WebAlerts Documentation

Release 0.0.4dev

Choongmin Lee

February 28, 2016

Contents

1 Quickstart **3**

2 Configuration options **5**

3 Logging **7**

4 API reference **9**
- 4.1 webalerts.sites . . . 9
- 4.2 webalerts.notifications . . . 10
- 4.3 webalerts.exceptions . . . 11

5 Indices and tables **13**

Python Module Index **15**

WebAlerts is a small Python framework that make notifications for new website posts matching specified patterns.

CHAPTER 1

Quickstart

The following is an example WebAlerts app:

```
from webalerts import App
app = App(config={
    'patterns': ['Samsung SSD (128|256)G'],
    'sites': {
        'clien': {
            'class': 'webalerts.sites.clien.Clien',
            'board_ids': ['sold'],
            'username': 'clienuser',
            'password': 'letmein',
        },
    },
    'notifications': {
        'email': {
            'class': 'webalerts.notifications.email.EmailNotification',
            'to_addrs': ['me@example.com'],
        },
    },
})
app.run()
```

It will check Clien every 5 minutes and send emails to me@example.com on posts about Samsung SSD 128G or 256G on the Sell board.

You can load the configurations from a YAML file. For example, the above configurations can be written as the following YAML file:

```
patterns: [Samsung SSD (128|256)G]
sites:
  clien:
    class: webalerts.sites.clien.Clien
    board_ids: [sold]
    password: letmein
    username: clienuser
notifications:
  email:
    class: webalerts.notifications.email.EmailNotification
    to_addrs: [me@example.com]
```

Then the program can be simplified as follows:

```
from webalerts import App
app = App()
app.from_yaml('config.yaml')
app.run()
```

CHAPTER 2

Configuration options

In essence, a WebAlert app is just a bunch of configurations. Below are the options you can use. Options without a default value are required.

patterns **(default: *None*)** List of regex patterns to match posts you want to be notified. Notifications will be sent if any of the patterns matches the title or the content of a post. If *patterns* is *None*, then it always matches any post.

check_interval **(default: 5)** Number of minutes between each loop where websites are checked for new posts. It should be an integer. The minimum value is 1.

notify_interval **(default: 5)** Number of minutes between each loop where notifications are sent. It should be an integer. The minimum vale is 1.

sites List of website settings, keyed by name. Each website accepts a different set of config options, although options like *username* and *password* are common for most sites. See `webalerts.sites` for more details. Below are general options for all sites:

class Site class object or its fully qualified name.

notifications **(default: all notifications)** Notification names that will be used for this site.

patterns **(default: global *patterns* value)** List of regex patterns to match posts you want to be notified. See the global *patterns* option for more.

check_interval **(default: global *check_interval* value)** Number of minutes between each loop where websites are checked for new posts. See the global *check_interval* option for more.

notifications List of notification settings, keyed by name. See `webalerts.notifications` for more details. Below are general options for all notifications options:

class Notification class object or its fully qualified name.

notify_interval **(default: global *notify_interval* value)** Number of minutes between each loop where notifications are sent. See the global *notify_interval* option for more.

CHAPTER 3

Logging

To enable debug logging with timestamps to standard out, put the following code before running the app:

```
import logging
logging.basicConfig(level=logging.DEBUG, format='%(asctime)s [%(name)s] [%(levelname)s] %(message)s')
```

For more details about configuring logging, see `logging` and `logging.config`.

Loggers are named after their module and class name, e.g. `App` class has a logger named `webalerts.app.App`.

CHAPTER 4

API reference

class webalerts.**App**(*config=None*)

A WebAlerts app object is initialized with configuration values in the provided *config* dictionary. For the list of config values, see Configuration options.

from_yaml(*name*)

Load config values from a YAML file.

run()

Start the main loop of the app that periodically collects new posts and feeds those posts to notification pipelines.

class webalerts.**Post**

Represent a generic post, a user's content published to a website. String parameters containing non-ASCII characters must be unicode.

Parameters

- **url** – URL of a post (required).
- **title** – Title of a post (required).
- **content** – text content of a post (required).
- **content_html** – HTML content of a post.
- **author** – Name of the original poster of a post.
- **author_id** – ID that uniquely identifies the original poster.
- **published** – `datetime` object containing the date and time when a post was published.

content_html_safe

A sanitized version of *content_html*. Notifications should use this value instead of *content_html*.

4.1 webalerts.sites

This package contains site-specific implementations on how to interact with the site.

A site class should be implemented in the following way:

- The constructor accepts a config dictionary. Typical configuration values include `username` and `password` for sites that require authentication.
- Implements `get_new_posts()`. It should return a list of `Post` objects since the last time it is called sorted by published time in ascending order, or an empty list if it is the first call. It must handle exceptions

expected in normal use and raise only instances of `SiteException` or `ConfigurationError` if necessary.

class `webalerts.sites.clien.`**`Clien`**(*config*)
Implementation for Clien. It accepts the following configuration options: *username*, *password*, *board_ids*.

username and *password* are your Clien username and password. Some boards in Clien require login to view posts.

board_ids is a list of identifiers of boards to watch. The identifier of a board can be found in its URL after `bo_table=`. Currently boards in special forms are not supported such as *Photos*.

