# Refresh JWT Token with ASP.NET (Core) (C#)

## Che cosa sono i Refresh Token

In breve, puoi pensare ai token di aggiornamento come:

Un token che può essere utilizzato per ricevere un nuovo token di accesso (nel nostro caso, token JWT) senza dover inviare nuovamente le credenziali.

Quindi, come possiamo implementarne uno?

Prima di ciò, tieni presente che è necessario apportare le seguenti modifiche al tuo endpoint di accesso esistente (per me è in / api / Auth / login):

1. Modificare l'oggetto di ritorno nell'endpoint di accesso in modo da restituire quanto segue:

{  
"accessToken": "<JWT Token>",  
"accessTokenExpiration": "<A value that tells up to when is the access token valid."  
"refreshToken": "<Refresh Token>"  
}

"AccessToken" - Questo è fondamentalmente il tuo token JWT.

"AccessTokenExpiration" - Questo è facoltativo. Ma questo rappresenta un valore che indica al client fino a quando il token di accesso è valido. Sta a te decidere se vuoi restituire un double per indicare i minuti. Di solito uso DateTimeOffset per questo.

"RefreshToken" - Qui è dove si posizionerà il token di aggiornamento che il client può utilizzare per ricevere un nuovo token JWT.

1. Modifica la modalità di generazione del token JWT assicurando che un identificatore per l'utente sia incluso nel token JWT tramite claims o tramite un oggetto:

private List<Claim> CreateClaims(string id)

{

var claims = new List<Claim>();

claims.Add(new Claim(ClaimTypes.Name, string id));

return claims;

}

public string GenerateToken(string appSecret, IEnumerable<Claim> claims, double expirationInMinutes)

{

var tokenHandler = new JwtSecurityTokenHandler();

var key = Encoding.ASCII.GetBytes(appSecret);

var claimsDictionary = new Dictionary<string, object>();

foreach (var claim in claims)

{

claimsDictionary.Add(claim.Type, claim.Value);

}

var tokenDescriptor = new SecurityTokenDescriptor

{

Subject = new ClaimsIdentity(claims),

Claims = claimsDictionary,

Expires = DateTime.UtcNow.AddMinutes(expirationInMinutes),

SigningCredentials =

new SigningCredentials(

new SymmetricSecurityKey(key),

SecurityAlgorithms.HmacSha256Signature)

};

var token = tokenHandler.CreateToken(tokenDescriptor);

return tokenHandler.WriteToken(token);

Si noti che è necessario aggiungere un reclamo che è possibile utilizzare per identificare facilmente l'utente in base al token JWT. Di solito uso Claim ClaimTypes.Name per questo.

1. Nel flusso di lavoro di accesso, assicurarsi che "Aggiorna flusso token durante l'accesso" e "Utilizzo di un token di aggiornamento" (vedere di seguito).

## Refresh Token durante il login

Questo flusso può essere posizionato prima o dopo la generazione di un token JWT.

Genera un token casuale:

public string GenerateRefreshToken()

{

var randomNumber = new byte[32];

using (var rng = RandomNumberGenerator.Create())

{

rng.GetBytes(randomNumber);

return Convert.ToBase64String(randomNumber);

}

}

L'idea alla base del valore del token è che deve essere il più casuale possibile ed è qualcosa che non può essere facilmente indovinato.

È possibile utilizzare la propria implementazione per generare il valore casuale, ma per esempio sopra ho usato System.Security.Cryptography.RandomNumberGenerator.

Alcuni suggeriscono che è possibile utilizzare GUID per questo, ma altri sostengono che il GUID abbia uno schema.

Alla fine, spetta a te decidere cosa usare.

2. Memorizza il tuo token di aggiornamento insieme alla scadenza nel tuo database / repository.

Questo è un passo molto importante. Ricorda che i nostri token di aggiornamento sono davvero casuali ed è davvero difficile da indovinare?

Bene, dobbiamo trovare un modo per convalidare se il token di aggiornamento fornito è valido o no. Quindi come lo facciamo? Conserva i token di aggiornamento da qualche parte.

Ecco una panoramica di tutto il flusso:

public RefreshToken GenerateRefreshToken(User user)

{

// Create the refresh token

RefreshToken refreshToken = new RefreshToken()

{

Token = GenerateRefreshToken(),

Expiration = DateTime.UtcNow.AddMinutes(35) // Make this configurable

}

// Add it to the list of of refresh tokens for the user

user.RefreshTokens.Add(refreshToken);

// Update the user along with the new refresh token

UserRepository.Update(user);

return refreshToken;

}

public string GenerateRefreshToken()

{

var randomNumber = new byte[32];

using (var rng = RandomNumberGenerator.Create())

{

rng.GetBytes(randomNumber);

return Convert.ToBase64String(randomNumber);

}

}

Ti starai chiedendo perché sto usando un "elenco di token di aggiornamento".

Il motivo di ciò è che se la tua app consente a un utente di accedere a più dispositivi, ognuno di questi avrà il proprio token di aggiornamento.

## Usare il refresh token

Ora che abbiamo dato un token di aggiornamento al client, la domanda successiva sarebbe come usarli nel modo giusto?

Crea un nuovo endpoint che consenta AnonymousAccess:

[AllowAnonymous]

[HttpPost]

[Route("refresh")]

public async Task<ActionResult<LoginResponse>>Refresh([FromBody]RefreshRequest request)

{

var loginResponse = AuthorizationService.Refresh(request.accessToken, refreshToken);

LoginResponse response = new LoginResponse();

response.AccessToken = loginResponse.AccessToken;

response.AccessTokenExpiration = loginResponse.AccessTokenExpiration;

response.RefreshToken = loginResponse.RefreshToken;

return response;

}

public class RefreshRequest

{

public string AccessToken { get; set; }

public string RefreshToken { get; set; }

}

public class LoginResponse

{

public string AccessToken { get; set; }

public DateTimeOffset AccessTokenExpiration { get; set; }

public string RefreshToken { get; set; }

}

Riprendere da <https://medium.com/@kedren.villena/refresh-jwt-token-with-asp-net-core-c-25c2c9ee984b>