Eggshell Documentation

Release 0.4.0

Nils Hempelmann

Jan 10, 2019

Contents

Eggshell is meant as a collection of utilities for PyWPS servers. It covers miscellaneous functionality like file handling and common netCDF operations.

Eggshell is part of the Birdhouse project.

Eggshell was inspired by FlyingPigeon. We encourage developers to contribute their own code to eggshell to make it one of the building blocks of climate related PyWPS services.

- Free software: Apache Software License 2.0
- Documentation: https://eggshell.readthedocs.io.

CHAPTER 1

Features

CHAPTER 2

Credits

This package was created with Cookiecutter and the audreyr/cookiecutter-pypackage project template.

2.1 Installation

2.1.1 Install with Conda

Install *eggshell* with the following command:

```
$ conda install -c birdhouse -c conda-forge birdhouse-eggshell
```

This is the preferred method to install Eggshell, as it will always install the most recent stable release.

2.1.2 Install from GitHub

The sources for Eggshell can be downloaded from the Github repo.

Check out code from the birdy GitHub repo and start the installation:

```
$ git clone git://github.com/bird-house/eggshell
$ cd eggshell
$ conda env create -f environment.yml
$ python setup.py install
```

2.2 Usage

Eggshell is organized in submodules which are ordered by thematic functionality.

To use Eggshell in a project:

```
import eggshell
```

Example:

```
from eggshell import utils
tar_output = utils.archive(['tas.nc', 'tasmax.nc'], output_dir=workdir)
```

2.2.1 Legacy Modules

Warning: The legacy modules are deprecated and only kept as reference.

The ledacy modules are from the *FlyingPigeon* project and need to be refactored for common usage. You can find them in the *legacy* package. Check the documentation in the API reference.

2.3 Contributing

Contributions are welcome, and they are greatly appreciated! Every little bit helps, and credit will always be given.

You can contribute in many ways:

2.3.1 Report Bugs

Report bugs at https://github.com/bird-house/eggshell/issues.

If you are reporting a bug, please include:

- Your operating system name and version.
- Any details about your local setup that might be helpful in troubleshooting.
- Detailed steps to reproduce the bug.

2.3.2 Fix Bugs

Look through the GitHub issues for bugs. Anything tagged with "bug" and "help wanted" is open to whoever wants to implement it.

2.3.3 Implement Features

Look through the GitHub issues for features. Anything tagged with "enhancement" and "help wanted" is open to whoever wants to implement it.

2.3.4 Write Documentation

Eggshell could always use more documentation, whether as part of the official Eggshell docs, in docstrings, or even on the web in blog posts, articles, and such.

2.3.5 Submit Feedback

The best way to send feedback is to file an issue at https://github.com/bird-house/eggshell/issues.

If you are proposing a feature:

- Explain in detail how it would work.
- Keep the scope as narrow as possible, to make it easier to implement.
- Remember that this is a volunteer-driven project, and that contributions are welcome :)

2.4 Development

2.4.1 Get Started!

Ready to contribute? Here's how to set up *eggshell* for local development.

1. Fork the *eggshell* repo on GitHub.
2. Clone your fork locally:

```
$ git clone git@github.com:your_name_here/eggshell.git
```

3. Install your local copy into a conda environment. Assuming you have conda installed, this is how you set up your fork for local development:

```
$ cd eggshell/
$ conda env create -f environment.yml
$ conda activate eggshell
$ pip install -r requirements_dev.txt # install develop tools like pytest
$ python setup.py develop
```

4. Create a branch for local development:

```
$ git checkout -b name-of-your-bugfix-or-feature
```

   Now you can make your changes locally.

5. When you're done making changes, check that your changes pass flake8 and the tests:

```
$ flake8
$ pytest
```

   Or use the Makefile:

```
$ make lint
$ make test
$ make test-all
```

6. Commit your changes and push your branch to GitHub:

```
$ git add .
$ git commit -m "Your detailed description of your changes."
$ git push origin name-of-your-bugfix-or-feature
```

7. Submit a pull request through the GitHub website.

2.4.2 Write Documentation

You can find the documentation in the *docs/source* folder. To generate the Sphinx documentation locally you can use the *Makefile*:

```
$ make docs
```

2.4.3 Pull Request Guidelines

Before you submit a pull request, check that it meets these guidelines:

1. The pull request should include tests.
2. If the pull request adds functionality, the docs should be updated. Put your new functionality into a function with a docstring, and add the feature to the list in README.rst.
3. The pull request should work for Python 2.7, 3.6 and 3.7, and for PyPy. Check https://travis-ci.org/bird-house/eggshell/pull_requests and make sure that the tests pass for all supported Python versions.

