Unicum Documentation

Release 0.3 [4 - Beta]

sonntagsgesicht, based on a fork of Deutsche Postbank [pbrisk

Wednesday, 18 September 2019

Contents

CHAPTER 1

Introduction

1.1 Python library *unicum*

unicum consists of multiple object implementations that implement various factory pattern. All types merge into on type *VisibleObject* and each type contributes bits and piece.

The visible obj focus on robust and simple construction from a dictionary via *PersistentObject* having values only simple types or containers containers of simple types.

These values are translated via *FatoryObject* into more complex structures which are take from a factory.

Or, alternatively, using *DataRange* into something similar to a *data_frame* type in *R*, a table with column and row names as well as common types for each column values.

Inheriting from *LinkedObject* provides functionality to swap or update attributes at runtime

1.2 Example Usage

Using *FactoryObject*:

```
>>> from unicum import FactoryObject

>>> class Currency(FactoryObject): __factory = dict()
>>> class EUR(Currency): pass
>>> class USD(Currency): pass

>>> EUR().register()  # registers USD() instance with class name 'EUR'
>>> eur = Currency('EUR')  # picks instance with key 'EUR' from currency cache
>>> eur == EUR()  # picks instance with key given by class name 'EUR' from␣
↪currency cache, too.

True

>>> eur2 = eur.__class__('EUR')  # picks instance with key 'EUR' from currency␣
↪cache
>>> eur == eur2

True

>>> usd = USD().register()  # registers USD() instance with class name 'USD'
>>> usd.register('usd')  # registers usd with name 'usd'
>>> usd == USD()

True

>>> eur == eur.__class__('USD')

False

>>> usd == eur.__class__('USD')

True

>>> usd == Currency('usd')

True
```

Using *LinkedObject*:

```
>>> from unicum import LinkedObject
```

1.3 Development Version

The latest development version can be installed directly from GitHub:

```
$ pip install --upgrade git+https://github.com/sonntagsgesicht/unicum.git
```

1.4 Contributions

Issues and Pull Requests are always welcome.

1.5 License

Code and documentation are available according to the Apache Software License (see LICENSE).

CHAPTER 2

Tutorial

2.1 using VisibleObject

We give a simple example of a everyday network of objects.

Background is a framework to build a schedule of classes in a school. First we introduce **Student**, **Teacher** and **ClassRoom** as basic items.

Then we add container (lists) of them. And finally a **Lesson** as a *struct* with a **Teacher**, a **ClassRoom** and a list of **Student**.

A list of **Lesson** gives a **Schedule** and lists of **Student**, **Teacher** and **ClassRoom** together with a **schedule** build a **School**.

```
# -*- coding: utf-8 -*-

# unicum
# ------
# Python library for simple object cache and factory.
#
# Author:   sonntagsgesicht, based on a fork of Deutsche Postbank [pbrisk]
# Version:  0.3, copyright Wednesday, 18 September 2019
# Website:  https://github.com/sonntagsgesicht/unicum
# License:  Apache License 2.0 (see LICENSE file)


import logging
import sys

sys.path.append('..')
logging.basicConfig()

from unicum import VisibleObject, VisibleAttributeList, VisibleObjectList,␣
↪VisibleDataRange


class Person(VisibleObject):
    def __init__(self, name=''):
        super(Person, self).__init__(name)
```

(continues on next page)

(continued from previous page)

```
        self._age_ = 0


class Student(Person):
    def __init__(self, name=''):
        super(Student, self).__init__(name)
        self._school_class_ = SchoolClass()


class Teacher(Person):
    pass


class ClassRoom(VisibleObject):
    pass


class StudentList(VisibleAttributeList):
    def __init__(self, iterable=None):
        super(StudentList, self).__init__(iterable, Student)


class TeacherList(VisibleAttributeList):
    def __init__(self, iterable=None):
        super(TeacherList, self).__init__(iterable, Teacher)


class ClassRoomList(VisibleAttributeList):
    def __init__(self, iterable=None):
        super(ClassRoomList, self).__init__(iterable, ClassRoom)


class SchoolClass(VisibleObject):
    def __init__(self, name=''):
        super(SchoolClass, self).__init__(name)
        self._students_ = StudentList()


class Lesson(VisibleObject):
    def __init__(self):
        super(Lesson, self).__init__()
        self._subject_ = ''
        self._teacher_ = Teacher()
        self._class_room_ = ClassRoom()
        self._school_class_ = SchoolClass()
        self._day_ = 'Monday'
        self._time_ = '8:30'
        self._hour_ = 1


class Schedule(VisibleAttributeList):
    def __init__(self, iterable=None, object_type=Lesson,
                 value_types=(float, int, str, type(None), VisibleObject)):
        super(Schedule, self).__init__(iterable, object_type, value_types)


class School(VisibleObject):
    def __init__(self):
        super(School, self).__init__()
        self._teachers_ = TeacherList()
        self._students_ = StudentList()
```

