext và Request, Session.

@HttpContextAccessor?.HttpContext?
.Session.GetString[]
@HttpContextAccessor?.HttpContext?
.Session.GetInt32[]

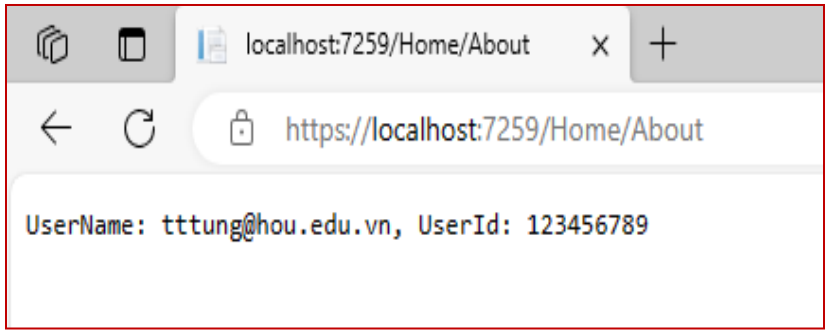
@HttpContextAccessor?.HttpContext?
.Session.SetString[]
@HttpContextAccessor?.HttpContext?
.Session.SetInt32[]



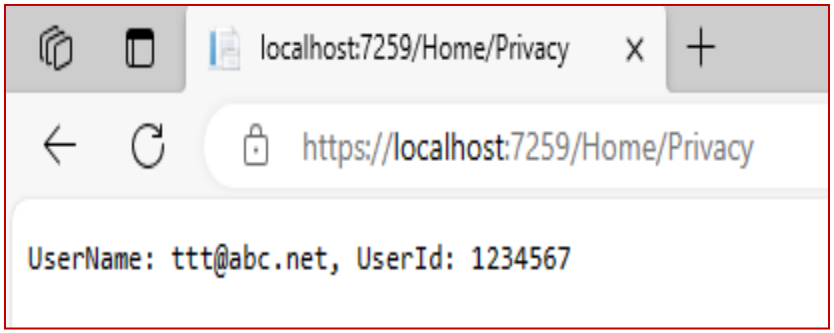


Session Trong ASP.NET Core MVC

7.2.2. Làm việc với Session trong ASP.net core MVC



Thông tin Session lưu trên Server



Thông tin Cookie lưu tại Browser

Xác thực và Phân quyền

7.3. Xác thực và Phân quyền

- Khi phát triển bất kỳ ứng dụng web nào, điều quan trọng nhất ta cần quan tâm là tính bảo mật của nó. Điều đó có nghĩa là ta cần đảm bảo rằng chỉ những **người dùng được xác thực và phân quyền** mới có thể truy cập trang web.

- Xác thực **(Authentication)** là một quá trình đảm bảo và xác nhận danh tính của người dùng. Nói cách khác, chúng ta có thể nói rằng đó là một quá trình để xác thực ai đó dựa trên một số nguồn dữ liệu đã được cấp phát và xác nhận.

- Phân quyền **(Authorization)** là một cơ chế bảo mật được sử dụng để xác định xem người dùng có quyền truy cập vào một tài nguyên cụ thể hay không. Điểm quan trọng nhất cần phải nhớ là xác thực xảy ra trước và dựa vào các thông tin xác thực ta có thể phân quyền truy cập tài nguyên, chức năng trên hệ thống.

Xác thực và Phân quyền

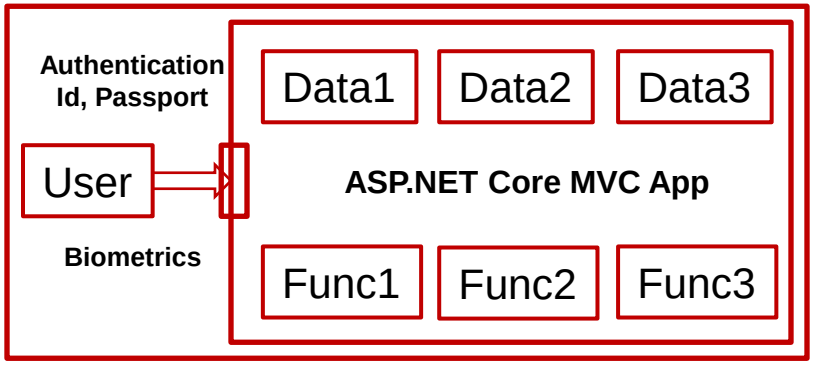
7.3.1 Mô hình xác thực phân quyền

Phân loại Authentication trong MVC:

