



Eötvös Loránd Tudományegyetem
Informatikai Kar

Webes alkalmazások fejlesztése

9. előadás

Webszolgáltatások hitelesítése (ASP.NET Core)

Cserép Máté

mcserep@inf.elte.hu

<https://mcserep.web.elte.hu>

Webszolgáltatások hitelesítése

Autentikáció és autorizáció szükségessége

- A webszolgáltatások alkalmazprogramozási interfészét (API) is megfelelő védelemmel kell ellátni
 - *autentikáció*: a végpontok elérésekor szükséges lehet ellenőrizni a felhasználó azonosságát (*identity*)
 - *autorizáció*: a végpontok elérésekor szükséges lehet ellenőrizni, hogy a felhasználó jogosult-e a művelet végrehajtására
- Amennyiben az API nincsen publikusan kiajánlva (pl. OpenAPI), a végpontok akkor is ismerté válhatnak
- Nem számíthatunk arra, hogy a felhasználó csak a megadott klienseket használja, tetszőleges HTTP kérést küldhet bármikor

Webszolgáltatások hitelesítése

Adatok titkosítása memóriában

- A teljes biztonsághoz a memóriában lévő kényes tartalmat is titkosítani kell, mivel az is potenciális támadási felület
 - lehetőség szerint csak a feldolgozás időtartamára szerepeljen titkosítatlan információ a memóriában, egyébként kódolva tároljuk
- Szövegek titkosított kezelésére szolgál a **SecureString** típus, amely alapvetően titkosítva tárolja a jelszót, és csak lekéréskor (**ToString**) dekódolja
 - a grafikus felületen a jelszavak titkosított bekérését a **PasswordBox** biztosítja, a **Password** tulajdonság feloldja a titkosítást (amely nem függőségi tulajdonság, így nem köthető)

Webszolgáltatások hitelesítése

Adatok titkosítása memóriában

- Pl.:

```
<Button Content="Bejelentkezés"
        Command="{Binding LoginCommand}"
        CommandParameter="{Binding
                        ElementName=passwordBox}" .../>
<!-- magát a jelszóbekérőt adjuk át -->

...
LoginCommand = new DelegateCommand(param =>
{
    _model.Login(UserName,
                  (param as PasswordBox).Password);
    // kiolvassuk a titkosított jelszót
    ...
})
```

Feladat: Valósítsuk meg az utazási ügynökség épületeit karbantartó asztali alkalmazást.

- adjunk lehetőséget képek megtekintésére, hozzáadására, törlésére
 - a képet fájlból töltjük be, majd átméretezzük (kis és nagy méretben, PNG formátumban)
 - a képeket egyedi azonosítóval látjuk el, valamint az épület azonosítójával
- a biztonság növelésére az adatkezelést autentikációhoz kötjük (*ASP.NET Core Identity* segítségével), így a felhasználónak előbb be kell jelentkezniük az alkalmazásba

Webszolgáltatások hitelesítése

Példa

Tervezés:

- létrehozunk egy vezérlőt (**BuildingImageController**), valamint egy adatátviteli típust (**ImageDTO**) a képkezeléshez
- a képeket alapvetően byte tömbként kezeljük, a szolgáltatás nem is ismeri azok képi tartalmát
- a képbetöltést egy segédtípusban (**ImageHandler**) végezzük
- a képek megjelenítéséhez átalakítást végzünk (**BuildingImageConverter**), ami **BitmapImage** típusra alakítja a tömböt, ezeket **Image** vezérlővel jelenítjük meg
- külön nézetbe szervezzük az épület adatainak megadását (**BuldingEditorWindow**)

