InjectCallableConversions

See the previous example as it has been processed in the steps ProcessMethodBodiesWithDuckTyping, ProcessSharedLocals, and ProcessClosures.

An adapter class is created for event handlers. This allows us to say for example "button.Click += MyMethod" without having to manually cast to the correct event handler type.

```
import System.Windows.Forms from System.Windows.Forms

[Boo.Lang.ModuleAttribute]
public final transient class TempModule(System.Object):

 private static def Main(argv as (System.String)) as System.Void:
  ___locals = TempModule.___locals3()
  f = System.Windows.Forms.Form()
  ___locals.___clickcount_0 = 0
  b = __eval__(
                   (___temp1 = System.Windows.Forms.Button()),
                    ___temp1.set_Text('Hello'),
                    ___temp1.set_Dock(System.Windows.Forms.DockStyle.Fill),

                    ___temp1)
  b.add_Click(
                    ___adaptor0.Adapt(
                      ___callable0(
                          TempModule.___closure2(___locals),
                               __addressof__(self.Invoke))))
  f.get_Controls().Add(b)
  f.ShowDialog()

 private def constructor():
  super()

 class ___locals3(System.Object):

  internal ___clickcount_0 as System.Int32

  public def constructor():
   super()

 class ___closure2(System.Object):

  internal _____locals4 as TempModule.___locals3

  public def constructor(_____locals4 as TempModule.___locals3):
   super()
   self._____locals4 = _____locals4

  public def Invoke() as System.Void:
   Boo.Lang.Builtins.print(
                             "you clicked me ${(
                                  self._____locals4.___clickcount_0 = (
                                     self._____locals4.___clickcount_0 + 1)
```

```
                                )} times")

public final class ___callable0(System.MulticastDelegate,
Boo.Lang.ICallable):

 public def constructor(instance as System.Object, method as
System.IntPtr):
  pass

 public virtual def Call(args as (System.Object)) as System.Object:
  pass

 public virtual def Invoke() as System.Void:
  pass

 public virtual def BeginInvoke(callback as System.AsyncCallback,
                                    asyncState as System.Object) as
System.IAsyncResult:
  pass

 public def BeginInvoke(callback as System.AsyncCallback) as
System.IAsyncResult:
  return self.BeginInvoke(callback, null)

 public def BeginInvoke() as System.IAsyncResult:
  return self.BeginInvoke(null, null)

 public virtual def EndInvoke(result as System.IAsyncResult) as
System.Void:
  pass

internal final class ___adaptor0(System.Object):

 protected __callable as ___callable0

 public def constructor(from_ as ___callable0):
  super()
  self.__callable = from_

 public def Invoke(sender as System.Object, e as System.EventArgs) as
System.Void:
  self.__callable.Invoke()

 public static def Adapt(from_ as ___callable0) as System.EventHandler:
```

```
    return System.EventHandler(
                         ___adaptor0(from_), __addressof__(self.Invoke))
```