Python

Aug 14, 2019

Contents

Contents:

CHAPTER 1

Tripal Library

Build Documentation

A Python library for interacting with Tripal.

It allows to load data into a remote Tripal instance, and explore its content, directly from Python code, or using a CLI (Command Line Interface).

See below for examples of what you can do using the `tripal` python module, and the `tripaille` CLI.

(in case you wonder: Tripaille's name comes from a bad french word play)

1.1 Requirements

The tripal_rest_api module needs to be installed on your Tripal instance (choose the master branch for Tripal 2, or the branch 7.x-3.x for Tripal 3).

1.2 Installation

```
$ pip install tripal

# On first use you'll need to create a config file to connect to the tripal server,␣
↪just run:

$ tripaille init
Welcome to Tripal's Tripaille!
TRIPAL_URL: http://localhost/
TRIPAL_USER: admin
TRIPAL_PASS: changeme
```

This will create a tripaille config file in ~/.tripaille.yml

1.3 Examples

```
    from tripal import TripalInstance
    ti = TripalInstance(tripal_url='http://localhost:8500/', username='admin',␣
↪password='changeme')

    # Create human species
    org = ti.organism.add_organism(genus="Homo", species="sapiens", common="Human",␣
↪abbr="H.sapiens")

    # Then display the list of organisms
    orgs = ti.organism.get_organisms()

    for org in orgs:
        print('{} {}'.format(org['genus'], org['species']))

    # Create an analysis
    an = ti.analysis.add_analysis(name="My cool analysis",
                                  program="Something",
                                  programversion="1.0",
                                  algorithm="Google",
                                  sourcename="src",
                                  sourceversion="2.1beta",
                                  sourceuri="http://example.org/",
                                  date_executed="2018-02-03")

    # Then display the list of organisms
    ans = ti.analysis.get_analyses()

    # And load some data
    ti.analysis.load_fasta(fasta="./test-data/Citrus_sinensis-scaffold00001.fasta",␣
↪analysis_id=ans[0]['analysis_id'], organism_id=orgs[0]['organism_id'])
    ti.analysis.load_gff3(gff="./test-data/Citrus_sinensis-orange1.1g015632m.g.gff3",␣
↪analysis_id=ans[0]['analysis_id'], organism_id=orgs[0]['organism_id'])
```

Or with the Tripaille client:

```
    $ tripaille organism add_organism --abbr H.sapiens --common Human Homo sapiens

    $ tripaille organism get_organisms
    [
        {
            "organism_id": "17",
            "abbreviation": "H.sapiens",
            "genus": "Homo",
            "species": "sapiens",
            "common_name": "Human",
            "infraspecific_name": null,
            "type_id": null,
            "comment": ""
        }
    ]

    # Then load some data
    $ tripaille analysis add_analysis \
        "My cool analysis" \
        "Something" \
```

(continues on next page)

(continued from previous page)

```
        "v1.0" \
        "src"


    $ tripaille analysis get_analyses
    [
        {
            "analysis_id": "68",
            "name": "My cool analysis",
            "description": "",
            "program": "Something",
            "programversion": "1.0",
            "timeexecuted": "2018-02-03 00:00:00"
        },
    ]

    $ tripaille analysis load_fasta \
        --analysis_id 68 \
        --sequence_type contig \
        --organism_id 17 \
        ./test-data/Citrus_sinensis-scaffold00001.fasta
```

1.4 History

- 3.2.1
  - Fix error when loading blast or interpro with empty description
- 3.2
  - Fix support for elasticsearch indexing with Tripal 3
- 3.1.1
  - Minor release to fix broken package at pypi, no code change
- 3.1
  - Add expression module to manage biomaterials and expression data (with tripal_expression module)
  - Add entity.get_bundles() and entity.publish() methods for Tripal 3
- 3.0
  - Added some support for Tripal 3
    - Add `job add_import_job` for generic Tripal 3 importer
    - Add preliminary code for entity management (waiting for https://github.com/tripal/tripal/issues/202)
    - sync and delete_orphans are not yet implemented (https://github.com/tripal/tripal/issues/337)
  - GFF3: removed bugged 'refresh' and 'remove' loading mode (no more available in Tripal3)
  - Renamed `organism get_organism_nodes` to `organism get_organisms_tripal` and `analysis get_analysis_nodes` to `analysis get_analyses_tripal`. Both now return Drupal Nodes for Tripal 2 or Entities for Tripal 3.
  - Added `delete_orphans` methods for organisms and analyses
  - Added tests

- 2.0.4
  - Small bug fixes
- 2.0.3
  - More reliable detection of job failures
- 2.0.2
  - Fix broken pip install tripal
- 2.0.1
  - Fix missing requirements
- 2.0
  - Rewritten most of the code, now working in a similar way as parsec or chakin
  - New cli tool named 'tripaille'
  - Tripal jobs can now be run directly by python-tripal and stdout and stderr are retrieved at the end of jobs.
  - Updated indexing code to latest tripal_elasticsearch module
- 1.9
  - Wait for job completion and get logs

1.5 License

CHAPTER 2

Commands

Tripaille is a set of wrappers for accessing Tripal. Each utility is implemented as a subcommand of `tripaille`. This section of the documentation describes these commands.

2.1 analysis

This section is auto-generated from the help text for the tripaille command `analysis`.

2.1.1 `add_analysis` command

Usage:

```
tripaille analysis add_analysis [OPTIONS] NAME PROGRAM PROGRAMVERSION
```

Help

Create an analysis

Output

Analysis information

Options:

```
--algorithm TEXT      analysis algorithm
--sourceversion TEXT  analysis sourceversion
--sourceuri TEXT      analysis sourceuri
--description TEXT    analysis description
--date_executed TEXT  analysis date_executed (yyyy-mm-dd)
-h, --help            Show this message and exit.
```

2.1.2 `delete_orphans` command

Usage:

```
tripaille analysis delete_orphans [OPTIONS]
```

Help

Delete orphans Drupal analysis nodes

Output

status

Options:

```
--job_name TEXT  Name of the job
--no_wait        Return immediately without waiting for job completion
-h, --help       Show this message and exit.
```

2.1.3 `get_analyses` command

Usage:

```
tripaille analysis get_analyses [OPTIONS]
```

Help

Get analyses

Output

Analysis information

Options:

```
--analysis_id TEXT     An analysis ID
--name TEXT            analysis name
--program TEXT         analysis program
--programversion TEXT  analysis programversion
--algorithm TEXT       analysis algorithm
--sourcename TEXT      analysis sourcename
--sourceversion TEXT   analysis sourceversion
--sourceuri TEXT       analysis sourceuri
--date_executed TEXT   analysis date_executed (yyyy-mm-dd)
-h, --help             Show this message and exit.
```

2.1.4 `get_analyses_tripal` command

Usage:

```
tripaille analysis get_analyses_tripal [OPTIONS]
```

Help

Get analysis entities

Output

Analysis entity/node information

Options:

```
--analysis_id INTEGER  An analysis entity/node ID
-h, --help             Show this message and exit.
```

