

# Biztonság

Albert István

[ialbert@aut.bme.hu](mailto:ialbert@aut.bme.hu)

Q.B. 221, 1662



Automatizálási és  
Alkalmazott  
Informatikai Tanszék

# Tartalom

## Fogalmak

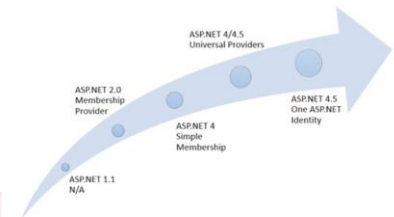
- Azonosítás (authentication)
  - > Hitelesítés – Ki az aki a kérést küldi ?
- Engedélyezés (authorization)
  - > Hozzáférés szabályozás – Milyen adatokat és szolgáltatásokat érhet el ez a személy ?
- Integritás (integrity)
  - > Hogyan biztosítható, hogy az adatokat útközben ne manipulálhassák ?

## Azonosítás, engedélyezés, integritás

	web szerver	ASP.NET	saját kód
Azonosítás	<ul style="list-style-type: none"><li>Anonymous, Basic</li><li>Windows Integrált (Kerberos)</li><li>Client Certificate</li></ul>	<ul style="list-style-type: none"><li>Windows Forms</li><li>OAuth – FB stb.</li></ul>	<ul style="list-style-type: none"><li>pl: DB, AD</li></ul>
Engedélyezés	<ul style="list-style-type: none"><li>NTFS</li><li>Fájlszintű</li></ul>	<ul style="list-style-type: none"><li>Mappe, fájl alapú</li><li>.NET szerep alapú</li></ul>	<ul style="list-style-type: none"><li>deklaratív</li><li>imperatív</li></ul>
Integritás	<ul style="list-style-type: none"><li>SSL</li></ul>	<ul style="list-style-type: none"><li>Data Protection</li></ul>	<ul style="list-style-type: none"><li>Data Protection</li></ul>

## ASP.NET azonosítás történelem

- Sok változat készült már
  - > PL SQL, no-SQL, Azure, OAuth támogatás miatt írták mindig újra



## Engedélyezés

- Szabályozza, hogy az adott erőforráshoz, funkcióhoz ki férhet hozzá?
- Például attribútumokkal szabályozható
  - > Controller (osztály) szinten
  - > Action (metódus) szinten
- AllowAnonymous attribútum
- Authorize attribútum
  - > Csak belépett felhasználók

## UserManager<TUser, TKey>

- A funkciók ebben az osztályban kerültek implementálásra
  - > Beléptetés, kiléptetés, email
- Sok külső szolgáltatást (függőséget) használ
  - > IUserStore<TUser, TKey>
  - > IValidator<TUser> – felhasználó vizsgálat
  - > IPasswordValidator<TUser> – jelszó vizsgálat
  - > .ctor-ban IUserStore, a többi propertyként (IoC)
- Minden aszinkron

## Adat védelem

- Érzékeny adat védelme a megbízhatatlan klientsől
  - > Például autentikációs token
- Célok
  - > **Integritás:** a kliens ne tudja hamisítani.
  - > **Titkosítás:** a kliens ne tudja olvasni.
  - > **Izoláció:** a szerver komponensek egymástól függetlenül tudjanak működni.

# Fogalmak

- Azonosítás (authentication)
  - > Hitelesítés – Ki az aki a kérést küldi ?
- Engedélyezés (authorization)
  - > Hozzáférés szabályozás – Milyen adatokat és szolgáltatásokat érhet el ez a személy ?
- Integritás (integrity)
  - > Hogyan biztosítható, hogy az adatokat útközben ne manipulálhassák ?

