WikibaseIntegrator

Release 0.12.4

LeMyst and WikibaseIntegrator contributors

Jun 07, 2023

CONTENTS

CHAPTER

ONE

WIKIBASEINTEGRATOR

1.1 Subpackages

1.1.1 wikibaseintegrator.datatypes

1.1.1.1 Subpackages

1.1.1.1.1 wikibaseintegrator.datatypes.extra

wikibaseintegrator.datatypes.extra.edtf

class wikibaseintegrator.datatypes.extra.edtf.**EDTF**(*value=None*, ***kwargs*)

Bases: *String*

Implements the Wikibase data type for Wikibase Extended Date/Time Format extension. More info at https://www.mediawiki.org/wiki/Extension:Wikibase_EDTF

Parameters

- **value** (*str | None*) –
- **kwargs** (*Any*) –

DTYPE = 'edtf'

__init__(*value=None*, ***kwargs*)

Constructor, calls the superclass BaseDataType

Parameters

- **value** (Optional[str]) – The string to be used as the value
- **kwargs** (*Any*) –

equals(*that*, *include_ref=False*, *fref=None*)

Tests for equality of two statements. If comparing references, the order of the arguments matters!!! self is the current statement, the next argument is the new statement. Allows passing in a function to use to compare the references 'fref'. Default is equality. fref accepts two arguments 'oldrefs' and 'newrefs', each of which are a list of references, where each reference is a list of statements

Return type
bool

Parameters

- **that** (Claim) –

- **include_ref** (*bool*) –
- **fref** (*Callable | None*) –

from_json(*json_data*)

Parameters
json_data (Dict[str, Any]) – a JSON representation of a Claim

Return type
Claim

get_json()

Return type
Dict[str, Any]

get_sparql_value()

Return type
str

has_equal_qualifiers(*other*)

Return type
bool

Parameters
other (Claim) –

property id: str | None

property mainsnak: *Snak*

parse_sparql_value(*value*, *type='literal'*, *unit='1'*)

Return type
bool

property qualifiers: *Qualifiers*

property qualifiers_order: List[str]

property rank: *WikibaseRank*

property references: *References*

static refs_equal(*olditem*, *newitem*)

tests for exactly identical references

Return type
bool

Parameters

- **olditem** (Claim) –
- **newitem** (Claim) –

remove(*remove=True*)

Return type
None

property removed: bool

set_value(*value=None*)

Parameters
value (*str | None*) –

property type: str | Dict

update(*claim*)

Return type
None

Parameters
claim (Claim) –

wikibaseintegrator.datatypes.extra.localmedia

class wikibaseintegrator.datatypes.extra.localmedia.**LocalMedia**(*value=None*, ***kwargs*)

Bases: *String*

Implements the Wikibase data type for Wikibase Local Media extension. More info at https://www.mediawiki.org/wiki/Extension:Wikibase_Local_Media

Parameters

- **value** (*str | None*) –
- **kwargs** (*Any*) –

DTYPE = 'localMedia'

__init__(*value=None*, ***kwargs*)

Constructor, calls the superclass BaseDataType

Parameters

- **value** (Optional[str]) – The string to be used as the value
- **kwargs** (*Any*) –

equals(*that*, *include_ref=False*, *fref=None*)

Tests for equality of two statements. If comparing references, the order of the arguments matters!!! self is the current statement, the next argument is the new statement. Allows passing in a function to use to compare the references 'fref'. Default is equality. fref accepts two arguments 'oldrefs' and 'newrefs', each of which are a list of references, where each reference is a list of statements

Return type
bool

Parameters

- **that** (Claim) –
- **include_ref** (*bool*) –
- **fref** (*Callable | None*) –

from_json(*json_data*)

Parameters
json_data (Dict[str, Any]) – a JSON representation of a Claim

Return type
Claim

get_json()

Return type
Dict[str, Any]

get_sparql_value()

Return type
str

has_equal_qualifiers(*other*)

Return type
bool

Parameters
other (Claim) –

property id: str | None

property mainsnak: *Snak*

parse_sparql_value(*value*, *type='literal'*, *unit='1'*)

Return type
bool

property qualifiers: *Qualifiers*

property qualifiers_order: List[str]

property rank: *WikibaseRank*

property references: *References*

static refs_equal(*olditem*, *newitem*)
tests for exactly identical references

Return type
bool

Parameters
- **olditem** (Claim) –
- **newitem** (Claim) –

remove(*remove=True*)

Return type
None

property removed: bool

set_value(*value=None*)

Parameters
value (*str | None*) –

property type: str | Dict

update(*claim*)

Return type
None

Parameters
claim (Claim) –

1.1.1.2 Submodules

1.1.1.2.1 wikibaseintegrator.datatypes.basedatatype

class wikibaseintegrator.datatypes.basedatatype.**BaseDataType**(*prop_nr=None*, ***kwargs*)

Bases: *Claim*

The base class for all Wikibase data types, they inherit from it

Parameters

- **prop_nr** (*Optional[Union[int, str]]*) –
- **kwargs** (*Any*) –

DTYPE = 'base-data-type'

__init__(*prop_nr=None*, ***kwargs*)

Constructor, will be called by all data types.

Parameters

- **prop_nr** (Union[str, int, None]) – The property number a Wikibase snak belongs to
- **kwargs** (*Any*) –

equals(*that*, *include_ref=False*, *fref=None*)

Tests for equality of two statements. If comparing references, the order of the arguments matters!!! self is the current statement, the next argument is the new statement. Allows passing in a function to use to compare the references 'fref'. Default is equality. fref accepts two arguments 'oldrefs' and 'newrefs', each of which are a list of references, where each reference is a list of statements

Return type
bool

Parameters

- **that** (Claim) –
- **include_ref** (*bool*) –
- **fref** (*Callable | None*) –

from_json(*json_data*)

Parameters
json_data (Dict[str, Any]) – a JSON representation of a Claim

Return type
Claim

get_json()

Return type
Dict[str, Any]

get_sparql_value()

Return type
str

has_equal_qualifiers(*other*)

Return type
bool

Parameters
other (Claim) –

property id: str | None

property mainsnak: *Snak*

parse_sparql_value(*value*, *type='literal'*, *unit='1'*)

Return type
bool

property qualifiers: *Qualifiers*

property qualifiers_order: List[str]

property rank: *WikibaseRank*

property references: *References*

static refs_equal(*olditem*, *newitem*)
tests for exactly identical references

Return type
bool

Parameters
- **olditem** (Claim) –
- **newitem** (Claim) –

remove(*remove=True*)

Return type
None

property removed: bool

set_value(*value=None*)

Parameters
value (*Any | None*) –

property type: str | Dict

update(*claim*)

Return type
None

Parameters
claim (Claim) –

1.1.1.2.2 wikibaseintegrator.datatypes.commonsmedia

class wikibaseintegrator.datatypes.commonsmedia.**CommonsMedia**(*value=None*, ***kwargs*)

Bases: *String*

Implements the Wikibase data type for Wikimedia commons media files

Parameters

- **value** (*str | None*) –
- **kwargs** (*Any*) –

DTYPE = 'commonsMedia'

__init__(*value=None*, ***kwargs*)

Constructor, calls the superclass BaseDataType

Parameters

- **value** (Optional[str]) – The string to be used as the value
- **kwargs** (*Any*) –

equals(*that*, *include_ref=False*, *fref=None*)

Tests for equality of two statements. If comparing references, the order of the arguments matters!!! self is the current statement, the next argument is the new statement. Allows passing in a function to use to compare the references 'fref'. Default is equality. fref accepts two arguments 'oldrefs' and 'newrefs', each of which are a list of references, where each reference is a list of statements

Return type
bool

Parameters

- **that** (Claim) –
- **include_ref** (*bool*) –
- **fref** (*Callable | None*) –

from_json(*json_data*)

Parameters
json_data (Dict[str, Any]) – a JSON representation of a Claim

Return type
Claim

get_json()

Return type
Dict[str, Any]

get_sparql_value()

Return type
str

has_equal_qualifiers(*other*)

Return type
bool

Parameters
other (Claim) –

property id: str | None

property mainsnak: *Snak*

parse_sparql_value(*value*, *type='literal'*, *unit='1'*)

Return type
bool

property qualifiers: *Qualifiers*

property qualifiers_order: List[str]

property rank: *WikibaseRank*

property references: *References*

static refs_equal(*olditem*, *newitem*)

tests for exactly identical references

Return type
bool

Parameters

- **olditem** (Claim) –
- **newitem** (Claim) –

remove(*remove=True*)

Return type
None

property removed: bool

set_value(*value=None*)

Parameters
value (*str | None*) –

property type: str | Dict

update(*claim*)

Return type
None

Parameters
claim (Claim) –

1.1.1.2.3 wikibaseintegrator.datatypes.externalid

class wikibaseintegrator.datatypes.externalid.**ExternalID**(*value=None*, ***kwargs*)

Bases: *String*

Implements the Wikibase data type 'external-id'

Parameters

- **value** (*str | None*) –
- **kwargs** (*Any*) –

DTYPE = 'external-id'

__init__(*value=None*, ***kwargs*)

Constructor, calls the superclass BaseDataType

Parameters

- **value** (Optional[str]) – The string to be used as the value
- **kwargs** (*Any*) –

equals(*that*, *include_ref=False*, *fref=None*)

Tests for equality of two statements. If comparing references, the order of the arguments matters!!! self is the current statement, the next argument is the new statement. Allows passing in a function to use to compare the references 'fref'. Default is equality. fref accepts two arguments 'oldrefs' and 'newrefs', each of which are a list of references, where each reference is a list of statements

Return type
bool

Parameters

- **that** (Claim) –
- **include_ref** (*bool*) –
- **fref** (*Callable | None*) –

from_json(*json_data*)

Parameters
json_data (Dict[str, Any]) – a JSON representation of a Claim

Return type
Claim

get_json()

Return type
Dict[str, Any]

get_sparql_value()

Return type
str

has_equal_qualifiers(*other*)

Return type
bool

Parameters
other (Claim) –

property id: str | None

property mainsnak: *Snak*

parse_sparql_value(*value*, *type='literal'*, *unit='1'*)

Return type
bool

property qualifiers: *Qualifiers*

property qualifiers_order: List[str]

property rank: *WikibaseRank*

property references: *References*

static refs_equal(*olditem*, *newitem*)

tests for exactly identical references

Return type
bool

Parameters

- **olditem** (Claim) –
- **newitem** (Claim) –

remove(*remove=True*)

Return type
None

property removed: bool

set_value(*value=None*)

Parameters
value (*str | None*) –

property type: str | Dict

update(*claim*)

Return type
None

Parameters
claim (Claim) –

1.1.1.2.4 wikibaseintegrator.datatypes.form

class wikibaseintegrator.datatypes.form.**Form**(*value=None*, ***kwargs*)

Bases: *BaseDataType*

Implements the Wikibase data type 'wikibase-form'

Parameters

- **value** (*str | None*) –
- **kwargs** (*Any*) –

DTYPE = 'wikibase-form'

__init__(*value=None*, ***kwargs*)

Constructor, calls the superclass BaseDataType

Parameters

- **value** (*str with the format "L<Lexeme ID>-F<Form>"*) – The form number to serve as a value using the format "L<Lexeme ID>-F<Form ID>" (example: L252248-F2)
- **prop_nr** (*str with a 'P' prefix followed by digits*) – The property number for this claim
- **snaktype** (*str*) – The snak type, either 'value', 'somevalue' or 'novalue'
- **references** (*A data type with subclass of BaseDataType*) – List with reference objects
- **qualifiers** (*A data type with subclass of BaseDataType*) – List with qualifier objects
- **rank** (*str*) – rank of a snak with value 'preferred', 'normal' or 'deprecated'
- **kwargs** (*Any*) –

equals(*that*, *include_ref=False*, *fref=None*)

Tests for equality of two statements. If comparing references, the order of the arguments matters!!! self is the current statement, the next argument is the new statement. Allows passing in a function to use to compare the references 'fref'. Default is equality. fref accepts two arguments 'oldrefs' and 'newrefs', each of which are a list of references, where each reference is a list of statements

Return type
bool

Parameters

- **that** (*Claim*) –
- **include_ref** (*bool*) –
- **fref** (*Callable | None*) –

from_json(*json_data*)

Parameters
json_data (Dict[str, Any]) – a JSON representation of a Claim

Return type
Claim

get_json()

Return type
Dict[str, Any]

get_sparql_value()

Return type
str

has_equal_qualifiers(*other*)

Return type
bool

Parameters
other (Claim) –

property id: str | None

property mainsnak: *Snak*

parse_sparql_value(*value*, *type='literal'*, *unit='1'*)

Return type
bool

property qualifiers: *Qualifiers*

property qualifiers_order: List[str]

property rank: *WikibaseRank*

property references: *References*

static refs_equal(*olditem*, *newitem*)

tests for exactly identical references

Return type
bool

Parameters

- **olditem** (Claim) –
- **newitem** (Claim) –

remove(*remove=True*)

Return type
None

property removed: bool

set_value(*value=None*)

Parameters
value (*str | None*) –

property type: str | Dict

update(*claim*)

Return type
None

Parameters
claim (Claim) –

1.1.1.2.5 wikibaseintegrator.datatypes.geoshape

class wikibaseintegrator.datatypes.geoshape.**GeoShape**(*value=None*, ***kwargs*)

Bases: *BaseDataType*

Implements the Wikibase data type 'geo-shape'

Parameters

- **value** (*str | None*) –
- **kwargs** (*Any*) –

DTYPE = 'geo-shape'

__init__(*value=None*, ***kwargs*)

Constructor, calls the superclass BaseDataType

Parameters

- **value** (Optional[str]) – The GeoShape map file name in Wikimedia Commons to be linked
- **kwargs** (Any) –

Keyword Arguments

- *prop_nr* (str) – The item ID for this claim
- *snaktype* (str) – The snak type, either 'value', 'somevalue' or 'novalue'
- *references* (References or list of Claim) – List with reference objects
- *qualifiers* (Qualifiers) – List with qualifier objects
- *rank* (WikibaseRank) – The snak type, either 'value', 'somevalue' or 'novalue'

equals(*that*, *include_ref=False*, *fref=None*)

Tests for equality of two statements. If comparing references, the order of the arguments matters!!! self is the current statement, the next argument is the new statement. Allows passing in a function to use to compare the references 'fref'. Default is equality. fref accepts two arguments 'oldrefs' and 'newrefs', each of which are a list of references, where each reference is a list of statements

Return type
bool

Parameters

- **that** (Claim) –
- **include_ref** (*bool*) –
- **fref** (*Callable | None*) –

from_json(*json_data*)

Parameters
json_data (Dict[str, Any]) – a JSON representation of a Claim

Return type
Claim

get_json()

Return type
Dict[str, Any]

get_sparql_value()

Return type
str

has_equal_qualifiers(*other*)

Return type
bool

Parameters
other (Claim) –

property id: str | None

property mainsnak: *Snak*

parse_sparql_value(*value*, *type='literal'*, *unit='1'*)

Return type
bool

property qualifiers: *Qualifiers*

property qualifiers_order: List[str]

property rank: *WikibaseRank*

property references: *References*

static refs_equal(*olditem*, *newitem*)
tests for exactly identical references

Return type
bool

Parameters
- **olditem** (Claim) –
- **newitem** (Claim) –

remove(*remove=True*)

Return type
None

property removed: bool

set_value(*value=None*)

Parameters
value (*str | None*) –

property type: str | Dict

update(*claim*)

Return type
None

Parameters
claim (Claim) –

1.1.1.2.6 wikibaseintegrator.datatypes.globecoordinate

class wikibaseintegrator.datatypes.globecoordinate.**GlobeCoordinate**(*latitude=None*, *longitude=None*, *altitude=None*, *precision=None*, *globe=None*, *wikibase_url=None*, ***kwargs*)

Bases: *BaseDataType*

Implements the Wikibase data type for globe coordinates

Parameters

- **latitude** (*float | None*) –
- **longitude** (*float | None*) –
- **altitude** (*float | None*) –
- **precision** (*float | None*) –
- **globe** (*str | None*) –
- **wikibase_url** (*str | None*) –
- **kwargs** (*Any*) –

DTYPE = 'globe-coordinate'

__init__(*latitude=None*, *longitude=None*, *altitude=None*, *precision=None*, *globe=None*, *wikibase_url=None*, ***kwargs*)

