owmeta Documentation

Release 0.12.4.dev0

owmeta

Jun 19, 2022

CONTENTS

Our main README is available online on Github.[1] This documentation contains additional materials beyond what is covered there.

Contents:

[1] http://github.com/openworm/owmeta

CHAPTER

ONE

OWMETA

1.1 owmeta package

1.1.1 owmeta

OpenWorm Unified Data Abstract Layer.

An introduction to owmeta can be found in the README on our Github page.

1.1.2 Subpackages

owmeta.commands package

Various commands of the same kind as `OWM`, mostly intended as sub-commands of `OWM`.

Submodules

owmeta.commands.biology module

class owmeta.commands.biology.**CellCmd**(*parent*, **args*, ***kwargs*)

Bases: object

Commands for dealing with biological cells

show(*cell_name_or_id*, *context=None*)

Show information about the cell

Parameters

cell_name_or_id
[str] Cell name or Cell URI

context
[str] Context to search in. Optional, defaults to the default context.

owmeta.data_trans package

Data translators

Some `DataSource` and `DataTranslator` types. Some deal with generic file types (e.g., comma-separated values) while others are specific to the format of a kind of file housed in owmeta.

Submodules

owmeta.data_trans.bibtex module

class owmeta.data_trans.bibtex.**BibTexDataSource**(**args*, *no_type_decl=False*, ***kwargs*)

Bases: `DSMixin`, `LocalFileDataSource`

File name
[`DatatypeProperty`] Attribute: `file_name`

Torrent file name
[`DatatypeProperty`] Attribute: `torrent_file_name`

MD5 hash
[`DatatypeProperty`] Attribute: `md5`

SHA-256 hash
[`DatatypeProperty`] Attribute: `sha256`

SHA-512 hash
[`DatatypeProperty`] Attribute: `sha512`

Input source
[`ObjectProperty`] Attribute: `source`

The data source that was translated into this one

Transformation
[`ObjectProperty`] Attribute: `transformation`

Information about the transformation process that created this object

Translation
[`ObjectProperty`] Attribute: `translation`

Information about the translation process that created this object

Description
[`DatatypeProperty`] Attribute: `description`

Free-text describing the data source

class_context = **owmeta_core.context.ClassContext(ident="http://schema.openworm.org/2020/07/sci")**

class owmeta.data_trans.bibtex.**BibTexDataTranslator**(**args*, *no_type_decl=False*, ***kwargs*)

Bases: `DTMixin`, `DataTranslator`

Input type(s): *BibTexDataSource*

Output type(s): *EvidenceDataSource*

input_type

alias of *BibTexDataSource*

output_type

alias of *EvidenceDataSource*

translate()

Notionally, this method takes one or more data sources, and translates them into some other data source that captures essentially the same information, but, possibly, in a different format. Additional sources can be passed in as well for auxiliary information which are not "translated" in their entirety into the output data source. Such auxiliarry data sources should be distinguished from the primary ones in the translation

Parameters

***args**
Input data sources

****kwargs**
Named input data sources

Returns

the output data source

class_context = owmeta_core.context.ClassContext(ident="http://schema.openworm.org/2020/07/sci")

class owmeta.data_trans.bibtex.**EvidenceDataSource**(**args*, *no_type_decl=False*, ***kwargs*)

Bases: DSMixin, DataSource

Context
[ObjectProperty] Attribute: evidence_context

The context

Input source
[ObjectProperty] Attribute: source

The data source that was translated into this one

Transformation
[ObjectProperty] Attribute: transformation

Information about the transformation process that created this object

Translation
[ObjectProperty] Attribute: translation

Information about the translation process that created this object

Description
[DatatypeProperty] Attribute: description

Free-text describing the data source

after_transform()

Called after Transformer.transform.

This method should handle any of the things that should happen for an output data source after Transformer.transform (or Translator.translate). This can include things like flushing output to files, closing file handles, and writing triples in a Context.

NOTE: Be sure to call this method via super() in sub-classes

class_context = owmeta_core.context.ClassContext(ident="http://schema.openworm.org/2020/07/sci")

context_property

"Context", a ObjectProperty: The context

owmeta.data_trans.common_data module

owmeta.data_trans.connections module

class owmeta.data_trans.connections.**ConnectomeCSVDataSource**(**args*, *no_type_decl=False*, ***kwargs*)

Bases: DSMixin, CSVDataSource

A CSV data source whose CSV file describes a neural connectome

Basically, this is just a marker type to indicate what's described in the CSV – there's no consistent schema

Parameters

commit_op
[CommitOp, optional] The operation to use for commiting the file changes. The default is COPY

class_context = owmeta_core.context.ClassContext(ident="http://schema.openworm.org/2020/07/sci/bio")

class owmeta.data_trans.connections.**NeuronConnectomeCSVTranslation**(**args*, *no_type_decl=False*, ***kwargs*)

Bases: GenericTranslation

class_context = owmeta_core.context.ClassContext(ident="http://schema.openworm.org/2020/07/sci/bio")

class owmeta.data_trans.connections.**NeuronConnectomeCSVTranslator**(**args*, *no_type_decl=False*, ***kwargs*)

Bases: DTMixin, CSVDataTranslator

Input type(s): *ConnectomeCSVDataSource*, *DataWithEvidenceDataSource*

Output type(s): *DataWithEvidenceDataSource*

output_type

alias of *DataWithEvidenceDataSource*

translation_type

alias of *NeuronConnectomeCSVTranslation*

make_translation(*sources*)

It's intended that implementations of BaseDataTranslator will override this method to make custom Translations according with how different arguments to *translate* are (or are not) distinguished.

The actual properties of a Translation subclass must be assigned within the *translate* method

Parameters

sources
[tuple] The sources that go into the translation. Sub-classes may choose to pass these to their superclass' make_translation method or not.

Returns

a description of the translation

translate(*data_source*, *neurons_source*, *muscles_source*)

Notionally, this method takes one or more data sources, and translates them into some other data source that captures essentially the same information, but, possibly, in a different format. Additional sources can be passed in as well for auxiliary information which are not "translated" in their entirety into the output data source. Such auxiliarry data sources should be distinguished from the primary ones in the translation

Parameters

***args**
Input data sources

****kwargs**
Named input data sources

Returns

the output data source

class_context = owmeta_core.context.ClassContext(ident="http://schema.openworm.org/2020/07/sci/bio")

class owmeta.data_trans.connections.**NeuronConnectomeSynapseClassTranslation**(**args*, *no_type_decl=False*, ***kwargs*)

Bases: GenericTranslation

class_context = owmeta_core.context.ClassContext(ident="http://schema.openworm.org/2020/07/sci/bio")

class owmeta.data_trans.connections.**NeuronConnectomeSynapseClassTranslator**(**args*, *no_type_decl=False*, ***kwargs*)

Bases: DTMixin, CSVDataTranslator

Adds synapse classes to existing connections

output_type

alias of *DataWithEvidenceDataSource*

translation_type

alias of *NeuronConnectomeSynapseClassTranslation*

make_translation(*sources*)

It's intended that implementations of BaseDataTranslator will override this method to make custom Translations according with how different arguments to *translate* are (or are not) distinguished.

The actual properties of a Translation subclass must be assigned within the *translate* method

Parameters

sources
[tuple] The sources that go into the translation. Sub-classes may choose to pass these to their superclass' make_translation method or not.

Returns

a description of the translation

translate(*data_source*, *neurotransmitter_source*)

Notionally, this method takes one or more data sources, and translates them into some other data source that captures essentially the same information, but, possibly, in a different format. Additional sources can be passed in as well for auxiliary information which are not "translated" in their entirety into the output data source. Such auxiliarry data sources should be distinguished from the primary ones in the translation

Parameters

***args**
Input data sources

****kwargs**
Named input data sources

Returns

the output data source

class_context = owmeta_core.context.ClassContext(ident="http://schema.openworm.org/2020/07/sci/bio")

owmeta.data_trans.context_merge module

class owmeta.data_trans.context_merge.**ContextMergeDataTranslator**(**args*, *no_type_decl=False*, ***kwargs*)

Bases: DTMixin, DataTranslator

Input type(s): owmeta_core.datasource.OneOrMore (*DataWithEvidenceDataSource*)

Output type(s): *DataWithEvidenceDataSource*

output_type

alias of *DataWithEvidenceDataSource*

translate(**sources*)

Notionally, this method takes one or more data sources, and translates them into some other data source that captures essentially the same information, but, possibly, in a different format. Additional sources can be passed in as well for auxiliary information which are not "translated" in their entirety into the output data source. Such auxiliarry data sources should be distinguished from the primary ones in the translation

Parameters

***args**
Input data sources

****kwargs**
Named input data sources

Returns

the output data source

class_context = owmeta_core.context.ClassContext(ident="http://schema.openworm.org/2020/07/sci")

owmeta.data_trans.data_with_evidence_ds module

class owmeta.data_trans.data_with_evidence_ds.**DataWithEvidenceDataSource**(**args*, *no_type_decl=False*, ***kwargs*)

Bases: DSMixin, DataSource

A data source that has an "evidence context" containing statements which support those in its "data context". The data source also has a combined context which imports both the data and evidence contexts. The data and evidence contexts have identifiers based on the data source's identifier and the combined context has the same identifier as the data source.

after_transform()

Called after Transformer.transform.

