JaMp - Un cliente para Jamendo

Simón Pena y Alumnos del Máster en Software Libre

Máster en Software Libre, 2009-2010. Edición de A Coruña
Igalia - Universidad Rey Juan Carlos

GUADEC-ES – 23 de julio de 2010

Universidad Rey Juan Carlos

igalia

Licencia

Contenido

Contenido

¿Qué es JaMp?

JaMp

- Ejercicio práctico en el módulo de desarrollo del Máster Software Libre
- Cliente para Jamendo para la plataforma GNOME
- Git como sistema de control de versiones
- Envío e integración de parches

¿Qué es Jamendo?

- Comunidad de música libre, legal e ilimitada, publicada bajo licencias Creative Commons
- 36299 álbumes publicados
- 226880 Reseñas de álbumes
- 809051 miembros activos

¿Por qué JaMp?

Intereses didácticos

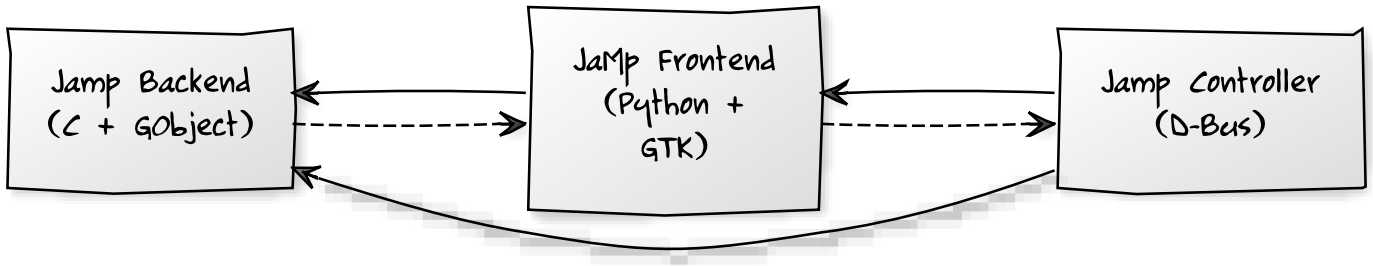
- Punto de entrada a la plataforma
- Familiarizarse con tecnologías habituales
  - Acceso y manipulación de contenidos web
  - Reproducción de los contenidos multimedia

Intereses prácticos

- Cliente dedicado (frente a usar plugins)
- Arquitectura solida y modular

Contenido

Arquitectura utilizada



Patrón arquitectónico Model-View-Controller

- Un *backend* para la recuperación de información de Jamendo y la reproducción multimedia
- Un *frontend* para la presentación de la información obtenida y la gestión del la reproducción
- Interconexión mediante D-Bus

Backend

Desarrollado en C con GObject

- Un proveedor de datos
  - Accede a Jamendo usando libsoup
  - Manipula los datos recibidos usando libxml
  - Expone un método *Query*
- Un gestor multimedia
  - Permite la reproducción de los contenidos usando gstreamer
  - Expone métodos *Play*, *Pause*, notificación de progreso...

Frontend

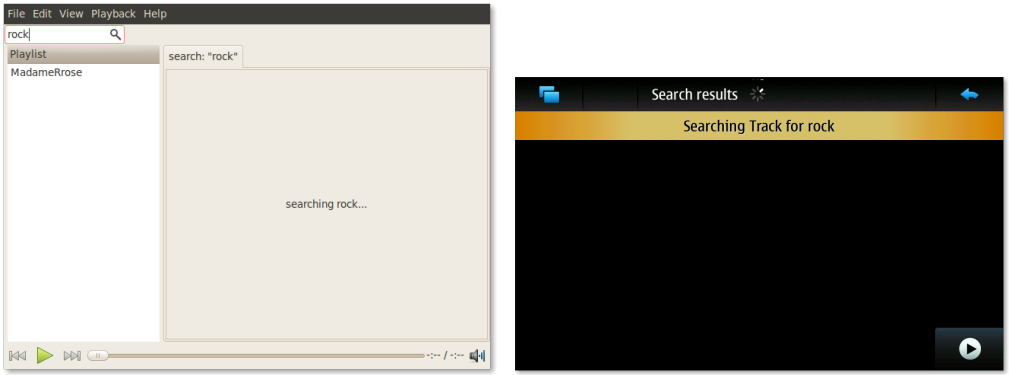


Figura: Clientes buscando “rock”

Desarrollado en Python

- Un cliente para GNOME, desarrollado con Glade
- Un cliente para Maemo, usando Hildon

