maggit Documentation

Release 0.1

Matthieu Gautier

December 10, 2015

Contents

Contents:

CHAPTER 1

Installing

1.1 Installing with pypi

Maggit is available on pypi. Just use it

```
$ pip install maggit
```

1.2 Installing from source

You can directly download sources and install from them

```
$ git clone https://gitlab.com/maggit/maggit.git
$ cd maggit
$ python3 setup.py install .
```

To test that everything is ok

```
$ pip install pytest
$ py.test
```

You are now ready to use Maggit. Read the Tutorial if you don't know how.

CHAPTER 2

Tutorial

2.1 Importing Maggit

This is pretty simple

```
import maggit
```

You can use the star import if you want

```
from maggit import *
```

For the rest of the tutorial, we'll assume that you import maggit and do not use the star import.

2.2 Get a repository

A *Repo* is one the central objects when interactig with maggit:

```
# Create a repository
repo = maggit.Repo()
```

By default, Maggit will look for a git repository in your current directory. If you want to explore a repository elsewhere, specify it:

```
repo = maggit.Repo("a/git/repository")
```

You don't have to specify a the root directory of the repository. If the directory you specify is not a git repository, maggit will look up in the parents directories.

2.3 References

Once we've got a repository, the first thing we may want to do is list it's references (branches, tags):

```
branches = repo.branches
print(repo.branches.keys())
```

repo.branches is a dictionnary mapping all branches of the repository

If we want manipulate the *master* branch, lets get it:

```
master = branches['master']
```

In the same way *repo.tags* is a dictionnary mapping all tags of the repository

All the references we get, whatever they are branches, lightweigh tag or real tag objects share a common api. The most interesting here, is to get the commit object pointed to by the ref:

```
commit = master.commit
```

2.4 Print the log of the 10th last commit of the master branch

We just have to go through the parent of the commit and print the message each time

```
commit = master.commit
for i in range(10):
    print(commit.message)
    commit = commit.parents[0]
```

2.5 Print the content of a file

There are several way of doing it.

The low level way is the following:

```
current = commit.tree
for part in path.split('/'):
    current = current.entries[part]
blob = current
print(blob.content)
```

But you can also use a high level api

```
entry = commit.get_file(path)
# the entry is a intermediate object making the link between a gitobject
# and a particular commit
blob = entry.gitObject
print(blob.content)
```

2.6 The commit object

Other important object in Maggit is naturally the git objects. A git object firstly allow us to access to the content of the object itself.

The commit object is one of them. Naturally it allows us to access the commit attributes:

```
commit.message
commit.first_line #The first line of the message
commit.author
commit.committer
commit.parents # The parents of the commit
commit.tree # The tree object
```

CHAPTER 3

maggit package api

3.1 maggit.db package

This module contains all the low level classs to handle a git repository.

Class defined in this module use io functions and provide a more consistant API to access content of the git repository.

Most users will not use those object directly but better use the high level API provided by the maggit package itself.

3.1.1 maggit.db.db module

class maggit.db.db.**Gitdb**(*rootdir*)

Bases: object

The Gitdb, unify all loose/pack io function to provide a coherent access to content of a git repository.

A Gitdb handle only the reading of git objects. Not the references, remotes, ...

Parameters **rootdir** (*path*) – The path of the objects directory (.git/objects)

blob_content(*sha*)

Read and parse a object assuming it is a blob.

Parameters **sha** – The sha of the object.

Returns The content of the blob (bytes).

Raises ValueError – If the object is not a blob.

blob_write(*content*)

commit_content(*sha*)

Read and parse a object assuming it is a commit.

Parameters **sha** – The sha of the object.

Returns

A tuple (tree, parents, message, author, committer) where:

- tree is the sha of the tree object of the commit(unhexlified bytes).
- parents is a list of sha of the parents commit.
- message is the message of the commit.
- author is the name (b'name <email> timestamp') of the author.

