# pylastfm Documentation

*Release 0.1*

Andi Albrecht

February 24, 2007

seeslow@users.sf.net

**Abstract**

This document describes the basic functionality of the `lastfm` module.

## Contents



The `lastfm` module provides access to the Last.fm / Audioscrobbler web services.

## 1 Download and installation

You have to ask? It's the only thing to do to your sprockets!

# 2 Getting started

## 2.1 Say Hello to RJ! (Profile Data)

First off, let's import the module and fetch some user profile data in XML format:

```
>>> import lastfm
>>> profile_xml = lastfm.user.profile_information("RJ", format=lastfm.FORMAT.xml)
>>> print profile_xml
<?xml version="1.0" encoding="UTF-8"?>
<profile id="1000002" cluster="2" username="RJ">
  <url>http://www.last.fm/user/RJ/</url>
  <realname>Richard Jones</realname>
  <mbox_sha1sum>1b374543545e01bc8d555a6a57c637f61f999fdf</mbox_sha1sum>
  <registered unixtime="1037793040">Nov 20, 2002</registered>
  <age>24</age>
  <gender>m</gender>
  <country>United Kingdom</country>
  <playcount>44547</playcount>
  <avatar>http://static.last.fm/avatar/0f4bda3a8e49e714c26ef610e2893454.jpg</avatar>
</profile>
>>>
```

Easy, isn't it. But it would be much easier, if we've got it objectified already. So let's request an etree object as return value:

```
>>> profile_dom = lastfm.user.profile_information("RJ", format=lastfm.FORMAT.etree)
>>> profile_dom
<<< <etree._ElementTree object at 0xb74be70c>
>>> profile_dom.getroot().get("username")
<<< 'RJ'
>>> profile_dom.xpath("//*/age")[0].text
<<< '24'
>>>
```

When you're requesting an ElementTree, the retrieved data will be parsed by the `etree` module without any modifications. You will have access to all fetched attributes and elements, but all values will be strings. Another way to retrieve pythonic return values is to request a Python object as provided by the `lastfm` module. These objects do some type conversion for you and provide additional methods for each object.

```
>>> profile = lastfm.user.profile_information("RJ")
>>> profile
<<< <Profile username='RJ'>
>>> profile.realname
<<< 'Richard Jones'
>>> profile.age
<<< 24
>>> profile.registered
<<< datetime.datetime(2002, 11, 20, 12, 50, 40)
```

As you can see, the `format` keyword isn't required, because it's the default format for all services that provide XML data and some of the values have been converted from strings to more pythonic types.

**Note:** Depending on the user's profile some of the available attributes have a `None` value, i.e. if a user hasn't give a real name, the `realname` attribute will be `None`.

The next section covers some of the methods provided by the `Profile` class.

## 2.2 What RJ likes most...

Now that we know RJ, what are his favourite artists? Let's have a look:

```
>>> import lastfm
>>> profile = lastfm.user.profile_information("RJ")
>>> artists_xml = profile.top_artists(format=lastfm.FORMAT.xml)
>>> print artists_xml
<?xml version="1.0" encoding="UTF-8"?>
<topartists user="RJ">
<artist>
    <name>Dream Theater</name>
    <mbid>28503ab7-8bf2-4666-a7bd-2644bfc7cb1d</mbid>
    <playcount>837</playcount>
    <rank>1</rank>
    <url>http://www.last.fm/music/Dream+Theater</url>
    <thumbnail>http:///storable/image/153635/small.jpg</thumbnail>
    <image>http://static.last.fm/proposedimages/sidebar/6/4209/432600.jpg</image>
</artist>
....
<artist>
    <name>The Doors</name>
    <mbid>9efff43b-3b29-4082-824e-bc82f646f93d</mbid>
    <playcount>140</playcount>
    <rank>49</rank>
    <url>http://www.last.fm/music/The+Doors</url>
    <thumbnail>http:///storable/image/148057/small.jpg</thumbnail>
    <image>http://static.last.fm/proposedimages/sidebar/6/344/408886.jpg</image>
</artist>
</topartists>
>>>
```

Uh, Dream Theater at rank 1 and The Doors as last... Ok, it's not my profile.

