Ruby GUI

احنا هنتكلم عن تصميم الوجهات الرسومية بإستخدام روبى وعدة toolkits

اولا

GTK

GTK هى library مختصة بتصميم الواجهات الرسومية GUI وانشئت بالأساس لتطوير GNU Image Manipulation Program او GIMP واسمها اختصار ل GIMP Toolkit
حاليا دخلت فى مشروعات كتيرة مش ال GIMP فقط زى تطوير GNOME GNU Network Object Model Environment وعدد كبير من البرامج

للمزيد تقدر تراجع مقالة وادى التقنية

http://itwadi.com/what_is_gtk

ومراجعة الموقع الرسمى

http://gtk.org

اولا التستيب

http://sourceforge.net/project/showfiles.php?group_id=53614

hello world

اولا عايزين ننشئ ويندو مشابهة لدي كبداية

الكود

```
require 'gtk2'
win=Gtk::Window.new   Gtk::Window::TOPLEVEL
win.show
Gtk.main
```

1- هنستدعى ال gtk ب

```
require 'gtk2'
```

2- هننشئ اوبجكت من ال Window class ونحدد انها TOPLEVEL مش POPUP

```
win=Gtk::Window.new   Gtk::Window::TOPLEVEL
```

3-نطلب عرض الويندو بإستخدام ال show method

```
win.show
```

4- نخلى ال GTK تبدأ حاجة اسمها ال main loop ودى خاصة بجعل ال GTK مستعدة للتعامل مع اى ايفينت يحصل للبرنامج "طبعا انت بتحدد طريقة التعامل "

hello world2

عيزين نعمل نافذة وجواها button زرا مكتوب عليه Hello, World
بكل بساطة نفس الكود السابق ولكن هنضيف اوبجكت من ال Button class

```
require "gtk2"

btn=Gtk::Button.new "Hello, World!"
win=Gtk::Window.new   Gtk::Window::TOPLEVEL
```

نضيف ال btn لل win بإستخدام ال add method

```
win.add btn
```

نستدعى ال show_all method لعرض كل الكنترولز الموجودة داخل ال container "ال win"

```
win.show_all
```

ندخل ال gtk فى ال main loop

```
Gtk.main
```

لتعديل العنوان الخاص بال win عدله بإستخدام ال title=

```
win.title="Hello!"
```

ليصبح كدة

وتقدر تعدل ال border_width عن طريق border_width=

```
win.border_width=15
```

جميل مثالنا دة بس ياريت بيعمل حاجة!؟ معقولة فى حد يدوس على button ومش يحصل اى ناتج ؟
تمام جدا
اول ماهيتم الضغط على ال button ال gtk هتستقبل event وليكن بإسم clicked واذا انت مش حددت callback "ميثود يتم تنفيذها عند حدوث الإيفنت" ال gtk هتتجاهلها
ملحوظة احيانا بيسمى ال event handler واحيانا ال بيتقال عليه slot وعلى ال ايفينت اسم signal فإحنا هنمشى بالتسمية الأصلية وهى ال signal/slot ال signal هى clicked مثلا وال slot هى ال callback اللى هيتم استدعائها

خلاص تمام تعالى ننفذ الكلام دة
احنا الأول عايزين نكتب ال callback اللى هتحصل

```
#declare on_btn_clicked callback
```

```
def on_btn_clicked (win)
  dialog = Gtk::MessageDialog.new(win,
                        Gtk::Dialog::DESTROY_WITH_PARENT,
                        Gtk::MessageDialog::INFO,
                        Gtk::MessageDialog::BUTTONS_OK,
                        "Hello World!")
  dialog.run
  dialog.destroy
end
```

احنا هنا
1- انشانا اوبجكت من ال MessageDialog classوحددنا
1- ال parent بتاع المسج
2- انه هيتقفل مع ال parent
3- يظهر ايكون INFO
4- الزراير اللى هتظهر هى ال OK بس
5- التكست بتاع الرسالة

وبعد كدا dialog.run لعرض الديالوج وبعد مالمستخدم يختار اختياره يتم قفله عن طريق ال destroy method
نربط ال callback دى بال btn object عن طريق ال signal_connect method

```
#connect clicked signal to "on_btn_clicked"
btn.signal_connect("clicked"){
  on_btn_clicked(win)
}
```

اللى هينتج لما تيجى تضغط على ال button دا

لما بنيجى نقفل ويندو معينة بيوصل لل gtk سيجنال بإسم delete-event
ودى هنخليها تعمل ريترن ب false معناها ان معالجة الإيفنت دا مش انتهت

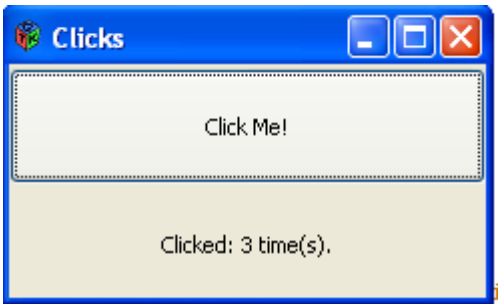
لسة فيتعمل raise لل destroy signal

```
win.signal_connect("delete-event"){
  false #processing aint done yet. "destroy" signal get raised.
  #true #will stop processing closing signal

}
```

وبعدها بيوصل سيجنال بإسم destroy ودى بنربطها بقفل الأبلكيشن نفسه من خلال ميثود بإسم Gtk.main_quit

```
win.signal_connect("destroy"){
  Gtk.main_quit
}
```

Clicks



لاحظ ان الويندو متقسمة لجزئين راسي اول جزء فيه button والتانى فيه label بيتكتب عليه عدد مرات الضغط على ال button
احنا ممكن نكتبها بالطريقة المعتادة وممكن نكتبها بطريقة كتير افضل بإستخدام ال OOP
تمام الأول عشان نخليهم متقسمين فى شكل معين افقى او رأسى بنستخدم Hbox (اختصار ل Horizontal Box) او Vbox (اختصار ل Vertical Box) تمام ؟ قشطة

ال vertical box بياخد ال widgets او بوكسز تانية فى صورة rows صفوف

ال horizontal box بياخد ال widgets او البوكسز التانى فى صورة Columns عواميد

لإنشاء box بتبدأ اولا ب homogeneous ودى معناها هل كل الأجزء متساوية فى العرض او الطول ليها ترجمة بإسم متجانسة اعتقد مناسبة ؟ والمعامل التانى لتحديد عرض الفاصل

@vbox=Gtk::VBox.new(false, 2)

packing:

pack_start(child, expand, fill, padding)

لوضع ال widget من الشمال لليمين او من فوق لتحت "صورة فطرية!"

pack_end(child, expand, fill, padding)

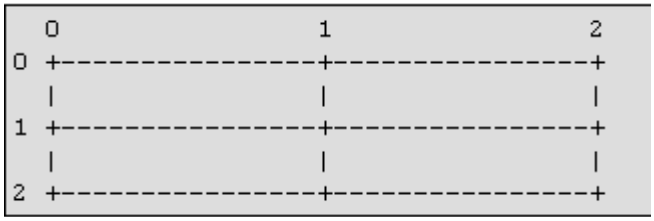
لوضع ال widget من اليمين للشمال او من تحت لفوق وهى موجود لل Hbox وال Vbox

expand: هل عايزه يكبر مع اى زيادة فى حجم الويندو؟

fill: فى حال التصغير هل يتم اخفاء جزء منه؟وليس تصغيره

padding: الهامش حوله

-استخدام الجداول



لإنشاء جدول بننشئه كالتالى

```
Gtk::Table.new( rows, columns, homogeneous )
```

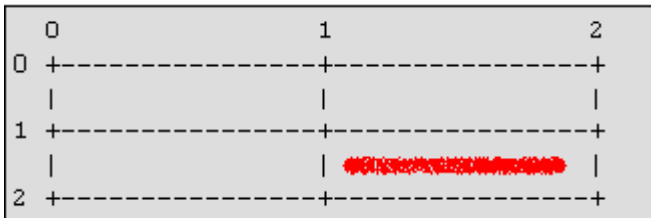
عدد الصفوف وعدد الأعمدة وهل متجانسين او لأ

لإضافة child بإستخدام ال attach method

```
.attach( child, left_attach, right_attach,
```

```
top_attach, bottom_attach, xoptions, yoptions,
xpadding, ypadding)
```

child: هو اللى هيتم اضافته فى الجدول
left_attach : العمود على يسار المكان
right_attach: العمود على يمين المكان
top_attach: الصف فوق المكان
bottom_attach: الصف تحت المكان
مثال للتوضيح



لو عايزين نحط widget معين فى الكورنر اليمين من جدول زى مانت شايف 2X2
بيقع بين الخطين الرأسين 1و 2 وهما دول ال right_attach, left_attach
بيقع بين الخطين الأفقيين 1 و 2 وهما دول ال top_attach, bottom_attach
xoptions: الرولز ل x
Gtk::FILL: لو الحدول اكبر من الويدجت فالويدجت هيتمدد ليشغل المساحة
Gtk::EXPAND :هنا الجدول هيتمدد اذا كان فى مساحة فى ال window
Gtk::SHRINK: اذا تم تصغير المساحة المتاحة للويدجت "مع تصغير الجدول مثلا" هيتم تصغيره

yoptions: الرولز ل y مشابهه ل x
xpadding: الهامش من ناحية ال x
ypadding: الهامش من ناحية ال y
لو انت مرضى مع الخيارات الأساسية تقدر تستخدم attach_defaults

```
attach_defaults( child, left_attach, right_attach,
top_attach, bottom_attach )
```

ودى هتخليك تدخل ال left, right, top, bottom attach فقط والباقى هيكون بالديفولت لل x,y options
Gtk::FILL | Gtk::EXPAND
وال x,y padding هيكون 0

فين ال fixed ؟ موجود fixed لعرض ال ويدجتس بتحديد ال مكان على الفورم ولكن "استخدام السابق افضل من حيث حماية طريقة وضعك لل ويدجتس من حيث التمدد والإنكماش وكدا"

دا هو الكود

```
require "gtk2"

class Win < Gtk::Window
  def initialize
    super Gtk::Window::TOPLEVEL #call super class constructor.

    @count=0
    @btn=Gtk::Button.new "Click Me!"
    @btn.signal_connect("clicked"){
      on_btn_clicked
    }
    @lbl=Gtk::Label.new
    @vbox=Gtk::VBox.new(false, 2)
    @vbox.pack_start(@btn, true, true, 0)
    @vbox.pack_start(@lbl, true, true, 0)

    add(@vbox)

    signal_connect("delete-event"){
      false
    }
    signal_connect("destroy"){
      Gtk.main_quit
    }

  end

  def on_btn_clicked
      @count+=1
      @lbl.label="Clicked: #{@count} time(s)."
  end
```

```
end


w=Win.new
w.title="Clicks"

w.show_all

Gtk.main
```

الفكرة اننا
1- نشتق كلاس جديد من ال GTK Window
2- نحدد ال ال widgets اللى هتتضاف على صورة instance variables عشان كدة عندنا وهنا هما @lbl و @btn
3- نربط ال clicked signal لل btn ب on_btn_clicked slot ونخزن عدد مرات الضغط عليه فى متغير بإسم @count مثلا ونعدل التكست الظاهر على ال@lbl (كل دا فى ال callback)

*ملحوظة تقدر تظهر markup فى ال label بإستخدم set_markup

```
 def on_btn_clicked
    @count+=1
    @lbl.label="Clicked: #{@count} time(s)."
 end
```

