vogue

Hassan Foroughi Asl, Maya Brandi

Oct 22, 2021

API AND CLI REFERENCE

Vogue is Clinical Genomics solution for capturing data from various places in the data flow and to trend the data over a longer period of time.

```
git clone https://github.com/Clinical-Genomics/vogue.git
cd vogue
pip install -e .
```

CHAPTER

ONE

RELEASE MODEL

Vogue development is organised on a flexible Git "Release Flow" branching system. This more or less means that we make releases in release branches which corresponds to stable versions of Vogue.

1.1 Steps to make a new release:

1) Create a release branch from master named `version_X.X.X`
2) Update change log with the new version.
3) Make a PR to master,
4) Merge release PR into master
5) Use `bumpversion` to change version accordingly: `bumpversion major` or `bumpversion minor` or `bumpversion patch`
6) Do `git push` and `git push --tags`

```
- Name PR `release version X.X.X`
- Justify if its a patch/minor/major version bump
- Paste the latest changelog to the text body
- get it approved and merge to master. **Dont delete the release branch!**
```

5) Make a new release.

```
- Name tag version as `vX.X.X`
- Set target to the release branch
- Make descriptive title
- Paste latest changelog to the text body
- Release!
```

1.2 Deploying to production

Use `update-vogue-prod.sh` script to update production both on Hasta and Clinical-db. **Please follow the development guide and ``servers`` repo when doing so. It is also important to keep those involved informed.**

CHAPTER

TWO

FRONT END

All views in vogue should be self-explanatory. There should be no further documentation needed to be able to interpret the content of the web page.

CHAPTER

THREE

BACK END

The trending database is a Mongo database consisting of following collections:

- **sample** - holds LIMS specific data on sample level. Anchoring identifier are LIMS sample ids.
- **sample_analysis** - holds data from diferent pipeliens on sample level. Anchoring identifier are lims sample ids.
- **flowcell** - holds LIMS specific data on run level. Anchoring identifier are flowcell ids.
- **application_tag** - holds application tag specific data. Anchoring identifier are application tags.

The load command of each collection is described below.

CHAPTER

FOUR

DATA FLOW

CHAPTER

FIVE

CLI

The CLI has two base commands - load and run. The load command is for loading various data into the trending database, and the run command is for running the web application.

5.1 Load sample

```
Usage: vogue load sample [OPTIONS]

  Read and load lims data for one or all samples. When loading many
  samples, the different options -f, -n, -d are used to delimit the subset
  of samples to load.

Options:
  -s, --sample-lims-id TEXT  Input sample lims id
  -m, --many                 Load all LIMS samples if no other options are
                             selected
  --dry-run                  Load from sample or not. (dry-run)
  -f, --load-from TEXT       load from this sample LIMS id. Use if load all
                             broke. Start where it ended
  -n, --new                  Use this flag if you only want to load samples
                             that do not exist in the database
  -d, --date TEXT            Update only samples delivered after date
  --help                     Show this message and exit.
```

5.2 Load analysis

```
Usage: vogue load analysis [OPTIONS]

  Read and load analysis results. These are either QC or analysis output
  files.

  The inputs are unique ID with an analysis config file (JSON/YAML) which
  includes analysis results matching the analysis model. Analysis types
  recognize the following keys in the input file: QC:multiqc_picard_dups,
  multiqc_picard_HsMetrics, multiqc_picard_AlignmentSummaryMetrics,
  multiqc_picard_insertSize microsalt:blast_pubmlst, quast_assembly,
  blast_resfinder_resistence, picard_markduplicate, microsalt_samtools_stats

Options:
  -s, --sample-id TEXT            Input sample id.  [required]
```

(continues on next page)

(continued from previous page)

```
  -a, --analysis-config PATH      Input config file. Accepted format: JSON,
                                  YAML  [required]
  -t, --analysis-type [QC|microsalt|all]
                                  Type of analysis results to load.
  -c, --analysis-case TEXT        The case that this sample belongs.
                                  It can be
                                  specified multiple times.  [required]
  -w, --analysis-workflow TEXT    Analysis workflow used.  [required]
  --workflow-version TEXT         Analysis workflow used.  [required]
  --is-case                       Specify this flag if input json is case
                                  level.
  --case-analysis-type [multiqc]  Specify the type for the case analysis. i.e.
                                  if it is multiqc output, then choose multiqc
  --dry                           Load from sample or not. (dry-run)
  --help                          Show this message and exit.
↪Show this message and exit.
```