# Szerep alapú biztonság a .NET-ben

- Identity interfész
  - > felhasználó neve: `string Name`
  - > azonosítás típusa
- IPrincipal interfész
  - > Szerepkörök: `bool IsInRole( string )` metódus
  - > hivatkozás az Identityre
- Minden szálhoz lekérdezhető/beállítható a felhasználó
  - > **RÉGI .NETBEN:** `Thread . CurrentPrincipal`
  - > Az aszinkron hívás minták ezt a biztonsági környezetet tipikusan átviszik a háttérszálakra is...
  - > Szerver oldalon tipikusan a `HttpContext/...`-en keresztül lehet elérni a kéréshez tartozó felhasználót

# Különböző implementációk

- GenericIdentity, GenericPrincipal osztályok
  - > Naív implementációk, stringek
- WindowsIdentity, WindowsPrincipal osztályok
  - > Operációs rendszer által támogatott implementációk, windowsos/AD-s felhasználó stb.
  - > WindowsIdentity . GetCurrent()
- Az autentikáció és autorizáció keverhető
  - > Windows-os autentikációs (WindowsIdentity)
  - > Saját, .NET-es autorizáció (GenericPrincipal)

# Imperatív ellenőrzés

- Amennyiben nincs elegendő információnk fordítási időben a felhasználóról, programkódból ellenőrizhetjük a feltételeket

```
if( ! ...IsInRole("@<domain>\Customer") )  
    throw new SecurityException( ... );
```

```
if( ! ...IsInRole( "ModifyBankAccount" ) )  
    throw new SecurityException( ... );
```

# Deklaratív ellenőrzés

- A kód biztonságához hasonlóan felhasználói szerepeket is megkövetelhetünk

```
[Authorize(Role=@"ModifyBankAccount")]  
public void DoIt( ... ) {...}
```

- Vagy tágabban, a csak azonosított felhasználókra

```
[Authorize]
```

- Nem azonosított felhasználókra

```
[AllowAnonymous]
```

# „Szerep” alapú biztonság

- Az autorizáció a mindenképp a kódba beégetett ellenőrzésekre épít
  - Előbb-utóbb mindenképp lesznek konstans sztringek, amelyeket a kód deklaratív vagy imperatív módon ellenőriz!
- Ezeket hívhatjuk „elemi jogoknak”, amik be vannak égetve a kódba, nem rugalmasak



# Jogosultsági rendszer rugalmassága

- Az elemi jogok felhasználókhoz rendelése
  - Használhatunk AD-t (org unit) vagy saját adatbázist
- A hozzárendelés összetettsége, ötletek:
  - Felhasználók csoportokba rendezése, a csoportok jelentik az „elemi jogokat”
  - A csoportok hierarchiát képeznek
  - Az elemi jogok csoportokhoz rendelése dinamikus
  - Helyettesítés kezelése, auditálási kérdések

# Azonosítás, engedélyezés, integritás

	web szerver	ASP.NET	saját kód
Azonosítás	<ul style="list-style-type: none"><li>• Anonymous, Basic</li><li>• Windows Integrált (Kerberos)</li><li>• Client Certificate</li></ul>	<ul style="list-style-type: none"><li>• Windows</li><li>• Forms</li><li>• OAuth – FB stb.</li></ul>	<ul style="list-style-type: none"><li>• pl: DB, AD</li></ul>
Engedélyezés	<ul style="list-style-type: none"><li>• NTFS</li><li>• Fájlszintű</li></ul>	<ul style="list-style-type: none"><li>• Mappa, fájl alapú</li><li>• .NET szerep alapú</li></ul>	<ul style="list-style-type: none"><li>• deklaratív</li><li>• Imperatív</li></ul>
Integritás	<ul style="list-style-type: none"><li>• SSL</li></ul>	Data Protection	Data Protection