- author is the name (b'name <email> timestamp') of the committer.

Return type bytes, list[bytes], bytes, bytes, bytes

Raises `ValueError` – If the object is not a commit.

commit_write(*treesha*, *message*, *author*, *authorDate*, *committer*, *committerDate*, *parents=[]*)

gen_pack_list()

get_full_sha(*prefix*)

Return the full Sha of the prefix

Parameters **prefix** (*bytes*) – The beginning of a sha.

Returns The corresponding (bytes).

Exemples:

```
>>> repo.get_full_sha(b'bf09f0a9')
<Sha b'bf09f0a9...'>
```

Raises `:Exception` – If number of object corresponding to prefix is not equal to one.

get_pack(*sha*)

Get a pack containing the sha

Parameters **sha** – The sha of the object

Returns class:*~maggit.io.pack.GitPack* containing the sha

Return type The

object_content(*sha*)

object_exists(*sha*)

Return True if the sha exists in the db

object_type(*sha*)

Return the type of the object associated to sha.

Parameters **sha** – The sha of the object.

Returns The type of the object.

tag_content(*sha*)

Read and parse a object assuming it is a tag.

Parameters **sha** – The sha of the object.

Returns

A tuple (object, objecttype, tag, tagger, message) where:

- object is the sha of the tagged object (unhexlified bytes).
- objecttype is the type of the tagged object.
- tag is the name of the tag.
- tagger is the name (b'name <email> timestamp') of the tagger.
- message is the message of the tag.

Return type bytes, bytes, bytes, bytes, bytes

Raises `ValueError` – If the object is not a tag.

tag_write(*objectsha*, *tag*, *tagger*, *tagDate*, *message*)

tree_content(*sha*)
Read and parse a object assuming it is a tree.

Parameters **sha** – The sha of the object.

Returns

A list of (path, (mode, sha)) where :

- path is the name of the entry.
- mode is the git mode.
- sha is the sha of the blob/tree object.

Return type List[Tuple[bytes, Tuple[bytes, bytes]]]

Raises `ValueError` – If the object is not a tree.

tree_write(*entries*)

3.1.2 maggit.db.repo module

class `maggit.db.repo.`**Repo**(*gitdir=None*, *disable_directoryLooking=False*, *bare=False*)
This is the low level Repository class.

The repo make the link between all other low level subsystems and recreate a coherent database.

Parameters

- **gitdir** (*path*) – The directory path of the repository,

  If is None, the current working directory is assumed.
- **disable_directoryLooking** (*bool*) – If True, assume that the gitdir is a valid path so do not search for a valid git repository in parents and gitdir must not be None.
- **bare** (*bool*) – Does the repository is a bare one. Only relevant if disable_directoryLooking is True. Else, the bare attribute is detected from the git repository structure.

HEAD
The current checkouted branch

branches
A iterator of branches in the repo

check_ref_exists(*ref*)
Check that a ref exist in the git bdd.

Parameters **ref** (*str*) – the ref to check.

Returns True if the ref exists, else False

Return type bool

get_full_sha(*value*)
Return the full sha of the value

classmethod **get_git_dir**(*dirToCheck=None*)

get_peel_ref_sha(*ref*)

get_ref_sha(*ref*)
Return the sha corresponding to the ref.

Parameters **ref** (*str*) – the ref to get.

Returns The sha corresponding to the ref

Return type bytes

classmethod **init_repo**(*gitdir*, *bare=False*)
Instantiate a new git repo at the given location.

tags
A iterator of tags in the repo

3.2 maggit.io package

This module contains all the low level functions necessary to Maggit to read, write and parse all git files.

Thoses files includes git objects, pack, packedrefs, config, index, ...

Thoses function do not provide high level API. There are mainly intended to be use by Maggit itself.

3.2.1 maggit.io.loose module

maggit.io.loose.**commit_parse**(*content*)
Parse a commit content.