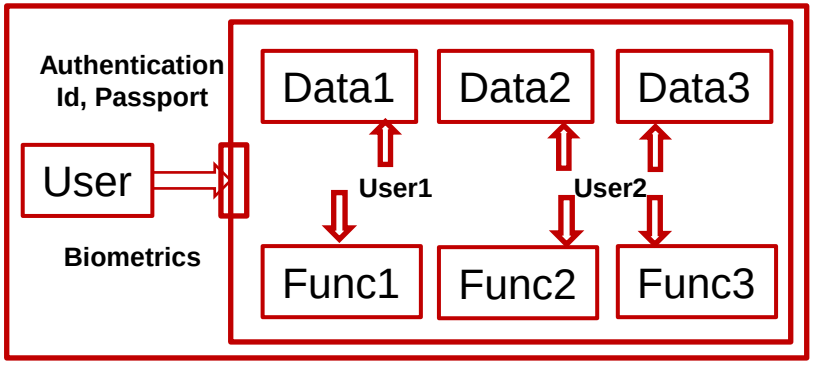
- **Form Authentication:** Trong loại xác thực này, người dùng cần cung cấp thông tin đăng nhập của mình thông qua Form
- **Windows Authentication:** được sử dụng cùng với xác thực IIS. Xác thực được thực hiện bởi IIS theo một trong ba cách như *basic*, *digest* hoặc *Integrated Windows Authentication*. ASP.NET sử dụng danh tính được xác thực để cho phép truy cập.
- **Passport Authentication:** Đây là một dịch vụ xác thực tập trung (dịch vụ trả phí) do Microsoft cung cấp, cung cấp một dịch vụ đăng nhập và hồ sơ cốt lõi cho các trang web thành viên.
- **None:** Không cung cấp xác thực. Đây là chế độ mặc định

ASP.NET MVC sử dụng 2 cách thiết lập

- **Form Authentication**
- **ASP.NET Identity**



Mô hình xác thực định danh



Mô hình phân quyền sau xác thực

ASP.NET Core MVC Identity

7.3.2 ASP.NET Identity

ASP.NET Identity là một hệ thống xác thực phân quyền hiện đại cho phép thực hiện tất cả các hoạt động kiểm soát người dùng bắt buộc trong ứng dụng web của mình, chẳng hạn như Đăng ký, Xác thực và Phân quyền.

Identity được thiết kế để thay thế các hệ thống kiểm soát người dùng ASP.NET trước đây. Nó có thể được sử dụng với tất cả các khung ASP.NET: ASP.NET MVC, API Web, v.v

ASP.NET Identity cung cấp đa dạng các chức năng quản trị tài khoản:

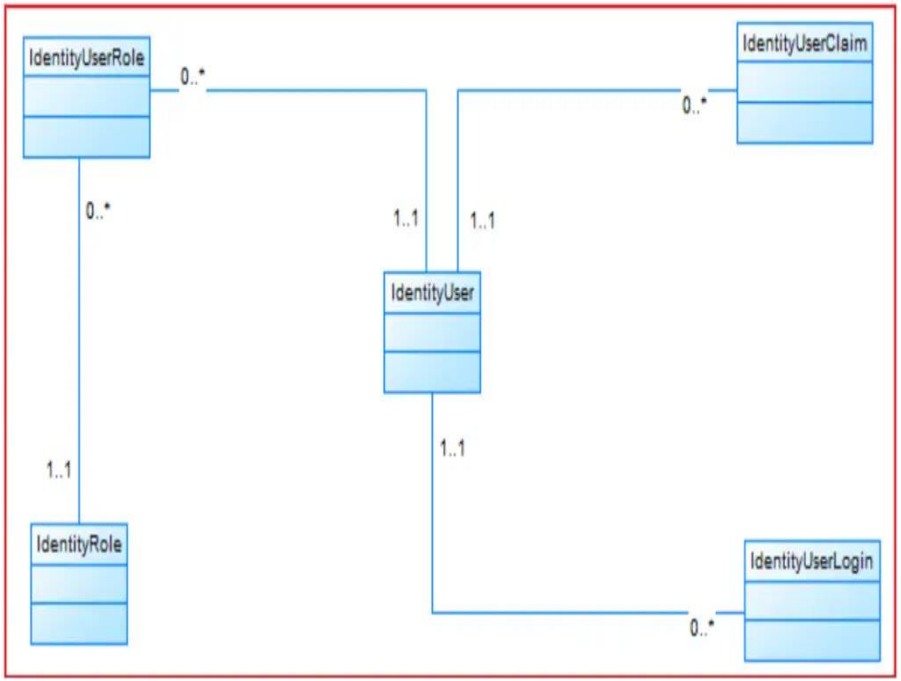
- Log in, Log off.
- Add, Update, and Delete Users.
- Authentication với API (Facebook, Google, Twitter, Microsoft).
- Locking and Unlocking a User Account.
- Hạn chế quyền truy cập vào các phần của trang web dựa trên các vai trò nhất định.
- Locking user account sau một số lần đăng nhập.
- Configure Password Policy.
- Password Reset.
- Two Factor Authentication.
- Account Confirmation Email.
- Quản lý theo vai trò Roles và Claims.

