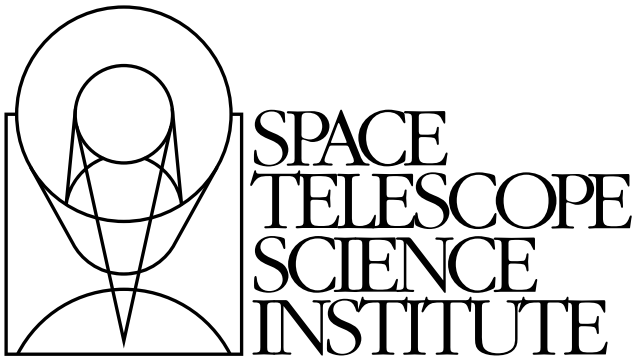

stsci.distutils Documentation

Release 0.3.2

Erik Bray, Mark Sienkiewicz, Christine Slocum

April 18, 2016

CONTENTS

1 Introduction **1**

2 Features **3**
2.1 Hook Scripts 3
2.2 Commands 5

3 Appendix **7**
3.1 API documentation 7
3.2 Changelog 10

Python Module Index **13**

Index **15**

CHAPTER

ONE

INTRODUCTION

This package contains utilities used to package some of STScI's Python projects; specifically those projects that comprise stsci_python and Astrolib.

It currently consists mostly of some setup_hook scripts meant for use with distutils2/packaging and/or d2to1, and a customized easy_install command meant for use with distribute.

This package is not meant for general consumption, though it might be worth looking at for examples of how to do certain things with your own packages, but YMMV.

CHAPTER

TWO

FEATURES

2.1 Hook Scripts

Currently the main features of this package are a couple of setup_hook scripts. In distutils2, a setup_hook is a script that runs at the beginning of any pysetup command, and can modify the package configuration read from setup.cfg. There are also pre- and post-command hooks that only run before/after a specific setup command (eg. build_ext, install) is run.

2.1.1 stsci.distutils.hooks.use_packages_root

If using the `packages_root` option under the `[files]` section of setup.cfg, this hook will add that path to `sys.path` so that modules in your package can be imported and used in setup. This can be used even if `packages_root` is not specified–in this case it adds `''` to `sys.path`.

2.1.2 stsci.distutils.hooks.version_setup_hook

Creates a Python module called version.py which currently contains four variables:

- `__version__` (the release version)
- `__svn_revision__` (the SVN revision info as returned by the `svnversion` command)
- `__svn_full_info__` (as returned by the `svn info` command)
- `__setup_datetime__` (the date and time that setup.py was last run).

These variables can be imported in the package's `__init__.py` for degugging purposes. The version.py module will *only* be created in a package that imports from the version module in its `__init__.py`. It should be noted that this is generally preferable to writing these variables directly into `__init__.py`, since this provides more control and is less likely to unexpectedly break things in `__init__.py`.

2.1.3 stsci.distutils.hooks.version_pre_command_hook

Identical to version_setup_hook, but designed to be used as a pre-command hook.

2.1.4 stsci.distutils.hooks.version_post_command_hook

The complement to version_pre_command_hook. This will delete any version.py files created during a build in order to prevent them from cluttering an SVN working copy (note, however, that version.py is *not* deleted from the build/

directory, so a copy of it is still preserved). It will also not be deleted if the current directory is not an SVN working copy. For example, if source code extracted from a source tarball it will be preserved.

2.1.5 stsci.distutils.hooks.tag_svn_revision

A setup_hook to add the SVN revision of the current working copy path to the package version string, but only if the version ends in .dev.

For example, `mypackage-1.0.dev` becomes `mypackage-1.0.dev1234`. This is in accordance with the version string format standardized by PEP 386.

This should be used as a replacement for the `tag_svn_revision` option to the egg_info command. This hook is more compatible with packaging/distutils2, which does not include any VCS support. This hook is also more flexible in that it turns the revision number on/off depending on the presence of `.dev` in the version string, so that it's not automatically added to the version in final releases.

This hook does require the `svnversion` command to be available in order to work. It does not examine the working copy metadata directly.

2.1.6 stsci.distutils.hooks.numpy_extension_hook

This is a pre-command hook for the build_ext command. To use it, add a `[build_ext]` section to your setup.cfg, and add to it:

```
pre-hook.numpy-extension-hook = stsci.distutils.hooks.numpy_extension_hook
```

This hook must be used to build extension modules that use Numpy. The primary side-effect of this hook is to add the correct numpy include directories to `include_dirs`. To use it, add 'numpy' to the 'include-dirs' option of each extension module that requires numpy to build. The value 'numpy' will be replaced with the actual path to the numpy includes.