Constructor, calls the superclass BaseDataType

Parameters

- **latitude** (Optional[float]) – Latitude in decimal format
- **longitude** (Optional[float]) – Longitude in decimal format
- **altitude** (Optional[float]) – Altitude (in decimal format?) (Always None at this moment)
- **precision** (Optional[float]) – Precision of the position measurement, default 1 / 3600

- **globe** (Optional[str]) – The globe entity concept URI (ex: http://www.wikidata.org/entity/Q2) or 'Q2'
- **wikibase_url** (Optional[str]) – The default wikibase URL, used when the globe is only an ID like 'Q2'. Use wbi_config['WIKIBASE_URL'] by default.
- **kwargs** (*Any*) –

equals(*that*, *include_ref=False*, *fref=None*)

Tests for equality of two statements. If comparing references, the order of the arguments matters!!! self is the current statement, the next argument is the new statement. Allows passing in a function to use to compare the references 'fref'. Default is equality. fref accepts two arguments 'oldrefs' and 'newrefs', each of which are a list of references, where each reference is a list of statements

Return type
bool

Parameters

- **that** (Claim) –
- **include_ref** (*bool*) –
- **fref** (*Callable | None*) –

from_json(*json_data*)

Parameters
json_data (Dict[str, Any]) – a JSON representation of a Claim

Return type
Claim

get_json()

Return type
Dict[str, Any]

get_sparql_value()

Return type
str

has_equal_qualifiers(*other*)

Return type
bool

Parameters
other (Claim) –

property id: str | None

property mainsnak: *Snak*

parse_sparql_value(*value*, *type='literal'*, *unit='1'*)

Return type
bool

property qualifiers: *Qualifiers*

property qualifiers_order: List[str]

property rank: *WikibaseRank*

property references: *References*

static refs_equal(*olditem*, *newitem*)

tests for exactly identical references

Return type
bool

Parameters

- **olditem** (Claim) –
- **newitem** (Claim) –

remove(*remove=True*)

Return type
None

property removed: bool

set_value(*latitude=None*, *longitude=None*, *altitude=None*, *precision=None*, *globe=None*, *wikibase_url=None*)

Parameters

- **latitude** (*float | None*) –
- **longitude** (*float | None*) –
- **altitude** (*float | None*) –
- **precision** (*float | None*) –
- **globe** (*str | None*) –
- **wikibase_url** (*str | None*) –

property type: str | Dict

update(*claim*)

Return type
None

Parameters
claim (Claim) –

1.1.1.2.7 wikibaseintegrator.datatypes.item

class wikibaseintegrator.datatypes.item.**Item**(*value=None*, ***kwargs*)

Bases: *BaseDataType*

Implements the Wikibase data type 'wikibase-item' with a value being another item ID

Parameters

- **value** (*str | int | None*) –
- **kwargs** (*Any*) –

DTYPE = 'wikibase-item'

__init__(*value=None*, ***kwargs*)

Constructor, calls the superclass BaseDataType

Parameters

- **value** (Union[str, int, None]) – The item ID to serve as the value
- **kwargs** (*Any*) –

equals(*that*, *include_ref=False*, *fref=None*)

Tests for equality of two statements. If comparing references, the order of the arguments matters!!! self is the current statement, the next argument is the new statement. Allows passing in a function to use to compare the references 'fref'. Default is equality. fref accepts two arguments 'oldrefs' and 'newrefs', each of which are a list of references, where each reference is a list of statements

Return type
bool

Parameters

- **that** (Claim) –
- **include_ref** (*bool*) –
- **fref** (*Callable | None*) –

from_json(*json_data*)

Parameters
json_data (Dict[str, Any]) – a JSON representation of a Claim

Return type
Claim

get_json()

Return type
Dict[str, Any]

get_sparql_value()

Return type
str

has_equal_qualifiers(*other*)

Return type
bool

Parameters
other (Claim) –

property id: str | None

property mainsnak: *Snak*

parse_sparql_value(*value*, *type='literal'*, *unit='1'*)

Return type
bool

property qualifiers: *Qualifiers*

property qualifiers_order: List[str]

property rank: *WikibaseRank*

property references: *References*

static refs_equal(*olditem*, *newitem*)

tests for exactly identical references

Return type
bool

Parameters

- **olditem** (Claim) –
- **newitem** (Claim) –

remove(*remove=True*)

Return type
None

property removed: bool

set_value(*value=None*)

Parameters
value (*str | int | None*) –

property type: str | Dict

update(*claim*)

Return type
None

Parameters
claim (Claim) –

1.1.1.2.8 wikibaseintegrator.datatypes.lexeme

class wikibaseintegrator.datatypes.lexeme.**Lexeme**(*value=None*, ***kwargs*)

Bases: *BaseDataType*

Implements the Wikibase data type 'wikibase-lexeme'

Parameters

- **value** (*str | int | None*) –
- **kwargs** (*Any*) –

DTYPE = 'wikibase-lexeme'

__init__(*value=None*, ***kwargs*)

Constructor, calls the superclass BaseDataType

Parameters

- **value** (Union[str, int, None]) – The lexeme number to serve as a value
- **kwargs** (*Any*) –

equals(*that*, *include_ref=False*, *fref=None*)

Tests for equality of two statements. If comparing references, the order of the arguments matters!!! self is the current statement, the next argument is the new statement. Allows passing in a function to use to compare the references 'fref'. Default is equality. fref accepts two arguments 'oldrefs' and 'newrefs', each of which are a list of references, where each reference is a list of statements

Return type
bool

Parameters

- **that** (Claim) –
- **include_ref** (*bool*) –
- **fref** (*Callable | None*) –

from_json(*json_data*)

Parameters
json_data (Dict[str, Any]) – a JSON representation of a Claim

Return type
Claim

get_json()

Return type
Dict[str, Any]

get_sparql_value()

Return type
str

has_equal_qualifiers(*other*)

Return type
bool

Parameters
other (Claim) –

property id: str | None

property mainsnak: *Snak*

parse_sparql_value(*value*, *type='literal'*, *unit='1'*)

Return type
bool

property qualifiers: *Qualifiers*

property qualifiers_order: List[str]

property rank: *WikibaseRank*

property references: *References*

static refs_equal(*olditem*, *newitem*)

tests for exactly identical references

Return type
bool

Parameters

- **olditem** (Claim) –
- **newitem** (Claim) –

remove(*remove=True*)

Return type
None

property removed: bool

set_value(*value=None*)

Parameters
value (*str | int | None*) –

property type: str | Dict

update(*claim*)

Return type
None

Parameters
claim (Claim) –

1.1.1.2.9 wikibaseintegrator.datatypes.math

class wikibaseintegrator.datatypes.math.**Math**(*value=None*, ***kwargs*)

Bases: *String*

Implements the Wikibase data type 'math' for mathematical formula in TEX format

Parameters

- **value** (*str | None*) –
- **kwargs** (*Any*) –

DTYPE = 'math'

__init__(*value=None*, ***kwargs*)

Constructor, calls the superclass BaseDataType

Parameters

- **value** (Optional[str]) – The string to be used as the value
- **kwargs** (*Any*) –

equals(*that*, *include_ref=False*, *fref=None*)

Tests for equality of two statements. If comparing references, the order of the arguments matters!!! self is the current statement, the next argument is the new statement. Allows passing in a function to use to compare the references 'fref'. Default is equality. fref accepts two arguments 'oldrefs' and 'newrefs', each of which are a list of references, where each reference is a list of statements

Return type
bool

Parameters

- **that** (Claim) –
- **include_ref** (*bool*) –
- **fref** (*Callable | None*) –

from_json(*json_data*)

Parameters
json_data (Dict[str, Any]) – a JSON representation of a Claim

Return type
Claim

get_json()

Return type
Dict[str, Any]

get_sparql_value()

Return type
str

has_equal_qualifiers(*other*)

Return type
bool

Parameters
other (Claim) –

property id: str | None

property mainsnak: *Snak*

parse_sparql_value(*value*, *type='literal'*, *unit='1'*)

Return type
bool

property qualifiers: *Qualifiers*

property qualifiers_order: List[str]

property rank: *WikibaseRank*

property references: *References*

static refs_equal(*olditem*, *newitem*)

tests for exactly identical references

Return type
bool

Parameters

- **olditem** (Claim) –
- **newitem** (Claim) –

remove(*remove=True*)

Return type
None

property removed: bool

set_value(*value=None*)

Parameters
value (*str | None*) –

property type: str | Dict

update(*claim*)

Return type
None

Parameters
claim (Claim) –

1.1.1.2.10 wikibaseintegrator.datatypes.monolingualtext

class wikibaseintegrator.datatypes.monolingualtext.**MonolingualText**(*text=None*, *language=None*, ***kwargs*)

Bases: BaseDataType

Implements the Wikibase data type for Monolingual Text strings

Parameters

- **text** (*str | None*) –
- **language** (*str | None*) –
- **kwargs** (*Any*) –

DTYPE = 'monolingualtext'

__init__(*text=None*, *language=None*, ***kwargs*)

Constructor, calls the superclass BaseDataType

Parameters

- **text** (Optional[str]) – The language specific string to be used as the value.
- **language** (Optional[str]) – Specifies the language the value belongs to.
- **kwargs** (*Any*) –

equals(*that*, *include_ref=False*, *fref=None*)

Tests for equality of two statements. If comparing references, the order of the arguments matters!!! self is the current statement, the next argument is the new statement. Allows passing in a function to use to compare the references 'fref'. Default is equality. fref accepts two arguments 'oldrefs' and 'newrefs', each of which are a list of references, where each reference is a list of statements

Return type
bool

Parameters

- **that** (Claim) –
- **include_ref** (*bool*) –
- **fref** (*Callable | None*) –

from_json(*json_data*)

Parameters
json_data (Dict[str, Any]) – a JSON representation of a Claim

Return type
Claim

get_json()

Return type
Dict[str, Any]

get_sparql_value()

Return type
str

has_equal_qualifiers(*other*)

Return type
bool

Parameters
other (Claim) –

property id: str | None

property mainsnak: *Snak*

parse_sparql_value(*value*, *type='literal'*, *unit='1'*)

Return type
bool

property qualifiers: *Qualifiers*

property qualifiers_order: List[str]

property rank: *WikibaseRank*

property references: *References*

static refs_equal(*olditem*, *newitem*)

tests for exactly identical references

Return type
bool

Parameters

- **olditem** (Claim) –
- **newitem** (Claim) –

remove(*remove=True*)

Return type
None

property removed: bool

set_value(*text=None*, *language=None*)

Parameters

- **text** (*str | None*) –
- **language** (*str | None*) –

property type: str | Dict

update(*claim*)

Return type
None

Parameters
claim (Claim) –

1.1.1.2.11 wikibaseintegrator.datatypes.musicalnotation

class wikibaseintegrator.datatypes.musicalnotation.**MusicalNotation**(*value=None*, ***kwargs*)

Bases: *String*

Implements the Wikibase data type 'musical-notation'

Parameters

- **value** (*str | None*) –
- **kwargs** (*Any*) –

DTYPE = 'musical-notation'

__init__(*value=None*, ***kwargs*)

Constructor, calls the superclass BaseDataType

Parameters

- **value** (Optional[str]) – The string to be used as the value
- **kwargs** (*Any*) –

equals(*that*, *include_ref=False*, *fref=None*)

Tests for equality of two statements. If comparing references, the order of the arguments matters!!! self is the current statement, the next argument is the new statement. Allows passing in a function to use to compare the references 'fref'. Default is equality. fref accepts two arguments 'oldrefs' and 'newrefs', each of which are a list of references, where each reference is a list of statements

Return type
bool

Parameters

- **that** (Claim) –
- **include_ref** (*bool*) –
- **fref** (*Callable | None*) –

from_json(*json_data*)

Parameters
json_data (Dict[str, Any]) – a JSON representation of a Claim

Return type
Claim

get_json()

Return type
Dict[str, Any]

get_sparql_value()

Return type
str

has_equal_qualifiers(*other*)

Return type
bool

Parameters
other (Claim) –

property id: str | None

property mainsnak: *Snak*

parse_sparql_value(*value*, *type='literal'*, *unit='1'*)

Return type
bool

property qualifiers: *Qualifiers*

property qualifiers_order: List[str]

property rank: *WikibaseRank*

property references: *References*

static refs_equal(*olditem*, *newitem*)

tests for exactly identical references

Return type
bool

Parameters

- **olditem** (Claim) –
- **newitem** (Claim) –

remove(*remove=True*)

Return type
None

property removed: bool

set_value(*value=None*)

Parameters
value (*str | None*) –

property type: str | Dict

update(*claim*)

Return type
None

Parameters
claim (Claim) –

1.1.1.2.12 wikibaseintegrator.datatypes.property

class wikibaseintegrator.datatypes.property.**Property**(*value=None*, ***kwargs*)

Bases: *BaseDataType*

Implements the Wikibase data type 'property'

Parameters

- **value** (*str | int | None*) –
- **kwargs** (*Any*) –

DTYPE = 'wikibase-property'

__init__(*value=None*, ***kwargs*)

Constructor, calls the superclass BaseDataType

Parameters

- **value** (*str with a 'P' prefix, followed by several digits or only the digits without the 'P' prefix*) – The property number to serve as a value
- **kwargs** (*Any*) –

equals(*that*, *include_ref=False*, *fref=None*)

Tests for equality of two statements. If comparing references, the order of the arguments matters!!! self is the current statement, the next argument is the new statement. Allows passing in a function to use to compare the references 'fref'. Default is equality. fref accepts two arguments 'oldrefs' and 'newrefs', each of which are a list of references, where each reference is a list of statements

Return type
bool

Parameters

- **that** (Claim) –
- **include_ref** (*bool*) –
- **fref** (*Callable | None*) –

from_json(*json_data*)

Parameters
json_data (Dict[str, Any]) – a JSON representation of a Claim

Return type
Claim

get_json()

Return type
Dict[str, Any]

get_sparql_value()

Return type
str

has_equal_qualifiers(*other*)

Return type
bool

Parameters
other (Claim) –

property id: str | None

property mainsnak: *Snak*

parse_sparql_value(*value*, *type='literal'*, *unit='1'*)

Return type
bool

property qualifiers: *Qualifiers*

property qualifiers_order: List[str]

property rank: *WikibaseRank*

property references: *References*

static refs_equal(*olditem*, *newitem*)

tests for exactly identical references

Return type
bool

Parameters

- **olditem** (Claim) –
- **newitem** (Claim) –

remove(*remove=True*)

Return type
None

property removed: bool

set_value(*value=None*)

Parameters
value (*str | int | None*) –

property type: str | Dict

update(*claim*)

Return type
None

Parameters
claim (Claim) –

1.1.1.2.13 wikibaseintegrator.datatypes.quantity

class wikibaseintegrator.datatypes.quantity.**Quantity**(*amount=None*, *upper_bound=None*, *lower_bound=None*, *unit='1'*, *wikibase_url=None*, ***kwargs*)

Bases: *BaseDataType*

Implements the Wikibase data type for quantities

Parameters

- **amount** (*str | int | float | None*) –
- **upper_bound** (*str | int | float | None*) –
- **lower_bound** (*str | int | float | None*) –
- **unit** (*str | int*) –
- **wikibase_url** (*str | None*) –
- **kwargs** (*Any*) –

DTYPE = 'quantity'

__init__(*amount=None*, *upper_bound=None*, *lower_bound=None*, *unit='1'*, *wikibase_url=None*, ***kwargs*)

Constructor, calls the superclass BaseDataType

Parameters

- **amount** (Union[str, int, float, None]) – The amount value
- **upper_bound** (Union[str, int, float, None]) – Upper bound of the value if it exists, e.g. for standard deviations
- **lower_bound** (Union[str, int, float, None]) – Lower bound of the value if it exists, e.g. for standard deviations
- **unit** (Union[str, int]) – The unit item URL or the QID a certain amount has been measured in (https://www.wikidata.org/wiki/Wikidata:Units). The default is dimensionless, represented by a '1'
- **wikibase_url** (Optional[str]) – The default wikibase URL, used when the unit is only an ID like 'Q2'. Use wbi_config['WIKIBASE_URL'] by default.
- **kwargs** (*Any*) –

equals(*that*, *include_ref=False*, *fref=None*)

Tests for equality of two statements. If comparing references, the order of the arguments matters!!! self is the current statement, the next argument is the new statement. Allows passing in a function to use to compare the references 'fref'. Default is equality. fref accepts two arguments 'oldrefs' and 'newrefs', each of which are a list of references, where each reference is a list of statements

