**L04\_E-shop v ASP.NET\_Příprava datové vrstvy**

V dnešním ASP.NET Core tutoriálu si nejprve vytvoříme ApplicationUser, upravíme ApplicationDbContext, nastavíme Identity a upravíme soubor appsettings.json.

## Datová vrstva

V datové vrstvě vytvoříme třídu ApplicationUser. Poté upravíme třídu ApplicationDbContext a soubor Eshop.Data.csproj.

### Třída ApplicationUser

Jelikož do budoucna budeme chtít možnosti uživatele naší aplikace rozšiřovat, nevystačíme si jen s **výchozí** třídou IdentityUser, která je nyní nastavena v souboru Program.cs.

Proto do **datové vrstvy** přidáme novou složku Models/. Do této složky následně přidáme novou třídu ApplicationUser, která bude z IdentityUser dědit. Rozšiřovat ji budeme až později:

**namespace** Eshop.Data.Models

{

**public** **class** ApplicationUser : IdentityUser

{

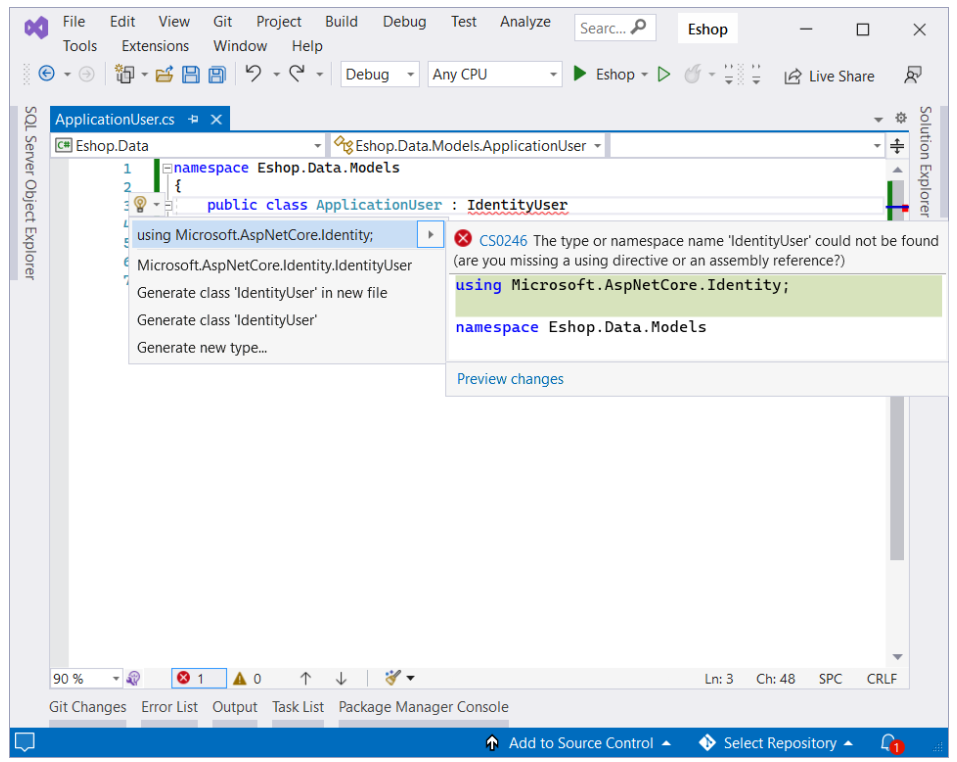
}

}

Při založení každé nové třídy vždy zkontrolujeme existenci modifikátoru přístupu třídy public, aby byla viditelná v jiných projektech.

### Snadné přidání jmenného prostoru

Do třídy ApplicationUser je potřeba, kvůli třídě IdentityUser, naimportovat jmenný prostor Microsoft.AspNetCore.Identity. To se dá velmi rychle provést pomocí funkce Quick Actions and Refactorings... (žárovka) ve Visual Studiu. K té se dostaneme buď kliknutím pravým tlačítkem na text IdentityUser a vybráním Quick Actions and Refactorings... anebo přesunutím kurzoru k textu IdentityUser a stisknutím Ctrl + .. Poté jen stačí z nabídky vybrat požadovaný using Microsoft.AspNetCore.Identity;:



Tímto způsobem budeme v tomto kurzu přidávat většinu jmenných prostorů.

### Třída ApplicationDbContext

Složku Data/ z projektu Eshop přesuneme do projektu Eshop.Data. Tato složka obsahuje třídu ApplicationDbContext a složku pro tzv. migrace Migrations/. ApplicationDbContext upravíme tak, aby pracoval s naším ApplicationUser:

**namespace** Eshop.Data

{

**public** **class** ApplicationDbContext : IdentityDbContext<ApplicationUser>

{

...