2.1.5 `load_blast` command

Usage:

```
tripaille analysis load_blast [OPTIONS] NAME PROGRAM PROGRAMVERSION
```

Help

Create a Blast analysis

Output

Loading information

Options:

```
--blast_ext TEXT          If looking for files in a directory, extension of the
                          blast result files
--blastdb TEXT            Name of the database blasted against (must be in the
                          Chado db table)
--blastdb_id TEXT         ID of the database blasted against (must be in the
                          Chado db table)
--blast_parameters TEXT   Blast parameters used to produce these results
--query_re TEXT           The regular expression that can uniquely identify the
                          query name. This parameters is required if the
                          feature name is not the first word in the blast query
                          name.
--query_type TEXT         The feature type (e.g. 'gene', 'mRNA', 'contig') of
                          the query. It must be a valid Sequence Ontology term.
--query_uniquename        Use this if the --query-re regular expression matches
                          unique names instead of names in the database.
--is_concat               If the blast result file is simply a list of
                          concatenated blast results.
--search_keywords         Extract keywords for Tripal search
--no_parsed TEXT          Maximum number of hits to parse per feature.
                          Default=all  [default: all]
--no_wait                 Do not wait for job to complete
--algorithm TEXT          analysis algorithm
--sourceversion TEXT      analysis sourceversion
--sourceuri TEXT          analysis sourceuri
--description TEXT        analysis description
--date_executed TEXT      analysis date_executed (yyyy-mm-dd)
-h, --help                Show this message and exit.
```

2.1.6 `load_fasta` command

Usage:

```
tripaille analysis load_fasta [OPTIONS] FASTA
```

Help

Load fasta sequences

Output

Loading information

Options:

```
--organism TEXT           Organism common name or abbreviation
--organism_id INTEGER     Organism ID
--analysis TEXT           Analysis name
--analysis_id INTEGER     Analysis ID
--sequence_type TEXT      Sequence type  [default: contig]
--re_name TEXT            Regular expression for the name
--re_uniquename TEXT      Regular expression for the unique name
--db_ext_id TEXT          External DB ID
--re_accession TEXT       Regular expression for the accession from external DB
--rel_type TEXT           Relation type (part_of or derives_from)
--rel_subject_re TEXT     Relation subject regular expression (used to extract
                          id of related entity)
--rel_subject_type TEXT   Relation subject type (must match already loaded
                          data, e.g. mRNA)
--method TEXT             Insertion method (insert, update or insup,
                          default=insup (Insert and Update))  [default: insup]
--match_type TEXT         Match type for already loaded features (name or
                          uniquename; default=uniquename; used for "Update
                          only" or "Insert and update" methods)'  [default:
                          uniquename]
--job_name TEXT           Name of the job
--no_wait                 Do not wait for job to complete
-h, --help                Show this message and exit.
```

2.1.7 `load_gff3` command

Usage:

```
tripaille analysis load_gff3 [OPTIONS] GFF
```

Help

Load GFF3 file

Output

Loading information

Options:

```
--organism TEXT               Organism common name or abbreviation
--organism_id INTEGER         Organism ID
--analysis TEXT               Analysis name
--analysis_id INTEGER         Analysis ID
--import_mode TEXT            Import mode (add_only=existing features won't be
                              touched, update=existing features will be
                              updated and obsolete attributes kept)')
                              [default: update]
--target_organism TEXT        In case of Target attribute in the GFF3, choose
```

(continues on next page)

(continued from previous page)

```
                                the organism abbreviation or common name to
                                which target sequences belong. Select this only
                                if target sequences belong to a different
                                organism than the one specified with --organism-
                                id. And only choose an organism here if all of
                                the target sequences belong to the same species.
                                If the targets in the GFF file belong to
                                multiple different species then the organism
                                must be specified using the
                                'target_organism=genus:species' attribute in the
                                GFF file.')
--target_organism_id INTEGER    In case of Target attribute in the GFF3, choose
                                the organism ID to which target sequences
                                belong. Select this only if target sequences
                                belong to a different organism than the one
                                specified with --organism-id. And only choose an
                                organism here if all of the target sequences
                                belong to the same species. If the targets in
                                the GFF file belong to multiple different
                                species then the organism must be specified
                                using the 'target_organism=genus:species'
                                attribute in the GFF file.')
--target_type TEXT              In case of Target attribute in the GFF3, if the
                                unique name for a target sequence is not unique
                                (e.g. a protein and an mRNA have the same name)
                                then you must specify the type for all targets
                                in the GFF file. If the targets are of different
                                types then the type must be specified using the
                                'target_type=type' attribute in the GFF file.
                                This must be a valid Sequence Ontology (SO)
                                term.')
--target_create                 In case of Target attribute in the GFF3, if the
                                target feature cannot be found, create one using
                                the organism and type specified above, or using
                                the 'target_organism' and 'target_type' fields
                                specified in the GFF file. Values specified in
                                the GFF file take precedence over those
                                specified above.')
--start_line INTEGER            The line in the GFF file where importing should
                                start
--landmark_type TEXT            A Sequence Ontology type for the landmark
                                sequences in the GFF fie (e.g. 'chromosome').
--alt_id_attr TEXT              When ID attribute is absent, specify which other
                                attribute can uniquely identify the feature.
--create_organism               Create organisms when encountering organism
                                attribute (these lines will be skip otherwise)
--re_mrna TEXT                  Regular expression for the mRNA name
--re_protein TEXT               Replacement string for the protein name
--job_name TEXT                 Name of the job
--no_wait                       Do not wait for job to complete
-h, --help                      Show this message and exit.
```

2.1.8 `load_go` command

Usage:

```
tripaille analysis load_go [OPTIONS] NAME PROGRAM PROGRAMVERSION
```

Help

Create a GO analysis

Output

Loading information

Options:

```
--organism TEXT         Organism common name or abbreviation
--organism_id INTEGER   Organism ID
--gaf_ext TEXT          If looking for files in a directory, extension of the
                        GAF files
--query_type TEXT       The feature type (e.g. 'gene', 'mRNA', 'contig') of the
                        query. It must be a valid Sequence Ontology term.
--query_matching TEXT   Method to match identifiers to features in the
                        database. ('name', 'uniquename' or 'dbxref')  [default:
                        uniquename]
--method TEXT           Import method ('add' or 'remove')  [default: add]
--name_column INTEGER   Column containing the feature identifiers (2, 3, 10 or
                        11; default=2).  [default: 2]
--re_name TEXT          Regular expression to extract the feature name from GAF
                        file.
--no_wait               Do not wait for job to complete
--algorithm TEXT        analysis algorithm
--sourceversion TEXT    analysis sourceversion
--sourceuri TEXT        analysis sourceuri
--description TEXT      analysis description
--date_executed TEXT    analysis date_executed (yyyy-mm-dd)
-h, --help              Show this message and exit.
```

2.1.9 `load_interpro` command

Usage:

```
tripaille analysis load_interpro [OPTIONS] NAME PROGRAM PROGRAMVERSION
```

