Glade-Tutorial mit PyGObject

Release 0.1

Anke K. (encarsia)

31.10.2017

Inhalt

Hinweis: Dies ist ein halbautomatisch generiertes E-Book aus den Artikeln der Tutorialreihe, die sich auf meiner GitHub Page befinden.

- Tutorial-Übersichtsseite
- Verzeichnis der Beispieldateien
- Download-Seite

Generiert am 31.10.2017 mit Sphinx.

KAPITEL 1

Tutorial-Reihe zu Glade

1.1 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.

1.2 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-Gänge-Menü*)
- Fortschrittsbalken und Levelbar (*Bars*)
- CSS (*Alles so schön bunt hier*)
- Spinbutton und Combobox (*Qual der Wahl*)
- ListStore und TreeView (*Überlistet*)
- TreeStore mit Sortierung und Filterung (*Ansichtssache*)
- Lokalisation mit locale und gettext (*Romani ite domum*)
- VTE-Terminal (*Exterminate!*)

- Dialoge (*Dialoge*)
- Programm als eigenständige GTK+-Anwendung (*Selbständig*)
- Icon, Headerbar und Kommandozeilenoptionen (*Desktopintegrationsbemühungen*)
- Dateiauswahldialog (*Dateiauswahldialog*)
- GSettings-Konfigurationssystem (*Das Konfigurationssystem GSettings*)
- Mediaplayer mit GStreamer (*Mediaplayer mit GStreamer*)
- Mediaplayer mit LibVLC (*Mediaplayer mit VLC*)
- Stack und Notebook (*Stacks und Notebooks*)

1.3 Dateien

- Verzeichnis der Beispieldateien: encarsia.github.io/listings
- E-Book

1.4 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).

1.5 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!)

1.6 Todo

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

KAPITEL 2

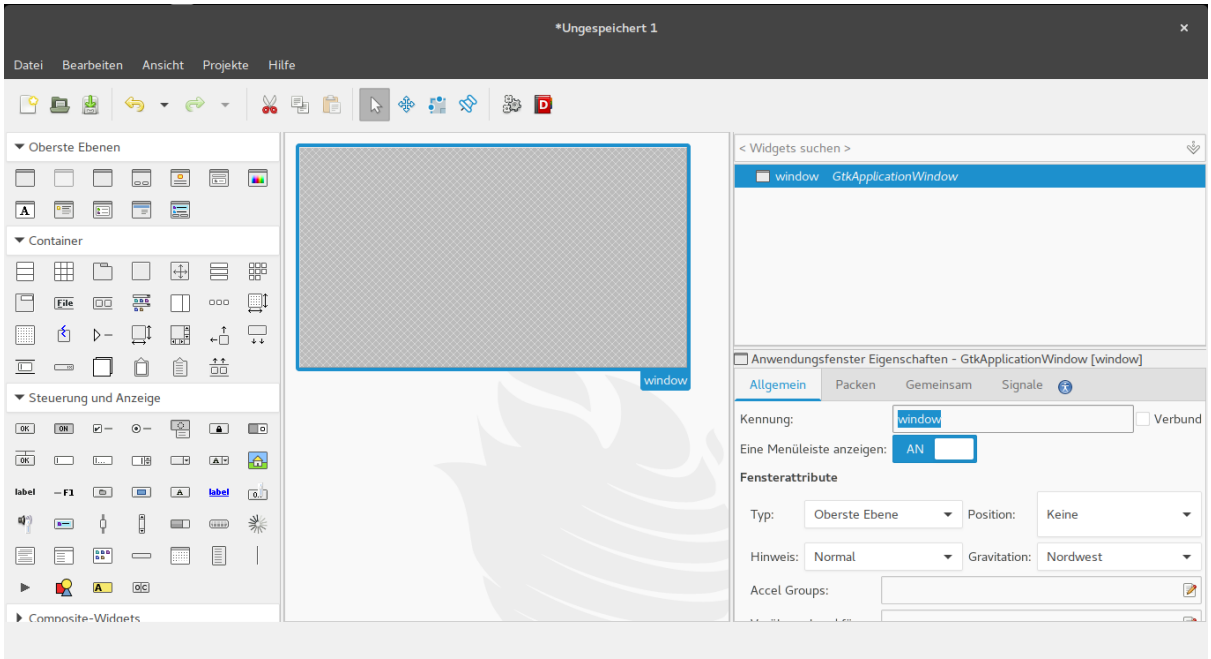
Fenster mit Aussicht

Minimalbeispiel

2.1 Glade

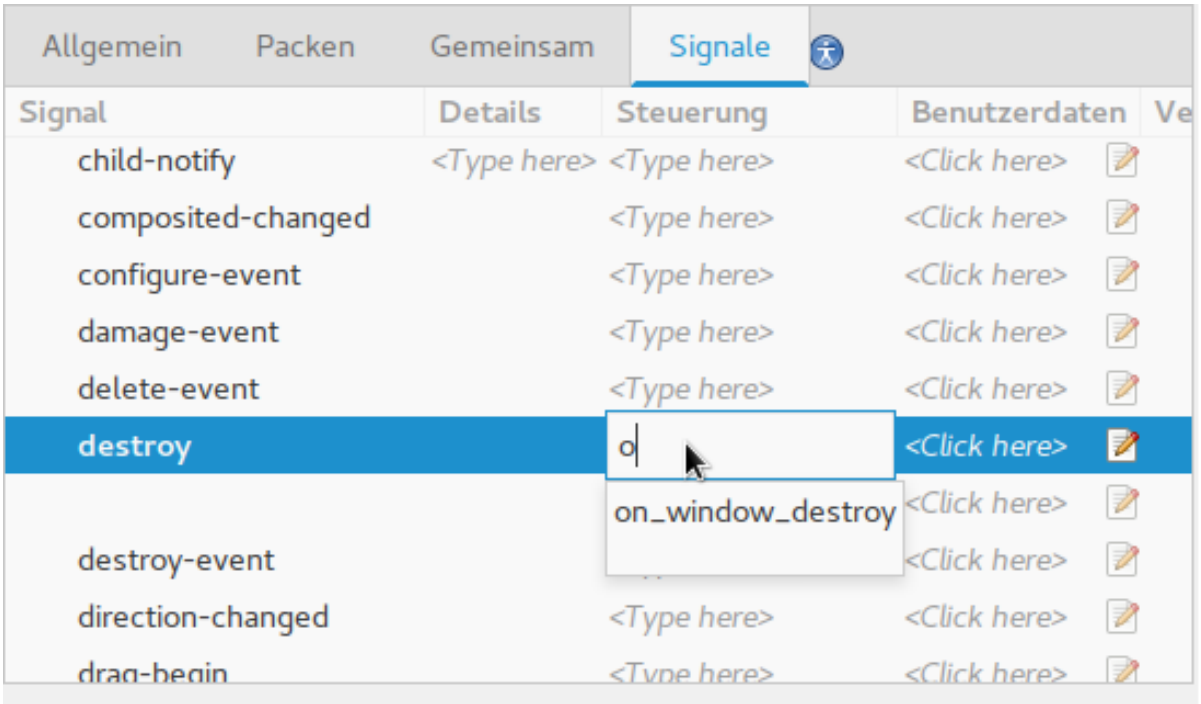
Nach dem Start präsentiert sich Glade dreigeteilt, links ist die Fenster-/Widget-Auswahl, in der Mitte die Projektansicht und rechts eine Baumansicht des Projekts, im unteren Bereich können Eigenschaften und Signale editiert werden.

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

Push the button

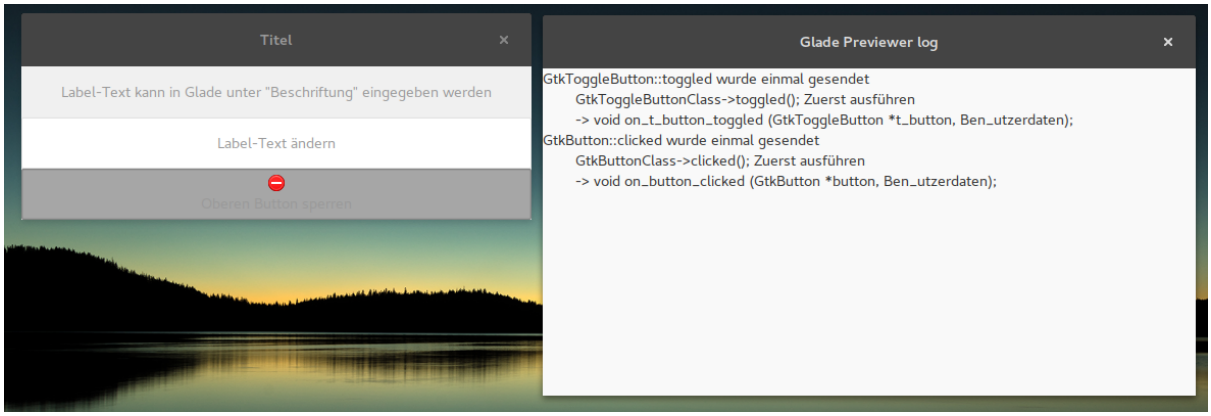
Buttons und Labels

3.1 Glade

Ausgehend vom letzten Beispiel (*Fenster mit Aussicht*) werden nun ein paar Elemente hinzugefügt, ein Label, ein Button und ein Togglebutton. Jedes Anzeigen- oder Steuerungselement benötigt je ein Container. In diesem Beispiel werden vertikale Boxen angelegt, diese lassen sich jederzeit erweitern, es ist auch möglich, Container ineinander zu verschachteln.

Den Elementen Button und Togglebutton wird auf *clicked* bzw. *toggled* ein Signal zugewiesen. Label dient nur der Anzeige von Text, hier wird kein Signal benötigt.

In der Vorschauansicht kann man testen, ob die korrekte Reaktion ausgelöst wird.



3.2 Python

Ein Klick auf den Button soll in der Labelanzeige einen anderen Text anzeigen, hier wird zufällig ein Element aus einer Liste ausgewählt.

Alle *Gtk.Builder*-Objekte können über die Funktion `get_object` angesprochen werden:

```
Gtk.Builder.get_object("name").funktion(options)

#Beispiel GtkLabel
Gtk.Builder.get_object("label_name").set_text("neuer Text")
```

Der Togglebutton soll die Labelanzeige leeren und Button inaktivieren und bei erneutem Klick wieder freigeben.

Der Zustand des Togglebuttons kann mit der Funktion `get_active()` abgerufen werden (gibt True/False zurück).

Abhängig vom verwendeten Widget erfordert die Signal-Funktion mindestens einen Parameter.

```
def on_t_button_toggled(self,widget):
    if widget.get_active():
        #do something
    else:
        #do something different
```

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="GtkImage" id="image1">
    <property name="visible">True</property>
    <property name="can_focus">False</property>
    <property name="stock">gtk-dialog-error</property>
  </object>
  <object class="GtkApplicationWindow" id="window">
    <property name="width_request">500</property>
    <property name="can_focus">False</property>
    <property name="title" translatable="yes">Titel</property>
    <property name="resizable">False</property>
    <signal name="destroy" handler="on_window_destroy" swapped="no"/>
    <child>
      <object class="GtkBox">
        <property name="width_request">200</property>
        <property name="visible">True</property>
        <property name="can_focus">False</property>
        <property name="orientation">vertical</property>
        <property name="homogeneous">True</property>
        <child>
          <object class="GtkLabel" id="label">
            <property name="visible">True</property>
            <property name="can_focus">False</property>
            <property name="label" translatable="yes">Label-Text kann in Glade␣
↪unter "Beschriftung" eingegeben werden</property>
          </object>
          <packing>
            <property name="expand">False</property>
            <property name="fill">True</property>
            <property name="position">0</property>
          </packing>
        </child>
        <child>
          <object class="GtkButton" id="button">
            <property name="label" translatable="yes">Label-Text ändern</property>
```

```
            <property name="visible">True</property>
            <property name="can_focus">True</property>
            <property name="receives_default">True</property>
            <signal name="clicked" handler="on_button_clicked" swapped="no"/>
          </object>
          <packing>
            <property name="expand">False</property>
            <property name="fill">True</property>
            <property name="position">1</property>
          </packing>
        </child>
        <child>
          <object class="GtkToggleButton" id="t_button">
            <property name="label" translatable="yes">Oberen Button sperren</
↪property>
            <property name="visible">True</property>
            <property name="can_focus">True</property>
            <property name="receives_default">True</property>
            <property name="image">image1</property>
            <property name="image_position">top</property>
            <property name="always_show_image">True</property>
            <signal name="toggled" handler="on_t_button_toggled" swapped="no"/>
          </object>
          <packing>
            <property name="expand">False</property>
            <property name="fill">True</property>
            <property name="position">2</property>
          </packing>
        </child>
      </object>
    </child>
    <child>
      <placeholder/>
    </child>
  </object>
</interface>
```

3.3.2 Python

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

import random
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_button_clicked(self,widget):
        new_text = random.choice(x.label_texts)
        x.builder.get_object("label").set_text(new_text)

    def on_t_button_toggled(self,widget):
        if widget.get_active():
            x.builder.get_object("label").set_text("")
            x.builder.get_object("button").set_sensitive(False)
        else:
```

```
            x.builder.get_object("button").set_sensitive(True)

class Example:

    def __init__(self):

        self.gladefile = "02_labelbutton.glade"

        self.label_texts = ["The things you used to own, now they own you.",
                            "I am Jack's complete lack of surprise. I am Jack's␣
↪Broken Heart.",
                            "On a long enough time line, the survival rate for␣
↪everyone drops to zero.",
                            "Sticking feathers up your butt does not make you a␣
↪chicken!",
                            "I am Jack's smirking revenge."]

        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()
```

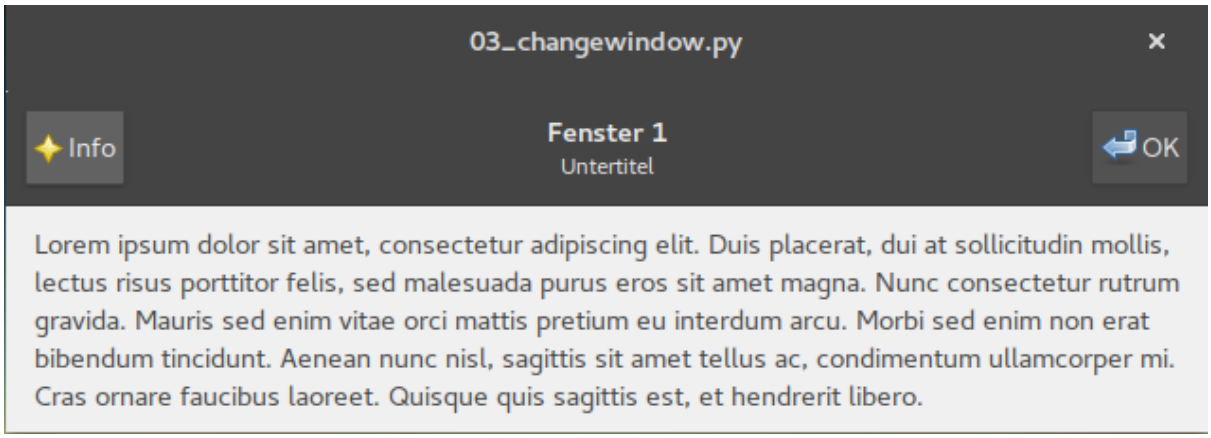
KAPITEL 4

Durchzug

Fenster und Dialoge öffnen und schließen

4.1 Glade

Mit Glade lassen sich verschiedene Fensterarten und Dialoge erstellen. Im Beispiel hat das Hauptfenster zwei Buttons, ein Button öffnet ein Info-Fenster, der andere schließt das Hauptfenster und öffnet ein anderes Fenster, das jeweils das gleiche tut.



Es werden insgesamt 7 Signale angelegt:

- **Fenster, jeweils**
  - Info-Button (Headerbar links): *clicked*
  - Wechsel-/“Ok“-Button (Headerbar rechts): *clicked*
  - Schließen/Beenden: *destroy*
- **Info-Dialog**
  - Schließen-Button: *destroy*

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- Generated with glade 3.20.0 -->
```

```
<interface>
  <requires lib="gtk+" version="3.20"/>
  <object class="GtkWindow" id="win1">
    <property name="can_focus">False</property>
    <property name="resizable">False</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_bottom">9</property>
        <property name="orientation">vertical</property>
        <property name="spacing">10</property>
        <child>
          <object class="GtkHeaderBar">
            <property name="visible">True</property>
            <property name="can_focus">False</property>
            <property name="title">Fenster 1</property>
            <property name="subtitle">Untertitel</property>
            <child>
              <object class="GtkButton" id="info_button1">
                <property name="label">gtk-about</property>
                <property name="visible">True</property>
                <property name="can_focus">True</property>
                <property name="receives_default">True</property>
                <property name="use_stock">True</property>
                <property name="always_show_image">True</property>
                <signal name="clicked" handler="on_info_button_clicked" swapped="no
↪"/>
              </object>
              <packing>
                <property name="position">1</property>
              </packing>
            </child>
            <child>
              <object class="GtkButton" id="button1">
                <property name="label">gtk-ok</property>
                <property name="visible">True</property>
                <property name="can_focus">True</property>
                <property name="receives_default">True</property>
                <property name="use_stock">True</property>
                <property name="always_show_image">True</property>
                <signal name="clicked" handler="on_button1_clicked" swapped="no"/>
              </object>
              <packing>
                <property name="pack_type">end</property>
                <property name="position">1</property>
              </packing>
            </child>
          </object>
          <packing>
            <property name="expand">False</property>
            <property name="fill">True</property>
            <property name="position">0</property>
          </packing>
        </child>
        <child>
          <object class="GtkLabel" id="label1">
            <property name="visible">True</property>
            <property name="can_focus">False</property>
            <property name="label" translatable="yes">Lorem ipsum dolor sit amet,␣
↪consectetur adipiscing elit. Duis placerat, dui at sollicitudin mollis, lectus␣
↪risus porttitor felis, sed malesuada purus eros sit amet magna. Nunc consectetur␣
↪rutrum gravida. Mauris sed enim vitae orci mattis pretium eu interdum arcu.␣
↪Morbi sed enim non erat bibendum tincidunt. Aenean nunc nisl, sagittis sit amet␣
↪eleifend ac, condimentum ullamcorper mi. Cras ornare faucibus laoreet. Quisque␣
↪quis sagittis est, et hendrerit libero.</property>
```

```
            <property name="wrap">True</property>
            <property name="max_width_chars">80</property>
          </object>
          <packing>
            <property name="expand">False</property>
            <property name="fill">True</property>
            <property name="position">1</property>
          </packing>
        </child>
      </object>
    </child>
    <child type="titlebar">
      <placeholder/>
    </child>
  </object>
  <object class="GtkAboutDialog" id="about_dialog">
    <property name="can_focus">False</property>
    <property name="title" translatable="yes">Info</property>
    <property name="resizable">False</property>
    <property name="modal">True</property>
    <property name="window_position">center-on-parent</property>
    <property name="destroy_with_parent">True</property>
    <property name="type_hint">dialog</property>
    <property name="deletable">False</property>
    <property name="transient_for">win1</property>
    <property name="program_name">Info Dialog</property>
    <property name="version">0.1</property>
    <property name="comments" translatable="yes">Platz für mehr Blabla</property>
    <property name="website">www.example.com</property>
    <property name="logo_icon_name">image-missing</property>
    <child internal-child="vbox">
      <object class="GtkBox">
        <property name="can_focus">False</property>
        <property name="orientation">vertical</property>
        <property name="spacing">2</property>
        <child internal-child="action_area">
          <object class="GtkButtonBox">
            <property name="can_focus">False</property>
            <property name="layout_style">end</property>
            <child>
              <object class="GtkButton" id="close_button">
                <property name="label">gtk-close</property>
                <property name="visible">True</property>
                <property name="can_focus">True</property>
                <property name="receives_default">True</property>
                <property name="use_stock">True</property>
                <signal name="clicked" handler="on_close_button_clicked" swapped=
↪"no"/>
              </object>
              <packing>
                <property name="expand">True</property>
                <property name="fill">True</property>
                <property name="position">0</property>
              </packing>
            </child>
          </object>
          <packing>
            <property name="expand">False</property>
            <property name="fill">False</property>
            <property name="position">0</property>
          </packing>
        </child>
      </object>
```

```
    </child>
    <child>
      <placeholder/>
    </child>
  </object>
  <object class="GtkWindow" id="win2">
    <property name="can_focus">False</property>
    <property name="resizable">False</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_bottom">9</property>
        <property name="orientation">vertical</property>
        <property name="spacing">10</property>
        <child>
          <object class="GtkHeaderBar">
            <property name="visible">True</property>
            <property name="can_focus">False</property>
            <property name="title">Fenster 2 </property>
            <property name="subtitle">Anderer Untertitel</property>
            <child>
              <object class="GtkButton" id="info_button2">
                <property name="label">gtk-about</property>
                <property name="visible">True</property>
                <property name="can_focus">True</property>
                <property name="receives_default">True</property>
                <property name="use_stock">True</property>
                <property name="always_show_image">True</property>
                <signal name="clicked" handler="on_info_button_clicked" swapped="no
↪"/>
              </object>
              <packing>
                <property name="position">1</property>
              </packing>
            </child>
            <child>
              <object class="GtkButton" id="button2">
                <property name="label">gtk-ok</property>
                <property name="visible">True</property>
                <property name="can_focus">True</property>
                <property name="receives_default">True</property>
                <property name="use_stock">True</property>
                <property name="always_show_image">True</property>
                <signal name="clicked" handler="on_button2_clicked" swapped="no"/>
              </object>
              <packing>
                <property name="pack_type">end</property>
                <property name="position">1</property>
              </packing>
            </child>
          </object>
          <packing>
            <property name="expand">False</property>
            <property name="fill">True</property>
            <property name="position">0</property>
          </packing>
        </child>
        <child>
          <object class="GtkLabel" id="label2">
            <property name="visible">True</property>
            <property name="can_focus">False</property>
```

```
            <property name="label" translatable="yes">In sagittis purus nec␣
↪eleifend dignissim. Curabitur venenatis eleifend leo ac tincidunt. Etiam ut␣
↪consequat neque. Aenean in libero placerat, iaculis est quis, blandit nulla.␣
↪Nulla euismod cursus nisl efficitur imperdiet. Sed vel augue vitae dui congue␣
↪eleifend id eu libero. Cras laoreet velit nibh, et pharetra ante pharetra id.␣
↪Nullam mollis arcu a nibh pulvinar, sed volutpat quam facilisis. Vivamus quis␣
↪leo quis orci aliquam fermentum. Donec varius accumsan nisi eu ullamcorper.␣
↪Integer condimentum, eros sit amet convallis vehicula, elit leo mattis risus,␣
↪quis suscipit turpis nibh sed nulla. Sed id justo ut magna commodo eleifend.␣
↪Praesent nunc arcu, elementum eu dolor nec, rutrum molestie mauris.</property>
            <property name="wrap">True</property>
            <property name="max_width_chars">80</property>
          </object>
          <packing>
            <property name="expand">False</property>
            <property name="fill">True</property>
            <property name="position">1</property>
          </packing>
        </child>
      </object>
    </child>
    <child type="titlebar">
      <placeholder/>
    </child>
  </object>
</interface>
```

4.2 Python

Die entscheidenden Funktionen in der Handhabung von Fenstern sind

```
#Fenster anzeigen
Gtk.Builder.get_object("name").show_all()
#Fenster ausblenden, kann mit show_all() reaktiviert werden
Gtk.Builder.get_object("name").hide_on_delete()
#Fenster schließen, Gtk wird dabei beendet
Gtk.main_quit()
```

Die Buttons zum Öffnen des Info-Dialogs und zum Beenden des Programms führen die jeweils identische Funktion aus; es werden demzufolge nur 5 Funktionen in der Handler-Klasse benötigt.

Das vollständige Beispiel ist dann:

```
#!/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_button1_clicked(self,widget):
        x.window.hide_on_delete()
        x.set_window("win2")

    def on_button2_clicked(self,widget):
```

```
        x.window.hide_on_delete()
        x.set_window("win1")

    def on_info_button_clicked(self,widget):
        x.about.show_all()

    def on_close_button_clicked(self,widget):
        x.about.hide_on_delete()

class Example:

    def __init__(self):

        self.builder = Gtk.Builder()
        self.builder.add_from_file("03_changewindow.glade")
        self.builder.connect_signals(Handler())

        self.about = self.builder.get_object("about_dialog")

        self.set_window("win1")

    def set_window(self,win):
        self.window = self.builder.get_object(win)
        self.window.show_all()

    def main(self):
        Gtk.main()

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

Siehe auch Fortsetzung-Artikel zu Dialogen (*Dialoge*).

KAPITEL 5

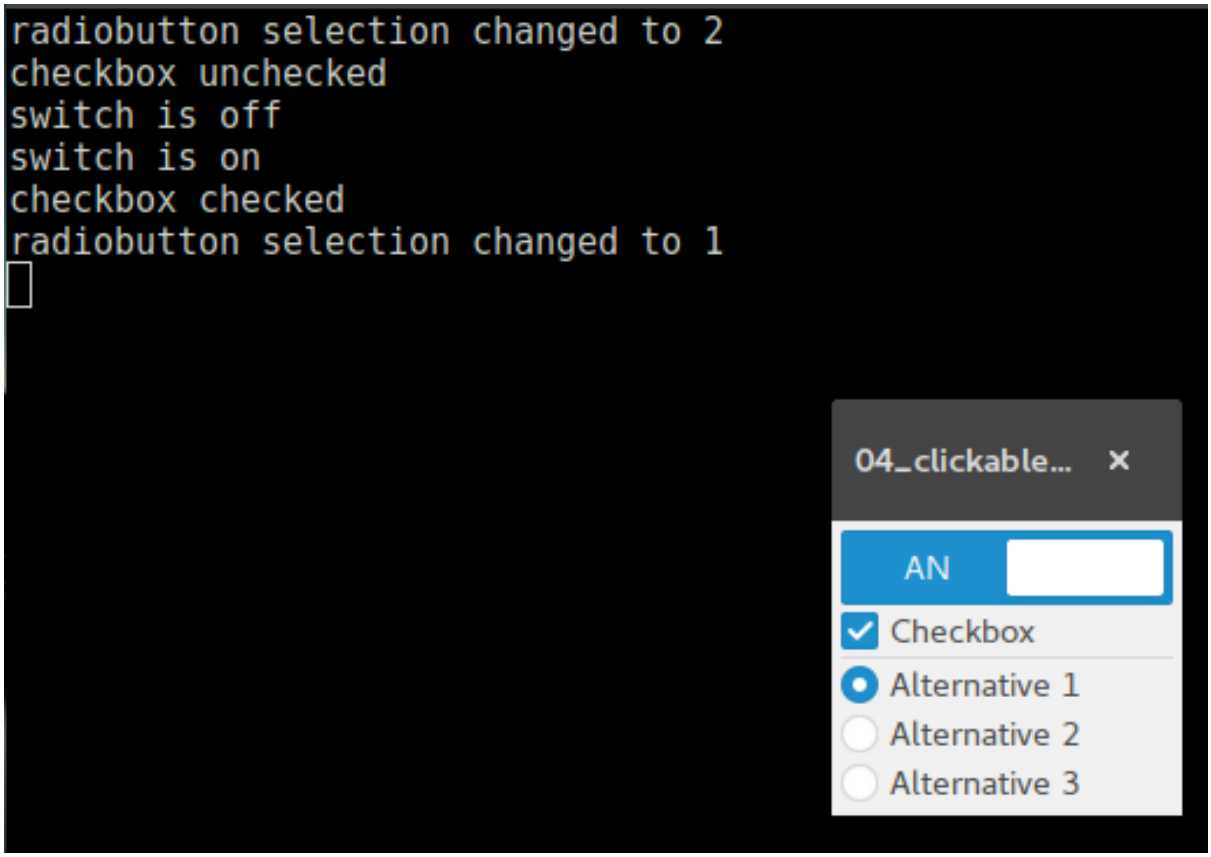
Clickbaiting

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.



5.1 Glade

5.1.1 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.

5.1.2 Checkbox

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

5.1.3 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.

