Manwë Documentation

Release 1.3.1

Martijn Vermaat <martijn@vermaat.name>

December 22, 2015

Contents

Warning: This is a work in progress, probably not yet ready for use!

Manwë is a Python client library for working with the Varda database for genomic variation frequencies. It also provides a command line interface to some of its functionality.

The main goal of Manwë is to offer the complete Varda API, but on an abstraction level that is nice to work with from Python code.

```
>>> import manwe
>>> session = manwe.Session()
>>> user = session.create_user('testlogin', 'password')
>>> user.dirty
False
>>> user.name = 'Test User'
>>> user.dirty
True
>>> user.save()
>>> user.dirty
False
>>> for sample in session.samples():
...     print sample.user.name
...
Rob Userman
Barry Robsfriend
Rob Userman
```

CHAPTER 1

User documentation

New users should probably start here.

1.1 Installation

The Manwë source code is hosted on GitHub. Supported Python versions for running Manwë are 2.7 and PyPy (unit tests are run automatically on these platforms using the Travis CI service). Manwë can be installed either via the Python Package Index (PyPI) or from the source code.

1.1.1 Latest release via PyPI

To install the latest release via PyPI using pip:

```
pip install manwe
```

1.1.2 Development version

You can also clone and use the latest development version directly from the GitHub repository:

```
git clone https://github.com/varda/manwe.git
cd manwe
python setup.py install
```

1.2 User guide

Most functionality in Manwë is accessed though a *manwe.Session* instance.

Todo: This guide.

1.3 Command line interface

Some of the functionality in Manwë is provided through a simple command line interface.

Since the average scientist is too lazy to write complete documentation, you'll just find a quick dump of the command line help output below.

```
martijn@hue:~$ manwe -h
usage: manwe [-h] [--config CONFIG_FILE]

                  {import-sample,activate,sample,add-user,user,data-source,download-data-source}
                  ...

Manwë command line interface.

optional arguments:
  -h, --help            show this help message and exit
  --config CONFIG_FILE  path to configuration file to use instead of looking
                        in default locations

subcommands:
  {import-sample,activate,sample,add-user,user,data-source,download-data-source}
                        subcommand help
    import-sample       import sample data
    activate            activate sample
    sample              show sample details
    add-user            add new API user
    user                show user details
    data-source         show data source details
    download-data-source
                        download data source and write data to standard output
```

CHAPTER 2

API reference

Documentation on a specific function, class or method can be found in the API reference.

2.1 API reference

2.1.1 manwe

Manwë, a Python client library and command line interface to the Varda database for genomic variation frequencies.

class manwe.**Session**(*api_root=None*, *token=None*, *config=None*, *log_level=20*)

Bases: *manwe.session.AbstractSession*

Session for interfacing the server API.

Example session:

```
>>> session = Session()
>>> sample = session.create_sample('Test')
>>> sample.uri
'/samples/1'
>>> sample.dirty
False
>>> sample.name = 'Test sample'
>>> sample.dirty
True
>>> sample.save()
>>> sample.dirty
False
```

annotation(*uri*)

Get a resource of type annotation.

Parameters **uri** (*str*) – URI for the annotation to retrieve.

Returns A resource of type annotation.

Return type *Annotation*

annotations(**args*, ***kwargs*)

Query an annotation resource collection.

Returns An annotation resource collection.

Return type *AnnotationCollection*

coverage (*uri*)
Get a resource of type coverage.

Parameters **uri** (*str*) – URI for the coverage to retrieve.

Returns A resource of type coverage.

Return type *Coverage*

coverages (**args*, ***kwargs*)
Query a coverage resource collection.

Parameters **sample** (*Sample*) – Filter collection by sample.

Returns A coverage resource collection.

Return type *CoverageCollection*

create_annotation (**args*, ***kwargs*)
Create an annotation resource.

Parameters

- **data_source** (*DataSource*) – Data source to annotate.
- **name** (*str*) – Human readable annotation name.
- **queries** (*dict(str, str)*) – Sample queries to calculate variant frequencies over. Keys are query identifiers (alphanumeric) and values are query expressions.

Returns An annotation resource.

Return type *Annotation*

create_coverage (**args*, ***kwargs*)
Create a coverage resource.

