
WikibaseIntegrator

Release 0.12.4

LeMyst and WikibaseIntegrator contributors

Jun 07, 2023

CONTENTS

1	wikibaseintegrator	1
1.1	Subpackages	1
1.1.1	wikibaseintegrator.datatypes	1
1.1.1.1	Subpackages	1
1.1.1.1.1	wikibaseintegrator.datatypes.extra	1
1.1.1.2	Submodules	5
1.1.1.2.1	wikibaseintegrator.datatypes.basedatatype	5
1.1.1.2.2	wikibaseintegrator.datatypes.commonsmmedia	7
1.1.1.2.3	wikibaseintegrator.datatypes.externalid	9
1.1.1.2.4	wikibaseintegrator.datatypes.form	11
1.1.1.2.5	wikibaseintegrator.datatypes.geoshape	13
1.1.1.2.6	wikibaseintegrator.datatypes.globecoordinate	15
1.1.1.2.7	wikibaseintegrator.datatypes.item	17
1.1.1.2.8	wikibaseintegrator.datatypes.lexeme	19
1.1.1.2.9	wikibaseintegrator.datatypes.math	21
1.1.1.2.10	wikibaseintegrator.datatypes.monolingualtext	23
1.1.1.2.11	wikibaseintegrator.datatypes.musicalnotation	25
1.1.1.2.12	wikibaseintegrator.datatypes.property	27
1.1.1.2.13	wikibaseintegrator.datatypes.quantity	29
1.1.1.2.14	wikibaseintegrator.datatypes.sense	32
1.1.1.2.15	wikibaseintegrator.datatypes.string	34
1.1.1.2.16	wikibaseintegrator.datatypes.tabulardata	36
1.1.1.2.17	wikibaseintegrator.datatypes.time	38
1.1.1.2.18	wikibaseintegrator.datatypes.url	41
1.1.2	wikibaseintegrator.entities	43
1.1.2.1	wikibaseintegrator.entities.baseentity	43
1.1.2.2	wikibaseintegrator.entities.item	45
1.1.2.3	wikibaseintegrator.entities.lexeme	48
1.1.2.4	wikibaseintegrator.entities.mediainfo	52
1.1.2.5	wikibaseintegrator.entities.property	55
1.1.3	wikibaseintegrator.models	58
1.1.3.1	wikibaseintegrator.models.aliases	58
1.1.3.2	wikibaseintegrator.models.basemodel	60
1.1.3.3	wikibaseintegrator.models.claims	60
1.1.3.4	wikibaseintegrator.models.descriptions	63
1.1.3.5	wikibaseintegrator.models.forms	64
1.1.3.6	wikibaseintegrator.models.labels	67
1.1.3.7	wikibaseintegrator.models.language_values	68
1.1.3.8	wikibaseintegrator.models.lemmas	70
1.1.3.9	wikibaseintegrator.models.qualifiers	72

1.1.3.10	wikibaseintegrator.models.references	73
1.1.3.11	wikibaseintegrator.models.senses	75
1.1.3.12	wikibaseintegrator.models.sitelinks	77
1.1.3.13	wikibaseintegrator.models.snaks	78
1.2	Submodules	80
1.2.1	wikibaseintegrator.wbi_backoff	80
1.2.2	wikibaseintegrator.wbi_config	80
1.2.3	wikibaseintegrator.wbi_enums	80
1.2.4	wikibaseintegrator.wbi_exceptions	82
1.2.5	wikibaseintegrator.wbi_fastrun	85
1.2.6	wikibaseintegrator.wbi_helpers	89
1.2.7	wikibaseintegrator.wbi_login	97
1.2.8	wikibaseintegrator.wikibaseintegrator	102
2	Changelog	103
2.1	Changelog	103
2.1.1	v0.12.4	103
2.1.2	v0.12.3	103
2.1.3	v0.12.2	103
2.1.4	v0.12.1	103
2.1.5	v0.12.0	103
2.1.6	v0.11.3	103
2.1.7	v0.12.0rc5	104
2.1.8	v0.12.0rc4	104
2.1.9	v0.11.2	104
2.1.10	v0.12.0rc3	104
2.1.11	v0.12.0rc2	104
2.1.12	v0.12.0rc1	104
2.1.13	v0.12.0.dev7	104
2.1.14	v0.12.0.dev6	104
2.1.15	v0.11.1	104
2.1.16	v0.12.0.dev5	104
2.1.17	v0.12.0.dev4	105
2.1.18	v0.12.0.dev3	105
2.1.19	v0.12.0.dev2	105
2.1.20	v0.11.0	105
2.1.21	v0.10.1	105
2.1.22	v0.10.0	105
2.1.23	v0.9.0	105
2.1.24	v0.8.2	105
2.1.25	v0.8.1	105
3	Indices and tables	107
	Python Module Index	109
	Index	111

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.commonsmmedia

class `wikibaseintegrator.datatypes.commonsmmedia.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*='I')

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='I'*)

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='I'*)

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='I'*)

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*='I')

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*='I', *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*='I')**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*='I')**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 *wbentity* 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 typeOptional[*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 typeOptional[*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 typeOptional[*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 typeOptional[*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 typeOptional[*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 Wiki-data 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 Wiki-data)
- **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 by 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.

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](#)

INDICES AND TABLES

- `genindex`
- `modindex`
- `search`

PYTHON MODULE INDEX

W

wikibaseintegrator.datatypes.basedatatype, 5
wikibaseintegrator.datatypes.commonsmmedia, 7
wikibaseintegrator.datatypes.externalid, 9
wikibaseintegrator.datatypes.extra.edtf, 1
wikibaseintegrator.datatypes.extra.localmedia, 3
wikibaseintegrator.datatypes.form, 11
wikibaseintegrator.datatypes.geoshape, 13
wikibaseintegrator.datatypes.globecoordinate, 15
wikibaseintegrator.datatypes.item, 17
wikibaseintegrator.datatypes.lexeme, 19
wikibaseintegrator.datatypes.math, 21
wikibaseintegrator.datatypes.monolingualtext, 23
wikibaseintegrator.datatypes.musicalnotation, 25
wikibaseintegrator.datatypes.property, 27
wikibaseintegrator.datatypes.quantity, 29
wikibaseintegrator.datatypes.sense, 32
wikibaseintegrator.datatypes.string, 34
wikibaseintegrator.datatypes.tabulardata, 36
wikibaseintegrator.datatypes.time, 38
wikibaseintegrator.datatypes.url, 41
wikibaseintegrator.entities.baseentity, 43
wikibaseintegrator.entities.item, 45
wikibaseintegrator.entities.lexeme, 48
wikibaseintegrator.entities.mediainfo, 52
wikibaseintegrator.entities.property, 55
wikibaseintegrator.models.aliases, 58
wikibaseintegrator.models.basemodel, 60
wikibaseintegrator.models.claims, 60
wikibaseintegrator.models.descriptions, 63
wikibaseintegrator.models.forms, 64
wikibaseintegrator.models.labels, 67
wikibaseintegrator.models.language_values, 68
wikibaseintegrator.models.lemmas, 70
wikibaseintegrator.models.qualifiers, 72
wikibaseintegrator.models.references, 73
wikibaseintegrator.models.senses, 75
wikibaseintegrator.models.sitelinks, 77
wikibaseintegrator.models.snaks, 78
wikibaseintegrator.wbi_backoff, 80
wikibaseintegrator.wbi_config, 80
wikibaseintegrator.wbi_enums, 80
wikibaseintegrator.wbi_exceptions, 82
wikibaseintegrator.wbi_fastrun, 85
wikibaseintegrator.wbi_helpers, 89
wikibaseintegrator.wbi_login, 97
wikibaseintegrator.wikibaseintegrator, 102

Symbols

<code>__init__()</code>	(<i>wikibaseintegrator.datatypes.basedatatype.BaseDataType</i> method), 5	<code>__init__()</code>	(<i>wikibaseintegrator.datatypes.string.String</i> method), 34
<code>__init__()</code>	(<i>wikibaseintegrator.datatypes.commonsmedia.CommonsMedia</i> method), 7	<code>__init__()</code>	(<i>wikibaseintegrator.datatypes.tabulardata.TabularData</i> method), 36
<code>__init__()</code>	(<i>wikibaseintegrator.datatypes.externalid.ExternalID</i> method), 9	<code>__init__()</code>	(<i>wikibaseintegrator.datatypes.time.Time</i> method), 38
<code>__init__()</code>	(<i>wikibaseintegrator.datatypes.extra.edtf.EDTF</i> method), 1	<code>__init__()</code>	(<i>wikibaseintegrator.datatypes.url.URL</i> method), 41
<code>__init__()</code>	(<i>wikibaseintegrator.datatypes.extra.localmedia.LocalMedia</i> method), 3	<code>__init__()</code>	(<i>wikibaseintegrator.entities.baseentity.BaseEntity</i> method), 43
<code>__init__()</code>	(<i>wikibaseintegrator.datatypes.form.Form</i> method), 11	<code>__init__()</code>	(<i>wikibaseintegrator.entities.item.ItemEntity</i> method), 45
<code>__init__()</code>	(<i>wikibaseintegrator.datatypes.geoshape.GeoShape</i> method), 13	<code>__init__()</code>	(<i>wikibaseintegrator.entities.lexeme.LexemeEntity</i> method), 49
<code>__init__()</code>	(<i>wikibaseintegrator.datatypes.globecoordinate.GlobeCoordinate</i> method), 15	<code>__init__()</code>	(<i>wikibaseintegrator.entities.mediainfo.MediaInfoEntity</i> method), 52
<code>__init__()</code>	(<i>wikibaseintegrator.datatypes.item.Item</i> method), 18	<code>__init__()</code>	(<i>wikibaseintegrator.entities.property.PropertyEntity</i> method), 55
<code>__init__()</code>	(<i>wikibaseintegrator.datatypes.lexeme.Lexeme</i> method), 19	<code>__init__()</code>	(<i>wikibaseintegrator.models.aliases.Alias</i> method), 58
<code>__init__()</code>	(<i>wikibaseintegrator.datatypes.math.Math</i> method), 21	<code>__init__()</code>	(<i>wikibaseintegrator.models.aliases.Aliases</i> method), 59
<code>__init__()</code>	(<i>wikibaseintegrator.datatypes.monolingualtext.MonolingualText</i> method), 23	<code>__init__()</code>	(<i>wikibaseintegrator.models.basemodel.BaseModel</i> method), 60
<code>__init__()</code>	(<i>wikibaseintegrator.datatypes.musicalnotation.MusicalNotation</i> method), 25	<code>__init__()</code>	(<i>wikibaseintegrator.models.claims.Claim</i> method), 60
<code>__init__()</code>	(<i>wikibaseintegrator.datatypes.property.Property</i> method), 27	<code>__init__()</code>	(<i>wikibaseintegrator.models.claims.Claims</i> method), 62
<code>__init__()</code>	(<i>wikibaseintegrator.datatypes.quantity.Quantity</i> method), 29	<code>__init__()</code>	(<i>wikibaseintegrator.models.descriptions.Descriptions</i> method), 63
<code>__init__()</code>	(<i>wikibaseintegrator.datatypes.sense.Sense</i> method), 32	<code>__init__()</code>	(<i>wikibaseintegrator.models.forms.Form</i> method), 64
		<code>__init__()</code>	(<i>wikibaseintegrator.models.forms.Forms</i> method), 65

