sphinxcontrib-jupyter Documentation

Release 19.2.1

QuantEcon Development Team

Jul 30, 2019

Contents:

This sphinx extension can be used to build a collection of Jupyter notebooks for Sphinx Projects.

Note: It has mainly been written to support the use case of scientific publishing and hasn't been well tested outside of this domain. Please provide feedback as an issue to this repository.

Requires: Sphinx >= 1.7.2 (for running tests).

One of the main benefits of writing Jupyter notebooks as `RST` files is to simplify the task of version control for large projects.

CHAPTER 1

Installation

To install the extension:

```
pip install sphinxcontrib-jupyter
```

to upgrade your current installation to the latest version:

```
pip install sphinxcontrib-jupyter --upgrade
```

Todo: Add installation via conda-forge

1.1 Alternative

Another way to get the **latest** version it is to install directly by getting a copy of the repository:

```
git clone https://github.com/QuantEcon/sphinxcontrib-jupyter
```

and then use

```
python setup.py install
```

1.2 Developers

For developers it can be useful to install using the *develop* option:

```
python setup.py develop
```

this will install the package into the *site-wide* package directory which is linked to the code in your local copy of the repository. It is **not** recommended to install this way for common use.

CHAPTER 2

Sphinx Setup

To initially setup a Sphinx project, please refer here.

Note: QuantEcon is currently developing a custom quickstart to assist with setting up a sphinx project customised to use this extension and provide more guidance with the configuration process.

Update the project `conf.py` file to include the jupyter extension and add the desired configuration settings (see *Extension Configuration* section for details):

```
extensions = ["sphinxcontrib.jupyter"]
```

once the extension is installed you can then run:

```
make jupyter
```

The *Extension Configuration* section includes details on how to configure the extension.

CHAPTER 3

Extension Configuration and Options

The options are split into the different parts of the compilation pipeline that are available in this extension:

3.1 Constructing Jupyter Notebooks

Options

- *jupyter_conversion_mode*
- *jupyter_static_file_path*
- *jupyter_header_block*
- *jupyter_default_lang*
- *jupyter_lang_synonyms*
- *jupyter_kernels*
- *jupyter_write_metadata*
- *jupyter_options*
- *jupyter_drop_solutions*
- *jupyter_drop_tests*
- *jupyter_ignore_no_execute:*
- *jupyter_ignore_skip_test*
- *jupyter_allow_html_only*
- *jupyter_target_html*
- *jupyter_images_markdown*

3.1.1 jupyter_conversion_mode

Specifies which writer to use when constructing notebooks.

Option	Description
"all" (**default**)	compile complete notebooks which include `markdown cells` and `code blocks`
"code"	compile notebooks that only contain the `code blocks`.

`conf.py` usage:

```
jupyter_conversion_mode = "all"
```

3.1.2 jupyter_static_file_path

Specify path to *_static* folder.

`conf.py` usage:

```
jupyter_static_file_path = ["source/_static"]
```

3.1.3 jupyter_header_block

Add a header block to every generated notebook by specifying an RST file

`conf.py` usage:

```
jupyter_header_block = ["source/welcome.rst"]
```

3.1.4 jupyter_default_lang

Specify default language for collection of RST files

`conf.py` usage:

```
jupyter_default_lang = "python3"
```

3.1.5 jupyter_lang_synonyms

Specify any language synonyms.

This will be used when parsing code blocks. For example, python and ipython have slightly different highlighting directives but contain code that can both be executed on the same kernel

`conf.py` usage:

```
jupyter_lang_synonyms = ["pycon", "ipython"]
```

3.1.6 jupyter_kernels

Specify kernel information for the jupyter notebook metadata.

This is used by jupyter to connect the correct language kernel and is **required** in `conf.py`.

`conf.py` usage:

```
jupyter_kernels = {
    "python3": {
        "kernelspec": {
            "display_name": "Python",
            "language": "python3",
            "name": "python3"
            },
        "file_extension": ".py",
    },
}
```

Todo: See Issue 196

3.1.7 jupyter_write_metadata

write time and date information at the top of each notebook as notebook metadata

Note: This option is slated to be deprecated

3.1.8 jupyter_options

An dict-type object that is used by dask to control execution

Todo: This option needs to be reviewed

3.1.9 jupyter_drop_solutions

Drop `code-blocks` that include `:class: solution`

Values
False (**default**)
True

Todo: This option needs to be reviewed

3.1.10 jupyter_drop_tests

Drop `code-blocks` that include ``:class: test`

Values
False (**default**)
True

Todo: This option needs to be reviewed

3.1.11 jupyter_ignore_no_execute:

Values
False (**default**)
True

When constructing notebooks this option can be enabled to ignore *:class: no-execute* for *code-blocks*. This is useful for *html* writer for pages that are meant to fail but shouldn't be included in *coverage* tests.

`conf.py` usage:

```
jupyter_ignore_no_execute = True
```

3.1.12 jupyter_ignore_skip_test

When constructing notebooks this option can be enabled to ignore *:class: skip-test* for *code-blocks*.

