

# Regex Sınıf

Başvuru



## Tanım

Ad Alanı: [System.Text.RegularExpressions](#)

Bütünleştirilmiş Kod: System.Text.RegularExpressions.dll

Sabit bir normal ifadeyi temsil eder.

### Bu makalede

[Tanım](#)

[Örnekler](#)

[Açıklamalar](#)

[Oluşturucular](#)

[Alanlar](#)

[Özellikler](#)


[Yöntemler](#)

[Belirtik Arabirim Kullanımları](#)

[Şunlara uygulanır](#)

[İş Parçacığı Güvenliği](#)

[Ayrıca bkz.](#)

C#	 Kopyala
<pre>public class Regex : System.Runtime.Serialization.ISerializable</pre>	

Devralma [Object](#) → [Regex](#)

Türetilmiş [System.Web.RegularExpressions.AspCodeRegex](#)  
[System.Web.RegularExpressions.AspEncodedExprRegex](#)  
[System.Web.RegularExpressions.AspExprRegex](#)  
[System.Web.RegularExpressions.CommentRegex](#)  
[System.Web.RegularExpressions.DatabindExprRegex](#)  
[Diğer...](#)

## Örnekler

Aşağıdaki örnek, bir dizedeki sözcüklerin tekrar tekrar tekrar olup olmadığını denetlemek için normal bir ifade kullanır. Normal ifade `\b(?:<word>\w+)\s+(\k<word>)\b` aşağıdaki tabloda gösterildiği gibi yorumlanabilir.

Desen	Description
<code>\b</code>	Eşleşmeyi bir sözcük sınırından başlatın.
<code>(?&lt;word&gt;\w+)</code>	Bir veya daha fazla sözcük karakterini bir sözcük sınıрыla eşleştirin. Yakalanan gruba adını verin <code>word</code> .
<code>\s+</code>	Bir veya daha fazla boşluk karakterini eşleştirin.
<code>(\k&lt;word&gt;)</code>	adlı <code>word</code> yakalanan grubu eşleştirin.
<code>\b</code>	Bir sözcük sınıрыla eşleş.

C#KopyalaÇalıştır

```
using System;
using System.Text.RegularExpressions;

public class Test
{
    public static void Main ()
    {
        // Define a regular expression for repeated words.
        Regex rx = new Regex(@"\b(?:<word>\w+)\s+(\k<word>)\b",
            RegexOptions.Compiled | RegexOptions.IgnoreCase);

        // Define a test string.
        string text = "The the quick brown fox fox jumps over the lazy dog dog.";

        // Find matches.
        MatchCollection matches = rx.Matches(text);

        // Report the number of matches found.
        Console.WriteLine("{0} matches found in:\n {1}",
            matches.Count,
            text);

        // Report on each match.
        foreach (Match match in matches)
```

```

    {
        GroupCollection groups = match.Groups;
        Console.WriteLine("'{'{0}'}' repeated at positions {1} and {2}",
            groups["word"].Value,
            groups[0].Index,
            groups[1].Index);
    }
}
// The example produces the following output to the console:
//      3 matches found in:
//      The the quick brown fox fox jumps over the lazy dog dog.
//      'The' repeated at positions 0 and 4
//      'fox' repeated at positions 20 and 25
//      'dog' repeated at positions 49 and 53

```

Aşağıdaki örnek, bir dizenin para birimi değerini temsil edip etmediğini veya para birimi değerini temsil etmek için doğru biçime sahip olup olmadığını denetlemek için normal ifadenin kullanımını gösterir. Bu durumda, normal ifade kullanıcının geçerli kültürü için , [CurrencyDecimalDigits](#), [NumberFormatInfo.CurrencySymbol](#), [NumberFormatInfo.NegativeSign](#) ve [NumberFormatInfo.PositiveSign](#) özelliklerinden [NumberFormatInfo.CurrencyDecimalSeparator](#) dinamik olarak oluşturulur. Sistemin geçerli kültürü en-US ise, sonuçta elde edilen normal ifade olur `^\s*(\+|-)?\s?\$?\s?(\d*\.\d{2}?)\{1\}$`. Bu normal ifade aşağıdaki tabloda gösterildiği gibi yorumlanabilir.

Desen	Description
<code>^</code>	Dizenin başlangıcından başlayın.
<code>\s*</code>	Sıfır veya daha fazla boşluk karakteriyle eşleş.
<code>(\+ -)?</code>	Pozitif işaretin veya negatif işaretin sıfır veya bir oluşumunu eşleştirin.
<code>\s?</code>	Sıfır veya bir beyaz boşluk karakterini eşleştirin.
<code>\\$?</code>	Dolar işaretinin sıfır veya bir oluşumunu eşleştirin.
<code>\s?</code>	Sıfır veya bir beyaz boşluk karakterini eşleştirin.
<code>\d*</code>	Sıfır veya daha fazla ondalık basamağı eşleştirin.
<code>\.?</code>	Sıfır veya bir ondalık nokta simgesiyle eşleş.
<code>\d{2}?</code>	İki ondalık basamağı sıfır veya bir kez eşleştirin.
<code>(\d*\.\d{2}?)\{1\}</code>	En az bir kez ondalık ayırıcı simgesiyle ayrılmış tam sayı ve kesirli basamak desenini eşleştirin.
<code>\$</code>	Dizenin sonunu eşleştirin.

