rosteron Python Module

Release 1.0.0

Jun 11, 2019

Class Reference

The `rosteron` module allows read-only access to rostering information in instances of RosterOn Mobile, a workforce management product from Allocate Software.

```
>>> import rosteron
>>> with rosteron.Session('https://rosteron.xyz.com.au/RosterOnProd/Mobile') as␣
↪session:
...     session.log_in('joe.bloggs', 'abc123')
...     snapshot = session.get_roster()
>>> print(snapshot)
<Snapshot (time=2019-06-10T08:03:12+00:00, len=19)>
>>> for item in snapshot[:3]:
...     print(item)
<Item (date=2019-06-11, title='ABCDE - Melbourne Office', detail=('10:30 - 18:06',␣
↪None, 'XYZ', 'Assistant'))>
<Item (date=2019-06-12, title='ABCDE - Melbourne Office', detail=('10:30 - 18:06',␣
↪None, 'XYZ', 'Assistant'))>
<Item (date=2019-06-13, title='ABCDE - Melbourne Office', detail=('10:30 - 18:06',␣
↪None, 'XYZ', 'Assistant'))>
```

Complete documentation is hosted on Read the Docs.

CHAPTER 1

Features

- Roster data includes server-side retrieval timestamps.
- Sessions automatically log out after use (when used in a `with` block).
- Meaningful Python exceptions are raised when problems arise.
- Requests & responses to/from RosterOn can optionally be logged to files for debugging.

CHAPTER 2

Installation

Install this module from PyPI using pip:

```
pip install rosteron
```

CHAPTER 3

Support

The `rosteron` module is fully documented. Bug reports, feature requests, and questions are welcome via the issue tracker.

Documentation https://python-rosteron.readthedocs.io

Issue tracker https://github.com/Lx/python-rosteron/issues

CHAPTER 4

Contribute

4.1 Sample responses from other RosterOn installations

Roster output is minimally structured on the assumption that each RosterOn instance formats its data differently (the author has only seen data from one RosterOn Mobile instance).

Roster response samples from other RosterOn Mobile instances would be very gratefully received, as these may demonstrate uniformity across all instances, which would allow future releases of this module to provide more structured output.

4.2 Source code

Pull requests are gratefully received and considered.

GitHub repository https://github.com/Lx/python-rosteron

CHAPTER 5

License

This project is licensed under the MIT License.

CHAPTER 6

Index

- genindex

6.1 `Session` Class

class `rosteron.`**`Session`**(*url: str*, *browser: mechanicalsoup.StatefulBrowser = StatefulBrowser()*)

A *`Session`* object represents a connection to a RosterOn server, managing logging in, roster *`Snapshot`* retrieval, logging out, and optional file-based logging of RosterOn HTTP requests & responses.

`Session` objects are context managers, enabling automatic session log-out if used in a `with` block:

```
with Session(...) as session:
    session.log_in(...)
    snapshot = session.get_roster()

# session will always be logged out by this point
```

Parameters

- **`url`** – the base URL of the **Mobile** version of the RosterOn instance, e.g. `https://rosteron.example.com.au/RosterOnProd/Mobile`. The correct URL can be obtained for a RosterOn Mobile instance by visiting its "Log In" page in a browser and copying the portion of the URL prior to `/Account/Login`.
- **`browser`** – if specified, a custom `mechanicalsoup.StatefulBrowser` instance. Not required in normal usage; primarily intended for testing & diagnostic purposes.

6.1.1 `log_in()` Method

`Session.`**`log_in`**(*username: str*, *password: str*)

Log in to RosterOn with the specified user credentials.

Parameters

- **`username`** – the RosterOn user whose shifts are to be retrieved.
- **`password`** – the relevant RosterOn user's password.

Raises

- ***`BadCredentialsError`*** – if the RosterOn server doesn't accept the provided credentials.
- ***`BadResponseError`*** – if the RosterOn server returns an unexpected response.

Returns

this *`Session`* object, such that a *`log_in()`* call can be used in a `with` block if desired:

```
with session.log_in(...):
    snapshot = session.get_roster()

# session will always be logged out by this point
```

6.1.2 `get_roster()` Method

`Session.`**`get_roster`**`()` → rosteron.Snapshot

Retrieve a snapshot of the logged-in user's roster.

Return type *`Snapshot`*

Raises

- ***`NotLoggedInError`*** – if no RosterOn user is logged in.
- ***`BadResponseError`*** – if the RosterOn server returns an unexpected response.

