Making metadata with Leaftag

A NEW LEAF

The experimental Leaftag provides a convenient means for associating files with search categories and other metadata.

BY OLIVER FROMMEL

Hierarchical directory systems are very useful for organizing files, but they can only help you to a certain point. If you have a collection of digital images, you may be happy with directories that reflect the content categories, such as landscapes, technology, or portraits. Subdirectories can give you more granular categories, such as family, friends, work, and so on, but if you have a large and complex collection, you still might not be able to find the image you are looking for.

Search techniques based on metadata provide a better solution. Users can associate keywords with files to speed up the search. Metadata gives a user the ability to create an organizational structure that does not rely on the filesystem.

One example of this metadata approach is the Gnome image management tool F-Spot, which uses a category system [1] based on tags (Figure 1). Besides the ability to sort images by timestamp, the program also supports category-based sorting. Unfortunately, these search features only work within the F-Spot tool, which runs its own database.

Of course, it would be more practical if you could search metadata in any program. This was the idea the developers had in mind when they implemented the Leaftag global metadata tagging system [2]. Leaftag lets you associate metadata *tags* with a file. The tag information is then maintained in a database that supports category-based searches. The Leaftag project leader is Christian Hammond, who works for Vmware; some of the Gnome-specific code in Leaftag comes from David Trowbridge.

Finicky Install

Leaftag is still at a relatively early stage of development, although the tagging feature works fine. The Leaftag distribution includes four components: the Libleaftag C library, the Tagutils, and the two Gnome-specific packages, Leaftag-Python and Leaftag-Gnome. There are no distribution-specific binary packages available at present, so you must be prepared to experiment and build Leaftag yourself. Leaftag does not ask for much in the line of current libraries; you don't need a CVS version or the like. If you intend to install Leaftag support for Gnome, you need at least Gnome 2.12 with the developer packages.

After downloading the four packages from [2], the first step is to compile and install Libleaftag, as the other packages all depend on the library. Building the library is very much like any other Autoconf-managed software. Start by running the *configure* script. The library stores category tags in an Sqlite database, so you will need the developer packages for Sqlite Version 2 – Sqlite 3 is not API-compatible.

If you run Configure without specifying any options, and then *make install* to distribute them on your disk, you might like to set the *PKG_CONFIG_PATH* environmental variable for the other packages to */usr/local/lib/pkgconfig*. If you don't, the packages will be unable to locate *leaftag.pc*, which the install script has copied to this path. Leaftag-Python also requires the Python-Gtk developer packages; these are *pygtk2-devel* for Fedora 4/5, and *python-gtk2-dev* for Ubuntu.

The most complex task is resolving the dependencies for Leaftag-Gnome. This component requires the Deskbar Applet [3], which is available for a few distributions but not typically as a recent version. Again this means downloading the sources and going through the build process, which in turn requires the *gnome-python2-applet* package for Fedora or *python-gnome2-dev* for Ubuntu.

After installing the Deskbar applet, you should be able to compile Leaftag-Gnome. A few more steps are required to integrate the Deskbar applet with the Gnome panel. If you do not specify a Configure prefix, the applet will end up in */usr/local* and will be invisible to the Bonobo server. You can either enter the path directly in the configuration file, */etc/bonobo-activation/bonobo-activation-config.xml*, or use a utility for the job:

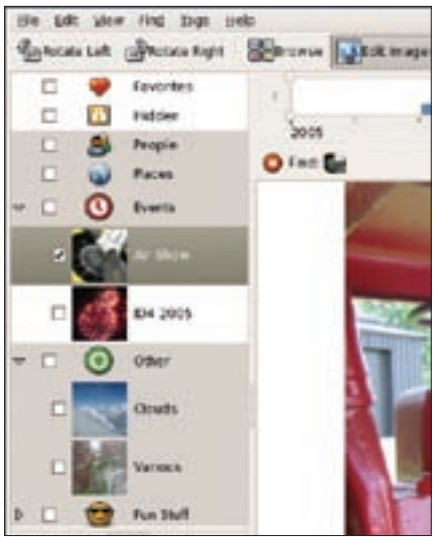

Figure 1: F-Spot uses tags to categorize photos; unfortunately, the tags are only available within the tool itself.