This method should handle any of the things that should happen for an output data source after Transformer.transform (or Translator.translate). This can include things like flushing output to files, closing file handles, and writing triples in a Context.

NOTE: Be sure to call this method via super() in sub-classes

class_context = owmeta_core.context.ClassContext(ident="http://schema.openworm.org/2020/07/sci")

combined_context_property

"Combined context", a ObjectProperty: Context importing both the data and evidence contexts

data_context_property

"Data context", a ObjectProperty: The context in which primary data for this data source is defined

evidence_context_property

"Evidence context", a ObjectProperty: The context in which evidence for the "Data context" is defined

owmeta.data_trans.neuron_data module

class owmeta.data_trans.neuron_data.**NeuronCSVDataSource**(**args*, *no_type_decl=False*, ***kwargs*)

Bases: DSMixin, CSVDataSource

BibTeX files

[DatatypeProperty] Attribute: *bibtex_files*

List of BibTeX files that are referenced in the csv file by entry ID

CSV file name

[DatatypeProperty] Attribute: csv_file_name

Header column names

[DatatypeProperty] Attribute: csv_header

CSV field delimiter

[DatatypeProperty] Attribute: csv_field_delimiter

Default value: ,

File name

[DatatypeProperty] Attribute: file_name

Torrent file name

[DatatypeProperty] Attribute: torrent_file_name

MD5 hash
[DatatypeProperty] Attribute: md5

SHA-256 hash
[DatatypeProperty] Attribute: sha256

SHA-512 hash
[DatatypeProperty] Attribute: sha512

Input source
[ObjectProperty] Attribute: source

The data source that was translated into this one

Transformation
[ObjectProperty] Attribute: transformation

Information about the transformation process that created this object

Translation
[ObjectProperty] Attribute: translation

Information about the translation process that created this object

Description
[DatatypeProperty] Attribute: description

Free-text describing the data source

Parameters

commit_op
[CommitOp, optional] The operation to use for commiting the file changes. The default is COPY

bibtex_files

"BibTeX files", a DatatypeProperty: List of BibTeX files that are referenced in the csv file by entry ID

class_context = owmeta_core.context.ClassContext(ident="http://schema.openworm.org/2020/07/sci/bio")

class owmeta.data_trans.neuron_data.**NeuronCSVDataTranslator**(**args*, *no_type_decl=False*, ***kwargs*)

Bases: DTMixin, CSVDataTranslator

Input type(s): *NeuronCSVDataSource*

Output type(s): *DataWithEvidenceDataSource*

input_type

alias of *NeuronCSVDataSource*

output_type

alias of *DataWithEvidenceDataSource*

translate(*data_source*)

Notionally, this method takes one or more data sources, and translates them into some other data source that captures essentially the same information, but, possibly, in a different format. Additional sources can be passed in as well for auxiliary information which are not "translated" in their entirety into the output data source. Such auxiliarry data sources should be distinguished from the primary ones in the translation

Parameters

***args**
Input data sources

****kwargs**
Named input data sources

Returns

the output data source

```
class_context =
owmeta_core.context.ClassContext(ident="http://schema.openworm.org/2020/07/sci/bio")
```

owmeta.data_trans.wormatlas module

class owmeta.data_trans.wormatlas.**WormAtlasCellListDataSource**(**args*, *no_type_decl=False*, ***kwargs*)

Bases: DSMixin, CSVDataSource

CSV file name
[DatatypeProperty] Attribute: csv_file_name

Header column names
[DatatypeProperty] Attribute: *csv_header*

CSV field delimiter
[DatatypeProperty] Attribute: *csv_field_delimiter*

Default value: ,

File name
[DatatypeProperty] Attribute: file_name

Torrent file name
[DatatypeProperty] Attribute: torrent_file_name

MD5 hash
[DatatypeProperty] Attribute: md5

SHA-256 hash
[DatatypeProperty] Attribute: sha256

SHA-512 hash
[DatatypeProperty] Attribute: sha512

Input source
[ObjectProperty] Attribute: source

The data source that was translated into this one

Transformation
[ObjectProperty] Attribute: transformation

Information about the transformation process that created this object

Translation
[ObjectProperty] Attribute: translation

Information about the translation process that created this object

Description

[DatatypeProperty] Attribute: description

Free-text describing the data source

Parameters

commit_op

[CommitOp, optional] The operation to use for commiting the file changes. The default is COPY

class_context = owmeta_core.context.ClassContext(ident="http://schema.openworm.org/2020/07/sci/bio")

csv_field_delimiter

"CSV field delimiter", a DatatypeProperty

Default value: ','

csv_header

"Header column names", a DatatypeProperty

class owmeta.data_trans.wormatlas.**WormAtlasCellListDataTranslation**(**args*, *no_type_decl=False*, ***kwargs*)

Bases: GenericTranslation

defined_augment()

This fuction must return False if *identifier_augment()* would raise an IdentifierMissingException. Override it when defining a non-standard identifier for subclasses of DataObjects.

identifier_augment()

Override this method to define an identifier in lieu of one explicity set.

One must also override *defined_augment()* to return True whenever this method could return a valid identifier. IdentifierMissingException should be raised if an identifier cannot be generated by this method.

Raises

IdentifierMissingException

class_context = owmeta_core.context.ClassContext(ident="http://schema.openworm.org/2020/07/sci/bio")

class owmeta.data_trans.wormatlas.**WormAtlasCellListDataTranslator**(**args*, *no_type_decl=False*, ***kwargs*)

Bases: DTMixin, CSVDataTranslator

Input type(s): *WormAtlasCellListDataSource*, *DataWithEvidenceDataSource*

Output type(s): *DataWithEvidenceDataSource*

output_type

alias of *DataWithEvidenceDataSource*

translation_type

alias of *WormAtlasCellListDataTranslation*

make_translation(*sources*)

It's intended that implementations of `BaseDataTranslator` will override this method to make custom `Translations` according with how different arguments to *translate* are (or are not) distinguished.

The actual properties of a `Translation` subclass must be assigned within the *translate* method

Parameters

sources
[tuple] The sources that go into the translation. Sub-classes may choose to pass these to their superclass' make_translation method or not.

Returns

a description of the translation

translate(*data_source*, *neurons_source*)

Notionally, this method takes one or more data sources, and translates them into some other data source that captures essentially the same information, but, possibly, in a different format. Additional sources can be passed in as well for auxiliary information which are not "translated" in their entirety into the output data source. Such auxiliarry data sources should be distinguished from the primary ones in the translation

Parameters

***args**
Input data sources

****kwargs**
Named input data sources

Returns

the output data source

class_context = owmeta_core.context.ClassContext(ident="http://schema.openworm.org/2020/07/sci/bio")

owmeta.data_trans.wormbase module

class owmeta.data_trans.wormbase.**CellWormBaseCSVTranslator**(**args*, *no_type_decl=False*, ***kwargs*)

Bases: `DTMixin`, `CSVDataTranslator`

Input type(s): *WormBaseCSVDataSource*

Output type(s): *DataWithEvidenceDataSource*

input_type

alias of *WormBaseCSVDataSource*

output_type

alias of *DataWithEvidenceDataSource*

translate(*data_source*)

Translate wormbase CSV dump into Cells, Neurons, and Muscles

class_context = owmeta_core.context.ClassContext(ident="http://schema.openworm.org/2020/07/sci/bio")

class owmeta.data_trans.wormbase.**WormBaseCSVDataSource**(**args*, *no_type_decl=False*, ***kwargs*)

Bases: DSMixin, CSVDataSource

CSV file name
[DatatypeProperty] Attribute: csv_file_name

Header column names
[DatatypeProperty] Attribute: *csv_header*

CSV field delimiter
[DatatypeProperty] Attribute: csv_field_delimiter

Default value: ,

File name
[DatatypeProperty] Attribute: file_name

Torrent file name
[DatatypeProperty] Attribute: torrent_file_name

MD5 hash
[DatatypeProperty] Attribute: md5

SHA-256 hash
[DatatypeProperty] Attribute: sha256

SHA-512 hash
[DatatypeProperty] Attribute: sha512

Input source
[ObjectProperty] Attribute: source

The data source that was translated into this one

Transformation
[ObjectProperty] Attribute: transformation

Information about the transformation process that created this object

Translation
[ObjectProperty] Attribute: translation

Information about the translation process that created this object

Description
[DatatypeProperty] Attribute: description

Free-text describing the data source

Parameters

commit_op
[CommitOp, optional] The operation to use for commiting the file changes. The default is COPY

class_context = owmeta_core.context.ClassContext(ident="http://schema.openworm.org/2020/07/sci/bio")

csv_header

"Header column names", a DatatypeProperty

class owmeta.data_trans.wormbase.**WormbaseIonChannelCSVDataSource**(**args*, *no_type_decl=False*, ***kwargs*)