Conexión mediante D-Bus

- Se usa D-Bus GLib en el lado del backend
- Todas las llamadas son asíncronas
- El modelo está totalmente desacoplado de la vista
- El controlador se puede reaprovechar entre distintas vistas

Contenido

Manejando señales (I)

Declarando la señal

Se instala en la inicialización de la clase (*pref_foo_class_init*)

```
jmp_mplayer_signals[ERROR] = g_signal_new ("error",
                                           G_TYPE_FROM_CLASS (klass),
                                           G_SIGNAL_RUN_LAST,
                                           0,
                                           NULL,
                                           NULL,
                                           g_cclosure_marshal_VOID__STRING,
                                           G_TYPE_NONE,
                                           1,
                                           G_TYPE_STRING);
```

Manejando señales (y II)

Emisión

Se envía la señal de acuerdo a la lógica de la aplicación (aquí, al recibir un error en *gstreamer*)

```
g_signal_emit (self, jmp_mplayer_signals[ERROR], 0, error_message);
```

Conexión

Nos conectamos a la señal en el cliente que use el componente (que podrá decidir qué comportamiento aplicar en función de los valores recibidos)

```
g_signal_connect (jmplayer, "error", G_CALLBACK (error_callback), NULL);
```

Marshallers y GClosures (I)

- Los *GClosure*s permiten representar funciones de callback.
- Existe un número de *GClosure*s predefinidos
  - g_cclosure_marshal_VOID__VOID
  - g_cclosure_marshal_VOID__BOOLEAN
  - ...
- Es habitual querer callbacks no predefinidos ¿Cómo?
- Usando *glib-genmarshal*

Marshallers y GClosures (I)

- Los *GClosure*s permiten representar funciones de callback.
- Existe un número de *GClosure*s predefinidos
  - g_cclosure_marshal_VOID__VOID
  - g_cclosure_marshal_VOID__BOOLEAN
  - ...
- Es habitual querer callbacks no predefinidos ¿Cómo?
- Usando *glib-genmarshal*

Marshallers y GClosures (I)

- Los *GClosure*s permiten representar funciones de callback.
- Existe un número de *GClosure*s predefinidos
  - g_cclosure_marshal_VOID__VOID
  - g_cclosure_marshal_VOID__BOOLEAN
  - ...
- Es habitual querer callbacks no predefinidos ¿Cómo?
- Usando *glib-genmarshal*

Marshallers y GClosures (II)

glib-genmarshal

- Permite crear GClosures personalizados
- Recibe como entrada una lista
  $RETURN\_VALUE : ARG_1, ARG_i, ARG_N$

Fichero marshal.list

```
VOID:INT64,INT64
VOID:STRING,POINTER
VOID:STRING,UINT,POINTER
```

Marshallers y GClosures (y III)

Soporte en autotools

```
jmp-marshal.h: marshal.list
glib-genmarshal --header --prefix=jmp_marshal marshal.list > $@

jmp-marshal.c: marshal.list
glib-genmarshal --body --prefix=jmp_marshal marshal.list > $@
```

Usándolo en una señal

```
..., jmp_marshal_VOID__INT64_INT64, G_TYPE_NONE,
     2, G_TYPE_INT64, G_TYPE_INT64, NULL);

static void
tick_cb (JmpMplayer *jmplayer, gint64 position, gint64 duration,
                gpointer user_data)
```

Contenido

Accediendo a Jamendo (I)

Usando libsoup

Encolando un mensaje

Encolamos una petición para comunicarnos de forma asíncrona

```
...
soup_session_queue_message (session, msg, process_response, cbdata);
...
```

Recibiendo la respuesta

Una vez recibida la respuesta, se gestionará aquí

```
static void
process_response (SoupSession *session, SoupMessage *msg,
    gpointer user_data);
```

Accediendo a Jamendo (II)

Analizando la respuesta: usando libxml2

```
static GList*
parse_xml (const char *buffer, int length, JmpRelation relation)
{
        xmlDocPtr doc = xmlReadMemory (buffer, length, NULL, NULL,
                                       XML_PARSE_NOBLANKS | XML_PARSE_RECOVER);
        if (!doc) return NULL;

        GList *result = NULL;

        xmlXPathContextPtr context = xmlXPathNewContext (doc);

        /* xpath is of the form "/data/album" */
        xmlXPathObjectPtr xpath_obj =
                xmlXPathEvalExpression (vos[relation].xpath, context);
        if (!xpath_obj) goto bail;
        ...
```