الربط

```
  @btn.signal_connect("clicked"){
   on_btn_clicked
  }
```

الهيكلية

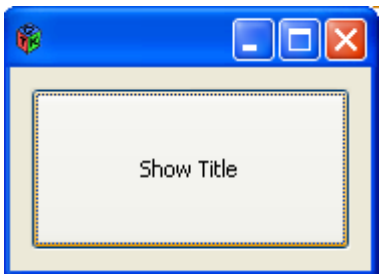
```
GObject
 |
 GtkObject
     +GtkWidget
     |    +GtkMisc
     |    |    +GtkLabel
     |    |    |    `GtkAccelLabel
     |    |    +GtkArrow
     |    |         `GtkImage
     |    +GtkContainer
     |    |    +GtkBin
     |    |    |    +GtkAlignment
```

```
|     |     |     +GtkFrame
|     |     |     |     `GtkAspectFrame
|     |     |     +GtkButton
|     |     |     |     +GtkToggleButton
|     |     |     |     |     `GtkCheckButton
|     |     |     |     |          `GtkRadioButton
|     |     |     |     `GtkOptionMenu
|     |     |     +GtkItem
|     |     |     |     +GtkMenuItem
|     |     |     |       +GtkCheckMenuItem
|     |     |     |       |     `GtkRadioMenuItem
|     |     |     |       +GtkImageMenuItem
|     |     |     |       +GtkSeparatorMenuItem
|     |     |     |       `GtkTearoffMenuItem
|     |     |     +GtkWindow
|     |     |     |     +GtkDialog
|     |     |     |     |     +GtkColorSelectionDialog
|     |     |     |     |     +GtkFileSelection
|     |     |     |     |     +GtkFontSelectionDialog
|     |     |     |     |     +GtkInputDialog
|     |     |     |     |     `GtkMessageDialog
|     |     |     |     `GtkPlug
|     |     |     +GtkEventBox
|     |     |     +GtkHandleBox
|     |     |     +GtkScrolledWindow
|     |     |     `GtkViewport
|     |     +GtkBox
|     |     |     +GtkButtonBox
|     |     |     |     +GtkHButtonBox
|     |     |     |     `GtkVButtonBox
|     |     |     +GtkVBox
|     |     |     |     +GtkColorSelection
|     |     |     |     +GtkFontSelection
|     |     |     |     `GtkGammaCurve
|     |     |     `GtkHBox
|     |     |          +GtkCombo
|     |     |          `GtkStatusbar
|     |     +GtkFixed
|     |     +GtkPaned
|     |     |     +GtkHPaned
|     |     |     `GtkVPaned
|     |     +GtkLayout
|     |     +GtkMenuShell
|     |     |     +GtkMenuBar
|     |     |     `GtkMenu
|     |     +GtkNotebook
|     |     +GtkSocket
|     |     +GtkTable
|     |     +GtkTextView
|     |     +GtkToolbar
|     |     `GtkTreeView
|     +GtkCalendar
|     +GtkDrawingArea
|     |     `GtkCurve
|     +GtkEditable
|     |     +GtkEntry
|     |       `GtkSpinButton
|     +GtkRuler
|     |     +GtkHRuler
```

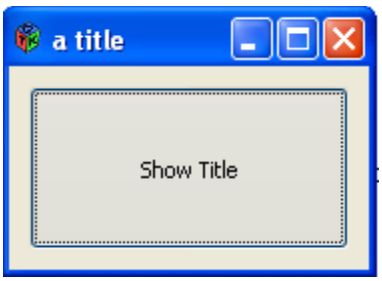
```
|     |       `GtkVRuler
|     +GtkRange
|     |     +GtkScale
|     |     |     +GtkHScale
|     |     |     `GtkVScale
|     |     `GtkScrollbar
|     |        +GtkHScrollbar
|     |        `GtkVScrollbar
|     +GtkSeparator
|     |     +GtkHSeparator
|     |     `GtkVSeparator
|     +GtkInvisible
|     +GtkPreview
|     `GtkProgressBar
+GtkAdjustment
+GtkCellRenderer
|     +GtkCellRendererPixbuf
|     +GtkCellRendererText
|     +GtkCellRendererToggle
+GtkItemFactory
+GtkTooltips
`GtkTreeViewColumn
```

Toggle Button

مشابه لل check box ودا ليه حالتين True او False



هنا وهو مش متنشط يعنى False



هنا هو متنشط يعنى active

اول مايداس عليه بيبعت toggled signal لل Gtk main loop فإحنا هنهندله ب callback ولتكن on_toggled مثلا كالتالى

```
def on_toggled
  if self.title==" "
    self.title="a title"
```

```
    else
      self.title=" "
    end
```

ونربطها بيه

```
    @tog_btn.signal_connect("toggled"){
      on_toggled
    }
```

كود المثال

```
class Win < Gtk::Window
  def initialize
    super Gtk::Window::TOPLEVEL
    self.border_width=10
    self.title=" "
    @tog_btn=Gtk::ToggleButton.new "Show Title"
    @tog_btn.signal_connect("toggled"){
      on_toggled
    }
    add @tog_btn

  end

  def on_toggled
    if self.title==" "
      self.title="a title"
    else
      self.title=" "
    end

  end
end
```

CheckButton

مشابة للسابق ولكن بشكل مختلف بعض الشئ

```
class Win < Gtk::Window
  def initialize
    super Gtk::Window::TOPLEVEL
    self.border_width=10
    self.title=" "
    @chk_btn=Gtk::CheckButton.new "Show Title"
    @chk_btn.signal_connect("toggled"){
      on_toggled
    }
    add @chk_btn

  end

  def on_toggled
    if self.title==" "
      self.title="a title"
    else
      self.title=" "
    end

  end
end

w=Win.new
w.show_all
Gtk.main
```

Adjustment

هى مش ويدجت ولكن بيستخدم فى تخزين ونقل معلومات محددة لإعدادات ويدجتس معينة زى السكرولبارز والسبينرز والراينجز وغيرهم

لعمل adjustment object بننشئه كالتالى

```
Gtk::Adjustment.new( value, lower, upper,
step_increment, page_increment, page_size )
```

تقدر تعتبرها ك model لويدجت وهو بيعرضهالك فى ال view بتاعته على كل حال هنشوف

ال value القيمة الأساسية

ال lower اقل قيمة

ال upper اعلى قيمة

ال step_increment مقدار الزيادة

Scale

فى منها افقى وفي رأسى لإنشاء الأوبجكتس منها

```
Gtk::VScale.new( adjustment )

Gtk::VScale.new( min, max, step )

Gtk::HScale.new( adjustment );

Gtk::HScale.new( min, max, step );
```

ياإما تباصى adjustment او تباصى اقل واكبر قيمة والزيادة

لإظهار القيمة مع ال scale او لأ تقدر تستخدم draw_value او set_draw_value وتديهم قيمة true او false فى حالة الإخفاء

-- هى true افتراضيا..

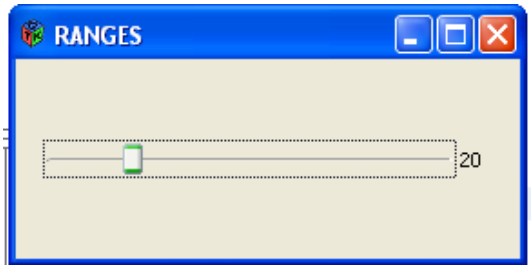
digits/set_digits لتحديد عدد الأرقام بعد العلامة العشرية اللى عايزها تظهر

value= pos

set_value_pos(pos)

تقدر تستخدمهم لتحديد المكان اللى هيظهر عليه قيمة ال value ودى بتاخد القيم

```
Gtk::POS_LEFT
Gtk::POS_RIGHT
Gtk::POS_TOP
Gtk::POS_BOTTOM
```



مثلا لعمل المثال دا هننشئه كالتالى

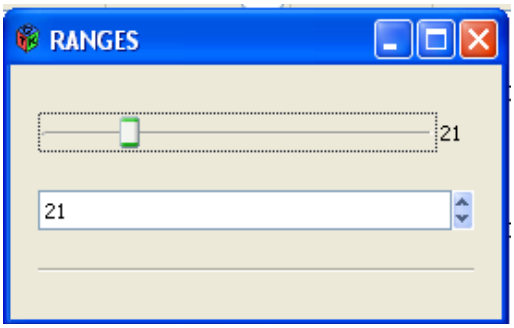
```
class Win<Gtk::Window
  def initialize
    super
    self.title="RANGES"
    @vbox=Gtk::VBox.new(false, 0)
    #@adj=Gtk::Adjustment.new(20, 0, 101, 1,1,1)
    #@hsc=Gtk::HScale.new(@adj)
    @hsc=Gtk::HScale.new(0, 101, 1)
    @hsc.digits=0
    @hsc.set_value_pos(Gtk::POS_RIGHT)
    @vbox.pack_start(@hsc, true, true, 0)

    add(@vbox)
    self.border_width=15

    signal_connect("delete-event"){
      false
    }
    signal_connect("destroy"){
      Gtk.main_quit
    }
    show_all

  end
end
```

update_policy

بتحدد امتة يتم تعديل ال value بال adjustment الخاصة بال range widget وترسل ال value_changed signal بتاخد قيم مثل Gtk::UPDATE_CONTINUOUS

بترسل عند حدوث اقل تغيير ممكن فى ال range
Gtk::UPDATE_DISCONTINUOUS
بترسل لما المستخدم يسيب الماوس ويكون ال range ثابت
Gtk::UPDATE_DELAYED

بترسل بمجرد ان المستخدم يسيب الماوس او يتوقف عن الحركة لفترة صغيرة
وبيتم التحكم فى ال policy من خلال
.set_update_policy(up_policy)
.update_policy=(up_policy)
للحصول على ال adjustment الخاصة بال range استخدم
.adjustment
لتعديل ال adjustment استخدم
.set_adjustment(adj)
.adjustment=(adj)



نيجى لمثال تانى هنا عندنا Scaleو SpinButton عايزين نربطهم ان لما يتغير قيمة اى منهم يتعدل فى التانية
هنا هنستخدم ال adjustment object للإتنين بحيث ان يتعدل قيمة ال value فيها للإتنين (بما انها ال model اللى بيعرضه كل من ال Scale, SpinButton)

كود المثال

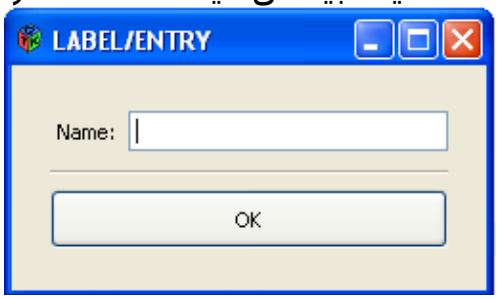
```
class Win<Gtk::Window
  def initialize
    super
    self.title="RANGES"
    @vbox=Gtk::VBox.new(false, 0)
    @adj=Gtk::Adjustment.new(20, 0, 101, 1,1,1)
    @hsc=Gtk::HScale.new(@adj)
    #@hsc=Gtk::HScale.new(0, 101, 1)
    @hsc.digits=0
```

```
    @hsc.set_value_pos(Gtk::POS_RIGHT)
    @hsep=Gtk::HSeparator.new

    @spin=Gtk::SpinButton.new(@adj)

    @vbox.pack_start(@hsc, true, true, 10)
    @vbox.pack_start(@spin, true, true,10)
    @vbox.pack_start(@hsep, true, true, 10)

    add(@vbox)
    self.border_width=15

    signal_connect("delete-event"){
      false
    }
    signal_connect("destroy"){
      Gtk.main_quit
    }

    show_all
  end
end
```

Label/Entry

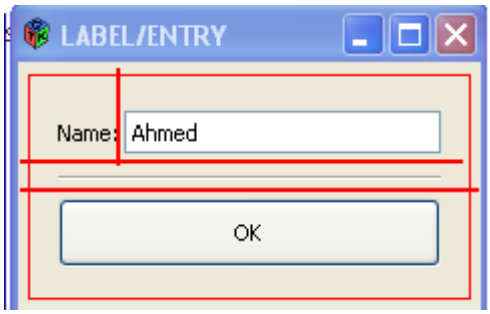
ال Label بيستخدم لعرض تكست ما بدون الحاجة لتغييره من المستخدم

ال Text Entry مشابه لل تكست فيلد بيدخل فيه المستخدم قيمة مطلوبة مثلا...



