Asp.TotpKit – TOTP (Time-based One-Time Password) Doğrulama Kütüphanesi

İçindekiler

1. Amaç

Asp.TotpKit, ASP.NET geliştiricileri için tasarlanmış, **Google Authenticator uyumlu** zaman tabanlı tek seferlik şifre (TOTP) doğrulama kütüphanesidir.
Projenizde güvenli iki faktörlü kimlik doğrulama (2FA) sistemini sıfırdan yazmadan, **üretim ortamına hazır** şekilde entegre etmenizi sağlar.

Bu kütüphane, sürekli tekrar eden şu işlemleri tek bir yerde toplar:

- QR kodu oluşturma,
- Secret key üretimi,
- Kod doğrulama (30 saniyelik aralıklarla),
- Cookie veya kullanıcı bazlı geçici oturum yönetimi,
- ASP.NET Identity veya custom kullanıcı modelleriyle uyum.

2. Özellikler

- RFC 6238 standardına uygun TOTP üretimi
- Google Authenticator ve Authy desteği
- 30 saniyelik otomatik döngü kodlar
- Toleranslı doğrulama (±30 saniye fark)
- Opsiyonel cookie tabanlı geçici oturum
- AppUser (Identity) veya custom model desteği
- IOptions<T> yapılandırması ile DI uyumlu
- Yalnızca QRCoder & .NET bağımlılığı
- Tam loglama ve hata yönetimi (ILogger)
- Email ve ID gibi kullanıcı bilgilerini opsiyonel bağlama

3. Mimari Yapı

Asp.TotpKit, 3 ana bileşenden oluşur:

1. **Interface** → ITotpService<TUser> ve ITotpUserInfo
    - Tüm metod imzalarını içerir (Generate, Validate, Verify, Disable...)
2. **Model** → TotpSetupResponse, TotpVerifyResult, TotpOptions, TotpInstructions
    - Veri transferi ve yapılandırma modelleri
3. **Service** → TotpService<TUser>
    - Gerçek implementasyon. HMAC tabanlı TOTP üretimi, QR code oluşturma, cookie işlemleri ve kullanıcı güncellemelerini yönetir.

4. Kurulum

Asp.TotpKit, ASP.NET Core projelerine kolayca entegre edilebilecek şekilde tasarlanmıştır. Kurulum süreci, Identity yapısı kullanılsa da kullanılmasa da basittir.
Aşağıdaki adımlar, hem AppUser (IdentityUser) tabanlı hem de **custom kullanıcı modeli** ile kullanım senaryolarını kapsar.

4.1. NuGet Paketini Ekle

```
dotnet add package Asp.TotpKit
```

4.2. Program.cs Üzerinden Servis Kaydı

Kütüphaneyi **Dependency Injection (DI)** konteynerine kaydetmek için AddTotpKit<TUser>() metodunu kullanın.

```
builder.Services.AddTotpKit<AppUser>(options =>
{
    options.AppName = "MyProject";
    options.Issuer = "MyProject.Auth";
    options.GetUserEmail = u => u.Email!;
    options.GetUserId = u => u.Id!;
});
```

Program.cs Entegrasyonu (Manual Scoped Registration):

```
builder.Services.AddScoped<ITotpService<AppUser>, TotpService<AppUser>>(sp =>
{
    var service = new TotpService<AppUser>(
        sp.GetRequiredService<UserManager<AppUser>>(),
        sp.GetRequiredService<IConfiguration>(),
        sp.GetRequiredService<ILogger<TotpService<AppUser>>>(),
        sp.GetRequiredService<IOptions<TotpOptions<AppUser>>>()
    );

    // 🔹 Local ortam (Swagger, Postman) için varsayılan cookie ayarları
    service.httpOnlyCookie = false;
    service.secureCookie = false;
    service.sameSiteMode = SameSiteMode.Lax;

    // 🔹 Production ortamı için önerilen güvenli ayarlar:
    // service.httpOnlyCookie = true;
    // service.secureCookie = true;
    // service.sameSiteMode = SameSiteMode.Lax;

    return service;
});
```

