dataflake.docbuilder

Release 2.3

Jens Vagelpohl

Feb 06, 2023

CONTENTS

This package provides a set of scripts to automate building Sphinx-based package documentation for packages hosted on a source code revision control server. At this point only Git repositories are supported.

- check out the current development trunk and all tagged versions
- build all Sphinx-based documentation in them, if it exists
- stitch together the trunk and release versions in a single HTML file to provide a simple jump-off page for all package versions
- rebuild the Sphinx documentation if there were any changes since the last build

CHAPTER

ONE

NARRATIVE DOCUMENTATION

Narrative documentation explaining how to use `dataflake.docbuilder`.

1.1 Installation

1.1.1 Prerequisites

For cloning remote software repositories the software uses the version control system client directly, so you must have the respective client package like *git* installed on your system.

1.1.2 Install with `pip`

```
$ pip install dataflake.docbuilder
```

1.1.3 Install with `zc.buildout`

Just add `dataflake.docbuilder` to the `eggs` setting(s) in your buildout configuration to have it pulled in automatically:

```
...
eggs =
    dataflake.docbuilder
...
```

1.2 Using dataflake.docbuilder

`dataflake.docbuilder` can be used in two ways. It defines a *Setuptools* entry point called `docbuilder`, which automatically creates a shell script `docbuilder` when the package is installed through *Setuptools*. If you are using zc.buildout, you can use the package directly as a `recipe` and configure the documentation builder within your buildout configuration file.

Warning: When the documentation build scripts build the *Sphinx* documentation, the downloaded packages are not fully installed. They only have their EGG-INFO structures created and are then added to `sys.path` for the time it takes to run the *Sphinx* builder so they can be imported. That means their dependencies are not installed

automatically. If there are dependencies that must be available before the package can be imported or before the *Sphinx* documentation can be built (such as third-party *Sphinx* extensions) you must make them available yourself. How to do so is shown below.

Warning: The document build script will call the Git script through the shell. Make sure you can check out or update the packages you want to document without receiving any prompts, e.g. for credentials, otherwise the script will just hang waiting for input on a prompt you will never see.

1.2.1 From `pip`

After installing `dataflake.docbuilder` using *Setuptools*, a shell script named `docbuilder` is created. This can be used to invoke the documentation build process and accepts several options, which you can discover yourself by running `docbuilder -h` or `docbuilder --help`:

- `-s` <URLS> or `--source=`<URLS>: This *mandatory* parameter, which can be given multiple times, contains an URL to a package's location in a software version control repository. You can prefix the URL with information about the revision control server used, if no prefix is given, Git is assumed:

```
[git]https://myserver/git/mypackage
https://github.com/organization/myotherpackage.git
```

- `-g` <GROUP> or `--grouping=`<GROUP>: You can group packages into groups to set them apart in the HTML output. A GROUP element consists of the package name and the group name, separated by a colon (":") character. Example: *dataflake.docbuilder:dataflake*.
- `-w` <PATH> or `--working-directory=`<PATH>: The `docbuilder` script will check out the packages and run the documentation build process in this folder. This parameter is *mandatory* as well.
- `-o` <PATH> or `--output-directory=`<PATH>: This is where the folder tree for the HTML output is stored, which links back into the build tree defined by the `working-directory` parameter. If it is not specified, the HTML output tree will end up in a folder named *html* inside the `working-directory`.
- `-t` or `--trunk-only`: This flag is unset by default. If you set this flag only documentation from the main development branch will be built.
- `-m` or `--max-tags`: When building documentation for the current development version and all tags this flag specifies how many tags to show on the main index page. If more tags exist, a link to a separate page is inserted that shows all tags for the given package. The default value is 5.
- `-v` or `--verbose`: Set the log verbosity. If `--v` is specified you will see more detailed logging output. If you specify it more than once all `Sphinx` documentation build output will be shown as well.
- `--index-template=`<PATH>: This optional parameter contains a filesystem path to a folder containing a *Sphinx* configuration (a `conf.py` file) and templates/static files. If you provide such a path `dataflake.docbuilder` will use it to build additional content for the `output-directory` folder and auto-generate an index file. Please note: This folder must not contain an `index.rst` document, as the index file will be auto-generated during the documentation build process. You can optionally provide a template named `index.rst.in` in the `index-template` folder, which will have the autogenerated package list appended at the end.
- `--index-name=`<NAME>: The file name, sans extension, for the index file. A ReST source file <NAME>`.rst` will be created containing links to the documented packages, and *Sphinx* will compile it to the final <NAME>`.html` output. The default value is `index`.

- `--docs-directory=<NAME>`: The folder name inside your software package checkout where *Sphinx* documentation is stored. By default, the folders *doc* and *docs* are searched. You can use this parameter multiple times to add other folder names to the default list.
- `-h` or `--help`: Show the help text.