5.2 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")
```

5.3 Listings

5.3.1 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>
```

5.3.2 Python

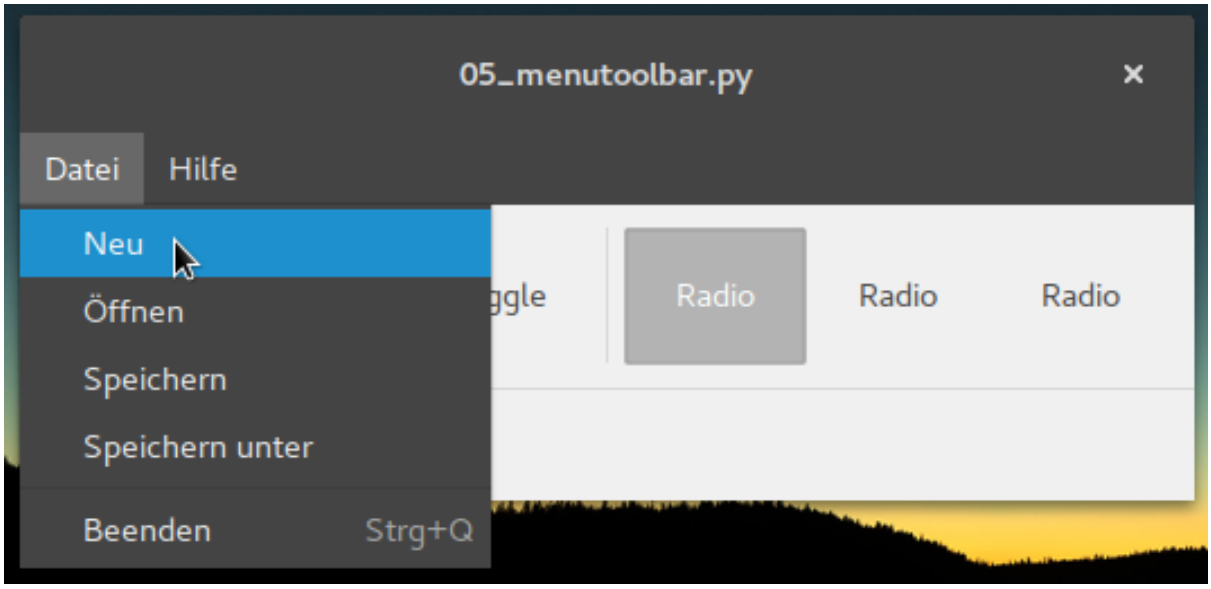
```
#!/usr/bin/python
# -*- coding: utf-8 -*-

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 6

Drei-Gänge-Menü

Menüs, Toolbars und Statusbars

6.1 Glade



6.1.1 Menü

Das Menü-Element findet man unter den Containerelementen, es benötigt aber selbst eine leere Box. Es wird zunächst ein Standard-Menü angelegt, das sich bequem über *„Edit. . . “* in einem separaten Fenster bearbeiten lässt.

Sehr simpel erfolgt das Anlegen von Shortcuts im „Edit“-Fenster unter *„Hierarchie* > *Eigenschaften* > *Tastenkürzel“*. Dort weist man der Tastenkombination ein Signal zu (im Falle von Menüeinträgen *activate*). Examplarisch wurden Shortcuts zum Beenden (Strg+Q) und zum Einblenden des About-Dialogs (Strg+I) angelegt.

6.1.2 Toolbar

Toolbars können verschiedene Widgets wie Buttons, Togglebuttons, Radiobuttons oder (Unter-)Menüs enthalten. Die Erstellung und Bearbeitung erfolgt analog zum Menü über „Edit. . . “.

6.1.3 Statusbar

In der Statusbar können kurze Meldungen/Nachrichten eingeblendet werden. Die Meldungen werden analog zu einer Liste behandelt, das Widget bietet die Funktionen `push` und `pop`.

6.2 Python

Um Nachrichten an die Statusbar zu senden, bedient man sich einfach der Funktion

```
widget.push(content_id,message)
```

Wenn man Meldungen ausschließlich „obendrauf“ einblendet, kann man als content_id eine beliebige Zahl angeben, zum Beispiel `0`.

6.3 Listings

6.3.1 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="resizable">False</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_bottom">9</property>
        <property name="orientation">vertical</property>
        <child>
          <object class="GtkMenuBar">
            <property name="visible">True</property>
            <property name="can_focus">False</property>
            <child>
              <object class="GtkMenuItem">
                <property name="visible">True</property>
                <property name="can_focus">False</property>
                <property name="label" translatable="yes">_Datei</property>
                <property name="use_underline">True</property>
                <child type="submenu">
                  <object class="GtkMenu">
                    <property name="visible">True</property>
                    <property name="can_focus">False</property>
                    <child>
                      <object class="GtkImageMenuItem">
                        <property name="label">gtk-new</property>
                        <property name="visible">True</property>
                        <property name="can_focus">False</property>
```

```
                        <property name="use_underline">True</property>
                        <property name="use_stock">True</property>
                        <signal name="activate" handler="on_nothing_here" swapped=
↪"no"/>
                      </object>
                    </child>
                    <child>
                      <object class="GtkImageMenuItem">
                        <property name="label">gtk-open</property>
                        <property name="visible">True</property>
                        <property name="can_focus">False</property>
                        <property name="use_underline">True</property>
                        <property name="use_stock">True</property>
                        <signal name="activate" handler="on_nothing_here" swapped=
↪"no"/>
                      </object>
                    </child>
                    <child>
                      <object class="GtkImageMenuItem">
                        <property name="label">gtk-save</property>
                        <property name="visible">True</property>
                        <property name="can_focus">False</property>
                        <property name="use_underline">True</property>
                        <property name="use_stock">True</property>
                        <signal name="activate" handler="on_nothing_here" swapped=
↪"no"/>
                      </object>
                    </child>
                    <child>
                      <object class="GtkImageMenuItem">
                        <property name="label">gtk-save-as</property>
                        <property name="visible">True</property>
                        <property name="can_focus">False</property>
                        <property name="use_underline">True</property>
                        <property name="use_stock">True</property>
                        <signal name="activate" handler="on_nothing_here" swapped=
↪"no"/>
                      </object>
                    </child>
                    <child>
                      <object class="GtkSeparatorMenuItem">
                        <property name="visible">True</property>
                        <property name="can_focus">False</property>
                      </object>
                    </child>
                    <child>
                      <object class="GtkImageMenuItem">
                        <property name="label">gtk-quit</property>
                        <property name="visible">True</property>
                        <property name="can_focus">False</property>
                        <property name="use_underline">True</property>
                        <property name="use_stock">True</property>
                        <signal name="activate" handler="on_window_destroy"␣
↪swapped="no"/>
                        <accelerator key="q" signal="activate" modifiers="GDK_
↪CONTROL_MASK"/>
                      </object>
                    </child>
                  </object>
                </child>
              </object>
            </child>
            <child>
```

```
            <object class="GtkMenuItem">
              <property name="visible">True</property>
              <property name="can_focus">False</property>
              <property name="label" translatable="yes">_Hilfe</property>
              <property name="use_underline">True</property>
              <child type="submenu">
                <object class="GtkMenu">
                  <property name="visible">True</property>
                  <property name="can_focus">False</property>
                  <child>
                    <object class="GtkImageMenuItem" id="menu_info">
                      <property name="label">gtk-about</property>
                      <property name="visible">True</property>
                      <property name="can_focus">False</property>
                      <property name="use_underline">True</property>
                      <property name="use_stock">True</property>
                      <signal name="activate" handler="on_info_button_clicked"␣
↪swapped="no"/>
                      <accelerator key="i" signal="activate" modifiers="GDK_
↪CONTROL_MASK"/>
                    </object>
                  </child>
                </object>
              </child>
            </object>
          </child>
        </object>
        <packing>
          <property name="expand">False</property>
          <property name="fill">True</property>
          <property name="position">1</property>
        </packing>
      </child>
      <child>
        <object class="GtkToolbar">
          <property name="visible">True</property>
          <property name="can_focus">False</property>
          <property name="toolbar_style">both</property>
          <child>
            <object class="GtkToolButton">
              <property name="visible">True</property>
              <property name="can_focus">False</property>
              <property name="label" translatable="yes">toolbutton</property>
              <property name="use_underline">True</property>
              <property name="stock_id">gtk-yes</property>
            </object>
            <packing>
              <property name="expand">False</property>
              <property name="homogeneous">True</property>
            </packing>
          </child>
          <child>
            <object class="GtkMenuToolButton">
              <property name="visible">True</property>
              <property name="can_focus">False</property>
              <property name="label" translatable="yes">menu</property>
              <property name="use_underline">True</property>
              <property name="stock_id">gtk-print</property>
              <child type="menu">
                <object class="GtkMenu">
                  <property name="visible">True</property>
                  <property name="can_focus">False</property>
                  <child>
```

```
                        <object class="GtkMenuItem">
                          <property name="visible">True</property>
                          <property name="can_focus">False</property>
                          <property name="label" translatable="yes">submenuitem 1</
↪property>
                          <property name="use_underline">True</property>
                        </object>
                      </child>
                      <child>
                        <object class="GtkMenuItem">
                          <property name="visible">True</property>
                          <property name="can_focus">False</property>
                          <property name="label" translatable="yes">submenuitem 2</
↪property>
                          <property name="use_underline">True</property>
                        </object>
                      </child>
                    </object>
                  </child>
                </object>
                <packing>
                  <property name="expand">False</property>
                  <property name="homogeneous">True</property>
                </packing>
              </child>
              <child>
                <object class="GtkToggleToolButton">
                  <property name="visible">True</property>
                  <property name="can_focus">False</property>
                  <property name="label" translatable="yes">Toggle</property>
                  <property name="use_underline">True</property>
                </object>
                <packing>
                  <property name="expand">False</property>
                  <property name="homogeneous">True</property>
                </packing>
              </child>
              <child>
                <object class="GtkSeparatorToolItem">
                  <property name="visible">True</property>
                  <property name="can_focus">False</property>
                </object>
                <packing>
                  <property name="expand">False</property>
                  <property name="homogeneous">True</property>
                </packing>
              </child>
              <child>
                <object class="GtkRadioToolButton" id="tb_radio1">
                  <property name="visible">True</property>
                  <property name="can_focus">False</property>
                  <property name="label" translatable="yes">Radio</property>
                  <property name="use_underline">True</property>
                  <property name="active">True</property>
                </object>
                <packing>
                  <property name="expand">False</property>
                  <property name="homogeneous">True</property>
                </packing>
              </child>
              <child>
                <object class="GtkRadioToolButton">
                  <property name="visible">True</property>
```

```
            <property name="can_focus">False</property>
            <property name="label" translatable="yes">Radio</property>
            <property name="use_underline">True</property>
            <property name="active">True</property>
            <property name="group">tb_radio1</property>
          </object>
          <packing>
            <property name="expand">False</property>
            <property name="homogeneous">True</property>
          </packing>
        </child>
        <child>
          <object class="GtkRadioToolButton">
            <property name="visible">True</property>
            <property name="can_focus">False</property>
            <property name="label" translatable="yes">Radio</property>
            <property name="use_underline">True</property>
            <property name="active">True</property>
            <property name="group">tb_radio1</property>
          </object>
          <packing>
            <property name="expand">False</property>
            <property name="homogeneous">True</property>
          </packing>
        </child>
      </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="GtkStatusbar" id="statusbar">
        <property name="visible">True</property>
        <property name="can_focus">False</property>
        <property name="margin_left">10</property>
        <property name="margin_right">10</property>
        <property name="margin_start">10</property>
        <property name="margin_end">10</property>
        <property name="margin_top">6</property>
        <property name="margin_bottom">6</property>
        <property name="orientation">vertical</property>
        <property name="spacing">2</property>
      </object>
      <packing>
        <property name="expand">False</property>
        <property name="fill">True</property>
        <property name="position">4</property>
      </packing>
    </child>
  </object>
```

```
    </child>
    <child>
      <placeholder/>
    </child>
  </object>
  <object class="GtkAboutDialog" id="about_dialog">
    <property name="can_focus">False</property>
    <property name="title" translatable="yes">Info</property>
    <property name="resizable">False</property>
    <property name="modal">True</property>
    <property name="window_position">center-on-parent</property>
    <property name="destroy_with_parent">True</property>
    <property name="type_hint">dialog</property>
    <property name="deletable">False</property>
    <property name="transient_for">window</property>
    <property name="program_name">Info Dialog</property>
    <property name="version">0.1</property>
    <property name="comments" translatable="yes">Platz für mehr Blabla</property>
    <property name="website">www.example.com</property>
    <property name="logo_icon_name">image-missing</property>
    <child internal-child="vbox">
      <object class="GtkBox">
        <property name="can_focus">False</property>
        <property name="orientation">vertical</property>
        <property name="spacing">2</property>
        <child internal-child="action_area">
          <object class="GtkButtonBox">
            <property name="can_focus">False</property>
            <property name="layout_style">end</property>
            <child>
              <object class="GtkButton" id="close_button">
                <property name="label">gtk-close</property>
                <property name="visible">True</property>
                <property name="can_focus">True</property>
                <property name="receives_default">True</property>
                <property name="use_stock">True</property>
                <signal name="clicked" handler="on_close_button_clicked" swapped=
↪"no"/>
              </object>
              <packing>
                <property name="expand">True</property>
                <property name="fill">True</property>
                <property name="position">0</property>
              </packing>
            </child>
          </object>
          <packing>
            <property name="expand">False</property>
            <property name="fill">False</property>
            <property name="position">0</property>
          </packing>
        </child>
      </object>
    </child>
    <child>
      <placeholder/>
    </child>
  </object>
</interface>
```

6.3.2 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_info_button_clicked(self,widget):
        x.sb_message("Öffne Info-Dialog")
        x.builder.get_object("about_dialog").show_all()

    def on_close_button_clicked(self,widget):
        x.sb_message("Schließe Info-Dialog")
        x.builder.get_object("about_dialog").hide_on_delete()

    def on_nothing_here(self,widget):
        x.sb_message("%s: Widget hat keine Funktion." % widget)

class Example:

    def __init__(self):

        self.builder = Gtk.Builder()
        self.builder.add_from_file("05_menutoolbar.glade")
        self.builder.connect_signals(Handler())

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

    def sb_message(self,message):
        self.builder.get_object("statusbar").push(0,message)

    def main(self):
        Gtk.main()

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

KAPITEL 7

Alles so schön bunt hier

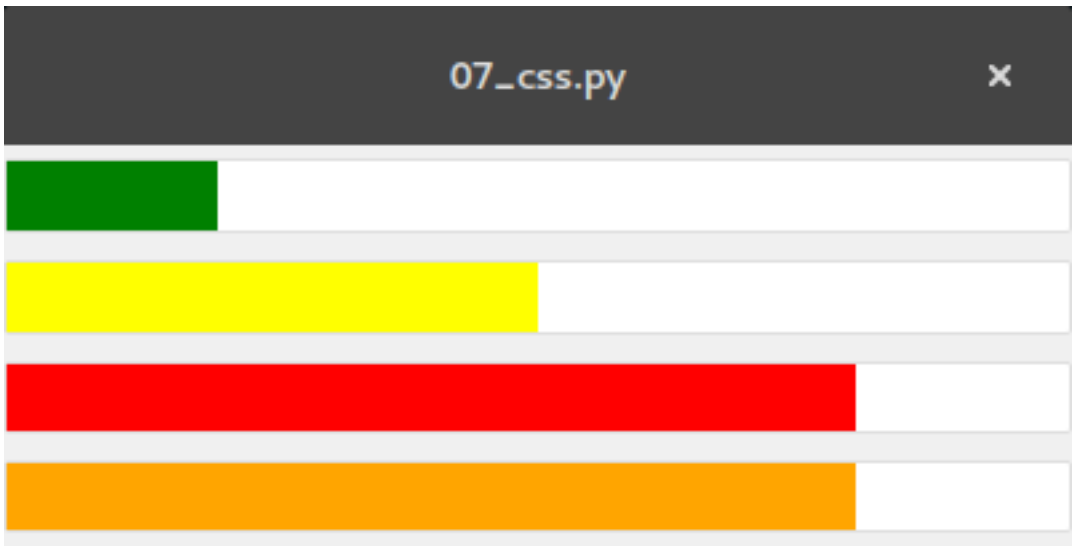
Schöner klicken mit Cascading Style Sheets

7.1 CSS

GTK+-Objekte lassen sich mit Hilfe von CSS im Layout verändern. Meiner unmaßgeblichen Ansicht nach sollte man es damit allerdings nicht übertreiben und das grundlegende Erscheinungsbild dem eingestellten Theme überlassen. Links:

- GTK, Python and CSS are an awesome combo - Grundlagen mit Beispiel
- Overview of CSS in GTK+ - ausführliche Übersicht mit vielen Beispielen
- A GTK+ update - Neuerungen seit GTK+ 3.20

7.2 Glade



Mit Glade werden nur die Fenster/Widgets angelegt, in diesem Beispiel vier Levelbars mit Werten, die CSS-Layout-Anweisungen erfolgen dann im Code.

7.3 Python

7.3.1 CSS

Layout-Anweisungen erfolgen nach dem Muster

```
widget [element] {
  font...
  color...
  background...
  ...
}
```

innerhalb einer String-Variblen, die von der Klasse `Gtk.CssProvider()` geladen werden.

7.3.2 Levelbar

Levelbars können, wie bereits im Artikel „Bars“ (*Bars*) angedeutet, in definierten Wertebereichen unterschiedliche Farben annehmen (um zum Beispiel einen kritischen Ladezustand zu visualisieren). Die vordefinierten Offset-Marker dafür sind:

- `low` (<=.25)
- `high` (<=.75)
- `full` (bis 1)

Die Werte können mit den Funktionen `get_offset_value` abgefragt bzw. mit `add_offset_value` angelegt oder verändert werden.

Im Beispiel wird der unteren Levelbar ein zusätzlicher Offsetmarker zwischen `high` und `full` angelegt, deshalb wird beim Wert von 0.8 dort im Gegensatz zur dritten Levelbar nicht der Marker für `full` ereicht.

```
self.bar.add_offset_value("alert",.9)
```

7.4 Listings

7.4.1 Glade

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- Generated with glade 3.20.0 -->
<interface>
  <requires lib="gtk+" version="3.20"/>
  <object class="GtkWindow" id="window">
    <property name="can_focus">False</property>
    <property name="default_width">400</property>
    <property name="default_height">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="orientation">vertical</property>
        <property name="homogeneous">True</property>
```

```
        <child>
          <object class="GtkLevelBar" id="lev1">
            <property name="visible">True</property>
            <property name="can_focus">False</property>
            <property name="margin_top">5</property>
            <property name="margin_bottom">5</property>
            <property name="value">0.20000000000000001</property>
          </object>
          <packing>
            <property name="expand">False</property>
            <property name="fill">True</property>
            <property name="position">0</property>
          </packing>
        </child>
        <child>
          <object class="GtkLevelBar" id="lev2">
            <property name="visible">True</property>
            <property name="can_focus">False</property>
            <property name="margin_top">5</property>
            <property name="margin_bottom">5</property>
            <property name="value">0.5</property>
          </object>
          <packing>
            <property name="expand">False</property>
            <property name="fill">True</property>
            <property name="position">1</property>
          </packing>
        </child>
        <child>
          <object class="GtkLevelBar" id="lev3">
            <property name="visible">True</property>
            <property name="can_focus">False</property>
            <property name="margin_top">5</property>
            <property name="margin_bottom">5</property>
            <property name="value">0.80000000000000004</property>
          </object>
          <packing>
            <property name="expand">False</property>
            <property name="fill">True</property>
            <property name="position">2</property>
          </packing>
        </child>
        <child>
          <object class="GtkLevelBar" id="lev4">
            <property name="visible">True</property>
            <property name="can_focus">False</property>
            <property name="margin_top">5</property>
            <property name="margin_bottom">5</property>
            <property name="value">0.80000000000000004</property>
          </object>
          <packing>
            <property name="expand">False</property>
            <property name="fill">True</property>
            <property name="position">3</property>
          </packing>
        </child>
      </object>
    </child>
    <child>
      <placeholder/>
    </child>
  </object>
</interface>
```

7.4.2 Python

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

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

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("07_css.glade")
        self.builder.connect_signals(Handler())

        css = b"""

levelbar trough block.filled.low {
    background-color: green;
}

levelbar trough block.filled.high {
    background-color: yellow;
}

levelbar trough block.filled.alert {
    background-color: orange;
}

levelbar trough block.filled.full {
    background-color: red;
}
"""
        #load css stylesheet
        style_provider = Gtk.CssProvider()
        style_provider.load_from_data(css)

        Gtk.StyleContext.add_provider_for_screen(
            Gdk.Screen.get_default(),
            style_provider,
            Gtk.STYLE_PROVIDER_PRIORITY_APPLICATION
        )

        self.bar = self.builder.get_object("lev4")
        self.bar.add_offset_value("alert",.9)

        print("low:  ",self.bar.get_offset_value("low"))
        print("high: ",self.bar.get_offset_value("high"))
        print("alert:",self.bar.get_offset_value("alert"))
        print("full: ",self.bar.get_offset_value("full"))

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

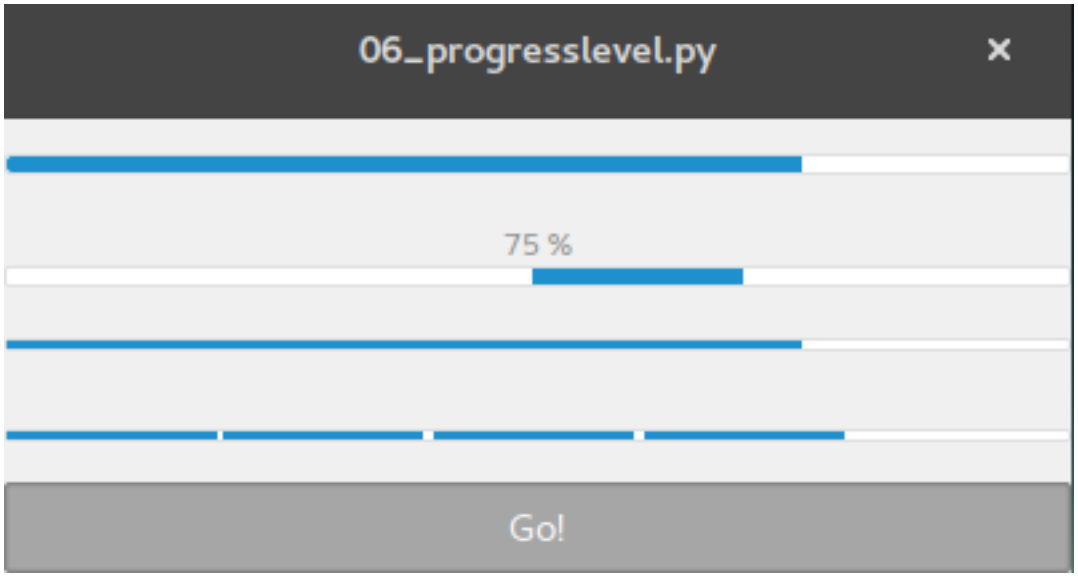
```
    def main(self):
        Gtk.main()

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

KAPITEL 8

Bars

Progressbars und Levelbars

8.1 Glade



8.1.1 Progressbar

Fortschrittsbalken zeigen für gewöhnlich den Status eines länger dauernden Prozesses an. Es gibt dabei zwei Modi:

1. verhältnismäßige Anzeige, der Fortschritt wird mit einem Wert zwischen 0 und 1 ausgedrückt
2. Aktivitätsmodus, ein beweglicher Block läuft nach zugewiesener Schrittweite hin und her

Zusätzlich hat das Widget eine optionale Textanzeige. Wird der Inhalt nicht spezifiziert, wird der Fortschritt in Prozent angezeigt.

8.1.2 Levelbar

Levelbars werden normalerweise als Füllstandsanzeiger genutzt. Der Füllstand wird dabei wie beim Fortschrittsbalken angezeigt, weitere Widget-Eigenschaften sind:

1. Zwei Anzeigenmodi:
   (a) *continuous*: ein einzelner Block repräsentiert den gegebenen Wert
   (b) *discrete*: Levelbar wird in eine festgelegte Anzahl von Blöcken geteilt, ein Block steht für einen Wertebereich
2. Festlegen von Minimal-/Maximalwert möglich, Standardwert ist 0 bzw. 1; beim Anzeigenmodus *discrete* entspricht der Maximalwert der Anzahl der Blöcke
3. Farbliche Änderungen des Balkens bei Überschreiten bestimmter Werte (siehe CSS-Artikel (*Alles so schön bunt hier*))

8.2 Python

8.2.1 Progressbar

Im Beispiel repräsentiert der erste Balken den Wert, der zweite befindet sich im Aktivitätsmodus. Möchte man bei letzterem trotzdem eine Prozentangabe im Textfeld darstellen, muss man diesen manuell einsetzen:

```
widget.pulse()
widget.set_text("%d %%" % perc_value)
```

8.2.2 Levelbar

Was `set_fraction` für Progressbar, ist `set_value` für Levelbar. Im *continuous*-Modus ist dies selbsterklärend, im *discrete*-Modus muss man bedenken, wie viele Blöcke definiert wurden:

```
widget.set_value(fraction*blocks)
```

Auf die Funktion (*Alles so schön bunt hier*) eingegangen.

8.3 Listings

8.3.1 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">400</property>
    <property name="default_height">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="orientation">vertical</property>
        <property name="homogeneous">True</property>
        <child>
```

```
          <object class="GtkProgressBar" id="prog1">
            <property name="visible">True</property>
            <property name="can_focus">False</property>
            <property name="text" translatable="yes">Text hier</property>
          </object>
          <packing>
            <property name="expand">False</property>
            <property name="fill">False</property>
            <property name="position">0</property>
          </packing>
        </child>
        <child>
          <object class="GtkProgressBar" id="prog2">
            <property name="visible">True</property>
            <property name="can_focus">False</property>
            <property name="pulse_step">0.14999999999999999</property>
            <property name="show_text">True</property>
          </object>
          <packing>
            <property name="expand">False</property>
            <property name="fill">False</property>
            <property name="position">1</property>
          </packing>
        </child>
        <child>
          <object class="GtkLevelBar" id="lev1">
            <property name="visible">True</property>
            <property name="can_focus">False</property>
          </object>
          <packing>
            <property name="expand">False</property>
            <property name="fill">False</property>
            <property name="position">2</property>
          </packing>
        </child>
        <child>
          <object class="GtkLevelBar" id="lev2">
            <property name="visible">True</property>
            <property name="can_focus">False</property>
            <property name="max_value">5</property>
            <property name="mode">discrete</property>
          </object>
          <packing>
            <property name="expand">False</property>
            <property name="fill">False</property>
            <property name="position">3</property>
          </packing>
        </child>
        <child>
          <object class="GtkButton" id="go">
            <property name="label" translatable="yes">Go!</property>
            <property name="visible">True</property>
            <property name="can_focus">True</property>
            <property name="receives_default">True</property>
            <property name="always_show_image">True</property>
            <signal name="clicked" handler="on_go_clicked" swapped="no"/>
          </object>
          <packing>
            <property name="expand">False</property>
            <property name="fill">False</property>
            <property name="position">7</property>
          </packing>
        </child>
```

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

8.3.2 Python

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

import time
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_go_clicked(self,widget):
        for i in range(101):
            x.progbar1.set_fraction(i/100)
            x.progbar2.pulse()
            x.progbar2.set_text("%d %%" % i)
            x.levbar1.set_value(i/100)
            x.levbar2.set_value((i/100)*5)
            time.sleep(.05)
            #interrupt main loop to update GUI
            while Gtk.events_pending(): Gtk.main_iteration()
        x.progbar2.set_fraction(1)

class Example:

    def __init__(self):

        self.builder = Gtk.Builder()
        self.builder.add_from_file("06_progresslevel.glade")
        self.builder.connect_signals(Handler())

        self.progbar1 = self.builder.get_object("prog1")
        self.progbar2 = self.builder.get_object("prog2")
        self.levbar1 = self.builder.get_object("lev1")
        self.levbar2 = self.builder.get_object("lev2")

        self.levbar2.add_offset_value("high",4)
        self.levbar2.add_offset_value("full",5)

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

    def main(self):
        Gtk.main()

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

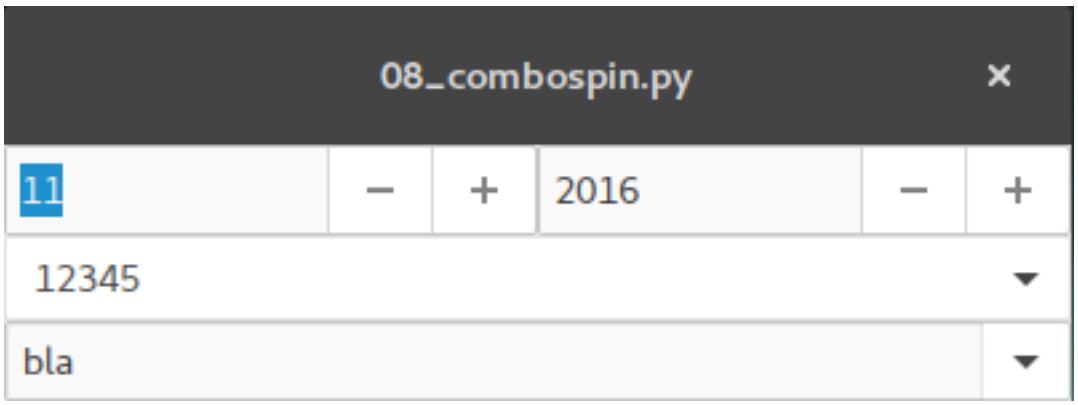
KAPITEL 9

Qual der Wahl

Spinbutton und Combobox

Die Widgets erleichtern die Eingabe bestimmer Werte, indem eine Listenauswahl oder ein Wertebereich und ggf. Standardwerte dazu vorgegeben werden. Die Eingabe ist normalerweise rein intuitiv über Mausklicks machbar, Tastatureingaben sind optional.

9.1 Glade



9.1.1 Spinbutton

Spinbuttons verfügen über zahlreiche, per Glade festlegbare Eigenschaften wie Mindest-/Höchst-/Standardwert. Zum Spinbutton gehört zwingend das Widget *adjustment*, das unter *„Allgemein > Attribute des Einstellknopfs > Stellgröße“* ausgewählt oder angelegt werden kann.

Im Beispiel repräsentieren die beiden Spinbuttons Monat und Jahr, wobei der Spinbutton für den Monat zyklisch angelegt wird, das heißt, nach dem Erreichen des Maximalwertes springt er auf den Mindestwert um. Dieses Verhalten löst das Signal *wrapped* aus und wird angelegt, um die Jahreszahl im zweiten Spinbutton passend umzuschalten.

9.1.2 Combobox

Es gibt in GTK+ zwei verschiedene Combobox-Widgets:

GtkComboboxText Die Auswahlliste des Dropdown-Menüs sind Strings und werden direkt im Widget erstellt.

GtkCombobox Die Daten für die Auswahlliste stammen aus einem Listen- oder Baumspeicher (ListStore oder TreeStore). In diesen können Datensätze mit verschiedenen Informationen gespeichert werden (siehe auch Artikel „Überlistet“ (*Überlistet*)).

Beide Widgets können zusätzlich ein optionales Eingabefeld besitzen. In diesem Fall muss *„Allgemein > Hat Eintrag“* aktiviert sein. Dies legt das interne Widget *GtkEntry* an. Wichtig ist, dass dieses unter „Gemeinsam > Widget Flags“ fokussierbar gemacht wird.

Im Beispiel gibt es zwei ComboboxText-Widgets. Das erste besitzt kein Eingabefeld, es ist also ausschließlich eine Auswahl unter den gegebenen Listenpunkten möglich, die Auswahlliste ist direkt in Glade eingegeben. Die zweite Combobox hat ein Eingabefeld, zu demonstrativen Zwecken werden die Listenpunkte direkt im Programm erstellt. Bei beiden wird das Signal *changed* abgefangen.

9.2 Python

9.2.1 Spinbutton

Der Wert eines Spinbutton lässt sich einfach per `get_value` bzw. `set_value` ermitteln bzw. festlegen. So werden im Beispiel zu Beginn die aktuellen Monats- und Jahreszahlen eingetragen und in der Funktion `on_spin_m_wrapped` beim Umschalten von 12 auf 1 die Jahreszahl um 1 erhöht und umgekehrt.

9.2.2 Combobox

Listeneinträge einer Combobox können einfach mit der Funktion `append` angefügt werden, wie in diesem Beispiel etwa