Parameters

- **sample** (*Sample*) – Sample the coverage resource is part of.
- **data_source** (*DataSource*) – Data source for the coverage resource.

Returns A coverage resource.

Return type *Coverage*

create_data_source (**args*, ***kwargs*)
Create a data source resource.

Parameters

- **name** (*str*) – Human readable data source name.
- **filetype** (*str*) – Data filetype. Possible values are `bed`, `vcf`, and `csv`.
- **gzipped** (*bool*) – Whether or not the data is compressed using gzip.
- **data** (*file-like object*) – Data blob.
- **local_file** (*str*) – A filename on the server filesystem. This can be used instead of *data*.

Returns A data source resource.

Return type *DataSource*

create_group (**args*, ***kwargs*)
Create a group resource.

Parameters **name** (*str*) – Human readable group name.

Returns A group resource.

Return type *Group*

create_sample(**args*, ***kwargs*)
Create a sample resource.

Parameters

- **name** (*str*) – Human readable sample name.
- **pool_size** (*int*) – Number of individuals in the sample.
- **coverage_profile** (*bool*) – Whether or not the sample has a coverage profile.
- **public** (*bool*) – Whether or not this sample is public.
- **notes** (*str*) – Human readable notes in Markdown format.
- **groups** (iterable(*DataSource*)) – Groups this sample is part of.

Returns A sample resource.

Return type *Sample*

create_user(**args*, ***kwargs*)
Create a user resource.

Parameters

- **login** (*str*) – Login name used for authentication.
- **password** (*str*) – Password used for authentication.
- **name** (*str*) – Human readable user name.
- **email** (*str*) – User e-mail address.
- **roles** (*iterable(str)*) – Roles for this user (values must be from `USER_ROLES`).

Returns A user resource.

Return type *User*

create_variant(**args*, ***kwargs*)
Create a variant resource.

Parameters

- **chromosome** (*str*) – Chromosome name.
- **position** (*int*) – Position of variant on *chromosome*.
- **reference** (*str*) – Reference allele.
- **observed** (*str*) – Observed allele.

Returns A variant resource.

Return type *Variant*

create_variation(**args*, ***kwargs*)
Create a variation resource.

Parameters

- **sample** (*Sample*) – Sample the variation resource is part of.

- **data_source** (*DataSource*) – Data source for the variation resource.
- **skip_filtered** (*bool*) – Discard entries in *data_source* marked as filtered.
- **use_genotypes** (*bool*) – Use per-sample genotype information from *data_source*.
- **prefer_genotype_likelihoods** (*bool*) – Prefer using genotype likelihoods from *data_source* instead of concrete genotypes.

Returns A variation resource.

Return type *Variation*

data_source(*uri*)
Get a resource of type data_source.

Parameters **uri** (*str*) – URI for the data_source to retrieve.

Returns A resource of type data_source.

Return type *DataSource*

data_sources(**args*, ***kwargs*)
Query a data source resource collection.

Parameters **user** (*User*) – Filter collection by user.

Returns A data source resource collection.

Return type *DataSourceCollection*

group(*uri*)
Get a resource of type group.

Parameters **uri** (*str*) – URI for the group to retrieve.

Returns A resource of type group.

Return type *Group*

groups(**args*, ***kwargs*)
Query a group resource collection.

Returns A group resource collection.

Return type *GroupCollection*

sample(*uri*)
Get a resource of type sample.

Parameters **uri** (*str*) – URI for the sample to retrieve.

Returns A resource of type sample.

Return type *Sample*

samples(**args*, ***kwargs*)
Query a sample resource collection.

Parameters

- **groups** (iterable(*DataSource*)) – Filter collection by groups.
- **public** (*bool*) – Filter collection by public/non-public.
- **user** (*User*) – Filter collection by user.

Returns A sample resource collection.

Return type *SampleCollection*

user(*uri*)
Get a resource of type user.

Parameters **uri** (*str*) – URI for the user to retrieve.

Returns A resource of type user.

Return type *User*

users(**args*, ***kwargs*)
Query a user resource collection.

Returns A user resource collection.

Return type *UserCollection*

variant(*uri*)
Get a resource of type variant.

Parameters **uri** (*str*) – URI for the variant to retrieve.

Returns A resource of type variant.