Return type
bool

Parameters

- **that** (Claim) –
- **include_ref** (*bool*) –
- **fref** (*Callable | None*) –

from_json(*json_data*)

Parameters
json_data (Dict[str, Any]) – a JSON representation of a Claim

Return type
Claim

get_json()

Return type
Dict[str, Any]

get_sparql_value()

Return type
str

has_equal_qualifiers(*other*)

Return type
bool

Parameters
other (Claim) –

property id: str | None

property mainsnak: *Snak*

parse_sparql_value(*value*, *type='literal'*, *unit='1'*)

Return type
bool

property qualifiers: *Qualifiers*

property qualifiers_order: List[str]

property rank: *WikibaseRank*

property references: *References*

static refs_equal(*olditem*, *newitem*)

tests for exactly identical references

Return type
bool

Parameters

- **olditem** (Claim) –
- **newitem** (Claim) –

remove(*remove=True*)

Return type
None

property removed: bool

set_value(*amount=None*, *upper_bound=None*, *lower_bound=None*, *unit='1'*, *wikibase_url=None*)

Parameters

- **amount** (*str | int | float | None*) –
- **upper_bound** (*str | int | float | None*) –
- **lower_bound** (*str | int | float | None*) –
- **unit** (*str | int*) –
- **wikibase_url** (*str | None*) –

property type: str | Dict

update(*claim*)

Return type
None

Parameters
claim (Claim) –

1.1.1.2.14 wikibaseintegrator.datatypes.sense

class wikibaseintegrator.datatypes.sense.**Sense**(*value=None*, ***kwargs*)

Bases: *BaseDataType*

Implements the Wikibase data type 'wikibase-sense'

Parameters

- **value** (*str | None*) –
- **kwargs** (*Any*) –

DTYPE = 'wikibase-sense'

__init__(*value=None*, ***kwargs*)

Constructor, calls the superclass BaseDataType

Parameters

- **value** (Optional[str]) – Value using the format "L<Lexeme ID>-S<Sense ID>" (example: L252248-S123)
- **kwargs** (*Any*) –

equals(*that*, *include_ref=False*, *fref=None*)

Tests for equality of two statements. If comparing references, the order of the arguments matters!!! self is the current statement, the next argument is the new statement. Allows passing in a function to use to compare the references 'fref'. Default is equality. fref accepts two arguments 'oldrefs' and 'newrefs', each of which are a list of references, where each reference is a list of statements

Return type
bool

Parameters

- **that** (Claim) –
- **include_ref** (*bool*) –
- **fref** (*Callable | None*) –

from_json(*json_data*)

Parameters
json_data (Dict[str, Any]) – a JSON representation of a Claim

Return type
Claim

get_json()

Return type
Dict[str, Any]

get_sparql_value()

Return type
str

has_equal_qualifiers(*other*)

Return type
bool

Parameters
other (Claim) –

property id: str | None

property mainsnak: *Snak*

parse_sparql_value(*value*, *type='literal'*, *unit='1'*)

Return type
bool

property qualifiers: *Qualifiers*

property qualifiers_order: List[str]

property rank: *WikibaseRank*

property references: *References*

static refs_equal(*olditem*, *newitem*)
tests for exactly identical references

Return type
bool

Parameters

- **olditem** (Claim) –
- **newitem** (Claim) –

remove(*remove=True*)

Return type
None

property removed: bool

set_value(*value=None*)

Parameters
value (*str | None*) –

property type: str | Dict

update(*claim*)

Return type
None

Parameters
claim (Claim) –

1.1.1.2.15 wikibaseintegrator.datatypes.string

class wikibaseintegrator.datatypes.string.**String**(*value=None*, ***kwargs*)

Bases: *BaseDataType*

Implements the Wikibase data type 'string'

Parameters

- **value** (*str | None*) –
- **kwargs** (*Any*) –

DTYPE = 'string'

__init__(*value=None*, ***kwargs*)

Constructor, calls the superclass BaseDataType

Parameters

- **value** (Optional[str]) – The string to be used as the value
- **kwargs** (*Any*) –

equals(*that*, *include_ref=False*, *fref=None*)

Tests for equality of two statements. If comparing references, the order of the arguments matters!!! self is the current statement, the next argument is the new statement. Allows passing in a function to use to compare the references 'fref'. Default is equality. fref accepts two arguments 'oldrefs' and 'newrefs', each of which are a list of references, where each reference is a list of statements

Return type
bool

Parameters

- **that** (Claim) –
- **include_ref** (*bool*) –
- **fref** (*Callable | None*) –

from_json(*json_data*)

Parameters
json_data (Dict[str, Any]) – a JSON representation of a Claim

Return type
Claim

get_json()

Return type
Dict[str, Any]

get_sparql_value()

Return type
str

has_equal_qualifiers(*other*)

Return type
bool

Parameters
other (Claim) –

property id: str | None

property mainsnak: *Snak*

parse_sparql_value(*value*, *type='literal'*, *unit='1'*)

Return type
bool

property qualifiers: *Qualifiers*

property qualifiers_order: List[str]

property rank: *WikibaseRank*

property references: *References*

static refs_equal(*olditem*, *newitem*)
tests for exactly identical references

Return type
bool

Parameters

- **olditem** (Claim) –
- **newitem** (Claim) –

remove(*remove=True*)

Return type
None

property removed: bool

set_value(*value=None*)

Parameters
value (*str | None*) –

property type: str | Dict

update(*claim*)

Return type
None

Parameters
claim (Claim) –

1.1.1.2.16 wikibaseintegrator.datatypes.tabulardata

class wikibaseintegrator.datatypes.tabulardata.**TabularData**(*value=None*, ***kwargs*)

Bases: *BaseDataType*

Implements the Wikibase data type 'tabular-data'

Parameters

- **value** (*str | None*) –
- **kwargs** (*Any*) –

DTYPE = 'tabular-data'

__init__(*value=None*, ***kwargs*)

Constructor, calls the superclass BaseDataType

Parameters

- **value** (Optional[str]) – Reference to tabular data file on Wikimedia Commons.
- **kwargs** (*Any*) –

equals(*that*, *include_ref=False*, *fref=None*)

Tests for equality of two statements. If comparing references, the order of the arguments matters!!! self is the current statement, the next argument is the new statement. Allows passing in a function to use to compare the references 'fref'. Default is equality. fref accepts two arguments 'oldrefs' and 'newrefs', each of which are a list of references, where each reference is a list of statements

Return type
bool

Parameters

- **that** (Claim) –
- **include_ref** (*bool*) –
- **fref** (*Callable | None*) –

from_json(*json_data*)

Parameters
json_data (Dict[str, Any]) – a JSON representation of a Claim

Return type
Claim

get_json()

Return type
Dict[str, Any]

get_sparql_value()

Return type
str

has_equal_qualifiers(*other*)

Return type
bool

Parameters
other (Claim) –

property id: str | None

property mainsnak: *Snak*

parse_sparql_value(*value*, *type='literal'*, *unit='1'*)

Return type
bool

property qualifiers: *Qualifiers*

property qualifiers_order: List[str]

property rank: *WikibaseRank*

property references: *References*

static refs_equal(*olditem*, *newitem*)
tests for exactly identical references

Return type
bool

Parameters

- **olditem** (Claim) –
- **newitem** (Claim) –

remove(*remove=True*)

Return type
None

property removed: bool

set_value(*value=None*)

Parameters
value (*str | None*) –

property type: str | Dict

update(*claim*)

Return type
None

Parameters
claim (Claim) –

1.1.1.2.17 wikibaseintegrator.datatypes.time

class wikibaseintegrator.datatypes.time.**Time**(*time=None*, *before=0*, *after=0*, *precision=None*, *timezone=0*, *calendarmodel=None*, *wikibase_url=None*, ***kwargs*)

Bases: *BaseDataType*

Implements the Wikibase data type with date and time values

Parameters

- **time** (*str | None*) –
- **before** (*int*) –
- **after** (*int*) –
- **precision** (*int | WikibaseDatePrecision | None*) –
- **timezone** (*int*) –
- **calendarmodel** (*str | None*) –
- **wikibase_url** (*str | None*) –
- **kwargs** (*Any*) –

DTYPE = 'time'

__init__(*time=None*, *before=0*, *after=0*, *precision=None*, *timezone=0*, *calendarmodel=None*, *wikibase_url=None*, ***kwargs*)

Constructor, calls the superclass BaseDataType

Parameters

- **time** (Optional[str]) – Explicit value for point in time, represented as a timestamp resembling ISO 8601. You can use the keyword 'now' to get the current UTC date.
- **prop_nr** – The property number for this claim
- **before** (int) – explicit integer value for how many units after the given time it could be. The unit is given by the precision.
- **after** (int) – explicit integer value for how many units before the given time it could be. The unit is given by the precision.
- **precision** (Union[int, *WikibaseDatePrecision*, None]) – Precision value for dates and time as specified in the Wikibase data model (https://www.wikidata.org/wiki/Special:ListDatatypes#time)
- **timezone** (int) – The timezone which applies to the date and time as specified in the Wikibase data model
- **calendarmodel** (Optional[str]) – The calendar model used for the date. URL to the Wikibase calendar model item or the QID.
- **wikibase_url** (*str | None*) –
- **kwargs** (*Any*) –

equals(*that*, *include_ref=False*, *fref=None*)

Tests for equality of two statements. If comparing references, the order of the arguments matters!!! self is the current statement, the next argument is the new statement. Allows passing in a function to use to

compare the references 'fref'. Default is equality. fref accepts two arguments 'oldrefs' and 'newrefs', each of which are a list of references, where each reference is a list of statements

Return type
bool

Parameters

- **that** (Claim) –
- **include_ref** (*bool*) –
- **fref** (*Callable | None*) –

from_json(*json_data*)

Parameters
json_data (Dict[str, Any]) – a JSON representation of a Claim

Return type
Claim

get_day()

Return type
int

get_json()

Return type
Dict[str, Any]

get_month()

Return type
int

get_sparql_value()

Return type
str

get_year()

Return type
int

has_equal_qualifiers(*other*)

Return type
bool

Parameters
other (Claim) –

property id: str | None

property mainsnak: *Snak*

parse_sparql_value(*value*, *type='literal'*, *unit='1'*)

Return type
bool

property qualifiers: *Qualifiers*

property qualifiers_order: List[str]

property rank: *WikibaseRank*

property references: *References*

static refs_equal(*olditem*, *newitem*)

tests for exactly identical references

Return type
bool

Parameters

- **olditem** (Claim) –
- **newitem** (Claim) –

remove(*remove=True*)

Return type
None

property removed: bool

set_value(*time=None*, *before=0*, *after=0*, *precision=None*, *timezone=0*, *calendarmodel=None*, *wikibase_url=None*)

Parameters

- **time** (*str | None*) –
- **before** (*int*) –
- **after** (*int*) –
- **precision** (*int | WikibaseDatePrecision | None*) –
- **timezone** (*int*) –
- **calendarmodel** (*str | None*) –
- **wikibase_url** (*str | None*) –

property type: str | Dict

update(*claim*)

Return type
None

Parameters
claim (Claim) –

1.1.1.2.18 wikibaseintegrator.datatypes.url

class wikibaseintegrator.datatypes.url.**URL**(*value=None*, ***kwargs*)

Bases: *BaseDataType*

Implements the Wikibase data type for URL strings

Parameters

- **value** (*str | None*) –
- **kwargs** (*Any*) –

DTYPE = 'url'

__init__(*value=None*, ***kwargs*)

Constructor, calls the superclass BaseDataType

Parameters

- **value** (Optional[str]) – The URL to be used as the value
- **kwargs** (*Any*) –

equals(*that*, *include_ref=False*, *fref=None*)

Tests for equality of two statements. If comparing references, the order of the arguments matters!!! self is the current statement, the next argument is the new statement. Allows passing in a function to use to compare the references 'fref'. Default is equality. fref accepts two arguments 'oldrefs' and 'newrefs', each of which are a list of references, where each reference is a list of statements

Return type
bool

Parameters

- **that** (Claim) –
- **include_ref** (*bool*) –
- **fref** (*Callable | None*) –

from_json(*json_data*)

Parameters
json_data (Dict[str, Any]) – a JSON representation of a Claim

Return type
Claim

get_json()

Return type
Dict[str, Any]

get_sparql_value()

Return type
str

has_equal_qualifiers(*other*)

Return type
bool

Parameters
other (Claim) –

property id: str | None

property mainsnak: *Snak*

parse_sparql_value(*value*, *type='literal'*, *unit='1'*)

Return type
bool

property qualifiers: *Qualifiers*

property qualifiers_order: List[str]

property rank: *WikibaseRank*

property references: *References*

static refs_equal(*olditem*, *newitem*)
tests for exactly identical references

Return type
bool

Parameters

- **olditem** (Claim) –
- **newitem** (Claim) –

remove(*remove=True*)

Return type
None

property removed: bool

set_value(*value=None*)

Parameters
value (*str | None*) –

property type: str | Dict

update(*claim*)

Return type
None

Parameters
claim (Claim) –

1.1.2 wikibaseintegrator.entities

1.1.2.1 wikibaseintegrator.entities.baseentity

class wikibaseintegrator.entities.baseentity.**BaseEntity**(*api=None*, *title=None*, *pageid=None*, *lastrevid=None*, *type=None*, *id=None*, *claims=None*, *is_bot=None*, *login=None*)

Bases: `object`

Parameters

- **api** (*Optional['WikibaseIntegrator']*) –
- **title** (*Optional[str]*) –
- **pageid** (*Optional[int]*) –
- **lastrevid** (*Optional[int]*) –
- **type** (*Optional[str]*) –
- **id** (*Optional[str]*) –
- **claims** (*Optional[Claims]*) –
- **is_bot** (*Optional[bool]*) –
- **login** (*Optional[_Login]*) –

ETYPE = 'base-entity'

__init__(*api=None*, *title=None*, *pageid=None*, *lastrevid=None*, *type=None*, *id=None*, *claims=None*, *is_bot=None*, *login=None*)

Parameters

- **api** (*WikibaseIntegrator | None*) –
- **title** (*str | None*) –
- **pageid** (*int | None*) –
- **lastrevid** (*int | None*) –
- **type** (*str | None*) –
- **id** (*str | None*) –
- **claims** (*Claims | None*) –
- **is_bot** (*bool | None*) –
- **login** (*_Login | None*) –

add_claims(*claims*, *action_if_exists=ActionIfExists.APPEND_OR_REPLACE*)

Parameters

- **claims** (Union[*Claim*, List[*Claim*], *Claims*]) – A Claim, list of Claim or just a Claims object to add to this Claims object.
- **action_if_exists** (*ActionIfExists*) – Replace or append the statement. You can force an addition if the declaration already exists. KEEP: The original claim will be kept and the new one will not be added (because there is already one with this property number) APPEND_OR_REPLACE: The new claim will be added only if the new one is different

(by comparing values) FORCE_APPEND: The new claim will be added even if already exists REPLACE_ALL: The new claim will replace the old one

Return type
BaseEntity

Returns
Return the updated entity object.

property api: *WikibaseIntegrator*

property claims: *Claims*

clear(***kwargs*)

Use the *clear* parameter of *wbeditentity* API call to clear the content of the entity. The entity will be updated with an empty dictionary.

Parameters
kwargs (Any) – More arguments for _write() and Python requests

Return type
Dict[str, Any]

Returns
A dictionary representation of the edited Entity

delete(*login=None*, *allow_anonymous=False*, *is_bot=None*, ***kwargs*)

Delete the current entity. Use the pageid first if available and fallback to the page title.

Parameters

- **login** (Optional[_Login]) – A wbi_login.Login instance
- **allow_anonymous** (bool) – Allow an unidentified edit to the MediaWiki API (default False)
- **is_bot** (Optional[bool]) – Flag the edit as a bot
- **reason** – Reason for the deletion. If not set, an automatically generated reason will be used.
- **deletetalk** – Delete the talk page, if it exists.
- **kwargs** (Any) – Any additional keyword arguments to pass to mediawiki_api_call_helper and requests.request

Returns
The data returned by the API as a dictionary

from_json(*json_data*)

Import a dictionary into BaseEntity attributes.

Parameters
json_data (Dict[str, Any]) – A specific dictionary from MediaWiki API

Return type
BaseEntity

Returns

get_entity_url(*wikibase_url=None*)

Return type
str

Parameters
wikibase_url (*str | None*) –

get_json()

To get the dict equivalent of the JSON representation of the entity.