Parameters **content** (*bytes*) – The content to parse (without header).

Returns

A tuple (tree, parents, message, author, committer) where:

- tree is the sha of the tree object of the commit(unhexlified bytes).
- parents is a list of sha of the parents commit.
- message is the message of the commit.
- author is the name (b'name <email> timestamp') of the author.
- author is the name (b'name <email> timestamp') of the committer.

Return type bytes, list[bytes], bytes, bytes

maggit.io.loose.**object_content**(*filepath*)
Return the content of a loose object.

Parameters **filepath** – The path of the loose object.

Returns

A tuple (type, content) where:

- type is the type of the object.
- content is the content of the object (without header).

Return type bytes, bytes

maggit.io.loose.**object_rawsha**(*type_*, *content*)
Generate the raw content and the sha of a object content.

Parameters

- **type** (*bytes*) – The type of the object.

- **content** (*bytes*) – The content of the object (without header).

Returns

A tuple (raw, sha) where:

- raw is the full content of the object (with header).
- sha is the sha of the object.

Return type bytes, bytes

maggit.io.loose.**object_sha**(*type_*, *content*)
Generate the sha of a object content.

Is the bit more performant than object_rawsha(...)[1] as the raw content is not generated.

Parameters

- **type** (*bytes*) – The type of the object.
- **content** (*bytes*) – The content of the object (without header).

Returns The sha of the object.

maggit.io.loose.**object_sha_from_raw**(*raw*)
Generate the sha of a object from its content.

Parameters **raw** (*bytes*) – The content of the object (with header)

Returns The sha of the object.

maggit.io.loose.**object_write**(*filepath*, *content*, *compress_level=1*)
Correctly create the loose object file.

Parameters

- **filepath** – The path of the loose object to write.
- **content** (*bytes*) – The full content (with header) to write.
- **compress_level** (*int*) – The compression level to use (default=1).

maggit.io.loose.**tag_parse**(*content*)
Parse a tag content.

Parameters **content** (*bytes*) – The content to parse (without header).

Returns

A tuple (object, objecttype, tag, tagger, message) where:

- object is the sha of the tagged object (unhexlified bytes).
- objecttype is the type of the tagged object.
- tag is the name of the tag.
- tagger is the name (b'name <email> timestamp') of the tagger.
- message is the message of the tag.

Return type bytes, bytes, bytes, bytes, bytes

maggit.io.loose.**tree_parse**(*content*)
Parse a tree content.

Parameters **content** (*bytes*) – The content to parse (without header).

Returns

A list of (path, (mode, sha)) where :

- path is the name of the entry.
- mode is the git mode.
- sha is the sha of the blob/tree object.

Return type List[Tuple[bytes, Tuple[bytes, bytes]]]

3.2.2 maggit.io.pack module

class maggit.io.pack.**GitPack**(*packfile*, *idxfile*)

A git pack file.

Parameters

- **packfile** (*path*) – The path of the packfile.
- **idxfile** (*GitPackIndex*) – The index associated to the pack.

read_object(*offset*)

Return the content of a object at a offset.

Parameters **offset** – The offset to read from.

Returns

A tuple (type, content) where:

- type is the type of the object.
- content is the content of the object (without header).

Return type bytes, bytes

class maggit.io.pack.**GitPackIndex**(*indexfile*)

A pack index

Parameters **indexfile** (*path*) – The path of the index file.

get_offset(*sha*)

Return the offset in the pack associated to the sha.

3.2.3 maggit.io.packedref module

maggit.io.packedref.**packed_ref_parse**(*filename*)

Parse a packed ref file.

Parameters **filename** (*path*) – The path of the packed ref.

Returns

A list of (ref, sha, peeledsha) where:

- ref is name of the ref
- sha is the associated sha
- peeledsha is the peeled sha associated to the ref if present. Else None.

Return type List[Tuple[bytes, bytes, bytes]]

class maggit.**Sha**
Bases: bytes

A Sha is a git sha. It means that it is the identifier of a git object.