هنا المستخدم المفروض يدخل تكست معين فى ال entry

اولا التصميم



ال Label و ال Entry هيكونو فى Hbox
هنضم ال Hbox السابق ل Vbox (ال Layout الرئيسى)
ونضم صف جديد فيه ال Hseparator
صف اخير يكون فيه ال btn
طيب نعمل كل دا

```
@lbl=Gtk::Label.new "Name:"
@entry=Gtk::Entry.new
@hbox=Gtk::HBox.new(false, 2)

@hbox.pack_start(@lbl, true, true, 0)
@hbox.pack_start(@entry, true, true, 0)

@btnok=Gtk::Button.new "OK"

@vbox=Gtk::VBox.new(false, 0)

@vbox.add @hbox

@hsep=Gtk::HSeparator.new

@vbox.pack_start(@hsep, true, true, 2)
@vbox.pack_end(@btnok, true, true, 2)

add(@vbox)
```

تمام نربط ال clicked signal الخاصة بال btnok بالميثود msg كالتالى

```
    def msg(name)
          md=Gtk::MessageDialog.new(self, Gtk::Dialog::DESTROY_WITH_PARENT,
                        Gtk::MessageDialog::INFO,
                        Gtk::MessageDialog::BUTTONS_OK,
                        "Hi #{name}"
                    )
      md.run
      md.destroy
    end
```

الكود للمثال

```
require "gtk2"

class Win<Gtk::Window
  def initialize
    super
    self.title="LABEL/ENTRY"
    #Better with tabels...

    self.border_width=20
    @lbl=Gtk::Label.new "Name:"
    @entry=Gtk::Entry.new
    @hbox=Gtk::HBox.new(false, 2)

    @hbox.pack_start(@lbl, true, true, 0)
    @hbox.pack_start(@entry, true, true, 0)

    @btnok=Gtk::Button.new "OK"

    @vbox=Gtk::VBox.new(false, 0)

    @vbox.add @hbox

    @hsep=Gtk::HSeparator.new

    @vbox.pack_start(@hsep, true, true, 2)
    @vbox.pack_end(@btnok, true, true, 2)

    add(@vbox)

    @btnok.signal_connect("clicked"){
       msg(@entry.text)
```

```
    }
    def msg(name)
          md=Gtk::MessageDialog.new(self, Gtk::Dialog::DESTROY_WITH_PARENT,
                          Gtk::MessageDialog::INFO,
                          Gtk::MessageDialog::BUTTONS_OK,
                          "Hi #{name}"
                      )
      md.run
      md.destroy
    end


    signal_connect("delete-event"){
      false
    }
    signal_connect("destroy"){
      Gtk.main_quit
    }
    show_all
  end

end

w=Win.new
Gtk.main
```

ال Label ليه وظايف كتيرة مثلا تحديد ال justification
تقدر تعدل ال text اللى عليه بإستخدام

```
.set_text(text)
.label=(text)
```

تقدر تخليه justified بإستخدام

```
.set_justify(justify_type)
.justify=(justify_type)
```

وبتاخد القيم

```
Gtk::JUSTIFY_LEFT
Gtk::JUSTIFY_RIGHT
Gtk::JUSTIFY_CENTER
Gtk::JUSTIFY_FILL
```

تقدر تدعم ال wrapping بإستخدام

.set_wrap(bool)

.wrap(bool)

GRWC(GTK/Ruby Word Count)

هنكتب برنامج WC ب ruby/GTK

اولا السرفس

```
class WordCount
  def initialize

  end

  def set_file(filename)
    begin
      @filename=filename
      @lines=File.readlines filename
      @txt=@lines.join
    rescue Exception => ex
      puts "File not found."
    end
  end

  def chars_count(count_spaces=true)
```

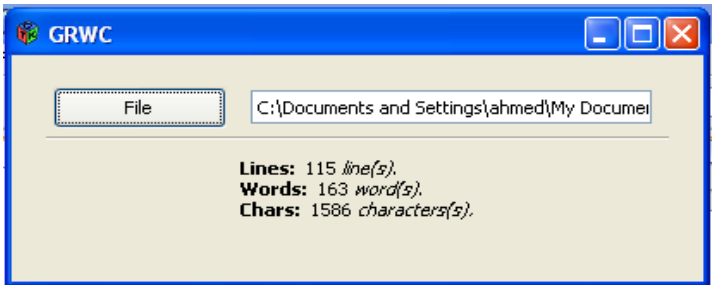
```
    if not count_spaces
      @txt.length
    else
      return @txt.gsub(/\s+/, '').length
    end
  end
  def words_count
    return @txt.split.length
  end
  def lines_count
    return @lines.length
  end

  def info_to_markup
    if @filename
      m= <<end
<b>Lines: </b> #{lines_count} <i>line(s)</i>.
<b>Words: </b> #{words_count} <i>word(s)</i>.
<b>Chars: </b> #{chars_count} <i>characters(s).</i>
end
    else
      return ''
```

```
    end
  end
end
```

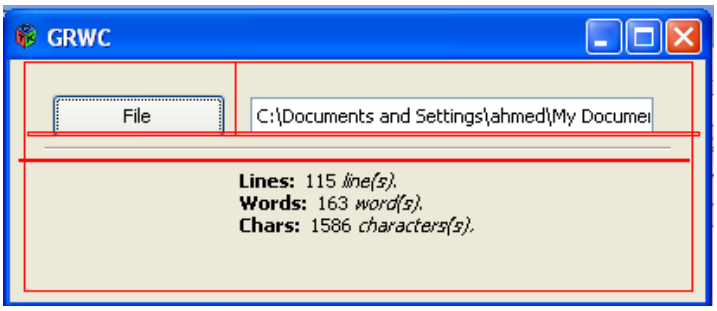
هنا فيه

info_to_markup

لنعرض فيها المعلومات بسهولة



تصميم الواجهة



التنفيذ

```
def init_comp
  self.title="GRWC"
  self.border_width=20
```

```
  @btn_file=Gtk::Button.new "File"
  @btn_file.signal_connect("clicked"){
    get_file
  }
  @entry_file=Gtk::Entry.new
  @entry_file.editable=false
  @hbox=Gtk::HBox.new(false, 0)
  @hbox.pack_start(@btn_file, true, true, 4)
  @hbox.pack_start(@entry_file, true, true, 10)
  @vbox=Gtk::VBox.new(false, 0)
  @vbox.add @hbox
  @hsep=Gtk::HSeparator.new
  @vbox.pack_start(@hsep, true, true, 4)
  @lblinfo=Gtk::Label.new
  @lblinfo.markup=@wc.info_to_markup
  @vbox.pack_start(@lblinfo, true, true, 4)
  add(@vbox)
```

ربطنا ال btn_file ب get_file ميثود المعرفة كالتالى

```
def get_file
      file_sel=Gtk::FileSelection.new("Choose File:")
      file_sel.ok_button.signal_connect("clicked"){
        @wc.set_file(file_sel.filename)
        @entry_file.text=file_sel.filename
        file_sel.hide
        @lblinfo.label=@wc.info_to_markup
      }
```

```
      file_sel.cancel_button.signal_connect( "clicked" ) { file_sel.hide }
      file_sel.show_all
  end
```

لإختيار ملف ما هننشئ اوبجكت من FileSelection

```
      file_sel=Gtk::FileSelection.new("Choose File:")
```

نربط ال ok_button فيه ب اللى احنا عايزينه وهنا ان نعمل set_file فى ال WordCount object ونعدل ال entry_file.text

```
      file_sel.ok_button.signal_connect("clicked"){
        @wc.set_file(file_sel.filename)
        @entry_file.text=file_sel.filename
        file_sel.hide
        @lblinfo.label=@wc.info_to_markup
      }
```

ونخفيه ونحط الناتج من wc.info_to_markup فى ال lblinfo.label

كود المثال

```
require "gtk2"
class WordCount
  def initialize
  end
  def set_file(filename)
    begin
      @filename=filename
      @lines=File.readlines filename
      @txt=@lines.join
    rescue Exception => ex
      puts "File not found."
    end
```

```
  end
  def chars_count(count_spaces=true)
    if not count_spaces
      @txt.length
    else
      return @txt.gsub(/\s+/, '').length
    end
  end



  def words_count
    return @txt.split.length
  end
  def lines_count
    return @lines.length
  end

  def info_to_markup
    if @filename
      m= <<end
<b>Lines: </b> #{lines_count} <i>line(s)</i>.
<b>Words: </b> #{words_count} <i>word(s)</i>.
<b>Chars: </b> #{chars_count} <i>characters(s).</i>
end
    else
      return ''
```

```
    end


  end
end


class Win<Gtk::Window
  def initialize
    super
    @wc=WordCount.new
    init_comp
    signal_connect("delete-event"){
      false
    }
    signal_connect("destroy"){
      Gtk.main_quit
    }
    show_all
  end
  def init_comp
    self.title="GRWC"
    self.border_width=20


    @btn_file=Gtk::Button.new "File"
    @btn_file.signal_connect("clicked"){
      get_file
    }
```

```
    @entry_file=Gtk::Entry.new
    @entry_file.editable=false
    @hbox=Gtk::HBox.new(false, 0)
    @hbox.pack_start(@btn_file, true, true, 4)
    @hbox.pack_start(@entry_file, true, true, 10)
    @vbox=Gtk::VBox.new(false, 0)
    @vbox.add @hbox
    @hsep=Gtk::HSeparator.new
    @vbox.pack_start(@hsep, true, true, 4)
    @lblinfo=Gtk::Label.new
    @lblinfo.markup=@wc.info_to_markup
    @vbox.pack_start(@lblinfo, true, true, 4)
    add(@vbox)
  end
  def get_file
      file_sel=Gtk::FileSelection.new("Choose File:")
      file_sel.ok_button.signal_connect("clicked"){
        @wc.set_file(file_sel.filename)
        @entry_file.text=file_sel.filename
        file_sel.hide
        @lblinfo.label=@wc.info_to_markup
      }
      file_sel.cancel_button.signal_connect( "clicked" ) { file_sel.hide }
      file_sel.show_all
  end
end
```

```
w=Win.new

Gtk.main
```

لسنترة الويندو تقدر تعدل ال window_position ل POS_CENTER

```
self.window_position=Gtk::Window::POS_CENTER
```

ComboBox

عبارة عن قايمة منسدلة فيها قيم معرفة مسبقا

لتخزين القايمة دى بنستخدم

.popdown_string=(a_list)

للحصول على القيمة الحالية منها

.entry.text

ال combo اهم سيجنال ليها هى ال changed signal

```
@cbo=Gtk::Combo.new

dists=["Slackware", "Ubuntu", "Fedora", "Debian"]

@cbo.popdown_strings=dists
```

تم ربط ال signal كالتالى

```
@cbo.entry.signal_connect("changed"){
  @lbl.label=@cbo.entry.text
}
```

كود المثال

```
require "gtk2"



class Win<Gtk::Window
  def initialize
    super

    init_comp

    signal_connect("delete-event"){
      false
    }
    signal_connect("destroy"){
      Gtk.main_quit
    }
```

```
  show_all

end
def init_comp
  self.title="COMBO"
  self.border_width=20
  self.window_position=Gtk::Window::POS_CENTER

  @vbox=Gtk::VBox.new(false, 0)

  @cbo=Gtk::Combo.new
  dists=["Slackware", "Ubuntu", "Fedora", "Debian"]
  @cbo.popdown_strings=dists
  @lbl=Gtk::Label.new
  @cbo.entry.signal_connect("changed"){
    @lbl.label=@cbo.entry.text
  }

  @vbox.pack_start(@cbo, true, true, 0)
  @vbox.pack_start(@lbl, true, true, 0)

  add(@vbox)
```

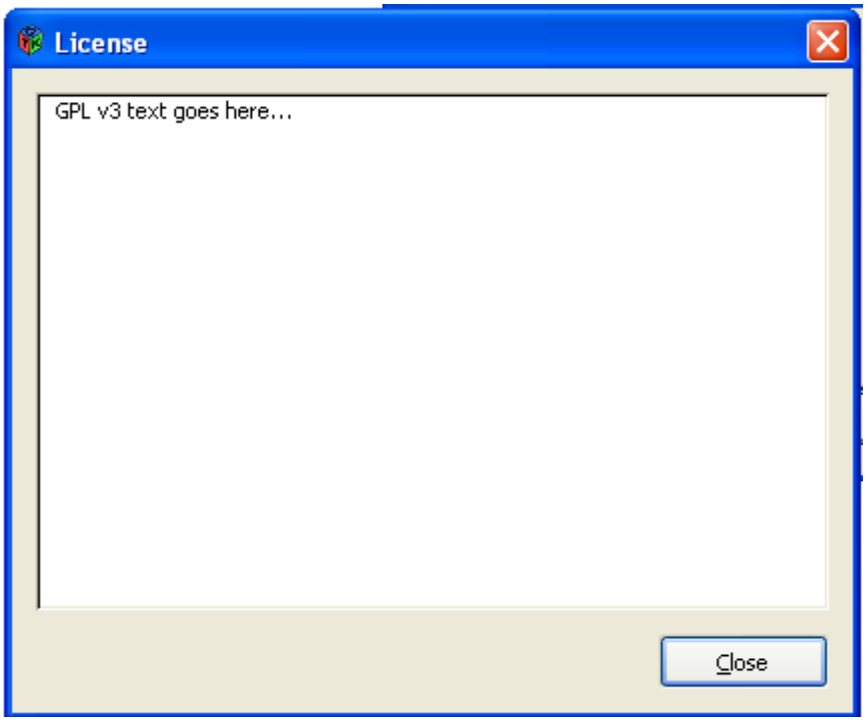
end

AboutDialog

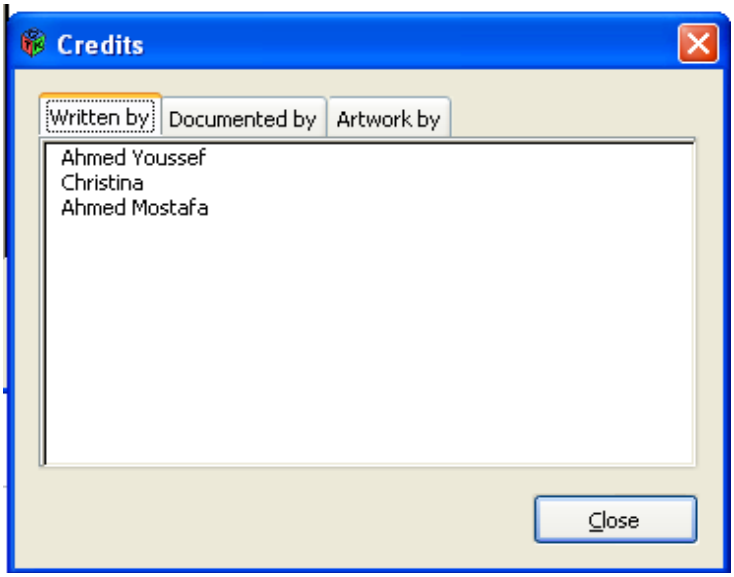
مهم جدا انك تكتب فى برنامج جزئية about فيها اسم البرنامج المبرمج والحقوق والرخصة واللوجو ورقم الإصدار والموقع الرسمى والكتاب والمترجمين والموثقين "كتبو التوثيق للبرنامج" واللى قامو بالتصاميم وهكذا

الصورة الأساسية

الرخصة "بتضيف تكست الرخصة"



اسم الكتاب والموثقين والرسامين وغيرهم هيكون موجود فى credits



تعالى نشوف اتكتب ازاى

اولا بتنشئ اوبجكت من ال Gtk::AboutDialog

```
aboutdlg=Gtk::AboutDialog.new
```

حدد اسم البرنامج

```
aboutdlg.name='RKONQ'
```

حدد رقم الإصدار

```
aboutdlg.version="0.9"
```

حدد اللوجو

```
pixbuf=Gdk::Pixbuf.new('firefox_logo.jpg', 50, 50)
aboutdlg.logo=pixbuf
```

حدد الحقوق

```
aboutdlg.copyright="(c) Ahmed Youssef"
```

التعليق على البرنامج

aboutdlg.comments="Simple WebBrowser"

الموقع الرسمى

aboutdlg.website="http://www.konq.org"

الرخصة

aboutdlg.license="GPL v3 text goes here..."