`__init__()` (*wikibaseintegrator.models.forms.Representations* method), 65
`__init__()` (*wikibaseintegrator.models.labels.Labels* method), 67
`__init__()` (*wikibaseintegrator.models.language_values.LanguageValue* method), 68
`__init__()` (*wikibaseintegrator.models.language_values.LanguageValues* method), 69
`__init__()` (*wikibaseintegrator.models.lemmas.Lemmas* method), 70
`__init__()` (*wikibaseintegrator.models.qualifiers.Qualifiers* method), 72
`__init__()` (*wikibaseintegrator.models.references.Reference* method), 73
`__init__()` (*wikibaseintegrator.models.references.References* method), 73
`__init__()` (*wikibaseintegrator.models.senses.Glosses* method), 75
`__init__()` (*wikibaseintegrator.models.senses.Sense* method), 76
`__init__()` (*wikibaseintegrator.models.senses.Senses* method), 76
`__init__()` (*wikibaseintegrator.models.sitelinks.Sitelink* method), 77
`__init__()` (*wikibaseintegrator.models.sitelinks.Sitelinks* method), 77
`__init__()` (*wikibaseintegrator.models.snaks.Snak* method), 78
`__init__()` (*wikibaseintegrator.models.snaks.Snaks* method), 79
`__init__()` (*wikibaseintegrator.wbi_exceptions.MWApiError* method), 82
`__init__()` (*wikibaseintegrator.wbi_exceptions.MaxRetriesReachedException* method), 83
`__init__()` (*wikibaseintegrator.wbi_exceptions.MissingEntityException* method), 83
`__init__()` (*wikibaseintegrator.wbi_exceptions.ModificationFailed* method), 83
`__init__()` (*wikibaseintegrator.wbi_exceptions.NonExistentEntityError* method), 84
`__init__()` (*wikibaseintegrator.wbi_exceptions.SaveFailed* method), 84

`__init__()` (*wikibaseintegrator.wbi_exceptions.SearchError* method), 85
`__init__()` (*wikibaseintegrator.wbi_fastrun.FastRunContainer* method), 85
`__init__()` (*wikibaseintegrator.wbi_helpers.BColors* method), 89
`__init__()` (*wikibaseintegrator.wbi_login.Clientlogin* method), 97
`__init__()` (*wikibaseintegrator.wbi_login.Login* method), 98
`__init__()` (*wikibaseintegrator.wbi_login.LoginError* method), 99
`__init__()` (*wikibaseintegrator.wbi_login.OAuth1* method), 100
`__init__()` (*wikibaseintegrator.wbi_login.OAuth2* method), 101
`__init__()` (*wikibaseintegrator.wikibaseintegrator.WikibaseIntegrator* method), 102

A

ActionIfExists (*class in wikibaseintegrator.wbi_enums*), 80
add() (*wikibaseintegrator.models.claims.Claims* method), 62
add() (*wikibaseintegrator.models.descriptions.Descriptions* method), 63
add() (*wikibaseintegrator.models.forms.Forms* method), 65
add() (*wikibaseintegrator.models.forms.Representations* method), 66
add() (*wikibaseintegrator.models.labels.Labels* method), 67
add() (*wikibaseintegrator.models.language_values.LanguageValues* method), 69
add() (*wikibaseintegrator.models.lemmas.Lemmas* method), 70
add() (*wikibaseintegrator.models.qualifiers.Qualifiers* method), 72
add() (*wikibaseintegrator.models.references.Reference* method), 73
add() (*wikibaseintegrator.models.references.References* method), 74
add() (*wikibaseintegrator.models.senses.Glosses* method), 75
add() (*wikibaseintegrator.models.senses.Senses* method), 77
add() (*wikibaseintegrator.models.snaks.Snaks* method), 79

`add_claims()` (*wikibaseintegrator.entities.baseentity.BaseEntity* method), 43
`add_claims()` (*wikibaseintegrator.entities.item.ItemEntity* method), 46
`add_claims()` (*wikibaseintegrator.entities.lexeme.LexemeEntity* method), 49
`add_claims()` (*wikibaseintegrator.entities.mediainfo.MediaInfoEntity* method), 52
`add_claims()` (*wikibaseintegrator.entities.property.PropertyEntity* method), 55
`Alias` (*class in wikibaseintegrator.models.aliaes*), 58
`Aliases` (*class in wikibaseintegrator.models.aliaes*), 59
`aliases` (*wikibaseintegrator.entities.item.ItemEntity* property), 46
`aliases` (*wikibaseintegrator.entities.mediainfo.MediaInfoEntity* property), 52
`aliases` (*wikibaseintegrator.entities.property.PropertyEntity* property), 56
`aliases` (*wikibaseintegrator.models.aliaes.Aliases* property), 59
`api` (*wikibaseintegrator.entities.baseentity.BaseEntity* property), 44
`api` (*wikibaseintegrator.entities.item.ItemEntity* property), 46
`api` (*wikibaseintegrator.entities.lexeme.LexemeEntity* property), 49
`api` (*wikibaseintegrator.entities.mediainfo.MediaInfoEntity* property), 52
`api` (*wikibaseintegrator.entities.property.PropertyEntity* property), 56
`APPEND_OR_REPLACE` (*wikibaseintegrator.wbi_enums.ActionIfExists* attribute), 80
`args` (*wikibaseintegrator.wbi_exceptions.MaxRetriesReachedException* attribute), 83
`args` (*wikibaseintegrator.wbi_exceptions.MissingEntityException* attribute), 83
`args` (*wikibaseintegrator.wbi_exceptions.ModificationFailed* attribute), 83
`args` (*wikibaseintegrator.wbi_exceptions.MWApiError* attribute), 82
`args` (*wikibaseintegrator.wbi_exceptions.NonExistentEntityError* attribute), 84
`args` (*wikibaseintegrator.wbi_exceptions.SaveFailed* attribute), 84
`args` (*wikibaseintegrator.wbi_exceptions.SearchError* attribute), 85
`args` (*wikibaseintegrator.wbi_login.LoginError* attribute), 99

B

`BaseDataType` (*class in wikibaseintegrator.datatypes.basedatatype*), 5
`BaseEntity` (*class in wikibaseintegrator.entities.baseentity*), 43
`BaseModel` (*class in wikibaseintegrator.models.basemodel*), 60
`BColors` (*class in wikibaseintegrator.wbi_helpers*), 89
`BILLION_YEARS` (*wikibaseintegrator.wbi_enums.WikibaseDatePrecision* attribute), 81
`BOLD` (*wikibaseintegrator.wbi_helpers.BColors* attribute), 89

C

`CENTURY` (*wikibaseintegrator.wbi_enums.WikibaseDatePrecision* attribute), 81
`check_language_data()` (*wikibaseintegrator.wbi_fastrun.FastRunContainer* method), 86
`Claim` (*class in wikibaseintegrator.models.claims*), 60
`Claims` (*class in wikibaseintegrator.models.claims*), 62
`claims` (*wikibaseintegrator.entities.baseentity.BaseEntity* property), 44
`claims` (*wikibaseintegrator.entities.item.ItemEntity* property), 46
`claims` (*wikibaseintegrator.entities.lexeme.LexemeEntity* property), 49
`claims` (*wikibaseintegrator.entities.mediainfo.MediaInfoEntity* property), 52
`claims` (*wikibaseintegrator.entities.property.PropertyEntity* property), 56
`claims` (*wikibaseintegrator.models.claims.Claims* property), 62
`claims` (*wikibaseintegrator.models.forms.Form* property), 64
`clear()` (*wikibaseintegrator.entities.baseentity.BaseEntity* method), 44
`clear()` (*wikibaseintegrator.entities.item.ItemEntity* method), 46
`clear()` (*wikibaseintegrator.entities.lexeme.LexemeEntity* method), 49
`clear()` (*wikibaseintegrator.entities.mediainfo.MediaInfoEntity* method), 52

<code>clear()</code>	(<i>wikibaseintegrator.entities.property.PropertyEntity</i> method), 56	<code>delete_page()</code>	(in module <i>wikibaseintegrator.wbi_helpers</i>), 89
<code>clear()</code>	(<i>wikibaseintegrator.models.qualifiers.Qualifiers</i> method), 72	DEPRECATED	(<i>wikibaseintegrator.wbi_enums.WikibaseRank</i> attribute), 81
<code>clear()</code>	(<i>wikibaseintegrator.models.references.References</i> method), 74	Descriptions	(class in <i>wikibaseintegrator.models.descriptions</i>), 63
<code>clear()</code>	(<i>wikibaseintegrator.wbi_fastrun.FastRunContainer</i> method), 86	descriptions	(<i>wikibaseintegrator.entities.item.ItemEntity</i> property), 47
<code>Clientlogin</code>	(class in <i>wikibaseintegrator.wbi_login</i>), 97	descriptions	(<i>wikibaseintegrator.entities.mediainfo.MediaInfoEntity</i> property), 53
<code>code</code>	(<i>wikibaseintegrator.wbi_exceptions.ModificationFailed</i> attribute), 83	descriptions	(<i>wikibaseintegrator.entities.property.PropertyEntity</i> property), 56
<code>code</code>	(<i>wikibaseintegrator.wbi_exceptions.MWApiError</i> attribute), 82		
<code>code</code>	(<i>wikibaseintegrator.wbi_exceptions.NonExistentEntity</i> attribute), 84	DTYPE	(<i>wikibaseintegrator.datatypes.basedatatype.BaseDataType</i> attribute), 5
<code>code</code>	(<i>wikibaseintegrator.wbi_exceptions.SaveFailed</i> attribute), 84	DTYPE	(<i>wikibaseintegrator.datatypes.commonsmmedia.CommonsMedia</i> attribute), 7
<code>CommonsMedia</code>	(class in <i>wikibaseintegrator.datatypes.commonsmmedia</i>), 7	DTYPE	(<i>wikibaseintegrator.datatypes.externalid.ExternalID</i> attribute), 9
<code>COMMONSMEDIA</code>	(<i>wikibaseintegrator.wbi_enums.WikibaseDatatype</i> attribute), 80	DTYPE	(<i>wikibaseintegrator.datatypes.extra.edtf.EDTF</i> attribute), 1
<code>continue_oauth()</code>	(<i>wikibaseintegrator.wbi_login.OAuth1</i> method), 100	DTYPE	(<i>wikibaseintegrator.datatypes.extra.localmedia.LocalMedia</i> attribute), 3
D		DTYPE	(<i>wikibaseintegrator.datatypes.form.Form</i> attribute), 11
<code>datatype</code>	(<i>wikibaseintegrator.entities.property.PropertyEntity</i> property), 56	DTYPE	(<i>wikibaseintegrator.datatypes.geoshape.GeoShape</i> attribute), 13
<code>datatype</code>	(<i>wikibaseintegrator.models.snaks.Snak</i> property), 79	DTYPE	(<i>wikibaseintegrator.datatypes.globecoordinate.GlobeCoordinate</i> attribute), 15
<code>datavalue</code>	(<i>wikibaseintegrator.models.snaks.Snak</i> property), 79	DTYPE	(<i>wikibaseintegrator.datatypes.item.Item</i> attribute), 17
<code>DAY</code>	(<i>wikibaseintegrator.wbi_enums.WikibaseDatePrecision</i> attribute), 81	DTYPE	(<i>wikibaseintegrator.datatypes.lexeme.Lexeme</i> attribute), 19
<code>DECADE</code>	(<i>wikibaseintegrator.wbi_enums.WikibaseDatePrecision</i> attribute), 81	DTYPE	(<i>wikibaseintegrator.datatypes.math.Math</i> attribute), 21
<code>delete()</code>	(<i>wikibaseintegrator.entities.baseentity.BaseEntity</i> method), 44	DTYPE	(<i>wikibaseintegrator.datatypes.monolingualtext.MonolingualText</i> attribute), 23
<code>delete()</code>	(<i>wikibaseintegrator.entities.item.ItemEntity</i> method), 46	DTYPE	(<i>wikibaseintegrator.datatypes.musicalnotation.MusicalNotation</i> attribute), 25
<code>delete()</code>	(<i>wikibaseintegrator.entities.lexeme.LexemeEntity</i> method), 49	DTYPE	(<i>wikibaseintegrator.datatypes.property.Property</i> attribute), 27
<code>delete()</code>	(<i>wikibaseintegrator.entities.mediainfo.MediaInfoEntity</i> method), 53	DTYPE	(<i>wikibaseintegrator.datatypes.quantity.Quantity</i> attribute), 29
<code>delete()</code>	(<i>wikibaseintegrator.entities.property.PropertyEntity</i> method),	DTYPE	(<i>wikibaseintegrator.datatypes.sense.Sense</i> at-