Sha are store in Maggit as a 20 bytes len.

hexbytes
The sha as a hexlified bytes

hexstr
The sha as a hexlified str

class maggit.**Repo**(*gitdir=None*, *disable_directoryLooking=False*, *bare=False*)
Bases: *maggit.db.repo.Repo*

This is the central piece of a git repository.

HEAD
The current checkouted branch

branches
A dict of branches in the repo

check_ref_exists(*ref*)
Check that a ref exist in the git bdd.

Parameters **ref** (*str*) – the ref to check.

Returns True if the ref exists, else False

Return type bool

get_blob(*sha*)
Get a blob object for the sha.

Parameters **sha** (*Sha*) – The sha of the blob.

Returns class:*~maggit.git_objects.Blob* object for this sha.

Return type A

Raises **:**Exception – If sha doesn't name a blob object.

get_commit(*sha*)
Get a commit object for the sha.

Parameters **sha** (*Sha*) – The sha of the commit.

Returns class:*~maggit.repo.Commit* object for this sha.

Return type A

Raises **:**Exception – If sha doesn't name a commit object.

get_full_sha(*prefix*)
Return the full Sha of the prefix

Parameters **prefix** (*bytes*) – The beginning of a sha.

Returns class:*~maggit.Sha* corresponding.

Return type The

Exemples:

```
>>> repo.get_full_sha(b'bf09f0a9')
<Sha b'bf09f0a9...'>
```

Raises : Exception – If number of object corresponding to prefix is not equal to one.

get_git_dir(*dirToCheck=None*)

get_object(*sha*, *type_=None*)
Get a git object for the sha.

Parameters

- **sha** (*Sha*) – The sha of the object.
- **type** (*ObjectType* or None) – The type of the object. If this is None (or not set), the type is detected automatically.

Returns class:~*maggit.git_objects.GitObject* object for this sha.

Return type A

Raises : Exception – If sha doesn't name a object.

get_peel_ref_sha(*ref*)

get_ref_sha(*name*)
Return the Sha corresponding to the reference name

Parameters **name** (*str*) – The reference name.

Returns class:~*maggit.Sha* corresponding.

Return type The

Exemples:

```
>>> repo.get_ref_sha('master')
<Sha >
```

Raises : Exception – If number of object corresponding to prefix is not equal to one.

get_tag(*sha*)
Get a tag object for the sha.

Parameters **sha** (*Sha*) – The sha of the tag.

Returns class:~*maggit.git_objects.Tag* object for this sha.

Return type A

Raises : Exception – If sha doesn't name a tag object.

get_tree(*sha*)
Get a tree object for the sha.

Parameters **sha** (*Sha*) – The sha of the tree.

Returns class:~*maggit.git_objects.Tree* object for this sha.

Return type A

Raises : Exception – If sha doesn't name a tree object.

init_repo(*gitdir*, *bare=False*)
Instantiate a new git repo at the given location.

tags
A dict of tags in the repo

class maggit.**ObjectType**
This is a enum listing possible type for a git object.

class maggit.repo.**Entry**(*repo*, *commit*, *path*, *mode*, *gitObject*)
A Entry represent a entry (file or directory) at a specific time.

We can somehow see a repository as a complex 2 dimentionnal array. Commits (and so the history) are rows. Files (and Trees) are columns.

In this situations, Entry are the cells of this array.

get_first_appearance()
Get the commit who firstly introduce the current version of the change.

Returns class:*~maggit.repo.Commit*

Return type A

parents
The previous versions of the files.

Previous versions can be equal to the current one if the current commit introduce no change on this file.

The length of the parents will most of the time be 1 but may be greater in case of merge.

class maggit.repo.**Commit**(*repo*, *sha*)

get_file(*path*)
Get an entry corresponding to the path in the commit

Parameters **path** (*str or Path*) – The path of the file to look at.