Webszolgáltatások hitelesítése

Példa

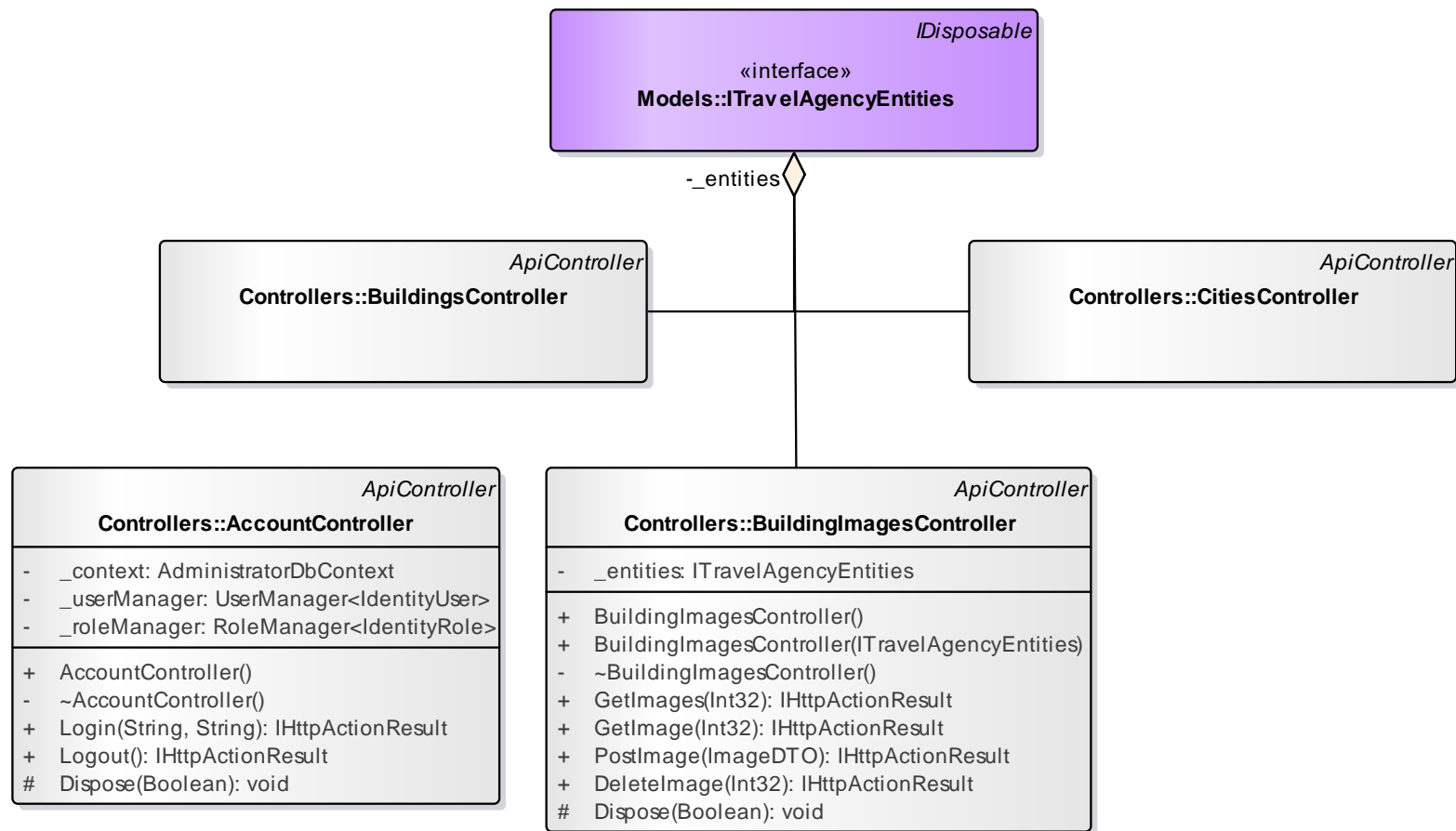
Tervezés:

- létrehozunk egy vezérlőt a felhasználókezeléshez (**AccountController**), ebben lehetőséget adunk bejelentkezésre (**Login**) és kijelentkezésre (**Logout**)
- a szolgáltatásban attribútum (**Authorize**) segítségével korlátozzuk az akciófüggvényekhez való hozzáférést (csak a rendszergazda csoportban lévő felhasználókra)
- kliens oldalon megjelenik a két új művelet a modellben (**LoginAsync**, **LogoutAsync**)
- a bejelentkezéshez egy külön nézetet (**LoginWindow**), valamint nézetmodellt (**LoginViewModel**) hozunk létre

