vcs Documentation

version 0.1.10

Marcin Kuzminski & Lukasz Balcerzak

October 21, 2010

Contents

Welcome to vcs's documentation!

`vcs` is abstraction layer over various version control systems. It is designed as feature-rich Python library with clear *API Reference*.

vcs uses Semantic Versioning

Features

- Common *API* for SCM *backends*
- Fetching repositories data lazily
- Simple caching mechanism so we don't hit repo too often
- Command line client (still basic)

Incoming

- Django app for mercurial hgserve replacement
- Simple commit api
- Smart and powerfull in memory Workdirs
- VCS based wiki

Documentation

Installation:

Quickstart

Say you don't want to install vcs or just want to begin with really fast tutorial? Not a problem, just follow sections below.

Prepere

We will try to show you how you can use vcs directly on repository. But hey, vcs is maintained within mercurial repository already, so why not use it? Simply run following commands in your shell

```
cd /tmp
hg clone http://bitbucket.org/marcinkuzminski/vcs/
cd vcs
```

Now run your python interpreter of choice:

```
$ python
>>>
```

Note

You may of course put your clone of vcs wherever you like but running python shell *inside* of it would allow you to use just cloned version of vcs.

Take the shortcut

There is no need to import everything from vcs - in fact, all you'd need is to import get_repo, at least for now. Then, simply initialize repository object by providing it's type and path.

```
>>> from vcs import get_repo
>>>
>>> # create mercurial repository representation at current dir
>>> # alias tells which backend should be used (see vcs.BACKENDS)
>>> repo = get_repo(alias='hg', path='.')
```

Basics

Let's ask repo about the content...

```
>>> root = repo.get_changeset().get_node('')
>>> print root.nodes # prints nodes of the RootNode
[<DirNode ''>, <DirNode 'docs'>, <DirNode 'tests'>, # ... (chopped)
>>>
>>> # get 10th changeset
>>> chset = repo.get_changeset(10)
>>> print chset
<MercurialChangeset at 10>
>>>
>>> # any backend would return latest changeset if revision is not given
>>> tip = repo.get_changeset()
>>> tip is repo.get_changeset('tip') # for mercurial backend 'tip' is allowed
True
```

```
>>> tip is repo.get_changeset(None) # any backend allow revision to be None (defaul
True
>>> tip.revision is repo.revisions[-1]
True
>>>
>>> # Iterate repository
>>> list(repo) == repo.changesets.values()
False
>>> # Those are not equal, as repo iterator returns only changesets for keys
>>> # from repo.revisions and repo.changesets is a dict caching calls for
>>> # each changeset; but repo iterator would always be a subset of cached
>>> # changesets
>>> set(list(repo)).issubset(set(repo.changesets.values()))
True
```

Walking

Now let's ask for nodes at revision 44 (http://bitbucket.org/marcinkuzminski/vcs/src/a0eada0b9e4e/)

```
>>> chset44 = repo.get_changeset(44)
>>> root = chset44.root
>>> # You may also use shorter one-liner:
>>> root = repo.get_changeset(44).get_node('')
```

Note

If you have to check this to believe, you may get raw id of the changeset and open browser on same changeset at bitbucket:

```
>>> print root.changeset.short_id
a0eada0b9e4e
```

This show us that 44 revision has hex of (shorter version): `a0eada0b9e4e` which you can follow on bitbucket at: http://bitbucket.org/marcinkuzminski/vcs/src/a0eada0b9e4e/

```
>>> print root.dirs
[<DirNode 'docs'>, <DirNode 'examples'>, <DirNode 'tests'>, <DirNode 'vcs'>]
```

Note

Nodes are objects representing files and directories within the repository revision.

```
>>> # Fetch vcs directory
>>> vcs = repo.get_changeset(44).get_node('vcs')
>>> print vcs.dirs
[<DirNode 'vcs/backends'>, <DirNode 'vcs/utils'>, <DirNode 'vcs/web'>]
>>> web_node = vcs.dirs[-1]
>>> web = repo.get_changeset(44).get_node(web_node.path)
>>> print web.nodes
[<DirNode 'vcs/web/simplevcs'>, <FileNode 'vcs/web/__init__.py'>]
>>> print web.files
[<FileNode 'vcs/web/__init__.py'>]
>>> web.files[0].content
''
```

```
>>> print vcs.files[0].content
"""
Various Version Control System management abstraction layer for Python.
"""

VERSION = (0, 0, 1, 'alpha')

__version__ = '.'.join((str(each) for each in VERSION[:4]))

__all__ = [
    'get_repo', 'get_backend', 'BACKENDS',
    'VCSError', 'RepositoryError', 'ChangesetError']

from vcs.backends import get_repo, get_backend, BACKENDS
from vcs.exceptions import VCSError, RepositoryError, ChangesetError


>>> chset44 = repo.get_changeset(44)
>>> chset44.get_node('vcs/web') is web
True
>>> # same if we span ``get_node`` methods:
>>> chset44.get_node('vcs').get_node('web') is web
True
```

