Glade-Tutorial mit PyGObject

Release 0.1

Anke K. (encarsia)

31.10.2017

Inhalt

Generated 5 paragraphs, 430 words, 2933 bytes of Lorem Ipsum

Generated on 31.10.2017.

KAPITEL 1

Tutorial-Reihe zu Glade

1.1 Themen

- Minimalbeispiel (*Fenster mit Aussicht*)
- Buttons und Labels (push-the-button)
- Fenster und Dialoge (durchzug)
- Schalter, Checkbox und Radiobutton (*Clickbaiting*)
- Menü, Toolbar und Statusbar (drei-gange-menu)
- Fortschrittsbalken und Levelbar (bars)
- CSS (css)
- Spinbutton und Combobox (qual-der-wahl)
- ListStore und TreeView (uberlistet)
- TreeStore mit Sortierung und Filterung (ansichtssache)
- Lokalisation mit locale und gettext (romani-ite-domum)
- VTE-Terminal (exterminate)
- Dialoge (dialoge)
- Programm als eigenständige GTK+-Anwendung (application)
- Icon, Headerbar und Kommandozeilenoptionen (application-fortsetzung)
- Dateiauswahldialog (fcdialog)
- GSettings-Konfigurationssystem (gsettings)
- Mediaplayer mit GStreamer (gst-player)
- Mediaplayer mit LibVLC (vlc-player)
- Stack und Notebook (*Stacks und Notebooks*)

Verzeichnis der Beispieldateien: encarsia.github.io/listings

1.2 Motivation

Bei der Erstellung der grafischen Oberfläche sowohl für gpt als auch NoN habe ich auf Glade zurückgegriffen, einem grafischen Werkzeug, mit dem man relativ einfach GTK+-Oberflächen erstellen kann.

Mit Glade erstellte Projektdateien sind GtkBuilder-XML-Dateien, die Verbindung zum eigentlichen Programm erfolgt über Signale, dabei werden zahlreiche Programmiersprachen unterstützt. Hier werde ich Python verwenden.

Da es in den letzten Jahren Versionssprünge sowohl bei Python als auch GTK+ gegeben hat (jeweils von 2.x auf 3.x), gibt es viele Dokumentationen und Tutorials, die nicht 1:1 anwendbar sind, d.h. die Funktionen sind meist gleich, nur die Syntax unterscheidet sich minimal (siehe Links).

An dieser Stelle versuche ich aktuell zu bleiben, derzeit mit Python 3.5.2 und Glade 3.20.0.

1.3 Nicht exklusiv

GTK+-Elemente können natürlich auch ohne Glade direkt im Quellcode des Programms erstellt werden. Es ist möglich, beide Optionen parallel zu verwenden oder auch im Entwicklungs-Verlauf das eine gegen das andere zu ersetzen.

Da Glade in verschiedenen Programmiersprachen eingesetzt werden kann, ist es ebenso denkbar, Programme in verschiedenen Sprachen mit derselben Oberfläche zu erstellen (migrieren).

1.4 Links

- The Python GTK+ 3 Tutorial - Grundlagen der Programmierung von GTK+-GUI mit Python
- Creating a GUI using PyGTK and Glade - Grundlagentutorial für PyGTK (Python 2.x)
- Programmieren mit Python und Glade - umfangreiches Tutorial auf Deutsch
- Python GObject Introspection API Reference - vollständige Dokumentation des GI-Moduls (bookmark this!)

1.5 Todo

- Textfelder mit Gtk.TextView und GtkSource.View
- Interaktion mit anderen Anwendungen mit und ohne Threading