	tribute), 32		
DTYPE	(<i>wikibaseintegrator.datatypes.string.String</i> attribute), 34	<code>equals()</code>	(<i>wikibaseintegrator.datatypes.property.Property</i> method), 27
DTYPE	(<i>wikibaseintegrator.datatypes.tabulardata.TabularData</i> attribute), 36	<code>equals()</code>	(<i>wikibaseintegrator.datatypes.quantity.Quantity</i> method), 30
DTYPE	(<i>wikibaseintegrator.datatypes.time.Time</i> attribute), 38	<code>equals()</code>	(<i>wikibaseintegrator.datatypes.sense.Sense</i> method), 32
DTYPE	(<i>wikibaseintegrator.datatypes.url.URL</i> attribute), 41	<code>equals()</code>	(<i>wikibaseintegrator.datatypes.string.String</i> method), 34
DTYPE	(<i>wikibaseintegrator.models.claims.Claim</i> attribute), 60	<code>equals()</code>	(<i>wikibaseintegrator.datatypes.tabulardata.TabularData</i> method), 36
E		<code>equals()</code>	(<i>wikibaseintegrator.datatypes.time.Time</i> method), 38
<code>edit_entity()</code>	(in module <i>wikibaseintegrator.wbi_helpers</i>), 90	<code>equals()</code>	(<i>wikibaseintegrator.datatypes.url.URL</i> method), 41
EDTF	(class in <i>wikibaseintegrator.datatypes.extra.edtf</i>), 1	<code>equals()</code>	(<i>wikibaseintegrator.models.claims.Claim</i> method), 60
ENDC	(<i>wikibaseintegrator.wbi_helpers.BColors</i> attribute), 89	<code>error_dict</code>	(<i>wikibaseintegrator.wbi_exceptions.ModificationFailed</i> attribute), 83
<code>equals()</code>	(<i>wikibaseintegrator.datatypes.basedatatype.BaseDataType</i> method), 5	<code>error_dict</code>	(<i>wikibaseintegrator.wbi_exceptions.MWApiError</i> attribute), 82
<code>equals()</code>	(<i>wikibaseintegrator.datatypes.commonsmedia.CommonsMedia</i> method), 7	<code>error_dict</code>	(<i>wikibaseintegrator.wbi_exceptions.NonExistentEntityError</i> attribute), 84
<code>equals()</code>	(<i>wikibaseintegrator.datatypes.externalid.ExternalID</i> method), 9	<code>error_dict</code>	(<i>wikibaseintegrator.wbi_exceptions.SaveFailed</i> attribute), 84
<code>equals()</code>	(<i>wikibaseintegrator.datatypes.extra.edtf.EDTF</i> method), 1	ETYPE	(<i>wikibaseintegrator.entities.baseentity.BaseEntity</i> attribute), 43
<code>equals()</code>	(<i>wikibaseintegrator.datatypes.extra.localmedia.LocalMedia</i> method), 3	ETYPE	(<i>wikibaseintegrator.entities.item.ItemEntity</i> attribute), 45
<code>equals()</code>	(<i>wikibaseintegrator.datatypes.form.Form</i> method), 11	ETYPE	(<i>wikibaseintegrator.entities.lexeme.LexemeEntity</i> attribute), 49
<code>equals()</code>	(<i>wikibaseintegrator.datatypes.geoshape.GeoShape</i> method), 13	ETYPE	(<i>wikibaseintegrator.entities.mediainfo.MediaInfoEntity</i> attribute), 52
<code>equals()</code>	(<i>wikibaseintegrator.datatypes.globecoordinate.GlobeCoordinate</i> method), 16	ETYPE	(<i>wikibaseintegrator.entities.property.PropertyEntity</i> attribute), 55
<code>equals()</code>	(<i>wikibaseintegrator.datatypes.item.Item</i> method), 18	<code>execute_sparql_query()</code>	(in module <i>wikibaseintegrator.wbi_helpers</i>), 90
<code>equals()</code>	(<i>wikibaseintegrator.datatypes.lexeme.Lexeme</i> method), 20	ExternalID	(class in <i>wikibaseintegrator.datatypes.externalid</i>), 9
<code>equals()</code>	(<i>wikibaseintegrator.datatypes.math.Math</i> method), 21	EXTERNALID	(<i>wikibaseintegrator.wbi_enums.WikibaseDatatype</i> attribute), 80
<code>equals()</code>	(<i>wikibaseintegrator.datatypes.monolingualtext.MonolingualText</i> method), 23	F	
<code>equals()</code>	(<i>wikibaseintegrator.datatypes.musicalnotation.MusicalNotation</i> method), 25	FAIL	(<i>wikibaseintegrator.wbi_helpers.BColors</i> attribute), 89

FastRunContainer (class in *wikibaseintegrator.wbi_fastrun*), 85
FORCE_APPEND (*wikibaseintegrator.wbi_enums.ActionIfExists* attribute), 80
Form (class in *wikibaseintegrator.datatypes.form*), 11
Form (class in *wikibaseintegrator.models.forms*), 64
FORM (*wikibaseintegrator.wbi_enums.WikibaseDatatype* attribute), 81
format2wbi() (in module *wikibaseintegrator.wbi_helpers*), 91
format_amount() (in module *wikibaseintegrator.wbi_helpers*), 91
format_query_results() (*wikibaseintegrator.wbi_fastrun.FastRunContainer* method), 86
Forms (class in *wikibaseintegrator.models.forms*), 65
forms (*wikibaseintegrator.entities.lexeme.LexemeEntity* property), 50
forms (*wikibaseintegrator.models.forms.Forms* property), 65
from_json() (*wikibaseintegrator.datatypes.basedatatype.BaseDataType* method), 5
from_json() (*wikibaseintegrator.datatypes.commonsmedia.CommonsMedia* method), 7
from_json() (*wikibaseintegrator.datatypes.externalid.ExternalID* method), 9
from_json() (*wikibaseintegrator.datatypes.extra.edtf.EDTF* method), 2
from_json() (*wikibaseintegrator.datatypes.extra.localmedia.LocalMedia* method), 3
from_json() (*wikibaseintegrator.datatypes.form.Form* method), 11
from_json() (*wikibaseintegrator.datatypes.geoshape.GeoShape* method), 13
from_json() (*wikibaseintegrator.datatypes.globecoordinate.GlobeCoordinate* method), 16
from_json() (*wikibaseintegrator.datatypes.item.Item* method), 18
from_json() (*wikibaseintegrator.datatypes.lexeme.Lexeme* method), 20
from_json() (*wikibaseintegrator.datatypes.math.Math* method), 22
from_json() (*wikibaseintegrator.datatypes.monolingualtext.MonolingualText* method), 24
from_json() (*wikibaseintegrator.datatypes.musicalnotation.MusicalNotation* method), 26
from_json() (*wikibaseintegrator.datatypes.property.Property* method), 28
from_json() (*wikibaseintegrator.datatypes.quantity.Quantity* method), 30
from_json() (*wikibaseintegrator.datatypes.sense.Sense* method), 32
from_json() (*wikibaseintegrator.datatypes.string.String* method), 34
from_json() (*wikibaseintegrator.datatypes.tabulardata.TabularData* method), 36
from_json() (*wikibaseintegrator.datatypes.time.Time* method), 39
from_json() (*wikibaseintegrator.datatypes.url.URL* method), 41
from_json() (*wikibaseintegrator.entities.baseentity.BaseEntity* method), 44
from_json() (*wikibaseintegrator.entities.item.ItemEntity* method), 47
from_json() (*wikibaseintegrator.entities.lexeme.LexemeEntity* method), 50
from_json() (*wikibaseintegrator.entities.mediainfo.MediaInfoEntity* method), 53
from_json() (*wikibaseintegrator.entities.property.PropertyEntity* method), 56
from_json() (*wikibaseintegrator.models.alias.Alias* method), 58
from_json() (*wikibaseintegrator.models.alias.Aliases* method), 59
from_json() (*wikibaseintegrator.models.claims.Claim* method), 60
from_json() (*wikibaseintegrator.models.claims.Claims* method), 62
from_json() (*wikibaseintegrator.models.descriptions.Descriptions* method), 63
from_json() (*wikibaseintegrator.models.forms.Form* method), 64
from_json() (*wikibaseintegrator.models.forms.Forms* method), 65
from_json() (*wikibaseintegrator.models.forms.Representations* method), 66
from_json() (*wikibaseintegrator.models.labels.Labels* method), 67
from_json() (*wikibaseintegrator.models.language_values.LanguageValue*

method), 68

from_json() (wikibaseintegrator.models.language_values.LanguageValues method), 69

from_json() (wikibaseintegrator.models.lemmas.Lemmas method), 70

from_json() (wikibaseintegrator.models.qualifiers.Qualifiers method), 72

from_json() (wikibaseintegrator.models.references.Reference method), 73

from_json() (wikibaseintegrator.models.references.References method), 74

from_json() (wikibaseintegrator.models.senses.Glosses method), 75

from_json() (wikibaseintegrator.models.senses.Sense method), 76

from_json() (wikibaseintegrator.models.senses.Senses method), 77

from_json() (wikibaseintegrator.models.sitelinks.Sitelinks method), 78

from_json() (wikibaseintegrator.models.snaks.Snak method), 79

from_json() (wikibaseintegrator.models.snaks.Snaks method), 79

fulltext_search() (in module wikibaseintegrator.wbi_helpers), 91

G

generate_edit_credentials() (wikibaseintegrator.wbi_login.Clientlogin method), 97

generate_edit_credentials() (wikibaseintegrator.wbi_login.Login method), 98

generate_edit_credentials() (wikibaseintegrator.wbi_login.OAuth1 method), 100

generate_edit_credentials() (wikibaseintegrator.wbi_login.OAuth2 method), 101

generate_entity_instances() (in module wikibaseintegrator.wbi_helpers), 91

GeoShape (class in wikibaseintegrator.datatypes.geoshape), 13

GEOSHAPE (wikibaseintegrator.wbi_enums.WikibaseDatatype attribute), 81

get() (wikibaseintegrator.entities.item.ItemEntity method), 47

get() (wikibaseintegrator.entities.lexeme.LexemeEntity method), 50

get() (wikibaseintegrator.entities.mediainfo.MediaInfoEntity method), 53

get() (wikibaseintegrator.entities.property.PropertyEntity method), 57

get() (wikibaseintegrator.models.aliases.Aliases method), 59

get() (wikibaseintegrator.models.claims.Claims method), 62

get() (wikibaseintegrator.models.descriptions.Descriptions method), 63

get() (wikibaseintegrator.models.forms.Forms method), 65

get() (wikibaseintegrator.models.forms.Representations method), 66

get() (wikibaseintegrator.models.labels.Labels method), 67

get() (wikibaseintegrator.models.language_values.LanguageValues method), 69

get() (wikibaseintegrator.models.lemmas.Lemmas method), 71

get() (wikibaseintegrator.models.qualifiers.Qualifiers method), 72

get() (wikibaseintegrator.models.references.References method), 74

get() (wikibaseintegrator.models.senses.Glosses method), 75

get() (wikibaseintegrator.models.senses.Senses method), 77

get() (wikibaseintegrator.models.sitelinks.Sitelinks method), 78

get() (wikibaseintegrator.models.snaks.Snaks method), 79

get_all_data() (wikibaseintegrator.wbi_fastrun.FastRunContainer method), 86

get_by_title() (wikibaseintegrator.entities.mediainfo.MediaInfoEntity method), 53

get_conflicting_entity_ids (wikibaseintegrator.wbi_exceptions.ModificationFailed property), 83

get_conflicting_entity_ids (wikibaseintegrator.wbi_exceptions.MWApiError property), 82

get_conflicting_entity_ids (wikibaseintegrator.wbi_exceptions.NonExistentEntityError property), 84

get_conflicting_entity_ids (wikibaseintegrator.wbi_exceptions.SaveFailed property), 84

