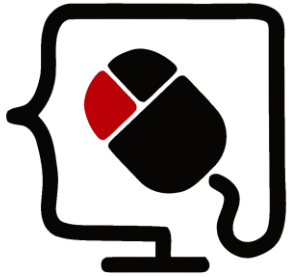


Национална програма
"Обучение за ИТ умения и кариера"
https://it-kariera.mon.bg/e-learning/

Министерството на
образованието и науката
https://www.mon.bg

Автентикация и авторизация

Authentication and Authorization

Съдържание

- Основна идентичност на ASP.NET
- Видове за удостоверяване
- Социални акаунти

Автентикация и Авторизация

Каква е разликата?

Автентикация срещу Авторизация [1/2]

- Автентикация
  - Процесът на проверка на самоличността на потребител или компютър
  - Въпроси: Кой си ти? Как го доказваш?
  - Поверителните данни могат да бъдат парола, смарт карта, външен маркер и т.н.
- Авторизация
  - Процесът на определяне на това, което на потребителя е разрешено да прави на компютър или мрежа
  - Въпроси: Какво можете да правите? Можете ли да видите тази страница?

Автентикация срещу Авторизация [2/2]

Authorization

What you can do

Authentication

Who you are

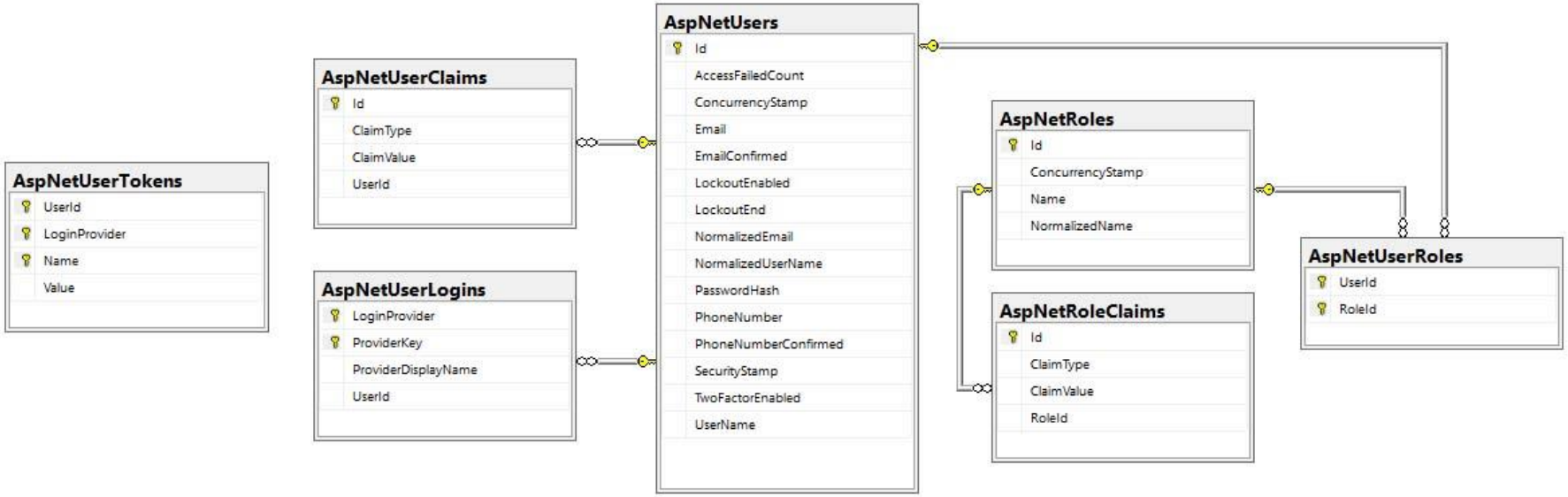
ASP.NET Core Идентичност

ASP.NET Core Идентичност [1/3]

- Системата ASP.NET Core Identity
  - Система за удостоверяване и упълномощаване за ASP.NET Core
  - Поддържа ASP.NET MVC, Pages, Web API (JWT), SignalR
  - Работи с потребители, потребителски профили, влизане / излизане, роли и т.н.
  - Работи със съгласието за бисквитки и GDPR
  - Поддържа външни доставчици за вход
    - Facebook, Google, Twitter и т.н.
  - Поддържа база данни, Azure, Active Directory, потребители на Windows и т.н.