Bu durumda normal ifade, geçerli bir para birimi dizesinin grup ayırıcı simgeleri içermediğini ve kesirli basamak içermediğini veya geçerli kültürün [CurrencyDecimalDigits](#) özelliği tarafından tanımlanan kesirli basamak sayısını içerdiğini varsayar.

C#

 Kopyala

 Çalıştır

```
using System;
using System.Globalization;
using System.Text.RegularExpressions;

public class Example
{
    public static void Main()
    {
        // Get the current NumberFormatInfo object to build the regular
        // expression pattern dynamically.
        NumberFormatInfo nfi = NumberFormatInfo.CurrentInfo;

        // Define the regular expression pattern.
        string pattern;
        pattern = @"^\s*[";
        // Get the positive and negative sign symbols.
        pattern += Regex.Escape(nfi.PositiveSign + nfi.NegativeSign) + @""]?";
        pattern += "\s?";
        // Get the currency symbol.
        pattern += Regex.Escape(nfi.CurrencySymbol) + @"?\s?";
        // Add integral digits to the pattern.
        pattern += @"(\d*";
        // Add the decimal separator.
        pattern += Regex.Escape(nfi.CurrencyDecimalSeparator) + "?";
        // Add the fractional digits.
        pattern += @"\d{";
        // Determine the number of fractional digits in currency values.
        pattern += nfi.CurrencyDecimalDigits.ToString() + "}?){1}$";

        Regex rgx = new Regex(pattern);

        // Define some test strings.
        string[] tests = { "-42", "19.99", "0.001", "100 USD",
                           ".34", "0.34", "1,052.21", "$10.62",
                           "+1.43", "-$0.23" };

        // Check each test string against the regular expression.
        foreach (string test in tests)
        {
            if (rgx.IsMatch(test))
                Console.WriteLine("{0} is a currency value.", test);
            else
                Console.WriteLine("{0} is not a currency value.", test);
        }
    }
}
```

```
// The example displays the following output:  
//      -42 is a currency value.  
//      19.99 is a currency value.  
//      0.001 is not a currency value.  
//      100 USD is not a currency value.  
//      .34 is a currency value.  
//      0.34 is a currency value.  
//      1,052.21 is not a currency value.  
//      $10.62 is a currency value.  
//      +1.43 is a currency value.  
//      -$0.23 is a currency value.
```

Bu örnekteki normal ifade dinamik olarak oluşturulduğundan, tasarım zamanında geçerli kültürün para birimi simgesinin, ondalık işaretinin veya pozitif ve negatif işaretlerin normal ifade dili işleçleri olarak normal ifade altyapısı tarafından yanlış yorumlanıp yorumlanamayacağını bilmiyoruz. Herhangi bir yanlış yorumlamayı önlemek için örnek, dinamik olarak oluşturulan her dizeyi yöntemine [Escape](#) geçirir.

## Açıklamalar

sınıfı, [Regex](#) .NET Framework normal ifade altyapısını temsil eder. Belirli karakter desenlerini bulmak için büyük miktarda metni hızla ayrıştırmak için kullanılabilir; metin alt dizelerini ayıklamak, düzenlemek, değiştirmek veya silmek için; ve ayıklanan dizeleri bir rapor oluşturmak üzere bir koleksiyona eklemek için.

### ⓘ Not

Birincil ilgi alanınız belirli bir desene uygun olup olmadığını belirleyerek bir dizeyi doğrulamaksa sınıfını **System.Configuration.RegexStringValidator** kullanabilirsiniz.

Normal ifadeleri kullanmak için, [Normal İfade Dili - Hızlı Başvuru'da](#) belgelenen söz dizimini kullanarak metin akışında tanımlamak istediğiniz deseni tanımlarsınız. Ardından, isteğe bağlı olarak bir [Regex](#) nesne örneği oluşturabilirsiniz. Son olarak, normal ifade deseni ile eşleşen metni değiştirme veya desen eşleşmesini tanımlama gibi bazı işlemleri gerçekleştiren bir yöntemi çağırırsınız.