If the package to be documented or its *Sphinx* documentation configuration needs additional packages to be imported and run, you need to make them available yourself by e.g. using `easy_install` or by adding them to your zc.buildout configuration.

1.2.2 From zc.buildout

In a zc.buildout configuration file, the `dataflake.docbuilder` package can be used directly as a recipe. The recipe will create a shell script that invokes the document build process with the options specified in the configuration stanza. Here's a simple example:

```
[buildout]
parts = docbuilderdocs

[docbuilderdocs]
recipe = dataflake.docbuilder
eggs =
    repoze.sphinx.autointerface
sources =
    https://github.com/dataflake/dataflake.docbuilder.git
```

This configuration will create a script named `docbuilderdocs` which builds the *Sphinx* documentation found in the `dataflake.docbuilder` development head and all tags.

The following keywords can be used with this recipe (documentation see above):

- `eggs`: If the package to be documented or its *Sphinx* documentation configuration needs additional packages to be imported and run, you need to list them here so they get pulled in automatically.
- `sources`: Equivalent to one or more `--source` parameters shown above. Mandatory.
- `groupings`: One or more `--grouping` parameters as shown above.
- `working-directory`: The `--working-directory` parameter shown above. If none is specified, a default of `{buildout:directory}/parts/<SCRIPTNAME>` is used.
- `output-directory`: The `--output-directory` parameter shown above
- `trunk-only`: The `--trunk-only` parameter shown above
- `max-tags`: The `--max-tags` parameter shown above
- `verbose`: The `--verbose` parameter shown above
- `index-template`: The `index-template` parameter shown above
- `index-name`: The `--index-name` parameter shown above
- `docs-directory`: The `--docs-directory` parameter shown above

1.3 Development

1.3.1 Bug tracker

For bug reports, suggestions or questions please use the GitHub issue tracker at https://github.com/dataflake/dataflake.docbuilder/issues.

1.3.2 Getting the source code

The source code is maintained on GitHub. To check out the main branch:

```
$ git clone https://github.com/dataflake/dataflake.docbuilder.git
```

You can also browse the code online at https://github.com/dataflake/dataflake.docbuilder

1.3.3 Preparing the development sandbox

The following steps only need to be done once to install all the tools and scripts needed for building, packaging and testing. First, create a Virtual environment. The example here uses Python 3.11, but any Python version supported by this package will work. Then install all the required tools:

```
$ cd dataflake.docbuilder
$ python3.11 -m venv .
$ bin/pip install -U pip wheel
$ bin/pip install -U setuptools zc.buildout tox twine
```

1.3.4 Building the documentation

`tox` is also used to build the Sphinx-based documentation. The input files are in the *docs* subfolder and the documentation build step will compile them to HTML. The output is stored in *docs/_build/html/*:

```
$ bin/tox -edocs
```

If the documentation contains doctests they are run as well.

1.4 Change log

1.4.1 2.3 (2023-02-06)

- Fix index page link creation

1.4.2 2.2 (2023-02-06)

- Fix breakage attempting to activate invalid distributions

1.4.3 2.1 (2022-02-06)

- Add support for older Git versions.

1.4.4 2.0 (2023-02-06)

- Drop support for documentation generated with `z3c.recipe.sphinx`.
- Drop support for Subversion and Mercurial.
- Drop option `copy-output`. Always copy HTML output to HTML output folder.
- Add support for Python 3.8, 3.9, 3.10, 3.11.
- Drop support for Python 2.7, 3.5, 3.6.

1.4.5 1.23 (2018-10-16)

- make sure to convert shell output to unicode on Python 3 and 2.
- before invoking `setup.py` commands in a subshell, make sure it is present.

1.4.6 1.22 (2018-07-22)

- add option `-n` to skip checkout and build of software packages

1.4.7 1.21 (2018-06-29)

- test and declare support for Python 3.7

1.4.8 1.20 (2017-06-01)

- use pkgutil-style namespace declaration
- package cleanup (`.gitignore`, `MANIFEST.in`, `README.rst`)
- docs cleanup (`Makefile`, `conf.py`)
- tests cleanup (`tox.ini`)
- remove unsupported documentation bits

1.4.9 1.19 (2017-05-29)

- Python 3 compatibility

1.4.10 1.18 (2017-05-29)

- added tox configuration to do PEP-8 checking with `flake8`
- PEP 8 code cleanup

1.4.11 1.17 (2017-05-27)

- If output folders don't exist yet create them instead of failing.
- Use the name `development` instead of `trunk` or `head` to designate the current development version in the rendered index page.
- If a version control URL does not specify which version control system is used, *Git* is now used as default instead of *Subversion*.
- Moved the repository to GitHub and its documentation to ReadTheDocs

1.4.12 1.16 (2014-11-25)

- added flag `-c/--copy-output` which will copy instead of link all HTML output into the folder designated as HTML output folder.

