python-webuntis Documentation

Release 0.1.9

Markus Unterwaditzer

Mar 03, 2017

Contents

Author Markus Unterwaditzer

Version 0.1.9

Project Page on GitHub

License *new-style BSD*

python-webuntis is a package for the API of WebUntis, a timetable system used for schools all around Europe. It is compatible with Python >= 2.6 and Python >= 3.3:

```
import webuntis

s = webuntis.Session(
    server='webuntis.grupet.at:8080',
    username='api',
    password='api',
    school='demo_inf',
    useragent='WebUntis Test'
)

s.login()

for klasse in s.klassen():
    print(klasse.name)

s.logout()
```

Output:

```
1A
2A
3A
[...]
```

Read More

CHAPTER 1

Index

Getting Started

Before we are getting started, there are a few things to know about WebUntis and its API:

Note:

- **You need an account for the API.** You can't access the API anonymously. It's designated that schools give each student a user account for the WebUntis server they're using. Many schools just make the timetable world-accessible though, preventing any use of the API. If you happen to be at such a school, you're a pitiful bastard.
- **The API is read-only.** And there's nothing you can do about it.
- **The API documentation does not explain the purpose of some methods.** So i can't do a much better job at it.
- **Different schools, different rules.** It is not neccessary that schools enter information about, for example, a teacher, in the correct format. It might happen that a school abuses the name field of a teacher to just write the teacher's initials in it. Testing is the only sane way out of this.
- **All dates and times are in local time.** WebUntis' JSON API returns time and date in local time, the timezones are configurable by the school. As if that wasn't ridiculous enough, there is no information on the timezone provided as part of the API. So this library can't make real efforts to handle date and time in a more professional manner.

Initially i started writing this library with the goal to abstract away all the pain that otherwise would result in direct interaction with the API. This is still an unreached goal. Some things like the problem with time and date is unsolvable. So if you think some part of the library could be easier to use, please let me know! I don't want python-webuntis to become as inconsistent and *weird* as the API it is wrapping.

Okay, let's install the webuntis package:

```
pip install webuntis
```

Here's the example from the *Intro* again:

```
import webuntis

s = webuntis.Session(
    server='webuntis.grupet.at:8080',
    username='api',
    password='api',
    school='demo_inf',
    useragent='WebUntis Test'
)

s.login()

for klasse in s.klassen():
    print(klasse.name)

s.logout()
```

So what does this do?

```
# a minimal example, these parameters are absolutely neccessary
s = webuntis.Session(
    server='webuntis.grupet.at:8080',
    username='api',
    password='api',
    school='demo_inf',
    useragent='WebUntis Test'
)
```

`webuntis.Session` represents a new session on a WebUntis server, which is bound to a school. This means that you will have to create a new session if you want to read data from a different school, even if it's on the same server.

But passing the credentials in doesn't mean a session is started immediately. You have to do it manually:

```
s.login()
```

This will raise an exception if you haven't provided the neccessary configuration (*i.e.* username, password, server, useragent, school).

So now that we're logged in, we can fetch some data. How about a list of all the classes of the school *demo_inf*? As the WebUntis software comes from Austria, these are called "klassen", which is German and means "classes". This name was probably choosen so there's no confusion between the classes of object-oriented programming languages and the classes that are actually important now.

Anyway, *python-webuntis* won't break that tradition:

```
for klasse in s.klassen():
    print(klasse.name)
```

We get a list-like, iterable object when calling `webuntis.Session.klassen()`, a `webuntis.objects.KlassenList` to be precise. This *KlassenList* contains multiple instances of `webuntis.objects.KlassenObject`. An instance of this object has multiple attributes, one of them being *name*.

At last, you get logged out with this:

```
s.logout()
```

You should always log out after doing your job, just like you should close a file after being done with it. For such reasons, Python has the with-statement, which you also can use to log yourself out automatically:

```
with webuntis.Session(...).login() as s:
    # work goes here

# now you're logged out, even if your code halted with exceptions before.
```

Where to go from here?

Session describes the `Session` class, which is the only class you will ever directly get in touch with.

Session

The Session Class

class `webuntis.`**`Session`**`(`***config*`)`

The origin of everything you want to do with the WebUntis API. Can be used as a context-manager to provide automatic log-out.

Configuration can be set with keyword arguments when initializing *`Session`*. Unless noted otherwise, they get saved in a dictionary located in the instance's *`config`* attribute and can be modified afterwards.