(continues on next page)

(continued from previous page)

```
        self._class_rooms_ = ClassRoomList()
        self._schedule_ = Schedule()


if __name__ == '__main__':
    School().register()  # turns School() into an `unicum` class (with only one␣
↪`unnamed` instance)
    School().modify_object('Schedule', Schedule())  # mark School().Schedule as␣
↪modified

    # fill the Schedule with Lessons
    School().get_property('Schedule').append(
        Lesson.create(
            Subject='Math',
            Teacher='Mr. Logan',
            SchoolClass='FreshMen',
            ClassRoom='Room 1',
            Time='8:30'
        ))

    School().get_property('Schedule').append(
        Lesson.create(
            Subject='Physics',
            Teacher='Mr. Logan',
            SchoolClass='Senior',
            ClassRoom='Room 2',
            Time='10:15'
        ))

    School().get_property('Schedule').append(
        Lesson.create(
            Subject='Math',
            Teacher='Mr. Logan',
            SchoolClass='Senior',
            ClassRoom='Room 2',
            Time='12:00'
        ))

    School().get_property('Schedule').append(
        Lesson.create(
            Subject='History',
            Teacher='Mrs. Smith',
            SchoolClass='Senior',
            ClassRoom='Room 2',
            Time='8:30'
        ))

    School().get_property('Schedule').append(
        Lesson.create(
            Subject='Sports',
            Teacher='Mrs. Smith',
            SchoolClass='FreshMen',
            ClassRoom='Hall',
            Time='10:15'
        ))

    School().get_property('Schedule').append(
        Lesson.create(
            Subject='History',
            Teacher='Mrs. Smith',
            SchoolClass='FreshMen',
```

(continues on next page)

(continued from previous page)

```
            ClassRoom='Room 1',
            Time='12:00'
        ))

    # fill VisibleAttributeList
    School().modify_object('Teachers', TeacherList(('Mr. Logan', 'Mrs. Smith')).
↪register())
    School().modify_object('Students', StudentList(('Tom', 'Ben', 'Luisa', 'Peter',
↪ 'Paul', 'Mary')).register())
    School().modify_object('ClassRooms', ClassRoomList(('Room 1', 'Room 2', 'Hall
↪')).register())

    # give students an assigned class which makes the object tree circular:
    # School().Students[0] in School().Students[0].SchoolClass.Students
    # (hence, the object tree cannot be drawn as a json at once.)
    SchoolClass('FreshMen').register().modify_object('Students', School().get_
↪property('Students')[:3])
    for s in SchoolClass('FreshMen').get_property('Students'):
        s.modify_object('SchoolClass', SchoolClass('FreshMen'))

    SchoolClass('Senior').register().modify_object('Students', School().get_
↪property('Students')[3:])
    for s in SchoolClass('Senior').get_property('Students'):
        s.modify_object('SchoolClass', SchoolClass('Senior'))

    # now all items are stored in - can can be reconstructed from School() json
    print(School().to_json(all_properties_flag=True, indent=2))

    """
    {
      "Name": "School",
      "Class": "School",
      "Module": "__main__",
      "ClassRooms": [
        [     "Class" ,  "Module" ,  "Name" ],
        [ "ClassRoom" ,"__main__" ,"Room 1" ],
        [ "ClassRoom" ,"__main__" ,"Room 2" ],
        [ "ClassRoom" ,"__main__" ,  "Hall" ]
      ],
      "Schedule": [
        [  "Class" ,"ClassRoom" ,  "Module" ,  "Name" ,"SchoolClass" ,"Subject" ,
↪ "Teacher" , "Time" ],
        [ "Lesson" ,   "Room 1" ,"__main__" ,"Lesson" ,   "FreshMen" ,   "Math" ,
↪"Mr. Logan" , "8:30" ],
        [ "Lesson" ,   "Room 2" ,"__main__" ,"Lesson" ,     "Senior" ,"Physics" ,
↪"Mr. Logan" ,"10:15" ],
        [ "Lesson" ,   "Room 2" ,"__main__" ,"Lesson" ,     "Senior" ,   "Math" ,
↪"Mr. Logan" ,"12:00" ],
        [ "Lesson" ,   "Room 2" ,"__main__" ,"Lesson" ,     "Senior" ,"History" ,
↪"Mrs. Smith" , "8:30" ],
        [ "Lesson" ,     "Hall" ,"__main__" ,"Lesson" ,   "FreshMen" , "Sports" ,
↪"Mrs. Smith" ,"10:15" ],
        [ "Lesson" ,   "Room 1" ,"__main__" ,"Lesson" ,   "FreshMen" ,"History" ,
↪"Mrs. Smith" ,"12:00" ]
      ],
      "Students": [
        [   "Class" ,  "Module" , "Name" ,"SchoolClass" ],
        [ "Student" ,"__main__" ,  "Tom" ,   "FreshMen" ],
        [ "Student" ,"__main__" ,  "Ben" ,   "FreshMen" ],
        [ "Student" ,"__main__" ,"Luisa" ,   "FreshMen" ],
        [ "Student" ,"__main__" ,"Peter" ,     "Senior" ],
```

