Wiki-Lyrics HOW-TO

March 28, 2007

1 Introduction

1.1 What is Wiki-Lyrics?

It's an Amarok lyrics script. However, it has some features that makes it “special”:

- It can get lyrics (and suggestions) from *a lot of sites.*
- It can be easily extended to support other sites.
- It automatically searches suggestions for the requested lyrics.
- It can upload lyrics, cover art and album data to Lyriki and/or LyricWiki (wiki's for lyrics).

1.2 How does it works

Wiki-Lyrics himself has a sort of plugin system: in the main script's directory there are some Ruby files named `lyrics_*.rb`, each one defining a plugin (a special Ruby class) that can get the lyrics from *one site.* The main script calls them when it needs to fetch lyrics and/or suggestions.

2 The plugins

2.1 Writing a plugin

First few lines are “standard”:

```
$LOAD_PATH << File.expand_path(File.dirname(__FILE__))
require 'utils/strings'
require 'lyrics'
```

This is to import some classes and modules we'll have to use.
Now we can create our class, let's call it FooLyrics:

```
class FooLyrics < Lyrics
```

First two methods provide basic information about the site and are self explanatory:

```
def FooLyrics.site_host()
    return 'www.lyrics.foo.bar'
end

def FooLyrics.site_name()
    return 'Foo'
end
```

OK, that was easy. We just created 2 class-methods that return the URL and the name of the site we are getting the lyrics from. Now, let's write some real code. Though the `Lyrics` class can be extended in several ways, in the most common scenario (and the one we'll explain here), there are 4 additional methods our class should implement:

```
def build_lyrics_fetch_data( artist, title, album=nil, year=nil )
    a = CGI.escape( artist )
    t = CGI.escape( title )
    return { 'url'=>"http://#{site_host()}/index.php?a=#{a}&t=#{t}.html" }
end
```

The method receives song information and must return a hash with at least one key, `'url'`, and optionally another one, `'post'`. As the method name suggests, the data returned by this method will be used to fetch the lyrics page for the song in question, with the `'url'` key being the URL of the page to retrieve. Normally the page will be retrieved through a `GET` request but if some parameters need to be posted to the site in order to get the lyrics, you can provide them with the optional key `'post'`, which expects a hash. When `'post'` is specified, the page will be retrieved through a `POST` request and the data provided will be sent to the site (note that `GET` parameters are specified as part of the URL).

```
def parse_lyrics( url, body, artist, title, album=nil, year=nil )
    lyrics_data = {}
    # isolate the lyrics in the received page body
    ...
    lyrics_data['lyrics'] = body
    return lyrics_data
end
```

The method receives the fetched page and it's URL and the information received by `build_lyrics_fetch_data`. It also has to return a hash with at least one key, `'lyrics'`, and other optional keys: `'artist'`, `'title'`, `'album'`, `'year'` and `'custom_data'`. If the page contains the lyrics to the requested song, they must be returned in the `'lyrics'` key as plain text (not html, though you can leave formatting tags such as `<b>`, `<i>` or `<u>`); otherwise the `'lyrics'` key should be set to `nil`. You might wonder why return any of the optional keys that are already provided as arguments. The reason is that Wiki-Lyrics does not only retrieves lyrics but can also submit song information to Lyriki and LyricWiki. Instead of only providing the lyrics to the song, Wiki-Lyrics tries to provide those sites with as much (relevant) information as possible. To account for the fact that some information can be missing from the files metadata (or just be inaccurate), all plugins should try to obtain as much information as possible from the lyrics page. The keys that are expected along the lyrics for every plugin (`'artist'`, `'title'`, `'album'` and `'year'`) will be automatically filled with the values provided to the method if not filled by the plugin (so don't bother filling them unless the data is obtained from the page itself). Any other information available can be returned in the `'custom_data'` key, which must contain yet another hash. For example both, Lyriki and LyricWiki plugins return song credits and lyricist this way (as `'credits'` and `'lyricist'` keys respectively).

```
def build_suggestions_fetch_data( artist, title, album=nil, year=nil )
    a = CGI.escape( artist )
    return { 'url'=>"http://#{site_host()}/index.php?search=#{a}.html" }
end
```

This method is similar to `build_lyrics_fetch_data` but, instead of the URL of the lyrics page, it must return the URL where suggestions for the given song can be found (same things noted before for the `'post'` key apply here).

```
def parse_suggestions( url, body, artist, title, album=nil, year=nil )
    suggestions = []
    # isolate the suggestions in the received page body
    body.split( /separator regexp/ ) # split the suggestions in the body
        # analyze the suggestion and extract it's url, artist and title
        suggestions << { 'url'=> s_url, 'artist'=>s_artist, 'title'=>s_title }
    end
    return suggestions
end
```

As the name suggests, this method must parse the suggestions found in the received page body and return them as an array of hashes (one hash per suggestion), each hash containing `'url'`, `'artist'` and `'title'` keys.

2.1.1 Useful functions

Wiki-Lyrics includes some functions to help you in some “standard” operations. Some of those are listed below[1]:

- `Strings.latin12utf8(text)` convert text from latin1 to utf-8
- `Strings.utf82latin1(text)` convert text from utf-8 to latin1
- `build_google_feeling_lucky_url(artist, title)` returns the URL of the “feeling lucky” google search

2.2 Save and install the plugin

To have Wiki-Lyrics automatically detect plugins, these have to be saved under the script's main directory following the `lyrics_PLUGINCLASSNAME.rb` filename convention (*mandatory*, otherwise the plugin won't be loaded).
Countinuing our example, we would have to save our plugin as `lyrics_FooLyrics.rb` under Wiki-Lyrics main directory, usually `$HOME/.kde/share/apps/amarok/scripts/wiki_lyrics`.

3 Enjoy

Now open Amarok, start Wiki-Lyrics script and configure it to use your new plugin.
And, of course, sing along ;-)

---

[1] For other commonly used functions take a look at the Ruby modules in the `utils` directory.