ASP.NET Core Идентичност [2/3]

- Обикновено данните за идентичност на ASP.NET Core се съхраняват в релационна база данни
  - Данните се запазват с помощта на Entity Framework Core
  - Имате известен контрол върху вътрешната схема на базата данни



ASP.NET Core Идентичност [3/3]

- Настройка на идентичността на ASP.NET
  - Използване на шаблоните на ASP.NET за проекти от Visual Studio
    - Последващо персонализиране
- На ръка
  - Инсталиране на NuGet пакети, ръчна конфигурация, създаване на EF карти (модели), преглеждане на модели, контролери, изгледи и т.н.
- Необходим пакет NuGet
  - Microsoft.AspNetCore.Identity.EntityFrameworkCore

Удостоверяване на шаблона на ASP.NET Core Project [1/2]

- ApplicationDbContext.cs
  - Съдържа контекста на данни на EF
  - Осигурява достъп до данните на приложението, използвайки модели на обекти
- Startup.cs
  - Може да конфигурира удостоверяване въз основа на бисквитки (или JWT)
  - Може да активира външно влизане (напр. Вход във Facebook)
  - Може да промени настройките за идентификация по подразбиране
  - Може да активира RoleManager с .AddRoles <IdentityRole> ()

Удостоверяване на шаблона на ASP.NET Core Project [2/2]

- Настройки на паролата - могат да бъдат определени в Startup.cs

```
public void ConfigureServices(IServiceCollection services)
{
  ...
  services.AddDefaultIdentity<IdentityUser>(options =>
      {
          // Password, lockout, emails, etc.
          options.Password.RequireNonAlphanumeric = false;
      })
    .AddDefaultUI(UIFramework.Bootstrap4)
    .AddRoles<IdentityRole>()
    .AddEntityFrameworkStores<ApplicationDbContext>()
}
```

Регистрация на потребител

```
var newUser = new ApplicationUser()
{
    UserName = "maria",
    Email = "mm@gmail.com",
    PhoneNumber = "+359 2 981 981"
};

var result = await userManager.CreateAsync(newUser, "S0m3@Pa$$");

if (result.Succeeded)
    // User registered
else
    // result.Errors holds the error messages
```

Потребителски Вход/Изход

- Вход

```
bool rememberMe = true;
bool shouldLockout = false;
var signInStatus = await signInManager.PasswordSignInAsync(
    "maria", "S0m3@Pa$$", rememberMe, shouldLockout);

if (signInStatus.Succeeded)
    // Sucessfull login
else
    // Login failed
```

- Изход

```
await signInManager.SignOutAsync();
```

ASP.NET Авторизиране

- Използвайте атрибутите [Authorize] и [AllowAnonymous], за да конфигурирате разрешен / анонимен достъп за контролер / действие

```
[Authorize]
public class AccountController : Controller {
  // GET: /Account/Login (anonymous)
  [AllowAnonymous]
  public async Task<IActionResult> Login(string returnUrl) { … }

  // POST: /Account/LogOff (for logged-in users only)
  [HttpPost]
  public async Task<IActionResult> Logout() { … }
}
```

Провери настоящият потребител

```
// GET: /Account/Roles (for logged-in users only)
[Authorize]
public ActionResult Roles()
{
    var currentUser = await userManager.GetUserAsync(this.User);
    var roles = await userManager.GetRolesAsync(currentUser);
    ...
}
```

```
// GET: /Account/Data (for logged-in users only)
[Authorize]
public ActionResult Data()
{
    var currentUserUsername = await userManager.GetUserName(this.User);
    var currentUserId = await userManager.GetUserIdAsync(this.User);
    ...
}
```

