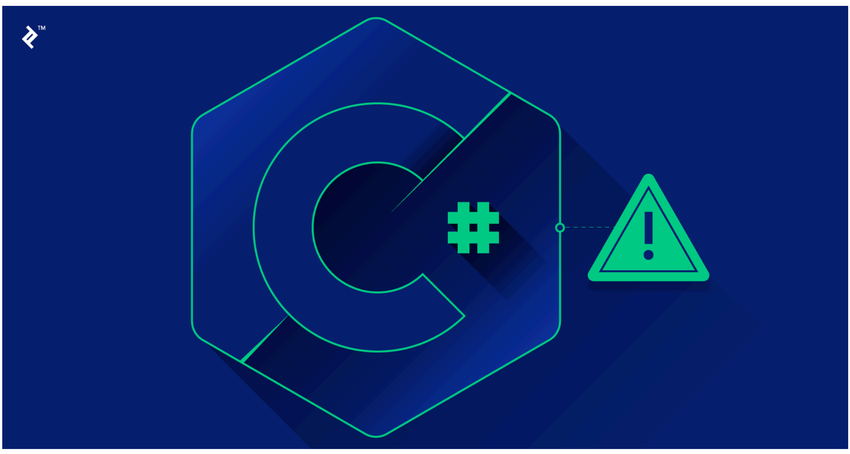
C# Switch 语句进阶：模式匹配详解与实例演示

作者：微信公众号：【架构师老卢】 11-26 8:18 157



概述：在C#中，switch语句的模式匹配在C# 7.0及以上版本中引入。以下是switch语句中常见的模式及其使用方法的示例：

在C#中，switch语句的模式匹配在C# 7.0及以上版本中引入。以下是switch语句中常见的模式及其使用方法的示例：

1. 类型模式：

优点： 用于检查对象的运行时类型，使代码更具可读性。

```
public static string GetObjectType(object obj)
{
    switch (obj)
    {
        case int i:
            return "整数类型";
        case string s:
            return "字符串类型";
        case double d:
            return "双精度浮点数类型";
        default:
            return "其他类型";
    }
}
```

2. 常量模式：

优点： 用于匹配对象是否等于某个常量值。

```
public static string GetDayOfWeekName(DayOfWeek day)
{
    switch (day)
    {
        case DayOfWeek.Monday:
            return "星期一";
        case DayOfWeek.Tuesday:
            return "星期二";
        case DayOfWeek.Wednesday:
            return "星期三";
        case DayOfWeek.Thursday:
            return "星期四";
        case DayOfWeek.Friday:
            return "星期五";
        default:
            return "其他";
    }
}
```

3. 组合模式：

优点： 允许将多个模式组合在一起，形成更复杂的匹配条件。

```
public static string GetInfo(object obj)
{
    switch (obj)
    {
        case int i when i > 0:
            return "正整数";
        case int i when i < 0:
            return "负整数";
        case string s when s.Length > 10:
            return "字符串长度大于10";
        default:
            return "其他";
    }
}
```

4. 属性模式：

优点： 用于匹配对象的属性，提供更灵活的条件判断。

```
public static string GetPersonInfo(object person)
{
    switch (person)
    {
        case { Age: > 18, Name: "Alice" }:
            return "成年人 Alice";
        case { Age: > 18, Name: "Bob" }:
            return "成年人 Bob";
        case { Age: <= 18, Name: "Alice" }:
            return "未成年人 Alice";
        default:
            return "其他";
    }
}

public class Person
{
    public string Name { get; set; }
    public int Age { get; set; }
}
```

5. 变量模式：

优点： 允许在模式中引入新的变量，提供更灵活的条件判断。

```
public static string GetVariablePattern(object obj)
{
    switch (obj)
    {
        case int i when i > 0:
            return $"正整数：{i}";
        case int i when i < 0:
            return $"负整数：{i}";
        case string s:
            return $"字符串：{s}";
        default:
            return "其他";
    }
}
```

- 模式匹配使得`switch`语句更为强大，能够更直观地表达条件逻辑。

- 不同的模式适用于不同的场景，根据需求选择合适的模式，提高代码的可读性和可维护性。
- 使用模式匹配可以减少代码中的重复，并提供更灵活的条件判断方式。