GitDict Documentation

Release 0.0.2

John Krauss

March 09, 2012

CONTENTS

CHAPTER

ONE

DICT

class gitdict.**GitDict**(*repo*, *key*, *autocommit*)

The GitDict object. Functions just like a dict, but it has a history and can be committed and merged.

These should be created with a DictRepository rather than instantiated directly.

autocommit = None

Whether changes should be automatically committed.

clear() → None. Remove all items from D.

commit(*author=None*, *committer=None*, *message=''*, *parents=None*)

Commit the dict to its repository.

Parameters

- **author** (*pygit2.Signature*) – (optional) The author of this commit. Defaults to global author.
- **committer** (*pygit2.Signature*) – (optional) The committer of this commit. Will default to global author.
- **message** (*string*) – (optional) The commit message. Defaults to a blank string.
- **parents** (*array*) – (optional) The 20-byte IDs of the parents of this commit. Defaults to the last commit.

copy() → a shallow copy of D

dirty

Whether this GitDict has uncommitted changes.

static fromkeys(*S*[, *v*]) → New dict with keys from S and values equal to v.
v defaults to None.

get(*k*[, *d*]) → D[k] if k in D, else d. d defaults to None.

has_key(*k*) → True if D has a key k, else False

items() → list of D's (key, value) pairs, as 2-tuples

iteritems() → an iterator over the (key, value) items of D

iterkeys() → an iterator over the keys of D

itervalues() → an iterator over the values of D

key

The String key for this dict in its repository.

keys() → list of D's keys

merge(*other*, *author=None*, *committer=None*)
Try to merge another GitDict into this one. If possible, will fast-forward the merged dict; otherwise, will attempt to merge in the intervening changes.

Parameters

- **other** (GitDict) – the GitDict to merge in.
- **author** (*pygit2.Signature*) – (optional) The author of this commit, if one is necessary. Defaults to global author.
- **committer** (*pygit2.Signature*) – (optional) The committer of this commit, if one is necessary. Will default to global author.

Returns True if the merge succeeded, False otherwise.

Return type boolean

pop(*k*[, *d*]) → v, remove specified key and return the corresponding value.
If key is not found, d is returned if given, otherwise KeyError is raised

popitem() → (k, v), remove and return some (key, value) pair as a
2-tuple; but raise KeyError if D is empty.

setdefault(*k*[, *d*]) → D.get(k,d), also set D[k]=d if k not in D

update(**args*, ***kwargs*)
Update as if this were a regular dict.

values() → list of D's values

viewitems() → a set-like object providing a view on D's items

viewkeys() → a set-like object providing a view on D's keys

viewvalues() → an object providing a view on D's values

CHAPTER

TWO

REPOSITORY

class gitdict.**DictRepository**(*repo_or_path=None*)

The DictRepository object.

Parameters **repo_or_path** (*string or pygit2.Repository*) – The path to a repository, or an existing pygit2.Repository object. If it is a path that does not exist, a new bare git repository will be initialized there. If it is a path that does exist, then the directory will be used as a bare git repository.

clone(*original*, *key*)

Clone a GitDict.

Parameters

- **original** (GitDict) – the GitDict to clone
- **key** (*string*) – where to clone to

Raises ValueError if to_key already exists.

create(*key*, *dict={}*, *autocommit=False*, *message='first commit'*, *author=None*, *committer=None*)

Create a new GitDict

Parameters

- **key** (GitDict) – The key of the new GitDict
- **dict** (*dict*) – (optional) The value of the dict. Defaults to empty.
- **autocommit** (*boolean*) – (optional) Whether the GitDict should automatically commit. Defaults to false.
- **message** (*string*) – (optional) Message for first commit. Defaults to "first commit".
- **author** (*pygit2.Signature*) – (optional) The signature for the author of the first commit. Defaults to global author.
- **committer** – (optional) The signature for the committer of the first commit. Defaults to author.

Returns the GitDict

Return type GitDict

fast_forward(*from_dict*, *to_dict*)

Fast forward a GitDict.

Parameters

- **from_dict** (GitDict) – the GitDict to fast forward.

- **to_dict** (GitDict) – the :class:‘GitDict <GitDict>‘to fast forward to.

get (*key*, *autocommit=False*)
Obtain the GitDict for a key.

Parameters

- **key** (*string*) – The key to look up.
- **default** (*dict*) – (optional) The default dict value if there is no existing value for the key. Defaults to an empty dict.
- **autocommit** (*boolean*) – (optional) Whether the GitDict should automatically commit. Defaults to false.

Returns the GitDict

Return type GitDict

Raises KeyError if there is no entry for key

raw_commit (*key*, *raw_dict*, *author*, *committer*, *message*, *parents*)
Commit a dict to this DictRepository. It is recommended that you use the GitDict commit method instead.

Parameters

- **raw_dict** (*dict*) – the data to commit.
- **author** (*pygit2.Signature*) – The author of the commit. If None, will be replaced with default.
- **committer** (*pygit2.Signature*) – The committer of this commit. If None, will be replaced with author.
- **message** (*string*) – The commit message.
- **parents** – A list of 20-byte object IDs of parent commits. An empty list means this is the first commit.

Returns The oid of the new commit.

Return type 20 bytes

CHAPTER

THREE

OTHER

class gitdict.**DictAuthor**(*name*, *email*)
Wrapper for a name and email to use when committing to git.

Parameters

- **name** (*string*) – The name to use when committing
- **email** (*string*) – The email to use when committing

email
The author's email.

name
The author's name.

signature(*time=None*, *offset=None*)
Generate a pygit2.Signature.

Parameters

- **time** (*int*) – (optional) the time for the signature, in UTC seconds. Defaults to current time.
- **offset** (*int*) – (optional) the time offset for the signature, in minutes. Defaults to the system offset.

Returns a signature

Return type pygit2.Signature

class gitdict.**DictDiff**(*dict1*, *dict2*)
A wrapper around json_diff's dict comparisons.

Parameters

- **dict1** (*dict*) – The first dict. Can be a GitDict.
- **dict2** (*dict*) – The second dict. Can be a GitDict.

appended
A dict of appended keys and their values.

conflict(*other*)
Determine whether this :class:'DictDiff <DictDiff>'s conflicts with another.

Parameters **other** (DictDiff) – The diff to compare.

Returns the conflicts

Return type DictConflict

removed
A dict of removed keys and their values.

updated
A dict of updated keys and their values.

PYTHON MODULE INDEX