Return type *Variant*

variants(**args*, ***kwargs*)
Query a variant resource collection.

Returns A variant resource collection.

Return type *VariantCollection*

variation(*uri*)
Get a resource of type variation.

Parameters **uri** (*str*) – URI for the variation to retrieve.

Returns A resource of type variation.

Return type *Variation*

variations(**args*, ***kwargs*)
Query a variation resource collection.

Parameters **sample** (*Sample*) – Filter collection by sample.

Returns A variation resource collection.

Return type *VariationCollection*

2.1.2 manwe.config

Manwë configuration object.

class manwe.config.**AttributeDictMixin**
Bases: object

Augment classes with a Mapping interface by adding attribute access.

Taken from Celery (*celery.datastructures.AttributeDictMixin*).

class manwe.config.**Config**
Bases: flask.config.Config, *manwe.config.AttributeDictMixin*

Dictionary with some extra ways to fill it from files or special dictionaries (see *flask.config.Config*) and attribute access.

Initialized with *manwe.default_config*.

2.1.3 manwe.default_config

Manwë default configuration settings.

manwe.default_config.**API_ROOT = 'http://127.0.0.1:5000'**
Varda API root endpoint.

manwe.default_config.**COLLECTION_CACHE_SIZE = 20**
Number of resources to query per collection request.

manwe.default_config.**DATA_BUFFER_SIZE = 1048576**
Size of chunks to yield from data iterator in bytes.

manwe.default_config.**TASK_POLL_WAIT = 2**
Time to wait between polling task state (in seconds).

manwe.default_config.**TOKEN = None**
Varda API authentication token.

manwe.default_config.**VERIFY_CERTIFICATE = True**
Whether or not to verify the API SSL certificate, or a path to a CA_BUNDLE file with certificates of trusted CAs.

2.1.4 manwe.errors

API custom exceptions.

exception manwe.errors.**ApiError**(*code*, *message*)
Bases: exceptions.Exception

exception manwe.errors.**BadRequestError**(*code*, *message*)
Bases: *manwe.errors.ApiError*

exception manwe.errors.**ForbiddenError**(*code*, *message*)
Bases: *manwe.errors.ApiError*

exception manwe.errors.**NotAcceptableError**(*code*, *message*)
Bases: *manwe.errors.ApiError*

exception manwe.errors.**NotFoundError**(*code*, *message*)
Bases: *manwe.errors.ApiError*

exception manwe.errors.**TaskError**(*code*, *message*)
Bases: *manwe.errors.ApiError*

exception manwe.errors.**UnauthorizedError**(*code*, *message*)
Bases: *manwe.errors.ApiError*

exception manwe.errors.**UnsatisfiableRangeError**(*code*, *message*)
Bases: *manwe.errors.ApiError*

2.1.5 manwe.fields

Manwë resource fields.

class manwe.fields.**Custom**(*from_api*, *to_api*, ***kwargs*)
Bases: *manwe.fields.Field*

Custom field definitions are parameterized with conversion functions.

class manwe.fields.**Field**(*key=None*, *mutable=False*, *hidden=False*, *default=None*, *doc=None*)
Bases: object

Base class for resource field definitions.

A field definition can convert field values from their API representation to their Python representation, and vice versa.

__init__(*key=None*, *mutable=False*, *hidden=False*, *default=None*, *doc=None*)
Create a field instance.

Parameters

- **key** (*str*) – Key by which this field is stored in the API.
- **mutable** (*bool*) – If *True*, field values can be modified.
- **hidden** (*bool*) – If *True*, field should not be shown.
- **default** – Default field value (as a Python value).
- **doc** (*str*) – Documentation string

default = None
Default field value (as an API value).

doc = None
Documentation string.

from_python(*value*)
Convert Python value to API value.

hidden = None
If *True*, field should not be shown.

key = None
Key by which this field is stored in the API. By default inherited from *name*.

mutable = None
If *True*, field values can be modified.

name
Name by which this field is available on the resource class.

to_python(*value*, *resource*)
Convert API value to Python value.

This gets called from field getters, so the user gets a nice Python value when accessing the field.