# Azonosítás a webserveren

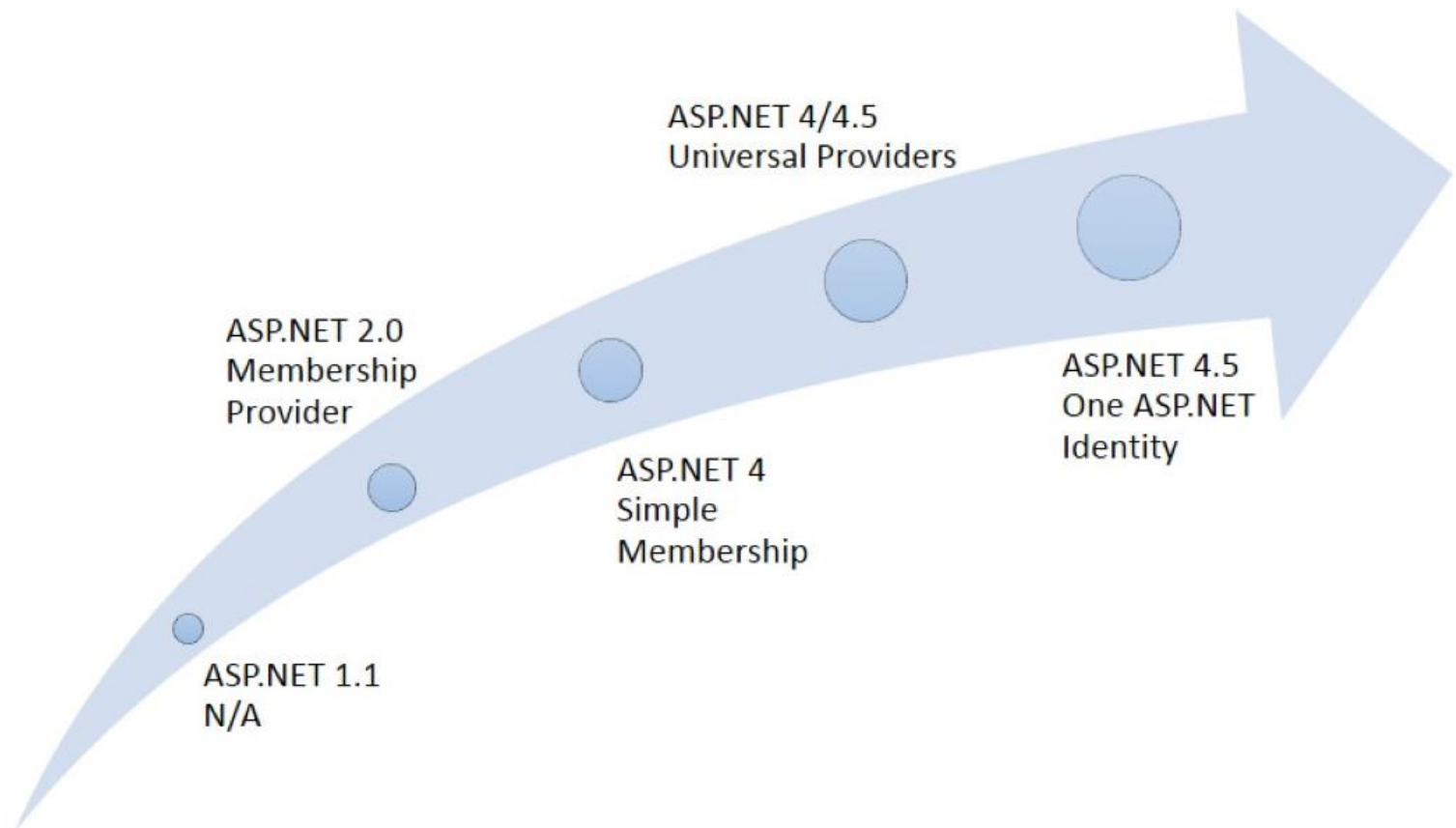
- Anonymous
  - > A *webserver* nem azonosít
  - > Forms, Közösségi, OAuth, ...
- Basic
  - > A webserver domainjében azonosít
  - > CLEAR TEXT felhasználónév és jelszó, SSL-lel már jó
- **Windows Integrated** authentication (automatikus)
  - > Single-Sign-On domainben lévő kliensen (böngészőfüggő)
  - > **Kerberos**: gyors, skálázható, proxyn keresztül is működik
  - > NTLM: proxyval nem megy, nem kell AD
- Client Certificate
  - > Nagyobb adminisztrációs költségek, lassabb, nagyon biztonságos, rugalmas, integritás védelem, kölcsönösség

# Engedélyezés a webszerveren

- IP cím ellenőrzése
  - > Csak szerver OS-en, de IPSec mindenütt van!
- IIS Web jogok ellenőrzése
  - > Nem felhasználófüggő (read, write, ...)
- NTFS jogok ellenőrzése
  - > Mappa és fájl alapú ellenőrzés az azonosított felhasználó alapján
  - > Csak azonosított windowsos felhasználók esetén!