2.4.4 Tips

To run a subset of tests:

```
$ pytest tests.test_utils
```

2.4.5 Deploying

A reminder for the maintainers on how to deploy. Make sure all your changes are committed (including an entry in CHANGES.rst). Then run:

```
$ bumpversion patch # possible: major / minor / patch
$ git push
$ git push --tags
```

Travis will then deploy to PyPI if tests pass.

2.5 Credits

2.5.1 Development Lead

- Nils Hempelmann <info@nilshempelmann.de>

2.5.2 Contributors

- David Huard <huard.david@ouranos.ca>
- Carsten Ehbrecht <ehbrecht@dkrz.de>

2.6 CHANGES

2.6.1 0.4.0 (2018-11-21)

Changes by PR #26:

- regenerated Eggshell project using *cookiecutter* template.
- moved previous code from *flyingpigeon* to *legacy* package.
- added *eggshell.utils* module.
- updated documentation to use conda environemnt.
- enabled Python 3.7 test on Travis.

2.6.2 0.3.0 (2018-07-30)

- updates from flyingpigeon (#21)
- added conda badges (#15)
- setup travis (#7)
- moved tests to top-level folder (#9)
- removed buildout (#8)
- moved heavy dependencies to subpackages (#5)

2.6.3 0.2.0 (2018-06-01)

- eggshell as conda package (#4)

2.6.4 0.1.0 (2018-02-13)

Montreal Release

- moved code to github.
- Initial Release.

2.7 Sphinx AutoAPI Index

This page is the top-level of your generated API documentation. Below is a list of all items that are documented here.

2.7.1 eggshell

Top-level package for Eggshell.

eggshell is a function library to be used in birds of birdhouse. eggshell is organised in subpackages ordered thematically.

Subpackages

eggshell.eo

EO subpackage of Eggshell.

eggshell.eo contains functions related to Earth-Obersvation analysis

Submodules

eggshell.eo.gdal_merge

Module Contents

eggshell.eo.gdal_merge.**progress**

eggshell.eo.gdal_merge.**__version__**

eggshell.eo.gdal_merge.**verbose = 0**

eggshell.eo.gdal_merge.**quiet = 0**

eggshell.eo.gdal_merge.**DoesDriverHandleExtension**(*drv*, *ext*)

eggshell.eo.gdal_merge.**GetExtension**(*filename*)

eggshell.eo.gdal_merge.**GetOutputDriversFor**(*filename*)

eggshell.eo.gdal_merge.**GetOutputDriverFor**(*filename*)

eggshell.eo.gdal_merge.**raster_copy**(*s_fh*, *s_xoff*, *s_yoff*, *s_xsize*, *s_ysize*, *s_band_n*, *t_fh*, *t_xoff*, *t_yoff*, *t_xsize*, *t_ysize*, *t_band_n*, *nodata=None*)

eggshell.eo.gdal_merge.**raster_copy_with_nodata**(*s_fh*, *s_xoff*, *s_yoff*, *s_xsize*, *s_ysize*, *s_band_n*, *t_fh*, *t_xoff*, *t_yoff*, *t_xsize*, *t_ysize*, *t_band_n*, *nodata*)

eggshell.eo.gdal_merge.**raster_copy_with_mask**(*s_fh*, *s_xoff*, *s_yoff*, *s_xsize*, *s_ysize*, *s_band_n*, *t_fh*, *t_xoff*, *t_yoff*, *t_xsize*, *t_ysize*, *t_band_n*, *m_band*)

eggshell.eo.gdal_merge.**names_to_fileinfos**(*names*)

Translate a list of GDAL filenames, into file_info objects.

names – list of valid GDAL dataset names.

Returns a list of file_info objects. There may be less file_info objects than names if some of the names could not be opened as GDAL files.

class eggshell.eo.gdal_merge.**file_info**

A class holding information about a GDAL file.

init_from_name(*self*, *filename*)

Initialize file_info from filename

filename – Name of file to read.

Returns 1 on success or 0 if the file can't be opened.

report(*self*)

copy_into(*self*, *t_fh*, *s_band=1*, *t_band=1*, *nodata_arg=None*)
Copy this files image into target file.