Getting meta data

Make vcs show us some meta information

Tags and branches

```
>>> print repo.branches
{'default': 'f1ffc1cfbae0', 'git': '735ca3f85433', 'web': '2e6a2bf9356c'}
# above values may vary
>>> print repo.tags
{'0.1.1': 'eb3a60fc9643', '0.1.2': 'a7e60bff65d5', 'tip': 'f1ffc1cfbae0'}
```

Give me a file, finally!

```
>>> root = repo.get_changeset(44).get_node('')
>>> backends = root.get_node('vcs/backends')
>>> backends.files
[<FileNode 'vcs/backends/__init__.py'>,
 <FileNode 'vcs/backends/base.py'>,
 <FileNode 'vcs/backends/hg.py'>]
>>> f = backends.get_node('hg.py')
>>> f.name
'hg.py'
>>> f.path
'vcs/backends/hg.py'
>>> f.size
8882
>>> f.last_changeset
<MercurialChangeset at 44>
>>> f.last_changeset.date
datetime.datetime(2010, 4, 14, 14, 8)
>>> f.last_changeset.message
'Cleaning up codes at base/mercurial backend'
>>> f.last_changeset.author
'Lukasz Balcerzak <lukasz.balcerzak@python-center.pl>'
```

```
>>>
>>> f.mimetype
'text/x-python'
>>>
>>> # Following would raise exception unless you have pygments installed
>>> f.lexer
<pygments.lexers.PythonLexer>
>>> f.lexer_alias # shortcut to get first of lexers' available aliases
'python'
>>> f.name
'hg.py'
>>>
>>> # wanna go back? why? oh, whatever...
>>> f.parent
<DirNode 'vcs/backends'>
>>>
>>> # is it cached? hell yeah...
>>> f is f.parent.get_node('hg.py') is repo.get_changeset(44).get_node('vcs/backend
True
```

How about history?

New in version 0.1.1.

It is possible to retrieve changesets for which file node has been changed and this is pretty damn simple. Let's say we want to see history of the file located at vcs/nodes.py.

```
>>> f = repo.get_changeset().get_node('vcs/nodes.py')
>>> print f.history
[<MercurialChangeset at 82>, <MercurialChangeset at 81>, <MercurialChange
...
```

Note that history attribute is computed lazily and returned list is reversed - changesets are retrieved from most recent to oldest.

Show me the difference!

Here we present naive implementation of diff table for the given file node located at vcs/nodes.py. First we have to get the node from repository. After that we retrieve last changeset for which the file has been modified and we create a html file using difflib.

```
>>> f = repo.get_changeset(82).get_node('vcs/nodes.py')
>>> f_old = repo.get_changeset(81).get_node(f.path)
>>> out = open('out.html', 'w')
>>> from difflib import HtmlDiff
>>> hd = HtmlDiff(tabsize=4)
>>> diffs = hd.make_file(f.content.split('\n'), f_old.content.split('\n'))
>>> out.write(diffs)
>>> out.close()
```

Installation

vcs is simply, pure python package. However, it makes use of various *version control systems* and thus, would require some third part libraries and they may have some deeper dependencies.

Requirements

Below is a table which shows requirements for each backend.

SCM	Backend	Alias	Requirements

Mercurial	`vcs.backend.hg`	`hg`	• mercurial >= 1.6
Git	`vcs.backend.git`	`git`	• git >= 1.7 • Dulwich >= 0.6

Install from Cheese Shop

Easiest way to install `vcs` is to run:

```
easy_install vcs
```

Or:

```
pip install vcs
```

If you prefer to install manually simply grab latest release from http://pypi.python.org/pypi/vcs, decompress archive and run:

```
python setup.py install
```

Development

In order to test the package you'd need all backends' underlying libraries (see table above) and unittest2 as we use it to run test suites.

Here is a full list of packages needed to run test suite:

Package	Homepage
unittest2	http://pypi.python.org/pypi/unittest2
mercurial	http://mercurial.selenic.com/
git	http://git-scm.com
dulwich	http://pypi.python.org/pypi/dulwich

API Reference

Nodes

Node

class `vcs.nodes.`**`Node`** (*path*, *kind*)

Simplest class representing file or directory on repository. SCM backends should use `FileNode` and `DirNode` subclasses rather than `Node` directly.
Node's `path` cannot start with slash as we oparete on *relative* paths only. Moreover, every single node is identified by the `path` attribute, so it cannot end with slash, too. Otherwise, path could lead to mistakes.