ASP.NET Core MVC Identity

7.3.2 ASP.NET Identity

Định danh người dùng: (**IdentityUser**)

- **Email**: Địa chỉ email
- **EmailConfirmed**: Xác nhận email.
- **PasswordHash**: Hash password (SHA256).
- **SecurityStamp**: Giá trị ngẫu nhiên khi thông tin thay đổi (password changed, login removed).
- **PhoneNumber**: Số điện thoại.
- **PhoneNumberConfirmed**: Xác nhận số điện thoại.
- **TwoFactorEnabled**: Xác thực 2 lớp.
- **LockoutEndDateUtc**: DateTime theo UTC.
- **LockoutEnabled**: Có thể khóa user?
- **AccessFailedCount**: Số lần đăng nhập sai.
- **Claims**: Khiếu nại lấy lại tài khoản.
- **Logins**: Tài khoản login của người dùng.
- **Id**: User ID (Primary Key).
- **UserName**: username



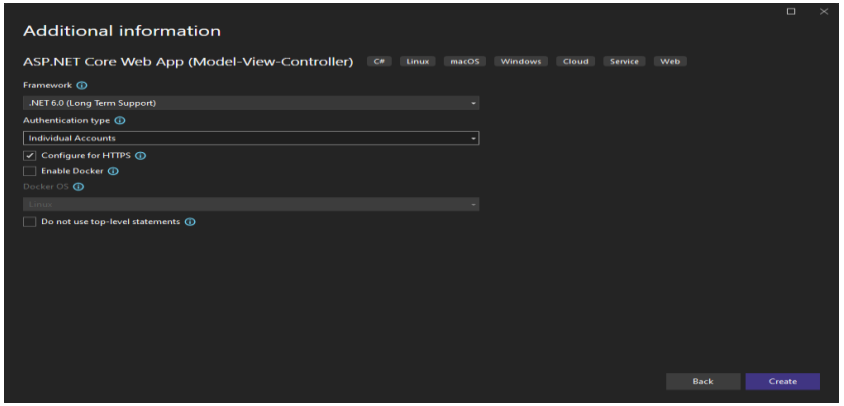
Mô hình ASP.NET Identity

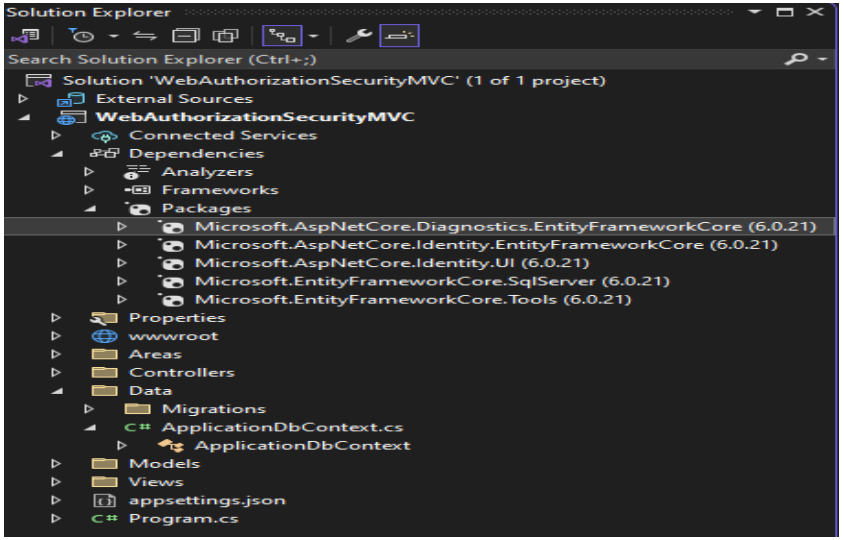
ASP.NET Core MVC Identity

7.3.2 ASP.NET Identity

Thiết lập CSDL Identity

- Start ASP.NET core MVC Project
- Chọn Authentication Type: **Individual Accounts**
- Cấu trúc ứng dụng Identity: Packages
  - **Identity.EntityFramworkCore**
  - **Identity.UI**