get_day() (wikibaseintegrator.datatypes.time.Time method), 39

get_edit_cookie() (wikibaseintegra-

`tor.wbi_login.Clientlogin method`), 97
`get_edit_cookie()` (`wikibaseintegrator.wbi_login.Login method`), 99
`get_edit_cookie()` (`wikibaseintegrator.wbi_login.OAuth1 method`), 100
`get_edit_cookie()` (`wikibaseintegrator.wbi_login.OAuth2 method`), 102
`get_edit_token()` (`wikibaseintegrator.wbi_login.Clientlogin method`), 98
`get_edit_token()` (`wikibaseintegrator.wbi_login.Login method`), 99
`get_edit_token()` (`wikibaseintegrator.wbi_login.OAuth1 method`), 101
`get_edit_token()` (`wikibaseintegrator.wbi_login.OAuth2 method`), 102
`get_entity_url()` (`wikibaseintegrator.entities.baseentity.BaseEntity method`), 44
`get_entity_url()` (`wikibaseintegrator.entities.item.ItemEntity method`), 47
`get_entity_url()` (`wikibaseintegrator.entities.lexeme.LexemeEntity method`), 50
`get_entity_url()` (`wikibaseintegrator.entities.mediainfo.MediaInfoEntity method`), 54
`get_entity_url()` (`wikibaseintegrator.entities.property.PropertyEntity method`), 57
`get_fastrun_container()` (in module `wikibaseintegrator.wbi_fastrun`), 88
`get_item()` (`wikibaseintegrator.wbi_fastrun.FastRunContainer method`), 87
`get_items()` (`wikibaseintegrator.wbi_fastrun.FastRunContainer method`), 87
`get_json()` (`wikibaseintegrator.datatypes.basedatatype.BaseDataType method`), 6
`get_json()` (`wikibaseintegrator.datatypes.commonsmmedia.CommonsMedia method`), 7
`get_json()` (`wikibaseintegrator.datatypes.externalid.ExternalID method`), 9
`get_json()` (`wikibaseintegrator.datatypes.extra.edtf.EDTF method`), 2
`get_json()` (`wikibaseintegrator.datatypes.extra.localmedia.LocalMedia method`), 4
`get_json()` (`wikibaseintegrator.datatypes.form.Form method`), 11
`get_json()` (`wikibaseintegrator.datatypes.geoshape.GeoShape method`), 14
`get_json()` (`wikibaseintegrator.datatypes.globecoordinate.GlobeCoordinate method`), 16
`get_json()` (`wikibaseintegrator.datatypes.item.Item method`), 18
`get_json()` (`wikibaseintegrator.datatypes.lexeme.Lexeme method`), 20
`get_json()` (`wikibaseintegrator.datatypes.math.Math method`), 22
`get_json()` (`wikibaseintegrator.datatypes.monolingualtext.MonolingualText method`), 24
`get_json()` (`wikibaseintegrator.datatypes.musicalnotation.MusicalNotation method`), 26
`get_json()` (`wikibaseintegrator.datatypes.property.Property method`), 28
`get_json()` (`wikibaseintegrator.datatypes.quantity.Quantity method`), 30
`get_json()` (`wikibaseintegrator.datatypes.sense.Sense method`), 32
`get_json()` (`wikibaseintegrator.datatypes.string.String method`), 34
`get_json()` (`wikibaseintegrator.datatypes.tabulardata.TabularData method`), 36
`get_json()` (`wikibaseintegrator.datatypes.time.Time method`), 39
`get_json()` (`wikibaseintegrator.datatypes.url.URL method`), 41
`get_json()` (`wikibaseintegrator.entities.baseentity.BaseEntity method`), 45
`get_json()` (`wikibaseintegrator.entities.item.ItemEntity method`), 47
`get_json()` (`wikibaseintegrator.entities.lexeme.LexemeEntity method`), 50
`get_json()` (`wikibaseintegrator.entities.mediainfo.MediaInfoEntity method`), 54
`get_json()` (`wikibaseintegrator.entities.property.PropertyEntity method`), 57
`get_json()` (`wikibaseintegrator.models.alias.Alias method`), 58
`get_json()` (`wikibaseintegrator.models.alias.Aliases method`), 59
`get_json()` (`wikibaseintegrator.models.claims.Claim method`), 61

`get_json()` (*wikibaseintegrator.models.claims.Claims* method), 62
`get_json()` (*wikibaseintegrator.models.descriptions.Descriptions* method), 63
`get_json()` (*wikibaseintegrator.models.forms.Form* method), 65
`get_json()` (*wikibaseintegrator.models.forms.Forms* method), 65
`get_json()` (*wikibaseintegrator.models.forms.Representations* method), 66
`get_json()` (*wikibaseintegrator.models.labels.Labels* method), 67
`get_json()` (*wikibaseintegrator.models.language_values.LanguageValue* method), 68
`get_json()` (*wikibaseintegrator.models.language_values.LanguageValues* method), 70
`get_json()` (*wikibaseintegrator.models.lemmas.Lemmas* method), 71
`get_json()` (*wikibaseintegrator.models.qualifiers.Qualifiers* method), 72
`get_json()` (*wikibaseintegrator.models.references.Reference* method), 73
`get_json()` (*wikibaseintegrator.models.references.References* method), 74
`get_json()` (*wikibaseintegrator.models.senses.Glosses* method), 75
`get_json()` (*wikibaseintegrator.models.senses.Sense* method), 76
`get_json()` (*wikibaseintegrator.models.senses.Senses* method), 77
`get_json()` (*wikibaseintegrator.models.snaks.Snak* method), 79
`get_json()` (*wikibaseintegrator.models.snaks.Snaks* method), 79
`get_language_data()` (*wikibaseintegrator.wbi_fastrun.FastRunContainer* method), 87
`get_languages` (*wikibaseintegrator.wbi_exceptions.ModificationFailed* property), 83
`get_languages` (*wikibaseintegrator.wbi_exceptions.MWApiError* property), 82
`get_languages` (*wikibaseintegrator.wbi_exceptions.NonExistentEntityError* property), 84
`get_languages` (*wikibaseintegrator.wbi_exceptions.SaveFailed* property), 85
`get_month()` (*wikibaseintegrator.datatypes.time.Time* method), 39
`get_prop_datatype()` (*wikibaseintegrator.wbi_fastrun.FastRunContainer* method), 87
`get_session()` (*wikibaseintegrator.wbi_login.Clientlogin* method), 98
`get_session()` (*wikibaseintegrator.wbi_login.Login* method), 99
`get_session()` (*wikibaseintegrator.wbi_login.OAuth1* method), 101
`get_session()` (*wikibaseintegrator.wbi_login.OAuth2* method), 102
`get_sparql_value()` (*wikibaseintegrator.datatypes.basedatatype.BaseDataType* method), 6
`get_sparql_value()` (*wikibaseintegrator.datatypes.commonsmmedia.CommonsMedia* method), 8
`get_sparql_value()` (*wikibaseintegrator.datatypes.externalid.ExternalID* method), 9
`get_sparql_value()` (*wikibaseintegrator.datatypes.extra.edtf.EDTF* method), 2
`get_sparql_value()` (*wikibaseintegrator.datatypes.extra.localmedia.LocalMedia* method), 4
`get_sparql_value()` (*wikibaseintegrator.datatypes.form.Form* method), 12
`get_sparql_value()` (*wikibaseintegrator.datatypes.geoshape.GeoShape* method), 14
`get_sparql_value()` (*wikibaseintegrator.datatypes.globecoordinate.GlobeCoordinate* method), 16
`get_sparql_value()` (*wikibaseintegrator.datatypes.item.Item* method), 18
`get_sparql_value()` (*wikibaseintegrator.datatypes.lexeme.Lexeme* method), 20
`get_sparql_value()` (*wikibaseintegrator.datatypes.math.Math* method), 22
`get_sparql_value()` (*wikibaseintegrator.datatypes.monolingualtext.MonolingualText* method), 24
`get_sparql_value()` (*wikibaseintegrator.datatypes.musicalnotation.MusicalNotation* method), 26
`get_sparql_value()` (*wikibaseintegrator.datatypes.property.Property* method), 28
`get_sparql_value()` (*wikibaseintegrator.datatypes.quantity.Quantity* method),

30
 get_sparql_value() (wikibaseintegrator.datatypes.sense.Sense method), 32
 get_sparql_value() (wikibaseintegrator.datatypes.string.String method), 34
 get_sparql_value() (wikibaseintegrator.datatypes.tabulardata.TabularData method), 36
 get_sparql_value() (wikibaseintegrator.datatypes.time.Time method), 39
 get_sparql_value() (wikibaseintegrator.datatypes.url.URL method), 41
 get_sparql_value() (wikibaseintegrator.models.claims.Claim method), 61
 get_user_agent() (in module wikibaseintegrator.wbi_helpers), 92
 get_year() (wikibaseintegrator.datatypes.time.Time method), 39
 GlobeCoordinate (class in wikibaseintegrator.datatypes.globecoordinate), 15
 GLOBECOORDINATE (wikibaseintegrator.wbi_enums.WikibaseDatatype attribute), 81
 Glosses (class in wikibaseintegrator.models.senses), 75
 grammatical_features (wikibaseintegrator.models.forms.Form property), 65

H

has_equal_qualifiers() (wikibaseintegrator.datatypes.basedatatype.BaseDataType method), 6
 has_equal_qualifiers() (wikibaseintegrator.datatypes.commonsmmedia.CommonsMedia method), 8
 has_equal_qualifiers() (wikibaseintegrator.datatypes.externalid.ExternalID method), 10
 has_equal_qualifiers() (wikibaseintegrator.datatypes.extra.edtf.EDTF method), 2
 has_equal_qualifiers() (wikibaseintegrator.datatypes.extra.localmedia.LocalMedia method), 4
 has_equal_qualifiers() (wikibaseintegrator.datatypes.form.Form method), 12
 has_equal_qualifiers() (wikibaseintegrator.datatypes.geoshape.GeoShape method), 14
 has_equal_qualifiers() (wikibaseintegrator.datatypes.globecoordinate.GlobeCoordinate method), 16
 has_equal_qualifiers() (wikibaseintegrator.datatypes.item.Item method), 18
 has_equal_qualifiers() (wikibaseintegrator.datatypes.lexeme.Lexeme method), 20

has_equal_qualifiers() (wikibaseintegrator.datatypes.math.Math method), 22
 has_equal_qualifiers() (wikibaseintegrator.datatypes.monolingualtext.MonolingualText method), 24
 has_equal_qualifiers() (wikibaseintegrator.datatypes.musicalnotation.MusicalNotation method), 26
 has_equal_qualifiers() (wikibaseintegrator.datatypes.property.Property method), 28
 has_equal_qualifiers() (wikibaseintegrator.datatypes.quantity.Quantity method), 30
 has_equal_qualifiers() (wikibaseintegrator.datatypes.sense.Sense method), 32
 has_equal_qualifiers() (wikibaseintegrator.datatypes.string.String method), 34
 has_equal_qualifiers() (wikibaseintegrator.datatypes.tabulardata.TabularData method), 36
 has_equal_qualifiers() (wikibaseintegrator.datatypes.time.Time method), 39
 has_equal_qualifiers() (wikibaseintegrator.datatypes.url.URL method), 41
 has_equal_qualifiers() (wikibaseintegrator.models.claims.Claim method), 61
 hash (wikibaseintegrator.models.references.Reference property), 73
 hash (wikibaseintegrator.models.snaks.Snak property), 79
 HEADER (wikibaseintegrator.wbi_helpers.BColors attribute), 89
 HUNDRED_THOUSAND_YEARS (wikibaseintegrator.wbi_enums.WikibaseDatePrecision attribute), 81

I

id (wikibaseintegrator.datatypes.basedatatype.BaseDataType property), 6
 id (wikibaseintegrator.datatypes.commonsmmedia.CommonsMedia property), 8
 id (wikibaseintegrator.datatypes.externalid.ExternalID property), 10
 id (wikibaseintegrator.datatypes.extra.edtf.EDTF property), 2
 id (wikibaseintegrator.datatypes.extra.localmedia.LocalMedia property), 4
 id (wikibaseintegrator.datatypes.form.Form property), 12
 id (wikibaseintegrator.datatypes.geoshape.GeoShape property), 14
 id (wikibaseintegrator.datatypes.globecoordinate.GlobeCoordinate property), 16
 id (wikibaseintegrator.datatypes.item.Item property), 18