Return type
Dict[str, Union[str, Dict[str, List]]]

Returns

property id: str | None

property lastrevid: int | None

property pageid: str | int | None

property title: str | None

property type: str

write_required(*base_filter=None*, *action_if_exists=ActionIfExists.REPLACE_ALL*, ***kwargs*)

Return type
bool

Parameters

- **base_filter** (*Optional[List[BaseDataType | List[BaseDataType]]]*) –
- **action_if_exists** (ActionIfExists) –
- **kwargs** (*Any*) –

1.1.2.2 wikibaseintegrator.entities.item

class wikibaseintegrator.entities.item.**ItemEntity**(*labels=None*, *descriptions=None*, *aliases=None*, *sitelinks=None*, ***kwargs*)

Bases: *BaseEntity*

Parameters

- **labels** (*Optional[Labels]*) –
- **descriptions** (*Optional[Descriptions]*) –
- **aliases** (*Optional[Aliases]*) –
- **sitelinks** (*Optional[Sitelinks]*) –
- **kwargs** (*Any*) –

ETYPE = 'item'

__init__(*labels=None*, *descriptions=None*, *aliases=None*, *sitelinks=None*, ***kwargs*)

Parameters

- **api** –
- **labels** (Optional[*Labels*]) –
- **descriptions** (Optional[*Descriptions*]) –

- **aliases** (Optional[*Aliases*]) –
- **sitelinks** (Optional[*Sitelinks*]) –
- **kwargs** (Any) –

Return type
None

add_claims(*claims*, *action_if_exists=ActionIfExists.APPEND_OR_REPLACE*)

Parameters

- **claims** (Union[*Claim*, List[*Claim*], *Claims*]) – A Claim, list of Claim or just a Claims object to add to this Claims object.
- **action_if_exists** (*ActionIfExists*) – Replace or append the statement. You can force an addition if the declaration already exists. KEEP: The original claim will be kept and the new one will not be added (because there is already one with this property number) APPEND_OR_REPLACE: The new claim will be added only if the new one is different (by comparing values) FORCE_APPEND: The new claim will be added even if already exists REPLACE_ALL: The new claim will replace the old one

Return type
BaseEntity

Returns
Return the updated entity object.

property aliases: *Aliases*

property api: *WikibaseIntegrator*

property claims: *Claims*

clear(***kwargs*)

Use the *clear* parameter of *wbeditentity* API call to clear the content of the entity. The entity will be updated with an empty dictionary.

Parameters
kwargs (Any) – More arguments for _write() and Python requests

Return type
Dict[str, Any]

Returns
A dictionary representation of the edited Entity

delete(*login=None*, *allow_anonymous=False*, *is_bot=None*, ***kwargs*)

Delete the current entity. Use the pageid first if available and fallback to the page title.

Parameters

- **login** (Optional[_Login]) – A wbi_login.Login instance
- **allow_anonymous** (bool) – Allow an unidentified edit to the MediaWiki API (default False)
- **is_bot** (Optional[bool]) – Flag the edit as a bot
- **reason** – Reason for the deletion. If not set, an automatically generated reason will be used.
- **deletetalk** – Delete the talk page, if it exists.

- **kwargs** (Any) – Any additional keyword arguments to pass to mediawiki_api_call_helper and requests.request

Returns
The data returned by the API as a dictionary

property descriptions: *Descriptions*

from_json(*json_data*)

Import a dictionary into BaseEntity attributes.

Parameters
json_data (Dict[str, Any]) – A specific dictionary from MediaWiki API

Return type
ItemEntity

Returns

get(*entity_id=None*, ***kwargs*)

Request the MediaWiki API to get data for the entity specified in argument.

Parameters

- **entity_id** (Union[str, int, None]) – The entity_id of the Item entity you want. Must start with a 'Q'.
- **kwargs** (Any) –

Return type
ItemEntity

Returns
an ItemEntity instance

get_entity_url(*wikibase_url=None*)

Return type
str

Parameters
wikibase_url (*str | None*) –

get_json()

To get the dict equivalent of the JSON representation of the Item.

Return type
Dict[str, Union[str, Dict]]

Returns
A dict representation of the Item.

property id: str | None

property labels: *Labels*

property lastrevid: int | None

new(***kwargs*)

Return type
ItemEntity

Parameters
kwargs (*Any*) –

property pageid: str | int | None

property sitelinks: Sitelinks

property title: str | None

property type: str

write(**kwargs*)

Write the ItemEntity data to the Wikibase instance and return the ItemEntity object returned by the instance. extend `_write()`

Parameters

- **data** – The serialized object that is used as the data source. A newly created entity will be assigned an 'id'.
- **summary** – A summary of the edit
- **login** – A login instance
- **allow_anonymous** – Force a check if the query can be anonymous or not
- **clear** – Clear the existing entity before updating
- **is_bot** – Add the bot flag to the query
- **kwargs** (Any) – More arguments for Python requests

Return type
ItemEntity

Returns
an ItemEntity of the response from the instance

write_required(*base_filter=None*, *action_if_exists=ActionIfExists.REPLACE_ALL*, ***kwargs*)

Return type
bool

Parameters

- **base_filter** (*Optional[List[BaseDataType | List[BaseDataType]]]*) –
- **action_if_exists** (ActionIfExists) –
- **kwargs** (*Any*) –

1.1.2.3 wikibaseintegrator.entities.lexeme

class wikibaseintegrator.entities.lexeme.**LexemeEntity**(*lemmas=None*, *lexical_category=None*, *language=None*, *forms=None*, *senses=None*, ***kwargs*)

Bases: *BaseEntity*

Parameters

- **lemmas** (*Optional[Lemmas]*) –
- **lexical_category** (*Optional[str]*) –

- **language** (*Optional[str]*) –
- **forms** (*Optional[Forms]*) –
- **senses** (*Optional[Senses]*) –
- **kwargs** (*Any*) –

ETYPE = 'lexeme'

__init__(*lemmas=None*, *lexical_category=None*, *language=None*, *forms=None*, *senses=None*, ***kwargs*)

Parameters

- **lemmas** (Lemmas | *None*) –
- **lexical_category** (*str* | *None*) –
- **language** (*str* | *None*) –
- **forms** (Forms | *None*) –
- **senses** (Senses | *None*) –
- **kwargs** (*Any*) –

add_claims(*claims*, *action_if_exists=ActionIfExists.APPEND_OR_REPLACE*)

Parameters

- **claims** (Union[*Claim*, List[*Claim*], *Claims*]) – A Claim, list of Claim or just a Claims object to add to this Claims object.
- **action_if_exists** (*ActionIfExists*) – Replace or append the statement. You can force an addition if the declaration already exists. KEEP: The original claim will be kept and the new one will not be added (because there is already one with this property number) APPEND_OR_REPLACE: The new claim will be added only if the new one is different (by comparing values) FORCE_APPEND: The new claim will be added even if already exists REPLACE_ALL: The new claim will replace the old one

Return type
BaseEntity

Returns
Return the updated entity object.

property api: *WikibaseIntegrator*

property claims: *Claims*

clear(***kwargs*)

Use the *clear* parameter of *wbeditentity* API call to clear the content of the entity. The entity will be updated with an empty dictionary.

Parameters
kwargs (Any) – More arguments for _write() and Python requests

Return type
Dict[str, Any]

Returns
A dictionary representation of the edited Entity

delete(*login=None*, *allow_anonymous=False*, *is_bot=None*, ***kwargs*)

Delete the current entity. Use the pageid first if available and fallback to the page title.

Parameters

- **login** (Optional[_Login]) – A wbi_login.Login instance
- **allow_anonymous** (bool) – Allow an unidentified edit to the MediaWiki API (default False)
- **is_bot** (Optional[bool]) – Flag the edit as a bot
- **reason** – Reason for the deletion. If not set, an automatically generated reason will be used.
- **deletetalk** – Delete the talk page, if it exists.
- **kwargs** (Any) – Any additional keyword arguments to pass to mediawiki_api_call_helper and requests.request

Returns

The data returned by the API as a dictionary

property forms: *Forms*

from_json(*json_data*)

Import a dictionary into BaseEntity attributes.

Parameters

json_data (Dict[str, Any]) – A specific dictionary from MediaWiki API

Return type

LexemeEntity

Returns

get(*entity_id*, ***kwargs*)

Return type

LexemeEntity

Parameters

- **entity_id** (*str | int*) –
- **kwargs** (*Any*) –

get_entity_url(*wikibase_url=None*)

Return type

str

Parameters

wikibase_url (*str | None*) –

get_json()

To get the dict equivalent of the JSON representation of the entity.

Return type

Dict[str, Union[str, Dict]]

Returns

property id: **str | None**

property language: str

property lastrevid: int | None

property lemmas: *Lemmas*

property lexical_category: str | None

new(***kwargs*)

Return type
LexemeEntity

Parameters
kwargs (*Any*) –

property pageid: str | int | None

property senses: *Senses*

property title: str | None

property type: str

write(***kwargs*)

Write the LexemeEntity data to the Wikibase instance and return the LexemeEntity object returned by the instance.

Parameters

- **data** – The serialized object that is used as the data source. A newly created entity will be assigned an 'id'.
- **summary** – A summary of the edit
- **login** – A login instance
- **allow_anonymous** – Force a check if the query can be anonymous or not
- **clear** – Clear the existing entity before updating
- **is_bot** – Add the bot flag to the query
- **kwargs** (Any) – More arguments for Python requests

Return type
LexemeEntity

Returns
an LexemeEntity of the response from the instance

write_required(*base_filter=None*, *action_if_exists=ActionIfExists.REPLACE_ALL*, ***kwargs*)

Return type
bool

Parameters

- **base_filter** (*Optional[List[*BaseDataType *| List[*BaseDataType*]]]*) –
- **action_if_exists** (ActionIfExists) –
- **kwargs** (*Any*) –

1.1.2.4 wikibaseintegrator.entities.mediainfo

class wikibaseintegrator.entities.mediainfo.**MediaInfoEntity**(*labels=None*, *descriptions=None*, *aliases=None*, ***kwargs*)

Bases: *BaseEntity*

Parameters

- **labels** (*Optional[Labels]*) –
- **descriptions** (*Optional[Descriptions]*) –
- **aliases** (*Optional[Aliases]*) –
- **kwargs** (*Any*) –

ETYPE = 'mediainfo'

__init__(*labels=None*, *descriptions=None*, *aliases=None*, ***kwargs*)

Parameters

- **api** –
- **labels** (Optional[*Labels*]) –
- **descriptions** (Optional[*Descriptions*]) –
- **aliases** (Optional[*Aliases*]) –
- **sitelinks** –
- **kwargs** (Any) –

Return type
None

add_claims(*claims*, *action_if_exists=ActionIfExists.APPEND_OR_REPLACE*)

Parameters

- **claims** (Union[*Claim*, List[*Claim*], *Claims*]) – A Claim, list of Claim or just a Claims object to add to this Claims object.
- **action_if_exists** (*ActionIfExists*) – Replace or append the statement. You can force an addition if the declaration already exists. KEEP: The original claim will be kept and the new one will not be added (because there is already one with this property number) APPEND_OR_REPLACE: The new claim will be added only if the new one is different (by comparing values) FORCE_APPEND: The new claim will be added even if already exists REPLACE_ALL: The new claim will replace the old one

Return type
BaseEntity

Returns
Return the updated entity object.

property aliases: *Aliases*

property api: *WikibaseIntegrator*

property claims: *Claims*

clear(***kwargs*)

Use the *clear* parameter of *wbeditentity* API call to clear the content of the entity. The entity will be updated with an empty dictionary.

Parameters
kwargs (Any) – More arguments for _write() and Python requests

Return type
Dict[str, Any]

Returns
A dictionary representation of the edited Entity

delete(*login=None*, *allow_anonymous=False*, *is_bot=None*, ***kwargs*)

Delete the current entity. Use the pageid first if available and fallback to the page title.

Parameters

- **login** (Optional[_Login]) – A wbi_login.Login instance
- **allow_anonymous** (bool) – Allow an unidentified edit to the MediaWiki API (default False)
- **is_bot** (Optional[bool]) – Flag the edit as a bot
- **reason** – Reason for the deletion. If not set, an automatically generated reason will be used.
- **deletetalk** – Delete the talk page, if it exists.
- **kwargs** (Any) – Any additional keyword arguments to pass to mediawiki_api_call_helper and requests.request

Returns
The data returned by the API as a dictionary

property descriptions: *Descriptions*

from_json(*json_data*)

Import a dictionary into BaseEntity attributes.

Parameters
json_data (Dict[str, Any]) – A specific dictionary from MediaWiki API

Return type
MediaInfoEntity

Returns

get(*entity_id*, ***kwargs*)

Return type
MediaInfoEntity

Parameters

- **entity_id** (*str | int*) –
- **kwargs** (*Any*) –

get_by_title(*titles*, *sites='commonswiki'*, ***kwargs*)

Return type
MediaInfoEntity

Parameters

- **titles** (*List[str] | str*) –
- **sites** (*str*) –
- **kwargs** (*Any*) –

get_entity_url(*wikibase_url=None*)

Return type
str

Parameters
wikibase_url (*str | None*) –

get_json()

To get the dict equivalent of the JSON representation of the entity.

Return type
Dict[str, Union[str, Dict]]

Returns

property id: str | None

property labels: *Labels*

property lastrevid: int | None

new(***kwargs*)

Return type
MediaInfoEntity

Parameters
kwargs (*Any*) –

property pageid: str | int | None

property title: str | None

property type: str

write(***kwargs*)

Write the MediaInfoEntity data to the Wikibase instance and return the MediaInfoEntity object returned by the instance.

Parameters

- **data** – The serialized object that is used as the data source. A newly created entity will be assigned an 'id'.
- **summary** – A summary of the edit
- **login** – A login instance
- **allow_anonymous** – Force a check if the query can be anonymous or not
- **clear** – Clear the existing entity before updating
- **is_bot** – Add the bot flag to the query
- **kwargs** (Any) – More arguments for Python requests

Return type
MediaInfoEntity

Returns
an MediaInfoEntity of the response from the instance

write_required(*base_filter=None*, *action_if_exists=ActionIfExists.REPLACE_ALL*, ***kwargs*)

Return type
bool

Parameters

- **base_filter** (*Optional[List[BaseDataType | List[BaseDataType]]]*) –
- **action_if_exists** (*ActionIfExists*) –
- **kwargs** (*Any*) –

1.1.2.5 wikibaseintegrator.entities.property

class wikibaseintegrator.entities.property.**PropertyEntity**(*datatype=None*, *labels=None*, *descriptions=None*, *aliases=None*, ***kwargs*)

Bases: *BaseEntity*

Parameters

- **datatype** (*Union[str, WikibaseDatatype, None]*) –
- **labels** (*Optional[Labels]*) –
- **descriptions** (*Optional[Descriptions]*) –
- **aliases** (*Optional[Aliases]*) –
- **kwargs** (*Any*) –

ETYPE = 'property'

__init__(*datatype=None*, *labels=None*, *descriptions=None*, *aliases=None*, ***kwargs*)

Parameters

- **datatype** (*str | WikibaseDatatype | None*) –
- **labels** (*Labels | None*) –
- **descriptions** (*Descriptions | None*) –
- **aliases** (*Aliases | None*) –
- **kwargs** (*Any*) –

add_claims(*claims*, *action_if_exists=ActionIfExists.APPEND_OR_REPLACE*)

Parameters

- **claims** (Union[*Claim*, List[*Claim*], *Claims*]) – A Claim, list of Claim or just a Claims object to add to this Claims object.
- **action_if_exists** (*ActionIfExists*) – Replace or append the statement. You can force an addition if the declaration already exists. KEEP: The original claim will be kept and the new one will not be added (because there is already one with this property number)

APPEND_OR_REPLACE: The new claim will be added only if the new one is different (by comparing values) FORCE_APPEND: The new claim will be added even if already exists REPLACE_ALL: The new claim will replace the old one

Return type
BaseEntity

Returns
Return the updated entity object.

property aliases: *Aliases*

property api: *WikibaseIntegrator*

property claims: *Claims*

clear(***kwargs*)

Use the *clear* parameter of *wbeditentity* API call to clear the content of the entity. The entity will be updated with an empty dictionary.