5.3 Load flowcell

```
Usage: vogue load flowcell [OPTIONS]

  Read and load LIMS data for one or all runs

Options:
  -r, --run-id TEXT  Run id for the run. Eg: 190510_A00689_0032_BHJLW2DSXX
  -a, --all-runs     Loads all flowcells found in LIMS.
  --dry              Load from flowcell or not. (dry-run)
  --help             Show this message and exit.
```

5.4 Load apptag

```
Usage: vogue load apptag [OPTIONS] APPLICATION_TAGS

  Reads json string with application tags. Eg:'[{"tag":"MELPCFR030",
  "category":"wgs",...},...]'

Options:
  --help  Show this message and exit.
```

5.5 Run

```
Usage: vogue run [OPTIONS]

  Run a local development server.

  This server is for development purposes only. It does not provide the
  stability, security, or performance of production WSGI servers.
```

(continues on next page)

(continued from previous page)

```
  The reloader and debugger are enabled by default if FLASK_ENV=development
  or FLASK_DEBUG=1.

Options:
  -h, --host TEXT                 The interface to bind to.
  -p, --port INTEGER              The port to bind to.
  --cert PATH                     Specify a certificate file to use HTTPS.
  --key FILE                      The key file to use when specifying a
                                  certificate.
  --reload / --no-reload          Enable or disable the reloader. By default
                                  the reloader is active if debug is enabled.
  --debugger / --no-debugger      Enable or disable the debugger. By default
                                  the debugger is active if debug is enabled.
  --eager-loading / --lazy-loader
                                  Enable or disable eager loading. By default
                                  eager loading is enabled if the reloader is
                                  disabled.
  --with-threads / --without-threads
                                  Enable or disable multithreading.
  --help                          Show this message and exit.
```

5.5.1 vogue

vogue package

Subpackages

vogue.adapter package

Submodules

vogue.adapter.plugin module

class vogue.adapter.plugin.**VogueAdapter**(*client=None*, *db_name=None*)

Bases: mongo_adapter.adapter.MongoAdapter

add_or_update_bioinfo_processed(*analysis_result: dict*)

Functionality to add or update analysis for processed bioinfo stat

add_or_update_bioinfo_raw(*analysis_result: dict*)

Functionality to add or update analysis for unprocessed aka raw bioinfo stat

add_or_update_bioinfo_samples(*analysis_result: dict*)

Functionality to add or update bioinfo analysis for sample level results

add_or_update_document(*document_news: dict*, *collection*)

Adds/updates a document in the database

app_tag(*tag*)

bioinfo_processed(*analysis_id: str*)

Functionality to get analyses results

bioinfo_raw(*analysis_id: str*)

Functionality to get analyses results

bioinfo_samples_aggregate(*pipe: list*)
Function to make a aggregation on the sample analysis colleciton

delete_sample()

find_genotype_plate(*plate_id: str*)
find all samples from plate

find_samples(*query: dict*) → list
Function to find samples in samples collection based on query

flowcell(*run_id*)

flowcells_aggregate(*pipe: list*)
Function to make a aggregation on the flowcell colleciton

genotype_analysis_aggregate(*pipe: list*)
Function to make a aggregation on the genotype analysis colleciton

get_all_reagent_label_names_grouped_by_category()
Function get all reagent label names grouped by category from the reagent_label_category colleciton

get_category(*app_tag*)
Function get category based on application tag from the application tag collection

get_reagent_label_categories()
Function to get all categories from label_category_collection

get_reagent_label_category(*reagent_label*)
Function get category based on application tag from the application tag collection

reagent_label_aggregate(*pipe: list*)
Function to make a aggregation on the reagent_label analysis colleciton

sample(*lims_id*)

sample_analysis(*analysis_id: str*)
Functionality to get analyses results

sample_collection_ids()

samples_aggregate(*pipe: list*)
Function to make a aggregation on the sample colleciton

setup(*db_name: str*)
Setup connection to a database

vogue.adapter.plugin.**check_dates**(*analysis_result*, *current_document*)
Function to pop analysysis results from tne new analysis if the results are older than the current results in the database

Module contents

vogue.build package

Submodules

vogue.build.application_tag module

vogue.build.application_tag.**build_application_tag**(*app_tag: dict*) → dict
Builds the application tag collection documents.