}

}

Jmenný prostor se nám automaticky změní na Eshop.Data.

### Soubor Eshop.Data.csproj

Dále ještě potřebujeme upravit soubor .csproj z naší **datové vrstvy**. Jedná se o typ souborů pro **.NET Core**, které umožňují aplikaci nastavit si některá **runtime nastavení** (ve starším .NET Frameworku byl pro podobný účel soubor App.config). K souboru se dostaneme tak, že v okně Solution Explorer dvakrát klikneme na projekt Eshop.Data. Do elementu <PropertyGroup> vložíme řádek s <GenerateRuntimeConfigurationFiles>:

...

**<PropertyGroup>**

...

**<GenerateRuntimeConfigurationFiles>**true**</GenerateRuntimeConfigurationFiles>**

**</PropertyGroup>**

...

## Aplikační vrstva

V aplikační vrstvě nastavíme Identity a upravíme soubor appsettings.json.

### Nastavení Identity

V souboru Program.cs:

* změníme volání builder.Services.AddDefaultIdentity() na builder.Services.AddIdentity() s naším ApplicationUser,
* nastavíme minimální **délku hesla** na 8 znaků,
* vypneme povinnost použití **nealfanumerických znaků**.
* nastavíme na false obvyklé **potvrzení** emailové adresy, které v naší aplikaci dále nebudeme vyžadovat,
* nastavíme na true použití **unikátní** e-mailové adresy,
* přidáme registraci **Razor stránek** pomocí builder.Services.AddRazorPages().

Změněná část souboru Program.cs vypadá teď následovně:

**ar** builder = WebApplication.CreateBuilder(**args**);

// Add services to the container.

**var** connectionString = builder.Configuration.GetConnectionString("DefaultConnection");

builder.Services.AddDbContext<ApplicationDbContext>(options =>

options.UseSqlServer(connectionString));

builder.Services.AddDatabaseDeveloperPageExceptionFilter();

builder.Services.AddIdentity<ApplicationUser, IdentityRole>(options =>

{

options.Password.RequiredLength = 8;

options.Password.RequireNonAlphanumeric = **false**;

options.SignIn.RequireConfirmedAccount = **false**;

options.User.RequireUniqueEmail = **true**;

})

.AddEntityFrameworkStores<ApplicationDbContext>();

builder.Services.AddControllersWithViews();

builder.Services.AddRazorPages();

**var** app = builder.Build();

...

### Nastavení connection stringu

Popis umístění databázového souboru najdeme v **konfiguračním** souboru appsettings.json. V něm můžeme najít např. takovéto [nastavení](csharp/asp-net-core/web-api/aspnet-web-api-databaze#_connection-string):

...

"ConnectionStrings": {

"DefaultConnection": "Server=(localdb)\\mssqllocaldb;Database=aspnet-EShop-05E9600A-095B-4366-A041-73CA2A6B0137;Trusted\_Connection=True;MultipleActiveResultSets=true"

},

...

Nastavení udává, že data budou uložena do souboru s poněkud nepěkným názvem aspnet-EShop-05E9600A-095B-4366-A041-73CA2A6B0137 (název souboru se u vás bude pravděpodobně lišit). V ASP.NET Core pracujeme při vývoji s databází **LocalDB**, což je vlastně jednodušší, lightweight verze SQL Server Express databáze. Jelikož se spouští na vyžádání a téměř bez potřeby jakékoliv konfigurace, hodí se nám zejména pro vývoj.

Při nasazení aplikace na server bychom pro sestavení connection stringu nejspíš využili proměnné prostředí nebo jiný přístup, každopádně bychom jej v konečném důsledku změnili právě tady.

**LocalDB** ukládá databazové .mdf soubory defaultně do složky C:/Users/<user>.

Vygenerovaný název si sice můžeme změnit, jestliže se nám nelíbí, avšak **musíme si dát pozor**, aby nekolidoval s názvem nějaké již existující databáze.  
My si název databáze změníme na aspnet-Eshop-6-0:

...

"ConnectionStrings": {

"DefaultConnection": "Server=(localdb)\\mssqllocaldb;Database=aspnet-Eshop-6-0;Trusted\_Connection=True;MultipleActiveResultSets=true"

},

...

### Předání connection stringu

Vraťme se zpět do souboru Program.cs. Zde můžeme vidět předání informací o **connection stringu** našemu kontextu ApplicationDbContext:

...

**var** builder = WebApplication.CreateBuilder(**args**);

// Add services to the container.

**var** connectionString = builder.Configuration.GetConnectionString("DefaultConnection");

builder.Services.AddDbContext<ApplicationDbContext>(options =>

options.UseSqlServer(connectionString));

...

To je pro tuto lekci vše.