Help

Create an Interpro analysis

Output

Loading information

Options:

```
--interpro_parameters TEXT  InterProScan parameters used to produce these
                            results
--query_re TEXT             The regular expression that can uniquely identify
                            the query name. This parameters is required if the
                            feature name is not the first word in the blast
                            query name.
--query_type TEXT           The feature type (e.g. 'gene', 'mRNA', 'contig')
                            of the query. It must be a valid Sequence Ontology
```

(continues on next page)

(continued from previous page)

```
                              term.
--query_uniquename            Use this if the query_re regular expression
                              matches unique names instead of names in the
                              database.
--parse_go                    Load GO annotation to the database
--no_wait                     Do not wait for job to complete
--algorithm TEXT              analysis algorithm
--sourceversion TEXT          analysis sourceversion
--sourceuri TEXT              analysis sourceuri
--description TEXT            analysis description
--date_executed TEXT          analysis date_executed (yyyy-mm-dd)
-h, --help                    Show this message and exit.
```

2.1.10 `sync` command

Usage:

```
tripaille analysis sync [OPTIONS]
```

Help

Synchronize an analysis

Output

status

Options:

```
--analysis TEXT     Analysis name
--analysis_id TEXT  ID of the analysis to sync
--job_name TEXT     Name of the job
--no_wait           Return immediately without waiting for job completion
-h, --help          Show this message and exit.
```

2.2 db

This section is auto-generated from the help text for the tripaille command `db`.

2.2.1 `get_dbs` command

Usage:

```
tripaille db get_dbs [OPTIONS]
```

Help

Get all dbs

Output

Dbs information

Options:

```
--db_id TEXT  A db ID
--name TEXT   filter on db name
-h, --help    Show this message and exit.
```

2.2.2 `get_mviews` command

Usage:

```
tripaille db get_mviews [OPTIONS]
```

Help

Get all materialized views

Output

materialized views information

Options:

```
--name TEXT  filter on mview name
-h, --help   Show this message and exit.
```

2.2.3 `index` command

Usage:

```
tripaille db index [OPTIONS]
```

Help

Schedule database indexing using elasticsearch

Output

Indexing information

Options:

```
--mode TEXT        Indexing mode: 'website' to index everything, 'table' to
                   index a single table (default: website)  [default: website]
--table TEXT       Table to index (only in 'table' mode)
--index_name TEXT  Index name (only in 'table' mode)
--queues INTEGER   Number of indexing task queues  [default: 10]
--fields TEXT      Fields to index (only in 'table' mode), syntax:
                   <field_name>|<field_type>, field_type should be one of
                   'string', 'keyword', 'date', 'long', 'double', 'boolean',
                   'ip', 'object', 'nested', 'geo_point', 'geo_shape', or
                   'completion'
--links TEXT       List of links to show to users, syntax: <column-where-to-
                   show-the-link>|</your/url/[any-column-name]>
--tokenizer TEXT   Tokenizer to use (one of 'standard', 'letter', 'lowercase',
                   'whitespace', 'uax_url_email', 'classic', 'ngram',
                   'edge_ngram', 'keyword', 'pattern', or 'path_hierarchy';
                   default='standard')  [default: standard]
--job_name TEXT    Name of the job
```

(continues on next page)

(continued from previous page)

```
--no_wait           Do not wait for job to complete
-h, --help          Show this message and exit.
```

2.2.4 `populate_mviews` command

Usage:

```
tripaille db populate_mviews [OPTIONS]
```

Help

Populate materialized views

Output

Loading information

Options:

```
--name TEXT  filter on mview name
--no_wait    Do not wait for job to complete
-h, --help   Show this message and exit.
```

2.3 entity

This section is auto-generated from the help text for the tripaille command `entity`.

2.3.1 `add_entity` command

Usage:

```
tripaille entity add_entity [OPTIONS] ENTITY
```

Help

Add a new entity to the database

Output

Entity information

Options:

```
--params TEXT  Values to populate the entity fields
-h, --help     Show this message and exit.
```

2.3.2 `get_bundles` command

Usage:

```
tripaille entity get_bundles [OPTIONS]
```

Help

Get the list of tripal bundles

Output

Bundles information

Options:

```
-h, --help  Show this message and exit.
```

2.3.3 `get_entities` command

Usage:

```
tripaille entity get_entities [OPTIONS]
```

Help

Get entities

Output

Entity information

Options:

```
--entity TEXT        Name of the entity type (e.g. Organism)
--entity_id INTEGER  ID of an entity
-h, --help           Show this message and exit.
```

2.3.4 `get_fields` command

Usage:

```
tripaille entity get_fields [OPTIONS] ENTITY
```

Help

Get the list of available fields for an entity

Output

Fields information

Options:

```
-h, --help  Show this message and exit.
```

2.3.5 `publish` command

Usage:

```
tripaille entity publish [OPTIONS]
```

Help

Publish entities (Tripal 3 only)

Output

status

Options:

```
--types TEXT     List of entity types to be published (e.g. Gene mRNA,
                 default: all)
--job_name TEXT  Name of the job
--no_wait        Return immediately without waiting for job completion
-h, --help       Show this message and exit.
```

2.4 expression

This section is auto-generated from the help text for the tripaille command `expression`.

2.4.1 `add_biomaterial` command

Usage:

```
tripaille expression add_biomaterial [OPTIONS] ORGANISM_ID FILE_PATH
```

Help

Add a new biomaterial to the database

Output

Job information

Options:

```
--no_wait   Do not wait for job to complete
-h, --help  Show this message and exit.
```

2.4.2 `add_expression` command

Usage:

```
tripaille expression add_expression [OPTIONS] ORGANISM_ID ANALYSIS_ID
```

Help

type organism_id str :param organism_id: Organism Id

Output

Loading information

Options:

```
--match_type TEXT             Match to features using either name or
                              uniquename. Default to uniquename  [default:
                              uniquename]
--biomaterial_provider TEXT   The contact who provided the biomaterial.
                              (optional, non functional in Tripal2)
--array_design TEXT           The array design associated with this analysis.
                              This is not required if the experimental data was
                              gathered from next generation sequencing methods.
                              (optional, non functional in Tripal2)
--assay_id TEXT               The id of the assay associated with the
                              experiment. (optional, non functional in Tripal2)
--acquisition_id TEXT         The id of the acquisition associated with the
                              experiment (optional, non functional in Tripal2)
--quantification_id TEXT      The id of the quantification associated with the
                              experiment (optional, non functional in Tripal2)
--file_extension TEXT         File extension for the file(s) to be loaded into
                              Chado. Do not include the ".". Not required for
                              matrix files. (optional)
--start_regex TEXT            A regular expression to describe the line that
                              occurs before the start of the expression data.
                              If the file has no header, this is not needed.
                              (optional)
--stop_regex TEXT             A regular expression to describe the line that
                              occurs after the end of the expression data. If
                              the file has no footer text, this is not needed.
                              (optional)
--use_column                  Set if the expression file is a column file
--no_wait                     Do not wait for job to complete
-h, --help                    Show this message and exit.
```

