

### 1. ****Trình bày khái niệm Authentication trong ASP.NET Core. Tại sao cần xác thực người dùng?****

**Authentication** là quá trình xác minh danh tính người dùng (user identity). Trong ASP.NET Core, authentication giúp đảm bảo rằng người dùng là ai mà họ khai báo.  
👉 **Lý do cần thiết:** để bảo vệ tài nguyên ứng dụng, đảm bảo chỉ người dùng hợp lệ mới truy cập được.

### 2. ****Mô tả quá trình xử lý xác thực khi người dùng đăng nhập vào một ứng dụng ASP.NET Core sử dụng Cookie Authentication.****

Người dùng nhập thông tin đăng nhập.

Ứng dụng xác thực thông tin với cơ sở dữ liệu.

Nếu đúng, hệ thống tạo **Authentication Ticket** và lưu vào cookie.

Cookie được gửi cùng với các request tiếp theo để xác minh người dùng.

### 3. ****So sánh Session-based Authentication và Token-based Authentication:****

| **Tiêu chí** | **Session-based** | **Token-based** |
| --- | --- | --- |
| Lưu trữ trạng thái | Server lưu session | Stateless (không lưu trạng thái) |
| Bảo mật | Có thể bị tấn công CSRF | Token (JWT) dễ truyền qua HTTPS, nhưng phải bảo mật tốt |
| Mở rộng hệ thống | Hạn chế khi scale | Dễ dàng mở rộng (client lưu token) |
| Tốc độ | Tốt trong nội bộ | Linh hoạt hơn qua các domain/API khác nhau |

### 4. ****Vai trò của SignInManager trong ASP.NET Core Identity:****

SignInManager<TUser>:

Kiểm tra thông tin đăng nhập.

Đăng nhập/đăng xuất người dùng.

Xử lý xác thực 2 bước, khoá tài khoản, ghi nhớ đăng nhập...

### 5. ****Giải thích sự khác biệt giữa Authentication và Authorization:****

**Authentication**: Xác minh bạn là ai. (ví dụ: đăng nhập).

**Authorization**: Bạn được phép làm gì. (ví dụ: admin mới có thể xóa bài viết). 📌 Ví dụ: Người dùng đăng nhập thành công (authentication), nhưng chỉ admin mới có quyền truy cập trang quản trị (authorization).

### 6. ****Ý nghĩa và vai trò của [AllowAnonymous] và [Authorize] trong ASP.NET Core MVC:****

[Authorize]: yêu cầu người dùng đã xác thực mới truy cập được.

[AllowAnonymous]: cho phép truy cập không cần xác thực, ngay cả khi controller/class đã có [Authorize].

### 7. ****Khi nào nên sử dụng [Authorize(Roles = "Admin")]?****

Khi bạn muốn giới hạn quyền truy cập chỉ cho người có role cụ thể — ví dụ "Admin":

[Authorize(Roles = "Admin")]public IActionResult ManageUsers() { ... }

Dùng khi bảo vệ chức năng quản trị hoặc nhạy cảm.

### 8. ****Trình bày luồng xử lý khi xác thực người dùng thất bại:****

Nếu xác thực thất bại:

Trả lại thông báo lỗi (sai username/password).

Hoặc redirect về trang đăng nhập.

Có thể cấu hình trong Startup.cs:

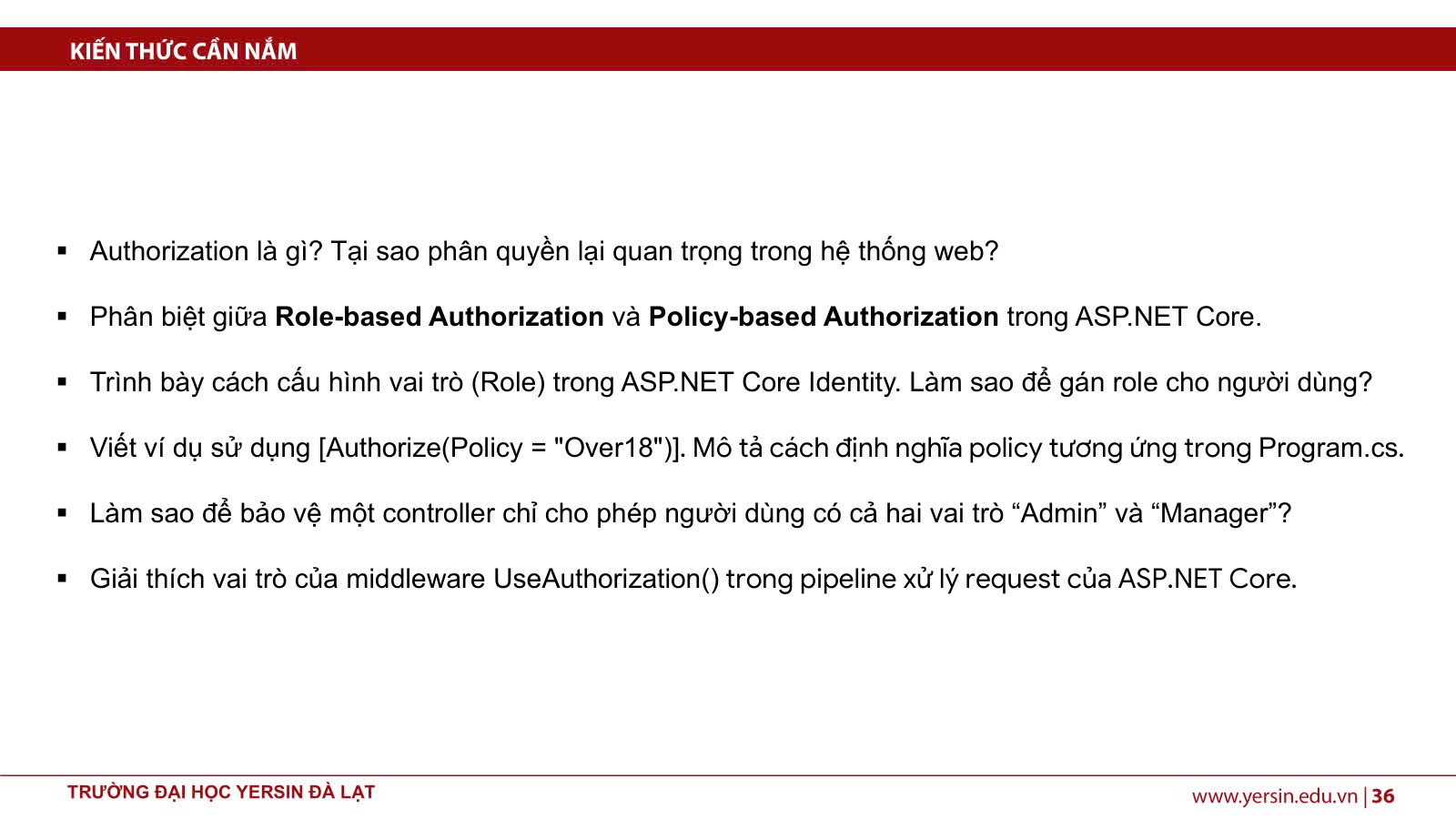
services.ConfigureApplicationCookie(options =>

{

options.LoginPath = "/Account/Login";

options.AccessDeniedPath = "/Account/AccessDenied";

});



### 1. ****Authorization là gì? Tại sao phân quyền lại quan trọng trong hệ thống web?****

**Authorization** là quá trình kiểm tra xem người dùng đã được cấp quyền truy cập vào một tài nguyên hoặc chức năng cụ thể hay chưa.

**Quan trọng vì:**

Bảo vệ dữ liệu nhạy cảm.

Giới hạn quyền truy cập tùy vai trò người dùng.

Tránh lạm quyền trong hệ thống.

### 2. ****Phân biệt Role-based Authorization và Policy-based Authorization trong ASP.NET Core:****

| **Tiêu chí** | **Role-based** | **Policy-based** |
| --- | --- | --- |
| Cách hoạt động | Dựa vào tên vai trò (Role) | Dựa vào quy tắc (policy) do dev định nghĩa |
| Linh hoạt | Ít linh hoạt | Rất linh hoạt (có thể kiểm tra tuổi, claims...) |
| Cách dùng | [Authorize(Roles = "Admin")] | [Authorize(Policy = "Over18")] |

### 3. ****Cấu hình vai trò (Role) trong ASP.NET Core Identity & gán role cho người dùng:****

**Tạo và gán Role:**

await roleManager.CreateAsync(new IdentityRole("Admin"));await userManager.AddToRoleAsync(user, "Admin");

**Cấu hình dịch vụ:**

services.AddIdentity<IdentityUser, IdentityRole>()

.AddEntityFrameworkStores<ApplicationDbContext>();

### 4. ****Ví dụ dùng [Authorize(Policy = "Over18")] và định nghĩa trong Program.cs:****

**Program.cs:**

builder.Services.AddAuthorization(options =>

{

options.AddPolicy("Over18", policy =>

policy.RequireClaim("Age", "18", "19", "20", "21", "22")); // hoặc dùng custom requirement

});