Subclasses for structured data (such as lists and dicts) should be careful to not return mutable structures here, since that would allow to bypass the field setter. For example, calling *field.append(v)* will not add *field* to the set of dirty fields and will not go through *from_python()*. Actually, it might not even modify the API value on the resource, because *to_python()* probably created a copy.

One solution for this, as implemented on Set, is to return an immutable field value (a *frozenset* in this case) and thereby force modifications through the field setter.

Another approach would be something similar to the *MutableDict* type in SQLAlchemy (see Mutation Tracking).

This does not apply to *Link* fields, where the value is itself a resource which should be modified using its own `resources.Resource.save()` method.

class `manwe.fields.`**Link**(*resource_key*, ***kwargs*)
Bases: *manwe.fields.Field*

Definition for a resource link.

__init__(*resource_key*, ***kwargs*)

Parameters **resource_key** (*str*) – Key for the linked resource.

from_python(*value*)
In request data, a resource link is represented by its URI (a string).

to_python(*value*, *resource*)
Create a `resources.Resource` instance from the resource URI.

Modifications of the returned resource should be saved by calling `resources.Resource.save()` on that resource.

class `manwe.fields.`**Queries**(*key=None*, *mutable=False*, *hidden=False*, *default=None*, *doc=None*)
Bases: *manwe.fields.Field*

Definition for a field containing annotation queries.

In the API, annotation queries are lists of dictionaries with *name* and *expression* items.

As a Python value, we represent this as a dictionary with keys the query names and values the query expressions.

2.1.6 manwe.resources

Manwë resources.

class `manwe.resources.`**Annotation**(*session*, *values*)
Bases: *manwe.resources.TaskedResource*

Class for representing an annotation resource.

annotated_data_source
Annotated data source (*DataSource* instance).

classmethod **create**(*session*, *data_source*, *name=None*, *queries=None*)
Create an annotation resource.

Parameters

- **data_source** (*DataSource*) – Data source to annotate.
- **name** (*str*) – Human readable annotation name.
- **queries** (*dict(str, str)*) – Sample queries to calculate variant frequencies over. Keys are query identifiers (alphanumeric) and values are query expressions.

Returns An annotation resource.

Return type *Annotation*

original_data_source
Original data source (*DataSource* instance).

class manwe.resources.**AnnotationCollection**(*session*)
Bases: *manwe.resources.ResourceCollection*

Class for representing an annotation resource collection as an iterator returning *Annotation* instances.

__init__(*session*)
Query an annotation resource collection.

Returns An annotation resource collection.

Return type *AnnotationCollection*

resource_class
alias of *Annotation*

class manwe.resources.**Coverage**(*session*, *values*)
Bases: *manwe.resources.TaskedResource*

Class for representing a coverage resource.

classmethod create(*session*, *sample*, *data_source*)
Create a coverage resource.

Parameters

- **sample** (*Sample*) – Sample the coverage resource is part of.
- **data_source** (*DataSource*) – Data source for the coverage resource.

Returns A coverage resource.

Return type *Coverage*

data_source
Coverage data (*DataSource* instance).

sample
Coverage is part of this *Sample*.

class manwe.resources.**CoverageCollection**(*session*, *sample=None*)
Bases: *manwe.resources.ResourceCollection*

Class for representing a coverage resource collection as an iterator returning *Coverage* instances.

__init__(*session*, *sample=None*)
Query a coverage resource collection.

Parameters **sample** (*Sample*) – Filter collection by sample.

Returns A coverage resource collection.

Return type *CoverageCollection*

resource_class
alias of *Coverage*

sample
Collection is filtered by this *Sample*.

class manwe.resources.**DataSource**(*session*, *values*)
Bases: *manwe.resources.Resource*

Class for representing a data source resource.

added
Date and time this data source was added.

classmethod **create**(*session*, *name*, *filetype*, *gzipped=False*, *data=None*, *local_file=None*)
Create a data source resource.

Parameters

- **name** (*str*) – Human readable data source name.
- **filetype** (*str*) – Data filetype. Possible values are `bed`, `vcf`, and `csv`.
- **gzipped** (*bool*) – Whether or not the data is compressed using gzip.
- **data** (*file-like object*) – Data blob.
- **local_file** (*str*) – A filename on the server filesystem. This can be used instead of *data*.

Returns A data source resource.

