

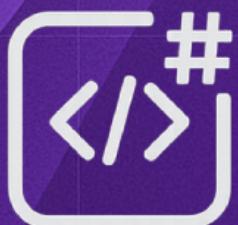
Extension Methods Nedir?

Extension methods, C# 3.0 ile hayatımıza giren ve mevcut tiplere kaynak kodunu değiştirmeden yeni metodlar eklememizi sağlayan güçlü bir özelliktir. Microsoft'un LINQ'i desteklemek için eklediği bu özellik, bugün modern C# programlamanın vazgeçilmez bir parçası haline geldi.

|static public [Return Tipi] [Method Adı] (this [Extension Yapılan Tip])

int
string
gibi

int
string
gibi



csharp
gunlukleri

Basit Bir Örnek Verirsek

```
// Basit ama güçlü bir örnek
public static class StringExtensions
{
    public static bool IsValidEmail(this string email)
    {
        if (string.IsNullOrWhiteSpace(email))
            return false;

        var regex = new Regex(@"^[\w\.-]+@[^\w\.-]+\.\w{2,}$");
        return regex.IsMatch(email);
    }
}

// Kullanımı
string userEmail = "developer@microsoft.com";
if (userEmail.IsValidEmail())
{
    // Email geçerli
}
```



csharp
gunlukleri

Bir Örnek Daha

0 references

```
internal class Program
```

```
{
```

0 references

```
private static void Main(string[] args)
```

```
{
```

```
    int sayi = 5;
```

```
    Console.WriteLine(sayi.Pow(1)); //Çıktı = 5
```

```
    Console.WriteLine(sayi.Pow(2)); //Çıktı = 25
```

```
    Console.WriteLine(sayi.Pow(3)); //Çıktı = 625
```

```
    Console.WriteLine(sayi.Pow(4)); //Çıktı = 390625
```

```
}
```

```
}
```

0 references

```
static public class Extension
```

```
{
```

4 references

```
static public int Pow(this int value, int powIndex = 2)
```

```
{
```

```
    for (int i = 0; i < powIndex - 1; i++)
```

```
    {
```

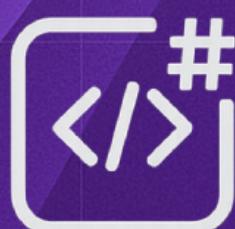
```
        value *= value;
```

```
    }
```

```
    return value;
```

```
}
```

```
}
```



csharp
gunlukleri

```
private static PropertyBuilder<T> GuidConfigs<T>()
{
    this.PropertyBuilder<T> propertyBuilder,
    bool.isRequired = true,
    string.columnType = "uniqueidentifier",
    bool.defaultNewId = false, // NEWID()
    bool.defaultSequential = false, // NEWSEQUENTIALID()
    bool.valueGeneratedOnAdd = false, // EF tarafından ValueGeneratedOnAdd
    Guid? defaultVal = null

    if (defaultNewId && defaultSequential)
        throw new ArgumentException("defaultNewId ve defaultSequential aynı anda true olamaz.");
    if (isRequired)
        propertyBuilderIsRequired();

    propertyBuilder.HasColumnType(columnType);

    if (valueGeneratedOnAdd)
        propertyBuilder.ValueGeneratedOnAdd();

    if (defaultNewId)
        propertyBuilder.HasValueSql("NEWID()");
    else if (defaultSequential)
        propertyBuilder.HasValueSql("NEWSEQUENTIALID()");
    else if (defaultVal is not null)
        propertyBuilder.HasValue(defaultVal);

    return propertyBuilder;
}
```



Yukarıdaki örnekte projenize özel EF extension örneği vardır. Bu sayede EF Configuration ayarlamalarında her seferinde aynı şeyleri yazmak yerine bir adet extension ile kolayca ekleme ve toplu yönetmeyi sağlayabilirsiniz.

Bu yöntemi fluent validation içinde kullanarak gereksiz tekrardan kurtulup toplu olarak kontrol sağlayabilirsiniz.

Extension methodlar her türlü tipte üretilmesi ve aynı class veya namespace içinde olmasına gerek olmadan herhangi bir yerden eklenebilme esnekliği sayesinde önemli bir etkiye sahiptir.

builder.Property(e => e.InsertedBy).RequiredGuidConfigs();

```
static public class JsonExtension
{
    15 references
    static public string ToJson<T>(this T obj) => JsonConvert.SerializeObject(obj);
    0 references
    static public T FromString<T>(this string str) where T : class, new() => JsonConvert.DeserializeObject<T>(str) ?? new T();
}
```

Yukarıdaki örnek basit bir örnek olsada projelerde çok fazla kullanmaktayım. Bu sayede geliştirme aşamasında istediğim yerde .ToJson() diyerek objeyi konsolda görebiliyorum.

Burada Json dönüşümü için Newtonsoft.Json kullandım. İleride değiştirmem gerekirse, dönüşümlerin çoğunu buradan yönettiğim için tek bir dosyada yapacağım json değişiklikleri ile tüm projenin json altyapısını değiştirebilirim.

Tabiki bazı yerlerde class üretip constructor içinde özel ayarlamalar yapmak gerekiyor. Fakat ufak işlerin extension kullanarak yapılması projelerde büyük avantaj ve hız kazandırmaktadır.

- ▷ [A C# DateTimeExtension.cs](#)
- ▷ [A C# GuidExtension.cs](#)
- ▷ [A C# HashingExtensions.cs](#)
- ▷ [✓ C# IConfigurationExtension.cs](#)
- ▷ [A C# IEnumerableExtension.cs](#)
- ▷ [A C# IntegerExtension.cs](#)
- ▷ [A C# IQueryableExtension.cs](#)
- ▷ [A C# JsonExtension.cs](#)
- ▷ [A C# ObjectExtension.cs](#)
- ▷ [A C# PropertyBuilderExtension.cs](#)
- ▷ [A C# StringExtension.cs](#)

Örnek bir projemin extension altyapısı.