```
[self.builder.get_object("comboboxtext2").append(None,entry) for entry in ("bla",
↪"blubb","ja","nein")]
```

Der aktuell angewählte Eintrag wird mit der Funktion `widget.set_active_text()` ermittelt, diese gibt auch den Text des optionalen Texteintragfeldes aus.

9.3 Listings

9.3.1 Glade

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- Generated with glade 3.20.0 -->
<interface>
  <requires lib="gtk+" version="3.20"/>
  <object class="GtkAdjustment" id="adj_j">
    <property name="lower">1234</property>
    <property name="upper">3000</property>
    <property name="step_increment">1</property>
    <property name="page_increment">10</property>
  </object>
  <object class="GtkAdjustment" id="adj_m">
    <property name="lower">1</property>
    <property name="upper">12</property>
    <property name="step_increment">1</property>
```

```
    <property name="page_increment">10</property>
  </object>
  <object class="GtkWindow" id="window">
    <property name="can_focus">False</property>
    <property name="default_width">400</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="orientation">vertical</property>
        <child>
          <object class="GtkBox">
            <property name="visible">True</property>
            <property name="can_focus">False</property>
            <property name="homogeneous">True</property>
            <child>
              <object class="GtkSpinButton" id="spin_m">
                <property name="visible">True</property>
                <property name="can_focus">True</property>
                <property name="placeholder_text" translatable="yes">Monat</
↪property>
                <property name="adjustment">adj_m</property>
                <property name="wrap">True</property>
                <signal name="wrapped" handler="on_spin_m_wrapped" swapped="no"/>
              </object>
              <packing>
                <property name="expand">False</property>
                <property name="fill">True</property>
                <property name="position">1</property>
              </packing>
            </child>
            <child>
              <object class="GtkSpinButton" id="spin_y">
                <property name="visible">True</property>
                <property name="can_focus">True</property>
                <property name="adjustment">adj_j</property>
                <property name="climb_rate">1</property>
              </object>
              <packing>
                <property name="expand">False</property>
                <property name="fill">True</property>
                <property name="position">2</property>
              </packing>
            </child>
          </object>
          <packing>
            <property name="expand">False</property>
            <property name="fill">True</property>
            <property name="position">0</property>
          </packing>
        </child>
        <child>
          <object class="GtkComboBoxText" id="comboboxtext1">
            <property name="visible">True</property>
            <property name="can_focus">False</property>
            <items>
              <item id="&lt;Geben Sie die Kennung ein&gt;" translatable="yes">eins
↪</item>
              <item translatable="yes">zwei</item>
              <item translatable="yes">12345</item>
              <item translatable="yes">mehr Listenblabla</item>
            </items>
```

```
            <signal name="changed" handler="on_comboboxtext1_changed" swapped="no"/
↪>
          </object>
          <packing>
            <property name="expand">False</property>
            <property name="fill">True</property>
            <property name="position">2</property>
          </packing>
        </child>
        <child>
          <object class="GtkComboBoxText" id="comboboxtext2">
            <property name="visible">True</property>
            <property name="can_focus">False</property>
            <property name="has_entry">True</property>
            <signal name="changed" handler="on_comboboxtext2_changed" swapped="no"/
↪>
            <child internal-child="entry">
              <object class="GtkEntry" id="entry">
                <property name="can_focus">True</property>
              </object>
            </child>
          </object>
          <packing>
            <property name="expand">False</property>
            <property name="fill">True</property>
            <property name="position">3</property>
          </packing>
        </child>
      </object>
    </child>
    <child type="titlebar">
      <placeholder/>
    </child>
  </object>
</interface>
```

9.3.2 Python

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

import datetime
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_spin_m_wrapped(self,widget):
        if widget.get_value() == 1:
            x.spin_y.set_value(x.spin_y.get_value()+1)
        else:
            x.spin_y.set_value(x.spin_y.get_value()-1)

    def on_comboboxtext1_changed(self,widget):
        print("Auswahl ComboBox 1:",widget.get_active_text())

    def on_comboboxtext2_changed(self,widget):
```

```
        print("Auswahl ComboBox 2:",widget.get_active_text())

class Example:

    def __init__(self):

        self.builder = Gtk.Builder()
        self.builder.add_from_file("08_combospin.glade")
        self.builder.connect_signals(Handler())

        #set current values for month/year
        self.builder.get_object("spin_m").set_value(datetime.datetime.now().month)
        self.spin_y = self.builder.get_object("spin_y")
        self.spin_y.set_value(datetime.datetime.now().year)

        #set combobox list values
        [self.builder.get_object("comboboxtext2").append(None,entry) for entry in (
↪"bla","blubb","ja","nein")]

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

    def main(self):
        Gtk.main()

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

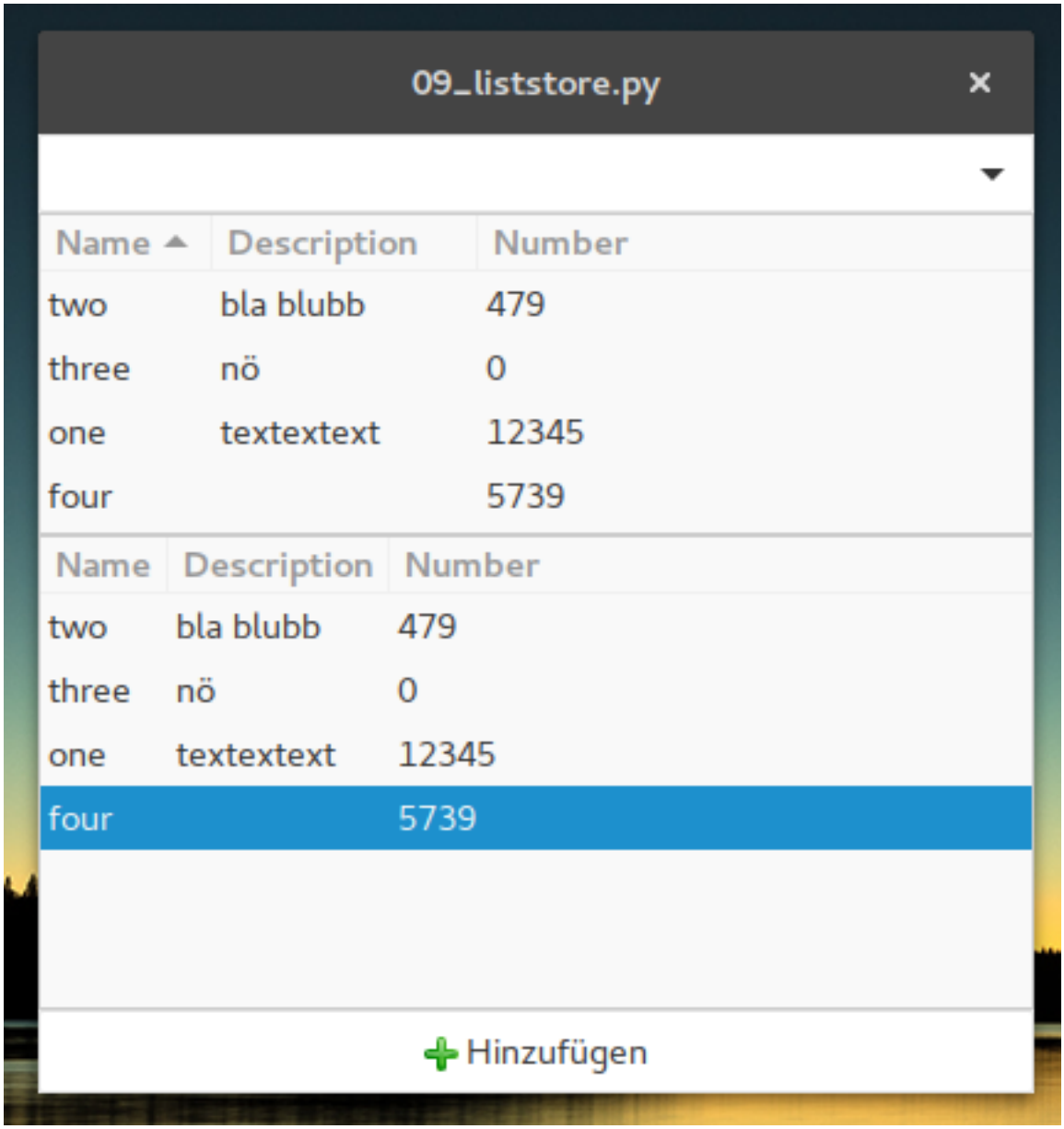
KAPITEL 10

Überlistet

Daten in ListStore speichern und mit ComboBox und TreeView anzeigen

Für die Speicherung und Anzeige von Daten in Listen- oder Tabellenform benötigt man in GTK+-Anwendungen verschiedene Elemente:

1. Im Modell werden die Daten verwaltet, es gibt zwei Typen:
   - *ListStore:* flache Liste, die Spalten können neben Text-, Zahlenwerten auch GTK+-Elemente (z.B. Buttons, Checkboxen) enthalten
   - *TreeStore:* funktioniert prinzipiell wie ListStore, Zeilen können ihrerseits Kind-Einträge besitzen, Daten können im Gegensatz zu ListStore nicht in Glade angegeben werden (TreeStore-Artikel (*Ansichtssache*))
2. Widgets:
   - *TreeView:* dieses Widget eignet sich zum Anzeigen, Sortieren, Bearbeiten von Daten, wird von beiden Modelltypen verwendet; es können parallel mehrere TreeView-Widgets angelegt werden, die auf die selbe Datenbasis (Modell) zurückgreifen, aber zum Beispiel verschiedene Spalten anzeigen
   - *ComboBox:* Comboboxen dienen der Auswahl aus einer gegebenen Liste, deren Datenbasis ein List- oder TreeStore sein kann (siehe Artikel zu Spinbutton und Combobox (*Qual der Wahl*))
   - *CellRenderers:* Unterwidgets, in denen die anzuzeigenden Daten, deren Layout und weitere Optionen wie Bearbeitbarkeit festgelegt werden



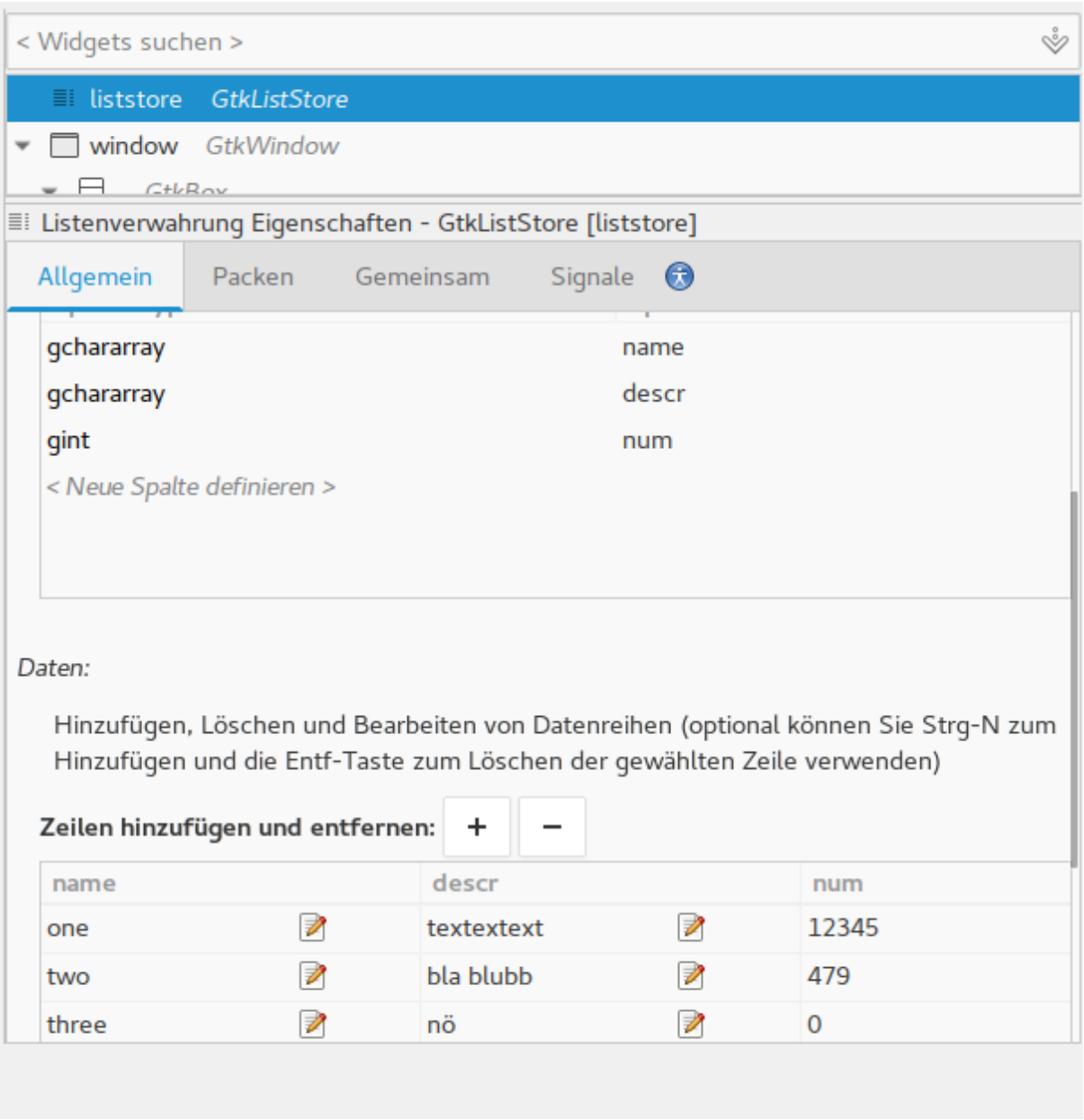
10.1 Glade

10.1.1 ListStore

Um die Vielseitigkeit von ListStore zu skizzieren, wird im Beispiel ein *Gtk.ListStore* (zu finden in der Elementauswahl links unter *„Sonstiges > Listenverwahrung“*) erstellt und von drei Widgets verwendet.

Zunächst werden ein paar Spalten erstellt. ListStore-Daten lassen sich direkt in Glade eingeben. Dies ist allerdings nur für wenige Zeilen und Spalten praktikabel und übersichtlich. Selbst wenige Daten würde ich immer direkt im Python-Code einlesen.

Wie man sieht, werden Änderungen im ListStore (Sortierung, Inhalt) sofort in allen Widgets aktualisiert, die auf dieses Objekt zugreifen. Für verschiedene Sortierungen des selben List-/TreeStores muss man *Gtk.TreeModelSort* anwenden (Beispiel siehe TreeStore-Artikel (*Ansichtssache*)).



10.1.2 Widgets

ComboBox Als „Baumansichtsmodell“ wird wie auch bei den folgenden Widgets der ListStore ausgewählt. Über *„Edit > Hierarchie“* ein CellRendererText hinzugefügt. Im ersten Feld („Text“) stellt man ein, aus welcher Spalte das Dropdown-Menü angezeigt werden soll. Um die Auswahl zu verarbeiten, wird das Signal *changed* belegt.

TreeView #1 Das erste TreeView-Widget wird innerhalb eines *Gtk.ScrolledWindow*-Containers angelegt. Wie bei ComboBox werden nun beliebige CellRenderer angelegt. Wird der Sortierungsanzeiger aktiviert, können die Spalten mit Klick auf den Spaltenkopf sortiert werden. In der Sortierspaltenkennung wird die Spalte angegeben, nach der sortiert werden soll, auf diese Weise kann man eine Spalte auch gemäß einer anderen Spalte sortieren (hier im Beispiel wird die mittlere Spalte nach der letzten sortiert, die Sortierung der beiden hinteren Spalten liefert also das gleiche Ergebnis.

TreeView #2 Das zweite TreeView-Widget wird innerhalb eines Sichtfeldes (*Gtk.Viewport*) erstellt. Dieser Container bietet keine Scrollbalken, das Widget vergrößert automatisch, so dass alle Zeilen sichtbar sind. Bei größeren Tabellen ist ein ScrolledWindow also praktikabler. Es werden die gleichen Daten angezeigt wie zuvor, allerdings ohne Sortierungsanzeiger, dafür wird die mittlere Spalte („Description“) editierbar gemacht

und erhält eine Funktion für das Signal *edited*.

Button Ein Klick auf den Button soll jeweils eine weitere Zeile zum ListStore hinzufügen, es wird also das *clicked*-Signal belegt.

10.2 Python

10.2.1 TreeStore

Die in TreeStore vorhandenen Zeilen lassen sich einfach über `for row in store` abrufen. Neue Zeilen lassen sich mit `append` hinzufügen, andere Optionen wären `insert` oder `remove`, um Zeilen an bestimmten Positionen einzufügen oder zu entfernen.

10.2.2 ComboBox

Normalerweise benötigt man für den Zugang zu einer Datenzeile einen *TreeIter*, das Objekt, das auf den Pfad im Modell zeigt (alternativ kann man diese auch über *TreePath* ansprechen).

```
iter,model = widget.get_active_iter(),widget.get_model()
row = model[iter]
print("Selection:",row[0])
```

10.2.3 Zellen bearbeiten

Das *edited*-Signal übergibt als Parameter die bearbeitete Zeile und den neuen Zelleninhalt. Dieser muss allerdings explizit als neuer Zelleninhalt übergeben werden, sonst zeigt die Zelle nach der Bearbeitung wieder den alten Inhalt an. Dafür kann man einfach die vom Widget übergebene Position (TreePath) statt des TreeIters verwenden.

```
def on_cellrenderer_descr_edited(self,widget,pos,edit):
    x.store[int(pos)][1] = edit
```

10.3 Listings

10.3.1 Glade

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- Generated with glade 3.20.0 -->
<interface>
  <requires lib="gtk+" version="3.20"/>
  <object class="GtkListStore" id="liststore">
    <columns>
      <!-- column-name name -->
      <column type="gchararray"/>
      <!-- column-name descr -->
      <column type="gchararray"/>
      <!-- column-name num -->
      <column type="gint"/>
    </columns>
    <data>
      <row>
        <col id="0" translatable="yes">one</col>
        <col id="1" translatable="yes">textextext</col>
        <col id="2">12345</col>
      </row>
```

```
      <row>
        <col id="0" translatable="yes">two</col>
        <col id="1" translatable="yes">bla blubb</col>
        <col id="2">479</col>
      </row>
      <row>
        <col id="0" translatable="yes">three</col>
        <col id="1" translatable="yes">nö</col>
        <col id="2">0</col>
      </row>
    </data>
  </object>
  <object class="GtkWindow" id="window">
    <property name="width_request">300</property>
    <property name="can_focus">False</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="orientation">vertical</property>
        <child>
          <object class="GtkComboBox" id="cbox">
            <property name="visible">True</property>
            <property name="can_focus">False</property>
            <property name="model">liststore</property>
            <property name="entry_text_column">0</property>
            <signal name="changed" handler="on_cbox_changed" swapped="no"/>
            <child>
              <object class="GtkCellRendererText"/>
              <attributes>
                <attribute name="text">0</attribute>
              </attributes>
            </child>
          </object>
          <packing>
            <property name="expand">False</property>
            <property name="fill">True</property>
            <property name="position">0</property>
          </packing>
        </child>
        <child>
          <object class="GtkBox">
            <property name="width_request">150</property>
            <property name="height_request">250</property>
            <property name="visible">True</property>
            <property name="can_focus">False</property>
            <property name="orientation">vertical</property>
            <child>
              <object class="GtkScrolledWindow">
                <property name="visible">True</property>
                <property name="can_focus">True</property>
                <property name="shadow_type">in</property>
                <child>
                  <object class="GtkTreeView">
                    <property name="visible">True</property>
                    <property name="can_focus">True</property>
                    <property name="model">liststore</property>
                    <property name="headers_clickable">False</property>
                    <child internal-child="selection">
                      <object class="GtkTreeSelection"/>
                    </child>
                    <child>
```

```
                        <object class="GtkTreeViewColumn">
                          <property name="title" translatable="yes">Name</property>
                          <property name="sort_indicator">True</property>
                          <property name="sort_column_id">0</property>
                          <child>
                            <object class="GtkCellRendererText"/>
                            <attributes>
                              <attribute name="text">0</attribute>
                            </attributes>
                          </child>
                        </object>
                      </child>
                      <child>
                        <object class="GtkTreeViewColumn">
                          <property name="title" translatable="yes">Description</
↪property>
                          <property name="sort_indicator">True</property>
                          <property name="sort_column_id">2</property>
                          <child>
                            <object class="GtkCellRendererText"/>
                            <attributes>
                              <attribute name="text">1</attribute>
                            </attributes>
                          </child>
                        </object>
                      </child>
                      <child>
                        <object class="GtkTreeViewColumn">
                          <property name="title" translatable="yes">Number</property>
                          <property name="sort_indicator">True</property>
                          <property name="sort_column_id">2</property>
                          <child>
                            <object class="GtkCellRendererText"/>
                            <attributes>
                              <attribute name="text">2</attribute>
                            </attributes>
                          </child>
                        </object>
                      </child>
                    </object>
                  </child>
                </object>
                <packing>
                  <property name="expand">True</property>
                  <property name="fill">True</property>
                  <property name="position">0</property>
                </packing>
              </child>
              <child>
                <object class="GtkViewport">
                  <property name="visible">True</property>
                  <property name="can_focus">False</property>
                  <child>
                    <object class="GtkTreeView">
                      <property name="visible">True</property>
                      <property name="can_focus">True</property>
                      <property name="model">liststore</property>
                      <child internal-child="selection">
                        <object class="GtkTreeSelection"/>
                      </child>
                      <child>
                        <object class="GtkTreeViewColumn">
                          <property name="title" translatable="yes">Name</property>
```

```
                        <child>
                          <object class="GtkCellRendererText"/>
                          <attributes>
                            <attribute name="text">0</attribute>
                          </attributes>
                        </child>
                      </object>
                    </child>
                    <child>
                      <object class="GtkTreeViewColumn">
                        <property name="title" translatable="yes">Description</
↪property>
                        <child>
                          <object class="GtkCellRendererText" id="cellrenderer_
↪descr">
                            <property name="editable">True</property>
                            <signal name="edited" handler="on_cellrenderer_descr_
↪edited" swapped="no"/>
                          </object>
                          <attributes>
                            <attribute name="text">1</attribute>
                          </attributes>
                        </child>
                      </object>
                    </child>
                    <child>
                      <object class="GtkTreeViewColumn">
                        <property name="title" translatable="yes">Number</property>
                        <child>
                          <object class="GtkCellRendererText"/>
                          <attributes>
                            <attribute name="text">2</attribute>
                          </attributes>
                        </child>
                      </object>
                    </child>
                  </object>
                </child>
              </object>
              <packing>
                <property name="expand">True</property>
                <property name="fill">True</property>
                <property name="position">1</property>
              </packing>
            </child>
          </object>
          <packing>
            <property name="expand">True</property>
            <property name="fill">True</property>
            <property name="position">1</property>
          </packing>
        </child>
        <child>
          <object class="GtkButton" id="add_row_button">
            <property name="label">gtk-add</property>
            <property name="visible">True</property>
            <property name="can_focus">True</property>
            <property name="receives_default">True</property>
            <property name="use_stock">True</property>
            <property name="always_show_image">True</property>
            <signal name="clicked" handler="on_add_row_button_clicked" swapped="no
↪"/>
          </object>
```

```
          <packing>
            <property name="expand">False</property>
            <property name="fill">True</property>
            <property name="position">2</property>
          </packing>
        </child>
      </object>
    </child>
    <child type="titlebar">
      <placeholder/>
    </child>
  </object>
</interface>
```

10.3.2 Python

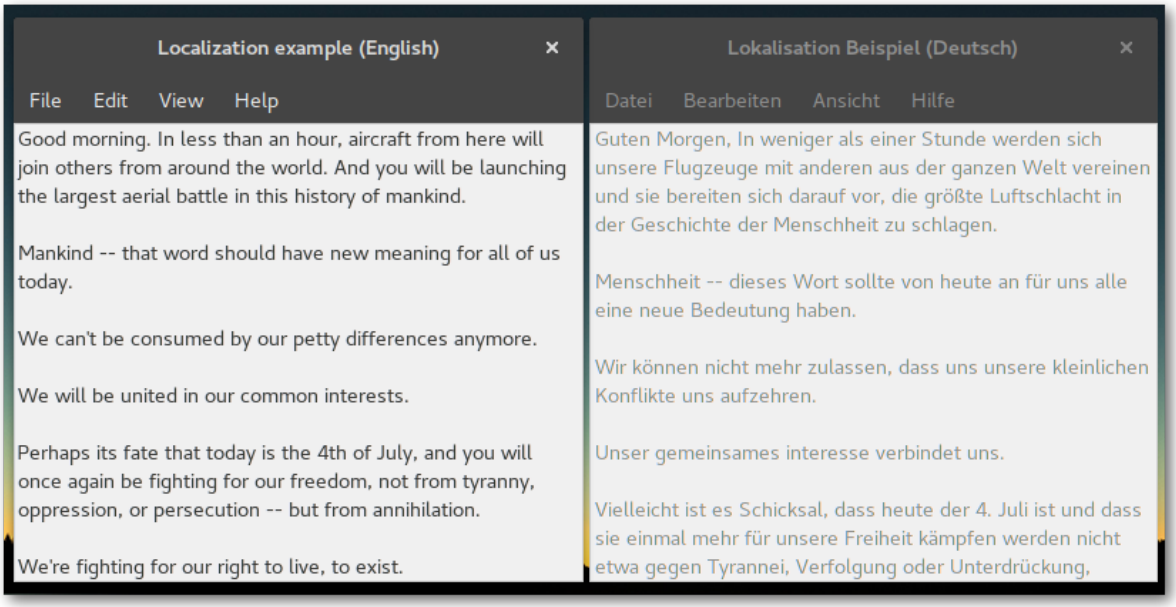
```
#!/usr/bin/python
# -*- coding: utf-8 -*-

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_cbox_changed(self,widget):
        iter,model = widget.get_active_iter(),widget.get_model()
        row = model[iter]
        print("Selection:",row[0])

    def on_cellrenderer_descr_edited(self,widget,pos,edit):
        x.store[int(pos)][1] = edit

    def on_add_row_button_clicked(self,widget):
        x.store.append(list(x.more_rows[len(x.store)-3]))
        #set button inactive when all rows are appended
        if len(x.store) == 7:
            x.button.set_sensitive(False)

class Example:

    def __init__(self):

        self.builder = Gtk.Builder()
        self.builder.add_from_file("09_liststore.glade")
        self.builder.connect_signals(Handler())

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

        self.button = self.builder.get_object("add_row_button")
        self.store = self.builder.get_object("liststore")

        #print all values
        [print(row[:]) for row in self.store]

        self.more_rows = [("four","",5739),
                          ("five","",120),
```

```
                                ("six","",4),
                                ("seven","lucky number",7)]

    def main(self):
        Gtk.main()

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

KAPITEL 11

Romani ite domum

Lokalisation mit gettext und locale



11.1 Glade

Strings von Labels oder Menüs sind standardmäßig als übersetzbar konfiguriert (Checkbox unter „Beschriftung“), insofern muss hier nichts weiter beachtet werden. Glade-Projektdateien werden direkt von GetText verarbeitet.

