

# ASP.NET MVC – Identity

הצגת שגיאות ולידצית צד שרת -  
בעמוד הרשמה

Register.cshtml

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

        <span asp-validation-summary="All"></span>

        <div class="form-group">
```

All = Properties and Model Errors

Model Errors = string.Empty Errors

```
ModelState.AddModelError(string.Empty, "'ALL/ModelOnly' error occurred.");
ModelState.AddModelError(nameof(AppUser.Email), "'ALL' Invalid email address.");
```

## קונטרולר להרשמה והתחברות: בקשת GET להרשמה: (הצגת הודעות שגיאה בהרשמה)

```
[HttpPost]
public async Task<IActionResult> Register(RegisterViewModel vm)
{
    if (ModelState.IsValid)
    {
        var user = new AppUser() { Language = vm.Language, Email = vm.Email, UserName = vm.Email };
        var result = await userManager.CreateAsync(user, vm.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(vm);
}
```

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

קונטרולר להרשמה והתחברות: בקשת POST להתנתקות:  
בקשת GET להצגת טופס התחברות:

```
[HttpPost]
public async Task<IActionResult> Logout()
{
    await signInManager.SignOutAsync();
    return Redirect("/");
}

//GET Identity/Auth/Login
public ActionResult Login()
{
    return View();
}
```

```
@model LoginViewModel
```

```
@{  
    ViewData["Title"] = "Login";  
    string returnUrl = ViewBag.ReturnUrl;  
}
```

```
<h1>@ViewData["Title"]</h1>
```

```
<div class="row">
```

```
    <form asp-action="Login" method="post">
```

```
        <div asp-validation-summary="All"></div>
```

```
        <div class="form-group">
```

```
            <label asp-for="Email"></label>
```

```
            <input asp-for="Email" class="form-control" />
```

```
            <span asp-validation-for="Email"></span>
```

```
        </div>
```

```
        <input type="hidden" name="ReturnUrl" value="@ReturnUrl" />
```

```
        <div class="form-group">
```

```
            <label asp-for="Password"></label>
```

```
            <input asp-for="Password" class="form-control" />
```

```
            <span asp-validation-for="Password"></span>
```

```
        </div>
```

```
        <button type="submit" class="btn btn-primary">Login</button>
```

```
    </form>
```

```
</div>
```

```
@section Scripts{
```

```
    <partial name="_ValidationScriptsPartial"/>
```

```
}
```

עיצוב להתחברות:  
(כולל הצגת שגיאות ולידציה צד שרת)

## קונטרולר לAuth: בקשת POST להתחברות:

```
[HttpPost]
public async Task<IActionResult> Login(LoginViewModel vm, string returnUrl = "/")
{
    if (ModelState.IsValid)
    {
        //var user = await userManager.FindByEmailAsync(vm.Email);
        //var result = signInManager.PasswordSignInAsync(user, vm.Password, true, false);
        var result = await signInManager.PasswordSignInAsync(vm.Email, vm.Password, true, false);

        if (result.Succeeded)
        {
            return Redirect(returnUrl);
        }

        ModelState.AddModelError(string.Empty, "Invalid Login Attempt");
    }
    return View(vm);
}
```

remember me

LockOut on failure



# הגדרות - LockOut on failure

Program.cs

```
var lockoutOptions = new LockoutOptions()
{
    DefaultLockoutTimeSpan = TimeSpan.FromMinutes(5),
    MaxFailedAccessAttempts = 5,
    AllowedForNewUsers = false
};

builder.Services
    .AddIdentity<AppUser, IdentityRole>(options =>
    {
        options.Lockout = lockoutOptions;
        options.User.RequireUniqueEmail = true;
        options.Password.RequireNonAlphanumeric = false;
        options.Password.RequireDigit = false;
        options.Password.RequireLowercase = false;
        options.Password.RequireUppercase = false;
        options.Password.RequiredLength = 6;
    })
    .AddEntityFrameworkStores<Lec7Context>();
```