Parameters **app_tag** (*dict*) – {'tag':'MELPCFR030', 'category':'wgs',...}

Returns {'_id':'MELPCFR030', 'category':'wgs'}

Return type mongo_application_tag(dict)

vogue.build.bioinfo_analysis module

vogue.build.bioinfo_analysis.**build_analysis**(*analysis_dict: dict*, *analysis_type: str*, *valid_analysis: list*, *current_analysis: dict*, *process_case=False*, *cleanup=False*)
Builds analysis dictionary based on input analysis_dict and prepares a mongo_doc.

If not process_case, then do not validate any keys in the analysis_dict. This will only load into bioinfo_raw.

If process_case, then extract valid keys from analysis_dict.

vogue.build.bioinfo_analysis.**build_bioinfo_sample**(*analysis_dict: dict*, *sample_id: str*, *process_case=False*)
Builds sample analysis from analysis_dict

analysis_dict is a processed dictionary, i.e. from bioinfo_processed

vogue.build.bioinfo_analysis.**build_mongo_case**(*analysis_dict: dict*, *case_analysis: dict*, *processed=False*)
Build a mongo case document dictionary

vogue.build.bioinfo_analysis.**build_processed_case**(*analysis_dict: dict*, *analysis_type: str*, *valid_analysis: list*, *cleanup=False*)
Builds an analysis dict from input information provided by user.

Input: analysis_dict: A dictionary of bioinfo stats to be prepared for bioinfo_processed collection analysis_type: A string for analysis_type to be extracted from from analysis_dict valid_analysis: A list of valid analysis to found within analysis_dict cleanup: Flag to cleanup unwanted keys from analysis_dict using info from valid_analysis and analysis_type

Output:

case_analysis: A dictionary with information about workflow and case_analysis_type(e.g. multiqc), workflow version, and date added.

vogue.build.bioinfo_analysis.**build_unprocessed_case**(*analysis_dict: dict*)
Prepare a case analysis dictionary

vogue.build.bioinfo_analysis.**extract_valid_analysis**(*analysis_dict: dict*, *analysis_type: str*, *valid_analysis: list*)
Extracts analysis dictionary based on input analysis_dict. This function will remove analysis json that are not part of the matching model. analysis_type is a single key matching ANALYSIS_SETS's first level keys.

Input: analysis_dict: A dictionary of bioinfo analysis stats. analysis_type: A string of analysis type. This is provided by user. valid_analysis: A list of analysis to be extracted from analysis dict.

Output: analysis: A dictionary of valid_analysis as keys extracted from analysis_dict

vogue.build.bioinfo_analysis.**get_common_keys**(*valid_analysis: list*, *analysis_type: str*)
Match a list of values with keys from a MODEL dictionary

input: valid_analysis as list output: analysis_common_keys as list

vogue.build.bioinfo_analysis.**update_mongo_doc_case**(*mongo_doc: dict*, *analysis_dict: dict*, *new_analysis: dict*)

Parameters

- **mongo_doc** – an existing analysis retrieved from MongoDB
- **analysis_dict** – a dictionary parsed from CLI
- **new_analysis** – new analysis dictionary to be loaded to MongoDB

Returns an updated mongo_doc from Args

Return type mongo_doc

Add or update mongo document for case data Adds or updates within processed or raw bioinfo collection

vogue.build.flowcell module

vogue.build.flowcell.**build_run**(*run: genologics.entities.Process*, *instrument: str*, *date: str*) → dict
Build flowcell document from lims data.

vogue.build.reagent_label module

vogue.build.reagent_label.**build_reagent_label**(*step: genologics.entities.Process*) → dict
Build reagent label document from lims data.

vogue.build.reagent_label_category module

vogue.build.reagent_label_category.**build_reagent_label_category**(*lims_reagent_label*) → dict
Build reagent label ctegory document from lims data.

vogue.build.sample module

vogue.build.sample.**build_sample**(*sample: genologics.entities.Sample*, *lims: genologics.lims.Lims*, *adapter*) → dict
Build lims sample

Module contents

vogue.commands package

Subpackages

vogue.commands.load package

Subpackages

vogue.commands.load.bioinfo package

Submodules

vogue.commands.load.bioinfo.base module

cli for handling bioinfo collections. Addition and update!

