UserStories Part 12 – Login og authentication

Du skal fortsætte med projektet scrumboard.

I denne opgave skal du lave en login side til dit ScrumBoard projekt. Login-funktionen vil benytte en "simpel" Cookie-baseret autentifikation uden tilknyttet database (dvs løsningen er ikke baseret på det mere advancerede ASP.Net Identity framework)

Opret en model-klasse User

I projektet ScrumboardLib i mappen model lav en klasse '**User**' der indeholder properties til **UserName**, **Password** og **Rolle** (til senere brug).

Husk: der skal være både en default konstruktør og en konstruktør til at initialisere properties.

Opret en MockUsers

I projektet Scrumboard i mappen MockData skal du lave en ny klasse '**MockUsers'**, som skal have en statisk liste users, initialiseret med et par User objekter, samt en statisk metode GetMockUsers( ), der returnere listen users.

EKSTRA: lav det som en tabel i en Database

Opret en UserService

I projektet Scrumboard i mappen Services skal du lave et interface '**IUserService**' med med metoderne (alternativt benyt det generiske interface IService):

a. List<User> GetAll()
b. User GetByString(String username)
c. User Create(User newUser)
d. User Delete(String username)
e. User Modify(User modifiedUser)

Husk at lave interfacet public

Du skal lige ledes lave en klasse '**UserService'**, som implementerer interfacet 'IUserService' og som har en statisk liste users, der initialiseres ved kald af MockUsers.GetMockUsers() (alternativt benyt det generiske klasse ServiceGeneric).

Tilpas Startup-klassen

Registrer UserService i ConfigureServices ved at tilføje: services.AddSingleton<UserService, UserService>();

Configurer din Razor applikation til at benytte Cookies

Der skal nu tilføjes konfiguration for den cookie-baserede autentification.

Du skal derfor tilføje følgende kode til ConfigureServices( ) metoden i Startup.cs for at understøtte at applikationen benytter Cookies:

```
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.AddAuthentication(
CookieAuthenticationDefaults.AuthenticationScheme).
   AddCookie(cookieOptions =>
   {
      cookieOptions.LoginPath = "/Login/Login";
   });

services.AddMvc().AddRazorPagesOptions(options =>
{
   options.Conventions.AuthorizeFolder("/UserStories");
}).SetCompatibilityVersion(CompatibilityVersion.Version_2_2);
```

og følgende kode: **`app.UseAuthentication( );`**

HUSK: mellem app.UseRouting(); og app.UseAuthorization(); i Configure( ) metoden

HUSK-HUSK: Relevante using statements (imports), fx using Microsoft.AspNetCore.Http; og using Microsoft.AspNetCore.Authentication.Cookies;

Opret en Login-side

I projektet Scrumboard i mappen pages skal du lave en ny mappe '**Login'**, og tilføj en ny Razor Pages: '**Login**' til mappen.

Tilpas Controlleren 'Login.cshtml.cs'

1. LogInModel skal have en static property der kan referere til en User der er logget ind:

   public static User LoggedInUser { get; set; } = null;

2. Der skal desuden være en reference til UserService:

   private IUserService _userService;

   Husk, at initialisere servicen i konstruktøren!

3. Og der skal være 3 properties: **UserName**, **Password** og **Message** af typen string.
   UserName skal annoteres med [BindProperty] og
   Password med [BindProperty, DataType(DataType.Password)]
   (Du venter med Rolle)

4. Til sidst skal OnPost( ) implementeres. Metoden skal først hente listen af Users via _userService og tjekke om brugeren med (UserName, Password) findes i listen.

   Hvis bruger findes sættes LoggedInUser til den aktuelle user og der oprettes en ny liste af Claim-objekter der initialiseres med et nyt Claim indeholdende vores brugers UserName.