### ⓘ Not

Bazı yaygın normal ifade desenleri için bkz. **Normal İfade Örnekleri**. Ayrıca, **Regular-Expressions.info'daki** gibi normal ifade desenlerinden oluşan bir dizi çevrimiçi kitaplık da vardır.

sınıfını [Regex](#) kullanma hakkında daha fazla bilgi için bu konudaki aşağıdaki bölümlere bakın:

- [Regex ve Dize Yöntemlerini Karşılaştırma](#)
- [Statik ve Örnek Yöntemleri Karşılaştırma](#)
- [Normal İfade İşlemleri Gerçekleştirme](#)
- [Bir Zaman Aşımı Değeri Tanımlama](#)

Normal ifade dili hakkında daha fazla bilgi için bkz . [Normal İfade Dili - Hızlı Başvuru](#) veya şu broşürlerden birini indirip yazdırmak:

[Word \(.docx\) biçiminde Hızlı Başvuru](#)

[PDF \(.pdf\) biçiminde Hızlı Başvuru](#)

## Regex ve Dize Yöntemlerini Karşılaştırma

sınıfı, [System.String](#) metinle desen eşleştirme gerçekleştirmek için kullanabileceğiniz çeşitli arama ve karşılaştırma yöntemleri içerir. Örneğin, [String.Contains](#), [String.EndsWith](#)ve [String.StartsWith](#) yöntemleri bir dize örneğinin belirtilen bir alt dize içerip içermediğini belirler; ve [String.IndexOf](#), [String.IndexOfAny](#), [String.LastIndexOf](#)ve [String.LastIndexOfAny](#) yöntemleri bir dizede belirtilen alt dizinin başlangıç konumunu döndürür. Belirli bir dizeyi [System.String](#) ararken sınıfının yöntemlerini kullanın. Dizede [Regex](#) belirli bir deseni ararken sınıfını kullanın. Daha fazla bilgi ve örnek için bkz . [.NET Normal İfadeleri](#).

[Açıklamalar'a Geri Dön](#)

## Statik ve Örnek Yöntemleri Karşılaştırma

Normal ifade desenini tanımladıktan sonra, bunu normal ifade altyapısına iki yoldan biriyle sağlayabilirsiniz:

- Normal ifadeyi temsil eden bir [Regex](#) nesne örneği oluşturarak. Bunu yapmak için normal ifade desenini bir [Regex](#) oluşturucuya geçirirsiniz. Nesne [Regex](#) sabittir; bir [Regex](#) nesneyi normal bir ifadeyle örneklediğinizde, bu nesnenin normal ifadesi değiştirilemez.
- Hem normal ifadeyi hem de aranacak metni ( `static` `Shared Visual Basic`) [Regex](#) yöntemine sağlayarak. Bu, açıkça bir nesne oluşturmadan normal bir [Regex](#) ifade kullanmanıza olanak tanır.

Tüm **Regex** desen belirleme yöntemleri hem statik hem de örnek aşırı yüklemelerini içerir.

Desenin kullanılabilmesi için normal ifade altyapısının belirli bir desen derlemesi gerekir. Nesneler sabit olduğundan **Regex** , bu bir sınıf oluşturucu veya statik yöntem çağrıldığında **Regex** oluşan tek seferlik bir yordamdır. Tek bir normal ifadeyi tekrar tekrar derleme gereksinimini ortadan kaldırmak için, normal ifade altyapısı statik yöntem çağrılarında kullanılan derlenmiş normal ifadeleri önbelleğe alır. Sonuç olarak, normal ifade desen eşleştirme yöntemleri statik ve örnek yöntemleri için karşılaştırılabilir performans sunar.

### ❗ Önemli

.NET Framework 1.0 ve 1.1 sürümlerinde, örneğinde veya statik yöntem çağrılarında kullanılan tüm derlenmiş normal ifadeler önbelleğe alınmıştı. .NET Framework 2.0'dan başlayarak yalnızca statik yöntem çağrılarında kullanılan normal ifadeler önbelleğe alınır.

Ancak önbelleğe alma işlemi aşağıdaki iki durumda performansı olumsuz etkileyebilir:

- Statik yöntem çağrılarını çok sayıda normal ifadeyle kullandığınızda. Varsayılan olarak, normal ifade altyapısı en son kullanılan 15 statik normal ifadeyi önbelleğe alır. Uygulamanız 15'ten fazla statik normal ifade kullanıyorsa, bazı normal ifadelerin yeniden derlenmiş olması gerekir. Bu yeniden derlemeyi önlemek için özelliğini artırabilirsiniz [Regex.CacheSize](#) .
- Daha önce derlenmiş normal ifadelerle yeni **Regex** nesnelerin örneğini oluştururken. Örneğin, aşağıdaki kod bir metin akışında yinelenen sözcükleri bulmak için normal bir ifade tanımlar. Örnekte tek bir normal ifade kullanılıyor olsa da, her metin satırını işlemek için yeni **Regex** bir nesne örneği oluşturur. Bu, normal ifadenin döngünün her yinelemesiyle yeniden derlenmesine neden olur.

C#

 Kopyala

```
StreamReader sr = new StreamReader(filename);
string input;
string pattern = @"b(\w+)\s1\b";
while (sr.Peek() >= 0)
{
    input = sr.ReadLine();
    Regex rgx = new Regex(pattern, RegexOptions.IgnoreCase);
    MatchCollection matches = rgx.Matches(input);
    if (matches.Count > 0)
    {
        Console.WriteLine("{0} ({1} matches):", input, matches.Count);
    }
}
```

```
        foreach (Match match in matches)
            Console.WriteLine("    " + match.Value);
    }
}
sr.Close();
```

Yeniden derlemeyi önlemek için, aşağıdaki yeniden yazılan örnekte gösterildiği gibi, bunu gerektiren tüm kodlar için erişilebilir olan tek [Regex](#) bir nesne örneği oluşturmanız gerekir.

