ASPNET Core 2.1 MVC – Autenticação

Prof. Henrique Batista da Silva

henriquebsilva@outlook.com.br

ASPNET Core 2.1 MVC

Pré-requisitos

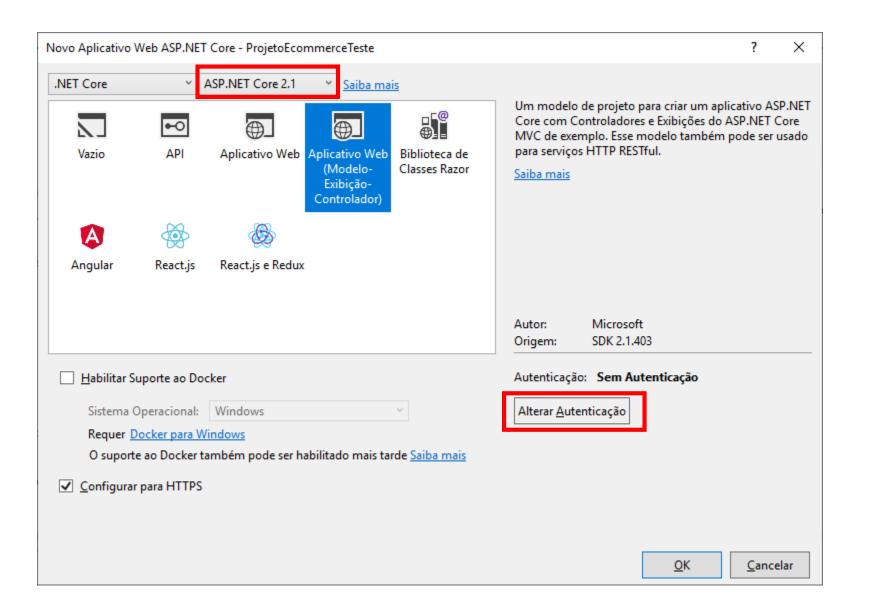
- SDK do .NET Core 2.1.403
- Visual Studio 2017 com "ASP.NET e desenvolvimento para a Web".

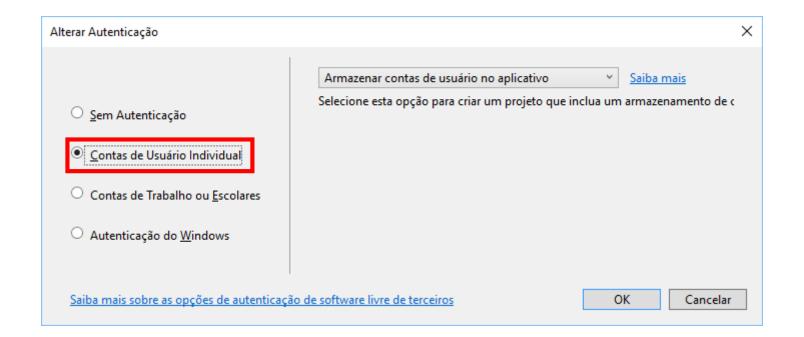
ASPNET Core 2.1 MVC – Autenticação

ASPNET Core 2.0 MVC

• No menu **Arquivo** do Visual Studio 2017, selecione **Novo** > **Projeto**.

 Crie um novo Aplicativo Web ASP.NET Core. Nomeie o projeto como EcommerceApp





Adicionando ferramentas de Scaffold

Observe que o framework já criou uma migração inicial (veja a pasta "Data"). No menu Ferramentas, selecione Gerenciador de pacotes
 NuGet > Console do Gerenciador de pacotes. Então digite os seguintes comandos:

Update-Database

Execute a aplicação

 Com o banco de dados criado anteriormente, execute a aplicação com Ctrl + F5

• Se der erro de certificação, clique em ir para o site.

Customização da aplicação

 Abra o PowerShell e vá até a pasta do projeto (diretório do arquivo Startup.cs) e instale a ferramenta codegenerator do aspnet core

dotnet tool install -g dotnet-aspnet-codegenerator

• Instale também o pacote para visualização do código do identity.