`get_parent_path ()`
Returns node's parent path or empty string if node is root.

`is_dir ()`
Returns `True` if node's kind is `NodeKind.DIR`, `False` otherwise.

`is_file ()`
Returns `True` if node's kind is `NodeKind.FILE`, `False` otherwise.

`is_root ()`

Returns `True` if node is a root node and `False` otherwise.

FileNode

class `vcs.nodes.`**`FileNode`** (*path*, *content=None*, *changeset=None*)

Class representing file nodes.

Attribute : path: path to the node, relative to repostiory's root

Attribute : content: if given arbitrary sets content of the file

Attribute : changeset: if given, first time content is accessed, callback

Only one of `content` and `changeset` may be given. Passing both would raise `NodeError` exception.

Parameters:

- **path** -- relative path to the node
- **content** -- content may be passed to constructor
- **changeset** -- if given, will use it to lazily fetch content

RemovedFileNode

class `vcs.nodes.`**`RemovedFileNode`** (*path*)

Dummy FileNode class - trying to access any public attribute except path, name, kind or state (or methods/attributes checking those two) would raise RemovedFileNodeError.

Parameters:

- **path** -- relative path to the node

DirNode

class `vcs.nodes.`**`DirNode`** (*path*, *nodes=()*, *changeset=None*)

DirNode stores list of files and directories within this node. Nodes may be used standalone but within repository context they lazily fetch data within same repositorty's changeset.

Only one of `nodes` and `changeset` may be given. Passing both would raise `NodeError` exception.

Parameters:

- **path** -- relative path to the node
- **nodes** -- content may be passed to constructor
- **changeset** -- if given, will use it to lazily fetch content
- **size** -- always 0 for `DirNode`

`get_node` **(*path*)**

Returns node from within this particular `DirNode`, so it is now allowed to fetch, i.e. node located at 'docs/api/index.rst' from node 'docs'. In order to access deeper nodes one must fetch nodes between them first - this would work:

```
docs = root.get_node('docs')
docs.get_node('api').get_node('index.rst')
```

Param : path - relative to the current node

> ### *Note*
>
> To access lazily (as in example above) node have to be initialized with related changeset object - without it node is out of context and may know nothing about anything else than nearest (located at same level) nodes.

RootNode

class `vcs.nodes.`**`RootNode`** (*nodes=()*, *changeset=None*)
DirNode being the root node of the repository.

Backends

Implemented Backends

Git Backend

Git backend implementation.

GitRepository

class `vcs.backends.git.`**`GitRepository`** (*repo_path*, *create=False*, *src_url=None*, *update_after_clone=False*)
Git repository backend.

`clone` (*url*, *update_after_clone*)
Tries to clone changes from external location. if update_after_clone is set To false it'll prevent the runing update on workdir

`get_changeset` (*revision=None*)
Returns `GitChangeset` object representing commit from git repository at the given revision or head (most recent commit) if None given.

`get_changesets` (*limit=10*, *offset=None*)
Return last n number of `MercurialChangeset` specified by limit attribute if None is given whole list of revisions is returned :param limit: int limit or None

`run_git_command` (*cmd*)
Runs given `cmd` as git command and returns tuple (returncode, stdout, stderr).

Note

This method exists only until log/blame functionality is implemented at Dulwich (see https://bugs.launchpad.net/bugs/645142). Parsing os command's output is road to hell...

Parameters:
- **cmd** -- git command to be executed

GitChangeset

class `vcs.backends.git.`**`GitChangeset`** (*repository*, *revision*)
Bases: **`vcs.backends.base.BaseChangeset`**
Represents state of the repository at single revision.

`id`
Returns same as `raw_id` attribute.

`raw_id`
Returns raw string identifing this changeset (40-length sha)

`short_id`

Returns shortened version of `raw_id` (first 12 characters)

revision
Returns integer representing changeset.

parents
Returns list of parents changesets.

added
Returns list of added `FileNode` objects.

changed
Returns list of changed `FileNode` objects.

removed
Returns list of removed `RemovedFileNode` objects.

