infocards Documentation

Release 0.2.0

Rafael Medina García

August 07, 2014

Contents

CHAPTER 1

Getting Started

This document will show you a quick introduction on how to easily install and use the infocards library.

1.1 Requirements

- fuzzywuzzy >= 0.2.1
- pg8000 >= 1.9.13
- PyMYSQL >= 0.6.2
- SQLAlchemy >= 0.9.7

1.2 Installation

infocards has been tested on **Python 2.7** and **3.4**, although it should work in other versions such as **2.5**. The library can be installed from source by downloading and extracting the compressed file and then running:

```
$ python setup.py install
```

setuptools will also try to download the additional libraries required in the code.

However, the **recommended installation method** is to directly use pip to download and install the package (including dependencies) from the package index:

```
$ pip install infocards
```

1.3 Creating an archive

Once installed, creating a new archive (or connecting to an already existing one) is as easy as this:

```
from infocards import Archive

ar = Archive(CONNECTION_INFORMATION) # Be creative with your archive name!
ar.create_archive() # Use this only if you are creating an archive from scratch
```

Done, your archive is ready and awaiting its cards. From version 0.2.0 and up, the library supports working with MySQL, PostgreSQL and SQLite databases. For details on the connection information for each database type, check *Archive*.

1.4 Inserting a new card

Cards contain very little information, so creating a new card is very easy:

```
from infocards import Archive

ar = Archive('myarchive.dat')

title = 'This is my new card'
description = 'What a nice little description'
content = 'You can be as creative as you want when writing your content!!'
tags = 'tags are simply words separated by a whitespace'

are.new_card(title, description, content, tags)
```

You must take into account, however, that there cannot be two cards with the same title in the archive.

1.5 Retrieving a card

Cards have a unique title. In order to get the information on a specific card, simply ask the archive for that title:

```
from infocards import Archive

ar = Archive('myarchive.dat')

my_card = ar.get_card('This is my new card') # If a card with that title exists, now you can access
```

1.6 Retrieving a list of all the cards

If you want a complete list of all the *Cards* in the archive, simply do:

```
from infocards import Archive

ar = Archive('myarchive.dat')

all_cards = ar.all()
```

1.7 Updating a card

Cards can also be modified/updated easily so that you can add that information you forgot when creating the card!:

```
from infocards import Archive

ar = Archive('myarchive.dat')

my_card = ar.get_card('This is my new card') # Need to modify the original card
my_card.title = 'I like this title better' # You can even change the title!
my_card.content = 'Cards are easy to use'

ar.update_card('This is my new card', my_card) # Card updated!
```

As renaming a card is possible, modifications require the old title as well as the new card information.

1.8 Removing a card

Of course, you can even remove *Cards* from your archive. Simply use the card title, exactly the same as when retrieving a card:

```
from infocards import Archive

ar = Archive('myarchive.dat')

ar.remove_card('This is my new card') # Gone!
```

1.9 Searching for cards

Having to know all the card titles by heart can (and will) become frustrating. For that reason, the library includes a simple search functionality:

```
from infocards import Archive

ar = Archive('myarchive.dat')

# Search for cards containing 'Python'
result = ar.search('Python') # List of cards
```

CHAPTER 2

Archive

This document contains some more detailed information on the archive.

2.1 Structure

The archive consists of a SQLite database file (you decide the name) with a single table named *cards*. This table has the following columns:

```
id (Integer)[primary key] -> row id
title (Text)[unique] -> card title
description (Text) -> card description
content (Text) -> card content
tags (Text) -> card tags
modified (DateTime) -> last modification date and time
```

The *id* and *modified* columns don't need to be taken into account when dealing with the cards. The *id* is there just for convenience, while the *modified* column is automatically set to the date and time in which the card is inserted/modified automatically by the archive.

2.2 Connection

As of version 0.2.0, *infocards* supports archives based on MySQL, PostgreSQL and SQLite databases. You can find the information required for each database type below.