C#

 Kopyala

```
StreamReader sr = new StreamReader(filename);
string input;
string pattern = @"\"b(\\w+)\\s\\1\\b";
Regex rgx = new Regex(pattern, RegexOptions.IgnoreCase);

while (sr.Peek() >= 0)
{
    input = sr.ReadLine();
    MatchCollection matches = rgx.Matches(input);
    if (matches.Count > 0)
    {
        Console.WriteLine("{0} ({1} matches):", input, matches.Count);
        foreach (Match match in matches)
            Console.WriteLine("    " + match.Value);
    }
}
sr.Close();
```

[Açıklamalar'a Geri Dön](#)

## Normal İfade İşlemleri Gerçekleştirme

İster bir [Regex](#) nesnenin örneğini oluşturup yöntemlerini çağırın, ister statik yöntemleri çağırın [Regex](#) , sınıfı aşağıdaki desen eşleştirme işlevini sunar:

- Eşleşmenin doğrulanması. Bir eşleşme olup olmadığını belirlemek için yöntemini çağırırsınız [IsMatch](#) .
- Tek bir eşleşme alınıyor. Bir dizede [Match](#) veya dizenin bir [Match](#) bölümünde ilk eşleşmeyi temsil eden bir nesneyi almak için yöntemini çağırırsınız. Sonraki eşleşmeler yöntemi çağırılarak [Match.NextMatch](#) alınabilir.
- Tüm eşleşmelerin alınması. Bir dizede [Matches](#) veya dizenin bir [System.Text.RegularExpressions.MatchCollection](#) bölümünde bulunan tüm eşleşmeleri temsil eden bir nesneyi almak için yöntemini çağırırsınız.



- Eşleşen metnin değiştirilmesi. Eşleşen metni değiştirmek için yöntemini çağırırsınız [Replace](#) . Değiştirme metni normal bir ifadeyle de tanımlanabilir. Buna ek olarak, bazı [Replace](#) yöntemler, değiştirme metnini program aracılığıyla tanımlamanızı sağlayan bir [MatchEvaluator](#) parametre içerir.
- Giriş dizesinin bölümlerinden oluşturulan bir dize dizisi oluşturma. Bir giriş dizesini [Split](#) normal ifade tarafından tanımlanan konumlara bölmek için yöntemini çağırırsınız.

Sınıfı, [Regex](#) desen eşleştirme yöntemlerine ek olarak birkaç özel amaçlı yöntem içerir:

- yöntemi, [Escape](#) normal ifade veya giriş dizesinde normal ifade işleçleri olarak yorumlanabilir tüm karakterlerin kaçışını verir.
- [Unescape](#) yöntemi bu kaçış karakterlerini kaldırır.
- yöntemi, [CompileToAssembly](#) önceden tanımlanmış normal ifadeler içeren bir derleme oluşturur. .NET Framework, ad alanında [System.Web.RegularExpressions](#) bu özel amaçlı derlemelerin örneklerini içerir.

[Açıklamalar'a Geri Dön](#)

## Bir Zaman Aşımı Değeri Tanımlama

.NET, desen eşleştirmede önemli güç ve esneklik sağlayan tam özellikli bir normal ifade dilini destekler. Bununla birlikte, güç ve esneklik bir maliyete neden olur: düşük performans riski. Kötü performans gösteren normal ifadelerin oluşturulması şaşırtıcı derecede kolaydır. Bazı durumlarda, aşırı geri izlemeyi kullanan normal ifade işlemleri, normal ifade deseni ile neredeyse eşleşen metinleri işlerken yanıt vermeyi durduruyor gibi görünebilir. .NET Normal İfade altyapısı hakkında daha fazla bilgi için bkz. [Normal İfade Davranışının Ayrıntıları](#). Aşırı geri izleme hakkında daha fazla bilgi için bkz. [Geri izleme](#).

.NET Framework 4.5'den başlayarak, aşırı geri izlemeyi sınırlamak için normal ifade eşleşmeleri için bir zaman aşımı aralığı tanımlayabilirsiniz. Normal ifade düzenine ve giriş metnine bağlı olarak, yürütme süresi belirtilen zaman aşımı aralığını aşabilir, ancak belirtilen zaman aşımı aralığından daha fazla zaman harcamaz. Normal ifade altyapısı zaman aşımına uğradıysa, bir [RegexMatchTimeoutException](#) özel durum oluşturur. Çoğu durumda bu, normal ifade düzeniyle neredeyse eşleşen metinleri eşleştirmeye çalışarak normal ifade altyapısının işlem gücünü boşa harcamasını önler. Ancak zaman aşımı aralığının çok düşük olduğunu veya geçerli makine yükünün performansta genel bir düşüşe neden olduğunu da gösterebilir.