Parameters

- **`username`** (*`str`*) – The username used for the API.
- **`password`** (*`str`*) – The password used for the API.
- **`server`** (*`str`*) – A host name, a URL, or a URL without path.

```
s = webuntis.Session(..., server='thalia.webuntis.com')
# 'https://thalia.webuntis.com/WebUntis/jsonrpc.do'

# Want to disable SSL?
# make sure there's NO SLASH at the end!
s.config['server'] = 'http://thalia.webuntis.com'
# 'http://thalia.webuntis.com/WebUntis/jsonrpc.do'

# or maybe use a completely different API endpoint?
s.config['server'] = 'http://thalia.webuntis.com/WebUntis/jsonrpc2.
↪do'
# 'http://thalia.webuntis.com/WebUntis/jsonrpc2.do'

# or just change the path?
s.config['server'] = 'thalia.webuntis.com/WebUntis/jsonrpc2.do'
# 'https://thalia.webuntis.com/WebUntis/jsonrpc2.do'

s.config['server'] = '!"$%/WebUntis/jsonrpc.do'
# ValueError: Not a valid hostname
```

- **`school`** (*`str`*) – A valid school name.
- **`useragent`** (*`str`*) – A string containing a useragent. Please include useful information, such as an email address, for the server maintainer. Just like you would do with the HTTP useragents of bots.

- **cachelen** (*int*) – The maximum size of the internal cache. All results are saved in it, but they only get used if you set the `from_cache` parameter on a session method to `True`. This parameter is not saved in the configuration dictionary.

  ```
  s.timetable(klasse=123)  # saves in cache
  s.timetable(klasse=123)  # fetch data again, override old value
  s.timetable(klasse=123, from_cache=True)  # get directly from cache
  ```

  The reason this cache was added is that the API only allows you to fetch a whole list of objects (teachers/schoolclasses/...), not single ones. It would seriously harm performance to fetch the whole list each time we want information about a single object. Without the cache, i sometimes experienced a performance decrease about twenty seconds, so i wouldn't set the `cachelen` to anything smaller than `5`.

  Default value is `20`.

  You can clear the cache using:

  ```
  s.cache.clear('timetable')  # clears all cached timetables
  s.cache.clear()  # clears everything from the cache
  ```

- **jsessionid** (*str*) – The session key to use. You usually shouldn't touch this.
- **login_repeat** (*int*) – The amount of times *python-webuntis* should try to login when finding no or an expired session. Default to `0`, meaning it won't do that.

login()

Initializes an authentication, provided we have the credentials for it.

Returns

The session. This is useful for jQuery-like command chaining:

```
s = webuntis.Session(...).login()
```

Raises *webuntis.errors.BadCredentialsError* – Username/Password missing or invalid.

Raises *webuntis.errors.AuthError* – Didn't recieve a session ID for unknown reasons.

logout(*suppress_errors=False*)

Log out of session

Parameters **suppress_errors** (*bool*) – Whether to suppress errors.

Raises *webuntis.errors.NotLoggedInError* – Can't log out because not logged in. Raised unless `suppress_errors` is `True`.

config = None

The config dictionary, filled with most keyword arguments from initialization.

cache = None

Contains the caching dictionary for requests.

Things you can do with the API

class webuntis.**Session**

departments(***kwargs*)

Get all departments.

Return type *webuntis.objects.DepartmentList*

`holidays`(***kwargs*)
Get all holidays.

Return type *webuntis.objects.HolidayList*

`klassen`(***kwargs*)
Get all school classes.

Parameters **schoolyear** (*webuntis.objects.SchoolyearObject* or an integer ID of it) – The schoolyear where the classes should be fetched from.

Return type *webuntis.objects.KlassenList*

`timetable`(***kwargs*)
Get the timetable for a specific school class and time period.

Parameters

- **start** (`datetime.datetime` or `datetime.date`) – The beginning of the time period.
- **end** (`datetime.datetime` or `datetime.date`) – The end of the time period.

Return type *webuntis.objects.PeriodList*

Furthermore you have to explicitly define a klasse, teacher, subject, room or student parameter containing the id or the object of the thing you want to get a timetable about:

```python
import datetime
today = datetime.date.today()
monday = today - datetime.timedelta(days=today.weekday())
friday = monday + datetime.timedelta(days=4)

klasse = s.klassen().filter(id=1)[0]  # schoolclass #1
tt = s.timetable(klasse=klasse, start=monday, end=friday)
```

Raises `ValueError`, `TypeError`

`rooms`(***kwargs*)
Get all rooms of a school.

Return type *webuntis.objects.RoomList*

`schoolyears`(***kwargs*)
Get all schoolyears.

