Setupext: gitversion

Release 1.1.0

January 06, 2015

Contents

CHAPTER 1

Wait... Why? What??

PEP440 codifies a version scheme for Python packages. This setuptools extension will generate Developmental Release and Local Version Label segments that identify the revision that is being built. PEP440 defines the following format for Python package versions:

```
version         = [epoch "!"] public-version ["+" local-version]
epoch           = digit+
public-version  = release-segment [pre-segment] [post-segment] [dev-segment]
local-version   = (letter | digit)+ ["." (letter|digit)+]
release-segment = digit+ ("." digit+)*
pre-segment     = "a" digit+ | "b" digit+ | "rc" digit+
post-segment    = ".post" digit+
dev-segment     = ".dev" digit+
```

It also recommends that package indecies only accept *final releases* which are defined as having a version that consists of only a release segment and an optional epoch. So why did I go through all of the trouble to create an extension for managing versions that should not be submitted to a package index? If you develop Python packages that are used inside the walls of a business, then you probably know exactly why – using a local Python Package Index that holds non-public packages is commonplace. It is also common to stage pre-release packages and builds from a CI server in an internal index. This is where this extension comes into play. It provides a consistent way to manage package versions throughout all stages of development.

Let's look at the state of this project as I am writing this document. The git history looks like the following:

```
* 3fdc192 - (HEAD, origin/initial-implementation, initial-implementation)
###### 9 more commits here
* 7ca1fd2 - (origin/master, master)
*   87d944e - Merge pull request #1 from dave-shawley/reorg (6 days ago)
|\
| * 04d0cca - (origin/reorg, reorg)
| ###### 9 more commits here
|/
* bd7ad3c - (tag: 0.0.0) SYN (4 months ago)
```

When I run **setup.py git_version** it sets the version to `0.0.0.post1.dev11`. The `0.0.0` portion is the release segment that is passed to the `setup` function in *setup.py*. The extension finds that tag in the history and counts the number of merges that have occurred since that tag – this value becomes the *post* version segment. In this case there has only been a single merge. Then it counts the number of commits since the last merge occurred – this value becomes the *development* version segment.

CHAPTER 2

How?

The easiest way to use this extension is to install it into your build environment and then use it from the command line when you generate and upload your distribution.

1. `pip install setupext-gitversion` into your build environment
2. Add the following lines to your *setup.cfg*:

```
[git_version]
version-file = LOCAL-VERSION
```

3. Add the following line to your *MANIFEST.in*:

   ```
   include LOCAL-VERSION
   ```

4. Modify your *setup.py* to append the contents of *LOCAL-VERSION* to your `version` keyword:

   ```
   version_suffix = ''
   try:
      with open('LOCAL-VERSION') as f:
         version_suffix = f.readline().strip()
   except IOError:
      pass

   setup(
      # normal keywords
      version='1.2.3' + version_suffix,
   )
   ```

   Where `1.2.3` is the tag of the last release package.

5. Add `git_version` to your *upload* or distribution generation command. You want to use something like the following:

   ```
   setup.py git_version sdist upload
   setup.py git_version bdist_egg upload
   ```

And that's it. That will embed SCM information into your package when your build a distribution. It is also smart enough to generate an empty suffix for a build from a tagged commit.

CHAPTER 3

Ok... Where?

Source	https://github.com/dave-shawley/setupext-gitversion
Status	https://travis-ci.org/dave-shawley/setupext-gitversion
Download	https://pypi.python.org/pypi/setupext-gitversion
Documentation	http://setupext-gitversion.readthedocs.org/en/latest
Issues	https://github.com/dave-shawley/setupext-gitversion

CHAPTER 4

Documentation

4.1 Usage

4.1.1 Command Line Synopsis

Usage: setup.py git_version *[options] packaging-command*

The **git_version** setuptools command updates the package's version metadata based on repository information. Since the update is only done on the metadata in-memory, this is only really useful in the same *setup.py* execution as a packaging command such as `sdist` or one of the `bdist` variants.

-C, --committish
Include the abbreviated version of the most recent committish as the local portion of the version number.

-V `FILE`**, --version-file** `FILE`
Writes the local segment of the version to *FILE* in addition to setting the in-memory version.

4.1.2 setup.cfg Example

```
[git_version]
version-file = LOCAL-VERSION
```

4.2 Version Calculation

PEP 440 defines a Python Package's version using the following grammar.

```
version   ::=   [epoch "!"]  public ["+" local]
epoch     ::=   digit+
public    ::=   release [pre][post][dev]
release   ::=   digit+ ("."  release)*
pre       ::=   "pre" digit+
post      ::=   "post" digit+
dev       ::=   "dev" digit+
local     ::=   (letter | digit)+ ("."  local)*
```

4.2.1 Public Version

The public portion of the version identifier is managed by your package. The best practice for managing the public version is to simply embed it within your package as a top-level attribute named `__version__` or `version`. You should use the version attribute to calculate the value passed to as the `version` keyword to `setuptools.setup()`.

```
# setup.py
import setuptools
import mypackage

setup(
   name='mypackage',
   version=mypackage.__version__,
   # ...
)
```

This extension searches for a git tag matching the public portion of the `version` keyword and uses it as the basis for constructing the post and development release segements.

4.2.2 Pre Release Segment

This extension does not define a value for the pre-release segment.

4.2.3 Post Release Segment

This extension defines the post-release segment as the number of merges since the tag associated with your package's version.

4.2.4 Development Release Segment

This extension defines the development release segment as the number of commits since the last merge.

4.2.5 Local Segment

This extension defines the local identifier as the first seven characters of the most recent commit. The local identifier is only included if the `--committish` flag is included and either the post or development segment is defined.

4.3 Setting up your environment

The easiest way to start working with this code is to set up a virtual environment and run `env/bin/pip -r dev-requirements.txt`. That will install the necessary testing tools. Then you can run everything else using `env/bin/python setup.py`:

- *setup.py nosetests* will run the tests using nose to test against the and generate a coverage report to stdout.
- *setup.py build_sphinx* will generate HTML documentation into *build/doc/html*. This is the doc set that is uploaded to Read The Docs.
- *setup.py flake8* will run the `flake8` utility and report on any static code analysis failures.

CHAPTER 5

Changelog

- 1.1.0 (6-Jan-2015)
  - Add *–committish* command line flag.
- 1.0.1 (3-Jan-2015)
  - Switch from using drone.io to travis-ci.org
- 1.0.0 (3-Jan-2015)
  - Update metadata version based on tag and git repo state.
  - Write local version to file when *–version-file* is specified.

Symbols

P

S