ASP.NET MVC – Identity

ספריות לעבודה עם Identity

Microsoft.AspNetCore.Identity.EntityFrameworkCore by Microsoft
ASP.NET Core Identity provider that uses Entity Framework Core.

Microsoft.EntityFrameworkCore.SqlServer by Microsoft
Microsoft SQL Server database provider for Entity Framework Core.

Microsoft.EntityFrameworkCore.Tools by Microsoft
Entity Framework Core Tools for the NuGet Package Manager Console in Visual Studio.

מודל למשתמש – יורש מהמחלקה המובנית IdentityUser

```
using Microsoft.AspNetCore.Identity;
using System.ComponentModel.DataAnnotations;

namespace MVCAuth.Models;

public class AppUser: IdentityUser
{
    [Required, MinLength(2) ,MaxLength(20)]
    public required string Language { get; set; }
}
```

מודל להרשמה

```
namespace MVCAuth.Areas.Identity.Models;

using System.ComponentModel.DataAnnotations;

public class RegisterViewModel
{
    [Required]
    [EmailAddress]
    [Display(Name = "Email")]
    public required string Email { get; set; }

    [Required]
    [DataType(DataType.Password)]
    [Display(Name = "Password")]
    public required string Password { get; set; }

    [DataType(DataType.Password)]
    [Display(Name = "Confirm password")]
    [Compare("Password", ErrorMessage = "The password and confirmation password do not match.")]
    public required string ConfirmPassword { get; set; }

    [Required, MinLength(2), MaxLength(20)]
    public required string Language { get; set; }
}
```

מודל להתחברות

```
using System.ComponentModel.DataAnnotations;

namespace MVCAuth.Areas.Identity.Models;

public class LoginViewModel
{
    [Required, EmailAddress]
    public required string Email { get; set; }

    [Required, MinLength(2), MaxLength(40)]

    public required string Password { get; set; }
}
```

Database Context – Entity Framework:

```
using Microsoft.AspNetCore.Identity;
using Microsoft.AspNetCore.Identity.EntityFrameworkCore;
using Microsoft.EntityFrameworkCore;
using MVCAuth.Models;

namespace MVCAuth.Data;

public class ApplicationDbContext(DbContextOptions<ApplicationDbContext> options) :
IdentityDbContext<AppUser, IdentityRole, string>(options)
{
    public DbSet<Product> Products { get; set; }
}
```

הגדרות בקובץ Program.cs

```
// Add services to the container.
var connectionString = builder.Configuration.GetConnectionString("DefaultConnection")
?? throw new InvalidOperationException("Connection string 'DefaultConnection' not
found.");
builder.Services.AddDbContext<ApplicationDbContext>(options =>
    options.UseSqlServer(connectionString));

builder.Services.AddIdentity<AppUser, IdentityRole>(options => {
    //  options.SignIn.RequireConfirmedAccount = true;
    options.User.RequireUniqueEmail = true;
    options.Password.RequiredLength = 8;
})
    .AddEntityFrameworkStores<ApplicationDbContext>();
```

1

```
app.UseRouting();

app.UseAuthentication();
app.UseAuthorization();

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

app.Run();
```

2

Database Context – Entity Framework: Package Manager Console

Add-Migration Auth

1

Update-Database

2

1

ניצור קונטרולר להרשמה/התחברות/התנתקות/הצגת פרטים
AuthController

2

נוסיף קישורים לסרגל הניווט – קישור להרשמה/התחברות/התנתקות/פרטי משתמש:

3

כדי לממש את הקישורים - יש לנו 2 מצבים:

1) המשתמש מחובר - נרצה להציג לו כפתור התנתקות וכפתור לעריכת פרטי משתמש.
2) המשתמש אינו מחובר – נציג קישורים לעמוד התחברות ולעמוד הרשמה.

