Sample GTK-SHARP application.

Summary

The following is a sample GTK# application written in Boo. Unlike some applications, this one actually does something useful - its an image (porn?) browser, and its cross platform to boot!

The code, with comments, comes in just over 100 lines. You will need the glade file below (make sure to save it on whatever output directory you compile the program to).

Thanks to incomp from IRC for spiffying up the sample a bit.

Prerequisites

GTK# Runtime
There are two different runtimes for windows users: the "Win32 Installer" and the "Win32 Runtime Installer." As poorly named as they are, all you need to know is that "Win32 Installer" is for developers and "Win32 Runtime Installer" is for end-users. Guess which one you are.

Mono and GTK# Runtime if you're running under Linux.

The Code.

```
import System
import System.IO
import Gtk from "gtk-sharp"
import Gdk from "gdk-sharp"
import Glade from "glade-sharp"

class PornBrowser():
 public static final phile = """pornbrowser.glade"""

 //Unfortunately, Gtk does not provide a builtin "IsSupported" function for images.
 public static final supportedExtensions = ( ".png", ".jpeg", ".jpg", ".gif", ".tga",
".bmp", ".ico", ".tiff" )

 def constructor():
  //Connect event handlers and mainPorn, aboutDialog, etc, to their respective
methods/handlers.
  for widget in ("mainPorn", "aboutDialog", "directorySelector"):
   gxml = Glade.XML(phile, widget, null)
   gxml.Autoconnect(self)

 def OnPornClose(sender as object, e as DeleteEventArgs):
 """The user is leaving the application via the Big Red 'X'"""
  Application.Quit()

 def OnPornQuit(sender as object, e as EventArgs):
 """The user is leaving the application via File-->Quit"""
  Application.Quit()

 def OnAboutActivate(sender as object, e as EventArgs):
 """The user is trying to view the about dialog"""
  aboutDialog.Show()

 def OnAboutDelete(sender as object, e as DeleteEventArgs):
  e.RetVal = true
  aboutDialog.Hide()
```

```
 def OnAboutResponse(sender as object, e as ResponseArgs):
  if e.ResponseId == ResponseType.Close:
   aboutDialog.Hide()

 def OnOpenDirectory(sender as object, e as EventArgs):
  #changed
  directorySelector.FileList.Sensitive = false
  directorySelector.Show()

 def OnDirectoryDelete(sender as object, e as DeleteEventArgs):
  e.RetVal = true
  directorySelector.Hide() //Don't worry, the "Destroy" event will kill it.

 def OnDirectoryResponse(sender as object, args as ResponseArgs):
  directorySelector.Hide()
  #changed
  if args.ResponseId == ResponseType.Ok and directorySelector.Filename.Length != 0:
   LoadDirectory(directorySelector.Filename)

 private def LoadDirectory(path as string):
  //Create a new store that has two columns; the first is a Pixbuf (image) the second,
a string.
  store = ListStore(Gdk.Pixbuf, string)
  pornThumbs.Model = store
  cell = CellRendererPixbuf()
  column = TreeViewColumn()
  column.PackStart(cell, false)

  //Bind the element in column 0 of the store to the pixbuf property in this column's
CellRendererPixbuf!
  column.AddAttribute(cell, "pixbuf", 0)
  column.Title = path

  //Detach the old column.
  if pornThumbs.GetColumn(0) is not null:
   pornThumbs.RemoveColumn(pornThumbs.GetColumn(0))

  //Finally, attach the new column so it will be updated in real time.
  pornThumbs.AppendColumn(column)
  files = Directory.GetFiles(path)
  validFiles = e for e as string in files if System.IO.FileInfo(e).Extension in
PornBrowser.supportedExtensions
  parse = do():
   for i as int, file as string in enumerate(validFiles):
    //The using construct ensures that the program's memory size does not inflate
wildly out of control.
    //Having these "image" floating around afterwards ensures lots of memory consuming
havoc until garbaged collected!
    #changed
    // If we couldn't read a file pass over it.
    try:
     using image = Gdk.Pixbuf(file):
      store.AppendValues( (image.ScaleSimple(128, 128, InterpType.Bilinear), file ) )
    except e:
     print " Couldn't read file $file "

  print "Indexing start: $(DateTime.Now)"
  //Update the treeview in real time by starting an asynchronous operation.
  parse.BeginInvoke( { print "Indexing complete: $(DateTime.Now)" } ) //Spin off a new
```

```
thread; fill the progress bar.

 def OnPictureActivated(sender as object, args as RowActivatedArgs):
  //Grabs the TreeIter object that represents the currently selected node -- icky
stuff with "ref" keyword. ;(
  x as TreeIter
  pornThumbs.Model.GetIter(x, args.Path)

  //In the second column we've stored the filename that represents the thumbnail image
in the treeview.
  //Grab this filename and then use it to inform the user what file he's viewing, as
well as loading it.
  image = pornThumbs.Model.GetValue(x, 1) as string
  pornImage.Pixbuf = Gdk.Pixbuf(image)
  imageName.Text = Path.GetFileName(image)

 //All of the widgets we want Autoconnect to hook up to Glade-created widgets in
pornBrowser.glade
 [Glade.Widget] aboutDialog as Gtk.Dialog
 [Glade.Widget] mainPorn as Gtk.Window
 [Glade.Widget] directorySelector as Gtk.FileSelection
 [Glade.Widget] pornImage as Gtk.Image
 [Glade.Widget] pornThumbs as Gtk.TreeView
 [Glade.Widget] imageName as Gtk.Label

//Run the application.
Application.Init()
```

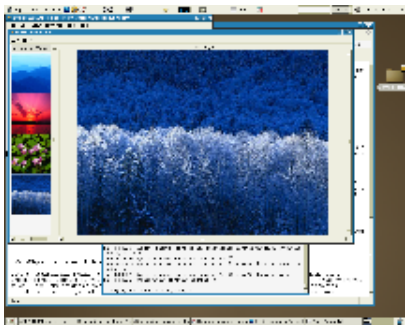
```
PornBrowser()
Application.Run()
```

The Glade File

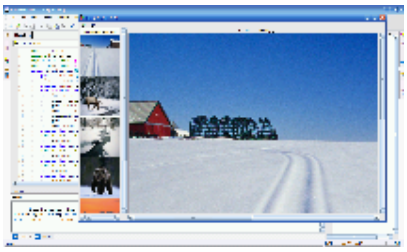
Pornbrowser.glade is the mark-up file that represents your fancy user interface ,built entirely in Glade. Put it in the output directory of the sample application (where the .exe resides).

The Screenshots

Amazingly, this application runs under Windows and Linux – without recompiling! Who needs Java anymore? 😊



The sample application running under Linux (no modifications required!)



A screenshot of the sample application running under Windows.