Webszolgáltatások hitelesítése

Példa

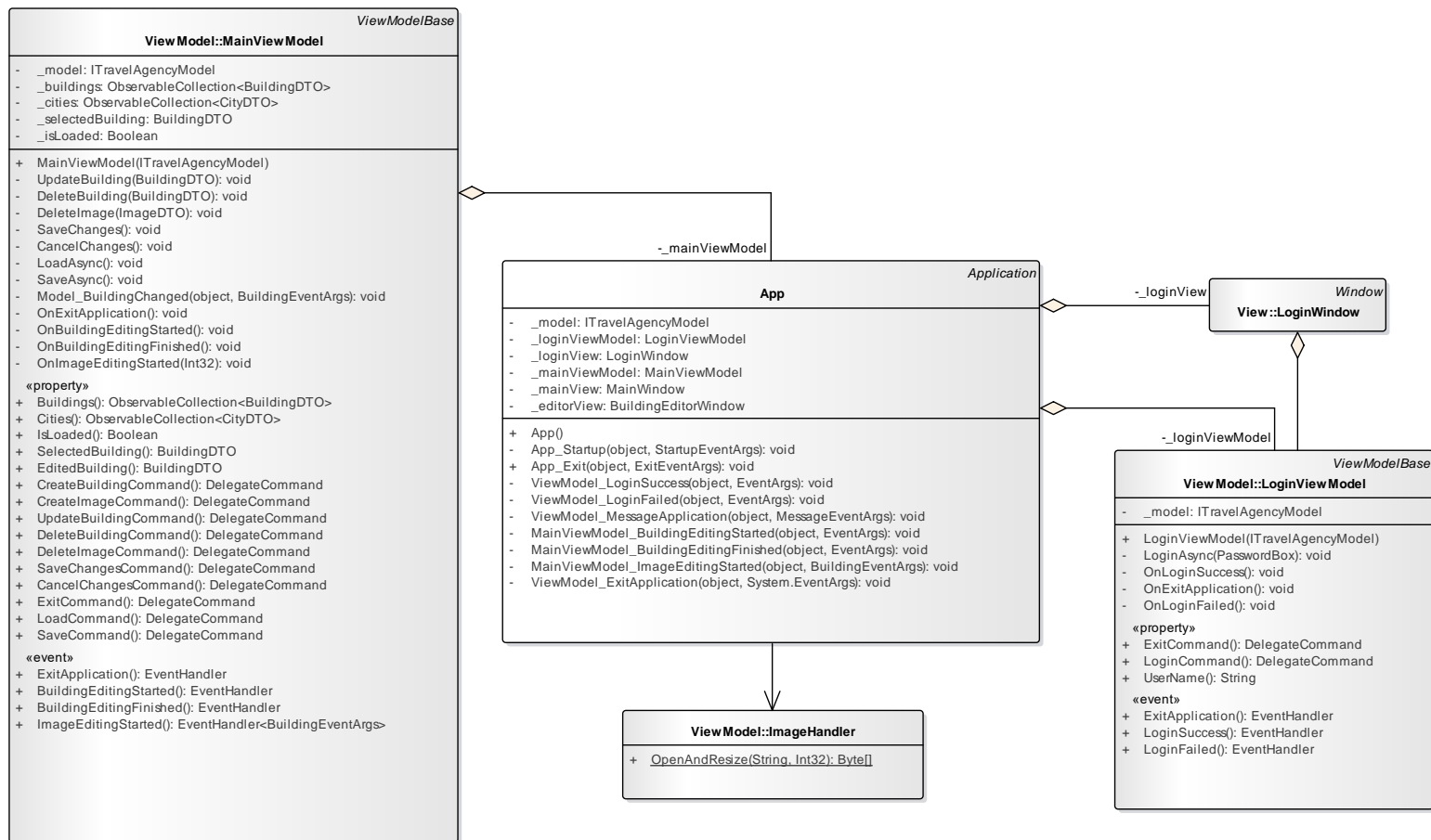
Tervezés (szolgáltatás):