Добави потребител в роля

```
var roleName = "Administrator";
var roleExists = await roleManager.RoleExistsAsync(roleName);

if (roleExists)
{
    var user = await userManager.GetUserAsync(User);
    var result = await userManager.AddToRoleAsync(user, roleName);

    if (result.Succeeded)
        // The user is now Administrator
}
```

Изисква влезлия потребител в определена роля

- Достъп само на потребителите в роля “Administrator":

```
[Authorize(Roles="Administrator")]
public class AdminController : Controller
{ … }
```

- Достъп, ако Ролята на потребителя е “User", “Student" или “Trainer":

```
[Authorize(Roles="User, Student, Trainer")]
public ActionResult Roles()
{ … }
```

Проверете ролята на потребителя, в която сте в момента

```
// GET: /Home/Admin (for logged-in admins only)
[Authorize]
public ActionResult Admin()
{
    if (this.User.IsInRole("Administrator"))
    {
        ViewBag.Message = "Welcome to the admin area!";
        return View();
    }

    return this.View("Unauthorized");
}
```

ASP.NET Core User Manager

- UserManager <TUser> - API за управление на потребителите в постоянен магазин

Category		
AddClaimsAsync(…)	FindByEmailAsync(…)	GenerateChangeEmailTokenAsync(…)
AddToRoleAsync(…)	FindByIdAsync(…)	GenerateEmailConfirmationTokenAsync(…)
IsInRoleAsync(…)	FindByNameAsync(…)	GeneratePasswordResetTokenAsync(…)
GetUserId(…)	GetClaimsAsync(…)	GetAuthenticationTokenAsync(…)
ConfirmEmailAsync(…)	GetEmailAsync(…)	IsEmailConfirmedAsync(…)
ChangeEmailAsync(…)	GetRolesAsync(…)	CreateSecurityTokenAsync(…)
CreateAsync(…)	GetUserAsync(…)	ResetPasswordAsync(…)
DeleteAsync(…)	CheckPasswordAsync(…)	RemoveFromRoleAsync(…)
Dispose(…)	UpdateAsync(…)	RemoveClaimsAsync(…)

Claims [1/3]

- Идентичността на базата на претенции е често срещана техника, използвана в приложенията
  - Приложенията придобиват информация за самоличност на своите потребители чрез искове
- Искът е твърдение, което един субект прави за себе си
  - Може да става въпрос за име, група, етническа принадлежност, привилегия, асоциация и т.н.
  - Субектът, който отправя искането, е доставчик
- Идентичността на базата на претенции опростява логиката за удостоверяване
  - Често се използва в отделни части на приложение или микроприложения
  - Не е необходим механизъм за създаване / промяна на акаунта

Claims [2/3]

- В ASP.NET Core проверките за автентичност на базата на претенции са декларативни
  - Програмистът ги вгражда срещу контролер или действие
  - Програмистът посочва необходимите претенции за достъп до функционалността
- Изискванията за искове се основават на политиката
  - Програмистът трябва да регистрира политика, изразяваща изисквания за претенции
- Претенциите са двойки име-стойност

Claims [3/3]

- Най-простият вид политика за искове проверява само за наличието на рекламация
  - Стойността на рекламацията не се проверява

```
public void ConfigureServices(IServiceCollection services) {
    ...
    services.AddAuthorization(options => {
        options.AddPolicy("EmployeeOnly", policy =>
policy.RequireClaim("EmployeeNumber")); });}
```

```
[Authorize(Policy = "EmployeeOnly")]
public IActionResult VacationBalance() {
    //This action is accessible only by Identities with the "EmployeeOnly"
Claim...
    return View(); }
```

Идентичност

Scaffolding ASP.NET Core Identity

- От ASP.NET Core 2.2, идентичността се предоставя като библиотека на клас Razor
- Скелето може да бъде конфигурирано да генерира изходен код
  - Ако трябва да промените кода и да промените поведението
- Повечето от необходимия код се генерира от скелета
  - Вашият проект ще се нуждае от актуализация, преди процесът да приключи
- Скелето генерира полезен файл ScaffoldingReadme.txt
  - Съдържа инструкции за това какво е необходимо за завършване на скелета
- Препоръчва се контрол на източника, преди да се направи опит за скеле