Again, let's get a little more pythonic:

```
>>> artists = profile.top_artists()
>>> artists
<<< <TopArtists username='RJ'>
>>> for artist in artists:
...     print artist.rank, artist.name
...
...
1 Dream Theater
2 Miles Davis
3 Aerosmith
>>>
```

You can get the same result without creating a `Profile` instance by calling the `lastfm.user.top_artists` function directly:

```
>>> artists = lastfm.user.top_artists("RJ")
>>> artists
<<< <TopArtists username='RJ'>
>>>
```

# 3 Advanced usage

## 3.1 Configuration

### Configuration Options

### Caching

| Option | Description | Type | Default |
|---|---|---|---|
| `cache.on` | Enables/disables the cache | bool | False |
| `cache.backend` | Caching backend: 'memory', 'filesystem' or an import path | str | 'memory' |

## 3.2 Caching

pylastfm provides some simple caching mechanisms. To enable caching you have to update the modules configuration. The simplest way to enable caching is to:

```
>>> import lastfm
>>> lastfm.config.update({"cache.on", True})
```

The `update` function will automatically reload all related modules, including the caching module with the new configuration options.

pylastfm provides two caching backends: The first is an In-Memory-Cache - which is the default, the second one is a filesystem based cache which keeps it's information accross sessions on a per user basis. You can choose between both backends by setting the `"cache.backend"` option. Here's a complete example:

```
>>> import lastfm
>>> lastfm.config.update({"cache.on" : True, "cache.backend" : "filesystem"})
>>> artists = lastfm.user.top_artists("RJ")
2007-02-20 12:22:08,280 DEBUG Retrieving 'http://ws.audioscrobbler.com/1.0/user/RJ/profile.xml'
2007-02-20 12:22:08,526 DEBUG Retrieving 'http://ws.audioscrobbler.com/1.0/user/RJ/topartists.x
>>> # Let's do it again!
>>> artists = lastfm.user.top_artists("RJ")
2007-02-20 12:22:17,300 DEBUG Retrieving 'http://ws.audioscrobbler.com/1.0/user/RJ/profile.xml'
2007-02-20 12:22:17,301 DEBUG Cached 'http://ws.audioscrobbler.com/1.0/user/RJ/profile.xml'
2007-02-20 12:22:17,302 DEBUG Retrieving 'http://ws.audioscrobbler.com/1.0/user/RJ/topartists.x
2007-02-20 12:22:17,302 DEBUG Cached 'http://ws.audioscrobbler.com/1.0/user/RJ/topartists.xml'
>>>
```

As you can see, calling the `top_artists` function a second time with the same parameters returns the information from the filesystem cache.

You can easily write your own cache backend by subclassing `lastfm.lib.cache.BaseCache` and setting the

"cache.backend" to an import path (i.e. "mymodule.MyCachingClass"). pylastfm will dynamically import your class and use it as it's caching backend.

### 3.3 Callbacks

### 3.4 PyGTK Integration

### 3.5 Error Handling

One error catch it all...: error.original_exc

## 4 lastfm — Interface to the Last.fm / Audioscrobbler web services

The spam module defines operations for handling cans of Spam. It knows the four generally available Spam varieties and understands both can sizes.

Because spamification requires UNIX process management, the module is only available on genuine UNIX systems.

The spam module defines the following functions:

**open** (*filename*[, *mode*[, *buffersize*]])
: Open the file *filename* as a can of Spam. The optional *mode* and *buffersize* arguments specify the read/write mode ('r' (default) or 'w') and the buffer size (default: system dependent).

**cansize**
: The default can size, in ounces. Legal values are 7 and 12. The default varies per supermarket. This variable should not be changed once the open() function has been called.

**exception error**
: Exception raised when an operation fails for a Spam specific reason. The exception argument is a string describing the reason of the failure.

### 4.1 Example

The following example demonstrates how to open a can of spam using the spam module.

```
>>> import spam
>>> can = spam.open('/etc/passwd')
>>> can.empty()
>>> can.close()
```

### 4.2 Spam Objects

Spam objects, as returned by open() above, have the following methods:

**empty**()
: Empty the can into the trash.

# A This is an Appendix

To create an appendix in a Python HOWTO document, use markup like this:

```
\appendix

\section{This is an Appendix}

To create an appendix in a Python HOWTO document, ....


\section{This is another}

Just add another \section{}, but don't say \appendix again.
```