Code I Found: A Warning about Object Comparisons

Posted At : August 27, 2010 2:24 PM | Posted By : Jon Hartmann
Related Categories: General

Last time I posted a "Code I Found" entry some people got upset... well I'm doing it again, but I'm pairing it with an actual warning about object value comparisons in strongly typed languages. This example shows both a valid problem and some dumb coding that complicates it.

The Code

This is the code I found, written it C# (variable names have been changed to protect the innocent). The intention of this code is to get a value from a drop down and compare it to the original value. If the new value is valid and is different, perform some actions to handle that change.

```
object newValue = Convert.ToInt32(dropdownOptions.SelectedValue);


if (((int)newValue > 0) && (MyObject.Value != newValue))
{
    MyObject.Value = newValue.ToString();
    ...
    OtherObject.DoSomething((int)newValue);
    ...
    AnotherObject.DosomethingElse((int)newValue);
}
```

When I glanced at this code the first time, I did't think much of it, because the use of `(int)` to cast `newValue` over and over again didn't really stand out. I did notice that the `newValue` was saved as an object, but I wasn't too concerned. It wasn't until I ran the code that things went a little weird.

The Problem

No matter what value I chose in the drop down (even the original value) this if block executed. I checked my Watch list while running and saw that `(MyObject.Value != newValue)` always evaluated to true, no matter what value I picked in the drop down. Looking closer, the watch list told me that both `MyObject.Value` and `newValue` may be the same value (say 5) and still `(MyObject.Value != newValue)` was true.

Not knowing what was going on, but finding It wasn't until I noticed the object casting that I started to get suspicious.object newValue = Convert.ToInt32(dropdownOptions.SelectedValue); to be a really odd statement, I went ahead and switched it to an `int`, and that when things started to unravel.

Once `newValue` was an int, the if statement complained that it couldn't compare an object with an int: `MyObject.Value` was an `object`, and it wasn't an int. What this means is that the `!=` statement was always going to be true, because an object cannot equal an int, even if they have the same value, because they are not the same *type*.

The Solution

I corrected the code as follows:

```
int newValue = Convert.ToInt32(dropdownOptions.SelectedValue);


if ((newValue > 0) && (Convert.ToInt32(MyObject.Value) != newValue))
{
    MyObject.Value = newValue.ToString();
    ...
    OtherObject.DoSomething(newValue);
    ...
    AnotherObject.DosomethingElse(newValue);
}
```

This not only fixes the logic problem, but actually simplifies the code.