11.2 Python

11.2.1 Übersetzbare Strings

Zur Übersetzung freigegebene Strings werden durch eine Einklammerung mit vorausgehendem Unterstrich markiert und beim Aufruf von `xgettext` erkannt:

```
_ = gettext.gettext
translatable_string = _("translate me")
```

11.2.2 (bind)textdomain einrichten

Nun muss man Python noch zeigen, unter welchem Namen und Pfad die MO-Dateien (siehe unten) zu finden sind:

```
locale.bindtextdomain(appname,locales_dir)
locale.textdomain(locales_dir)
gettext.bindtextdomain(appname,locales_dir)
gettext.textdomain(appname)
builder.set_translation_domain(appname)
```

`set_translation_domain` muss vor dem Laden der Glade-Datei(en) aufgerufen werden.

11.3 GetText

11.3.1 POT

POT steht für Portable Object Template und ist dem Namen zufolge die Vorlage für Übersetzungen. Diese Datei enthält alle übersetzbaren Strings. Nachdem eine leere POT-Datei erstellt wurde, ruft man nun `xgettext` nach folgendem Muster für alle Quelldateien auf:

```
$ xgettext --options -o output.pot sourcefile.ext
```

Die erkannten Strings werden nach dem Schema

```
#: sourcefile.ext:line number
msgid "translatable string"
msgstr ""
```

der angegebenen POT-Datei hinzugefügt. Die Markierung der Fundstelle(n) des Strings kann mit der Option `--no-location` verhindert werden.

Für das Beispiel wird also je ein Aufruf für die Glade- und Python-Datei benötgt:

```
$ xgettext --sort-output --keyword=translatable --language=Glade -j -o 10_
↪localization/TUT.pot 10_lokalisation.glade
$ xgettext --language=Python -j -o 10_localization/TUT.pot 10_lokalisation.py
```

Mit der Option `-j` (`--join-existing`) wird eine bestehende Datei um zusätzliche Strings ergänzt und funktioniert deshalb sowohl bei der Initiierung (vorher einfach mit `touch template.pot` die leere Datei erstellen) als auch bei erneutem Aufruf zum Aktualisieren neuer Strings.

11.3.2 PO

Die übersetzten Strings werden in jeweils einer PO-Datei gespeichert. Eine neue Übersetzung legt man mit

```
$ msginit --input=source.pot --locale=xx
# xx=language code
```

an, das eine PO-Datei mit dem Namen xx.po (z.B. de.po) anlegt. Diese kann direkt im Texteditor oder mittels Tools wie PoEdit bearbeitet werden. Die deutschsprachige Lokalisation wird also angelegt mit

```
$ msginit --input=TUT.pot --locale=de
```

Wird die POT-Datei verändert, kann man die PO-Dateien mit `msgmerge` abgleichen und anschließend die neuen Strings übesetzen:

```
$ msgmerge lang.po template.pot > new_lang.po
```

11.3.3 MO

MO-Dateien sind auf Maschinenlesbarkeit optimierte PO-Dateien und letztlich die, die vom Programm benutzt werden. Unterhalb der angegebenen *bindtextdomain* liegen die Lokalisationsdateien nach der Verzeichnisstruktur `(path/to/bindtextdomain)/locale/language code/LC_MESSAGES/appname.po`

Im Beispiel wird die bindtextdomain einfach im lokalen Verzeichnis angelegt, die erzeugte *de.po* wird mit `msgfmt` in die MO-Datei überführt:

```
$ msgfmt --output locale/de/LC_MESSAGES/TUT.mo de.po
```

11.4 Listings

11.4.1 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">Localization example (English)</
↪property>
    <property name="default_width">400</property>
    <property name="default_height">400</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="orientation">vertical</property>
        <child>
          <object class="GtkMenuBar">
            <property name="visible">True</property>
            <property name="can_focus">False</property>
            <child>
              <object class="GtkMenuItem">
                <property name="visible">True</property>
                <property name="can_focus">False</property>
                <property name="label" translatable="yes">_File</property>
                <property name="use_underline">True</property>
                <child type="submenu">
                  <object class="GtkMenu">
                    <property name="visible">True</property>
```

```
                    <property name="can_focus">False</property>
                    <child>
                      <object class="GtkImageMenuItem">
                        <property name="label">gtk-new</property>
                        <property name="visible">True</property>
                        <property name="can_focus">False</property>
                        <property name="use_underline">True</property>
                        <property name="use_stock">True</property>
                      </object>
                    </child>
                    <child>
                      <object class="GtkImageMenuItem">
                        <property name="label">gtk-open</property>
                        <property name="visible">True</property>
                        <property name="can_focus">False</property>
                        <property name="use_underline">True</property>
                        <property name="use_stock">True</property>
                      </object>
                    </child>
                    <child>
                      <object class="GtkImageMenuItem">
                        <property name="label">gtk-save</property>
                        <property name="visible">True</property>
                        <property name="can_focus">False</property>
                        <property name="use_underline">True</property>
                        <property name="use_stock">True</property>
                      </object>
                    </child>
                    <child>
                      <object class="GtkImageMenuItem">
                        <property name="label">gtk-save-as</property>
                        <property name="visible">True</property>
                        <property name="can_focus">False</property>
                        <property name="use_underline">True</property>
                        <property name="use_stock">True</property>
                      </object>
                    </child>
                    <child>
                      <object class="GtkSeparatorMenuItem">
                        <property name="visible">True</property>
                        <property name="can_focus">False</property>
                      </object>
                    </child>
                    <child>
                      <object class="GtkImageMenuItem">
                        <property name="label">gtk-quit</property>
                        <property name="visible">True</property>
                        <property name="can_focus">False</property>
                        <property name="use_underline">True</property>
                        <property name="use_stock">True</property>
                      </object>
                    </child>
                  </object>
                </child>
              </object>
            </child>
            <child>
              <object class="GtkMenuItem">
                <property name="visible">True</property>
                <property name="can_focus">False</property>
                <property name="label" translatable="yes">_Edit</property>
                <property name="use_underline">True</property>
                <child type="submenu">
```

```
            <object class="GtkMenu">
              <property name="visible">True</property>
              <property name="can_focus">False</property>
              <child>
                <object class="GtkImageMenuItem">
                  <property name="label">gtk-cut</property>
                  <property name="visible">True</property>
                  <property name="can_focus">False</property>
                  <property name="use_underline">True</property>
                  <property name="use_stock">True</property>
                </object>
              </child>
              <child>
                <object class="GtkImageMenuItem">
                  <property name="label">gtk-copy</property>
                  <property name="visible">True</property>
                  <property name="can_focus">False</property>
                  <property name="use_underline">True</property>
                  <property name="use_stock">True</property>
                </object>
              </child>
              <child>
                <object class="GtkImageMenuItem">
                  <property name="label">gtk-paste</property>
                  <property name="visible">True</property>
                  <property name="can_focus">False</property>
                  <property name="use_underline">True</property>
                  <property name="use_stock">True</property>
                </object>
              </child>
              <child>
                <object class="GtkImageMenuItem">
                  <property name="label">gtk-delete</property>
                  <property name="visible">True</property>
                  <property name="can_focus">False</property>
                  <property name="use_underline">True</property>
                  <property name="use_stock">True</property>
                </object>
              </child>
            </object>
          </child>
        </object>
      </child>
      <child>
        <object class="GtkMenuItem">
          <property name="visible">True</property>
          <property name="can_focus">False</property>
          <property name="label" translatable="yes">_View</property>
          <property name="use_underline">True</property>
        </object>
      </child>
      <child>
        <object class="GtkMenuItem">
          <property name="visible">True</property>
          <property name="can_focus">False</property>
          <property name="label" translatable="yes">_Help</property>
          <property name="use_underline">True</property>
          <child type="submenu">
            <object class="GtkMenu">
              <property name="visible">True</property>
              <property name="can_focus">False</property>
              <child>
                <object class="GtkImageMenuItem">
```

```
                        <property name="label">gtk-about</property>
                        <property name="visible">True</property>
                        <property name="can_focus">False</property>
                        <property name="use_underline">True</property>
                        <property name="use_stock">True</property>
                      </object>
                    </child>
                  </object>
                </child>
              </object>
            </child>
          </object>
          <packing>
            <property name="expand">False</property>
            <property name="fill">True</property>
            <property name="position">0</property>
          </packing>
        </child>
        <child>
          <object class="GtkScrolledWindow">
            <property name="visible">True</property>
            <property name="can_focus">True</property>
            <property name="shadow_type">in</property>
            <child>
              <object class="GtkViewport">
                <property name="visible">True</property>
                <property name="can_focus">False</property>
                <child>
                  <object class="GtkLabel">
                    <property name="visible">True</property>
                    <property name="can_focus">False</property>
                    <property name="label" translatable="yes">Good morning. In␣
↪less than an hour, aircraft from here will join others from around the world.␣
↪And you will be launching the largest aerial battle in this history of mankind.

Mankind -- that word should have new meaning for all of us today.

We can't be consumed by our petty differences anymore.

We will be united in our common interests.

Perhaps its fate that today is the 4th of July, and you will once again be␣
↪fighting for our freedom, not from tyranny, oppression, or persecution -- but␣
↪from annihilation.

We're fighting for our right to live, to exist.

And should we win the day, the 4th of July will no longer be known as an American␣
↪holiday, but as the day when the world declared in one voice:

"We will not go quietly into the night!
We will not vanish without a fight!
We're going to live on!
We're going to survive!"

Today, we celebrate our Independence Day!</property>
                    <property name="wrap">True</property>
                  </object>
                </child>
              </object>
            </child>
          </object>
          <packing>
```

```
            <property name="expand">True</property>
            <property name="fill">True</property>
            <property name="position">1</property>
          </packing>
        </child>
      </object>
    </child>
    <child>
      <placeholder/>
    </child>
  </object>
</interface>
```

11.4.2 Python

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

import os
import locale
import gettext

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

_ = gettext.gettext

class Handler:

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

class Example:

    def __init__(self):

        #setting up localization
        locales_dir = os.path.join(os.getcwd(),'10_localization','locale')
        appname = 'TUT'

        #required for showing Glade file translations
        locale.bindtextdomain(appname,locales_dir)
        locale.textdomain(locales_dir)
        #required for code translations
        gettext.bindtextdomain(appname,locales_dir)
        gettext.textdomain(appname)

        self.builder = Gtk.Builder()
        self.builder.set_translation_domain(appname)

        self.builder.add_from_file("10_lokalisation.glade")
        self.builder.connect_signals(Handler())

        #translatable strings
        print(_("It's a trap!"))
        print(_("""These aren't the droids you're looking for.\n"""))

        #not translatable
        nonono = """\"Jar Jar is the key to all of this.\""""
        george = "...ruined it."
```

```
        print(nonono,george)

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

    def main(self):
        Gtk.main()

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

11.4.3 POT

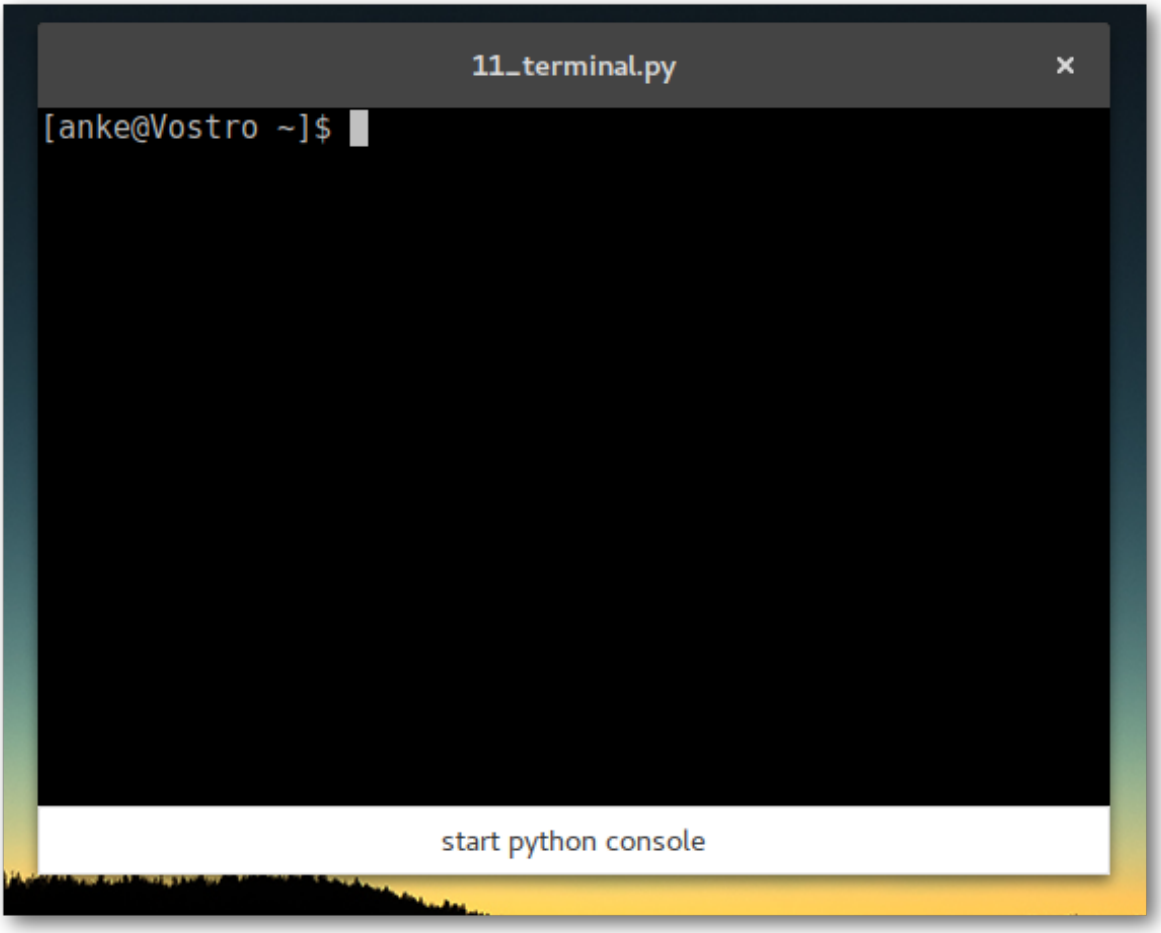
```
# SOME DESCRIPTIVE TITLE.
# Copyright (C) YEAR THE PACKAGE'S COPYRIGHT HOLDER
# This file is distributed under the same license as the PACKAGE package.
# FIRST AUTHOR <EMAIL@ADDRESS>, YEAR.
#
#, fuzzy
msgid ""
msgstr ""
"Project-Id-Version: PACKAGE VERSION\n"
"Report-Msgid-Bugs-To: \n"
"POT-Creation-Date: 2016-11-28 13:06+0100\n"
"PO-Revision-Date: YEAR-MO-DA HO:MI+ZONE\n"
"Last-Translator: FULL NAME <EMAIL@ADDRESS>\n"
"Language-Team: LANGUAGE <LL@li.org>\n"
"Language: \n"
"MIME-Version: 1.0\n"
"Content-Type: text/plain; charset=UTF-8\n"
"Content-Transfer-Encoding: 8bit\n"

msgid ""
"Good morning. In less than an hour, aircraft from here will join others from "
"around the world. And you will be launching the largest aerial battle in "
"this history of mankind.\n"
"\n"
"Mankind -- that word should have new meaning for all of us today.\n"
"\n"
"We can't be consumed by our petty differences anymore.\n"
"\n"
"We will be united in our common interests.\n"
"\n"
"Perhaps its fate that today is the 4th of July, and you will once again be "
"fighting for our freedom, not from tyranny, oppression, or persecution -- "
"but from annihilation.\n"
"\n"
"We're fighting for our right to live, to exist.\n"
"\n"
"And should we win the day, the 4th of July will no longer be known as an "
"American holiday, but as the day when the world declared in one voice:\n"
"\n"
"\"We will not go quietly into the night!\n"
"We will not vanish without a fight!\n"
"We're going to live on!\n"
"We're going to survive!\"\n"
"\n"
"Today, we celebrate our Independence Day!"
msgstr ""

msgid "It's a trap!"
```

```
msgstr ""

msgid "Localization example (English)"
msgstr ""

msgid "These aren't the droids you're looking for.\n"
msgstr ""

msgid "_Edit"
msgstr ""

msgid "_File"
msgstr ""

msgid "_Help"
msgstr ""

msgid "_View"
msgstr ""
```

KAPITEL 12

Exterminate!

Das VTE-Terminal-Widget



12.1 Glade

Das Widget findet man in der Widget-Seitenleiste ganz unten und stellt ein fertiges Terminal bereit. Um das Terminal auf `exit` zu schließen, muss das Signal *child-exited* abgefangen werden.

Ein Klick auf den Button soll innerhalb dieses Terminals eine Python-Konsole starten, hier wird also das *clicked*-Signal belegt.

12.2 Python

Elemente außerhalb des *Gtk*-Moduls, die mit Glade verwendet werden, müssen als *GObject*-Typ registriert werden (dies betrifft beispielsweise auch das *GtkSourceView*-Widget (Modul *GtkSource*):

```
GObject.type_register(Vte.Terminal)
```

Das Terminal wird mit der Funktion `spawn_sync` initiiert, die ganze 7 Parameter erwartet. Die Dokumentation liefert Details, für eine einfache Bash kommt man mit viel Defaults und Nones aus:

```
terminal.spawn_sync(
        Vte.PtyFlags.DEFAULT,
        None,
        ["/bin/bash"],
        None,
        GLib.SpawnFlags.DEFAULT,
        None,
        None,
        )
```

Um eine Eingabe an die Konsole zu schicken, bedarf es der Funktion `feed_child`. Als Parameter müssen übergeben werden zum einen der String (inklusive *newline*, um einen Befehl auszuführen) und die Länge des Strings:

```
command = "python\n"
x.terminal.feed_child(command,len(command))
```

Die Ausgabe ins Terminal kann mit der Funktion `get_text()` abgefangen werden. Die Funktion gibt ein Tupel zurück, dessen erstes Element der Ausgabestring ist. Dieser enthält allerdings den gesamten Terminalinhalt, also auch viele Leerzeilen, die sich mit herkömmlichen String-Operationen beseitigen lassen.

```
widget.get_text()[0].rstrip()
```

12.3 Listings

12.3.1 Glade

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- Generated with glade 3.20.0 -->
<interface>
  <requires lib="gtk+" version="3.20"/>
  <requires lib="vte-2.91" version="0.46"/>
  <object class="GtkApplicationWindow" id="window">
    <property name="can_focus">False</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="orientation">vertical</property>
        <child>
          <object class="VteTerminal" id="term">
            <property name="visible">True</property>
            <property name="can_focus">True</property>
            <property name="has_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_window_destroy" swapped="no"/>
          </object>
          <packing>
            <property name="expand">False</property>
            <property name="fill">True</property>
            <property name="position">0</property>
          </packing>
        </child>
        <child>
          <object class="GtkButton" id="button">
            <property name="label" translatable="yes">start python console</
↪property>
            <property name="visible">True</property>
            <property name="can_focus">True</property>
            <property name="receives_default">True</property>
            <signal name="clicked" handler="on_button_clicked" swapped="no"/>
          </object>
          <packing>
            <property name="expand">False</property>
            <property name="fill">False</property>
            <property name="position">2</property>
          </packing>
        </child>
      </object>
    </child>
    <child type="titlebar">
      <placeholder/>
    </child>
  </object>
</interface>
```

12.3.2 Python

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

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

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

class Handler:

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

    def on_button_clicked(self,widget):
```

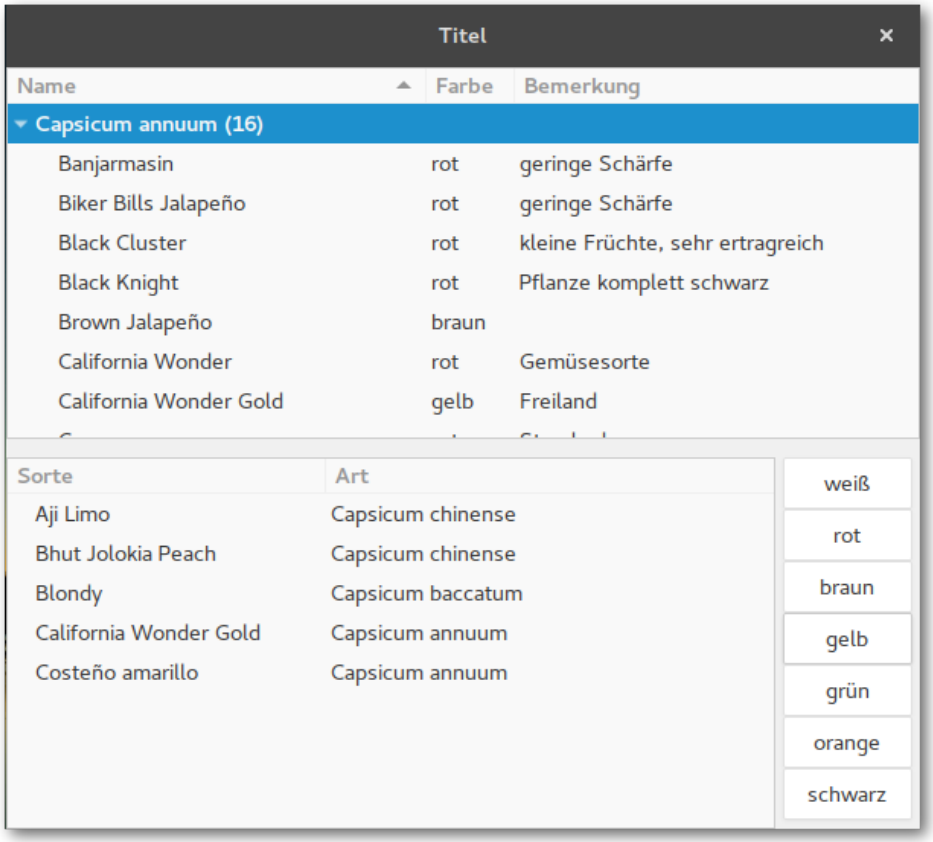
```
        command = "python\n"
        x.terminal.feed_child(command,len(command))

class Example:

    def __init__(self):

        self.builder = Gtk.Builder()
        GObject.type_register(Vte.Terminal)

        self.builder.add_from_file("11_terminal.glade")
        self.builder.connect_signals(Handler())

        self.terminal = self.builder.get_object("term")
        self.terminal.spawn_sync(
            Vte.PtyFlags.DEFAULT,
            None,
            ["/bin/bash"],
            None,
            GLib.SpawnFlags.DEFAULT,
            None,
            None,
            )

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

    def main(self):
        Gtk.main()

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

KAPITEL 13

Ansichtssache

Daten anzeigen mit TreeStore

(Fortsetzung zum ListStore-Artikel (*Überlistet*))



13.1 TreeStore vs. ListStore

Im Gegensatz zum ListStore können Zeilen eines TreeStores ihrerseits Kind-Elemente besitzen, die `append`-Funktion benötigt demzufolge ein weiteren Parameter, der einen Bezug zu einer anderen Datenzeile anzeigt:

```
#append row to liststore
store.append([value1,value2,value3])

#append row to treestore
store.append(parent,[value1,value2,value3])
```

Der Wert der Variable *parent* ist entweder

- **None**, wenn die Zeile keine übergeordnete Zeile besitzt, oder
- **TreeIter**, der zur übergeordneten Zeile zeigt

Der TreeIter wird beim Erstellen einer Zeile erzeugt, untergeordnete Zeilen werden nach folgendem Schema angelegt:

```
row1 = store.append(None,[value1,value2,value3])
row2 = store.append(row1,[value1,value2,value3])
```

Man erhält den *TreeIter*-Wert einer Zeile am einfachsten über die `get_selection`-Funktion des *GtkTreeSelection*-Widgets von TreeView (wird automatisch angelegt).

13.2 Glade

Im Beispiel werden zwei TreeStores und die jeweils enthaltenen Spalten angelegt, dazu die TreeView-Widgets zur Anzeige.

13.2.1 TreeModelSort

Spalten lassen sich mit der Funktion `set_sort_column_id` einfach sortieren. Wendet man diese Funktion direkt auf TreeStore an, werden logischerweise alle TreeView-Widgets, die darauf zurückgreifen, sortiert.

Für diese Fälle muss man *TreeModelSort*-Elemente „zwischenschalten“, d.h. man erstellt aus der Widget-Seitenleiste unter *„Sonstiges > Sortierung für Baumansichtsmodell“* (4. Eintrag) ein Widget und weist ihm den gewünschten TreeStore zu (einzige Option unter „Allgemein“). Anschließend ersetzt man im TreeView das Modell mit dem eben erstellten TreeModelSort.

Die Sortierungsfunktion führt man wie zuvor, nur auf das TreeModelSort-Objekt, aus.

13.2.2 TreeModelFilter

TreeModelFilter ermöglicht die Darstellung bestimmter Zeilen, in Glade wird wie bei TreeModelSort verfahren, zuerst das Element anlegen (3. Eintrag unter „Sonstige“), anschließend erfolgen die Zuweisungen zum Modell und TreeView.

Im gewählten Beispiel sollen Sorten nach der Fruchtfarbe sortiert werden, es wird also noch ein Container für Buttons benötigt, also eine *GtkButtonBox*.

13.2.3 Formatierung aus dem Modell laden

Neben den anzuzeigenden Spalten gibt es im ersten TreeStore eine Spalte „weight“. Der Wert in dieser Spalte wird dazu verwendet, die Zelle in Fettschrift darzustellen. Dazu wird in den Eigenschaften des CellRenderers unter *Schriftgewicht* die entsprechende Spalte angegeben (der Wert für normale Schrift ist 400). Analog dazu können beispielsweise auch Zellen eingefärbt oder weitere Schriftformatierungen vorgenommen werden.