```
namespace Lec7.ViewModels;

public class ManageViewModel
{
    public string? Password { get; set; } = null;

    public string? NewPassword { get; set; } = null;

    public string? Language { get; set; } = null;
}
```



```
public async Task<IActionResult> Manage()
{
    var user = await userManager.GetUserAsync(User);
    if (user == null)
    {
        return Redirect("/");
    }
    ViewBag.UserName = user.UserName;

    var vm = new ManageViewModel() { Language = user.Language };
    return View(vm);
}
```

```
@model ManageViewModel
```

```
@{
```

```
    ViewData["Title"] = "Manage";  
    string username = ViewBag.UserName;
```

```
}
```

```
<h1>Edit Your Profile @username</h1>
```

```
<div class="row">
```

```
    <form asp-action="Manage" method="post">  
        <div asp-validation-summary="All"></div>
```

```
        <div class="form-group">  
            <label asp-for="Password"></label>  
            <input class="form-control" type="password" asp-for="Password" />  
            <span asp-validation-for="Password"></span>  
        </div>
```

```
        <div class="form-group">  
            <label asp-for="NewPassword"></label>  
            <input class="form-control" type="password" asp-for="NewPassword" />  
            <span asp-validation-for="NewPassword"></span>  
        </div>
```

```
        <div class="form-group">  
            <label asp-for="Language"></label>  
            <input class="form-control" asp-for="Language" />  
            <span asp-validation-for="Language"></span>  
        </div>
```

```
        <button type="submit" class="btn btn-primary mt-2">Save Changes</button>
```

```
    </form>
```

```
</div>
```

```
@section Scripts{
```

```
    <partial name="_ValidationScriptsPartial"/>
```

```
}
```

עריכת פרטים קובץ עיצוב Manage.cshtml

[HttpPost]

```
public async Task<IActionResult> Manage(ManageViewModel vm)
{
    var user = await userManager.GetUserAsync(User);

    if (user is null)
    {
        return Redirect("/");
    }
    if (vm.Password is not null && vm.NewPassword is not null)
    {
        //userManager.ChangePasswordAsync
        var result = await userManager.ChangePasswordAsync(user, vm.Password, vm.NewPassword);

        if (!result.Succeeded)
        {
            foreach (var error in result.Errors)
            {
                ModelState.AddModelError("", error.Description);
            }
            return View(vm); //the new pass is 2 short/passwords dont match
        }
    }

    if (vm.Language is not null)
    {
        user.Language = vm.Language;
        await userManager.UpdateAsync(user);
    }
    return Redirect("/");
}
```

עריכת פרטים מתודת POST בController

הגנה על Controller כך שיאפשר רק למשתמש מחובר להכנס:

```
[Authorize]
public class ProductsController : Controller
{
    private readonly Lec7Context _context;

    public ProductsController(Lec7Context context)
    {
        _context = context;
    }

    // GET: Products
    public async Task<IActionResult> Index()
    {
        return View(await _context.Product.ToListAsync());
    }
}
```

אפשר לשים את האנוטציה הזאת מעל מחלקה או מעל מתודה  
בהתאם לצורך בפרויקט

כשנבדוק נראה שהRedirect Url מוגדר לכתובת ברירת מחדל  
שאינה נכונה מבחינתנו – פתרון בעמוד הבא

## שינוי הגדרת נתיב להתחברות (ברירת מחדל של הספרייה)

1