Bases: DSMixin, CSVDataSource

CSV file name
[DatatypeProperty] Attribute: csv_file_name

Header column names
[DatatypeProperty] Attribute: *csv_header*

CSV field delimiter
[DatatypeProperty] Attribute: csv_field_delimiter

Default value: ,

File name
[DatatypeProperty] Attribute: file_name

Torrent file name
[DatatypeProperty] Attribute: torrent_file_name

MD5 hash
[DatatypeProperty] Attribute: md5

SHA-256 hash
[DatatypeProperty] Attribute: sha256

SHA-512 hash
[DatatypeProperty] Attribute: sha512

Input source
[ObjectProperty] Attribute: source

The data source that was translated into this one

Transformation
[ObjectProperty] Attribute: transformation

Information about the transformation process that created this object

Translation
[ObjectProperty] Attribute: translation

Information about the translation process that created this object

Description
[DatatypeProperty] Attribute: description

Free-text describing the data source

Parameters

commit_op
[CommitOp, optional] The operation to use for commiting the file changes. The default is COPY

class_context = owmeta_core.context.ClassContext(ident="http://schema.openworm.org/2020/07/sci/bio")

csv_header

"Header column names", a DatatypeProperty

class owmeta.data_trans.wormbase.**WormbaseIonChannelCSVTranslator**(**args*, *no_type_decl=False*, ***kwargs*)

Bases: DTMixin, CSVDataTranslator

Input type(s): *WormbaseIonChannelCSVDataSource*

Output type(s): *DataWithEvidenceDataSource*

input_type

alias of *WormbaseIonChannelCSVDataSource*

output_type

alias of *DataWithEvidenceDataSource*

translate(*data_source*)

Notionally, this method takes one or more data sources, and translates them into some other data source that captures essentially the same information, but, possibly, in a different format. Additional sources can be passed in as well for auxiliary information which are not "translated" in their entirety into the output data source. Such auxiliarry data sources should be distinguished from the primary ones in the translation

Parameters

***args**
Input data sources

****kwargs**
Named input data sources

Returns

the output data source

class_context = owmeta_core.context.ClassContext(ident="http://schema.openworm.org/2020/07/sci/bio")

class owmeta.data_trans.wormbase.**WormbaseTextMatchCSVDataSource**(**args*, *no_type_decl=False*, ***kwargs*)

Bases: DSMixin, CSVDataSource

initial_cell_column
[DatatypeProperty] Attribute: *initial_cell_column*

The index of the first column with a cell name

cell_type
[DatatypeProperty] Attribute: *cell_type*

The type of cell to be produced

CSV file name
[DatatypeProperty] Attribute: csv_file_name

Header column names
[DatatypeProperty] Attribute: csv_header

CSV field delimiter
[DatatypeProperty] Attribute: csv_field_delimiter

Default value: ,

File name
[DatatypeProperty] Attribute: file_name

Torrent file name
[DatatypeProperty] Attribute: torrent_file_name

MD5 hash
[DatatypeProperty] Attribute: md5

SHA-256 hash
[DatatypeProperty] Attribute: sha256

SHA-512 hash
[DatatypeProperty] Attribute: sha512

Input source
[ObjectProperty] Attribute: source

The data source that was translated into this one

Transformation
[ObjectProperty] Attribute: transformation

Information about the transformation process that created this object

Translation
[ObjectProperty] Attribute: translation

Information about the translation process that created this object

Description
[DatatypeProperty] Attribute: description

Free-text describing the data source

Parameters

commit_op
[CommitOp, optional] The operation to use for commiting the file changes. The default is COPY

cell_type

"cell_type", a DatatypeProperty: The type of cell to be produced

class_context = owmeta_core.context.ClassContext(ident="http://schema.openworm.org/2020/07/sci/bio")

initial_cell_column

"initial_cell_column", a DatatypeProperty: The index of the first column with a cell name

class owmeta.data_trans.wormbase.**WormbaseTextMatchCSVTranslator**(**args*, *no_type_decl=False*, ***kwargs*)

Bases: DTMixin, CSVDataTranslator

Input type(s): *WormbaseTextMatchCSVDataSource*

Output type(s): *DataWithEvidenceDataSource*

input_type

alias of *WormbaseTextMatchCSVDataSource*

output_type

alias of *DataWithEvidenceDataSource*

`translate`(*data_source*)

Notionally, this method takes one or more data sources, and translates them into some other data source that captures essentially the same information, but, possibly, in a different format. Additional sources can be passed in as well for auxiliary information which are not "translated" in their entirety into the output data source. Such auxiliarry data sources should be distinguished from the primary ones in the translation

Parameters

***args**
Input data sources

****kwargs**
Named input data sources

Returns

`the output data source`

`class_context = owmeta_core.context.ClassContext(ident="http://schema.openworm.org/2020/07/sci/bio")`

1.1.3 Submodules

owmeta.bibtex module

owmeta.bibtex.**bibtex_to_document**(*bibtex_entry*, *context=None*)

Takes a single BibTeX entry and translates it into a *Document* object

owmeta.bibtex.**load**(*bibtex_file*)

Load BibTeX records from a file

Parameters

bibtex_file
[file object] File containing one or more BibTex records

Returns

`bibtexparser.bibdatabase.BibDatabase`
Records represented in the string

owmeta.bibtex.**load_from_file_named**(*file_name*)

Loads from a file with the given name

Parameters

file_name
[`str`] Name of the bibtex file to open

Returns

`bibtexparser.bibdatabase.BibDatabase`
Records from the named file

owmeta.bibtex.**loads**(*bibtex_string*)

Load BibTeX records from a string

Parameters

bibtex_string
[`str`] Text of one or more BibTex records

Returns

bibtexparser.bibdatabase.BibDatabase
Records represented in the string

owmeta.bibtex.**parse_bibtex_into_documents**(*file_name*, *context=None*)

Parses BibTeX records into a dictionary of *Document* instances

Parameters

bibtex_file
[file object] File containing one or more BibTeX records

Returns

dict
Document instances from the records in the file

owmeta.bibtex_customizations module

bibtexparser customizations

owmeta.bibtex_customizations.**author**(*record*)

Split author field by the string 'and' into a list of names.

Parameters

record
[dict] the record

Returns

dict
the given record with any updates applied

owmeta.bibtex_customizations.**customizations**(*record*)

Standard owmeta bibtexparser customizations

Includes: *url*, *note_url*, *doi*, *listify*, and *author*

Parameters

record
[dict] the record

Returns

dict
the given record with any updates applied

owmeta.bibtex_customizations.**doi**(*record*)

Adds a doi URI to the record if there's a doi entry in the record

Parameters

record
[dict] the record to update

Returns

dict
the given record with any updates applied

owmeta.bibtex_customizations.**listify**(*record*)

Turns every value in the record into a list except for ENTRYTYPE and ID

owmeta.bibtex_customizations.**listify_one**(*record*, *name*)

If the given field name does not have a list value, then updates the record by turning that value into a list.

Parameters

record
[dict] The record to update

name
[str] The name of the field to turn into a list

Returns

dict
the given record with any updates applied

owmeta.bibtex_customizations.**note_url**(*record*)

Extracts URLs from note entries in the given record

Parameters

record
[dict] the record

Returns

dict
the given record with any updates applied

owmeta.bibtex_customizations.**url**(*record*)

Merges any URL from \url{...} in howpublished, and any existing link or url values in the record and normalizes them into a list in the url field of the record

Parameters

record
[dict] the record

Returns

dict
the given record with any updates applied

owmeta.biology module

class owmeta.biology.**BiologyType**(**args*, *no_type_decl=False*, ***kwargs*)

Bases: DataObject

owmeta.cell module

class owmeta.cell.**Cell**(**args*, *no_type_decl=False*, ***kwargs*)

Bases: *BiologyType*

A biological cell.

All cells with the same *name* are considered to be the same object.

Parameters

name
[str] The name of the cell

lineageName
[str] The lineageName of the cell

Examples

```
>>> from owmeta_core.quantity import Quantity
>>> c = Cell(lineageName="AB plapaaaap",
...     divisionVolume=Quantity("600","(um)^3"))
```

blast()

Return the blast name.

Example:

```
>>> c = Cell(name="ADAL", lineageName='AB ')
>>> c.blast()
'AB'
```

Note that this isn't a Property. It returns the blast cell part of a *lineageName* value.

property description

A description of the cell

property divisionVolume

The volume of the cell at division

property lineageName

The lineageName of the cell

property name

The 'adult' name of the cell typically used by biologists when discussing C. elegans

owmeta.cell_common module

owmeta.channel module

class owmeta.channel.**Channel**(**args*, *no_type_decl=False*, ***kwargs*)

Bases: *BiologyType*

A biological ion channel.

defined_augment()

This fuction must return False if *identifier_augment()* would raise an IdentifierMissingException. Override it when defining a non-standard identifier for subclasses of DataObjects.

identifier_augment()

Override this method to define an identifier in lieu of one explicity set.

One must also override *defined_augment()* to return True whenever this method could return a valid identifier. IdentifierMissingException should be raised if an identifier cannot be generated by this method.

Raises

IdentifierMissingException

property appearsIn

Cell types in which the ion channel has been expressed

property description

A description of the ion channel

property expression_pattern

A pattern of expression of this cell within an organism

property gene_WB_ID

Wormbase ID of the encoding gene

property gene_class

Classification of the encoding gene

property gene_name

Name of the gene that codes for this ion channel

property model

Get experimental models of this ion channel

property models

Alias to *model*

property name

Ion channel's name

property proteins

Proteins associated with this channel

property subfamily

Ion channel's subfamily

class owmeta.channel.**ExpressionPattern**(**args*, *no_type_decl=False*, ***kwargs*)

Bases: *BiologyType*

defined_augment()

This fuction must return False if *identifier_augment()* would raise an IdentifierMissingException. Override it when defining a non-standard identifier for subclasses of DataObjects.

identifier_augment()

Override this method to define an identifier in lieu of one explicity set.