13.3 Python

13.3.1 TreeModelSort

Durch die Positionsabfrage von `GtkTreeSelection.get_selected()` erhält man ein Tupel (model,pos), *pos* von *model* zeigt dabei auf TreeModelSort (bzw. analog auf TreeModelFilter), nicht auf TreeStore und erfordert eine Konvertierung:

```
model,pos = selection.get_selected()
converted_iter = treesort.convert_iter_to_child_iter(pos)
store.set_value(converted_iter,column,value)
```

13.3.2 TreeModelFilter

Zunächst muss eine Filterfunktion erstellt werden, in der die Sichtbarkeit von Zeilen definiert wird, im Beispiel also die Variable *self.color*:

```
def color_filter_func(self,model,iter,data):
    if model[iter][2] == self.color:
        return True
    else:
        return False
```

Die Funktion wird zunächst nach dem Schema

```
treefilter.set_visible_func(filter_func)
```

zugewiesen, jede Filterung wird dann per `refilter()` ausgelöst, also wenn das Button-Signal ausgelöst wird:

```
def on_button_clicked(self,widget):
    x.color = widget.get_label()
    x.obj("treefilter").refilter()
```

13.4 Listings

13.4.1 Glade

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- Generated with glade 3.20.0 -->
<interface>
  <requires lib="gtk+" version="3.20"/>
  <object class="GtkTreeStore" id="filterstore">
    <columns>
      <!-- column-name var -->
      <column type="gchararray"/>
      <!-- column-name spec -->
      <column type="gchararray"/>
      <!-- column-name color -->
      <column type="gchararray"/>
    </columns>
  </object>
  <object class="GtkTreeModelFilter" id="treefilter">
    <property name="child_model">filterstore</property>
  </object>
  <object class="GtkTreeStore" id="store">
    <columns>
```

```
      <!-- column-name name -->
      <column type="gchararray"/>
      <!-- column-name spec -->
      <column type="gchararray"/>
      <!-- column-name color -->
      <column type="gchararray"/>
      <!-- column-name note -->
      <column type="gchararray"/>
      <!-- column-name weight -->
      <column type="gint"/>
      <!-- column-name counter -->
      <column type="gint"/>
    </columns>
  </object>
  <object class="GtkTreeModelSort" id="treesort">
    <property name="model">store</property>
  </object>
  <object class="GtkApplicationWindow" id="window">
    <property name="width_request">600</property>
    <property name="height_request">500</property>
    <property name="can_focus">False</property>
    <property name="title" translatable="yes">Titel</property>
    <signal name="destroy" handler="on_window_destroy" swapped="no"/>
    <child>
      <object class="GtkBox" id="box">
        <property name="visible">True</property>
        <property name="can_focus">False</property>
        <property name="orientation">vertical</property>
        <property name="spacing">12</property>
        <property name="homogeneous">True</property>
        <child>
          <object class="GtkScrolledWindow">
            <property name="visible">True</property>
            <property name="can_focus">True</property>
            <property name="shadow_type">in</property>
            <child>
              <object class="GtkTreeView" id="view">
                <property name="visible">True</property>
                <property name="can_focus">True</property>
                <property name="model">treesort</property>
                <property name="headers_clickable">False</property>
                <property name="rules_hint">True</property>
                <child internal-child="selection">
                  <object class="GtkTreeSelection" id="selection"/>
                </child>
                <child>
                  <object class="GtkTreeViewColumn">
                    <property name="title" translatable="yes">Name</property>
                    <property name="sort_order">descending</property>
                    <property name="sort_column_id">5</property>
                    <child>
                      <object class="GtkCellRendererText"/>
                      <attributes>
                        <attribute name="text">0</attribute>
                        <attribute name="weight">4</attribute>
                      </attributes>
                    </child>
                  </object>
                </child>
                <child>
                  <object class="GtkTreeViewColumn">
                    <property name="title" translatable="yes">Farbe</property>
                    <child>
```

```
                    <object class="GtkCellRendererText"/>
                    <attributes>
                      <attribute name="text">2</attribute>
                    </attributes>
                  </child>
                </object>
              </child>
              <child>
                <object class="GtkTreeViewColumn">
                  <property name="title" translatable="yes">Bemerkung</property>
                  <child>
                    <object class="GtkCellRendererText" id="cellrenderer_note">
                      <property name="editable">True</property>
                      <signal name="edited" handler="on_cellrenderer_note_edited
↪" swapped="no"/>
                    </object>
                    <attributes>
                      <attribute name="text">3</attribute>
                    </attributes>
                  </child>
                </object>
              </child>
            </object>
          </child>
        </object>
        <packing>
          <property name="expand">False</property>
          <property name="fill">True</property>
          <property name="position">0</property>
        </packing>
      </child>
      <child>
        <object class="GtkBox">
          <property name="visible">True</property>
          <property name="can_focus">False</property>
          <child>
            <object class="GtkScrolledWindow">
              <property name="visible">True</property>
              <property name="can_focus">True</property>
              <property name="shadow_type">in</property>
              <property name="propagate_natural_width">True</property>
              <child>
                <object class="GtkTreeView">
                  <property name="visible">True</property>
                  <property name="can_focus">True</property>
                  <property name="model">treefilter</property>
                  <child internal-child="selection">
                    <object class="GtkTreeSelection"/>
                  </child>
                  <child>
                    <object class="GtkTreeViewColumn">
                      <property name="title" translatable="yes">Sorte</property>
                      <child>
                        <object class="GtkCellRendererText"/>
                        <attributes>
                          <attribute name="text">0</attribute>
                        </attributes>
                      </child>
                    </object>
                  </child>
                  <child>
                    <object class="GtkTreeViewColumn">
                      <property name="title" translatable="yes">Art</property>
```

```
                        <child>
                          <object class="GtkCellRendererText"/>
                          <attributes>
                            <attribute name="text">1</attribute>
                          </attributes>
                        </child>
                      </object>
                    </child>
                  </object>
                </child>
              </object>
              <packing>
                <property name="expand">True</property>
                <property name="fill">True</property>
                <property name="position">0</property>
              </packing>
            </child>
            <child>
              <object class="GtkButtonBox" id="buttonbox">
                <property name="visible">True</property>
                <property name="can_focus">False</property>
                <property name="orientation">vertical</property>
                <property name="layout_style">start</property>
                <child>
                  <placeholder/>
                </child>
              </object>
              <packing>
                <property name="expand">False</property>
                <property name="fill">True</property>
                <property name="padding">5</property>
                <property name="pack_type">end</property>
                <property name="position">1</property>
              </packing>
            </child>
          </object>
          <packing>
            <property name="expand">False</property>
            <property name="fill">True</property>
            <property name="position">2</property>
          </packing>
        </child>
      </object>
    </child>
    <child type="titlebar">
      <placeholder/>
    </child>
  </object>
</interface>
```

13.4.2 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()

    def on_button_clicked(self,widget):
        x.color = widget.get_label()
        x.obj("treefilter").refilter()

    def on_cellrenderer_note_edited(self,widget,row,edit):
        model,pos = x.obj("selection").get_selected()
        conv_iter = x.obj("treesort").convert_iter_to_child_iter(pos)
        x.obj("store").set_value(conv_iter,3,edit)

class Example:

    def __init__(self):

        self.builder = Gtk.Builder()
        self.builder.add_from_file("12_treestore.glade")
        self.builder.connect_signals(Handler())
        self.obj = self.builder.get_object

        #read data from text file
        with open("12_capsicum.txt") as f:
            data = [line.strip("\n") for line in f]

        colors = set()
        species = set()
        varieties = []
        for line in data:
            variety = line.split(";")
            colors.add(variety[2])
            species.add(variety[1])
            try:
                variety[3]
            except IndexError:
                variety.append("")
            varieties.append(variety)

        #append lines to 1st treestore
        for s in species:
            counter = 0
            row = self.obj("store").append(None,[None,None,None,None,800,None])
            for v in varieties:
                if v[1] == s:
                    self.obj("store").append(row,[v[0],v[1],v[2],v[3],400,None])
                    counter += 1
            self.obj("store").set_value(row,0,"%s (%d)" % (s,counter))
            self.obj("store").set_value(row,5,counter)

        #append lines to 2nd treestore
        [self.obj("filterstore").append(None,[v[0],v[1],v[2]]) for v in varieties]

        #create buttons in buttonbox
        for c in colors:
            button = Gtk.Button.new_with_label(c)
            button.connect("clicked",Handler().on_button_clicked)
            self.obj("buttonbox").add(button)

        self.obj("view").expand_all()
        self.obj("treesort").set_sort_column_id(5, Gtk.SortType.DESCENDING)
        self.obj("treefilter").set_visible_func(self.color_filter_func)
        self.obj("window").show_all()
```

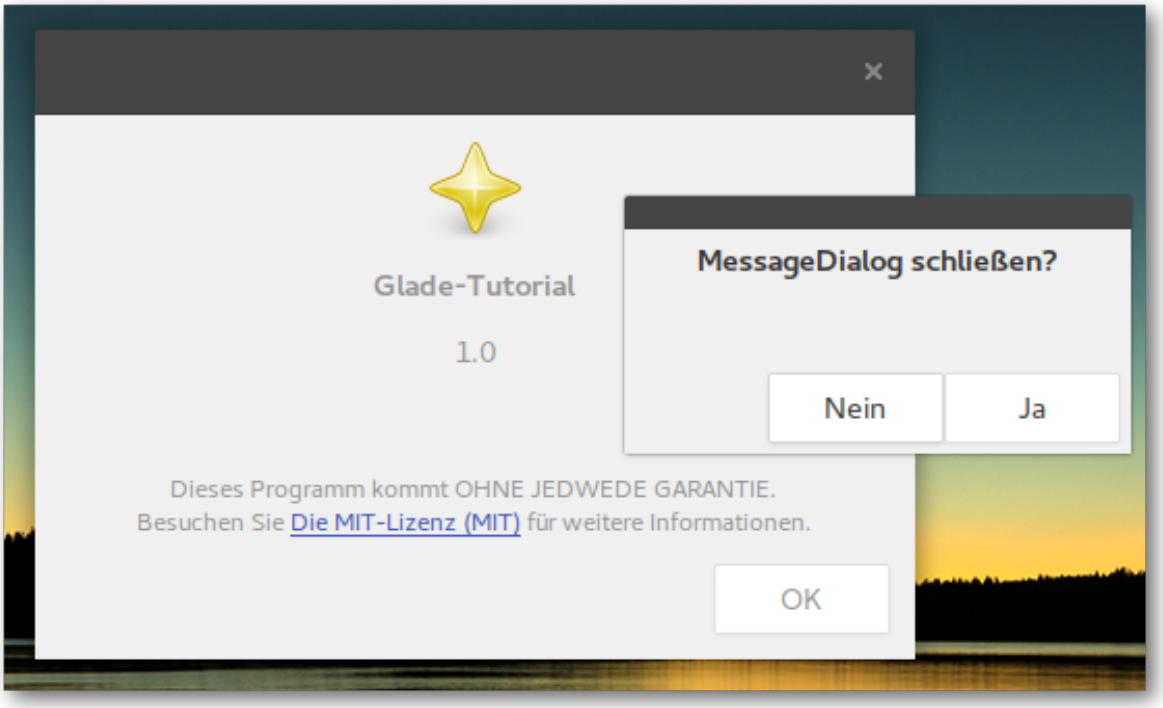
```
    def color_filter_func(self,model,iter,data):
        if model[iter][2] == self.color:
            return True
        else:
            return False

    def main(self):
        Gtk.main()

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

KAPITEL 14

Dialoge

Anzeige von Dialogfenstern

Dialoge sind ergänzende Fenster zur Anwendung und dienen der Interaktion mit dem Benutzer, in denen Informationen angezeigt werden oder Eingaben vom Benutzer abgefragt werden können. Die *GtkDialog*-Klasse bietet einige Unterklassen für gebräuchliche Anzeigen und Abfragen, wie die im Beispiel verwendeten *About*- und *MessageDialog* (Artikel zum FileChooserDialog (*Dateiauswahldialog*)).



14.1 Glade

Dialog-Widgets findet man unter *„Oberste Ebene“* neben den Fenster-Widgets.

Dialoge sind ergänzende Fenster, um den Fokus des Nutzers zu lenken. Sie können direkt an ein übergeordnetes Fenster angeheftet werden, mindestens aber müssen sie unter *„Allgemein > Fensterattribute > Vorübergehend für:“* einem Eltern-Fenster zugeordnet werden. Sie erscheinen dadurch nicht als separates Fenster in der Übersicht und erben ein vorhandenes Icon.

14.1.1 AboutDialog

Das „About“-Dialogfenster bietet in der Regel Informationen zum Projekt, darunter Version, Lizenz, beteiligte Programmierer, Übersetzer etc. Dies alles lässt sich sehr einfach direkt in Glade angeben.

14.1.2 MessageDialog

Der MessageDialog ist ein Standarddialog zum Anzeigen oder Abfragen von Informationen. Er ist so konfiguriert, dass er keine eigene Fensterdekoration besitzt und nicht als Fenster in der Taskbar erscheint. Außerdem bietet er die Möglichkeit, Standardbuttons einzurichten.

14.1.3 Buttons und Responses

Dialoge besitzen bereits intern über eine *GtkButtonBox*, die mit beliebigen Buttons befüllt werden kann. Dieser Bereich ist als „intern action_area“ gekennzeichnet.

Im Gegensatz zu Buttons in normalen Fenstern müssen in Dialogen keine Signale auf *clicked* angelegt werden, sondern man legt in den Button-Eigenschaften unter *„Allgemein“* eine Antwortkennung (Response) fest (int) und belegt das Signal *response* des *GtkDialog*.

Standardbuttons wie im MessageDialog auswählbar besitzen eine vorgegebene Response (siehe Python GI API Reference):

- **Ok** -5
- **Abbrechen** -6
- **Schließen** -7
- **Ja** -8
- **Nein** -9
- **[X]** -4

Der große Vorteil der Responses besteht darin, dass sie sich direkt auf das Dialog-Objekt beziehen; man kann die Responses in einer Funktion verarbeiten und muss dies nicht für jeden einzelnen Button vornehmen.

14.1.4 Wiederherstellbare Dialoge

Das Problem von per *destroy*-Signal geschlossenen Fenstern besteht darin, dass sie sich nicht wieder aufrufen lassen. Deshalb wird stattdessen das Signal *delete-event* belegt.

14.2 Python

14.2.1 Responses

Beim Auslösen des *response*-Signals wird die Antwortkennung als Parameter übergeben, so kann wie bereits erwähnt, jede innerhalb einer einzelnen Funktion verarbeitet werden:

```
def on_dialog_response(self,widget,response):
    if response == 0:
        widget.hide_on_delete()
    elif response == 1:
        do.something()
    elif response == (2 or 3):
        do.something.different()
```

14.2.2 Delete-event

Mit der Funktion `hide_on_delete()` ausgeblendete Dialoge oder reguläre Fenster lassen sich mit `show_all()` wieder anzeigen:

```
def on_dialog_delete_event(self,widget,event):
    widget.hide_on_delete()
    return True
```

14.2.3 Mehrere Glade-Dateien

Wie bereits erwähnt (*Fenster mit Aussicht*), können mehrere Dateien für Fenster und Dialoge innerhalb eines Projektes verwendet werden. Allerdings ist es nicht möglich, diese dateiübergreifend aneinanderzubinden, es wird die `set_transient_for`-Funktion von *GtkWindow* benötigt:

```
dialog.set_transient_for(mainwindow)
```

14.3 Listings

14.3.1 Glade

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- Generated with glade 3.20.0 -->
<interface>
  <requires lib="gtk+" version="3.20"/>
  <object class="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>
      <object class="GtkBox">
        <property name="visible">True</property>
        <property name="can_focus">False</property>
        <property name="orientation">vertical</property>
        <child>
          <object class="GtkButton" id="aboutbutton">
            <property name="label" translatable="yes">GtkAboutDialog</property>
            <property name="visible">True</property>
            <property name="can_focus">True</property>
            <property name="receives_default">True</property>
            <signal name="clicked" handler="on_aboutbutton_clicked" swapped="no"/>
          </object>
          <packing>
            <property name="expand">False</property>
            <property name="fill">True</property>
            <property name="position">0</property>
          </packing>
```

```
        </child>
        <child>
          <object class="GtkButton" id="messagebutton">
            <property name="label" translatable="yes">GtkMessageDialog</property>
            <property name="visible">True</property>
            <property name="can_focus">True</property>
            <property name="receives_default">True</property>
            <signal name="clicked" handler="on_messagebutton_clicked" swapped="no"/
↪>
          </object>
          <packing>
            <property name="expand">False</property>
            <property name="fill">True</property>
            <property name="position">1</property>
          </packing>
        </child>
      </object>
    </child>
    <child type="titlebar">
      <placeholder/>
    </child>
  </object>
  <object class="GtkAboutDialog" id="aboutdialog">
    <property name="can_focus">False</property>
    <property name="type_hint">dialog</property>
    <property name="transient_for">window</property>
    <property name="attached_to">window</property>
    <property name="program_name">Glade-Tutorial</property>
    <property name="version">1.0</property>
    <property name="logo_icon_name">help-about</property>
    <property name="license_type">mit-x11</property>
    <signal name="delete-event" handler="on_dialog_delete_event" swapped="no"/>
    <signal name="response" handler="on_dialog_response" swapped="no"/>
    <child internal-child="vbox">
      <object class="GtkBox">
        <property name="can_focus">False</property>
        <property name="orientation">vertical</property>
        <property name="spacing">2</property>
        <child internal-child="action_area">
          <object class="GtkButtonBox">
            <property name="can_focus">False</property>
            <property name="layout_style">end</property>
            <child>
              <object class="GtkButton" id="button1">
                <property name="label">gtk-ok</property>
                <property name="visible">True</property>
                <property name="can_focus">True</property>
                <property name="receives_default">True</property>
                <property name="use_stock">True</property>
              </object>
              <packing>
                <property name="expand">True</property>
                <property name="fill">True</property>
                <property name="position">1</property>
              </packing>
            </child>
          </object>
          <packing>
            <property name="expand">False</property>
            <property name="fill">False</property>
            <property name="position">0</property>
          </packing>
        </child>
```

```
      </object>
    </child>
    <action-widgets>
      <action-widget response="-8">button1</action-widget>
    </action-widgets>
    <child type="titlebar">
      <placeholder/>
    </child>
  </object>
  <object class="GtkMessageDialog" id="messdialog">
    <property name="can_focus">False</property>
    <property name="type_hint">dialog</property>
    <property name="transient_for">window</property>
    <property name="buttons">yes-no</property>
    <property name="text" translatable="yes">&lt;b&gt;MessageDialog schließen?&lt;/
↪b&gt;</property>
    <property name="use_markup">True</property>
    <signal name="response" handler="on_dialog_response" swapped="no"/>
    <child internal-child="vbox">
      <object class="GtkBox">
        <property name="can_focus">False</property>
        <property name="orientation">vertical</property>
        <property name="spacing">2</property>
        <child internal-child="action_area">
          <object class="GtkButtonBox">
            <property name="can_focus">False</property>
            <property name="homogeneous">True</property>
            <property name="layout_style">end</property>
          </object>
          <packing>
            <property name="expand">False</property>
            <property name="fill">False</property>
            <property name="position">0</property>
          </packing>
        </child>
      </object>
    </child>
    <child type="titlebar">
      <placeholder/>
    </child>
  </object>
</interface>
```

14.3.2 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_dialog_delete_event(self,widget,event):
        widget.hide_on_delete()
        return True
```

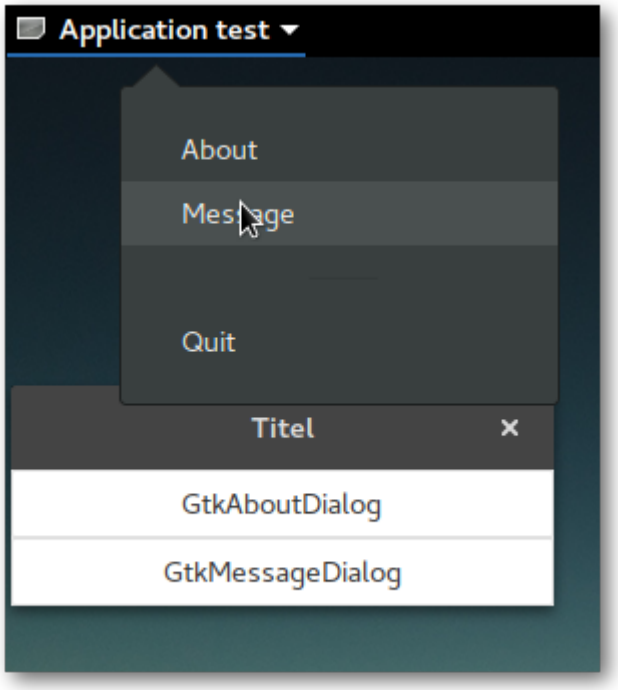
```
    def on_aboutbutton_clicked(self,widget):
        x.obj("aboutdialog").show_all()

    def on_messagebutton_clicked(self,widget):
        x.obj("messdialog").format_secondary_text("")
        x.obj("messdialog").show_all()

    def on_dialog_response(self,widget,response):
        if response == -8:
            widget.hide_on_delete()
        elif response == -9:
            widget.format_secondary_text("Doch!")

class Example:

    def __init__(self):

        builder = Gtk.Builder()
        builder.add_from_file("13_dialoge.glade")
        builder.connect_signals(Handler())

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

    def main(self):
        Gtk.main()

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

KAPITEL 15

Selbständig

Programm als GtkApplication laufen lassen

GtkApplication handhabt verschiedene wichtige Aspekte einer GTK+-Anwendung, wie etwa der GTK+-Initialisierung, dem Sessionmanagement und der Desktopintegration.



15.1 XML-Dateien

15.1.1 Glade

In Glade verändert sich im Prinzip nichts. Als Hauptfenster sollten *GtkApplicationWindows* zum Einsatz kommen. Als Beispiel wird hier das Gladefile aus dem Artikel zu Dialogen (*Dialoge*) wieder verwendet.

15.1.2 GMenu

Die GNOME-Shell unterstützt Appmenüs, erreichbar über das obere Panel. Die XML-Datei muss so formatiert sein, dass sie als *GioMenu* erkannt wird:

```
<?xml version="1.0"?>
<interface>
 <menu id="appmenu">
    <section>
      <item>
        <attribute name="label" translatable="yes">Menu Item</attribute>
        <attribute name="action">app.item</attribute>
      </item>
    </section>
  </menu>
</interface>
```

Von Glade selbst würde diese XML-Datei als veraltetes Format erkannt, aber sie lässt sich trotzdem von *GtkBuilder* laden und anschließend kann man die Identifier nutzen.

15.2 Python

15.2.1 Initialisierung von *GtkApplication*

Bei der Initialisierung wird eine *application_id*- und *flags*-Angabe benötigt. Letztere können in der Regel bei 0 bzw. FLAGS_NONE belassen werden (siehe Gio.ApplicationFlags), die Konventionen für die application_id sind hier dokumentiert.

Die Application kann nun mit verschiedenen Signalen verbunden werden, die zu bestimmten Ereignissen ausgelöst werden, aber es muss mindestens `activate` verbunden werden:

```
def __init__(self):

    self.app = Gtk.Application.new("org.application.test", 0)
    #self.app.connect("startup", self.on_app_startup) #optional
    self.app.connect("activate", self.on_app_activate)
    #self.app.connect("shutdown", self.on_app_shutdown) #optional

def on_app_activate(self, app):

    #setting up GtkBuilder etc.
    ...
    ...
    ...
```

15.2.2 Appmenu

Wie oben bereits erwähnt, lässt sich die GMenu-XML von *GtkBuilder* laden, dann wird das Menü der Application zugewiesen:

```
builder.add_from_file("menu.ui")
app.set_app_menu(builder.get_object("appmenu"))
```

Die zu den Menüeinträgen verknüpften Funktionen müssen nun als Actions, genauer *GioSimpleActions*, erstellt und analog zur herkömmlichen Signalverknüpfung über `connect` verbunden werden.

```
def add_simple_action(self, name, callback):
    action = Gio.SimpleAction.new(name)
```

```
    action.connect("activate", callback)
    self.app.add_action(action)
```

Im Beispiel werden Actions zum Aufrufen der Dialoge erstellt.

15.2.3 Starten und Beenden

GtkApplication übernimmt die Handhabung des GTK+-Mainloops, das heißt, es nicht mehr notwendig GTK+ manuell zu starten oder zu beenden. Stattdessen werden `run()` und `quit()` verwendet:

```
Gtk.main()       ->  app.run(argv)
Gtk.main_quit()  ->  app.quit()
```

Beendet man das Programm über den [X]-Button oder den „Schließen“-Eintrag des Appmenus (immer vorhanden), wird automatisch das „shutdown“-Signal ausgelöst (siehe oben). Das heißt, es müssen keine entsprechenden Signale definiert werden. „Shutdown“ wird auch ausgelöst, wenn bei der Initialisierung nicht mit einer Funktion verbunden wird.

15.3 Links

- GNOME Developer: GtkApplication
- How to use GTK+ 3 in Python to manage your whole application
- Stackoverflow: How to create a complete menu using GIO Actions in PyGI GTK?

15.4 Listings

15.4.1 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>
      <object class="GtkBox">
        <property name="visible">True</property>
        <property name="can_focus">False</property>
        <property name="orientation">vertical</property>
        <child>
          <object class="GtkButton" id="aboutbutton">
            <property name="label" translatable="yes">GtkAboutDialog</property>
            <property name="visible">True</property>
            <property name="can_focus">True</property>
            <property name="receives_default">True</property>
            <signal name="clicked" handler="on_aboutbutton_clicked" swapped="no"/>
          </object>
          <packing>
            <property name="expand">False</property>
            <property name="fill">True</property>
            <property name="position">0</property>
          </packing>
```

```
        </child>
        <child>
          <object class="GtkButton" id="messagebutton">
            <property name="label" translatable="yes">GtkMessageDialog</property>
            <property name="visible">True</property>
            <property name="can_focus">True</property>
            <property name="receives_default">True</property>
            <signal name="clicked" handler="on_messagebutton_clicked" swapped="no"/
↪>
          </object>
          <packing>
            <property name="expand">False</property>
            <property name="fill">True</property>
            <property name="position">1</property>
          </packing>
        </child>
      </object>
    </child>
    <child type="titlebar">
      <placeholder/>
    </child>
  </object>
  <object class="GtkAboutDialog" id="aboutdialog">
    <property name="can_focus">False</property>
    <property name="type_hint">dialog</property>
    <property name="transient_for">window</property>
    <property name="attached_to">window</property>
    <property name="program_name">Glade-Tutorial</property>
    <property name="version">1.0</property>
    <property name="logo_icon_name">help-about</property>
    <property name="license_type">mit-x11</property>
    <signal name="delete-event" handler="on_dialog_delete_event" swapped="no"/>
    <signal name="response" handler="on_dialog_response" swapped="no"/>
    <child internal-child="vbox">
      <object class="GtkBox">
        <property name="can_focus">False</property>
        <property name="orientation">vertical</property>
        <property name="spacing">2</property>
        <child internal-child="action_area">
          <object class="GtkButtonBox">
            <property name="can_focus">False</property>
            <property name="layout_style">end</property>
            <child>
              <object class="GtkButton" id="button1">
                <property name="label">gtk-ok</property>
                <property name="visible">True</property>
                <property name="can_focus">True</property>
                <property name="receives_default">True</property>
                <property name="use_stock">True</property>
              </object>
              <packing>
                <property name="expand">True</property>
                <property name="fill">True</property>
                <property name="position">1</property>
              </packing>
            </child>
          </object>
          <packing>
            <property name="expand">False</property>
            <property name="fill">False</property>
            <property name="position">0</property>
          </packing>
        </child>
```

```
      </object>
    </child>
    <action-widgets>
      <action-widget response="-8">button1</action-widget>
    </action-widgets>
    <child type="titlebar">
      <placeholder/>
    </child>
  </object>
  <object class="GtkMessageDialog" id="messdialog">
    <property name="can_focus">False</property>
    <property name="type_hint">dialog</property>
    <property name="transient_for">window</property>
    <property name="buttons">yes-no</property>
    <property name="text" translatable="yes">&lt;b&gt;MessageDialog schließen?&lt;/
↪b&gt;</property>
    <property name="use_markup">True</property>
    <signal name="response" handler="on_dialog_response" swapped="no"/>
    <child internal-child="vbox">
      <object class="GtkBox">
        <property name="can_focus">False</property>
        <property name="orientation">vertical</property>
        <property name="spacing">2</property>
        <child internal-child="action_area">
          <object class="GtkButtonBox">
            <property name="can_focus">False</property>
            <property name="homogeneous">True</property>
            <property name="layout_style">end</property>
          </object>
          <packing>
            <property name="expand">False</property>
            <property name="fill">False</property>
            <property name="position">0</property>
          </packing>
        </child>
      </object>
    </child>
    <child type="titlebar">
      <placeholder/>
    </child>
  </object>
</interface>
```

15.4.2 GMenu

```
<?xml version="1.0"?>
<interface>
  <menu id="appmenu">
    <section>
      <item>
        <attribute name="label" translatable="yes">About</attribute>
        <attribute name="action">app.about</attribute>
      </item>
      <item>
        <attribute name="label" translatable="yes">Message</attribute>
        <attribute name="action">app.message</attribute>
      </item>
    </section>
    <section>
      <item>
        <attribute name="label" translatable="yes">Quit</attribute>
```

```
        <attribute name="action">app.quit</attribute>
        <attribute name="accel">&lt;Primary&gt;q</attribute>
      </item>
    </section>
  </menu>
</interface>
```

15.4.3 Python

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

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

class Handler:

    def on_window_destroy(self,window):
        window.close()

    def on_dialog_delete_event(self,widget,event):
        widget.hide_on_delete()
        return True

    def on_aboutbutton_clicked(self,widget):
        app.obj("aboutdialog").show_all()

    def on_messagebutton_clicked(self,widget):
        app.obj("messdialog").format_secondary_text("")
        app.obj("messdialog").show_all()

    def on_dialog_response(self,widget,response):
        if response == -8:
            widget.hide_on_delete()
        elif response == -9:
            widget.format_secondary_text("Doch!")

class ExampleApp:

    def __init__(self):

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

    def on_app_startup(self, app):
        print("Gio.Application startup signal emitted")

    def on_app_activate(self, app):
        print("Gio.Application activate signal emitted")
        builder = Gtk.Builder()
        builder.add_from_file("13_dialoge.glade")
        builder.add_from_file("14_giomenu.ui")
        builder.connect_signals(Handler())

        app.set_app_menu(builder.get_object("appmenu"))
        self.obj = builder.get_object
```

