lesana

Release 0.10.0dev

Elena ``of Valhalla''

Nov 30, 2024

CONTENTS:

lesana is a python3 library to organize collections of various kinds. It is designed to have a data storage / serialization format that is friendly to git and other VCSs, but decent performances.

To reach this aim it uses yaml as its serialization format, which is easy to store in a VCS, share between people and syncronize between different computers, but it also keeps an index of this data in a local xapian database in order to allow for fast searches.

lesana supports collections of any kind, as long as their entries can be described with a mostly flat dictionary of fields of the types described in the documentation file `field_types`.

Some example collection schemas are provided, but one big strenght of lesana is the ability to customize your collection with custom fields by simply writing a personalized `settings.yaml`.

CHAPTER

ONE

INSTALLATION

The recommended way to install lesana is to use the packages available for your distribution; see e.g. the list of distributions that provide lesana on repology.

Alternatively, the source code for lesana can be downloaded from the git repository at https://git.sr.ht/~valhalla/lesana; and releases are made on pypi.

lesana expects to run on a POSIX-like system and requires the following dependencies:

- python3
- xapian
- ruamel.yaml
- jinja2
- dateutil
- GitPython optional, to add git support.

Under debian (and derivatives), the packages to install are:

```
apt install python3-jinja2 python3-ruamel.yaml python3-xapian \
            python3-dateutil python3-git
```

lesana can be run in place from the git checkout / extracted tarball; to use `setup.py` you will also need setuptools (e.g. from the `python3-setuptools` package under debian and derivatives).

CHAPTER

TWO

CONTRIBUTING

Lesana is hosted on sourcehut:

- bug tracker
- mailing lists
- git repository
- CI

CHAPTER

THREE

LICENSE

CHAPTER

FOUR

DOCUMENTATION

The documentation for the latest development version of lesana can be browsed online at https://lesana.trueelena.org; PDF and epub versions are also available[1].

The author can be contacted via email: webmaster AT trueelena DOT org.

[1] Everything is also available via onion, at http://aublvconhsld6cvcf3dbibffzih2un5bicp3s3b5qmkskof26p3pssqd.onion/

CHAPTER

FIVE

CONTENTS

5.1 User Documentation

Documentation that is useful for everybody.

5.1.1 Getting Started (Command Line)

lesana can be used from the command line through the `lesana` command; for more details run `lesana help`.

Many commands will try to open some file in an editor: they will attempt to use, in this order, `$EDITOR`, `sensible-editor` or as a fallback `vi`, which should be installed on any POSIX-like system.

To start a new collection, create a directory and run `lesana init` into it:

```
mkdir $DIRECTORY
cd $DIRECTORY
lesana init
```

It will create the basic file structure of a lesana collection, including a `settings.yaml` skeleton and it will initialize a git repository (use `--no-git` to skip this part and ignore all further git commands).

It will then open `settings.yaml` in an editor: fill in your values for the available variables, and define your list of fields; see *The settings file* for details. Then save and exit, and you are now ready to commit the configuration for your new collection, as the changes have already been added to git:

```
git commit -m 'Collection settings'
```

An empty collection is not very interesting: let us start adding new entries:

```
lesana new
```

It will again open an editor on a skeleton of entry where you can fill in the values. When you close the editor it will print the entry id, that you can use e.g. to edit again the same entry:

```
lesana edit $ENTRY_ID
```

After you've added a few entries, you can now search for some word that you entered in one of the indexed fields:

```
lesana search some words
```

this will also print the entry ids of matching items, so that you can open them with `lesana edit`. See *Search syntax* for more details on the search syntax.

If you're using git, entries will be autoadded to the staging area, but you need to commit them, so that you can choose how often you do so.

Search results are limited by default to 12 matches; to get all results for your query you can use the option `--all`. This is especially useful when passing the results to a template:

```
lesana search --template templates/my_report.html --all \
    <some search terms> \
    > some_search_terms-report.html
```

will generate an html file based on the jinja2 template `templates/my_report.html` with all the entries found for those search terms.