Return type *DataSource*

data
Iterator yielding data as chunks.

filetype
Data filetype.

gzipped
If *True*, *data* is compressed using gzip.

name
Human readable data source name.

user
Data source is owned by this *User*.

class manwe.resources.**DataSourceCollection**(*session*, *user=None*)
Bases: *manwe.resources.ResourceCollection*

Class for representing a data source resource collection as an iterator returning *DataSource* instances.

__init__(*session*, *user=None*)
Query a data source resource collection.

Parameters **user** (*User*) – Filter collection by user.

Returns A data source resource collection.

Return type *DataSourceCollection*

resource_class
alias of *DataSource*

user
Collection is filtered by this *User*.

class manwe.resources.**Group**(*session*, *values*)
Bases: *manwe.resources.Resource*

Class for representing a group resource.

classmethod **create**(*session*, *name*)
Create a group resource.

Parameters **name** (*str*) – Human readable group name.

Returns A group resource.

Return type *Group*

name
Human readable group name.

class manwe.resources.**GroupCollection**(*session*)
Bases: *manwe.resources.ResourceCollection*

Class for representing a group resource collection as an iterator returning *Group* instances.

__init__(*session*)
Query a group resource collection.

Returns A group resource collection.

Return type *GroupCollection*

resource_class
alias of *Group*

class manwe.resources.**Resource**(*session*, *values*)
Bases: object

Base class for representing server resources.

Resource fields are defined as class attributes by Field instances.

__init__(*session*, *values*)
Create a representation for a server resource from a dictionary.

Parameters

- **session** (*Session*) – Manwë session.
- **values** (*dict*) – Dictionary with field values (using API keys and values).

classmethod create(*session*, *values=None*, *files=None*)
Create a new resource on the server and return a representation for it.

Parameters

- **session** (*Session*) – Manwë session.
- **values** (*dict*) – Dictionary with field values (using Python names and values).
- **files** (*dict(str, file-like object)*) – Open file objects.

Every subclass should override this with an informative docstring.

dirty
True if there are any unsaved changes on this resource, *False* otherwise.

key = None
Key for this resource type.

refresh(*skip_dirty=False*)
Refresh resource with data from the server.

Parameters **skip_dirty** (*bool*) – If *True*, don't refresh field values with unsaved changes.

save()
Send any unsaved changes on this resource to the server and refresh with data from the server.

save_fields(***values*)
Send field values specified by keyword arguments to the server and refresh with data from the server (skipping dirty field values).

Keyword arguments use Python names and values.

session = None
The session this resource is attached to as `.Session`.

uri
Resource URI.

class manwe.resources.**ResourceCollection**(*session*, *values=None*)
Bases: object

Base class for representing server resource collections, iterators returning *Resource* instances.

Collection filters are defined as class attributes by `Field` instances (and must not be mutable).

__init__(*session*, *values=None*)
Create a representation for a server resource collection.

Parameters

- **session** (*Session*) – Manwë session.
- **values** (*dict*) – Dictionary with field values (using Python names and values).

Every subclass should override this with an informative docstring.

cache_size
Number of resources to query per collection request.

next()
Return the next resource in the collection.

reset()
Reset resource collection iterator.

resource_class = None
Resource class to use for instantiating resources in this collection.

session = None
The session this resource collection is attached to as `.Session`.

size = None
The total number of resources in this collection as last reported by the server. Note that the actual number of resources produced by the collection iterator might deviate from this number, and this is why there is not *__len__* property defined.

class manwe.resources.**Sample**(*session*, *values*)
Bases: *manwe.resources.Resource*

Class for representing a sample resource.

active
If *True*, the sample is active.

added
Date and time this sample was added.

coverage_profile
If *True*, the sample has a coverage profile.

classmethod create(*session*, *name*, *pool_size=1*, *coverage_profile=True*, *public=False*, *notes=None*, *groups=None*)
Create a sample resource.

Parameters

- **name** (*str*) – Human readable sample name.
- **pool_size** (*int*) – Number of individuals in the sample.
- **coverage_profile** (*bool*) – Whether or not the sample has a coverage profile.
- **public** (*bool*) – Whether or not this sample is public.
- **notes** (*str*) – Human readable notes in Markdown format.
- **groups** (iterable(*DataSource*)) – Groups this sample is part of.