Values
False (**default**)
True

`conf.py` usage:

```
jupyter_ignore_skip_test = True
```

3.1.13 jupyter_allow_html_only

Enable this option to allow `.. only:: html` pass through to the notebooks.

Values
False (**default**)
True

`conf.py` usage:

```
jupyter_allow_html_only = True
```

3.1.14 jupyter_target_html

Enable this option to generate notebooks that favour the inclusion of `html` in notebooks to support more advanced features.

Values
False (**default**)
True

Supported Features:

1. html based table support
2. image inclusion as `html` figures

`conf.py` usage:

```
jupyter_target_html = True
```

3.1.15 jupyter_images_markdown

Force the inclusion of images as native markdown

Values
False (**default**)
True

Note: when this option is enabled the *:scale:* option is not supported in RST.

`conf.py` usage:

```
jupyter_images_markdown = True
```

3.2 Executing Notebooks

3.2.1 jupyter_execute_nb

Enables the execution of generated notebooks

Values
False (**default**)
True

Todo: deprecate this option in favour of jupyter_execute_notebooks

3.2.2 jupyter_execute_notebooks

Enables the execution of generated notebooks

Values
False (**default**)
True

`conf.py` usage:

```
jupyter_execute_notebooks = True
```

3.2.3 jupyter_dependency_lists

Dependency of notebooks on other notebooks for execution can also be added to the configuration file above in the form of a dictionary. The key/value pairs will contain the names of the notebook files.

`conf.py` usage:

```
# add your dependency lists here
jupyter_dependency_lists = {
   'python_advanced_features' : ['python_essentials','python_oop'],
   'discrete_dp' : ['dp_essentials'],
}
```

3.2.4 jupyter_number_workers

Specify the number cores to use with dask

Values
Integer (**default** = 1)

`conf.py` usage:

jupyter_number_workers = 4

3.2.5 jupyter_threads_per_worker

Specify the number of threads per worker for dask

Values
Integer (**default** = 1)

`conf.py` usage:

jupyter_threads_per_worker = 1

3.3 Converting Notebooks to HTML

Options

- *jupyter_generate_html*
- *jupyter_html_template*
- *jupyter_make_site*
- *jupyter_download_nb*
- *jupyter_images_urlpath*

3.3.1 jupyter_generate_html

Enable sphinx to generate HTML versions of notebooks

Values
False (**default**)
True

`conf.py` usage:

```
jupyter_generate_html = True
```

3.3.2 jupyter_html_template

Specify path to nbconvert html template file

Note: Documentation on nbconvert templates can be found here

`conf.py` usage:

```
jupyter_html_template = "theme/template/<file>.tpl"
```

3.3.3 jupyter_make_site

Enable sphinx to construct a complete website

Todo: Document all the extra elements this option does over jupyter_generate_html

This option:

1. fetches coverage statistics if coverage is enabled.

`conf.py` usage:

```
jupyter_make_site = True
```

3.3.4 jupyter_download_nb

Request Sphinx to generate a collection of download notebooks to support a website

`conf.py` usage:

```
jupyter_download_nb = True
```

3.3.5 jupyter_images_urlpath

Apply a url prefix when writing images in Jupyter notebooks. This is useful when paired with `jupyter_download_nb` so that download notebooks are complete with web referenced images.

`conf.py` usage:

```
jupyter_images_urlpath = "s3://<path>/_static/img/"
```

3.4 Computing Coverage Statistics

3.4.1 jupyter_make_coverage

Enable coverage statistics to be computed

Values
False (**default**)
True

3.4.2 jupyter_template_coverage_file_path

Provide path to template coverage file

Todo: Document format for template

`conf.py` usage:

```
jupyter_template_coverage_file_path = "theme/templates/<file>.json"
```

It can also be useful to have multiple configurations when working on a large project, such as generating notebooks for working on locally and editing and compiling the project for HTML in a deployment setting. Further details on how to manage large projects can be found here.

An example *conf.py* is available here

CHAPTER 4

RST Conversion Gallery

Note: A minimum configured sphinx repo is available here which generates a sample notebook

Examples

The test suite, located here provides examples of conversions between RST and the Jupyter notebook which form the test cases for this extension. It can be a useful resource to check how elements are converted if they are not contained in this gallery.

4.1 code-blocks

The following code in the **.rst** file

```
Code blocks
-----------

This is a collection to test various code-blocks

This is a **.. code::** directive

.. code:: python

    this = 'is a code block'
    x = 1
    no = 'really!'
    p = argwhere(x == 2)

This is another **.. code::** directive

.. code:: python

    from pylab import linspace
    t = linspace(0, 1)
    x = t**2

This is a **::** directive

::

    from pylab import *
    x = logspace(0, 1)
    y = x**2
    figure()
    plot(x, y)
    show()
```

will look as follows in the jupyter notebook

Code blocks

This is a collection to test various code-blocks

This is a **.. code::** directive

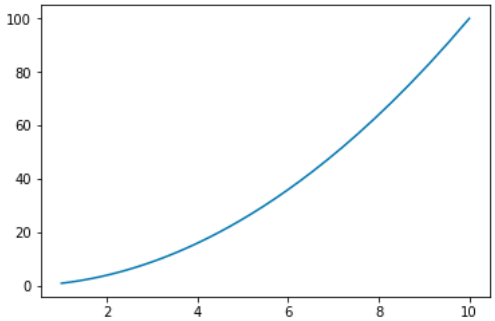
In [5]:
```
this = 'is a code block'
x = 1
no = 'really!'
p = argwhere(x == 2)
```

