
virtualenvwrapper Documentation

Release 4.7.3.dev7

Doug Hellmann

April 05, 2017

1	Features	3
2	Introduction	5
3	Details	9
3.1	Installation	9
3.2	Command Reference	13
3.3	Customizing Virtualenvwrapper	22
3.4	Project Management	31
3.5	Tips and Tricks	32
3.6	For Developers	34
3.7	Existing Extensions	36
3.8	Why virtualenvwrapper is (Mostly) Not Written In Python	37
3.9	CHANGES	39
3.10	Glossary	69
4	References	71
5	Support	73
5.1	Shell Aliases	73
6	License	75

virtualenvwrapper is a set of extensions to Ian Bicking's [virtualenv](#) tool. The extensions include wrappers for creating and deleting virtual environments and otherwise managing your development workflow, making it easier to work on more than one project at a time without introducing conflicts in their dependencies.

Features

1. Organizes all of your virtual environments in one place.
2. Wrappers for managing your virtual environments (create, delete, copy).
3. Use a single command to switch between environments.
4. Tab completion for commands that take a virtual environment as argument.
5. User-configurable hooks for all operations (see *Per-User Customization*).
6. Plugin system for more creating sharable extensions (see *Extending Virtualenvwrapper*).

Introduction

The best way to explain the features `virtualenvwrapper` gives you is to show it in use.

First, some initialization steps. Most of this only needs to be done one time. You will want to add the command to `source /usr/local/bin/virtualenvwrapper.sh` to your shell startup file, changing the path to `virtualenvwrapper.sh` depending on where it was installed by pip.

```
$ pip install virtualenvwrapper
...
$ export WORKON_HOME=~/.Envs
$ mkdir -p $WORKON_HOME
$ source /usr/local/bin/virtualenvwrapper.sh
$ mkvirtualenv env1
Installing
setuptools.....
.....
.....done.
virtualenvwrapper.user_scripts Creating /Users/dhellmann/Envs/env1/bin/predeactivate
virtualenvwrapper.user_scripts Creating /Users/dhellmann/Envs/env1/bin/postdeactivate
virtualenvwrapper.user_scripts Creating /Users/dhellmann/Envs/env1/bin/preactivate
virtualenvwrapper.user_scripts Creating /Users/dhellmann/Envs/env1/bin/postactivate  New python execu
(env1)$ ls $WORKON_HOME
env1 hook.log
```

Now we can install some software into the environment.

```
(env1)$ pip install django
Downloading/unpacking django
  Downloading Django-1.1.1.tar.gz (5.6Mb): 5.6Mb downloaded
  Running setup.py egg_info for package django
Installing collected packages: django
  Running setup.py install for django
    changing mode of build/scripts-2.6/django-admin.py from 644 to 755
    changing mode of /Users/dhellmann/Envs/env1/bin/django-admin.py to 755
Successfully installed django
```

We can see the new package with `ls sitepackages`:

```
(env1)$ ls sitepackages
Django-1.1.1-py2.6.egg-info      easy-install.pth
setuptools-0.6.10-py2.6.egg    pip-0.6.3-py2.6.egg
django                          setuptools.pth
```

Of course we are not limited to a single `virtualenv`:

```
(env1)$ ls $WORKON_HOME
env1          hook.log
(env1)$ mkvirtualenv env2
Installing setuptools.....
.....
.....done.
virtualenvwrapper.user_scripts Creating /Users/dhellmann/Envs/env2/bin/predeactivate
virtualenvwrapper.user_scripts Creating /Users/dhellmann/Envs/env2/bin/postdeactivate
virtualenvwrapper.user_scripts Creating /Users/dhellmann/Envs/env2/bin/preactivate
virtualenvwrapper.user_scripts Creating /Users/dhellmann/Envs/env2/bin/postactivate  New python execut
(env2)$ ls $WORKON_HOME
env1          env2          hook.log
```

Switch between environments with `workon`:

```
(env2)$ workon env1
(env1)$ echo $VIRTUAL_ENV
/Users/dhellmann/Envs/env1
(env1)$
```

The `workon` command also includes tab completion for the environment names, and invokes customization scripts as an environment is activated or deactivated (see *Per-User Customization*).

```
(env1)$ echo 'cd $VIRTUAL_ENV' >> $WORKON_HOME/postactivate
(env1)$ workon env2
(env2)$ pwd
/Users/dhellmann/Envs/env2
```

postmkvirtualenv is run when a new environment is created, letting you automatically install commonly-used tools.

```
(env2)$ echo 'pip install sphinx' >> $WORKON_HOME/postmkvirtualenv
(env3)$ mkvirtualenv env3
New python executable in env3/bin/python
Installing setuptools.....
.....done.
virtualenvwrapper.user_scripts Creating /Users/dhellmann/Envs/env3/bin/predeactivate
virtualenvwrapper.user_scripts Creating /Users/dhellmann/Envs/env3/bin/postdeactivate
virtualenvwrapper.user_scripts Creating /Users/dhellmann/Envs/env3/bin/preactivate
virtualenvwrapper.user_scripts Creating /Users/dhellmann/Envs/env3/bin/postactivate
Downloading/unpacking sphinx
  Downloading Sphinx-0.6.5.tar.gz (972Kb): 972Kb downloaded
  Running setup.py egg_info for package sphinx
    no previously-included directories found matching 'doc/_build'
Downloading/unpacking Pygments>=0.8 (from sphinx)
  Downloading Pygments-1.3.1.tar.gz (1.1Mb): 1.1Mb downloaded
  Running setup.py egg_info for package Pygments
Downloading/unpacking Jinja2>=2.1 (from sphinx)
  Downloading Jinja2-2.4.tar.gz (688Kb): 688Kb downloaded
  Running setup.py egg_info for package Jinja2
    warning: no previously-included files matching '*' found under directory 'docs/_build/doctrees'
Downloading/unpacking docutils>=0.4 (from sphinx)
  Downloading docutils-0.6.tar.gz (1.4Mb): 1.4Mb downloaded
  Running setup.py egg_info for package docutils
Installing collected packages: docutils, Jinja2, Pygments, sphinx
  Running setup.py install for docutils
  Running setup.py install for Jinja2
```

```
Running setup.py install for Pygments
Running setup.py install for sphinx
  no previously-included directories found matching 'doc/_build'
  Installing sphinx-build script to /Users/dhellmann/Envs/env3/bin
  Installing sphinx-quickstart script to /Users/dhellmann/Envs/env3/bin
  Installing sphinx-autogen script to /Users/dhellmann/Envs/env3/bin
Successfully installed docutils Jinja2 Pygments sphinx (env3)$
(venv3)$ which sphinx-build
/Users/dhellmann/Envs/env3/bin/sphinx-build
```

Through a combination of the existing functions defined by the core package (see *Command Reference*), third-party plugins (see *Extending Virtualenvwrapper*), and user-defined scripts (see *Per-User Customization*) virtualenvwrapper gives you a wide variety of opportunities to automate repetitive operations.

Installation

Supported Shells

virtualenvwrapper is a set of shell *functions* defined in Bourne shell compatible syntax. Its automated tests run under these shells on OS X and Linux:

- bash
- ksh
- zsh

It may work with other shells, so if you find that it does work with a shell not listed here please let me know. If you can modify it to work with another shell without completely rewriting it, then send a pull request through the [bitbucket project page](#). If you write a clone to work with an incompatible shell, let me know and I will link to it from this page.

Windows Command Prompt

David Marble has ported virtualenvwrapper to Windows batch scripts, which can be run under Microsoft Windows Command Prompt. This is also a separately distributed re-implementation. You can download [virtualenvwrapper-win](#) from PyPI.

MSYS

It is possible to use virtualenv wrapper under [MSYS](#) with a native Windows Python installation. In order to make it work, you need to define an extra environment variable named `MSYS_HOME` containing the root path to the MSYS installation.

```
export WORKON_HOME=$HOME/.virtualenvs
export MSYS_HOME=/c/msys/1.0
source /usr/local/bin/virtualenvwrapper.sh
```

or:

```
export WORKON_HOME=$HOME/.virtualenvs
export MSYS_HOME=C:\msys\1.0
source /usr/local/bin/virtualenvwrapper.sh
```

Depending on your MSYS setup, you may need to install the [MSYS mktemp](#) binary in the `MSYS_HOME/bin` folder.

PowerShell

Guillermo López-Anglada has ported virtualenvwrapper to run under Microsoft's PowerShell. We have agreed that since it is not compatible with the rest of the extensions, and is largely a re-implementation (rather than an adaptation), it should be distributed separately. You can download [virtualenvwrapper-powershell](#) from PyPI.

Python Versions

virtualenvwrapper is tested under Python 2.6-3.4.

Basic Installation

virtualenvwrapper should be installed into the same global site-packages area where virtualenv is installed. You may need administrative privileges to do that. The easiest way to install it is using `pip`:

```
$ pip install virtualenvwrapper
```

or:

```
$ sudo pip install virtualenvwrapper
```

Warning: virtualenv lets you create many different Python environments. You should only ever install virtualenv and virtualenvwrapper on your base Python installation (i.e. NOT while a virtualenv is active) so that the same release is shared by all Python environments that depend on it.

An alternative to installing it into the global site-packages is to add it to your user local directory (usually `~/local`).

```
$ pip install --user virtualenvwrapper
```

Shell Startup File

Add three lines to your shell startup file (`.bashrc`, `.profile`, etc.) to set the location where the virtual environments should live, the location of your development project directories, and the location of the script installed with this package:

```
export WORKON_HOME=$HOME/.virtualenvs
export PROJECT_HOME=$HOME/Devel
source /usr/local/bin/virtualenvwrapper.sh
```

After editing it, reload the startup file (e.g., run `source ~/.bashrc`).

Lazy Loading

An alternative initialization script is provided for loading virtualenvwrapper lazily. Instead of sourcing `virtualenvwrapper.sh` directly, use `virtualenvwrapper_lazy.sh`. If `virtualenvwrapper.sh` is not on your `$PATH`, set `VIRTUALENVWRAPPER_SCRIPT` to point to it.

```
export WORKON_HOME=$HOME/.virtualenvs
export PROJECT_HOME=$HOME/Devel
export VIRTUALENVWRAPPER_SCRIPT=/usr/local/bin/virtualenvwrapper.sh
source /usr/local/bin/virtualenvwrapper_lazy.sh
```

Warning: When the lazy-loading version of the startup script is used, tab-completion of arguments to virtualenvwrapper commands (such as environment names) is not enabled until after the first command has been run. For example, tab completion of environments does not work for the first instance of *workon*.

Quick-Start

1. Run: `workon`
2. A list of environments, empty, is printed.
3. Run: `mkvirtualenv temp`
4. A new environment, `temp` is created and activated.
5. Run: `workon`
6. This time, the `temp` environment is included.

Configuration

virtualenvwrapper can be customized by changing environment variables. Set the variables in your shell startup file *before* loading `virtualenvwrapper.sh`.

Location of Environments

The variable `WORKON_HOME` tells virtualenvwrapper where to place your virtual environments. The default is `$HOME/.virtualenvs`. If the directory does not exist when virtualenvwrapper is loaded, it will be created automatically.

Location of Project Directories

The variable `PROJECT_HOME` tells virtualenvwrapper where to place your project working directories. The variable must be set and the directory created before *mkproject* is used.

See also:

- *Project Management*

Project Linkage Filename

The variable `VIRTUALENVWRAPPER_PROJECT_FILENAME` tells virtualenvwrapper how to name the file linking a virtualenv to a project working directory. The default is `.project`.

See also:

- *Project Management*

Enable Project Directory Switching

The variable `VIRTUALENVWRAPPER_WORKON_CD` controls whether the working directory is changed during the post activate phase. The default is 1, to enable changing directories. Set the value to 0 to disable this behavior for all invocations of `workon`.

See also:

- *workon*

Location of Hook Scripts

The variable `VIRTUALENVWRAPPER_HOOK_DIR` tells virtualenvwrapper where the *user-defined hooks* should be placed. The default is `$WORKON_HOME`.

See also:

- *Per-User Customization*

Location of Hook Logs

The variable `VIRTUALENVWRAPPER_LOG_FILE` tells virtualenvwrapper where the logs for the hook loader should be written. The default is to not log from the hooks.