# ASP.NET azonosítás történelem

- Sok változat készült már
  - > Pl. SQL, no-SQL, Azure, OAuth támogatás miatt írták mindig újra



# ASP.NET Core Identity - célok 1.

- Egységes beléptetés a különböző webes keretrendszerhez
  - > MVC, WebForms, WebPages, **Web API**, SignalR, ...
- Egyszerűen bővíthető felhasználói adatok
  - > Például születési adatok, stb.
- Beépített megoldás relációs adatbázishoz:
  - > EF, Code First: egyszerűen módosítható adatbázis
  - > Bővíthető, más jellegű tár bevezetése egyszerű
- Tesztelhető, tesztekben egyszerűen felhasználható

# ASP.NET Core Identity - célok 2.

- Egyszerű szerepkezelés
- Claim-based autentikáció támogatás
  - > Rugalmasabb, mint a szerep alapú
- OAuth támogatás: Facebook, Google, Live, ...
- [Azure] Active Directory támogatás
- OWIN integráció: kompatibilis a moduláris webes middleware-rel a monolitikus System.Web helyett
- NuGet csomag támogatás: egyszerűbb integráció, gyorsabb fejlesztési ciklusok

# ASP.NET Core Identity

- Klasszikus felhasználónév + jelszó megoldás
- Külső szolgáltatás használata
  - > Facebook, Google, Microsoft, ...
- Háttér tár
  - > Microsoft SQL Server
  - > Vagy bármi más...



# DI alapú – szolgáltatás regisztrálás

```
// This method gets called by the runtime. Use this method to
// register the services.
0 references
public void ConfigureServices(IServiceCollection services)
{
    // Add framework services.
    services.AddDbContext<ApplicationDbContext>(options =>
        options.UseSqlServer(Configuration.GetConnectionString("DefaultConnection")));

    services.AddIdentity<ApplicationUser, IdentityRole>()
        .AddEntityFrameworkStores<ApplicationDbContext>()
        .AddDefaultTokenProviders();
}
```

# Be kell állítani

```
services.Configure<IdentityOptions>(options =>
{
    // Password settings
    options.Password.RequireDigit = true;
    options.Password.RequiredLength = 2;
    options.Password.RequireNonAlphanumeric = false;
    options.Password.RequireUppercase = true;
    options.Password.RequireLowercase = false;

    // Lockout settings
    options.Lockout.DefaultLockoutTimeSpan = TimeSpan.FromMinutes(30);
    options.Lockout.MaxFailedAccessAttempts = 10;

    // Cookie settings
    options.Cookies.ApplicationCookie.ExpireTimeSpan = TimeSpan.FromDays(150);
    options.Cookies.ApplicationCookie.LoginPath = "/Account/LogIn";
    options.Cookies.ApplicationCookie.LogoutPath = "/Account/LogOff";

    // User settings
    options.User.RequireUniqueEmail = true;
});
```

# Middleware komponens a pipeline-ban


// This method gets called by the runtime. Use this method to confi

0 references | Albert István, 364 days ago | 1 author, 1 change

```
public void Configure(IApplicationBuilder app, IHostingEnvironment  
{  
    loggerFactory.AddConsole(Configuration.GetSection("Logging"));  
    loggerFactory.AddDebug();
```

