Practical GNOME Programming with Ruby

FOSDEM 2004 Tutorial

Laurent Sansonetti - `lrz@gnome.org`

Nikolai Weibull - `pcp@pcppopper.org`

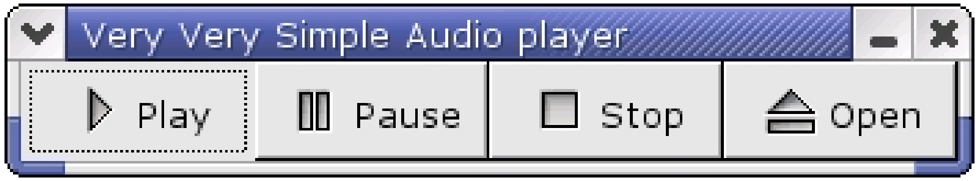
`http://ruby-gnome2.sourceforge.jp`

Goal

Our goal is to demonstrate that Ruby is

- Useful to glue complex components together
- Mature enough for real GNOME development
- Sexy

by developing a very, very simple audio player:



TOC

Contents:

TOC

Contents:

The Ruby-GNOME2 Project

- Since 2002
- Current release: 0.8.1
- Has 16 developers worldwide
- Supports 17 GNOME libraries
- Resources in English, and i18n *in progress* (French, Italian, and Japanese)

The Ruby-GNOME2 Project

Supported libraries:

Name	Implementation Status	Tutorials?	API Reference?	Examples?
ATK	***	No	No	Yes
GConf	***	Yes	Yes	Yes
GDK	***	No	Yes	Yes
GLib	***	No	No	Yes
GStreamer	***	No	Yes	Yes
GTK	***	Yes	Yes	Yes
GnomeCanvas	***	No	No	Yes
GtkGLExt	***	No	Yes	Yes
Libgda	***	Yes	Yes	Yes
Libglade	***	No	No	Yes
Libgnome	***	No	No	Yes
GdkPixbuf	**	No	No	Yes
GnomeVFS	**	No	No	Yes
Libart	**	No	No	Yes
Pango	**	No	No	Yes
GtkHtml	*	No	No	Yes
GtkSourceView	*	No	No	Yes

The Ruby-GNOME2 Project

RBBR (RuBy BRowser):

- Inspects classes/modules:
  - Native stuff: methods, constants, ancestors, . . .
  - GLib stuff: signals, properties, enum/flags, . . .
- Displays API reference (if exists)
- Shows GNOME stock items/icons
- Usable (HIG compliant)
- i18n (Belarusian, English, French, Japanese)

TOC

Contents:

User Interface Design

Glade is a GTK+/GNOME user interface builder.
Glade can:

- generate source code (C/C++/Perl/.../Ruby) for the user interface: the bad
- generate an XML file (`.glade`) loadable by Libglade: the good
- crash: the ugly

User Interface Design

Ruby/Libglade in action:

```
require 'libglade2'

engine = GladeXML.new('my_project.glade') do |handler_name|
    # connect the signal handler somewhere
    # we need to return a method reference
    lambda{ puts handler_name + " was called!" }
end

window = engine['my_window'] # window is a Gtk::Window
window.title = "My Application"

button = engine['my_button'] # button is a Gtk::Button
button.text = "My Button"
button.sensitive = false
:
:
```

User Interface Design

Let’s play a bit with Glade!

TOC

Contents:

Multimedia Design

GStreamer is

- A multimedia framework
- A set of components (plugins) for
  - input
  - codecs (audio, video, . . .)
  - filters / processors
  - output
- Network transparent
- Extensible
- Fast
- Still under heavy development

Multimedia Design

GStreamer deals with pipelines.

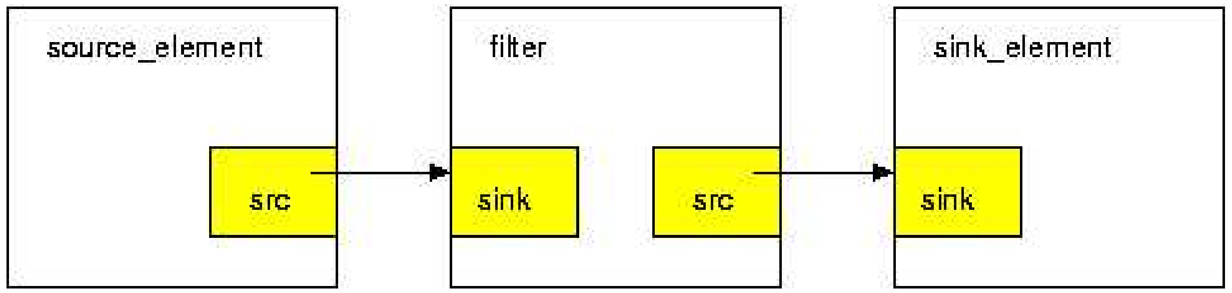
- *Pipelines* are sets of elements
- *Elements* are sets of pads
- *Pads* are connection points between elements

Elements are generally one of the following:

- A source - provides data
- A filter - tranforms data
- A sink - consumes data

Multimedia Design

GStreamer pipeline overview:



Multimedia Design

Pipelines are what we get after connecting elements in a sequence, much like connecting commands in a shell. We will discuss three pipelines:

- `filesrc ! mad ! osssink`
- `filesrc ! spider ! osssink`
- `gnomevfssrc ! spider ! osssink`

Multimedia Design

The `filesrc ! mad ! osssink` pipeline consists of:

- `filesrc` - provides data from a file on disk
- `mad` - decodes MPEG audio using libmad
- `osssink` - sends audio to soundcard using OSS

Multimedia Design

Positive

- Simple, works very well

Negative

- Can only decode files on disk
- Can only decode MPEG audio

Multimedia Design

We can do better!
The `filesrc ! spider ! osssink` pipeline consists of:

- `filesrc` - provides data from a file on disk
- `spider` - finds type of data and creates proper decoder
- `osssink` - sends audio to soundcard using OSS

Multimedia Design

Positive

- Works for any kind of media type with audio data
- Works well in most cases
- Simple to set up

Negative

- Can only decode files on disk
- Can fail to find type
- Can fail to find decoder
- Slower than using proper decoder immediately

Multimedia Design

We can still do better!
The `gnomevfssrc ! spider ! osssink` pipeline consists of:

- `gnomevfssrc` - provides data from an URL using GnomeVFS
- `spider` - finds type of data and creates proper decoder
- `osssink` - sends audio to soundcard using OSS

Multimedia Design

Positive

- Works for anything that can be accessed through an URL
- Works for any kind of media type with audio data
- Works well in most cases
- Simple to set up

Negative

- Adds GnomeVFS dependency

TOC

Contents:

User Interface Design

Let's write some code!

TOC

Contents:

Questions

Ask away!