Python Interpreter, virtualenv, and \$PATH

During startup, `virtualenvwrapper.sh` finds the first `python` and `virtualenv` programs on the `$PATH` and remembers them to use later. This eliminates any conflict as the `$PATH` changes, enabling interpreters inside virtual environments where virtualenvwrapper is not installed or where different versions of virtualenv are installed. Because of this behavior, it is important for the `$PATH` to be set **before** sourcing `virtualenvwrapper.sh`. For example:

```
export PATH=/usr/local/bin:$PATH
source /usr/local/bin/virtualenvwrapper.sh
```

To override the `$PATH` search, set the variable `VIRTUALENVWRAPPER_PYTHON` to the full path of the interpreter to use and `VIRTUALENVWRAPPER_VIRTUALENV` to the full path of the `virtualenv` binary to use. Both variables *must* be set before sourcing `virtualenvwrapper.sh`. For example:

```
export VIRTUALENVWRAPPER_PYTHON=/usr/local/bin/python
export VIRTUALENVWRAPPER_VIRTUALENV=/usr/local/bin/virtualenv
source /usr/local/bin/virtualenvwrapper.sh
```

Default Arguments for virtualenv

If the application identified by `VIRTUALENVWRAPPER_VIRTUALENV` needs arguments, they can be set in `VIRTUALENVWRAPPER_VIRTUALENV_ARGS`. The same variable can be used to set default arguments to be passed to `virtualenv`. For example, set the value to `--no-site-packages` to ensure that all new environments are isolated from the system `site-packages` directory.

```
export VIRTUALENVWRAPPER_VIRTUALENV_ARGS='--no-site-packages'
```


Temporary Files

virtualenvwrapper creates temporary files in `$TMPDIR`. If the variable is not set, it uses `/tmp`. To change the location of temporary files just for virtualenvwrapper, set `VIRTUALENVWRAPPER_TMPDIR`.

Site-wide Configuration

Most UNIX systems include the ability to change the configuration for all users. This typically takes one of two forms: editing the *skeleton* files for new accounts or editing the global startup file for a shell.

Editing the skeleton files for new accounts means that each new user will have their private startup files preconfigured to load virtualenvwrapper. They can disable it by commenting out or removing those lines. Refer to the documentation for the shell and operating system to identify the appropriate file to edit.

Modifying the global startup file for a given shell means that all users of that shell will have virtualenvwrapper enabled, and they cannot disable it. Refer to the documentation for the shell to identify the appropriate file to edit.

Upgrading to 2.9

Version 2.9 includes the features previously delivered separately by `virtualenvwrapper.project`. If you have an older version of the project extensions installed, remove them before upgrading.

Upgrading from 1.x

The shell script containing the wrapper functions has been renamed in the 2.x series to reflect the fact that shells other than bash are supported. In your startup file, change `source /usr/local/bin/virtualenvwrapper_bashrc` to `source /usr/local/bin/virtualenvwrapper.sh`.

Command Reference

All of the commands below are to be used on the Terminal command line.

Managing Environments

mkvirtualenv

Create a new environment, in the `WORKON_HOME`.

Syntax:

```
mkvirtualenv [-a project_path] [-i package] [-r requirements_file] [virtualenv options] ENVNAME
```

All command line options except `-a`, `-i`, `-r`, and `-h` are passed directly to `virtualenv`. The new environment is automatically activated after being initialized.

```
$ workon
$ mkvirtualenv mynewenv
New python executable in mynewenv/bin/python
Installing setuptools.....
.....
```

```

.....
done.
(mynewenv)$ workon
mynewenv
(mynewenv)$

```

The `-a` option can be used to associate an existing project directory with the new environment.

The `-i` option can be used to install one or more packages (by repeating the option) after the environment is created.

The `-r` option can be used to specify a text file listing packages to be installed. The argument value is passed to `pip -r` to be installed.

See also:

- [premkvirtualenv](#)
- [postmkvirtualenv](#)
- [requirements file format](#)

mktmpenv

Create a new virtualenv in the `WORKON_HOME` directory.

Syntax:

```
mktmpenv [(-c|--cd) | (-n|--no-cd)] [VIRTUALENV_OPTIONS]
```

A unique virtualenv name is generated.

If `-c` or `--cd` is specified the working directory is changed to the virtualenv directory during the post-activate phase, regardless of the value of `VIRTUALENVWRAPPER_WORKON_CD`.

If `-n` or `--no-cd` is specified the working directory is **not** changed to the virtualenv directory during the post-activate phase, regardless of the value of `VIRTUALENVWRAPPER_WORKON_CD`.

```

$ mktmpenv
Using real prefix '/Library/Frameworks/Python.framework/Versions/2.7'
New python executable in 1e513ac6-616e-4d56-9aa5-9d0a3b305e20/bin/python
Overwriting 1e513ac6-616e-4d56-9aa5-9d0a3b305e20/lib/python2.7/distutils/__init__.py
with new content
Installing setuptools.....
.....done.
This is a temporary environment. It will be deleted when deactivated.
(1e513ac6-616e-4d56-9aa5-9d0a3b305e20) $

```

lsvirtualenv

List all of the environments.

Syntax:

```
lsvirtualenv [-b] [-l] [-h]
```

- b** Brief mode, disables verbose output.
- l** Long mode, enables verbose output. Default.
- h** Print the help for lsvirtualenv.

See also:

- [get_env_details](#)

showvirtualenv

Show the details for a single virtualenv.

Syntax:

```
showvirtualenv [env]
```

See also:

- [get_env_details](#)

rmvirtualenv

Remove an environment, in the WORKON_HOME.

Syntax:

```
rmvirtualenv ENVNAME
```

You must use *deactivate* before removing the current environment.

```
(mynewenv)$ deactivate
$ rmvirtualenv mynewenv
$ workon
$
```

See also:

- [prermvirtualenv](#)
- [postrmvirtualenv](#)

cpvirtualenv

Duplicate an existing virtualenv environment. The source can be an environment managed by virtualenvwrapper or an external environment created elsewhere.

Warning: Copying virtual environments is not well supported. Each virtualenv has path information hard-coded into it, and there may be cases where the copy code does not know it needs to update a particular file. **Use with caution.**

Syntax:

```
cpvirtualenv ENVNAME [TARGETENVNAME]
```

Note: Target environment name is required for WORKON_HOME duplications. However, target environment name can be omitted for importing external environments. If omitted, the new environment is given the same name as the original.

```
$ workon
$ mkvirtualenv source
New python executable in source/bin/python
Installing setuptools.....
.....
done.
(source)$ cpvirtualenv source dest
Making script /Users/dhellmann/Devel/virtualenvwrapper/tmp/dest/bin/easy_install relative
Making script /Users/dhellmann/Devel/virtualenvwrapper/tmp/dest/bin/easy_install-2.6 relative
Making script /Users/dhellmann/Devel/virtualenvwrapper/tmp/dest/bin/pip relative
Script /Users/dhellmann/Devel/virtualenvwrapper/tmp/dest/bin/postactivate cannot be made relative (it
Script /Users/dhellmann/Devel/virtualenvwrapper/tmp/dest/bin/postdeactivate cannot be made relative (it
Script /Users/dhellmann/Devel/virtualenvwrapper/tmp/dest/bin/preactivate cannot be made relative (it
Script /Users/dhellmann/Devel/virtualenvwrapper/tmp/dest/bin/predeactivate cannot be made relative (it
(dest)$ workon
dest
source
(dest)$
```

See also:

- [*precvirtualenv*](#)
- [*postcpvirtualenv*](#)
- [*premkvirtualenv*](#)
- [*postmkvirtualenv*](#)

allvirtualenv

Run a command in all virtualenvs under WORKON_HOME.

Syntax:

```
allvirtualenv command with arguments
```

Each virtualenv is activated, bypassing activation hooks, the current working directory is changed to the current virtualenv, and then the command is run. Commands cannot modify the current shell state, but can modify the virtualenv.

```
$ allvirtualenv pip install -U pip
```

Controlling the Active Environment

workon

List or change working virtual environments

Syntax:

```
workon [(-c|--cd)|(-n|--no-cd)] [environment_name|"."]
```

If no environment_name is given the list of available environments is printed to stdout.

If -c or --cd is specified the working directory is changed to the project directory during the post-activate phase, regardless of the value of VIRTUALENVWRAPPER_WORKON_CD.

If `-n` or `--no-cd` is specified the working directory is **not** changed to the project directory during the post-activate phase, regardless of the value of `VIRTUALENVWRAPPER_WORKON_CD`.

If `"."` is passed as the environment name, the name is derived from the base name of the current working directory (contributed by Matias Saguir).

```
$ workon
$ mkvirtualenv env1
  New python executable in env1/bin/python
Installing setuptools.....
.....
done.
(env1)$ mkvirtualenv env2
New python executable in env2/bin/python
Installing setuptools.....
.....
done.
(env2)$ workon
env1
env2
(env2)$ workon env1
(env1)$ echo $VIRTUAL_ENV
/Users/dhellmann/Devel/virtualenvwrapper/tmp/env1
(env1)$ workon env2
(env2)$ echo $VIRTUAL_ENV
/Users/dhellmann/Devel/virtualenvwrapper/tmp/env2
(env2)$
```

See also:

- [*predeactivate*](#)
- [*postdeactivate*](#)
- [*preactivate*](#)
- [*postactivate*](#)
- [*Enable Project Directory Switching*](#)

deactivate

Switch from a virtual environment to the system-installed version of Python.

Syntax:

```
deactivate
```

Note: This command is actually part of `virtualenv`, but is wrapped to provide before and after hooks, just as `workon` does for `activate`.

```
$ workon
$ echo $VIRTUAL_ENV

$ mkvirtualenv env1
New python executable in env1/bin/python
Installing setuptools.....
```

```
.....  
.....  
done.  
(env1)$ echo $VIRTUAL_ENV  
/Users/dhellmann/Devel/virtualenvwrapper/tmp/env1  
(env1)$ deactivate  
$ echo $VIRTUAL_ENV  
  
$
```

See also:

- *predeactivate*
- *postdeactivate*

Quickly Navigating to a virtualenv

There are two functions to provide shortcuts to navigate into the currently-active virtualenv.

cdvirtualenv

Change the current working directory to \$VIRTUAL_ENV.

Syntax:

```
cdvirtualenv [subdir]
```

Calling `cdvirtualenv` changes the current working directory to the top of the virtualenv (\$VIRTUAL_ENV). An optional argument is appended to the path, allowing navigation directly into a subdirectory.

```
$ mkvirtualenv env1  
New python executable in env1/bin/python  
Installing setuptools.....  
.....  
done.  
(env1)$ echo $VIRTUAL_ENV  
/Users/dhellmann/Devel/virtualenvwrapper/tmp/env1  
(env1)$ cdvirtualenv  
(env1)$ pwd  
/Users/dhellmann/Devel/virtualenvwrapper/tmp/env1  
(env1)$ cdvirtualenv bin  
(env1)$ pwd  
/Users/dhellmann/Devel/virtualenvwrapper/tmp/env1/bin
```

cdsitepackages

Change the current working directory to the site-packages for \$VIRTUAL_ENV.

Syntax:

```
cdsitepackages [subdir]
```

Because the exact path to the site-packages directory in the virtualenv depends on the version of Python, `cdsitepackages` is provided as a shortcut for `cdvirtualenv`

lib/python\${pyvers}/site-packages. An optional argument is also allowed, to specify a directory hierarchy within the site-packages directory to change into.

```
$ mkvirtualenv env1
New python executable in env1/bin/python
Installing setuptools.....
.....
done.
(env1)$ echo $VIRTUAL_ENV
/Users/dhellmann/Devel/virtualenvwrapper/tmp/env1
(env1)$ cd sitepackages PyMOTW/bisect/
(env1)$ pwd
/Users/dhellmann/Devel/virtualenvwrapper/tmp/env1/lib/python2.6/site-packages/PyMOTW/bisect
```

lssitepackages

Calling `lssitepackages` shows the content of the `site-packages` directory of the currently-active `virtualenv`.

Syntax:

```
lssitepackages
```

```
$ mkvirtualenv env1
New python executable in env1/bin/python
Installing setuptools.....
.....
done.
(env1)$ $ workon env1
(env1)$ lssitepackages
setuptools-0.6.10-py2.6.egg      pip-0.6.3-py2.6.egg
easy-install.pth              setuptools.pth
```

Path Management

add2virtualenv

Adds the specified directories to the Python path for the currently-active `virtualenv`.

Syntax:

```
add2virtualenv directory1 directory2 ...
```

Sometimes it is desirable to share installed packages that are not in the system `site-packages` directory and which should not be installed in each `virtualenv`. One possible solution is to symlink the source into the environment `site-packages` directory, but it is also easy to add extra directories to the `PYTHONPATH` by including them in a `.pth` file inside `site-packages` using `add2virtualenv`.

1. Check out the source for a big project, such as Django.
2. Run: `add2virtualenv path_to_source`.
3. Run: `add2virtualenv`.
4. A usage message and list of current “extra” paths is printed.
5. Use option `-d` to remove the added path.

The directory names are added to a path file named `_virtualenv_path_extensions.pth` inside the `site-packages` directory for the environment.

Based on a contribution from James Bennett and Jannis Leidel.

toggleglobalsitepackages

Controls whether the active virtualenv will access the packages in the global Python `site-packages` directory.

Syntax:

```
toggleglobalsitepackages [-q]
```

Outputs the new state of the virtualenv. Use the `-q` switch to turn off all output.

```
$ mkvirtualenv env1
New python executable in env1/bin/python
Installing setuptools.....
.....
done.
(env1)$ toggleglobalsitepackages
Disabled global site-packages
(env1)$ toggleglobalsitepackages
Enabled global site-packages
(env1)$ toggleglobalsitepackages -q
(env1)$
```

Project Directory Management

See also:

Project Management

mkproject

Create a new virtualenv in the `WORKON_HOME` and project directory in `PROJECT_HOME`.