Returns A sample resource.

Return type *Sample*

groups
Sample is part of these groups (*Group* instances).

name
Human readable sample name.

notes
Human readable notes in Markdown format.

pool_size
Number of individuals.

public
If *True*, the sample is public.

user
Sample is owned by this *User*.

class manwe.resources.**SampleCollection**(*session*, *groups=None*, *public=None*, *user=None*)
Bases: *manwe.resources.ResourceCollection*

Class for representing a sample resource collection as an iterator returning *Sample* instances.

__init__(*session*, *groups=None*, *public=None*, *user=None*)
Query a sample resource collection.

Parameters

- **groups** (iterable(*DataSource*)) – Filter collection by groups.
- **public** (*bool*) – Filter collection by public/non-public.
- **user** (*User*) – Filter collection by user.

Returns A sample resource collection.

Return type *SampleCollection*

groups
Collection is filtered by these groups (*Group* instances).

public
Collection is filtered by this public state.

resource_class
alias of *Sample*

user
Collection is filtered by this *User*.

class manwe.resources.**Task**(*resource*)
Bases: object

Represents a server task.

Instances are equiped with the parent resource which they use to query task state. This means refreshing the parent resource is directly visible on the task.

error
The error object as a *TaskError* if *state* is failure, *None* otherwise.

failure
Whether or not task is in failure state.

If *True*, *error* is set.

progress
Task progress in the range *0* to *100* if *state* is running, *None* otherwise.

resubmit()
Resubmit task.

running
Whether or not task is in running state.

If *True*, *progress* is set.

state
Task state. Possible values are waiting, running, succes, and failure.

success
Whether or not task is in success state.

wait()
Block while *state* is waiting or running, polling the server every *TASK_POLL_WAIT* seconds.

wait_and_monitor()
Iterator, yielding *progress* while *state* is waiting or running, polling the server every *TASK_POLL_WAIT* seconds.

After that, yield *100*, or raise *error* if *state* is failure.

waiting
Whether or not task is in waiting state.

class manwe.resources.**TaskedResource**(*session*, *values*)
Bases: *manwe.resources.Resource*

Base class for representing server resources with tasks.

task
Server task (*Task* instance).

class manwe.resources.**User**(*session*, *values*)
Bases: *manwe.resources.Resource*

Class for representing a user resource.

added
Date and time this user was added.

classmethod create(*session*, *login*, *password*, *name=None*, *email=None*, *roles=None*)
Create a user resource.

Parameters

- **login** (*str*) – Login name used for authentication.
- **password** (*str*) – Password used for authentication.
- **name** (*str*) – Human readable user name.
- **email** (*str*) – User e-mail address.
- **roles** (*iterable(str)*) – Roles for this user (values must be from `USER_ROLES`).

Returns A user resource.

Return type *User*

email
Email address.

login
Login name.

name
Human readable user name.

password
Password used for authentication.

roles
Roles for this user.

class `manwe.resources.`**UserCollection**(*session*)
Bases: *manwe.resources.ResourceCollection*

Class for representing a user resource collection as an iterator returning *User* instances.

__init__(*session*)
Query a user resource collection.

Returns A user resource collection.

Return type *UserCollection*

resource_class
alias of *User*

class `manwe.resources.`**Variant**(*session*, *values*)
Bases: *manwe.resources.Resource*

Class for representing a variant resource.

annotate(*queries=None*)
Annotate this variant with the observed frequencies over sets of samples

Parameters **queries** (*dict(str, str)*) – Sample queries to calculate variant frequencies over. Keys are query identifiers (alphanumeric) and values are query expressions.

Returns Variant observation frequencies. Keys are query identifiers and values are dictionaries with *coverage*, *frequency*, *frequency_het*, and *frequency_hom*.

Return type dict(str, dict)

chromosome
Chromosome name.

classmethod create(*session*, *chromosome*, *position*, *reference=''*, *observed=''*)
Create a variant resource.

Parameters

- **chromosome** (*str*) – Chromosome name.
- **position** (*int*) – Position of variant on *chromosome*.
- **reference** (*str*) – Reference allele.
- **observed** (*str*) – Observed allele.

Returns A variant resource.