(continues on next page)

(continued from previous page)

```
        [ "Student" ,"__main__" , "Paul" ,      "Senior" ],
        [ "Student" ,"__main__" , "Mary" ,      "Senior" ]
      ],
      "Teachers": [
        [   "Class" ,  "Module" ,       "Name" ],
        [ "Teacher" ,"__main__" , "Mr. Logan" ],
        [ "Teacher" ,"__main__" ,"Mrs. Smith" ]
      ]
    }
    """

    # for didactic purpose we set Schedule as a VisibleDataRange and all other␣
↪lists as VisibleObjectList
    # (since `modify_object` would cast a VisibleObjectList to a TeacherList,␣
↪StudentList or ClassRoomList
    #  we have to workaround here.)

    School()._teachers_ = VisibleObjectList(School().get_property('Teachers'))
    School()._students_ = VisibleObjectList(School().get_property('Students'))
    School()._class_rooms_ = VisibleObjectList(School().get_property('ClassRooms'))
    School()._schedule_ = VisibleDataRange(School().get_property('Schedule').to_
↪serializable())

    # now we can not reconstructed from School() json as teachers, students and␣
↪class rooms are only given by name
    print(School().to_json(all_properties_flag=True, indent=2))

    """
    {
      "Name": "School",
      "Class": "School",
      "Module": "__main__",
      "ClassRooms": [
        "Room 1",
        "Room 2",
        "Hall"
      ],
      "Schedule": [
        [ null , "Class" ,"ClassRoom" ,  "Module" ,  "Name" ,"SchoolClass" ,
↪"Subject" ,   "Teacher" , "Time" ],
        [    0 ,"Lesson" ,   "Room 1" ,"__main__" ,"Lesson" ,   "FreshMen" ,
↪"Math" , "Mr. Logan" , "8:30" ],
        [    1 ,"Lesson" ,   "Room 2" ,"__main__" ,"Lesson" ,     "Senior" ,
↪"Physics" , "Mr. Logan" ,"10:15" ],
        [    2 ,"Lesson" ,   "Room 2" ,"__main__" ,"Lesson" ,     "Senior" ,
↪"Math" , "Mr. Logan" ,"12:00" ],
        [    3 ,"Lesson" ,   "Room 2" ,"__main__" ,"Lesson" ,     "Senior" ,
↪"History" ,"Mrs. Smith" , "8:30" ],
        [    4 ,"Lesson" ,     "Hall" ,"__main__" ,"Lesson" ,   "FreshMen" ,
↪"Sports" ,"Mrs. Smith" ,"10:15" ],
        [    5 ,"Lesson" ,   "Room 1" ,"__main__" ,"Lesson" ,   "FreshMen" ,
↪"History" ,"Mrs. Smith" ,"12:00" ]
      ],
      "Students": [
        "Tom",
        "Ben",
        "Luisa",
        "Peter",
        "Paul",
        "Mary"
      ],
```