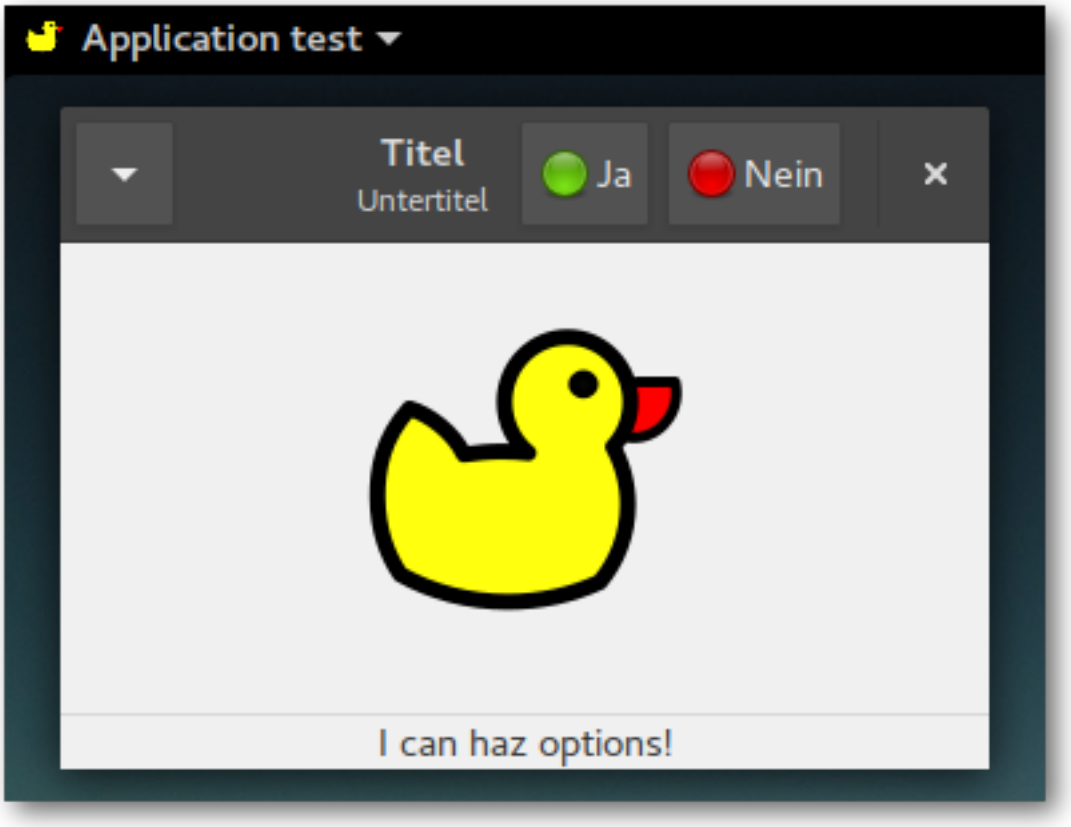
```
        self.obj("window").set_application(app)

        #display application name in upper panel of the GNOME Shell
        self.obj("window").set_wmclass("Application test","Application test")
        self.obj("window").show_all()

        self.add_simple_action("about", self.on_action_about_activated)
        self.add_simple_action("message", self.on_action_message_activated)
        self.add_simple_action("quit", self.on_action_quit_activated)

    def on_app_shutdown(self, app):
        print("Gio.Application shutdown signal emitted")

    def add_simple_action(self, name, callback):
        action = Gio.SimpleAction.new(name)
        action.connect("activate", callback)
        self.app.add_action(action)

    def on_action_about_activated(self, action, user_data):
        self.obj("aboutdialog").show_all()

    def on_action_message_activated(self, action, user_data):
        Handler().on_messagebutton_clicked(self)

    def on_action_quit_activated(self, action, user_data):
        self.app.quit()

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

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

KAPITEL 16

Desktopintegrationsbemühungen

Desktopintegration: Icon, Headerbar, Kommandozeilenoptionen

(Fortsetzung zum Artikel GtkApplication (*Selbständig*))



16.1 Glade

16.1.1 Icon

Einem Fenster lässt sich direkt in Glade unter *„Allgemein > Darstellung > Symboldatei“* ein Icon auswählen. Das Problem dabei ist, dass Glade Bilddateien nur anzeigt, wenn sie sich im selben Verzeichnis wie die Gladedatei selbst befinden, auch wenn man ein anderes Verzeichnis auswählt.

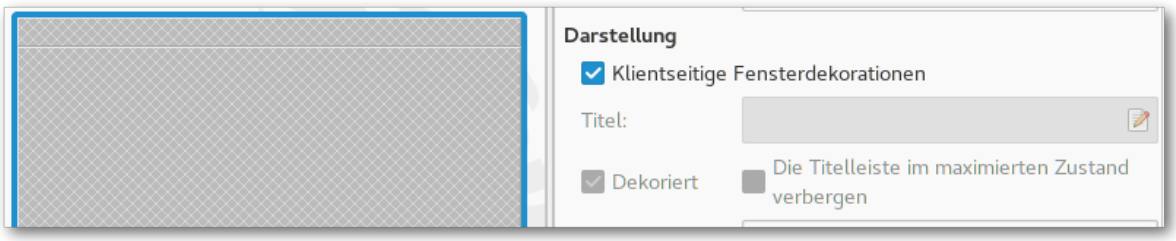
Am einfachsten behebt man dies, indem man die Gladedatei in einem Texteditor bearbeitet und den (relativen) Pfad zum Icon angibt. Diese Einstellung bleibt auch erhalten, wenn die Datei später mit Glade bearbeitet und gespeichert wird:

```
<object class="GtkApplicationWindow" id="window">
  ...
  <!-- <property name="icon">duckyou.svg</property> -->
  <property name="icon">../files/duckyou.svg</property>
  ...
```

16.1.2 Headerbar

Die Headerbar wurde mit GNOME 3.10 eingeführt und vereint Titelleiste und Toolbar in einem Widget, d.h neben Titel und Untertitel können rechts und/oder links verschiedene Widgets (Menüs, Buttons) angelegt sowie clientseitige Fensterkontrollknöpfe angezeigt werden.

Die Headerbar ist optional. Möchte man sie nutzen, muss in den Fenstereinstellungen *„Allgemein > Darstellung > Klienseitige Fensterdekoration“* ausgewählt werden. Daraufhin erscheint im oberen Bereich des Fensters ein reservierter Bereich, in dem die Headerbar platziert wird. Wird die Headerbar außerhalb davon platziert, wird weiterhin zusätzlich die normale Titelleiste angezeigt.



16.2 Kommandozeilenoptionen

GtkApplication stellt die erforderlichen Mittel für anwendungseigene Kommandozeilenoptionen zur Verfügung (Handling command line options in GApplication).

16.2.1 Optionen anlegen

Verfügbare Optionen werden mit der Funktion `add_main_option_entries(entrylist)` hinzugefügt. Diese Einträge haben das Format *GLib.OptionEntry*, welches allerlei Parameter besitzt.

```
def __init__(self):
    self.app = Gtk.Application.new("org.application.test", Gio.ApplicationFlags(0))
    self.app.add_main_option_entries([
        self.create_option_entry("--version", description="Show version numbers␣
↪and exit"),
        self.create_option_entry("--setlabel", description="Set label widget",
↪arg=GLib.OptionArg.STRING,),
        self.create_option_entry("--bollocks", description="Additional test option␣
↪- exit"),
```

```
    ])

def create_option_entry(self,long_name, short_name=None, flags=0, arg=GLib.
↪OptionArg.NONE,arg_data=None, description=None, arg_description=None):
    option = GLib.OptionEntry()
    option.long_name = long_name.lstrip('-')
    option.short_name = 0 if not short_name else short_name.lstrip('-')
    option.flags = flags
    option.arg = arg
    option.arg_data = arg_data
    option.description = description
    option.arg_description = arg_description
    return option
```

16.2.2 Signal verbinden

Der *GtkApplication*-eigene „handle-local-options"-Handler verarbeitet die Optionen. Sobald Optionen angelegt sind, wird dieses Signal noch vor dem „startup"-Signal ausgelöst

```
self.app.connect("handle-local-options", self.on_local_option)
```

16.2.3 Optionen verarbeiten

Die an die Handler-Funktion übergebene `option` ist ein Element der Klasse *GLib.VariantDict*. Mit `contains("option")` lässt sich nach der übergebenen Option suchen.

```
def on_local_option(self, app, option):
    if option.contains("option1"):
        #do something and exit normally
        return 0
    elif option.contains("option2"):
        #do something different and exit
        return 0
    elif option.contains("option3"):
        #do more and continue
    return -1
```

Ein übergebener String kann extrahiert werden, indem *GLib.VariantDict* mit `end()` in *GLib.Variant* konvertiert wird, das sich wiederum mit `keys()` auslesen lässt:

```
var = GLib.VariantDict.end(option)
option_string = var[var.keys()[0]]
```

Ein Return-Wert ist zwingend erforderlich, er entspricht dabei dem Exit-Status:

- **-1**: Anwendung wird weiter ausgeführt
- **0**: erfolgreiche Ausführung, Anwendung wird beendet, „startup/activate" werden nicht ausgeführt
- **1** bzw. positiver Wert: nicht erfolgreiche Ausführung, Anwendung wird beendet

16.2.4 Optionen übergeben

Die Option, die immer verfügbar ist, ist `--help`. Hier werden unter „Anwendungsoptionen" die angelegten Optionen samt Beschreibung aufgeführt. Die Optionen können wie definiert angegeben werden:

```
$ python script.py --version
Python: 3.6.0
GTK+:   3.22.6
```

```
```

oder mit `--setlabel` einen String an *Gtk.Label* übergeben:

```
$ python script.py --setlabel "I can haz options!"
```

16.3 Listings

16.3.1 Glade

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- Generated with glade 3.20.0 -->
<interface>
  <requires lib="gtk+" version="3.20"/>
  <object class="GtkMenu" id="menu">
    <property name="visible">True</property>
    <property name="can_focus">False</property>
    <child>
      <object class="GtkMenuItem">
        <property name="visible">True</property>
        <property name="can_focus">False</property>
        <property name="label" translatable="yes">MenuItem 1</property>
        <property name="use_underline">True</property>
      </object>
    </child>
    <child>
      <object class="GtkMenuItem">
        <property name="visible">True</property>
        <property name="can_focus">False</property>
        <property name="label" translatable="yes">MenuItem 2</property>
        <property name="use_underline">True</property>
      </object>
    </child>
    <child>
      <object class="GtkMenuItem">
        <property name="visible">True</property>
        <property name="can_focus">False</property>
        <property name="label" translatable="yes">MenuItem 3</property>
        <property name="use_underline">True</property>
      </object>
    </child>
  </object>
  <object class="GtkApplicationWindow" id="window">
    <property name="width_request">400</property>
    <property name="height_request">300</property>
    <property name="can_focus">False</property>
    <property name="title" translatable="yes">Titel</property>
    <property name="icon">../files/duckyou.svg</property>
    <property name="show_menubar">False</property>
    <child>
      <object class="GtkBox">
        <property name="visible">True</property>
        <property name="can_focus">False</property>
        <property name="orientation">vertical</property>
        <child>
          <object class="GtkImage">
            <property name="visible">True</property>
            <property name="can_focus">False</property>
            <property name="pixbuf">../files/duckyou.svg</property>
          </object>
```

```
          <packing>
            <property name="expand">True</property>
            <property name="fill">True</property>
            <property name="position">0</property>
          </packing>
        </child>
        <child>
          <object class="GtkLabel" id="label">
            <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="pack_type">end</property>
            <property name="position">1</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">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="title">Titel</property>
        <property name="subtitle">Untertitel</property>
        <property name="show_close_button">True</property>
        <child>
          <object class="GtkMenuButton">
            <property name="visible">True</property>
            <property name="can_focus">True</property>
            <property name="receives_default">True</property>
            <property name="popup">menu</property>
            <child>
              <placeholder/>
            </child>
          </object>
        </child>
        <child>
          <object class="GtkButton">
            <property name="label">gtk-no</property>
            <property name="visible">True</property>
            <property name="can_focus">True</property>
            <property name="receives_default">True</property>
            <property name="use_stock">True</property>
            <property name="always_show_image">True</property>
          </object>
          <packing>
            <property name="pack_type">end</property>
            <property name="position">2</property>
          </packing>
        </child>
```

```
        <child>
          <object class="GtkButton">
            <property name="label">gtk-yes</property>
            <property name="visible">True</property>
            <property name="can_focus">True</property>
            <property name="receives_default">True</property>
            <property name="use_stock">True</property>
            <property name="always_show_image">True</property>
          </object>
          <packing>
            <property name="pack_type">end</property>
            <property name="position">2</property>
          </packing>
        </child>
      </object>
    </child>
  </object>
</interface>
```

16.3.2 Python

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

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

class ExampleApp:

    def __init__(self):

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

        self.app.add_main_option_entries([
            self.create_option_entry("--version", description="Show version␣
↪numbers and exit"),
            self.create_option_entry("--setlabel", description="Set label widget",
↪arg=GLib.OptionArg.STRING),
            self.create_option_entry("--bollocks", description="Additional test␣
↪option - exit"),
        ])

        self.app.connect("handle-local-options", self.on_local_option)
        self.app.connect("activate", self.on_app_activate)

    def on_local_option(self, app, option):
        self.option_string = ""
        if option.contains("version"):
            var = GLib.VariantDict.end(option)
            print("Python: {}".format(sys.version[:5]))
            print("GTK+:   {}.{}.{}".format(Gtk.MAJOR_VERSION, Gtk.MINOR_VERSION,␣
↪Gtk.MICRO_VERSION))
            return 0
        elif option.contains("bollocks"):
            return 1
        elif option.contains("setlabel"):
            var = GLib.VariantDict.end(option)
            self.option_string = var[var.keys()[0]]
```

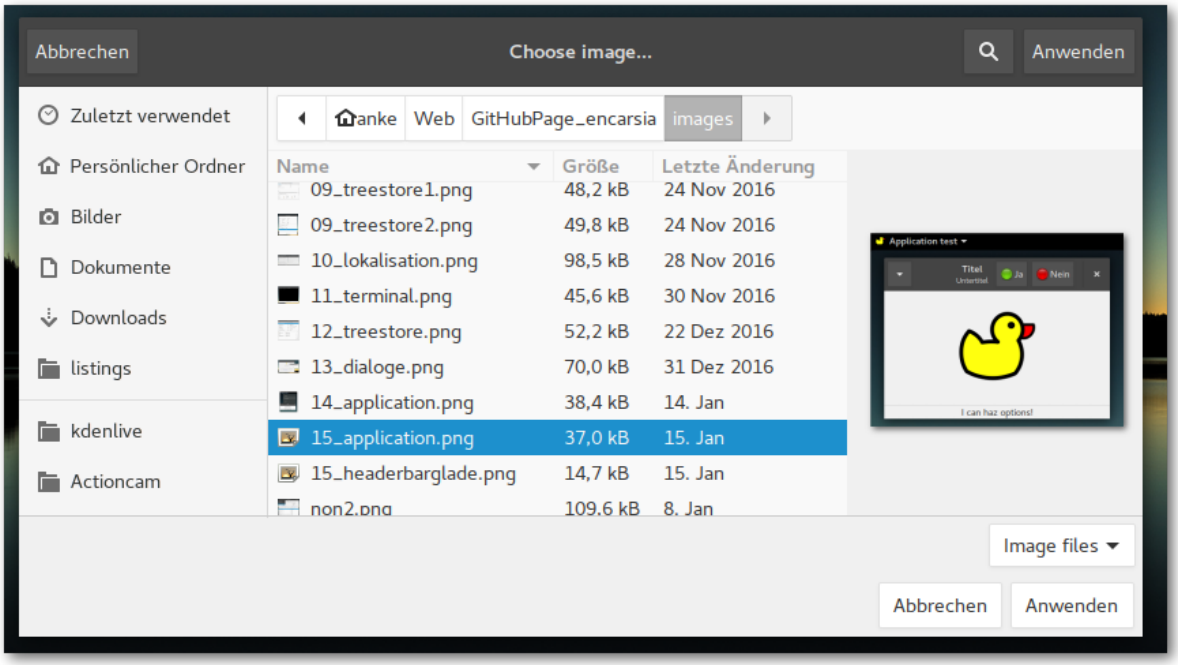
```
        return -1

    def create_option_entry(self,long_name, short_name=None, flags=0, arg=GLib.
↪OptionArg.NONE,arg_data=None, description=None, arg_description=None):
        option = GLib.OptionEntry()
        option.long_name = long_name.lstrip('-')
        option.short_name = 0 if not short_name else short_name.lstrip('-')
        option.flags = flags
        option.arg = arg
        option.arg_data = arg_data
        option.description = description
        option.arg_description = arg_description
        return option

    def on_app_activate(self, app):
        builder = Gtk.Builder()
        builder.add_from_file("15_application.glade")

        self.obj = builder.get_object
        self.obj("window").set_application(app)
        self.obj("label").set_text(self.option_string)

        #display application name in upper panel of the GNOME Shell
        self.obj("window").set_wmclass("Application test","Application test")
        self.obj("window").show_all()

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

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

KAPITEL 17

Dateiauswahldialog

FileChooserDialog

Der *GtkFileChooserDialog* ist eine Subclass von *GtkDialog* (siehe Artikel zu Dialogen (*Dialoge*)) und ermöglicht das Auswählen und Speichern von Dateien oder Ordnern.



17.1 Glade

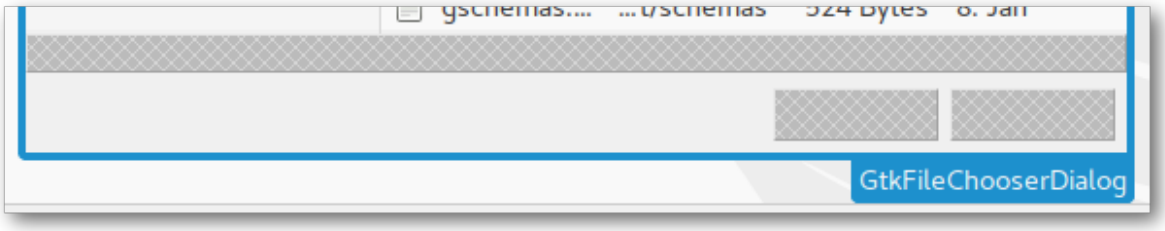
Den Dialog findet man in der Widget-Seitenleiste oben unter *„Oberste Ebene“*. Neben dem Dateibrowser besitzt er eine erweiterbare interne *GtkBox* für weitere Widgets sowie eine *GtkButtonBox* als interne „action area“ für Buttons.

Es ist erforderlich anzugeben, für welche Aktion der Dialog gedacht ist, was *Gtk.FileChooserAction* entspricht (siehe Python GI API Reference): Datei öffnen oder speichern, Ordner auswählen oder anlegen.

17.1.1 Action area und Responses

Responses sind Antwortkennungen, die beim Auslösen des Signals *response* übergeben werden. Buttons in der „action area“ werden jeweils Response-Werte zugewiesen anstatt das *clicked*-Signal der Buttons zu nutzen.

Standardmäßig wird die „action area“ unter dem Dateibrowserbereich angelegt.



Verwendet man den *FileChooserDialog* ohne Glade (siehe unten), werden die Buttons in der Headerbar angezeigt. Letzteres sollte aber vermutlich der Standard sein, da es eine Warnung ausgegeben wird, die die Funktionalität des Dialogs allerdings nicht beeinträchtigt:

```
Gtk-WARNING **: Content added to the action area of a dialog using header bars
```

Diese Meldung wird nicht angezeigt, wenn man darauf verzichtet, in Glade Buttons zur intern action area hinzuzufügen. Dies betrifft auch andere Dialogarten.

Legt man nun in Glade eine Headerbar mit Buttons an, ist es standardmäßig nicht möglich, diesen Buttons Response-Werte zuzuweisen.

Dafür gibt es (mindestens) zwei Lösungsmöglichkeiten:

XML-Datei

Man legt die Headerbar mit Button(s) an, anschließend öffnet man die Glade-Datei in einem Texteditor und fügt dem Element `<action-widgets>` die entsprechenden Zeilen hinzu:

```
<object class="GtkFileChooserDialog" id="filechooser_dialog">
  <property abc ></property>
  <property xyz ></property>
  <!-- usw. -->
  <action-widgets>
    <!-- Buttons innerhalb der action area -->
    <action-widget response="0">button1</action-widget>
    <action-widget response="1">button2</action-widget>
    <!-- Button in Headerbar -->
    <action-widget response="-1">hb_button</action-widget>
  </action-widgets>
  <!-- usw. -->
</object>
```

Dies funktioniert zwar, ist aber ganz sicher nicht so gedacht, weil diese Änderung beim erneuten Bearbeiten der Glade-Datei wieder rückgängig gemacht wird.

add_action_widget-Funktion

Mit der Funktion `add_action_widget` können aktivierbare Widgets zur action area hinzugefügt und damit ebenfalls per *response*-Signal verarbeitet werden. Dies sind Widgets der *Gtk.Activatable*-Klasse und beinhaltet die Widgets *Buttons*, *MenuItem*, *RecentChooserMenu*, *Switch* und *ToolItem*.

Ein Button wird nach dem Schema

```
widget.add_action_widget(button,response)
```

hinzugefügt. Wichtig ist es, beim Button die Widget-Eigenschaft „can-default“ zu aktivieren:

```
button.set_property("can-default",True)
```

Im Beispiel erhält der Dialog die beiden Standardbuttons „Anwenden“/“Abbrechen“:

```
button = Gtk.Button.new_from_stock(Gtk.STOCK_CANCEL)
button.set_property("can-default",True)
self.obj("filechooser_dialog").add_action_widget(button, Gtk.ResponseType.CANCEL)
button = Gtk.Button.new_from_stock(Gtk.STOCK_APPLY)
button.set_property("can-default",True)
self.obj("filechooser_dialog").add_action_widget(button, Gtk.ResponseType.OK)
```

Um die Dateiauswahl auch auf Doppelklick zu ermöglichen, wird neben des *response*-Signals noch das Signal *file-activated* benötigt.

17.1.2 Vorschau-Widget

Der Dialog besitzt die Option ein Vorschau-Widget einzubinden. Dafür aktiviert man in den Dialog-Eigenschaften *„Vorschau-Widget aktiv“* und wählt unter *„Vorschau-Widget“* ein freies Widget (z.B. ein *GtkImage*). Möglicherweise muss man dieses Widget zunächst in ein leeres Container-Widget erstellen und dort in einen freien Bereich ziehen.

Wenn eine Aktualisierung der Vorschau angefordert wird, wird das Signal *update-preview* ausgelöst.

17.1.3 FileFilter

FileFilter dienen dazu, Dateien bestimmten Musters anzuzeigen. Pro Filter können mehrere (shell style glob) Patterns oder MIME-Types angegeben werden.

Den Filter findet man in Glade unter *„Sonstiges“*. Im Dialog kann man in den allgemeinen Widget-Einstellungen den gewünschten Filter auswählen. Dies entspricht der `set_filter`-Funktion.

17.2 Python

17.2.1 Dialog ohne Glade

Der *FileChooserDialog* lässt sich auch ziemlich einfach ohne Glade realisieren, zudem lassen sich die oben genannten Probleme mit Buttons in der Headerbar vermeiden. Der Dialog wird nach folgendem Schema erstellt:

```
dialog = Gtk.FileChooserDialog("window title",
                               parent_window,
                               file_chooser_action,
                               (button1,response1,
                               button2,response2))
```

Der Dialog wird dann direkt aufgerufen und verarbeitet:

```
response = dialog.run()
if response == response1:
    ...
elif response == response2:
    ...
dialog.destroy()
```

17.2.2 FileFilter

Es gibt zwei Möglichkeiten, einen *Filefilter* anzuwenden:

1. Ohne Wahl. Der anzuwendende Filter ist voreingestellt:

```
dialog.set_filter(filter)
```

2. Wahl per Dropdown-Menü: Der Nutzer kann zwischen mehreren vorgegebenen Filtern wählen:

```
dialog.add_filter(filter1)
dialog.add_filter(filter2)
...
```

17.3 Listings

17.3.1 Glade

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- Generated with glade 3.20.0 -->
<interface>
  <requires lib="gtk+" version="3.20"/>
  <object class="GtkFileFilter" id="filefilter">
    <mime-types>
      <mime-type>image/*</mime-type>
    </mime-types>
  </object>
  <object class="GtkFileFilter" id="jpg_filter">
    <mime-types>
      <mime-type>image/jpeg</mime-type>
    </mime-types>
  </object>
  <object class="GtkFileFilter" id="png_filter">
    <mime-types>
      <mime-type>image/png</mime-type>
    </mime-types>
  </object>
  <object class="GtkImage" id="preview">
    <property name="width_request">200</property>
    <property name="visible">True</property>
    <property name="can_focus">False</property>
    <property name="margin_right">5</property>
  </object>
  <object class="GtkApplicationWindow" id="window">
    <property name="width_request">300</property>
    <property name="height_request">200</property>
    <property name="can_focus">False</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="orientation">vertical</property>
        <property name="homogeneous">True</property>
        <child>
          <object class="GtkButton" id="file_button">
            <property name="label" translatable="yes">Choose an image file...</
↪property>
            <property name="visible">True</property>
            <property name="can_focus">True</property>
```

```
            <property name="receives_default">True</property>
            <signal name="clicked" handler="on_file_button_clicked" swapped="no"/>
          </object>
          <packing>
            <property name="expand">False</property>
            <property name="fill">True</property>
            <property name="position">0</property>
          </packing>
        </child>
        <child>
          <object class="GtkButton" id="dir_button">
            <property name="label" translatable="yes">Choose folder...</property>
            <property name="visible">True</property>
            <property name="can_focus">True</property>
            <property name="receives_default">True</property>
            <signal name="clicked" handler="on_dir_button_clicked" swapped="no"/>
          </object>
          <packing>
            <property name="expand">False</property>
            <property name="fill">True</property>
            <property name="position">1</property>
          </packing>
        </child>
      </object>
    </child>
    <child type="titlebar">
      <placeholder/>
    </child>
  </object>
  <object class="GtkFileChooserDialog" id="filechooser_dialog">
    <property name="width_request">800</property>
    <property name="height_request">500</property>
    <property name="can_focus">False</property>
    <property name="type_hint">dialog</property>
    <property name="transient_for">window</property>
    <property name="attached_to">window</property>
    <property name="preview_widget">preview</property>
    <property name="use_preview_label">False</property>
    <signal name="delete-event" handler="on_dialog_close" swapped="no"/>
    <signal name="file-activated" handler="on_filechooser_dialog_file_activated"␣
↪swapped="no"/>
    <signal name="response" handler="on_filechooser_dialog_response" swapped="no"/>
    <signal name="update-preview" handler="on_filechooser_dialog_update_preview"␣
↪swapped="no"/>
    <child internal-child="vbox">
      <object class="GtkBox" id="fcbox">
        <property name="can_focus">False</property>
        <property name="orientation">vertical</property>
        <child internal-child="action_area">
          <object class="GtkButtonBox">
            <property name="can_focus">False</property>
            <child>
              <object class="GtkButton" id="button2">
                <property name="label">gtk-cancel</property>
                <property name="visible">True</property>
                <property name="can_focus">True</property>
                <property name="receives_default">True</property>
                <property name="use_stock">True</property>
              </object>
              <packing>
                <property name="expand">True</property>
                <property name="fill">True</property>
                <property name="position">2</property>
```