<code>id</code> (<code>wikibaseintegrator.datatypes.lexeme.Lexeme</code> property), 20	82
<code>id</code> (<code>wikibaseintegrator.datatypes.math.Math</code> property), 22	
<code>id</code> (<code>wikibaseintegrator.datatypes.monolingualtext.MonolingualText</code> property), 24	
<code>id</code> (<code>wikibaseintegrator.datatypes.musicalnotation.MusicalNotation</code> property), 26	
<code>id</code> (<code>wikibaseintegrator.datatypes.property.Property</code> property), 28	
<code>id</code> (<code>wikibaseintegrator.datatypes.quantity.Quantity</code> property), 30	
<code>id</code> (<code>wikibaseintegrator.datatypes.sense.Sense</code> property), 33	
<code>id</code> (<code>wikibaseintegrator.datatypes.string.String</code> property), 35	
<code>id</code> (<code>wikibaseintegrator.datatypes.tabulardata.TabularData</code> property), 37	
<code>id</code> (<code>wikibaseintegrator.datatypes.time.Time</code> property), 39	
<code>id</code> (<code>wikibaseintegrator.datatypes.url.URL</code> property), 42	
<code>id</code> (<code>wikibaseintegrator.entities.baseentity.BaseEntity</code> property), 45	
<code>id</code> (<code>wikibaseintegrator.entities.item.ItemEntity</code> property), 47	
<code>id</code> (<code>wikibaseintegrator.entities.lexeme.LexemeEntity</code> property), 50	
<code>id</code> (<code>wikibaseintegrator.entities.mediainfo.MediaInfoEntity</code> property), 54	
<code>id</code> (<code>wikibaseintegrator.entities.property.PropertyEntity</code> property), 57	
<code>id</code> (<code>wikibaseintegrator.models.claims.Claim</code> property), 61	
<code>id</code> (<code>wikibaseintegrator.models.forms.Form</code> property), 65	
<code>info</code> (<code>wikibaseintegrator.wbi_exceptions.ModificationFailed</code> attribute), 83	
<code>info</code> (<code>wikibaseintegrator.wbi_exceptions.MWApiError</code> attribute), 82	
<code>info</code> (<code>wikibaseintegrator.wbi_exceptions.NonExistentEntityError</code> attribute), 84	
<code>info</code> (<code>wikibaseintegrator.wbi_exceptions.SaveFailed</code> attribute), 85	
<code>init_language_data()</code> (<code>wikibaseintegrator.wbi_fastrun.FastRunContainer</code> method), 88	
<code>Item</code> (class in <code>wikibaseintegrator.datatypes.item</code>), 17	
<code>ITEM</code> (<code>wikibaseintegrator.wbi_enums.WikibaseDatatype</code> attribute), 81	
<code>ItemEntity</code> (class in <code>wikibaseintegrator.entities.item</code>), 45	
K	
<code>KEEP</code> (<code>wikibaseintegrator.wbi_enums.ActionIfExists</code> attribute), 80	
<code>KNOWN_VALUE</code> (<code>wikibaseintegrator.wbi_enums.WikibaseSnakType</code> attribute),	
L	
<code>Labels</code> (class in <code>wikibaseintegrator.models.labels</code>), 67	
<code>labels</code> (<code>wikibaseintegrator.entities.item.ItemEntity</code> property), 47	
<code>labels</code> (<code>wikibaseintegrator.entities.mediainfo.MediaInfoEntity</code> property), 54	
<code>labels</code> (<code>wikibaseintegrator.entities.property.PropertyEntity</code> property), 57	
<code>language</code> (<code>wikibaseintegrator.entities.lexeme.LexemeEntity</code> property), 50	
<code>language</code> (<code>wikibaseintegrator.models.alias.Alias</code> property), 58	
<code>language</code> (<code>wikibaseintegrator.models.language_values.LanguageValue</code> property), 69	
<code>LanguageValue</code> (class in <code>wikibaseintegrator.models.language_values</code>), 68	
<code>LanguageValues</code> (class in <code>wikibaseintegrator.models.language_values</code>), 69	
<code>lastrevid</code> (<code>wikibaseintegrator.entities.baseentity.BaseEntity</code> property), 45	
<code>lastrevid</code> (<code>wikibaseintegrator.entities.item.ItemEntity</code> property), 47	
<code>lastrevid</code> (<code>wikibaseintegrator.entities.lexeme.LexemeEntity</code> property), 51	
<code>lastrevid</code> (<code>wikibaseintegrator.entities.mediainfo.MediaInfoEntity</code> property), 54	
<code>lastrevid</code> (<code>wikibaseintegrator.entities.property.PropertyEntity</code> property), 57	
<code>Lemmas</code> (class in <code>wikibaseintegrator.models.lemmas</code>), 70	
<code>lemmas</code> (<code>wikibaseintegrator.entities.lexeme.LexemeEntity</code> property), 51	
<code>Lexeme</code> (class in <code>wikibaseintegrator.datatypes.lexeme</code>), 19	
<code>LEXEME</code> (<code>wikibaseintegrator.wbi_enums.WikibaseDatatype</code> attribute), 81	
<code>lexeme_add_form()</code> (in module <code>wikibaseintegrator.wbi_helpers</code>), 92	
<code>lexeme_add_sense()</code> (in module <code>wikibaseintegrator.wbi_helpers</code>), 92	
<code>lexeme_edit_form()</code> (in module <code>wikibaseintegrator.wbi_helpers</code>), 93	
<code>lexeme_edit_sense()</code> (in module <code>wikibaseintegrator.wbi_helpers</code>), 93	

`lexeme_remove_form()` (in module `wikibaseintegrator.wbi_helpers`), 93

`lexeme_remove_sense()` (in module `wikibaseintegrator.wbi_helpers`), 94

`LexemeEntity` (class in `wikibaseintegrator.entities.lexeme`), 48

`lexical_category` (`wikibaseintegrator.entities.lexeme.LexemeEntity` property), 51

`LocalMedia` (class in `wikibaseintegrator.datatypes.extra.localmedia`), 3

`Login` (class in `wikibaseintegrator.wbi_login`), 98

`LoginError`, 99

M

`mainsnak` (`wikibaseintegrator.datatypes.basedatatype.BaseDataType` property), 6

`mainsnak` (`wikibaseintegrator.datatypes.commonsmmedia.CommonsMedia` property), 8

`mainsnak` (`wikibaseintegrator.datatypes.externalid.ExternalID` property), 10

`mainsnak` (`wikibaseintegrator.datatypes.extra.edtf.EDTF` property), 2

`mainsnak` (`wikibaseintegrator.datatypes.extra.localmedia.LocalMedia` property), 4

`mainsnak` (`wikibaseintegrator.datatypes.form.Form` property), 12

`mainsnak` (`wikibaseintegrator.datatypes.geoshape.GeoShape` property), 14

`mainsnak` (`wikibaseintegrator.datatypes.globecoordinate.GlobeCoordinate` property), 16

`mainsnak` (`wikibaseintegrator.datatypes.item.Item` property), 18

`mainsnak` (`wikibaseintegrator.datatypes.lexeme.Lexeme` property), 20

`mainsnak` (`wikibaseintegrator.datatypes.math.Math` property), 22

`mainsnak` (`wikibaseintegrator.datatypes.monolingualtext.MonolingualText` property), 24

`mainsnak` (`wikibaseintegrator.datatypes.musicalnotation.MusicalNotation` property), 26

`mainsnak` (`wikibaseintegrator.datatypes.property.Property` property), 28

`mainsnak` (`wikibaseintegrator.datatypes.quantity.Quantity` property),

31

`mainsnak` (`wikibaseintegrator.datatypes.sense.Sense` property), 33

`mainsnak` (`wikibaseintegrator.datatypes.string.String` property), 35

`mainsnak` (`wikibaseintegrator.datatypes.tabulardata.TabularData` property), 37

`mainsnak` (`wikibaseintegrator.datatypes.time.Time` property), 39

`mainsnak` (`wikibaseintegrator.datatypes.url.URL` property), 42

`mainsnak` (`wikibaseintegrator.models.claims.Claim` property), 61

`Math` (class in `wikibaseintegrator.datatypes.math`), 21

`MATH` (`wikibaseintegrator.wbi_enums.WikibaseDatatype` attribute), 81

`MaxRetriesReachedException`, 82

`MediaInfoEntity` (class in `wikibaseintegrator.entities.mediainfo`), 52

`mediawiki_api_call()` (in module `wikibaseintegrator.wbi_helpers`), 94

`mediawiki_api_call_helper()` (in module `wikibaseintegrator.wbi_helpers`), 95

`merge_items()` (in module `wikibaseintegrator.wbi_helpers`), 95

`merge_lexemes()` (in module `wikibaseintegrator.wbi_helpers`), 95

`messages` (`wikibaseintegrator.wbi_exceptions.ModificationFailed` attribute), 83

`messages` (`wikibaseintegrator.wbi_exceptions.MWApiError` attribute), 82

`messages` (`wikibaseintegrator.wbi_exceptions.NonExistentEntityError` attribute), 84

`messages` (`wikibaseintegrator.wbi_exceptions.SaveFailed` attribute), 85

`messages_names` (`wikibaseintegrator.wbi_exceptions.ModificationFailed` attribute), 83

`messages_names` (`wikibaseintegrator.wbi_exceptions.MWApiError` attribute), 82

`messages_names` (`wikibaseintegrator.wbi_exceptions.NonExistentEntityError` attribute), 84

`messages_names` (`wikibaseintegrator.wbi_exceptions.SaveFailed` attribute), 85

`MILLENNIUM` (`wikibaseintegrator.wbi_enums.WikibaseDatePrecision` at-

tribute), 81

MILLION_YEARS (wikibaseintegrator.wbi_enums.WikibaseDatePrecision attribute), 81

MissingEntityException, 83

ModificationFailed, 83

module

- wikibaseintegrator.datatypes.basedatatype, 5
- wikibaseintegrator.datatypes.commonsmmedia, 7
- wikibaseintegrator.datatypes.externalid, 9
- wikibaseintegrator.datatypes.extra.edtf, 1
- wikibaseintegrator.datatypes.extra.localmedia, 3
- wikibaseintegrator.datatypes.form, 11
- wikibaseintegrator.datatypes.geoshape, 13
- wikibaseintegrator.datatypes.globecoordinate, 15
- wikibaseintegrator.datatypes.item, 17
- wikibaseintegrator.datatypes.lexeme, 19
- wikibaseintegrator.datatypes.math, 21
- wikibaseintegrator.datatypes.monolingualtext, 23
- wikibaseintegrator.datatypes.musicalnotation, 25
- wikibaseintegrator.datatypes.property, 27
- wikibaseintegrator.datatypes.quantity, 29
- wikibaseintegrator.datatypes.sense, 32
- wikibaseintegrator.datatypes.string, 34
- wikibaseintegrator.datatypes.tabulardata, 36
- wikibaseintegrator.datatypes.time, 38
- wikibaseintegrator.datatypes.url, 41
- wikibaseintegrator.entities.baseentity, 43
- wikibaseintegrator.entities.item, 45
- wikibaseintegrator.entities.lexeme, 48
- wikibaseintegrator.entities.mediainfo, 52
- wikibaseintegrator.entities.property, 55
- wikibaseintegrator.models.aliases, 58
- wikibaseintegrator.models.basemodel, 60
- wikibaseintegrator.models.claims, 60
- wikibaseintegrator.models.descriptions, 63
- wikibaseintegrator.models.forms, 64
- wikibaseintegrator.models.labels, 67
- wikibaseintegrator.models.language_values, 68
- wikibaseintegrator.models.lemmas, 70
- wikibaseintegrator.models.qualifiers, 72
- wikibaseintegrator.models.references, 73
- wikibaseintegrator.models.senses, 75
- wikibaseintegrator.models.sitelinks, 77
- wikibaseintegrator.models.snaks, 78
- wikibaseintegrator.wbi_backoff, 80
- wikibaseintegrator.wbi_config, 80
- wikibaseintegrator.wbi_enums, 80
- wikibaseintegrator.wbi_exceptions, 82
- wikibaseintegrator.wbi_fastrun, 85
- wikibaseintegrator.wbi_helpers, 89
- wikibaseintegrator.wbi_login, 97
- wikibaseintegrator.wikibaseintegrator, 102