One must also override *defined_augment()* to return True whenever this method could return a valid identifier. IdentifierMissingException should be raised if an identifier cannot be generated by this method.

Raises

IdentifierMissingException

property description

Natural language description of the expression pattern

property wormbaseID

Alias to *wormbaseid*

property wormbaseURL

The URL for the expression pattern in Wormbase

property wormbaseid

The ID for the expression pattern in Wormbase

owmeta.channel_common module

owmeta.channel_common.**CHANNEL_RDF_TYPE = rdflib.term.URIRef('http://schema.openworm.org/2020/07/Channel')**

Shared RDF type for channels

owmeta.channelworm module

class owmeta.channelworm.**ChannelModel**(**args*, *no_type_decl=False*, ***kwargs*)

Bases: DataObject

A model for an ion channel.

There may be multiple models for a single channel.

Example usage:

```
>>> from owmeta_core.quantity import Quantity

# Create a ChannelModel
>>> cm = PatchClampChannelModel(key='ca_boyle',
...     gating='voltage',
...     ion='Ca',
...     conductance=Quantity.parse('10pS'))
```

class_context = owmeta_core.context.ClassContext(ident="http://schema.openworm.org/2020/07/sci/bio")

property conductance

The conductance of this ion channel. This is the initial value, and should be entered as a Quantity object.

property gating

The gating mechanism for this channel (“voltage” or name of ligand(s))

property ion

The type of ion this channel selects for

property modelType

The type of model employed to describe a channel

property neuroML

Property for attaching NeuroML documents to resources

class owmeta.channelworm.**HomologyChannelModel**(**args*, *no_type_decl=False*, ***kwargs*)

Bases: *ChannelModel*

class_context =
owmeta_core.context.ClassContext(ident="http://schema.openworm.org/2020/07/sci/bio")

class owmeta.channelworm.**PatchClampChannelModel**(**args*, *no_type_decl=False*, ***kwargs*)

Bases: *ChannelModel*

class_context =
owmeta_core.context.ClassContext(ident="http://schema.openworm.org/2020/07/sci/bio")

class owmeta.channelworm.**PatchClampExperiment**(**args*, *no_type_decl=False*, ***kwargs*)

Bases: *Experiment*

Store experimental conditions for a patch clamp experiment.

property Ca_concentration

Calcium concentration

property Cl_concentration

Chlorine concentration

property blockers

Channel blockers used for this experiment

property cell

The cell this experiment was performed on

property cell_age

Age of the cell

class_context =
owmeta_core.context.ClassContext(ident="http://schema.openworm.org/2020/07/sci/bio")

property initial_voltage

Starting voltage of the patch clamp

property ion_channel

The ion channel being clamped

property membrane_capacitance

Initial membrane capacitance

property mutants

Type(s) of mutants being used in this experiment

property patch_type

Type of patch clamp being used ('voltage' or 'current')

property pipette_solution

Type of solution in the pipette

owmeta.cli_hints module

owmeta.command module

class owmeta.command.**OWMEvidence**(*parent*)

Bases: object

Commands for evidence

get(*identifier*, *rdf_type=None*)

Retrieves evidence for the given object. If there are multiple types for the object, the evidence for only one type will be shown, but you can specify which type should be used.

Parameters

identifier
[str] The object to show evidence for

rdf_type
[str] Type of the object to show evidence

owmeta.connection module

class owmeta.connection.**Connection**(**args*, *no_type_decl=False*, ***kwargs*)

Bases: *BiologyType*

property number

The weight of the connection

property post_cell

The post-synaptic cell

property pre_cell

The pre-synaptic cell

property synclass

The kind of Neurotransmitter (if any) sent between *pre_cell* and *post_cell*

property syntype

The kind of synaptic connection. ‘gapJunction’ indicates a gap junction and ‘send’ a chemical synapse

property termination

Where the connection terminates. Inferred from type of post_cell at initialization

owmeta.document module

exception owmeta.document.**PubmedRetrievalException**

Bases: Exception

exception owmeta.document.**WormbaseRetrievalException**

Bases: Exception

class owmeta.document.**BaseDocument**(**args*, *no_type_decl=False*, ***kwargs*)

Bases: DataObject

class_context = owmeta_core.context.ClassContext(ident="http://schema.openworm.org/2020/07/sci")

class owmeta.document.**Document**(**args*, *no_type_decl=False*, ***kwargs*)

Bases: *BaseDocument*

A representation of some document.

Possible keys include:

```
pmid, pubmed: a pubmed id or url (e.g., 24098140)
wbid, wormbase: a wormbase id or url (e.g., WBPaper00044287)
doi: a Digitial Object id or url (e.g., s00454-010-9273-0)
uri: a URI specific to the document, preferably usable for accessing
     the document
```

Parameters

bibtex
[str] A string containing a single BibTeX entry. Parsed during initialization, but not saved thereafter. optional

doi
[str] A Digital Object Identifier (DOI). optional

pubmed
[str] A PubMed ID (PMID) or URL that points to a paper. Ignored if 'pmid' is provided. optional

wormbase
[str] An ID or URL from WormBase that points to a record. Ignored if *wbid* or *wormbaseid* are provided. optional

defined_augment()

This fuction must return False if *identifier_augment()* would raise an IdentifierMissingException. Override it when defining a non-standard identifier for subclasses of DataObjects.

identifier_augment()

Override this method to define an identifier in lieu of one explicity set.

One must also override *defined_augment()* to return True whenever this method could return a valid identifier. IdentifierMissingException should be raised if an identifier cannot be generated by this method.

Raises

IdentifierMissingException

update_from_pubmed(*read_size=65536*, ***kwargs*)

Update the document attributes from NCBI Entrez API using the pubmed attribute

Parameters

chunk_size
[int] The number of bytes to pass to `requests.Response.iter_content`. This *may* reduce runtime memory requirements for the request.

****kwargs**
Passed on as arguments to `requests.Session.get`

update_from_wormbase(*replace_existing=False*, ***kwargs*)

Queries WormBase.org for additional data to fill in the *Document*.

If replace_existing is set to `True`, then existing values will be cleared.

Parameters

replace_existing
[bool] Whether to replace values that are already set for a given property

****kwargs**
Passed on as arguments to `requests.Session.get`

property author

An author of the document

class_context = owmeta_core.context.ClassContext(ident="http://schema.openworm.org/2020/07/sci")

property date

Alias to year

property doi

A Digital Object Identifier (DOI), optional

property pmid

A PubMed ID (PMID) that points to a paper

property title

The title of the document

property uri

A non-standard URI for the document

property wbid

An ID from WormBase.org that points to a record, optional

property wormbaseid

An alias to *wbid*

property year

The year (e.g., publication year) of the document

class owmeta.document.**SourcedFrom**(**args*, ***kwargs*)

Bases: `ObjectProperty`

Indicates which document provided the source for an object

owner_type

alias of BaseDataObject

value_type

alias of *BaseDocument*

lazy = True

If True, then the property is not attached to an instance until the property is set or queried.

multiple = False

If True, then the property will only maintain a single staged value at a time. No effort is made to check how many values are stored in the RDF graph.

rdf_type_class = None

owmeta.documentContext module

class owmeta.documentContext.**DocumentContext**(**args*, ***kwargs*)

Bases: Context

A Context that corresponds to a document.

class owmeta.documentContext.**DocumentContextMeta**(*name*, *typ*, *dct*)

Bases: ContextMeta

owmeta.evidence module

exception owmeta.evidence.**EvidenceError**

Bases: Exception

class owmeta.evidence.**Evidence**(**args*, *no_type_decl=False*, ***kwargs*)

Bases: DataObject

A representation which provides evidence, for a group of statements.

Attaching evidence to an set of statements is done like this:

```
>>> from owmeta.connection import Connection
>>> from owmeta.evidence import Evidence
>>> from owmeta_core.context import Context
```

Declare contexts:

```
>>> ACTX = Context(ident="http://example.org/data/some_statements")
>>> BCTX = Context(ident="http://example.org/data/some_other_statements")
>>> EVCTX = Context(ident="http://example.org/data/some_statements#evidence")
```

Make statements in ACTX and BCTX contexts:

```
>>> ACTX(Connection)(pre_cell="VA11", post_cell="VD12", number=3)
>>> BCTX(Connection)(pre_cell="VA11", post_cell="VD12", number=2)
```

In EVCTX, state that a that a certain document supports the set of statements in ACTX, but refutes the set of statements in BCTX:

```
>>> doc = EVCTX(Document)(author='White et al.', date='1986')
>>> EVCTX(Evidence)(reference=doc, supports=ACTX.rdf_object)
>>> EVCTX(Evidence)(reference=doc, refutes=BCTX.rdf_object)
```

Finally, save the contexts:

```
>>> ACTX.save_context()
>>> BCTX.save_context()
>>> EVCTX.save_context()
```

One note about the *reference* predicate: the reference should, ideally, be an unambiguous link to a peer-reviewed piece of scientific literature detailing methods and data analysis that supports the set of statements. However, in gather data from pre-existing sources, going to that level of specificity may be difficult due to deficient query capability at the data source. In such cases, a broader reference, such as a `Website` with information which guides readers to a peer-reviewed article supporting the statement is sufficient.

defined_augment()

This fuction must return False if *identifier_augment()* would raise an `IdentifierMissingException`. Override it when defining a non-standard identifier for subclasses of DataObjects.

identifier_augment()

Override this method to define an identifier in lieu of one explicity set.

