C# dilinde extension metodlar, mevcut sınıflara yeni metodlar eklemenin bir yoludur. Bu metodlar, özellikle mevcut sınıfın veya yapının kaynak kodunu değiştiremeyeceğiniz durumlarda faydalıdır. Extension metodlar, statik bir sınıfın içinde statik olarak tanımlanır ve `this` anahtar kelimesi kullanılarak genişletilecek sınıf veya yapının örneğini parametre olarak alır.

### Extension Metod Oluşturma Adımları

1. \*\*Statik Sınıf Oluşturun\*\*: Extension metodlar, statik sınıfların içinde tanımlanmalıdır.

2. \*\*Statik Metod Tanımlayın\*\*: Metod statik olmalı ve genişletmek istediğiniz sınıfın bir örneğini `this` anahtar kelimesi ile parametre olarak almalıdır.

3. \*\*Namespace Ekleme\*\*: Extension metodu kullanmak istediğiniz dosyaya, metodun bulunduğu namespace'i ekleyin.

### Örnek: Extension Metod Oluşturma

#### 1. Adım: Statik Sınıf ve Statik Metod Tanımlama

```csharp

using System;

namespace ExtensionMethods

{

public static class StringExtensions

{

public static bool IsPalindrome(this string str)

{

int min = 0;

int max = str.Length - 1;

while (true)

{

if (min > max)

{

return true;

}

char a = str[min];

char b = str[max];

if (char.ToLower(a) != char.ToLower(b))

{

return false;

}

min++;

max--;

}

}

}

}

```

Bu örnekte, `StringExtensions` adında bir statik sınıf oluşturduk ve `IsPalindrome` adında bir extension metod tanımladık. Bu metod, bir `string`'in palindrome olup olmadığını kontrol eder.

#### 2. Adım: Extension Metodunu Kullanma

Extension metodları kullanmak için, metodun tanımlandığı namespace'i kullanmak istediğiniz dosyaya eklemeniz gerekir.

```csharp

using System;

using ExtensionMethods; // Extension metodların bulunduğu namespace

class Program

{

static void Main()

{

string example = "Madam";

bool isPalindrome = example.IsPalindrome();

Console.WriteLine($"Is the word '{example}' a palindrome? {isPalindrome}");

}

}

```

Bu örnekte, `IsPalindrome` extension metodunu `string` türünde bir örnek üzerinde kullandık.

### Özet

- \*\*Statik Sınıf\*\*: Extension metodlar statik bir sınıf içinde tanımlanmalıdır.

- \*\*Statik Metod\*\*: Metod statik olmalı ve `this` anahtar kelimesi ile genişletilecek türü parametre olarak almalıdır.

- \*\*Namespace\*\*: Extension metodları kullanmak için doğru namespace'in eklenmesi gereklidir.

Bu şekilde, mevcut sınıflara veya yapılara yeni işlevler ekleyebilir ve onları daha esnek bir şekilde kullanabilirsiniz.

extension metodları oluştururken çeşitli seçenekler ve yaklaşımlar kullanabilirsiniz. İşte bazı yaygın senaryolar ve örnekler:

### 1. Temel Veri Türleri İçin Extension Metodlar

#### Örnek: `int` Türü İçin Extension Metod

```csharp

public static class IntExtensions

{

public static bool IsEven(this int number)

{

return number % 2 == 0;

}

}

```

Kullanım:

```csharp

int num = 10;

bool isEven = num.IsEven();

Console.WriteLine(isEven); // True

```

### 2. Koleksiyonlar İçin Extension Metodlar

#### Örnek: `IEnumerable<T>` İçin Extension Metod

```csharp

using System.Collections.Generic;

using System.Linq;

public static class EnumerableExtensions

{

public static IEnumerable<T> TakeEveryNth<T>(this IEnumerable<T> source, int n)

{

if (n <= 0)

throw new ArgumentException("N must be greater than 0.", nameof(n));

int index = 0;

foreach (var item in source)

{

if (index++ % n == 0)

yield return item;

}

}

}

```

Kullanım:

```csharp

List<int> numbers = new List<int> { 1, 2, 3, 4, 5, 6, 7, 8, 9, 10 };

var everySecondNumber = numbers.TakeEveryNth(2);

Console.WriteLine(string.Join(", ", everySecondNumber)); // 1, 3, 5, 7, 9

```

### 3. Daha Karmaşık Nesneler İçin Extension Metodlar

#### Örnek: `DateTime` Türü İçin Extension Metod

```csharp

public static class DateTimeExtensions

{

public static int CalculateAge(this DateTime birthDate)

{

int age = DateTime.Today.Year - birthDate.Year;

if (birthDate.Date > DateTime.Today.AddYears(-age)) age--;

return age;

}

}

```

Kullanım:

```csharp

DateTime birthDate = new DateTime(1990, 6, 1);

int age = birthDate.CalculateAge();

Console.WriteLine(age); // Doğum tarihine bağlı olarak yaş hesaplanır

```

### 4. Şartlı Extension Metodlar

Bazı durumlarda, metodun belirli koşullara bağlı olarak çalışmasını isteyebilirsiniz.

#### Örnek: `string` İçin Şartlı Extension Metod

```csharp

public static class StringExtensions

{

public static bool IsNullOrEmptyOrWhiteSpace(this string str)

{

return string.IsNullOrEmpty(str) || string.IsNullOrWhiteSpace(str);

}

}

```

Kullanım:

```csharp

string text = " ";

bool isEmpty = text.IsNullOrEmptyOrWhiteSpace();

Console.WriteLine(isEmpty); // True

```

### 5. Generic Extension Metodlar

Extension metodları generic yaparak, farklı türlerde çalışmasını sağlayabilirsiniz.

#### Örnek: `IList<T>` İçin Generic Extension Metod

```csharp

using System.Collections.Generic;

public static class ListExtensions

{

public static void AddRange<T>(this IList<T> list, IEnumerable<T> items)

{

foreach (var item in items)

{

list.Add(item);

}

}

}

```

Kullanım:

```csharp

List<int> numbers = new List<int> { 1, 2, 3 };

numbers.AddRange(new List<int> { 4, 5, 6 });

Console.WriteLine(string.Join(", ", numbers)); // 1, 2, 3, 4, 5, 6

```

### Özet

- \*\*Temel Veri Türleri\*\*: `int`, `string`, `DateTime` gibi türler için extension metodlar oluşturabilirsiniz.

- \*\*Koleksiyonlar\*\*: `IEnumerable<T>`, `List<T>` gibi koleksiyonlar için extension metodlar oluşturabilirsiniz.

- \*\*Generic Metodlar\*\*: Generic extension metodlar yazarak, farklı türler üzerinde çalışabilen metodlar oluşturabilirsiniz.

- \*\*Şartlı Metodlar\*\*: Belirli koşullara bağlı olarak çalışan extension metodlar yazabilirsiniz.

Extension metodlar, mevcut sınıfların ve yapıların işlevselliğini artırmanın esnek ve güçlü bir yoludur. Farklı senaryolara uygun extension metodlar yazarak kodunuzu daha okunabilir ve yeniden kullanılabilir hale getirebilirsiniz.