```
            </packing>
          </child>
          <child>
            <object class="GtkButton" id="button1">
              <property name="label">gtk-apply</property>
              <property name="visible">True</property>
              <property name="can_focus">True</property>
              <property name="receives_default">True</property>
              <property name="use_stock">True</property>
            </object>
            <packing>
              <property name="expand">True</property>
              <property name="fill">True</property>
              <property name="position">3</property>
            </packing>
          </child>
        </object>
        <packing>
          <property name="expand">False</property>
          <property name="fill">False</property>
          <property name="position">0</property>
        </packing>
      </child>
    </object>
  </child>
  <action-widgets>
    <action-widget response="-6">button2</action-widget>
    <action-widget response="-5">button1</action-widget>
  </action-widgets>
  <child type="titlebar">
    <object class="GtkHeaderBar">
      <property name="visible">True</property>
      <property name="can_focus">False</property>
      <property name="title">Choose image...</property>
      <property name="show_close_button">True</property>
    </object>
  </child>
  <action-widgets>
    <action-widget response="-6">button2</action-widget>
    <action-widget response="-5">button1</action-widget>
  </action-widgets>
 </object>
</interface>
```

17.3.2 Python

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

import os
import sys
import gi
gi.require_version('Gtk', '3.0')
from gi.repository import Gtk, Gio, GdkPixbuf

class Handler:

    def on_window_destroy(self,window):
        window.close()

    def on_dialog_close(self,widget,*event):
```

```
        widget.hide_on_delete()
        return True

    def on_filechooser_dialog_response(self,widget,response):
        if response == -6:
            print("Cancel")
        elif response == -5:
            print("File selection: %s" % widget.get_filename())
        self.on_dialog_close(widget)

    def on_filechooser_dialog_file_activated(self,widget):
        self.on_filechooser_dialog_response(widget,-5)

    def on_filechooser_dialog_update_preview(self,widget):
        if widget.get_filename() != None and os.path.isfile(widget.get_filename()):
            pixbuf = GdkPixbuf.Pixbuf.new_from_file_at_scale(widget.get_filename(),
↪200,200,True)
            app.obj("preview").set_from_pixbuf(pixbuf)

    def on_file_button_clicked(self,widget):
        app.obj("filechooser_dialog").show_all()

    def on_dir_button_clicked(self,widget):

        dialog = Gtk.FileChooserDialog("Choose a folder",
                                       app.obj("window"),
                                       Gtk.FileChooserAction.SELECT_FOLDER,
                                       (Gtk.STOCK_CANCEL, Gtk.ResponseType.CANCEL,
                                       Gtk.STOCK_APPLY, Gtk.ResponseType.OK))
        dialog.set_default_size(600, 300)

        response = dialog.run()
        if response == Gtk.ResponseType.OK:
            print("Folder selection: %s" % dialog.get_filename())
        elif response == Gtk.ResponseType.CANCEL:
            print("Cancel")

        dialog.destroy()

class ExampleApp:

    def __init__(self):

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

    def on_app_activate(self, app):
        builder = Gtk.Builder()
        builder.add_from_file("16_filechooser.glade")
        builder.connect_signals(Handler())

        self.obj = builder.get_object
        self.obj("window").set_application(app)
        self.obj("window").set_wmclass("Filechooser example","Filechooser example")
        self.obj("window").show_all()

        #add filters to filechooser dialog
        self.obj("filefilter").set_name("Image files")
        self.obj("filechooser_dialog").add_filter(self.obj("filefilter"))
        self.obj("png_filter").set_name("PNG files")
        self.obj("filechooser_dialog").add_filter(self.obj("png_filter"))
```

```
        self.obj("jpg_filter").set_name("JPG files")
        self.obj("filechooser_dialog").add_filter(self.obj("jpg_filter"))

        #add buttons to headerbar of Glade generated dialog
        button = Gtk.Button.new_from_stock(Gtk.STOCK_CANCEL)
        button.set_property("can-default",True)
        self.obj("filechooser_dialog").add_action_widget(button, Gtk.ResponseType.
↪CANCEL)
        button = Gtk.Button.new_from_stock(Gtk.STOCK_APPLY)
        button.set_property("can-default",True)
        self.obj("filechooser_dialog").add_action_widget(button, Gtk.ResponseType.
↪OK)

    def on_app_shutdown(self, app):
        self.app.quit()

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

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

KAPITEL 18

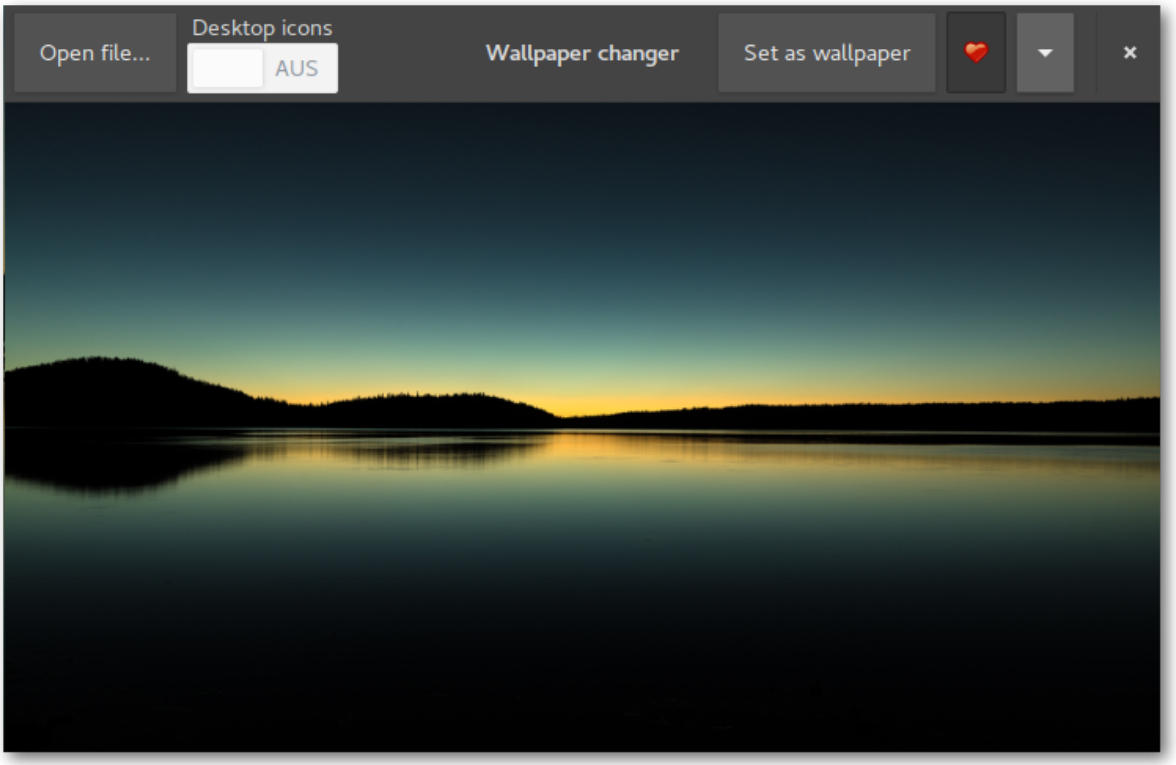
Das Konfigurationssystem GSettings

Das GNOME-eigene Konfigurationssystem GSettings

GSettings ist GNOMEs zentrales Konfigurationssystem für Anwendungen. Es ist die Schnittstelle für verschiedenmögliche Backends, gemeinhin ist dies dconf.

Mittels grafischem (dconf-editor) oder Kommandozeilentool (gsettings) lassen sich Konfigurationen abfragen und manipulieren.

Das folgende Beispiel kann Hintergrundbilder laden, festlegen und bookmarken/favorisieren.



18.1 Schemas

Um eine Konfiguration für eine Anwendung zu erstellen, muss diese in einer Schema-Datei definiert werden. Diese Datei ist eine XML-formatierte Datei, die anschließend in sein maschinenlesbares Äquivalent überführt werden muss.

Ein Beispiel für eine Schema-Datei mit einer festzulegenden Eigenschaft (key) wäre etwa:

```
<schemalist>
  <schema id="org.gtk.Test" path="/org/gtk/Test/">

    <key name="string-key" type="s">
      <default>""</default>
      <summary>A string</summary>
      <description>
        Configuration key defined for a string. Default value is set to an empty␣
↪string.
      </description>
    </key>

  </schema>
</schemalist>
```

Die Dateibenennung folgt der Vorgabe „`schema.id.gschema.xml`“. Das Standardinstallationsverzeichnis für Schema-Dateien ist `/usr/share/glib-2.0/schemas`. Schema-Dateien können auch außerhalb dieses Verzeichnisses genutzt werden (z.B. lokal, zu Testzwecken), sie werden dann aber nicht vom dconf-editor angezeigt.

Die erforderliche Kompilierung erfolgt mit

```
glib-compile-schemas /path/to/schema/files/
#default directory
glib-compile-schemas /usr/share/glib-2.0/schemas/
```

Die kompilierte und nun von GSettings verwendete Datei ist `gschemas.compiled`.

18.2 Glade

Das Beispiel soll Bilder anzeigen, dafür wird das Widget *GtkImage* benötigt. Alle Steuerungselemente werden in der Headerbar untergebracht:

- „Open File“-Button: öffnet einen FileChooserDialog (*Dateiauswahldialog*)
- Switch: schaltet Desktop-Icons an oder ab
- „Fav“-Togglebutton: bookmarkt angezeigte Dateien, zeigt an, ob angezeigte Datei als Favorit markiert ist
- „Set as wallpaper“-Button: angezeigte Datei als Hintergrundbild verwenden
- *MenuButton*: unterhalb des Buttons wird eine Liste der favorisierten Dateien angezeigt, die von dort aus aufgerufen werden können

18.3 Python

18.3.1 Globales Schema laden

Eine bestehende Konfiguration zu laden, geschieht einfach per

```
setting = Gio.Settings.new("full.schema.path")
#load desktop background configuration
setting = Gio.Settings.new("org.gnome.desktop.background")
```

18.3.2 Lokales Schema laden

Bei einem lokal gespeicherten Schema muss der Ort der `schemas.compiled` angegeben werden, bevor die Konfiguration geladen werden kann:

```
schema_source = Gio.SettingsSchemaSource.new_from_directory(os.getcwd(),
            Gio.SettingsSchemaSource.get_default(), False)
schema = Gio.SettingsSchemaSource.lookup(schema_source,"org.example.wallpaper-
↪changer",False)
setting = Gio.Settings.new_full(schema, None, None)
```

18.3.3 Widget verknüpfen

Es ist möglich, GSettings-Eigenschaften direkt an Widgets zu binden. Diese können dann bidirektional Zustände anzeigen bzw. man kann Änderungen an ihnen vornehmen:

```
setting.bind("setting-key", widget, property, Gio.SettingsBindFlags...)
```

Im Beispiel wäre dies also

```
self.bg_setting.bind("show-desktop-icons", self.obj("switch"), "active", Gio.
↪SettingsBindFlags.DEFAULT)
```

Der Schalter zeigt beim Start die aktuelle Einstellung an. Eine Änderung des Status ist sofort wirksam.

18.3.4 Werte abrufen und festlegen

Eigenschaften können mit `get_"type"` und `set_"type"` ermittelt und festgelegt werden. Die relevante Funktion ist vom festgelegten Schlüsseltyp abhängig, also `get_string` und `set_string` für Zeichenketten, `get_int` und `set_int` für Ganzzahlen usw. (siehe PyGObject API Reference).

Wird der Wert einer Eigenschaft per `get_value(key)` abgefragt, wird dies immer als Wert des Typs GLib.Variant zurückgegeben. Entsprechend erwartet die Funktion `set_value(key)` ebenfalls diesen Typ.

Die Inhalte dieser Werte lassen sich einfach in simple Datentypen konvertieren, z.B.

```
#return string
setting.get_value(key).get_string()
#return anything (list, string, bool etc.)
setting.get_value(key).unpack()
```

Umgekehrt lassen sich reguläre Datentypen nach folgendem Muster als *GLib.Variant*-Typ ausdrücken und an GSettings übergeben:

```
setting.set_value(key, GLib.Variant(string_type, value)
```

Eine Liste der verfügbaren Stringtypen finden sich in der GNOME Developer-Dokumentation.

Im Beispiel wird auf diese Art die Favoritenliste aktualisiert:

```
app_setting.set_value("favourites", GLib.Variant('as',fav_list))
```

18.4 Links

- Standardeinstellungen in GSettings ändern (German)
- Introduction to GSettings in Python
- Short Example of GSettings Bindings with Python using a Gtk Switch
- GSettings - flexible configuration system
- dconf (ubuntuusers-Wiki)
- First steps with GSettings

18.5 Listings

18.5.1 Schema

```
<?xml version="1.0" encoding="utf-8"?>

<schemalist>

  <schema path="/org/example/wallpaper-changer/" id="org.example.wallpaper-changer
↪">

    <key name="favourites" type="as">
      <default>[]</default>
      <summary>List of favourite wallpapers</summary>
      <description>
        Add or remove entry by pressing the 'fav' toggle button.
      </description>
    </key>

  </schema>

</schemalist>
```

18.5.2 Glade

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- Generated with glade 3.20.0 -->
<interface>
  <requires lib="gtk+" version="3.20"/>
  <object class="GtkFileFilter" id="filefilter">
    <mime-types>
      <mime-type>image/*</mime-type>
    </mime-types>
  </object>
  <object class="GtkImage" id="image1">
    <property name="visible">True</property>
    <property name="can_focus">False</property>
    <property name="icon_name">emblem-favorite</property>
  </object>
  <object class="GtkMenu" id="menu">
    <property name="visible">True</property>
    <property name="can_focus">False</property>
  </object>
  <object class="GtkApplicationWindow" id="window">
    <property name="can_focus">False</property>
```

```
    <signal name="destroy" handler="on_window_destroy" swapped="no"/>
    <signal name="size-allocate" handler="on_window_size_allocate" swapped="no"/>
    <child>
      <object class="GtkImage" id="image_area">
        <property name="width_request">400</property>
        <property name="height_request">300</property>
        <property name="visible">True</property>
        <property name="can_focus">False</property>
        <property name="stock">gtk-missing-image</property>
      </object>
    </child>
    <child type="titlebar">
      <object class="GtkHeaderBar">
        <property name="visible">True</property>
        <property name="can_focus">False</property>
        <property name="title">Wallpaper changer</property>
        <property name="has_subtitle">False</property>
        <property name="show_close_button">True</property>
        <child>
          <object class="GtkButton" id="open_button">
            <property name="label" translatable="yes">Open file...</property>
            <property name="visible">True</property>
            <property name="can_focus">True</property>
            <property name="receives_default">True</property>
            <signal name="clicked" handler="on_open_button_clicked" swapped="no"/>
          </object>
        </child>
        <child>
          <object class="GtkBox">
            <property name="visible">True</property>
            <property name="can_focus">False</property>
            <property name="orientation">vertical</property>
            <child>
              <object class="GtkLabel">
                <property name="visible">True</property>
                <property name="can_focus">False</property>
                <property name="label" translatable="yes">Desktop icons</property>
              </object>
              <packing>
                <property name="expand">False</property>
                <property name="fill">True</property>
                <property name="position">0</property>
              </packing>
            </child>
            <child>
              <object class="GtkSwitch" id="switch">
                <property name="visible">True</property>
                <property name="can_focus">True</property>
              </object>
              <packing>
                <property name="expand">False</property>
                <property name="fill">True</property>
                <property name="position">1</property>
              </packing>
            </child>
          </object>
          <packing>
            <property name="position">2</property>
          </packing>
        </child>
        <child>
          <object class="GtkMenuButton" id="fav_menu">
            <property name="visible">True</property>
```

```
            <property name="can_focus">True</property>
            <property name="receives_default">True</property>
            <property name="popup">menu</property>
            <child>
              <placeholder/>
            </child>
          </object>
          <packing>
            <property name="pack_type">end</property>
            <property name="position">1</property>
          </packing>
        </child>
        <child>
          <object class="GtkButton" id="setwp_button">
            <property name="label" translatable="yes">Set as wallpaper</property>
            <property name="visible">True</property>
            <property name="can_focus">True</property>
            <property name="receives_default">True</property>
            <signal name="clicked" handler="on_setwp_button_clicked" swapped="no"/>
          </object>
          <packing>
            <property name="pack_type">end</property>
            <property name="position">3</property>
          </packing>
        </child>
        <child>
          <object class="GtkToggleButton" id="fav_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="toggled" handler="on_fav_button_toggled" swapped="no"/>
          </object>
          <packing>
            <property name="pack_type">end</property>
            <property name="position">3</property>
          </packing>
        </child>
      </object>
    </child>
  </object>
  <object class="GtkImage" id="preview">
    <property name="width_request">200</property>
    <property name="visible">True</property>
    <property name="can_focus">False</property>
    <property name="margin_right">5</property>
  </object>
  <object class="GtkFileChooserDialog" id="filechooser_dialog">
    <property name="width_request">800</property>
    <property name="height_request">600</property>
    <property name="can_focus">False</property>
    <property name="type_hint">dialog</property>
    <property name="transient_for">window</property>
    <property name="attached_to">window</property>
    <property name="filter">filefilter</property>
    <property name="preview_widget">preview</property>
    <property name="use_preview_label">False</property>
    <signal name="delete-event" handler="on_dialog_close" swapped="no"/>
    <signal name="file-activated" handler="on_filechooser_dialog_file_activated"␣
↪swapped="no"/>
    <signal name="response" handler="on_filechooser_dialog_response" swapped="no"/>
    <signal name="update-preview" handler="on_filechooser_dialog_update_preview"␣
↪swapped="no"/>
```

```
    <child internal-child="vbox">
      <object class="GtkBox" id="fcbox">
        <property name="can_focus">False</property>
        <property name="orientation">vertical</property>
        <child internal-child="action_area">
          <object class="GtkButtonBox">
            <property name="can_focus">False</property>
            <child>
              <placeholder/>
            </child>
          </object>
          <packing>
            <property name="expand">False</property>
            <property name="fill">False</property>
            <property name="position">0</property>
          </packing>
        </child>
      </object>
    </child>
    <child type="titlebar">
      <object class="GtkHeaderBar">
        <property name="visible">True</property>
        <property name="can_focus">False</property>
        <property name="title">Choose image file</property>
        <property name="show_close_button">True</property>
      </object>
    </child>
  </object>
</interface>
```

18.5.3 Python

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

import sys
import os
import gi
gi.require_version('Gtk','3.0')
from gi.repository import Gtk, Gio, GLib, GdkPixbuf

class Handler:

    def on_window_destroy(self,window):
        window.close()

    def on_dialog_close(self,widget,*event):
        widget.hide_on_delete()
        return True

    def on_filechooser_dialog_response(self,widget,response):
        if response == 1:
            self.on_dialog_close(widget)
        elif response == 0:
            app.uri = widget.get_filename()
            app.draw_pixbuf(app.uri)
            app.handle_fav(app.uri)
            self.on_dialog_close(widget)

    def on_filechooser_dialog_file_activated(self,widget):
        self.on_filechooser_dialog_response(widget,0)
```

```
    def on_open_button_clicked(self,widget):
        app.obj("filechooser_dialog").show_all()

    def on_setwp_button_clicked(self,widget):
        app.bg_setting.set_string("picture-uri","file://%s" % app.uri)

    def on_window_size_allocate(self,widget,size):
        app.draw_pixbuf(app.uri)

    def on_filechooser_dialog_update_preview(self,widget):
        if widget.get_filename() != None and os.path.isfile(widget.get_filename()):
            pixbuf = GdkPixbuf.Pixbuf.new_from_file_at_scale(widget.get_filename(),
↪200,200,True)
            app.obj("preview").set_from_pixbuf(pixbuf)

    def on_fav_button_toggled(self,widget):
        if widget.get_active():
            #add file to fav_list if not in list
            if app.uri not in app.fav_list:
                app.fav_list.append(app.uri)
        else:
            #remove file from fav_list if in list
            if app.uri in app.fav_list:
                app.fav_list.remove(app.uri)
        #update GSettings entry for favourites
        app.app_setting.set_value("favourites", GLib.Variant('as', app.fav_list))
        #update fav list in popup menu
        popup = app.obj("menu")
        #remove all items
        for i in popup.get_children():
            popup.remove(i)
        #reload all items from fav_list
        for fav in app.fav_list:
            #only label menuitem with filename instead of path
            item=Gtk.MenuItem(os.path.split(fav)[1])
            item.connect("activate",self.on_choose_fav_from_menu,fav)
            popup.append(item)
        popup.show_all()

    def on_choose_fav_from_menu(self,widget,file):
        app.uri = file
        app.draw_pixbuf(file)
        app.handle_fav(file)

class ExampleApp:

    def __init__(self):

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

    def on_app_activate(self, app):
        builder = Gtk.Builder()
        builder.add_from_file("17_gsettings.glade")
        builder.connect_signals(Handler())
        self.obj = builder.get_object

        #load existing GSettings application config
        self.bg_setting = Gio.Settings.new("org.gnome.desktop.background")
        #get_value returns Gio formatted file path
```

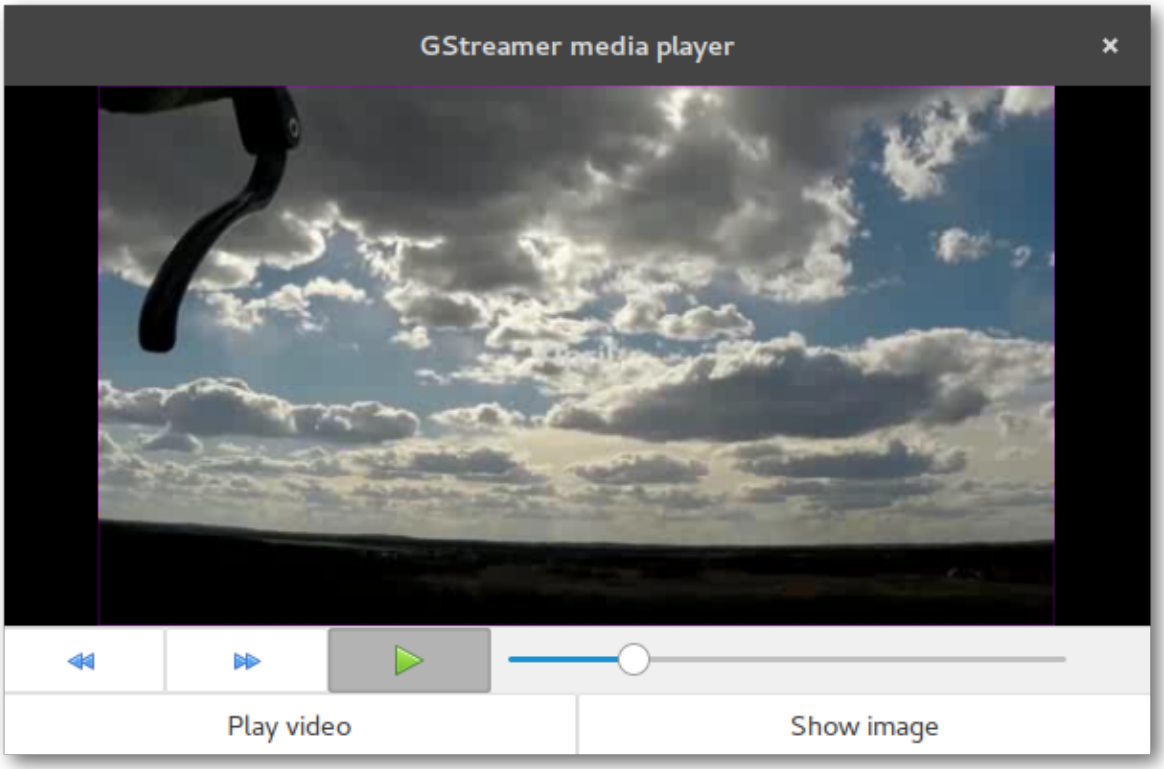
```
        file = self.bg_setting.get_value("picture-uri")
        #convert path into string
        self.uri = file.get_string()[7:]
        #bind GSettings key to GTK+ object
        self.bg_setting.bind("show-desktop-icons", self.obj("switch"), "active",␣
↪Gio.SettingsBindFlags.DEFAULT)

        #add GSettings schema from compiled XML file located in current directory␣
↪(only recommended for test use, standard location: /usr/share/glib-2.0/schemas/)
        schema_source = Gio.SettingsSchemaSource.new_from_directory(os.getcwd(),
                Gio.SettingsSchemaSource.get_default(), False)
        schema = Gio.SettingsSchemaSource.lookup(schema_source,"org.example.
↪wallpaper-changer",False)
        self.app_setting = Gio.Settings.new_full(schema, None, None)
        #convert value (GLib.Variant) into native list
        self.fav_list = self.app_setting.get_value("favourites").unpack()

        self.obj("window").set_application(app)
        self.obj("window").set_wmclass("Wallpaper changer","Wallpaper changer")
        self.obj("window").show_all()

        self.draw_pixbuf(self.uri)
        self.handle_fav(self.uri)

    def draw_pixbuf(self,file):
        size=self.obj("image_area").get_allocation()
        pixbuf = GdkPixbuf.Pixbuf.new_from_file_at_scale(file,size.width,size.
↪height,True)
        self.obj("image_area").set_from_pixbuf(pixbuf)

    def handle_fav(self,uri):
        #set toggle button to correct state
        if uri in self.fav_list:
            self.obj("fav_button").set_active(True)
        else:
            self.obj("fav_button").set_active(False)

    def on_app_shutdown(self, app):
        self.app.quit()

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

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

KAPITEL 19

Mediaplayer mit GStreamer

Mediaplayer mit GStreamer 1.x realisieren

GStreamer ist ein Multimedia-Framework, das zum Anzeigen und (De-)Kodieren von Mediendateien verwendet werden kann.



19.1 Glade

- **Darstellungsbereich der Mediendatei:** Widget *Gtk.DrawingArea*
- **Steuerungselemente:** Vor-/Zurückspulen (*Gtk.Button*), Pause (*Gtk.Togglebutton*)

- **Medienauswahl:** Buttons, um Video- oder Bilddatei anzuzeigen

19.2 Python

19.2.1 Player einrichten

Elemente und Pipelines

GStreamer handhabt alle möglichen Arten von Medienflüssen. Jeder Schritt in dieser Verarbeitungskette wird per *Element* definiert und in *Pipelines* verbunden. Eine solche Pipeline besteht typischerweise aus „source“-, „filter“-/“decode“- und „sink“-Elementen.

```
-------------------------------------------------------
|  Pipeline                                           |
|                                                     |
|  -------------   ----------------   -------------- |
|  | source    |   | filter       |   | sink       | |
|  |           |->>| decoder      |->>|            | |
|  | Quelle    |   | Verarbeitung |   | Ausgabe    | |
|  -------------   ----------------   -------------- |
-------------------------------------------------------
```

Nach diesem Prinzip wird dies mittels *Gst*-Modul umgesetzt:

```
#init Gst and create pipeline
Gst.init()
pipeline = Gst.Pipeline()

#create elements
src = Gst.ElementFactory.make("filesrc","source")
decode = Gst.ElementFactory.make("decodebin","decode")
sink = Gst.ElementFactory.make("xvimagesink")

#configure elements
src.set_property("location",file_location)

#add elements to pipeline
pipeline.add(src)
pipeline.add(decode)
pipeline.add(sink)

#link elements together
src.link(decode)
decode.link(sink)
```

Fertige Pipelines

Es besteht auch beispielsweise die Möglichkeit, Audio- und Videosignale voneinander getrennt werden, indem jeweils ein „videosink“ und ein „audiosink“ erstellt usw. Auf der anderen Seite gibt es vorgefertigte Pipelines für Standardaufgaben wie etwa das Abspielen von Medien. Ein solches Element ist „playbin“, das den Code signifikant vereinfacht:

```
Gst.init(None)
player = Gst.ElementFactory.make("playbin","player")
sink = Gst.ElementFactory.make("xvimagesink")
player.set_property("uri",uri_of_file)
player.set_property("video-sink", sink)
```

19.2.2 Und los!

Eine Pipeline oder ein „playbin“-Element können nun über *Gst.STATE* gesteuert werden:

```
player.set_state(Gst.State.PLAYING)
player.set_state(Gst.State.PAUSED)
```

19.2.3 Fortschrittsanzeige

Die Fortschrittsanzeige ist an dieser Stelle keine *Gtk.ProgressBar* sondern eine horizontale *Gtk.Scale*. Mit diesem Widget lässt sich nicht nur eine Position anzeigen, sondern auch per Maus setzen. Für letzteres wird das Signal *value-changed* benötigt. Streng genommen ist das Signal *change-value* an dieser Stelle die sauberere Lösung, die im nachfolgenden Beitrag (*Mediaplayer mit VLC*)_ zur Umsetzung des Mediaplayers mit LibVLC verwendet wird.

19.3 Möglichkeiten und Limitierungen

Bei der Einarbeitung in GStreamer stolpert man (an dieser Stelle generalisiert die Autorin weitgehend und möglicherweise unbegründet) über diverse Hürden:

Es gibt eine Reihe von Tutorials. Die Umsetzung wird durch zwei Umstände erschwert:

1. Die primäre Sprache von und mit GStreamer ist C. Mit Python steht man eher auf experimentellem Boden.
2. Durch die Versionssprünge sowohl bei GStreamer (von 0.10 auf 1.x) als auch Python (2.x auf 3.x) funktionieren viele ältere Anleitungen nicht mehr ohne weiteres.

Es gibt weiterhin Effekte, die sich mir nicht erschließen. Das in diesem Artikel aufgeführte Beispiel funktioniert nicht, wenn das Gtk-Fenster eine Headerbar enthält. Es sollte mit der Verwendung von „gtksink“ lösbar sein, aber dies wiederum verweigert das Abspielen in einem zugewiesenen Widget. Die Komplexität und Mächtigkeit von GStreamer ist massiv verwirrend.