(continues on next page)

(continued from previous page)

```
      "Teachers": [
        "Mr. Logan",
        "Mrs. Smith"
      ]
    }
    """

    # but we can extract all items we have so far and reconstruct from them
    for obj in VisibleObject.filter():
        print(VisibleObject(obj).to_json(all_properties_flag=False, indent=2))
        print()
        pass

    """
    {
      "Name": "School",
      "Class": "School",
      "Module": "__main__",
      "ClassRooms": [
        "Room 1",
        "Room 2",
        "Hall"
      ],
      "Schedule": [
        [ null , "Class" ,"ClassRoom" ,  "Module" ,  "Name" ,"SchoolClass" ,
↪"Subject" ,   "Teacher" , "Time" ],
        [    0 ,"Lesson" ,   "Room 1" ,"__main__" ,"Lesson" ,   "FreshMen" ,
↪"Math" , "Mr. Logan" , "8:30" ],
        [    1 ,"Lesson" ,   "Room 2" ,"__main__" ,"Lesson" ,     "Senior" ,
↪"Physics" , "Mr. Logan" ,"10:15" ],
        [    2 ,"Lesson" ,   "Room 2" ,"__main__" ,"Lesson" ,     "Senior" ,
↪"Math" , "Mr. Logan" ,"12:00" ],
        [    3 ,"Lesson" ,   "Room 2" ,"__main__" ,"Lesson" ,     "Senior" ,
↪"History" ,"Mrs. Smith" , "8:30" ],
        [    4 ,"Lesson" ,     "Hall" ,"__main__" ,"Lesson" ,   "FreshMen" ,
↪"Sports" ,"Mrs. Smith" ,"10:15" ],
        [    5 ,"Lesson" ,   "Room 1" ,"__main__" ,"Lesson" ,   "FreshMen" ,
↪"History" ,"Mrs. Smith" ,"12:00" ]
      ],
      "Students": [
        "Tom",
        "Ben",
        "Luisa",
        "Peter",
        "Paul",
        "Mary"
      ],
      "Teachers": [
        "Mr. Logan",
        "Mrs. Smith"
      ]
    }

    {
      "Name": "Mr. Logan",
      "Class": "Teacher",
      "Module": "__main__"
    }

    {
      "Name": "Mrs. Smith",
```

(continues on next page)

(continued from previous page)


```
  "Class": "Teacher",
  "Module": "__main__"
}

{
  "Name": "Tom",
  "Class": "Student",
  "Module": "__main__",
  "SchoolClass": "FreshMen"
}

{
  "Name": "Ben",
  "Class": "Student",
  "Module": "__main__",
  "SchoolClass": "FreshMen"
}

{
  "Name": "Luisa",
  "Class": "Student",
  "Module": "__main__",
  "SchoolClass": "FreshMen"
}

{
  "Name": "Peter",
  "Class": "Student",
  "Module": "__main__",
  "SchoolClass": "Senior"
}

{
  "Name": "Paul",
  "Class": "Student",
  "Module": "__main__",
  "SchoolClass": "Senior"
}

{
  "Name": "Mary",
  "Class": "Student",
  "Module": "__main__",
  "SchoolClass": "Senior"
}

{
  "Name": "Room 1",
  "Class": "ClassRoom",
  "Module": "__main__"
}

{
  "Name": "Room 2",
  "Class": "ClassRoom",
  "Module": "__main__"
}

{
  "Name": "Hall",
  "Class": "ClassRoom",
```


(continues on next page)

