pymzqc

Release v1.0.0

Mathias Walzer

Mar 11, 2022

CONTENTS:

CHAPTER

ONE

INTRODUCTION

A python library to creaete and use mzQC files. Specifically, the library facilitates access to mzQC files in form of a **directly usable object representation of mzQC** and offers additional functionality to:

- serialise
- deserialise
- syntactic checks
- semantic checks

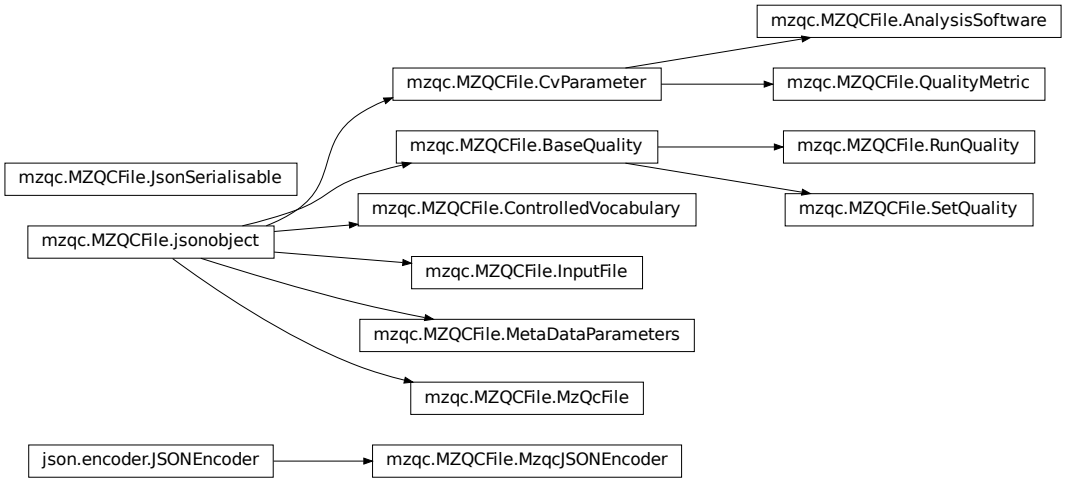
CHAPTER

TWO

CODE STRUCTURE

The library consists of three modules, the syntactic check module, the semantic check module, and the object representation module for (de-)serialisation.

2.1 Object Representation Overview



CHAPTER

THREE

MZQC PACKAGE

3.1 Submodules

3.2 mzqc.MZQCFile module

class mzqc.MZQCFile.**AnalysisSoftware**(*accession: str = ''*, *name: str = ''*, *description: str = ''*, *value: str = ''*, *unit: str = ''*, *version: str = ''*, *uri: str = ''*)

Bases: *mzqc.MZQCFile.CvParameter*

AnalysisSoftware Object representation for mzQC schema type AnalysisSoftware

class mzqc.MZQCFile.**BaseQuality**(*metadata: Optional[*mzqc.MZQCFile.MetaDataParameters*] = None*, *qualityMetrics: Optional[List[*mzqc.MZQCFile.QualityMetric*]] = None*)

Bases: *mzqc.MZQCFile.jsonobject*

BaseQuality Object representation for mzQC schema type BaseQuality

class mzqc.MZQCFile.**ControlledVocabulary**(*name: str = ''*, *uri: str = ''*, *version: str = ''*)

Bases: *mzqc.MZQCFile.jsonobject*

ControlledVocabulary Object representation for mzQC schema type ControlledVocabulary

class mzqc.MZQCFile.**CvParameter**(*accession: str = ''*, *name: str = ''*, *description: str = ''*, *value: Optional[Union[int, str, float, List[int], List[str], List[float], List[List[int]], List[List[str]], List[List[float]], Dict[str, List]]] = None*, *unit: str = ''*)

Bases: *mzqc.MZQCFile.jsonobject*

CvParameter Object representation for mzQC schema type CvParameter

class mzqc.MZQCFile.**InputFile**(*location: str = ''*, *name: str = ''*, *fileFormat: Optional[*mzqc.MZQCFile.CvParameter*] = None*, *fileProperties: Optional[List[*mzqc.MZQCFile.CvParameter*]] = None*)

Bases: *mzqc.MZQCFile.jsonobject*

InputFile Object representation for mzQC schema type InputFile

class mzqc.MZQCFile.**JsonSerialisable**

Bases: `object`

JsonSerialisable Main structure template for mzQC objects

Sets the foundation for a mzQC object to be readily (de-)serialisable with standard python json handling code. Facilitates reading and writing of complex objects.

classmethod FromJson(*json_str*, *complete=False*)

FromJson Main method for deserialisation

Accounts for neccessary object rectification due to same-attribute class footprints. N.B.: for this to work the class init variables must be same name as the corresponding member attributes (self).