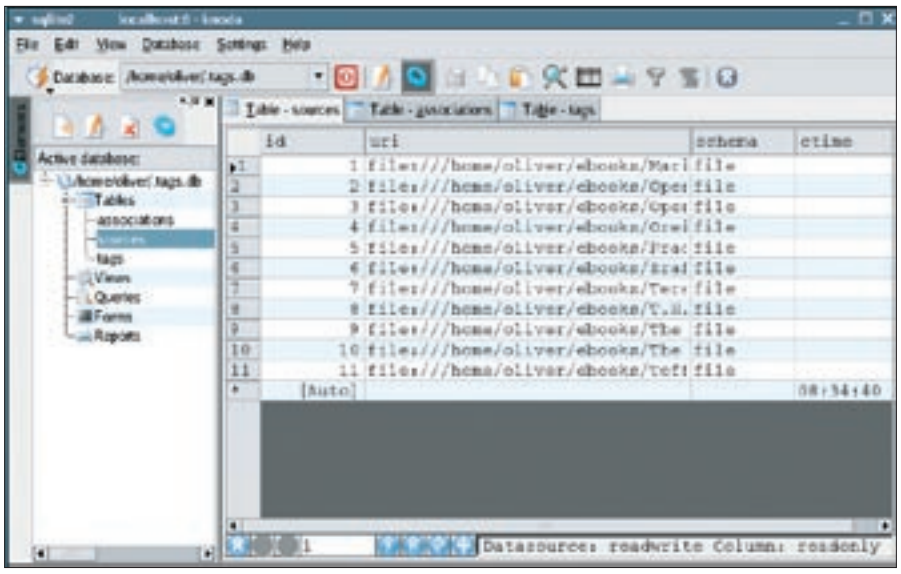

Figure 2: Leaftag is based on an Sqlite database. The data file is located in $HOME/.tags.db and contains three tables.

```
bonobo-activation-sysconf ➥
--add-directory=➥
/usr/local/lib/bonobo/servers
```

To make the Deskbar applet visible in the applet menu, you need to kill the necessary components and let them autostart:

```
killall ➥
bonobo-activation-server
killall gnome-panel
```

The final component is the set of command line tools, or Tagutils, which should not cause you any difficulty.

Tagging at the Command Line

The program that tags files with your own category tags is called *tagutils*. This program supports five commands that are appended to the command names: *ls*, *tag*, *tagprop*, *tags*, and *untag*. To create a new tag, just tag one or more files with it: *tagutils family image.jpg*. Instead of a local file, users can specify URLs and thus tag non-local resources.

To tag a whole directory, rather than a single file, *tagutils tag* has an *-r* option. A combination of *tagutils untag tag file* will remove a tag name.

Of course, you can assign more than one tag to a file. To do so, just run the Tagutils command with the required tags. You can create, read, modify, and delete tag properties using *tagutils tagprop*. The purpose of this tag properties feature is currently a little unclear, but properties such as *description*, *hidden*, and *image* are a fair indication of where this feature is heading. Programs that set tags can use properties to display descriptions and images for categories or to hide specific tags.

tagutils tags gives you a list of the tags you have used. There is no way of deleting tags at this time. A combination of *tagutils ls tag* outputs the files and URLs tagged with a specific tag.

Behind the Scenes

In the background, files, tags, and associations are managed using an Sqlite database (*$HOME/.tags.db*), which contains three tables (Figure 2).

The keys are not hashes or anything similar, but simple filenames. This approach is not exactly robust; if you change the filename or path, Tagutils will lose track of the file.

The *tagutils ls tag* command does not output error messages, probably to avoid nasty looking "not found" messages in GUI-based front-ends. After moving a tagged file, if you move it back to its original location, Tagutils will find it again – none of this has any effect on the database.

Many users have suggested the alternative approach of using the filesystem's extended attributes, but this would cause a number of issues. For one thing, tags would no longer be portable; that is, they would be lost if you moved files to another computer. For another thing, extended attributes are system global, whereas the Leaftag developers intended to implement user-specific tags right from the outset.

Another option would be to carry on down the Leaftag path, and either modify all file utilities, or the desktop or kernel VFS layers, to allow for database updates. This is the approach that Apple took with file resources on Mac OS X, although Apple does not use a central repository, opting instead for a file-oriented approach that uses hidden directories.

Despite its weaknesses, Leaftag has been welcomed by various Gnome projects. The current Leaftag architecture means that developers who want to use tags have to add support to their own code. David Trowbridge tried adding Leaftag support with Gimmie, a kind of personal information manager for Gnome. You can watch the demo clip at [4]. And the modern memo manager Tomboy at least has experimental Leaftag support[5]. ■

INFO

[1] F-Spot: *http://f-spot.org*

[2] Leaftag: *http://www.chipx86.com/wiki/Leaftag*

[3] Deskbar applet: *http://raphael.slinckx.net/deskbar*

[4] Leaftag and Gimmie: *http://david.navi.cx/blog/?p=77*

[5] Leaftag and Tomboy: *http://www.helsinki.fi/~pakaste/blog/tomboy_leaftag.html*