Return type *webuntis.objects.SchoolyearList*

`statusdata`(***kwargs*)
Information about lesson types and period codes, specifically about the colors used to highlight them in the web-interface of WebUntis.

Return type *webuntis.objects.StatusData*

`subjects`(***kwargs*)
Get all subjects.

Return type *webuntis.objects.SubjectList*

`teachers`(***kwargs*)
Get all teachers.

Return type *webuntis.objects.TeacherList*

Objects and Models

Note: The classes listed here never should be instantiated directly. Instead, use the methods of *webuntis.Session*.

class webuntis.objects.**ColorInfo**(*data*, *parent=None*, *session=None*)

Bases: *webuntis.objects.Result*

An object containing information about a lesson type or a period code:

```
>>> lstype = s.statusdata().lesson_types[0]
>>> lstype.name
'ls'
>>> lstype.forecolor
'000000'
>>> lstype.backcolor
'ee7f00'
```

```
>>> pcode = s.statusdata().period_codes[0]
>>> pcode.name
'cancelled'
>>> pcode.forecolor
'FFFFFF'
>>> pcode.backcolor
'FF0000'
```

backcolor

The background color used in the web interface and elsewhere

forecolor

The foreground color used in the web interface and elsewhere

name

The name of the LessonType or PeriodCode

class webuntis.objects.**DepartmentList**(*data*, *parent=None*, *session=None*)

Bases: *webuntis.objects.ListResult*

A list of departments, in form of *DepartmentObject* instances.

class webuntis.objects.**DepartmentObject**(*data*, *parent=None*, *session=None*)

Bases: *webuntis.objects.ListItem*

Represents a department

long_name

Long name, such as *Raum Erste A*. Not predictable.

name

short name such as *R1A*

class webuntis.objects.**HolidayList**(*data*, *parent=None*, *session=None*)

Bases: *webuntis.objects.ListResult*

A list of holidays, in form of *HolidayObject* instances.

class `webuntis.objects.`**`HolidayObject`**(*data*, *parent=None*, *session=None*)
Bases: `webuntis.objects.ListItem`

Represents a single holiday.

`end`
The end of the holiday

`name`
Name, such as *Nationalfeiertag*.

`short_name`
Abbreviated form of the name

`start`
The start date of the holiday, as a datetime object.

class `webuntis.objects.`**`KlassenList`**(*data*, *parent=None*, *session=None*)
Bases: `webuntis.objects.ListResult`

A list of school classes, in form of `KlassenObject` instances.

class `webuntis.objects.`**`KlassenObject`**(*data*, *parent=None*, *session=None*)
Bases: `webuntis.objects.ListItem`

Represents a school class.

`long_name`
Long name of class

`name`
Name of class

class `webuntis.objects.`**`ListItem`**(*data*, *parent=None*, *session=None*)
Bases: `webuntis.objects.Result`

ListItems represent an item in a `Result`. They don't contain methods to retrieve data.

class `webuntis.objects.`**`ListResult`**(*data*, *parent=None*, *session=None*)
Bases: `webuntis.objects.Result`

A list-like version of `Result` that takes a list and returns a list of objects, containing a list value each.

`filter`(***criterions*)
Return a list of all objects, filtered by attributes:

```
foo = s.klassen().filter(id=1)  # is kind-of the same as
foo = [kl for kl in s.klassen() if kl.id == 1]

# We can also use sets to match multiple values.
bar = s.klassen().filter(name={'1A', '2A', '3A', '4A'})
# is kind-of the same as
bar = [kl for kl in s.klassen()
       if kl.id in {'1A', '2A', '3A', '4A'}]

# Since ``filter`` returns a ListResult itself too, we can chain
# multiple calls together:
bar = s.klassen().filter(id=4, name='7A')  # is the same as
bar = s.klassen().filter(id=4).filter(name='7A')
```

`filter()` is also used when using the `in` operator on a `ListResult`:

```
we_have_it = {'name': '6A'} in s.klassen()  # same as
we_have_it = bool(s.klassen().filter(name='6A'))
```

Note: This is only available because it looks nicer than list comprehensions or generator expressions. Depending on your usecase alternatives to this method may be faster.

class `webuntis.objects.`**`PeriodList`**(*data*, *parent=None*, *session=None*)

Bases: *`webuntis.objects.ListResult`*

Aka timetable, a list of periods, in form of *`PeriodObject`* instances.

`to_table`(*dates=None*, *times=None*)

Creates a table-like structure out of the periods. Useful for rendering timetables in HTML and other markup languages.