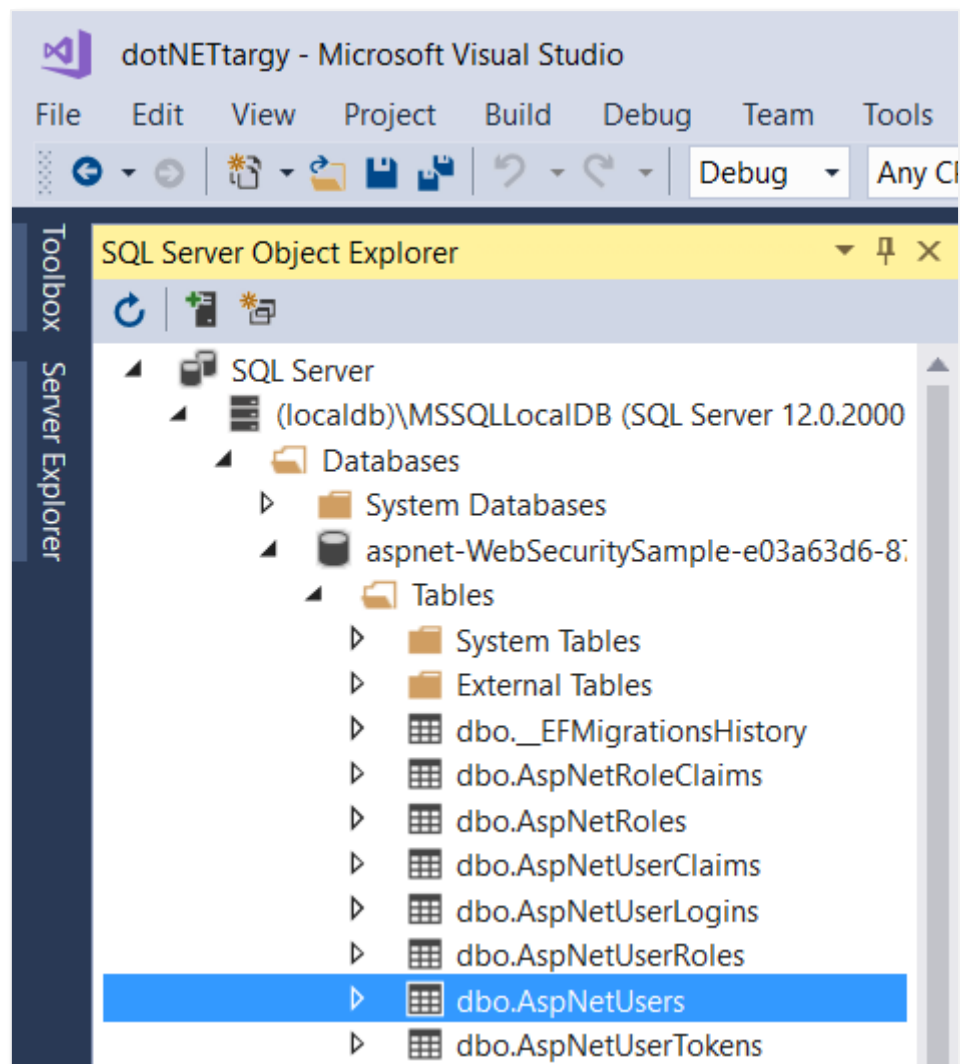
```
    app.UseIdentity();
```

```
if (er  
{
```

 (extension) `IApplicationBuilder UseIdentity()`  
Enables ASP.NET identity for the current application.

```
    app.UseDeveloperExceptionPage();  
    app.UseDatabaseErrorPage();
```

# Létrehozza az adatbázist



# Featureset

- Saját jelszó erősség, logout házirend
- Két-faktoros belépés (SMS, QR kód stb)
- Fiók megerősítés
- Fiók kizárás
- Kilépés mindenhol (Security Stamp)
- Elfelejtett jelszó kezelés
- Felhasználó ellenőrzés
- ...

# Tetszőleges provider használható

- „Beépítve”: NuGet csomagban az Microsofttól
  - > Facebook
  - > Twitter
  - > Google
  - > Microsoft
- Egyéb, mások által karban tartott csomagok:
  - > LinkedIn
  - > Instagram
  - > Reddit
  - > Github, Yahoo, Tumblr, Pinterest, Pocket, Flickr, Vimeo, Cribble, SoundCloud, VK

# Windows autentikáció

- A host (IIS, IISExpress, WebListener, ...) végzi, azt kell konfigurálni, például:

```
{  
  "iisSettings": {  
    "windowsAuthentication": true,  
    "anonymousAuthentication": false,  
    "iisExpress": {  
      "applicationUrl": "http://localhost:  
      "sslPort": 0  
    }  
  }  
} // additional options trimmed  
}
```

WindowsAuthSample

Application Configuration: N/A Platform: N/A

Build

Build Events

Package

Debug

Signing

Resources

Launch: IIS Express

Application arguments: Arguments to be passed to the application

Working directory: Absolute path to working directory Browse...

