PyOpenMensa Documentation

Release 0.95.0

Malte Swart

Jan 23, 2018

Contents

1 Install PyOpenMensa **3**
- 1.1 Requirements . . . 3
- 1.2 Via git . . . 3
- 1.3 Via git submodule . . . 3

2 Creating OpenMensa Canteen Feeds **5**
- 2.1 Tutorial to generate OpenMensa Feeds . . . 5
- 2.2 Parse helpers . . . 5
- 2.3 Feed API . . . 7

3 Indices and tables **11**

Python Module Index **13**

This small python library helps you to work with OpenMensa by:

- **support writing canteen feeds**: *Creating OpenMensa Canteen Feeds* makes it very easy to generate a valid OpenMensa Feed V2
- **python wrapper for OpenMensa data**: Access data (canteens, meals) transparent from OpenMensa. **(in development)**

More information about OpenMensa and all possibilities for developers can be found in the OpenMensa Documentation.

Contents:

CHAPTER 1

Install PyOpenMensa

Currently only installation via git is supported, but building packages is planned (help is welcome).

1.1 Requirements

- **Python >= 3.2**, Python 2.6 and Python 2.7 are currently also supported, but Python 3 is recommended.

No additional packages or libraries are needed.

1.2 Via git

```
git clone git://github.com/mswart/pyopenmensa
```

1.3 Via git submodule

If you use PyOpenMensa in a project with git as source code management tool, you can add pyopenmensa via a git submodule:

```
git submodule add pyopenmensa git://github.com/mswart/pyopenmensa
git commit pyopenmensa
```

The handling of git submodules is sometings tricky: e.g. submodule are not used per default. The git book describes all needed steps.

CHAPTER 2

Creating OpenMensa Canteen Feeds

2.1 Tutorial to generate OpenMensa Feeds

2.2 Parse helpers

2.2.1 Date parsing

The first parameter of *BaseBuilder.addMeal()* specify the date. It should be a datetime.date. *LazyBuilder.addMeal()* supports also str as type: pyopenmensa tries to extract a date from it with *extractDate()*. Around the date itself can be garbage strings. The following date formats are recognized as 6th March 2013:

- **2013-03-06**, 2013-03-6, 2013-3-06, 2013-3-6
- **13-03-06**, 13-03-6, 13-3-06, 13-3-6
- **06.03.2013**, 6.03.2013, 06.3.2013, 6.3.2013
- **06.03.13**, 6.03.13, 06.3.13, 6.3.13
- **06. März 2013**, 6. März 2013
- **06. Maerz 2013**, 6. Maerz 2013
- **06. März 13**, 6. März 13
- **06. Maerz 13**, 6. Maerz 13

The white spaces and the dots in the last four formats are optional.

2.2.2 Prices

OpenMensa and pyopenmensa support also meal prices. The price can be specified for different roles. See the Documentation about OpenMensa Feed V2 for all supported roles.

Internally pyopenmensa counts in cents (integers). But strings and floats can be converted with *buildPrices()*. The following numbers are recognized as 3 Euro and 9 Cents:

- "3.09 €"
- "3,09 €"
- "3.09€"
- "3,09€"
- "3.09"
- "3,09"

Also these ways:

- a dictionary with a role to prices mapping:

```
buildPrices({'student': '3.64 €', 'employee': 3.84, 'others': 414})
```

- a iterator about prices and a iterator about roles:

```
buildPrices(['3.64€', 3.84, 414], ('student', 'employee', 'role'))
```

- base prices and additional costs for other roles:

```
buildPrices(3.64, default='student', addtional={'employee': '0.20€', 'others': 50}
↪)
```

will create:

```
{'student': 364, 'employee': 384, 'others': 414}
```

The fifth and sixth parameter of *LazyBuilder.addMeal()* are passed to *buildPrices()* - so there is normally no need to call it directly.

2.2.3 Legend helpers and notes

For every meal a feed can specify a list of additional notes. These notes are passed to *addMeal* as fourth parameter. The name contains in many cases food additive information (e.g. with number in brackets).

The *LazyBuilder* class helps with the extraction and converting of these data:

1. Use *LazyBuilder.setLegendData()* to define the food additive to identifier (like numbers) mapping. It uses *buildLegend()* to create the legend.

   You can pass a dictionary as parameter or a text as first.