2.4.3 `delete_biomaterials` command

Usage:

```
tripaille expression delete_biomaterials [OPTIONS]
```

Help

Delete some biomaterials

Output

status

Options:

```
--names TEXT         JSON list of biomaterial names to delete. (optional)
--organism_id TEXT   Organism id from which to delete biomaterials (optional)
--analysis_id TEXT   Analysis id from which to delete biomaterials (optional)
--job_name TEXT      Name of the job (optional)
--no_wait            Return immediately without waiting for job completion
-h, --help           Show this message and exit.
```

2.4.4 `get_biomaterials` command

Usage:

```
tripaille expression get_biomaterials [OPTIONS]
```

Help

List biomaterials in the database

Output

Job information

Options:

```
--provider_id TEXT     Limit query to the selected provider
--biomaterial_id TEXT  Limit query to the selected biomaterial
--organism_id TEXT     Limit query to the selected organism
--dbxref_id TEXT       Limit query to the selected ref
-h, --help             Show this message and exit.
```

2.4.5 `sync_biomaterials` command

Usage:

```
tripaille expression sync_biomaterials [OPTIONS]
```

Help

Synchronize some biomaterials

Output

status

Options:

```
--ids TEXT       JSON list of ids of biomaterials to be synced (default: all)
--max_sync TEXT  Maximum number of features to sync (default: all)
--job_name TEXT  Name of the job
--no_wait        Return immediately without waiting for job completion
-h, --help       Show this message and exit.
```

2.5 feature

This section is auto-generated from the help text for the tripaille command `feature`.

2.5.1 `delete_orphans` command

Usage:

```
tripaille feature delete_orphans [OPTIONS]
```

Help

Delete orphans Drupal feature nodes

Output

status

Options:

```
--job_name TEXT  Name of the job
--no_wait        Return immediately without waiting for job completion
-h, --help       Show this message and exit.
```

2.5.2 `get_features_tripal` command

Usage:

```
tripaille feature get_features_tripal [OPTIONS]
```

Help

Get features entities

Output

Feature entity/node information

Options:

```
--feature_id INTEGER  A feature entity/node ID
-h, --help            Show this message and exit.
```

2.5.3 `sync` command

Usage:

```
tripaille feature sync [OPTIONS]
```

Help

Synchronize some features (Tripal 2 only)

Output

status

Options:

```
--organism TEXT     Common name of the organism to sync
--organism_id TEXT  ID of the organism to sync
--max_sync TEXT     Maximum number of features to sync (default: all)
--types TEXT        List of types of records to be synced (e.g. gene mRNA,
                    default: all)
--ids TEXT          List of names of records to be synced (e.g. gene0001,
                    default: all)
--job_name TEXT     Name of the job
--no_wait           Return immediately without waiting for job completion
-h, --help          Show this message and exit.
```

2.6 job

This section is auto-generated from the help text for the tripaille command `job`.

2.6.1 `add_import_job` command

Usage:

```
tripaille job add_import_job [OPTIONS] NAME IMPORTER INPUT_FILE ARGUMENTS
```

Help

Schedule a new import job

Output

Job information

Options:

```
--priority INTEGER  An integer score to prioritize the job  [default: 10]
-h, --help          Show this message and exit.
```

2.6.2 `add_job` command

Usage:

```
tripaille job add_job [OPTIONS] NAME MODULE CALLBACK ARGUMENTS
```

Help

Schedule a new job

Output

Job information

Options:

```
--priority INTEGER  An integer score to prioritize the job  [default: 10]
-h, --help          Show this message and exit.
```

2.6.3 `get_jobs` command

Usage:

```
tripaille job get_jobs [OPTIONS]
```

Help

Get all jobs

Output

Jobs information

Options:

```
--job_id INTEGER  job id
-h, --help        Show this message and exit.
```

2.6.4 `get_logs` command

Usage:

```
tripaille job get_logs [OPTIONS] STDOUT STDERR
```

Help

Get job output

Output

Output information

Options:

```
-h, --help  Show this message and exit.
```

2.6.5 `run_jobs` command

Usage:

```
tripaille job run_jobs [OPTIONS]
```

Help

Run jobs in queue. There is no way to trigger a single job execution.

Output

Job information

Options:

```
--wait      Wait for job completion  [default: True]
-h, --help  Show this message and exit.
```

2.6.6 `wait` command

Usage:

```
tripaille job wait [OPTIONS] JOB_ID
```

Help

Wait for a job completion

Output

Job information

Options:

```
-h, --help  Show this message and exit.
```

2.7 organism

This section is auto-generated from the help text for the tripaille command `organism`.

2.7.1 `add_organism` command

Usage:

```
tripaille organism add_organism [OPTIONS] GENUS SPECIES
```

Help

Add a new organism to the database

Output

Organism information

Options:

```
--common TEXT               The common name of the organism
--abbr TEXT                 The abbreviation of the organism
--comment TEXT              A comment / description
--infraspecific_rank TEXT   The type name of infraspecific name for any taxon
                            below the rank of species. Must be one of
                            ['subspecies', 'varietas', 'subvariety', 'forma',
                            'subforma']
--infraspecific_name TEXT   The infraspecific name for this organism.
-h, --help                  Show this message and exit.
```

2.7.2 `delete_orphans` command

Usage:

```
tripaille organism delete_orphans [OPTIONS]
```

Help

Delete orphans Drupal organism nodes

Output

status

Options:

```
--job_name TEXT  Name of the job
--no_wait        Return immediately without waiting for job completion
-h, --help       Show this message and exit.
```

2.7.3 `get_organisms` command

Usage:

```
tripaille organism get_organisms [OPTIONS]
```

Help

Get organisms from chado table

Output

Organism information

Options:

```
--organism_id TEXT  An organism ID
--genus TEXT        The genus of the organism
--species TEXT      The species of the organism
--common TEXT       The common name of the organism
--abbr TEXT         The abbreviation of the organism
--comment TEXT      A comment / description
-h, --help          Show this message and exit.
```

2.7.4 `get_organisms_tripal` command

Usage:

```
tripaille organism get_organisms_tripal [OPTIONS]
```

Help

Get organism entities

Output

Organism entity information

Options:

```
--organism_id INTEGER  An organism entity ID
-h, --help             Show this message and exit.
```

2.7.5 `get_taxonomic_ranks` command

Usage:

```
tripaille organism get_taxonomic_ranks [OPTIONS]
```

Help

Get taxonomic ranks

Output

Taxonomic ranks

Options:

```
-h, --help  Show this message and exit.
```

2.7.6 `sync` command

Usage:

```
tripaille organism sync [OPTIONS]
```

Help

Synchronize an organism

Output

status

Options:

```
--organism TEXT     Common name of the organism to sync
--organism_id TEXT  ID of the organism to sync
--job_name TEXT     Name of the job
--no_wait           Return immediately without waiting for job completion
-h, --help          Show this message and exit.
```

2.8 phylogeny

This section is auto-generated from the help text for the tripaille command `phylogeny`.