Parameters
kwargs (Any) – More arguments for _write() and Python requests

Return type
Dict[str, Any]

Returns
A dictionary representation of the edited Entity

property datatype: **str | *WikibaseDatatype* | None**

delete(*login=None*, *allow_anonymous=False*, *is_bot=None*, ***kwargs*)

Delete the current entity. Use the pageid first if available and fallback to the page title.

Parameters

- **login** (Optional[_Login]) – A wbi_login.Login instance
- **allow_anonymous** (bool) – Allow an unidentified edit to the MediaWiki API (default False)
- **is_bot** (Optional[bool]) – Flag the edit as a bot
- **reason** – Reason for the deletion. If not set, an automatically generated reason will be used.
- **deletetalk** – Delete the talk page, if it exists.
- **kwargs** (Any) – Any additional keyword arguments to pass to mediawiki_api_call_helper and requests.request

Returns
The data returned by the API as a dictionary

property descriptions: *Descriptions*

from_json(*json_data*)

Import a dictionary into BaseEntity attributes.

Parameters
json_data (Dict[str, Any]) – A specific dictionary from MediaWiki API

Return type
PropertyEntity

Returns

get(*entity_id*, ***kwargs*)

Return type
PropertyEntity

Parameters

- **entity_id** (*str | int*) –
- **kwargs** (*Any*) –

get_entity_url(*wikibase_url=None*)

Return type
str

Parameters
wikibase_url (*str | None*) –

get_json()

To get the dict equivalent of the JSON representation of the entity.

Return type
Dict[str, Union[str, Any]]

Returns

property id: str | None

property labels: *Labels*

property lastrevid: int | None

new(***kwargs*)

Return type
PropertyEntity

Parameters
kwargs (*Any*) –

property pageid: str | int | None

property title: str | None

property type: str

write(***kwargs*)

Write the PropertyEntity data to the Wikibase instance and return the PropertyEntity object returned by the instance.

Parameters

- **data** – The serialized object that is used as the data source. A newly created entity will be assigned an 'id'.
- **summary** – A summary of the edit
- **login** – A login instance
- **allow_anonymous** – Force a check if the query can be anonymous or not
- **clear** – Clear the existing entity before updating

- **is_bot** – Add the bot flag to the query
- **kwargs** (Any) – More arguments for Python requests

Return type
PropertyEntity

Returns
an PropertyEntity of the response from the instance

write_required(*base_filter=None*, *action_if_exists=ActionIfExists.REPLACE_ALL*, ***kwargs*)

Return type
bool

Parameters

- **base_filter** (*Optional[List[BaseDataType | List[BaseDataType]]]*) –
- **action_if_exists** (*ActionIfExists*) –
- **kwargs** (*Any*) –

1.1.3 wikibaseintegrator.models

1.1.3.1 wikibaseintegrator.models.aliases

class wikibaseintegrator.models.aliases.**Alias**(*language*, *value=None*)

Bases: *LanguageValue*

Parameters

- **language** (*str*) –
- **value** (*Optional[str]*) –

__init__(*language*, *value=None*)

Parameters

- **language** (*str*) –
- **value** (*str | None*) –

from_json(*json_data*)

Return type
LanguageValue

Parameters
json_data (*Dict[str, str]*) –

get_json()

Return type
Dict[str, Optional[str]]

property language: str

remove()

Return type
LanguageValue

property removed: bool

property value: str | None

The value of the LanguageValue instance. :return: A string with the value of the LanguageValue instance.

class wikibaseintegrator.models.aliases.**Aliases**(*language=None*, *value=None*)

Bases: *BaseModel*

Parameters

- **language** (*Optional[str]*) –
- **value** (*Optional[str]*) –

__init__(*language=None*, *value=None*)

Parameters

- **language** (*str | None*) –
- **value** (*str | None*) –

property aliases: Dict[str, List[*Alias*]]

from_json(*json_data*)

Return type
Aliases

Parameters
json_data (*Dict[str, List]*) –

get(*language=None*)

Return type
Optional[List[*Alias*]]

Parameters
language (*str | None*) –

get_json()

Return type
Dict[str, List]

set(*language=None*, *values=None*, *action_if_exists=ActionIfExists.APPEND_OR_REPLACE*)

Return type
Aliases

Parameters

- **language** (*str | None*) –
- **values** (*str | List | None*) –
- **action_if_exists** (ActionIfExists) –

1.1.3.2 wikibaseintegrator.models.basemodel

class wikibaseintegrator.models.basemodel.**BaseModel**

Bases: object

__init__()

1.1.3.3 wikibaseintegrator.models.claims

class wikibaseintegrator.models.claims.**Claim**(*qualifiers=None*, *rank=None*, *references=None*, *snaktype=WikibaseSnakType.KNOWN_VALUE*)

Bases: *BaseModel*

Parameters

- **qualifiers** (*Optional[Qualifiers]*) –
- **rank** (*Optional[WikibaseRank]*) –
- **references** (*Optional[Union[References, List[Union[Claim, List[Claim]]]]]*) –
- **snaktype** (*WikibaseSnakType*) –

DTYPE = 'claim'

__init__(*qualifiers=None*, *rank=None*, *references=None*, *snaktype=WikibaseSnakType.KNOWN_VALUE*)

Parameters

- **qualifiers** (Optional[*Qualifiers*]) –
- **rank** (Optional[*WikibaseRank*]) –
- **references** (Union[*References*, List[Union[*Claim*, List[*Claim*]]], None]) – A References object, a list of Claim object or a list of list of Claim object
- **snaktype** (*WikibaseSnakType*) –

Return type
None

equals(*that*, *include_ref=False*, *fref=None*)

Tests for equality of two statements. If comparing references, the order of the arguments matters!!! self is the current statement, the next argument is the new statement. Allows passing in a function to use to compare the references 'fref'. Default is equality. fref accepts two arguments 'oldrefs' and 'newrefs', each of which are a list of references, where each reference is a list of statements

Return type
bool

Parameters

- **that** (*Claim*) –
- **include_ref** (*bool*) –
- **fref** (*Callable | None*) –

from_json(*json_data*)

Parameters
json_data (Dict[str, Any]) – a JSON representation of a Claim

Return type
Claim

get_json()

Return type
Dict[str, Any]

abstract get_sparql_value()

Return type
str

has_equal_qualifiers(*other*)

Return type
bool

Parameters
other (Claim) –

property id: str | None

property mainsnak: *Snak*

property qualifiers: *Qualifiers*

property qualifiers_order: List[str]

property rank: *WikibaseRank*

property references: *References*

static refs_equal(*olditem*, *newitem*)

tests for exactly identical references

Return type
bool

Parameters

- **olditem** (Claim) –
- **newitem** (Claim) –

remove(*remove=True*)

Return type
None

property removed: bool

property type: str | Dict

update(*claim*)

Return type
None

Parameters
claim (Claim) –

class wikibaseintegrator.models.claims.**Claims**

Bases: *BaseModel*

__init__()

Return type
None

add(*claims*, *action_if_exists=ActionIfExists.REPLACE_ALL*)

Parameters

- **claims** (Union[*Claims*, List[*Claim*], *Claim*]) – A Claim, list of Claim or just a Claims object to add to this Claims object.
- **action_if_exists** (*ActionIfExists*) – Replace or append the statement. You can force an addition if the declaration already exists. Defaults to REPLACE_ALL. KEEP: The original claim will be kept and the new one will not be added (because there is already one with this property number) APPEND_OR_REPLACE: The new claim will be added only if the new one is different (by comparing values) FORCE_APPEND: The new claim will be added even if already exists REPLACE_ALL: The new claim will replace the old one

Return type
Claims

Returns
Return the updated Claims object.

property claims: Dict[str, List[*Claim*]]

from_json(*json_data*)

Return type
Claims

Parameters
json_data (*Dict[str, Any]*) –

get(*property*)

Return type
List[*Claim*]

Parameters
property (*str*) –

get_json()

Return type
Dict[str, List]

remove(*property=None*)

Return type
None

Parameters
property (*str | None*) –

1.1.3.4 wikibaseintegrator.models.descriptions

class wikibaseintegrator.models.descriptions.**Descriptions**

Bases: *LanguageValues*

__init__()

Return type
None

add(*language_value*)

Add a LanguageValue object to the list

Parameters
language_value (*LanguageValue*) – A LanguageValue object

Return type
LanguageValues

Returns
The current LanguageValues object

from_json(*json_data*)

Create a new Descriptions object from a JSON/dict object.

Parameters
json_data (Dict[str, Dict]) – A dict object who use the same format as Wikibase.

Return type
Descriptions

Returns
The newly created or updated object.

get(*language=None*)

Get a LanguageValue object with the specified language. Use the default language in wbi_config if none specified.

Parameters
language (Optional[str]) – The requested language.

Return type
Optional[*LanguageValue*]

Returns
The related LanguageValue object or None if none found.

get_json()

Get a formatted dict who respect the Wikibase format.

Return type
Dict[str, Dict]

Returns
A dict using Wikibase format.

set(*language=None*, *value=None*, *action_if_exists=ActionIfExists.REPLACE_ALL*)

Create or update the specified language with the valued passed in arguments.

Parameters

- **language** (Optional[str]) – The desired language.
- **value** (Optional[str]) – The desired value of the LanguageValue object. Use None to delete an existing LanguageValue object from the list.
- **action_if_exists** (*ActionIfExists*) – The action if the LanguageValue object is already defined. Can be ActionIfExists.REPLACE_ALL (default) or ActionIfExists.KEEP.

Return type
Optional[*LanguageValue*]

Returns
The created or updated LanguageValue. None if there's no LanguageValue object with this language.

property values: Dict[str, *LanguageValue*]

A dict of LanguageValue with the language as key.

1.1.3.5 wikibaseintegrator.models.forms

class wikibaseintegrator.models.forms.**Form**(*form_id=None*, *representations=None*, *grammatical_features=None*, *claims=None*)

Bases: *BaseModel*

Parameters

- **form_id** (*Optional[str]*) –
- **representations** (*Optional[Representations]*) –
- **grammatical_features** (*Optional[Union[str, int, List[str]]]*) –
- **claims** (*Optional[Claims]*) –

__init__(*form_id=None*, *representations=None*, *grammatical_features=None*, *claims=None*)

Parameters

- **form_id** (*str | None*) –
- **representations** (Representations | *None*) –
- **grammatical_features** (*str | int | List[str] | None*) –
- **claims** (Claims | *None*) –

property claims

from_json(*json_data*)

Return type
Form

Parameters
json_data (*Dict[str, Any]*) –

get_json()

Return type
Dict[str, Union[str, Dict, List]]

property grammatical_features

property id

property representations

class wikibaseintegrator.models.forms.**Forms**

Bases: *BaseModel*

__init__()

Return type
None

add(*form*)

Return type
Forms

Parameters
form (Form) –

property forms: Dict

from_json(*json_data*)

Return type
Forms

Parameters
json_data (*List[Dict]*) –

get(*id*)

Return type
Form

Parameters
id (*str*) –

get_json()

Return type
List[Dict]

class wikibaseintegrator.models.forms.**Representations**

Bases: *LanguageValues*

__init__()

Return type
None

add(*language_value*)

Add a LanguageValue object to the list

Parameters
language_value (*LanguageValue*) – A LanguageValue object

Return type
LanguageValues

Returns
The current LanguageValues object

from_json(*json_data*)

Create a new LanguageValues object from a JSON/dict object.

Parameters
json_data (Dict[str, Dict]) – A dict object who use the same format as Wikibase.

Return type
LanguageValues

Returns
The newly created or updated object.

get(*language=None*)

Get a LanguageValue object with the specified language. Use the default language in wbi_config if none specified.

Parameters
language (Optional[str]) – The requested language.

Return type
Optional[*LanguageValue*]

Returns
The related LanguageValue object or None if none found.

get_json()

Get a formatted dict who respect the Wikibase format.

Return type
Dict[str, Dict]

Returns
A dict using Wikibase format.

set(*language=None*, *value=None*, *action_if_exists=ActionIfExists.REPLACE_ALL*)

Create or update the specified language with the valued passed in arguments.

Parameters

- **language** (Optional[str]) – The desired language.
- **value** (Optional[str]) – The desired value of the LanguageValue object. Use None to delete an existing LanguageValue object from the list.
- **action_if_exists** (*ActionIfExists*) – The action if the LanguageValue object is already defined. Can be ActionIfExists.REPLACE_ALL (default) or ActionIfExists.KEEP.

Return type
Optional[*LanguageValue*]

Returns
The created or updated LanguageValue. None if there's no LanguageValue object with this language.

property values: Dict[str, *LanguageValue*]
A dict of LanguageValue with the language as key.

1.1.3.6 wikibaseintegrator.models.labels

class wikibaseintegrator.models.labels.**Labels**
Bases: *LanguageValues*

__init__()

Return type
None

add(*language_value*)
Add a LanguageValue object to the list

Parameters
language_value (*LanguageValue*) – A LanguageValue object

Return type
LanguageValues

Returns
The current LanguageValues object

from_json(*json_data*)
Create a new Labels object from a JSON/dict object.

Parameters
json_data (Dict[str, Dict]) – A dict object who use the same format as Wikibase.

Return type
Labels

Returns
The newly created or updated object.

get(*language=None*)
Get a LanguageValue object with the specified language. Use the default language in wbi_config if none specified.

Parameters
language (Optional[str]) – The requested language.

Return type
Optional[*LanguageValue*]

Returns
The related LanguageValue object or None if none found.

get_json()

Get a formatted dict who respect the Wikibase format.

Return type
Dict[str, Dict]

Returns
A dict using Wikibase format.

set(*language=None*, *value=None*, *action_if_exists=ActionIfExists.REPLACE_ALL*)

Create or update the specified language with the valued passed in arguments.

Parameters

- **language** (Optional[str]) – The desired language.
- **value** (Optional[str]) – The desired value of the LanguageValue object. Use None to delete an existing LanguageValue object from the list.
- **action_if_exists** (*ActionIfExists*) – The action if the LanguageValue object is already defined. Can be ActionIfExists.REPLACE_ALL (default) or ActionIfExists.KEEP.

Return type
Optional[*LanguageValue*]

Returns
The created or updated LanguageValue. None if there's no LanguageValue object with this language.

property values: Dict[str, *LanguageValue*]

A dict of LanguageValue with the language as key.

1.1.3.7 wikibaseintegrator.models.language_values

class wikibaseintegrator.models.language_values.**LanguageValue**(*language*, *value=None*)

Bases: *BaseModel*

Parameters

- **language** (*str*) –
- **value** (*Optional[str]*) –

__init__(*language*, *value=None*)

Parameters

- **language** (*str*) –
- **value** (*str | None*) –

from_json(*json_data*)

Return type
LanguageValue

Parameters
json_data (*Dict[str, str]*) –

get_json()

Return type
Dict[str, Optional[str]]

property language: str

remove()

Return type
LanguageValue

property removed: bool

property value: str | None

The value of the LanguageValue instance. :return: A string with the value of the LanguageValue instance.

class wikibaseintegrator.models.language_values.**LanguageValues**

Bases: *BaseModel*

__init__()

Return type
None

add(*language_value*)

Add a LanguageValue object to the list

Parameters
language_value (*LanguageValue*) – A LanguageValue object

Return type
LanguageValues

Returns
The current LanguageValues object

from_json(*json_data*)

Create a new LanguageValues object from a JSON/dict object.

Parameters
json_data (Dict[str, Dict]) – A dict object who use the same format as Wikibase.

Return type
LanguageValues

Returns
The newly created or updated object.

get(*language=None*)

Get a LanguageValue object with the specified language. Use the default language in wbi_config if none specified.

Parameters
language (Optional[str]) – The requested language.

Return type
Optional[*LanguageValue*]

Returns
The related LanguageValue object or None if none found.

get_json()

Get a formatted dict who respect the Wikibase format.

Return type
Dict[str, Dict]

Returns
A dict using Wikibase format.

set(*language=None*, *value=None*, *action_if_exists=ActionIfExists.REPLACE_ALL*)

Create or update the specified language with the valued passed in arguments.

Parameters

- **language** (Optional[str]) – The desired language.
- **value** (Optional[str]) – The desired value of the LanguageValue object. Use None to delete an existing LanguageValue object from the list.
- **action_if_exists** (*ActionIfExists*) – The action if the LanguageValue object is already defined. Can be ActionIfExists.REPLACE_ALL (default) or ActionIfExists.KEEP.

Return type
Optional[*LanguageValue*]

Returns
The created or updated LanguageValue. None if there's no LanguageValue object with this language.

property values: Dict[str, *LanguageValue*]

A dict of LanguageValue with the language as key.