If later on you want to clone the repository elsewhere (using regular git commands) you can use `git init` in the new repository to install the hooks to manage updating the local cache every time the repository is updated via git, and then run `lesana index` to prepare the first version of the cache.

5.1.2 The settings file

The file `settings.yaml` defines the properties of a collection.

It is a yaml file with a dict of properties and their values.

`name`:
the human readable name of the collection.

`lang`:
the language of the collection; valid values are listed in the xapian stemmer documentation and are usually either the English name or the two letter ISO639 code of a language.

`entry_label`:
a jinja2 template used to show an entry in the interface; beside the entry fields two useful variables are `eid` for the full entry ID and `short_id` for the short version.

`default_sort`:
a list of field names (possibly prefixed by + or -) that are used by default to sort results of searches in the collection. The fields must be marked as sortable in their definition, see below.

`search_aliases`:
a dict of <name>: <search snippet> which can be used in a query template. For more details see *Search templates and search_aliases*

`fields`:
The list of fields used by the collection, as described below.

Field definitions

`name`:
a name for the field (computer readable: keeping it lowercase alphabetic ascii is probably safer).

`type`:
the type of the field: valid fields are listed in *lesana.types module* (see the `name` property for each field)

`index`:
whether this field should be indexed: valid values are `free` for fields that should be available in the free text search and `field` for fields that should only be available by specifying the field name in the search.

sortable:
boolean; whether this field is sortable. Sortable fields enable sorting the results and search by ranges, but having too many sortable fields make the search more resurce intensive.

help:
a description for the field; this is e.g. added to new entries as a comment.

default:
the default value to use when creating an entry.

prefix:
the optional term prefix used inside xapian: if you don't know what this means you should avoid using this, otherwise see Term Prefixes on the xapian documentation for details.

Some field types may add other custom properties.

`list` properties

list:
the type of the entries in the list; note that neither lists of non uniform values nor lists of lists are supported (if you need those you can use the `yaml` generic type, or write your own derivative with an additional type).

`integer` properties

auto:
automatic manipulation of the field contents.

The value of `increment` will autoincrement the value at every update.

The reference command-line client will run this update before editing an entry, allowing further changes from the user; a command line user can then decide to abort this change through the usual git commands.

Other clients may decide to use a different workflow.

increment:
the amount by which an `auto: increment` field is incremented (negative values are of course allowed). Default is 1.

`decimal` properties

precision:
if this property is set, every value in this field will get rounded to the given number of decimals.

With this property it is possible to store decimal values as YAML floats instead of strings.

`date` and `datetime` properties

`auto`:
: automatic manipulation of the field contents.

  The following values are supported.

  `creation`
  : autofill the field at creation time with the current UTC time (`datetime`) or local zone day (`date`).

  `update`
  : autofill the field when it is updated with the current UTC time (`datetime`) or local zone day (`date`).

    The reference command line client will run this update before editing an entry, allowing further changes from the user; a command line user can then decide to abort this change through the usual git commands.

    Other clients may decide to use a different workflow.

`values`

The `string`, `text`, `list` and numeric types can have a property `value` with a list of valid values for that field.

An empty value is always allowed.

For the `list` type, each element of the list is checked, not the whole list.

5.1.3 Search syntax

Searches in lesana use the human readable query string format defined by xapian.

The simplest search is just a list of terms: e.g. searching for `thing object` will find entries where either `thing` or `object` is present in one of the fields with `free` indexing.

It is also possible to specify that a term must be in one specific field: the syntax for this is the name of the field follwed by `:` and the term, e.g. `name:object` will search for entries with the term `object` in the `name` field.

Search queries can of course include the usual logical operators `AND`, `OR` and `NOT`.

More modifiers are available; see the Query Parser documentation from xapian for details.

Search templates and `search_aliases`

In some contexts, search queries are rendered as jinja2 templates with the contents of the `search_aliases` property as set in `settings.yaml`.