2.8.1 `sync` command

Usage:

```
tripaille phylogeny sync [OPTIONS]
```

Help

Synchronize some phylotree

Output

status

Options:

```
--max_sync TEXT  Maximum number of features to sync (default: all)
--job_name TEXT  Name of the job
--no_wait        Return immediately without waiting for job completion
-h, --help       Show this message and exit.
```

CHAPTER 3

tripal package

3.1 Subpackages

3.1.1 tripal.analysis package

Module contents

class tripal.analysis.**AnalysisClient**(*tripalinstance*, ***requestArgs*)

Bases: *tripal.client.Client*

Manage Tripal analyses

add_analysis(*name*, *program*, *programversion*, *sourcename*, *algorithm=None*, *sourceversion=None*, *sourceuri=None*, *description=None*, *date_executed=None*)

Create an analysis

Parameters

- **name** (*str*) – analysis name
- **program** (*str*) – analysis program
- **programversion** (*str*) – analysis programversion
- **algorithm** (*str*) – analysis algorithm
- **sourcename** (*str*) – analysis sourcename
- **sourceversion** (*str*) – analysis sourceversion
- **sourceuri** (*str*) – analysis sourceuri
- **description** (*str*) – analysis description
- **date_executed** (*str*) – analysis date_executed (yyyy-mm-dd)

Return type dict

Returns Analysis information

delete_orphans(*job_name=None*, *no_wait=None*)
Delete orphans Drupal analysis nodes

Parameters

- **job_name** (*str*) – Name of the job
- **no_wait** (*bool*) – Return immediately without waiting for job completion

Return type str

Returns status

get_analyses(*analysis_id=None*, *name=None*, *program=None*, *programversion=None*, *algorithm=None*, *sourcename=None*, *sourceversion=None*, *sourceuri=None*, *date_executed=None*)
Get analyses

Parameters

- **analysis_id** (*str*) – An analysis ID
- **name** (*str*) – analysis name
- **program** (*str*) – analysis program
- **programversion** (*str*) – analysis programversion
- **algorithm** (*str*) – analysis algorithm
- **sourcename** (*str*) – analysis sourcename
- **sourceversion** (*str*) – analysis sourceversion
- **sourceuri** (*str*) – analysis sourceuri
- **date_executed** (*str*) – analysis date_executed (yyyy-mm-dd)

Return type list of dict

Returns Analysis information

get_analyses_tripal(*analysis_id=None*)
Get analysis entities

Parameters **analysis_id** (*int*) – An analysis entity/node ID

Return type list of dict

Returns Analysis entity/node information

load_blast(*name*, *program*, *programversion*, *sourcename*, *blast_output*, *blast_ext=None*, *blastdb=None*, *blastdb_id=None*, *blast_parameters=None*, *query_re=None*, *query_type=None*, *query_uniquename=False*, *is_concat=False*, *search_keywords=False*, *no_parsed=u'all'*, *no_wait=False*, *algorithm=None*, *sourceversion=None*, *sourceuri=None*, *description=u''*, *date_executed=None*)
Create a Blast analysis

Parameters

- **name** (*str*) – analysis name
- **program** (*str*) – analysis program
- **programversion** (*str*) – analysis programversion
- **sourcename** (*str*) – analysis sourcename

- **blast_output** (*str*) – Path to the Blast file to load (single XML file, or directory containing multiple XML files)
- **blast_ext** (*str*) – If looking for files in a directory, extension of the blast result files
- **blastdb** (*str*) – Name of the database blasted against (must be in the Chado db table)
- **blastdb_id** (*str*) – ID of the database blasted against (must be in the Chado db table)
- **blast_parameters** (*str*) – Blast parameters used to produce these results
- **query_re** (*str*) – The regular expression that can uniquely identify the query name. This parameters is required if the feature name is not the first word in the blast query name.
- **query_type** (*str*) – The feature type (e.g. 'gene', 'mRNA', 'contig') of the query. It must be a valid Sequence Ontology term.
- **query_uniquename** (*bool*) – Use this if the –query-re regular expression matches unique names instead of names in the database.
- **is_concat** (*bool*) – If the blast result file is simply a list of concatenated blast results.
- **search_keywords** (*bool*) – Extract keywords for Tripal search
- **no_parsed** (*str*) – Maximum number of hits to parse per feature. Default=all
- **no_wait** (*bool*) – Do not wait for job to complete
- **algorithm** (*str*) – analysis algorithm
- **sourceversion** (*str*) – analysis sourceversion
- **sourceuri** (*str*) – analysis sourceuri
- **description** (*str*) – analysis description
- **date_executed** (*str*) – analysis date_executed (yyyy-mm-dd)

Return type str

Returns Loading information

load_fasta(*fasta*, *organism=None*, *organism_id=None*, *analysis=None*, *analysis_id=None*, *sequence_type=u'contig'*, *re_name=None*, *re_uniquename=None*, *db_ext_id=None*, *re_accession=None*, *rel_type=None*, *rel_subject_re=None*, *rel_subject_type=None*, *method=u'insup'*, *match_type=u'uniquename'*, *job_name=None*, *no_wait=False*)

Load fasta sequences

Parameters

- **fasta** (*str*) – Path to the Fasta file to load
- **organism** (*str*) – Organism common name or abbreviation
- **organism_id** (*int*) – Organism ID
- **analysis** (*str*) – Analysis name
- **analysis_id** (*int*) – Analysis ID
- **sequence_type** (*str*) – Sequence type
- **re_name** (*str*) – Regular expression for the name
- **re_uniquename** (*str*) – Regular expression for the unique name
- **db_ext_id** (*str*) – External DB ID

- **re_accession** (*str*) – Regular expression for the accession from external DB
- **rel_type** (*str*) – Relation type (part_of or derives_from)
- **rel_subject_re** (*str*) – Relation subject regular expression (used to extract id of related entity)
- **rel_subject_type** (*str*) – Relation subject type (must match already loaded data, e.g. mRNA)
- **method** (*str*) – Insertion method (insert, update or insup, default=insup (Insert and Update))
- **match_type** (*str*) – Match type for already loaded features (name or uniquename; default=uniquename; used for "Update only" or "Insert and update" methods)'
- **job_name** (*str*) – Name of the job
- **no_wait** (*bool*) – Do not wait for job to complete

Return type str

Returns Loading information

load_gff3(*gff*, *organism=None*, *organism_id=None*, *analysis=None*, *analysis_id=None*, *import_mode=u'update'*, *target_organism=None*, *target_organism_id=None*, *target_type=None*, *target_create=False*, *start_line=None*, *landmark_type=None*, *alt_id_attr=None*, *create_organism=False*, *re_mrna=None*, *re_protein=None*, *job_name=None*, *no_wait=False*)

