**Attribute Nedir ?**

C# **Attributes**, bir programda kodun derleme zamanında veya çalışma zamanında davranışını değiştirmek, ek bilgi sağlamak veya işlemek için kullanılan **metadata**bileşenleridir. Genellikle koda eklenen etiketler olarak da düşünülebilirler. Bu etiketler, bir özelliği veya davranışı tanımlamak için kullanılır ve çeşitli programlama senaryolarında büyük esneklik sağlarlar.

C# **Attributes**, genellikle [**AttributeName**] şeklinde kullanılır, burada **AttributeName**, belirli bir özelliği veya davranışı tanımlayan önceden tanımlanmış veya özel olarak oluşturulmuş bir **attribute**ismidir. Örneğin, [**Serializable**] **attribute'ü**, bir sınıfın serileştirilebilir olduğunu belirtmek için kullanılırken, [**Obsolete**] **attribute'ü**, bir kodun artık kullanılmadığını belirtmek için kullanılır.

**Nerelerde Kullanılır ?**

* Kod hakkında ek bilgi vermek (örn: [Obsolete])
* Çalışma zamanında (runtime) bazı özel işlemler yapmak
* Veri doğrulama (validation)
* Serileştirme / JSON veya XML dönüşümü
* Unit testlerde işaretleme
* Web API'lerde davranış değiştirme

| **Kullanım Yeri** | **Açıklama** |
| --- | --- |
| [Obsolete] | Eskimiş bir metod/sınıf varsa uyarı verdirir. |
| [Required] | Bir alan boş geçilemez. Genelde form/doğrulama işlemlerinde. |
| [Serializable] | Nesnenin serileştirilebilir (JSON, XML) olduğunu belirtir. |
| [HttpGet], [HttpPost] | Web API controller’larında hangi HTTP isteğine yanıt verileceğini tanımlar. |
| **Örnek :** |  |

[Obsolete("Bu metod artık kullanılmıyor, YeniMetod'u kullanın.")]

public void EskiMetod()

{

Console.WriteLine("Eski metod");

}

EskiMetod(); // Derleyici seni uyarır: Bu metod eski, kullanma!

public class Kullanici

{

[Required] // Formdan veri alırken, Ad boşsa hata verir

public string Ad { get; set; }

}

**Örnek (Özelleştirilmiş Attribute)**

**[AttributeUsage(AttributeTargets.Method | AttributeTargets.Class)]**

public class SevvalAttribute : Attribute

{

public string Importance { get; }

public SevvalAttribute(string importance)

{

Importance = importance;

}

}

//kullanim

[Sevval("Yüksek")]

public class ImportantClass

{

[Sevval("Kritik")]

public void CriticalMethod()

{

Console.WriteLine("Bu metod kritik bir öneme sahiptir.");

}

}

**Attribute Okuma (Reflection ile):**

Type type = typeof(ImportantClass);

object[] attributes = type.GetCustomAttributes(typeof(SevvalAttribute), false);

foreach (SevvalAttribute attr in attributes)

{

Console.WriteLine($"Sınıfın önemi: {attr.Importance}");

}

MethodInfo method = type.GetMethod("CriticalMethod");

attributes = method.GetCustomAttributes(typeof(SevvalAttribute), false);

foreach (SevvalAttribute attr in attributes)

{

Console.WriteLine($"Metodun önemi: {attr.Importance}");

}

**Attribute'lar ile veri doğrulama (validation) nasıl yapılabilir ?**

Attribute'lar ile veri doğrulama, C# dilinde bir sınıfın özelliklerine doğrudan doğrulama kuralları eklemeyi sağlar. Özellikle System.ComponentModel.DataAnnotations isim alanındaki [Required], [StringLength], [Range], [EmailAddress] gibi hazır attribute’lar kullanılarak, bir verinin boş olup olmadığı, belirli bir uzunlukta veya aralıkta olup olmadığı gibi kurallar tanımlanabilir. Bu doğrulama kuralları, çalışma zamanında Validator sınıfı yardımıyla otomatik olarak kontrol edilir ve olası hatalar kullanıcıya bildirilebilir. Bu yöntem, özellikle kullanıcı girdilerinin kontrol edilmesi gereken durumlarda kullanışlıdır ve kodun daha okunabilir, düzenli ve merkezi bir yapıda olmasını sağlar.