vogue.commands.load.bioinfo.bioinfo_process module

Functionality to add or update to processed bioinfo collection

vogue.commands.load.bioinfo.bioinfo_raw module

Add or update bioinfo results to bioinfo raw collection

vogue.commands.load.bioinfo.bioinfo_sample module

Add or update analysis results for samples from bioinfo_processed into bioinf_sample collection

Module contents

Submodules

vogue.commands.load.application_tag module

vogue.commands.load.base module

vogue.commands.load.flowcell module

vogue.commands.load.genotype module

vogue.commands.load.reagent_label module

vogue.commands.load.reagent_label_category module

vogue.commands.load.sample module

vogue.commands.load.temp module

Module contents

Submodules

vogue.commands.base module

Module with CLI commands for vogue The CLI is intended for development/testing purpose only. To run in a production setting please refer to documentation for suggestions how.

Module contents

vogue.constants package

Submodules

vogue.constants.constants module

vogue.constants.lims_constants module

Module contents

vogue.load package

Submodules

vogue.load.application_tag module

vogue.load.application_tag.**load_aplication_tags**(*adapter*, *json_list*)

Will go through all application tags in json_list and add/update them to trending-db.

Parameters

- **adapter** (*adapter.VogueAdapter*) –
- **json_list** (*list(dict)*) – [{'tag':'MELPCFR030', 'category':'wgs',...},...]

vogue.load.bioinfo_analysis module

vogue.load.bioinfo_analysis.**load_analysis**(*adapter*, *lims_id*, *analysis*, *processed=False*, *is_sample=False*, *dry_run=False*)

Load information for a bioinfo analysis

vogue.load.flowcell module

vogue.load.flowcell.**load_all**(*adapter*, *lims*)
Function to load all lims flowcell into the database

vogue.load.flowcell.**load_one**(*adapter*, *run*)
Function to load one lims flowcell into the database

vogue.load.flowcell.**load_recent**(*adapter*, *lims*, *the_date*)
Function to load all lims flowcell into the database

vogue.load.genotype module

vogue.load.genotype.**load_sample**(*adapter*, *genotype_sample_string*)

vogue.load.reagent_label module

vogue.load.reagent_label.**load_all**(*adapter*, *lims*)
Function to load reagent_labels from all lims flowcells into the database

vogue.load.reagent_label.**load_one**(*adapter*, *step*)
Function to load reagent_labels from a step into the database

vogue.load.reagent_label.**load_recent**(*adapter*, *lims*, *the_date*)
Function to load reagent_labels from all lims flowcells run after the_date into the database

vogue.load.reagent_label_category module

vogue.load.reagent_label_category.**load_all**(*adapter*, *lims*, *categories*)
Function to load reagent_labels from a step into the database

vogue.load.sample module

vogue.load.sample.**load_all**(*adapter*, *lims*, *start_sample=None*)
Function to load all lims samples into the database

vogue.load.sample.**load_all_dry**()

vogue.load.sample.**load_one**(*adapter*, *lims_sample=None*, *lims=None*)
Function to load one lims sample into the database

vogue.load.sample.**load_one_dry**(*lims_sample*, *lims*, *adapter*)

vogue.load.sample.**load_recent**(*adapter*, *lims*, *the_date*)
Function to load all lims samples into the database

Module contents

vogue.models package

Submodules

vogue.models.bioinfo_analysis module

Module contents

vogue.parse package

Subpackages

vogue.parse.build package

Submodules

vogue.parse.build.flowcell module

vogue.parse.build.flowcell.**filter_none**(*mongo_dict*)
Function to filter out Nones and NaN from a dict.

vogue.parse.build.flowcell.**run_data**(*run*)
Function to get run info from lanes in a lims sequecing process. Reformates the data to be part of a document in the flowcell database.