**Controller:**

[Authorize(Policy = "Over18")]public IActionResult RestrictedArea() { ... }

### 5. ****Bảo vệ controller cho người có cả hai vai trò “Admin” và “Manager”:****

ASP.NET Core không hỗ trợ mặc định kiểm tra "có cả hai vai trò", nhưng bạn có thể viết custom policy:

builder.Services.AddAuthorization(options =>

{

options.AddPolicy("AdminAndManager", policy =>

policy.RequireAssertion(context =>

context.User.IsInRole("Admin") && context.User.IsInRole("Manager")));

});

Controller:

[Authorize(Policy = "AdminAndManager")]public IActionResult AdminManagerOnly() { ... }

### 6. ****Vai trò của middleware**** UseAuthorization() ****trong pipeline:****

Middleware UseAuthorization():

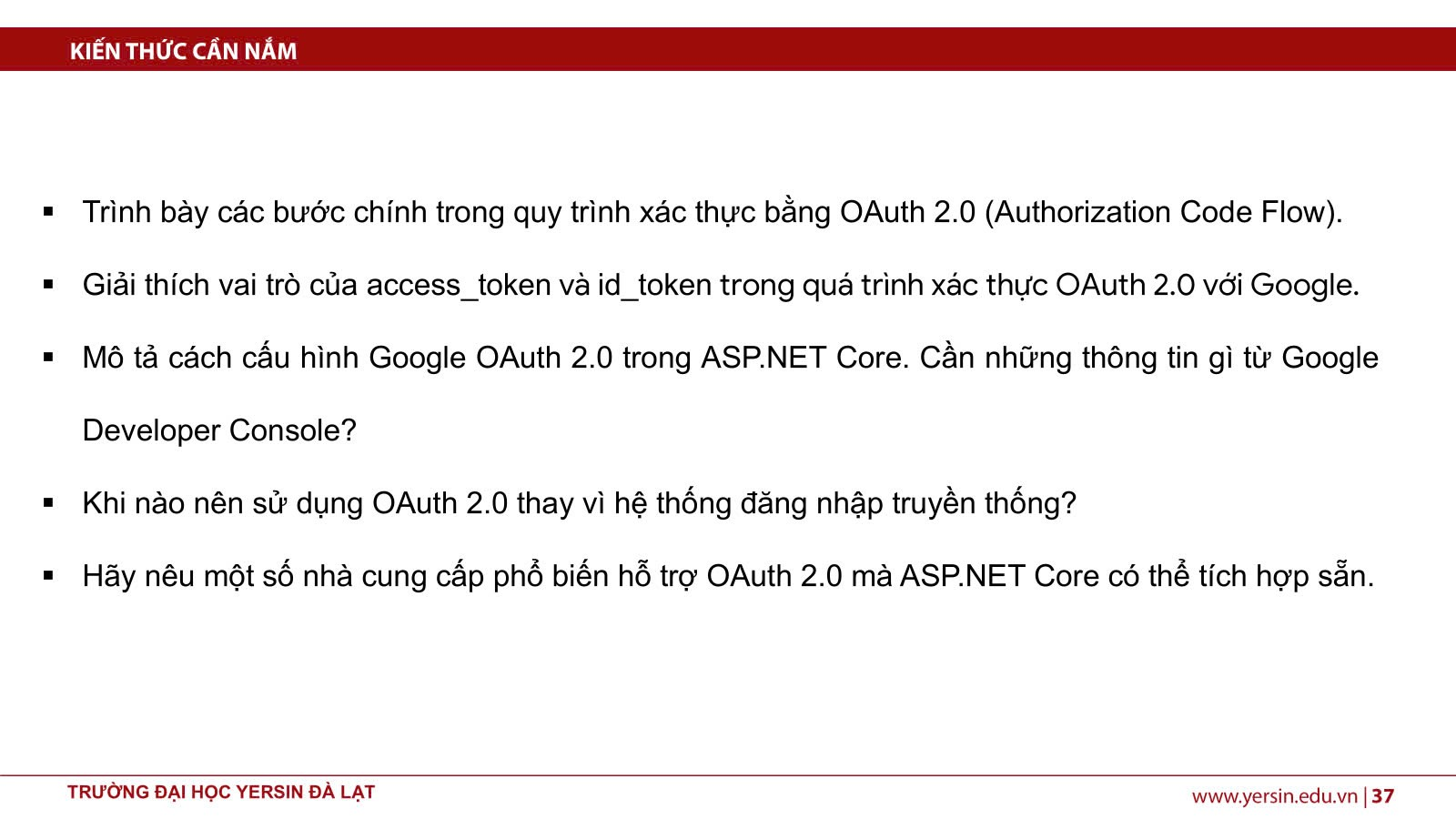
Kiểm tra các attribute như [Authorize] trong controller/action.