2.2.1 MySQL

infocards uses the PyMySQL module in order to connect to MySQL databases. The parameters for the connection are passed to the *Archive* constructor like so:

```
from infocards import Archive

ar = Archive(
    mysql = DATABASE_HOST,
    user = DATABASE_USER,
    passwd = DATABASE_PASSWORD,
    port = DATABASE_PORT,
    db = DATABASE_NAME)
```

2.2.2 PostgreSQL

For PostgreSQL databases, the pg8000 module is used. It's parameters are similar to those of the MySQL database:

```
from infocards import Archive

ar = Archive(
    postgresql = DATABASE_HOST
    user = DATABASE_USER,
    passwd = DATABASE_PASSWORD,
    port = DATABASE_PORT,
    db = DATABASE_NAME,
    ssl = USE_SSL) # Boolean value
```

The main difference is the included *ssl* parameter, which is used to indicate whether the connector should use SSL or not (default is false). Some databases, such as the Heroku ones, may need the SSL set to *True*.

2.2.3 SQLite

Connection to SQLite databases is done through the builtin *sqlite* module in Python:

```
from infocards import Archive

ar = Archive(
    sqlite = DATABASE_PATH)
```

SQLite databases are file-based, therefore you have to specify the absolute path of the file to connect to. If the file does not exist, it may be created by the Archive.

2.3 Search

The *search* functionality of the archive can be tunned a bit in order to obtain different results. Modyfing the *likelihood* parameter (default is 80) allows you to specify the percentage on which two words are considered similar. The *relevance* represents the percentage (default is 50) of search query words that must be included in the card tag list in order to consider that card as relevant to the search:

```
from infocards import Archive

# ar = Archive(...)

cards = ar.search("this is my search query", likelihood=80, relevance=50)
```

CHAPTER 3

Cards

This document contains some more detailed information on cards.

3.1 Structure

Card objects have the same structure that is shown in the *Archive* model structure, and hence you can access all of its parameters easily:

```
# card = Card(...)

title = card.title
description = card.description
content = card.content
tags = card.tags
modified = card.modified
```

The *title*, *description* and *content* attributes are strings, while *tags* is a list of strings and *modified* is a datetime stamp.

3.2 Tag strings and lists

Although tags are stored in the archive as a single string, in which each tag is separated by a whitespace, the *Card* object has its tags accesible as a list of strings. This list is alphabetically sorted, in lowercase and does not contain any repeated values (the library will make sure all of the three rules are present when creating a new card).

In order to convert a *tag string* to a *tag list* or viceversa, you can use the following static methods in the *card module*:

```
from infocards.card import Card

my_tag_string = "This is my tag list and it is awesome"

my_tag_list = Card.tag_list(my_tag_string)
# ['and', 'awesome', 'is', 'it', 'list', 'my', 'tag', 'this']

my_new_tag_string = Card.tag_string(my_tag_list)
# 'and awesome is it list my tag this'
```

3.3 Modify values of a card

As shown in the *Getting Started* document, editing a card is as simple as changing the value of its attributes. For example:

```
from infocards.card import Card

# card = Card(...)

# Changing card title
card.title = 'This is the new title'

# Changing description
card.description = 'This is the new description'

# Deleting content
card.content = ''

# Adding a tag
card.tags.append('python')
```

It would not make much sense to modify the *modified* attribute, as it is set automatically when inserting or modifying the card.

CHAPTER 4

Package Reference

4.1 archive module

class infocards.archive.**Archive**(**kwargs)
Bases: builtins.object

Database connection.

**kwargs contains the database connection information. Check the documentation on the *_create_engine()* method for details.

all()
Obtain a list of all the cards stored in the archive.

Returns list of Card objects

create_archive()
Create the corresponding schemas in the database. Must be used when connecting to an empty database.

get_card(*title*)
Obtain a card from the archive.

This is a direct search, so the title (or the row id) must be written as in the stored card.