KAPITEL 2

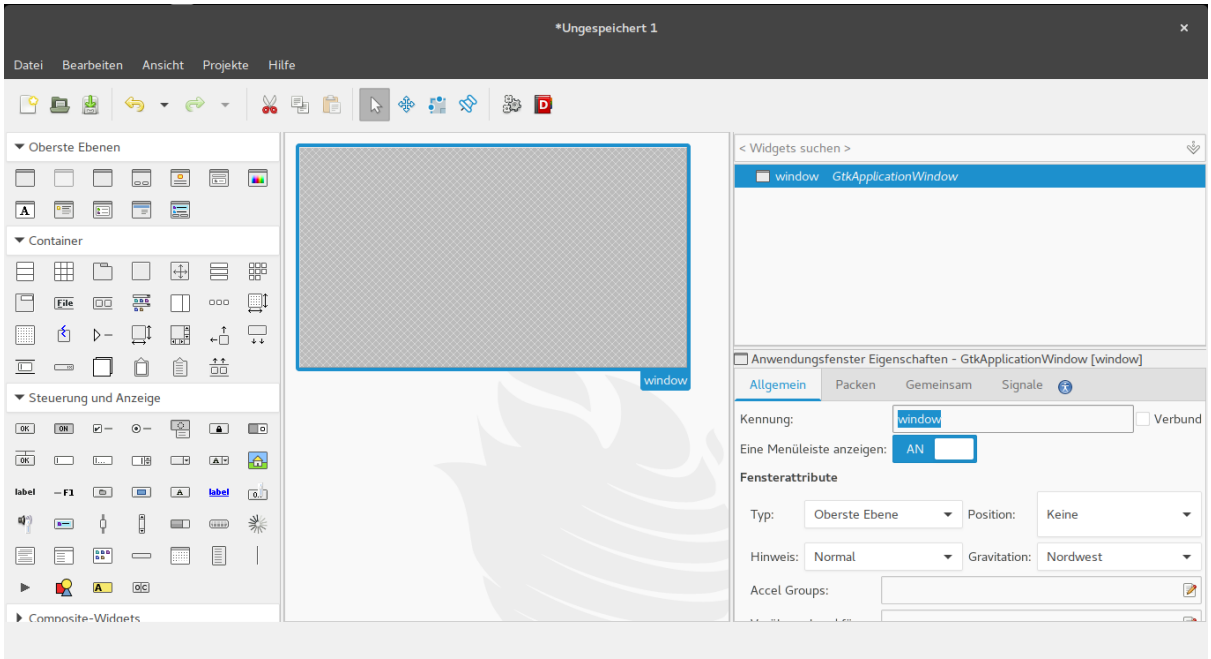
Fenster mit Aussicht

Minimalbeispiel

2.1 Glade

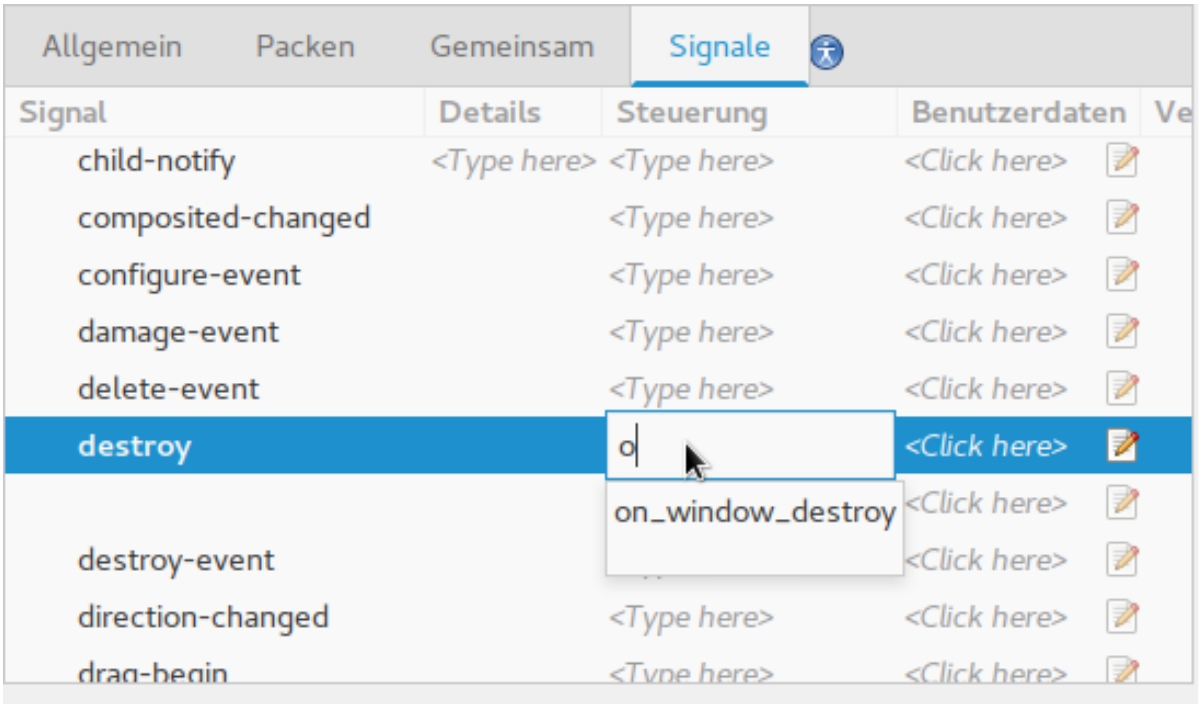
Nach dem Start präsentiert sich Glade dreigeteilt, links ist die Fenster-/Widget-Auswahl, in der Mitte die Projektansicht und rechts eine Baumansicht des Projekts, im unteren Bereich können Eigenschaften und Signale editiert werden.

Nun erstellt man ein Fenster und gibt ihm eine Kennung. Mit dieser Kennung wird das Objekt im Programmcode angesprochen.



Um die Ausführung von Funktionen durch ein Widget zu initiieren, müssen sie mit Signalen gekoppelt werden. Signale können je nach Objektart verschieden ausgelöst werden, durch Anklicken, Markieren, Editieren, Schalten etc.