MonolingualText (class in wikibaseintegrator.datatypes.monolingualtext), 23

MONOLINGUALTEXT (wikibaseintegrator.wbi_enums.WikibaseDatatype attribute), 81

MONTH (wikibaseintegrator.wbi_enums.WikibaseDatePrecision attribute), 81

MusicalNotation (class in wikibaseintegrator.datatypes.musicalnotation), 25

MUSICALNOTATION (wikibaseintegrator.wbi_enums.WikibaseDatatype attribute), 81

MWApiError, 82

N

new() (wikibaseintegrator.entities.item.ItemEntity method), 47

new() (wikibaseintegrator.entities.lexeme.LexemeEntity method), 51

new() (wikibaseintegrator.entities.mediainfo.MediaInfoEntity method), 54

new() (wikibaseintegrator.entities.property.PropertyEntity method), 57

NO_VALUE (wikibaseintegrator.wbi_enums.WikibaseSnakType attribute), 82

NonExistentEntityError, 83

NORMAL (wikibaseintegrator.wbi_enums.WikibaseRank attribute), 81

O

OAuth1 (class in wikibaseintegrator.wbi_login), 99

OAuth2 (class in wikibaseintegrator.wbi_login), 101

OKBLUE (wikibaseintegrator.wbi_helpers.BColors attribute), 89

OKCYAN (wikibaseintegrator.wbi_helpers.BColors attribute), 89

OKGREEN (wikibaseintegrator.wbi_helpers.BColors attribute), 89

P

[pageid](#) ([wikibaseintegrator.entities.baseentity.BaseEntity](#) property), [45](#)
[pageid](#) ([wikibaseintegrator.entities.item.ItemEntity](#) property), [48](#)
[pageid](#) ([wikibaseintegrator.entities.lexeme.LexemeEntity](#) property), [51](#)
[pageid](#) ([wikibaseintegrator.entities.mediainfo.MediaInfoEntity](#) property), [54](#)
[pageid](#) ([wikibaseintegrator.entities.property.PropertyEntity](#) property), [57](#)
[parse_sparql_value\(\)](#) ([wikibaseintegrator.datatypes.basedatatype.BaseDataType](#) method), [6](#)
[parse_sparql_value\(\)](#) ([wikibaseintegrator.datatypes.commonsmedia.CommonsMedia](#) method), [8](#)
[parse_sparql_value\(\)](#) ([wikibaseintegrator.datatypes.externalid.ExternalID](#) method), [10](#)
[parse_sparql_value\(\)](#) ([wikibaseintegrator.datatypes.extra.edtf.EDTF](#) method), [2](#)
[parse_sparql_value\(\)](#) ([wikibaseintegrator.datatypes.extra.localmedia.LocalMedia](#) method), [4](#)
[parse_sparql_value\(\)](#) ([wikibaseintegrator.datatypes.form.Form](#) method), [12](#)
[parse_sparql_value\(\)](#) ([wikibaseintegrator.datatypes.geoshape.GeoShape](#) method), [14](#)
[parse_sparql_value\(\)](#) ([wikibaseintegrator.datatypes.globecoordinate.GlobeCoordinate](#) method), [16](#)
[parse_sparql_value\(\)](#) ([wikibaseintegrator.datatypes.item.Item](#) method), [18](#)
[parse_sparql_value\(\)](#) ([wikibaseintegrator.datatypes.lexeme.Lexeme](#) method), [20](#)
[parse_sparql_value\(\)](#) ([wikibaseintegrator.datatypes.math.Math](#) method), [22](#)
[parse_sparql_value\(\)](#) ([wikibaseintegrator.datatypes.monolingualtext.MonolingualText](#) method), [24](#)
[parse_sparql_value\(\)](#) ([wikibaseintegrator.datatypes.musicalnotation.MusicalNotation](#) method), [26](#)
[parse_sparql_value\(\)](#) ([wikibaseintegrator.datatypes.property.Property](#) method), [28](#)
[parse_sparql_value\(\)](#) ([wikibaseintegrator.datatypes.quantity.Quantity](#) method), [31](#)

[parse_sparql_value\(\)](#) ([wikibaseintegrator.datatypes.sense.Sense](#) method), [33](#)
[parse_sparql_value\(\)](#) ([wikibaseintegrator.datatypes.string.String](#) method), [35](#)
[parse_sparql_value\(\)](#) ([wikibaseintegrator.datatypes.tabulardata.TabularData](#) method), [37](#)
[parse_sparql_value\(\)](#) ([wikibaseintegrator.datatypes.time.Time](#) method), [39](#)
[parse_sparql_value\(\)](#) ([wikibaseintegrator.datatypes.url.URL](#) method), [42](#)
[PREFERRED](#) ([wikibaseintegrator.wbi_enums.WikibaseRank](#) attribute), [82](#)
[Property](#) (class in [wikibaseintegrator.datatypes.property](#)), [27](#)
[PROPERTY](#) ([wikibaseintegrator.wbi_enums.WikibaseDatatype](#) attribute), [81](#)
[property_number](#) ([wikibaseintegrator.models.snaks.Snak](#) property), [79](#)
[PropertyEntity](#) (class in [wikibaseintegrator.entities.property](#)), [55](#)

Q

[Qualifiers](#) (class in [wikibaseintegrator.models.qualifiers](#)), [72](#)
[qualifiers](#) ([wikibaseintegrator.datatypes.basedatatype.BaseDataType](#) property), [6](#)
[qualifiers](#) ([wikibaseintegrator.datatypes.commonsmedia.CommonsMedia](#) property), [8](#)
[qualifiers](#) ([wikibaseintegrator.datatypes.externalid.ExternalID](#) property), [10](#)
[qualifiers](#) ([wikibaseintegrator.datatypes.extra.edtf.EDTF](#) property), [2](#)
[qualifiers](#) ([wikibaseintegrator.datatypes.extra.localmedia.LocalMedia](#) property), [4](#)
[qualifiers](#) ([wikibaseintegrator.datatypes.form.Form](#) property), [12](#)
[qualifiers](#) ([wikibaseintegrator.datatypes.geoshape.GeoShape](#) property), [14](#)
[qualifiers](#) ([wikibaseintegrator.datatypes.globecoordinate.GlobeCoordinate](#) property), [16](#)
[qualifiers](#) ([wikibaseintegrator.datatypes.item.Item](#) property), [18](#)
[qualifiers](#) ([wikibaseintegrator.datatypes.lexeme.Lexeme](#) property), [20](#)

- qualifiers** (*wikibaseintegrator.datatypes.math.Math* property), 22
qualifiers (*wikibaseintegrator.datatypes.monolingualtext.MonolingualText* property), 24
qualifiers (*wikibaseintegrator.datatypes.musicalnotation.MusicalNotation* property), 26
qualifiers (*wikibaseintegrator.datatypes.property.Property* property), 28
qualifiers (*wikibaseintegrator.datatypes.quantity.Quantity* property), 31
qualifiers (*wikibaseintegrator.datatypes.sense.Sense* property), 33
qualifiers (*wikibaseintegrator.datatypes.string.String* property), 35
qualifiers (*wikibaseintegrator.datatypes.tabulardata.TabularData* property), 37
qualifiers (*wikibaseintegrator.datatypes.time.Time* property), 39
qualifiers (*wikibaseintegrator.datatypes.url.URL* property), 42
qualifiers (*wikibaseintegrator.models.claims.Claim* property), 61
qualifiers (*wikibaseintegrator.models.qualifiers.Qualifiers* property), 72
qualifiers_order (*wikibaseintegrator.datatypes.basedatatype.BaseDataType* property), 6
qualifiers_order (*wikibaseintegrator.datatypes.commonsmedia.CommonsMedia* property), 8
qualifiers_order (*wikibaseintegrator.datatypes.externalid.ExternalID* property), 10
qualifiers_order (*wikibaseintegrator.datatypes.extra.edtf.EDTF* property), 2
qualifiers_order (*wikibaseintegrator.datatypes.extra.localmedia.LocalMedia* property), 4
qualifiers_order (*wikibaseintegrator.datatypes.form.Form* property), 12
qualifiers_order (*wikibaseintegrator.datatypes.geoshape.GeoShape* property), 14
qualifiers_order (*wikibaseintegrator.datatypes.globecoordinate.GlobeCoordinate* property), 16
qualifiers_order (*wikibaseintegrator.datatypes.item.Item* property), 19
qualifiers_order (*wikibaseintegrator.datatypes.lexeme.Lexeme* property), 20
qualifiers_order (*wikibaseintegrator.datatypes.math.Math* property), 22
qualifiers_order (*wikibaseintegrator.datatypes.monolingualtext.MonolingualText* property), 24
qualifiers_order (*wikibaseintegrator.datatypes.musicalnotation.MusicalNotation* property), 26
qualifiers_order (*wikibaseintegrator.datatypes.property.Property* property), 28
qualifiers_order (*wikibaseintegrator.datatypes.quantity.Quantity* property), 31
qualifiers_order (*wikibaseintegrator.datatypes.sense.Sense* property), 33
qualifiers_order (*wikibaseintegrator.datatypes.string.String* property), 35
qualifiers_order (*wikibaseintegrator.datatypes.tabulardata.TabularData* property), 37
qualifiers_order (*wikibaseintegrator.datatypes.time.Time* property), 40
qualifiers_order (*wikibaseintegrator.datatypes.url.URL* property), 42
qualifiers_order (*wikibaseintegrator.models.claims.Claim* property), 61
Quantity (class in *wikibaseintegrator.datatypes.quantity*), 29
QUANTITY (*wikibaseintegrator.wbi_enums.WikibaseDatatype* attribute), 81
- ## R
- rank** (*wikibaseintegrator.datatypes.basedatatype.BaseDataType* property), 6
rank (*wikibaseintegrator.datatypes.commonsmedia.CommonsMedia* property), 8
rank (*wikibaseintegrator.datatypes.externalid.ExternalID* property), 10
rank (*wikibaseintegrator.datatypes.extra.edtf.EDTF* property), 2
rank (*wikibaseintegrator.datatypes.extra.localmedia.LocalMedia* property), 4
rank (*wikibaseintegrator.datatypes.form.Form* property), 12
rank (*wikibaseintegrator.datatypes.geoshape.GeoShape* property), 14
rank (*wikibaseintegrator.datatypes.globecoordinate.GlobeCoordinate* property), 16