One must also override *defined_augment()* to return True whenever this method could return a valid identifier. `IdentifierMissingException` should be raised if an identifier cannot be generated by this method.

Raises

IdentifierMissingException

class_context = owmeta_core.context.ClassContext(ident="http://schema.openworm.org/2020/07/sci")

property reference

The resource providing evidence supporting/refuting the attached context

property refutes

A context naming a set of statements which are refuted by the attached reference

property supports

A context naming a set of statements which are supported by the attached reference

owmeta.evidence.**evidence_for**(*qctx*, *ctx*, *evctx=None*)

Returns an iterable of Evidence

Parameters

qctx

[`object`] an object supported by evidence. If the object is a `Context` with no identifier, then the query considers statements 'staged' (rather than stored) in the context

ctx

[`Context`] Context that bounds where we look for statements about `qctx`. The contexts for statements found in this context are the actual targets of Evidence.supports statements.

evctx
[Context] if the Evidence.supports statements should be looked for somewhere other than ctx, that can be specified in evctx. optional

owmeta.evidence.**query_context**(*graph*, *qctx*)

graph
[rdflib.graph.Graph] Graph where we can find the contexts for statements in qctx

qctx
[owmeta.context.Context] Container for statements

owmeta.experiment module

class owmeta.experiment.**Experiment**(**args*, *no_type_decl=False*, ***kwargs*)

Bases: DataObject

Generic class for storing information about experiments

Should be overridden by specific types of experiments (example: see PatchClampExperiment in channelworm.py).

Overriding classes should have a list called "conditions" that contains the names of experimental conditions for that particular type of experiment. Each of the items in "conditions" should also be either a DatatypeProperty or ObjectProperty for the experiment as well.

get_conditions()

Return conditions and their associated values in a dict.

class_context = owmeta_core.context.ClassContext(ident="http://schema.openworm.org/2020/07/sci")

property reference

Supporting article for this experiment.

owmeta.muscle module

class owmeta.muscle.**BodyWallMuscle**(**args*, *no_type_decl=False*, ***kwargs*)

Bases: *Muscle*

A somatic muscle cell that lies close under the skin and basal lamina of C. elegans and allows the worm to move

class owmeta.muscle.**Muscle**(**args*, *no_type_decl=False*, ***kwargs*)

Bases: *Cell*

A single muscle cell.

See what neurons innervate a muscle:

Example:

```
>>> mdr21 = Muscle('MDR21')
>>> innervates_mdr21 = mdr21.innervatedBy()
>>> len(innervates_mdr21)
4
```

property innervatedBy

Neurons synapsing with this muscle

property neurons

Alias to *innervatedBy*

property receptor

Alias to *receptors*

property receptors

Receptor types expressed by this type of muscle

owmeta.my_neuroml module

class owmeta.my_neuroml.**NeuroML**(**args*, ***kwargs*)

Bases: DataUser

classmethod generate(*o*, *t=2*)

Get a NeuroML object that represents the given object. The `type` determines what content is included in the NeuroML object:

Parameters

- **o** – The object to generate neuroml from
- **t** – The what kind of content should be included in the document - 0=full morphology+biophysics - 1=cell body only+biophysics - 2=full morphology only

Returns

A NeuroML object that represents the given object.

Return type

NeuroMLDocument

classmethod write(*o*, *n*)

Write the given neuroml document object out to a file :param o: The NeuroMLDocument to write :param n: The name of the file to write to

owmeta.network module

class owmeta.network.**Network**(**args*, *no_type_decl=False*, ***kwargs*)

Bases: *BiologyType*

A network of neurons

aneuron(*name*)

Get a neuron by name.

Example:

```
# Grabs the representation of the neuronal network
>>> net = Worm().get_neuron_network()

# Grab a specific neuron
>>> aval = net.aneuron('AVAL')
```

(continues on next page)

(continued from previous page)

```
>>> aval.type()
set([u'interneuron'])
```

Parameters
name – Name of a c. elegans neuron

Returns
Neuron corresponding to the name given

Return type
owmeta.neuron.Neuron

defined_augment()

This fuction must return False if *identifier_augment()* would raise an IdentifierMissingException. Override it when defining a non-standard identifier for subclasses of DataObjects.

identifier_augment()

Override this method to define an identifier in lieu of one explicity set.

One must also override *defined_augment()* to return True whenever this method could return a valid identifier. IdentifierMissingException should be raised if an identifier cannot be generated by this method.

Raises

IdentifierMissingException

interneurons()

Get all interneurons

Returns
A iterable of all interneurons

Return type
iter(*Neuron*)

motor()

Get all motor

Returns
A iterable of all motor neurons

Return type
iter(*Neuron*)

neuron_names()

Gets the complete set of neurons' names in this network.

Example:

```
# Grabs the representation of the neuronal network
>>> net = Worm().get_neuron_network()

>>> len(set(net.neuron_names()))
302
>>> set(net.neuron_names())
set(['VB4', 'PDEL', 'HSNL', 'SIBDR', ... 'RIAL', 'MCR', 'LUAL'])
```

sensory()

Get all sensory neurons

Returns
A iterable of all sensory neurons

Return type
iter(*Neuron*)

property neuron

Returns a set of all Neuron objects in the network

property neurons

Alias to *neuron*

property synapse

Returns a set of all synapses in the network

property synapses

Alias to *synapse*

property worm

The worm connected to the network

owmeta.neuroml module

class owmeta.neuroml.**NeuroMLDocument**(**args*, *no_type_decl=False*, ***kwargs*)

Bases: DataObject

Represents a NeuroML document

The document may be represented literally in the RDF graph using `xml_content` or stored elsewhere and included by reference with *document_url*.

Example:

```
>>> embedded_nml = NeuroMLDocument(key='embedded_ex', content="""\
... <?xml version="1.0" encoding="UTF-8"?>
... <neuroml xmlns="http://www.neuroml.org/schema/neuroml2"
...     xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
...     xsi:schemaLocation="http://www.neuroml.org/schema/neuroml2
...     https://raw.github.com/NeuroML/NeuroML2/master/Schemas/NeuroML2/NeuroML_
↪v2beta.xsd"
...     id="k_slow">
...     <ionChannel id="k_slow" conductance="10pS" type="ionChannelHH" species="k">
...         <notes>K slow channel from Boyle and Cohen 2008</notes>
...         <gateHHtauInf id="n" instances="1">
...             <timeCourse type="fixedTimeCourse" tau="25.0007 ms"/>
...             <steadyState type="HHSigmoidVariable" rate="1" scale="15.8512 mV"␣
↪midpoint="19.8741 mV"/>
...         </gateHHtauInf>
...     </ionChannel>
... </neuroml>""")

>>> external_nml = NeuroMLDocument(ident='external_ex',
...     document_url='')
```

class_context = owmeta_core.context.ClassContext(ident="http://schema.openworm.org/2020/07/sci/bio")

property content

XML content for the document. Should be a complete NeuroML document rather than a fragment.

property document_url

URL where the XML content of the document can be retrieved

class owmeta.neuroml.NeuroMLProperty(*args, **kwargs)

Bases: ObjectProperty

Property for attaching NeuroML documents to resources

value_type

alias of *NeuroMLDocument*

rdf_type_class = None

owmeta.neuron module

class owmeta.neuron.ConnectionProperty(*args, **kwargs)

Bases: CustomProperty

A representation of the connection between neurons. Either a gap junction or a chemical synapse

TODO: Add neurotransmitter type. TODO: Add connection strength

contextualize(*context*)

Return an object with the given context. If the provided `context` is `None`, then `self` MUST be returned unmodified. Prefer to override `contextualize_agument` which will be called from this method.

It is generally not correct to set a field on the object and return the same object as this would change the context for other users of the object. Also, returning a copy of the object is usually inappropriate for mutable objects. Immutable objects may maintain a 'context' property and return a copy of themselves with that property set to the provided `context` argument.

get(*pre_post_or_either='pre'*, ***kwargs*)

Get a list of connections associated with the owning neuron.

Parameters

pre_post_or_either: str
What kind of connection to look for. 'pre': Owner is the source of the connection 'post': Owner is the destination of the connection 'either': Owner is either the source or destination of the connection

Returns

list of Connection

get_terms(*pre_post_or_either='pre'*, ***kwargs*)

Get a list of connection identifiers associated with the owning neuron.

Parameters

pre_post_or_either: str
What kind of connection to look for. 'pre': Owner is the source of the connection 'post': Owner is the destination of the connection 'either': Owner is either the source or destination of the connection

Returns

list of Connection

set(*conn*, ***kwargs*)

Add a connection associated with the owner Neuron

Parameters

conn
[*owmeta.connection.Connection*] connection associated with the owner neuron

Returns

A owmeta.neuron.Connection

class owmeta.neuron.**Neighbor**(**args*, ***kwargs*)

Bases: CustomProperty

contextualize(*context*)

Return an object with the given context. If the provided `context` is `None`, then `self` MUST be returned unmodified. Prefer to override `contextualize_agument` which will be called from this method.