> ***Note***
>
> Remember that those `RemovedFileNode` instances are only dummy `FileNode` objects and trying to access most of it's attributes or methods would raise `NodeError` exception.

get_file_annotate (*path*)
Returns a list of three element tuples with lineno,changeset and line
TODO: This function now uses os underlying 'git' command which is generally not good. Should be replaced with algorithm iterating commits.

get_file_changeset (*path*)
Returns last commit of the file at the given `path`.

get_file_content (*path*)
Returns content of the file at given `path`.

get_file_history (*path*)
Returns history of file as reversed list of `Changeset` objects for which file at given `path` has been modified.
TODO: This function now uses os underlying 'git' and 'grep' commands which is generally not good. Should be replaced with algorithm iterating commits.

get_file_size (*path*)
Returns size of the file at given `path`.

get_node (*path*)

get_nodes (*path*)

walk (*topurl=''*)
Similar to os.walk method. Insted of filesystem it walks through changeset starting at given `topurl`. Returns generator of tuples (topnode, dirnodes, filenodes).

GitInMemoryChangeset

class vcs.backends.git.**GitInMemoryChangeset** (*repository*)
Bases: **vcs.backends.base.BaseInMemoryChangeset**

add (filenodes*)**
Marks given `FileNode` objects as *to be committed*.

Raises:

- **NodeAlreadyExistsError** -- if node with same path exists at latest changeset
- **NodeAlreadyAddedError** -- if node with same path is already marked as *added*

change (**filenodes*)
Marks given `FileNode` objects to be *changed* in next commit.

Raises:

- **EmptyRepositoryError** -- if there are no changesets yet
- **NodeAlreadyExistsError** -- if node with same path is already marked to be *changed*
- **NodeAlreadyRemovedError** -- if node with same path is already marked to be *removed*
- **NodeDoesNotExistError** -- if node doesn't exist in latest changeset
- **NodeNotChangedError** -- if node hasn't really be changed

commit (*message*, *author*, ***kwargs*)
Performs in-memory commit.

get_ipaths ()
Returns generator of paths from nodes marked as added, changed or removed.

get_paths ()
Returns list of paths from nodes marked as added, changed or removed.

remove (**filenodes*)
Marks given `FileNode` (or `RemovedFileNode`) objects to be *removed* in next commit.

Raises:

- **EmptyRepositoryError** -- if there are no changesets yet
- **NodeDoesNotExistError** -- if node does not exist in latest changeset
- **NodeAlreadyRemovedError** -- if node has been already marked to be *removed*
- **NodeAlreadyChangedError** -- if node has been already marked to be *changed*

reset ()
Resets this instance to initial state (cleans `added`, `changed` and `removed` lists).

Mercurial Backend

Created on Apr 8, 2010

author: marcink,lukaszb

MercurialRepository

class `vcs.backends.hg.`**MercurialRepository** (*repo_path*, *create=False*, *baseui=None*, *src_url=None*, *update_after_clone=False*)
Mercurial repository backend
Raises RepositoryError if repository could not be find at the given `repo_path`.

Parameters:

- **repo_path** -- local path of the repository
- **create=False** -- if set to True, would try to craete repository if it does not exist rather than raising exception
- **baseui=None** -- user data
- **src_url=None** -- would try to clone repository from given location
- **update_after_clone=False** -- sets update of working copy after making a clone

`get_changeset` (***`revision=None`***)
Returns `MercurialChangeset` object representing repository's changeset at the given `revision`.

`get_changesets` (***`limit=10`***, ***`offset=None`***)
Return last n number of `MercurialChangeset` specified by limit attribute if None is given whole list of revisions is returned

Parameters:

- **limit** -- int limit or None
- **offset** -- int offset

`pull` (***`url`***)
Tries to pull changes from external location.

MercurialChangeset

class `vcs.backends.hg.`**`MercurialChangeset`** (*`repository`*, *`revision`*)
Bases: **`vcs.backends.base.BaseChangeset`**
Represents state of the repository at the single revision.

`id`
Returns shorter version of mercurial's changeset hexes.

`raw_id`
Returns raw string identifing this changeset (40-length hex)

`short_id`
Returns shortened version of `raw_id` (first 12 characters)

`revision`
Returns integer representing changeset.

`parents`
Returns list of parents changesets.

`added`
Returns list of added `FileNode` objects.

`changed`
Returns list of changed `FileNode` objects.

`removed`
Returns list of removed `RemovedFileNode` objects.