The values of those search aliases should be valid search snippets with the syntax documented above; it's usually a good idea to wrap them in parenthesis, so that they are easier to use in complex queries; e.g.:

```
my_alias: '(name:object OR name:thing)'
```

can correctly be used in a query like:

```
{{ my_alias }} AND description:shiny
```

5.1.4 Moving Data between Collections

Entries can be exported from a lesana collection to another using the `lesana export` command and a jinja2 template.

The template should generate a yaml file that is a valid lesana entry for the destination collection and it can use the fields of the starting collection as variables. The variable `entry` is also available and gives complete access to the entry of the original collection, so fields with names that aren't valid jinja templates can be accessed as `entry.data[<field-name>]`.

E.g. to convert between a collection with fields `name`, `short-desc`, `desc` to a collection with fields `name`, `description` one could use the following template:

```
name: {{ name }}
description: |
    {{ entry.data.[short-desc] }}

    {{ desc | indent(width=4, first=False) }}
```

From the origin collection you can then run the command:

```
lesana export <path/to/destination/collection> <path/to/template>
```

to export all entries.

You can also export just a subset of entries by adding a xapian query with the parameter `--query`; you can test the search using:

```
lesana search --all <some search terms>
```

and then actually run the export with:

```
lesana search --query '<some search terms>' <destination> <template>
```

note that in this second command the spaces in the search query have to be protected from the shell.

5.1.5 lesana derivatives

Front-ends

Collector

Collector is a Gtk3 app to manage collection inventories throught yaml files, which also works on GNU/Linux mobile devices.

linkopedia

linkopedia is a read-only web interface to Lesana collections.

5.2 Developer Documentation

Documentation that is useful for developers who are using lesana as a library.

5.2.1 Promises

Semantic versioning

This project uses semver.

Collection format stability

Future versions of lesana will be able to read collections written by older versions.

Older versions in the same mayor release will also be able to work concurrently on the same repository.

If in the future a change of formats will be required, conversions scripts will be written in a way that will make them as stable as possibile, and will have enought test data to keep them maintained for the time being.

Disposable cache

Contrary to the yaml files, the xapian cache is considered disposable: from time to time there may be a need to delete the cache and reindex everything, either because of an upgrade or to perform repository mainteinance.

Of course, effort will be made to reduce the need for this so that it only happens sporadically, but it will probably never completely disappear.

5.3 Contributor Documentation

Documentation that is useful for contributors of lesana.

5.3.1 Release procedure

- Check that the version number in setup.py and in docs/source/conf.py is correct.
- Check that the changelog is up to date.
- Tag the version you are preparing:

```
$ git tag -s v$VERSION
$ git push
$ git push --tags
```

for the tag content use something like:

```
Version $VERSION

* contents
* of the relevant
* changelog
```

- Generate the distribution files:

```
$ python3 -m build
```

- Upload

```
$ twine upload -s dist/*
```

- Send the release announce to:

```
valhalla/lesana-announce@lists.sr.ht, ~valhalla/lesana-discuss@lists.sr.ht
```

- Close the bugs marked as pending_release on https://todo.sr.ht/~valhalla/lesana.

5.4 Man Pages

5.4.1 lesana-edit

SYNOPSIS

lesana edit [–help] [–collection <collection>] [–no-git] <entry>

DESCRIPTION

Lesana edit will open an existing entry (specified by id or partial id) in an editor, so that it can be changed.

If the collection is configured to use git, after the editor has been closed, it will add the file to the git staging area, unless `--no-git` is given.

OPTIONS

-h, --help Prints an help message and exits.

--collection COLLECTION, -c COLLECTION The collection to work on. Default is `.`

--no-git Don't add the new entry to git.

5.4.2 lesana-export

SYNOPSIS

lesana export [-h] [–collection COLLECTION] [–query QUERY]
destination template

DESCRIPTION

Lesana export converts entries from one lesana collection to another, using a jinja2 template.

OPTIONS

-h, --help Prints an help message and exits.

--collection COLLECTION, -c COLLECTION The collection to work on. Default is `.`

--query QUERY, -q QUERY Xapian query to search in the collection

5.4.3 lesana-index

SYNOPSIS

lesana index [–help] [–collection COLLECTION] [files [files ...]]