Um in diesem Beispiel das Programmfenster mit dem Schließen-Button zu schließen, wird das Signal *destroy* benötigt. Beim Funktionsnamen hilft die Vorschlagsfunktion nach dem Schema `on_kennung_signal`. Ich empfehle, diesen Vorschlägen im allgemeinen zu folgen, sie erleichtern die Tipparbeit.



Glade selbst erzeugt keinen Programmcode, sondern eine XML-Datei des Typs *GtkBuilder*.

2.2 Python

First things first. Die *GtkBuilder*-Funktionen stehen im *Gtk*-Modul aus den Python GObject Introspection-Bindings zur Verfügung:

```
import gi
gi.require_version('Gtk','3.0')
from gi.repository import Gtk
```

Nach dem Aufruf von `Gtk.Builder()` wird die Glade-Datei geladen.

```
builder.add_from_file(gladefile)
```

Um die Übersicht zu bewahren, können dies auch mehrere Dateien sein, es sollte allerdings auf eine eindeutige Kennung geachtet werden. Bei doppelten gleichen Kennungen kann nur die zuletzt geladene mit `get_object(kennung)` angesprochen werden.

Anschließend werden die Signale verbunden. Meine Empfehlung ist hier, die dazugehörigen Funktionen der Übersicht wegen in eine eigene Klasse auszulagern.

```
self.builder.connect_signals(Handler())
```

Dieses Beispiel-Skript öffnet ein leeres Fenster, das per Schließen-Button beendet werden kann.

2.3 Ohne Glade

Das oben konstruierte Beispiel entspricht dem Basisbeispiel im Python GTK+ 3 Tutorial:

```
import gi
gi.require_version('Gtk', '3.0')
from gi.repository import Gtk

win = Gtk.Window()
win.connect("delete-event", Gtk.main_quit)
win.show_all()
Gtk.main()
```

Man sollte sich von der Kürze dieses Beispiels nicht täuschen lassen. Die eigentlichen Elemente, Boxen, Widget, Buttons, Leisten etc. fehlen hier komplett.

2.4 Listings

2.4.1 Glade

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- Generated with glade 3.20.0 -->
<interface>
  <requires lib="gtk+" version="3.20"/>
  <object class="GtkApplicationWindow" id="window">
    <property name="can_focus">False</property>
    <property name="title" translatable="yes">Titel</property>
    <signal name="destroy" handler="on_window_destroy" swapped="no"/>
    <child>
      <placeholder/>
    </child>
    <child>
      <placeholder/>
    </child>
  </object>
</interface>
```

2.4.2 Python

```
#!/usr/bin/python
# -*- coding: utf-8 -*-

import gi
gi.require_version('Gtk','3.0')
from gi.repository import Gtk

class Handler:

    def on_window_destroy(self,*args):
        Gtk.main_quit()

class Example:

    def __init__(self):

        self.builder = Gtk.Builder()
        self.builder.add_from_file("01_minimal.glade")
        self.builder.connect_signals(Handler())

        window = self.builder.get_object("window")
        window.show_all()
```

```
    def main(self):
        Gtk.main()