ASP.NET Core Identity

- ApplicationUser.cs - може да добави функционалност на потребителя
- Разширява потребителската информация за приложението ASP.NET Core, получено от IdentityUser
  - Id (уникален идентификационен номер на потребител, низ, съдържащ GUID)
    - Например 313c241a-29ed-4398-b185-9a143bbd03ef
  - Потребителско име (уникално потребителско име), напр. Мария
- Имейл (имейл адрес - може да бъде уникален), напр. mm@gmail.com
- Може да съдържа допълнителни полета, напр. име, фамилия, дата на раждане

Пълен контрол идентичността

Пълен контрол над идентичността [1/2]

- Поведението за идентичност по подразбиране се заменя с такъв

```
public void ConfigureServices(IServiceCollection services)
{
    ...
    services.AddIdentity<IdentityUser, IdentityRole>()
        // services.AddDefaultIdentity<IdentityUser>()
        .AddEntityFrameworkStores<ApplicationDbContext>()
        .AddDefaultTokenProviders();
    ...
}
```

- Внедрявате потребителски и потребителска роля

Пълен контрол над идентичността [1/2]

- Конфигуриране на LoginPath, LogoutPath, AccessDeniedPath

```
public void ConfigureServices(IServiceCollection services)
{
    ...
    services.ConfigureApplicationCookie(options =>
    {
        options.LoginPath = $"/Identity/Account/Login";
        options.LogoutPath = $"/Identity/Account/Logout";
        options.AccessDeniedPath = $"/Identity/Account/AccessDenied";
    });
    ...
}
```

