**使用 as 和 is 运算符安全地进行强制转换（C# 编程指南）**

由于对象是多态的，因此基类类型的变量可以保存派生类型。若要访问派生类型的方法，需要将值强制转换回该派生类型。不过，在这些情况下，如果只尝试进行简单的强制转换，会导致引发 [InvalidCastException](https://msdn.microsoft.com/zh-cn/library/system.invalidcastexception.aspx) 的风险。这就是 C# 提供 [is](https://msdn.microsoft.com/zh-cn/library/scekt9xw.aspx) 和 [as](https://msdn.microsoft.com/zh-cn/library/cscsdfbt.aspx) 运算符的原因。您可以使用这两个运算符来测试强制转换是否会成功，而没有引发异常的风险。通常，**as** 运算符更高效一些，因为如果可以成功进行强制转换，它会实际返回强制转换值。而 **is** 运算符只返回一个布尔值。因此，如果只想确定对象的类型，而无需对它进行实际强制转换，则可以使用 is 运算符。

下面的示例演示如何使用 **is** 和 **as** 运算符从一个引用类型强制转换为另一个引用类型，而没有引发异常的风险。此示例还演示如何对 **as** 运算符使用可以为 null 值的类型。

class SafeCasting

{

class Animal

{

public void Eat() { Console.WriteLine("Eating."); }

public override string ToString()

{

return "I am an animal.";

}

}

class Mammal : Animal { }

class Giraffe : Mammal { }

class SuperNova { }

static void Main()

{

SafeCasting app = new SafeCasting();

// Use the is operator to verify the type.

// before performing a cast.

Giraffe g = new Giraffe();

app.UseIsOperator(g);

// Use the as operator and test for null

// before referencing the variable.

app.UseAsOperator(g);

// Use the as operator to test

// an incompatible type.

SuperNova sn = new SuperNova();

app.UseAsOperator(sn);

// Use the as operator with a value type.

// Note the implicit conversion to int? in

// the method body.

int i = 5;

app.UseAsWithNullable(i);

double d = 9.78654;

app.UseAsWithNullable(d);

// Keep the console window open in debug mode.

System.Console.WriteLine("Press any key to exit.");

System.Console.ReadKey();

}

void UseIsOperator(Animal a)

{

if (a is Mammal)

{

Mammal m = (Mammal)a;

m.Eat();

}

}

void UseAsOperator(object o)

{

Mammal m = o as Mammal;

if (m != null)

{

Console.WriteLine(m.ToString());

}

else

{

Console.WriteLine("{0} is not a Mammal", o.GetType().Name);

}

}

void UseAsWithNullable(System.ValueType val)

{

int? j = val as int?;

if (j != null)

{

Console.WriteLine(j);

}

else

{

Console.WriteLine("Could not convert " + val.ToString());

}

}

}