DESCRIPTION

Lesana index adds some entries to the xapian cache, listed by filename (by default all of the files found in the items directory).

OPTIONS

-h, --help Prints an help message and exits.

--collection COLLECTION, -c COLLECTION The collection to work on. Default is `.`

--reset Delete the existing xapian cache before indexing.

5.4.4 lesana-init

SYNOPSIS

lesana init [–help] [–collection <collection>] [–no-git]

DESCRIPTION

lesana init initializes a new lesana collection.

It will create the directory (if it does not exist) and, unless `--no-git` is specified it will initialize it as a git repository, create a `.gitignore` file with some relevant contents and add hooks to update the local cache when the files are changed via git.

It will then create a skeleton `settings.yaml` file and open it in an editor to start configuring the collection.

When leaving the editor, again unless `--no-git` is used, it will add this file to the git staging area, but not commit it.

It is safe to run this command on an existing repository, e.g. to install the hooks on a new clone, but it will overwrite the hooks themselves even if they have been changed by the user.

OPTIONS

--help, -h Prints an help message and exits.

--collection COLLECTION, -c COLLECTION The directory where the collection will be initialized. Default is .

--no-git Do not use git in the current collection.

5.4.5 lesana-new

SYNOPSIS

lesana new [–help] [–collection <collection>] [–no-git]

DESCRIPTION

Lesana new creates a new lesana entry.

It will create an empty entry and open an editor so that it can be filled.

If the collection is configured to use git, after the editor has been closed, it will add the file to the git staging area, unless `--no-git` is given.

OPTIONS

-h, --help Prints an help message and exits.

--collection COLLECTION, -c COLLECTION The collection to work on. Default is `.`

--no-git Don't add the new entry to git.

5.4.6 lesana-rm

SYNOPSIS

lesana rm [-h] [–collection COLLECTION] entries [entries . . .]

DESCRIPTION

Lesana rm removes an entry from the collection, removing both the file and the cached entry.

OPTIONS

-h, --help Prints an help message and exits.

--collection COLLECTION, -c COLLECTION The collection to work on. Default is `.`

5.4.7 lesana

SYNOPSIS

lesana [–help] <command>

DESCRIPTION

lesana is a tool to organize collections of various kinds. It is designed to have a data storage / serialization format that is friendly to git and other VCSs, but decent performances.

To reach this aim it uses yaml as its serialization format, which is easy to store in a VCS, share between people and synchronize between different computers, but it also keeps an index of this data in a local xapian database in order to allow for fast searches.

lesana supports collections of any kind, as long as their entries can be described with a mostly flat dictionary of fields of the types described in the documentation file `field_types`.

Some example collection schemas are provided, but one big strength of lesana is the ability to customize your collection with custom fields either by simply writing a personalized `settings.yaml`.

OPTIONS

-h, --help Prints an help message and exits.

COMMANDS

new(1)
Creates a new entry.

edit(1)
Edits an existing entry.

show(1)
Shows an existing entry.

index(1)
Index some entries in the xapian cache.

search(1)
Searches for entries in the xapian cache.

export(1)
Exports entries from one lesana collection to another

init(1)
Initialize a new lesana collection

rm(1)
Removes an entry.

TEXT EDITOR

Many lesana subcommands will try to open files in a text editor chosen as follows:

- first, the value of $EDITOR is tried
- then the command `sensible-editor`, as available under e.g. Debian and its derivatives
- lastly, it will try to fallback to `vi`, which should be available under any posix system.

5.4.8 lesana-search

SYNOPSIS

lesana search [–help] [–collection COLLECTION] [–template TEMPLATE]
[–offset OFFSET] [–pagesize PAGESIZE] [–all] [–expand-query-template] [–sort FIELD1 [–sort FIELD2 . . .]] [query [query . . .]]

DESCRIPTION

Lesana search allows one to make searches in the collection and render the results.