As Clien does not provide public API, it works by parsing HTML markup of pages returned by the web server. It may not work at any time as the site owner has not explicitly granted scripted accesses to the site and the markup of the site is subject to change.

class `webalerts.sites.cgv.`**`CGV`**(*config*)
Implementation for CGV Cinemas. Get notified and get the best seat! Don't forget setting *patterns* to *None* in configuration for this site, if the global *patterns* is set.

Configuration options:

username CGV username.

password CGV password.

data List of tuples (*movie_name*, *movie_format*, *theater_name*, *time_range*, *date_range*, *seat_range*). You will be notified if any of the specified seats in the specified date and time range for the specified movie, format, and theater is found. All strings containing non-ASCII characters must be unicode. Movie, format, and theater names should be exact matches. It is recommended to find those in the official site.

time_range is a tuple of length 2, consisting of strings for times in HH:MM format. Hour can be larger than 23, so `'25:00'` is a valid time string. If you do not want to restrict times, set it to *None*.

date_range is a list of strings, either date in yyyymmdd format or short weekday names such as 'mon', 'fri'. `'weekdays'` and `'weekends'` are shortcuts for `['mon', 'tue', 'wed', 'thu', 'fri']` and `['sat', 'sun']` respectively. `'today'` is the current date in Korea Standard Time (UTC+9).

seat_range is a list of seat names such as 'A1' and 'F18'.

The following is an example tuple: `('Gravity', 'IMAX3D', 'Wangsimni', ('18:00', '24:30'), ['1130', 'fri', 'weekends'], ['F16', 'F17', 'G16', 'G17'])`

4.2 webalerts.notifications

This package contains notification implementations.

A notification class should be implemented in the following way:

- The constructor accepts a config dictionary.
- Implements `notify()`. It should take a list of posts and do what it is supposed to do, such as sending emails to users. It must handle all exceptions expected in normal use and raise only instances of `NotificationException` or `ConfigurationError` if necessary.

class `webalerts.notifications.email.`**`EmailNotification`**(*config*)
Send emails to users on matched posts.

Although you can specify any SMTP server to use to send emails, it is not recommended to use your own mail server as many email services refuse to receive emails from unknown sources. If you want to send emails using your Gmail account, set *host* to `'smtp.gmail.com'`, *port* to 587, *secure* to *True*.

Configuration options:

to_addrs List of email addresses to which notifications are sent.

from_addr **(default: `'WebAlerts <webalerts@localhost>'`)** "From" address of notification emails.

host **(default: *None*)** Optional host parameter used to create a `smtplib.SMTP` instance.

port **(default: *None*)** Optional port parameter used to create a `smtplib.SMTP` instance.

secure **(default: *False*)** Whether the SMTP connection should be secure or not.

username **(default: *None*)** SMTP username.

password **(default: *None*)** SMTP password.

style **(default: see the source)** CSS styles to be placed in in <head>.

template **(default: see the source)** HTML template for each posts.

layout **(default: see the source)** HTML template for the whole email.

class `webalerts.notifications.console.`**`ConsoleNotification`**(*config*)
Print the titles and URLs of the matched posts to the standard out.

It is intended to be used for debugging. There are no configuration values for this notification.

4.3 webalerts.exceptions

exception `webalerts.exceptions.`**`ConfigurationError`**
Bases: `webalerts.exceptions.WebAlertsException`

Raised when there is an error in configurations.

exception `webalerts.exceptions.`**`LoginError`**
Bases: `webalerts.exceptions.SiteException`

Raised when website authentication fails.

exception `webalerts.exceptions.`**`NotificationException`**
Bases: `webalerts.exceptions.WebAlertsException`

Raised when a notification-related error occurs, e.g. it fails to send emails.

exception `webalerts.exceptions.`**`ParseError`**
Bases: `webalerts.exceptions.SiteException`

Raised when it fails to parse the returned HTML.

exception `webalerts.exceptions.`**`SiteException`**
Bases: `webalerts.exceptions.WebAlertsException`

Raised when a website-related error occurs, e.g. it fails to login or there is a network problem.

exception `webalerts.exceptions.`**`WebAlertsException`**
Bases: `exceptions.Exception`

Root class of all exceptions defined in webalerts.

CHAPTER 5

Indices and tables

- *genindex*
- *modindex*
- *search*

W

webalerts, 9
webalerts.exceptions, 11
webalerts.notifications, 10
webalerts.notifications.console, 11
webalerts.notifications.email, 10
webalerts.sites, 9
webalerts.sites.cgv, 10
webalerts.sites.clien, 10

A

App (class in webalerts), 9

C

CGV (class in webalerts.sites.cgv), 10
Clien (class in webalerts.sites.clien), 10
ConfigurationError, 11
ConsoleNotification (class in webalerts.notifications.console), 11
content_html_safe (webalerts.Post attribute), 9

E

EmailNotification (class in webalerts.notifications.email), 10

F

from_yaml() (webalerts.App method), 9

L

LoginError, 11

N

NotificationException, 11

P

ParseError, 11
Post (class in webalerts), 9

R

run() (webalerts.App method), 9

S

SiteException, 11

W

webalerts (module), 9
webalerts.exceptions (module), 11
webalerts.notifications (module), 10
webalerts.notifications.console (module), 11
webalerts.notifications.email (module), 10
webalerts.sites (module), 9
webalerts.sites.cgv (module), 10
webalerts.sites.clien (module), 10
WebAlertsException, 11