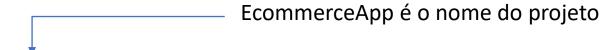
dotnet add package Microsoft. Visual Studio. Web. Code Generation. Design -- version 2.1.1

dotnet restore

Restaura as dependências e as ferramentas de um projeto

Gerando o código do Identity

• Execute o comando abaixo para gerar código do identity para que possamos customizar a aplicação. No aspnet core 2.1 estes código não são mais gerados quando o app é criado.



dotnet aspnet-codegenerator identity -dc EcommerceApp.Data.ApplicationDbContext --files Account.Register

- No visual Studio veja os arquivos criados dentro de Areas->Identity->Pages->Account
- Abra o arquivo "Register.cshtml" e veja o código da página de cadastro e altere de acordo com o código do próximo slide:

```
<div class="row">
   <div class="col-md-4">
       <form asp-route-returnUrl="@Model.ReturnUrl" method="post">
       <h4>Crie nova conta.</h4>
       <hr />
       <div asp-validation-summary="All" class="text-danger"></div>
       <div class="form-group">
           <label asp-for="Input.Email"></label>
           <input asp-for="Input.Email" class="form-control" />
           <span asp-validation-for="Input.Email" class="text-danger"></span>
       </div>
       <div class="form-group">
           <label asp-for="Input.Password"></label>
           <input asp-for="Input.Password" class="form-control" />
           <span asp-validation-for="Input.Password" class="text-danger"></span>
       </div>
       <div class="form-group">
           <label asp-for="Input.ConfirmPassword"></label>
           <input asp-for="Input.ConfirmPassword" class="form-control" />
           <span asp-validation-for="Input.ConfirmPassword" class="text-danger"></span>
       </div>
           <button type="submit" class="btn btn-default">Registrar/button>
       </form>
   </div>
</div>
```

Gerando o código do Identity

• Execute a aplicação com Ctrl + F5

• Explore o site criado e vá em Register.

Gerando o código do Identity

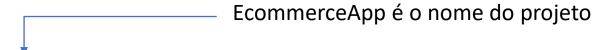
• No asp.net core 2.1 a Microsoft não utiliza mais a classe AccountController para o código do Identity.

 Todo o código é gerado pela PageMode (no nosso exemplo anterior é chamado de "Register.chhtml.cs")

• Observe que usamos o _signInManager para autenticar o usuário e _userManager para gerenciar o usuário.

Gerando o código de login

 Até agora fizemos apenas o cadastro de um novo usuário. Faremos agora o login deste usuário. Retorne a janela do PowerShell e execute o comando abaixo



dotnet aspnet-codegenerator identity -dc ProjetoEcommerceTeste.Data.ApplicationDbContext --files Account.Login

• Observe o código do login criado.

Autorização

• Vamos utilizar o conceito de Claims do Identity 2.1 (antes utilizávamos as "Roles").

 Acesse o arquivo "Register.cshtml.cs", veja o método "OnPostAsync" (método chamado quando um usuário é criado) e modifique o código conforme próximo slide para acrescentamos o tipo de usuário (Admin ou User)

```
if (result.Succeeded)
   _logger.LogInformation("User created a new account with password.");
   await _userManager.AddClaimAsync(user, new Claim("Tipo", "Admin"));
   var code = await _userManager.GenerateEmailConfirmationTokenAsync(user);
   var callbackUrl = Url.Page(
       "/Account/ConfirmEmail",
       pageHandler: null,
       values: new { userId = user.Id, code = code },
       protocol: Request.Scheme);
   await emailSender.SendEmailAsync(Input.Email, "Confirm your email",
       $"Please confirm your account by <a</pre>
       href='{HtmlEncoder.Default.Encode(callbackUrl)}'>clicking here</a>.");
   await _signInManager.SignInAsync(user, isPersistent: false);
   return LocalRedirect(returnUrl);
```

• Execute a aplicação

• Crie o usuário admin e volte ao código da aplicação

```
"UserEmail": "admin@ecommerce.com",
"UserPassword": "P@ssw0rd"
```

