为什么要引入可空值类型？

因为值类型存储在线程栈上，因此值类型变量永远不会为null，但某些情况下值类型会为null。

举个例子，在数据库中给某一列的数据类型定义为Int32然后映射到FCL（Framework Class Library）的数据类型，但数据库中的一个列可能允许这个Int32类型的值为空（即某一列上没有任何值），那么将数据库与程序连接后尝试使用C#程序处理数据库数据时就会出现问题——因为CLR中无法将Int32值表示为null。

Ps：SQL语言本身并不面向CLR（这就是为什么会出现Int32可以为null）。但是，Microsoft SQL Server提供了一种称为SQL CLR的技术，它允许开发人员使用.NET Framework中的任何CLR语言（如C#或VB.NET）编写存储过程、触发器、用户定义函数和用户定义类型。这些对象可以在SQL Server中部署并像传统的T-SQL对象一样使用。

为了解决CLR规定值类型不能为空，产生和数据库数据间的问题。CLR引入了可空值类型的概念。

CLR是如何实现可空值类型的？

可空值类型是通过使用System.Nullable<T>结构来实现的。这个结构包含两个字段：一个用于存储值类型的实际值，另一个用于指示该值是否为null。

System.Nullable<T>的逻辑表示：

图片包含 文本

描述已自动生成

电脑屏幕的照片上有文字

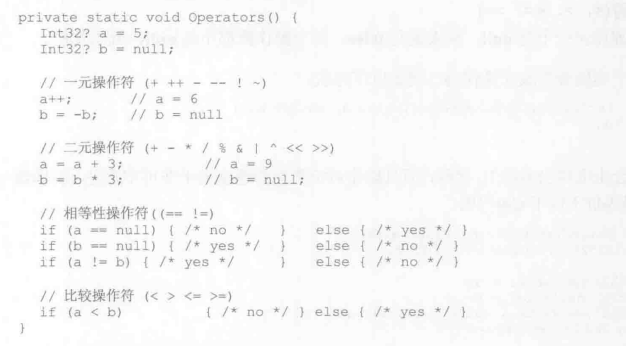
描述已自动生成

C#对可空值类型的支持：

1.简化语法比如“Int32?”等价于“Nullable<Int32>”

2.允许可空实例使用操作符

例子：



对于可空值类型，C#是这样解析这些操作符的（特别注意两Boolean类型之间的三值操作）

电脑屏幕的照片上有文字

描述已自动生成

Ps：因为重载了大量的操作符，操作可空值类型实例会生成大量IL代码，因此操作可空值类型的实例的速度要慢于非可空类型

3.空接合操作符（“??”操作符）

空接合操作符，??左边值不为null，那么它会返回左值，如果左边为null它会返回右侧指定的默认值。

Ps：空接合操作符既可用于引用类型也可用于值类型

相比于?：操作符，使用??操作符可以使得代码更易读更容易被理解。

CLR对可空值类型的支持：

（上面的都是C#语言对可空值类型的支持，因为无论是操作符重载还是类型名的简化，都是C#编译器层面上对可空值类型语法的优化，CLR对这些操作符重载等东西一无所知）

1.可空值类型的装箱

当CLR对可空值类型实例进行装箱时，会检查它是否为null，如果为null则不装箱任何东西直接返回null；如果不为null，CLR从实例中取出值进行装箱返回堆上的引用到栈上的变量。

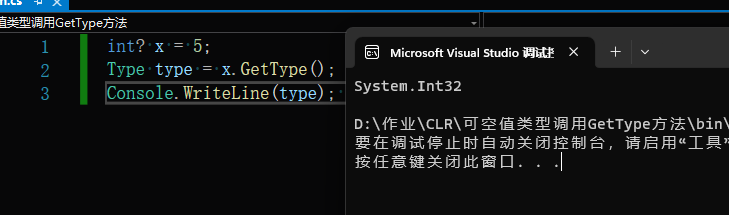
2.可空值类型的拆箱

CLR允许将已装箱的值类型T拆箱为T或者一个Nullable<T>，如果对已装箱值的引用为null，拆箱为可空值类型时会将这个可空值类型的实例的值设为null。

3.通过可空值类型调用GetType

在Nullable<T>上调用GetType，返回的类型为T而非Nullable<T>

例子：



4.通过可空值类型调用接口方法

因为可空值类型没有实现IComparable<T>接口，因此在转型为接口类型IComparable<T>时会十分繁琐，但CLR允许使用简化语法让直接装箱可以通过编译

下面是书上P395的例子：