☒ Launch URL: Absolute or relative URL

Environment variables:

Name	Value
ASPNETCORE_ENVIRONMENT	Development

Add Remove

Web Server Settings

App URL: http://localhost:52171/

☐ Enable SSL

☐ Enable Anonymous Authentication

☒ Enable Windows Authentication

# Windows autentikáció az alkalmazásban

- Ha a host eldobja a kérést, mindegy mit állítunk az alkalmazásban!
- Az Authorize és AllowAnonymous attribútumoknak csak akkor van hatásuk, ha a névtelen kérést is átengedjük a hoston
- Ehhez speciális konfiguráció kell, például:

```
// IISDefaults requires the following import:  
// using Microsoft.AspNetCore.Server.IISIntegration;  
services.AddAuthentication(IISDefaults.AuthenticationScheme);
```



# Megszemélyesítés

- Alapértelmezetten nincs, minden szál a process nevében fut!
  - > Ritkán kell a windowsos felhasználó nevében elvégezni műveletet, például fájl hozzáférés stb.
- `HttpContext.Current.User.Identity`: ez egy `WindowsIdentity`
- `WindowsIdentity.RunImpersonated`
  - > token: a `WindowsIdentity` objektum access tokenje
  - > delegate: **a felhasználó nevében végrehajtandó kód**

# Engedélyezés

- Szabályozza, hogy az adott erőforráshoz, funkcióhoz ki férhet hozzá?
- Például attribútumokkal szabályozható
  - > Controller (osztály) szinten
  - > Action (metódus) szinten
- AllowAnonymous attribútum
- Authorize attribútum
  - > Csak belépett felhasználók

# Szerep szintű engedélyezés

- Authorize attribútum
  - > Roles paraméter
  - > VAGY kapcsolat: a roles paraméterben felsorolva vesszővel
  - > ÉS kapcsolat: több Authorize attribútum
  - > Controller és action szinten

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

# Házirend (policy) alapú megoldás

- Plusz egy indirekció bevezetése
- Házirend definiálása

```
public void ConfigureServices(IServiceCollection services)
{
    services.AddAuthorization(options =>
    {
        options.AddPolicy("RequireAdministratorRole",
            policy => policy.RequireRole("Administrator"));
    });
}
```

- Házirend hivatkozása

```
[Authorize(Policy = "RequireAdministratorRole")]
0 references
public IActionResult Shutdown()
{
    return View();
}
```

# Egyéb autorizációs lehetőségek

- Claim-based engedélyezés
  - > Az azonosító szolgáltató adatokat (claimek) rendel a felhasználóhoz
  - > Ezeket a claimeket lehet használni engedélyezéskor
  - > Például: a felhasználó 18 évnél idősebb
- Saját autorizációs logika
- Mindkettő a házirendként jelenik meg
- Imperatív ellenőrzés kódból

# Autorizációs szolgáltatás

- Deklaratív logika nem elég, mert szükség van a kérés egyéb paramétereire
  - > Például a felhasználó melyik tárgy eredményeit akarja szerkeszteni
- Saját `IAuthorizationService` implementáció
  - > `AuthorizationHandler` alaposztály segítségével
- A kapott paraméterek alapján eldönti, hogy végrehajtható-e a művelet
- Csak egy sémát ad a kód alapú ellenőrzésre és segít összegyűjteni a validációs logikát egy helyre