הספריה נותנת לנו Services להתחברות הרשמה וכן משתנה שנגיש לכל Controller בירושה בשם User

נוח יותר ליצור Partial View לטובת הנושא כדי ליצור הפרדה נושאית

```
public class AuthController : Controller
{
    0 references
    public IActionResult Login()
    {
        return View();
    }

    0 references
    public IActionResult Register()
    {
        return View();
    }

    0 references
    public IActionResult Logout()
    {
        return View();
    }

    0 references
    public IActionResult Manage()
    {
        return View();
    }
}
```

הקונטרולר:

כדי לממש את הקישורים – יש לנו 2 מצבים:

1) המשתמש מחובר – נרצה להציג לו כפתור התנתקות וכפתור לעריכת פרטי משתמש.
2) המשתמש אינו מחובר – נציג קישורים לעמוד התחברות ולעמוד הרשמה.

לשם כך ניצור ***Partial View*** לטובת הנושא כדי ליצור הפרדה נושאית נקרא לקובץ ***_LoginPartial.cshtml***

הספריה נותנת לנו Services להתחברות הרשמה וכן משתנה שנגיש לכל Controller בירושה בשם User

אפשר וכך נהוג – להשתמש ב*Services* הללו ב*Razor View* וכך להנגיש את מצב ההתחברות של המשתמש לכלל העמודים (כלומר זה צריך להיות ב*Index* *(Layout*

לטובת עיצוב של הPartial נעתיק את הHTML הבא מקובץ Layout
לקובץ _LoginPartial.cshtml ונשנה אותו כרצוננו

```
<body>
    <header>
        <nav class="navbar navbar-expand-sm navbar-toggleable-sm navbar-light bg-
            <div class="container-fluid">
                <a class="navbar-brand" asp-area="" asp-controller="Home" asp-act
<html lang="en">
                <nav class="navbar navbar-expand-sm navbar-toggleable-sm navbar-light bg-white borde
                    <span class="navbar-toggler-icon"></span>
                </button>
                <div class="navbar-collapse collapse d-sm-inline-flex justify-con
                    <ul class="navbar-nav flex-grow-1">
                        <li class="nav-item">
                            <a class="nav-link text-dark" asp-area="" asp-control
                        </li>
                        <li class="nav-item">
                            <a class="nav-link text-dark" asp-area="" asp-control
                        </li>
                    </ul>
                </div>
            </div>
        </nav>
    </header>
```

עיצוב - הקובץ _LoginPartial.cshtml

_LoginPartial.cshtml

```
<ul class="navbar-nav">
    <li class="nav-item">
        <a class="nav-link text-dark" asp-controller="Auth" asp-action="Login">Login</a>
    </li>
    <li class="nav-item">
        <a class="nav-link text-dark" asp-controller="Auth" asp-action="Register">Register</a>
    </li>

    <li class="nav-item">
        <a class="nav-link text-dark" asp-controller="Auth" asp-action="Manage">Manage</a>
    </li>

    <li class="nav-item">
        <a class="nav-link text-dark" asp-controller="Auth" asp-action="Logout">Logout</a>
    </li>
</ul>
```

עכשיו יש לנו קישורים אבל הם מוצגים ללא תנאי –
לטובת הצגה בתנאי – נזריק את הServices של הספריה Identity

_LoginPartial.cshtml

```
@using Microsoft.AspNetCore.Identity
@inject UserManager<AppUser> userManager
@inject SignInManager<AppUser> signInManager

<ul class="navbar-nav">

    @if (!signInManager.IsSignedIn(User))
    {
        <li class="nav-item">
            <a class="nav-link text-dark" asp-controller="Auth" asp-action="Login">Login</a>
        </li>
        <li class="nav-item">
            <a class="nav-link text-dark" asp-controller="Auth" asp-action="Register">Register</a>
        </li>
    }
    else
    {
        <li class="nav-item">
            <a class="nav-link text-dark" asp-controller="Auth" asp-action="Manage">Manage</a>
        </li>

        <li class="nav-item">
            <a class="nav-link text-dark" asp-controller="Auth" asp-action="Logout">Logout</a>
        </li>
    }
</ul>
```