This method will compute the overlap area of the file_info objects file, and the target gdal.Dataset object, and copy the image data for the common window area. It is assumed that the files are in a compatible projection ... no checking or warping is done. However, if the destination file is a different resolution, or different image pixel type, the appropriate resampling and conversions will be done (using normal GDAL promotion/demotion rules).

t_fh – gdal.Dataset object for the file into which some or all of this file may be copied.

Returns 1 on success (or if nothing needs to be copied), and zero one failure.

eggshell.eo.gdal_merge.**Usage**()

eggshell.eo.gdal_merge.**main**(*argv=None*)

eggshell.nc

NC subpackage of Eggshell

eggshell.nc contains functions related to netCDF4 processing

Submodules

eggshell.nc.calculation

Module Contents

eggshell.nc.calculation.**LOGGER**

eggshell.nc.calculation.**fieldmean**(*resource*)
calculating of a weighted field mean

Parameters **resource** – str or list of str containing the netCDF files paths

Return list timeseries of the averaged values per timestep

eggshell.nc.calculation.**remove_mean_trend**(*fana*, *varname*)
Removing the smooth trend from 3D netcdf file

eggshell.nc.cdo_utils

Placeholder for cdo based functions

eggshell.nc.nc_fetch

Module Contents

eggshell.nc.nc_fetch.**paths**

eggshell.nc.nc_fetch.**LOGGER**

eggshell.nc.nc_fetch.**_PRESSUREDATA_ = ['NCEP_slp', 'NCEP_z1000', 'NCEP_z925', 'NCEP_z850',**

eggshell.nc.nc_fetch.**_EOBSVARIABLES_ = ['tg', 'tx', 'tn', 'rr']**

eggshell.nc.nc_fetch.**reanalyses**(*start=1948*, *end=None*, *variable='slp'*, *dataset='NCEP'*, *tim-res='day'*, *getlevel=True*)

Fetches the reanalysis data (NCEP, 20CR or ERA_20C) to local file system

Parameters

- **start** – int for start year to fetch source data
- **end** – int for end year to fetch source data (if None, current year will be the end)
- **variable** – variable name (default='slp'), geopotential height is given as e.g. z700
- **dataset** – default='NCEP'

Return list list of path/files.nc

eggshell.nc.nc_fetch.**get_level**(*resource*, *level*)

eggshell.nc.nc_fetch.**write_fileinfo**(*resource*, *filepath=False*)

write path and filenames to a text file

Parameters

- **ressource** – list of files to be documented
- **filepath** – if True the absolute filepath is written out as well (default = False)

Return txt textfile with appropriate information

eggshell.nc.nc_utils

Module Contents

eggshell.nc.nc_utils.**LOGGER**

class eggshell.nc.nc_utils.**CookieNetCDFTransfer**(*request*, *opendap_hostnames=[]*)

__enter__(*self*)

__exit__(*self*, *exc_type*, *exc_val*, *exc_tb*)

eggshell.nc.nc_utils.**aggregations**(*resource*)

aggregates netcdf files by experiment. Aggregation examples: CORDEX: EUR-11_ICHEC-EC-EARTH_historical_r3i1p1_DMI-HIRHAM5_v1_day CMIP5: We collect for each experiment all files on the time axis: 200101-200512, 200601-201012, ... Time axis is sorted by time. :param resource: list of netcdf files :return: dictionary with key=experiment

eggshell.nc.nc_utils.**get_auth_cookie**(*pywps_request*)

eggshell.nc.nc_utils.**opendap_or_download**(*resource*, *auth_tkt_cookie={}*, *output_path=None*, *max_nbytes=10000000000*)

Check for OPEnDAP support, if not download the resource. :param resource: url of a NetCDF resource :param output_path: where to save the non-OPEnDAP resource :param max_nbytes: maximum file size for download, default: 1 gb :return str: the original url if OPEnDAP is supported or path of saved file

eggshell.nc.nc_utils.**get_coordinates**(*resource*, *variable=None*, *unrotate=False*)

reads out the coordinates of a variable :param resource: netCDF resource file :param variable: variable name :param unrotate: If True the coordinates will be returned for unrotated pole :returns list, list: latitudes , longitudes

eggshell.nc.nc_utils.**get_index_lat**(*resource*, *variable=None*)
returns the dimension index of the latiude values :param resource: list of path(s) to netCDF file(s) of one Dataset :param variable: variable name :return int: index

eggshell.nc.nc_utils.**get_variable**(*resources*)
Guess main variables in a NetCDF file. (compare nc.ocg_utils.get_variable)

Parameters **resources** – netCDF4.Dataset

Return list names of main variables

The main variables are the one with highest dimensionality and size. The time, lon, lat variables and variables that are defined as bounds are automatically ignored.

eggshell.nc.nc_utils.**sort_by_filename**(*resource*, *historical_concatination=False*)
Sort a list of files with CORDEX-conformant file names.