6.1.3 `log_out()` Method

`Session.`**`log_out`**`()` → None

If a user is logged in to RosterOn, log them out; otherwise, do nothing.

This method is called automatically if the *`Session`* is used in a `with` block:

```
with Session(...) as session:
    session.log_in(...)
    snapshot = session.get_roster()

# session will always be logged out by this point
```

Raises ***`BadResponseError`*** – if a user is logged in **and** the RosterOn server returns an unexpected response while attempting to log out.

6.1.4 `save_logs()` Method

`Session.`**`save_logs`**`(`*directory: str*`)` → None

Log, to the specified directory, all RosterOn server requests & responses made over the life of the *`Session`*. Intended only for diagnostic purposes. Login credentials are not logged.

Each request/response will be saved to `<yyyymmddThhmmss.microseconds>Z-<purpose>-<n>.txt` in the specified directory, where:

- `<yyyymmddThhmmss.microseconds>Z` is the date & time of the initial request in UTC;
- `<purpose>` is the type of output expected for the operation triggering the initial request (`login`, `home`, `roster`, or `logout`); and
- `n` is `0` for the initial request/response pair in one operation, and a higher number for each subsequent request/response pair in that operation.

The typical *`Session`* usage of logging in, retrieving the roster, and logging out triggers requests & responses that would be logged as such:

```
20190610T042837.160169Z-login-0.txt
20190610T042838.576616Z-home-0.txt
20190610T042838.576616Z-home-1.txt
20190610T042838.934080Z-roster-0.txt
20190610T042839.134057Z-logout-0.txt
20190610T042839.134057Z-logout-1.txt
```

Each file will contain the date & time of the request, the request method & URL (not login credentials), and the server response (including status and headers):

```
2019-06-10 04:28:37.160169+00:00
GET https://rosteron.xyz.com.au/RosterOnProd/Mobile/Account/Login
200 OK

Date: Mon, 10 Jun 2019 04:28:38 GMT
Content-Type: text/html; charset=utf-8

<!DOCTYPE html>
...
```

Parameters **`directory`** – The directory where the requests & responses will be logged, which is assumed to exist and have appropriate write permissions.

6.2 `Snapshot` Class

class rosteron.**Snapshot**(*time: datetime.datetime, items: Sequence[Item]*)

A *`Snapshot`* object represents the state of a RosterOn user's roster at a specific point in time.

`Snapshot` objects are returned directly by the *`Session.get_roster()`* method, are *subscriptable*, are *iterable*, and have a *length* corresponding to the number of contained *`Items`*:

```
>>> snapshot[0] == snapshot.items[0]
True
>>> [item for item in snapshot][0] == snapshot.items[0]
True
>>> len(snapshot) == len(snapshot.items)
True
```

Parameters

- **`time`** – a `datetime` object holding the server's timestamp at which the roster was retrieved.
- **`items`** – a `tuple` of *`Item`* objects comprising the roster.

6.3 `Item` Class

class rosteron.**Item**(*date: datetime.date, title: str, detail: Sequence[Optional[str]]*)

An *Item* object represents one item on the roster.

Item objects aren't returned directly; they are instead returned within a *Snapshot* object.

Data in an *Item* is minimally structured on the assumption that each RosterOn instance formats its data differently (the author has only seen data from one RosterOn Mobile instance). Future releases of this module can provide more structured output if samples are provided from other RosterOn Mobile instances.

Parameters

- **date** – a `date` object.
- **title** – the text from the title above the roster item, minus the date and following separator.
- **detail** – a `tuple` of `str`/`None` values, where each value is either:
  - a string extracted from a `<p>` element in the roster item; or
  - `None` where an empty `<p>` element is encountered.

6.4 `BadCredentialsError` Exception

class rosteron.exceptions.**BadCredentialsError**(*username: str*)

BadCredentialsError exceptions are raised when RosterOn rejects the supplied username & password during a login operation.