Authentication type: Individual Accounts

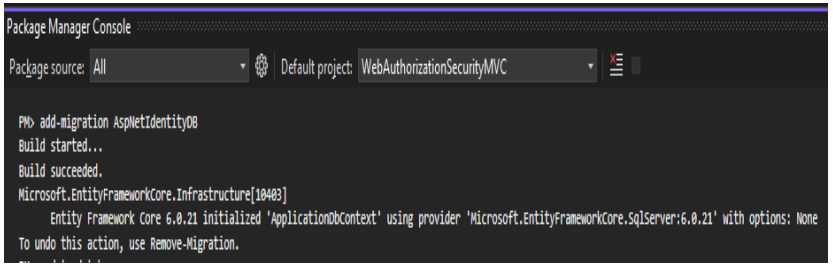
ASP.NET Core MVC Identity

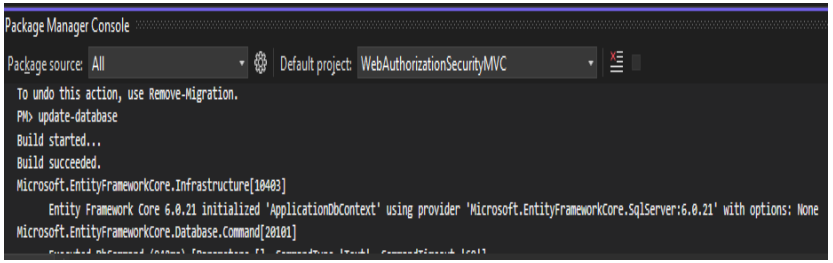
7.3.2 Xác thực định danh

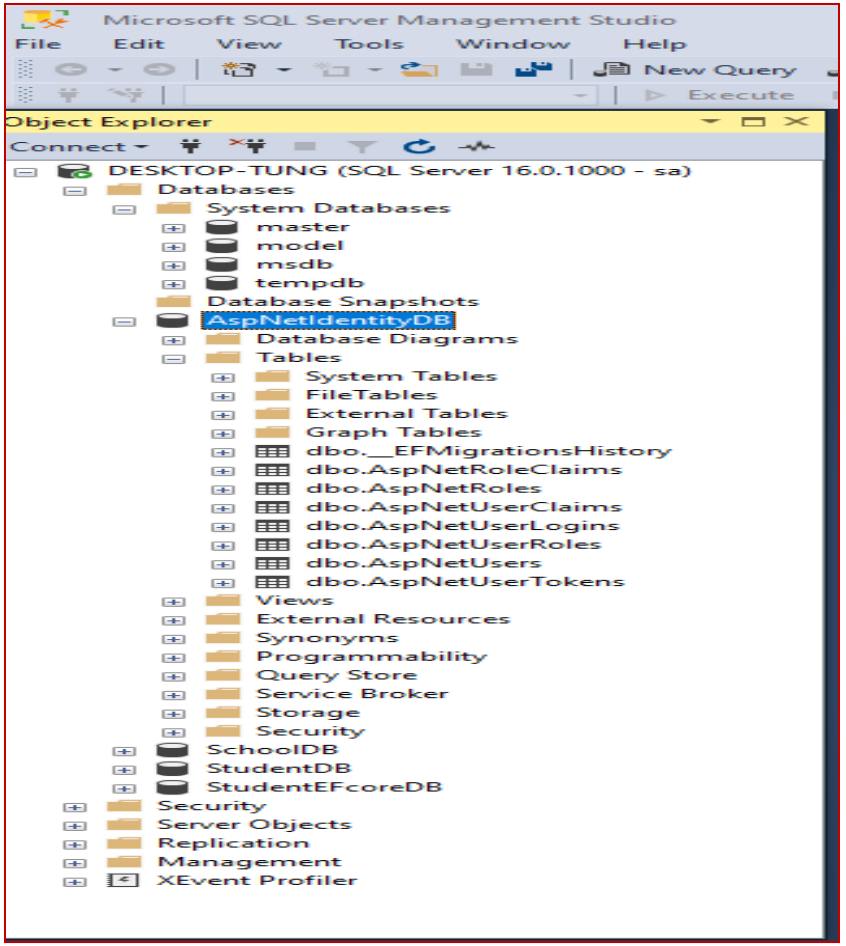
Thiết lập CSDL Identity

Migrate và cập nhật CSDL (PM)