Parameters

- **classself** (*self*) – The objects class self
- **json_str** (*str*) – The JSON string to be deserialised
- **complete** (*bool, optional*) – Flag to indicate if the whole JSON is to be returned deserialised, or just the *mzQC* entry (default).

Returns The deserialised JSON string

Return type MzQcFile object

classmethod ToJson(*obj*, *readability=0*, *complete=True*)

ToJson Main method for serialisation

Parameters

- **classself** (*self*) – The objects class self
- **obj** (*object*) – The object to be serialised
- **readability** (*int, optional*) – The indentation level, by default 0 (=no indentation, 1=minor indentation on MZQC objects, >1 heavy indentation for max. human readability)
- **complete** (*bool, optional*) – Flag to indicate if the object is to be left without the enclosing *mzQC* key or if the JSON is to be amended to full schema compliance (default).

Returns The serialisation result

Return type str

classmethod class_mapper(*d*)

class_mapper Maps incoming objects to their respective definition

Allows every registered object to 'know' its type map incuding recursing into its attributes. Can be used as object_hook in the json load process.

Parameters

- **classself** (*self*) – The objects class self
- **d** (*dict*) – The dictionary mapping attributes

Returns Returns an object of the 'outer-most' class

Return type class object

Raises **ValueError** – If expected date strings are invalid.

classmethod complex_handler(*obj*)

complex_handler Handles the in-depth serialisations necessary

Facilitates the correct serialisation for each type of object (within the registered mzQC JsonSerialisable context).

Parameters

- **classself** (*self*) – The objects class self
- **obj** (*object*) – The object to be deserialised

Returns The correct object deconstruction into its deserialisable bits

Return type obj

Raises **TypeError** – In case a given object cannot be serialised with the given set of functionalities.

mappings: Dict[str, Any] = {frozenset({'name', 'uri', 'version'}): <class 'mzqc.MZQCFile.ControlledVocabulary'>, frozenset({'accession', 'description', 'name', 'unit', 'value'}): <class 'mzqc.MZQCFile.QualityMetric'>, frozenset({'accession', 'description', 'name', 'unit', 'uri', 'value', 'version'}): <class 'mzqc.MZQCFile.AnalysisSoftware'>, frozenset({'fileFormat', 'fileProperties', 'location', 'name'}): <class 'mzqc.MZQCFile.InputFile'>, frozenset({'analysisSoftware', 'inputFiles', 'label'}): <class 'mzqc.MZQCFile.MetaDataParameters'>, frozenset({'metadata', 'qualityMetrics'}): <class 'mzqc.MZQCFile.SetQuality'>, frozenset({'contactAddress', 'contactName', 'controlledVocabularies', 'creationDate', 'description', 'runQualities', 'setQualities', 'version'}): <class 'mzqc.MZQCFile.MzQcFile'>}

classmethod register(*cls*)

register The method for class registration in the class mapping process

Each registered class gets mapped.

Parameters

- **classself** (*self*) – the objects class self
- **cls** (*object*) – the class type

Returns the class type

Return type cls

static time_helper(*da: str*) → datetime.datetime

time_helper Helper method for ISO8601 string of various length consumption

Used on JSON datetime object string representation will handle length and return python datetime objects.

Parameters **da** (*str*) – JSON datetime object string representation

Returns Python datetime object including the same amount detail provided

Return type datetime

class mzqc.MZQCFile.**MetaDataParameters**(*label: str = ''*, *inputFiles: Optional[List[*mzqc.MZQCFile.InputFile*]] = None*, *analysisSoftware: Optional[List[*mzqc.MZQCFile.AnalysisSoftware*]] = None*)

Bases: *mzqc.MZQCFile.jsonobject*

MetaDataParameters Object representation for mzQC schema type MetaDataParameters

class mzqc.MZQCFile.**MzQcFile**(*creationDate: Union[datetime.datetime, str] = datetime.datetime(2022, 3, 11, 17, 4, 1)*, *version: str = '1.0.0'*, *contactName: str = ''*, *contactAddress: str = ''*, *description: str = ''*, *runQualities: Optional[List[*mzqc.MZQCFile.RunQuality*]] = None*, *setQualities: Optional[List[*mzqc.MZQCFile.SetQuality*]] = None*, *controlledVocabularies: Optional[List[*mzqc.MZQCFile.ControlledVocabulary*]] = None*)

Bases: *mzqc.MZQCFile.jsonobject*

MzQcFile Object representation for mzQC schema type MzQcFile

class mzqc.MZQCFile.**MzqcJSONEncoder**(**, skipkeys=False*, *ensure_ascii=True*, *check_circular=True*, *allow_nan=True*, *sort_keys=False*, *indent=None*, *separators=None*, *default=None*)

Bases: json.encoder.JSONEncoder

MzqcJSONEncoder The encoder used to facilitate indented encoding

Handles the string encoding and formatting of the serialised objects.

iterencode(*o*, *_one_shot=False*)
Encode the given object and yield each string representation as available.

