Jup2Jek Documentation

Release 1.0.0

Matt Pewsey

Dec 22, 2018

Contents

CHAPTER 1

About

The Jup2Jek package is a tool for converting Jupyter notebooks to markdown for use as posts on Jekyll websites, such those hosted using GitHub Pages.

CHAPTER 2

Installation & Usage

To use the package, first install it via pip:

```
pip install git+https://github.com/mpewsey/jup2jek#egg=jup2jek
```

Next, create a configuration file *jup2jek.ini*, with the following sections and place it in your website project's root directory. Configure the options as you feel is suited to your particular project.

```
# jup2jek.ini
[JUP2JEK]
# The directory containing your posts and Jupyter notebooks to be converted
posts = _posts
# The directory where assets generated by converted notebooks will be placed
assets = assets/jupyter
```

Warning: The specified assets path should contain no other files. Otherwise those files will be deleted during the conversion process.

From your root directory, run the below command to convert notebooks.

```
jup2jek
```

The method performs the following actions:

1. All notebooks contained in the designated posts folder are found and converted to markdown using the jupyter nbconvert command.
2. If the nbconvert command generates an assets folder, the assets are moved to the designated site assets folder at the same relative path.
3. Asset paths referenced within the markdown files are updated to the website path.

Each time you add additional notebooks, you will need to run this command again.

Notebooks should be named in the same format as typical Jekyll posts, i.e. YYYY-MM-DD-post.ipynb. In addition, the first cell of the Jupyter notebook should contain YAML front matter, as would be contained in an typical post. This cell should be formatted as an nbconvert cell so that is it converted exactly as entered. For example:

```
---
layout: post
title: Post Title
categories: jekyll
date: 2018-03-28
---
```

CHAPTER 3

API Documentation

3.1 Jup2Jek (`jup2jek`)

3.1.1 Classes

The following classes are used for notebook conversion and script parsing:

Jup2Jek(root[, options])	A class for converting Jupyter notebooks to markdowns for use on Jekyll websites.
Jup2JekArgParser()	Command line argument parser for *jup2jek* script.

jup2jek.Jup2Jek

class jup2jek.**Jup2Jek**(*root*, *options=None*)

A class for converting Jupyter notebooks to markdowns for use on Jekyll websites.

Parameters

root [str] The path to the website root directory.

options [dict] The path to the configuration file. If None, then the default configuration file *root/jup2jek.ini* will be used.

Examples

The contents of an example configuration file are below:

```
>>> [JUP2JEK]
>>> posts = _posts
>>> assets = assets/jupyter
```

posts The relative path to the posts directory from the website root. This is *_posts* for default Jekyll site configurations.

assets The relative path to the assets folder where generated notebook assets will be stored. This is relative to the website root.

> **Warning:** This folder will be cleared each time *convert_notebooks()* is called. Therefore, the folder should be separate from those used for other website assets. Otherwise those assets will be deleted.

Methods

assets_path()	Returns the assets path.
convert(path)	Converts the notebook at the specified path to markdown via the *jupyter nbconvert [notebook path] –to markdown* command.
convert_notebooks()	Converts all Jupyter notebooks within the posts folder specified in the class options to markdown for use on Jekyll websites.
load_options(path)	Loads the options from the specified configuration file.
notebooks()	Returns a list of notebook paths to be converted.
posts_path()	Returns the posts path.
write_default_options(path)	Writes a configuration file with the default options.

jup2jek.Jup2Jek.assets_path

Jup2Jek.**assets_path**()
Returns the assets path.

jup2jek.Jup2Jek.convert

Jup2Jek.**convert**(*path*)
Converts the notebook at the specified path to markdown via the *jupyter nbconvert [notebook path] –to markdown* command.

Parameters

path [str] Path to the jupyter notebook.

jup2jek.Jup2Jek.convert_notebooks

Jup2Jek.**convert_notebooks**()
Converts all Jupyter notebooks within the posts folder specified in the class options to markdown for use on Jekyll websites. The convert process operates as follows:

1. All notebooks contained in the designated posts folder are found and converted to markdown using the jupyter nbconvert command.
2. If the nbconvert command generates an assets folder, the assets are moved to the designated site assets folder at the same relative path.