الكتاب

aboutdlg.authors=['Ahmed Youssef', 'Christina', 'Ahmed Mostafa']

الموثقين

aboutdlg.documenters=['Fahad', 'Zaid']

الفنانين

aboutdlg.artists=['Sami']

كود المثال

```
require "gtk2"

aboutdlg=Gtk::AboutDialog.new
aboutdlg.name='RKONQ'
aboutdlg.version="0.9"
pixbuf=Gdk::Pixbuf.new('firefox_logo.jpg', 50, 50)
aboutdlg.logo=pixbuf
aboutdlg.copyright="(c) Ahmed Youssef"
aboutdlg.comments="Simple WebBrowser"
```

aboutdlg.website="http://www.konq.org"

aboutdlg.license="GPL v3 text goes here..."

aboutdlg.authors=['Ahmed Youssef', 'Christina', 'Ahmed Mostafa']

aboutdlg.artists=['Sami']

aboutdlg.documenters=['Fahad', 'Zaid']

aboutdlg.run

aboutdlg.destroy

Frame

الفريم هو عبارة عن Container تقدر تضيفله label زى فى الصورة Distros او Langs

هنا عندنا MainVBOX وفيه 2 frames

الأول Distros والتانى Langs

ال Distros Frame فيه Vbox فيه صفين الأول Combo والتانى Label

ال Langs Frame فيه Vbox فيه صفين الأول Combo والتانى Label

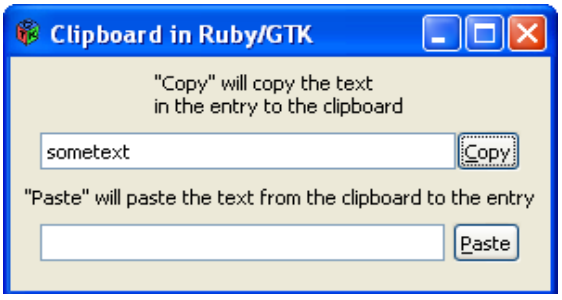
تمام ننفذ التصميم

```
def init_comp
   self.title="COMBO"
   self.border_width=20
   self.window_position=Gtk::Window::POS_CENTER

   @mvbox=Gtk::VBox.new(false, 0)


   @vbox1=Gtk::VBox.new(false, 0)
   @frame1=Gtk::Frame.new
```

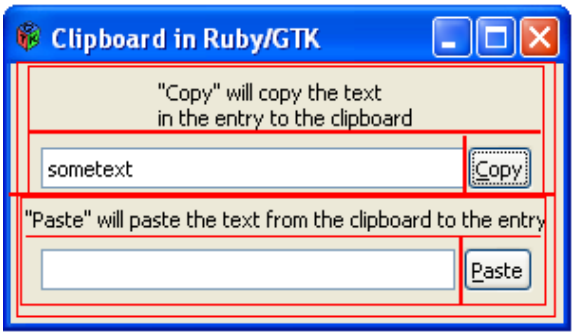
```
@frame1.label="Distros:"

@cbodists=Gtk::Combo.new
dists=["Slackware", "Ubuntu", "Fedora", "Debian"]
@cbodists.popdown_strings=dists
@lbl1=Gtk::Label.new

@cbodists.entry.signal_connect("changed"){
  @lbl1.label=@cbodists.entry.text
}
@vbox1.pack_start(@cbodists, true, true, 10)
@vbox1.pack_start(@lbl1, true, true, 10)

@frame1.add(@vbox1)

@mvbox.pack_start(@frame1, true, true, 5)

@vbox2=Gtk::VBox.new(false, 0)
@frame2=Gtk::Frame.new
@frame2.label="Langs:"
langs=["Ruby", "Python", "Perl", "Pascal"]
```

```
@cbolangs=Gtk::Combo.new
@cbolangs.popdown_strings=langs
@lbl2=Gtk::Label.new
@cbolangs.entry.signal_connect("changed"){
  @lbl2.label=@cbolangs.entry.text
}
@vbox2.pack_start(@cbolangs, true, true, 10)
@vbox2.pack_start(@lbl2, true, true, 10)
@frame2.add(@vbox2)

@mvbox.pack_start(@frame2, true, true, 0)
add(@mvbox)
end
```

Clipboard



تصميم الواجهة




```
  def init_comp

    self.title="CLIPBOARD"

    self.border_width=20

    self.window_position=Gtk::Window::POS_CENTER



    @vbox1=Gtk::VBox.new(false, 0)

    @lbl1=Gtk::Label.new %Q["Copy" will copy the text\nin the entry to the
clipboard]

    @hbox1=Gtk::HBox.new(false, 0)

    @e1=Gtk::Entry.new

    @bcopy=Gtk::Button.new("Copy")

    @bcopy.signal_connect("clicked", @e1){|w, e|

       clipboard = e.get_clipboard(Gdk::Selection::CLIPBOARD)

       clipboard.text = @e1.text



    }

    @hbox1.pack_start(@e1, true, true, 0)
```

```
    @hbox1.pack_start(@bcopy, true, true, 4)
    @vbox1.pack_start(@lbl1, true, true, 0)
    @vbox1.pack_start(@hbox1, true, true, 0)



    @vbox2=Gtk::VBox.new(false, 0)
    @hbox2=Gtk::HBox.new(false, 0)
    @e2=Gtk::Entry.new
    @bcut=Gtk::Button.new("Paste")
    @bcut.signal_connect("clicked", @e2){|w, e|
        clipboard = e.get_clipboard(Gdk::Selection::CLIPBOARD)
        clipboard.request_text {|board, text, data|
        @e2.text = text
      }
    }
    @lbl2=Gtk::Label.new %Q["Paste" will paste the text from the clipboard to
the entry]
    @hbox2.pack_start(@e2, true, true, 0)
    @hbox2.pack_start(@bcut, true, true, 4)


    @vbox2.pack_start(@lbl2, true, true, 0)
    @vbox2.pack_start(@hbox2, true, true, 0)
```

```
  @mvbox=Gtk::VBox.new(false, 0)

  @mvbox.pack_start(@vbox1, true, true, 0)
  @mvbox.pack_start(@vbox2, true, true, 0)

  add(@mvbox)

end
```

الربط الأول لعملية ال copy فى ال clipboard

```
  @bcopy.signal_connect("clicked", @e1){|w, e|
     clipboard = e.get_clipboard(Gdk::Selection::CLIPBOARD)
     clipboard.text = @e1.text

  }
```

الأول نحصل على ال clipboard

```
     clipboard = e.get_clipboard(Gdk::Selection::CLIPBOARD)
```

ونخزن فيها ال تكست الموجود فى @e1.text

الربط الثانى لعملية ال paste فى ال clipboard

```
  @bcut.signal_connect("clicked", @e2){|w, e|
```

```
        clipboard = e.get_clipboard(Gdk::Selection::CLIPBOARD)
        clipboard.request_text {|board, text, data|
        @e2.text = text
      }
    }
```

الكود

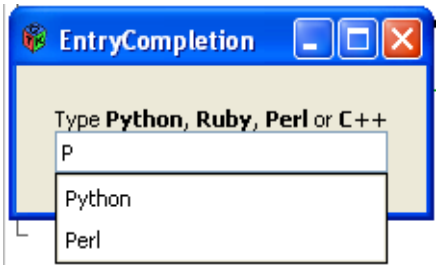
```
require "gtk2"



class Win<Gtk::Window
  def initialize
    super

    init_comp

    signal_connect("delete-event"){
      false
    }
    signal_connect("destroy"){
      Gtk.main_quit
```

```
    }
    show_all

  end
  def init_comp
    self.title="CLIPBOARD"
    self.border_width=20
    self.window_position=Gtk::Window::POS_CENTER



    @vbox1=Gtk::VBox.new(false, 0)
    @lbl1=Gtk::Label.new %Q["Copy" will copy the text\nin the entry to the clipboard]
    @hbox1=Gtk::HBox.new(false, 0)
    @e1=Gtk::Entry.new
    @bcopy=Gtk::Button.new("Copy")
    @bcopy.signal_connect("clicked", @e1){|w, e|
       clipboard = e.get_clipboard(Gdk::Selection::CLIPBOARD)
       clipboard.text = @e1.text

    }
    @hbox1.pack_start(@e1, true, true, 0)
```

```
    @hbox1.pack_start(@bcopy, true, true, 4)
    @vbox1.pack_start(@lbl1, true, true, 0)
    @vbox1.pack_start(@hbox1, true, true, 0)



    @vbox2=Gtk::VBox.new(false, 0)
    @hbox2=Gtk::HBox.new(false, 0)
    @e2=Gtk::Entry.new
    @bcut=Gtk::Button.new("Paste")
    @bcut.signal_connect("clicked", @e2){|w, e|
        clipboard = e.get_clipboard(Gdk::Selection::CLIPBOARD)
        clipboard.request_text {|board, text, data|
        @e2.text = text
      }
    }
    @lbl2=Gtk::Label.new %Q["Paste" will paste the text from the clipboard to
the entry]
    @hbox2.pack_start(@e2, true, true, 0)
    @hbox2.pack_start(@bcut, true, true, 4)


    @vbox2.pack_start(@lbl2, true, true, 0)
    @vbox2.pack_start(@hbox2, true, true, 0)
```

```
    @mvbox=Gtk::VBox.new(false, 0)

    @mvbox.pack_start(@vbox1, true, true, 0)
    @mvbox.pack_start(@vbox2, true, true, 0)

    add(@mvbox)

  end

end

w=Win.new

Gtk.main
```

EntryCompletion



ال GTK فى منها كتير مبنى على مبدأ ال MVC ولعمل عملية الإكمال التلقائى دى هتطلب منك ان يكون عندك ليست مخزنة بالفعل (model)

خلاص ننشئ المودل

```
def get_model
  lstore=Gtk::ListStore.new String
  ["Python", "Ruby", "C++", "Perl"].each{|w|
    iter=lstore.append
    iter[0]=w
  }
  return lstore
end
```

هنا المودل عبارة عن List من ال Strings بتضم "Python", "Ruby", "C++", "Perl"

ننشئ ال EntryCompletion كالتالى

```
  @ecompl=Gtk::EntryCompletion.new
```

نسندله ال model

```
  @ecompl.model=get_model
```

العمود اللى هيتم البحث فيه من ال model

```
  @ecompl.text_column = 0
```

ننشئ Entry

```
@entry=Gtk::Entry.new
```

نسند ليه ال completion object

```
@entry=Gtk::Entry.new
```

كود المثال

```
#
# To change this template, choose Tools | Templates
# and open the template in the editor.


require "gtk2"



class Win<Gtk::Window
  def initialize
    super

    init_comp

    signal_connect("delete-event"){
      false
```