Webszolgáltatások hitelesítése

Példa

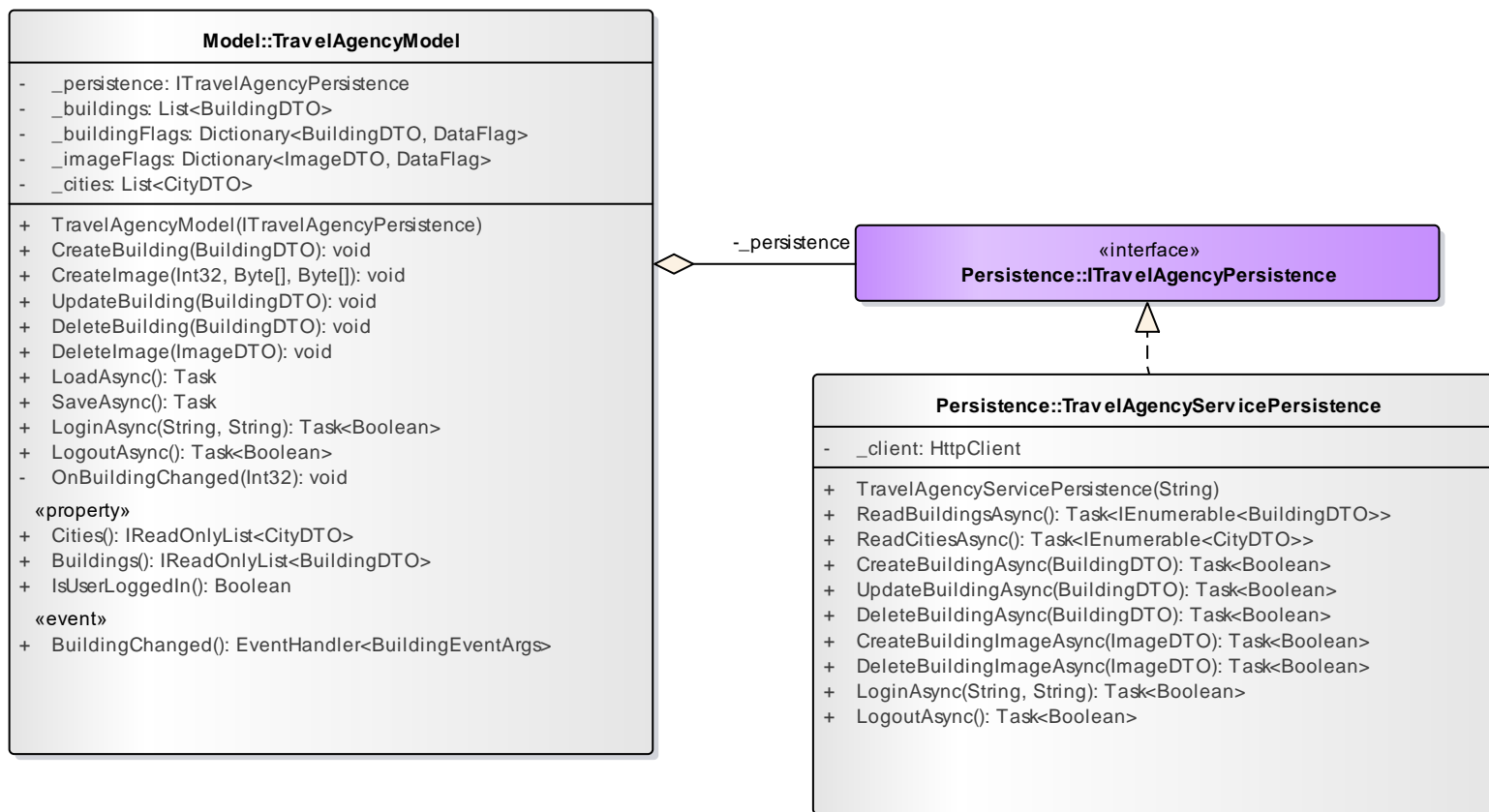
Tervezés (kliens):



