hgapi Documentation

Release 1.7.3

fredrik@haard.se

November 13, 2016

Contents:

CHAPTER 1

hgapi Package

1.1 `hgapi` Module

exception `hgapi.hgapi.`**`HgException`**(*msg*, *exit_code=None*)
Exception class allowing a exit_code parameter and member to be used when calling Mercurial to return exit code.

class `hgapi.hgapi.`**`Repo`**(*path*, *user=None*)
A representation of a Mercurial repository.

classmethod **`command`**(*path*, *env*, **args*)
Run a hg command in path and return the result.

Raise on error.

`config`(*section*, *key*)
Return the value of a configuration variable.

`configbool`(*section*, *key*)
Return a config value as a boolean value.

Empty values, the string 'false' (any capitalization), and '0' are considered False, anything else is True

`configlist`(*section*, *key*)
Return a config value as a list.

Will try to create a list delimited by commas, or whitespace if no commas are present.

`get_branch_names`()
Returns a list of branch names from the repo.

`get_branches`()
Returns a list of branches from the repo, including versions.

If get_active_only is True, then only return active branches.

`hg_add`(*filepath=None*)
Add a file to the repo.

when no filepath is given, all files are added to the repo.

`hg_addremove`(*filepath=None*)
Add a file to the repo.

When no filepath is given, all files are added and removed to and respectively from the repo.

`hg_archive`(*destination*, *revision=None*, *archive_type=None*)
Archive a repository.

Creates an archive of a single revision in the specified destination.

If revision is not supplied the default is the parent of the repository's working directory (tip).

If archive_type is not supplied mercurial will determine the type based on the file extension. If there is no file extension the default is "files".

`hg_branch`(*branch_name=None*)
Create a branch or get a branch name.

If branch_name is not None, the branch is created. Otherwise the current branch name is returned.

classmethod `hg_clone`(*url*, *path*, **args*)
Clone repository at given *url* to *path*, then return repo object to *path*.

`hg_command`(**args*)
Run a hg command.

`hg_commit`(*message*, *user=None*, *date=None*, *files=[]*, *close_branch=False*, *amend=False*, *message_file=None*)
Commit changes to the repository.

`hg_diff`(*rev_a=None*, *rev_b=None*, *filenames=None*)
Get a unified diff as returned by 'hg diff'.

rev_a and rev_b are passed as -r <rev> arguments to the call, filenames are expected to be an iterable of file names.

Returns a list of dicts where every dict has a 'filename' and 'diff' field, where with diff being the complete diff for the file including header (diff -r xxxx -r xxx...).

`hg_heads`(*short=False*)
Get a list with the node identifiers of all open heads. If short is given and is not False, return the short form of the node id.

`hg_id`()
Get the output of the hg id command (truncated node).

`hg_incoming`(*remote=u'default'*)
Get incoming changesets for a certain remote.

`hg_init`()
Initialize a new repo.

`hg_log`(*identifier=None*, *limit=None*, *template=None*, *branch=None*, ***kwargs*)
Get repositiory log.

`hg_merge`(*reference*, *preview=False*)
Merge reference to current.

With 'preview' set to True get a list of revision numbers containing all revisions that would have been merged.

`hg_move`(*source*, *destination*)
Move a file in the repo.

`hg_node`()
Get the full node id of the current revision.

`hg_outgoing`(*remote=u'default'*)
Get outgoing changesets for a certain remote.

hg_paths()
Get remote repositories.

hg_pull(*source=None*)
Pull changes to this repo.

hg_push(*destination=None*)
Push changes from this repo.

hg_remove(*filepath*)
Remove a file from the repo

hg_rename(*source*, *destination*)
Move a file in the repo. This is hg_more.

hg_rev()
Get the revision number of the current revision.

hg_revert(*all=False*, **files*)
Revert repository.

classmethod hg_root(*path*)
Return the root (top) of the path.

When no path is given, current working directory is used. Raises HgException when no repo is available.

hg_status(*empty=False*, *clean=False*)
Get repository status.

Returns a dict containing a *change char* -> *file list* mapping, where change char is in:

```
A, M, R, !, ?
```

Example after adding one.txt, modifying a_folder/two.txt and three.txt:

```
{'A': ['one.txt'], 'M': ['a_folder/two.txt', 'three.txt'],
'!': [], '?': [], 'R': []}
```

If empty is set to non-False value, don't add empty lists. If clean is set to non-False value, add clean files as well (-A)

hg_tag(**tags*, ***kwargs*)

Add one or more tags to the current revision.

Add one or more tags to the current revision, or revision given by passing 'rev' as a keyword argument:

```
>>> repo.hg_tag('mytag', rev=3)
```

hg_tags()
Get all tags from the repo.

Returns a dict containing tag: shortnode mapping

hg_update(*reference*, *clean=False*)
Update to the revision identified by reference.

classmethod hg_version()
Return the version number of Mercurial.

read_config()
Read the configuration as seen with 'hg showconfig'.

Is called by __init__ - only needs to be called explicitly to reflect changes made since instantiation.

revision(*identifier*)
Get the identified revision as a Revision object.

revisions(*slice_*)
Returns a list of Revision objects for the given slice

class hgapi.hgapi.**Revision**(*json_log*)
A representation of a revision. Available fields are:

```
node, rev, author, branch, parents, date, tags, desc
```

A Revision object is equal to any other object with the same value for node.

CHAPTER 2

Indices and tables

- genindex
- modindex
- search

Python Module Index

Index