2.1.7 stsci.distutils.hooks.is_display_option

This is not actually a hook, but is a useful utility function that can be used in writing other hooks. Basically, it returns `True` if setup.py was run with a "display option" such as –version or –help. This can be used to prevent your hook from running in such cases.

2.1.8 stsci.distutils.hooks.glob_data_files

A pre-command hook for the install_data command. Allows filename wildcards as understood by `glob.glob()` to be used in the data_files option. This hook must be used in order to have this functionality since it does not normally exist in distutils.

This hook also ensures that data files are installed relative to the package path. data_files shouldn't normally be installed this way, but the functionality is required for a few special cases.

2.2 Commands

2.2.1 build_optional_ext

This serves as an optional replacement for the default built_ext command, which compiles C extension modules. Its purpose is to allow extension modules to be *optional*, so that if their build fails the rest of the package is still allowed to be built and installed. This can be used when an extension module is not definitely required to use the package.

To use this custom command, add:

```
commands = stsci.distutils.command.build_optional_ext.build_optional_ext
```

under the `[global]` section of your package's setup.cfg. Then, to mark an individual extension module as optional, under the setup.cfg section for that extension add:

```
optional = True
```

Optionally, you may also add a custom failure message by adding:

```
fail_message = The foobar extension module failed to compile.
               This could be because you lack such and such headers.
               This package will still work, but such and such features
               will be disabled.
```

CHAPTER

THREE

APPENDIX

3.1 API documentation

3.1.1 stsci.distutils.hooks

stsci.distutils.hooks.**glob_data_files**(*command_obj*)

A pre-command hook for the install_data command allowing wildcard patterns to be used in the data_files option.

Also ensures that data files with relative paths as their targets are installed relative install_lib.

Config Usage:

```
[files]
data_files =
    target_directory = source_directory/*.foo
    other_target_directory = other_source_directory/*

[install_data]
pre-hook.glob-data-files = stsci.distutils.hooks.glob_data_files
```

stsci.distutils.hooks.**is_display_option**(*ignore=None*)

A hack to test if one of the arguments passed to setup.py is a display argument that should just display a value and exit. If so, don't bother running this hook (this capability really ought to be included with distutils2).

Optionally, ignore may contain a list of display options to ignore in this check. Each option in the ignore list must contain the correct number of dashes.

stsci.distutils.hooks.**numpy_extension_hook**(*command_obj*)

A distutils2 pre-command hook for the build_ext command needed for building extension modules that use NumPy.

To use this hook, add 'numpy' to the list of include_dirs in setup.cfg section for an extension module. This hook will replace 'numpy' with the necessary numpy header paths in the include_dirs option for that extension.

Note: Although this function uses numpy, stsci.distutils does not depend on numpy. It is up to the distribution that uses this hook to require numpy as a dependency.

Config Usage:

```
[extension=mypackage.extmod]
sources =
    foo.c
    bar.c
include_dirs = numpy
```

```
[build_ext]
pre-hook.numpy-extension = stsci.distutils.hooks.numpy_extension_hook
```

stsci.distutils.hooks.**tag_svn_revision**(*config*)

A setup_hook to add the SVN revision of the current working copy path to the package version string, but only if the version ends in .dev.

For example, `mypackage-1.0.dev` becomes `mypackage-1.0.dev1234`. This is in accordance with the version string format standardized by PEP 386.

This should be used as a replacement for the `tag_svn_revision` option to the egg_info command. This hook is more compatible with packaging/distutils2, which does not include any VCS support. This hook is also more flexible in that it turns the revision number on/off depending on the presence of `.dev` in the version string, so that it's not automatically added to the version in final releases.

This hook does require the `svnversion` command to be available in order to work. It does not examine the working copy metadata directly.

Config Usage:

```
[global]
setup_hooks = stsci.distutils.hooks.tag_svn_revision
```

You should write exactly this in your package's `__init__.py`:

```
from .version import *
```

stsci.distutils.hooks.**use_packages_root**(*config*)

Adds the path specified by the 'packages_root' option, or the current path if 'packages_root' is not specified, to sys.path. This is particularly useful, for example, to run setup_hooks or add custom commands that are in your package's source tree.

Use this when the root of your package source tree is not in the same directory with the setup.py

Config Usage:

```
[files]
packages_root = lib
# for installing pkgname from lib/pkgname/*.py

[global]
setup_hooks = stsci.distutils.hooks.use_packages_root
```

stsci.distutils.hooks.**version_post_command_hook**(*command_obj*)

Deprecated since version 0.3: This hook was meant to complement *version_pre_command_hook()*, also deprecated.

