Nirvana Documentation

Release 0.0.1

Nick Wilson

Nov 17, 2017

Contents

1 Overview **3**

2 Installation **5**

3 User Guide **7**

4 Developer Guide **9**

5 Sitemap **11**
5.1 User Guide 11
5.2 Developer Guide 12

6 Indices and tables **15**

Warning: This project is still in the early planning/development phase. I'm starting with some Documentation Driven Development (DDD) where I write documentation before code to help ensure a good, usable interface design. I was going to link to Wikipedia's article on DDD but apparently they don't have one. Hmm... I swear I'm not crazy. Well, anyways, there's a good chance that nothing you see here actually exists yet. If it does, it probably doesn't work.

Warning: The NirvanaHQ API hasn't been released yet. I don't have any insider information about how the API will work other than what can be observed from the web app (which uses the API and is subject to change without notice). I'm delaying work on the lower-level part of this library that communicates with the API and am instead focusing on the higher-level interface for developer's using this library.

This is a Python library that provides access to the web-based task management software from NirvanaHQ, which is based on David Allen's popular Getting Things Done (GTD) methodology.

Note: Other than being huge fans, the developers of this library are not affiliated with NirvanaHQ in any way, and either is this library.

CHAPTER 1

Overview

The goal of this library is to provide full access to NirvanaHQ's API for use in Python programs. Let's start by showing you some of the things this library can do.

Authenticating with the API and accessing the user's data is simple:

```
>>> import nirvana
>>> auth = nirvana.auth.BasicAuth('username', 'password')
>>> nirv = nirvana.Nirvana(auth)
```

Let's print a welcome message to the user:

```
>>> print("Welcome, {0}!".format(nirv.user.first_name))
Welcome, Nick!
```

Next actions are easily accessible:

```
>>> for task in nirv.tasks.next():
...     print(task.name)
Task 1
Task 2
```

To create a new task:

```
>>> from nirvana.models import Task
>>> nirv.tasks.create('Task 3', state=Task.NEXT)
<Task: Task 3>
```

When you're done, synchronize the changes back to NirvanaHQ:

```
>>> nirv.sync()
```

Keep reading for more details on how you can use this library to access the rest of the features provided by NirvanaHQ's API.

CHAPTER 2

Installation

The easiest way to install the Nirvana library is with pip:

```
$ pip install nirvana
```

See the *installation* section of the *user guide* for more information and other installation options.

CHAPTER 3

User Guide

See the *user guide* for more details on using the Nirvana library in your own Python application.

CHAPTER 4

Developer Guide

Want to help develop this library? Great! See the *developer guide*.

CHAPTER 5

Sitemap

5.1 User Guide

Todo

Provide a intro/overview of the user guide.

Table of contents:

5.1.1 Supported Pythons and Platforms

Python 2.6 and 2.7 are currently supported. There are no plans to support 2.5, but 3.x is coming "soon".

I develop and test this library on Mac and Linux. I'd like to be friendly to other platforms (e.g., Windows) as well, but it may be a while before I get around to setting up a development environment under Windows to test it.

5.1.2 Installation

The recommended method of installing the latest official release of the Nirvana library is from the Python Package Index (PyPI). For the more adventurous, the most recent development version can be installed from the source code.

Python Package Index (PyPI)

The latest official release of the Nirvana library is available on the Python Package Index and can be installed with pip:

```
$ pip install nirvana
```

Source Code

The source code is hosted on GitHub. Once the source code is obtained, the Nirvana library can be installed by running the following command from the nirvana-python directory:

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

5.1.3 Library Structure

The Nirvana library is split into two parts:

1. The low-level interface is a thin Python layer on top of NirvanaHQ's API. It allows your Python programs to make calls directly with the API. Using the library at this level requires knowledge of the API that can be obtained from their API documentation.
2. The high-level interface builds on top of the low-level interface to provide nice, object-oriented, *pythonic* access to a user's tasks and related data. This is the way most developers will use this library. The documentation on this site should be sufficient for using the library at this level. NirvanaHQ's API documentation should not be required.

5.1.4 API Reference

Todo

(Waiting for code) Use autodoc to make use of the docstrings in the code.

5.2 Developer Guide

Todo

Provide a intro/overview of the developer guide.

Table of contents:

5.2.1 Documentation TODOs

This document lists any TODO items found in the documentation using the sphinx.ext.todo extension to Sphinx. If nothing shows up here, either there are no TODOs or, more likely, the `todo_include_todos` option was not set in Sphinx when this documentation was built.

Todo

Provide a intro/overview of the developer guide.

(The original entry is located in /home/docs/checkouts/readthedocs.org/user_builds/nirvana-python/checkouts/latest/docs/dev/index.rst, line 4.)

Todo

(Waiting for code) Use autodoc to make use of the docstrings in the code.

(The original entry is located in /home/docs/checkouts/readthedocs.org/user_builds/nirvana-python/checkouts/latest/docs/user/api/index.rst, line 4.)

Todo

Provide a intro/overview of the user guide.

(The original entry is located in /home/docs/checkouts/readthedocs.org/user_builds/nirvana-python/checkouts/latest/docs/user/index.rst, line 4.)

CHAPTER 6

Indices and tables

- genindex
- modindex
- search