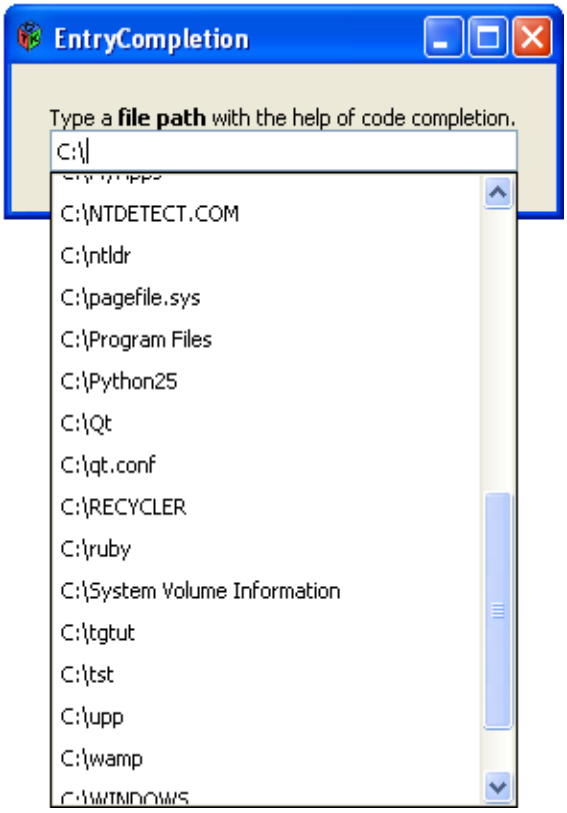
```
    }
    signal_connect("destroy"){
      Gtk.main_quit
    }
    show_all

  end
  def init_comp
    self.title="EntryCompletion"
    self.border_width=20
    self.window_position=Gtk::Window::POS_CENTER

    @mvbox=Gtk::VBox.new(false, 0)
    @lbl=Gtk::Label.new
    @lbl.set_markup("Type <b>Python</b>, <b>Ruby</b>, <b>Perl</b> or
<b>C++</b>")
    @mvbox.pack_start(@lbl, true, true, 0)
    @ecompl=Gtk::EntryCompletion.new
    @ecompl.model=get_model
    @ecompl.text_column = 0

    @entry=Gtk::Entry.new
```

```
    @entry.completion=@ecompl



    @mvbox.pack_start(@entry, true, true, 0)

    add(@mvbox)

  end

  def get_model
    lstore=Gtk::ListStore.new String
    ["Python", "Ruby", "C++", "Perl"].each{|w|
      iter=lstore.append
      iter[0]=w
    }
    return lstore
  end

end

w=Win.new
```

Gtk.main

المثال من ال gtk-demo

طيب عايزين نعمل حاجة مفيدة بردو مثلا path completer ؟ نايس



الفكرة اننا نكاش ال ملفات اللى فى ال path اللى المستخدم هيكتبه ك model وندخله للل entrycompletion object

```
def get_model

    lstore=Gtk::ListStore.new String


      Dir.entries(@entry.text).each{|f|

          iter=lstore.append

          iter[0]=@entry.text<<f
```

```
            #puts f
        }
         return lstore
  end
```

التصميم

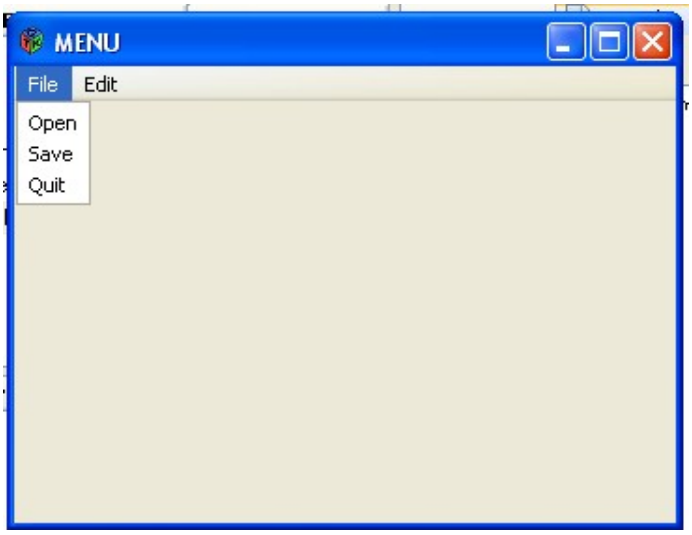
```
  def init_comp
    self.title="EntryCompletion"
    self.border_width=20
    self.window_position=Gtk::Window::POS_CENTER



    @mvbox=Gtk::VBox.new(false, 0)
    @lbl=Gtk::Label.new
    @lbl.set_markup("Type a <b>file path</b> with the help of code
completion.")
    @mvbox.pack_start(@lbl, true, true, 0)
    @ecompl=Gtk::EntryCompletion.new



    @entry=Gtk::Entry.new
    @entry.text="C:\\"
```

```
  @ecompl.model=get_model
  @ecompl.text_column=0
  @entry.completion=@ecompl

  @entry.signal_connect("changed"){|w|
    w.completion.model=get_model if File.directory?(w.text)
  }

  @mvbox.pack_start(@entry, true, true, 0)

  add(@mvbox)

end
```

هنا ربطنا ال changed signal بال get_model كالتالى

```
  @entry.signal_connect("changed"){|w|
    w.completion.model=get_model if File.directory?(w.text)
  }
```

Menus



فى عندنا هنا MenuBar فيه 2 MenuItems وهما File, Edit

فى ال file_menu (ال menu الخاصة بال file menu item) عندنا بتشمل 3 items

Open, Save, Quit

وال edit_menu فيها Copy, Cut, Paste

تمام اولا لإنشاء منيو بار

```
mbar = Gtk::MenuBar.new
```

ال items اللى عليها file, edit وماهما الا MenuItems فإنشئهم كالتالى

```
filemitem = Gtk::MenuItem.new('File')
editmitem = Gtk::MenuItem.new('Edit')
```

نعمل ال menus اللى هتظهر عند الضغط على filemitem او editmitem

```
filemenu = Gtk::Menu.new
editmenu = Gtk::Menu.new
```

ال filemenu بتشمل ال open, save, quit menu items

```
file_open_item = Gtk::MenuItem.new('Open')
```

```
filemenu.add file_open_item


file_save_item = Gtk::MenuItem.new('Save')
filemenu.add file_save_item


file_quit_item = Gtk::MenuItem.new('Quit')
filemenu.add file_quit_item
```

بننشئ الأوبجكت وبعد كدا بنضيفه للmenu بإستخدام .add

نفس الفكرة لeditmenu

```
edit_copy_item=Gtk::MenuItem.new("Copy")
editmenu.add edit_copy_item
edit_cut_item=Gtk::MenuItem.new("Cut")
editmenu.add edit_cut_item
edit_paste_item=Gtk::MenuItem.new("Paste")
editmenu.add edit_paste_item
```

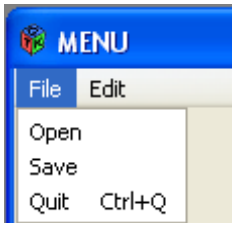
نحدد ان الfilemenu والeditmenu هما الsubmenus الخاصين بي ,filemitem editmitem

```
filemitem.set_submenu filemenu
editmitem.set_submenu editmenu
```

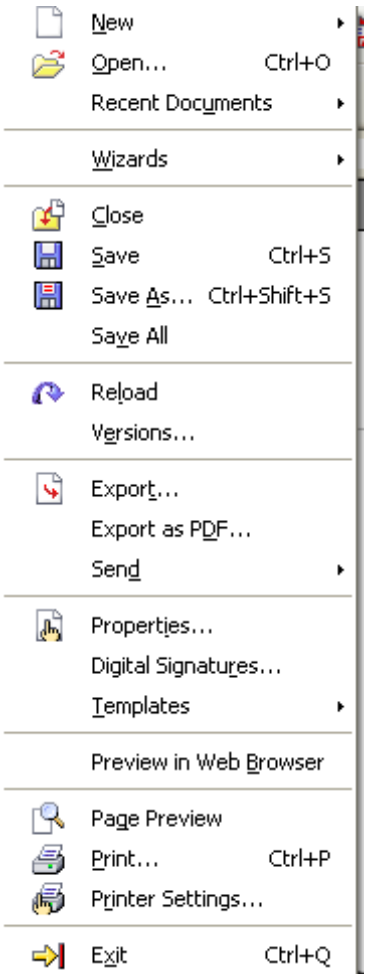
نضيف الfilemitem, editmitem للmenubar

```
mbar.append filemitem
mbar.append editmitem
```

استخدام ال AcclGroup



هى عبارة عن keyboard accelerators او "مسرعات" بإستخدام الإختصارات زى احيانا لما تدوس Alt+F4 مثلا فى ال open office



الفكرة اننا نضيف مسرع على ال menuitem ودا بيتم بكل بساطة

1- انشئ ال AccelGroup object

accgroup=Gtk::AccelGroup.new

2- اربطه بال window

عن طريق

.add_accel_group

window.add_accel_group accgroup

3- اربط ال menuitem اللى انت عايزه مثلا ال quit

```
file_quit_item.add_accelerator("activate", accgroup, Gdk::Keyval::GDK_q,
Gdk::Window::CONTROL_MASK, Gtk::ACCEL_VISIBLE)
```

هنا بنقول اربط ال activate بال accgroup المعرفة سابقا بالتحديد ب "حرف ال q والكنترول" مع إظهار الإختصار على المنيو Ctrl+Q

اربط ال activate signal بال Gtk::main_quit

```
file_quit_item.signal_connect("activate"){

  Gtk::main_quit

}
```

وبس كدا!

--تقدر تراجع الموقع الرسمى بخصوص ال ويدجتس الإضافية زى

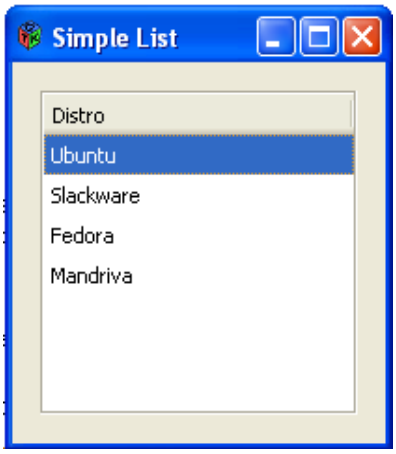
Check Menu Item

Radio Menu Item

Separator Menu Item

Tearoff Menu Item

TreeView



ال treeview بتعتمد على مبدأ ال mvc يعنى بتعرض (view) الداتا (ال model)

هنا المودل زى مانت شايف ليست

["Ubuntu", "Slackware", "Fedora", "Mandriva"]

الأول نكتب ميثود لإنشاء المودل

```
def get_model
    lstore=Gtk::ListStore.new String
    ["Ubuntu", "Slackware", "Fedora", "Mandriva"].each{|dist|
     iter=lstore.append
     iter[0]=dist
    }
    return lstore

end
```

ال iter بيعبر عن صف ال ListStore وال 0 بتعبر عن عمود فيه

انشئ scrollbar ليحتوى ال treeview

الصورة العامة

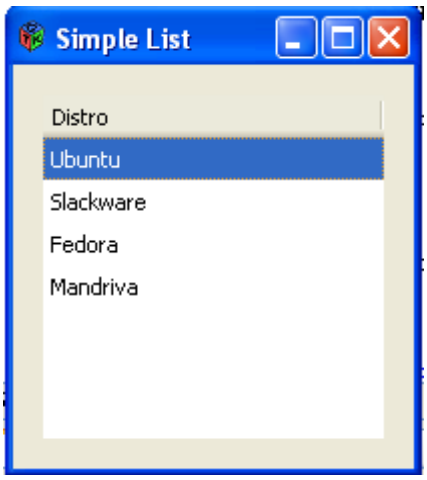
```
Gtk::ScrolledWindow.new( hadjustment, vadjustment )
```

فهننشئه كدا

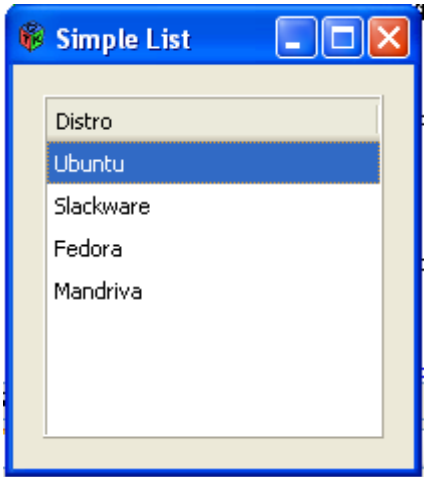
```
@sw = Gtk::ScrolledWindow.new(nil, nil)
```