Parameters **run** (*Process*) – lims Process instance of sequencing type

Returns

run info per lane.

eq: {'Lane 1': {'% Aligned R2': 0.94, '% Bases >=Q30 R1': 90.67, '% Bases >=Q30 R2': 88.84,...}, 'Lane 2': {'% Aligned R2': 0.92, '% Bases >=Q30 R1': 91.67, '% Bases >=Q30 R2': 83.84,...}}

avg_data (dict): average run info over all lanes. eg: {'% Phasing R2': 0.09, '% Bases >=Q30': 89.755, ...}

Return type lane_data (dict)

vogue.parse.build.reagent_label module

vogue.parse.build.reagent_label.**filter_none**(*mongo_dict*)
Function to filter out Nones and NaN from a dict.

vogue.parse.build.reagent_label.**get_define_step_data**(*pool*)
Search the artifact history for the define steps. Input:

pool: lims artifact - Input to bcl step

Returns

dict keys: sample ids value: target reads (udf 'Reads to sequence (M)')

define_step: lims process the define step

flowcell_target_reads: int the summe of the udf 'Reads to sequence (M)' from all outarts in the step

Return type define_step_outputs

vogue.parse.build.reagent_label.**reagent_label_data**(*bcl_step*)

This function takes as input a bcl conversion and demultiplexing step. From that step it goes back in artifact history to the prevoius Define step. Both step types exist in the Nova Seq workflow. From the output artifacts of the bcl step, index_total_reads is calculated:

index_total_reads: the sum of '# Reads' from all artifact with a specific index flowcell_total_reads: the sum of '# Reads' from all output artifacts

From the output artifacts of the define step, index_target_reads and flowcell_target_reads are fetched:
index_target_reads: fetched from the 'Reads to sequence (M)' udf of the output artifact with a specific index flowcell_target_reads: the sum of the 'Reads to sequence (M)' udf of all the output artifacts

vogue.parse.build.sample module

vogue.parse.build.sample.**datetime2date**(*date: datetime.datetime*) → None.datetime.date

Convert datetime.datetime to datetime.date

vogue.parse.build.sample.**get_concentration_and_nr_defrosts**(*application_tag: str*, *lims_id: str*, *lims: genologics.lims.Lims*) → dict

Get concentration and nr of defrosts for wgs illumina PCR-free samples.

Find the latest artifact that passed through a concentration_step and get its concentration_udf. –> concentration Go back in history to the latest lot_nr_step and get the lot_nr_udf from that step. –> lotnr Find all steps where the lot_nr was used. –> all_defrosts Pick out those steps that were performed before our lot_nr_step –> defrosts_before_this_process Count defrosts_before_this_process. –> nr_defrosts

vogue.parse.build.sample.**get_final_conc_and_amount_dna**(*application_tag: str*, *lims_id: str*, *lims: genologics.lims.Lims*) → dict

Find the latest artifact that passed through a concentration_step and get its concentration. Then go back in history to the latest amount_step and get the amount.

vogue.parse.build.sample.**get_latest_input_artifact**(*process_type: str*, *lims_id: str*, *lims: genologics.lims.Lims*) → genologics.entities.Artifact

Returns the input artifact related to lims_id and the step that was latest run.

vogue.parse.build.sample.**get_library_size**(*sample_id: str*, *lims: genologics.lims.Lims*, *size_steps: List[str]*, *workflow: str*) → int

Getting the udf Size (bp) that in fact is set on the aggregate qc librar validation step.

vogue.parse.build.sample.**get_microbial_library_concentration**(*application_tag: str*, *lims_id: str*, *lims: genologics.lims.Lims*) → float

Check only samples with mictobial application tag. Get concentration_udf from concentration_step.

vogue.parse.build.sample.**get_number_of_days**(*first_date: datetime.datetime*, *second_date: datetime.datetime*) → int

Get number of days between different time stamps.

vogue.parse.build.sample.**get_output_artifact**(*process_types: list*, *lims_id: str*, *lims: genologics.lims.Lims*, *last: bool = True*) → genologics.entities.Artifact
Returns the output artifact related to lims_id and the step that was first/latest run.

If last = False return the first artifact

vogue.parse.build.sample.**str_to_datetime**(*date: str*) → datetime.datetime
Convert str to datetime

vogue.parse.build.sample_analysis module

class vogue.parse.build.sample_analysis.**Mip_dna**(*case*)
Bases: object

Class to prepare mip case_analysis results for mip_dna results in the sample_analysis collection

build_mip_dna_sample(*sample_id*)
Bulding the mip analysis for one sample. Returns {} if the date 'added' is empty.

vogue.parse.build.sample_analysis.**get_latest_analysis**(*case*, *analysis_type*)
Get the latest analysis of anaöysis_type from one case

vogue.parse.build.sample_analysis.**reduce_keys**(*dict_long_keys*)
Cut keys generated in the multiqc report. First entry is allways lims sample ID

class vogue.parse.build.sample_analysis.**uSalt**(*project*)
Bases: object

Class to prepare uSalt case_analysis results for uSalt results in the sample_analysis collection

build_uSalt_sample(*sample_id*)
Bulding the uSalt analysis for one sample. Returns {} if the date 'added' is empty.