Özel durumu nasıl işleyeceksiniz, özel durumun nedenlerine bağlıdır. Zaman aşımı aralığı çok düşük ayarlandığından veya aşırı makine yükünden dolayı özel durum oluşursa, zaman aşımı aralığını artırabilir ve eşleşen işlemi yeniden deneyebilirsiniz. Normal ifade aşırı geri izlemeyi gerektirdiğinden özel durum oluşursa, bir eşleşme olmadığını varsayabilir ve isteğe bağlı olarak, normal ifade desenini değiştirmenize yardımcı olacak bilgileri günlüğe kaydedebilirsiniz.

Normal ifade nesnesinin örneğini oluştururken oluşturucuyu [Regex\(String, RegexOptions, TimeSpan\)](#) çağırarak zaman aşımı aralığı ayarlayabilirsiniz. Statik yöntemler için, parametresi olan `matchTimeout` bir eşleştirme yönteminin aşırı yüklemesini çağırarak zaman aşımı aralığı ayarlayabilirsiniz. Zaman aşımı değerini açıkça ayarlamazsanız, varsayılan zaman aşımı değeri aşağıdaki gibi belirlenir:

- Varsa, uygulama genelinde zaman aşımı değerini kullanarak. Bu, nesnenin örneği oluşturulduğu veya statik yöntem çağırısının yapıldığı [Regex](#) uygulama etki alanı için geçerli olan herhangi bir zaman aşımı değeri olabilir. Bir değer dize gösterimini [TimeSpan](#) "REGEX\_DEFAULT\_MATCH\_TIMEOUT" özelliğine atamak için yöntemini çağırarak [AppDomain.SetData](#) uygulama genelinde zaman aşımı değerini ayarlayabilirsiniz.
- Uygulama genelinde zaman aşımı değeri [InfiniteMatchTimeout](#) ayarlanmamışsa değerini kullanarak.

### ❗ Önemli

Tüm normal ifade desen eşleştirme işlemlerinde bir zaman aşımı değeri ayarlamanızı öneririz. Daha fazla bilgi için bkz. [Normal İfadeler için En İyi Yöntemler](#).

[Açıklamalar'a Geri Dön](#)

## Oluşturucular

<a href="#">Regex()</a>	<a href="#">Regex</a> sınıfının yeni bir örneğini başlatır.
<a href="#">Regex(SerializationInfo, StreamingContext)</a>	Serileştirilmiş verileri kullanarak sınıfının yeni bir örneğini <a href="#">Regex</a> başlatır.
<a href="#">Regex(String)</a>	Belirtilen normal ifade için sınıfının yeni bir örneğini <a href="#">Regex</a> başlatır.
<a href="#">Regex(String, RegexOptions)</a>	Deseni <a href="#">Regex</a> değiştiren seçeneklerle belirtilen normal ifade için sınıfının yeni bir örneğini başlatır.

`Regex(String, RegexOptions, TimeSpan)`

Belirtilen normal ifade için sınıfın `Regex` yeni bir örneğini başlatır; deseni değiştiren seçenekler ve bir desen eşleştirme yönteminin zaman aşımına uğramadan önce ne kadar süreyle eşleşme denemesi gerektiğini belirten bir değer.

## Alanlar

<code>capnames</code>	yöntemi tarafından oluşturulan bir <code>Regex</code> nesne tarafından <code>CompileToAssembly</code> kullanılır.
<code>caps</code>	yöntemi tarafından oluşturulan bir <code>Regex</code> nesne tarafından <code>CompileToAssembly</code> kullanılır.
<code>capsize</code>	yöntemi tarafından oluşturulan bir <code>Regex</code> nesne tarafından <code>CompileToAssembly</code> kullanılır.
<code>capslist</code>	yöntemi tarafından oluşturulan bir <code>Regex</code> nesne tarafından <code>CompileToAssembly</code> kullanılır.
<code>factory</code>	yöntemi tarafından oluşturulan bir <code>Regex</code> nesne tarafından <code>CompileToAssembly</code> kullanılır.
<code>InfiniteMatchTimeout</code>	Desen eşleştirme işleminin zaman aşımına neden olmaması gerektiğini belirtir.
<code>internalMatchTimeout</code>	İşlem zaman aşımına uğramadan önce desen eşleştirme işleminde geçen en uzun süre.
<code>pattern</code>	yöntemi tarafından oluşturulan bir <code>Regex</code> nesne tarafından <code>CompileToAssembly</code> kullanılır.
<code>roptions</code>	yöntemi tarafından oluşturulan bir <code>Regex</code> nesne tarafından <code>CompileToAssembly</code> kullanılır.

