Open Object Rexx V4 APIs

Tutorial

W. David Ashley
Rexx Language Symposium
April 2008

Goals for the V4 APIs

- Use C++ as the normal access method
- Provide different contexts for routines, methods, threads and exits
- Reduce the coding effort to produce extension libraries
- Maintain and enhance platform independence
- Make ooRexx objects directly available to the C++ programmer

Getting Started

- #include <oorexxapi.h>
- Source files should be C++. This may mean that they have a .cpp extension on some platforms
- There are new prototypes for routines(functions). There are similar prototypes for Methods
- Argument conversion and testing is all automatic

The oorexxapi.h File

The oorexxapi.h file includes a set of platform independent and platform dependent header files. These will all be installed automatically for your platform.

Contexts

- Routines, methods, threads and exits all have a context associated with them.
- It is provided to the programmer automatically through the new prototypes.
- The variable <u>*context*</u> is automatically set and made available to the programmer inside a routine, method, etc.
- The <u>*context*</u> variable is a pointer to a host of C++ methods that can provide just about any service you might need.

Example ooRexx Method in C++

```
RexxMethod2(int,                            // Return type
            GrxColorButtonNew,              // Object_method name
            OSELF, self,                    // Self
            OPTIONAL_CSTRING, colorstr)// Color string
{
    GtkWidget *myWidget;
    GdkColor color;

    if (colorstr = NULL) {
        myWidget = gtk_color_button_new();
    }
    else {
        gdk_color_parse(colorstr, &color);
        myWidget = gtk_color_button_new_with_color(&color);
    }
    context->SetObjectVariable("!POINTER", context->NewPointer(myWidget));
    g_object_set_data(G_OBJECT(myWidget), "OORXOBJECT", self);

    return 0;
}
```

Declaring the Previous Method

```
::class GtkColorButton public subclass GtkButton

::METHOD init EXTERNAL "LIBRARY rexxgtk GrxColorButtonNew"
::METHOD 'title=' EXTERNAL "LIBRARY rexxgtk GrxColorButtonSetTitle"
::METHOD set_title EXTERNAL "LIBRARY rexxgtk GrxColorButtonSetTitle"
::METHOD color EXTERNAL "LIBRARY rexxgtk GrxColorButtonGetColor"
::METHOD get_color EXTERNAL "LIBRARY rexxgtk GrxColorButtonGetColor"
::METHOD set_color EXTERNAL "LIBRARY rexxgtk GrxColorButtonSetColor"
::METHOD 'color=' EXTERNAL "LIBRARY rexxgtk GrxColorButtonSetColor"
::METHOD signal_connect EXTERNAL "LIBRARY rexxgtk
GrxColorButtonSetColor"

-- The following are the methods that can be connected to signals. By
default
-- they do nothing.

::method signal_color_set
return
```

Invoking the Previous Method

```
-- Create a color button with the default color
MyButton1 = .GtkColorButton~new()

-- Create a color button with a specific color
MyButton2 = .GtkColorButton~new('#313131')


::requires 'rexxgtk.cls'
```

Connecting Signals (callbacks)

```
RexxMethod1(RexxObjectPtr,               // Return type
            GrxButtonSignalConnect,      // Object_method name
            CSTRING, name)               // Signal name
{
    RexxPointerObject rxptr = (RexxPointerObject)context-
>GetObjectVariable("!POINTER");
    GtkWidget *myWidget = (GtkWidget *)context->PointerValue(rxptr);
    cbcb *cblock;

    if (strcmp(name, "clicked") == 0) {
        cblock = (cbcb *)malloc(sizeof(cbcb));
        cblock->instance = context->threadContext->instance;
        cblock->signal_name = "signal_clicked";
        g_signal_connect(G_OBJECT(myWidget), "clicked",
                         G_CALLBACK(signal_func_0), cblock);
        return context->True();
    }

    RexxObjectPtr parent = context->GetSuper();
    return context->SendMessage0(parent, name);
}
```

Executing Signals (callbacks)

```
static void signal_func_0(GtkWidget *widget,
                          gpointer data)
{
    cbcb *cblock = (cbcb *)data;
    RexxObjectPtr rxobj = (RexxObjectPtr)g_object_get_data(G_OBJECT(widget),
"OORXOBJECT");
    RexxThreadContext *context;

    cblock->instance->AttachThread(&context);
    context->SendMessage0(rxobj, ((cbcb *)data)->signal_name);
    context->DetachThread();
    return;
}
```