

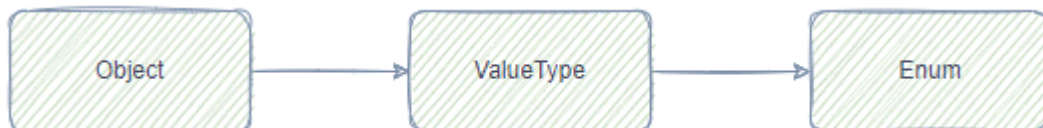
第07期-细说枚举类型

2022年3月23日 9:56

类型说明

枚举类型 是由基础整型数值类型的一组命名常量定义的值类型。

```
public abstract class Enum : ValueType, IComparable, IConvertible, IFormattable
```



```
enum Season
{
    Spring,
    Summer,
    Autumn,
    Winter
}
```

```
enum Season : ushort
{
    Spring,
    Summer,
    Autumn,
    Winter
}
```

默认从 0 索引开始，可自定义，从自定义 +1 类推。

实例化枚举

```
Season status0 = Season.Spring;
```

```
Season status1 = new Season();
```

```
Season status2 = (Season)1;
```

位标志的枚举类型

```
[Flags]
public enum Days
{
    None      = 0b_0000_0000, // 0
    Monday    = 0b_0000_0001, // 1
    Tuesday   = 0b_0000_0010, // 2
    Wednesday = 0b_0000_0100, // 4
}
```

```

Thursday = 0b_0000_1000, // 8
Friday   = 0b_0001_0000, // 16
Saturday = 0b_0010_0000, // 32
Sunday   = 0b_0100_0000, // 64
Weekend   = Saturday | Sunday
}

public class FlagsEnumExample
{
    public static void Main()
    {
        Days meetingDays = Days.Monday | Days.Wednesday | Days.Friday;
        Console.WriteLine(meetingDays);
        // Output:
        // Monday, Wednesday, Friday

        Days workingFromHomeDays = Days.Thursday | Days.Friday;
        Console.WriteLine($"Join a meeting by phone on {meetingDays &
workingFromHomeDays}");
        // Output:
        // Join a meeting by phone on Friday

        bool isMeetingOnTuesday = (meetingDays & Days.Tuesday) ==
Days.Tuesday;
        Console.WriteLine($"Is there a meeting on Tuesday:
{isMeetingOnTuesday}");
        // Output:
        // Is there a meeting on Tuesday: False

        var a = (Days)37;
        Console.WriteLine(a);
        // Output:
        // Monday, Wednesday, Saturday
    }
}

```

HasFlag(Enum) 可确定当前实例中是否设置了一个或多个位域。

类型转换

```

int value1 = 2;

Season result = (Season)value;

int value2 = (int)result;

Season season = (Season)Enum.ToObject(typeof(Season), 2);

var number = Convert.ChangeType(Season.Spring,
Enum.GetUnderlyingType(typeof(Season)));

```

Enum.Parse(Type, String) 和 Enum.TryParse<TEnum>(String, TEnum)

转换之前的判断

```

bool result = Enum.IsDefined(typeof(Season), 2)

```

格式化字符串

```
string[] formats = { "G", "F", "D", "X" };

Season status = Season.Autumn;

foreach (var fmt in formats)
{
    Console.WriteLine(status.ToString(fmt));
}
```

访问枚举的成员

```
string[] names = Enum.GetNames(typeof(Season));

Array values = Enum.GetValues(typeof(Season));
```

为枚举类型添加方法

```
public enum Grades { F = 0, D = 1, C = 2, B = 3, A = 4 };

public static class Extensions
{
    public static Grades minPassing = Grades.D;

    public static bool Passing(this Grades grade)
    {
        return grade >= minPassing;
    }
}
```

枚举显示扩展

```
public enum Season
{
    [Display(Name = "春")]
    Spring,

    [Display(Name = "夏")]
    Summer,

    [Display(Name = "秋")]
    Autumn,

    [Display(Name = "冬")]
    Winter
}
```

```
public static class EnumDisplayExtensions
```

```

{
    public static string? GetDisplayName(this Enum @enum)
    {
        return @enum.GetType()
            .GetMember(@enum.ToString())
            .FirstOrDefault()
            ?.GetCustomAttribute<DisplayAttribute>(false)
            ?.Name
            ?? @enum.ToString();
    }
}

```

```

string? str = Season.Autumn.GetDisplayName();

```

设计准则

- PascalCasing
- 单数类型名称，除非其值为位域
- 在简单枚举上提供零值