Parameters

- **resource** – netCDF file
- **historical_concatination** – if True (default=False), appropriate historial runs will be sorted to the rcp datasets

Return dictionary {'drs_filename': [list of netCDF files]}

eggshell.nc.nc_utils.**get_frequency**(*resource*)
returns the frequency as set in the metadata (see also metadata.get_frequency)

Parameters **resource** – NetCDF file

Return str frequency

eggshell.nc.nc_utils.**get_timerange**(*resource*)
returns from/to timestamp of given netcdf file(s).

Parameters **resource** – list of path(s) to netCDF file(s)

Returns netcdf.datetime.datetime start, end

eggshell.nc.nc_utils.**get_time**(*resource*)
returns all timestamps of given netcdf file as datetime list.

Parameters **resource** – NetCDF file(s)

:return : list of timesteps

eggshell.nc.nc_utils.**get_values**(*resource*, *variable=None*)
returns the values for a list of files of files belonging to one dataset

Parameters

- **resource** – list of files
- **variable** – variable to be picked from the files (if not set, variable will be detected)

Returs numpy.array values

eggshell.nc.nc_utils.**drs_filename**(*resource*, *skip_timestamp=False*, *skip_format=False*, *variable=None*, *rename_file=False*, *add_file_path=False*)
generates filename according to the data reference syntax (DRS) based on the metadata in the resource. http://cmip-pcmdi.llnl.gov/cmip5/docs/cmip5_data_reference_syntax.pdf https://pypi.python.org/pypi/drslib :param add_file_path: if add_file_path=True, path to file will be added (default=False) :param resource: netcdf file :param skip_timestamp: if True then from/to timestamp != added to the filename

(default: False)

Parameters

- **variable** – appropriate variable for filename, if not set (default), variable will be determined. For files with more than one data variable, the variable parameter has to be defined (default:) example: variable='tas'
- **rename_file** – rename the file. (default: False)

Returns str DRS filename

eggshell.nc.nc_utils.**sort_by_time**(*resource*)
Sort a list of files by their time variable.

Parameters **resource** – File path.

Returns Sorted file list.

eggshell.nc.ocg_utils

Module Contents

eggshell.nc.ocg_utils.**LOGGER**

eggshell.nc.ocg_utils.**temp_groups**

eggshell.nc.ocg_utils.**call**(*resource=[]*, *variable=None*, *dimension_map=None*, *agg_selection=True*, *calc=None*, *calc_grouping=None*, *conform_units_to=None*, *crs=None*, *memory_limit=None*, *prefix=None*, *regrid_destination=None*, *regrid_options='bil'*, *level_range=None*, *geom=None*, *output_format_options=None*, *search_radius_mult=2.0*, *select_nearest=False*, *select_ugid=None*, *spatial_wrapping=None*, *t_calendar=None*, *time_region=None*, *time_range=None*, *dir_output=None*, *output_format='nc'*)
Call OCGIS operation.

Parameters

- **resource** – Input netCDF file.
- **variable** – variable in the input file to be picked
- **dimension_map** – dimension map in case of unconventional storage of data
- **agg_selection** – For aggregation of in case of mulitple polygons geoms
- **calc** – ocgis calc syntax for calculation partion
- **calc_grouping** – time aggregate grouping
- **cdover** – OUTDATED use py-cdo ('python', by default) or cdo from the system ('system')
- **conform_units_to** –
- **crs** – coordinate reference system
- **memory_limit** – limit the amount of data to be loaded into the memory at once if None (default) free memory is detected by birdhouse
- **level_range** – subset of given levels
- **prefix** – string for the file base name
- **regrid_destination** – file path with netCDF file with grid for output file