- Migration: **add-migration DBName**
- Update: **update-database**



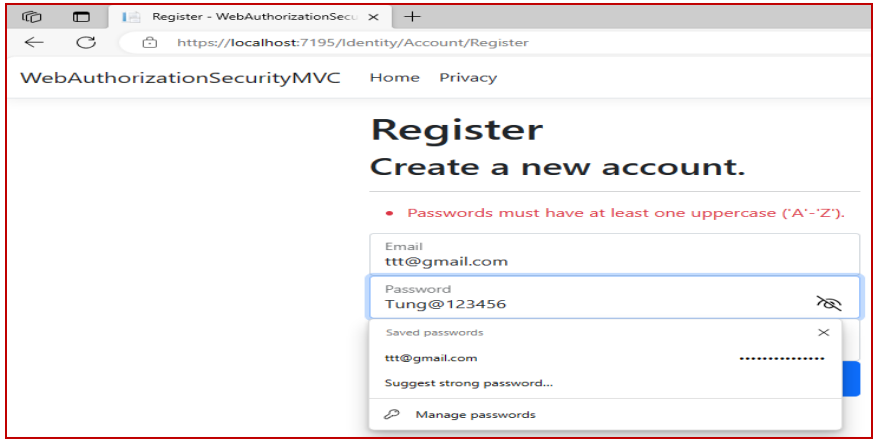


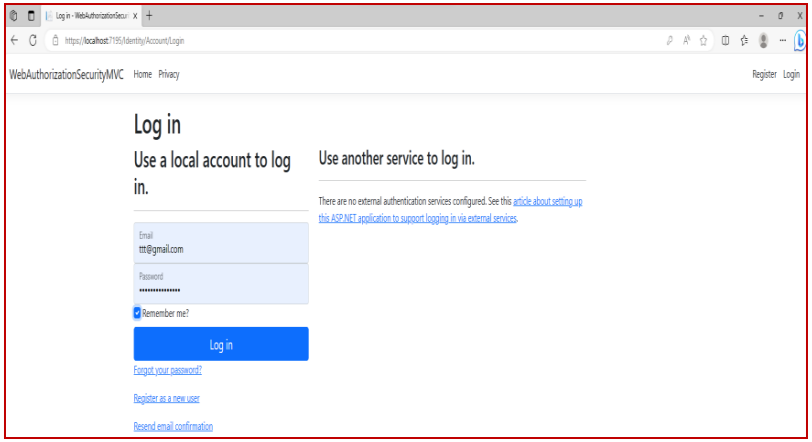


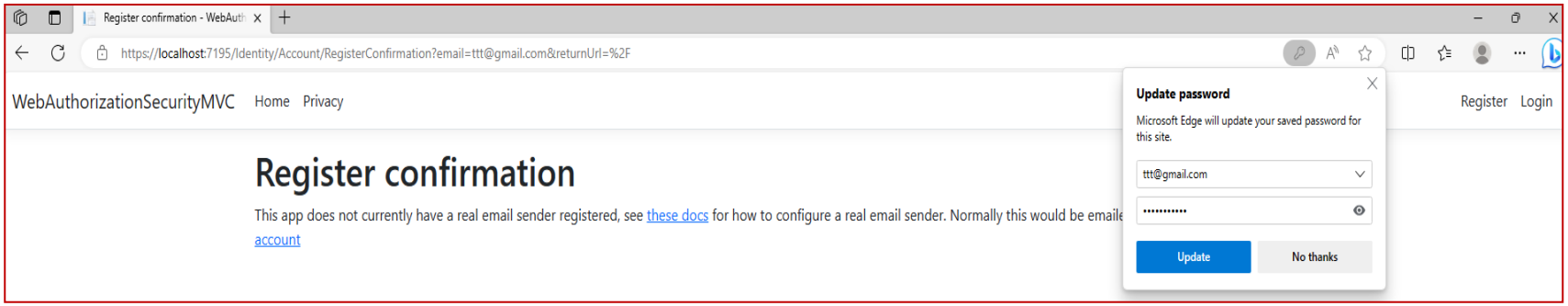
ASP.NET Core MVC Identity

7.3.2 Xác thực định danh

Identity UI:







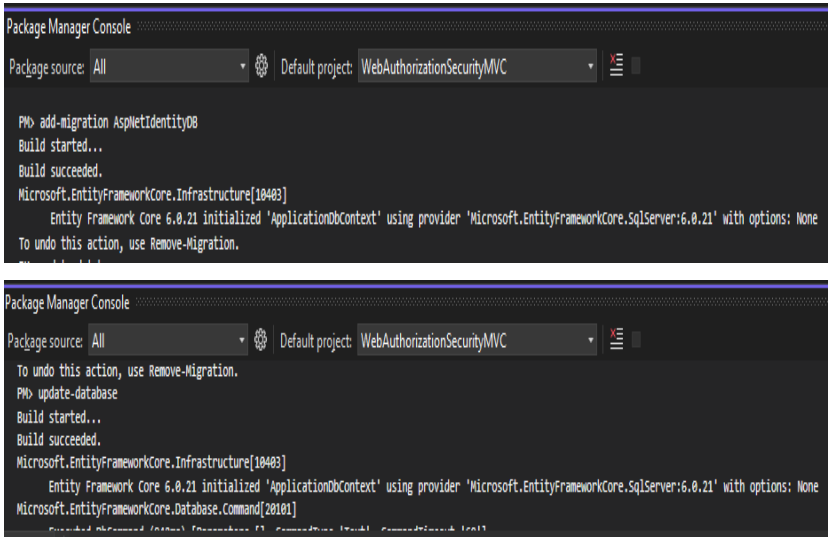
ASP.NET Core MVC Identity

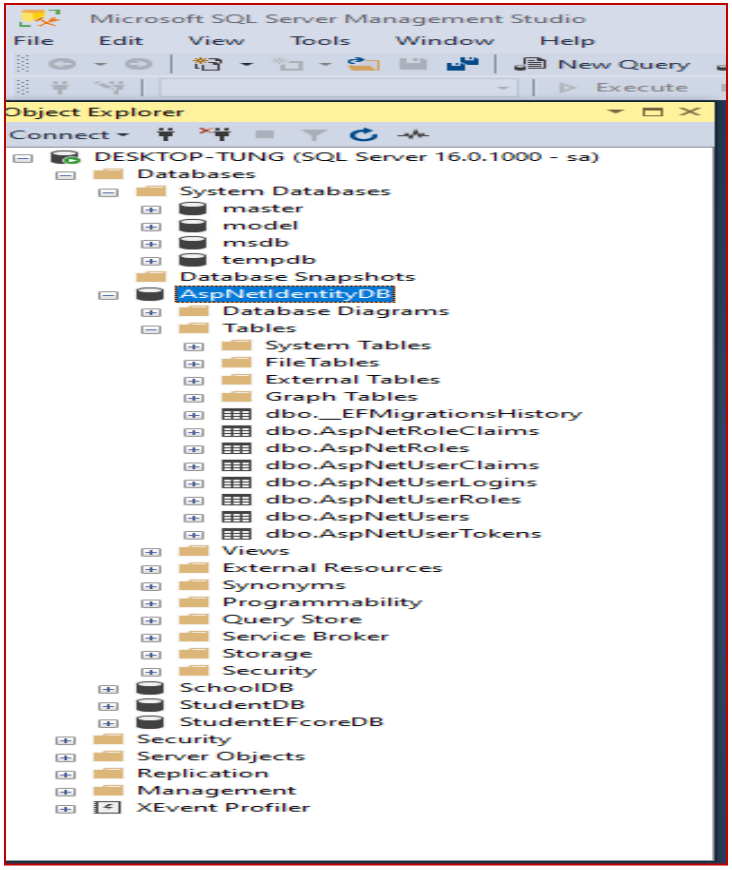
7.3.2 Xác thực định danh

Thiết lập CSDL Identity

Migrate và cập nhật CSDL (PM)