It is generally not correct to set a field on the object and return the same object as this would change the context for other users of the object. Also, returning a copy of the object is usually inappropriate for mutable objects. Immutable objects may maintain a 'context' property and return a copy of themselves with that property set to the provided `context` argument.

get(***kwargs*)

Get a list of neighboring neurons.

Parameters

See parameters for owmeta.connection.Connection

Returns

list of *Neuron*

get_terms(***kwargs*)

Get a list of neighboring neurons.

Parameters

See parameters for owmeta.connection.Connection

Returns

list of *Neuron*

set(*other*, ***kwargs*)

Set the value of this property

Derived classes must override.

class owmeta.neuron.**Neuron**(**args*, *no_type_decl=False*, ***kwargs*)

Bases: *Cell*

A neuron.

See what neurons express some neuropeptide

Example:

```
# Grabs the representation of the neuronal network
>>> net = P.Worm().get_neuron_network()

# Grab a specific neuron
>>> aval = net.aneuron('AVAL')

>>> aval.type()
set([u'interneuron'])

#show how many connections go out of AVAL
>>> aval.connection.count('pre')
77

>>> aval.name()
u'AVAL'

#list all known receptors
>>> sorted(aval.receptors())
[u'GGR-3', u'GLR-1', u'GLR-2', u'GLR-4', u'GLR-5', u'NMR-1', u'NMR-2', u'UNC-8']

#show how many chemical synapses go in and out of AVAL
>>> aval.Syn_degree()
90
```

Parameters

name
[str] The name of the neuron.

Attributes

neighbor
[CustomProperty] Get neurons connected to this neuron if called with no arguments, or with arguments, state that neuronName is a neighbor of this Neuron

connection
[CustomProperty] Get a set of Connection objects describing chemical synapses or gap junctions between this neuron and others

`GJ_degree`()

Get the degree of this neuron for gap junction edges only

Returns
total number of incoming and outgoing gap junctions

Return type
int

`Syn_degree`()

Get the degree of this neuron for chemical synapse edges only

Returns
total number of incoming and outgoing chemical synapses

Return type
int

contextualize(*context*)

Return an object with the given context. If the provided `context` is `None`, then `self` MUST be returned unmodified. Prefer to override `contextualize_agument` which will be called from this method.

It is generally not correct to set a field on the object and return the same object as this would change the context for other users of the object. Also, returning a copy of the object is usually inappropriate for mutable objects. Immutable objects may maintain a 'context' property and return a copy of themselves with that property set to the provided `context` argument.

get_incidents(*type=0*)

Get neurons which synapse at this neuron

property innexin

Innexin types associated with this neuron

property neuropeptide

Name of the gene corresponding to the neuropeptide produced by this neuron

property neurotransmitter

Neurotransmitters associated with this neuron

property receptor

The receptor types associated with this neuron

property receptors

Alias to *receptor*

property type

The neuron type (i.e., sensory, interneuron, motor)

owmeta.plot module

class owmeta.plot.**Plot**(**args*, *no_type_decl=False*, ***kwargs*)

Bases: `DataObject`

Object for storing plot data in owmeta.

Parameters

data
[2D `list` (`list` of `lists`)] List of XY coordinates for this Plot.

Example usage
[:]

```
>>> pl = Plot([[1, 2], [3, 4]])
>>> pl.get_data()
# [[1, 2], [3, 4]]
```

get_data()

Get the data stored for this plot.

set_data(*data*)

Set the data attribute, which is user-facing, as well as the serialized _data_string attribute, which is used for db storage.

class_context = owmeta_core.context.ClassContext(ident="http://schema.openworm.org/2020/07/sci")

owmeta.sources module

owmeta.sources.**owm_data**(*ns*)

Sources based on objects external to owmeta (e.g., files, websites)

owmeta.translators module

owmeta.utils module

Common utilities for translation, massaging data, etc., that don't fit elsewhere in owmeta

owmeta.website module

class owmeta.website.**Website**(**args*, *no_type_decl=False*, ***kwargs*)

Bases: *BaseDocument*

A representation of a website

defined_augment()

This fuction must return False if *identifier_augment()* would raise an `IdentifierMissingException`. Override it when defining a non-standard identifier for subclasses of DataObjects.

identifier_augment()

Override this method to define an identifier in lieu of one explicity set.

One must also override *defined_augment()* to return True whenever this method could return a valid identifier. `IdentifierMissingException` should be raised if an identifier cannot be generated by this method.

Raises

IdentifierMissingException

class_context = owmeta_core.context.ClassContext(ident="http://schema.openworm.org/2020/07/sci")

property title

The official name for the website

property url

A URL for the website

owmeta.worm module

class owmeta.worm.**Worm**(**args*, *no_type_decl=False*, ***kwargs*)

Bases: *BiologyType*

A representation of the whole worm

defined_augment()

True if the name is defined

get_neuron_network()

Return the neuron network of the worm.

Example:

```
# Grabs the representation of the neuronal network
>>> net = P.Worm().get_neuron_network()

# Grab a specific neuron
>>> aval = net.aneuron('AVAL')

>>> aval.type()
set([u'interneuron'])

#show how many connections go out of AVAL
>>> aval.connection.count('pre')
77
```

Returns
An object to work with the network of the worm

Return type
owmeta.Network

get_semantic_net()

Get the underlying semantic network as an RDFLib Graph

Returns
A semantic network containing information about the worm

Return type
rdflib.ConjunctiveGraph

identifier_augment(**args*, ***kwargs*)

Result is derived from the name property

muscles()

Get all Muscle objects attached to the Worm.

Example:

```
>>> muscles = P.Worm().muscles()
>>> len(muscles)
96
```

Returns
A set of all muscles

Return type
set

property cell

A type of cell in the worm

property muscle

A type of muscle which is in the worm

property name

Alias to *scientific_name*

property neuron_network

The neuron network of the worm

property scientific_name

Scientific name for the organism

owmeta.worm_common module

CHAPTER

TWO

FOR USERS

2.1 owmeta Data Sources

The sources of data for owmeta are stored in the OpenWormData repository. A few `DataTranslators` translate these data into common owmeta data sources. You can list these by running:

```
owm source list
```

and you can show some of the properties of a data source by running:

```
owm source show $SOURCE_IDENTIFIER
```

For instance, you can run the following to see the top-level data source, try:

```
owm source show http://openworm.org/data
```

This will print out summary descriptions of the sources that contribute to the main data source.

2.1.1 A Note on owmeta Data

Below, each major element of the worm's anatomy that owmeta stores data on is considered individually. The data being used is tagged by source in a superscript, and the decisions made during the curation process (if any) are described.

2.1.2 Neurons

- Neuron names[2]: Extracted from WormBase. Dynamic version on this google spreadsheet. Staged in this csv file. Parsed by this method.
- Neuron types[1]: Extracted from WormAtlas.org. Staged in this csv file. Parsed by this method.
- Cell descriptions[1]: Extracted from WormAtlas.org. Staged in this tsv file. Parsed by this method.
- Lineage names[1]: Extracted from WormAtlas.org. Dynamic version on this google spreadsheet. Staged in this tsv file. Parsed by this method.

[2]
- Harris, T. W., Antoshechkin, I., Bieri, T., Blasiar, D., Chan, J., Chen, W. J., … Sternberg, P. W. (2010). WormBase: a comprehensive resource for nematode research. Nucleic Acids Research, 38(Database issue), D463–7. http://doi.org/10.1093/nar/gkp952
- Lee, R. Y. N., & Sternberg, P. W. (2003). Building a cell and anatomy ontology of Caenorhabditis elegans. Comparative and Functional Genomics, 4(1), 121–6. http://doi.org/10.1002/cfg.248

[1] Altun, Z.F., Herndon, L.A., Wolkow, C.A., Crocker, C., Lints, R. and Hall, D. H. (2015). WormAtlas. Retrieved from http://www.wormatlas.org - WormAtlas Complete Cell List

- Neurotransmitters[1]: Extracted from WormAtlas.org. Dynamic version on this google spreadsheet. Staged in this csv file. Parsed by this method.
- Neuropeptides[Page 41, 1]: Extracted from WormAtlas.org. Dynamic version on this google spreadsheet. Staged in this csv file. Parsed by this method.
- Receptors[Page 41, 1]: Extracted from WormAtlas.org. Dynamic version on this google spreadsheet. Staged in this csv file. Parsed by this method.
- Innexins[Page 41, 1]: Extracted from WormAtlas.org. Dynamic version on this google spreadsheet. Staged in this csv file. Parsed by this method.

Gene expression data below, additional to that extracted from WormAtlas concerning receptors, neuropeptides, neurotransmitters and innexins are parsed by this method:

- Monoamine secretors and receptors, neuropeptide secretors and receptors[4]: Dynamic version on this google spreadsheet. Staged in this csv file.

2.1.3 Muscle cells

- Muscle names[Page 41, 2]: Extracted from WormBase. Dynamic version on this google spreadsheet. Staged in this csv file. Parsed by this method.
- Cell descriptions[Page 41, 1]: Extracted from WormAtlas.org. Dynamic version on this google spreadsheet. Staged in this tsv file. Parsed by this method.
- Lineage names[Page 41, 1]: Extracted from WormAtlas.org. Dynamic version on this google spreadsheet. Staged in this tsv file. Parsed by this method.
- Neurons that innervate each muscle[3]: Extracted from data personally communicated by S. Cook. Staged in this csv file. Parsed by this method.