[rank](#) ([wikibaseintegrator.datatypes.item.Item](#) property), 19
[rank](#) ([wikibaseintegrator.datatypes.lexeme.Lexeme](#) property), 20
[rank](#) ([wikibaseintegrator.datatypes.math.Math](#) property), 22
[rank](#) ([wikibaseintegrator.datatypes.monolingualtext.MonolingualText](#) property), 24
[rank](#) ([wikibaseintegrator.datatypes.musicalnotation.MusicalNotation](#) property), 26
[rank](#) ([wikibaseintegrator.datatypes.property.Property](#) property), 28
[rank](#) ([wikibaseintegrator.datatypes.quantity.Quantity](#) property), 31
[rank](#) ([wikibaseintegrator.datatypes.sense.Sense](#) property), 33
[rank](#) ([wikibaseintegrator.datatypes.string.String](#) property), 35
[rank](#) ([wikibaseintegrator.datatypes.tabulardata.TabularData](#) property), 37
[rank](#) ([wikibaseintegrator.datatypes.time.Time](#) property), 40
[rank](#) ([wikibaseintegrator.datatypes.url.URL](#) property), 42
[rank](#) ([wikibaseintegrator.models.claims.Claim](#) property), 61
[reconstruct_statements\(\)](#) ([wikibaseintegrator.wbi_fastrun.FastRunContainer](#) method), 88
[Reference](#) (class in [wikibaseintegrator.models.references](#)), 73
[References](#) (class in [wikibaseintegrator.models.references](#)), 73
[references](#) ([wikibaseintegrator.datatypes.basedatatype.BaseDataType](#) property), 6
[references](#) ([wikibaseintegrator.datatypes.commonsmmedia.CommonsMedia](#) property), 8
[references](#) ([wikibaseintegrator.datatypes.externalid.ExternalID](#) property), 10
[references](#) ([wikibaseintegrator.datatypes.extra.edtf.EDTF](#) property), 2
[references](#) ([wikibaseintegrator.datatypes.extra.localmedia.LocalMedia](#) property), 4
[references](#) ([wikibaseintegrator.datatypes.form.Form](#) property), 12
[references](#) ([wikibaseintegrator.datatypes.geoshape.GeoShape](#) property), 14
[references](#) ([wikibaseintegrator.datatypes.globecoordinate.GlobeCoordinate](#) property), 17
[references](#) ([wikibaseintegrator.datatypes.item.Item](#) property), 19
[references](#) ([wikibaseintegrator.datatypes.lexeme.Lexeme](#) property), 20
[references](#) ([wikibaseintegrator.datatypes.math.Math](#) property), 22
[references](#) ([wikibaseintegrator.datatypes.monolingualtext.MonolingualText](#) property), 24
[references](#) ([wikibaseintegrator.datatypes.musicalnotation.MusicalNotation](#) property), 26
[references](#) ([wikibaseintegrator.datatypes.property.Property](#) property), 28
[references](#) ([wikibaseintegrator.datatypes.quantity.Quantity](#) property), 31
[references](#) ([wikibaseintegrator.datatypes.sense.Sense](#) property), 33
[references](#) ([wikibaseintegrator.datatypes.string.String](#) property), 35
[references](#) ([wikibaseintegrator.datatypes.tabulardata.TabularData](#) property), 37
[references](#) ([wikibaseintegrator.datatypes.time.Time](#) property), 40
[references](#) ([wikibaseintegrator.datatypes.url.URL](#) property), 42
[references](#) ([wikibaseintegrator.models.claims.Claim](#) property), 61
[references](#) ([wikibaseintegrator.models.references.References](#) property), 74
[refs_equal\(\)](#) ([wikibaseintegrator.datatypes.basedatatype.BaseDataType](#) static method), 6
[refs_equal\(\)](#) ([wikibaseintegrator.datatypes.commonsmmedia.CommonsMedia](#) static method), 8
[refs_equal\(\)](#) ([wikibaseintegrator.datatypes.externalid.ExternalID](#) static method), 10
[refs_equal\(\)](#) ([wikibaseintegrator.datatypes.extra.edtf.EDTF](#) static method), 2
[refs_equal\(\)](#) ([wikibaseintegrator.datatypes.extra.localmedia.LocalMedia](#) static method), 4
[refs_equal\(\)](#) ([wikibaseintegrator.datatypes.form.Form](#) static method), 12
[refs_equal\(\)](#) ([wikibaseintegrator.datatypes.geoshape.GeoShape](#) static method), 14

`tor.datatypes.geoshape.GeoShape` static method), 14
`refs_equal()` (`wikibaseintegrator.datatypes.globecoordinate.GlobeCoordinate` static method), 17
`refs_equal()` (`wikibaseintegrator.datatypes.item.Item` static method), 19
`refs_equal()` (`wikibaseintegrator.datatypes.lexeme.Lexeme` static method), 20
`refs_equal()` (`wikibaseintegrator.datatypes.math.Math` static method), 22
`refs_equal()` (`wikibaseintegrator.datatypes.monolingualtext.MonolingualText` static method), 24
`refs_equal()` (`wikibaseintegrator.datatypes.musicalnotation.MusicalNotation` static method), 26
`refs_equal()` (`wikibaseintegrator.datatypes.property.Property` static method), 28
`refs_equal()` (`wikibaseintegrator.datatypes.quantity.Quantity` static method), 31
`refs_equal()` (`wikibaseintegrator.datatypes.sense.Sense` static method), 33
`refs_equal()` (`wikibaseintegrator.datatypes.string.String` static method), 35
`refs_equal()` (`wikibaseintegrator.datatypes.tabulardata.TabularData` static method), 37
`refs_equal()` (`wikibaseintegrator.datatypes.time.Time` static method), 40
`refs_equal()` (`wikibaseintegrator.datatypes.url.URL` static method), 42
`refs_equal()` (`wikibaseintegrator.models.claims.Claim` static method), 61
`remove()` (`wikibaseintegrator.datatypes.basedatatype.BaseDataType` method), 6
`remove()` (`wikibaseintegrator.datatypes.commonsmmedia.CommonsMedia` method), 8
`remove()` (`wikibaseintegrator.datatypes.externalid.ExternalID` method), 10
`remove()` (`wikibaseintegrator.datatypes.extra.edtf.EDTF` method), 2
`remove()` (`wikibaseintegrator.datatypes.extra.localmedia.LocalMedia` method), 4
`remove()` (`wikibaseintegrator.datatypes.form.Form` method), 12
`remove()` (`wikibaseintegrator.datatypes.geoshape.GeoShape` method), 14
`remove()` (`wikibaseintegrator.datatypes.globecoordinate.GlobeCoordinate` method), 17
`remove()` (`wikibaseintegrator.datatypes.item.Item` method), 19
`remove()` (`wikibaseintegrator.datatypes.lexeme.Lexeme` method), 21
`remove()` (`wikibaseintegrator.datatypes.math.Math` method), 23
`remove()` (`wikibaseintegrator.datatypes.monolingualtext.MonolingualText` method), 25
`remove()` (`wikibaseintegrator.datatypes.musicalnotation.MusicalNotation` method), 27
`remove()` (`wikibaseintegrator.datatypes.property.Property` method), 29
`remove()` (`wikibaseintegrator.datatypes.quantity.Quantity` method), 31
`remove()` (`wikibaseintegrator.datatypes.sense.Sense` method), 33
`remove()` (`wikibaseintegrator.datatypes.string.String` method), 35
`remove()` (`wikibaseintegrator.datatypes.tabulardata.TabularData` method), 37
`remove()` (`wikibaseintegrator.datatypes.time.Time` method), 40
`remove()` (`wikibaseintegrator.datatypes.url.URL` method), 42
`remove()` (`wikibaseintegrator.models.alias.Alias` method), 58
`remove()` (`wikibaseintegrator.models.claims.Claim` method), 61
`remove()` (`wikibaseintegrator.models.claims.Claims` method), 62
`remove()` (`wikibaseintegrator.models.language_values.LanguageValue` method), 69
`remove()` (`wikibaseintegrator.models.qualifiers.Qualifiers` method), 72
`remove()` (`wikibaseintegrator.models.references.References` method), 74
`remove()` (`wikibaseintegrator.models.senses.Sense` method), 76
`remove_claims()` (in module `wikibaseintegrator`), 12

- tor.wbi_helpers*), 96
- removed (*wikibaseintegrator.datatypes.basedatatype.BaseDataType* property), 6
- removed (*wikibaseintegrator.datatypes.commonsmmedia.CommonsMedia* property), 8
- removed (*wikibaseintegrator.datatypes.externalid.ExternalID* property), 10
- removed (*wikibaseintegrator.datatypes.extra.edtf.EDTF* property), 2
- removed (*wikibaseintegrator.datatypes.extra.localmedia.LocalMedia* property), 4
- removed (*wikibaseintegrator.datatypes.form.Form* property), 12
- removed (*wikibaseintegrator.datatypes.geoshape.GeoShape* property), 14
- removed (*wikibaseintegrator.datatypes.globecoordinate.GlobeCoordinate* property), 17
- removed (*wikibaseintegrator.datatypes.item.Item* property), 19
- removed (*wikibaseintegrator.datatypes.lexeme.Lexeme* property), 21
- removed (*wikibaseintegrator.datatypes.math.Math* property), 23
- removed (*wikibaseintegrator.datatypes.monolingualtext.MonolingualText* property), 25
- removed (*wikibaseintegrator.datatypes.musicalnotation.MusicalNotation* property), 27
- removed (*wikibaseintegrator.datatypes.property.Property* property), 29
- removed (*wikibaseintegrator.datatypes.quantity.Quantity* property), 31
- removed (*wikibaseintegrator.datatypes.sense.Sense* property), 33
- removed (*wikibaseintegrator.datatypes.string.String* property), 35
- removed (*wikibaseintegrator.datatypes.tabulardata.TabularData* property), 37
- removed (*wikibaseintegrator.datatypes.time.Time* property), 40
- removed (*wikibaseintegrator.datatypes.url.URL* property), 42
- removed (*wikibaseintegrator.models.alias.Alias* property), 58
- removed (*wikibaseintegrator.models.claims.Claim* property), 61
- removed (*wikibaseintegrator.models.language_values.LanguageValue* property), 69
- REPLACE_ALL (*wikibaseintegrator.wbi_enums.ActionIfExists* attribute), 80
- Representations (class in *wikibaseintegrator.models.forms*), 65
- representations (*wikibaseintegrator.models.forms.Form* property), 65
- ## S
- SaveFailed, 84
- search_entities() (in module *wikibaseintegrator.wbi_helpers*), 96
- SearchError, 85
- Sense (class in *wikibaseintegrator.datatypes.sense*), 32
- Sense (class in *wikibaseintegrator.models.senses*), 76
- SENSE (*wikibaseintegrator.wbi_enums.WikibaseDatatype* attribute), 81
- Senses (class in *wikibaseintegrator.models.senses*), 76
- senses (*wikibaseintegrator.entities.lexeme.LexemeEntity* property), 51
- set() (*wikibaseintegrator.models.alias.Aliases* method), 59
- set() (*wikibaseintegrator.models.descriptions.Descriptions* method), 64
- set() (*wikibaseintegrator.models.forms.Representations* method), 66
- set() (*wikibaseintegrator.models.labels.Labels* method), 68
- set() (*wikibaseintegrator.models.language_values.LanguageValues* method), 70
- set() (*wikibaseintegrator.models.lemmas.Lemmas* method), 71
- set() (*wikibaseintegrator.models.qualifiers.Qualifiers* method), 72
- set() (*wikibaseintegrator.models.senses.Glosses* method), 75
- set() (*wikibaseintegrator.models.sitelinks.Sitelinks* method), 78
- set_value() (*wikibaseintegrator.datatypes.basedatatype.BaseDataType* method), 6
- set_value() (*wikibaseintegrator.datatypes.commonsmmedia.CommonsMedia* method), 8
- set_value() (*wikibaseintegrator.datatypes.externalid.ExternalID* method), 10