The exception message includes the supplied username.

```
>>> from rosteron import exceptions
>>> raise exceptions.BadCredentialsError('joe.bloggs')
Traceback (most recent call last):
  File "<input>", line 1, in <module>
rosteron.exceptions.BadCredentialsError: RosterOn rejected the login credentials␣
↪for username 'joe.bloggs'
```

6.5 `NotLoggedInError` Exception

class rosteron.exceptions.**NotLoggedInError**

NotLoggedInError exceptions are raised when *get_roster()* is called on a *Session* where a user has not yet successfully logged in.

```
>>> from rosteron import exceptions
>>> raise exceptions.NotLoggedInError
Traceback (most recent call last):
  File "<input>", line 1, in <module>
rosteron.exceptions.NotLoggedInError: a RosterOn user must successfully log in␣
↪before a roster can be retrieved
```

6.6 BadResponseError Exception

class rosteron.exceptions.**BadResponseError**(*purpose: str*)

BadResponseError exceptions are raised when the RosterOn server returns a response that doesn't satisfy the needs of the current operation.

This could happen when an incorrect *Session* URL is used, when the RosterOn server is down, when a login error other than "bad username/password" occurs, or when logout occurs at an unexpected time.

The exception message includs the type of output that was expected (`login`, `home`, `roster`, or `logout`).

```
>>> from rosteron import exceptions
>>> raise exceptions.BadResponseError('login')
Traceback (most recent call last):
  File "<input>", line 1, in <module>
rosteron.exceptions.BadResponseError: RosterOn returned an unexpected response␣
↪for an operation expecting 'login'
```

6.7 RosterOnError Exception

class rosteron.exceptions.**RosterOnError**

RosterOnError exceptions are never raised directly.

This exception class exists solely as a base class for all other RosterOn-related exception classes, to enable "catch-all" error-handling when the specifics of the failure (beyond the fact that it is RosterOn-related) are unimportant:

```
try:
    ...
except RosterOnError:
    print('There was a RosterOn problem; continuing')
```

6.8 Private Methods & Classes

Knowledge of the following methods & classes (which are not part of the public `rosteron` module API) is only of benefit if further developing the `rosteron` module.

6.8.1 Private Session Methods

_browse() Method

Session.**_browse**(*url_fragment: Optional[str], purpose: str*) → rosteron._Response

Note the current client time, browse to the next page, log the response in case *save_logs()* is called later, and attempt to build a corresponding *_Response* object.

Parameters

- **url_fragment** – if specified, the URL (minus the base *Session* URL) that will be navigated to; if not specified, the current page's selected form will be submitted.
- **purpose** – the type of output expected by this navigation/submission (`login`, `home`, `roster`, or `logout`). Used in *BadResponseError* messages and response logging.

Return type *_Response*

Raises ***BadResponseError*** – if the response doesn't have the expected RosterOn page traits.

`__exit__()` Method

Session.**__exit__**(*exc_type*, *exc_val*, *exc_tb*) → bool

Ensure that the RosterOn user is logged out. Called at the end of any `with` block that uses this *Session* object.

The parameters describe the exception raised inside the `with` block, if any, and are not used.

Returns `False`, to indicate that any exception that occurred should propagate to the caller rather than be suppressed.

6.8.2 Private `_Response` Class

class rosteron.**_Response**(*time: datetime.datetime*, *id: str*, *content: bs4.Tag*)

A semi-evaluated RosterOn response, returned by the *Session._browse()* method.

Every interesting response from RosterOn Mobile conveniently holds its interesting content in a structure like this:

```
...
<div data-role="page" id="account-login">
    ...
    <div data-role="content">
        <!-- content of interest within -->
    </div>
</div>
...
```

The `id` attribute can be used to very confidently (and cheaply) determine the intent of the page. The content itself should only be further processed if that ID is as expected.

Parameters

- **time** – ideally the time as returned by the server in the response header; failing that, the client time when the request was started.
- **id** – the page ID as specified by the `id` attribute in the `<div data-role="page">` element.
- **content** – the `<div data-role="content">` element as a `bs4.Tag` object.

6.8.3 Private `_LogEntry` Class

class rosteron.**_LogEntry**(*time: datetime.datetime*, *response: requests.Response*, *purpose: str*)

A saved, timestamped RosterOn request/response pair for potential later logging to file.

These are constructed in *Session._browse()*, appended to the `Session._log` `list`, and emitted on request by *Session.save_logs()* as files.

Parameters

- **time** – the client time when the request was started.

- **`response`** – the final returned `requests.Response` object, which holds its corresponding request in its `request` attribute and intermediate responses (if any) in its `history` attribute.
- **`purpose`** – the type of output that was expected by this operation (`login`, `home`, `roster`, or `logout`). Used in the filename.

Index