The section *Search syntax* in the full documentation describes the query syntax in more detail; it is available online at https://lesana.trueelena.org/user/search.html or it may be installed on your system (e.g. in Debian and derivatives it will be at `/usr/share/doc/lesana/html/user/search.html`).

By default entries are printed according to the `entry_label` from the `settings.yaml` file, but they can be rendered according to a jinja2 template.

If no query is specified, it will default to `'*'`, i.e. search all entries: thus `lesana search --all` will print all entries, while just `lesana search` will print the first 12 entries, possibly according to the relevant sorting options.

OPTIONS

-h, --help Prints an help message and exits.

--collection COLLECTION, -c COLLECTION The collection to work on. Default is `.`

--template TEMPLATE, -t TEMPLATE Template to use when displaying results

--offset OFFSET .

--pagesize PAGESIZE .

--all Return all available results

--sort Sort the results by a sortable field.

This option can be added multiple times; prefix the name of the field with `-` to reverse the results (e.g. `--sort='-date'`).

--expand-query-template Render search_aliases in the query as a jinja2 template

5.4.9 lesana-get-values

SYNOPSIS

lesana search [–help] [–collection COLLECTION] [–template TEMPLATE]
–field FIELD [query [query . . .]]

DESCRIPTION

Lesana get-values will list all values found in a field and the number of entries where that value has been found.

A template can be specified to format the results.

Extracting the values from a sortable field is significantly more efficient than doing so from a non-sortable field, but adding too many sortable fields can make general searches and indexing slower.

OPTIONS

-h, --help Prints an help message and exits.

--collection COLLECTION, -c COLLECTION The collection to work on. Default is `.`

--template TEMPLATE, -t TEMPLATE Template to use when displaying results

--field Name of the desired field.

5.4.10 lesana-show

SYNOPSIS

lesana show [–help] [–collection COLLECTION] [–template TEMPLATE] <entry>

DESCRIPTION

`lesana show` will print an entry (specified by id or partial id) to stdout.

A template can be specified with `--template <template>` to pretty print entries.

OPTIONS

-h, --help Prints an help message and exits.

--collection COLLECTION, -c COLLECTION The collection to work on. Default is `.`

--template TEMPLATE, -t TEMPLATE Use the specified template to display results.

TEMPLATES

The templates used by `lesana show` are jinja2 templates.

The entry fields are available as variables, and the full entry is available as the variable `entry` and can be used to give access to fields with names that aren't valid jinja2 variables e.g. as `entry.data[<field-name>]`.

5.5 lesana reference

5.5.1 lesana package

Subpackages

lesana.data package

Submodules

lesana.collection module

class lesana.collection.**Collection**(*directory=None*, *itemdir='items'*)

Bases: `object`

PARSER_FLAGS = 23

entries_from_short_eid(*seid*)

entry_from_eid(*eid*)

entry_from_rendered_template(*template*, *data*)

get_all_documents()

Yield all documents in the collection.

Note that the results can't be sorted, even if the collection has a default_sort; if you need sorted values you need to use a regular search with a query of '*'

get_all_search_results()

get_field_values(*field*, *querystring='*'*)

get_search_results(*offset=0*, *pagesize=12*)

get_template(*template_fname*, *searchpath='.'*)

git_add_files(*files=[]*)

property indexed_fields

classmethod init(*directory=None*, *git_enabled=True*, *edit_file=None*, *settings={}*)

Initialize a lesana repository

directory defaults to . if git_enabled is True, git support is enabled and if possible a git repository is initalized. edit_file is a syncronous function that runs on a filename (possibly opening the file in an editor) and should manage its own errors.

remove_entries(*eids*)

remove_file(*fname*)

render_query_template(*query*)

Render a query template, filling it with search_aliases.

save_entries(*entries=[]*)

start_search(*querystring*, *sort_by=None*)

Prepare a search for querystring.

update_cache(*fnames=None*, *reset=False*)

Update the xapian db with the data in files.

`fnames` is a list of *basenames* of files in `self.itemdir`.