1.1.3.8 wikibaseintegrator.models.lemmas

class wikibaseintegrator.models.lemmas.**Lemmas**

Bases: *LanguageValues*

__init__()

Return type
None

add(*language_value*)

Add a LanguageValue object to the list

Parameters
language_value (*LanguageValue*) – A LanguageValue object

Return type
LanguageValues

Returns
The current LanguageValues object

from_json(*json_data*)

Create a new Lemmas object from a JSON/dict object.

Parameters
json_data (Dict[str, Dict]) – A dict object who use the same format as Wikibase.

Return type
Lemmas

Returns
The newly created or updated object.

get(*language=None*)

Get a LanguageValue object with the specified language. Use the default language in wbi_config if none specified.

Parameters
language (Optional[str]) – The requested language.

Return type
Optional[*LanguageValue*]

Returns
The related LanguageValue object or None if none found.

get_json()

Get a formatted dict who respect the Wikibase format.

Return type
Dict[str, Dict]

Returns
A dict using Wikibase format.

set(*language=None*, *value=None*, *action_if_exists=ActionIfExists.REPLACE_ALL*)

Create or update the specified language with the valued passed in arguments.

Parameters

- **language** (Optional[str]) – The desired language.
- **value** (Optional[str]) – The desired value of the LanguageValue object. Use None to delete an existing LanguageValue object from the list.
- **action_if_exists** (*ActionIfExists*) – The action if the LanguageValue object is already defined. Can be ActionIfExists.REPLACE_ALL (default) or ActionIfExists.KEEP.

Return type
Optional[*LanguageValue*]

Returns
The created or updated LanguageValue. None if there's no LanguageValue object with this language.

property values: Dict[str, *LanguageValue*]

A dict of LanguageValue with the language as key.

1.1.3.9 wikibaseintegrator.models.qualifiers

class wikibaseintegrator.models.qualifiers.**Qualifiers**

Bases: *BaseModel*

__init__()

Return type
None

add(*qualifier*, *action_if_exists=ActionIfExists.REPLACE_ALL*)

Return type
Qualifiers

Parameters

- **qualifier** (Snak | Claim) –
- **action_if_exists** (ActionIfExists) –

clear(*property=None*)

Return type
Qualifiers

Parameters
property (*str | int | None*) –

from_json(*json_data*)

Return type
Qualifiers

Parameters
json_data (*Dict[str, List]*) –

get(*property*)

Return type
List[*Snak*]

Parameters
property (*str | int*) –

get_json()

Return type
Dict[str, List]

property qualifiers

remove(*qualifier*)

Return type
Qualifiers

Parameters
qualifier (Snak | Claim) –

set(*qualifiers*)

Return type
Qualifiers

Parameters
qualifiers (*Qualifiers | List[Snak | Claim] | None*) –

1.1.3.10 wikibaseintegrator.models.references

class wikibaseintegrator.models.references.**Reference**(*snaks=None*, *snaks_order=None*)

Bases: *BaseModel*

Parameters

- **snaks** (*Optional[Snaks]*) –
- **snaks_order** (*Optional[List]*) –

__init__(*snaks=None*, *snaks_order=None*)

Parameters

- **snaks** (*Snaks | None*) –
- **snaks_order** (*List | None*) –

add(*snak=None*, *action_if_exists=ActionIfExists.REPLACE_ALL*)

Return type
Reference

Parameters

- **snak** (*Snak | Claim | None*) –
- **action_if_exists** (*ActionIfExists*) –

from_json(*json_data*)

Return type
Reference

Parameters
json_data (*Dict[str, Any]*) –

get_json()

Return type
Dict[str, Union[Dict, List]]

property hash

property snaks

property snaks_order

class wikibaseintegrator.models.references.**References**

Bases: *BaseModel*

__init__()

Return type
None

add(*reference=None*, *action_if_exists=ActionIfExists.REPLACE_ALL*)

Return type
References

Parameters

- **reference** (Reference | Claim | *None*) –
- **action_if_exists** (ActionIfExists) –

clear()

Return type
References

from_json(*json_data*)

Return type
References

Parameters
json_data (*List[Dict]*) –

get(*hash=None*)

Return type
Optional[*Reference*]

Parameters
hash (*str | None*) –

get_json()

Return type
List[Dict]

property references: List[*Reference*]

remove(*reference_to_remove*)

Return type
bool

Parameters
reference_to_remove (Claim | Reference) –

1.1.3.11 wikibaseintegrator.models.senses

class wikibaseintegrator.models.senses.**Glosses**

Bases: *LanguageValues*

__init__()

Return type
None

add(*language_value*)

Add a LanguageValue object to the list

Parameters
language_value (*LanguageValue*) – A LanguageValue object

Return type
LanguageValues

Returns
The current LanguageValues object

from_json(*json_data*)

Create a new LanguageValues object from a JSON/dict object.

Parameters
json_data (Dict[str, Dict]) – A dict object who use the same format as Wikibase.

Return type
LanguageValues

Returns
The newly created or updated object.

get(*language=None*)

Get a LanguageValue object with the specified language. Use the default language in wbi_config if none specified.

Parameters
language (Optional[str]) – The requested language.

Return type
Optional[*LanguageValue*]

Returns
The related LanguageValue object or None if none found.

get_json()

Get a formatted dict who respect the Wikibase format.

Return type
Dict[str, Dict]

Returns
A dict using Wikibase format.

set(*language=None*, *value=None*, *action_if_exists=ActionIfExists.REPLACE_ALL*)

Create or update the specified language with the valued passed in arguments.

Parameters

- **language** (Optional[str]) – The desired language.
- **value** (Optional[str]) – The desired value of the LanguageValue object. Use None to delete an existing LanguageValue object from the list.
- **action_if_exists** (*ActionIfExists*) – The action if the LanguageValue object is already defined. Can be ActionIfExists.REPLACE_ALL (default) or ActionIfExists.KEEP.

Return type
Optional[*LanguageValue*]

Returns
The created or updated LanguageValue. None if there's no LanguageValue object with this language.

property values: Dict[str, *LanguageValue*]

A dict of LanguageValue with the language as key.

class wikibaseintegrator.models.senses.**Sense**(*sense_id=None*, *glosses=None*, *claims=None*)

Bases: *BaseModel*

Parameters

- **sense_id** (*Optional[str]*) –
- **glosses** (*Optional[Glosses]*) –
- **claims** (*Optional[Claims]*) –

__init__(*sense_id=None*, *glosses=None*, *claims=None*)

Parameters

- **sense_id** (*str | None*) –
- **glosses** (Glosses | *None*) –
- **claims** (Claims | *None*) –

from_json(*json_data*)

Return type
Sense

Parameters
json_data (*Dict[str, Any]*) –

get_json()

Return type
Dict[str, Union[str, Dict]]

remove()

Return type
Sense

class wikibaseintegrator.models.senses.**Senses**

Bases: *BaseModel*

__init__()

Return type
None

add(*sense*, *action_if_exists=ActionIfExists.REPLACE_ALL*)

Return type
Senses

Parameters

- **sense** (Sense) –
- **action_if_exists** (ActionIfExists) –

from_json(*json_data*)

Return type
Senses

Parameters
json_data (*List[Dict]*) –

get(*id*)

Return type
Optional[*Sense*]

Parameters
id (*str*) –

get_json()

Return type
List[Dict]

1.1.3.12 wikibaseintegrator.models.sitelinks

class wikibaseintegrator.models.sitelinks.**Sitelink**(*site=None*, *title=None*, *badges=None*)

Bases: *BaseModel*

Parameters

- **site** (*Optional[str]*) –
- **title** (*Optional[str]*) –
- **badges** (*Optional[List[str]]*) –

__init__(*site=None*, *title=None*, *badges=None*)

Parameters

- **site** (*str | None*) –
- **title** (*str | None*) –
- **badges** (*List[str] | None*) –

class wikibaseintegrator.models.sitelinks.**Sitelinks**

Bases: *BaseModel*

__init__()

Return type
None

from_json(*json_data*)

Return type
Sitelinks

Parameters
json_data (*Dict[str, Dict]*) –

get(*site=None*)

Return type
Optional[*Sitelink*]

Parameters
site (*str | None*) –

set(*site*, *title=None*, *badges=None*)

Return type
Sitelink

Parameters

- **site** (*str*) –
- **title** (*str | None*) –
- **badges** (*List[str] | None*) –

1.1.3.13 wikibaseintegrator.models.snaks

class wikibaseintegrator.models.snaks.**Snak**(*snaktype=WikibaseSnakType.KNOWN_VALUE*, *property_number=None*, *hash=None*, *datavalue=None*, *datatype=None*)

Bases: *BaseModel*

Parameters

- **snaktype** (WikibaseSnakType) –
- **property_number** (*Optional[str]*) –
- **hash** (*Optional[str]*) –
- **datavalue** (*Optional[Dict]*) –
- **datatype** (*Optional[str]*) –

__init__(*snaktype=WikibaseSnakType.KNOWN_VALUE*, *property_number=None*, *hash=None*, *datavalue=None*, *datatype=None*)

Parameters

- **snaktype** (WikibaseSnakType) –
- **property_number** (*str | None*) –
- **hash** (*str | None*) –

- **datavalue** (*Dict | None*) –
- **datatype** (*str | None*) –

property datatype

property datavalue

from_json(*json_data*)

Return type
Snak

Parameters
json_data (*Dict[str, Any]*) –

get_json()

Return type
Dict[str, str]

property hash

property property_number

property snaktype

class wikibaseintegrator.models.snaks.**Snaks**

Bases: *BaseModel*

__init__()

Return type
None

add(*snak*)

Return type
Snaks

Parameters
snak (Snak) –

from_json(*json_data*)

Return type
Snaks

Parameters
json_data (*Dict[str, List]*) –

get(*property*)

Return type
List[*Snak*]

Parameters
property (*str*) –

get_json()

Return type
Dict[str, List]

1.2 Submodules

1.2.1 wikibaseintegrator.wbi_backoff

WikibaseIntegrator implementation of backoff python library.

wikibaseintegrator.wbi_backoff.**wbi_backoff_backoff_hdlr**(*details*)

wikibaseintegrator.wbi_backoff.**wbi_backoff_check_json_decode_error**(*e*)

Check if the error message is "Expecting value: line 1 column 1 (char 0)" if not, its a real error and we shouldn't retry

Return type
bool

1.2.2 wikibaseintegrator.wbi_config

Config global options Options can be changed at run time. See tests/test_backoff.py for usage example

Options: BACKOFF_MAX_TRIES: maximum number of times to retry failed request to wikidata endpoint.

Default: None (retry indefinitely) To disable retry, set value to 1

BACKOFF_MAX_VALUE: maximum number of seconds to wait before retrying. wait time will increase to this number
Default: 3600 (one hour)

USER_AGENT: Complementary user agent string used for http requests. Both to Wikibase api, query service and others.
See: https://meta.wikimedia.org/wiki/User-Agent_policy

1.2.3 wikibaseintegrator.wbi_enums

class wikibaseintegrator.wbi_enums.**ActionIfExists**(*value*)

Bases: Enum

Action to take if a statement with a property already exists on the entity.

APPEND_OR_REPLACE: Add the new element to the property if it does not exist, otherwise replace the existing element. FORCE_APPEND: Forces the addition of the new element to the property, even if it already exists. KEEP: Does nothing if the property already has elements stated. REPLACE_ALL: Replace all elements with the same property number.

APPEND_OR_REPLACE = 1

FORCE_APPEND = 2

KEEP = 3

REPLACE_ALL = 4

class wikibaseintegrator.wbi_enums.**WikibaseDatatype**(*value*)

Bases: Enum

An enumeration.

COMMONSMEDIA = 'commonsMedia'

EXTERNALID = 'external-id'

FORM = 'wikibase-form'

GEOSHAPE = 'geo-shape'

GLOBECOORDINATE = 'globe-coordinate'

ITEM = 'wikibase-item'

LEXEME = 'wikibase-lexeme'

MATH = 'math'

MONOLINGUALTEXT = 'monolingualtext'

MUSICALNOTATION = 'musical-notation'

PROPERTY = 'wikibase-property'

QUANTITY = 'quantity'

SENSE = 'wikibase-sense'

STRING = 'string'

TABULARDATA = 'tabular-data'

TIME = 'time'

URL = 'url'

class wikibaseintegrator.wbi_enums.**WikibaseDatePrecision**(*value*)

Bases: Enum

An enumeration.

BILLION_YEARS = 0

CENTURY = 7

DAY = 11

DECADE = 8

HUNDRED_THOUSAND_YEARS = 4

MILLENNIUM = 6

MILLION_YEARS = 3

MONTH = 10

YEAR = 9

class wikibaseintegrator.wbi_enums.**WikibaseRank**(*value*)

Bases: Enum

An enumeration.

DEPRECATED = 'deprecated'

NORMAL = 'normal'

PREFERRED = 'preferred'

class wikibaseintegrator.wbi_enums.**WikibaseSnakType**(*value*)

Bases: Enum

The snak type of the Wikibase data snak, three values possible, depending if the value is a known (value), not existent (novalue) or unknown (somevalue). See Wikibase documentation.

KNOWN_VALUE = 'value'

NO_VALUE = 'novalue'

UNKNOWN_VALUE = 'somevalue'

1.2.4 wikibaseintegrator.wbi_exceptions

exception wikibaseintegrator.wbi_exceptions.**MWApiError**(*error_dict*)

Bases: Exception

Base class for MediaWiki API error handling

Parameters
error_dict (*Dict[str, Any]*) –

__init__(*error_dict*)

Parameters
error_dict (*Dict[str, Any]*) –

args

code: str

error_dict: Dict[str, Any]

property get_conflicting_entity_ids: List[str]

Compute the list of conflicting entities from the error messages.

Returns
A list of conflicting entities or an empty list

property get_languages: List[str]

Compute a list of language identifiers from the error messages. Indicating the language which triggered the error.

Returns
A list of language identifiers or an empty list

info: str

messages: List[Dict[str, Any]]

messages_names: List[str]

with_traceback()

Exception.with_traceback(tb) – set self.__traceback__ to tb and return self.

exception wikibaseintegrator.wbi_exceptions.**MaxRetriesReachedException**

Bases: Exception

__init__(**args*, ***kwargs*)

args

with_traceback()

Exception.with_traceback(tb) – set self.__traceback__ to tb and return self.

exception wikibaseintegrator.wbi_exceptions.**MissingEntityException**

Bases: Exception

__init__(**args*, ***kwargs*)

args

with_traceback()

Exception.with_traceback(tb) – set self.__traceback__ to tb and return self.

exception wikibaseintegrator.wbi_exceptions.**ModificationFailed**(*error_dict*)

Bases: *MWApiError*

When the API return a 'modification-failed' error

Parameters
error_dict (*Dict[str, Any]*) –

__init__(*error_dict*)

Parameters
error_dict (*Dict[str, Any]*) –

args

code: str

error_dict: Dict[str, Any]

property get_conflicting_entity_ids: List[str]

Compute the list of conflicting entities from the error messages.

Returns
A list of conflicting entities or an empty list

property get_languages: List[str]

Compute a list of language identifiers from the error messages. Indicating the language which triggered the error.

Returns
A list of language identifiers or an empty list

info: str

messages: List[Dict[str, Any]]

messages_names: List[str]

with_traceback()

Exception.with_traceback(tb) – set self.__traceback__ to tb and return self.

exception wikibaseintegrator.wbi_exceptions.**NonExistentEntityError**(*error_dict*)

Bases: *MWApiError*

Parameters
error_dict (*Dict[str, Any]*) –

__init__(*error_dict*)

Parameters
error_dict (*Dict[str, Any]*) –

args

code: str

error_dict: Dict[str, Any]

property get_conflicting_entity_ids: List[str]

Compute the list of conflicting entities from the error messages.

Returns
A list of conflicting entities or an empty list

property get_languages: List[str]

Compute a list of language identifiers from the error messages. Indicating the language which triggered the error.

Returns
A list of language identifiers or an empty list

info: str

messages: List[Dict[str, Any]]

messages_names: List[str]

with_traceback()

Exception.with_traceback(tb) – set self.__traceback__ to tb and return self.

exception wikibaseintegrator.wbi_exceptions.**SaveFailed**(*error_dict*)

Bases: *MWApiError*

When the API return a 'save-failed' error

Parameters
error_dict (*Dict[str, Any]*) –

__init__(*error_dict*)

Parameters
error_dict (*Dict[str, Any]*) –

args

code: str

error_dict: Dict[str, Any]

property get_conflicting_entity_ids: List[str]

Compute the list of conflicting entities from the error messages.