Syntax:

```
mkproject [-f|--force] [-t template] [virtualenv_options] ENVNAME
```

-f, --force Create the virtualenv even if the project directory already exists

The `template` option may be repeated to have several templates used to create a new project. The templates are applied in the order named on the command line. All other options are passed to `mkvirtualenv` to create a virtual environment with the same name as the project.

```
$ mkproject myproj
New python executable in myproj/bin/python
Installing setuptools.....
.....
done.
Creating /Users/dhellmann/Devel/myproj
(myproj)$ pwd
/Users/dhellmann/Devel/myproj
```



```
(myproj)$ echo $VIRTUAL_ENV
/Users/dhellmann/Envs/myproj
(myproj)$
```

See also:

- *premkproject*
- *postmkproject*

setvirtualenvproject

Bind an existing virtualenv to an existing project.

Syntax:

```
setvirtualenvproject [virtualenv_path project_path]
```

The arguments to `setvirtualenvproject` are the full paths to the virtualenv and project directory. An association is made so that when `workon` activates the virtualenv the project is also activated.

```
$ mkproject myproj
New python executable in myproj/bin/python
Installing setuptools.....
.....
done.
Creating /Users/dhellmann/Devel/myproj
(myproj)$ mkvirtualenv myproj_new_libs
New python executable in myproj/bin/python
Installing setuptools.....
.....
done.
Creating /Users/dhellmann/Devel/myproj
(myproj_new_libs)$ setvirtualenvproject $VIRTUAL_ENV $(pwd)
```

When no arguments are given, the current virtualenv and current directory are assumed.

Any number of virtualenvs can refer to the same project directory, making it easy to switch between versions of Python or other dependencies for testing.

cdproject

Change the current working directory to the one specified as the project directory for the active virtualenv.

Syntax:

```
cdproject
```

Managing Installed Packages**wipeenv**

Remove all of the installed third-party packages in the current virtualenv.

Syntax:

wipeenv

Other Commands

virtualenvwrapper

Print a list of commands and their descriptions as basic help output.

Syntax:

```
virtualenvwrapper
```

Customizing Virtualenvwrapper

virtualenvwrapper adds several hook points you can use to change your settings, shell environment, or other configuration values when creating, deleting, or moving between environments. These hooks are exposed in two ways. *Per-User Customization* allows a user to perform generic actions for every virtualenv in your environment, including customization of virtualenv creation. *Extending Virtualenvwrapper* makes it possible to share common behaviors between systems and developers.

Per-User Customization

The end-user customization scripts are either *sourced* (allowing them to modify your shell environment) or *run* as an external program at the appropriate trigger time.

The global scripts applied to all environments should be placed in the directory named by `VIRTUALENVWRAPPER_HOOK_DIR`, which by default will be equal to `WORKON_HOME`. The local scripts should be placed in the `bin` directory of the virtualenv.

Example Usage

As a Django developer, you likely want `DJANGO_SETTINGS_MODULE` to be set, and if you work on multiple projects, you want it to be specific to the project you are currently working on. Wouldn't it be nice if it was set based on the active virtualenv? You can achieve this with *Per-User Customization* as follows.

If your `WORKON_HOME` is set to `~/virtualenvs`:

```
vim ~/virtualenvs/premkvirtualenv
```

Edit the file so it contains the following (for a default Django setup):

```
# Automatically set django settings for the virtualenv echo "export
DJANGO_SETTINGS_MODULE=${1.settings}" >> "$1/bin/activate"
```

Create a new virtualenv, and you should see `DJANGO_SETTINGS_MODULE` in your env!

get_env_details

Global/Local both

Argument(s) env name

Sourced/Run run

`$VIRTUALENVWRAPPER_HOOK_DIR/get_env_details` is run when `workon` is run with no arguments and a list of the virtual environments is printed. The hook is run once for each environment, after the name is printed, and can print additional information about that environment.

initialize

Global/Local global

Argument(s) None

Sourced/Run sourced

`$VIRTUALENVWRAPPER_HOOK_DIR/initialize` is sourced when `virtualenvwrapper.sh` is loaded into your environment. Use it to adjust global settings when `virtualenvwrapper` is enabled.

premkvirtualenv

Global/Local global

Argument(s) name of new environment

Sourced/Run run

`$VIRTUALENVWRAPPER_HOOK_DIR/premkvirtualenv` is run as an external program after the virtual environment is created but before the current environment is switched to point to the new env. The current working directory for the script is `$WORKON_HOME` and the name of the new environment is passed as an argument to the script.

postmkvirtualenv

Global/Local global

Argument(s) none

Sourced/Run sourced

`$VIRTUALENVWRAPPER_HOOK_DIR/postmkvirtualenv` is sourced after the new environment is created and activated. If the `-a <project_path>` flag was used, the link to the project directory is set up before this script is sourced.

precpvirtualenv

Global/Local global

Argument(s) name of original environment, name of new environment

Sourced/Run run

`$VIRTUALENVWRAPPER_HOOK_DIR/precpvirtualenv` is run as an external program after the source environment is duplicated and made relocatable, but before the `premkvirtualenv` hook is run or the current environment is switched to point to the new env. The current working directory for the script is `$WORKON_HOME` and the names of the source and new environments are passed as arguments to the script.

postcpvirtualenv

Global/Local global

Argument(s) none

Sourced/Run sourced

`$VIRTUAL_ENVWRAPPER_HOOK_DIR/postcpvirtualenv` is sourced after the new environment is created and activated.

preactivate

Global/Local global, local

Argument(s) environment name

Sourced/Run run

The global `$VIRTUAL_ENVWRAPPER_HOOK_DIR/preactivate` script is run before the new environment is enabled. The environment name is passed as the first argument.

The local `$VIRTUAL_ENV/bin/preactivate` hook is run before the new environment is enabled. The environment name is passed as the first argument.

postactivate

Global/Local global, local

Argument(s) none

Sourced/Run sourced

The global `$VIRTUAL_ENVWRAPPER_HOOK_DIR/postactivate` script is sourced after the new environment is enabled. `$VIRTUAL_ENV` refers to the new environment at the time the script runs.

This example script adds a space between the virtual environment name and your old PS1 by making use of `_OLD_VIRTUAL_PS1`.

```
PS1="( `basename \"$VIRTUAL_ENV\"` ) $_OLD_VIRTUAL_PS1"
```

The local `$VIRTUAL_ENV/bin/postactivate` script is sourced after the new environment is enabled. `$VIRTUAL_ENV` refers to the new environment at the time the script runs.

This example script for the PyMOTW environment changes the current working directory and the `PATH` variable to refer to the source tree containing the PyMOTW source.

```
pymotw_root=/Users/dhellmann/Documents/PyMOTW
cd $pymotw_root
PATH=$pymotw_root/bin:$PATH
```

predeactivate

Global/Local local, global

Argument(s) none

Sourced/Run sourced

The local `$VIRTUAL_ENV/bin/predeactivate` script is sourced before the current environment is deactivated, and can be used to disable or clear settings in your environment. `$VIRTUAL_ENV` refers to the old environment at the time the script runs.

The global `$VIRTUALENVWRAPPER_HOOK_DIR/predeactivate` script is sourced before the current environment is deactivated. `$VIRTUAL_ENV` refers to the old environment at the time the script runs.

postdeactivate

Global/Local local, global

Argument(s) none

Sourced/Run sourced

The `$VIRTUAL_ENV/bin/postdeactivate` script is sourced after the current environment is deactivated, and can be used to disable or clear settings in your environment. The path to the environment just deactivated is available in `$VIRTUALENVWRAPPER_LAST_VIRTUAL_ENV`.

prermvirtualenv

Global/Local global

Argument(s) environment name

Sourced/Run run

The `$VIRTUALENVWRAPPER_HOOK_DIR/prermvirtualenv` script is run as an external program before the environment is removed. The full path to the environment directory is passed as an argument to the script.

postrmvirtualenv

Global/Local global

Argument(s) environment name

Sourced/Run run

The `$VIRTUALENVWRAPPER_HOOK_DIR/postrmvirtualenv` script is run as an external program after the environment is removed. The full path to the environment directory is passed as an argument to the script.

premkproject

Global/Local global

Argument(s) name of new project

Sourced/Run run

`$WORKON_HOME/premkproject` is run as an external program after the virtual environment is created and after the current environment is switched to point to the new env, but before the new project directory is created. The current working directory for the script is `$PROJECT_HOME` and the name of the new project is passed as an argument to the script.

postmkproject

Global/Local global

Argument(s) none

Sourced/Run sourced

`$WORKON_HOME/postmkproject` is sourced after the new environment and project directories are created and the virtualenv is activated. The current working directory is the project directory.

Extending Virtualenvwrapper

Long experience with home-grown solutions for customizing a development environment has proven how valuable it can be to have the ability to automate common tasks and eliminate persistent annoyances. Carpenters build jigs, software developers write shell scripts. `virtualenvwrapper` continues the tradition of encouraging a craftsman to modify their tools to work the way they want, rather than the other way around.

There are two ways to attach your code so that `virtualenvwrapper` will run it: End-users can use shell scripts or other programs for personal customization, e.g. automatically performing an action on every new virtualenv (see *Per-User Customization*). Extensions can also be implemented in Python by using `Setuptools` entry points, making it possible to share common behaviors between systems and developers.

Use the hooks provided to eliminate repetitive manual operations and streamline your development workflow. For example, set up the `pre_activate` and `post_activate` hooks to trigger an IDE to load a project file to reload files from the last editing session, manage time-tracking records, or start and stop development versions of an application server. Use the `initialize` hook to add entirely new commands and hooks to `virtualenvwrapper`. And the `pre_mkvirtualenv` and `post_mkvirtualenv` hooks give you an opportunity to install basic requirements into each new development environment, initialize a source code control repository, or otherwise set up a new project.

Defining an Extension

Note: `Virtualenvwrapper` is delivered with a plugin for creating and running the user customization scripts (`user_scripts`). The examples below are taken from the implementation of that plugin.

Code Organization

The Python package for `virtualenvwrapper` is a *namespace package*. That means multiple libraries can install code into the package, even if they are not distributed together or installed into the same directory. Extensions can (optionally) use the `virtualenvwrapper` namespace by setting up their source tree like:

- `virtualenvwrapper/`
 - `__init__.py`
 - `user_scripts.py`

And placing the following code in `__init__.py`:

```
"""virtualenvwrapper module
"""
```

```
__import__('pkg_resources').declare_namespace(__name__)
```

Note: Extensions can be loaded from any package, so using the `virtualenvwrapper` namespace is not required.

Extension API

After the package is established, the next step is to create a module to hold the extension code. For example, `virtualenvwrapper/user_scripts.py`. The module should contain the actual extension entry points. Supporting code can be included, or imported from elsewhere using standard Python code organization techniques.

The API is the same for every extension point. Each uses a Python function that takes a single argument, a list of strings passed to the hook loader on the command line.

```
def function_name(args):
    # args is a list of strings passed to the hook loader
```

The contents of the argument list are defined for each extension point below (see *Extension Points*).

Extension Invocation

Direct Action Plugins can attach to each hook in two different ways. The default is to have a function run and do some work directly. For example, the `initialize()` function for the user scripts plugin creates default user scripts when `virtualenvwrapper.sh` is loaded.

```
def initialize(args):
    for filename, comment in GLOBAL_HOOKS:
        make_hook(os.path.join('$WORKON_HOME', filename), comment)
    return
```

Modifying the User Environment There are cases where the extension needs to update the user's environment (e.g., changing the current working directory or setting environment variables). Modifications to the user environment must be made within the user's current shell, and cannot be run in a separate process. To have code run in the user's shell process, extensions can define hook functions to return the text of the shell statements to be executed. These *source* hooks are run after the regular hooks with the same name, and should not do any work of their own.

The `initialize_source()` hook for the user scripts plugin looks for a global initialize script and causes it to be run in the current shell process.

```
def initialize_source(args):
    return """
#
# Run user-provided scripts
#
[ -f "$WORKON_HOME/initialize" ] && source "$WORKON_HOME/initialize"
"""
```

Warning: Because the extension is modifying the user's working shell, care must be taken not to corrupt the environment by overwriting existing variable values unexpectedly. Avoid creating temporary variables where possible, and use unique names where variables cannot be avoided. Prefixing variables with the extension name is a good way to manage the namespace. For example, instead of `temp_file` use `user_scripts_temp_file`. Use `unset` to release temporary variable names when they are no longer needed.

Warning: `virtualenvwrapper` works under several shells with slightly different syntax (`bash`, `sh`, `zsh`, `ksh`). Take this portability into account when defining source hooks. Sticking to the simplest possible syntax usually avoids problems, but there may be cases where examining the `SHELL` environment variable to generate different syntax for each case is the only way to achieve the desired result.

Registering Entry Points

The functions defined in the plugin need to be registered as *entry points* in order for virtualenvwrapper's hook loader to find them. Entry points are configured in the `setup.py` (or `setup.cfg` when using `pbr`) for your package by mapping the entry point name to the function in the package that implements it.