Return type *Variant*

observed
Observed allele.

position
Position of variant on *chromosome*.

reference
Reference allele.

class manwe.resources.**VariantCollection**(*session*)
Bases: *manwe.resources.ResourceCollection*

Class for representing a variant resource collection as an iterator returning *Variant* instances.

__init__(*session*)
Query a variant resource collection.

Returns A variant resource collection.

Return type *VariantCollection*

resource_class
alias of *Variant*

class manwe.resources.**Variation**(*session*, *values*)
Bases: *manwe.resources.TaskedResource*

Class for representing a variation resource.

classmethod create(*session*, *sample*, *data_source*, *skip_filtered=True*, *use_genotypes=True*, *prefer_genotype_likelihoods=False*)
Create a variation resource.

Parameters

- **sample** (*Sample*) – Sample the variation resource is part of.
- **data_source** (*DataSource*) – Data source for the variation resource.
- **skip_filtered** (*bool*) – Discard entries in *data_source* marked as filtered.
- **use_genotypes** (*bool*) – Use per-sample genotype information from *data_source*.
- **prefer_genotype_likelihoods** (*bool*) – Prefer using genotype likelihoods from *data_source* instead of concrete genotypes.

Returns A variation resource.

Return type *Variation*

data_source
Variation data (*DataSource* instance).

sample
Variation is part of this *Sample*.

class manwe.resources.**VariationCollection**(*session*, *sample=None*)
Bases: *manwe.resources.ResourceCollection*

Class for representing a variation resource collection as an iterator returning *Variation* instances.

__init__(*session*, *sample=None*)
Query a variation resource collection.

Parameters **sample** (*Sample*) – Filter collection by sample.

Returns A variation resource collection.

Return type *VariationCollection*

resource_class
alias of *Variation*

sample
Collection is filtered by this *Sample*.

class manwe.resources.**classproperty**(*getter*)
Bases: object

Decorator for defining computed class attributes, dual to the *property* built-in complementary to the *classmethod* built-in.

Example usage:

```
>>> class Foo(object):
...     x= 4
...     @classproperty
...     def number(cls):
...         return cls.x
...
>>> Foo().number
4
>>> Foo.number
4
```

Copied from bobince.

2.1.7 manwe.session

Manwë sessions.

class manwe.session.**AbstractSession**(*api_root=None*, *token=None*, *config=None*, *log_level=20*)
Bases: object

Abstract session for interfacing the server API.

Subclasses should have a dictionary of resources in their *_collections* attribute.

__init__(*api_root=None*, *token=None*, *config=None*, *log_level=20*)
Create a session.

Parameters

- **api_root** (*str*) – Varda API root endpoint.
- **token** (*str*) – Varda API authentication token.
- **config** (config.Config) – Manwë configuration object (*api_root* and *token* take precedence).

- **log_level** (*logging.LOG_LEVEL*) – Control the level of log messages you will see. Use *log_level=logging.DEBUG* to troubleshoot.

get(**args*, ***kwargs*)
Short for *request()* where *method* is GET.

patch(**args*, ***kwargs*)
Short for *request()* where *method* is PATCH.

post(**args*, ***kwargs*)
Short for *request()* where *method* is POST.

request(*method*, *uri*, ***kwargs*)
Send HTTP request to server.

Raises requests.RequestException Exception occurred while handling an API request.

set_log_level(*log_level*)
Control the level of log messages you will see.

manwe.session.**stringify**(*value*)
Serialize *value* to a *str* parsable by Varda.

Only works one level deep, so no nesting of structured data.

```
>>> stringify(34)
'34'
>>> stringify(False)
'false'
>>> stringify([4,2,6])
'4,2,6'
>>> stringify({'a': False, 'b': True})
'a:false,b:true'
```

CHAPTER 3

Additional notes

3.1 Development

Development of Manwë happens on GitHub: https://github.com/varda/manwe

3.1.1 Contributing

Contributions to Manwë are very welcome! They can be feature requests, bug reports, bug fixes, unit tests, documentation updates, or anything els you may come up with.

3.1.2 Coding style

In general, try to follow the PEP 8 guidelines for Python code and PEP 257 for docstrings.

3.1.3 Unit tests

To run the unit tests with pytest, just run:

```
$ py.test
```

The tests use Varda, so you need to have that installed (just the Python package, no configuration needed) along with its dependencies.