Note

Remember that those `RemovedFileNode` instances are only dummy `FileNode` objects and trying to access most of it's attributes or methods would raise `NodeError` exception.

get_file_annotate (*path*)
Returns a list of three element tuples with lineno,changeset and line

get_file_changeset (*path*)
Returns last commit of the file at the given `path`.

get_file_content (*path*)
Returns content of the file at given `path`.

get_file_history (*path*)
Returns history of file as reversed list of `Changeset` objects for which file at given `path` has been modified.

get_file_size (*path*)
Returns size of the file at given `path`.

get_node (*path*)
Returns `Node` object from the given `path`. If there is no node at the given `path`, `ChangesetError` would be raised.

get_nodes (*path*)
Returns combined `DirNode` and `FileNode` objects list representing state of changeset at the given `path`. If node at the given `path` is not instance of `DirNode`, ChangesetError would be raised.

walk (*topurl=''*)
Similar to os.walk method. Insted of filesystem it walks through changeset starting at given `topurl`. Returns generator of tuples (topnode, dirnodes, filenodes).

MercurialInMemoryChangeset

class vcs.backends.hg.**MercurialInMemoryChangeset** (*repository*)
Bases: **vcs.backends.base.BaseInMemoryChangeset**

add (**filenodes*)
Marks given `FileNode` objects as *to be committed*.

Raises:

- **NodeAlreadyExistsError** -- if node with same path exists at latest changeset
- **NodeAlreadyAddedError** -- if node with same path is already marked as *added*

change (**filenodes*)
Marks given `FileNode` objects to be *changed* in next commit.

Raises:

- **EmptyRepositoryError** -- if there are no changesets yet
- **NodeAlreadyExistsError** -- if node with same path is already marked to be *changed*
- **NodeAlreadyRemovedError** -- if node with same path is already marked to be *removed*
- **NodeDoesNotExistError** -- if node doesn't exist in latest changeset
- **NodeNotChangedError** -- if node hasn't really be changed

commit (*message*, *author*, ***kwargs*)

get_ipaths ()
Returns generator of paths from nodes marked as added, changed or removed.

get_paths ()
Returns list of paths from nodes marked as added, changed or removed.

remove (filenodes*)**
Marks given `FileNode` (or `RemovedFileNode`) objects to be *removed* in next commit.

Raises:

- **EmptyRepositoryError** -- if there are no changesets yet
- **NodeDoesNotExistError** -- if node does not exist in latest changeset
- **NodeAlreadyRemovedError** -- if node has been already marked to be *removed*
- **NodeAlreadyChangedError** -- if node has been already marked to be *changed*

reset ()
Resets this instance to initial state (cleans `added`, `changed` and `removed` lists).

Base Backend

Created on Apr 8, 2010

author: marcink,lukaszb

class `vcs.backends.base.`**`BaseChangeset`**
Each backend should implement it's changeset representation.
Attributes

`repository`
repository object within which changeset exists

`id`
may be `raw_id` or i.e. for mercurial's tip just `tip`

`raw_id`
raw changeset representation (i.e. full 40 length sha for git backend)

`short_id`
shortened (if apply) version of `raw_id`; it would be simple shortcut for `raw_id[:12]` for git/mercurial backends or same as `raw_id` for subversion

`revision`
revision number as integer

`files`
list of `FileNode` (`Node` with NodeKind.FILE) objects

`dirs`
list of `DirNode` (`Node` with NodeKind.DIR) objects

`nodes`
combined list of `Node` objects

`author`
author of the changeset, as unicode

`message`
message of the changeset, as unicode

`parents`
list of parent changesets

`last`

`True` if this is last changeset in repository, `False` otherwise; trying to access this attribute while there is no changesets would raise `EmptyRepositoryError`

`get_file_changeset` (***`path`***)

Returns last commit of the file at the given `path`.

`get_file_content` (***`path`***)

Returns content of the file at the given `path`.

`get_file_history` (***`path`***)

Returns history of file as reversed list of `Changeset` objects for which file at given `path` has been modified.

`get_file_size` (***`path`***)

Returns size of the file at the given `path`.

`get_node` (***`path`***)

Returns `Node` object from the given `path`.

Raises NodeDoesNotExistError: if there is no node at the given `path`

`get_nodes` (***`path`***)

Returns combined `DirNode` and `FileNode` objects list representing state of changeset at the given `path`.

Raises ChangesetError: if node at the given `path` is not instance of `DirNode`

`walk` (***`topurl`***`=''`)

Similar to os.walk method. Insted of filesystem it walks through changeset starting at given `topurl`. Returns generator of tuples (topnode, dirnodes, filenodes).

class `vcs.backends.base.`**`BaseInMemoryChangeset`** (*`repository`*)

Represents differences between repository's state (most recent head) and changes made *in place*.