(continued from previous page)

```
    "Module": "__main__"
  }

  {
    "Name": "FreshMen",
    "Class": "SchoolClass",
    "Module": "__main__",
    "Students": [
      [   "Class" ,  "Module" , "Name" ,"SchoolClass" ],
      [ "Student" ,"__main__" ,  "Tom" ,   "FreshMen" ],
      [ "Student" ,"__main__" ,  "Ben" ,   "FreshMen" ],
      [ "Student" ,"__main__" ,"Luisa" ,   "FreshMen" ]
    ]
  }

  {
    "Name": "Senior",
    "Class": "SchoolClass",
    "Module": "__main__",
    "Students": [
      [   "Class" ,  "Module" , "Name" ,"SchoolClass" ],
      [ "Student" ,"__main__" ,"Peter" ,     "Senior" ],
      [ "Student" ,"__main__" , "Paul" ,     "Senior" ],
      [ "Student" ,"__main__" , "Mary" ,     "Senior" ]
    ]
  }
  """
```

2.2 setup a web api service

The second demo shows how to build a simple web service.

```
# -*- coding: utf-8 -*-

# unicum
# ------
# Python library for simple object cache and factory.
#
# Author:   sonntagsgesicht, based on a fork of Deutsche Postbank [pbrisk]
# Version:  0.3, copyright Wednesday, 18 September 2019
# Website:  https://github.com/sonntagsgesicht/unicum
# License:  Apache License 2.0 (see LICENSE file)


from datetime import datetime
from hashlib import md5

from flask import Flask, request, jsonify
from flask.helpers import make_response

from unicum import SessionHandler, VisibleObject


class DemoServer(Flask):
    """ restful api class """

    def __init__(self, session_handler=SessionHandler(), *args, **kwargs):

        # store session properties
```

(continues on next page)

(continued from previous page)

```
        self._session_handler = session_handler

        # initialize Flask
        kwargs['import_name'] = kwargs.get('import_name', 'unicum_web_service')
        super(DemoServer, self).__init__(*args, **kwargs)
        self.config['JSONIFY_PRETTYPRINT_REGULAR'] = False

        # initialize url routes/rules to manage session
        self.add_url_rule('/', view_func=self._start_session, methods=["GET"])
        self.add_url_rule('/<session_id>', view_func=self._validate_session,␣
↪methods=["GET"])
        self.add_url_rule('/<session_id>', view_func=self._stop_session, methods=[
↪"DELETE"])
        self.add_url_rule('/<session_id>/<func>', view_func=self._call_session,␣
↪methods=["GET", "POST"])

    # manage sessions
    def _start_session(self):
        """ starts a session """
        assert request.method == 'GET'

        hash_str = str(request.remote_addr) + str(datetime.now())
        session_id = md5(hash_str.encode()).hexdigest()

        session_id = self._session_handler.start_session(session_id)
        return make_response(session_id, 200)

    def _validate_session(self, session_id):
        result = self._session_handler.validate_session(session_id)
        return make_response(jsonify(result), 200)

    def _call_session(self, session_id, func=''):
        """ create object """

        assert request.method in ('GET', 'POST')
        if session_id not in request.base_url:
            return make_response(jsonify('session id %s does not match.' % session_
↪id), 500)

        # get key word arguments
        kwargs = dict()
        if request.method == 'GET':
            kwargs = request.args
        elif request.method == 'POST':
            kwargs = request.get_json(force=True)

        result = self._session_handler.call_session(session_id, func, kwargs)

        if isinstance(result, (bool, int, float, str)):
            result = str(result)
        else:
            result = jsonify(result)

        return make_response(result)

    def _stop_session(self, session_id):
        """ closes a session """
        assert request.method in ('DELETE', 'GET')
        assert session_id in request.base_url

        result = self._session_handler.stop_session(session_id)
```

(continues on next page)

(continued from previous page)

```
        return make_response(jsonify(result), 200)

    # manage server
    def _shutdown(self):
        for session_id in self._sessions:
            self._session_handler.stop_session(session_id)

        request.environ.get('werkzeug.server.shutdown')()
        res = 'shutting down...'
        return make_response(jsonify(res))


class DemoObject(VisibleObject):
    def __init__(self, *args, **kwargs):
        super(DemoObject, self).__init__(*args, **kwargs)
        self._folder_ = ''
        self._float_ = 0.


if __name__ == '__main__':
    import requests
    from _thread import start_new_thread

    ################################################
    # start server at http://127.0.0.1:64001
    ################################################

    url, port = '127.0.0.1', '64001'
    start_new_thread(DemoServer(SessionHandler('demo_server', 'DemoObject')).run,␣
↪(url, port))

    ################################################
    # start session
    ################################################

    base_url = 'http://%s:%s/' % (url, port)
    session_id = requests.get(url=base_url)

    ################################################
    # call session
    ################################################

    # ----------------------------------------------
    # create object
    # ----------------------------------------------

    url = base_url + session_id.text
    name = 'MyName'
    folder = 'MyFolder'
    res = requests.get(
        url=url + '/create',
        params={
            'name': name,
            'register_flag': True
        })
    assert res.text == name

    # ----------------------------------------------
    # modify object
    # ----------------------------------------------
```