x = Example()
x.main()
```

KAPITEL 3

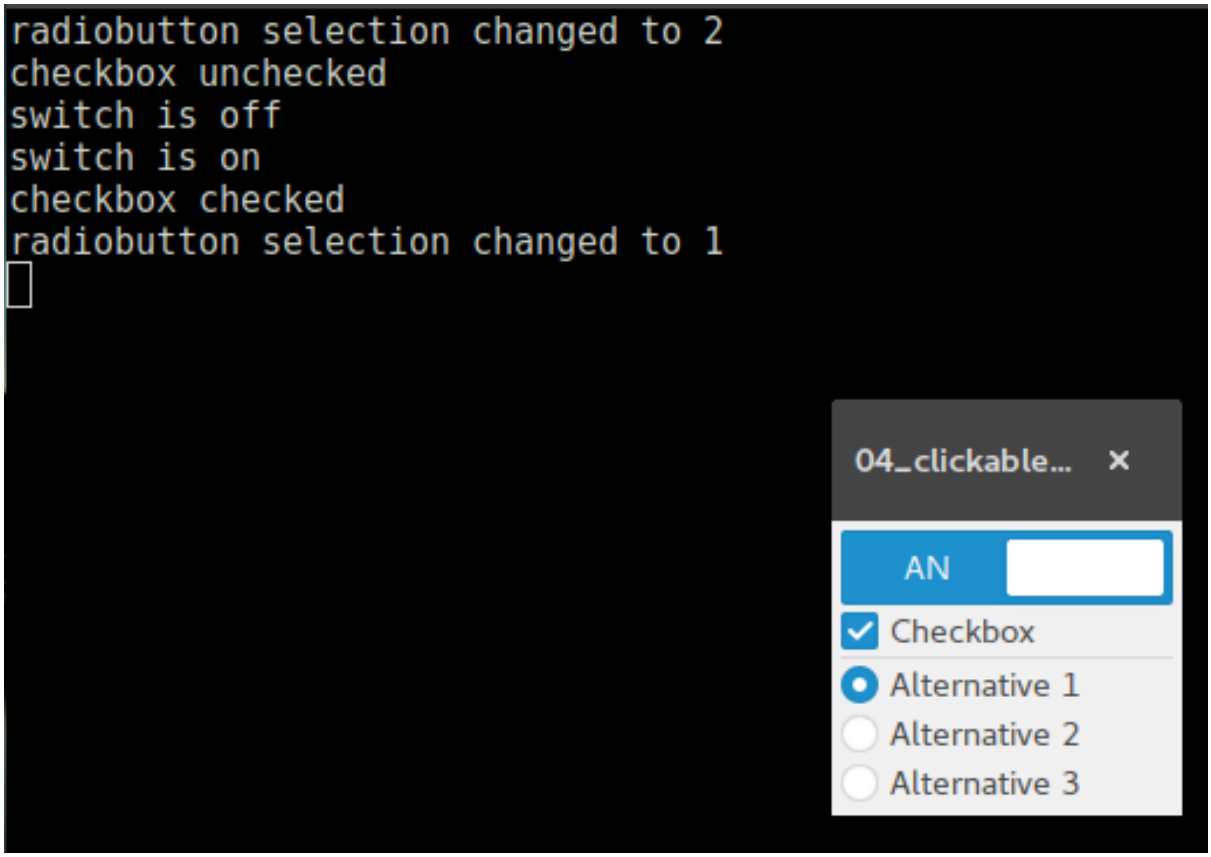
Clickbaiting

Switch, Checkbox, Radiobutton - mehr Elemente zum Anklicken

In diesem Artikel wird exemplarisch die Verwendung von Switches, Checkboxen und Radiobuttons vorgestellt. Folgend werden weitere Steuerungs- und Anzeigenelemente verwendet, es wird aber kein Anspruch auf Vollständigkeit erhoben, da die Verwendungsprozedur praktisch nach folgendem Schema funktioniert:

1. Container (Box, Leiste etc.) für Element anlegen
2. Element hinzufügen
3. Element mit einer Bezeichnung versehen (bei Elementen ohne Interaktion wie Boxen oder Trennlinien kann darauf verzichtet werden)
4. gewünschtem Signal eine Funktion zuweisen
5. (optional) Signal-/Funktionsaufruf in der Vorschaufunktion testen
6. Funktion im Programmcode schreiben

Alle verfügbaren GTK+-Klassen und ihre Funktionen findet man unter Python GI API Reference >> Gtk 3.0 >> Classes.



3.1 Glade

3.1.1 Switch oder Schalter

Ein Switch ist ein einfacher Ein-/Aus-Schalter mit, Überraschung!, zwei Zuständen. Der Zustand lässt sich über das Signal *state_set* abrufen.

3.1.2 Checkbox

Checkboxen sind Togglebuttons in anderem Outfit, hier wird demnach das Signal *toggled* belegt.

3.1.3 Radiobutton

Radiobuttons dienen der Auswahl _eines_ Listenpunktes aus einer gegebenen Liste. Das Element selbst funktioniert ebenfalls wie ein Togglebutton (das Signal *toggled* zuweisen).

Zusätzlich werden die zusammengehörigen Listenpunkte zu einer Gruppe zugeordet. Dies bewerkstelligt man einfach, indem man alle Radiobuttons unter „Allgemein > Knopfattribute > Gruppe“ an einem „führenden Radiobutton“ ausrichtet.

3.2 Python

Da Checkbox und Radiobutton Togglebuttons sind, wird hier der Status über die Funktion `widget.get_active()` abgerufen.

Beim Switch wird dem Signal *state_set* ein Parameter übergeben, der True/False ausgibt:

```
def on_switch_state_set(self,widget,state):
    if state is True:
        print("switch is on")
    else:
        print("switch is off")
```

3.3 Listings

3.3.1 Glade

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- Generated with glade 3.20.0 -->
<interface>
  <requires lib="gtk+" version="3.20"/>
  <object class="GtkWindow" id="window">
    <property name="can_focus">False</property>
    <property name="default_width">150</property>
    <signal name="destroy" handler="on_window_destroy" swapped="no"/>
    <child>
      <object class="GtkBox">
        <property name="visible">True</property>
        <property name="can_focus">False</property>
        <property name="margin_left">4</property>
        <property name="margin_right">4</property>
        <property name="margin_top">4</property>
        <property name="margin_bottom">4</property>
        <property name="orientation">vertical</property>
        <property name="spacing">1</property>
        <child>
          <object class="GtkSwitch" id="switch">
            <property name="visible">True</property>
            <property name="can_focus">True</property>
            <property name="active">True</property>
            <signal name="state-set" handler="on_switch_state_set" swapped="no"/>
          </object>
          <packing>
            <property name="expand">False</property>
            <property name="fill">True</property>
            <property name="position">0</property>
          </packing>
        </child>
        <child>
          <object class="GtkCheckButton" id="cbutton">
            <property name="label" translatable="yes">Checkbox</property>
            <property name="visible">True</property>
            <property name="can_focus">True</property>
            <property name="receives_default">False</property>
            <property name="active">True</property>
            <property name="draw_indicator">True</property>
            <signal name="toggled" handler="on_cbutton_toggled" swapped="no"/>
          </object>
          <packing>
            <property name="expand">False</property>
            <property name="fill">True</property>
            <property name="position">2</property>
          </packing>
        </child>
        <child>
          <object class="GtkSeparator">
```

```
            <property name="visible">True</property>
            <property name="can_focus">False</property>
          </object>
          <packing>
            <property name="expand">False</property>
            <property name="fill">True</property>
            <property name="position">3</property>
          </packing>
        </child>
        <child>
          <object class="GtkRadioButton" id="rbutton1">
            <property name="label" translatable="yes">Alternative 1</property>
            <property name="visible">True</property>
            <property name="can_focus">True</property>
            <property name="receives_default">False</property>
            <property name="active">True</property>
            <property name="draw_indicator">True</property>
            <signal name="toggled" handler="on_rbutton1_toggled" swapped="no"/>
          </object>
          <packing>
            <property name="expand">False</property>
            <property name="fill">True</property>
            <property name="position">4</property>
          </packing>
        </child>
        <child>
          <object class="GtkRadioButton" id="rbutton2">
            <property name="label" translatable="yes">Alternative 2</property>
            <property name="visible">True</property>
            <property name="can_focus">True</property>
            <property name="receives_default">False</property>
            <property name="active">True</property>
            <property name="draw_indicator">True</property>
            <property name="group">rbutton1</property>
            <signal name="toggled" handler="on_rbutton2_toggled" swapped="no"/>
          </object>
          <packing>
            <property name="expand">False</property>
            <property name="fill">True</property>
            <property name="position">5</property>
          </packing>
        </child>
        <child>
          <object class="GtkRadioButton" id="rbutton3">
            <property name="label" translatable="yes">Alternative 3</property>
            <property name="visible">True</property>
            <property name="can_focus">True</property>
            <property name="receives_default">False</property>
            <property name="active">True</property>
            <property name="draw_indicator">True</property>
            <property name="group">rbutton1</property>
            <signal name="toggled" handler="on_rbutton3_toggled" swapped="no"/>
          </object>
          <packing>
            <property name="expand">False</property>
            <property name="fill">True</property>
            <property name="position">6</property>
          </packing>
        </child>
      </object>
    </child>
    <child>
      <placeholder/>
```

```
    </child>
  </object>
</interface>
```

3.3.2 Python

```
#!/usr/bin/python
# -*- coding: utf-8 -*-

import gi
gi.require_version('Gtk','3.0')
from gi.repository import Gtk

class Handler:

    def on_window_destroy(self,*args):
        Gtk.main_quit()

    def on_switch_state_set(self,widget,state):
        if state is True:
            print("switch is on")
        else:
            print("switch is off")

    def on_cbutton_toggled(self,widget):
        if widget.get_active():
            print("checkbox checked")
        else:
            print("checkbox unchecked")

    def on_rbutton1_toggled(self,widget):
        if widget.get_active():
            print("radiobutton selection changed to 1")

    def on_rbutton2_toggled(self,widget):
        if widget.get_active():
            print("radiobutton selection changed to 2")

    def on_rbutton3_toggled(self,widget):
        if widget.get_active():
            print("radiobutton selection changed to 3")

class Example:

    def __init__(self):

        self.gladefile = "04_clickableelements.glade"
        self.builder = Gtk.Builder()
        self.builder.add_from_file(self.gladefile)
        self.builder.connect_signals(Handler())

        window = self.builder.get_object("window")
        window.show_all()

    def main(self):
        Gtk.main()

x = Example()
x.main()
```