2.1.4 Connectome

- Gap junctions between neurons[3]: Extracted from data personally communicated by S. Cook. Staged in this csv file. Parsed by this method.
- Synapses between neurons[3]: Extracted from data personally communicated by S. Cook. Staged in this csv file. Parsed by this method.

Curation note

There was another source of *C. elegans* connectome data that was created by members of the OpenWorm project that has since been retired. The history of this spreadsheet is mostly contained in this forum post We decided to use the Emmons data set[3] as the authoritative source for connectome data, as it is the very latest version and updated version of the *C. elegans* connectome that we are familiar with.

---

[4] Bentley B., Branicky R., Barnes C. L., Chew Y. L., Yemini E., Bullmore E. T., Vertes P. E., Schafer W. R. (2016) The Multilayer Connectome of Caenorhabditis elegans. PLoS Comput Biol 12(12): e1005283. http://doi.org/10.1371/journal.pcbi.1005283

[3] Emmons, S., Cook, S., Jarrell, T., Wang, Y., Yakolev, M., Nguyen, K., Hall, D. Whole-animal C. elegans connectomes. C. Elegans Meeting 2015 http://abstracts.genetics-gsa.org/cgi-bin/celegans15s/wsrch15.pl?author=emmons&sort=ptimes&sbutton=Detail&absno=155110844&sid=668862

2.1.5 Data Source References

2.2 Requirements for data storage in OpenWorm

Our OpenWorm database captures facts about *C. elegans*. The database stores data for generating model files and together with annotations describing the origins of the data. Below are a set of recommendations for implementation of the database organized around an RDF (Resource Description Framework) model.

2.2.1 Interface

Access is through a Python library which communicates with the database. This library serves the function of providing an object oriented view on the database that can be accessed through the Python scripts commonly used in the project. The *api* is described separately.

2.2.2 Data modeling

Biophysical and anatomical data are included in the database. A sketch of some features of the data model is below. Also included in our model are the relationships between these types. Given our choice of data types, we do not model the individual interactions between cells as entities in the database. Rather these are described by generic predicates in an RDF triple. For instance, neuron A synapsing with muscle cell B would give a statement (A, synapsesWith, B), but A synapsing with neuron C would also have (A, synapsesWith, C).

Nervous system

For the worm's nervous system, we capture a few important data types (listed *below*). These correspond primarily to the anatomical structures and chemicals which are necessary for the worm to record external and internal stimuli and activate its body in response to those stimuli.

Data types

A non-exhaustive list of neurological data types in our C. elegans database:

- receptor types identified in the nerve cell
- neurons
- ion channels
- neurotransmitters
- muscle receptors

Development

C. elegans has very stable cell division patterns in the absence of mutations. This means that we can capture divisions in our database as static 'daughterOf' relationships. The theory of differentiation codes additionally gives an algorithmic description to the growth patterns of the worm which describes signals transmitted between developing cells. In order to test this theory we would like to leverage existing photographic data indicating the volume of cells at the time of their division as this relates to the differentiation code stored by the cell. Progress on this issue is documented on GitHub

Aging

Concurrently with development, we would like to begin modeling the effects of aging on the worm. Aging typically manifests in physiological changes due to transcription errors or cell death. These physiological changes can be represented abstractly as parameters to the function of biological entities. See GitHub for further discussion.

2.2.3 Information assurance

Provenance

Tracking the origins of facts stated in the database demands a method of annotating statements in our database. Providing citations for facts must be as simple as providing a global identifier (e.g., URI, DOI) or a local identifier (e.g., Bibtex identifier, Pubmed ID). With owmeta, supporting information can be attached to named graphs, which are groupings of statements with a URI attached to them. A named graph can have as many or as few statements as desired. Furthermore, a given triple can occur in multiple named graphs. Further details for the attachment of evidence using this technique are given in the *api*.

In line with current practices for communication through the source code management platform, GitHub, we track responsibility for new uploads to the database through the OpenWormData Git repository. Each named graph is canonicalized – essentially, triples are sorted and written to a text file – and committed to a Git repository which gives us, at least, an email address and a timestamp for all modifications.

Access control

Data in owmeta are distributed as a bundle, a packaging structure which contains a set of canonicalized named graphs and, optionally, some files. Responsibility for restricting who can modify a bundle is, in the first instance, up to the bundle creator. When the bundle is actually distributed, the responsibility then falls on the distributor to ensure authentication of the bundle's provider and integrity of the bundle.

In OpenWorm, we create bundles from the OpenWormData GitHub repository. Access to the repository is managed by senior OpenWorm contributors. Bundles are deployed to Google Drive with write access controlled by Mark Watts. You can fetch OpenWorm bundles by adding a remote like this:

```
owm bundle remote add google-drive 'https://drive.google.com/uc?
↪id=1NYAcKdcvoFu5c7Nz3l4hK5UacG_eD56V&authuser=0&export=download'
```

`google-drive` can be substituted with any string.

2.2.4 Miscellaneous

Versioning

Experimental methods are constantly improving in biological research. These improvements may require updating the data we reference or store internally. However, in making updates we must not immediately expunge older content, breaking links created by internal and external agents. Instead, we utilize bundle versioning to track revisions to the data. Each successive release of the bundle increments the bundle version number.

2.2.5 Why RDF?

RDF offers advantages in resilience to schema additions and increased flexibility in integrating data from disparate sources.[1] These qualities can be valued by comparison to relational database systems. Typically, schema changes in a relational database require extensive work for applications using it.[2] In the author's experience, RDF databases offer more freedom in restructuring. Also, for data integration, SPARQL, the standard language for querying over RDF has Federated queries which allow for nearly painless integration of external SPARQL endpoints with existing queries.

The advantage of local storage of the database that goes along with each copy of the library is that the data will have the version number of the library. This means that data can be 'deprecated' along with a deprecated version of the library. This also will prevent changes made to a volatile database that break downstream code that uses the library.

2.3 Adding Data to *YOUR* OpenWorm Database

So, you've got some biological data about the worm and you'd like to save it in owmeta, but you don't know how it's done?

You've come to the right place!

A few biological entities (e.g., Cell, Neuron, Muscle, Worm) are pre-coded into owmeta. The full list is available in the *API*. If these entities already cover your use-case, then all you need to do is add values for the appropriate fields and save them. If you have data already loaded into your database, then you can load objects from it:

```
>>> from owmeta.neuron import Neuron
>>> n = Neuron.query()
>>> n.receptor('UNC-13')
owmeta_core.statement.Statement(...obj=owmeta_core.dataobject_property.
↪ContextualizedPropertyValue(rdflib.term.Literal(u'UNC-13')), context=None)
>>> for x in n.load():
...     do_something_with_unc13_neuron(n)  # doctest.SKIP
```

If you need additional entities it's easy to create them. Documentation for this is provided here.

Typically, you'll want to attach the data that you insert to entities already in the database. This allows you to recover objects in a hierarchical fashion from the database later. *Worm*, for instance, has a property, `neuron_network`, which points to the *Network* which should contain all neural cells and synaptic connections. To initialize the hierarchy you would do something like:

```
>>> from owmeta_core.context import Context
>>> from owmeta.worm import Worm
>>> from owmeta.network import Network
```

(continues on next page)

[1] http://answers.semanticweb.com/questions/19183/advantages-of-rdf-over-relational-databases
[2] http://research.microsoft.com/pubs/118211/andy%20maule%20-%20thesis.pdf

(continued from previous page)

```
>>> ctx = Context('http://example.org/c-briggsae')
>>> w = ctx(Worm)('C. briggsae') # The name is optional and currently defaults to 'C.␣
↪elegans'
>>> nn = ctx(Network)()          # make a neuron network
>>> w.neuron_network(nn)         # attach to the worm the neuron network
owmeta_core.statement.Statement(...)
>>> n = ctx(Neuron)('NeuronX')   # make a neuron
>>> n.receptor('UNC-13')         # state that the neuron has a UNC-13 type receptor
owmeta_core.statement.Statement(...)
>>> nn.neuron(n)                 # attach to the neuron network
owmeta_core.statement.Statement(...)
>>> ctx.save()           # save all of the data attached to the worm
```

It is possible to create objects without attaching them to anything and they can still be referenced by calling load on an instance of the object's class as in `n.load()` above. This also points out another fact: you don't have to set up the hierarchy for each insert in order for the objects to be linked to existing entities. If you have previously set up connections to an entity (e.g., `Worm('C. briggsae')`), assuming you *only* have one such entity, you can refer to things attached to it without respecifying the hierarchy for each script. The database packaged with owmeta should have only one Worm and one Network.

Remember that once you've made all of the statements, you must save the context in which the statements are made.