ال shadow type

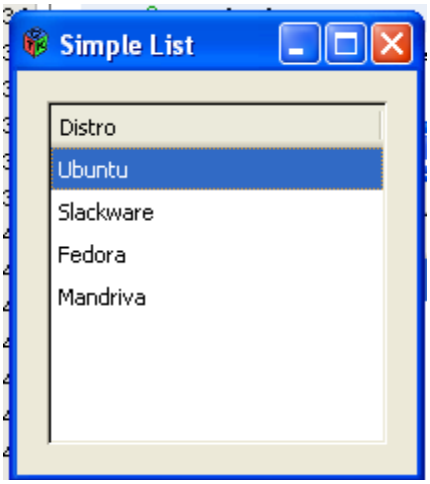
الديفولت None



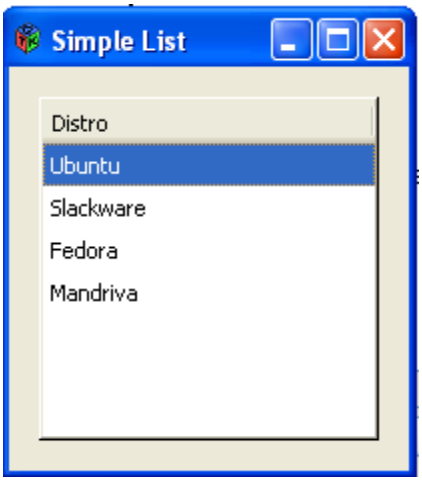
ETCHED_OUT



IN



OUT



@sw.shadow_type = Gtk::SHADOW_ETCHED_IN

@sw.set_policy(Gtk::POLICY_NEVER, Gtk::POLICY_AUTOMATIC)

هنا بنحدد ال policy

AUTOMATIC: ظهور سكرولبار او لأ هيتم اتوماتيك

ALWAYS: ظهور سكرولبار حتى فى عدم الحاجة إليه

NEVER: عدم ظهوره

ال TreeView بقة

@treev=Gtk::TreeView.new get_model

الديفولت للمودل nil

تقدر تعمله set من خلال

.set_model(model)

.model=(model)

ناقص انا ننشئ ال columns الخاصة بال treeview ونضيفهم
(هنا مش عايزين غير واحد بس)

coldis=Gtk::TreeViewColumn.new("Distro", renderer, 'text'=>0)

الأول ال title

التانى بيعبر عن ال renderer وهنا هيكون text renderer

```
renderer = Gtk::CellRendererText.new
```

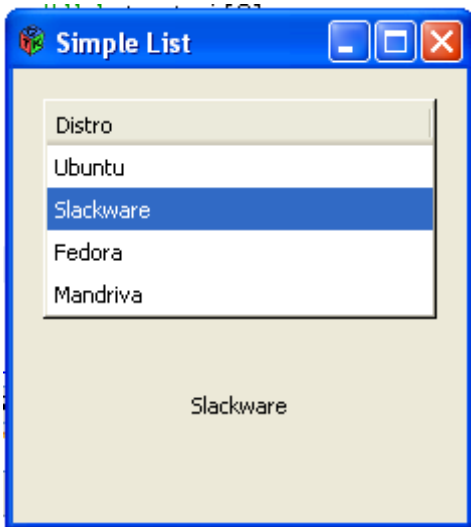
ال text يعرض اى جزء من الصف فى المودل ؟ هنا ال 0 راجع المودل

```
iter[0]=dist
```

وبس ضيفه لل treeview بإستخدام append_column

```
@treev.append_column coldis
```

changed signal



احنا بنعمل ليست ليه ؟ اكيد عشان نختار منها حاجة فعشان نعرف ايه اللى المستخدم اختاره لازم نربط ميثود معينة بال changed signal

1- نجيب ال selection

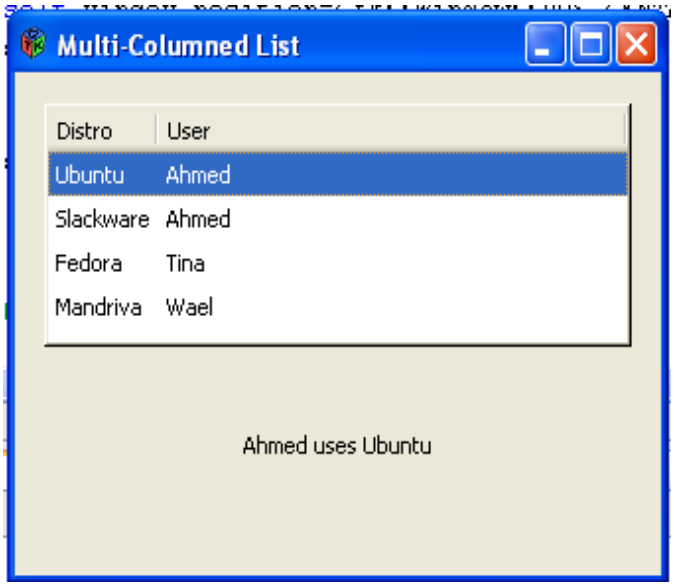
```
@sel=@treev.selection
```

2- نربط ال changed signal كالتالى -على فرض ان في label بإسم @lbl

```
@sel.signal_connect("changed"){

  i=@sel.selected
```

```
    @lbl.text=i[0]
  }
```

Multi-Columned



الأول هنعمل ال مودل (ال ListStore هتكون فيها ال row من عنصرين)

```
    lstore=Gtk::ListStore.new String, String
 def get_model
    lstore=Gtk::ListStore.new String, String

    [["Ahmed","Ubuntu"], ["Ahmed","Slackware"], ["Tina", "Fedora"], ["Wael",
"Mandriva"]].each{|user, dist|
      iter=lstore.append
      iter[0]=user
      iter[1]=dist
    }
    return lstore
 end
```

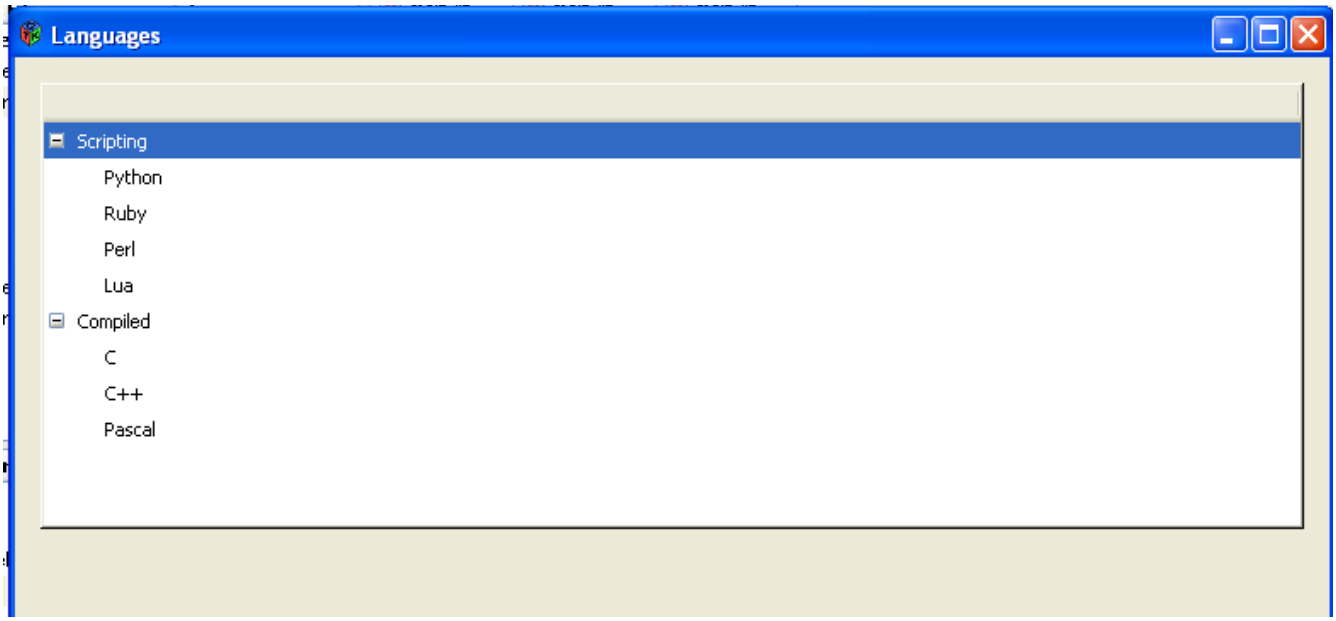
init_comp الى

```
def init_comp

    self.title="Multi-Columned List"
    self.set_size_request 200, 200 #400X400
    self.border_width=15
    self.window_position=Gtk::Window::POS_CENTER #Centerin the window
    signal_connect("delete-event"){
      false
    }
    signal_connect("destroy"){
      Gtk::main_quit
    }

    @mvbox=Gtk::VBox.new(false, 0)

    @sw = Gtk::ScrolledWindow.new(nil, nil)
    @sw.shadow_type = Gtk::SHADOW_OUT
    @sw.set_policy(Gtk::POLICY_NEVER, Gtk::POLICY_AUTOMATIC)

    @treev=Gtk::TreeView.new get_model

    #renderer
```

```
renderer = Gtk::CellRendererText.new
#column
colusers=Gtk::TreeViewColumn.new("User", renderer, 'text'=>0)
coldis=Gtk::TreeViewColumn.new("Distro", renderer, 'text'=>1)

@treev.append_column coldis
@treev.append_column colusers

@sw.add(@treev)

#lbl to display selection.

@lbl=Gtk::Label.new

@sel=@treev.selection

@sel.signal_connect("changed"){
  i=@sel.selected
  @lbl.text= i[0] + " uses " + i[1]
}

@mvbox.pack_start(@sw, true, true, 0)
@mvbox.pack_start(@lbl, true,true, 5)
add @mvbox
```

```
  end




  def quit_app
    Gtk.main_quit
  end


  def get_model
      lstore=Gtk::ListStore.new String, String


      [["Ahmed","Ubuntu"], ["Ahmed","Slackware"], ["Tina", "Fedora"], ["Wael",
"Mandriva"]].each{|user, dist|
        iter=lstore.append
        iter[0]=user
        iter[1]=dist
      }
      return lstore
  end


end
```

الوقتى عايزين نعمل حاجة مشابهه لكدا



بردو عن طريق ال TreeView

root :scripting

childS: python, ruby, perl, lua

root :compiled

childS: C, C++, Pascal

وصف برمجى

```
langs=[["Scripting",
          ["Python", "Ruby", "Perl", "Lua"]
        ],
        [
          "Compiled",
          ["C", "C++","Pascal"]
        ]
```

```
    ]
```

لعمل حاجةزى كدا هنستخدم ال TreeStore بدل من ال ListStore (واضح لأننا بنعمل tree!)

```
tstore=Gtk::TreeStore.new String


langs.each{|type, arr|
  rootiter=tstore.append(nil)
  rootiter.set_value(0, type)
  for lang in arr
    lnode=tstore.append(rootiter)
    lnode.set_value(0, lang)
  end


}
return tstore
```

شرحها

ال langs بتتقسم ل arrays كل واحدة منهم بتضم type و array تانية فيها ال langs

هناخد ال type ونخليه root

وناخد كل lang من ال langs الموجودة فى ال array التانية ونخليها child لل root!

ونضيفها بكل بساطة لل treeview

```
@treev=Gtk::TreeView.new get_model


#renderer
renderer = Gtk::CellRendererText.new
#column
```

```
collangs=Gtk::TreeViewColumn.new("", renderer, 'text'=>0)


@treev.append_column collangs
```

تقدر تعرف ايه اللى اللى مستخدم اختاره عن طريق ربط ال changed signal

```
@sel=@treev.selection
@sel.signal_connect("changed"){
  i=@sel.selected
  if i
    @lbl.text= i[0]
  end
}
```

ال treeview ويدجت فيه حاجات كتير جدا وصعب نغطيها هنا

تقدر تراجع هنا

http://ruby-gnome2.sourceforge.jp/hiki.cgi?tut-treeview

وتقدر تراجع

http://scentric.net/tutorial/treeview-tutorial.html

Ruby SE

الأول ال main layout هيكون Vbox

بيضم

1- ال menubar

2- ال textview

3- ال statusbar

كود البرنامج بتاعنا

main.rb

```
require "gtk2"
require "main_window"

w=MainWindow.new
Gtk.main
```

ال MainWindow دى عبارة عن ال GUI بتاعت برنامجنا موجودة فى main_window.rb

```
class MainWindow < Gtk::Window
```

هنكتب ميثود init_comp لإنشاء الكومبوننتس على ال Window عند انشاءها

```
  def initialize
    super
    @mod=false
    @filename=nil
    init_comp
    show_all
  end
```

@mod: لنختبر على هتم تغيير ال text او لأ

@filename: الفايل الحالى اللى بيتم عرضه

نروح ل init_comp method

```
def init_comp
```

1- نظبط خصائص الويندو بتاعتنا

```
    self.title="SE"
    self.set_size_request 400, 400 #400X400
    self.window_position=Gtk::Window::POS_CENTER #Centerin the window

   signal_connect("delete-event"){
      false
    }
    signal_connect("destroy"){
      Gtk.main_quit
    }
```