This partial copy of virtualenvwrapper's `setup.cfg` illustrates how the `initialize()` and `initialize_source()` entry points are configured.

```
virtualenvwrapper.initialize =
    user_scripts = virtualenvwrapper.user_scripts:initialize
    project = virtualenvwrapper.project:initialize
virtualenvwrapper.initialize_source =
    user_scripts = virtualenvwrapper.user_scripts:initialize_source
virtualenvwrapper.pre_mkvirtualenv =
    user_scripts = virtualenvwrapper.user_scripts:pre_mkvirtualenv
virtualenvwrapper.post_mkvirtualenv_source =
    user_scripts = virtualenvwrapper.user_scripts:post_mkvirtualenv_source
virtualenvwrapper.pre_cpvirtualenv =
    user_scripts = virtualenvwrapper.user_scripts:pre_cpvirtualenv
virtualenvwrapper.post_cpvirtualenv_source =
    user_scripts = virtualenvwrapper.user_scripts:post_cpvirtualenv_source
virtualenvwrapper.pre_rmvirtualenv =
    user_scripts = virtualenvwrapper.user_scripts:pre_rmvirtualenv
virtualenvwrapper.post_rmvirtualenv =
    user_scripts = virtualenvwrapper.user_scripts:post_rmvirtualenv
virtualenvwrapper.project.pre_mkproject =
    project = virtualenvwrapper.project:pre_mkproject
virtualenvwrapper.project.post_mkproject_source =
    project = virtualenvwrapper.project:post_mkproject_source
virtualenvwrapper.pre_activate =
    user_scripts = virtualenvwrapper.user_scripts:pre_activate
virtualenvwrapper.post_activate_source =
    project = virtualenvwrapper.project:post_activate_source
    user_scripts = virtualenvwrapper.user_scripts:post_activate_source
virtualenvwrapper.pre_deactivate_source =
    user_scripts = virtualenvwrapper.user_scripts:pre_deactivate_source
virtualenvwrapper.post_deactivate_source =
    user_scripts = virtualenvwrapper.user_scripts:post_deactivate_source
virtualenvwrapper.get_env_details =
    user_scripts = virtualenvwrapper.user_scripts:get_env_details

[pbr]
warnerrors = true

[wheel]
universal = true

[build_sphinx]
source-dir = docs/source
build-dir = docs/build
all_files = 1
```

The `entry_points` section maps the *group names* to lists of entry point specifiers. A different group name is defined by virtualenvwrapper for each extension point (see [Extension Points](#)).

The entry point specifiers are strings with the syntax `name = package.module:function`. By convention, the *name* of each entry point is the plugin name, but that is not required (the names are not used).

See also:

- [namespace packages](#)
- [Extensible Applications and Frameworks](#)

The Hook Loader

Extensions are run through a command line application implemented in `virtualenvwrapper.hook_loader`. Because `virtualenvwrapper.sh` is the primary caller and users do not typically need to run the app directly, no separate script is installed. Instead, to run the application, use the `-m` option to the interpreter:

```
$ python -m virtualenvwrapper.hook_loader -h
Usage: virtualenvwrapper.hook_loader [options] <hook> [<arguments>]
```

Manage hooks for virtualenvwrapper

Options:

```
-h, --help          show this help message and exit
-s, --source        Print the shell commands to be run in the current
                   shell
-l, --list          Print a list of the plugins available for the given
                   hook
-v, --verbose       Show more information on the console
-q, --quiet         Show less information on the console
-n NAMES, --name=NAMES
                   Only run the hook from the named plugin
```

To run the extensions for the initialize hook:

```
$ python -m virtualenvwrapper.hook_loader -v initialize
```

To get the shell commands for the initialize hook:

```
$ python -m virtualenvwrapper.hook_loader --source initialize
```

In practice, rather than invoking the hook loader directly it is more convenient to use the shell function, `virtualenvwrapper_run_hook` to run the hooks in both modes.:

```
$ virtualenvwrapper_run_hook initialize
```

All of the arguments given to shell function are passed directly to the hook loader.

Logging

The hook loader configures logging so that messages are written to `$WORKON_HOME/hook.log`. Messages also may be written to `stderr`, depending on the verbosity flag. The default is for messages at *info* or higher levels to be written to `stderr`, and *debug* or higher to go to the log file. Using logging in this way provides a convenient mechanism for users to control the verbosity of extensions.

To use logging from within your extension, simply instantiate a logger and call its `info()`, `debug()` and other methods with the messages.

```
import logging
log = logging.getLogger(__name__)

def pre_mkvirtualenv(args):
```

```
log.debug('pre_mkvirtualenv %s', str(args))
# ...
```

See also:

- [Standard library documentation for logging](#)
- [PyMOTW for logging](#)

Extension Points

The extension point names for native plugins follow a naming convention with several parts: `virtualenvwrapper.(pre|post)_<event>[_source]`. The `<event>` is the action taken by the user or `virtualenvwrapper` that triggers the extension. `(pre|post)` indicates whether to call the extension before or after the event. The suffix `_source` is added for extensions that return shell code instead of taking action directly (see *Modifying the User Environment*).

get_env_details

The `virtualenvwrapper.get_env_details` hooks are run when `workon` is run with no arguments and a list of the virtual environments is printed. The hook is run once for each environment, after the name is printed, and can be used to show additional information about that environment.

initialize

The `virtualenvwrapper.initialize` hooks are run each time `virtualenvwrapper.sh` is loaded into the user's environment. The initialize hook can be used to install templates for configuration files or otherwise prepare the system for proper plugin operation.

pre_mkvirtualenv

The `virtualenvwrapper.pre_mkvirtualenv` hooks are run after the virtual environment is created, but before the new environment is activated. The current working directory for when the hook is run is `$WORKON_HOME` and the name of the new environment is passed as an argument.

post_mkvirtualenv

The `virtualenvwrapper.post_mkvirtualenv` hooks are run after a new virtual environment is created and activated. `$VIRTUAL_ENV` is set to point to the new environment.

pre_activate

The `virtualenvwrapper.pre_activate` hooks are run just before an environment is enabled. The environment name is passed as the first argument.

post_activate

The `virtualenvwrapper.post_activate` hooks are run just after an environment is enabled. `$VIRTUAL_ENV` is set to point to the current environment.

pre_deactivate

The `virtualenvwrapper.pre_deactivate` hooks are run just before an environment is disabled. `$VIRTUAL_ENV` is set to point to the current environment.

post_deactivate

The `virtualenvwrapper.post_deactivate` hooks are run just after an environment is disabled. The name of the environment just deactivated is passed as the first argument.

pre_rmvirtualenv

The `virtualenvwrapper.pre_rmvirtualenv` hooks are run just before an environment is deleted. The name of the environment being deleted is passed as the first argument.

post_rmvirtualenv

The `virtualenvwrapper.post_rmvirtualenv` hooks are run just after an environment is deleted. The name of the environment being deleted is passed as the first argument.

Adding New Extension Points

Plugins that define new operations can also define new extension points. No setup needs to be done to allow the hook loader to find the extensions; documenting the names and adding calls to `virtualenvwrapper_run_hook` is sufficient to cause them to be invoked.

The hook loader assumes all extension point names start with `virtualenvwrapper.` and new plugins will want to use their own namespace qualifier to append to that. For example, the `project` extension defines new events around creating project directories (pre and post). These are called `virtualenvwrapper.project.pre_mkproject` and `virtualenvwrapper.project.post_mkproject`. These are invoked with:

```
virtualenvwrapper_run_hook project.pre_mkproject $project_name
```

and:

```
virtualenvwrapper_run_hook project.post_mkproject
```

respectively.

Project Management

A *project directory* is associated with a `virtualenv`, but usually contains the source code under active development rather than the installed components needed to support the development. For example, the project directory may contain the source code checked out from a version control system, temporary artifacts created by testing, experimental files not committed to version control, etc.

A project directory is created and bound to a `virtualenv` when `mkproject` is run instead of `mkvirtualenv`. To bind an existing project directory to a `virtualenv`, use `setvirtualenvproject`.

Using Templates

A new project directory can be created empty, or populated using one or more *template* plugins. Templates should be specified as arguments to *mkproject*. Multiple values can be provided to apply more than one template. For example, to check out a Mercurial repository from a project on bitbucket and create a new Django site, combine the *bitbucket* and *django* templates.

```
$ mkproject -t bitbucket -t django my_site
```

See also:

- *Templates*
- *Location of Project Directories*
- *Project Linkage Filename*
- *Enable Project Directory Switching*

Tips and Tricks

This is a list of user-contributed tips for making virtualenv and virtualenvwrapper even more useful. If you have tip to share, drop me an email or post a comment on [this blog post](#) and I'll add it here.

zsh Prompt

From Nat (was blogger.com/profile/16779944428406910187):

Using zsh, I added some bits to `$WORKON_HOME/post (de) activate` to show the active virtualenv on the right side of my screen instead.

in `postactivate`:

```
PS1="$ _OLD_VIRTUAL_PS1"  
_OLD_RPROMPT="$RPROMPT"  
RPROMPT="%${fg_bold[white]}% (env: %${fg[green]}% `basename \"$VIRTUAL_ENV\"`%${fg_bold[white]}%)
```

and in `postdeactivate`:

```
RPROMPT="$ _OLD_RPROMPT"
```

Adjust colors according to your own personal tastes or environment.

Updating cached `$PATH` entries

From Nat (was blogger.com/profile/16779944428406910187):

I also added the command `'rehash'` to `$WORKON_HOME/postactivate` and `$WORKON_HOME/postdeactivate` as I was having some problems with zsh not picking up the new paths immediately.

Creating Project Work Directories

Via James:

In the `postmkvirtualenv` script I have the following to create a directory based on the project name, add that directory to the `python` path and then `cd` into it:

```
proj_name=$(basename $VIRTUAL_ENV)
mkdir $HOME/projects/$proj_name
add2virtualenv $HOME/projects/$proj_name
cd $HOME/projects/$proj_name
```

In the `postactivate` script I have it set to automatically change to the project directory when I use the `workon` command:

```
proj_name=$(basename $VIRTUAL_ENV)
cd ~/projects/$proj_name
```

Automatically Run `workon` When Entering a Directory

Justin Abrahms posted about some code he added to his shell environment to look at the directory each time he runs `cd`. If it finds a `.venv` file, it activates the environment named within. On leaving that directory, the current `virtualenv` is automatically deactivated.

Harry Marr wrote a similar function that works with `git` repositories.

Installing Common Tools Automatically in New Environments

Via rizumu (was rizumu.myopenid.com):

I have this `postmkvirtualenv` to install the get a basic setup.

```
$ cat postmkvirtualenv
#!/usr/bin/env bash
curl -O http://python-distribute.org/distribute_setup.p... />python distribute_setup.py
rm distribute_setup.py
easy_install pip==dev
pip install Mercurial
```

Then I have a `pip` requirement file with my dev tools.

```
$ cat developer_requirements.txt
ipdb
ipython
pastescript
nose
http://douglatornell.ca/software/python/Nosy-1.0.tar.gz
coverage
sphinx
grin
pyflakes
pep8
```

Then each project has it's own `pip` requirement file for things like `PIL`, `psycopg2`, `django-apps`, `numpy`, etc.

Changing the Default Behavior of `cd`

Via *mae*:

This is supposed to be executed after `workon`, that is as a `postactivate` hook. It basically overrides `cd` to know about the VENV so instead of doing `cd` to go to `~` you will go to the `venv` root, IMO very handy and I can't live without it anymore. If you pass it a proper path then it will do the right thing.

```
cd () {
    if (( $# == 0 ))
    then
        builtin cd $VIRTUAL_ENV
    else
        builtin cd "$@"
    fi
}

cd
```

And to finally restore the default behaviour of `cd` once you bailout of a VENV via a `deactivate` command, you need to add this as a `postdeactivate` hook:

```
unset -f cd
```

Clean up environments on exit

Via *Michael*:

When you use a temporary `virtualenv` via `mktmpenv` or if you have a `post_deactivate` hook, you have to actually run `deactivate` to clean up the temporary environment or run the hook, respectively. It's easy to forget and just exit the shell. Put the following in `~/ .bash_logout` (or your shell's equivalent file) to always deactivate environments before exiting the shell:

```
[ "$VIRTUAL_ENV" ] && deactivate
```

For Developers

If you would like to contribute to `virtualenvwrapper` directly, these instructions should help you get started. Patches, bug reports, and feature requests are all welcome through the [BitBucket site](#). Contributions in the form of patches or pull requests are easier to integrate and will receive priority attention.

Note: Before contributing new features to `virtualenvwrapper` core, please consider whether they should be implemented as an extension instead.

Building Documentation

The documentation for `virtualenvwrapper` is written in `reStructuredText` and converted to HTML using `Sphinx`. The build itself is driven by `make`. You will need the following packages in order to build the docs:

- `Sphinx`
- `docutils`
- `sphinxcontrib-bitbucket`

Once all of the tools are installed into a virtualenv using `pip`, run `make html` to generate the HTML version of the documentation:

```
$ make html
rm -rf virtualenvwrapper/docs
(cd docs && make html SPHINXOPTS="-c sphinx/pkg")
sphinx-build -b html -d build/doctrees -c sphinx/pkg source build/html
Running Sphinx v0.6.4
loading pickled environment... done
building [html]: targets for 2 source files that are out of date
updating environment: 0 added, 2 changed, 0 removed
reading sources... [ 50%] command_ref
reading sources... [100%] developers

looking for now-outdated files... none found
pickling environment... done
checking consistency... done
preparing documents... done
writing output... [ 33%] command_ref
writing output... [ 66%] developers
writing output... [100%] index

writing additional files... search
copying static files... WARNING: static directory '/Users/dhellmann/Devel/virtualenvwrapper/plugins/'
done
dumping search index... done
dumping object inventory... done
build succeeded, 1 warning.

Build finished. The HTML pages are in build/html.
cp -r docs/build/html virtualenvwrapper/docs
```

The output version of the documentation ends up in `./virtualenvwrapper/docs` inside your sandbox.