KAPITEL 4

Stacks und Notebooks

Inhalte organisiert anzeigen

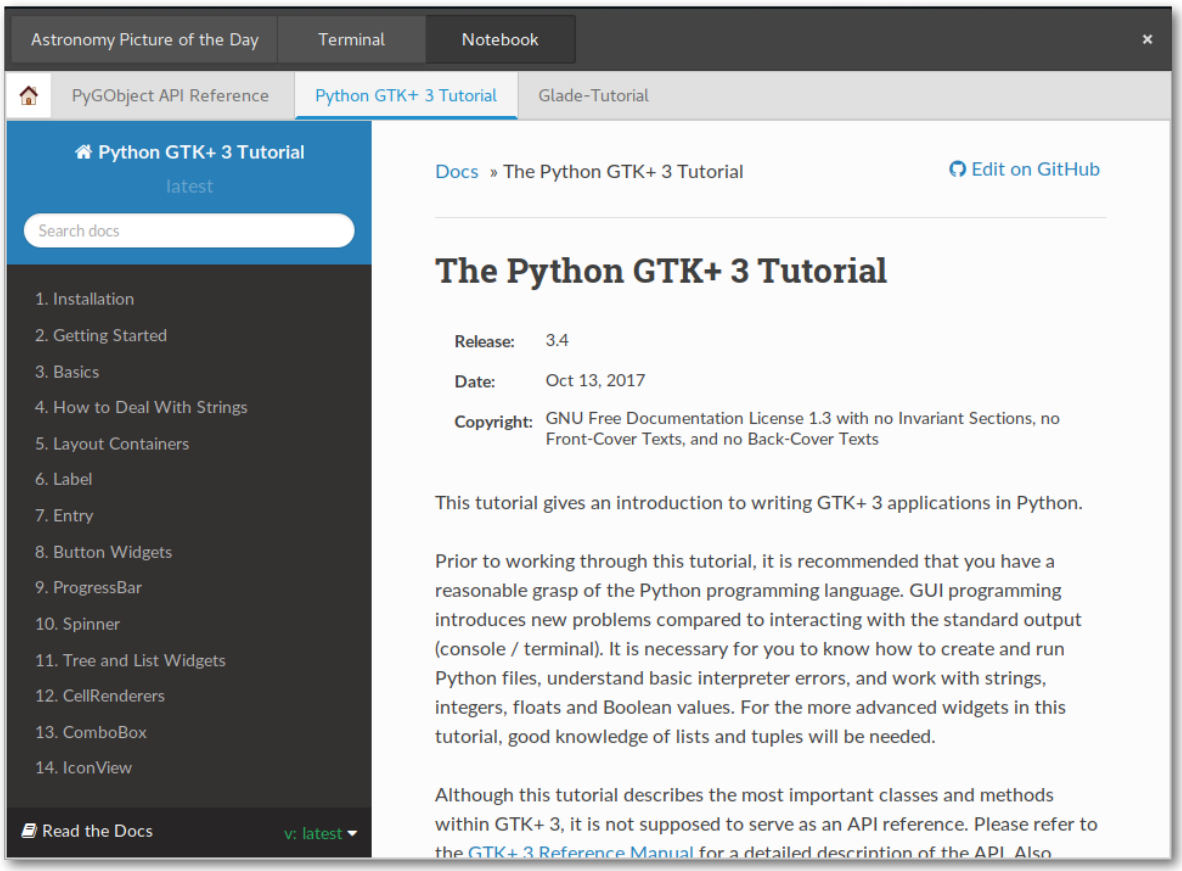
Gtk.Stack und *Gtk.Notebook* sind Layout-Container, die ihrerseits beliebige Widgets enthalten können.

Ein Notebook stellt ein mehrseitiges Layout mit klassischer Tab-Funktionalität zur Verfügung. Stacks bieten die gleiche Grundfunktionalität, nämlich mehrere Container innerhalb eines Widgets zu enthalten, zwischen denen man hin- und herschalten kann.

Der Hauptunterschied besteht darin, dass das Bedienelement des Stacks als separates Widget verwendet werden muss (*Gtk.StackSwitcher*). Verschiedene Stackswitcher können dabei auf den selben Stack zugreifen. Weiterhin lassen sich Stackswitcher in Headerbars platzieren, außerdem werden animierte Überblenden zwischen den Stack-Seiten unterstützt.

Stacks passen sich subjektiv besser in die GNOME-Umgebung ein, bieten aber nicht ganz so große Funktionalität wie Notebooks.

Das Beispiel enhält ein Fenster mit Stack, in dessen dritter Seite ein Notebook enthalten ist, das verschiedene Webseiten anzeigt.



4.1 Glade

4.1.1 Stack

Ein Stack, zu finden in der Sidebar unter „Container“, und dessen Unterseiten lassen sich einfach in Glade erstellen und bearbeiten. Als Unterwidgets kommen im Beispiel *Gtk.Image*, Vte.Terminal (exterminate)_ und *Gtk.Notebook* zum Einsatz.

Das Stackswitcher-Widget befindet sich unter „Steuerung und Anzeige“ und wird der Headerbar hinzugefügt. Es kann aber auch in reguläre Container-Widgets wie einer Box platziert und die Unterseiten horizontal oder vertikal angezeigt werden. Unter *„Allgemein > Stapel“* wird der Stack ausgewählt, auf den sich das Widget beziehen soll. Die anzuzeigende Seitenbezeichnung wird im jeweiligen Stack-Unterwidget unter *„Packen > Titel“* festgelegt. Dies funktioniert aber erst, nachdem einer Unterseite ein Widget hinzugefügt wurde. Standardmäßig ist dies zunächst leer.