## Özellikler

<code>CacheSize</code>	Derlenmiş normal ifadelerin geçerli statik önbelleğindeki en fazla girdi sayısını alır veya ayarlar.
<code>CapNames</code>	Adlandırılmış yakalama gruplarını dizin değerleriyle eşleyen bir sözlük alır veya ayarlar.
<code>Caps</code>	Numaralandırılmış yakalama gruplarını dizin değerleriyle eşleyen bir sözlük alır veya ayarlar.
<code>MatchTimeout</code>	Geçerli örneğin zaman aşımı aralığını alır.

Options	Oluşturucuya <a href="#">Regex</a> geçirilen seçenekleri alır.
RightToLeft	Normal ifadenin sağdan sola doğru arama yapıp yapmadığına ilişkin bir değer alır.

## Yöntemler

<a href="#">CompileToAssembly(Regex CompilationInfo[], Assembly Name)</a>	Belirtilen <a href="#">Regex</a> bir veya daha fazla nesneyi adlandırılmış bir derlemeye derler.
<a href="#">CompileToAssembly(Regex CompilationInfo[], Assembly Name, CustomAttribute Builder[])</a>	Belirtilen özniteliklerle bir veya daha fazla belirtilen <a href="#">Regex</a> nesneyi adlandırılmış bir derlemeye derler.
<a href="#">CompileToAssembly(Regex CompilationInfo[], Assembly Name, CustomAttribute Builder[], String)</a>	Belirtilen <a href="#">Regex</a> bir veya daha fazla nesneyi ve belirtilen kaynak dosyasını belirtilen özniteliklere sahip adlandırılmış bir derlemeye derler.
<a href="#">Equals(Object)</a>	Belirtilen nesnenin geçerli nesneye eşit olup olmadığını belirler. (Devralındığı yer: <a href="#">Object</a> )
<a href="#">Escape(String)</a>	En az sayıda karakterden (\, *, +, ?,  , {, [, (, ^, \$, ., # ve boşluk) kaçış kodlarıyla değiştirerek kaçış karakterinden kaçır. Bu, normal ifade altyapısına bu karakterleri meta karakter olarak değil sözcük anlamıyla yorumlamasını emreder.
<a href="#">GetGroupNames()</a>	Normal ifade için grup adlarını yakalama dizisi döndürür.
<a href="#">GetGroupNumbers()</a>	Bir dizideki grup adlarına karşılık gelen grup numaralarını yakalama dizisi döndürür.
<a href="#">GetHashCode()</a>	Varsayılan karma işlevi işlevi görür. (Devralındığı yer: <a href="#">Object</a> )
<a href="#">GetType()</a>	<a href="#">Type</a> Geçerli örneğini alır. (Devralındığı yer: <a href="#">Object</a> )
<a href="#">GroupNameFrom Number(Int32)</a>	Belirtilen grup numarasına karşılık gelen grup adını alır.
<a href="#">GroupNumberFrom Name(String)</a>	Belirtilen grup adına karşılık gelen grup numarasını döndürür.
<a href="#">InitializeReferences()</a>	yöntemi tarafından oluşturulan bir <a href="#">Regex</a> nesne tarafından <a href="#">CompileToAssembly</a> kullanılır.
<a href="#">IsMatch(String)</a>	Oluşturucuda belirtilen normal ifadenin <a href="#">Regex</a> belirtilen giriş

	dizesinde eşleşme bulup bulmadığını gösterir.
IsMatch(String, Int32)	Oluşturucuda belirtilen normal ifadenin <b>Regex</b> , dizede belirtilen başlangıç konumundan başlayarak belirtilen giriş dizesinde bir eşleşme bulup bulmadığını gösterir.
IsMatch(String, String)	Belirtilen normal ifadenin belirtilen giriş dizesinde eşleşme bulup bulmadığını gösterir.
IsMatch(String, String, Regex Options)	Belirtilen normal ifadenin belirtilen eşleştirme seçeneklerini kullanarak belirtilen giriş dizesinde eşleşme bulup bulmadığını gösterir.
IsMatch(String, String, Regex Options, TimeSpan)	Belirtilen normal ifadenin, belirtilen eşleştirme seçeneklerini ve zaman aşımı aralığını kullanarak belirtilen giriş dizesinde eşleşme bulup bulmadığını gösterir.
Match(String)	Oluşturucuda belirtilen normal ifadenin ilk oluşumu için belirtilen giriş dizesini <b>Regex</b> arar.
Match(String, Int32)	Giriş dizesinde, dizede belirtilen başlangıç konumundan başlayarak normal ifadenin ilk oluşumunu arar.
Match(String, Int32, Int32)	Belirtilen başlangıç konumundan başlayarak ve yalnızca belirtilen sayıda karakterde arama yaparak normal ifadenin ilk oluşumu için giriş dizesini arar.
Match(String, String)	Belirtilen normal ifadenin ilk oluşumu için belirtilen giriş dizesini arar.
Match(String, String, Regex Options)	Belirtilen eşleştirme seçeneklerini kullanarak belirtilen normal ifadenin ilk oluşumu için giriş dizesini arar.
Match(String, String, Regex Options, TimeSpan)	Belirtilen eşleştirme seçeneklerini ve zaman aşımı aralığını kullanarak belirtilen normal ifadenin ilk oluşumu için giriş dizesini arar.
Matches(String)	Normal ifadenin tüm oluşumları için belirtilen giriş dizesini arar.
Matches(String, Int32)	Belirtilen giriş dizesinde, dizede belirtilen başlangıç konumundan başlayarak normal ifadenin tüm oluşumlarını arar.
Matches(String, String)	Belirtilen normal ifadenin tüm oluşumları için belirtilen giriş dizesini arar.
Matches(String, String, Regex Options)	Belirtilen eşleştirme seçeneklerini kullanarak belirtilen giriş dizesinde belirtilen normal ifadenin tüm oluşumlarını arar.
Matches(String, String, Regex Options, TimeSpan)	Belirtilen eşleşen seçenekleri ve zaman aşımı aralığını kullanarak belirtilen giriş dizesinde belirtilen normal ifadenin tüm oluşumlarını arar.