- Migration: **add-migration DBName**
- Update: **update-database**





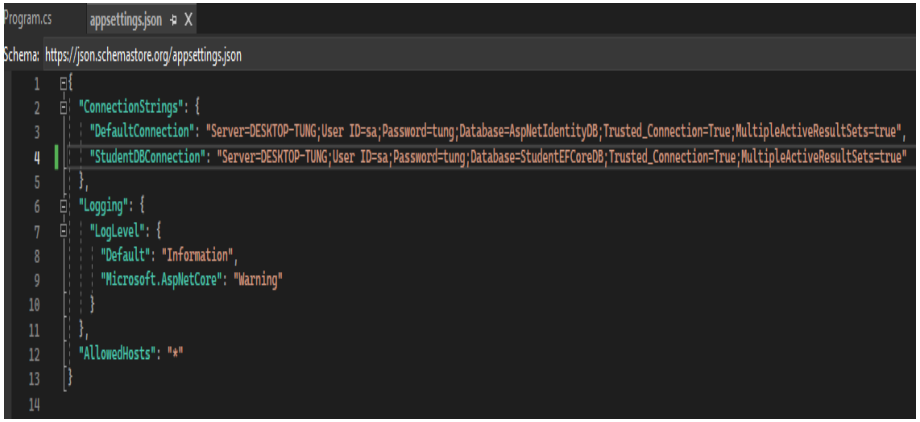
Migration Database

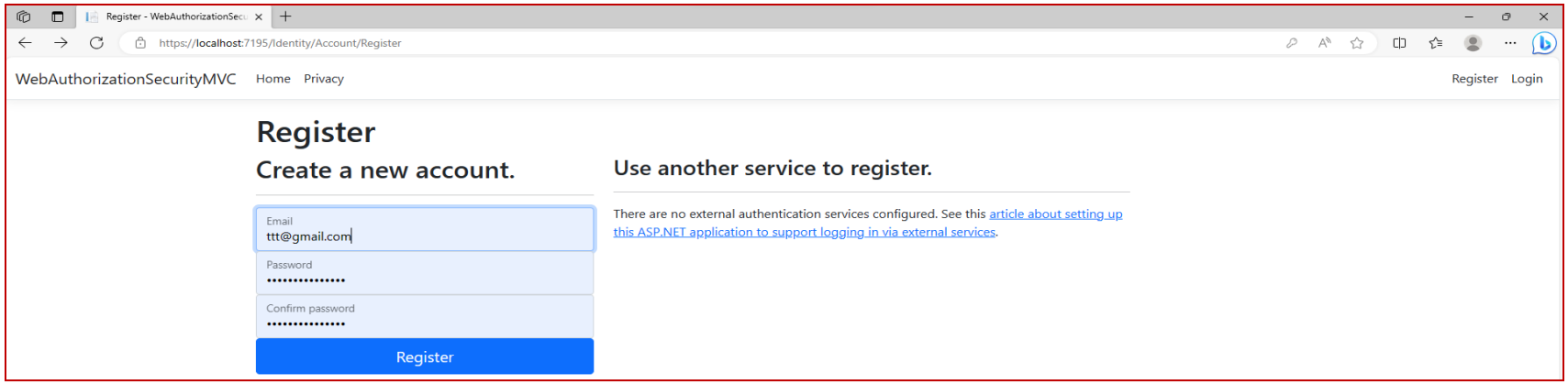
ASP.NET Core MVC Identity

7.3.3 Tích hợp Identity

Tích hợp CSDL Identity và CSDL Ứng dụng

- Kết nối DBContext với CSDL ứng dụng
- Start Run (Debug) Project
- Register New Account
  - Chọn Migration
  - Refresh