Webszolgáltatások hitelesítése

Példa

Tervezés (kliens):



Webszolgáltatások hitelesítése

Példa

Megvalósítás (BuildingImagesController.cs):

```
[Authorize(Roles = "administrator")]
    // csak bejelentkezett adminisztrátoroknak
public IActionResult PostImage([FromBody]
    ImageDTO image)
{
    BuildingImage buildingImage = new ...
    ...
    _context.SaveChanges();
    return CreatedAtAction(nameof(GetImage),
        new { id = buildingImage.Id },
        buildingImage.Id);
    // csak az azonosítót küldjük vissza
    ...
}
```

Webszolgáltatások hitelesítése

Példa

Megvalósítás (ImageHandler.cs):

```
public static Byte[] OpenAndResize(String path,
                                     Int32 height)
{
    BitmapImage image = new BitmapImage();
    // kép betöltése
    image.BeginInit();
    image.UriSource = new Uri(path);
    image.DecodePixelHeight = height;
    // megadott méretre
    image.EndInit();
}
```

Webszolgáltatások hitelesítése

Példa

Megvalósítás (ImageHandler.cs):

```
PngBitmapEncoder encoder =  
    new PngBitmapEncoder();  
    // átalakítás PNG formátumra  
encoder.Frames.Add(BitmapFrame.Create(image));  
  
using (MemoryStream stream =  
    new MemoryStream())  
    // átalakítás byte-tömbre  
{  
    encoder.Save(stream);  
    return stream.ToArray();  
}  
}
```

Webszolgáltatások hitelesítése

Hitelesítési módszerek

- A webes alkalmazások életciklusa miatt két kérés között nem őrzik meg az állapotot. Minden kérés felhasználó által történő közvetlen hitelesítése (pl. felhasználónév-jelszó párossal) körülményes, automatizált megoldásra van szükség.
- Hozzáférési token (*access token*) alapú hitelesítés
 - a sikeres felhasználónév-jelszó alapú autentikáció után a kliensnek egy hozzáférési tokent adunk át, amelyet perzisztálunk az alkalmazásban is
 - a további kérésekhez a kliens elküldi a korábban kapott hozzáférési tokent
 - a biztonság kritikus eleme a titkosított csatorna (HTTPS)

Webszolgáltatások hitelesítése

Hozzáférési tokenek

- A hozzáférési token átadása a kliens és a szerver között különböző módokon történhet
 - alkalmazások közötti kommunikáció esetén
 - GET/POST paraméterben, vagy
 - HTTP fejlécben (konvencionálisan **Authorization**)
 - emberi felhasználóval böngészőn keresztüli kommunikáció esetén jellemzően sütiben
 - az *ASP.NET Core Identity* erre alapértelmezetten a **.AspNetCore.Identity.Application** sütit használja
 - az alkalmazásban a szükséges információt az **AspNetUsers** tábla **SecurityStamp** mezője tárolja

Webszolgáltatások hitelesítése

Hozzáférési tokenek

- A hozzáférési token jellemzően adott ideig érvényes csak.
 - Lejárata lehet csúszóablakos (*sliding expiration*), azaz minden sikeres kérés után határozott ideig meghosszabbításra kerül
 - Tipikusan ilyet használunk süti alapú autentikációkor
 - Alkalmazások közötti kommunikációhoz jellemzően rögzített élettartamú hozzáférési tokenet használunk
 - Bejelentkezéskor a hozzáférési token (*access token*) mellett egy frissítő token (*refresh token*) is kapunk
 - Új hozzáférési a frissítő token használatával kérhető
 - A hozzáférési token sűrűn (*pl. 30 perc*), a frissítő token ritkábban jár le (*pl. 30 nap*), vagy egyáltalán nem