<code>MemberwiseClone()</code>	Geçerli <code>Object</code> öğesinin sıg bir kopyasını oluşturur. (Devralındığı yer: <code>Object</code> )
<code>Replace(String, Match Evaluator)</code>	Belirtilen giriş dizesinde, belirtilen normal ifadeyle eşleşen tüm dizeleri bir temsilci tarafından döndürülen bir <code>MatchEvaluator</code> dizeyle değiştirir.
<code>Replace(String, Match Evaluator, Int32)</code>	Belirtilen giriş dizesinde, normal ifade deseniyle eşleşen belirtilen en fazla dize sayısını temsilci tarafından <code>MatchEvaluator</code> döndürülen bir dizeyle değiştirir.
<code>Replace(String, Match Evaluator, Int32, Int32)</code>	Belirtilen giriş alt dizesinde, normal ifade deseniyle eşleşen belirtilen en fazla dize sayısını bir temsilci tarafından <code>MatchEvaluator</code> döndürülen bir dizeyle değiştirir.
<code>Replace(String, String)</code>	Belirtilen bir giriş dizesinde, normal ifade deseniyle eşleşen tüm dizeleri belirtilen bir değiştirme dizesiyle değiştirir.
<code>Replace(String, String, Int32)</code>	Belirtilen giriş dizesinde, normal ifade deseniyle eşleşen belirtilen en fazla sayıda dizeyi belirtilen bir değiştirme dizesiyle değiştirir.
<code>Replace(String, String, Int32, Int32)</code>	Belirtilen bir giriş alt dizesinde, normal ifade deseniyle eşleşen belirtilen en fazla dize sayısını belirtilen bir değiştirme dizesiyle değiştirir.
<code>Replace(String, String, Match Evaluator)</code>	Belirtilen giriş dizesinde, belirtilen normal ifadeyle eşleşen tüm dizeleri bir temsilci tarafından <code>MatchEvaluator</code> döndürülen bir dizeyle değiştirir.
<code>Replace(String, String, Match Evaluator, RegexOptions)</code>	Belirtilen giriş dizesinde, belirtilen normal ifadeyle eşleşen tüm dizeleri bir temsilci tarafından döndürülen bir <code>MatchEvaluator</code> dizeyle değiştirir. Belirtilen seçenekler eşleşen işlemi değiştirir.
<code>Replace(String, String, Match Evaluator, RegexOptions, TimeSpan)</code>	Belirtilen giriş dizesinde, belirtilen normal ifadeyle eşleşen tüm alt dizeleri temsilci tarafından döndürülen bir <code>MatchEvaluator</code> dizeyle değiştirir. Ek parametreler eşleşen işlemi değiştiren seçenekleri ve eşleşme bulunamazsa zaman aşımı aralığını belirtir.
<code>Replace(String, String, String)</code>	Belirtilen bir giriş dizesinde, belirtilen normal ifadeyle eşleşen tüm dizeleri belirtilen bir değiştirme dizesiyle değiştirir.
<code>Replace(String, String, String, RegexOptions)</code>	Belirtilen bir giriş dizesinde, belirtilen normal ifadeyle eşleşen tüm dizeleri belirtilen bir değiştirme dizesiyle değiştirir. Belirtilen seçenekler eşleşen işlemi değiştirir.
<code>Replace(String, String, String, RegexOptions, TimeSpan)</code>	Belirtilen bir giriş dizesinde, belirtilen normal ifadeyle eşleşen tüm dizeleri belirtilen bir değiştirme dizesiyle değiştirir. Ek parametreler eşleşen işlemi değiştiren seçenekleri ve eşleşme bulunamazsa zaman aşımı aralığını belirtir.
<code>Split(String)</code>	Giriş dizesini, oluşturunca belirtilen normal ifade deseni