Parameters **title** (*str*) – title of the card to get

Raises **NoCardFound** raised when the card does not exist

new_card(*title=''*, *description=''*, *content=''*, *tags=''*)
Create a new card for the archive.

Parameters

- **title** (*str*) – title of the card
- **description** (*str*) – description of the card
- **content** (*str*) – plain text content of the card
- **tags** (*str*) – sequence of tags identifying the card separated by whitespaces

Raises **InsertError** raised when a card already exists in the archive

remove_card(*title*)
Remove a card from the archive.

Parameters **title** (*str*) – title of the card to remove

Raises **NoCardFound** raised when the card cannot be found

search (*query*, *likelihood=80*, *relevance=50*)
Search for cards using the specified query.

A list of tags is created from the *title* and *tags* of the card and then compared with the query. If the percentage of query words present in the card list is greater or equal than *relevance*, then that card is added to the result.

If the query is empty, then a list of all the cards in the archive is returned.

Parameters

- **query** (*str*) – search query
- **likelihood** (*int*) – percentage for which two words are considered to be alike (0-100)
- **relevance** (*int*) – percentage for which a search query is considered relevant to the card. (0-100)

Returns list of **Card**

update_card (*title*, *new_card*)
Update the information of a card.

Parameters

- **title** (*str*) – title of the card to update
- **new_card** (*Card*) – *Card* object with the updated information

Raises NoCardFound raised when the card to update is not found

4.2 card module

class infocards.card.**Card** (*title*, *description*, *content*, *tags*, *modified*)
Bases: builtins.object

Cards have a simple structure and contain a small amount of information stored as plain text in the *Archive*.

Parameters

- **title** (*str*) – title of the card
- **description** (*str*) – a small description of the card
- **content** (*str*) – content of the card
- **tags** (*str*) – tags are stored as a sequence of words and/or sentences separated by whitespaces in the archive. However, the *Card* object will store the tags as a list for easier access.
- **modified** (*datetime*) – last modification's date and time

Note that in order to be able to search for cards, it is necessary to include tags.

static normalize (*tag_list=None*, *tag_string=None*)
Normalize a tag string or list.

Normalization is achieved by removing repeated words and transforming everyword into lowercase.

Parameters

- **tag_list** (*list*) – tag list to normalize (useful when modifying card tags)
- **tag_string** (*str*) – tag string to normalize

static **tag_list**(*tag_string*)
Obtain a tag list from a string.

This method is called when a new card is created. It will remove duplicate tags, set the tags to lowercase and sort the list when created.

Parameters **tag_string** (*str*) – string of tags separated by whitespaces

static **tag_string**(*tag_list*)
Obtain a string of tags from a list. Each tag will be separated by a whitespace.

Parameters **tag_list** (*list*) – list of tags to convert

4.3 exceptions module

exception infocards.exceptions.**ArchiveError**
Bases: builtins.Exception

Base class for the archive exceptions.

exception infocards.exceptions.**InsertError**(*message*)
Bases: infocards.exceptions.ArchiveError

Raised when an error occurs whilst trying to insert a new card in the archive.

Parameters **message** (*str*) – error explanation

exception infocards.exceptions.**NoCardFound**(*message*)
Bases: infocards.exceptions.ArchiveError

Raised when a specific card does not exist.

Parameters **message** (*str*) – error explanation

exception infocards.exceptions.**ParamError**(*message*)
Bases: infocards.exceptions.ArchiveError

Raised when function parameters are not valid.

Parameters **message** (*str*) – error explanation

exception infocards.exceptions.**SearchError**(*message*)
Bases: infocards.exceptions.ArchiveError

Raised when an error occurs when searching.

Parameters **message** (*str*) – error explanation

infocards is a small library that can be used to manage database archives containing information in a card-like structure.

Here you can find all the necessary information for *Getting Started* or have a look at the *Archive* or *Cards* details, or even the *Package Reference*

a

c

e

i