

第13期-可为空的值类型

2022年3月23日 9:56

可空值类型介绍

可为空的值类型都是泛型 `System.Nullable<T>` 结构的实例。需要表示基础值类型的**未定义值**时，通常使用可为空的值类型。

```
double? pi = 3.14;
char? letter = 'a';

int m2 = 10;
int? m = m2;

bool? flag = null;

// An array of a nullable value type:
int?[] arr = new int?[10];
```

检查类型

```
int? a = 42;

if (a is int valueOfA)
{
    Console.WriteLine($"a is {valueOfA}");
}
else
{
    Console.WriteLine("a does not have a value");
}
```

`Nullable<T>.HasValue` 判断是否有值

```
int? b = 10;
if (b.HasValue)
{
    Console.WriteLine($"b is {b.Value}");
}
else
{
    Console.WriteLine("b does not have a value");
}
// Output:
// b is 10
```

可为空的值类型转换为基础类型

```
int? a = 28;  
int b = a ?? -1;
```

强制转换

```
int? n = null;  
int n2 = (int)n;
```

提升的运算符

```
int? a = 10;  
int? b = null;  
int? c = 10;
```

```
a++;           // a is 11  
a = a * c;     // a is 110  
a = a + b;     // a is null
```

可空布尔值的运算比较

x	y	x&y	x y
true	true	true	true
true	false	false	true
true	null	null	true
false	true	false	true
false	false	false	false
false	null	false	null
null	true	null	true
null	false	false	null
null	null	null	null

逻辑比较运算符

```
int? a = 10;  
Console.WriteLine($"{a} >= null is {a >= null}");  
Console.WriteLine($"{a} < null is {a < null}");  
Console.WriteLine($"{a} == null is {a == null}");  
// Output:  
// 10 >= null is False  
// 10 < null is False
```

```
// 10 == null is False

int? b = null;
int? c = null;
Console.WriteLine($"null >= null is {b >= c}");
Console.WriteLine($"null == null is {b == c}");
// Output:
// null >= null is False
// null == null is True
```

判断某个类型是否为可空的值类型

获取可以为 null 的类型的基础类型参数

```
bool IsNullable(Type type) => Nullable.GetUnderlyingType(type) != null;
```

切勿使用 `Object.GetType` 获取可空值类型

```
int? a = 17;
Type typeOfA = a.GetType();
Console.WriteLine(typeOfA.FullName);
// Output:
// System.Int32
```

请勿使用 `is` 运算符来确定实例是否是可为空的值类型。

```
int? a = 14;
if (a is int)
{
    Console.WriteLine("int? instance is compatible with int");
}

int b = 17;
if (b is int?)
{
    Console.WriteLine("int instance is compatible with int?");
}
// Output:
// int? instance is compatible with int
// int instance is compatible with int?
```

参考资料

可为空的值类型

<https://docs.microsoft.com/zh-cn/dotnet/csharp/language-reference/builtin-types/nullable-value-types>