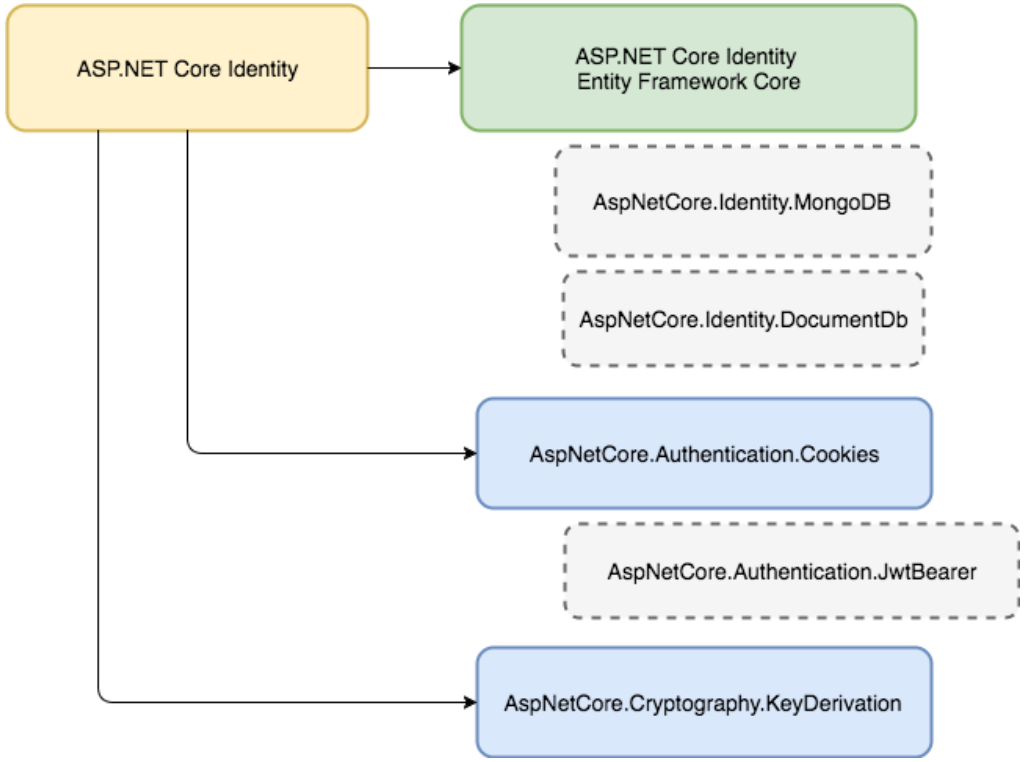
Видове Удостоверяване

Видове за удостоверяване

- Видове удостоверяване в ASP.NET Core приложения:
  - Удостоверяване и упълномощаване на базата на бисквитки (идентичност)
  - Удостоверяване и упълномощаване на Windows
  - Облачно удостоверяване и упълномощаване
  - JSON Уеб токени (JWT) удостоверяване и авторизация

Удостоверяване и упълномощаване въз основа на бисквитки

- Базираната на бисквитки автентикация е механизмът за удостоверяване на приложението ASP.NET Core
- Удостоверяването е изцяло на базата на бисквитки
- Това е основна разлика от ASP.NET MVC
- Главницата се основава на претенции



Удостоверяване и упълномощаване на Windows

- Windows auth е по-сложен механизъм за автентификация
  - Разчита на операционната система за удостоверяване на потребителите
  - Акредитивните данни се хешират, преди да бъдат изпратени през мрежата
  - Най-подходящ за интранет среда
    - Клиенти, потребители, сървъри принадлежат към един и същ домейн на Windows (AD)

Облачно удостоверяване и упълномощаване

- Облачното основание auth е по-модерен подход за удостоверяване
  - Работата по удостоверяване и упълномощаване се възлага на външни изпълнители
  - Външната платформа управлява функционалността на потребителя
  - Гарантира гъвкавост и скорост
  - Стратегически отделя функцията auth от другите

JWT удостоверяване и упълномощаване

- JSON Web Tokens е модерен механизъм за авторство, базиран на JavaScript
  - Компактен и самостоятелен
  - Фокусиран върху подписани символи
    - Работете с претенции
  - Данните са криптирани
  - Използва се за auth & обмен на информация
  - Често използван при разработване на REST
  - Изключително проста за разбиране
  - Използва се в приложения Angular / React / Vue.js / Blazor

Социални Акаунти

Социални Акаунти [1/3]

- Позволяването на потребителите да влизат със съществуващите си идентификационни данни е удобно
  - Прехвърля сложността на управлението на процеса на влизане към трета страна
  - Подобрява потребителското изживяване, като свежда до минимум техните авторски дейности
  - ASP.NET Core поддържа вградени външни доставчици за вход за:
    - Google
    - Facebook
    - Twitter
    - Microsoft

```
public void ConfigureServices(IServiceCollection services) {
    ...
    services.AddAuthentication()
        .AddGoogle(googleOptions => { ... })
        .AddFacebook(facebookOptions => { ... })
        .AddTwitter(twitterOptions => { ... })
        .AddMicrosoftAccount(microsoftOptions => { ... });
    ...
}
```

Социални Акаунти [2/3]

- Всеки доставчик на външно влизане има определен API за разработчици
  - Трябва да конфигурирате приложението си там, преди да го използвате
  - Това приложение ще ви предостави пълномощия
    - Идентификационен номер на приложението
    - Таен ключ
  - Тези идентификационни данни ще бъдат използвани от API на външен доставчик
  - Вие се удостоверявате с тях, когато изпращате заявка
  - Тези идентификационни данни не трябва да се съхраняват публично

Социални Акаунти [3/3]

- Пример: Facebook

```
public void ConfigureServices(IServiceCollection services)
{
    ...
    services.AddAuthentication()
        .AddFacebook(facebookOptions => {
            facebookOptions.AppId =
                  Configuration["Authentication:Facebook:AppId"];
            facebookOptions.AppSecret =
                  Configuration["Authentication:Facebook:AppSecret"];
    });
    ...
}
```

Обобщение

- Основна идентичност на ASP.NET
- Видове за удостоверяване
- Социални акаунти

Национална програма
"Обучение за ИТ умения и кариера"
https://it-kariera.mon.bg

Министерството на
образованието и науката
https://www.mon.bg