Running Tests

The test suite for `virtualenvwrapper` uses `shunit2` and `tox`. The `shunit2` source is included in the `tests` directory, but `tox` must be installed separately (`pip install tox`).

To run the tests under `bash`, `zsh`, and `ksh` for Python 2.4 through 2.7, run `tox` from the top level directory of the hg repository.

To run individual test scripts, use a command like:

```
$ tox tests/test_cd.sh
```

To run tests under a single version of Python, specify the appropriate environment when running `tox`:

```
$ tox -e py27
```

Combine the two modes to run specific tests with a single version of Python:

```
$ tox -e py27 tests/test_cd.sh
```

Add new tests by modifying an existing file or creating new script in the `tests` directory.

Creating a New Template

virtualenvwrapper.project templates work like virtualenvwrapper plugins. The *entry point* group name is `virtualenvwrapper.project.template`. Configure your entry point to refer to a function that will **run** (source hooks are not supported for templates).

The argument to the template function is the name of the project being created. The current working directory is the directory created to hold the project files (`$PROJECT_HOME/$envname`).

Help Text

One difference between project templates and other virtualenvwrapper extensions is that only the templates specified by the user are run. The `mkproject` command has a help option to give the user a list of the available templates. The names are taken from the registered entry point names, and the descriptions are taken from the docstrings for the template functions.

Existing Extensions

Below is a list of some of the extensions available for use with virtualenvwrapper.

emacs-desktop

Emacs `desktop-mode` lets you save the state of emacs (open buffers, kill rings, buffer positions, etc.) between sessions. It can also be used as a project file similar to other IDEs. The `emacs-desktop` plugin adds a trigger to save the current desktop file and load a new one when activating a new virtualenv using `workon`.

user_scripts

The `user_scripts` extension is delivered with virtualenvwrapper and enabled by default. It implements the user customization script features described in *Per-User Customization*.

vim-virtualenv

`vim-virtualenv` is Jeremy Cantrell's plugin for controlling virtualenvs from within vim. When used together with virtualenvwrapper, `vim-virtualenv` identifies the virtualenv to activate based on the name of the file being edited.

Templates

Below is a list of some of the templates available for use with `mkproject`.

bitbucket

The `bitbucket` extension automatically clones a mercurial repository from the specified bitbucket project.

django

The `django` extension automatically creates a new Django project.

SublimeText

Song Jin has created a template plugin for automatically generating the project files used by SublimeText. See the `sublime_projectfile_maker` page for details.

See also:

- *Creating a New Template*

Why virtualenvwrapper is (Mostly) Not Written In Python

If you look at the source code for `virtualenvwrapper` you will see that most of the interesting parts are implemented as shell functions in `virtualenvwrapper.sh`. The hook loader is a Python app, but doesn't do much to manage the `virtualenvs`. Some of the most frequently asked questions about `virtualenvwrapper` are “Why didn't you write this as a set of Python programs?” or “Have you thought about rewriting it in Python?” For a long time these questions baffled me, because it was always obvious to me that it had to be implemented as it is. But they come up frequently enough that I feel the need to explain.

tl;dr: POSIX Made Me Do It

The choice of implementation language for `virtualenvwrapper` was made for pragmatic, rather than philosophical, reasons. The wrapper commands need to modify the state and environment of the user's *current shell process*, and the only way to do that is to have the commands run *inside that shell*. That resulted in me writing `virtualenvwrapper` as a set of shell functions, rather than separate shell scripts or even Python programs.

Where Do POSIX Processes Come From?

New POSIX processes are created when an existing process invokes the `fork()` system call. The invoking process becomes the “parent” of the new “child” process, and the child is a full clone of the parent. The *semantic* result of `fork()` is that an entire new copy of the parent process is created. In practice, optimizations are normally made to avoid copying more memory than is absolutely necessary (frequently via a copy-on-write system). But for the purposes of this explanation it is sufficient to think of the child as a full replica of the parent.

The important parts of the parent process that are copied include dynamic memory (the stack and heap), static stuff (the program code), resources like open file descriptors, and the *environment variables* exported from the parent process. Inheriting environment variables is a fundamental aspect of the way POSIX programs pass state and configuration information to one another. A parent can establish a series of `name=value` pairs, which are then given to the child process. The child can access them through functions like `getenv()`, `setenv()` (and in Python through `os.environ`).

The choice of the term *inherit* to describe the way the variables and their contents are passed from parent to child is significant. Although a child can change its own environment, it cannot directly change the environment settings of its parent because there is no system call to modify the parental environment settings.

How the Shell Runs a Program

When a shell receives a command to be executed, either interactively or by parsing a script file, and determines that the command is implemented in a separate program file, it uses `fork()` to create a new process and then inside that process it uses one of the `exec` functions to start the specified program. The language that program is written in doesn't make any difference in the decision about whether or not to `fork()`, so even if the “program” is a shell script written in the language understood by the current shell, a new process is created.

On the other hand, if the shell decides that the command is a *function*, then it looks at the definition and invokes it directly. Shell functions are made up of other commands, some of which may result in child processes being created, but the function itself runs in the original shell process and can therefore modify its state, for example by changing the working directory or the values of variables.

It is possible to force the shell to run a script directly, and not in a child process, by *sourcing* it. The `source` command causes the shell to read the file and interpret it in the current process. Again, as with functions, the contents of the file may cause child processes to be spawned, but there is not a second shell process interpreting the series of commands.

What Does This Mean for virtualenvwrapper?

The original and most important features of virtualenvwrapper are automatically activating a virtualenv when it is created by `mkvirtualenv` and using `workon` to deactivate one environment and activate another. Making these features work drove the implementation decisions for the other parts of virtualenvwrapper, too.

Environments are activated interactively by sourcing `bin/activate` inside the virtualenv. The `activate` script does a few things, but the important parts are setting the `VIRTUAL_ENV` variable and modifying the shell's search path through the `PATH` variable to put the `bin` directory for the environment on the front of the path. Changing the path means that the programs installed in the environment, especially the python interpreter there, are found before other programs with the same name.

Simply running `bin/activate`, without using `source` doesn't work because it sets up the environment of the *child* process, without affecting the parent. In order to source the activate script in the interactive shell, both `mkvirtualenv` and `workon` also need to be run in that shell process.

Why Choose One When You Can Have Both?

The hook loader is one part of virtualenvwrapper that *is* written in Python. Why? Again, because it was easier. Hooks are discovered using `setuptools` entry points, because after an entry point is installed the user doesn't have to take any other action to allow the loader to discover and use it. It's easy to imagine writing a hook to create new files on the filesystem (by installing a package, instantiating a template, etc.).

How, then, do hooks running in a separate process (the Python interpreter) modify the shell environment to set variables or change the working directory? They cheat, of course.

Each hook point defined by virtualenvwrapper actually represents two hooks. First, the hooks meant to be run in Python are executed. Then the "source" hooks are run, and they *print out* a series of shell commands. All of those commands are collected, saved to a temporary file, and then the shell is told to source the file.

Starting up the hook loader turns out to be way more expensive than most of the other actions virtualenvwrapper takes, though, so I am considering making its use optional. Most users customize the hooks by using shell scripts (either globally or in the virtualenv). Finding and running those can be handled by the shell quite easily.

Implications for Cross-Shell Compatibility

Other than requests for a full-Python implementation, the other most common request is to support additional shells. `fish` comes up a lot, as do various Windows-only shells. The officially *Supported Shells* all have a common enough syntax that the same implementation works for each. Supporting other shells would require rewriting much, if not all, of the logic using an alternate syntax – those other shells are basically different programming languages. So far I have dealt with the ports by encouraging other developers to handle them, and then trying to link to and otherwise promote the results.

Not As Bad As It Seems

Although there are some special challenges created by the the requirement that the commands run in a user's interactive shell (see the many bugs reported by users who alias common commands like `rm` and `cd`), using the shell as a programming language holds up quite well. The shells are designed to make finding and executing other programs easy, and especially to make it easy to combine a series of smaller programs to perform more complicated operations. As that's what virtualenvwrapper is doing, it's a natural fit.

See also:

- [Advanced Programming in the UNIX Environment](#) by W. Richard Stevens & Stephen A. Rago
- [Fork \(operating system\)](#) on Wikipedia
- [Environment variable](#) on Wikipedia
- [Linux implementation of fork\(\)](#)

CHANGES

- Merged in [erickmk/virtualenvwrapper/erickmk/command_refrst-edited-online-with-bitbuc-1491225971803](#) (pull request #58)
- Update sentence to make it more clear
- `command_ref.rst` edited online with Bitbucket
- Merged in [sambrightman/virtualenvwrapper](#) (pull request #52)
- Fix spelling mistake in error message
- Merged in [lonetwin/virtualenvwrapper](#) (pull request #48)

4.7.2

- Baseline testing to python27
- Fixing naming in tests
- Merged in [phd/virtualenvwrapper](#) (pull request #46)
- Ignore `*.pyo` byte-code files
- Fix docs: fix URLs whenever possible, change protocol to https
- Add `wipeenv` and `allvirtualenv` for lazy loading
- Remove one-time functions from the environment
- Fix the problem with lazy completion for bash
- Last set of docs
- Docs fixes
- Updating to virtualenvwrapper
- Last set of docs
- Docs fixes
- Merged in [fix/issue-282-link-to-virtualenvwrapper](#) (pull request #49)

- Updating to virtualenvwrapper
- Unset previously defined cd function rather than redefine it
- Merged in ismailsunni/virtualenvwrapper/ismailsunni/command_refrst-edited-online-with-bitbucket-1454377958615 (pull request #44)
- command_ref.rst edited online with Bitbucket Adding -d for remove extra path
- use a ref instead of hard-coded link in new tip
- Merged in kojiromike/virtualenvwrapper/deactivate-on-logout-tip (pull request #43)
- Add Deactivate-on-Logout Tip
- update REAMDE with new bug tracker URL
- more dir fixes for El Capitan
- add testing for python 3.5
- temporary dir fixes for OS X El Capitan (10.11)
- update to work with tox 2.1.1
- Merged in jveatch/virtualenvwrapper/fix-py26-logging (pull request #41)
- Pass stream as arg rather than kwarg to avoid py26 conflict. Fixes issue #274. StreamHandler arg was named strm in python 2.6
- enhance verbose output of hook loader
- Merged in erilem/virtualenvwrapper/user-scheme-installation (pull request #38)
- Change install docs to use --user

4.7.0

- Merged in gnawbybol/virtualenvwrapper/support_MINGW64 (pull request #36)
- Detect MSYS if MSYSTEM is MINGW64
- Merged in kdeldycke/virtualenvwrapper/kdeldycke/restore-overridden-cd-command-to-its-def-1435073839852 (pull request #34)
- Restore overridden cd command to its default builtin behaviour

4.6.0

- remove some explicit tox environments
- Merged in jessamynsmith/virtualenvwrapper/py34 (pull request #30)
- quiet some of the lsvirtualenv tests
- add test for previous patch
- Merged in robsonpeixoto/virtualenvwrapper/bug/265 (pull request #33)
- Removes empty when list all virtualenvs
- Merged in justinabrahms/virtualenvwrapper/justinabrahms/update-links-and-name-for-venv-post-1431982402822 (pull request #32)
- Update links and name for venv post

- Added testing and updated docs for python 3.4
- Merged in jessamynsmith/virtualenvwrapper/env_with_space (pull request #28)
- Changes as per code review
- Added tests to verify that cpvirtualenv, lsvirtualenv, and mkproject work with spaces in env names
- Made rmvirtualenv work with spaces
- Added tests for leading spaces (trailing spaces don't work in Linux, so don't test them)
- Made lsvirtualenv and allvirtualenv work with spaces in env names
- Made cd command work with space in virtualenv name
- Fixed ordering in asserts for workon tests
- Made workon fully support virtualenvs with spaces in names
- fix default for VIRTUALENVWRAPPER_WORKON_CD

4.5.0

- Add -c/-n options to mktmpenv
- update mktmpenv test to assert changed directory
- Add test for creating venv with space in name

4.4.1

- Touch temporary file after a name is created
- document 'workon .' and give attribution
- Support "workon ."
- fix pep8 error
- make cd after workon optional
- fix sphinx build
- Merged in hjwp/virtualenvwrapper (pull request #25)
- Stop mangling the python argument to virtualenv
- ignore -f lines in pip freeze output
- Merged in bittner/virtualenvwrapper (pull request #22)
- hacked attempt to get round MSYS_HOME environ dependency on windows/git-bash/msys
- Change "distribute" to "setuptools" in docs
- Merged in jessamynsmith/virtualenvwrapper (pull request #23)
- Override tox's desire to install pre-releases
- Reworded the documentation around user scripts vs plugin creation, to make it more clear which one you need. Also added a simple example of user scripts
- do not install distribute in test environments
- Correct spelling of "Bitbucket"