3.1.4 Versioning

A normal version number takes the form X.Y.Z where X is the major version, Y is the minor version, and Z is the patch version. Development versions take the form X.Y.Z.dev where X.Y.Z is the closest future release version.

Note that this scheme is not 100% compatible with SemVer which would require X.Y.Z-dev instead of X.Y.Z.dev but compatibility with setuptools is more important for us. Other than that, version semantics are as described by SemVer.

Releases are published at PyPI and available from the GitHub git repository as tags.

Release procedure

Releasing a new version is done as follows:

1. Make sure the section in the `CHANGES` file for this release is complete and there are no uncommitted changes.

   Note: Commits since release X.Y.Z can be listed with `git log vX.Y.Z..` for quick inspection.

2. Update the `CHANGES` file to state the current date for this release and edit `manwe/__init__.py` by updating *__date__* and removing the `dev` value from *__version_info__*.

   Commit and tag the version update:

```
git commit -am 'Bump version to X.Y.Z'
git tag -a 'vX.Y.Z'
git push --tags
```

3. Upload the package to PyPI:

```
python setup.py sdist upload
```

4. Add a new entry at the top of the `CHANGES` file like this:

```
Version X.Y.Z+1
---------------

Release date to be decided.
```

   Increment the patch version and add a `dev` value to *__version_info__* in `manwe/__init__.py` and commit these changes:

```
git commit -am 'Open development for X.Y.Z+1'
```

3.2 Changelog

Here you can see the full list of changes between each Manwë release.

3.2.1 Version 1.3.2

Release date to be decided.

3.2.2 Version 1.3.1

Released on October 7th 2015.

- Fix bug in command line interface *samples show*.
- Wait for import task to complete in command line interface.

3.2.3 Version 1.3.0

Released on October 6th 2015.

- Bump API compatibility to *>=3.0.0,<4.0.0*.

- Wait for annotation task to complete in command line interface.
- Resource tasks (get state, wait for completion, resubmit).
- Ability to save ad-hoc specified field values (*Resource.save_fields()*).
- Refresh fields on save.
- Optionally skip dirty fields in refresh.

3.2.4 Version 1.2.1

Released on October 2nd 2015.

- Add VERIFY_CERTIFICATE config setting.
- Allow API mounted on a subpath.

3.2.5 Version 1.2.0

Released on September 23rd 2015.

- Configurable collection cache and buffer size.
- Add *Resource.refresh()* method .
- Add *ResourceCollection.reset()* method.
- More complete Sphinx API documentation.
- Better Python API discoverability (through tab completion on the session and resource instances, and better docstrings).
- Stream file uploads, effectively supporting large files without loading them in memory.

3.2.6 Version 1.0.0

Released on September 13th 2015.

- Bump API compatibility to >=*2.1.0*,<*3.0.0*.
- Many new functions in a more complete command line interface.
- Restructure command line interface.
- Support more collection filters.
- Improve handling fields of type set.
- Support for annotation queries.
- Support for sample groups.
- Implement equality check on resources.
- Better discoverability of API (tab completion of the session object).
- New and simplified configuration file format (it is a Python module).
- Refactored and more complete unit tests.

3.2.7 Version 0.2.0

Released on August 31st 2015.

- Bump API compatibility to *>=1.0.0,<2.0.0*.
- Moved git repository to GitHub Varda organisation.
- Do not send uninitialized collection arguments.
- Annotate regions.
- Annotate a VCF file from the commandline.
- Import VCF/BED for existing samples.
- Get task status.
- Use API token instead of login and password.
- Fix sample activation.
- Support unset optional fields in resource creation.

3.2.8 Version 0.1.0

Released on May 11th 2013.

First public release.

3.3 Copyright

Manwë is licensed under the MIT License, meaning you can do whatever you want with it as long as all copies include these license terms. The full license text can be found below.

3.3.1 Authors

Manwë is written and maintained by Martijn Vermaat.

- Leiden University Medical Center <humgen@lumc.nl>
- Martijn Vermaat <martijn@vermaat.name>

3.3.2 License

HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

CHAPTER 4

Indices and tables

- genindex
- modindex
- search

m

Symbols

A

B

C

D

S

T

U

V

W