Returns

A list of conflicting entities or an empty list

property get_languages: List[str]

Compute a list of language identifiers from the error messages. Indicating the language which triggered the error.

Returns

A list of language identifiers or an empty list

info: str

messages: List[Dict[str, Any]]

messages_names: List[str]

with_traceback()

Exception.with_traceback(tb) – set self.__traceback__ to tb and return self.

exception wikibaseintegrator.wbi_exceptions.**SearchError**

Bases: Exception

__init__(**args*, ***kwargs*)

args

with_traceback()

Exception.with_traceback(tb) – set self.__traceback__ to tb and return self.

1.2.5 wikibaseintegrator.wbi_fastrun

class wikibaseintegrator.wbi_fastrun.**FastRunContainer**(*base_data_type*, *mediawiki_api_url=None*, *sparql_endpoint_url=None*, *wikibase_url=None*, *base_filter=None*, *use_refs=False*, *case_insensitive=False*)

Bases: object

Parameters

- **base_data_type** (*Type[BaseDataType]*) –
- **mediawiki_api_url** (*Optional[str]*) –
- **sparql_endpoint_url** (*Optional[str]*) –
- **wikibase_url** (*Optional[str]*) –
- **base_filter** (*Optional[List[BaseDataType | List[BaseDataType]]]*) –
- **use_refs** (*bool*) –
- **case_insensitive** (*bool*) –

__init__(*base_data_type*, *mediawiki_api_url=None*, *sparql_endpoint_url=None*, *wikibase_url=None*, *base_filter=None*, *use_refs=False*, *case_insensitive=False*)

Parameters

- **base_data_type** (*Type[BaseDataType]*) –
- **mediawiki_api_url** (*Optional[str]*) –
- **sparql_endpoint_url** (*Optional[str]*) –
- **wikibase_url** (*Optional[str]*) –
- **base_filter** (*Optional[List[BaseDataType | List[BaseDataType]]]*) –
- **use_refs** (*bool*) –
- **case_insensitive** (*bool*) –

check_language_data(*qid*, *lang_data*, *lang*, *lang_data_type*, *action_if_exists=ActionIfExists.APPEND_OR_REPLACE*)

Method to check if certain language data exists as a label, description or aliases :type qid: `str` :param qid: Wikibase item id :type lang_data: `List` :param lang_data: list of string values to check :type lang: `str` :param lang: language code :type lang_data_type: `str` :param lang_data_type: What kind of data is it? 'label', 'description' or 'aliases'? :type action_if_exists: *ActionIfExists* :param action_if_exists: If aliases already exist, APPEND_OR_REPLACE or REPLACE_ALL :rtype: `bool` :return: boolean

Parameters

- **qid** (*str*) –
- **lang_data** (*List*) –
- **lang** (*str*) –
- **lang_data_type** (*str*) –
- **action_if_exists** (*ActionIfExists*) –

Return type
bool

clear()

convenience function to empty this fastrun container

Return type
`None`

format_query_results(*r*, *prop_nr*)

r is the results of the sparql query in _query_data and is modified in place *prop_nr* is needed to get the property datatype to determine how to format the value

Return type
`None`

Parameters

- **r** (*List*) –
- **prop_nr** (*str*) –

***r* is a list of dicts. The keys are:**
sid: statement ID item: the subject. the item this statement is on v: the object. The value for this statement unit: property unit pq: qualifier property qval: qualifier value qunit: qualifier unit ref: reference ID pr: reference property rval: reference value

get_all_data()

Return type
Dict[str, Dict]

get_item(*claims*, *cqid=None*)

Parameters

- **claims** (Union[List[*Claim*], *Claims*, *Claim*]) – A list of claims the entity should have
- **cqid** (Optional[str]) –

Return type
Optional[str]

Returns
An entity ID, None if there is more than one.

get_items(*claims*, *cqid=None*)

Get items ID from a SPARQL endpoint

Parameters

- **claims** (Union[List[*Claim*], *Claims*, *Claim*]) – A list of claims the entities should have
- **cqid** (Optional[str]) –

Return type
Optional[Set[str]]

Returns
a list of entity ID or None

Exception
if there is more than one claim

get_language_data(*qid*, *lang*, *lang_data_type*)

get language data for specified qid

Parameters

- **qid** (str) – Wikibase item id
- **lang** (str) – language code
- **lang_data_type** (str) – 'label', 'description' or 'aliases'

Return type
List[str]

Returns
list of strings

If nothing is found:
If lang_data_type == label: returns [''] If lang_data_type == description: returns [''] If lang_data_type == aliases: returns []

get_prop_datatype(*prop_nr*)

Return type
Optional[str]

Parameters
prop_nr (*str*) –

`init_language_data`(*lang*, *lang_data_type*)

Initialize language data store

Parameters

- **lang** (str) – language code
- **lang_data_type** (str) – 'label', 'description' or 'aliases'

Return type
None

Returns
None

`reconstruct_statements`(*qid*)

Return type
List[*BaseDataType*]

Parameters
qid (*str*) –

`update_frc_from_query`(*r*, *prop_nr*)

Return type
None

Parameters

- **r** (*List*) –
- **prop_nr** (*str*) –

`write_required`(*data*, *action_if_exists=ActionIfExists.REPLACE_ALL*, *cqid=None*)

Check if a write is required

Parameters

- **data** (List[*Claim*]) –
- **action_if_exists** (*ActionIfExists*) –
- **cqid** (Optional[str]) –

Return type
bool

Returns
Return True if the write is required

wikibaseintegrator.wbi_fastrun.`get_fastrun_container`(*base_filter=None*, *use_refs=False*, *case_insensitive=False*)

Return type
FastRunContainer

Parameters

- **base_filter** (*Optional[List[BaseDataType | List[BaseDataType]]]*) –
- **use_refs** (*bool*) –

- **case_insensitive** (*bool*) –

1.2.6 wikibaseintegrator.wbi_helpers

Multiple functions or classes that can be used to interact with the Wikibase instance.

class wikibaseintegrator.wbi_helpers.**BColors**

Bases: object

Default colors for pretty outputs.

BOLD = '\x1b[1m'

ENDC = '\x1b[0m'

FAIL = '\x1b[91m'

HEADER = '\x1b[95m'

OKBLUE = '\x1b[94m'

OKCYAN = '\x1b[96m'

OKGREEN = '\x1b[92m'

UNDERLINE = '\x1b[4m'

WARNING = '\x1b[93m'

__init__()

wikibaseintegrator.wbi_helpers.**delete_page**(*title=None*, *pageid=None*, *reason=None*, *deletetalk=False*, *watchlist='preferences'*, *watchlistexpiry=None*, *login=None*, ***kwargs*)

Delete a page

Parameters

- **title** (Optional[str]) – Title of the page to delete. Cannot be used together with pageid.
- **pageid** (Optional[int]) – Page ID of the page to delete. Cannot be used together with title.
- **reason** (Optional[str]) – Reason for the deletion. If not set, an automatically generated reason will be used.
- **deletetalk** (bool) – Delete the talk page, if it exists.
- **watchlist** (str) – Unconditionally add or remove the page from the current user's watchlist, use preferences (ignored for bot users) or do not change watch. One of the following values: nochange, preferences, unwatch, watch
- **watchlistexpiry** (Optional[str]) – Watchlist expiry timestamp. Omit this parameter entirely to leave the current expiry unchanged.
- **login** (Optional[_Login]) – A wbi_login.Login instance
- **kwargs** (Any) –

Return type

Dict

Returns

wikibaseintegrator.wbi_helpers.**edit_entity**(*data*, *id=None*, *type=None*, *baserevid=None*, *summary=None*, *clear=False*, *is_bot=False*, *tags=None*, *site=None*, *title=None*, ***kwargs*)

Creates a single new Wikibase entity and modifies it with serialised information.

Parameters

- **data** (Dict) – The serialized object that is used as the data source. A newly created entity will be assigned an 'id'.
- **id** (Optional[str]) – The identifier for the entity, including the prefix. Use either id or site and title together.
- **type** (Optional[str]) – Set this to the type of the entity to be created. One of the following values: form, item, lexeme, property, sense
- **baserevid** (Optional[int]) – The numeric identifier for the revision to base the modification on. This is used for detecting conflicts during save.
- **summary** (Optional[str]) – Summary for the edit. Will be prepended by an automatically generated comment.
- **clear** (bool) – If set, the complete entity is emptied before proceeding. The entity will not be saved before it is filled with the "data", possibly with parts excluded.
- **is_bot** (bool) – Mark this edit as bot.
- **login** – A login instance
- **tags** (Optional[List[str]]) – Change tags to apply to the revision.
- **site** (Optional[str]) – An identifier for the site on which the page resides. Use together with title to make a complete sitelink.
- **title** (Optional[str]) – Title of the page to associate. Use together with site to make a complete sitelink.
- **kwargs** (Any) – More arguments for Python requests

Return type
Dict

Returns
The answer from the Wikibase API

wikibaseintegrator.wbi_helpers.**execute_sparql_query**(*query*, *prefix=None*, *endpoint=None*, *user_agent=None*, *max_retries=1000*, *retry_after=60*)

Static method which can be used to execute any SPARQL query

Parameters

- **prefix** (Optional[str]) – The URI prefixes required for an endpoint, default is the Wikidata specific prefixes
- **query** (str) – The actual SPARQL query string
- **endpoint** (Optional[str]) – The URL string for the SPARQL endpoint. Default is the URL for the Wikidata SPARQL endpoint
- **user_agent** (Optional[str]) – Set a user agent string for the HTTP header to let the Query Service know who you are.

- **max_retries** (int) – The number time this function should retry in case of header reports.
- **retry_after** (int) – the number of seconds should wait upon receiving either an error code or the Query Service is not reachable.

Return type
Dict[str, Dict]

Returns
The results of the query are returned in JSON format

wikibaseintegrator.wbi_helpers.**format2wbi**(*entitytype*, *json_raw*, *allow_anonymous=True*, *wikibase_url=None*, ***kwargs*)

Return type
BaseEntity

Parameters

- **entitytype** (*str*) –
- **json_raw** (*str*) –
- **allow_anonymous** (*bool*) –
- **wikibase_url** (*str | None*) –

wikibaseintegrator.wbi_helpers.**format_amount**(*amount*)

A formatting function mostly used for Quantity datatype. :type amount: Union[int, str, float] :param amount: A int, float or str you want to pass to Quantity value. :rtype: str :return: A correctly formatted string amount by Wikibase standard.

Parameters
amount (*int | str | float*) –

Return type
str

wikibaseintegrator.wbi_helpers.**fulltext_search**(*search*, *max_results=50*, *allow_anonymous=True*, ***kwargs*)

Perform a fulltext search on the mediawiki instance. It's an exception to the "only wikibase related function" rule! WikibaseIntegrator is focused on wikibase-only functions to avoid spreading out and covering all functions of MediaWiki.

Parameters

- **search** (str) – Search for page titles or content matching this value. You can use the search string to invoke special search features, depending on what the wiki's search backend implements.
- **max_results** (int) – How many total pages to return. The value must be between 1 and 500.
- **allow_anonymous** (bool) – Allow anonymous interaction with the MediaWiki API. 'True' by default.
- **kwargs** (Any) – Extra parameters for mediawiki_api_call_helper()

Return type
List[Dict[str, Any]]

Returns

wikibaseintegrator.wbi_helpers.**generate_entity_instances**(*entities*, *allow_anonymous=True*, ***kwargs*)

A method which allows for retrieval of a list of Wikidata entities. The method generates a list of tuples where the first value in the tuple is the entity's ID, whereas the second is the new instance of a subclass of BaseEntity containing all the data of the entity. This is most useful for mass retrieval of entities.

Parameters

- **entities** (Union[str, List[str]]) – A list of IDs. Item, Property or Lexeme.
- **allow_anonymous** (bool) – Allow anonymous edit to the MediaWiki API. Disabled by default.
- **kwargs** (*Any*) –

Return type
List[Tuple[str, *BaseEntity*]]

Returns
A list of tuples, first value in the tuple is the entity's ID, second value is the instance of a subclass of BaseEntity with the corresponding entity data.

wikibaseintegrator.wbi_helpers.**get_user_agent**(*user_agent*)

Return a user agent string suitable for interacting with the Wikibase instance.

Parameters
user_agent (Optional[str]) – An optional user-agent. If not provided, will generate a default user-agent.

Return type
str

Returns
A correctly formatted user agent.

wikibaseintegrator.wbi_helpers.**lexeme_add_form**(*lexeme_id*, *data*, *baserevid=None*, *tags=None*, *is_bot=False*, ***kwargs*)

Adds Form to Lexeme

Parameters

- **lexeme_id** – ID of the Lexeme, e.g. L10
- **data** – The serialized object that is used as the data source.
- **baserevid** (Optional[int]) – Base Revision ID of the Lexeme, if edit conflict check is wanted.
- **tags** (Optional[List[str]]) – Change tags to apply to the revision.
- **is_bot** (bool) – Mark this edit as bot.
- **kwargs** (Any) –

Return type
Dict

Returns

wikibaseintegrator.wbi_helpers.**lexeme_add_sense**(*lexeme_id*, *data*, *baserevid=None*, *tags=None*, *is_bot=False*, ***kwargs*)

Adds a Sense to a Lexeme

Parameters

- **lexeme_id** – ID of the Lexeme, e.g. L10
- **data** – JSON-encoded data for the Sense, i.e. its glosses
- **baserevid** (Optional[int]) – Base Revision ID of the Lexeme, if edit conflict check is wanted.
- **tags** (Optional[List[str]]) – Change tags to apply to the revision.
- **is_bot** (bool) – Mark this edit as bot.
- **kwargs** (Any) –

Return type
Dict

Returns

wikibaseintegrator.wbi_helpers.**lexeme_edit_form**(*form_id*, *data*, *baserevid=None*, *tags=None*, *is_bot=False*, ***kwargs*)

Edits representations and grammatical features of a Form

Parameters

- **form_id** (str) – ID of the Form or the concept URI, e.g. L10-F2
- **data** – The serialized object that is used as the data source.
- **baserevid** (Optional[int]) – Base Revision ID of the Lexeme, if edit conflict check is wanted.
- **tags** (Optional[List[str]]) – Change tags to apply to the revision.
- **is_bot** (bool) – Mark this edit as bot.
- **kwargs** (Any) –

Return type
Dict

Returns

wikibaseintegrator.wbi_helpers.**lexeme_edit_sense**(*sense_id*, *data*, *baserevid=None*, *tags=None*, *is_bot=False*, ***kwargs*)

Edits glosses of a Sense

Parameters

- **sense_id** (str) – ID of the Sense or the concept URI, e.g. L10-S2
- **data** – The serialized object that is used as the data source.
- **baserevid** (Optional[int]) – Base Revision ID of the Lexeme, if edit conflict check is wanted.
- **tags** (Optional[List[str]]) – Change tags to apply to the revision.
- **is_bot** (bool) – Mark this edit as bot.
- **kwargs** (Any) –

Return type
Dict

Returns

wikibaseintegrator.wbi_helpers.**lexeme_remove_form**(*form_id*, *baserevid=None*, *tags=None*, *is_bot=False*, ***kwargs*)

Removes Form from Lexeme

Parameters

- **form_id** (str) – ID of the Form or the concept URI, e.g. L10-F2
- **baserevid** (Optional[int]) – Base Revision ID of the Lexeme, if edit conflict check is wanted.
- **tags** (Optional[List[str]]) – Change tags to apply to the revision.
- **is_bot** (bool) – Mark this edit as bot.
- **kwargs** (Any) –

Return type
Dict

Returns

wikibaseintegrator.wbi_helpers.**lexeme_remove_sense**(*sense_id*, *baserevid=None*, *tags=None*, *is_bot=False*, ***kwargs*)

Adds Form to Lexeme

Parameters

- **sense_id** (str) – ID of the Sense, e.g. L10-S20
- **baserevid** (Optional[int]) – Base Revision ID of the Lexeme, if edit conflict check is wanted.
- **tags** (Optional[List[str]]) – Change tags to apply to the revision.
- **is_bot** (bool) – Mark this edit as bot.
- **kwargs** (Any) –

Return type
Dict

Returns

wikibaseintegrator.wbi_helpers.**mediawiki_api_call**(*method*, *mediawiki_api_url=None*, *session=None*, *max_retries=100*, *retry_after=60*, ***kwargs*)

A function to call the MediaWiki API.