Attributes

`repository`

repository object for this in-memory-changeset

`added`

list of `FileNode` objects marked as *added*

`changed`

list of `FileNode` objects marked as *changed*

`removed`

list of `FileNode` or `RemovedFileNode` objects marked to be *removed*

`add` (***`*filenodes`***)

Marks given `FileNode` objects as *to be committed*.

Raises:

- **NodeAlreadyExistsError** -- if node with same path exists at latest changeset
- **NodeAlreadyAddedError** -- if node with same path is already marked as *added*

`change` (***`*filenodes`***)

Marks given `FileNode` objects to be *changed* in next commit.

Raises:

- **EmptyRepositoryError** -- if there are no changesets yet
- **NodeAlreadyExistsError** -- if node with same path is already marked to be *changed*
- **NodeAlreadyRemovedError** -- if node with same path is already marked to be *removed*
- **NodeDoesNotExistError** -- if node doesn't exist in latest changeset
- **NodeNotChangedError** -- if node hasn't really be changed

`commit` **(*`message`*, `**`*`kwargs`*)**

Commits local (from working directory) changes and returns newly created `Changeset`. Updates repository's `revisions` list.

Raises CommitError: if any error occurs while committing

`get_ipaths ()`

Returns generator of paths from nodes marked as added, changed or removed.

`get_paths ()`

Returns list of paths from nodes marked as added, changed or removed.

`remove` **(`*`*`filenodes`*)**

Marks given `FileNode` (or `RemovedFileNode`) objects to be *removed* in next commit.

Raises:

- **EmptyRepositoryError** -- if there are no changesets yet
- **NodeDoesNotExistError** -- if node does not exist in latest changeset
- **NodeAlreadyRemovedError** -- if node has been already marked to be *removed*
- **NodeAlreadyChangedError** -- if node has been already marked to be *changed*

`reset ()`

Resets this instance to initial state (cleans `added`, `changed` and `removed` lists).

class `vcs.backends.base.`**`BaseRepository`** (*`repo_path`*, *`create=False`*, `**`*`kwargs`*)

Base Repository for final backends

Attributes

`repo`
: object from external api

`revisions`
: list of all available revisions' ids, in ascending order

`changesets`
: storage dict caching returned changesets

`path`
: absolute path to the repository

`branches`
: branches as list of changesets

`tags`
: tags as list of changesets

Initializes repository. Raises RepositoryError if repository could not be find at the given `repo_path` or directory at `repo_path` exists and `create` is set to True.

Parameters:

- **repo_path** -- local path of the repository
- **create=False** -- if set to True, would try to craete repository.
- **src_url=None** -- if set, should be proper url from which repository would be cloned; requires `create` parameter to be set to True - raises RepositoryError if src_url is set and create evaluates to False

`add (`*`filenode`*`, **`*`kwargs`*`)`

Commit api function that will add given `FileNode` into this repository.

Raises:

- **NodeAlreadyExistsError** -- if there is a file with same path already in repository
- **NodeAlreadyAddedError** -- if given node is already marked as *added*

`commit (`*`message`*`, **`*`kwargs`*`)`

Persists current changes made on this repository and returns newly created changeset.

Raises NothingChangedError: if no changes has been made

`get_changeset (`*`revision=None`*`)`

Returns instance of `Changeset` class. If `revision` is None, most recenent changeset is returned.

Raises EmptyRepositoryError: if there are no revisions

`get_changesets (`*`limit=10`*`, `*`offset=None`*`)`

Return last n number of `Changeset` objects specified by limit attribute if None is given whole list of revisions is returned

Param : `limit`: int limit or None

Param : `offset`: int offset

`get_state ()`

Returns dictionary with `added`, `changed` and `removed` lists containing `FileNode` objects.

`is_valid ()`

Validates repository.

`remove (`*`filenode`*`, **`*`kwargs`*`)`

Commit api function that will remove given `FileNode` into this repository.

Raises:

- **EmptyRepositoryError** -- if there are no changesets yet
- **NodeDoesNotExistError** -- if there is no file with given path

class `vcs.backends.base.`**`BaseWorkdir`** (*`repository`*)

Working directory representation of single repository.

Attribute : repository: repository object of working directory

`commit (`*`message`*`, **`*`kwargs`*`)`

Commits local (from working directory) changes and returns newly created `Changeset`. Updates repository's `revisions` list.

Raises CommitError: if any error occurs while committing

get_added **()**
Returns list of `FileNode` objects marked as *new* in working directory.

get_changed **()**
Returns list of `FileNode` objects *changed* in working directory.

get_removed **()**
Returns list of `RemovedFileNode` objects marked as *removed* in working directory.

get_status **()**
Returns dict with `added`, `changed`, `removed` and `untracked` lists.

get_untracked **()**
Returns list of `FileNode` objects which are present within working directory however are not tracked by repository.

update **(*revision=None*)**
Fetches content of the given revision and populates it within working directory.

