GNAT Tutorial

GNAT is Not A Tagger

GNAT creates RDF files to describe a local music collection so that you can link your local music collection into the semantic web.

Download

Download the motools package (which contains GNAT) on http://sourceforge.net/projects/motools

Installation

- You need to have Python 2.4 or 2.5 (http://python.org/download/)
- Get Easy Install

  http://peak.telecommunity.com/DevCenter/EasyInstall
- Use Easy Install to install rdflib:

```
$ easy_install -U 'rdflib<3a'
```

- Use Easy Install to install mutagen:

```
$ easy_install mutagen
```

- Use Easy Install to install musicBrainz2:

```
$ easy_install python-musicbrainz2
```

You can now run gnat. This is done via AudioCollection.py:

```
■ $ python AudioCollection.py [options] command filepath
```

Generate RDF from embedded metadata

The most basic use of GNAT involves looking up your local music's metadata from embedded tags. GNAT matches this metadata to the MusicBrainz database and generates RDF containing a URI to the database entry. The following will generate one RDF file per directory:

```
$ python AudioCollection.py metadata path/to/music
```

If it does not find the artist-album-song combination in the MusicBrainz database then no URI will be recorded for that song. If no URIs are recorded for the contents of a directory then no RDF file is generated for that directory.

Generate RDF from CSV list

If you don't have access to the media files you wish to generated RDF for but have the metadata already stored in a file, you can still use GNAT. In order to do so, a metadata list must be converted into Comma Separated Values (CSV).
Inside of the GNAT source directory you will find the file test.csv. This file will be used to show CSV functionality with GNAT.

The CSV must take one of these two forms:

```
URI, Artist Name, Album Name, Track Name
```

or

```
URI, Artist Name, Album Name, Track Name, Track Number
```

The URI may be any unique identifier you wish, but for the generated RDF to be useful, we suggest using identifiers from a HTTP namespace you control, or local file:/// URIs

```
file:///music/TestCollection/track001.mp3
```

In order to do this run the following command:

```
$ python AudioCollection.py metadata test.csv
```

or (if you would prefer simple csv output):

```
$ python AudioCollection.py –c metadata test.csv
```

Generate MusicBrainz fingerprints and metadata

Another way to find MusicBrainz entries is to use audio fingerprints. If you have tracks in your collection that have incorrect or missing metadata you can use the MusicIP fingerprinting method to attempt to locate the URIs on MusicBrainz. The retrieved URIs will then be used the generate local RDF files in the same way as the previous methods.

To run the fingerprinting:

```
$ python AudioCollection.py fingerprint path/to/music
```