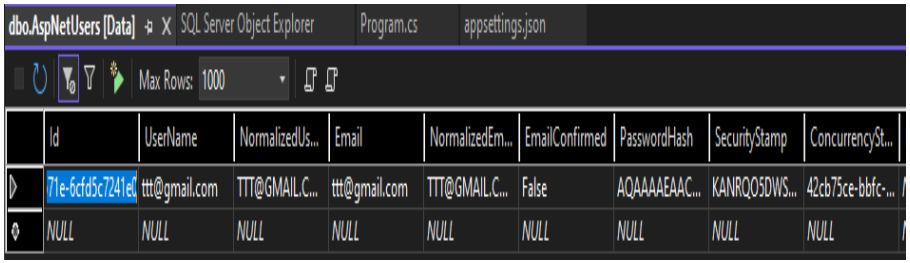


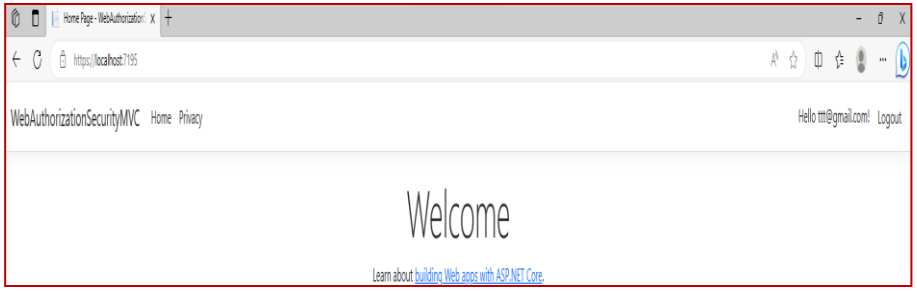
ASP.NET Core MVC Identity

7.3.2 Xác thực định danh

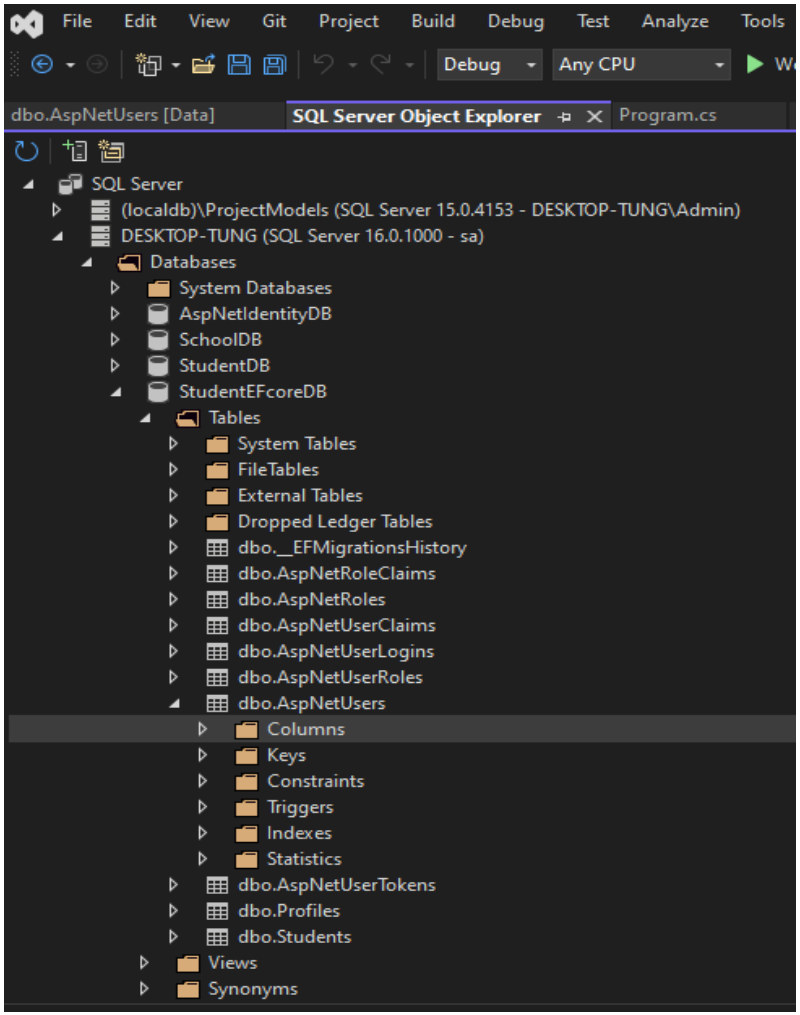
Tích hợp CSDL Identity và CSDL Ứng dụng

CSDL của ứng dụng sẽ tự động tích hợp **Identity**





Asp.Net App Identity Database



An ninh trong ASP.NET core MVC

7.4. An ninh trong Asp.Net core MVC

- Trong các ứng dụng web, bảo mật là một mối quan tâm lớn. Một số lưu ý về an ninh cần xem xét khi phát triển ứng dụng. ASP.NET Core MVC là một nền tảng khá phổ biến được sử dụng để phát triển các ứng dụng web.

- Tuy nhiên, các ứng dụng web trên nền tảng này đã được chứng minh là có thể bị tấn công từ các nguồn khác nhau. Vì vậy, chúng ta cần nhận biết các mối nguy cơ và xây dựng biện pháp bảo vệ dữ liệu của ứng dụng

- Phiên bản .Net core 6.0 (long-term support) và 7.0 đã được tăng cường nhiều tính năng an ninh bảo mật cho ứng dụng Web. Tuy nhiên, một số tính năng được tích hợp mặc định, còn lại chúng cần được tích hợp trong ứng dụng để tăng cường an ninh hệ thống

An ninh trong ASP.NET core MVC

7.4. An ninh trong Asp.Net core MVC

Phân loại tấn công và phòng chống

- **Cross-Site Scripting, CS Request Forgery (XSS/CSRF):** Chèn mã lệnh độc hại thông qua QueryString hoặc Form Input với mục đích đánh cắp thông tin bí mật như tài khoản hoặc thông tin xác thực khác, cookie và session. Để phòng chống cần kích hoạt URLEncode và DataAnnotation, Tag-Helper Anti Forgery.

- **SQL Injection:** là loại tấn công phổ CSDL biến nhất bằng cácch tiêm mã vào chuỗi SQLQuery để truy cập cơ sở dữ liệu và người dùng trái phép có thể thao tác với CSDL đó. Giải pháp phòng chống thường dùng là sử dụng Query với tham số, Thủ tục lưu, các loại CSDL ORM như Entity Framwork.

- **Error handling:** Mã xử thông báo lỗi trong ứng dụng web có thể tiết lộ nhiều thông tin nhạy cảm như thông tin cấu hình cơ sở dữ liệu, tên bảng, quy trình được lưu trữ, cấu trúc dữ liệu và cấu trúc mã hóa lập trình cho người dùng. Giải pháp thường là thiết lập cấu hình ReRoute và Redirect về trang thông báo lỗi tập trung khi có lỗi xẩy ra.

An ninh trong ASP.NET core MVC

7.4. An ninh trong Asp.Net core MVC

Phân loại tấn công và phòng chống

- **Enforce SSL (Secure Sockets Layer) and HSTS: Bắt buộc dùng** Lớp bảo mật để thiết lập kết nối an toàn hoặc được mã hóa giữa Client với Server. Với SSL, các Request/Response được truyền giữa trình duyệt và Web Server sẽ được mã hóa để duy trì tính toàn vẹn của dữ liệu. HSTS là web security policy bảo vệ ứng dụng web của khỏi các cuộc tấn công giao thức mức thấp và chiếm quyền điều khiển cookie. Nó buộc máy chủ web giao tiếp qua kết nối HTTPS. Nó luôn từ chối các kết nối HTTP không an toàn.

- **Tấn công XXE (XML External Entity):** Nếu ứng dụng phân tích cú pháp tệp XML được tải lên bởi người dùng cuối, nguy cơ có thể bị tấn công XXE (XML External Entity). Một dạng tấn công từ chối dịch vụ dạng XXE gọi là “billion laughs attact. Giải pháp thường dùng thiết lập chuẩn DTD (Data Type Definition).

- **Authentication and Session management:** Xác thực và quản lý phiên không đúng cách. Hầu hết các ứng dụng web đều có mô-đun xác thực và ta cần chú ý về những dữ liệu sử dụng trong các modul này. Những lỗi có thể mắc như không sử dụng SSL, lưu cookie dạng plaintext, không xóa cookie xác thực sau khi logout thành công.

KẾT THÚC

Q & A

CHÚC CÁC BẠN ĐẠT KẾT QUẢ TỐT