If no files have been passed, add everything.

if `reset` the existing xapian db is deleted before indexing

Return the number of files that have been added to the cache.

update_field(*query*, *field*, *value*)

class lesana.collection.**Entry**(*collection*, *data={}*, *fname=None*)

Bases: `object`

auto()

Update all fields of this entry, as required by the field settings.

This is called by the reference client before an edit, so that the user can make further changes.

Note that the stored file is not changed: if you need it you need to save the entry yourself.

empty_data()

get_data()

property idterm

render(*template*, *searchpath='.'*)

property short_id

validate()

property yaml_data

exception lesana.collection.**TemplatingError**

Bases: `Exception`

Raised when there are errors rendering a jinja template

lesana.command module

class lesana.command.**Command**(*collection_class=<class 'lesana.collection.Collection'>*, *entry_class=<class 'lesana.collection.Entry'>*)

Bases: Command

add_arguments(*parser: ArgumentParser*)

Add argparse arguments to an existing parser.

Override this method to add arguments to a subcommand.

class lesana.command.**Edit**(*collection_class=<class 'lesana.collection.Collection'>*, *entry_class=<class 'lesana.collection.Entry'>*)

Bases: *Command*, ExternalEditorMixin

Edit a lesana entry

arguments = [(['--collection', '-c'], {'help': 'The collection to work on (default .)'}), (['--no-git'], {'help': "Don't add the new entry to git", 'action': 'store_false', 'dest': 'git'}), (['eid'], {'help': 'eid of an entry to edit'})]

main()

Main code of this subcommand.

Override this method to implement the actual program.

class lesana.command.**Export**(*collection_class=<class 'lesana.collection.Collection'>*, *entry_class=<class 'lesana.collection.Entry'>*)

Bases: *Command*

Export entries to a different collection

arguments = [(['--collection', '-c'], {'help': 'The collection to work on (default .)'}), (['--query', '-q'], {'help': 'Xapian query to search in the collection'}), (['destination'], {'help': 'The collection to export entries to'}), (['template'], {'help': 'Template to convert entries'})]

main()

Main code of this subcommand.

Override this method to implement the actual program.

class lesana.command.**GetValues**(*collection_class=<class 'lesana.collection.Collection'>*, *entry_class=<class 'lesana.collection.Entry'>*)

Bases: *Command*

List all values for one field, with entry counts.

arguments = [(['--collection', '-c'], {'help': 'The collection to work on (default .)'}), (['--field', '-f'], {'help': 'Name of the field', 'required': True}), (['--template', '-t'], {'help': 'Template to use when displaying results'}), (['query'], {'help': 'Xapian query to limit the count search " + "in the collection', 'nargs': '*', 'default': '*'})]

main()

Main code of this subcommand.

Override this method to implement the actual program.

name: Optional[str] = 'get-values'

The name used to call this subcommand from the command line.

If this property is none, the default is the name of the class set to lowercase.

class lesana.command.**Index**(*collection_class=<class 'lesana.collection.Collection'>*, *entry_class=<class 'lesana.collection.Entry'>*)

Bases: *Command*

Index entries in a lesana collection

arguments = [(['--collection', '-c'], {'help': 'The collection to work on (default .)'}), (['--reset'], {'action': 'store_true', 'help': 'Delete the existing index and reindex from scratch.'}), (['files'], {'help': 'List of files to index (default: everything)', 'default': None, 'nargs': '*'})]

main()

Main code of this subcommand.

Override this method to implement the actual program.

class lesana.command.**Init**(*collection_class=<class 'lesana.collection.Collection'>*, *entry_class=<class 'lesana.collection.Entry'>*)

Bases: *Command*, ExternalEditorMixin

Initialize a lesana collection

arguments = [(['--collection', '-c'], {'help': 'The directory to work on (default .)', 'default': '.'}), (['--no-git'], {'help': 'Skip setting up git in this directory', 'action': 'store_false', 'dest': 'git'})]

main()

Main code of this subcommand.