Check out the example from the repository for clarification.

Parameters

- **`dates`** – An iterable of `datetime.date` objects that definetly should be included in the table. If this parameter is `None`, the timetable is just as wide as it has to be, leaving out days without periods.
- **`times`** – An iterable of `datetime.time` objects that definetly should be included in the table. If this parameter is `None`, the timetable is just as tall as it has to be, leaving out hours without periods.

Returns A list containing "rows", which in turn contain "hours", which contain *`webuntis.objects.PeriodObject`* instances which are happening at the same time.

class `webuntis.objects.`**`PeriodObject`**(*data*, *parent=None*, *session=None*)

Bases: *`webuntis.objects.ListItem`*

Represents a time range, where lessons/subjects may be held.

`code`

May be:

- `None` – There's nothing special about this period.
- `"cancelled"` – Cancelled
- `"irregular"` – Substitution/"Supplierung"/Not planned event

`end`

The end date/time of the period.

`klassen`

A *`KlassenList`* containing the classes which are attending this period.

`rooms`

The rooms (*`RoomList`*) where this period is taking place at. This also is not used for multiple lessons, but rather for a single lesson that is actually occuring at multiple locations (?).

`start`

The start date/time of the period, as datetime object.

`subjects`

A *`SubjectList`* containing the subjects which are topic of this period. This is not used for things like multiple language lessons (*e.g.* Latin, Spanish, French) – each of those will get placed in their own period.

`teachers`
A list of *`TeacherObject`* instances, which are attending this period.

`type`
May be:

- `"ls"` – Normal lesson
- `"oh"` – Office hour
- `"sb"` – Standby
- `"bs"` – Break Supervision
- `"ex"` – Examination

class `webuntis.objects.`**`Result`**(*data*, *parent=None*, *session=None*)
Bases: `object`

Base class used to represent most API objects.

Parameters

- **`data`** – Usually JSON data that should be represented.

  In the case of *`ListResult`*, however, it might also be a list of JSON mixed with *`ListItem`* objects.

- **`parent`** – (optional) A result object this result should be the child of. If given, the session will be inherited.
- **`session`** – Mandatory if `parent` is not supplied. Overrides the parent's inherited session.

`id`
The ID of this element.

An ID is needed for the object to be hashable. Therefore a result may bring its own implementation of this method even though the original API response didn't contain any ID.

class `webuntis.objects.`**`RoomList`**(*data*, *parent=None*, *session=None*)
Bases: *`webuntis.objects.ListResult`*

A list of rooms, in form of *`RoomObject`* instances.

class `webuntis.objects.`**`RoomObject`**(*data*, *parent=None*, *session=None*)
Bases: *`webuntis.objects.ListItem`*

Represents a physical room. Such as a classroom, but also the physics lab or whatever.

`long_name`
The long name of the room. Such as "Physics lab".

`name`
The short name of the room. Such as PHY.

class `webuntis.objects.`**`SchoolyearList`**(*data*, *parent=None*, *session=None*)
Bases: *`webuntis.objects.ListResult`*

A list of schoolyears, in form of *`SchoolyearObject`* instances.

`current`
Returns the current schoolyear in form of a *`SchoolyearObject`*

class `webuntis.objects.`**`SchoolyearObject`**(*data*, *parent=None*, *session=None*)
Bases: *`webuntis.objects.ListItem`*

Represents a schoolyear.

end
The end date

is_current
Boolean, check if this is the current schoolyear:

```
>>> y = s.schoolyears()
>>> y.current.id
7
>>> y.current.is_current
True
>>> y.filter(id=y.current.id).is_current
True
```

name
"2010/2011"

start
The start date of the schoolyear, as datetime object

class `webuntis.objects.`**`StatusData`**(*data*, *parent=None*, *session=None*)
Bases: `webuntis.objects.Result`

Information about lesson types and period codes and their colors.

lesson_types
A list of `ColorInfo` objects, containing information about all lesson types defined

period_codes
A list of `ColorInfo` objects, containing information about all period codes defined

class `webuntis.objects.`**`SubjectList`**(*data*, *parent=None*, *session=None*)
Bases: `webuntis.objects.ListResult`

A list of subjects, in form of `SubjectObject` instances.

class `webuntis.objects.`**`SubjectObject`**(*data*, *parent=None*, *session=None*)
Bases: `webuntis.objects.ListItem`

Represents a subject.

long_name
Long name of subject, such as *Physics*

name
Short name of subject, such as *PHY*

class `webuntis.objects.`**`TeacherList`**(*data*, *parent=None*, *session=None*)
Bases: `webuntis.objects.ListResult`