Load GFF3 file

Parameters

- **gff** (*str*) – Path to the GFF file to load
- **organism** (*str*) – Organism common name or abbreviation
- **organism_id** (*int*) – Organism ID
- **analysis** (*str*) – Analysis name
- **analysis_id** (*int*) – Analysis ID
- **import_mode** (*str*) – Import mode (add_only=existing features won't be touched, update=existing features will be updated and obsolete attributes kept)')
- **target_organism** (*str*) – In case of Target attribute in the GFF3, choose the organism abbreviation or common name to which target sequences belong. Select this only if target sequences belong to a different organism than the one specified with –organism-id. And only choose an organism here if all of the target sequences belong to the same species. If the targets in the GFF file belong to multiple different species then the organism must be specified using the 'target_organism=genus:species' attribute in the GFF file.')
- **target_organism_id** (*int*) – In case of Target attribute in the GFF3, choose the organism ID to which target sequences belong. Select this only if target sequences belong to a different organism than the one specified with –organism-id. And only choose an organism here if all of the target sequences belong to the same species. If the targets in the GFF file belong to multiple different species then the organism must be specified using the 'target_organism=genus:species' attribute in the GFF file.')
- **target_type** (*str*) – In case of Target attribute in the GFF3, if the unique name for a target sequence is not unique (e.g. a protein and an mRNA have the same name) then you must specify the type for all targets in the GFF file. If the targets are of different types

then the type must be specified using the 'target_type=type' attribute in the GFF file. This must be a valid Sequence Ontology (SO) term.')

- **target_create** (*bool*) – In case of Target attribute in the GFF3, if the target feature cannot be found, create one using the organism and type specified above, or using the 'target_organism' and 'target_type' fields specified in the GFF file. Values specified in the GFF file take precedence over those specified above.')
- **start_line** (*int*) – The line in the GFF file where importing should start
- **landmark_type** (*str*) – A Sequence Ontology type for the landmark sequences in the GFF fie (e.g. 'chromosome').
- **alt_id_attr** (*str*) – When ID attribute is absent, specify which other attribute can uniquely identify the feature.
- **create_organism** (*bool*) – Create organisms when encountering organism attribute (these lines will be skip otherwise)
- **re_mrna** (*str*) – Regular expression for the mRNA name
- **re_protein** (*str*) – Replacement string for the protein name
- **job_name** (*str*) – Name of the job
- **no_wait** (*bool*) – Do not wait for job to complete

Return type str

Returns Loading information

load_go(*name*, *program*, *programversion*, *sourcename*, *gaf_output*, *organism=None*, *organism_id=None*, *gaf_ext=None*, *query_type=None*, *query_matching=u'uniquename'*, *method=u'add'*, *name_column=2*, *re_name=None*, *no_wait=False*, *algorithm=None*, *sourceversion=None*, *sourceuri=None*, *description=None*, *date_executed=None*)

Create a GO analysis

Parameters

- **organism** (*str*) – Organism common name or abbreviation
- **organism_id** (*int*) – Organism ID
- **name** (*str*) – analysis name
- **program** (*str*) – analysis program
- **programversion** (*str*) – analysis programversion
- **sourcename** (*str*) – analysis sourcename
- **gaf_output** (*str*) – Path to the GAF file to load (single file, or directory containing multiple GAF files)
- **gaf_ext** (*str*) – If looking for files in a directory, extension of the GAF files
- **query_type** (*str*) – The feature type (e.g. 'gene', 'mRNA', 'contig') of the query. It must be a valid Sequence Ontology term.
- **query_matching** (*str*) – Method to match identifiers to features in the database. ('name', 'uniquename' or 'dbxref')
- **method** (*str*) – Import method ('add' or 'remove')
- **name_column** (*int*) – Column containing the feature identifiers (2, 3, 10 or 11; default=2).

- **re_name** (*str*) – Regular expression to extract the feature name from GAF file.
- **no_wait** (*bool*) – Do not wait for job to complete
- **algorithm** (*str*) – analysis algorithm
- **sourceversion** (*str*) – analysis sourceversion
- **sourceuri** (*str*) – analysis sourceuri
- **description** (*str*) – analysis description
- **date_executed** (*str*) – analysis date_executed (yyyy-mm-dd)

Return type str

Returns Loading information

load_interpro(*name*, *program*, *programversion*, *sourcename*, *interpro_output*, *interpro_parameters=None*, *query_re=None*, *query_type=None*, *query_uniquename=False*, *parse_go=False*, *no_wait=False*, *algorithm=None*, *sourceversion=None*, *sourceuri=None*, *description=u”*, *date_executed=None*)

Create an Interpro analysis

Parameters

- **name** (*str*) – analysis name
- **program** (*str*) – analysis program
- **programversion** (*str*) – analysis programversion
- **sourcename** (*str*) – analysis sourcename
- **interpro_output** (*str*) – Path to the InterProScan file to load (single XML file, or directory containing multiple XML files)
- **interpro_parameters** (*str*) – InterProScan parameters used to produce these results
- **query_re** (*str*) – The regular expression that can uniquely identify the query name. This parameters is required if the feature name is not the first word in the blast query name.
- **query_type** (*str*) – The feature type (e.g. ‘gene’, ‘mRNA’, ‘contig’) of the query. It must be a valid Sequence Ontology term.
- **query_uniquename** (*bool*) – Use this if the query_re regular expression matches unique names instead of names in the database.
- **parse_go** (*bool*) – Load GO annotation to the database
- **no_wait** (*bool*) – Do not wait for job to complete
- **algorithm** (*str*) – analysis algorithm
- **sourceversion** (*str*) – analysis sourceversion
- **sourceuri** (*str*) – analysis sourceuri
- **description** (*str*) – analysis description
- **date_executed** (*str*) – analysis date_executed (yyyy-mm-dd)

Return type str

Returns Loading information

sync (*analysis=None*, *analysis_id=None*, *job_name=None*, *no_wait=None*)
Synchronize an analysis

Parameters

- **analysis** (*str*) – Analysis name
- **analysis_id** (*str*) – ID of the analysis to sync
- **job_name** (*str*) – Name of the job
- **no_wait** (*bool*) – Return immediately without waiting for job completion

Return type str

Returns status

3.1.2 tripal.db package

Module contents

class tripal.db.**DbClient** (*tripalinstance*, ***requestArgs*)
Bases: *tripal.client.Client*

Access Tripal/Chado database

get_dbs (*db_id=None*, *name=None*)
Get all dbs

Parameters

- **db_id** (*str*) – A db ID
- **name** (*str*) – filter on db name

Return type list of dict

Returns Dbs information

get_mviews (*name=None*)
Get all materialized views

Parameters **name** (*str*) – filter on mview name

Return type list of dict

Returns materialized views information

index (*mode=u'website'*, *table=None*, *index_name=None*, *queues=10*, *fields=[]*, *links={}*, *tokenizer=u'standard'*, *token_filters=[]*, *exposed=False*, *index_url=None*, *job_name=None*, *no_wait=False*)
Schedule database indexing using elasticsearch