(continues on next page)

(continued from previous page)

```
    res = requests.get(
        url=url + '/modify_object',
        params={
            'self': name,
            'property_name': 'Folder',
            'property_value_variant': folder
        })
    assert res.text == name

    res = requests.get(
        url=url + '/modify_object',
        params={
            'self': name,
            'property_name': 'Float',
            'property_value_variant': 123.321
        })
    assert res.text == name

    # ---------------------------------------------
    # get properties
    # ---------------------------------------------

    res = requests.get(
        url=url + '/get_property',
        params={
            'self': name,
            'property_name': 'Class'
        })
    assert res.text == 'DemoObject'

    res = requests.get(
        url=url + '/get_property',
        params={
            'self': name,
            'property_name': 'Folder'
        })
    assert res.text == folder

    res = requests.get(
        url=url + '/get_property',
        params={
            'self': name,
            'property_name': 'Float'
        })
    assert abs(float(res.text) - 123.321) < 1e-10

    ################################################
    # close session
    ################################################

    session_id = requests.delete(url=url)

    ################################################
    # stop server
    ################################################

    requests.delete(url=base_url)
```

2.3 call service from spread sheet

The third demo is a demo_workbook.xlsm which shows how call the web service from a spread sheet.

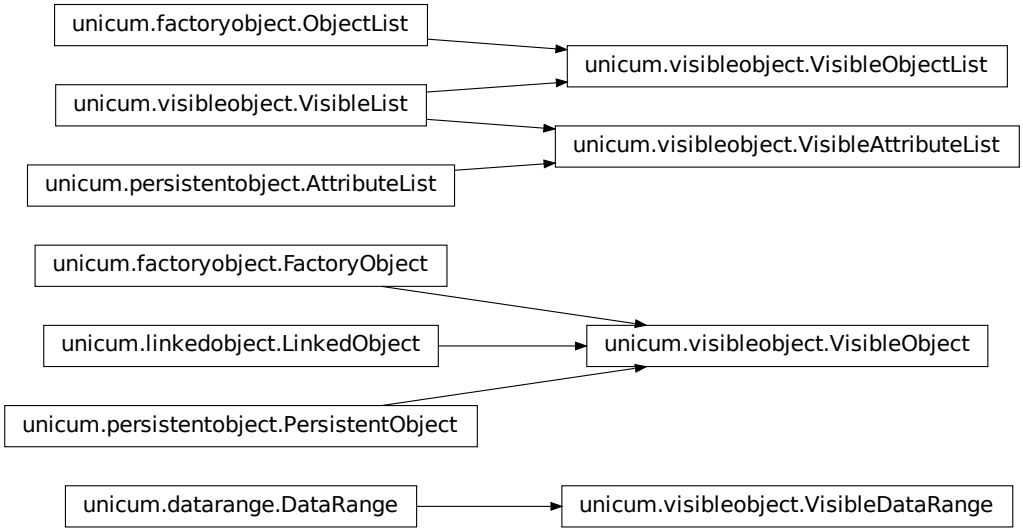
CHAPTER 3

API Documentation

3.1 Class List

factoryobject.FactoryObject	Objects identified by name
factoryobject.ObjectList	
linkedobject.LinkedObject	links from linked_obj to (obj, attribute) with obj.attribute = linked_obj
persistentobject.PersistentObject	
persistentobject.AttributeList	object list class
datarange.DataRange	
visibleobject.VisibleObject	
visibleobject.VisibleObjectList	
visibleobject.VisibleAttributeList	
visibleobject.VisibleDataRange	
session.SessionHandler	api session handler for multiprocessing sessions