Returns class:*~maggit.repo.Entry* corresponding.

Return type A

Raise: KeyError if the path is not existing.

get_first_appearance(*path*)
Return the commit where the present version of path was introduce.

Parameters **path** (*str or path*) – The path of the file.

Returns class:*maggit.repo.Commit*

Return type A

Exemples:

```
>>> first_appearance_commit = this_commit.get_first_appearance(path)
>>> # first_appearance_commit is the first one, so previous version differs
>>> parent = first_appearance_commit.parents[0]
>>> assert first_appearance_commit.get_file(path).gitObject != parent.get_file(path).git
>>> # from this_commit to first_appearance_commit, there is no change
>>> current = this_commit
>>> while current != first_appearance_commit:
```

```
...     assert current.get_file(path).gitObject == this_commit.get_file(path).gitObject
...     current = current.parents[0]
```

get_first_appearances(*root=None*, *depthLimit=None*)
Return the first appearances for all entry in root.

This is mostly equivalent to *{path:commit.get_first_appearance(path) for path in commit.tree.entries}* (if root is None). But a way more performant as diving into history is made once.

Parameters

- **root** (*str or Path*) – In wich subdirectory we must look.
- **depthLimit** (*int*) – The commit limit number we go in history If depthLimit is specified, and for a entry the first appearance is older than depthLimit, the entry will be present in the dictionnary with a None value.

Returns class:*maggit.repo.Commit*).

Return type A dict of (Path,

3.3 maggit.git_objects module

class maggit.git_objects.**GitObject**(*repo*, *sha*)
The base class for all git objects.

Git objects are conceptualy constant. However as we try to be lazy, slots are not fill at object creation and set when user read it. So GitObject are not constant but behave as if they were.

repo
Repo

The repo associated with the object.

sha
Sha

The sha of the object.

gitType
ObjectType

The type of the object.

class maggit.git_objects.**Blob**(*repo*, *sha*)
Bases: *maggit.git_objects.GitObject*

A blob object.

content
bytes

This is the content of the blob.

class maggit.git_objects.**Tree**(*repo*, *sha*)
Bases: *maggit.git_objects.GitObject*

A blob object.

entries
unmutable mapping

This is the entries of the tree.

class maggit.git_objects.**Commit**(*repo*, *sha*)
Bases: *maggit.git_objects.GitObject*

A commit object.

tree
maggit.git_objects.Tree

The tree object associated with the commit.

parents
tuple

The parents of the commits. Most of the time, there will only one parent. In case of branch merge, there will be more than one parent.

author
bytes

The author of the commit. (This is the raw content in the commit. This means that it is a bytes with the name, email and commit timestamp)

committer
bytes

The committer of the commit. (This is the raw content in the commit. This means that it is a bytes with the name, email and commit timestamp)

first_line
str

The first line of the commit message.

message
str

The full commit message (including the first line).

class maggit.git_objects.**Tag**(*repo*, *sha*)
Bases: *maggit.git_objects.GitObject*

A tag object.

object
GitObject

The git object tagged by this tag.

objectType
ObjectType

The type of the git object tagged by this tag.

tag
str

The name of the tag.

tagger
bytes

The person who create the tag. (This is the raw content in the tag. This means that it is a bytes with the name, email and tag timestamp)

first_line
str

The first line of the tag message.

message
str

The full tag message (including the first line).

3.4 maggit.refs module

class maggit.refs.**Ref**(*repo*, *sha*)
Bases: maggit.refs.BaseRef

commit

repo

sha

class maggit.refs.**Branche**(*repo*, *branchName*)
Bases: maggit.refs.BaseRef

commit

name

repo

sha

class maggit.refs.**Tag**(*repo*, *tagName*)
Bases: maggit.refs.BaseRef

commit

name

object

repo

sha

CHAPTER 4

Indices and tables

- genindex
- modindex
- search

m

N

O

P

R

S

T