19.4 Links

- Tutorial on using GStreamer Python Bindings in org-mode
- gstreamer-python-player/seek.py example
- GStreamer List of Elements and Plugins
- GStreamer documentation
- Using GStreamer 1.0 with Python
- Mediaplayer mit VLC (*Mediaplayer mit VLC*)_

19.5 Listings

19.5.1 Glade

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- Generated with glade 3.20.0 -->
<interface>
  <requires lib="gtk+" version="3.16"/>
  <object class="GtkAdjustment" id="adjustment">
    <property name="upper">100</property>
```

```
    <property name="step_increment">1</property>
    <property name="page_increment">10</property>
  </object>
  <object class="GtkImage" id="image1">
    <property name="visible">True</property>
    <property name="can_focus">False</property>
    <property name="stock">gtk-media-rewind</property>
  </object>
  <object class="GtkImage" id="image2">
    <property name="visible">True</property>
    <property name="can_focus">False</property>
    <property name="stock">gtk-media-forward</property>
  </object>
  <object class="GtkImage" id="image3">
    <property name="visible">True</property>
    <property name="can_focus">False</property>
    <property name="stock">gtk-media-pause</property>
  </object>
  <object class="GtkWindow" id="window">
    <property name="can_focus">False</property>
    <property name="title" translatable="yes">GStreamer media player</property>
    <property name="default_width">600</property>
    <property name="default_height">350</property>
    <signal name="destroy" handler="on_window_destroy" swapped="no"/>
    <child>
      <object class="GtkBox" id="box1">
        <property name="visible">True</property>
        <property name="can_focus">False</property>
        <property name="orientation">vertical</property>
        <child>
          <object class="GtkDrawingArea" id="play_here">
            <property name="visible">True</property>
            <property name="can_focus">False</property>
          </object>
          <packing>
            <property name="expand">True</property>
            <property name="fill">True</property>
            <property name="position">0</property>
          </packing>
        </child>
        <child>
          <object class="GtkSeparator" id="separator1">
            <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">1</property>
          </packing>
        </child>
        <child>
          <object class="GtkBox" id="box3">
            <property name="visible">True</property>
            <property name="can_focus">False</property>
            <child>
              <object class="GtkButtonBox" id="buttonbox1">
                <property name="visible">True</property>
                <property name="can_focus">False</property>
                <property name="layout_style">start</property>
                <child>
                  <object class="GtkButton" id="backward">
                    <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_backward_clicked" swapped=
↪"no"/>
                  </object>
                  <packing>
                    <property name="expand">True</property>
                    <property name="fill">True</property>
                    <property name="position">0</property>
                  </packing>
                </child>
                <child>
                  <object class="GtkButton" id="forward">
                    <property name="visible">True</property>
                    <property name="can_focus">True</property>
                    <property name="receives_default">True</property>
                    <property name="image">image2</property>
                    <property name="always_show_image">True</property>
                    <signal name="clicked" handler="on_forward_clicked" swapped="no
↪"/>
                  </object>
                  <packing>
                    <property name="expand">True</property>
                    <property name="fill">True</property>
                    <property name="position">1</property>
                  </packing>
                </child>
                <child>
                  <object class="GtkToggleButton" id="playpause_togglebutton">
                    <property name="visible">True</property>
                    <property name="can_focus">True</property>
                    <property name="receives_default">True</property>
                    <property name="image">image3</property>
                    <signal name="toggled" handler="on_playpause_togglebutton_
↪toggled" swapped="no"/>
                  </object>
                  <packing>
                    <property name="expand">True</property>
                    <property name="fill">True</property>
                    <property name="position">2</property>
                  </packing>
                </child>
              </object>
              <packing>
                <property name="expand">False</property>
                <property name="fill">False</property>
                <property name="position">0</property>
              </packing>
            </child>
            <child>
              <object class="GtkScale" id="progress">
                <property name="width_request">300</property>
                <property name="visible">True</property>
                <property name="can_focus">True</property>
                <property name="halign">center</property>
                <property name="margin_left">5</property>
                <property name="margin_right">5</property>
                <property name="adjustment">adjustment</property>
                <property name="fill_level">100</property>
                <property name="round_digits">1</property>
                <property name="draw_value">False</property>
```

```
            </object>
            <packing>
              <property name="expand">False</property>
              <property name="fill">False</property>
              <property name="position">1</property>
            </packing>
          </child>
        </object>
        <packing>
          <property name="expand">False</property>
          <property name="fill">True</property>
          <property name="position">2</property>
        </packing>
      </child>
      <child>
        <object class="GtkBox" id="box2">
          <property name="visible">True</property>
          <property name="can_focus">False</property>
          <property name="homogeneous">True</property>
          <child>
            <object class="GtkButton" id="vbutton">
              <property name="label" translatable="yes">Play video</property>
              <property name="visible">True</property>
              <property name="can_focus">True</property>
              <property name="receives_default">True</property>
              <signal name="clicked" handler="on_vbutton_clicked" swapped="no"/>
            </object>
            <packing>
              <property name="expand">True</property>
              <property name="fill">True</property>
              <property name="position">0</property>
            </packing>
          </child>
          <child>
            <object class="GtkButton" id="ibutton">
              <property name="label" translatable="yes">Show image</property>
              <property name="visible">True</property>
              <property name="can_focus">True</property>
              <property name="receives_default">True</property>
              <signal name="clicked" handler="on_ibutton_clicked" swapped="no"/>
            </object>
            <packing>
              <property name="expand">True</property>
              <property name="fill">True</property>
              <property name="position">1</property>
            </packing>
          </child>
        </object>
        <packing>
          <property name="expand">False</property>
          <property name="fill">True</property>
          <property name="position">3</property>
        </packing>
      </child>
    </object>
  </child>
  <child>
    <placeholder/>
  </child>
 </object>
</interface>
```

19.5.2 Python

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

import os
import time
import gi

gi.require_version('Gtk','3.0')
gi.require_version('Gst','1.0')
gi.require_version('GstVideo','1.0')

from gi.repository import Gst, Gtk, GLib, GstVideo

class GenericException(Exception):
    pass

class Handler:

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

    def on_playpause_togglebutton_toggled(self,widget):
        if app.playpause_button.get_active():
            img = Gtk.Image.new_from_stock(Gtk.STOCK_MEDIA_PLAY,Gtk.IconSize.
↪BUTTON)
            widget.set_property("image",img)
            app.pause()
        else:
            img = Gtk.Image.new_from_stock(Gtk.STOCK_MEDIA_PAUSE,Gtk.IconSize.
↪BUTTON)
            widget.set_property("image",img)
            app.play()

    def on_forward_clicked(self,widget):
        app.skip_time()

    def on_backward_clicked(self,widget):
        app.skip_time(-1)

    def on_progress_value_changed(self,widget):
        app.on_slider_seek

    def on_vbutton_clicked(self,widget):
        app.clear_playbin()
        app.setup_player("mediaplayer.avi")
        if app.playpause_button.get_active() is True:
            app.playpause_button.set_active(False)
        else:
            app.play()

    def on_ibutton_clicked(self,widget):
        app.clear_playbin()
        app.setup_player("mediaplayer.jpg")
        app.pause()

class GstPlayer:

    def __init__(self):

        #init GStreamer
        Gst.init(None)
```

```
        #setting up builder
        builder = Gtk.Builder()
        builder.add_from_file("19_gst_player.glade")
        builder.connect_signals(Handler())

        self.movie_window = builder.get_object("play_here")
        self.playpause_button = builder.get_object("playpause_togglebutton")
        self.slider = builder.get_object("progress")
        self.slider_handler_id = self.slider.connect("value-changed", self.on_
↪slider_seek)

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

        #setting up videoplayer
        self.player = Gst.ElementFactory.make("playbin","player")
        self.sink = Gst.ElementFactory.make("xvimagesink")
        self.sink.set_property("force-aspect-ratio",True)

    def setup_player(self,f):
        #file to play must be transmitted as uri
        uri = "file://"+os.path.abspath(f)
        self.player.set_property("uri",uri)

        #make playbin play in specified DrawingArea widget instead of separate,␣
↪GstVideo needed
        win_id = self.movie_window.get_property('window').get_xid()
        self.sink.set_window_handle(win_id)
        self.player.set_property("video-sink",self.sink)

    def play(self):
        self.is_playing = True
        self.player.set_state(Gst.State.PLAYING)
        #starting up a timer to check on the current playback value
        GLib.timeout_add(1000, self.update_slider)

    def pause(self):
        self.is_playing = False
        self.player.set_state(Gst.State.PAUSED)

    def current_position(self):
        status,position = self.player.query_position(Gst.Format.TIME)
        return position

    #skip 20 seconds on forward/backward button
    def skip_time(self,direction=1):
        app.player.seek_simple(Gst.Format.TIME,  Gst.SeekFlags.FLUSH | Gst.
↪SeekFlags.KEY_UNIT, self.current_position() + float(20) * Gst.SECOND * direction␣
↪)

    def update_slider(self):
        if not self.is_playing:
            return False # cancel timeout
        else:
            success, self.duration = self.player.query_duration(Gst.Format.TIME)
            #adjust duration and position relative to absolute scale of 100
            self.mult = 100 / ( self.duration / Gst.SECOND )
            if not success:
                raise GenericException("Couldn't fetch duration")
            #fetching the position, in nanosecs
            success, position = self.player.query_position(Gst.Format.TIME)
            if not success:
```

```
                raise GenericException("Couldn't fetch current position to update␣
↪slider")

            # block seek handler so we don't seek when we set_value()
            self.slider.handler_block(self.slider_handler_id)

            self.slider.set_value( float(position) / Gst.SECOND * self.mult)

            self.slider.handler_unblock(self.slider_handler_id)
        return True # continue calling every x milliseconds

    def on_slider_seek(self,widget):
        seek_time = app.slider.get_value()
        self.player.seek_simple(Gst.Format.TIME,  Gst.SeekFlags.FLUSH | Gst.
↪SeekFlags.KEY_UNIT, seek_time * Gst.SECOND / self.mult)

    def clear_playbin(self):
        try:
            self.player.set_state(Gst.State.NULL)
        except:
            pass

    def main(self):
        Gtk.main()

app = GstPlayer()
app.main()
```

19.5.3 Beispieldateien

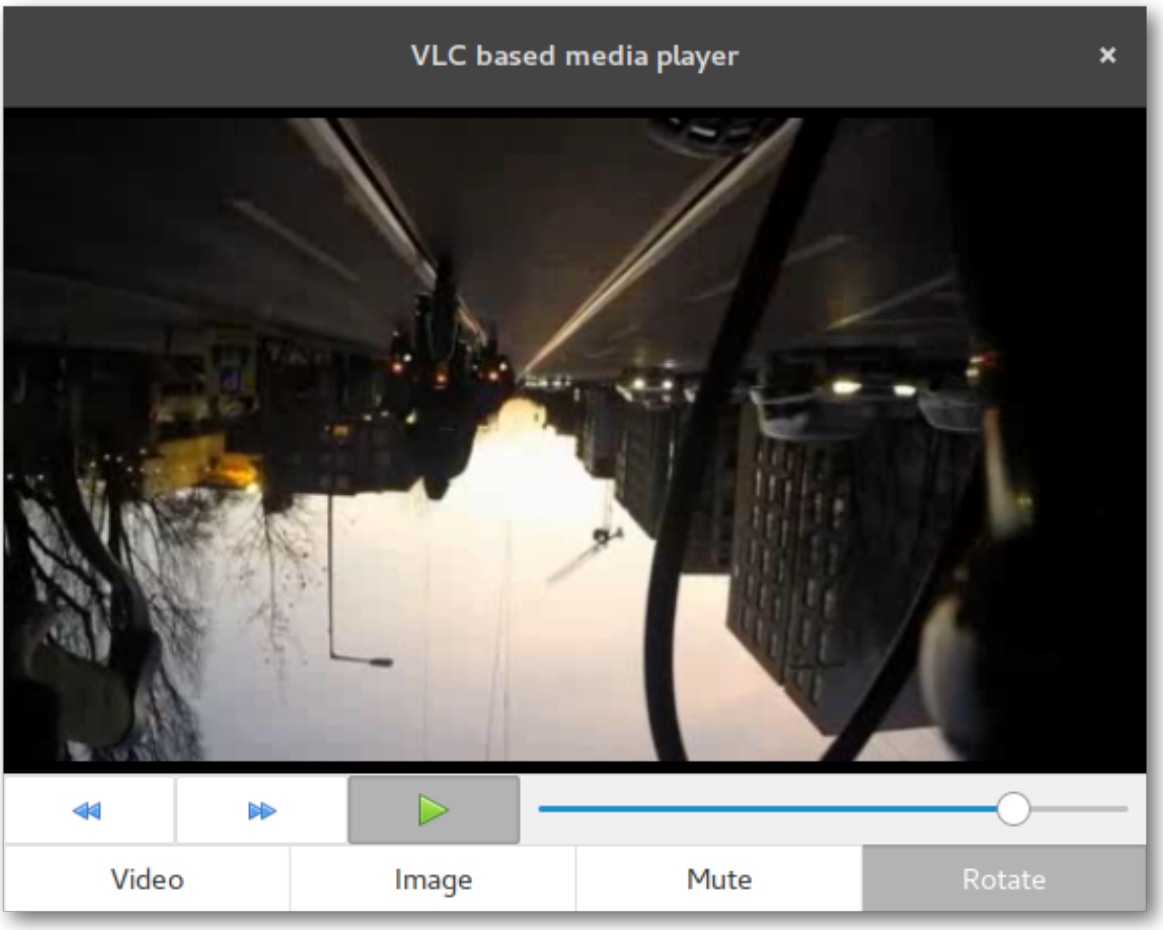
- Video (13,7 MB)
- Image (553 kB)

KAPITEL 20

Mediaplayer mit VLC

Mediaplayer mit LibVLC realisieren

VLC ist nicht nur ein Multimediaplayer, sondern auch ein Framework, zu dem Python-Bindings verfügbar sind. In diesem Beispiel wird analog zum GStreamer-Artikel (*Mediaplayer mit GStreamer*)_ ein einfacher Mediaplayer mittels LibVLC umgesetzt.



20.1 LibVLC

Voraussetzung für die Verwendung ist die Installation der Python-Bindings. Diese sind unter der Paketbezeichnung `python-vlc` zu finden.

20.2 Glade

- **Darstellungsbereich der Mediendatei:** Widget *Gtk.DrawingArea*
- **Steuerungselemente:** Vor-/Zurückspulen (*Gtk.Button*), Pause (*Gtk.Togglebutton*)
- **Medienauswahl:** Buttons, um Video- oder Bilddatei anzuzeigen
- **Playback manipulieren:** Buttons zum Stummschalten und Drehen des Videos

20.3 Python

20.3.1 Player einrichten

Der VLC-Player wird initiiert, sobald das dazugehörige Widget, in diesem Fall also *Gtk.DrawingArea* gezeichnet wird. Dazu wird das Signal `realize` genutzt, das grundsätzlich für die Klasse der Widgets verfügbar ist.

```
vlcOptions = "--no-xlib"
win_id = widget.get_window().get_xid()
setup_player(vlcOptions)
vlcInstance = vlc.Instance(options)
player = vlcInstance.media_player_new()
player.set_xwindow(win_id)
```

Als Optionen können Kommandozeilenoptionen von VLC übergeben werden. Im Beispiel wird beim Klick auf den „Rotate“-Button das Bild um 180° gedreht. Der Player wird erneut initiiert und die zusätzliche Option `--video-filter=transform{type=180}` übergeben.

20.3.2 Medium abspielen

Wie auch der GStreamer-Player kann der VLC-Player viele Video-/Audio- oder Bild-Formate anzeigen bzw. abspielen.

```
player.set_mrl(file_url)
#Datei abspielen
player.play()
#Pause/Play-Schalter
player.pause()
```

20.3.3 Positionsanzeige

Die Umsetzung des Fortschrittsbalkens und die Nutzung als Schiebereglers gestaltet sich ziemlich einfach.

```
#Position abfragen
player.get_position()
#Position bestimmen
player.set_position(val)
```

Der Wertebereich liegt dabei zwischen 0 und 1. Das Problem bei diesen Funktionen ist, dass sie relativ ressourcenintensiv arbeiten und das Playback mitunter verruckelt ist. Die Lösung im hiesigen Beispiel besteht darin, `get_position`-Abfragen zu umgehen, indem die Regler-Position herangezogen wird.

20.4 Möglichkeiten und Limitierungen

Die Nutzung der LibVLC-Python-Bindings erweist sich als einfach und angesichts der GStreamer-Umsetzung als geradezu intuitiv. Auch das „Headerbar-Problem“ besteht nicht.

Auf der anderen Seite greift man hier auf großes Projekt zurück, man muss VLC und die Python-Bindings installiert haben anstatt einfach das GStreamer-Modul aus dem GObject Introspection-Repository zu verwenden. Auch ist im Test der Ressourcenverbrauch von VLC gegenüber GStreamer größer.

20.5 Links

- LibVLC Python bindings documentation
- How to Build a Python Media Player using LibVLC and GTK+
- How to get video duration in seconds?
- Mediaplayer mit GStreamer (*Mediaplayer mit GStreamer*)_

20.6 Listings

20.6.1 Glade

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- Generated with glade 3.20.0 -->
<interface>
  <requires lib="gtk+" version="3.16"/>
  <object class="GtkAdjustment" id="adjustment">
    <property name="upper">100</property>
    <property name="step_increment">1</property>
    <property name="page_increment">10</property>
  </object>
  <object class="GtkImage" id="image1">
    <property name="visible">True</property>
    <property name="can_focus">False</property>
    <property name="stock">gtk-media-rewind</property>
  </object>
  <object class="GtkImage" id="image2">
    <property name="visible">True</property>
    <property name="can_focus">False</property>
    <property name="stock">gtk-media-forward</property>
  </object>
  <object class="GtkImage" id="image3">
    <property name="visible">True</property>
    <property name="can_focus">False</property>
    <property name="stock">gtk-media-pause</property>
  </object>
  <object class="GtkApplicationWindow" id="window">
    <property name="width_request">600</property>
    <property name="height_request">500</property>
    <property name="can_focus">False</property>
    <property name="default_width">440</property>
    <property name="default_height">250</property>
```

```
    <child>
      <object class="GtkBox" id="box1">
        <property name="visible">True</property>
        <property name="can_focus">False</property>
        <property name="orientation">vertical</property>
        <child>
          <object class="GtkDrawingArea" id="play_here">
            <property name="visible">True</property>
            <property name="can_focus">False</property>
            <signal name="realize" handler="on_play_here_realize" swapped="no"/>
          </object>
          <packing>
            <property name="expand">True</property>
            <property name="fill">True</property>
            <property name="position">0</property>
          </packing>
        </child>
        <child>
          <object class="GtkSeparator" id="separator1">
            <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">1</property>
          </packing>
        </child>
        <child>
          <object class="GtkBox" id="box3">
            <property name="visible">True</property>
            <property name="can_focus">False</property>
            <child>
              <object class="GtkButtonBox" id="buttonbox1">
                <property name="visible">True</property>
                <property name="can_focus">False</property>
                <property name="layout_style">start</property>
                <child>
                  <object class="GtkButton" id="backward">
                    <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_backward_clicked" swapped=
↪"no"/>
                  </object>
                  <packing>
                    <property name="expand">True</property>
                    <property name="fill">True</property>
                    <property name="position">0</property>
                  </packing>
                </child>
                <child>
                  <object class="GtkButton" id="forward">
                    <property name="visible">True</property>
                    <property name="can_focus">True</property>
                    <property name="receives_default">True</property>
                    <property name="image">image2</property>
                    <property name="always_show_image">True</property>
                    <signal name="clicked" handler="on_forward_clicked" swapped="no
↪"/>
                  </object>
```

```
                <packing>
                  <property name="expand">True</property>
                  <property name="fill">True</property>
                  <property name="position">1</property>
                </packing>
              </child>
              <child>
                <object class="GtkToggleButton" id="playpause_togglebutton">
                  <property name="visible">True</property>
                  <property name="can_focus">True</property>
                  <property name="receives_default">True</property>
                  <property name="image">image3</property>
                  <property name="always_show_image">True</property>
                  <signal name="toggled" handler="on_playpause_togglebutton_
↪toggled" swapped="no"/>
                </object>
                <packing>
                  <property name="expand">True</property>
                  <property name="fill">True</property>
                  <property name="position">2</property>
                </packing>
              </child>
            </object>
            <packing>
              <property name="expand">False</property>
              <property name="fill">False</property>
              <property name="position">0</property>
            </packing>
          </child>
          <child>
            <object class="GtkScale" id="progress">
              <property name="width_request">300</property>
              <property name="visible">True</property>
              <property name="can_focus">True</property>
              <property name="halign">center</property>
              <property name="margin_left">5</property>
              <property name="margin_right">5</property>
              <property name="adjustment">adjustment</property>
              <property name="fill_level">100</property>
              <property name="round_digits">1</property>
              <property name="draw_value">False</property>
              <signal name="change-value" handler="on_progress_change_value"␣
↪swapped="no"/>
            </object>
            <packing>
              <property name="expand">False</property>
              <property name="fill">False</property>
              <property name="position">1</property>
            </packing>
          </child>
        </object>
        <packing>
          <property name="expand">False</property>
          <property name="fill">True</property>
          <property name="position">2</property>
        </packing>
      </child>
      <child>
        <object class="GtkButtonBox" id="buttonbox2">
          <property name="visible">True</property>
          <property name="can_focus">False</property>
          <property name="homogeneous">True</property>
          <property name="layout_style">expand</property>
```

```
            <child>
              <object class="GtkButton" id="vbutton">
                <property name="label" translatable="yes">Video</property>
                <property name="visible">True</property>
                <property name="can_focus">True</property>
                <property name="receives_default">True</property>
                <signal name="clicked" handler="on_vbutton_clicked" swapped="no"/>
              </object>
              <packing>
                <property name="expand">True</property>
                <property name="fill">True</property>
                <property name="position">0</property>
              </packing>
            </child>
            <child>
              <object class="GtkButton" id="ibutton">
                <property name="label" translatable="yes">Image</property>
                <property name="visible">True</property>
                <property name="can_focus">True</property>
                <property name="receives_default">True</property>
                <signal name="clicked" handler="on_ibutton_clicked" swapped="no"/>
              </object>
              <packing>
                <property name="expand">True</property>
                <property name="fill">True</property>
                <property name="position">1</property>
              </packing>
            </child>
            <child>
              <object class="GtkToggleButton" id="mute">
                <property name="label" translatable="yes">Mute</property>
                <property name="visible">True</property>
                <property name="can_focus">True</property>
                <property name="receives_default">True</property>
                <signal name="toggled" handler="on_mute_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="GtkToggleButton" id="rotate">
                <property name="label" translatable="yes">Rotate</property>
                <property name="visible">True</property>
                <property name="can_focus">True</property>
                <property name="receives_default">True</property>
                <signal name="toggled" handler="on_rotate_toggled" swapped="no"/>
              </object>
              <packing>
                <property name="expand">True</property>
                <property name="fill">True</property>
                <property name="position">3</property>
              </packing>
            </child>
          </object>
          <packing>
            <property name="expand">False</property>
            <property name="fill">True</property>
            <property name="position">4</property>
          </packing>
        </child>
```

```
      </object>
    </child>
    <child type="titlebar">
      <object class="GtkHeaderBar">
        <property name="visible">True</property>
        <property name="can_focus">False</property>
        <property name="title">VLC based media player</property>
        <property name="show_close_button">True</property>
        <child>
          <placeholder/>
        </child>
      </object>
    </child>
  </object>
</interface>
```

20.6.2 Python

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

import sys
import os
import vlc
import subprocess
import time

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

from gi.repository import Gtk, Gio, Gdk, GLib

class Handler:

    def on_play_here_realize(self, widget):
        vlcOptions = "--no-xlib"
        self.win_id = widget.get_window().get_xid()
        self.setup_player(vlcOptions)
        self.player.audio_set_mute(False)
        self.is_playing = False

    def on_rotate_toggled(self, widget):
        pos = self.player.get_position()
        self.player.stop()
        self.player.release()
        self.vlcInstance.release()
        if widget.get_active():
            vlcOptions = "--no-xlib --video-filter=transform{type=180}"
        else:
            vlcOptions = "--no-xlib"
        self.setup_player(vlcOptions)
        self.player.set_mrl(self.video)
        self.player.play()
        self.player.set_position(pos)
        if not self.is_playing:
            time.sleep(.05)
            self.player.pause()

    def setup_player(self, options):
        self.vlcInstance = vlc.Instance(options)
        self.player = self.vlcInstance.media_player_new()
```

```
        self.player.set_xwindow(self.win_id)

    def on_backward_clicked(self, widget):
        skip_pos = go.slider.get_value() - 10
        if skip_pos < 0:
            self.player.set_position(0)
            go.slider.set_value(0)
        else:
            self.player.set_position(skip_pos / 100)
            go.slider.set_value(skip_pos)

    def on_forward_clicked(self, widget):
        skip_pos = go.slider.get_value() + 10
        if skip_pos > 100:
            self.player.pause()
            self.player.set_position(0.99)
            go.slider.set_value(100)
        else:
            self.player.set_position(skip_pos / 100)
            go.slider.set_value(skip_pos)

    def on_playpause_togglebutton_toggled(self, widget):
        if widget.get_active():
            img = Gtk.Image.new_from_stock(Gtk.STOCK_MEDIA_PLAY,Gtk.IconSize.
↪BUTTON)
            widget.set_property("image",img)
            self.is_playing = False
        else:
            img = Gtk.Image.new_from_stock(Gtk.STOCK_MEDIA_PAUSE,Gtk.IconSize.
↪BUTTON)
            widget.set_property("image",img)
            self.is_playing = True
        self.player.pause()
        GLib.timeout_add(1000, self.update_slider)

    def on_vbutton_clicked(self, widget):
        self.video = "file://"+os.path.abspath("mediaplayer.avi")
        self.duration = go.get_duration(self.video)
        self.player.set_mrl(self.video)
        self.is_playing = True
        go.slider.set_value(0)
        go.obj("playpause_togglebutton").set_active(False)
        go.obj("playpause_togglebutton").set_sensitive(True)
        go.obj("mute").set_sensitive(True)
        go.obj("rotate").set_sensitive(True)
        self.player.play()
        GLib.timeout_add(1000, self.update_slider)

    def on_ibutton_clicked(self, widget):
        image = "file://"+os.path.abspath("mediaplayer.jpg")
        self.player.set_mrl(image)
        self.is_playing = False
        self.player.play()
        go.obj("playpause_togglebutton").set_sensitive(False)
        go.obj("mute").set_sensitive(False)
        go.obj("rotate").set_sensitive(False)

    def on_mute_toggled(self, widget):
        if widget.get_active():
            widget.set_label("Unmute")
        else:
            widget.set_label("Mute")
        self.player.audio_toggle_mute()
```

```
    def on_progress_change_value(self, widget, scroll, value):
        self.player.set_position(value / 100)
        widget.set_value(value)

    def update_slider(self):
        if not self.is_playing:
            return False # cancel timeout
        else:
            pos = go.slider.get_value()
            new_pos = (pos + 100 / self.duration)
            go.slider.set_value(new_pos)
            if new_pos > 100:
                self.is_playing = False
        return True # continue calling every x milliseconds

class VlcPlayer:

    def __init__(self):
        self.app = Gtk.Application.new("org.media.player", Gio.ApplicationFlags(0))
        self.app.connect("activate", self.on_app_activate)

    def on_app_activate(self, app):
        #setting up builder
        builder = Gtk.Builder()
        builder.add_from_file("20_vlc_player_headerbar.glade")
        builder.connect_signals(Handler())
        self.obj = builder.get_object
        #slider position is float between 0..100
        self.slider = self.obj("progress")
        window = self.obj("window")
        window.set_application(app)
        window.show_all()

    def get_duration(self,video):
        command = ['ffprobe',
                        '-v', "error",
                        '-show_entries', "format=duration",
                        '-of', 'default=noprint_wrappers=1:nokey=1',
                        video
                        ]
        ffprobe_cmd = subprocess.run(command, stdout=subprocess.PIPE)
        #stdout of subprocess is byte variable, convert into float then into␣
↪integer
        return int(float(ffprobe_cmd.stdout.decode()))

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

go = VlcPlayer()
go.run(None)
```

20.6.3 Beispieldateien

- Video (13,7 MB)
- Image (553 kB)

KAPITEL 21

Stacks und Notebooks

Inhalte organisiert anzeigen

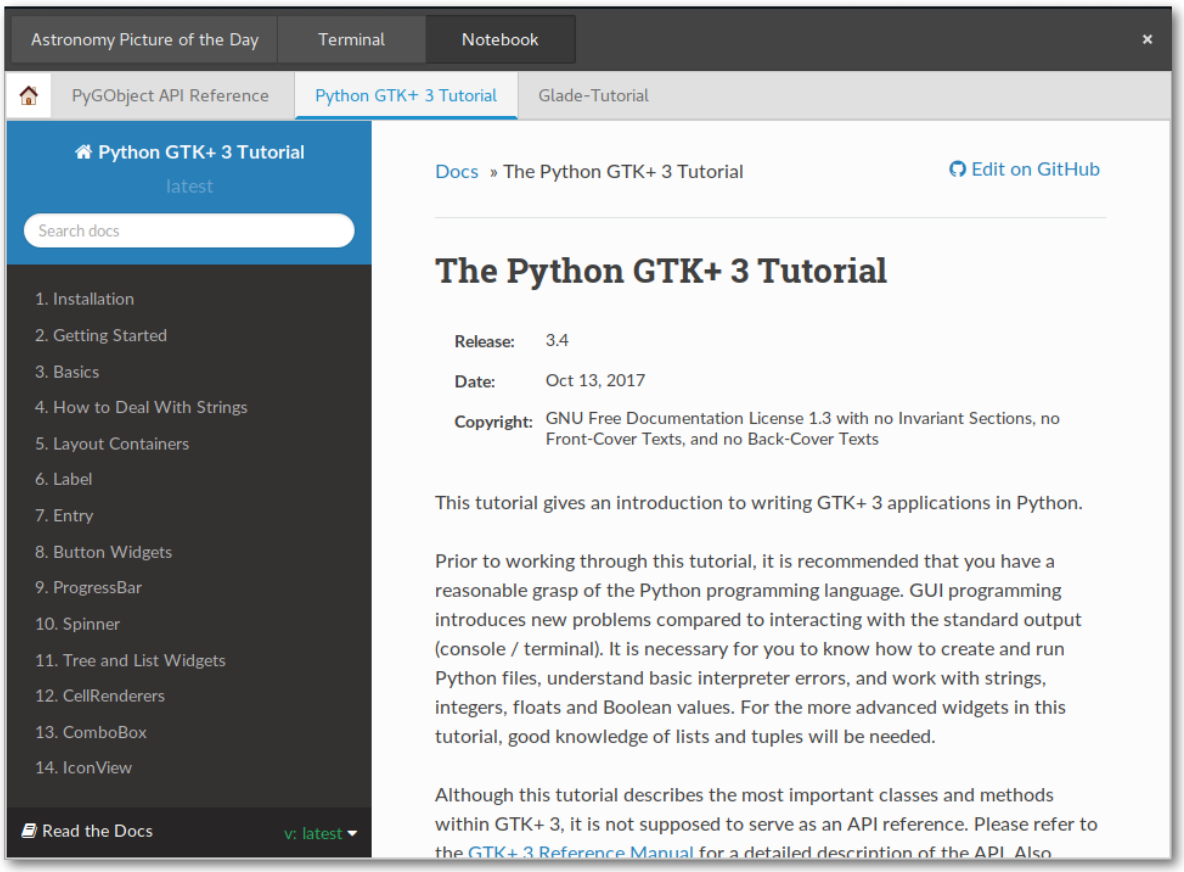
Gtk.Stack und *Gtk.Notebook* sind Layout-Container, die ihrerseits beliebige Widgets enthalten können.

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.



21.1 Glade

21.1.1 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.

21.1.2 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 (:ref: uberlistet (*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.

21.2 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.

21.2.1 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()
```

21.3 Listings

21.3.1 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)
```

21.3.2 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>
```