   Additionaly also a text with a regex is accepted. The regex is used to extract this mapping from the given text. Therefore the regex needs to have two named groups *name* and *value*. Per default the regex extracts a number with a following bracket until the next number with a bracket.

2. Pass the name with food additives and other optional notes to *LazyBuilder.addMeal()*. All numbers and letters in brackets will be removed from the name and the name in the legend data is added to the notes list.

Example:

```
canteen.setLegendData('1) Schwein a)Farbstoff')
canteen.addMeal('2013-05-02', 'Hauptgerichte', 'Gulasch(1,a)', ['Mit Süßspeise'])
# is equal to
canteen.addMeal('2013-05-02', 'Hauptgerichte', 'Gulasch', ['Mit Süßspeise', 'Schwein',
↪ 'Farbstoff'])
```

2.3 Feed API

2.3.1 Base Canteen Feed Builder

class pyopenmensa.feed.**BaseBuilder**(*version=None*)

This class represents and stores all information about OpenMensa canteens. It helps writing new python parsers with helper and shortcuts methods. So the complete object can be converted to a valid OpenMensa v2 xml feed string.

static _handleDate(*date*)

Internal method that is used to handle/convert input date. It raises a `ValueError` if the type is no `datetime.date`. This method should be overwritten in subclasses to support other date input types.

Parameters **date** – input to be handled/converted

Return type datetime.date

addMeal(*date*, *category*, *name*, *notes=None*, *prices=None*)

This is the main helper, it adds a meal to the canteen. The following data are needed:

Parameters

- **date** (*datetime.date*) – Date for the meal
- **category** (*str*) – Name of the meal category
- **meal** (*str*) – Meal name.

Raises

- **ValueError** – if the meal name is empty or longer that 250 characters
- **ValueError** – if the price role is unknown
- **ValueError** – if the category name is empty
- **ValueError** – if note list contains empty note
- **TypeError** – if the price value is not an integer

Additional the following data are also supported:

Parameters

- **notes** (*list*) – List of notes
- **prices** (*dict*) – Price of the meal; Every key must be a string for the role of the persons who can use this tariff; The value is the price in Euro Cents, The site of the OpenMensa project offers more detailed information.

clearDay(*date*)

Remove all stored information about this date (meals or closed information).

Parameters **date** (*datetime.date*) – Date of the day

dayCount()

Return the number of dates for which information are stored.

Return type int

hasMealsFor(*date*)
Checks whether for this day are information stored.

Parameters **date** (*datetime.date*) – Date of the day

Return type bool

setDayClosed(*date*)
Define that the canteen is closed on this date. If a day is closed, all stored meals for this day will be removed.

Parameters **date** (*datetime.date*) – Date of the day

toTag(*output*)
This methods adds all data of this canteen as canteen xml tag to the given xml Document.

toXMLFeed() uses this method to create the XML Feed. So there is normally no need to call it directly.

Parameters **output** (*xml.dom.DOMImplementation.createDocument*) – XML Document to which the data should be added

toXMLFeed()
Convert this cateen information into string which is a valid OpenMensa v2 xml feed

Return type str

2.3.2 Lazy Canteen Feed Builder

class pyopenmensa.feed.**LazyBuilder**(**args*, ***kwargs*)
An extended builder class which uses a set of helper and auto-converting functions to reduce common converting tasks

Variables **extra_regex** – None: regex to be passed to extractNotes, Use *None* to use default regex provided by this module.

addMeal(*date*, *category*, *name*, *notes=None*, *prices=None*, *roles=None*)
Same as *BaseBuilder.addMeal()* but uses helper functions to convert input parameters into needed types. Meals names are auto-shortend to the allowed 250 characters. The following paramer is new:

Parameters **roles** – Is passed as role parameter to *buildPrices()*

setAdditionalCharges(*default*, *additional*)
This is a helper function, which fast up the calculation of prices. It is useable if the canteen has fixed additional charges for special roles.

Parameters

- **default** (*str*) – specifies for which price role the price of addMeal are.
- **additional** (*dict*) – Defines the extra costs (value) for other roles (key).

setLegendData(**args*, ***kwargs*)
Set or genernate the legend data from this canteen. Uses *buildLegend()* for genernating

Dates

pyopenmensa.feed.**extractDate**(*text*)
Tries to extract a date from a given str.

Parameters **text** (*str*) – Input date. A datetime.date object is passed thought without modification.