- Update issue tracker URL

4.3.2

- build universal wheels
- Merged in [das_g/virtualenvwrapper/das_g/removed-gratuitous-preposition-1413208408920](#) (pull request #19)
- removed gratuitous preposition
- Fix test invocation for zsh
- add -q option to cd for zsh
- make run_tests use the SHELL var to run test script

4.3.1

- pep8 and test updates for previous commit
- Make postmkproject use VIRTUALENVWRAPPER_HOOK_DIR
- Tell tox it is ok to run shells not installed in the virtualenv
- Set VIRTUALENVWRAPPER_SCRIPT correctly for different shells
- Merged in [nishikar/virtualenvwrapper](#) (pull request #14)
- changed phrasing of environment not found message
- Add tests for wipenv with editable packages
- Remove obsolete information about pip environment vars
- Replace manually maintained history with ChangeLog
- Update doc build to fail if there are warnings

4.3

- remove announce.rst; move to blogging repository
- Merged in [erikb85/virtualenvwrapper/erikb85/run-user-postactivate-after-changing-dir-1401272364804](#) (pull request #15)
- Run User Postactivate after changing dirs
- add link to sublimetext extension
- moved environment exists check below active environment check
- added no such environment prompt to rmvirtualenv if it does not exist
- updated pep8
- clean up script mode changes
- forgotten comment
- trailing whitespace removed
- tabs expanded; mode difference
- changed comments and mode for sourced scripts

- ignore bin, include, lib
- mode constant for sourced-only files
- do not specify a version for pbr
- Merged in mjbrooks/virtualenvwrapper (pull request #12)
- use VIRTUALENVWRAPPER_ENV_BIN_DIR throughout
- Extract basic help text from the script
- Add list of commands as basic help output
- update author email
- clean up comment about zsh behavior in lazy
- Fix syntax error (empty *if* block)

4.2

- update docs for 4.2 release
- update history for previous change; fix syntax issue in previous change
- update history
- Do not create hooks for rmproject
- make setvirtualenvproject honor relative paths
- Ensure hook directory exists
- fix indentation in virtualenvwrapper_lazy.sh
- use valid syntax for creating tmpdir under linux
- stop python 3.2 tests
- stop using distribute for packaging the test templates
- fix merge issue from previous commit
- Fix mkvirtualenv -a relative paths
- minor: tabs to spaces
- Fix zsh crash caused by lazily loading the completions
- Fix hint in error message, when virtualenvwrapper_run_hook failed
- changed spelling of proj_name calculation
- Fix *which* with virtualenvwrapper_lazy.sh
- use virtualenvwrapper_cd in project plugin
- document new force option in history
- Merged in claymcclure/virtualenvwrapper (pull request #2)
- update history for doc fix from dirn
- Merged in dirn/virtualenvwrapper/dirn/fix-documentation-for-allvirtualenv-the-1375587964876 (pull request #4)
- update history for cd command fix

- consolidate 'ls' tests
- update test to handle change easy_install
- ignore any egg directories created while packaging
- add tests to make sure we override cd properly
- Merged in isbadawi/virtualenvwrapper (pull request #5)
- Always use virtualenvwrapper_cd instead of cd
- Fix documentation for allvirtualenv
- Document *mkproject -force* usage
- Mention sphinxcontrib-bitbucket requirement
- Merged in mrdbtr/virtualenvwrapper (pull request #3)
- add tmp- prefix to temporary envs
- Preserve quoting for allvirtualenv command arguments
- Add *mkproject -force* option
- Remove extraneous punctuation

4.1.1

- update history for 4.1.1
- Merged in mordred/virtualenvwrapper (pull request #1)
- Take advantage of pbr 0.5.19
- Working on packaging issue with 4.1 release

4.1

- prep for 4.1 release
- fix pep8 issue in user_scripts.py
- quiet cdproject test
- one more parallel test issue
- use pbr for packaging
- Allow tests to run in parallel
- Fix virtualenv detection with spaces in WORKON_HOME
- add license file
- Fix problem lsvirtualenv after previous commit
- Add allvirtualenv command
- Ensure that -p and -python options are consistent
- quiet tests
- add test for mkvirtualenv w/ site-packages
- ignore emacs TAGS file

- Provide a way to extend the lazy-loader
- Add wipeenv command
- Update ignore file
- remove trailing whitespace in tox.ini
- Quote paths
- Skip pushd/popd test under ksh
- Run the cdproject test in a subshell
- Show more details when running under zsh
- add doc explaining implementation choices
- add a warning to cpvirtualenv command docs
- fix rst in announcement file
- fix home page url
- add python 3.3 classifier
- Added tag 4.0 for changeset 2ba65a13f804

4.0

- Prepare for 4.0 release
- Update Python 3 compatibility
- assume setuptools is available during the installation
- update tested-under version lists
- add attribution for previous fix to the history file
- Correct script name in error message
- reorg test runner to remove redundant test runs
- flake8 fixes for setup.py
- Prep 3.7.1 release
- Make `-python` option to `mkvirtualenv` not sticky
- Fix project template listing when none installed
- note change in the history file
- better prefix and fix for other help functions
- prevent `workon_help` from polluting the global namespace
- Fixed broken screencast link
- Merged in dasevilla/virtualenvwrapper/link-fix (pull request #33)
- Update link to requirements docs
- Added tag 3.7 for changeset 303ff1485acb

3.7

- update version number
- Apply style to sphinx config file
- add link to flake8 in history
- use flake8 instead of pep8 for style checking
- Turn off logging by default
- Add help option to workon
- Add `-help` option to mkproject
- merge readme filename change
- Merged in jeffbyrnes/virtualenvwrapper (pull request #32)
- merge Add complete-time load to lazy loader
- Merged in upsuper/virtualenvwrapper (pull request #29)
- fix issue with toggleglobalsitepackages tests that was hidden by old test virtualenv
- show which virtualenv is used in tests
- do not check in test output
- Use `$_VIRTUALENVWRAPPER_API` instead of listing functions
- merge exclusion rules for doc build artifacts
- Added tag 3.6.1 for changeset c180ccae77b4

3.6.1

- prepare 3.6.1 release
- Rename READMEs to be RST
- Added exclusion for docs/en, docs/es, and docs/ja to `.hgignore`
- Add complete-time load to lazy loader
- Fix link to setvirtualenvproject command
- merge fix for relative python interpreter option to mkvirtualenv
- Replace `realpath` with a more portable way of converting a relative path to an absolute path
- Fix typo in documentation
- Fix `-python` switch for virtualenv
- fix markup typo in announcement
- Added tag 3.6 for changeset 002a0ccdcc7a

3.6

- update version number before release, 2
- update version number before release

- fix pep8 issues with setup.py
- fix pep8 issues with sphinx conf file
- Fix virtualenvwrapper_show_workon_options under zsh with chpwd
- update history for previous change
- Update documentation to point to the real file where add2virtualenv command adds directories to PYTHON-PATH
- update the links to the translated versions of the documentation
- change to the default theme for readthedocs.org
- move es and ja versions of docs to their own repositories
- add attribution to history file for previous patch
- fix issue with add2virtualenv and noclobber setting in shell; fixes #137
- pep8 cleanup
- fix lazy-loader function definitions under zsh; fixes #144
- use the right virtualenv binary to get help; fixes #148
- convert hook loader to use stevedore
- fix reference in announcement
- Added tag 3.5 for changeset c93b81815391

3.5

- bump version number and update announcement text
- fix whitespace and rename a few worker functions to be consistent with the rest
- document previous changes
- Use “command” to avoid aliases or functions that mask common utilities. fixes #119
- quiet some test operations and check for error codes before continuing
- allow the caller to control which shells are used for tests; unset variables that might be inherited and give the wrong idea about what the current shell is for a test; export SHELL to point to the current shell
- add test for lazy loading via workon; addresses #144
- update docs with link to virtualenvwrapper-win port; fixes #140
- clean up cpvirtualenv documentation
- if cpvirtualenv fails to create the target directory, return an error code
- document cpvirtualenv addition
- merged upstream
- Forgot to uncommit the remove workon_home in teardown
- update README with supported python versions
- Did not mean to commit isitepackages
- Update cpvirtualenv utilizing virtualenv-clone and allowing for external virutalenvs to be added to WORKON_HOME

- fix xref endpoint used in install.rst
- Added tag 3.4 for changeset 07905d9135ac

3.4

- bump version
- update announcement
- clarify warning on tab completion
- add lazy loader
- move error reporting for bad python interpreter closer to where the error occurs
- Invoke the initialization hooks directly when testing for error with Python
- hide error messages
- fix section heading in announce blog post so the version number does not appear twice
- update announcement file for 3.3 release
- fix the requirement name
- remove old copy of requirements file
- add requirements file to try readthedocs again
- Added tag 3.3 for changeset 45877370548e

3.3

- prepare 3.3 release
- attribution for previous merge
- Merged in agriffis/virtualenvwrapper (pull request #22)
- clean up RST formatting
- attribution for previous merge
- Merged in barberj/vew/fix_installing_requirements_after_cd (pull request #21)
- Use spaces for indentation consistently instead of mixed spaces/tabs. No functional changes
- Quoting arguments to expandpath to allow for spaces in the arguments
- Update to get fully qualified path of requirements in case a directory change occurs before pip is called
- Clean up the temporary file in the virtualenvwrapper_run_hook error returns
- attribution for previous merge
- Merged in agriffis/virtualenvwrapper (pull request #20)
- Fix error handling in virtualenvwrapper_tempfile; the typeset builtin will return success even if the command-substitution fails, so put them on separate lines
- catch -help option to mkvirtualenv; fixes #136
- Remove the trap from virtualenvwrapper_tempfile; the function is called in a command substitution, so the trap fires immediately to remove the file. There are ways to accomplish this, but they're complex and the caller is already explicitly rm'ing the file

- attribution for merging pull request 17
- merge in hook listing and pep8 fixes
- pep8 changes
- Merged in bwanamarko/virtualenvwrapper (pull request #17)
- print the list of core hooks if no hook name is given in list mode
- attribution for previous merges
- Check that required test shells are available ahead of running tests. This avoids accidentally running tests with /bin/sh (dash) on Debian, which eventually deletes the ~/.virtualenvs directory. (Whoops.)
- Enforce running run_tests under tox by setting/checking an env var
- another fix for msys users * using lssitepackages * keep \$site_packages in quotes in case of spaces
- fix bug for MSYS users - makes several folders, fails on shell startup * if \$WORKON_HOME not defined, or folder missing, then when mkdir called must * pass \$WORKON_HOME in double-quotes "\$WORKON_HOME" because there might be * spaces that will be interpreted separately * e.g. C:Documents and Settings.virtualenv makes 3 folders: * "C:Documents", "~/and" & "~/Settings/.virtualenv"
- update shell function virtualenvwrapper_get_site_packages_dir * let MSYS users use lssitepackages & cdsitepackages * replace \$VIRTUAL_ENV/bin with \$VIRTUAL_ENV/\$VIRTUALENVWRAPPER_ENV_BIN_DIR
- attribution for documentation work
- reset the default language
- revised the Japanese translation in plugins.rst
- revised the Japanese translation in index.rst
- merged the changes (r369:550) in extensions.rst
- merged the changes (r369:550) for Japanese translation in projects.rst
- merged the changes (r369:550) for Japanese translation in extensions.rst
- merged the changes (r369:550) for Japanese translation in developers.rst
- merged the changes (r369:550) for Japanese translation in tips.rst
- merged the changes (r369:550) for Japanese translation in scripts.rst
- merged the changes (r369:550) for Japanese translation
- merged the changes (r369:550) for Japanese translation
- changed LANGUAGE settings "en" to "ja"
- merged from original
- add attribution to history file for ralphbean's changes
- merge in permission changes from ralphbean
- Bypass the test for missing virtualenv if the user has it installed to the subset of the path needed for the shunit2 framework to function properly. Add a test for having VIRTUALENVWRAPPER_VIRTUALENV set to a program that does not exist
- Removed shebangs from scripts non-executable site-packages files
- Removed execution bit on virtualenvwrapper.sh
- update announcement blog post for 3.2

- Added tag 3.2 for changeset dccf1a1abf4e

3.2

- bump version number
- Add a link target name for the rmvirtualenv command
- Use distutils to get the site-packages directory. Fixes #112
- more global test header cleanup
- Centralize setup of variables for tests. Change WORKON_HOME and PROJECT_HOME for tests to make them unique across runs, allowing simultaneous test runs in different sandboxes
- update history for previous merge
- Merged in ciberglo/virtualenvwrapper (pull request #13)
- add history details about license classification change
- Merged in ralphbean/virtualenvwrapper (pull request #14)
- attribution for previous commit
- Fix typo in documentation reported by Nick Martin
- Changed trove classifiers from BSD to MIT (like the README indicates.)
- add test for removing several environments
- changing rmvirtualenv message: Erasing -> Removing
- support to remove several environments at once
- remove blank spaces
- use typeset instead of local and provide attribution for the original fix
- Make project_dir local so it doesn't clobber other variables
- Added tag 3.1 for changeset ebbb3ba81687