Parameters

- **method** (str) – 'GET' or 'POST'
- **mediawiki_api_url** (Optional[str]) –
- **session** (Optional[Session]) – If a session is passed, it will be used. Otherwise, a new requests session is created
- **max_retries** (int) – If api request fails due to rate limiting, maxlag, or readonly mode, retry up to *max_retries* times
- **retry_after** (int) – Number of seconds to wait before retrying request (see max_retries)
- **kwargs** (Any) – Any additional keyword arguments to pass to requests.request

Return type
Dict

Returns
The data returned by the API as a dictionary

wikibaseintegrator.wbi_helpers.**mediawiki_api_call_helper**(*data*, *login=None*, *mediawiki_api_url=None*, *user_agent=None*, *allow_anonymous=False*, *max_retries=1000*, *retry_after=60*, *maxlag=5*, *is_bot=False*, ***kwargs*)

A simplified function to call the MediaWiki API. Pass the data, as a dictionary, related to the action you want to call, all commons options will be automatically managed.

Parameters

- **data** (Dict[str, Any]) – A dictionary containing the JSON data to send to the API
- **login** (Optional[_Login]) – A wbi_login._Login instance
- **mediawiki_api_url** (Optional[str]) – The URL to the MediaWiki API (default Wikidata)
- **user_agent** (Optional[str]) – The user agent (Recommended for Wikimedia Foundation instances)
- **allow_anonymous** (bool) – Allow an unidentified edit to the MediaWiki API (default False)
- **max_retries** (int) – The maximum number of retries
- **retry_after** (int) – The timeout between each retry
- **maxlag** (int) – If applicable, the maximum lag allowed for the replication (An lower number reduce the load on the replicated database)
- **is_bot** (bool) – Flag the edit as a bot
- **kwargs** (Any) – Any additional keyword arguments to pass to requests.request

Return type
Dict

Returns
The data returned by the API as a dictionary

wikibaseintegrator.wbi_helpers.**merge_items**(*from_id*, *to_id*, *login=None*, *ignore_conflicts=None*, *is_bot=False*, ***kwargs*)

A static method to merge two items

Parameters

- **from_id** (str) – The ID to merge from. This parameter is required.
- **to_id** (str) – The ID to merge to. This parameter is required.
- **login** (Optional[_Login]) – A wbi_login.Login instance
- **ignore_conflicts** (Optional[List[str]]) – List of elements of the item to ignore conflicts for. Can only contain values of “description”, “sitelink” and “statement”
- **is_bot** (bool) – Mark this edit as bot.
- **kwargs** (*Any*) –

Return type
Dict

wikibaseintegrator.wbi_helpers.**merge_lexemes**(*source*, *target*, *login=None*, *summary=None*, *is_bot=False*, ***kwargs*)

A static method to merge two lexemes

Parameters

- **source** (str) – The ID to merge from. This parameter is required.
- **target** (str) – The ID to merge to. This parameter is required.
- **login** (Optional[_Login]) – A wbi_login.Login instance
- **summary** (Optional[str]) – Summary for the edit.
- **is_bot** (bool) – Mark this edit as bot.
- **kwargs** (*Any*) –

Return type
Dict

wikibaseintegrator.wbi_helpers.**remove_claims**(*claim_id*, *summary=None*, *baserevid=None*, *is_bot=False*, ***kwargs*)

Delete a claim from an entity

Parameters

- **claim_id** (str) – One GUID or several (pipe-separated) GUIDs identifying the claims to be removed. All claims must belong to the same entity.
- **summary** (Optional[str]) – Summary for the edit. Will be prepended by an automatically generated comment.
- **baserevid** (Optional[int]) – The numeric identifier for the revision to base the modification on. This is used for detecting conflicts during save.
- **is_bot** (bool) – Mark this edit as bot.
- **kwargs** (*Any*) –

Return type
Dict

wikibaseintegrator.wbi_helpers.**search_entities**(*search_string*, *language=None*, *strict_language=False*, *search_type='item'*, *max_results=50*, *dict_result=False*, *allow_anonymous=True*, ***kwargs*)

Performs a search for entities in the Wikibase instance using labels and aliases. You can have more information on the parameters in the MediaWiki API help (https://www.wikidata.org/w/api.php?action=help&modules=wbsearchentities)

Parameters

- **search_string** (str) – A string which should be searched for in the Wikibase instance (labels and aliases)
- **language** (Optional[str]) – The language in which to perform the search. This only affects how entities are selected. Default is 'en' from wbi_config. You can see the list of languages for Wikidata at https://www.wikidata.org/wiki/Help:Wikimedia_language_codes/lists/all (Use the WMF code)
- **strict_language** (bool) – Whether to disable language fallback. Default is 'False'.
- **search_type** (str) – Search for this type of entity. One of the following values: form, item, lexeme, property, sense, mediainfo

- **max_results** (int) – The maximum number of search results returned. The value must be between 0 and 50. Default is 50
- **dict_result** (bool) – Return the results as a detailed dictionary instead of a list of IDs.
- **allow_anonymous** (bool) – Allow anonymous interaction with the MediaWiki API. 'True' by default.
- **kwargs** (*Any*) –

Return type
List[Dict[str, Any]]

1.2.7 wikibaseintegrator.wbi_login

Login class for Wikidata. Takes authentication parameters and stores the session cookies and edit tokens.

class wikibaseintegrator.wbi_login.**Clientlogin**(*user=None*, *password=None*, *mediawiki_api_url=None*, *token_renew_period=1800*, *user_agent=None*)

Bases: _Login

Parameters

- **user** (*str | None*) –
- **password** (*str | None*) –
- **mediawiki_api_url** (*str | None*) –
- **token_renew_period** (*int*) –
- **user_agent** (*str | None*) –

__init__(*user=None*, *password=None*, *mediawiki_api_url=None*, *token_renew_period=1800*, *user_agent=None*)

This class is used to log in with a user account

Parameters

- **user** (Optional[str]) – The username
- **password** (Optional[str]) – The password
- **mediawiki_api_url** (Optional[str]) – The URL to the MediaWiki API (default Wikidata)
- **token_renew_period** (int) – Seconds after which a new token should be requested from the Wikidata server
- **user_agent** (Optional[str]) – UA string to use for API requests.

generate_edit_credentials()

Request an edit token and update the cookie_jar in order to add the session cookie

Return type
RequestsCookieJar

Returns
Returns a json with all relevant cookies, aka cookie jar

get_edit_cookie()

Can be called in order to retrieve the cookies from an instance of wbi_login.Login

Return type
RequestsCookieJar

Returns
Returns a json with all relevant cookies, aka cookie jar

get_edit_token()

Can be called in order to retrieve the edit token from an instance of wbi_login.Login

Return type
Optional[str]

Returns
returns the edit token

get_session()

Returns the requests.Session object used for the login.

Return type
Session

Returns
Object of type requests.Session()

class wikibaseintegrator.wbi_login.**Login**(*user=None*, *password=None*, *mediawiki_api_url=None*, *token_renew_period=1800*, *user_agent=None*)

Bases: _Login

Parameters

- **user** (*str | None*) –
- **password** (*str | None*) –
- **mediawiki_api_url** (*str | None*) –
- **token_renew_period** (*int*) –
- **user_agent** (*str | None*) –

__init__(*user=None*, *password=None*, *mediawiki_api_url=None*, *token_renew_period=1800*, *user_agent=None*)

This class is used to log in with a bot password

Parameters

- **user** (Optional[str]) – The user of the bot password (format <User>@<BotUser>)
- **password** (Optional[str]) – The password generated byt the MediaWiki
- **mediawiki_api_url** (Optional[str]) – The URL to the MediaWiki API (default Wikidata)
- **token_renew_period** (int) – Seconds after which a new token should be requested from the Wikidata server
- **user_agent** (Optional[str]) – UA string to use for API requests.

generate_edit_credentials()

Request an edit token and update the cookie_jar in order to add the session cookie

Return type
RequestsCookieJar

Returns
Returns a json with all relevant cookies, aka cookie jar

get_edit_cookie()

Can be called in order to retrieve the cookies from an instance of wbi_login.Login

Return type
RequestsCookieJar

Returns
Returns a json with all relevant cookies, aka cookie jar

get_edit_token()

Can be called in order to retrieve the edit token from an instance of wbi_login.Login

Return type
Optional[str]

Returns
returns the edit token

get_session()

Returns the requests.Session object used for the login.

Return type
Session

Returns
Object of type requests.Session()

exception wikibaseintegrator.wbi_login.**LoginError**

Bases: Exception

Raised when there is an issue with the login

__init__(**args*, ***kwargs*)

args

with_traceback()

Exception.with_traceback(tb) – set self.__traceback__ to tb and return self.

class wikibaseintegrator.wbi_login.**OAuth1**(*consumer_token=None*, *consumer_secret=None*, *access_token=None*, *access_secret=None*, *callback_url='oob'*, *mediawiki_api_url=None*, *mediawiki_index_url=None*, *token_renew_period=1800*, *user_agent=None*)

Bases: _Login

Parameters

- **consumer_token** (*str | None*) –
- **consumer_secret** (*str | None*) –
- **access_token** (*str | None*) –

- **access_secret** (*str | None*) –
- **callback_url** (*str*) –
- **mediawiki_api_url** (*str | None*) –
- **mediawiki_index_url** (*str | None*) –
- **token_renew_period** (*int*) –
- **user_agent** (*str | None*) –

__init__(*consumer_token=None*, *consumer_secret=None*, *access_token=None*, *access_secret=None*, *callback_url='oob'*, *mediawiki_api_url=None*, *mediawiki_index_url=None*, *token_renew_period=1800*, *user_agent=None*)

This class is used to interact with the OAuth1 API.

Parameters

- **consumer_token** (Optional[str]) – The consumer token
- **consumer_secret** (Optional[str]) – The consumer secret
- **access_token** (Optional[str]) – The access token (optional)
- **access_secret** (Optional[str]) – The access secret (optional)
- **callback_url** (str) – The callback URL used to finalize the handshake
- **mediawiki_api_url** (Optional[str]) – The URL to the MediaWiki API (default Wikidata)
- **mediawiki_index_url** (Optional[str]) – The URL to the MediaWiki index (default Wikidata)
- **token_renew_period** (int) – Seconds after which a new token should be requested from the Wikidata server
- **user_agent** (Optional[str]) – UA string to use for API requests.

continue_oauth(*oauth_callback_data=None*)

Continuation of OAuth procedure. Method must be explicitly called in order to complete OAuth. This allows external entities, e.g. websites, to provide tokens through callback URLs directly.

Parameters
oauth_callback_data (Optional[str]) – The callback URL received to a Web app

Return type
None

Returns

generate_edit_credentials()

Request an edit token and update the cookie_jar in order to add the session cookie

Return type
RequestsCookieJar

Returns
Returns a json with all relevant cookies, aka cookie jar

get_edit_cookie()

Can be called in order to retrieve the cookies from an instance of wbi_login.Login

Return type
RequestsCookieJar

Returns
Returns a json with all relevant cookies, aka cookie jar

`get_edit_token()`

Can be called in order to retrieve the edit token from an instance of wbi_login.Login

Return type
Optional[str]

Returns
returns the edit token

`get_session()`

Returns the requests.Session object used for the login.

Return type
Session

Returns
Object of type requests.Session()

`class wikibaseintegrator.wbi_login.OAuth2(consumer_token=None, consumer_secret=None, mediawiki_api_url=None, mediawiki_rest_url=None, token_renew_period=1800, user_agent=None)`

Bases: _Login

Parameters

- **consumer_token** (*str | None*) –
- **consumer_secret** (*str | None*) –
- **mediawiki_api_url** (*str | None*) –
- **mediawiki_rest_url** (*str | None*) –
- **token_renew_period** (*int*) –
- **user_agent** (*str | None*) –

`__init__(consumer_token=None, consumer_secret=None, mediawiki_api_url=None, mediawiki_rest_url=None, token_renew_period=1800, user_agent=None)`

This class is used to interact with the OAuth2 API.

Parameters

- **consumer_token** (Optional[str]) – The consumer token
- **consumer_secret** (Optional[str]) – The consumer secret
- **mediawiki_api_url** (Optional[str]) – The URL to the MediaWiki API (default Wikidata)
- **mediawiki_rest_url** (Optional[str]) – The URL to the MediaWiki REST API (default Wikidata)
- **token_renew_period** (int) – Seconds after which a new token should be requested from the Wikidata server
- **user_agent** (Optional[str]) – UA string to use for API requests.

generate_edit_credentials()

Request an edit token and update the cookie_jar in order to add the session cookie

Return type
RequestsCookieJar

Returns
Returns a json with all relevant cookies, aka cookie jar

get_edit_cookie()

Can be called in order to retrieve the cookies from an instance of wbi_login.Login

Return type
RequestsCookieJar

Returns
Returns a json with all relevant cookies, aka cookie jar

get_edit_token()

Can be called in order to retrieve the edit token from an instance of wbi_login.Login

Return type
Optional[str]

Returns
returns the edit token

get_session()

Returns the requests.Session object used for the login.

Return type
Session

Returns
Object of type requests.Session()

1.2.8 wikibaseintegrator.wikibaseintegrator

Main class of the Library.

class wikibaseintegrator.wikibaseintegrator.**WikibaseIntegrator**(*is_bot=False*, *login=None*)

Bases: object

Parameters

- **is_bot** (*bool*) –
- **login** (*Optional[_Login]*) –

__init__(*is_bot=False*, *login=None*)

This function initializes a WikibaseIntegrator instance to quickly access different entity type instances.

Parameters

- **is_bot** (bool) – declare if the bot flag must be set when you interact with the MediaWiki API.
- **login** (Optional[_Login]) – a wbi_login instance needed when you try to access a restricted MediaWiki instance.

CHAPTER

TWO

CHANGELOG

2.1 Changelog

2.1.1 v0.12.4

Released on 2023-06-07 - GitHub - PyPI

2.1.2 v0.12.3

Released on 2023-01-09 - GitHub - PyPI

2.1.3 v0.12.2

Released on 2022-11-13 - GitHub - PyPI

2.1.4 v0.12.1

Released on 2022-09-04 - GitHub - PyPI

2.1.5 v0.12.0

Released on 2022-07-14 - GitHub - PyPI

2.1.6 v0.11.3

Released on 2022-07-07 - GitHub - PyPI

2.1.7 v0.12.0rc5

Released on 2022-07-03 - GitHub - PyPI

2.1.8 v0.12.0rc4

Released on 2022-06-02 - GitHub - PyPI

2.1.9 v0.11.2

Released on 2022-04-25 - GitHub - PyPI

2.1.10 v0.12.0rc3

Released on 2022-03-24 - GitHub - PyPI

2.1.11 v0.12.0rc2

Released on 2022-02-18 - GitHub - PyPI

2.1.12 v0.12.0rc1

Released on 2022-01-28 - GitHub - PyPI

2.1.13 v0.12.0.dev7

Released on 2022-01-10 - GitHub - PyPI

2.1.14 v0.12.0.dev6

Released on 2021-11-17 - GitHub - PyPI

2.1.15 v0.11.1

Released on 2021-10-23 - GitHub - PyPI

2.1.16 v0.12.0.dev5

Released on 2021-09-27 - GitHub - PyPI

2.1.17 v0.12.0.dev4

Released on 2021-09-13 - GitHub - PyPI

2.1.18 v0.12.0.dev3

Released on 2021-09-09 - GitHub - PyPI

2.1.19 v0.12.0.dev2

Released on 2021-08-26 - GitHub - PyPI

2.1.20 v0.11.0

Released on 2021-06-06 - GitHub - PyPI

2.1.21 v0.10.1

Released on 2021-05-01 - GitHub - PyPI

2.1.22 v0.10.0

Released on 2021-02-11 - GitHub - PyPI

2.1.23 v0.9.0

Released on 2020-10-09 - GitHub - PyPI

2.1.24 v0.8.2

Released on 2020-08-07 - GitHub - PyPI

2.1.25 v0.8.1

Released on 2020-08-05 - GitHub - PyPI

CHAPTER

THREE

INDICES AND TABLES

- genindex
- modindex
- search

PYTHON MODULE INDEX

W

INDEX

Symbols

A

B

C

D

E

F

G

H

I

K

L

M

N

O

P

Q

R

S

T

U

V

W

Y