# UserManager<TUser, TKey>

- A funkciók ebben az osztályban kerültek implementálásra
  - > Beléptetés, kiléptetés, email
- Sok külső szolgáltatást (függőséget) használ
  - > IUserStore<TUser, TKey>
  - > IUserValidator<TUser> – felhasználó vizsgálat
  - > IPasswordValidator<TUser> – jelszó vizsgálat
  - > .ctor-ban IUserStore, a többi propertyként (IoC)
- Minden aszinkron

# IUserStore<TUser, TKey>

- Minimális funkciók
  - > Létrehozás, törlés, frissítés, keresés (név/id)
- Az átadott UserStore objektum más I...Store interfészeket is implementálhat
  - > IUserLoginStore: user – login összekapcsolás, pl Facebook stb.
  - > IRoleStore: szerepek tárolása
  - > IUserRoleStore: felhasználók szerepekhez rendelése
  - > IUserPasswordStore: jelszó hash-ek tárolása
- A UserManager osztály ezekre az implementációkra épít (SupportsUserRole, ...)



# IdentityResult: általános eredmény

- bool Succeeded: sikeres hívás vagy sem
- IEnumerable<string> Errors: hibák listája
- A metódusok általában Task<IdentityResult>-ot adnak vissza

# Bővíthető felhasználói adatok

```
// Add profile data for application users by
13 references
public class ApplicationUser : IdentityUser
{
    0 references
    public string FullName { get; set; }
    0 references
    public DateTime BirthDate { get; set; }
}
```

```
// Add framework services.
services.AddDbContext<ApplicationDbContext>(options =>
    options.UseSqlServer(Configuration.GetConnectionString("DefaultConnection")));

services.AddIdentity<ApplicationUser, IdentityRole>()
    .AddEntityFrameworkStores<ApplicationDbContext>()
    .AddDefaultTokenProviders();
```

```
public class ApplicationDbContext
    : IdentityDbContext<ApplicationUser>
{
    0 references
    public ApplicationDbContext(
        DbContextOptions<ApplicationDbContext> options)
        : base(options)
    { }
}
```

```
...public class IdentityDbContext<TUser> : IdentityDbContext<TUser, IdentityRole, string> where TUser : IdentityUser
{
    ...public IdentityDbContext(DbContextOptions options);
    ...protected IdentityDbContext();
}
```

# Adat védelem

- Érzékeny adat védelme a megbízhatatlan kientől
  - > Például autentikációs token
- Célok
  - > **Integritás:** a kliens ne tudja hamisítani.
  - > **Titkosítás:** a kliens ne tudja olvasni.
  - > **Izoláció:** a szerver komponensek egymástól függetlenül tudjanak működni.

# Egyszerű példa

```
IDataProtector protector;
```

1 reference

```
public void RunSample()
{
    Console.WriteLine("Enter input: ");
    string input = Console.ReadLine();

    // protect the payload
    string protectedPayload = protector.Protect(input);
    Console.WriteLine($"Protect returned: {protectedPayload}");

    // unprotect the payload
    string unprotectedPayload = protector.Unprotect(protectedPayload);
    Console.WriteLine($"Unprotect returned: {unprotectedPayload}");
}
```

```
C:\WINDOWS\system32\cmd.exe
Enter input: alibaba
Protect returned: CfDJ8PLFcYspzOtJth9
h-5WLI8BlGZaxrBlyZ6NFmimnWqP7_Ii6KptV
vmGRiLZoBLom1hhDxNpBIb9akoC6Sgur_obHi
fht2iw9YBydoSEIKQumXoBd-cHxcMnob1ZlQP
BrGw
Unprotect returned: alibaba
Press any key to continue . . .
```