ال title, size, position

2- نعمل ال menus بتاعتنا ونربطها ب methods مثلا لما يضغط على New يبقة on_new

وعلى Save يبقة on_save وهكذا

```
@mbar=Gtk::MenuBar.new
  @fileitem=Gtk::MenuItem.new("File")
  @filenewitem=Gtk::MenuItem.new("New")
  @filenewitem.signal_connect("activate"){|w|
    on_new
  }
  @fileopenitem=Gtk::MenuItem.new("Open")
  @fileopenitem.signal_connect("activate"){|w|
    on_open_file
  }
  @filesaveitem=Gtk::MenuItem.new("Save")
  @filesaveitem.signal_connect("activate"){|w|
    on_save
  }

  @filesaveasitem=Gtk::MenuItem.new("Save as")
  @filesaveasitem.signal_connect("activate"){|w|
    save_as
  }
  #@sep=Gtk::SeparatorMenuItem.new
```

```
@filequititem=Gtk::MenuItem.new("Exit")
@filequititem.signal_connect("activate"){|w|
  quit_app
}

@filemenu=Gtk::Menu.new
@filemenu.add @filenewitem
@filemenu.add @fileopenitem
@filemenu.add @filesaveitem
@filemenu.add @filesaveasitem
@filemenu.add Gtk::SeparatorMenuItem.new
@filemenu.add @filequititem

@fileitem.set_submenu @filemenu
@mbar.append(@fileitem)

@edititem=Gtk::MenuItem.new("Edit")
@editcopyitem=Gtk::MenuItem.new("Copy")
@editcopyitem.signal_connect("activate"){|w|
  on_copy
}

@editcutitem=Gtk::MenuItem.new("Cut")
@editcutitem.signal_connect("activate"){|w|
```

```
  on_cut
}
@editpasteitem=Gtk::MenuItem.new("Paste")
@editpasteitem.signal_connect("activate"){|w|
  on_paste
}
@editselectallitem=Gtk::MenuItem.new("Select all")
@editselectallitem.signal_connect("activate"){|w|
  on_select_all
}
@editfinditem=Gtk::MenuItem.new("Find")
@editfinditem.signal_connect("activate"){|w|
  on_find
}
@editreplaceitem=Gtk::MenuItem.new("Replace")
@editreplaceitem.signal_connect("activate"){|w|
  on_replace
}

@editmenu=Gtk::Menu.new
@editmenu.add @editcopyitem
@editmenu.add @editcutitem
@editmenu.add @editpasteitem
@editmenu.add Gtk::SeparatorMenuItem.new
```

```
@editmenu.add @editselectallitem

@editmenu.add @editfinditem

@editmenu.add @editreplaceitem



@edititem.set_submenu @editmenu

@mbar.add(@edititem)


@aboutitem=Gtk::MenuItem.new("Help")

@aboutse=Gtk::MenuItem.new("About SE..")

@aboutse.signal_connect("activate"){|w|

  on_about_se

}

@aboutgtk=Gtk::MenuItem.new("About GTK+..")


@aboutmenu=Gtk::Menu.new

@aboutmenu.add @aboutse

@aboutmenu.add @aboutgtk


@aboutitem.set_submenu @aboutmenu

@mbar.add @aboutitem


@mvbox.pack_start(@mbar, false, false, 0)
```

ننشئ ال TextView

```
@tview=Gtk::TextView.new
```

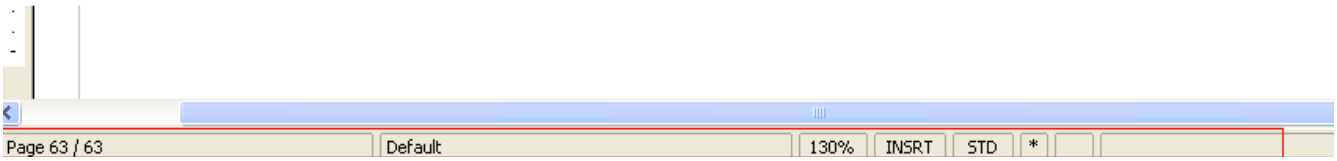
نحصل على ال buffer منه

```
@buf=@tview.buffer
```

نضيف ال @tview لل mvbox

```
@mvbox.pack_start(@tview, true, true, 0)
```

ننشئ اوبجكت من ال Statusbar (هو عبارة عن ويدجت بيستخدم فى عرض ملخص للحالة البرنامج




```
@sts=Gtk::Statusbar.new
```

نضيفه للل mvbox

```
@mvbox.pack_start(@sts, false, false, 0)
```

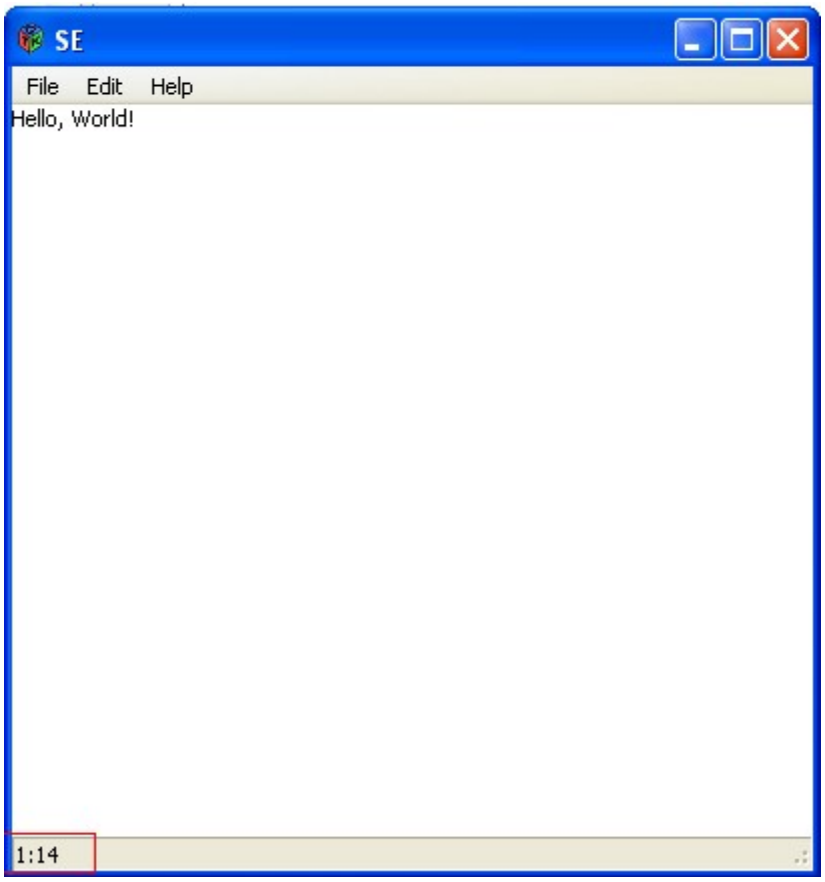
نخلص من ال @sts ووظيفته بالمرة

احنا عايزين رقم الصف والعمود اللى هيتم ادخال الحرف عندهم تمام؟

ال @buf هو اوبجكت من ال TextBuffer وفيه علامات جواه وفى علامات تقدر تنشئها منها insert mark وهى علامة بتشير لمكان الإدخال الحالى فكل الفكرة اننا نسحب منه iterator عن ال mark دى وال iterator دا هيكون فيه حاجات زى ال line, line_offset اللى بتعبر عن الصف والعمود

```
@buf.signal_connect("mark-set"){

  @sts.pop(0)

  iter=@buf.get_iter_at_mark(@buf.get_mark("insert"))

  r=iter.line+1 #Starts from zero.

  c=iter.line_offset+1 #Starts from zero.

  @sts.push(0, "#{r}:#{c}")
```

}



نايس

تعالى نشوف الميثودز اللى هيقدمها ال SE

```
def clear
  @filename=nil
  @buf.text=""
end
```

بتشيل ال buffer اللى على ال textview وتخلى ال @filename ل nil

بتحصل لما يجى يتعمل new file

```
def on_new
  if @filename
```

```
    show_saving_dlg
  end
  clear
  self.title="[New]"
end
```

لو فى فايل مفتوح هيظهر رسالة هل عايز تسيفه الأول ولا لأ؟ وتحول ال title بتاع ال SE ل [NEW]

```
def on_open_file
    file_sel=Gtk::FileSelection.new("Open File:")
    file_sel.ok_button.signal_connect("clicked"){
    @filename=file_sel.filename
    self.title=File.basename(@filename)
    read_file(@filename)
    file_sel.hide

    }
    file_sel.cancel_button.signal_connect( "clicked" ) { file_sel.hide }

    file_sel.show_all
end
```

هنا بنعرض ديالوج لإختيار فايل معين وقرايته بإستخدام ال read_file (هنشوفها الوقتى) + نعمل set لل @filename ونعدل ال title بتاع الويندو لل basename الخاص بالفايل دا

```
def read_file(path)
  f=File.new(path)
```

```
  @buf.text=f.readlines.join


end
```

بتستخدم لقراية ملف فى path معين

```
def on_save
  if @filename
    save
  else
    save_as
  end
end
```

لحفظ ملف ما اذا كان مفتوح يتم حفظه مباشرة لو لأ يتم التحويل ل save_as

```
def save
  if @filename
    f=File.new(@filename, "w")
    f.write(@buf.text)
    f.close
  end


end
```

اذا كان الملف مفتوح @filename لايساوى nil هيتم الحفظ عليه لو لأ مش هيحصل حاجة...

```
def save_as
  get_file_to_save_as
```

```
end
```

بتحولنا لميثود get_file_to_save_as

```
def get_file_to_save_as
    file_sel=Gtk::FileSelection.new("Save as:")
    file_sel.ok_button.signal_connect("clicked"){
      file_sel.hide
      f=File.new(file_sel.filename, "w")
      f.write(@buf.text)
      f.close
    }
    file_sel.cancel_button.signal_connect( "clicked" ) { file_sel.hide }
    file_sel.show_all
end
```

بتعرض ديالوج للمسار الحفظ وتسيف مع الضغط على ال ok

```
def quit_app
  Gtk.main_quit
end
```

هنربطها بال ال quititem فى ال file menu

```
def on_copy
  @tview.signal_emit("copy_clipboard")
end
```

لعملية ال copy لل clipboard كل اللى عليك انك ترسل copy_clipboard signal

```
def on_cut
   @tview.signal_emit("cut_clipboard")
end
```

لعملية ال cut لل clipboard كل اللى عليك انك ترسل cut_clipboard signal

لعلمية ال paste كل ماعليك هو ارسال paste_clipboard

```
def on_paste
   @tview.signal_emit("paste_clipboard")
end
```

```
def on_select_all
  @buf.place_cursor(@buf.end_iter)
  @buf.move_mark(@buf.get_mark("selection_bound"), @buf.start_iter)
end
```

لإختيار كل السطور كل بننقل ال cursor لل end_iter فى النهاية

ونجيب الحدود بين ال start_iter وال end_iter

```
def on_about_se
  abdl=Gtk::AboutDialog.new
  abdl.name="SE"
  abdl.version="0.9"
  abdl.authors=["Ahmed Youssef"]
  abdl.copyright="(c) Ahmed Youssef"
  abdl.comments="My fancy text editor"
```

```
    abdl.website="http://www.programming-fr34ks.net"
    abdl.artists=["Sami"]
    abdl.license="GPL v3 text goes here..."
    abdl.documenters=[“Ahmed Youssef”]
    abdl.run
    abdl.destroy
  end
```

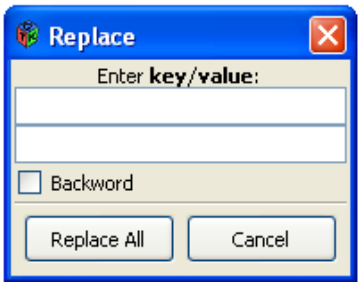
عند الضغط على about

```
  def on_about_gtk
    #pass
  end
```

معلومات عن gtk تقدر تجيبها من gtk.org

تمام كدا ناقص ال Find/Replace

انا هعمل ال replace وهسيب ال find لهدف



دى ال GUI بتاعت الديالوج هنكتبها

replace_dialog.rb

```
class ReplaceDialog < Gtk::Dialog
```

فى 3 متغيرت ال key, value و backword

```
  @key=nil
```

```
@val=nil
@backword=false
```

getters

```
def getkey
  @key
end

def getval
  @val
end

def backword?
  return @backword
end
```

constructor الى

```
def initialize
  super
  init_comp
  show_all
end
```

components الى

```
def init_comp
```

```
    self.title="Replace"
    @lbl=Gtk::Label.new
    @lbl.markup="Enter <b>key/value</b>:"
    @entrykey=Gtk::Entry.new
    @entrykey.signal_connect("changed"){|w|
      @key=w.text
    }
    @entryval=Gtk::Entry.new
    @entryval.signal_connect("changed"){|w|
      @val=w.text
    }

    @chk=Gtk::CheckButton.new "Backword"
    @chk.signal_connect("toggled"){
      @backword=!@backword
    }
    self.vbox.pack_start(@lbl, false, false, 0)
    self.vbox.pack_start(@entrykey, false, false, 0)
    self.vbox.pack_start(@entryval, false, false, 0)
    self.vbox.pack_start(@chk, false, false, 0)
    #should it be enhanced to [find_next]...
    add_buttons(["Replace All", Gtk::Dialog::RESPONSE_OK], ["Cancel",
Gtk::Dialog::RESPONSE_CANCEL])
  end
```