Future capabilities:

- Adding propositional logic to support making statements about all entities matching some conditions without needing to `load()` and `save()` them from the database.
- Statements like:

  ```
  ctx = Context('http://example.org/c-briggsae')
  w = ctx.stored(Worm)()
  w.neuron_network.neuron.receptor('UNC-13')
  l = list(w.load()) # Get a list of worms with neurons expressing 'UNC-13'
  ```

  currently, to do the equivalent, you must work backwards, finding all neurons with UNC-13 receptors, then getting all networks with those neurons, then getting all worms with those networks:

  ```
  worms = set()
  n = ctx.stored(Neuron)()
  n.receptor('UNC-13')
  for ns in n.load():
      nn = ctx.stored(Network)()
      nn.neuron(ns)
      for z in nn.load():
          w = ctx.stored(Worm)()
          w.neuron_network(z)
          worms.add(w)
  l = list(worms)
  ```

  It's not difficult logic, but it's 8 extra lines of code for a, conceptually, very simple query.

- Also, queries like:

```
l = list(ctx.stored(Worm)('C. briggsae').neuron_network.neuron.receptor()) # get a␣
↪list
#of all receptors expressed in neurons of C. briggsae
```

Again, not difficult to write out, but in this case it actually gives a much longer query time because additional values are queried in a `load()` call that are never returned.

We'd also like operators for composing many such strings so:

```
ctx.stored(Worm)('C. briggsae').neuron_network.neuron.get('receptor', 'innexin') #␣
↪list
#of (receptor, innexin) values for each neuron
```

would be possible with one query and thus not requiring parsing and iterating over neurons twice–it's all done in a single, simple query.

2.3.1 Contexts

Above, we used contexts without explaining them. In natural languages, our statements are made in a context that influences how they should be interpreted. In owmeta, that kind of context-sensitivity is modeled by using `owmeta.context.Context` objects. To see what this looks like, let's start with an example.

Basics

I have data about widgets from BigDataWarehouse (BDW) that I want to translate into RDF using owmeta, but I don't want put them with my other widget data since BDW data may conflict with mine. Also, if get more BDW data, I want to be able to relate these data to that. A good way to keep data which are made at distinct times or which come from different, possibly conflicting, sources is using contexts. The code below shows how to do that:

```
>>> from rdflib import ConjunctiveGraph
>>> from owmeta_core.context import Context
>>> # from mymod import Widget  # my own OWM widget model
>>> # from bdw import Load # BigDataWarehouse API

>>> # Create a Context with an identifier appropriate to this BDW data import
>>> ctx = Context('http://example.org/data/imports/BDW_Widgets_2017-2018')
>>> ctx.mapper.process_class(Widget)

>>> # Create a context manager using the default behavior of reading the
>>> # dictionary of current local variables
>>> with ctx(W=Widget) as c:
...     for record in Load(data_set='Widgets2017-2018'):
...         # declares Widgets in this context
...         c.W(part_number=record.pnum,
...             fullness=record.flns,
...             hardiness=record.hrds)
Widget(ident=rdflib.term.URIRef(...))


>>> # Create an RDFLib graph as the target for the data
>>> g = ConjunctiveGraph()
```

(continues on next page)

(continued from previous page)

```
>>> # Save the data
>>> ctx.save(g)

>>> # Serialize the data in the nquads format so we can see that all of our
>>> # statements are in the proper context
>>> print(g.serialize(format='nquads', encoding='UTF-8').decode('UTF-8'))
<http://example.org/BDW/entities/Widget#12> <http...> <http://example.org/data/imports/
↪BDW_Widgets_2017-2018> .
<http://example.org/BDW/entities/Widget#12> <...
```

If you've worked with lots of data before, this kind of pattern should be familiar. You can see how, with later imports, you would follow the naming scheme to create new contexts (e.g., `http://example.org/data/imports/BDW_Widgets_2018-2019`). These additional contexts could then have separate metadata attached to them or they could be compared:

```
>>> len(list(ctx(Widget)().load()))
1
>>> len(list(ctx18(Widget)().load()))  # 2018-2019 context
3
```

Context Metadata

Contexts, because they have identifiers just like any other objects, so we can make statements about them as well. An essential statement is imports: Contexts import other contexts, which means, if you follow owmeta semantics, that when you query objects from the importing context, that the imported contexts will also be available to query.

2.4 Software Versioning

The owmeta library follows the semanitc versioning scheme. For the sake of versioning, the software interface consists of:

1. Extensions to the **owm** command line defined
2. All "public" definitions (i.e., those whose names do not begin with '_') in the *owmeta* package, sub-packages, and sub-modules
3. The format of RDF data generated by subclasses of `owmeta_core.dataobject.DataObject` and defined in the *owmeta* package, sub-packages, and sub-modules
4. The API documentation for the *owmeta* package, sub-packages, and sub-modules

In addition, any changes to the packages released on PyPI mandates at least a patch version increment.

For Git, our software version control system, software releases will be represented as tags in the form `v$semantic_version` with all components of the semantic version represented.

2.4.1 Documentation versioning

The documentation will have a distinct version number from the software. The version numbers for the documentation must change at least as often as the software versioning since the relationship of the documentation to the software necessarily changes. However, changes _only_ to the non-API documentation will not be a cause for a change to any of the components of the software version number. For documentation releases which coincide with software releases, the documentation version number will simply be the software version number. Any subsequent change to documentation between software releases will compel an increase in the documentation version number by one. The documentation version number for such documentation releases will be represented as `${software_version}+docs${documentation_increment}`.

2.5 Python Release Compatibility

All Python releases will be supported until they reach their official end-of-life, typically reported as "Release Schedule" PEPs (search "release schedule" on the PEP index) Thereafter, any regressions due to dependencies of owmeta dropping support for an EOL Python version, or due to a change in owmeta making use of a feature in a still-supported Python release will only be fixed for the sake of OpenWorm projects when requested by an issue on our tracker or for other projects when a compelling case can be made.

This policy is intended to provide support to most well-maintained projects which depend on owmeta while not overburdening developers.

CHAPTER

THREE

FOR DEVELOPERS

3.1 Testing in owmeta

3.1.1 Preparing for tests

Within the owmeta project directory, owmeta can be installed for development and testing like this:

```
pip install --editable .
```

The project database should be populated like:

```
owm clone https://github.com/openworm/OpenWormData.git
```

3.1.2 Running tests

Tests should be run via setup.py like:

```
pytest
```

you can pass options to `pytest` like so:

```
pytest -k ChannelTest
```

3.1.3 Writing tests

Tests are written using Python's unittest. In general, a collection of closely related tests should be in one file. For selecting different classes of tests, tests can also be tagged using pytest marks like:

```
@pytest.mark.tag
class TestClass(unittest.TestCase):
    ...
```

Currently, marks are used to distinguish between unit-level tests and others which have the `inttest` mark. All marks are listed in pytest.ini under 'markers'.

3.1.4 Data Bundle Tests

The tests in DataIntegrityTest.py require that the openworm/owmeta-data bundle is installed. Normally, these will run in the CI environment. If you are doing work that affects what goes in the bundle, you can install new versions of the bundle and run the tests with `pytest -m data_bundle`.

3.2 Adding documentation

Documentation for owmeta is housed in two locations:

1. In the top-level project directory as `INSTALL.md` and `README.md`.
2. As a Sphinx project under the `docs` directory

By way of example, to add a page about useful facts concerning C. elegans to the documentation, include an entry in the list under `toctree` in `docs/index.rst` like:

```
worm-facts
```

and create the file `worm-facts.rst` under the `docs` directory and add a line:

```
.. _worm-facts:
```

to the top of your file, remembering to leave an empty line before adding all of your wonderful worm facts.

You can get a preview of what your documentation will look like when it is published by running `sphinx-build` on the docs directory:

```
sphinx-build -w sphinx-errors docs build_destination
```

The docs will be compiled to html which you can view by pointing your web browser at `build_destination/index.html`. If you want to view the documentation locally with the ReadTheDocs theme you'll need to download and install it.

3.2.1 API Documentation

API documentation is generated by the Sphinx autodoc extension. The format should be easy to pick up on, but a reference is available here. Just add a docstring to your function/class/method and add an `automodule` line to `owmeta/__init__.py` and your class should appear among the other documented classes.

3.2.2 Substitutions

Project-wide substitutions can be (conservatively!) added to allow for easily changing a value over all of the documentation. Currently defined substitutions can be found in `conf.py` in the `rst_epilog` setting. More about substitutions

3.2.3 Conventions

If you'd like to add a convention, list it here and start using it. It can be reviewed as part of a pull request.

1. Narrative text should be wrapped at 80 characters.
2. Long links should be extracted from narrative text. Use your judgement on what 'long' is, but if it causes the line width to stray beyond 80 characters that's a good indication.

3.3 owmeta coding standards

Pull requests are *required* to follow the PEP-8 Guidelines for contributions of Python code to owmeta, with some exceptions noted below. Compliance can be checked with the `pep8` tool and these command line arguments:

```
--max-line-length=120 --ignore=E261,E266,E265,E402,E121,E123,E126,E226,E24,E704,E128
```

Refer to the pep8 documentation for the meanings of these error codes.

Lines of code should only be wrapped before 120 chars for readability. Comments and string literals, including docstrings, can be wrapped to a shorter length.

Some violations can be corrected with `autopep8`.

CHAPTER

FOUR

ISSUES

CHAPTER

FIVE

INDICES AND TABLES

- genindex
- modindex
- search

PYTHON MODULE INDEX

O

INDEX

D

E

G

H

I

L

M

N

O

P

Q

R

S

T

U

V

W

Y