4.1.2 Notebook

Das Notebook findet sich ebenfalls unter „Container“. Die Steuerungseinheit des Tabs ist ein bei Erstellung einer Seite generiertes Label-Child-Widget. Als Container-Widgets der Unterseiten werden hier *Gtk.ScrolledWindows* verwendet. Diese benötigt man auch z.B. für die Anzeige von (längeren) Tabellen (siehe auch Artikel zu List-/TreeStores Nr. 1 (uberlistet)_ und Nr. 2 (ansichtssache)_).

Die Tab-Leiste des Notebooks bietet die Möglichkeit, sowohl am Anfang als auch am Ende ein Container-Widget bereitzustellen (unter *„Allgemein > Start-Aktion/End-Aktion“*), in dem zum Beispiel feste Buttons untergebracht werden können. Im Beispiel wird am Anfang ein „Home“-Button eingerichtet.

4.2 Python

Für das Umherschalten zwischen Stack-Unterseiten und Notebook-Tabs werden keine Signale benötigt. Im Beispiel werden nur zwei Signale benötigt, einmal für das Abfangen des „exit“-Kommandos innerhalb des Terminals und für den Button in der Notebook-Tableiste.

4.2.1 WebKit2

Die Darstellung von Webseiten wird im Beispiel von WebKit2 erledigt. Das zentrale Modul dabei ist `WebKit2.WebView`. Ein neues WebView-Objekt selbst ist bereits ein scrollbares Gtk+-Widget in einem *Gtk.Viewport*, muss also laut API-Referenz nicht mehr in ein *Gtk.ScrolledWindow* platziert werden. Dies funktionierte im Test zwar für *Gtk.Stack*, nicht aber für *Gtk.Notebook*, weshalb dort trotzdem als „Unterlage“ ein ScrolledWindow-Widget verwendet wird.

Das WebView-Widget wird nach folgendem Muster erstellt:

```
#create new WebView widget
webview = WebKit2.WebView()
#send URL to widget
webview.load_uri("http://google.com")
#add webview to notebook
notebook.add(webview)
#add webview to stack
stack.add_titled(webview, name, "StackSwitcher title")

webview.show()
```

4.3 Listings

4.3.1 Python

```
#!/usr/bin/python
# -*- coding: utf-8 -*-

import sys
import urllib.request

import gi
gi.require_version('Gtk','3.0')
gi.require_version('Vte', '2.91')
gi.require_version('WebKit2', '4.0')

from gi.repository import Gtk, Gio, Vte, GObject, GLib, WebKit2

class Handler:

    def on_term_child_exited(self, widget, event):
        #reset and setup terminal on exit command
        widget.reset(True, True)
        app.stack_console()

    def on_home_button_clicked(self, widget):
        #reload given URL in current tab
        page = app.obj("notebook").get_current_page()
        app.nbtabs[page][2].load_uri(app.nbtabs[page][1])

class ExampleApp:
```

```
    def __init__(self):

        self.app = Gtk.Application.new("org.application.test", Gio.
↪ApplicationFlags(0))
        self.app.connect("activate", self.on_app_activate)

    def on_app_activate(self, app):
        GObject.type_register(Vte.Terminal)
        builder = Gtk.Builder()
        builder.add_from_file("21_stacknotebook.glade")
        builder.connect_signals(Handler())
        self.obj = builder.get_object
        self.obj("window").set_application(app)
        self.obj("window").set_wmclass("Tutorial application","Tutorial application
↪")
        self.obj("window").show_all()

        #get window content
        self.stack_image()
        self.stack_console()
        self.stack_notebook()

    def run(self, argv):
        self.app.run(argv)

    def stack_image(self):
        #download and show NASA Astonomy Picture of the Day
        URL = "https://apod.nasa.gov"
        source = urllib.request.urlopen(URL).read().decode("utf-8")
        img_start = source.find("<IMG SRC=")
        img_end = source.find("alt=")
        img = source[img_start+10:img_end-2]
        IMGURL = "https://apod.nasa.gov/apod/"+img
        urllib.request.urlretrieve(IMGURL, "apod.jpg")
        self.obj("image").set_from_file("apod.jpg")

    def stack_console(self):
        #setup terminal
        self.obj("term").spawn_sync(
            Vte.PtyFlags.DEFAULT,
            None,
            ["/bin/bash"],
            None,
            GLib.SpawnFlags.DEFAULT,
            None,
            None,
            )

    def stack_notebook(self):

        self.nbtabs = [
                    ["gi_doc", "https://lazka.github.io/pgi-docs/"],
                    ["gtk_tut", "http://python-gtk-3-tutorial.readthedocs.io/en/
↪latest/index.html"],
                    ["glade_tut", "https://encarsia.github.io/posts/tutorial-reihe-
↪glade/"]
                    ]

        for tab in self.nbtabs:
            webview = WebKit2.WebView()
            tab.append(webview)
            webview.load_uri(tab[1])
```

```
            self.obj(tab[0]).add(webview)
            webview.show()

app = ExampleApp()
app.run(sys.argv)
```