Override this method to implement the actual program.

class lesana.command.**Lesana**

Bases: MainCommand

Manage collections

commands: Iterable['Command'] = (<lesana.command.New object>, <lesana.command.Edit object>, <lesana.command.Show object>, <lesana.command.Index object>, <lesana.command.Search object>, <lesana.command.GetValues object>, <lesana.command.Update object>, <lesana.command.Export object>, <lesana.command.Init object>, <lesana.command.Remove object>)

The subcommands: a tuple of *Command* subclasses.

class lesana.command.**New**(*collection_class=<class 'lesana.collection.Collection'>*, *entry_class=<class 'lesana.collection.Entry'>*)

Bases: *Command*, ExternalEditorMixin

Create a new entry

arguments = [(['--collection', '-c'], {'help': 'The collection to work on (default .)'}), (['--no-git'], {'help': "Don't add the new entry to git", 'action': 'store_false', 'dest': 'git'})]

main()

Main code of this subcommand.

Override this method to implement the actual program.

class lesana.command.**Remove**(*collection_class=<class 'lesana.collection.Collection'>*, *entry_class=<class 'lesana.collection.Entry'>*)

Bases: *Command*

Remove an entry from a collection

arguments = [(['--collection', '-c'], {'help': 'The collection to work on (default .)'}), (['entries'], {'help': 'List of entries to remove', 'nargs': '+'})]

main()

Main code of this subcommand.

Override this method to implement the actual program.

name: Optional[str] = 'rm'

The name used to call this subcommand from the command line.

If this property is none, the default is the name of the class set to lowercase.

class lesana.command.**Search**(*collection_class=<class 'lesana.collection.Collection'>*, *entry_class=<class 'lesana.collection.Entry'>*)

Bases: *Command*

Search for entries

arguments = [(['--collection', '-c'], {'help': 'The collection to work on (default .)'}), (['--template', '-t'], {'help': 'Template to use when displaying results'}), (['--offset'], {'type': <class 'int'>}), (['--pagesize'], {'type': <class 'int'>}), (['--all'], {'action': 'store_true', 'help': 'Return all available results'}), (['--sort'], {'action': 'append', 'help': 'Sort results by a sortable field'}), (['--expand-query-template', '-e'], {'action': 'store_true', 'help': 'Render search_aliases in the query as a jinja2 template'}), (['query'], {'help': 'Xapian query to search in the collection', 'nargs': '*', 'default': '*'})]

main()

Main code of this subcommand.

Override this method to implement the actual program.

class lesana.command.**Show**(*collection_class=<class 'lesana.collection.Collection'>*, *entry_class=<class 'lesana.collection.Entry'>*)

Bases: *Command*

Show a lesana entry

arguments = [(['--collection', '-c'], {'help': 'The collection to work on (default .)'}), (['--template', '-t'], {'help': 'Use the specified template to display results.'}), (['eid'], {'help': 'eid of an entry to edit'})]

main()

Main code of this subcommand.

Override this method to implement the actual program.

class lesana.command.**Update**(*collection_class=<class 'lesana.collection.Collection'>*, *entry_class=<class 'lesana.collection.Entry'>*)

Bases: *Command*

Update a field in multiple entries

arguments = [(['--collection', '-c'], {'help': 'The collection to work on (default .)'}), (['--field', '-f'], {'help': 'The field to change'}), (['--value', '-t'], {'help': 'The value to set'}), (['query'], {'help': 'Xapian query to search in the collection', 'nargs': '+'})]

main()

Main code of this subcommand.

Override this method to implement the actual program.

lesana.command.**main**()

lesana.templating module

Custom jinja2 filters and other templating helpers

class lesana.templating.**Environment**(**args*, ***kwargs*)

Bases: Environment

A customized jinja2 environment that includes our filters.

lesana.templating.**to_yaml**(*data*)

Return the yaml representation of data.

lesana.types module

Type checkers for lesana fields.