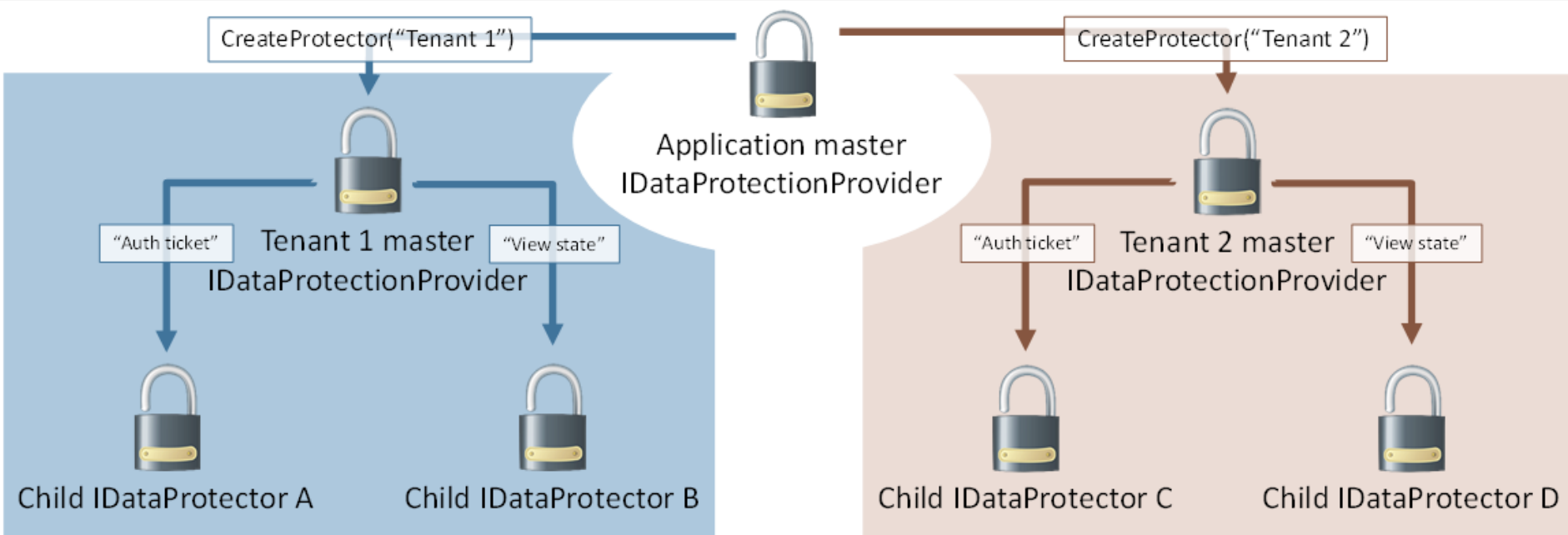
# Izoláció

- Minden komponens saját kulcsokat kap
  - > A CreateProtector metódus paramétere adja meg a komponens nevét, ez alapján válnak szét a kulcsok
  - > A különböző névvel rendelkező komponensek nem látják, olvashatják egymás titkosított adatait
  - > A név nem titkos, egyszerűen az egyediséget biztosítja

```
// the 'provider' parameter is provided by DI
0 references
public MyClass(IDataProtectionProvider provider)
{
    protector = provider.CreateProtector("BME.AUT.dotNET.Security");
}
```

# Tenant szintű izoláció

- Egy DataProtector további DataProtectorokat hozhat létre további nevekkel
  - > A szolgáltatás komponensek hierarchiába fűzhetők
  - > Teljes és felülről vezérelt az izoláció



# Konfiguráció

- A felhasználás módja-kódja nem változik
- A szolgáltatások konfigurálhatók
  - > „Encrypt at rest”
  - > Tanusítvány, DPAPI, ...

```
serviceCollection.AddDataProtection()  
    .PersistKeysToFileSystem( new DirectoryInfo(@"c:\temp\") )  
    .ProtectKeysWithDpapiNG("CERTIFICATE=HashId:3BCE558E2AD3E0E34A7743  
        flags: DpapiNGProtectionDescriptorFlags.None)  
    .DisableAutomaticKeyGeneration()  
    .SetDefaultKeyLifetime(TimeSpan.FromDays(14))  
    .UseCryptographicAlgorithms(new AuthenticatedEncryptionSettings()  
    {  
        EncryptionAlgorithm = EncryptionAlgorithm.AES_256_CBC,  
        ValidationAlgorithm = ValidationAlgorithm.HMACSHA256  
    });
```

# Kérdések?

Albert István  
[ialbert@aut.bme.hu](mailto:ialbert@aut.bme.hu)