Boxing and Unboxing in VB.NET

Posted by **Hannes Du Preez** on **May 22nd, 2015**

Introduction

Being in the programming game for so long, you tend to forget the basic principles of the .NET language. Today's topic is no exception. For experienced developers, a topic such as boxing and unboxing might already be second nature, but for the inexperienced programmer, this topic is vital, and the sooner you learn it, the better.

Boxing

Now, let me get this straight. This is not Floyd Mayweather versus Manny Pacquiao. I just had to get that out of the way....

Boxing is simply the process of converting a value type to any interface type implemented by the current value type; the most common being the Value object. Boxing is used to store value types in the garbage-collected heap.

This means that, once a variable is boxed, it can hold a separate value in a separate memory location. When a value type is boxed, a new object must be allocated and constructed.

A simple example of Boxing looks like this:

```
1. Dim i As Integer
2. i = 123
3.
4. Dim o As Objecto = i
```

The value of i gets copied into the variable o, because these two value types are compatible. Another example of Boxing is:

```
1. Dim i As Integer
2. i = 123
3. Dim L As Long
4. L = i
```

The variable i is compatible with the Long object, so it could be easily copied. For more information on Value types, please read the following:

https://msdn.microsoft.com/en-us/library/t63sy5hs.aspx

For more information regarding the Garbage Collector and Garbage Collection, please read the following:

https://msdn.microsoft.com/en-us/library/ee787088%28v=vs.110%29.aspx

Unboxing

Unboxing simply extracts the value type from the object or the interface type. In contrast to boxing, the cast required for unboxing is not as expensive computationally. Unboxing is a conversion from the type Object to a value type or from an interface type to a value type that implements the interface.

An unboxing operation consists of the following:

- Determining if the value to be unboxed is a boxed value.
- Copying the value from the instance into the value type variable.

A simple example of **Unboxing** is as follows:

```
1. Dim i As Integer
2. Dim o As Object
3. o = 123
4. i = DirectCast(o, i)
```

Here, the numeric value is initially of Type Object; then, it gets converted to an Integer value. For more information on Reference types, have a read through here:

https://msdn.microsoft.com/en-us/library/aa711899%28v=vs.71%29.aspx

For more information regarding all the **.NET Framework** types, have a look through these:

https://msdn.microsoft.com/en-us/library/k4tx24as%28v=vs.90%29.aspx

Our Project

Let's do a proper example. Open Visual Basic and create a new Console application.

Declare the following variables:

```
1. ' Integer
2.    Dim _intvalue As Integer = 15
3.
4.    ' object type
5.    Dim _Obj As [Object] = 200
```

intValue gets created with the value of 15 and as an Integer value type. Obj gets created as an Object type with the value of 200 stored inside it.

Add the following code to **Box** a variable:

```
1.    ' Boxing
2.    _Obj = DirectCast(_intvalue, [Object])
3.
4.    System.Console.WriteLine(_Obj)
5.
6.    System.Console.Read()
```

Once run this, you will see the value 15 displayed. Why? Well, because the Obj object now contains the boxed value of the intValue variable.

Add the following code to **Unbox** the value:

```
1.    ' UnBoxing
2.    _intvalue = CInt(_Obj)
3.
4.    System.Console.WriteLine(_intvalue)
```

```
5.
6.     System.Console.Read()
```

The Obj value gets unboxed back into intValue and now again holds the value of 15. Your full code looks like the following:

```
1. Imports System
2.
3. Imports System.Collections.Generic
4. Imports System.Text
5. Module Module1
6.
7.     Sub Main()
8.         ' Boxing
9.
10.        ' Integer
11.        Dim _intvalue As Integer = 15
12.
13.        ' object type
14.        Dim _Obj As [Object] = 200
15.
16.        ' Boxing
17.        _Obj = DirectCast(_intvalue, [Object])
18.
19.        System.Console.WriteLine(_Obj)
20.
21.        System.Console.Read()
22.
23.        ' UnBoxing
24.        _intvalue = CInt(_Obj)
25.
26.        System.Console.WriteLine(_intvalue)
27.
28.        System.Console.Read()
29.
30.    End Sub
31.
32. End Module
```

When run, your screen will look like Figure 1:

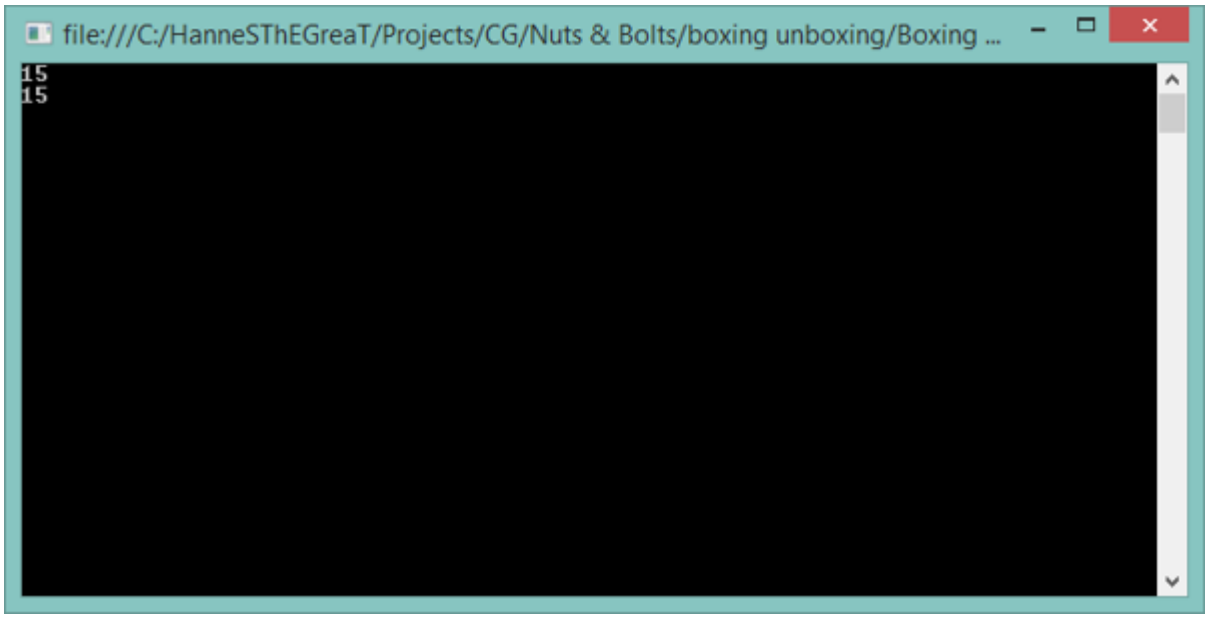

Figure 1: The results of our program

This working sample is included with this article.

Conclusion

Short, but to the point. I. Hope you have enjoyed today's article. Until next time, cheers!

Downloads

- Boxing and Unboxing.zip