 Acesse o arquivo "Register.cshtml.cs" novamente, veja o método "OnPostAsync" (método chamado quando um usuário é criado) e modifique o código conforme próximo slide para acrescentamos o tipo de usuário que agora será "user"

```
if (result.Succeeded)
   _logger.LogInformation("User created a new account with password.");
   await _userManager.AddClaimAsync(user, new Claim("Tipo", "User"));
   var code = await _userManager.GenerateEmailConfirmationTokenAsync(user);
   var callbackUrl = Url.Page(
       "/Account/ConfirmEmail",
       pageHandler: null,
       values: new { userId = user.Id, code = code },
       protocol: Request.Scheme);
   await emailSender.SendEmailAsync(Input.Email, "Confirm your email",
       $"Please confirm your account by <a</pre>
       href='{HtmlEncoder.Default.Encode(callbackUrl)}'>clicking here</a>.");
   await signInManager.SignInAsync(user, isPersistent: false);
   return LocalRedirect(returnUrl);
```

• Pronto, a partir deste momento qualquer usuário cadastrado na aplicação será do tipo "User" e teremos apenas um único admin.

• O Admin cadastro poderia ser feito com um script SQL nas tabelas do Identity.

• Iremos configurar mais tarde o permissionamento de cada usuário dentro das áreas do site. Mas por enquanto iremos criar os modelos.

Testando a aplicação

- Pressione Ctrl + F5 para executar o aplicativo sem modo depuração.
 - Melhor pois qualquer alteração no código da página pode ser visualizada apenas com uma atualização do browser.
 - A página deverá abrir automaticamente no seu browser. O aplicativo é executado pelo IIS. Observe que ele é executado em http://localhost:porta

Faça login com o usuário admin.

```
"UserEmail": "admin@ecommerce.com",
"UserPassword": "P@ssw0rd"
```

ASPNET Core 2.1 MVC – Autorização

Restringindo páginas apenas para usuários autenticados

• Vamos limitar determinado locais da página apenas para usuários autorizados. No nosso caso apenas o "Admin" poderá ter acesso ao app. "User" e anônimos terão acesso apenas a Home do site.

 Acesse o arquivo Startup.cs e nele vamos criar uma política de acesso (policy) para o admin

```
public void ConfigureServices(IServiceCollection services)
   services.Configure<CookiePolicyOptions>(options =>
       // This lambda determines whether user consent for non-essential cookies is needed
       for a given request.
       options.CheckConsentNeeded = context => true;
       options.MinimumSameSitePolicy = SameSiteMode.None;
   });
   services.AddDbContext<ApplicationDbContext>(options =>
       options.UseSqlServer(
           Configuration.GetConnectionString("DefaultConnection")));
   services.AddDefaultIdentity<IdentityUser>()
                                                                          Tipo de Claim criado
       .AddEntityFrameworkStores<ApplicationDbContext>();
   services.AddMvc().SetCompatibilityVersion(CompatibilityVersion.Version 2 1);
   services.AddAuthorization(options => {
       options.AddPolicy("SiteAdmin", policy => policy.RequireClaim("Tipo", "Admin"));
   });
                                                           Nome do Claim criado
                       Nome da policy
```

Restringindo páginas apenas para usuários autenticados

• Acesse o arquivo HomeController e altere conforme código abaixo

Observe que agora estamos aplicando a política em toda a app. Ou seja somente usuário Admin poderá acessar o site

Porém, observe que para o Index.html vamos permitir que qualquer usuário, autenticado ou não, acesse a home do site. Veja que [AllowAnonymous] Se sobrepõe a qualquer política de acesso. Mesmo usuário autenticado que não sejam Admin do site terão acesso.

Restringindo páginas apenas para usuários autenticados

Faça o mesmo procedimento com as demais Controllers

 Salve o arquivo, volte na página, faça logout e veja que agora é possível acessar a página da home sem login, mas as outras parte não.

• Faça login com o Admin e veja que ele acessa todo o site.

Referências

 https://docs.microsoft.com/ptbr/aspnet/core/security/authentication/identity?view=aspnetcore-2.1&tabs=visual-studio