Geany Newsletter #5

Contents

1 About Geany

Geany is a small and lightweight Integrated Development Environment. It was developed to provide a small and fast IDE, which has only a few dependencies from other packages. Another goal was to be as independent as possible from a special Desktop Environment like KDE or GNOME - Geany only requires the GTK2 runtime libraries.

More information about Geany can be found at geany.org.

2 New translations and updates

Since our last newsletter a number of translations has been updated or newly added to Geany. New translations are:

- Arabian
- Indonesian
- Lithuanian
- Mongolian (back in 2011)

But also translations like German, Kazakh, Hungarian, Italian, Traditional Chinese and Swedish translations have been updated during the last roughly four month.

3 Wiki available

We set up a wiki for additional documentation and resources related to Geany at `http://wiki.geany.org`. Anyone can contribute to the wiki simply by registering and then logging in.

In the wiki you can find configuration snippets and tips, snippets for various programming languages and many additional tags files to enhance Geany's autocompletion features.

Everybody is welcome to add additional useful content to the wiki.

4 C++ plugins supported

Geany's public plugin API headers have been updated to support inclusion into C++ code. Most of the changes involve adding *extern "C" {...}* blocks around the public headers' code (by way of GLIB's *G_BEGIN_DECLS* and *G_END_DECLS* macros) to make them easier to include, so the C++ code doesn't need to do this.

You can now write plugins in C++ and they will be loadable by Geany at run-time. Of course using Geany's API will still involve using C in your code, but the rest of your plugin can use whatever C++ features you want. You can even use gtkmm [1] in your plugin if you want.

Any of the symbols Geany looks up at run-time must not have their names mangled by the compiler. To avoid this, put that code inside an *extern "C"* block.

Here's an example of Geany's Hello World plugin from the Plugin HowTo [2] ported to C++:

```
#include <geanyplugin.h>

class HelloWorld
{
	private:
		gchar *hello_message;
		GtkWidget *main_menu_item;

	public:
		HelloWorld(const gchar *message);
		~HelloWorld();
		void SayHelloWorld();
};

static HelloWorld *hello;

extern "C"
{
	GeanyPlugin     *geany_plugin;
	GeanyData       *geany_data;
	GeanyFunctions  *geany_functions;

	PLUGIN_VERSION_CHECK(211)
	PLUGIN_SET_INFO("HelloWorld C++",
	                                "Just another tool to say hello world, this
	                                "1.0", "John Doe <john.doe@example.org>");

	void plugin_init(GeanyData *data)
```

```
        {
                hello = new HelloWorld("Hello C++ World");
        }

        void plugin_cleanup(void)
        {
                delete hello;
        }

        static void on_menu_item_clicked(GtkMenuItem *item, gpointer user_data)
        {
                hello->SayHelloWorld();
        }
}

HelloWorld::HelloWorld(const gchar *message)
{
        hello_message = g_strdup(message);
        main_menu_item = gtk_menu_item_new_with_mnemonic("Hello World");
        gtk_widget_show(main_menu_item);
        gtk_container_add(GTK_CONTAINER(geany->main_widgets->tools_menu), main_menu_
        g_signal_connect(main_menu_item, "activate", G_CALLBACK(on_menu_item_clicked
}

HelloWorld::~HelloWorld()
{
        g_free(hello_message);
        gtk_widget_destroy(main_menu_item);
}

void HelloWorld::SayHelloWorld()
{
        dialogs_show_msgbox(GTK_MESSAGE_INFO, "%s", hello_message);
}
```

It's important to note that the dynamic library loading mechanism that loads plugins is C functionality and does not know about C++ constructors. This means that global and static objects in the plugin will *not* have their constructors called when the plugin is loaded. Use dynamically created objects as show in the above example.

These changes will be available in the next Geany release but you can start using them right away in your C++ plugins if you Build Geany From Git [3].

1. http://developer.gnome.org/gtkmm-tutorial/2.24/sec-basics-gobj-and-wrap.

html.en

2. http://www.geany.org/manual/reference/howto.html
3. http://www.geany.org/Download/Git

5 Plugins

Notes from the plugin section.

5.1 New Plugins

5.1.1 GeanyPyflakes

Pyflakes is a command line tool that statically analyzes python program and detects two kinds of errors: unused imports and undefined symbols. geany-pyflakes runs pyflakes in the background and parses its output. Afterwards puts markers on lines with errors and adds the output to the panel at the bottom of editor (the one with console, todo, etc.). Geany-pyflakes is available at its project pages at http://code.google.com/p/geany-pyflakes/

Another way to check your Python code is described inside the wiki at http://wiki.geany.org/howtos/check_python_code

5.1.2 GeniusPaste

GeniusPaste is a plugin which is adding the possibility to paste your code from Geany into different pastebins. It supports this services:

- codepad.org
- pastebin.com
- pastebin.geany.org
- dpaste.de
- sprunge.us

During the paste process GeniusPaste detects automatically the syntax of your code and paste it with syntax highlighting enabled. Once this is done it is also able to redirect you to the pasted code opening a new browser tab.

5.2 GeanyPG

GeanyPG is a plugin that allows the user to encrypt, decrypt text and verify signatures with GnuPG from inside Geany. It's created by Hans Alves and is part of the geany-plugins project.

After the plugin has been installed successfully, it can be loaded from inside Geany's plugin manager which will add a new menu item into the Tools menu offering functions of the plugin.

To decrypt or encrypt, just select the interesting parts and choose the function you wish -- If none text has been selected, the whole document will be processed. In case you like to verify a signature obviously you will have to select the whole block.

When encrypting a message you can choose to sign at the same time. If a passphrase is needed, the GPGME library will decide how the user is prompted. Usually this will use gpg-agent. If gpg-agent is disabled, pinentry with one of its frontends will be used.

6 Geany local

6.1 Geany at Chemnitzer Linuxtage 2012 (March 17th, 18th)

As last year, Geany had a booth a Chemnitzer Linuxtage 2012 in German city Chemnitz. Our booth was again located next to the guys of Xfce as well as next (that was different to last year) to 2 lecture rooms. Even though the event wasn't as much crowded as last year, a lot of people were passing by asking some question or just saying hello. So Enrico and Frank had a lot of questions to answer and a lot of feedback to respond to.

7 Geany Packages for Fedora

There are new packages unofficially available for Fedora. One is containing the Geany Themes Matthew maintains at GitHub [1], the other one provides the tags files listed in the Geany Wiki [2]. The packages are not yet in Fedoras official repositories but available at Dominic's Fedora People space [3]. Note the geany-themes package is intended to work with current Git versions of Geany only. A x86_64 package from the current Git master as well as an SRPM for rebuilding is also available at [3].

The geany-tags package is split into subpackages containing the tags for each programming language. Currently these are: geany-tags-c, geany-tags-php and geany-tags-python. They can be installed independently from each other, of course.

Contact Dominic if you have suggestions for improvements.

1. https://github.com/codebrainz/geany-themes

2. http://wiki.geany.org/tags/start
3. http://dmaphy.fedorapeople.org/

8 About this newsletter

This newsletter has been created in cooperation by people from Geany's international community. Contributors to this newsletter and the infrastructure behind it, ordered by alphabet:

- Dominic Hopf
- Enrico Tröger
- Frank Lanitz
- Lex Trotman
- Matthew Brush