3.1

- prepare release 3.1
- quote the path as we are editing the pth file; fixes #132
- update history file for previous change
- associate project before enabling the new virtualenv; fixes #122
- add tags to announce.rst
- add a couple of debugging lines to the generated scripts
- Added tag 3.0.1 for changeset 14cf7e58d321

3.0.1

- package release 3.0.1; fixes #126
- Add test files to the sdist package. Addresses #126
- Remove /usr/bin since apparently there are times when virtualenv is installed there due to vendor packages. Fixes #127
- Added tag 3.0 for changeset 434b87ebc24a

3.0

- fix version info in trove classifiers, take 2
- fix version info in trove classifiers
- use the version of python in the current virtualenv to install the template project into the tox virtualenv during the test
- merge in support for python 3.2
- bump version number, update history, prepare announcement
- remove redundant test
- use the version of python in the virtualenv instead of depending on the PATH
- use packages available for python 3
- use packages that can be installed under python 3 to test the -i option to mkvirtualenv
- get the output in a way that makes it work properly with grep
- include virtualenv in the test dependencies
- fix shell expression to get the python version
- fix indentation
- py3k compatibility
- py3k compatibility
- py3k compatibility
- Added tag 2.11.1 for changeset 12a1e0b65313

2.11.1

- update history and version number for bug release
- Skipping re-initialization in subshells breaks tab completion, so go ahead and take the performance hit. Closes #121
- quiet some tests
- announcement for 2.11 release
- Added tag 2.11 for changeset ff4d492c873c

2.11

- bump version number for release
- add VIRTUALENVWRAPPER_PROJECT_FILENAME; resolves issue 120
- make log files group writable; resolves #62
- shortcut initialization if it has run before
- Remove support for Python 2.4 and 2.5. Update tests to work with virtualenv 1.7, where `--no-site-packages` is now the default
- Add note about `-a` option to history file and clarify its description in the docs a bit
- documentation for `-a <project_path>` flag
- test for `'mkvirtualenv -a <project_path>'`
- add `-a project_path` to `mkvirtualenv` usage summary
- associate a project with a venv at creation
- fix link to Justin Lily's helper post
- Added tag 2.10.1 for changeset 9e10c201a500

2.10.1

- bump version to 2.10.1; closes #114
- improve test for `mktmpenv` with options; addresses #114
- change `mktmpenv` to always create an env name for the user; addresses #114
- update announcement text
- bump version number
- document previous fix in history
- strip spaces from template names; fixes #111
- fix template listing for python 2.4, which does not support the `-m` option with namespace packages
- if `uuid` is not available, use `random` to generate a name for the new environment
- Use old style string formatting instead of the `format` method to retain python 2.4 and 2.5 support
- add test to ensure templates are applied correctly
- get the version number from the packaging scripts
- add `mktmpenv` command from `virtualenvwrapper.tmpenv`
- add `-i` option to `mkvirtualenv`
- more test quieting
- quiet tests and add intermediate check for delete
- fix use of `sed` in `add2virtualenv` to be more portable
- quiet test
- Merged in `miracle2k/virtualenvwrapper` (pull request #6)
- merge in linux changes

- ignore temporary files created by editor
- variable name changes and other cleanup so the script does not bomb under ksh on ubuntu 11.04
- run each test script in every shell before moving to the next script
- Make add2virtualenv tests work again, add new test code for new features
- Update lssitepackages to work with new.pth filename
- Merged upstream
- add link to changelog in readme
- fix version number in history
- update announcement file
- Upgrade instructions
- Clean up help functions. Add documentation for new -r option to mkvirtualenv
- Add -r option to mkvirtualenv to install base requirements after the environment is created. Fix argument processing in mkproject so the correct template names are preserved
- merge virtualenvwrapper.project features into virtualenvwrapper
- convert function definition format so typeset works under ksh
- Merged upstream
- add link to powershell port
- Added tag 2.8 for changeset 279244c0fa41

2.8

- set version in history and update announcement
- Added tag 2.8 for changeset 7e0abe005937
- bump version number
- merge in patches from noirbizarre to add support for MSYS environment; clean up doc addition; fix resulting problem is lsvirtualenv
- Identify another --no-site-packages test and add one for cpvirtualenv using the default args variable; addresses #102
- add test for --no-site-packages flag after cpvirtualenv; addresses #102
- Escape uses of cd in case it is aliased. addresses #101
- add a test to verify pushd/popd behavior; addresses #101
- Set is_msys to False when not in MSYS shell
- Avoid declaring the 'command_exists' function for a one shot use
- Replaced all remaining 'bin' occurrences by \$VIRTUALENVWRAPPER_ENV_BIN_DIR
- Use VIRTUALENVWRAPPER_VIRTUALENV in cpvirtualenv. fixes #104
- Merged in sharat87/virtualenvwrapper (pull request #1)
- Update documentation about mktemp
- VIRTUALENVWRAPPER_VIRTUALENV_ARGS not working with >1 args on zsh

- User scripts should be called based on new \$VIRTUALENVWRAPPER_ENV_BIN_DIR variable
- add some debugging and a test to try to reproduce problem with log directory variable; addresses #95
- move tab completion initialization; expand support for tab completion in zsh (fixes #97)
- Added support for getopts with fallback on getopt
- Improved variable name: VIRTUALENVWRAPPER_ENV_BIN_DIR instead of script_folder and is_msys instead of msys
- Document MSys installation
- Allow Win32 and Unix paths for MSYS_HOME variable
- Added msys paths support
- update announce file
- Added tag 2.7.1 for changeset b20cf787d8e1

2.7.1

- bump version number for bug release
- set log dir and hook dir variables after WORKON_HOME is set; fixes #94
- link to documentation about installing into user directory
- further installation doc clarification
- add a warning about installing into a virtualenv
- clarify instructions for running tests; fixes #92
- report an error if there are no test scripts
- Added tag 2.7 for changeset ea378ef00313

2.7

- update version and draft announcement
- add grep fix to history
- remove -e option from all calls to grep for better portability; fixes #85
- nicer titles for configuration section
- reorg install docs to separate the customization stuff; add some comments about site-wide installation; fixes #87
- make it possible to remove a virtualenv while inside it; fixes #83
- pass VIRTUALENVWRAPPER_VIRTUALENV_ARGS when calling VIRTUALENVWRAPPER_VIRTUALENV; fixes #89; fixes #87
- add link to vim-virtualenv
- enable tab completion for showvirtualenv; fixes #78
- clean up test instructions for developers; fixes #75
- clear configuration variables before running tests
- fix typo in cpvirtualenv; fixes #71

- Add VIRTUALENVWRAPPER_LOG_DIR variable
- Use VIRTUALENVWRAPPER_HOOK_DIR to control where the hooks are defined
- doc updates for VIRTUALENVWRAPPER_VIRTUALENV
- fix tests to work under ksh on ubuntu 10.10 by using alternate syntax for capturing messages sent to stderr
- fix tempdir tests to work on ubuntu 10.10
- merge pmclanahan's test changes and toggleglobalsitepackages
- Add attribution for recent patches to the history file
- fix tests for changes to virtualenvwrapper_verify_workon_home
- suppress hook loader messages in tests
- change verbosity level when creating hook scripts so the messages can be suppressed in tests
- Added docs for the toggleglobalsitepackages command
- Added "toggleglobalsitepackages" command. Added tests for the new command
- Modified the test runner to reliably use the intended shells
- fix arg handling for lsvirtualenv under zsh - fixes issue #86
- remove the custom functions from the sphinx config, since rtd does not support them
- trying readthedocs again
- ignore .orig files created by hg
- fix lsvirtualenv to read args in zsh
- remove the download url since I upload packages to pypi now
- translated 2.6.2/2.6.3 history into Japanese
- fixes issue 79 by enclosing WORKON_HOME in quotes
- merged from original
- Added tag 2.6.3 for changeset 246ce68795ea

2.6.3

- tweak history
- Added tag 2.6.3 for changeset e7582879df06
- more doc build changes
- add upload target
- Added tag 2.6.2 for changeset 625d85d3136f

2.6.2

- fix doc build for readthedocs.org
- add test for space in WORKON_HOME to address #79
- add a test to verify that when virtualenv fails to create an environment the hook scripts are not run. see #76
- merged a few fixes and updated history

- update history
- merge in japanese translation of documentation, with a few markup fixes; disable spelling extension until there is a python 2.7 installer for it
- add spelling extension
- Added Japanese translation for the documentation Added to make html/website for the Japanese documentation Added the Japanese documentation link in original English index.rst
- restore download url
- Added tag 2.6.1 for changeset 445a58d5a05a

2.6.1

- version 2.6.1
- fixes issue #73 by changing virtualenvwrapper_get_python_version to only include the major and minor numbers
- add supported version info to readme so it appears on pypi page
- Added tag 2.6 for changeset b0f27c65fa64

2.6

- bump version to 2.6 and document updates
- avoid specifying text mode when creating hook scripts (fixes #68)
- closes #70 by adding a list of supported shells and python versions to documentation and trove classifiers
- fix #60 by setting install_requires instead of requires
- change the way we determine the python version
- convert test scripts to use tox instead of home-grown multi-version system in the Makefile
- create the WORKON_HOME dir if it doesn't exist
- fix platforms definition so upload to pypi will work
- Added tag 2.5.3 for changeset dc74f106d8d2

2.5.3

- point release before uploading sdist
- Added tag 2.5.2 for changeset f71ffbb996c4

2.5.2

- Make lsvirtualenv work under zsh using patch from Zach Voase. Fixes #64
- Added tag 2.5.1 for changeset 2ab678413a29

2.5.1

- fix workon to list in brief mode
- Added tag 2.5 for changeset 80e2fcda77ac

2.5

- bump version
- add docs for showvirtualenv
- add showvirtualenv and re-implement lsvirtualenv with it
- Added tag 2.4 for changeset a85d80e88996

2.4

- tweak history file
- Added tag 2.4 for changeset 64f858d461d4
- add lsvirtualenv command with -l option
- Added tag 2.3 for changeset b9d4591458bb

2.3

- add get_env_details hook
- Added tag 2.2.2 for changeset 266a166f80da

2.2.2

- bump version to 2.2.2
- check exit code of virtualenv before proceeding (fixes #56)
- use single quotes around regex with \$ (see #55)
- update history with changes (see #57)
- escape more commands (see #57)
- incorporate patch from fredpalmer to escape grep calls (fixes #57)
- Added tag 2.2.1 for changeset 87d60f20a715

2.2.1

- fix #50 by escaping rm before calling it
- Added tag 2.2.1 for changeset 66a89d019905
- bump version to 2.2.1
- convert path deriving code in startup of script to function so it is easier to test
- escape dollar sign in regex to resolve #53

- add tests for GREP_OPTIONS problem (ref #51)
- unset GREP_OPTIONS before to use grep
- add support and bug tracker link to readme and docs
- ignore missing files in trap cleanup (see #38)
- address #37 with wording change in docs
- update history
- address issue #46 by escaping the calls to which
- Added tag 2.2 for changeset d5c5faecc92d

2.2

- bump version number
- more test refinements
- add trap to remove temporary file, see #38
- more tempfile fixes
- changes to make the tests run on my linux host
- mention changes to address ticket 35 in history
- addresses ticket 35 by adding debugging instrumentation
- since we always use the same config dir, set it once
- unify sphinx config files
- use the sphinxcontrib.bitbucket extension for links to the issues and changesets in history.rst
- update history with recent changes
- fix tests; clean up contributed changes
- Fixing a bug in the call to mktemp
- Some cleanup after talking with dhellmann
- First pass at speeding things up by making fewer calls into Python. Needs review
- review for text added by Doug about the translation
- show python version in test progress messages
- fix #44 by updating the tests to run with python 2.7b1
- fix #43 by switching the way the hook loader is run
- Added tag 2.1.1 for changeset 7540fc7d8e63

2.1.1

- setting up for a release
- fix #42 by quieting the errors/warnings
- fix #41 by using the cached python where the wrappers are installed
- fix formatting of seealso block

- link to Manuel's home page instead of just the translation
- add link back to english docs
- add attribution for Manuel
- add link from english to spanish docs; update history
- shift output directory for html build so the sdist package looks nicer
- merge in spanish translation
- another attempt to address #35
- added italic to deactivation
- announce translation
- english paragraph removed
- README translated
- first revision
- index revision
- markup fix
- aspell to plugins and fix some paragraphs
- aspell for script
- aspell to install
- aspell to index
- aspell to hooks
- aspell for extensions
- aspell for developers
- aspell for command_ref
- another paragraph
- almost done for plugins.rst
- continue the translation
- continue the translation of plugins.rst
- remove the option that copy the static files: we don't have file to copy and it generate a WARNING in the sphinx compilation
- markup fixed
- remove translation from the toctree
- scripts.rst translated to spanish
- remove old version of translations.rst, we don't need this file anymore
- I don't think that we need to translate the ChangeLog
- extensions.rst translated
- tips.rst translated
- rst markup fixed

