# MerX Code Projesi Dökümantasyonu

## Proje Hakkında Genel Bilgi

MerX Code, C# ve WPF kullanılarak geliştirilen bir gerçek zamanlı sözdizimi vurgulayıcı uygulamasıdır. Bu proje, kullanıcıların kod yazarken belirteç türlerini anında farklı renklerle görebileceği bir platform sunar. Örneğin, bir if anahtar kelimesi yazıldığında bu kelime hemen Açık mavi (koyu tema) veya Mor (açık tema) renkte görünür. Uygulama, iki farklı tema seçeneği (açık ve koyu tema) ile gelir ve AvalonEdit adında bir metin editör kontrolü kullanır. Ancak, vurgulama mantığı tamamen proje boyunca tasarlanmış kodlarla gerçekleştirilir; AvalonEdit'in hazır vurgulama özellikleri kullanılmaz.

## Dil ve Dilbilgisi Seçimi

Proje için C# programlama dili tercih edilir, çünkü hem öğrenmesi kolaydır hem de WPF ile grafik arayüz geliştirmek için uygundur. .NET Framework, Windows üzerinde uyumluluk sağlar. Dilbilgisi olarak, C tarzı bir dilin basitleştirilmiş bir versiyonu benimsenir. Bu dilbilgisi, if, while, return gibi anahtar kelimeleri, değişken tanımlamalarını (int, string, void), atama işlemlerini ve operatörlerle ifadeleri destekler. Örneğin, şu şekilde ifadeler yazılabilir:

```
int x = 10;
if (x > 5) {
    return x;
}
```

Bu dilbilgisi, ayrıştırıcıda tanımlı kurallarla işlenir.

## Sözdizimi Analizi Süreci

Sözdizimi analizi, leksikal analiz ve sözdizimi analizi olmak üzere iki aşamadan oluşur.

### Leksikal Analiz

Leksikal analiz, giriş metnini belirteçlere ayırmak için Lexer sınıfını kullanır. Lexer, metni karakter karakter tarar ve her bir parçayı bir belirteç olarak tanımlar. Desteklenen belirteç türleri şunlardır:

- **Keyword**: if, while, int, return gibi anahtar kelimeler.
- **Identifier**: Değişken adları, örneğin x veya myVariable.
- **Number**: Sayılar, örneğin 123 veya 45.67.

- **String**: Çift tırnak içindeki metinler, örneğin "hello".
- **Operator**: +, ==, <= gibi operatörler.
- **Whitespace**: Boşluklar ve satır sonları.
- **Comment**: // veya /* */ ile yapılan yorumlar.
- **Unknown**: Tanınmayan karakterler.

Lexer, bir durum makinesi gibi çalışır. Örneğin, bir rakamla karşılaştığında ReadNumber metodunu çağırır ve sayıyı tamamen okuyana kadar devam eder. Bir harf gördüğünde ise ReadIdentifierOrKeyword ile bunun bir anahtar kelime mi yoksa değişken adı mı olduğunu kontrol eder.

## Sözdizimi Analizi

Sözdizimi analizi, Parser sınıfıyla gerçekleştirilir. Bu sınıf, leksikal analizden gelen belirteçleri alır ve üstten-aşağı (top-down) bir yaklaşım ile dilbilgisi kurallarına göre kontrol eder. Recursive bir şekilde ilerleyen ayrıştırıcı, örneğin bir if ifadesini ayrıştırırken önce if anahtar kelimesini, sonra parantez içindeki ifadeyi ve ardından bloğu doğrular. Dilbilgisi kuralları şöyledir:

- Statement → if (Expression) BlockOrStatement [ else BlockOrStatement ]
- Statement → while (Expression) BlockOrStatement
- Statement → return [Expression] ;
- Statement → ( int | string | void ) Identifier [ = Expression ] ;
- Statement → Identifier = Expression ;
- Expression → Term { ( + | - | == | != | > | < | >= | <= ) Term }
- Term → Factor { ( * | / ) Factor }
- Factor → Number | Identifier | String | ( Expression )
- BlockOrStatement → { { Statement } } | Statement

Sözdizimi doğruysa true, değilse false döndürülür.