1.4.13 1.15 (2014-11-21)

- switched documentation to point to the new Git repository
- use new bootstrap.py script

1.4.14 1.14 (2011-11-29)

- Add a flag `-m/--max-tags` to set the number of package versions (tags) to be shown on the generated main page. If more tags exist, a link to a separate page is provided that shows all versions.

1.4.15 1.13 (2011-10-31)

- Catch additional errors upon building
- Ignoring empty tag lists when asking Git for tags

1.4.16 1.12 (2011-08-30)

- Be more resilient when simple ReST text compilation or Sphinx building fails. Now the whole documentation build process won't just fail at that point.
- provide more meaningful log messages when running with the `-v` option.

1.4.17 1.11 (2011-08-09)

- Now you can use `Git` alongside `Mercurial` and `Subversion` to use as version control system.

1.4.18 1.10 (2011-08-09)

- Taking more control of logging by defining our own logger and suppressing standard Sphinx log output. The new script flag `-v` or `--verbose` enables the user to determine what to show. Without it, only serious warnings are shown. With `-v` specified once you will see script progress output and notes about Sphinx build warnings. With `-vv` all Sphinx output is shown as well.

1.4.19 1.9 (2011-08-09)

- Now using pkg_resources.parse_version to parse the tag names and produce correct release ordering for each package
- Instead of using a flag to set the revision control system across all packages you now specify the revision control system per package with a simple prefix:

```
[hg]http://myserver/hg/mypackage
[svn]https://myservr/svnmypackage
```

  For backwards compatibility, all URLs without prefix are assumed to point to a Subversion repository.

1.4.20 1.8 (2011-08-05)

- Feature: You can now use either `Subversion` or `Mercurial` to check out documented packages.

1.4.21 1.7 (2010-08-03)

- Feature: If no standard package documentation can be found, the setuptools `long_description` settings is used as a last fallback to at least generate a single page for a package.
- Feature: To style the `long_description` fallback ReST documentation, a new parameter `fallback-css` can be used to provide a path to a CSS file.

1.4.22 1.6 (2010-07-31)

- Bug: If the `z3csphinx-output-directory` was set, all its contained packages ended up on the index document. Now this only happens if no SVN source URLs are otherwise provided. If they are, only packages from those source URLs are considered for linking on the index document.

1.4.23 1.5 (2010-07-31)

- Feature: If you generate some documentation via *z3c.recipe.sphinxdoc* and want to stitch links to it into the generated index file, you can use the new `z3csphinx-output-directory` parameter to point the script to the generated package documentation root folder.

1.4.24 1.4 (2010-07-31)

- Bug: Don't clean up intermediate files, otherwise it is not possible to re-use a template folder for creating several separate pages into an output folder.
- Bug: Clean up group header creation to avoid header level mixups.
- Bug: When creating a missing required index.rst, use a template file if it exists.

1.4.25 1.3 (2010-07-30)

- Feature: Added a script and buildout option `index-name` to specify the file name (without extension) for the index page. With this option you can safely build the index page into an existing *Sphinx* documentation folder without overwriting or changing the existing `index.rst` file and its HTML equivalent. The default continues to be `index.rst`, though.
- Feature removed: It is no longer possible to create a simple HTML index page without using *Sphinx* and a minimal *Sphinx* configuration.

1.4.26 1.2 (2010-07-29)

- Feature: Add new script option `-g`/`--grouping` and zc.buildout option `grouping` to group packages.
- Miscellaneous: Renamed the zc.buildout option *source* to *sources* since it contains one or more elements.
- Miscellaneous: Removed the version pinning on the Sphinx dependency since our other dependency (repoze.sphinx.autointerface) is now compatible with Sphinx 1.0.
- Bug: If pkg_resources.find_distributions cannot find valid Egg distributions we still force the tag folder itself into the pkg_resources.working_set as a fallback.

1.4.27 1.1 (2010-07-25)

- Feature: The user can now provide a Sphinx configuration folder path that will be used to generate additional content for the documentation root folder.
- Factoring: Moved the DocsBuilder class into its own module.
- Factoring: Save run state on the documentation builder class instead of handing it around
- Cosmetic: Use a flat hierarchy when creating the HTML output links instead of a folder per package. Only a single index page needs to be created that way.

1.4.28 1.0 (2010-07-23)

- Initial release

CHAPTER

TWO

SUPPORT

If you need commercial support for this software package, please visit https://www.zetwork.com.

CHAPTER

THREE

INDICES AND TABLES

- genindex
- glossary