Database register CLASS REGISTRATION

THORSTEN FISCHER

Which Windows user does not know the beloved registry, in which the system and diverse programs store configuration data – or again, maybe not, or only partly. In any case, this application is very easy to use with a mouse. It can be made even simpler.

As always, everything – and this is a promise – will get better and nicer. This time, what's involved is the possibility of storing, managing and querying configuration data for programs at a single, central location – at the start of a program for instance. Gnome developers have begun to bring a mechanism for this to the start line, which they have named GConf.

GConf is to make a proper entrance in Version 2.0 of Gnome, but is already available in a packet in the unstable tree of the source code. A few new programs (Nautilus and Evolution among them) already demand GConf for their compilation. The library will not remain limited to Gnome however, but will ultimately also be able to be used for pure Gtk+ and X-applications, KDE programs and even from the command line.

According to Havoc Pennington, Gnome is at the stage of Windows 3.x in terms of the configuration of applications, because each program stores its own settings in separate files. The registration database under Win9x may be a shrewd step, but it does have diverse restrictions:

According to the article, GConf wants to use a two-rail system to provide both a library and an architecture, which is to be designed as follows.

The concept

In theory GConf ought to offer a whole heap of advantages, once it is finished and refined. And so it does: We are talking about free software in the initial stages with considerable claims to what will eventually come about.

This begins with the independence of the back end, in which the actual data is to be stored. The current implementation stores everything in XML files, which are best suited to hierarchical organisations, if only files are involved.

But a binary format should also be possible as in the registry, and anyone who wants to get really stuck in can also control access via LDAP or even any SQL database of their choice. The idea behind this is that the administrator of a network can simply seek out whatever suits their needs in terms of infrastructure.

Data is stored in key/value pairs, similarly to the method used in the registry. The individual keys should each be given their own field for documentation, which is then shown by means of an application with which the values can be changed. In addition to this, other information should be stored, such as when the key was last altered. It is not apparent, from the documentation so far available, whether there might be a mechanism provided which could prevent or privilege the editing of certain keys.

There is also going to be a data notification service. In an age of components, when every

Model – View – Controller (MVC)

This three-stage concept consists of a model, one or more views keyed to this model, and finally one or more controllers. These components perform the following tasks:

- *In one model, the structure of the data is conceived. For GConf that means, firstly, designing the entire database, and secondly, giving some detailed thought as to the keys which are to be included.*
- *A view is a specific way of looking at data. This view can correspond to the model, in that all data is simply made accessible, or (and this would usually be the case) a certain section is shown. Whenever the model changes, the views also have to be adapted.*
- *Finally, data can be changed from outside using a controller.*

program can communicate with every other program, it makes sense to propagate changes in the network so that all applications for which it is desirable are reconfigured accordingly.

This concept is also of interest because this method can be used to reconfigure running programs. For example, with a setting which stipulates that all toolbars should no longer display icons, and ensures that this change immediately takes effect in all programs. I do not know, however, whether I am the only person for whom this sounds like an extremely interesting way to upset a whole network of computers.

Each key has, as under Windows, an unequivocal name. To show the hierarchy, a system is used which is reminiscent of the file system under Unix: a key could for example be called */programs/editor/window/font*. The values are tagged, so are defined in advance, such as for example string, integer or by another data type.

More than just a daemon

To be able to use GConf, a daemon named Gconfd must be running. There should always be exactly one daemon per user, which can communicate via CORBA with applications.

A high level of abstraction, which comes into play, for example, with some widgets in the Gnome Application Library, bears the name **MVC**. Its concept is explained in more detail in the Model – View – Controller boxout.

There are several different types of data with which one can fiddle around in Gconf: GConfEngine, is used to address a configuration database, in most cases this involves connection to a running Gconfd daemon. Then there are GConfValue, a structure which encases the tagged values from a key/value pair, and GConfClient, a client which can communicate with a GConfEngine, plus various tasty titbits such as caching and finally GConfError, a structure for handling errors.

Listing 1: The client

```
1:
2: #include <gconf/gconf-client.h>
3: #include <gtk/gtk.h>
4:
5: void callback (GtkWidget* entry, gpointer user data)
6: {
7:   GConfClient *gclient;
8:   gchar      *str;
9:
10:   client = GCONF_CLIENT (user_data);
11:
12:   str = gtk_editable_get_chars (GTK_EDITABLE (entry), 0, -1);
13:
14:   gconf_client_set_string (client, "/extra/uk/thefrog/linuxmagazine/gconf", str, NULL);
15:
16:   g_free(str);
17: }
18:
19: int main(int argc, char *argv [])
20: {
21:   GtkWidget   *window;
22:   GtkWidget   *entry;
23:   GConfClient *gclient;
24:
25:   gtk_init (&argc, &argv);
26:   gconf_init (argc, argv, NULL);
27:
28:   window = gtk_window_new (GTK_WINDOW_TOPLEVEL);
29:   entry = gtk_entry_new ();
30:
31:  gtk_container_add (GTK_CONTAINER (window), entry);
32:
33:  gclient = gconf_client_new ();
34:
35:  gconf_client_add_dir (gclient, "/extra/uk/thefrog/linuxmagazine/gconf", GCONF_CLIENT_PR
ELOAD_NONE, NULL);
36:
37:  gtk_signal_connect (GTK_OBJECT (entry), "activate", GTK_SIGNAL_FUNC (cal
lback), gclient);
38:
39:  gtk_widget_show_all (window);
40:
41:  gtk_main();
42:
43:  return 0;
44: }
```