Cleans up a previously generated version.py in order to avoid clutter.

Only removes the file if we're in an SVN working copy and the file is not already under version control.

stsci.distutils.hooks.**version_pre_command_hook**(*command_obj*)

Deprecated since version 0.3: Use *version_setup_hook()* instead; it's generally safer to check/update the version.py module on every setup.py run instead of on specific commands.

This command hook creates an version.py file in each package that requires it. This is by determining if the package's `__init__.py` tries to import or import from the version module.

version.py will not be created in packages that don't use it. It should only be used by the top-level package of the project.

stsci.distutils.hooks.**version_setup_hook**(*config*)
Creates a Python module called version.py which contains these variables:

- __version__ (the release version)
- __svn_revision__ (the SVN revision info as returned by the svnversion command)
- __svn_full_info__ (as returned by the svn info command)
- __setup_datetime__ (the date and time that setup.py was last run).
- __vdate__ (the release date)

These variables can be imported in the package's __init__.py for degugging purposes. The version.py module will *only* be created in a package that imports from the version module in its __init__.py. It should be noted that this is generally preferable to writing these variables directly into __init__.py, since this provides more control and is less likely to unexpectedly break things in __init__.py.

Config Usage:

```
[global]
setup-hooks = stsci.distutils.hooks.version_setup_hook
```

You should write exactly this in your package's __init__.py:

```
from .version import *
```

3.1.2 stsci.distutils.svnutils

Functions for getting and saving SVN info for distribution.

stsci.distutils.svnutils.**get_svn_info**(*path='.'*)
Uses svn info to get the full information about the working copy at the given path.

stsci.distutils.svnutils.**get_svn_version**(*path='.'*)
Uses svnversion to get just the latest revision at the given path.

3.1.3 stsci.distutils.versionutils

Utilities for dealing with package version info.

See also stsci.distutils.svnutils which specifically deals with adding SVN info to version.py modules.

stsci.distutils.versionutils.**clean_version_py**(*package_dir*, *package*)
Removes the generated version.py module from a package, but only if we're in an SVN working copy.

stsci.distutils.versionutils.**package_uses_version_py**(*package_root*, *package*, *module_name='version'*)
Determines whether or not a version.py module should exist in the given package. Returns the full path to the version.py module, regardless of whether it already exists. Otherwise returns False.

This works by checking whether or not there are any imports from the 'version' module in the package's __init__.py.

You should write this in your package's __init__.py:

```
from .version import *
```

stsci.distutils.versionutils.**update_setup_datetime**(*filename='version.py'*)
Update the version.py with the last time a setup command was run.

3.2 Changelog

3.2.1 0.3.8 (unreleased)

- Nothing changed yet.

3.2.2 0.3.7 (2013-12-23)

- Avoid using `Popen.stdout` directly in the version.py SVN revision auto-update script to avoid possible ResourceWarnings on Python >= 3.2. See https://github.com/spacetelescope/PyFITS/issues/45

3.2.3 0.3.6 (2013-11-21)

- Fixed a syntax error in Python 3 that was introduced in 0.3.5. This could occur very early in the setup such that it bailed before even 2to3 could run on the rest of the package.

3.2.4 0.3.5 (2013-11-18)

- Fixed an obscure issue that could occur when trying to install with easy_install on Python 2 systems that have lib2to3 installed but have never used it.

3.2.5 0.3.4 (2013-07-31)

- Updated the check for `__loader__` added in v0.3.3 to only perform that check on Python >= 3.3, since the issue doesn't apply to older Python versions.

3.2.6 0.3.3 (2013-07-25)

- Updated the import-time SVN revision update mechanism in the `version.py` module generated by the `version_setup_hook` to avoid running when not in a dev version of the package. This saves time on importing released packages when installed on users' systems.
- Added a workaround to a bug on Python 3.3 that could cause stsci.distutils to crash during installation.

3.2.7 0.3.2 (2013-03-27)

- Fixed a bug in the version hook that could occur if the svnversion command fails.
- Updated the template for the version.py module generated by the version hook so that `from .version import *` will work for applications.
- Added a `__vdate__` variable in version.py which may contain a release date string by specifying a `vdate` option in the `[metadata]` section of setup.cfg.
- Added a `stsci_distutils_version` variable in version.py containing the version of stsci.distutils used to generate the file–useful primarily for debugging purposes.
- Version 0.3.1 added a new zest.releaser hooks to ensure that source distributes are created as tar.gz files instead of zip files–this was left out of the changelog for 0.3.1.
- The tar.gz zest.releaser hook is updated in 0.3.2 to only run on stsci packages.

