

BÀI 4: XÂY DỰNG ỨNG DỤNG WEB BÁN HÀNG VỚI ASP.NET CORE MVC (PHẦN 2)

Sau khi học xong bài này, sinh viên có thể nắm được:

- Sử dụng Identity Core để thực hiện các chức năng quản lý người dùng trong ứng dụng
- Kỹ năng sử dụng Area trong ASP.NET Core MVC để phân tách và tổ chức các chức năng quản trị ra khỏi phần còn lại của ứng dụng web. Điều này giúp tăng cường bảo mật và rõ ràng trong cấu trúc ứng dụng.
- Tạo khu vực Admin để chứa các chức năng như quản lý người dùng, cài đặt hệ thống, báo cáo, và các công cụ quản trị khác.

4.1 Bài thực hành

4.1.1 Yêu cầu

Tích hợp Identity Core vào ứng dụng web bán hàng ở bài thực hành LAB 2 để quản lý người dùng.

Thiết lập một Area Admin trong ứng dụng web bán hàng để tách biệt chức năng dành cho người quản trị.

4.2 Hướng dẫn thực hiện

4.2.1 Cấu Hình ASP.NET Core Identity

Cấu hình ASP.NET Core Identity trong `ApplicationDbContext` và `Program.cs`:

- *ApplicationDbContext.cs*

```
using Microsoft.AspNetCore.Identity.EntityFrameworkCore;
using Microsoft.EntityFrameworkCore;
using YourNamespace.Models; // Thay thế bằng namespace thực tế của bạn

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

    public DbSet<Product> Products { get; set; }
    public DbSet<Category> Categories { get; set; }
    // Các DbSet khác nếu cần
}
```

- *Program.cs*

```
using Microsoft.AspNetCore.Identity;
using Microsoft.EntityFrameworkCore;
using WebsiteBanHang.DataAccess;
using WebsiteBanHang.Repositories;

var builder = WebApplication.CreateBuilder(args);

// Add services to the container.
builder.Services.AddControllersWithViews();

builder.Services.AddDbContext<ApplicationDbContext>(options =>
    options.UseSqlServer(builder.Configuration.GetConnectionString("DefaultConnection")));

builder.Services.AddIdentity<IdentityUser, IdentityRole>()
    .AddDefaultTokenProviders()
    .AddDefaultUI()
    .AddEntityFrameworkStores<ApplicationDbContext>();

builder.Services.AddRazorPages();

builder.Services.AddScoped<IProductRepository, EFProductRepository>();
builder.Services.AddScoped<ICategoryRepository, EFCategoryRepository>();

var app = builder.Build();

// Configure the HTTP request pipeline.
if (!app.Environment.IsDevelopment())
{
    app.UseExceptionHandler("/Home/Error");
}
app.UseStaticFiles();

app.UseRouting();
app.UseAuthentication();

app.UseAuthorization();

app.MapRazorPages();
app.MapControllerRoute(
    name: "default",
    pattern: "{controller=Home}/{action=Index}/{id?}");
```

```
app.Run();
```

- *Cập nhật Database*

```
dotnet ef migrations add InitialCreateIdentity --project WebsiteBanHang  
dotnet ef database update --project WebsiteBanHang
```

4.2.2 Scaffolding ASP.NET Core Identity

Sử dụng ASP.NET Core scaffolder để thêm các view đăng nhập và đăng ký:

1. Trong Visual Studio, click chuột phải vào project, chọn **Add -> New Scaffolded Item**.
2. Chọn **Identity** và click **Add**.
3. Chọn các **pages** bạn muốn thêm, ví dụ: **Login** và **Register**.
4. Chọn **Add**.

Add Identity

Select an existing layout page, or specify a new one:
/Areas/Identity/Pages/Account/Manage/_Layout.cshtml
(Leave empty if it is set in a Razor _viewstart file)