- typo fixed on english documentation
- fix the Makefile to generate the website documentation for 'en' and 'es' languages
- reorder the documents files in docs/LANGUAGE folders and modify the rules in the Makefile to build the documentation
- merge from Doug commit. Added the base.html template to make the website documentation
- Makefile modified to build "es" documentation
- put the base template in the repository
- developers.rst translated
- typo fixed
- continue plugins.rst translation
- Fix typo found by humitos
- starting with "Defining an Extension"
- start to translate plugins.rst
- hooks translated
- translations in the index page
- added some translated topics
- added the translation for install.rst
- index.rst translated to spanish
- fixed the right bug :)
- update announcement for 2.1
- add emacs directive to readme
- Added tag 2.1 for changeset 241df6c36860

2.1

- bump version
- rotate log file when it grows too big
- do not include website html in sdist
- do not include html docs inside virtualenvwrapper dir to avoid conflicts with other packages using that namespace
- fix mkvirtualenv -h
- doc updates
- add references to new extensions
- add -n and -l options to hook loader
- update docs with examples
- handle empty workon_home dir properly
- support nondestructive argument to deactivate
- include a date value in the filename

- fix #34 by using python's tempfile module instead of a shell command
- add hooks for cpvirtualenv; make deactivate work better under ksh
- Update docs for mkvirtualenv to fix #30
- fix #33 with improved installation instructions and a better error message
- use tempfile to create temporary files instead of the process id so the filenames are less predictable
- update contributing info
- add attribution for research work for ksh port
- add support for ksh (fixes #25)
- copy dist file to desktop after building
- Added tag 2.0.2 for changeset 6a51a81454ae

2.0.2

- update version and history
- fix #32 by removing use of 'with' ; add tests for python 2.6 and 2.5
- sort ignore lines and add build directory
- Added tag 2.0.1 for changeset 91e1124c6831

2.0.1

- update version and history
- add documentation about temp files
- fix #29 by checking TMPDIR and using a default if no value is found
- save draft of email for announcing new releases on python-announce
- Added tag 2.0 for changeset 54713c4552c2

2.0

- fix install dir for web docs
- Added tag 2.0 for changeset 485e1999adf0
- move todo list out of hg repo
- add namespace package declaration
- include more motivational background
- add help to Makefile
- merge 2.0 changes into tip
- status update
- even more doc cleanup
- doc restructuring

- remove redundant 'source' from cli
- more doc cleanup
- more doc cleanup
- update extension entry point docs
- move make_hooks functionality into user_scripts, since they are related
- start overhauling doc content
- test cleanup and enhancement
- add VIRTUALENVWRAPPER_LAST_VIRTUAL_ENV variable for postdeactivate scripts
- use the user's current shell as the default interpreter in the hook script
- quiet hook loader
- minor doc updates and formatting changes
- comment out debug logging
- all existing tests are passing again
- convert more hooks; stop running tests when we see a failure or error
- implement initialize hooks
- start implementing hook loader and a couple of sample hooks
- rename wrapper script
- add register rule
- update installation test
- don't need pavement.py any more
- add rules for updating website
- set version in Makefile before building html
- more tasks
- add test rules
- start moving from paver back to make and distribute
- reorg todo list
- add todo list and design notes for hook scripts
- Added tag 1.27 for changeset d64869519c2e

1.27

- add explicit check for virtualenv in the test
- Added tag 1.27 for changeset 3edf5f224815
- bump version; pre-release code cleanup
- add note about relocatable side-effect
- undo merge, tests moved to separate files
- touch up tests

- flush formatting prints
- quiet tests
- ignore build files created by tests
- added test that copied virtualenv exists
- resolve conflict on tests dispatch
- added script to setup.py
- add testpackage setup.py
- Added tag 1.26 for changeset 51eef82a39d4

1.26

- preparing version 1.26 for release
- fix #26 by quieting the error message during init and only showing it when an action is explicitly taken by the user
- remove directories likely to contain a site-wide virtualenv installation and hide the error message because we expect mkvirtualenv to fail
- break up the tests to make it easier to run only part of them
- run all tests on all shells
- Added tag 1.25 for changeset 06229877a640

1.25

- add cdsitepackages arg handling from William McVey
- Added test for cdsitepackages with argument
- Updated with expanded capability of cdsitepackages to cd to a subdir
- Added tab completion and pathname argument handling to cdsitepackages
- I didn't know about 'sed -i', makes this a lot easier
- When echoing the current list of paths, do not include the 'import' lines
- Test for existence of path file was broken, used the wrong test
- New -d option to 'add2virtualenv' which allows removal of a path previously added
- Make sure that paths added via 'add2virtualenv' always end up being listed *before* regularly installed packages in sys.path. This ensures that you can always use the command to replace an installed package with a out-of-virtualenv version
- Added tag 1.24.2 for changeset f31869779141

1.24.2

- update history and bump version
- update history
- add user-provided tips to the docs

- switch doc theme for packaged docs; add link to Rich Leland's screencast
- Added tag 1.24.1 for changeset 4a8870326d84

1.24.1

- bump version num before new release
- add license and home page info to top of script
- Added tag 1.24 for changeset b243d023094b

1.24

- bump version and update history
- fix preactivate scripts; warn for existing scripts that need to be executable but are not
- Added tag 1.23 for changeset e55e8a54de7b

1.23

- prep for release
- test both mkvirtualenv hooks
- fix the postmkvirtualenv hook
- Added tag 1.22 for changeset c50385e9c99b

1.22

- bump version
- Added tag 1.22 for changeset eddb2921783c
- automatically create hook scripts
- add mode specification for emacs
- update README instructions
- Added tag 1.21 for changeset 2190584becc7

1.21

- update version for new release
- Added tag 1.21 for changeset c11ee7913230
- verify that virtualenv is installed; correct use of python to fix the WORKON_HOME value; more tests
- improve handling for missing WORKON_HOME variable or directory; add test for #18 - can't reproduce
- Added tag 1.20 for changeset ed873ac408ff

1.20

- prepare release
- minor code cleanup
- added simple lssitepackages test
- lssitepackages now also shows contents of virtualenv_path_extensions.pth, if that file exists
- added a white-line at the end
- added lssitepackages info
- added lssitepackages command
- moved main website source files
- Added tag 1.19 for changeset 8af191bfa3c8

1.19

- fix for ticket #14: relative paths don't work with add2virtualenv
- incorporate patch from Sascha Brossmann to fix #15
- Applying my own ridiculous formatting to the README file. Give me 72 characters or give me death!
- Added tag 1.18 for changeset 24190e878fa8

1.18

- bump version number
- don't forget the destdir info
- add basic developer info to the documentation
- add docs for deactivate to resolve issue #12
- fix issue #10 by removing warning and using an error at runtime
- Added tag 1.17.1 for changeset 10fbaab7da41

1.17.1

- update pavement to use sphinxcontrib.paverutils
- Added tag 1.17 for changeset 749030a692a0

1.17

- add installation test task
- incorporate personal site templates into a build that lets me generate hosted docs
- formatting tweaks
- add feature list; clean up hook list; fix bug in warning message generation
- cannot run package from command line, so just warn on import

- create a simple python package and include the documentation in it so it is installed by default
- clean up and update docs, reduce size of readme, start working on packaging changes
- import documentation contribution from Steve Steiner
- run the tests under zsh as well as explicitly invoking bash
- Added tag 1.16 for changeset 7d9dbc84f25d

1.16

- bump version
- remove todo list
- Redirect all error messages from stdout to stderr Added directory completion for cdvirtualenv
- Allow cdvirtualenv to take an argument which is a directory under the virtualenv root to change into
- Added tag 1.15 for changeset bddfacc3c8fde

1.15

- prep release 1.15
- error handling in mkvirtualenv
- add tests to sdist package
- Added tag 1.14 for changeset 6e54ea32a9d1

1.14

- use dist_dir option for sdist command
- Added tag 1.14 for changeset caf3f2a31fdd
- update version #
- Added tag 1.14 for changeset e31542a0d946
- update change list
- fix virtualenvwrapper_show_workon_options to use find again
- rewrite tests using shutil2
- experimental version of deactivate wrapper
- Added tag 1.13 for changeset 7c40caf6ce6f

1.13

- add test.sh to manifest
- Added tag 1.13 for changeset 8e73805a97e1
- fix for issue #5
- Added tag 1.12 for changeset dda0e4d36a91

1.12

- fix verification in navigation functions and add tests
- Add a couple of quick-navigation helper functions
- add attribution
- check return code from `virtualenvwrapper_verify_workon_home` everywhere and return an error code if validation fails
- Update quick setup instructions to make them a little easier to follow and to fix a mistake in the order of some of the steps
- Added tag 1.11 for changeset 511994f15d58

1.11

- run global `postactivatehook` before local; move release not to the correct version
- merge ChrisHas35's `postactivatehook` changes
- start 1.11 with optimization suggestion from ChrisHas35
- Added tag 1.10 for changeset 274d4576d606
- add global `postactivate` hook. related to #3
- remove unnecessary `egrep` calls on `show_workon_options`. fixes #4

1.10

- update change history
- Updated 'workon' to use `find`, to avoid problems with colored 'ls' output
- Added tag 1.9 for changeset d8112e52eadc

1.9

- add more hooks based on suggestion from Chris Hasenpflug; add documentation
- Added tag 1.8.1 for changeset 8417344df8ff

1.8.1

- bump version number
- Added tag 1.8.1 for changeset dca76424222e
- fix argument processing in `mkvirtualenv`
- Added tag 1.8 for changeset ea5f27af83bb

1.8

- Fix for processing the argument list in mkvirtualenv from jorgevargas (#1)
- Added tag 1.7 for changeset 32f2a081d649

1.7

- Clean up TODO list and svn keywords. Add license section to README
- Added tag 1.7 for changeset 54aa96a1c09f
- Ignore files generated by paver and the build process. Use a fixed version string in the pavement.py file
- update tags
- convert from make to paver 1.0
- patches to rmvirtualenv to make it work with zsh from Byron Clark
- add note about zsh completion support
- add zsh completion support, courtesy of Ted Leung
- add docs; fix space issues
- remove premature release
- add path management feature contributed by James Bennett
- fix another typo, TEST, then add another useful message when the user tries to remove an active environment
- fix spelling mistake

1.6.1

- bug fix from John Shimek
- Add tab completion based on Arthur Koziel's version at <http://arthurkoziel.com/2008/10/11/virtualenvwrapper-bash-completion/>
- fix the download url

1.3

- add setup.py and related pieces, including minimal docs
- usability patches from Alex Satrapa
- notes about what I still need to do
- cleanup
- predeactivate and postactivate hooks
- go ahead and change to the environment after creating it
- look for the workdir script and run it if we find it
- update comments
- add attribution

- keywords

1.0

- first copy
- start new project

Glossary

project directory Directory associated with a virtualenv, usually located elsewhere and containing more permanent development artifacts such as local source files, test data, etc. (see *Location of Project Directories*)

template Input to *mkproject* that configures the *project directory* to contain default files. (see *Extending Virtualenvwrapper*)

References

`virtualenv`, from Ian Bicking, is a pre-requisite to using these extensions.

For more details, refer to the column I wrote for the May 2008 issue of Python Magazine: [virtualenvwrapper | And Now For Something Completely Different](#).

Rich Leland has created a short [screencast](#) showing off the features of `virtualenvwrapper`.

Manuel Kaufmann has [translated this documentation into Spanish](#).

Tetsuya Morimoto has [translated this documentation into Japanese](#).

Support

Join the [virtualenvwrapper Google Group](#) to discuss issues and features.

Report bugs via the [bug tracker on BitBucket](#).

Shell Aliases

Since `virtualenvwrapper` is largely a shell script, it uses shell commands for a lot of its actions. If your environment makes heavy use of shell aliases or other customizations, you may encounter issues. Before reporting bugs in the bug tracker, please test *without* your aliases enabled. If you can identify the alias causing the problem, that will help make `virtualenvwrapper` more robust.

License

Copyright Doug Hellmann, All Rights Reserved

Permission to use, copy, modify, and distribute this software and its documentation for any purpose and without fee is hereby granted, provided that the above copyright notice appear in all copies and that both that copyright notice and this permission notice appear in supporting documentation, and that the name of Doug Hellmann not be used in advertising or publicity pertaining to distribution of the software without specific, written prior permission.

DOUG HELLMANN DISCLAIMS ALL WARRANTIES WITH REGARD TO THIS SOFTWARE, INCLUDING ALL IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS, IN NO EVENT SHALL DOUG HELLMANN BE LIABLE FOR ANY SPECIAL, INDIRECT OR CONSEQUENTIAL DAMAGES OR ANY DAMAGES WHATSOEVER RESULTING FROM LOSS OF USE, DATA OR PROFITS, WHETHER IN AN ACTION OF CONTRACT, NEGLIGENCE OR OTHER TORTIOUS ACTION, ARISING OUT OF OR IN CONNECTION WITH THE USE OR PERFORMANCE OF THIS SOFTWARE.

P

project directory, 69

T

template, 69