Custom model kullanıyorsanız:

```
builder.Services.AddTotpKit<MyUser>(options =>
{
    options.AppName = "MyProject";
    options.Issuer = "MyCompany.Security";
    options.GetUserEmail = u => u.EmailAddress!;
    options.GetUserId = u => u.Uuid.ToString();
});
```

Not: AppUser veya MyUser sınıfınızın ITotpUserInfo arayüzünü uygulaması gerekir.
Bu, TOTP'ye ait alanların (Secret, Enabled, VerifiedDate vb.) saklanabilmesini sağlar.

4.3. AppUser / Custom User Model Tanımı

Asp.TotpKit, kullanıcıya özel TOTP bilgilerini (secret, doğrulama zamanı vb.) kaydetmek için bir arayüz kullanır:
ITotpUserInfo.

Bu arayüzü uygulayan her sınıf, TOTP işlemlerini destekler.
Varsayılan olarak ASP.NET Identity kullanıyorsanız AppUser sınıfına aşağıdaki alanları eklemeniz yeterlidir:

```
public class AppUser : IdentityUser, ITotpUserInfo
{
    public string? ToptSecret { get; set; }                // Base32 Secret Key
    public bool IsToptEnabled { get; set; }                // TOTP aktif mi
    public DateTime? ToptVerifiedDate { get; set; }        // Son doğrulama tarihi
    public bool IsTotpResetRequested { get; set; }         // Reset isteği
    public DateTime? TotpResetRequestedAt { get; set; }    // Reset zamanı
}
```

Eğer **Identity** kullanmıyor ve kendi kullanıcı tablonuzu yönetiyorsanız,
aynı arayüzü kendi modelinize entegre edebilirsiniz:

```
public class MyUser : ITotpUserInfo
{
    public Guid Uuid { get; set; }
    public string EmailAddress { get; set; } = "";
    public string? ToptSecret { get; set; }
    public bool IsToptEnabled { get; set; }
    public DateTime? ToptVerifiedDate { get; set; }
    public bool IsTotpResetRequested { get; set; }
    public DateTime? TotpResetRequestedAt { get; set; }
}
```

Bu sayede kütüphane, ister Identity kullansın ister kendi ORM modelinizi,
kullanıcı nesnesini tanır ve TotpService otomatik olarak o yapıyı kullanır.

4.4. IConfiguration / appsettings.json Kullanımı

Asp.TotpKit, IOptions<T> model bağlamasını desteklediği için konfigürasyonları Program.cs yerine appsettings.json üzerinden de alabilir.

Bu sayede projeniz production'a çıktığında, ortam bazlı ayarları değiştirmek çok daha kolay olur.

Örnek yapılandırma:

```
"AppSettings": {
  "AppName": "MyEnterpriseApp",
  "TotpIssuer": "MyEnterprise.Security"
}
```

Eğer Program.cs içinde bu alanlar belirtilmemişse:

```
options.AppName = ...
options.Issuer = ...
```

kütüphane bu değerleri otomatik olarak IConfiguration içinden okur.

Bu da, *"birçok projede ortak konfigürasyon paylaşmak"* isteyen ekipler için büyük kolaylıktır.
Örneğin staging, production ve local ortamlar için ayrı appsettings dosyaları kullanabilirsiniz.

4.5. Cookie Davranışı (Canlı ve Local Ortam Farkı)

Asp.TotpKit, giriş sonrası kullanıcı kimliği doğrulama sürecinde **geçici cookie** kullanmanıza izin verir.
Bu, özellikle iki aşamalı kimlik doğrulama (login → TOTP kodu) akışlarında kullanılır.