Warning: this part of the code is still in flux and it may change significantly in a future release.

class lesana.types.**LesanaBoolean**(*field*, *types*, *value_index=None*)

Bases: *LesanaType*

A boolean value

empty()

load(*data*)

name = 'boolean'

class lesana.types.**LesanaDate**(*field*, *types*, *value_index=None*)

Bases: *LesanaType*

A date

auto(*value*)

Return an updated value.

If the field settings auto is update return the current date, otherwise the old value.

empty()

load(*data*)

name = 'date'

class lesana.types.**LesanaDatetime**(*field*, *types*, *value_index=None*)

Bases: *LesanaType*

A datetime

auto(*value*)

Return an updated value.

If the field settings `auto` is `update` return the current datetime, otherwise the old value.

empty()

load(*data*)

name = 'datetime'

class lesana.types.**LesanaDecimal**(*field*, *types*, *value_index=None*)

Bases: *LesanaType*

A fixed point number

Because of a limitation of the yaml format, these should be stored quoted as a string, to avoid being loaded back as floats.

Alternatively, the property `precision` can be used to force all values to be rounded to that number of decimals.

empty()

load(*data*)

name = 'decimal'

class lesana.types.**LesanaFile**(*field*, *types*, *value_index=None*)

Bases: *LesanaString*

A path to a local file.

Relative paths are assumed to be relative to the base lesana directory (i.e. where .lesana lives)

name = 'file'

class lesana.types.**LesanaFloat**(*field*, *types*, *value_index=None*)

Bases: *LesanaType*

A floating point number

empty()

load(*data*)

name = 'float'

class lesana.types.**LesanaGeo**(*field*, *types*, *value_index=None*)

Bases: *LesanaString*

A Geo URI

load(*data*)

name = 'geo'

class lesana.types.**LesanaInt**(*field*, *types*, *value_index=None*)

Bases: *LesanaType*

An integer number

auto(*value*)

Return an updated value.

If the field settings `auto` is `increment` return the value incremented by the value of the field setting `increment` (default 1).

empty()

load(*data*)

name = 'integer'

class lesana.types.**LesanaList**(*field*, *types*, *value_index=None*)

Bases: *LesanaType*

A list of other values

empty()

index(*doc*, *indexer*, *value*)

Index a value for this field type.

Override this for types that need any kind of special treatment to be indexed.

See LesanaList for an idea on how to do so.

load(*data*)

name = 'list'

class lesana.types.**LesanaString**(*field*, *types*, *value_index=None*)

Bases: *LesanaType*

A string of unicode text

empty()

load(*data*)

name = 'string'

class lesana.types.**LesanaText**(*field*, *types*, *value_index=None*)

Bases: *LesanaString*

A longer block of unicode text

name = 'text'

class lesana.types.**LesanaTimestamp**(*field*, *types*, *value_index=None*)

Bases: *LesanaType*

A unix timestamp, assumed to be UTC

empty()

load(*data*)

name = 'timestamp'

class lesana.types.**LesanaType**(*field*, *types*, *value_index=None*)

Bases: object

Base class for lesana field types.

allowed_value(*value*)

Check whether a value is allowed in this field.

Return the value itself or raise LesanaValueError if the value isn't valid.

auto(*value*)

Return an updated value, as appropriate for the field.

Default is to return the value itself, but types can use their configuration to e.g. increment a numerical value or return the current date(time).

empty()

index(*doc*, *indexer*, *value*)

Index a value for this field type.

Override this for types that need any kind of special treatment to be indexed.

See LesanaList for an idea on how to do so.

load(*data*)

class lesana.types.**LesanaURL**(*field*, *types*, *value_index=None*)

Bases: *LesanaString*

An URL

name = 'url'

exception lesana.types.**LesanaValueError**

Bases: ValueError

Raised in case of validation errors.

class lesana.types.**LesanaYAML**(*field*, *types*, *value_index=None*)

Bases: *LesanaType*

Free YAML contents (no structure is enforced)

empty()

load(*data*)

name = 'yaml'

CHAPTER

SIX

INDICES AND TABLES

- genindex
- modindex
- search

PYTHON MODULE INDEX

INDEX

M

N

P

R

S

T

U

V

Y