	tarafından tanımlanan konumlardaki bir alt dize dizisine <b>Regex</b> böler.
<b>Split(String, Int32)</b>	Bir giriş dizesini, oluşturucuda belirtilen normal ifade tarafından tanımlanan konumlarda, belirtilen en fazla sayıda alt dize dizisine <b>Regex</b> böler.
<b>Split(String, Int32, Int32)</b>	Bir giriş dizesini, oluşturucuda belirtilen normal bir ifade tarafından tanımlanan konumlarda bir alt dize dizisine <b>Regex</b> belirtilen en fazla sayıda böler. Normal ifade deseni araması, giriş dizesinde belirtilen karakter konumunda başlar.
<b>Split(String, String)</b>	Bir giriş dizesini normal ifade deseni tarafından tanımlanan konumlarda bir alt dize dizisine böler.
<b>Split(String, String, Regex Options)</b>	Bir giriş dizesini, belirtilen normal ifade deseni tarafından tanımlanan konumlarda bir alt dize dizisine böler. Belirtilen seçenekler eşleşen işlemi değiştirir.
<b>Split(String, String, Regex Options, TimeSpan)</b>	Bir giriş dizesini, belirtilen normal ifade deseni tarafından tanımlanan konumlarda bir alt dize dizisine böler. Ek parametreler, eşleşen işlemi değiştiren seçenekleri ve eşleşme bulunamazsa zaman aşımı aralığını belirtir.
<b>ToString()</b>	Oluşturucuya geçirilen normal ifade desenini <b>Regex</b> döndürür.
<b>Unescape(String)</b>	Giriş dizesindeki tüm kaçış karakterlerini dönüştürür.
<b>UseOptionC()</b>	yöntemi tarafından oluşturulan bir <b>Regex</b> nesne tarafından <b>CompileToAssembly</b> kullanılır.
<b>UseOptionR()</b>	yöntemi tarafından oluşturulan bir <b>Regex</b> nesne tarafından <b>CompileToAssembly</b> kullanılır.
<b>ValidateMatchTimeout(TimeSpan)</b>	Zaman aşımı aralığının kabul edilebilir bir aralık içinde olup olmadığını denetler.

## Belirtik Arabirim Kullanımları

<b>ISerializable.GetObjectData(SerializationInfo, StreamingContext)</b>	Bir <b>SerializationInfo</b> nesneyi geçerli <b>Regex</b> nesnenin seri durumdan çıkarılması için gereken verilerle doldurur.
---	---

## Şunlara uygulanır

Ürün	Sürümler
------	----------

Ürün	Sürümler
.NET	Core 1.0, Core 1.1, Core 2.0, Core 2.1, Core 2.2, Core 3.0, Core 3.1, 5, 6, 7 Preview 4
.NET Framework	1.1, 2.0, 3.0, 3.5, 4.0, 4.5, 4.5.1, 4.5.2, 4.6, 4.6.1, 4.6.2, 4.7, 4.7.1, 4.7.2, 4.8
.NET Standard	1.0, 1.1, 1.2, 1.3, 1.4, 1.6, 2.0, 2.1
UWP	10.0
Xamarin.iOS	10.8
Xamarin.Mac	3.0

## İş Parçacığı Güvenliği

[Regex](#) Sınıfı sabit (salt okunur) ve iş parçacığı güvenlidir. [Regex](#) nesneleri herhangi bir iş parçacığında oluşturulabilir ve iş parçacıkları arasında paylaşılabilir. Daha fazla bilgi için bkz. [İş Parçacığı Güvenliği](#).

## Ayrıca bkz.

- [RegexStringValidator](#)
- [.NET Normal İfadeleri](#)
- [Normal İfade Dili Öğeleri](#)
- [Normal İfadeler - Hızlı Başvuru \(Word biçiminde indir\)](#)
- [Normal İfadeler - Hızlı Başvuru \(PDF biçiminde indir\)](#)

## Önerilen içerik

### [String.Split Yöntem \(System\)](#)

Bu örnekte belirtilen bir dizenin veya Unicode karakter dizisinin öğeleriyle sınırlandırılmış alt dizeleri içeren bir dize dizisi döndürür.

### [String.Contains Yöntem \(System\)](#)

Belirtilen karakterin bu dize içinde olup olmadığını belirten bir değer döndürür.



### String.Substring Yöntem (System)

Bu örnekten bir alt dize alır. Bu üye aşırı yüklendi. Sözdizimi, kullanım ve örnekleri dahil olmak üzere bu üyeye ilgili eksiksiz bilgi için aşırı yükleme listesindeki ada tıklayın.

### String.Trim Yöntem (System)

Geçerli dizeden belirtilen bir karakter kümesinin baştaki ve sondaki tüm oluşumlarının kaldırıldığı yeni bir dize döndürür.

### String.Remove Yöntem (System)

Geçerli dizeden belirtilen sayıda karakterin silindiği yeni bir dize döndürür.

### Regex.IsMatch Yöntem (System.Text.RegularExpressions)

Normal ifadenin giriş dizisinde eşleşme bulup bulmadığını gösterir.

### String.Compare Yöntem (System)

Belirtilen String iki nesneyi karşılaştırır ve sıralama düzenindeki göreceli konumlarını gösteren bir tamsayı döndürür.

Daha fazla göster ▼