Varsayılan davranış:

```
_totpService.httpOnlyCookie = false;
_totpService.secureCookie = false;
_totpService.sameSiteMode = SameSiteMode.Lax;
```

Bu ayarlar, **local geliştirme ortamı (Swagger/Postman)** için uygundur.
Çünkü HTTPS zorlaması ve HttpOnly kısıtlaması devre dışıdır.

Production ortamı:

Canlı sunucularda aşağıdaki gibi yapılandırmanız gerekir:

```
_totpService.httpOnlyCookie = true;
_totpService.secureCookie = true;
_totpService.sameSiteMode = SameSiteMode.Lax;
```

HttpOnly → cookie'ye sadece sunucu erişebilir, XSS riskini azaltır.
Secure → cookie yalnızca HTTPS bağlantısında gönderilir.
SameSite → CSRF koruması sağlar. "Lax" önerilen moddur.

Eğer frontend ve backend farklı domainlerde çalışıyorsa (CORS senaryosu),
SameSiteMode.None ayarlayabilirsiniz.
Bu durumda Secure = true zorunludur (tarayıcı kısıtlaması).

4.7.Setup Flow

Adım	Açıklama
1	Asp.TotpKit paketini yükle (dotnet add package)
2	AppUser ya da MyUser modeline ITotpUserInfo uygula
3	Program.cs içine AddTotpKit<TUser>() servisini ekle
4	(Opsiyonel) appsettings.json içine AppName & Issuer yaz
5	(Opsiyonel) Cookie ayarlarını yapılandır
6	ITotpService<TUser>'i constructor'a inject edip kullan

5. Metot Referansı

Metot	Açıklama	Dönüş Tipi
GenerateSecretKeyAsync(user)	Kullanıcı için yeni Base32 TOTP secret key üretir ve kaydeder.	Task<string>
GenerateQrCodeUriAsync(user, secretKey)	Google Authenticator ile uyumlu QR kod URI'si oluşturur.	Task<string>

GenerateQrCodeImageAsync(qrCodeUri)	QR kodu PNG formatında byte[] olarak üretir.	Task<byte[]>
ValidateTotpCodeAsync(user, code)	Girilen 6 haneli kodun geçerli olup olmadığını kontrol eder.	Task<bool>
ValidateLoginCodeAsync(user, code)	Login sırasında TOTP kodunu doğrular.	Task<bool>
GenerateSetupAsync(user)	Kullanıcıya özel secret ve QR bilgilerini döner.	Task<TotpSetupResponse>
VerifyCodeAsync(user, code)	İlk kurulum sonrası TOTP'yi aktif eder.	Task<TotpVerifyResult>
EnableTotpAsync(user, code)	Doğrulanmış kullanıcı için TOTP'yi etkinleştirir.	Task<bool>
DisableTotpAsync(user)	TOTP'yi kapatır, secret'ı temizler.	Task<bool>
HasTotpSetup(user)	Kullanıcının TOTP aktifliği kontrol edilir.	bool
SetTotpTempCookie(response, userId, expiryMinutes)	Login sürecinde geçici cookie oluşturur.	void
GetTotpTempCookie(request)	Cookie içinden kullanıcı ID'sini döner.	string
ClearTotpTempCookie(response)	Cookie'yi siler.	void

6. Sonuç ve Kapanış

Bu kütüphaneyi, her projede aynı TOTP doğrulama yapısını baştan kurmaktan sıkıldığım için geliştirdim.
Amacım, **herkesin kolayca anlayabileceği kadar sade**, ama **profesyonel düzeyde güvenli** bir 2FA sistemi oluşturmaktı.

Asp.TotpKit, yalnızca bir "kod doğrulama servisi" değil, aynı zamanda tekrar etmeyen, temiz ve genişletilebilir bir güvenlik mimarisidir.

Kendi projelerinde kullan, test et, incele.
Eksik, hatalı veya geliştirilebilir bir yön görürsen her zaman geri bildirim gönder.
Benim için bu proje sadece bir kütüphane değil — **yeniden kullanılabilir güvenliğin kültürüdür.**