Webszolgáltatások hitelesítése

Hozzáférési tokenek

- A hozzáférési tokenek két fajtáját különböztethetjük meg:
 - átlátszatlan (*opaque*) tokenek, ahol a token egy, a szerveren tárolt erőforrást hivatkozik, de nem hordoz önálló tartalmat
 - áttetsző (*transparent*) tokenek, ahol a token kódolt tartalmat hordoz, így nem csak autentikációra, hanem autorizációra, vagy általánosan bármilyen információ hordozására is alkalmas
 - a token digitálisan aláírható, így ellenőrizhető, hogy tartalma a szerver általi kiállítás óta nem módosult-e
 - a token akár titkosítható is, ha fontos, hogy a tartalma ne legyen könnyen olvasható
 - ilyen specifikáció a *JSON Web Tokens* (JWT) szabvány

Webszolgáltatások hitelesítése

JSON Web Tokens (JWT)

- A *JSON Web Tokens* (JWT) alkalmazások közötti biztonságos információ átadására szolgál
 - Sikeres felhasználónév-jelszó alapú autentikáció után a szerver előállít egy JWT-t, amely a felhasználó azonossága mellett egyéb információkat (pl. hozzáférési jogosultságok) is tartalmaz, ezeket *claim*-eknek nevezzük. A JWT-t a szerver digitális aláírással látja el és átadja a kliensnek.
 - A kliens a JWT-t olvashatja (amennyiben nem titkosított), valamint a szerver felé a további kérésekhez mint egy hozzáférési tokenet visszaküldi.
 - A szerver ellenőrzi, hogy a kapott JWT tartalma a digitális aláírásnak megfelel-e, nem került-e módosításra (*tampering*). Ha rendben van, akkor a tartalma hiteles, így nem szükséges pl. a jogosultsági körök külön betöltése.

Webszolgáltatások hitelesítése

JSON Web Tokens (JWT)

- A JWT token JSON formátumú és három részből áll:
 - *Header*: metainformációk a tokenről
 - *Payload*: a token hordozott tartalma
 - *Signature*: a token digitális aláírása
- A JWT token három része karakterkódolási problémák elkerülése végett [base64](#) kódolásra kerül és pontokkal elválasztva kerül elküldésre (így akár GET paraméter is lehet)
 - formátum: xxxxx.yyyyyy.zzzzz, pl.:

eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpc3MiOiJodHRwczovL2VsdGUuaHUvIiwic3ViIjoiaMTIzNDU2Nzg5IiwibmFtZSI6IiRlc3p0IEVsZWsiLCJhZG1pbSI6dHJlZX0.U56N4FGI21Dw2GVAACzv7PRxTck7jvsszN23GKHn4ak

Webszolgáltatások hitelesítése

JSON Web Tokens (JWT)

- A JWT *header* adja meg a digitális aláírás algoritmusát (HMAC SHA256, RSA, ECDSA), és a token típusát (JWT).

```
{  
  "alg": "HS256",  
  "typ": "JWT"  
}
```
- A JWT *payload* tartalmazza a hordozott információt, pl.:

```
{  
  "iss": "https://elte.hu/",  
  "sub": "123456789",  
  "name": "Teszt Elek",  
  "admin": true  
}
```

Webszolgáltatások hitelesítése

JSON Web Tokens (JWT)

- A *payload* tartalmazza a JWT *claim*-eket, 3 kategória szerint:
 - Vannak a JWT specifikációban előre definiált, ún. regisztrált claim-ek, mint pl. **iss** (*issuer*), **exp** (*expiration time*) vagy a **sub** (*subject*). A regisztrált claim-ek mind 3 karakteres névvel rendelkeznek a tömörség végett.
 - Vannak konvencionálisan használt, publikus claim-ek, mint pl. a **name**, az **email** vagy a **birthdate**.
 - Definiálhatunk tetszőleges *privát claim*-et is, ilyen volt az **admin** az előző példában.