Return type datetime.date

Prices

pyopenmensa.feed.**convertPrice**(*variant*, *regex=None*, *short_regex=None*, *none_regex=<_sre.SRE_Pattern object>*)

Helper function to convert the given input price into integers (cents count). int, float and str are supported

Parameters

- **variant** – Price
- **regex** (*re.compile*) – Regex to convert str into price. The re should contain two named groups *euro* and *cent*
- **short_regex** (*re.compile*) – Short regex version (no cent part) group *euro* should contain a valid integer.
- **none_regex** (*re.compile*) – Regex to detect that no value is provided if the input data is str, the normal regex do not match and this regex matches *None* is returned.

Return type int/None

pyopenmensa.feed.**buildPrices**(*data*, *roles=None*, *regex=<_sre.SRE_Pattern object>*, *default=None*, *additional={}*)

Create a dictionary with price information. Multiple ways are supported.

Return type dict: keys are role as str, values are the prices as cent count

pyopenmensa.feed.**default_price_regex = <_sre.SRE_Pattern object>**

Compiled regular expression objects

Notes and Legends

pyopenmensa.feed.**buildLegend**(*legend=None*, *text=None*, *regex=None*, *key=<function <lambda>>*)

Helper method to build or extend a legend from a text. The given regex will be used to find legend inside the text.

Parameters

- **legend** (*dict*) – Initial legend data
- **text** (*str*) – Text from which should legend information extracted. None means do no extraction.
- **regex** (*str*) – Regex to find legend part inside the given text. The regex should have a named group *name* (key) and a named group *value* (value).
- **key** (*callable*) – function to map the key to a legend key

Return type dict

pyopenmensa.feed.**default_legend_regex = '(?P<name>(\\d|[a-z])+)\\)\\s*(?P<value>\\w+((\\s+**

str(object='') -> string

Return a nice string representation of the object. If the argument is a string, the return value is the same object.

pyopenmensa.feed.**extractNotes**(*name*, *notes*, *legend=None*, *regex=None*, *key=<function <lambda>>*)

This functions uses legend data to extract e.g. (1) references in a meal name and add these in full text to the notes.

Parameters

- **name** (*str*) – The meal name
- **notes** (*list*) – The initial list of notes for this meal
- **legend** (*dict*) – The legend data. Use *None* to skip extraction. The key is searched inside the meal name (with the given regex) and if found the value is added to the notes list.
- **regex** (*re.compile*) – The regex to find legend references in the meal name. The regex must have exactly one group which identifies the key in the legend data. If you pass None the *default_extra_regex* is used. Only compiled regex are supported.
- **key** (*callable*) – function to map the key to a legend key

Return type tuple with name and notes

pyopenmensa.feed.**default_extra_regex = <_sre.SRE_Pattern object>**

Compiled regular expression objects

CHAPTER 3

Indices and tables

- genindex
- modindex
- search

Python Module Index

p

pyopenmensa.feed, 7

Index

Symbols

_handleDate() (pyopenmensa.feed.BaseBuilder static method), 7

A

addMeal() (pyopenmensa.feed.BaseBuilder method), 7
addMeal() (pyopenmensa.feed.LazyBuilder method), 8

B

BaseBuilder (class in pyopenmensa.feed), 7
buildLegend() (in module pyopenmensa.feed), 9
buildPrices() (in module pyopenmensa.feed), 9

C

clearDay() (pyopenmensa.feed.BaseBuilder method), 7
convertPrice() (in module pyopenmensa.feed), 9

D

dayCount() (pyopenmensa.feed.BaseBuilder method), 7
default_extra_regex (in module pyopenmensa.feed), 10
default_legend_regex (in module pyopenmensa.feed), 9
default_price_regex (in module pyopenmensa.feed), 9

E

extractDate() (in module pyopenmensa.feed), 8
extractNotes() (in module pyopenmensa.feed), 9

H

hasMealsFor() (pyopenmensa.feed.BaseBuilder method), 7

L

LazyBuilder (class in pyopenmensa.feed), 8

P

pyopenmensa.feed (module), 7

S

setAdditionalCharges() (pyopenmensa.feed.LazyBuilder method), 8
setDayClosed() (pyopenmensa.feed.BaseBuilder method), 8
setLegendData() (pyopenmensa.feed.LazyBuilder method), 8

T

toTag() (pyopenmensa.feed.BaseBuilder method), 8
toXMLFeed() (pyopenmensa.feed.BaseBuilder method), 8