- **geom** – name of shapefile stored in birdhouse shape cabinet
- **output_format_options** – output options for netCDF e.g compression level()
- **regrid_destination** – file containing the targed grid (griddes.txt or netCDF file)
- **regrid_options** – methods for regridding: 'bil' = Bilinear interpolation 'bic' = Bicubic interpolation 'dis' = Distance-weighted average remapping 'nn' = nearest neighbour 'con' = First-order conservative remapping 'laf' = largest area fraction reamapping
- **search_radius_mult** – search radius for point geometries. All included gridboxes will be returned
- **select_nearest** – nearest neighbour selection for point geometries
- **select_ugid** – ugid for appropriate polygons
- **spatial_wrapping** – how to handle coordinates in case of subsets, options: None (default), 'wrap', 'unwrap'
- **time_region** – select single month
- **time_range** – sequence of two datetime.datetime objects to mark start and end point
- **(default= curdir)** (*dir_output*) –
- **output_format** –

Returns output file path

eggshell.nc.ocg_utils.**calc_grouping**(*grouping*)
translate time grouping abbreviation (e.g 'JJA') into the appropriate ocgis calc_grouping syntax

Parameters **grouping** – time group abbreviation allowed values: "yr", "mon", "sem", "OND-JFM", "AMJJAS", "DJF", "MAM", "JJA", "SON"

Returns list calc_grouping conformant to ocgis syntax

eggshell.nc.ocg_utils.**get_variable**(*resource*)
detects processable variable name in netCDF file based on ocgis (compare guess_main_variables)

Parameters **resource** – filepath sting or sorted list for netcdf file(s)

Returns str variable name

eggshell.nc.ocg_utils.**has_variable**(*resource*, *variable*)

eggshell.plot

Visual subpackage of eggshell

eggshell.visual contains functions for ploting and data visualisation purposes. the main dependancies are matplotlib and cartopy, for that proj4 is required.

Submodules

eggshell.plot.plt_eodata

Module Contents

eggshell.plot.plt_eodata.**LOGGER**

eggshell.plot.plt_eodata.**plot_products**(*products, extend=[10, 20, 5, 15], dir_output='.'*)
plot the products extends of the search result

Parameters

- **products** – output of sentinelapi search
- **dir_output** – path to folder where to store the returned graphic

Parm extend extend of the background map to be plotted

Return png map of extents

eggshell.plot.plt_ncdata

visualization of netCDF data

Module Contents

eggshell.plot.plt_ncdata.**LOGGER**

eggshell.plot.plt_ncdata.**plot_extend**(*resource*, *file_extension='png'*)
plots the extend (domain) of the values stored in a netCDF file:

Parm resource path to netCDF file

Parameters **file_extension** – file format of the graphic. if file_extension=None a matplotlib figure will be returned

Return graphic graphic in specified format

eggshell.plot.plt_ncdata.**spaghetti**(*resouces*, *variable=None*, *title=None*, *file_extension='png'*, *dir_output='.'*)
creates a png file containing the appropriate spaghetti plot as a field mean of the values.

Parameters

- **resouces** – list of files containing the same variable
- **variable** – variable to be visualised. If None (default), variable will be detected
- **title** – string to be used as title

Retruns str path to png file

eggshell.plot.plt_ncdata.**uncertainty**(*resouces*, *variable=None*, *ylim=None*, *title=None*, *file_extension='png'*, *window=None*, *dir_output='.'*)
creates a png file containing the appropriate uncertainty plot.

Parameters

- **resouces** – list of files containing the same variable
- **variable** – variable to be visualised. If None (default), variable will be detected
- **title** – string to be used as title
- **window** – windowsize of the rolling mean

Returns str path/to/file.png

eggshell.plot.plt_ncdata.**plot_spatial_analog**(*ncfile*, *variable='dissimilarity'*, *cmap='viridis'*, *title='Spatial analog'*)
Return a matplotlib Figure instance showing a map of the dissimilarity measure.

eggshell.plot.plt_utils

Module Contents

eggshell.plot.plt_utils.**LOGGER**

eggshell.plot.plt_utils.**fig2plot**(*fig*, *file_extension='png'*, *dir_output='.'*, *bbox_inches='tight'*, *dpi=300*, *facecolor='w'*, *edgecolor='k'*, *figsize=(20, 10)*)

saving a matplotlib figure to a graphic

Parameters

- **fig** – matplotlib figure object
- **dir_output** – directory of output plot
- **file_extension** – file file_extension (default='png')

Return str path to graphic

class eggshell.plot.plt_utils.**MidpointNormalize**(*vmin=None*, *vmax=None*, *midpoint=None*, *clip=False*)

Bases:matplotlib.colors.Normalize

__call__(*self*, *value*, *clip=None*)

eggshell.plot.plt_utils.**concat_images**(*images*, *orientation='v'*, *dir_output='.'*)

concatenation of images.