Utils

This module provides some useful tools for `vcs` like annotate/diff html output. It also includes some internal helpers.

Public API

- *Annotate utils*
- *Diffs utils*
- *Helpers*

Private API

- *Lazy attributes utils*

Annotate utils

`vcs.utils.annotate.`**annotate_highlight** (*filenode*, *annotate_from_changeset_func=None*, *order=None*, *headers=None*, ***options*)
Returns html portion containing annotated table with 3 columns: line numbers, changeset information and pygmentized line of code.

Parameters:

- **filenode** -- FileNode object
- **annotate_from_changeset_func** -- function taking changeset and returning single annotate cell; needs break line at the end
- **order** -- ordered sequence of `ls` (line numbers column), `annotate` (annotate column), `code` (code column); Default is `['ls', 'annotate', 'code']`
- **headers** -- dictionary with headers (keys are whats in `order` parameter)

Diffs utils

class `vcs.utils.diffs.`**DiffProcessor** (*udiff*, *differ='udiff'*)
Give it a unified diff and it returns a list of the files that were mentioned in the diff together with a dict of meta information that can be used to render it in a HTML template.

Parameters:
- **udiff** -- a text in udiff format

`as_html` *`(table_class='code-difftable', line_class='line', new_lineno_class='lineno old', old_lineno_class='lineno new', code_class='code')`*
Return udiff as html table with customized css classes

`copy_iterator ()`
make a fresh copy of generator, we should not iterate thru an original as it's needed for repeating operations on this instance of DiffProcessor

`prepare ()`
Prepare the passed udiff for HTML rendering. It'l return a list of dicts

`raw_diff ()`
Returns raw string as udiff

`vcs.utils.diffs.`**`get_gitdiff`** (*`filenode_old`*, *`filenode_new`*)
Returns mercurial style git diff between given `filenode_old` and `filenode_new`.

`vcs.utils.diffs.`**`get_udiff`** (*`filenode_old`*, *`filenode_new`*)
Returns unified diff between given `filenode_old` and `filenode_new`.

Helpers

Utitlites aimed to help achieve mostly basic tasks.

`vcs.utils.helpers.`**`get_highlighted_code`** (*name*, *code*, *type*=`'terminal'`)
If pygments are available on the system then returned output is colored. Otherwise unchanged content is returned.

`vcs.utils.helpers.`**`get_scm`** (*path*, *search_recursively=False*)
Returns one of alias from `ALIASES` (in order of precedence same as shortcuts given in ALIASES) and top working dir path for the given argument. If no scm-specific directory is found, `VCSError` is raised unless `search_recursively` is set to `True` - in that case, function would try to find scm starting at given `path` and moving up. If we can't go up any further, `VCSError` is raised.

`vcs.utils.helpers.`**`run_command`** (*cmd*, **args*)
Runs command on the system with given `args`.

Lazy attributes utils

`vcs.utils.lazy.`**`LazyProperty`**
Decorator for easier creation of `property` from potentially expensive to calculate attribute of the class.
Usage:

```
class Foo(object):
    @LazyProperty
    def bar(self):
        print 'Calculating self._bar'
        return 42
```

Taken from http://blog.pythonisito.com/2008/08/lazy-descriptors.html and used widely.

Web

Django extension

Simple VCS

Django pluggable appliation build on top of vcs.

Models

class `vcs.web.simplevcs.models.`**`Repository`** (**args*, ***kwargs*)
Repository(id, alias, path)

class `vcs.web.simplevcs.models.`**`RepositoryInfo`** (**args*, ***kwargs*)
RepositoryInfo(id, repository_id, clone_count, push_count, size)

`get_repo_size ()`
Returns current size of repository directory.

Views

`vcs.web.simplevcs.views.`**`browse_repository`** (*request*, *template_name*, *repository=None*, *repository_path=None*, *repository_alias=None*, *revision=None*, *node_path=''*, *extra_context={}*)
Generic repository browser view.
Required arguments:

- Either `repository` or (`repository_path` *and* `repository_alias` is required
- `template_name`: The full name of a template to use in rendering the page.

Optional arguments:

- `revision`: object with which backend may identify changeset. In example, mercurial backend recognizes *hex* or **short** strings, revision numbers (given as integer or string) or `tip` string. If none specified, most recent changeset (sometimes called `tip`) is taken.
- `node_path`: relative location of the node in the repository (location to file or directory). By default this is an empty string which would retrieve root node from repository.
- `extra_context`: A dictionary of values to add to the template context. By default, this is an empty dictionary.