Nếu không đủ quyền, trả về 403 Forbidden hoặc redirect tới trang login.

Phải đặt **sau** UseAuthentication() trong Program.cs:

app.UseAuthentication();

app.UseAuthorization();



### 1. ****Các bước chính trong quy trình OAuth 2.0 (Authorization Code Flow):****

1. **User gửi yêu cầu đăng nhập** → được redirect đến trang xác thực của bên thứ ba (Google, Facebook...).
2. **User đăng nhập & cấp quyền** → Bên thứ ba redirect về redirect\_uri với một **authorization code**.
3. **Ứng dụng backend** dùng authorization code để gọi đến token endpoint và nhận về **access\_token** và **id\_token**.
4. **Sử dụng token** để truy cập tài nguyên hoặc xác minh danh tính người dùng.

### 2. ****Vai trò của**** access\_token ****và**** id\_token****:****

access\_token:

Dùng để **gọi API** (ví dụ: API của Google).

Thường không chứa thông tin người dùng

id\_token:

Dùng để **xác minh danh tính người dùng** (OpenID Connect).

Chứa thông tin như tên, email, thời gian đăng nhập...

### 3. ****Cấu hình Google OAuth 2.0 trong ASP.NET Core:****

Cần các thông tin từ **Google Developer Console**:

Client ID

Client Secret

Redirect URI (phải đúng với cấu hình trong app và Google)

**Ví dụ cấu hình:**

services.AddAuthentication()

.AddGoogle(options =>

{

options.ClientId = "your-client-id";

options.ClientSecret = "your-client-secret";

});

### 4. ****Khi nào nên dùng OAuth 2.0 thay vì đăng nhập truyền thống?****

Nên dùng OAuth 2.0 khi:

Cho phép đăng nhập qua **bên thứ ba** (Google, Facebook...).

Không muốn quản lý mật khẩu người dùng.

Ứng dụng mobile/web cần **uỷ quyền truy cập tài nguyên** từ server khác.

### 5. ****Một số nhà cung cấp OAuth 2.0 phổ biến hỗ trợ ASP.NET Core:****

**Google**

**Facebook**

**Microsoft**

**GitHub**

**Twitter**

**LinkedIn**

✅ ASP.NET Core hỗ trợ tích hợp sẵn thông qua Microsoft.AspNetCore.Authentication.\*



### 1. ****ASP.NET Core Identity là gì? Chức năng chính:****

ASP.NET Core Identity là hệ thống quản lý người dùng **tích hợp sẵn** trong ASP.NET Core.  
**Chức năng chính gồm:**

Đăng ký / Đăng nhập người dùng.

Xác thực và phân quyền người dùng.

Quản lý vai trò (Roles).

Hỗ trợ OAuth, OpenID Connect.

Mở rộng dễ dàng (thêm thông tin như tuổi, địa chỉ...).

### 2. ****So sánh giữa**** UserManager****,**** RoleManager ****và**** SignInManager****:****

| **Thành phần** | **Chức năng chính** |
| --- | --- |
| UserManager | Quản lý người dùng: tạo, xóa, tìm kiếm, đổi mật khẩu. |
| RoleManager | Quản lý vai trò: tạo role, gán role, kiểm tra role. |
| SignInManager | Xử lý đăng nhập, đăng xuất, xác minh thông tin đăng nhập. |

### 3. ****Tạo user mới và gán role trong ASP.NET Core Identity:****

var user = new IdentityUser { UserName = "john", Email = "john@example.com" };var result = await \_userManager.CreateAsync(user, "P@ssw0rd");

if (result.Succeeded)

{

await \_userManager.AddToRoleAsync(user, "Admin");

}

### 4. ****Kiểm tra mật khẩu người dùng với**** PasswordSignInAsync****:****

var result = await \_signInManager.PasswordSignInAsync("john", "P@ssw0rd", isPersistent: false, lockoutOnFailure: false);

if (result.Succeeded)

{

// Đăng nhập thành công

}

### 5. ****Lưu thông tin bổ sung cho người dùng:****

**Bước 1: Kế thừa từ IdentityUser để thêm trường:**

public class ApplicationUser : IdentityUser

{

public int Age { get; set; }

public string Address { get; set; }

}

**Bước 2: Cập nhật DbContext:**

public class AppDbContext : IdentityDbContext<ApplicationUser>

{

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

}

**Bước 3: Khi tạo user, gán thêm dữ liệu:**

var user = new ApplicationUser { UserName = "john", Email = "john@example.com", Age = 25, Address = "Đà Lạt" };await \_userManager.CreateAsync(user, "P@ssw0rd");