Parameters

- **images** – list of images
- **orientation** – vertical ('v' default) or horizontal ('h') concatenation
- **dir_output** – output directory

Return string path to image

eggshell.plot.plt_utils.**pdfmerge**(*pdfs*, *dir_output='.'*)

merge a list of pdfs

Parameters **pdfs** – list of pdf files

Parm dir_output output directory

Retun str merged pdf

Submodules

eggshell.config

WPS servers often need to specify a number of paths for processes to find data, shapefiles, caches and determine where outputs are stored. To make sure all birds use the same architecture, eggshell provides a *Paths* class to help with this.

Module Contents

eggshell.config.**LOGGER**

class eggshell.config.**Paths**(*module*)
Bases:object

This class facilitates the configuration of WPS birds.

top_level
return the top level directory of a WPS bird

data
Return the path to the data directory.

shapefiles
Return the path to the geographic data directory.

testdata
Return the path to the test data directory.

cache
Return the path to the server cache directory.

outputpath
Return the server directory for process outputs.

outputurl
Return the server URL for process outputs.

eggshell.config.**esgfsearch_distrib**()
TODO

eggshell.config.**esgfsearch_url**()
Return the server configuration value for the ESGF search node URL.

eggshell.dependencies

This module is used to manage optional dependencies.

Example usage:

```
from eggshell.dependencies import netCDF4 as nc
```

Module Contents

eggshell.dependencies.**netCDF4**

eggshell.exceptions

Module Contents

exception eggshell.exceptions.**CalculationException**
Bases:Exception

eggshell.log

Progress and errors in WPS processes are logged by the server. The initialization of the log file for each process is done using the *init_process_logger()*.

Module Contents

eggshell.log.**init_process_logger**(*filename=None*)
Connect and initialize the logging mechanism to a given file.

Parameters **filename** (*str*) – Logging file name. Defaults to log.txt

eggshell.utils

Utitility functions.

Module Contents

eggshell.utils.**paths**

eggshell.utils.**LOGGER**

eggshell.utils.**archive**(*resources*, *format='tar'*, *dir_output=None*, *mode=None*)
Compresses a list of files into an archive.

Parameters

- **resources** – list of files to be stored in archive
- **format** – archive format. Options: tar (default), zip
- **dir_output** – path to output folder (default: tempory folder)
- **mode** – for format='tar': 'w' or 'w:' open for writing without compression 'w:gz' open for writing with gzip compression 'w:bz2' open for writing with bzip2 compression 'w|' open an uncompressed stream for writing 'w|gz' open a gzip compressed stream for writing 'w|bz2' open a bzip2 compressed stream for writing

  for foramt='zip': read "r", write "w" or append "a"

Return str archive path/filname.ext

eggshell.utils.**download_file**(*url*, *out=None*, *verify=False*)

eggshell.utils.**local_path**(*url*)

eggshell.utils.**download**(*url*, *cache=False*)
Downloads URL using the Python requests module to the current directory.

Parameters

- **cache** – if True then files will be downloaded to a cache directory.
- **url** – url adress of the target file location

Return str filename

eggshell.utils.**extract_archive**(*resources*, *dir_output=None*)
extracts archives (tar/zip)

Parameters

- **resources** – list of archive files (if netCDF files are in list, they are passed and returnd as well in the return).
- **dir_output** – define a directory to store the results (default: tempory folder).

Return list [list of extracted files]

eggshell.utils.**rename_complexinputs**(*complexinputs*)
TODO: this method is just a dirty workaround to rename input files according to the url name.

eggshell.utils.**address_append**(*address*)
Formats a URL/URI to be more easily read with libraries such as "rasterstats"

Parameters **address** – URL/URI to a potential zip or tar file

Returns URL/URI prefixed for archive type

Package Contents

eggshell.**__author__ = Nils Hempelmann**

eggshell.**__email__ = info[@]nilshempelmann.de**

eggshell.**__version__ = 0.4.0**

Python Module Index

e

Index

Symbols

A

C

D

E

F

G

H

I

L

M

N

O

P

Q

R

S

T

U

V

W