Parameters

- **mode** (*str*) – Indexing mode: 'website' to index the website , 'nodes' for the website nodes, 'entities' for the website entities (Tripal 3), 'table' to index a single table, 'gene' to build a Gene search index (Tripal 3 only) (default: website) ('website' default to 'nodes' for Tripal 2, 'entities' for Tripal 3)
- **table** (*str*) – Table to index (only in 'table' mode)
- **index_name** (*str*) – Index name (only in 'table' mode)
- **queues** (*int*) – Number of indexing task queues (Tripal 2 only)

- **fields** (*list of str*) – Fields to index (only in 'table' mode), syntax: <field_name>|<field_type>, field_type should be one of 'string' (Tripal2), 'text' (Tripal3), 'keyword', 'date', 'long', 'double', 'boolean', 'ip', 'object', 'nested', 'geo_point', 'geo_shape', or 'completion'
- **links** (*list of str*) – List of links to show to users, syntax: <column-where-to-show-the-link>|</your/url/[any-column-name]> (Tripal 2 only)
- **tokenizer** (*str*) – Tokenizer to use (only in 'table' mode) (one of 'standard', 'letter', 'lowercase', 'whitespace', 'uax_url_email', 'classic', 'ngram', 'edge_ngram', 'keywordx', 'pattern', or 'path_hierarchy'; default='standard')
- **token_filters** (*list of str*) – Token filters (Tripal 3 only) (only in 'table' mode) (available filters are 'standard', 'asciifolding', 'length', 'lowercase', 'uppercase') (Default to ['standard', 'lowercase'])
- **exposed** (*bool*) – "Expose the index (read-only) to other websites
- **index_url** (*str*) – In order for other sites to link back to your results page, you must specify a path where the form for this index can be reached
- **job_name** (*str*) – Name of the job
- **no_wait** (*bool*) – Do not wait for job to complete

Return type str

Returns Indexing information

populate_mviews(*name=None*, *no_wait=None*)
Populate materialized views

Parameters

- **name** (*str*) – filter on mview name
- **no_wait** (*bool*) – Do not wait for job to complete

Return type str

Returns Loading information

tune()
Setup default entity index priority for whole website indexing

Return type dict

Returns "Status"

3.1.3 tripal.entity package

Module contents

class tripal.entity.**EntityClient**(*tripalinstance*, ***requestArgs*)
Bases: *tripal.client.Client*

Manage any type of Tripal entities

add_entity(*entity*, *params={}*)
Add a new entity to the database

Parameters

- **entity** (*str*) – Name of the entity

- **params** (*dict*) – Values to populate the entity fields

Return type dict

Returns Entity information

get_bundles()
Get the list of tripal bundles

Return type list of dict

Returns Bundles information

get_entities(*entity=None*, *entity_id=None*)
Get entities

Parameters

- **entity** (*str*) – Name of the entity type (e.g. Organism)
- **entity_id** (*int*) – ID of an entity

Return type list of dict

Returns Entity information

get_fields(*entity*)
Get the list of available fields for an entity

Parameters **entity** (*str*) – Name of the entity

Return type dict

Returns Fields information

publish(*types=[]*, *job_name=None*, *no_wait=None*)
Publish entities (Tripal 3 only)

Parameters

- **types** (*list of str*) – List of entity types to be published (e.g. Gene mRNA, default: all)
- **job_name** (*str*) – Name of the job
- **no_wait** (*bool*) – Return immediately without waiting for job completion

Return type str

Returns status

3.1.4 tripal.expression package

Module contents

class tripal.expression.**ExpressionClient**(*tripalinstance*, ***requestArgs*)
Bases: *tripal.client.Client*

Manage Tripal expressions

add_biomaterial(*organism_id*, *file_path*, *file_type*, *analysis_id=None*, *no_wait=False*)
Add a new biomaterial file to the database

Parameters

- **organism_id** (*str*) – The id of the associated organism

- **analysis_id** (*str*) – The id of the associated analysis. Required for TripalV3
- **file_path** (*str*) – The path to the biomaterial file
- **file_type** (*str*) – The type of the biomaterial file (xml, tsv or csv)
- **no_wait** (*bool*) – Do not wait for job to complete

Return type dict

Returns Job information

add_expression(*organism_id*, *analysis_id*, *file_path*, *match_type=u'uniquename'*, *array_design_id=None*, *quantification_units=None*, *file_extension=None*, *start_regex=None*, *stop_regex=None*, *seq_type=None*, *use_column=False*, *no_wait=False*)

Add an expression file to tripal

Parameters

- **organism_id** (*str*) – Organism Id
- **analysis_id** (*str*) – Id of the analysis
- **match_type** (*str*) – Match to features using either name or uniquename. Default to uniquename
- **file_path** (*str*) – Path to the expression file, or directory containing multiple expression files
- **array_design_id** (*str*) – The array design ID associated with this analysis. (Non functional in Tripal2)
- **quantification_units** (*str*) – The units associated with the loaded values (ie, FPKM, RPKM, raw counts).
- **file_extension** (*str*) – File extension for the file(s) to be loaded into Chado. Do not include the ".". Not required for matrix files.
- **start_regex** (*str*) – A regular expression to describe the line that occurs before the start of the expression data. If the file has no header, this is not needed.
- **seq_type** (*str*) – Specify the feature type to associate the data with. (Tripal3 only)
- **stop_regex** (*str*) – A regular expression to describe the line that occurs after the end of the expression data. If the file has no footer text, this is not needed.
- **use_column** (*bool*) – Set if the expression file is a column file
- **no_wait** (*bool*) – Do not wait for job to complete

Return type str

Returns Loading information

delete_biomaterials(*names=[]*, *organism_id=u''*, *analysis_id=u''*, *job_name=u''*, *no_wait=False*)

Delete some biomaterials

Parameters

- **names** (*str*) – JSON list of biomaterial names to delete. (optional)
- **organism_id** (*str*) – Organism id from which to delete biomaterials (optional)
- **analysis_id** (*str*) – Analysis id from which to delete biomaterials (optional)
- **no_wait** (*bool*) – Return immediately without waiting for job completion

- **job_name** (*str*) – Name of the job (optional)

Return type str

Returns status

get_biomaterials(*biomaterial_name=u”*, *provider_id=u”*, *biomaterial_id=u”*, *organism_id=u”*, *dbxref_id=u”*)
List biomaterials in the database

Parameters

- **organism_id** (*str*) – Limit query to the selected organism
- **biomaterial_id** (*str*) – Limit query to the selected biomaterial
- **biomaterial_name** (*str*) – Limit query to the selected biomaterial
- **provider_id** (*str*) – Limit query to the selected provider
- **dbxref_id** (*str*) – Limit query to the selected ref

Return type dict

Returns Biomaterial list

get_biomaterials_tripal(*biomaterial_id=None*)
Get Biomaterial entities

Parameters **biomaterial_id** (*int*) – A biomaterial entity ID

Return type list of dict

Returns Organism entity information

sync_biomaterials(*ids=u’[]’*, *max_sync=u”*, *job_name=None*, *no_wait=False*)
Synchronize some biomaterials