set_value() (*wikibaseintegrator.datatypes.extra.edtf.EDTF method*), 3
set_value() (*wikibaseintegrator.datatypes.extra.localmedia.LocalMedia method*), 4
set_value() (*wikibaseintegrator.datatypes.form.Form method*), 12
set_value() (*wikibaseintegrator.datatypes.geoshape.GeoShape method*), 14
set_value() (*wikibaseintegrator.datatypes.globecoordinate.GlobeCoordinate method*), 17
set_value() (*wikibaseintegrator.datatypes.item.Item method*), 19
set_value() (*wikibaseintegrator.datatypes.lexeme.Lexeme method*), 21
set_value() (*wikibaseintegrator.datatypes.math.Math method*), 23
set_value() (*wikibaseintegrator.datatypes.monolingualtext.MonolingualText method*), 25
set_value() (*wikibaseintegrator.datatypes.musicalnotation.MusicalNotation method*), 27
set_value() (*wikibaseintegrator.datatypes.property.Property method*), 29
set_value() (*wikibaseintegrator.datatypes.quantity.Quantity method*), 31
set_value() (*wikibaseintegrator.datatypes.sense.Sense method*), 33
set_value() (*wikibaseintegrator.datatypes.string.String method*), 35
set_value() (*wikibaseintegrator.datatypes.tabulardata.TabularData method*), 37
set_value() (*wikibaseintegrator.datatypes.time.Time method*), 40
set_value() (*wikibaseintegrator.datatypes.url.URL method*), 42
Sitelink (*class in wikibaseintegrator.models.sitelinks*), 77
Sitelinks (*class in wikibaseintegrator.models.sitelinks*), 77
sitelinks (*wikibaseintegrator.entities.item.ItemEntity property*), 48
Snak (*class in wikibaseintegrator.models.snaks*), 78
Snaks (*class in wikibaseintegrator.models.snaks*), 79
snaks (*wikibaseintegrator.models.references.Reference property*), 73
snaks_order (*wikibaseintegrator.models.references.Reference property*), 73
snaktype (*wikibaseintegrator.models.snaks.Snak property*), 79
String (*class in wikibaseintegrator.datatypes.string*), 34
STRING (*wikibaseintegrator.wbi_enums.WikibaseDatatype attribute*), 81
T
TabularData (*class in wikibaseintegrator.datatypes.tabulardata*), 36
TABULARDATA (*wikibaseintegrator.wbi_enums.WikibaseDatatype attribute*), 81
Time (*class in wikibaseintegrator.datatypes.time*), 38
TIME (*wikibaseintegrator.wbi_enums.WikibaseDatatype attribute*), 81
title (*wikibaseintegrator.entities.baseentity.BaseEntity property*), 45
title (*wikibaseintegrator.entities.item.ItemEntity property*), 48
title (*wikibaseintegrator.entities.lexeme.LexemeEntity property*), 51
title (*wikibaseintegrator.entities.mediainfo.MediaInfoEntity property*), 54
title (*wikibaseintegrator.entities.property.PropertyEntity property*), 57
type (*wikibaseintegrator.datatypes.basedatatype.BaseDataType property*), 7
type (*wikibaseintegrator.datatypes.commonsmmedia.CommonsMedia property*), 8
type (*wikibaseintegrator.datatypes.externalid.ExternalID property*), 10
type (*wikibaseintegrator.datatypes.extra.edtf.EDTF property*), 3
type (*wikibaseintegrator.datatypes.extra.localmedia.LocalMedia property*), 5
type (*wikibaseintegrator.datatypes.form.Form property*), 12
type (*wikibaseintegrator.datatypes.geoshape.GeoShape property*), 15
type (*wikibaseintegrator.datatypes.globecoordinate.GlobeCoordinate property*), 17
type (*wikibaseintegrator.datatypes.item.Item property*), 19
type (*wikibaseintegrator.datatypes.lexeme.Lexeme property*), 21
type (*wikibaseintegrator.datatypes.math.Math property*), 23
type (*wikibaseintegrator.datatypes.monolingualtext.MonolingualText property*), 25

type (*wikibaseintegrator.datatypes.musicalnotation.MusicalNotation* property), 27
 type (*wikibaseintegrator.datatypes.property.Property* property), 29
 type (*wikibaseintegrator.datatypes.quantity.Quantity* property), 31
 type (*wikibaseintegrator.datatypes.sense.Sense* property), 33
 type (*wikibaseintegrator.datatypes.string.String* property), 35
 type (*wikibaseintegrator.datatypes.tabulardata.TabularData* property), 37
 type (*wikibaseintegrator.datatypes.time.Time* property), 40
 type (*wikibaseintegrator.datatypes.url.URL* property), 42
 type (*wikibaseintegrator.entities.baseentity.BaseEntity* property), 45
 type (*wikibaseintegrator.entities.item.ItemEntity* property), 48
 type (*wikibaseintegrator.entities.lexeme.LexemeEntity* property), 51
 type (*wikibaseintegrator.entities.mediainfo.MediaInfoEntity* property), 54
 type (*wikibaseintegrator.entities.property.PropertyEntity* property), 57
 type (*wikibaseintegrator.models.claims.Claim* property), 61

U

UNDERLINE (*wikibaseintegrator.wbi_helpers.BColors* attribute), 89
 UNKNOWN_VALUE (*wikibaseintegrator.wbi_enums.WikibaseSnakType* attribute), 82
 update() (*wikibaseintegrator.datatypes.basedatatype.BaseDataType* method), 7
 update() (*wikibaseintegrator.datatypes.commonsmmedia.CommonsMedia* method), 8
 update() (*wikibaseintegrator.datatypes.externalid.ExternalID* method), 10
 update() (*wikibaseintegrator.datatypes.extra.edtf.EDTF* method), 3
 update() (*wikibaseintegrator.datatypes.extra.localmedia.LocalMedia* method), 5
 update() (*wikibaseintegrator.datatypes.form.Form* method), 12
 update() (*wikibaseintegrator.datatypes.geoshape.GeoShape* method), 15
 update() (*wikibaseintegrator.datatypes.globecoordinate.GlobeCoordinate* method), 17
 update() (*wikibaseintegrator.datatypes.item.Item* method), 19
 update() (*wikibaseintegrator.datatypes.lexeme.Lexeme* method), 21
 update() (*wikibaseintegrator.datatypes.math.Math* method), 23
 update() (*wikibaseintegrator.datatypes.monolingualtext.MonolingualText* method), 25
 update() (*wikibaseintegrator.datatypes.musicalnotation.MusicalNotation* method), 27
 update() (*wikibaseintegrator.datatypes.property.Property* method), 29
 update() (*wikibaseintegrator.datatypes.quantity.Quantity* method), 31
 update() (*wikibaseintegrator.datatypes.sense.Sense* method), 33
 update() (*wikibaseintegrator.datatypes.string.String* method), 35
 update() (*wikibaseintegrator.datatypes.tabulardata.TabularData* method), 37
 update() (*wikibaseintegrator.datatypes.time.Time* method), 40
 update() (*wikibaseintegrator.datatypes.url.URL* method), 42
 update() (*wikibaseintegrator.models.claims.Claim* method), 61
 update_frc_from_query() (*wikibaseintegrator.wbi_fastrun.FastRunContainer* method), 88
 URL (class in *wikibaseintegrator.datatypes.url*), 41
 URL (*wikibaseintegrator.wbi_enums.WikibaseDatatype* attribute), 81

V

value (*wikibaseintegrator.models.aliases.Alias* property), 59
 value (*wikibaseintegrator.models.language_values.LanguageValue* property), 69
 values (*wikibaseintegrator.models.descriptions.Descriptions* property), 64
 values (*wikibaseintegrator.models.forms.Representations* property), 67

- values (*wikibaseintegrator.models.labels.Labels* property), 68
- values (*wikibaseintegrator.models.language_values.LanguageValues* property), 70
- values (*wikibaseintegrator.models.lemmas.Lemmas* property), 71
- values (*wikibaseintegrator.models.senses.Glosses* property), 76
- W**
- WARNING (*wikibaseintegrator.wbi_helpers.BColors* attribute), 89
- wbi_backoff_backoff_hdlr() (in module *wikibaseintegrator.wbi_backoff*), 80
- wbi_backoff_check_json_decode_error() (in module *wikibaseintegrator.wbi_backoff*), 80
- WikibaseDatatype (class in *wikibaseintegrator.wbi_enums*), 80
- WikibaseDatePrecision (class in *wikibaseintegrator.wbi_enums*), 81
- WikibaseIntegrator (class in *wikibaseintegrator.wikibaseintegrator*), 102
- wikibaseintegrator.datatypes.basedatatype module, 5
- wikibaseintegrator.datatypes.commonsmmedia module, 7
- wikibaseintegrator.datatypes.externalid module, 9
- wikibaseintegrator.datatypes.extra.edtf module, 1
- wikibaseintegrator.datatypes.extra.localmedia module, 3
- wikibaseintegrator.datatypes.form module, 11
- wikibaseintegrator.datatypes.geoshape module, 13
- wikibaseintegrator.datatypes.globecoordinate module, 15
- wikibaseintegrator.datatypes.item module, 17
- wikibaseintegrator.datatypes.lexeme module, 19
- wikibaseintegrator.datatypes.math module, 21
- wikibaseintegrator.datatypes.monolingualtext module, 23
- wikibaseintegrator.datatypes.musicalnotation module, 25
- wikibaseintegrator.datatypes.property module, 27
- wikibaseintegrator.datatypes.quantity module, 29
- wikibaseintegrator.datatypes.sense module, 32
- wikibaseintegrator.datatypes.string module, 34
- wikibaseintegrator.datatypes.tabulardata module, 36
- wikibaseintegrator.datatypes.time module, 38
- wikibaseintegrator.datatypes.url module, 41
- wikibaseintegrator.entities.baseentity module, 43
- wikibaseintegrator.entities.item module, 45
- wikibaseintegrator.entities.lexeme module, 48
- wikibaseintegrator.entities.mediainfo module, 52
- wikibaseintegrator.entities.property module, 55
- wikibaseintegrator.models.alias module, 58
- wikibaseintegrator.models.basemodel module, 60
- wikibaseintegrator.models.claims module, 60
- wikibaseintegrator.models.descriptions module, 63
- wikibaseintegrator.models.forms module, 64
- wikibaseintegrator.models.labels module, 67
- wikibaseintegrator.models.language_values module, 68
- wikibaseintegrator.models.lemmas module, 70
- wikibaseintegrator.models.qualifiers module, 72
- wikibaseintegrator.models.references module, 73
- wikibaseintegrator.models.senses module, 75
- wikibaseintegrator.models.sitelinks module, 77
- wikibaseintegrator.models.snaks module, 78
- wikibaseintegrator.wbi_backoff module, 80
- wikibaseintegrator.wbi_config module, 80
- wikibaseintegrator.wbi_enums module, 80
- wikibaseintegrator.wbi_exceptions module, 82
- wikibaseintegrator.wbi_fastrun module, 82

`module`, 85
`wikibaseintegrator.wbi_helpers`
 `module`, 89
`wikibaseintegrator.wbi_login`
 `module`, 97
`wikibaseintegrator.wikibaseintegrator`
 `module`, 102
`WikibaseRank` (class in `wikibaseintegrator.wbi_enums`), 81
`WikibaseSnakType` (class in `wikibaseintegrator.wbi_enums`), 82
`with_traceback()` (`wikibaseintegrator.wbi_exceptions.MaxRetriesReachedException` method), 83
`with_traceback()` (`wikibaseintegrator.wbi_exceptions.MissingEntityException` method), 83
`with_traceback()` (`wikibaseintegrator.wbi_exceptions.ModificationFailed` method), 83
`with_traceback()` (`wikibaseintegrator.wbi_exceptions.MWApiError` method), 82
`with_traceback()` (`wikibaseintegrator.wbi_exceptions.NonExistentEntityError` method), 84
`with_traceback()` (`wikibaseintegrator.wbi_exceptions.SaveFailed` method), 85
`with_traceback()` (`wikibaseintegrator.wbi_exceptions.SearchError` method), 85
`with_traceback()` (`wikibaseintegrator.wbi_login.LoginError` method), 99
`write()` (`wikibaseintegrator.entities.item.ItemEntity` method), 48
`write()` (`wikibaseintegrator.entities.lexeme.LexemeEntity` method), 51
`write()` (`wikibaseintegrator.entities.mediainfo.MediaInfoEntity` method), 54
`write()` (`wikibaseintegrator.entities.property.PropertyEntity` method), 57
`write_required()` (`wikibaseintegrator.entities.baseentity.BaseEntity` method), 45
`write_required()` (`wikibaseintegrator.entities.item.ItemEntity` method), 48
`write_required()` (`wikibaseintegrator.entities.lexeme.LexemeEntity` method), 51
`write_required()` (`wikibaseintegrator.entities.mediainfo.MediaInfoEntity` method), 55
`write_required()` (`wikibaseintegrator.entities.property.PropertyEntity` method), 58
`write_required()` (`wikibaseintegrator.wbi_fastrun.FastRunContainer` method), 88

Y
`YEAR` (`wikibaseintegrator.wbi_enums.WikibaseDatePrecision` attribute), 81