A list of teachers, in form of `TeacherObject` instances.

class `webuntis.objects.`**`TeacherObject`**(*data*, *parent=None*, *session=None*)
Bases: `webuntis.objects.ListItem`

Represents a teacher.

fore_name
fore name of the teacher

long_name
surname of teacher

name
full name of the teacher

surname
surname of teacher

Errors and Exceptions

python-webuntis tries to cover as many error codes recieved by the API as possible.

exception `webuntis.errors.`**`Error`**
Bases: `exceptions.Exception`

Superclass for all *python-webuntis*-specific errors, never gets raised directly.

exception `webuntis.errors.`**`RemoteError`**
Bases: *`webuntis.errors.Error`*, `exceptions.IOError`

There was some kind of error while interacting with the server.

`request` = **None**
The decoded JSON request which lead to this error, if available.

`result` = **None**
The decoded JSON response/result which lead to this error, if available.

`code` = **None**
The error code, if available.

exception `webuntis.errors.`**`MethodNotFoundError`**
Bases: *`webuntis.errors.RemoteError`*

The JSON-RPC method was not found. This really should not occur.

exception `webuntis.errors.`**`AuthError`**
Bases: *`webuntis.errors.RemoteError`*

Errors while logging in.

exception `webuntis.errors.`**`BadCredentialsError`**
Bases: *`webuntis.errors.AuthError`*, `exceptions.ValueError`

Invalid or missing username or password.

exception `webuntis.errors.`**`NotLoggedInError`**
Bases: *`webuntis.errors.AuthError`*

The session expired or we never logged in.

exception `webuntis.errors.`**`DateNotAllowed`**
Bases: *`webuntis.errors.RemoteError`*

The selected date range (for timetable) is not allowed.

Changelog

- 0.1.9:
  - Add requests as a dependency and use it instead of urllib. Big security improvement due to certificate validation.

- 0.1.8:
  - Rewrote testsuite. Should work with both py.test and nosetest.
  - Removed support for Python 3.2 and 3.1. In earlier days i also was a bit sloppy with unicode strings vs bytestrings. That sloppiness has been partially fixed.
  - Instead of showing a default error message when the error code is not recognized, webuntis will now try to use the error message sent in the response. See 67d6fa2.
- 0.1.7:
  - Bugfixes, as always.
  - `webuntis.errors.BadCredentialsError` now subclasses `ValueError`.
  - **Backwards incompatible:** Completely changed the API of `webuntis.objects.PeriodList.to_table()`, along with a rewrite of that function. Basically it doesn't accept a width parameter anymore, but sets of dates and times that should occur in the table. It now also pairs a `datetime.date` object with a set of hours instead of the weekday number.
- 0.1.6:
  - Just documentation improvements (simplifying) and internal restructuring.
- 0.1.5:
  - Bugfixes
  - Major internal restructuring.
    * Now caching result objects instead of JSON
    * Added true hierarchial inheritance for Result objects.
  - New `login_repeat` option that automatically refreshes your session if neccessary. See `webuntis.Session`.
  - `in` operator is now supported by `webuntis.objects.ListResult`
  - `webuntis.objects.ListResult.filter()` now returns a `webuntis.objects.ListResult` instead of a normal list.
  - **Backwards incompatible:** `webuntis.objects.PeriodObject` used to have a `type` attribute that returned things like `"cancelled"` or `"irregular"`. Due to me having read the API documentation too quickly, this is not like the `type` returned from the WebUntis API. So `type` is now renamed to `code` and the new `type` is something completely different.
- 0.1.4:
  - Updates to match changes in API.
  - Better docs.
  - Less bugs.
- 0.1.3:
  - Bugfix: Would crash at midnight times.
- 0.1.2:
  - Another bugfix wave.
  - Switched to nosetests, make management of tests easier.
  - Somehow i spelled “lesson” as “lession” throughout the whole module, in method names and elsewhere. This is fixed now, but it might break programs that are currently relying on that spelling error.

- 0.1.1:
  - Bugfixes
  - Added support for tox
  - Actual Python 2.6 support
- 0.1.0: First version of python-webuntis

Credits and License

Note: The following 3-clause BSD license applies to the full sourcecode shipped as part of python-webuntis, as well as to the documentation.

The license text basically says that you are free to modify and use python-webuntis, provided that the copyright sticks around. Furthermore, you may not use the author's name to promote derivatives of python-webuntis.

Python Module Index

Index