C# Typeof, GetType, is

2013-03-25 PV

I've seen many people use the following code:

Type t = typeof(obj1);

if (t == typeof(int))

// Some code here

But I know you could also do this:

if (obj1.GetType() == typeof(int))

// Some code here

Or this:

if (obj1 is int)

// Some code here

Personally, I feel the last one is the cleanest, but is there something I'm missing? Which one is the best to use, or is it personal preference?

All are different.

* typeof takes a type name (which you specify at compile time).
* GetType gets the runtime type of an instance.
* is returns true if an instance is in the inheritance tree.

The is operator does *not* check if the runtime type of the operand is *exactly* the given type; rather, it checks to see if the runtime type *is compatible with* the given type:

class Animal {}

class Tiger : Animal {}

...

object x = new Tiger();

bool b1 = x is Tiger; // true

bool b2 = x is Animal; // true also! Every tiger is an animal.

But checking for type *identity* with reflection checks for *identity*, not for *compatibility*

bool b3 = x.GetType() == typeof(Tiger); // true

bool b4 = x.GetType() == typeof(Animal); // false! even though x is an animal

If that's not what you want, then you probably want IsAssignableFrom:

bool b5 = typeof(Tiger).IsAssignableFrom(x.GetType()); // true

bool b6 = typeof(Animal).IsAssignableFrom(x.GetType()); // true! A variable of type Animal may be assigned a Tiger.