Parameters

- **ids** (*str*) – JSON list of ids of biomaterials to be synced (default: all)
- **max_sync** (*str*) – Maximum number of features to sync (default: all)
- **job_name** (*str*) – Name of the job
- **no_wait** (*bool*) – Return immediately without waiting for job completion

Return type str

Returns status

3.1.5 tripal.feature package

Module contents

class tripal.feature.**FeatureClient**(*tripalinstance*, ***requestArgs*)
Bases: *tripal.client.Client*

Manage Tripal features

delete_orphans(*job_name=None*, *no_wait=None*)
Delete orphans Drupal feature nodes

Parameters

- **job_name** (*str*) – Name of the job

- **no_wait** (*bool*) – Return immediately without waiting for job completion

Return type str

Returns status

get_features(*feature_id=None*)
Get features entities

Parameters **feature_id** (*int*) – A feature entity/node ID

Return type list of dict

Returns Feature entity/node information

get_features_tripal(*feature_id=None*)
Get features entities

Parameters **feature_id** (*int*) – A feature entity/node ID

Return type list of dict

Returns Feature entity/node information

sync(*organism=None*, *organism_id=None*, *max_sync=u''*, *types=[]*, *ids=[]*, *job_name=None*, *no_wait=None*)
Synchronize some features (Tripal 2 only)

Parameters

- **organism** (*str*) – Common name of the organism to sync
- **organism_id** (*str*) – ID of the organism to sync
- **max_sync** (*str*) – Maximum number of features to sync (default: all)
- **types** (*list of str*) – List of types of records to be synced (e.g. gene mRNA, default: all)
- **ids** (*list of str*) – List of names of records to be synced (e.g. gene0001, default: all)
- **job_name** (*str*) – Name of the job
- **no_wait** (*bool*) – Return immediately without waiting for job completion

Return type str

Returns status

3.1.6 tripal.job package

Module contents

class tripal.job.**JobClient**(*tripalinstance*, ***requestArgs*)
Bases: *tripal.client.Client*

Manage Tripal jobs

add_import_job(*name*, *importer*, *input_file*, *arguments*, *priority=10*)
Schedule a new import job

Parameters

- **name** (*str*) – The name of the job

- **importer** (*str*) – The Tripal importer to use (e.g. FASTAImporter)
- **input_file** (*str*) – Local path to the file to import
- **arguments** (*str*) – A JSON string representing an array of arguments (e.g. "['some', 'arg', 42, 'foo']")
- **priority** (*int*) – An integer score to prioritize the job

Return type dict

Returns Job information

add_job(*name*, *module*, *callback*, *arguments*, *priority=10*)
Schedule a new job

Parameters

- **name** (*str*) – The name of the job
- **module** (*str*) – The Tripal module name to invoke
- **callback** (*str*) – The Tripal module callback function to invoke
- **arguments** (*str*) – A JSON string representing an array of arguments (e.g. "['some', 'arg', 42, 'foo']")
- **priority** (*int*) – An integer score to prioritize the job

Return type dict

Returns Job information

get_jobs(*job_id=None*)
Get all jobs

Parameters **job_id** (*int*) – job id

Return type list of dict

Returns Jobs information

get_logs(*stdout*, *stderr*)
Get job output

Parameters

- **stdout** (*str*) – Path to stdout file, as returned by run_jobs
- **stderr** (*str*) – Path to stderr file, as returned by run_jobs

Return type dict

Returns Output information

run_jobs(*wait=True*)
Run jobs in queue. There is no way to trigger a single job execution.

Parameters **wait** (*bool*) – Wait for job completion

Return type dict

Returns Job information

wait(*job_id*)
Wait for a job completion

Parameters **job_id** (*int*) – job id

Return type dict

Returns Job information

3.1.7 tripal.organism package

Module contents

class tripal.organism.**OrganismClient**(*tripalinstance*, ***requestArgs*)

Bases: *tripal.client.Client*

Manage Tripal organisms

add_organism(*genus*, *species*, *common=None*, *abbr=None*, *comment=None*, *infraspecific_rank=None*, *infraspecific_name=None*)

Add a new organism to the database

Parameters

- **genus** (*str*) – The genus of the organism
- **species** (*str*) – The species of the organism
- **common** (*str*) – The common name of the organism
- **abbr** (*str*) – The abbreviation of the organism
- **comment** (*str*) – A comment / description
- **infraspecific_rank** (*str*) – The type name of infraspecific name for any taxon below the rank of species. Must be one of ['subspecies', 'varietas', 'subvariety', 'forma', 'subforma']
- **infraspecific_name** (*str*) – The infraspecific name for this organism.

Return type dict

Returns Organism information

delete_orphans(*job_name=None*, *no_wait=None*)

Delete orphans Drupal organism nodes

Parameters

- **job_name** (*str*) – Name of the job
- **no_wait** (*bool*) – Return immediately without waiting for job completion

Return type str

Returns status

get_organisms(*organism_id=None*, *genus=None*, *species=None*, *common=None*, *abbr=None*, *comment=None*)

Get organisms from chado table

Parameters

- **organism_id** (*str*) – An organism ID
- **genus** (*str*) – The genus of the organism
- **common** (*str*) – The common name of the organism
- **abbr** (*str*) – The abbreviation of the organism

- **species** (*str*) – The species of the organism
- **comment** (*str*) – A comment / description

Return type list of dict

Returns Organism information

get_organisms_tripal(*organism_id=None*)
Get organism entities

Parameters **organism_id** (*int*) – An organism entity ID

Return type list of dict

Returns Organism entity information

get_taxonomic_ranks()
Get taxonomic ranks

Return type list of dict

Returns Taxonomic ranks

sync(*organism=None*, *organism_id=None*, *job_name=None*, *no_wait=None*)
Synchronize an organism

Parameters

- **organism** (*str*) – Common name of the organism to sync
- **organism_id** (*str*) – ID of the organism to sync
- **job_name** (*str*) – Name of the job
- **no_wait** (*bool*) – Return immediately without waiting for job completion

Return type str

Returns status

3.1.8 tripal.phylogeny package

Module contents

class tripal.phylogeny.**PhylogenyClient**(*tripalinstance*, ***requestArgs*)
Bases: *tripal.client.Client*

Manage Tripal phylogeny

sync(*max_sync=u"*, *job_name=None*, *no_wait=None*)
Synchronize some phylotree

Parameters

- **max_sync** (*str*) – Maximum number of features to sync (default: all)
- **job_name** (*str*) – Name of the job
- **no_wait** (*bool*) – Return immediately without waiting for job completion

Return type str

Returns status

3.2 Submodules

3.3 tripal.client module

Base tripal client

class tripal.client.**Client**(*tripalinstance*, ***requestArgs*)

Bases: object

Base client class implementing methods to make queries to the server

CLIENT_BASE = u'/tripal_api/'

3.4 Module contents

class tripal.**TripalInstance**(*tripal_url=u'http://localhost/tripal/'*, *username=u'admin'*, *password=u'changeme'*, *auth_login=None*, *auth_password=None*, ***kwargs*)

Bases: object

Python Module Index

t

Index

O

P

R

S

T

W