_Layout.cshtml

```
<body>
    <header>
        <nav class="navbar navbar-expand-sm navbar-toggleable-sm navbar-light bg-white bo
            <div class="container-fluid">
                <a class="navbar-brand" asp-area="Home" asp-controller="Home" asp-action=
                <button class="navbar-toggler" type="button" data-bs-toggle="collapse" da
                        aria-expanded="false" aria-label="Toggle navigation">
                    <span class="navbar-toggler-icon"></span>
                </button>
                <div class="navbar-collapse collapse d-sm-inline-flex justify-content-bet
                    <ul class="navbar-nav flex-grow-1">
                        <li class="nav-item">
                            <a class="nav-link text-dark" asp-area="Home" asp-controller=
                        </li>
                        <li class="nav-item">
                            <a class="nav-link text-dark" asp-area="Home" asp-controller=
                        </li>
                    </ul>
                    <partial name="_LoginPartial" />
                </div>
            </div>
        </nav>
```

קונטרולר להרשמה והתחברות: בקשת GET להרשמה:

```
using Microsoft.AspNetCore.Identity;
using Microsoft.AspNetCore.Mvc;
using MVCAuth.Areas.Identity.Models;
using MVCAuth.Models;

namespace MVCAuth.Areas.Identity.Controllers;

public class AuthController(UserManager<AppUser>
userManager, SignInManager<AppUser> signInManager) :
Controller
{
    // GET: Identity/Auth/Login
    public ActionResult Register()
    {
        return View();
    }
```

עיצוב
להרשמה:

```
@model RegisterViewModel
@{
    ViewData["Title"] = "Register";
}
<h1>@ViewData["Title"]</h1>
<div class="row">
    <form asp-action="Register" method="post">

        <div class="form-group">
            <label asp-for="Email"></label>
            <input asp-for="Email" class="form-control" />
            <span asp-validation-for="Email" class="text-danger"></span>
        </div>
        <div class="form-group">
            <label asp-for="Language"></label>
            <input asp-for="Language" class="form-control" />
            <span asp-validation-for="Language" class="text-danger"></span>
        </div>
        <div class="form-group">
            <label asp-for="Password"></label>
            <input asp-for="Password" class="form-control" type="password" />
            <span asp-validation-for="Password" class="text-danger"></span>
        </div>
        <div class="form-group">
            <label asp-for="ConfirmPassword"></label>
            <input asp-for="ConfirmPassword" class="form-control" type="password" />
            <span asp-validation-for="ConfirmPassword" class="text-danger"></span>
        </div>
        <button type="submit" class="btn btn-primary">Register</button>
    </form>
</div>

@section Scripts {
    <partial name="_ValidationScriptsPartial" />
}
```

קונטרולר להרשמה והתחברות: בקשת POST להרשמה:

```
// POST: Identity/Auth/Register
[HttpPost]
[ValidateAntiForgeryToken]
public async Task<IActionResult> Register(RegisterViewModel model)
{
    if (ModelState.IsValid)
    {
        var user = new AppUser { UserName = model.Email, Email = model.Email, Language =
model.Language };
        var result = await userManager.CreateAsync(user, model.Password);

        if (result.Succeeded)
        {
            await signInManager.SignInAsync(user, isPersistent: true);
            return Redirect("/");
        }

        foreach (var error in result.Errors)
        {
            ModelState.AddModelError(string.Empty, error.Description);
        }
    }

    return View(model);
}
```

צריך כאן את הלולאה כי הבדיקה מבוצעת ע"י
אובייקט של הרשמה והתחברות ולא במנגנון הרגיל
של ולידציות

שיעורי בית:

מומלץ לחזור על המצגת ולנסות לממש בעצמכם את כל מה שמימשנו בכיתה בפרוייקט חדש לטובת התרגול.

כמו תמיד – מוזמנים להגיע לשיעור הבא עם שאלות שעלו לכם במהלך התרגול.