```
builder.Services
    .AddIdentity<AppUser, IdentityRole>(options =>
    {
        options.Lockout = lockoutOptions;
        options.User.RequireUniqueEmail = true;
        options.Password.RequireNonAlphanumeric = false;
        options.Password.RequireDigit = false;
        options.Password.RequireLowercase = false;
        options.Password.RequireUppercase = false;
        options.Password.RequiredLength = 6;
    })
    .AddEntityFrameworkStores<Lec7Context>();
```

יש חשיבות לסדר – זה צריך להיות אחרי

builder.Services.AddIdentity

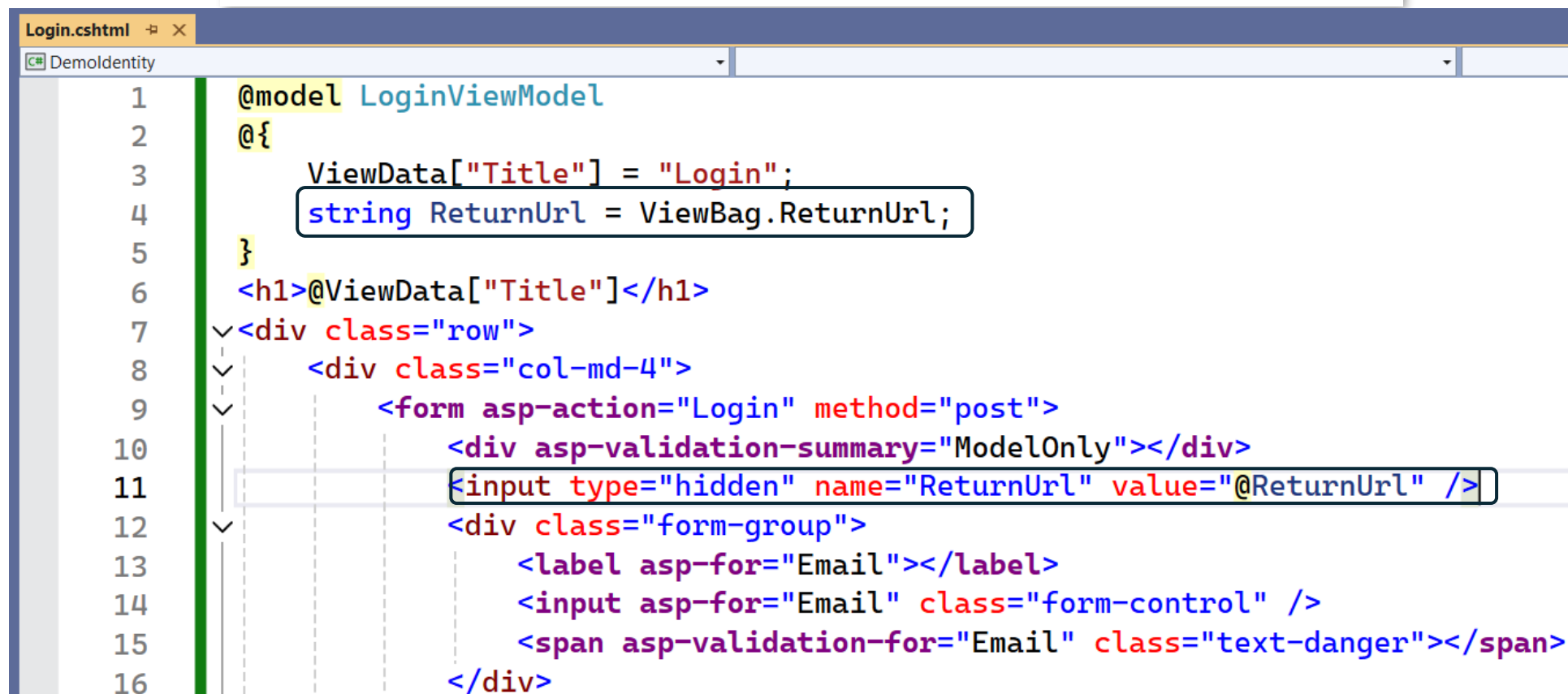


2

```
builder.Services.ConfigureApplicationCookie(options => {
    options.LoginPath = "/Auth/Login";
    options.AccessDeniedPath = "/Auth/AccessDenied";
});
```

נרצה לזכור את הכתובת שממנה נווט המשתמש לLogin כדי להחזירו לפעולה שרצה במקור:

```
public IActionResult Login(string returnUrl = "/")
{
    ViewBag.ReturnUrl = returnUrl;
    return View();
}
```



```
1 @model LoginViewModel
2 @{
3     ViewData["Title"] = "Login";
4     string returnUrl = ViewBag.ReturnUrl;
5 }
6 <h1>@ViewData["Title"]</h1>
7 <div class="row">
8 <div class="col-md-4">
9     <form asp-action="Login" method="post">
10         <div asp-validation-summary="ModelOnly"></div>
11         <input type="hidden" name="ReturnUrl" value="@ReturnUrl" />
12         <div class="form-group">
13             <label asp-for="Email"></label>
14             <input asp-for="Email" class="form-control" />
15             <span asp-validation-for="Email" class="text-danger"></span>
16         </div>
```

נרצה לזכור את הכתובת שממנה נווט המשתמש לLogin כדי להחזירו לפעולה שרצה במקור:

[HttpPost]

0 references

```
public async Task<IActionResult> Login(LoginViewModel vm, string returnUrl = "/")
{
    if (ModelState.IsValid)
    {
        var result = await signInManager.PasswordSignInAsync(vm.Email, vm.Password, isPersistent: true,
        if (result.Succeeded)
        {
            return Redirect(returnUrl);
        }
        ModelState.AddModelError(string.Empty, "Invalid login attempt.");
    }
    return View();
}
```

זריעה של  
הדטה-בייס

```
protected override void OnModelCreating(ModelBuilder builder)
{
    base.OnModelCreating(builder);
```

```
    var hasher = new PasswordHasher<AppUser>();
```

```
    // Seed roles
```

```
    builder.Entity<IdentityRole>().HasData(
```

```
        new IdentityRole { Id = "1", Name = "Admin", NormalizedName = "ADMIN" },
```

```
        new IdentityRole { Id = "2", Name = "User", NormalizedName = "USER" }
```

```
    );
```



## משתמשים וRoles

```
// Create a new user
var user = new AppUser
{
    Id = Guid.NewGuid().ToString(), // Generate a new GUID for the user ID
    UserName = "Tomerbu@gmail.com",
    NormalizedUserName = "TOMERBU@GMAIL.COM",
    Email = "Tomerbu@gmail.com",
    NormalizedEmail = "TOMERBU@GMAIL.COM",
    Language = "CSharp",
    EmailConfirmed = true,
    SecurityStamp = Guid.NewGuid().ToString()
};
user.PasswordHash = hasher.HashPassword(user, "123456");

// Seed the user
builder.Entity<AppUser>().HasData(user);

// Assign the Admin role to the user
builder.Entity<IdentityUserRole<string>>().HasData(
    new IdentityUserRole<string> { RoleId = "1", UserId = user.Id }
);
```

Add-Migration  
SeedDatabase  
Update-Database

זריעה של  
הדטה-בייס

```
// GET: Movies/Create  
[Authorize(Roles = "Admin")]  
0 references  
public IActionResult Create()  
{  
    return View();  
}
```

הגדרת מתודה  
למשתמש מסוג  
Admin בלבד:

## בהוספת משתמש – נשייך אותו לתפקיד של User (שיוך של Role).

```
public async Task<IActionResult> Register(RegisterViewModel vm)
{
    if (ModelState.IsValid)
    {
        var user = new AppUser()
        {
            Email = vm.Email,
            EyeColor = vm.Language,
            UserName = vm.Email
        };
        IdentityResult result = await userManager.CreateAsync(user, vm.Password);
        if (result.Succeeded)
        {
            await userManager.AddToRoleAsync(user, "User");
            await signInManager.SignInAsync(user, isPersistent: true); return Redirect("/");
        }
        foreach (var error in result.Errors)
        {
            ModelState.AddModelError(string.Empty, error.Description);
        }
    }
    return View();
}
```

## שיעורי בית:

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

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