هنا لل ديالوج فى vbox جاهز تقدر تضيف فيه مباشرة زى مافى الكود كدا

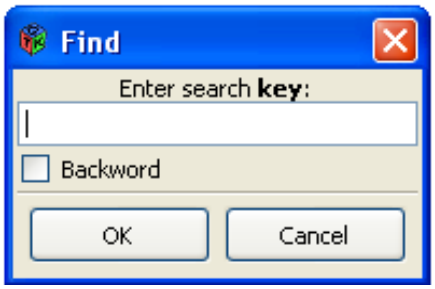
اضافة ال buttons الخاصة بالديالوج مثلا عن طريق add_buttons ونحدد ال response بتاع وهكذا

```
add_buttons(["Replace All", Gtk::Dialog::RESPONSE_OK], ["Cancel",
Gtk::Dialog::RESPONSE_CANCEL])
```

نرجع لل main_window

```
def on_replace

    rd=ReplaceDialog.new



    res=rd.run

    k,v, back=rd.getkey, rd.getval, rd.backword?

    if res==Gtk::Dialog::RESPONSE_OK

      if k and v

        puts "Replace #{k} with #{v}"

        puts "Backword? #{back}"

        while @buf.text.index(k)

          @buf.text = @buf.text.sub(k, v)

        end

      end

    end



    rd.destroy



end
```

هنستدعى الديالوج اللى لسه كاتبينه وفى حال وجود key, value المستخدم كتبهم هيتم استبدالهم فى ال @buf.text



find_dialog.rb

```
require "gtk2"
class FindDialog < Gtk::Dialog
  def initialize
    super
    init_comp
    @searchkey=nil
    @backword=false
    show_all
  end
  def searchkey
    @searchkey
  end
  def backword
    @backword
  end
  def init_comp
```

```
    self.title="Find"


    @lbl=Gtk::Label.new
    @lbl.markup="Enter search <b>key</b>:"


    @entry=Gtk::Entry.new
    @entry.signal_connect("changed"){|w|
      @searchkey=w.text
    }
    @chk=Gtk::CheckButton.new "Backword"
    @chk.signal_connect("toggled"){
      @backword=!@backword
    }


    self.vbox.pack_start(@lbl, false, false, 0)
    self.vbox.pack_start(@entry, false, false, 0)
    self.vbox.pack_start(@chk, false, false, 0)
    #should be enhanced to [find_next]...
    add_buttons(["OK", Gtk::Dialog::RESPONSE_OK], ["Cancel",
Gtk::Dialog::RESPONSE_CANCEL])



  end
end
```

المطلوب منك فى النهاية هو انهاء ال

```
def on_find
.....
end
```

تلميح هتحتاج تستخدم ال marks/iterators

الكود النهائى

main.rb

```
require "gtk2"
require "main_window"

w=MainWindow.new
Gtk.main
```

main_window.rb

```
require "gtk2"
require "find_dialog"
require "replace_dialog"

class MainWindow < Gtk::Window
  def initialize
    super
    @mod=false
```

```
  @filename=nil
  init_comp
  show_all
end
def init_comp
  self.title="SE"
  self.set_size_request 400, 400 #400X400
  self.window_position=Gtk::Window::POS_CENTER #Centerin the window

  signal_connect("delete-event"){
    false
  }
  signal_connect("destroy"){
    Gtk.main_quit
  }

  @mvbox=Gtk::VBox.new(false, 0)
  @mbar=Gtk::MenuBar.new
  @fileitem=Gtk::MenuItem.new("File")
  @filenewitem=Gtk::MenuItem.new("New")
  @filenewitem.signal_connect("activate"){|w|
    on_new
  }
  @fileopenitem=Gtk::MenuItem.new("Open")
```

```
@fileopenitem.signal_connect("activate"){|w|
  on_open_file
}
@filesaveitem=Gtk::MenuItem.new("Save")
@filesaveitem.signal_connect("activate"){|w|
  on_save
}

@filesaveasitem=Gtk::MenuItem.new("Save as")
@filesaveasitem.signal_connect("activate"){|w|
  save_as
}
#@sep=Gtk::SeparatorMenuItem.new
@filequititem=Gtk::MenuItem.new("Exit")
@filequititem.signal_connect("activate"){|w|
  quit_app
}

@filemenu=Gtk::Menu.new
@filemenu.add @filenewitem
@filemenu.add @fileopenitem
@filemenu.add @filesaveitem
@filemenu.add @filesaveasitem
@filemenu.add Gtk::SeparatorMenuItem.new
```

```
@filemenu.add @filequititem

@fileitem.set_submenu @filemenu
@mbar.append(@fileitem)

@edititem=Gtk::MenuItem.new("Edit")
@editcopyitem=Gtk::MenuItem.new("Copy")
@editcopyitem.signal_connect("activate"){|w|
  on_copy
}

@editcutitem=Gtk::MenuItem.new("Cut")
@editcutitem.signal_connect("activate"){|w|
  on_cut
}
@editpasteitem=Gtk::MenuItem.new("Paste")
@editpasteitem.signal_connect("activate"){|w|
  on_paste
}
@editselectallitem=Gtk::MenuItem.new("Select all")
@editselectallitem.signal_connect("activate"){|w|
  on_select_all
}
@editfinditem=Gtk::MenuItem.new("Find")
```

```
@editfinditem.signal_connect("activate"){|w|
  on_find
}
@editreplaceitem=Gtk::MenuItem.new("Replace")
@editreplaceitem.signal_connect("activate"){|w|
  on_replace
}

@editmenu=Gtk::Menu.new
@editmenu.add @editcopyitem
@editmenu.add @editcutitem
@editmenu.add @editpasteitem
@editmenu.add Gtk::SeparatorMenuItem.new
@editmenu.add @editselectallitem
@editmenu.add @editfinditem
@editmenu.add @editreplaceitem


@edititem.set_submenu @editmenu
@mbar.add(@edititem)

@aboutitem=Gtk::MenuItem.new("Help")
@aboutse=Gtk::MenuItem.new("About SE..")
@aboutse.signal_connect("activate"){|w|
```

```
  on_about_se
}
@aboutgtk=Gtk::MenuItem.new("About GTK+..")

@aboutmenu=Gtk::Menu.new
@aboutmenu.add @aboutse
@aboutmenu.add @aboutgtk

@aboutitem.set_submenu @aboutmenu
@mbar.add @aboutitem

@mvbox.pack_start(@mbar, false, false, 0)

@sts=Gtk::Statusbar.new
@tview=Gtk::TextView.new
@buf=@tview.buffer
@buf.signal_connect("mark-set"){
  @sts.pop(0)
  iter=@buf.get_iter_at_mark(@buf.get_mark("insert"))
  r=iter.line+1 #Starts from zero.
  c=iter.line_offset+1 #Starts from zero.
  @sts.push(0, "#{r}:#{c}")

}
```

```
  @mvbox.pack_start(@tview, true, true, 0)



  @mvbox.pack_start(@sts, false, false, 0)
  add(@mvbox)



end

def clear
  @filename=nil
  @buf.text=""
end

def on_new
  if @filename
    show_saving_dlg
  end
  clear
  self.title="[New]"
end

def on_open_file
```

```
      file_sel=Gtk::FileSelection.new("Open File:")
      file_sel.ok_button.signal_connect("clicked"){
      @filename=file_sel.filename
      self.title=File.basename(@filename)
      read_file(@filename)
      file_sel.hide

      }
      file_sel.cancel_button.signal_connect( "clicked" ) { file_sel.hide }

      file_sel.show_all
end
def read_file(path)
  f=File.new(path)
  @buf.text=f.readlines.join

end

def on_save
  if @filename
    save
  else
    save_as
  end
```

```
end

def show_saving_dlg

  md=Gtk::MessageDialog.new(self, Gtk::Dialog::MODAL,
                            Gtk::MessageDialog::INFO,
                            Gtk::MessageDialog::BUTTONS_YES_NO,
                            "Save?")

  res=md.run

  if res==Gtk::Dialog::RESPONSE_YES
    puts "OK!"
    on_save
  elsif res==Gtk::Dialog::RESPONSE_NO
    puts "NO!"
    clear
  end
  md.destroy


end
def get_file_to_save_as
```

```
    file_sel=Gtk::FileSelection.new("Save as:")
    file_sel.ok_button.signal_connect("clicked"){
      file_sel.hide
      f=File.new(file_sel.filename, "w")
      f.write(@buf.text)
      f.close
    }
    file_sel.cancel_button.signal_connect( "clicked" ) { file_sel.hide }



    file_sel.show_all

end
def save_as
  get_file_to_save_as

end



def save
  if @filename
    f=File.new(@filename, "w")
    f.write(@buf.text)
    f.close
```

```
  end

end

def quit_app
  Gtk.main_quit
end

def on_copy
  @tview.signal_emit("copy_clipboard")
end

def on_cut
   @tview.signal_emit("cut_clipboard")
end

def on_paste
   @tview.signal_emit("paste_clipboard")
end

def on_select_all
  @buf.place_cursor(@buf.end_iter)
  @buf.move_mark(@buf.get_mark("selection_bound"), @buf.start_iter)
end
```

```
def on_find
    fd=FindDialog.new




end
def on_replace
    rd=ReplaceDialog.new


    res=rd.run
    k,v, back=rd.getkey, rd.getval, rd.backword?
    if res==Gtk::Dialog::RESPONSE_OK
      if k and v
        puts "Replace #{k} with #{v}"
        puts "Backword? #{back}"
        while @buf.text.index(k)
          @buf.text = @buf.text.sub(k, v)
        end
      end
    end


    rd.destroy
```

```
  end

  def on_about_se
    abdl=Gtk::AboutDialog.new
    abdl.name="SE"
    abdl.version="0.9"
    abdl.authors=["Ahmed Youssef"]
    abdl.copyright="(c) Ahmed Youssef"
    abdl.comments="My fancy text editor"
    abdl.website="http://www.programming-fr34ks.net"
    abdl.artists=["Sami"]
    abdl.license="GPL v3 text goes here..."
    abdl.documenters=["Ahmed Youssef"]
    abdl.run
    abdl.destroy
  end

  def on_about_gtk
    #pass
  end

end

replace_dialog
```

```
require "gtk2"

class ReplaceDialog < Gtk::Dialog
  @key=nil
  @val=nil
  @backword=false

  def getkey
    @key
  end

  def getval
    @val
  end

  def backword?
    return @backword
  end

  def initialize
    super
    init_comp
    show_all
  end
```

```
def init_comp
  self.title="Replace"
  @lbl=Gtk::Label.new
  @lbl.markup="Enter <b>key/value</b>:"
  @entrykey=Gtk::Entry.new
  @entrykey.signal_connect("changed"){|w|
    @key=w.text
  }
  @entryval=Gtk::Entry.new
  @entryval.signal_connect("changed"){|w|
    @val=w.text
  }

  @chk=Gtk::CheckButton.new "Backword"
  @chk.signal_connect("toggled"){
    @backword=!@backword
  }

  self.vbox.pack_start(@lbl, false, false, 0)
  self.vbox.pack_start(@entrykey, false, false, 0)
  self.vbox.pack_start(@entryval, false, false, 0)
  self.vbox.pack_start(@chk, false, false, 0)
  #should it be enhanced to [find_next]...
  add_buttons(["Replace All", Gtk::Dialog::RESPONSE_OK], ["Cancel",
```

```
Gtk::Dialog::RESPONSE_CANCEL])



  end
end

find_dialog
require "gtk2"
class FindDialog < Gtk::Dialog
  def initialize
    super
    init_comp
    @searchkey=nil
    @backword=false
    show_all
  end
  def searchkey
    @searchkey
  end
  def backword
    @backword
  end
  def init_comp
    self.title="Find"
```

```
    @lbl=Gtk::Label.new
    @lbl.markup="Enter search <b>key</b>:"

    @entry=Gtk::Entry.new
    @entry.signal_connect("changed"){|w|
      @searchkey=w.text
    }
    @chk=Gtk::CheckButton.new "Backword"
    @chk.signal_connect("toggled"){
      @backword=!@backword
    }



    self.vbox.pack_start(@lbl, false, false, 0)
    self.vbox.pack_start(@entry, false, false, 0)
    self.vbox.pack_start(@chk, false, false, 0)
    #should be enhanced to [find_next]...
    add_buttons(["OK", Gtk::Dialog::RESPONSE_OK], ["Cancel",
Gtk::Dialog::RESPONSE_CANCEL])



  end
end
```

للمزيد عن ال TextView widget

http://www.bravegnu.org/gtktext/