Webszolgáltatások hitelesítése

JSON Web Tokens (JWT)

- A JWT *signature* a *header* és a *payload* digitális aláírása, base64 enkódolva.
 - HMAC SHA256 algoritmus használatakor egy titkos kulcs (*secret*) használatával kerül egy *hash* érték előállításra, pszeudo-kóddal:

```
signature = HMACSHA256(  
    base64(header) + "." + base64(payload) ,  
    secret)
```
- A kulcsot mi adjuk meg (konfigurációs érték) és csak a digitális aláírást kiállító szerver ismerheti.

Webszolgáltatások hitelesítése

JWT használata ASP.NET Core keretrendszerben

- Az *ASP.NET Core* integráltan támogatja a JWT alapú autentikációt és autorizációt
 - az **Authorize** attribútumnak megadhatjuk, hogy használjon JWT-t az alapértelmezett *Identity* alapú autorizáció helyett (`JwtBearerDefaults.AuthenticationScheme`)
- A **Program** osztály **WebApplicationBuilder** objektumán ezt az új alapértelmezésnek is konfigurálhatjuk:

```
builder.Services.AddAuthentication(options => {  
    options.DefaultAuthenticateScheme =  
        JwtBearerDefaults.AuthenticationScheme;  
    options.DefaultChallengeScheme =  
        JwtBearerDefaults.AuthenticationScheme;  
});
```

Webszolgáltatások hitelesítése

JWT használata ASP.NET Core keretrendszerben

- Ugyanitt engedélyezhetjük a JWT alapú hitelesítést (`AddJwtBearer`) és konfigurálhatjuk a token validálását:

```
builder.Services.AddAuthentication(...)  
    .AddJwtBearer(options => {  
        options.TokenValidationParameters =  
            new TokenValidationParameters  
            {  
                ValidIssuer = "https://elte.hu",  
                // kiállító ellenőrzése  
                IssuerSigningKey =  
                    new SymmetricSecurityKey(  
                        Encoding.UTF8.GetBytes("secret key"))  
                // digitális aláírás ellenőrzése  
            };  
    });
```


Webszolgáltatások hitelesítése

JWT előállítás

- Sikeres felhasználónév-jelszó párossal történő hitelesítés után generálhatunk egy JWT-t a kliensnek:

```
var tokenDesc = new SecurityTokenDescriptor
{
    Subject = new ClaimsIdentity(new[] {
        new Claim(ClaimTypes.Name, user.UserName)
    }),
    // JWT-ben rögzítendő claim-ek
    Issuer = "https://elte.hu", // kiállító
    Expires = DateTime.Now.AddMinutes(30), // lejárat
    SigningCredentials = new SigningCredentials(
        new SymmetricSecurityKey(
            Encoding.UTF8.GetBytes("secret key")),
        SecurityAlgorithms.HmacSha256)
    // digitális aláírás beállítása
};
```

Webszolgáltatások hitelesítése

JWT előállítás

- A JWT autorizációs adatokat, így a felhasználó szerepkörét is tartalmazhatja, ezeket is adjuk hozzá:

```
var roles = await userManager.GetRolesAsync(user);  
foreach (var role in roles)  
    tokenDesc.Subject.AddClaim(  
        new Claim(ClaimTypes.Role, role));
```

- Állítsuk elő a JWT sztringet:

```
var handler = new JwtSecurityTokenHandler();  
var jwtToken = handler.CreateToken(tokenDesc);  
string tokenStr = handler.WriteToken(jwtToken);
```

- A JWT tokent a kliensnek visszaadhatjuk a válasz törzsében vagy akár HTTP fejlécben

```
Response.Headers.Add("Bearer", tokenStr);
```

Webszolgáltatások hitelesítése

JWT token átadása a szervertől a kliensnek

- A kliens oldalon a JWT token kiolvasható a válasz HTTP üzenet fejlécéből:

```
using (HttpClient client = new HttpClient())
{
    HttpResponseMessage response =
        await client.GetAsync("api/account/login");
    if (response.IsSuccessStatusCode)
    {
        token = response.Headers.GetValues("Bearer")
            .FirstOrDefault();
        // JWT token
    }
}
```