   Dernæst oprettes et ClaimsIdentity - objekt med vores nye liste af Claims og der laves et SignIn med denne ClaimsIdentity:

```
public async Task<IActionResult> OnPost()
{

   List<User> users = _userService.Users;
   foreach (User user in users)
   {

      if (UserName == user.UserName && Password == user.Password)
      {

         LoggedInUser = user;

         var claims = new List<Claim> { new Claim(ClaimTypes.Name, UserName)
      };

      var claimsIdentity = new ClaimsIdentity(claims,
                        CookieAuthenticationDefaults.AuthenticationScheme);

      await HttpContext.SignInAsync(
            CookieAuthenticationDefaults.AuthenticationScheme,
            new ClaimsPrincipal(claimsIdentity));

      return RedirectToPage("/UserStories/Index");
      }
   }

   Message = "Invalid attempt";
   return Page();
}
```

Tilpas View - 'Login.cshtml'

Her tilføjes html-elementer så bruger kan indtaste UserName og Password mm ala:

```
<div class="container">
   <div>
   @Model.Message
   </div>
   <form method="post">
      <div class="form-group">
         <label asp-for="UserName" class="col-form-label "></label>
         <div>
         <input asp-for="UserName" />
         </div>
      </div>
      <div class="form-group">
         <label asp-for="Password" class="col-form-label"></label>
         <div>
         <input asp-for="Password" />
         </div>
     </div>
     <div class="form-group">
     <button class="btn btn-outline-light">Log in</button>
     </div>
   </form>
</div>
```

Afprøv programmet

Opret Logout-side

I projektet Scrumboard i mappen pages/Login skal du lave en ny Razor Pages: '**Logout**' til mappen.

Tilpas Controlleren 'Logout.cshtml.cs'

Siden skal kun have en funktion OnGet( ):

```
public async Task<IActionResult> OnGet()
{
   LoginPageModel.LoggedInUser = null;

   Await  HttpContext.SignOutAsync(
                CookieAuthenticationDefaults.AuthenticationScheme);
   return RedirectToPage("/index");
}
```

Denne funktion skal kaldes, når der skal logges af.

Tilpas view - 'Logout.cshtml'

Du skal ikke lave noget her (view bliver ikke vist)

Tilpas Layout-side med login-faciliteter

I projektet Scrumboard i mappen pages/shared skal du rette filen '_Layout.cshtml'.
Der skal nu tilføjes knapper i menuen til Login og Logout:

Udskift følgende linje (ca linje 21): med:

Efter (ca linje 31)
<li class="nav-item">
  Privacy
:

Indsæt:

```
@{
if (LoginModel.LoggedInUser == null)
{
<li class="nav-item">
<a class="nav-link" asp-area="" asp-page="/Login/Login">Login</a>
</li>
}
else
{
<li class="nav-item nav-link mr-3">
User: @LoginModel.LoggedInUser.UserName
</li>

<li class="nav-item">
<a class="nav-link" asp-area="" asp-page="/LogIn/Logout">Logout</a>
</li>
}
}
</ul>
```

Bemærk: Der skal tilføjes en using til LogIn, tilføj følgende som linie 1 i filen: @using ItemRazor.Pages.LogIn

Lige lidt oprydning af Index-siden

Når applikationen startes kan der ligge gamle login-cookies. Derfor skal vi lige sikre os at den tidligere user er "signed out". Det gøres ved at tilføje følgende kode til OnGet( ) metoden i Index.cshtml.cs

```
public void OnGet()
{
if (LoginPageModel.LoggedInUser == null)
{
HttpContext.SignOutAsync(CookieAuthenticationDefaults.AuthenticationScheme)
;
}
}
```

Prøv din applikation

Extra LÆS

- Role-based authorization in ASP.NET Core | Microsoft Docs
- Claims-based authorization in ASP.NET Core | Microsoft Docs

Tilføj brug af roller til din applikation

Læs: Razor Pages authorization conventions in ASP.NET Core | Microsoft Docs

Tilføj beskyttelse af dine sider