Module contents

vogue.parse.load package

Submodules

vogue.parse.load.bioinfo_analysis module

vogue.parse.load.bioinfo_analysis.**inspect_analysis_result**(*analysis_dict: dict*)
Takes input analysis_dict dictionary and validates entries.

Checks for there are at least two keys in analysis_dict dictionary. If there is less than two, or the key doesn't exist, disqualifies the file and returns False

Module contents

Module contents

vogue.server package

Subpackages

vogue.server.static package

Module contents

vogue.server.utils package

Submodules

vogue.server.utils.reagent_labels module

vogue.server.utils.utils module

Module contents

Submodules

vogue.server.auto module

vogue.server.extentions module

vogue.server.views module

Module contents

vogue.tools package

Submodules

vogue.tools.cli_utils module

vogue.tools.cli_utils.**add_doc**(*docstring*)
A decorator for adding docstring. Taken shamelessly from stackexchange.

vogue.tools.cli_utils.**check_file**(*fname*)
Check file exists and readable.

vogue.tools.cli_utils.**concat_dict_keys**(*my_dict: dict*, *key_name=''*, *out_key_list=['multiqc:multiqc_picard_dups, multiqc_picard_HsMetrics, multiqc_picard_AlignmentSummaryMetrics, multiqc_picard_insertSize', 'microsalt:blast_pubmlst, quast_assembly, blast_resfinder_resistence, picard_markduplicate, microsalt_samtools_stats', 'multiqc:multiqc_picard_dups, multiqc_picard_HsMetrics, multiqc_picard_AlignmentSummaryMetrics, multiqc_picard_insertSize', 'microsalt:blast_pubmlst, quast_assembly, blast_resfinder_resistence, picard_markduplicate, microsalt_samtools_stats']*)
Recursively create a list of key:key1,key2 from a nested dictionary

vogue.tools.cli_utils.**convert_defaultdict_to_regular_dict**(*inputdict: dict*)
Recursively convert defaultdict to dict.

vogue.tools.cli_utils.**convert_dot**(*string*)
replaces dot with underscore

vogue.tools.cli_utils.**dict_replace_dot**(*obj*)
recursively replace all dots in json.load keys.

vogue.tools.cli_utils.**json_read**(*fname*)
Reads JSON file and returns dictionary. Returns error if can't read.

vogue.tools.cli_utils.**recursive_default_dict**()
Recursivly create defaultdict.

vogue.tools.cli_utils.**yaml_read**(*fname*)
Reads YAML file and returns dictionary. Returns error if can't read.

Module contents

Submodules

vogue.exceptions module

exception vogue.exceptions.**InsertError**(*message: str*, *code: Optional[int] = 405*)
Bases: *vogue.exceptions.VogueRestError*

exception vogue.exceptions.**MissingApplicationTag**
Bases: Exception

exception vogue.exceptions.**VogueError**(*message: str*)
Bases: Exception

exception vogue.exceptions.**VogueRestError**(*message: str*, *code: Optional[int] = None*)
Bases: *vogue.exceptions.VogueError*

Module contents

5.5.2 Build Doc

If you'd like to create Sphinx documentation locally, follow the steps explained below locally. Tested on Conda 4.6.X

1. Create a conda environment:

```
conda create -n vogue_doc -c bioconda -c conda-forge python=3.6 pip
conda activate vogue_doc
```

2. Install Sphinx and extensions:

```
cd docs
pip install -r requirements.txt -r ../requirements-dev.txt -r ../requirements.txt
```

3. Build docs:

```
sphinx-apidoc -o source/ ../vogue
sphinx-build -T -E -b html -d _build/doctrees-readthedocs -D language=en . _build/html
```

4. View docs (`open` or similar command from your OS):

```
open _build/html/index.html
```

PYTHON MODULE INDEX

V

INDEX

R

S

U

V

Y