## Vurgulama Sistemi

Vurgulama, uygulamanın temel özelliğidir ve gerçek zamanlı çalışır. Kullanıcı metni her değiştirdiğinde Editor_TextChanged olayı tetiklenir. Bu olayda, metin leksikal analize tabi tutulur ve her belirteç uygun renkte vurgulanır. Bunun için HighlightingColorizer adında özel bir sınıf kullanılır. Bu sınıf, AvalonEdit'in metin satırlarını renklendirmesini sağlar. Belirteç türleri için belirlenen renkler şunlardır:

- **Keyword**: Koyu temada cyan, açık temada dark magenta.
- **Identifier**: Koyu temada beyaz, açık temada siyah.
- **Number**: Koyu temada sarı, açık temada koyu yeşil.
- **String**: Koyu temada turuncu, açık temada çikolata rengi.
- **Operator**: Koyu temada magenta, açık temada koyu mor.
- **Comment**: Koyu temada yeşil, açık temada gri.
- **Unknown**: Her iki temada kırmızı.

Tema yönetimi, ThemeManager sınıfıyla gerçekleştirilir. Kullanıcı, arayüzdeki düğmelerle temayı değiştirebilir.

## Grafik Arayüz (GUI) Tasarımı

Arayüz, WPF ve AvalonEdit kullanılarak tasarlanır. AvalonEdit, yalnızca metin editörü olarak işlev görür; vurgulama mantığı tamamen kendi kodlarımla yapılır. Arayüzün bileşenleri şunlardır:

- **Metin Editörü**: Kod yazımı için AvalonEdit'in TextEditor kontrolü kullanılır. Satır numaralarını otomatik gösterir ve Consolas yazı tipiyle kod yazma deneyimini zenginleştirir.
- **Tema Düğmeleri**: Üstte bir StackPanel içinde "Açık Tema" ve "Karanlık Tema" adlı iki düğme bulunur. Bu düğmelerle tema geçişi sağlanır.
- **Simge ve Tasarım**: Pencereye özel bir simge (logo.ico) eklenir ve pencere boyutu 800x450 olarak sabitlenir.

Arayüz, sade ve kullanıcı dostu bir tasarıma sahiptir. Kullanıcı, kod yazarken belirteçlerin anında renklendiğini fark eder.

## Kod Yapısı

Proje, aşağıdaki temel dosyalardan oluşur:

- **App.xaml ve App.xaml.cs**: Uygulamanın başlangıç noktasıdır. Temel ayarlar burada yapılır.
- **MainWindow.xaml ve MainWindow.xaml.cs**: Ana pencereyi ve arayüzü tanımlar. Tüm olaylar (metin değişimi, tema değişimi) burada işlenir.
- **SyntaxHighlighter.cs**: Leksikal analiz, sözdizimi analizi ve tema yönetimi için Lexer, Parser, Token, ve ThemeManager sınıflarını içerir.

Bu dosyalar, projenin farklı yönlerini ele alır ve uyumlu bir şekilde çalışır.

## Sonuç

MerX Code, kod yazımını görsel ve anlaşılır hale getirmeyi amaçlayan bir projedir. Gerçek zamanlı sözdizimi vurgulama, leksikal ve sözdizimi analizi gibi özelliklerle kullanıcılara pratik bir deneyim sunar. WPF ve AvalonEdit ile tasarlanan grafik arayüz, sade ve işlevsel bir yapıya sahiptir. Uygulama, farklı tema seçenekleri ve en az yedi belirteç türünün vurgulanmasıyla dikkat çeker. Gelecekte, hata mesajları gösterme veya daha geniş bir dilbilgisi desteği eklenmesi gibi geliştirmeler hedeflenir.