Webszolgáltatások hitelesítése

JWT token átadása a kienstől a szervernek

- A kliens a további kérésekhez a JWT token az *Authorization* fejléchez adja a *Bearer* prefixszel (pl. **Bearer eyJhb...n4ak**), itt az ASP.NET Core keresni fogja.

```
using (HttpClient client = new HttpClient())
{
    client.DefaultRequestHeaders.Authorization =
        new AuthenticationHeaderValue("Bearer",
            token); // JWT token hozzáadása az
                  // Authorization HTTP fejléchez

    HttpResponseMessage response =
        await client.PostAsJsonAsync("...", <object>);
    // ... válasz feldolgozása
}
```

Webszolgáltatások hitelesítése

Kliens alkalmazások

- A webszolgáltatások a szerver-oldali backend alkalmazás általános körű felhasználását teszik lehetővé.
- Tetszőleges platformú kliens alkalmazás nyújthat kliens oldali felületet a webszolgáltatáshoz.
 - Asztali alkalmazás, mobil alkalmazás, weboldal
 - Más webszolgáltatás is felhasználhatja
- A backend webszolgáltatás és a frontend kliensek eltérő platformokon és programozási nyelveken, eltérő fejlesztő csapatok által, párhuzamosan is fejleszthetők.

Webszolgáltatások hitelesítése

Külső kliens alkalmazások engedélyezése

- A JWT alapú autentikációval és autorizációval szerver oldali szolgáltatásunkhoz tetszőleges kliens alkalmazás kapcsolódhat, hogy a felhasználók felhatalmazásával műveleteket végezzen
 - pl. a Spotify fiókunk a Facebook hírfalunkra posztoljon
 - Az autentikációs és autorizációs protokollt külön specifikáció szabályozza (pl. OAuth2, OpenId), a JWT a tokenek formátuma lehet. (OpenId esetén kötelezően.)
 - A protokoll írja le a tokenek fajtáját, tartalmát, kibocsátásának és visszavonásának rendjét, stb.
 - Léteznek alternatív megoldások is, pl. a SAML (*Security Assertion Markup Language*) egyben a protokollt és saját, XML alapú token formátumát is definiálja.
-

Webszolgáltatások hitelesítése

Példa

Feladat: Valósítsuk meg az utazási ügynökség épületeit karbantartó asztali alkalmazást.

- a süti alapú hitelesítés helyett használjunk hozzáférési tokeneket a felhasználók autentikálására és autorizálására
- a webszolgáltatás az asztali alkalmazásnak a sikeres bejelentkezés után egy rövid lejáratú JWT tokenet adjon át, amely hordozza a felhasználó azonosságát és jogköreit
- a kliens kapjon egy átlátszatlan, nem lejáró frissítő tokenet is, amellyel megújíthatja a hozzáférési JWT tokenét
- a megoldást integráljuk a felhasználókezelés meglévő *ASP.NET Core Identity* alapú implementációjával

Webszolgáltatások hitelesítése

Példa

Tervezés:

- a webszolgáltatásban JWT generálást a **JwtService** osztály végzi, amely megvalósítja az **IJwtService** interfészt
- az **AccountController** vezérlő **Login** akciója előállítja a hozzáférési JWT tokenet, és a frissítő tokenel visszaadja
 - a JWT megújítását a **RefreshTokens** akció végzi, a frissítő tokenet a **Guest** entitás tárolja perzisztensen
- a **Program** osztályban regisztráljuk a JWT alapú autentikációt és autorizációt, mint alapértelmezettet
 - az **appsettings.json** fájlban megadjuk a konfiguráció dinamikusan változtatható paramétereit

Webszolgáltatások hitelesítése

Példa

Tervezés:

- a kliens oldalon a **TravelAgencyServicePersistence** osztályban minden kéréshez hozzáadjuk a JWT-t az *Authorization* HTTP fejlécben
 - a JWT-t és a frissítő tokenet a bejelentkezéskor kapjuk vissza a webszolgáltatástól
- sikertelen kérés esetén megújítjuk a hozzáférési JWT-t a frissítő token segítségével
 - a folyamatot segédeljárásba szervezzük ki (**SendRequest**)