3.2 Inheritance Diagram



3.3 Visible Objects

class unicum.visibleobject.**VisibleObject**(**args*, ***kwargs*)

Bases: *unicum.factoryobject.FactoryObject*, *unicum.linkedobject.LinkedObject*, *unicum.persistentobject.PersistentObject*

get_property(*property_name*, *property_item_name=None*)

to_serializable(*level=0*, *all_properties_flag=False*, *recursive=True*)

to_json(*all_properties_flag=False*, *property_order=('Name', 'Class', 'Module')*, ***kwargs*)

to_range(*all_properties_flag=False*)

classmethod from_serializable(*item*, *register_flag=False*)

core class method to create visible objects from a dictionary

classmethod from_json(*json_str*)

classmethod from_range(*range_list*, *register_flag=True*)

core class method to create visible objects from a range (nested list)

classmethod create(*name=None*, *register_flag=False*, ***kwargs*)

class unicum.visibleobject.**VisibleList**

Bases: list

register()

class unicum.visibleobject.**VisibleObjectList**(*iterable=None*, *object_type=<class 'unicum.visibleobject.VisibleObject'>*)

Bases: *unicum.factoryobject.ObjectList*, *unicum.visibleobject.VisibleList*

class unicum.visibleobject.**VisibleAttributeList**(*iterable=None*, *object_type=<class 'unicum.visibleobject.VisibleObject'>*, *value_types=(<class 'float'>, <class 'int'>, <class 'str'>, <class 'NoneType'>, <class 'unicum.visibleobject.VisibleObject'>)*)

Bases: *unicum.persistentobject.AttributeList*, *unicum.visibleobject.VisibleList*

class unicum.visibleobject.**VisibleDataRange**(*iterable=None*, *value_types=(<class 'float'>, <class 'int'>, <class 'str'>, <class 'NoneType'>, <class 'unicum.visibleobject.VisibleObject'>)*, *none_alias=(None, ' ', '', 'None')*)

Bases: *unicum.datarange.DataRange*

3.4 SessionHandler

class unicum.session.**SessionHandler**(*pkg_name='unicum'*, *cls_name='VisibleObject'*, *cast_types={}*)

Bases: `object`

api session handler for multiprocessing sessions

Parameters

- **pkg_name** – module containing relevant classes
- **cls_name** – default class (inherited from unicum.VisibleObject)
- **types** – additional dict of types to cast arguments

Standard type conversion is following a naming convention. So if an arguments ends with *int* the value will be casted with the type `int` given as value to the key *int* in **types**.

Same with *number*, *year*, *month*, *day* and *long*. Similar we cast *float* and *value* to a `float`, *string*, *str* and *name* to a `str` as well as *bool* and *flag* to `bool`

Anything ending with *variant* would be ignored.

And finally, the value of *cls* will be replaced by an attribute of **pkg_name** of the same name and the value of *self* will be replaced by an **cls_name** instance.

start_session(*session_id*)
: starts a session with given session_id

validate_session(*session_id*)
: checks wether a session with given session id exists

call_session(*session_id*, *func=''*, *kwargs={}*)
: calls the session and makes a function call with kwargs (which will be casted accordingly)

stop_session(*session_id*)
: closes a session with given session_id

3.5 Base Objects

3.5.1 Factory Objects

class unicum.factoryobject.**FactoryType**

Bases: `type`

classmethod get(*key*, *default=None*)

class unicum.factoryobject.**FactoryObject**(**args*, ***kwargs*)
Bases: object

Objects identified by name

register(**names*)

remove()

to_serializable(*level=0*, *all_properties_flag=False*, *recursive=True*)

classmethod from_serializable(*item*)

classmethod filter(*filter_func=None*)

classmethod get(*key*, *default=None*)

classmethod keys()

classmethod values()

classmethod items()

class unicum.factoryobject.**ObjectList**(*iterable=None*, *object_type=<class 'unicum.factoryobject.FactoryObject'>*)
Bases: list

index(*item*, *start=None*, *stop=None*)
Return first index of value.