Accediendo a Jamendo (y III)

Analizando la respuesta: usando libxml2

```
xmlNodeSetPtr nodeset = xpath_obj->nodesetval;
if (nodeset->nodeNr > 0)
        result = generate_list (nodeset, relation);

/* generate_list */
for (i = 0; i < nodes->nodeNr; i++) {
        xmlNodePtr node = nodes->nodeTab[i];
        if (node->type == XML_ELEMENT_NODE) {
                GObject *item = generate_vo (node->children, relation);
                if (item)
                        list = g_list_prepend (list, item);
        }
}

/* generate_vo */
for (cur_node = a_node; cur_node; cur_node = cur_node->next) {
         if (cur_node->type == XML_ELEMENT_NODE) {
                 char *value = xmlNodeGetContent(cur_node);
                 item_set (item, cur_node->name, value, relation);
```

Contenido

Utilizando D-Bus

Pasos

- Crear una interfaz XML exponiendo los métodos
- Crear un servicio que implemente la interfaz, empleando D-Bus GLib
- Integrar en autotools
- Acceder desde Python

Utilizando D-Bus (II)

Definiendo una interfaz

```
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE node PUBLIC
  "-//freedesktop//DTD D-Bus Object Introspection 1.0//EN"
  "http://standards.freedesktop.org/dbus/1.0/introspect.dtd">
<node name="/">
  <interface name="org.mswl.jamp">
    <method name="Play">
    </method>
    <method name="GetVolume">
      <arg type="d" name="volume" direction="out" />
    </method>
    <method name="Seek">
      <arg type="x" name="seek_position" direction="in" />
    </method>
    <signal name="tick">
      <arg type="x" name="position" />
      <arg type="x" name="duration" />
    </signal>
  </interface>
</node>
```

Utilizando D-Bus (III)

Implementando la interfaz

```
#include <dbus/dbus-glib.h>
...
connection = dbus_g_bus_get (DBUS_BUS_SESSION, &error);
...
#include <dbus/dbus-glib-bindings.h>
...
gboolean
jmp_mplayer_service_seek (JmpMplayerService *self, gint64 seek_position,
                          GError **error)
{
        return jmp_mplayer_seek (self->priv->player, seek_position);
}
...
gboolean
jmp_mplayer_service_get_volume (JmpMplayerService *self, double *volume,
                                GError **error)
{
        *volume = jmp_mplayer_get_volume (self->priv->player);
        return TRUE;
}
```

Utilizando D-Bus (III)

Configurando D-Bus

```
...
DBusGProxy *proxy = dbus_g_proxy_new_for_name (self->priv->connection,
                                               DBUS_SERVICE_DBUS,
                                               DBUS_PATH_DBUS,
                                               DBUS_INTERFACE_DBUS);
...
org_freedesktop_DBus_request_name (proxy,
                                   MPLAYER_SERVICE_NAME,
                                   0, &request_name_result,
                                   &error));
...
dbus_g_connection_register_g_object (self->priv->connection,
                                     MPLAYER_SERVICE_OBJECT_PATH,
                                     G_OBJECT (self));
```

Utilizando D-Bus (y IV)

Conectando desde Python

```
bus = dbus.SessionBus()

class PlaybackManager:
    def __init__(self, tick_cb=None):
        self._bus = bus
        self._tick_cb = tick_cb
        player_object = self._bus.get_object("org.mswl.jamp",
                                             "/Player")
        self._player_interface = dbus.Interface(player_object,
                                            dbus_interface="org.mswl.jamp")
        if self._tick_cb:
            self._player_interface.connect_to_signal("tick",
                                                     self._tick_cb)
    def seek(self, value):
        self._player_interface.Seek(value)
...
def tick_signal_handler(self, position, duration):
    ...
```

Contenido

Port para Maemo

¿Qué es Maemo?

- Plataforma que corre en dispositivos móviles de Nokia
  - Maemo 3: Internet Tablet 770
  - Maemo 4: Internet Tablets N800 y N810
  - Maemo 5: N900
- Alrededor de los dispositivos se forma la comunidad Maemo

Port para Maemo

¿En qué fijarse?

Se utiliza Hildon

```
hildon.StackableWindow
hildon.GtkButton(gtk.HILDON_SIZE_AUTO)
hildon.PickerButton(gtk.HILDON_SIZE_FINGER_HEIGHT,
                    hildon.BUTTON_ARRANGEMENT_HORIZONTAL)
```

La aplicación no debe parecer bloqueada

```
hildon.hildon_gtk_window_set_progress_indicator(self, True)
banner = hildon.hildon_banner_show_information(self, "", message)
banner.set_timeout(milliseconds)
```

Port para Maemo

¿En qué fijarse?