Example

Now it's time for a little example. The two listings, 1 and 2, contain a client and a controller demonstrating the function of GConf. GConf does have to be installed for this, either by doing it explicitly as described briefly in the following Installation boxout, or by having installed the Ximian desktop.

GConf has to be initialised like Gtk+. This then creates a new client, which is transferred via the connection with a signal to the callback function. If one now types some text into the input box and finishes with the Return key, the content of the box is read out

Listing 2: The controller

```
1: #include <gconf/gconf-client.h>
2: #include <gtk/gtk.h>
3:
4: void callback (GConfClient* client, guint cnxn id, const gchar* key, GConfValue* value, gbo?
olean is default, gpointer user data)
5: {
6:   GtkWidget *label;
7:
8:   label = GTK WIDGET (user data);
9:
10:   if (value == NULL)
11:   {
12:     gtk label set(GTK LABEL(label), " leer ");
13:   } else {
14:     if (value->type == GCONF VALUE STRING)
15:     {
16:       gtk label set (GTK LABEL (label), gconf value string(value));
17:     } else {
18:       gtk label set (GTK LABEL (label), " wrong type! ");
19:     }
20:   }
21: }
22:
23: int main(int argc, char *argv [])
24: {
25:   GtkWidget   *window;
26:   GtkWidget   *label;
27:   GConfClient *gclient;
28:   gchar       *str;
29:
30:   gtk init (&amp;argc, &amp;argv);
31:   gconf init (argc, argv, NULL);
32:
33:   gclient = gconf client new();
34:   window = gtk window new (GTK WINDOW TOPLEVEL);
35:
36:   str = gconf client get string (gclient, "/extra/uk/thefrog/linuxmagazine/gconf", NULL);
37:
38:   if (str)
39:   {
40:     label = gtk label new (" leer ");
41:   } else {
42:     label = gtk label new (str);
43:   }
44:
45:   if (str)
46:   {
47:     g free (str);
48:   }
49:
50:   gtk container add (GTK CONTAINER (window), label);
51:
52:   gconf client add dir (gclient, "/extra/uk/thefrog/linuxmagazine", GCONF CLIENT PRELOA?
D NONE, NULL);
53:   gconf client notify add (gclient, "/extra/uk/thefrog/linuxmagazine/gconf", callback, la?
bel, NULL, NULL);
54:
55:   gtk widget show all (window);
56:
57:   gtk main();
58:
59:   return 0;
60: }
```

The author

Thorsten Fischer is supposed to be studying IT and/or Media Consultancy. He also works as a developer at MapMedia.

and written into the callback as a value in the key.

As can be seen from the Include-lines, GConf is not actually Gnome-dependent. GConfClient still needs, at the moment, the link to Gtk+, but this defect will soon be consigned to oblivion.

In the second listing only any already existing key will be read out and shown in a label. At this point it is important to throw away the pointer *str* with *g_free()* again, as one gets a copy back. Once the label has been inserted into the window, the program is instructed to react to changes in the respective directory.

Then the callback function is invoked if the client in Listing 1 is used. The remaining lines in the callback essentially serve to test the read-out value for the correct type – a string. If the key contains no string as value, a corresponding pointer is inserted. If no value whatsoever has been set, a reference is also made to this.
The key used can be found under the extra hierarchy, which has been installed for additions by third parties. I think a further subdivision through domain names, could also be very sensible.

GConf in Gnome 2.0

It has already been mentioned that GConf will only be fully integrated into Gnome in Version 2.0. This also involves the installation of a few little things. Firstly, a global client for a logged-on user is going to run, which can be addressed via simple functions of the Gnome-API. Then there will be dialogs in the style of Gnome and diverse other functions, which are necessary in order to use Gconf with ease as a programmer.
Happy Gnoming! ■

Info

Feature article on GConf by Havoc Pennington:
http://developer.gnome.org/feature/current/index.html
GConf API Documentation:http://developer.gnome.org/doc/API/gconf/index.html
Gnome FTP: ftp://ftp.gnome.org/pub/GNOME/unstable/sources/GConf/
Ximian website:http://www.ximian.com

■

GConf installation

There are no great surprises in the installation: After obtaining the source code from the FTP site, all you need is the instruction:

```
tar xzvf GConf-0.x.x.tar.gz
cd GConf-0.x.x
./configure
make
make install
```

Here, of course, the final step must be performed by a privileged user, normally root. Please take note of the instructions that appear at the end of the execution of the installation script.