For example:

```
for chunk in JSONEncoder().iterencode(bigobject):
    mysocket.write(chunk)
```

class mzqc.MZQCFile.**QualityMetric**(*accession: str = ''*, *name: str = ''*, *description: str = ''*, *value: Optional[Union[int, str, float, List[int], List[str], List[float], List[List[int]], List[List[str]], List[List[float]], Dict[str, List]]] = None*, *unit: str = ''*)
Bases: *mzqc.MZQCFile.CvParameter*

QualityMetric Object representation is passed for its more concrete derivatives

class mzqc.MZQCFile.**RunQuality**(*metadata: Optional[*mzqc.MZQCFile.MetaDataParameters*] = None*, *qualityMetrics: Optional[List[*mzqc.MZQCFile.QualityMetric*]] = None*)
Bases: *mzqc.MZQCFile.BaseQuality*

QualityMetric Object representation is passed for its more general basis

class mzqc.MZQCFile.**SetQuality**(*metadata: Optional[*mzqc.MZQCFile.MetaDataParameters*] = None*, *qualityMetrics: Optional[List[*mzqc.MZQCFile.QualityMetric*]] = None*)
Bases: *mzqc.MZQCFile.BaseQuality*

SetQuality Object representation is passed for its more general basis

class mzqc.MZQCFile.**jsonobject**
Bases: `object`

jsonobject Proxy object for better integration of mzQC objects

Useful for testing and validity checks as __eq__ is overridden to compare all attributes as well.

mzqc.MZQCFile.**rectify**(*obj*)
rectify Rectifies objects according to their position in the local hierarchy

Carries out the neccessary object rectification due to same-attribute class footprints. Rectification depends on the object position in the local object hierarchy.

Parameters **obj** (*object*) – The object to be rectified

Returns The rectified object

Return type object

3.3 mzqc.SemanticCheck module

class mzqc.SemanticCheck.**SemanticCheck**(*version: str = ''*)
Bases: `object`

validate(*mzqc_obj:* mzqc.MZQCFile.MzQcFile)

exception mzqc.SemanticCheck.**SemanticError**(*message*, *validator=<unset>*, *path=()*, *cause=None*, *context=()*, *validator_value=<unset>*, *instance=<unset>*, *schema=<unset>*, *schema_path=()*, *parent=None*)

Bases: jsonschema.exceptions.ValidationError

Base class for exceptions in this module.

3.4 mzqc.SyntaxCheck module

class mzqc.SyntaxCheck.**SyntaxCheck**(*version: str = 'main'*)

Bases: object

SyntaxCheck class for syntax validations of mzQC objects (after JSON dump)

Using member function validate of the SytnaxCheck class, mzQC objects can be checked for correct syntax in its built-in serialisation.

validate(*mzqc_str: str*)

3.5 Module contents

CHAPTER

FOUR

MZQC

CHAPTER

FIVE

USAGE EXAMPLES

I am a fan of quick hands-on overviews to get to know software, that I might or might not use in the future. Several hands-on examples can be found here. A video version for ASMS'22 is on youtube:

The copy&paste essentials:

5.1 Load

```
from mzqc import MZQCFile as qc
with open("nameOfYourFile.mzQC", "r") as file:
    my_run_qualities = qc.JsonSerialisable.FromJson(file)
```

5.2 Access elements

see schema for a general overview of available elements.

```
# An in-memory mzQC file will still have the same hierarchical structure as the schema
print(my_run_qualities.description)

# JSON arrays can be used like python lists
for m in my_run_qualities.qualityMetrics:
    print(m.name)

# You can traverse the hierarchy with standard python member access notation ('.')
# and get to the bottom of things (like a metric value).
ms2_number = my_run_qualities.qualityMetrics[2].value
```

5.3 Store

```
inmem_file = qc.JsonSerialisable.ToJson(mzqc, readability=1)
with open("nameOfYourFile.mzQC", "w") as file:
    file.write(inmem_file)
```

CHAPTER

SIX

INDICES AND TABLES

- modindex
- search

PYTHON MODULE INDEX

m

INDEX