Template context:
In addition to `extra_context`, the template's context will be:

- `repository`: same what was given or computed from `repository_path` and `repository_alias`
- `changeset`: based on the given `revision` or tip if none given
- `root`: repository's node on the given `node_path`
- `readme_node`: `FileNode` instance if returning `root` is a `DirNode` and a readme file can be found at it's file listing (can be a file which name starts with *README*, with or without any extension, case is irrelevant); if no readme file could be found, None is returned

`vcs.web.simplevcs.views.`**`diff_file`** (*request*, *file_path*, *template_name*, *repository=None*, *repository_path=None*, *repository_alias=None*, *revision_old=None*, *revision_new=None*, *extra_context=None*)
Generic repository browser view showing diff of specified file.
Required arguments:

- Either `repository` or (`repository_path` *and* `repository_alias` is required)
- `file_path`: relative location of the file node in the repository.
- `revision_old`: object identifying changeset at backend.
- `revision_new`: object identifying changeset at backend.

- `template_name`: The full name of a template to use in rendering the page.

Optional arguments:

- `extra_context`: A dictionary of values to add to the template context. By default, this is an empty dictionary.

Template context:

In addition to `extra_context`, the template's context will be:

- `repository`: same what was given or computed from `repository_path` and `repository_alias`
- `file_old`: `FileNode` retrieved by backend for given `revision_old` param
- `file_new`: `FileNode` retrieved by backend for given `revision_new` param
- `diff_content`: unified diff for retrieved `file_old` and `file_new` contents

`vcs.web.simplevcs.views.`**`diff_changeset`** (*`request`*, *`template_name`*, *`repository=None`*, *`repository_path=None`*, *`repository_alias=None`*, *`revision=None`*, *`extra_context=None`*)

Generic repository browser view showing diffs for given revision.

Required arguments:

- Either `repository` or (`repository_path` *and* `repository_alias` is required
- `revision`: object identifying changeset at backend.
- `template_name`: The full name of a template to use in rendering the page.

Optional arguments:

- `extra_context`: A dictionary of values to add to the template context. By default, this is an empty dictionary.

Template context:

In addition to `extra_context`, the template's context will be:

- `repository`: same what was given or computed from `repository_path` and `repository_alias`
- `changeset`: `Changeset` retrieved by backend for given `revision` param; nodes from `added` and `changed` attributes of the changeset have one extra attribute: `diff` which is instance of `vcs.utils.diffs.DiffProcessor`

`vcs.web.simplevcs.views.`**`hgserve`** (*`request`*, *`repo_path`*, *`login_required=True`*, *`auth_callback=None`*)

Returns mimic of mercurial response. Would raise `NotMercurialRequest` if request is not recognized as one comming from mercurial agent.

Parameters:

- **repo_path** -- path to local mercurial repository on which request would be made
- **login_required=True** -- if set to False, would not require user to authenticate at all (with one exception, see note below)

Note

by default `VCS_ALWAYS_REQUIRE_LOGIN` is set to True and if not changed, would cause this function to require authentication from all requests (and `login_required` would *NOT* be checked). If, on the other hand, settings are configured that `VCS_ALWAYS_REQUIRE_LOGIN` is False, then authorization would be required only if login_required is True.

Parameters:

- **auth_callback=None** -- callable function, may be passed only if login_required was True; would be called *after* authorization, with `user` as parameter; may be used i.e. for permission checks

Semi generic views which helps to reponse for requests comming from specific SCM clients.

`vcs.web.simplevcs.views.hg.`**`hgserve`** (*`request`*, *`repo_path`*, *`login_required=True`*, *`auth_callback=None`*)

Returns mimic of mercurial response. Would raise `NotMercurialRequest` if request is not recognized as one comming from mercurial agent.

Parameters:

- **repo_path** -- path to local mercurial repository on which request would be made
- **login_required=True** -- if set to False, would not require user to authenticate at all (with one exception, see note below)

Note

by default `VCS_ALWAYS_REQUIRE_LOGIN` is set to True and if not changed, would cause this function to require authentication from all requests (and `login_required` would *NOT* be checked). If, on the other hand, settings are configured that `VCS_ALWAYS_REQUIRE_LOGIN` is False, then authorization would be required only if login_required is True.

Parameters:

- **auth_callback=None** -- callable function, may be passed only if login_required was True; would be called *after* authorization, with `user` as parameter; may be used i.e. for permission checks

Other topics

- *genindex*
- *search*

Index

Python Module Index