3.2.8 0.3.1 (2012-06-28)

- Fixed a bug where console output from svn-related programs was assumed to be ascii, leading to possible crashes on non-English systems.

3.2.9 0.3 (2012-02-20)

- The `glob_data_files` hook became a pre-command hook for the install_data command instead of being a setup-hook. This is to support the additional functionality of requiring data_files with relative destination paths to be install relative to the package's install path (i.e. site-packages).
- Dropped support for and deprecated the easier_install custom command. Although it should still work, it probably won't be used anymore for stsci_python packages.
- Added support for the `build_optional_ext` command, which replaces/extends the default `build_ext` command. See the README for more details.
- Added the `tag_svn_revision` setup_hook as a replacement for the setuptools-specific tag_svn_revision option to the egg_info command. This new hook is easier to use than the old tag_svn_revision option: It's automatically enabled by the presence of `.dev` in the version string, and disabled otherwise.
- The `svn_info_pre_hook` and `svn_info_post_hook` have been replaced with `version_pre_command_hook` and `version_post_command_hook` respectively. However, a new `version_setup_hook`, which has the same purpose, has been added. It is generally easier to use and will give more consistent results in that it will run every time setup.py is run, regardless of which command is used. `stsci.distutils` itself uses this hook–see the `setup.cfg` file and `stsci/distutils/__init__.py` for example usage.
- Instead of creating an `svninfo.py` module, the new `version_` hooks create a file called `version.py`. In addition to the SVN info that was included in `svninfo.py`, it includes a `__version__` variable to be used by the package's `__init__.py`. This allows there to be a hard-coded `__version__` variable included in the source code, rather than using pkg_resources to get the version.
- In `version.py`, the variables previously named `__svn_version__` and `__full_svn_info__` are now named `__svn_revision__` and `__svn_full_info__`.
- Fixed a bug when using stsci.distutils in the installation of other packages in the `stsci.*` namespace package. If stsci.distutils was not already installed, and was downloaded automatically by distribute through the setup_requires option, then `stsci.distutils` would fail to import. This is because the way the namespace package (nspkg) mechanism currently works, all packages belonging to the nspkg *must* be on the import path at initial import time.

  So when installing stsci.tools, for example, if `stsci.tools` is imported from within the source code at install time, but before `stsci.distutils` is downloaded and added to the path, the `stsci` package is already imported and can't be extended to include the path of `stsci.distutils` after the fact. The easiest way of dealing with this, it seems, is to delete `stsci` from `sys.modules`, which forces it to be reimported, now the its `__path__` extended to include `stsci.distutil`'s path.
- Added zest.releaser hooks for tweaking the development version string template, and for uploading new releases to STScI's local package index.

3.2.10 0.2.2 (2011-11-09)

- Fixed check for the issue205 bug on actual setuptools installs; before it only worked on distribute. setuptools has the issue205 bug prior to version 0.6c10.

- Improved the fix for the issue205 bug, especially on setuptools. setuptools, prior to 0.6c10, did not back of sys.modules either before sandboxing, which causes serious problems. In fact, it's so bad that it's not enough to add a sys.modules backup to the current sandbox: It's in fact necessary to monkeypatch setuptools.sandbox.run_setup so that any subsequent calls to it also back up sys.modules.

3.2.11 0.2.1 (2011-09-02)

- Fixed the dependencies so that setuptools is requirement but 'distribute' specifically. Previously installation could fail if users had plain setuptools installed and not distribute

3.2.12 0.2 (2011-08-23)

- Initial public release

S

stsci.distutils.hooks, 7
stsci.distutils.svnutils, 9
stsci.distutils.versionutils, 9

C

clean_version_py() (in module stsci.distutils.versionutils), 9

G

get_svn_info() (in module stsci.distutils.svnutils), 9
get_svn_version() (in module stsci.distutils.svnutils), 9
glob_data_files() (in module stsci.distutils.hooks), 7

I

is_display_option() (in module stsci.distutils.hooks), 7

N

numpy_extension_hook() (in module stsci.distutils.hooks), 7

P

package_uses_version_py() (in module stsci.distutils.versionutils), 9

S

stsci.distutils.hooks (module), 7
stsci.distutils.svnutils (module), 9
stsci.distutils.versionutils (module), 9

T

tag_svn_revision() (in module stsci.distutils.hooks), 8

U

update_setup_datetime() (in module stsci.distutils.versionutils), 9
use_packages_root() (in module stsci.distutils.hooks), 8

V

version_post_command_hook() (in module stsci.distutils.hooks), 8
version_pre_command_hook() (in module stsci.distutils.hooks), 8
version_setup_hook() (in module stsci.distutils.hooks), 8