Raises ValueError if the value is not present.

get(*item*, *default=None*)

append(*value*)
Append object to the end of the list.

insert(*index*, *value*)
Insert object before index.

extend(*iterable*)
Extend list by appending elements from the iterable.

to_serializable(*level=0*, *all_properties_flag=False*, *recursive=True*)

classmethod from_serializable(*item*)

3.5.2 Linked Objects

class unicum.linkedobject.**WeakAttrLink**(*obj*, *attr*)
Bases: object

ref_obj

attr

class unicum.linkedobject.**LinkedObject**
Bases: object

links from linked_obj to (obj, attribute) with obj.attribute = linked_obj

register_link(*obj*, *attr=None*)
creates link from obj.attr to self :param obj: object to register link to :param attr: attribute name to register link to

remove_link(*obj*, *attr=None*)
removes link from obj.attr

update_link()
redirects all links to self (the new linked object)

clean_up_link_dict()

3.5.3 Persistent Objects

class unicum.persistentobject.**PersistentObject**(**args*, ***kwargs*)
Bases: object

STARTS_WITH = '_'

ENDS_WITH = '_'

JSON_INDENT = 2

is_modified

classmethod from_serializable(*object_dict*)
core class method to create visible objects from a dictionary

to_serializable(*level=0*, *all_properties_flag=False*, *recursive=True*)

modify_object(*property_name*, *property_value_variant=None*)
api visible method for modifying visible object properties

Parameters

- **property_name** (*string, list or dict*) – property name
- **property_value_variant** (*various or None*) – property value, must be *None* if property_name is of type *dict*

Returns modified object

Return type unicum.lfojbect.VisibleObject

class unicum.persistentobject.**PersistentList**
Bases: list

classmethod from_serializable(*item*)

to_serializable(*level=0*, *all_properties_flag=False*, *recursive=True*)

class unicum.persistentobject.**PersistentDict**
Bases: dict

classmethod from_serializable(*item*)

to_serializable(*level=0*, *all_properties_flag=False*, *recursive=True*)

class unicum.persistentobject.**AttributeList**(*iterable=None*, *object_type=<class 'unicum.persistentobject.PersistentObject'>*, *value_types=(<class 'float'>, <class 'int'>, <class 'str'>, <class 'NoneType'>)*)
Bases: list

object list class

classmethod from_serializable(*item*)

append(*value*)
Append object to the end of the list.

index(*value*, *start=None*, *stop=None*)
Return first index of value.

Raises ValueError if the value is not present.

insert(*index*, *value*)
Insert object before index.

extend(*iterable*)
Extend list by appending elements from the iterable.

to_serializable(*level=0*, *all_properties_flag=False*, *recursive=True*)

keys(*level=0*, *all_properties_flag=False*)

values(*level=0*, *all_properties_flag=False*)

items(*level=0*, *all_properties_flag=False*)

3.5.4 DataRange Object

class unicum.datarange.**DataRange**(*iterable=None*, *value_types=(<class 'float'>, <class 'int'>, <class 'str'>, <class 'NoneType'>)*, *none_alias=(None, ' ', '', 'None')*, ***kwargs*)

Bases: object

update(*other*)

keys()

values()

items()

get(*key*, *default=None*)

pop(*key*, *default=None*)

popitem()

row_append(*row_key*, *value_list*)
append a new row to a DataRange

Parameters

- **row_key** – a string
- **value_list** – a list

col_append(*col_key*, *value_list*)
append a new row to a DataRange

Parameters

- **row_key** – a string
- **value_list** – a list

row_keys()

col_keys()

row(*item*)

col(*item*)

item_list

total_list

to_serializable(*level=0*, *all_properties_flag=False*, *recursive=True*)

transpose()

append(*key*, *value*)

extend(*other*)

insert(*item*, *key*, *value=None*)

CHAPTER 4

Releases

These changes are listed in decreasing version number order.

4.1 Release 0.3

Release date was Wednesday, 18 September 2019

- migration to python 3 (by dropping python 2 support)
- more documentation
- adding rest api support (incl. flask web service demo)
- adding storage backend (currently by file or sqllite3)

4.2 Release 0.2

Release date was December 31th, 2017

4.3 Release 0.1

Release date was July 7th, 2017

CHAPTER 5

Indices and tables

- genindex
- modindex
- search

Python Module Index

u

Index

J

K

L

M

O

P

R

S

T

U

V

W