Se utiliza Hildon

```
hildon.StackableWindow
hildon.GtkButton(gtk.HILDON_SIZE_AUTO)
hildon.PickerButton(gtk.HILDON_SIZE_FINGER_HEIGHT,
                    hildon.BUTTON_ARRANGEMENT_HORIZONTAL)
```

La aplicación no debe parecer bloqueada

```
hildon.hildon_gtk_window_set_progress_indicator(self, True)
banner = hildon.hildon_banner_show_information(self, "", message)
banner.set_timeout(milliseconds)
```

Port para Maemo

Claridad y sencillez, “finger friendly”

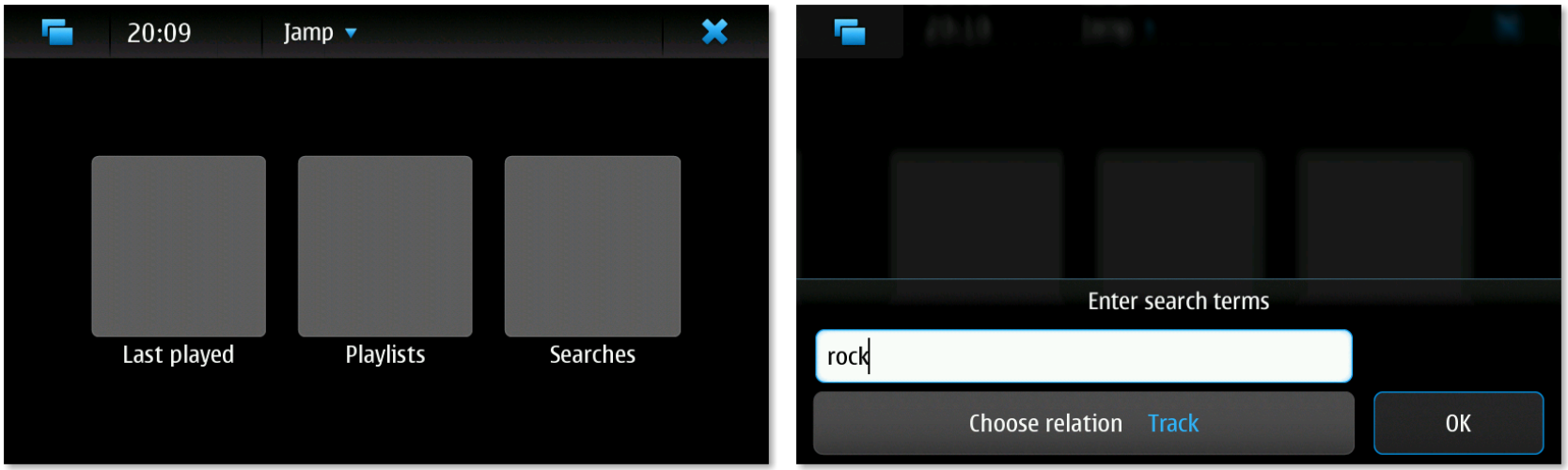


Figura: Pantalla de bienvenida y búsqueda

Contenido

Objetivos futuros (I)

Distribución

- Empaquetar JaMp
- Crear un repositorio *PPA* para Ubuntu

Interacción con Jamendo

- Aumentar las APIs implementadas
- Emplear Grilo para el acceso a los datos

Objetivos futuros (y II)

Ports

- Completar port para Maemo
- Emplear MAFW en Maemo
- Llevar a otras plataformas

Otras posibilidades

- Emplear otros toolkits gráficos

Contenido

¡Únete a nosotros!

¿Por qué?

- Te interesa Jamendo
- Te interesa conocer las tecnologías que usamos
- Te interesa llevar JaMp a una nueva plataforma
- Quieres corregir algún error que hayas visto hoy ;)

Descarga el código `http://gitorious.org/mswl2010/jamp`

Ponte en contacto con nosotros `jamp-devel@googlegroups.com`

¡¡Gracias!!

Referencias

Señales	http://library.gnome.org/devel/gobject/stable/signal.html
D-Bus	http://www.freedesktop.org/wiki/Software/dbus
Hildon	http://wiki.maemo.org/Hildon
PyMaemo	http://pymaemo.garage.maemo.org/, http://wiki.maemo.org/PyMaemo/UI_tutorial