☐ Override all files

Choose files to override

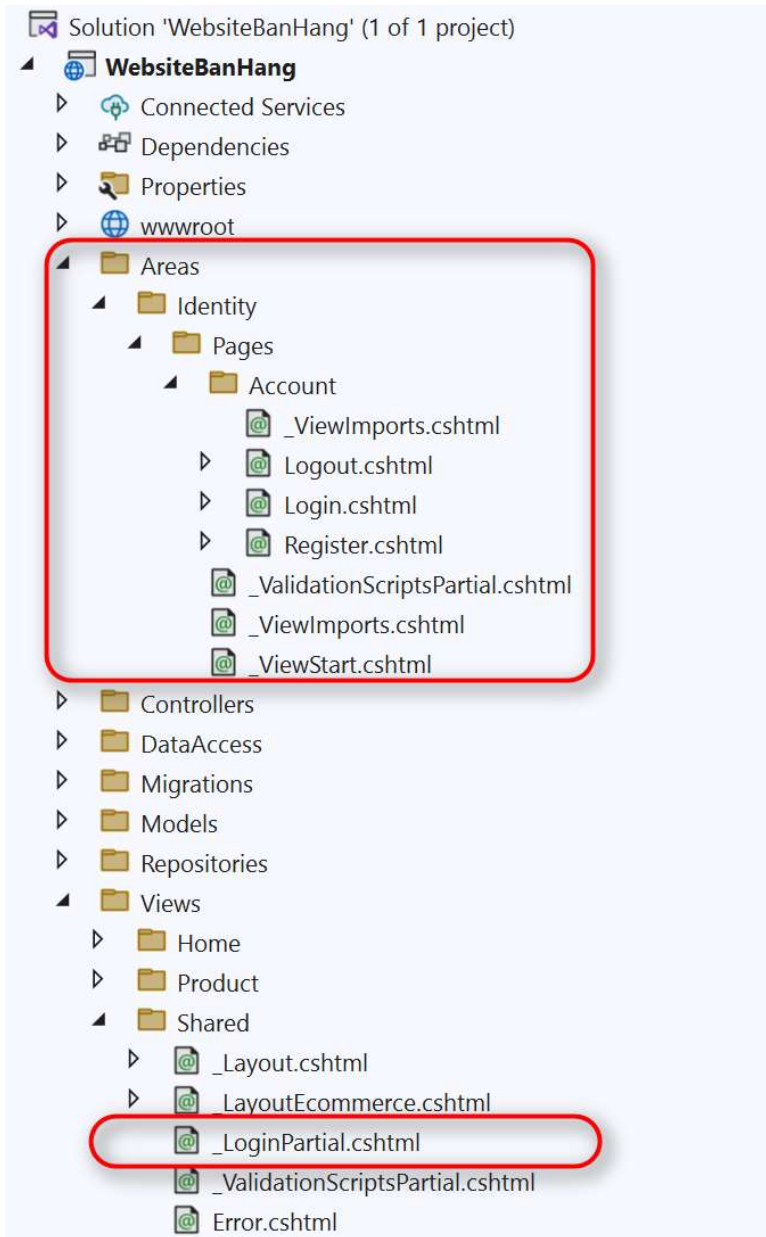
<input type="checkbox"/> Account\StatusMessage	<input type="checkbox"/> Account\AccessDenied	<input type="checkbox"/> Account\ConfirmEmail
<input type="checkbox"/> Account\ConfirmEmailChange	<input type="checkbox"/> Account\ExternalLogin	<input type="checkbox"/> Account\ForgotPassword
<input type="checkbox"/> Account\ForgotPasswordConfirmation	<input type="checkbox"/> Account\Lockout	<input checked="" type="checkbox"/> Account>Login
<input type="checkbox"/> Account>LoginWith2fa	<input type="checkbox"/> Account>LoginWithRecoveryCode	<input checked="" 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 checked="" type="checkbox"/> Account\Register
<input type="checkbox"/> Account\RegisterConfirmation	<input type="checkbox"/> Account\ResendEmailConfirmation	<input type="checkbox"/> Account\ResetPassword
<input type="checkbox"/> Account\ResetPasswordConfirmation		

DbContext class:

Database provider:

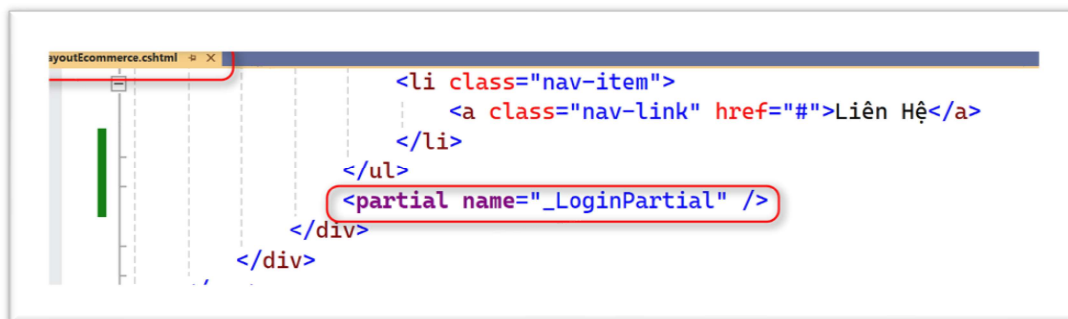
User class:

Add Cancel



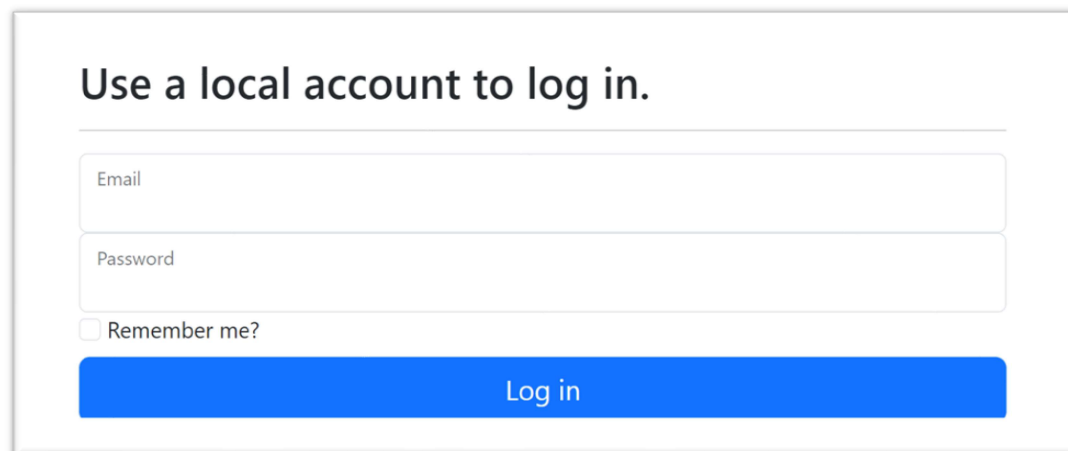
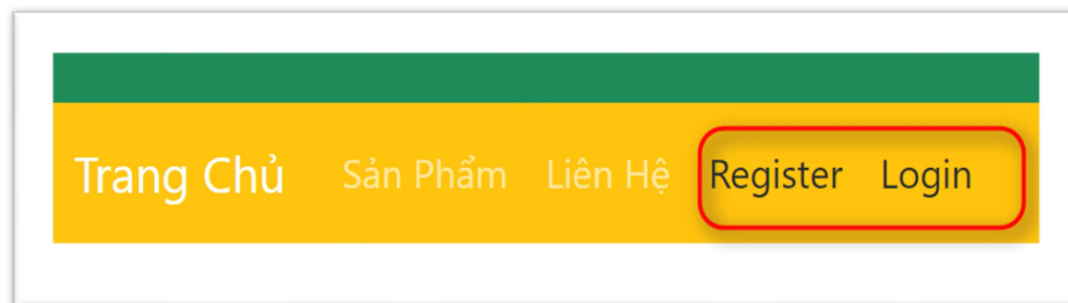
4.2.3 Thêm Liên Kết Đăng Ký và Đăng Nhập

- Cập nhật `'_Layout.cshtml'` để thêm các liên kết đăng ký và đăng nhập:



```
layoutcommerce.cshtml
<li class="nav-item">
  <a class="nav-link" href="#">Liên Hệ</a>
</li>
</ul>
<partial name="_LoginPartial" />
</div>
</div>
```

Kết quả



Use a local account to log in.

Email

Password

☐ Remember me?

Log in

article about setting up this ASP.NET application to support logging in via external services.'"/>

Thay đổi giao diện trang đăng ký và đăng nhập cho phù hợp với trang _Layout mặc định.

4.2.4 Phần vùng Area

Để tạo và thiết lập một Area Admin trong ứng dụng web bán hàng với ASP.NET Core MVC, bạn có thể làm theo các bước chi tiết sau:

1. Tạo Area Admin

- Trong Visual Studio, chuột phải vào project, chọn Add -> New Scaffolded Item
- Chọn Area, đặt tên là `Admin`, và tạo area.

2. Tạo Controllers và Views trong Admin Area:

- Trong `Areas/Admin`, tạo các thư mục `Controllers` và `Views`.
- Tạo `AdminController` trong `Areas/Admin/Controllers` với attribute `[Area("Admin")]`.
- Tạo các view tương ứng trong `Areas/Admin/Views/Admin`.

3. Cấu Hình Routing cho Admin Area:

- Trong `Program.cs`, thêm cấu hình routing cho admin area:

```
endpoints.MapControllerRoute(  
    name: "admin",
```

```
pattern: "{area:exists}/{controller=Home}/{action=Index}/{id?}"  
);
```

4. Thêm Chức Năng Quản Lý Sản Phẩm:

- Tạo `ProductController` trong `Areas/Admin/Controllers`.
- Thêm các action `Index`, `Add`, `Edit`, `Delete` cho việc quản lý sản phẩm.
- Tạo các view tương ứng trong `Areas/Admin/Views/Product`.

5. Quản Lý Đơn Hàng:

- Tương tự, tạo `OrderController` trong `Areas/Admin` và các view tương ứng.

6. Phân Quyền Truy Cập:

- Sử dụng `[Authorize(Roles = "Admin")]` để hạn chế quyền truy cập chỉ cho admin.

7. Tạo Điều Hướng trong Admin Area:

- Trong `_Layout.cshtml` của Admin Area, thêm liên kết tới các chức năng như quản lý sản phẩm, đơn hàng.

4.3 Yêu cầu bổ sung

- Thêm các thông tin bổ sung cho trang đăng ký như: **Address, Age, FirstName, LastName,...vvv**
- Xây dựng lại giao diện cho các trang đăng nhập, đăng ký
- Xây dựng một trang layout mới với mẫu tùy chỉnh phù hợp với một trang web bán hàng, áp dụng trang layout mới này cho tất cả các trang con.
- Triển khai hoàn chỉnh các chức năng cho người quản trị như Thêm/Xóa/Sửa Sản phẩm, danh mục, đơn hàng, ...vvv