3. Asset paths referenced within the markdown files are updated to the website path.

jup2jek.Jup2Jek.load_options

Jup2Jek.**load_options**(*path*)
Loads the options from the specified configuration file.

Parameters

path [str] Configuration file path.

jup2jek.Jup2Jek.notebooks

Jup2Jek.**notebooks**()
Returns a list of notebook paths to be converted.

jup2jek.Jup2Jek.posts_path

Jup2Jek.**posts_path**()
Returns the posts path.

jup2jek.Jup2Jek.write_default_options

static Jup2Jek.**write_default_options**(*path*)
Writes a configuration file with the default options.

Parameters

path [str] Directory for writing the configuration file. The file name should not be included and will be appended as part of the method.

jup2jek.Jup2JekArgParser

class jup2jek.**Jup2JekArgParser**
Command line argument parser for *jup2jek* script.

Methods

add_argument(dest, ... [, name, name])	
add_argument_group(*args, **kwargs)	
add_mutually_exclusive_group(**kwargs)	
add_subparsers(**kwargs)	
convert_arg_line_to_args(arg_line)	
error(message)	Prints a usage message incorporating the message to stderr and exits.
exit([status, message])	

Continued on next page

Table 3 – continued from previous page

format_help()
format_usage()
format_version()
get_default(dest)
parse_args([args, namespace])
parse_known_args([args, namespace])
print_help([file])
print_usage([file])
print_version([file])
register(registry_name, value, object)
set_defaults(**kwargs)

jup2jek.Jup2JekArgParser.add_argument

Jup2JekArgParser.**add_argument**(*dest*, ..., *name=value*, ...) *add_argument(option_string*, *option_string*, ..., *name=value*, ...)

jup2jek.Jup2JekArgParser.add_argument_group

Jup2JekArgParser.**add_argument_group**(**args*, ***kwargs*)

jup2jek.Jup2JekArgParser.add_mutually_exclusive_group

Jup2JekArgParser.**add_mutually_exclusive_group**(***kwargs*)

jup2jek.Jup2JekArgParser.add_subparsers

Jup2JekArgParser.**add_subparsers**(***kwargs*)

jup2jek.Jup2JekArgParser.convert_arg_line_to_args

Jup2JekArgParser.**convert_arg_line_to_args**(*arg_line*)

jup2jek.Jup2JekArgParser.error

Jup2JekArgParser.**error**(*message: string*)

Prints a usage message incorporating the message to stderr and exits.

If you override this in a subclass, it should not return – it should either exit or raise an exception.

jup2jek.Jup2JekArgParser.exit

Jup2JekArgParser.**exit**(*status=0*, *message=None*)

jup2jek.Jup2JekArgParser.format_help

Jup2JekArgParser.**format_help**()

jup2jek.Jup2JekArgParser.format_usage

Jup2JekArgParser.**format_usage**()

jup2jek.Jup2JekArgParser.format_version

Jup2JekArgParser.**format_version**()

jup2jek.Jup2JekArgParser.get_default

Jup2JekArgParser.**get_default**(*dest*)

jup2jek.Jup2JekArgParser.parse_args

Jup2JekArgParser.**parse_args**(*args=None*, *namespace=None*)

jup2jek.Jup2JekArgParser.parse_known_args

Jup2JekArgParser.**parse_known_args**(*args=None*, *namespace=None*)

jup2jek.Jup2JekArgParser.print_help

Jup2JekArgParser.**print_help**(*file=None*)

jup2jek.Jup2JekArgParser.print_usage

Jup2JekArgParser.**print_usage**(*file=None*)

jup2jek.Jup2JekArgParser.print_version

Jup2JekArgParser.**print_version**(*file=None*)

jup2jek.Jup2JekArgParser.register

Jup2JekArgParser.**register**(*registry_name*, *value*, *object*)

jup2jek.Jup2JekArgParser.set_defaults

Jup2JekArgParser.**set_defaults**(***kwargs*)

Python Module Index

j

Index