4.3.2 Glade

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- Generated with glade 3.20.1 -->
<interface>
  <requires lib="gtk+" version="3.20"/>
  <requires lib="vte-2.91" version="0.50"/>
  <object class="GtkImage" id="image1">
    <property name="visible">True</property>
    <property name="can_focus">False</property>
    <property name="icon_name">go-home</property>
  </object>
  <object class="GtkApplicationWindow" id="window">
    <property name="width_request">800</property>
    <property name="height_request">600</property>
    <property name="can_focus">False</property>
    <child>
      <object class="GtkStack" id="stack">
        <property name="visible">True</property>
        <property name="can_focus">False</property>
        <property name="transition_type">crossfade</property>
        <child>
          <object class="GtkImage" id="image">
            <property name="visible">True</property>
            <property name="can_focus">False</property>
            <property name="stock">gtk-missing-image</property>
          </object>
          <packing>
            <property name="name">page0</property>
            <property name="title" translatable="yes">Astronomy Picture of the Day
↪</property>
          </packing>
        </child>
        <child>
          <object class="VteTerminal" id="term">
            <property name="visible">True</property>
            <property name="can_focus">True</property>
            <property name="hscroll_policy">natural</property>
            <property name="vscroll_policy">natural</property>
            <property name="encoding">UTF-8</property>
            <property name="scroll_on_keystroke">True</property>
            <property name="scroll_on_output">False</property>
            <signal name="child-exited" handler="on_term_child_exited" swapped="no
↪"/>
          </object>
          <packing>
            <property name="name">page3</property>
            <property name="title" translatable="yes">Terminal</property>
            <property name="position">1</property>
          </packing>
        </child>
        <child>
          <object class="GtkNotebook" id="notebook">
            <property name="visible">True</property>
            <property name="can_focus">True</property>
```

```
            <child>
              <object class="GtkScrolledWindow" id="gi_doc">
                <property name="visible">True</property>
                <property name="can_focus">True</property>
                <property name="shadow_type">in</property>
                <child>
                  <placeholder/>
                </child>
              </object>
            </child>
            <child type="tab">
              <object class="GtkLabel">
                <property name="visible">True</property>
                <property name="can_focus">False</property>
                <property name="label" translatable="yes">PyGObject API Reference
↪</property>
              </object>
              <packing>
                <property name="tab_fill">False</property>
              </packing>
            </child>
            <child>
              <object class="GtkScrolledWindow" id="gtk_tut">
                <property name="visible">True</property>
                <property name="can_focus">True</property>
                <property name="shadow_type">in</property>
                <child>
                  <placeholder/>
                </child>
              </object>
              <packing>
                <property name="position">1</property>
              </packing>
            </child>
            <child type="tab">
              <object class="GtkLabel">
                <property name="visible">True</property>
                <property name="can_focus">False</property>
                <property name="label" translatable="yes">Python GTK+ 3 Tutorial</
↪property>
              </object>
              <packing>
                <property name="position">1</property>
                <property name="tab_fill">False</property>
              </packing>
            </child>
            <child>
              <object class="GtkScrolledWindow" id="glade_tut">
                <property name="visible">True</property>
                <property name="can_focus">True</property>
                <property name="shadow_type">in</property>
                <child>
                  <placeholder/>
                </child>
              </object>
              <packing>
                <property name="position">2</property>
              </packing>
            </child>
            <child type="tab">
              <object class="GtkLabel">
                <property name="visible">True</property>
                <property name="can_focus">False</property>
```

```
                <property name="label" translatable="yes">Glade-Tutorial</property>
              </object>
              <packing>
                <property name="position">2</property>
                <property name="tab_fill">False</property>
              </packing>
            </child>
            <child type="action-start">
              <object class="GtkButton" id="home_button">
                <property name="visible">True</property>
                <property name="can_focus">True</property>
                <property name="receives_default">True</property>
                <property name="image">image1</property>
                <property name="always_show_image">True</property>
                <signal name="clicked" handler="on_home_button_clicked" swapped="no
↪"/>
              </object>
              <packing>
                <property name="tab_fill">False</property>
              </packing>
            </child>
          </object>
          <packing>
            <property name="name">page1</property>
            <property name="title" translatable="yes">Notebook</property>
            <property name="position">2</property>
          </packing>
        </child>
      </object>
    </child>
    <child type="titlebar">
      <object class="GtkHeaderBar">
        <property name="visible">True</property>
        <property name="can_focus">False</property>
        <property name="show_close_button">True</property>
        <child>
          <object class="GtkStackSwitcher">
            <property name="visible">True</property>
            <property name="can_focus">False</property>
            <property name="stack">stack</property>
          </object>
        </child>
      </object>
    </child>
  </object>
</interface>
```