This is another **.. code::** directive

In [3]:
```
from pylab import linspace
t = linspace(0, 1)
x = t**2
```

This is a **::** directive

In [4]:
```
from pylab import *
x = logspace(0, 1)
y = x**2
figure()
plot(x, y)
show()
```



4.2 images and figures

The following code in the .rst file

```
Images
======

Collection of tests for **.. image::** and **.. figure::** directives

Image
-----

`Docutils Reference <http://docutils.sourceforge.net/docs/ref/rst/directives.html
↪#images>`__

Most basic image directive

.. image:: _static/hood.jpg

A scaled down version with 25 % width
```

(continues on next page)

(continued from previous page)

```
.. image:: _static/hood.jpg
:width: 25 %

A height of 50px

.. image:: _static/hood.jpg
:height: 50px

Figure
------

`Docutils Reference <http://docutils.sourceforge.net/docs/ref/rst/directives.html
↪#figure>`__

Testing the **.. figure::** directive

.. figure:: _static/hood.jpg
:scale: 50 %
```

will look as follows in the jupyter notebook

Image

Docutils Reference

Most basic image directive

A scaled down version with 25 % width

A height of 50px

Figure

Docutils Reference

Testing the **.. figure::** directive

Warning: if `jupyter_images_markdown = True` then the `:scale:`, `:height:` and *:width:'* attributes will be ignored.

4.3 jupyter-directive

The following code in the .rst file

```
Jupyter Directive
=================

This is a set of tests related to the Jupyter directive


The following jupyter directive with cell-break option should
split this text and the text that follows into different IN
blocks in the notebook

.. jupyter::
    :cell-break:

This text should follow in a separate cell.
```

will look as follows in the jupyter notebook

Jupyter Directive

This is a set of tests related to the Jupyter directive

The following jupyter directive with cell-break option should split this text and the text that follows into different IN blocks in the notebook

This text should follow in a separate cell.

4.4 links

The following code in the .rst file

```
.. _links:

Links
-----

Links are generated as markdown references to jump between notebooks and
the sphinx link machinery is employed to track links across documents.

An external link to another `notebook (as full file) <links_target.ipynb>`_

This is a paragraph that contains `a google hyperlink`_.

.. _a google hyperlink: https://google.com.au

- An inline reference to :ref:`another document <links_target>`

Special Cases
-------------

The following link has ( and ) contained within them that doesn't render nicely in␣
↪markdown. In this case the extension will substitute ( with `%28` and ) with `%29`

Thinking back to the mathematical motivation, a `Field <https://en.wikipedia.org/wiki/
↪Field_\(mathematics\)>`_ is an `Ring` with a few additional properties
```

will look as follows in the jupyter notebook

Links

Links are generated as markdown references to jump between notebooks and the sphinx link machinery is employed to track links across documents.

An external link to another notebook (as full file)

This is a paragraph that contains a google hyperlink.

- An inline reference to another document

Special Cases

The following link has (and) contained within them that doesn't render nicely in markdown. In this case the extension will substitute (with %28 and) with %29

Thinking back to the mathematical motivation, a Field is an Ring with a few additional properties

4.5 math

The following code in the .rst file

```
Math
----

Inline maths with inline role: :math:`x^3+\frac{1+\sqrt{2}}{\pi}`

Inline maths using dollar signs (not supported yet): $x^3+\frac{1+\sqrt{2}}{\pi}$ as␣
↪the
backslashes are removed.

.. math::

x^3+\frac{1+\sqrt{2}}{\pi}

check math with some more advanced LaTeX, previously reported as an issue.

.. math::

    \mathbb P\{z = v \mid x \}
    = \begin{cases}
        f_0(v) & \mbox{if } x = x_0, \\
        f_1(v) & \mbox{if } x = x_1
    \end{cases}

and labeled test cases

.. math::
   :label: firsteq

    \mathbb P\{z = v \mid x \}
    = \begin{cases}
        f_0(v) & \mbox{if } x = x_0, \\
        f_1(v) & \mbox{if } x = x_1
    \end{cases}
```

(continues on next page)

(continued from previous page)

```
Further Inline
--------------

A continuation Ramsey planner at :math:`t \geq 1` takes
:math:`(x_{t-1}, s_{t-1}) = (x_-, s_-)` as given and before
:math:`s` is realized chooses
:math:`(n_t(s_t), x_t(s_t)) = (n(s), x(s))` for :math:`s \in  {\cal S}`

Referenced Math
---------------

Simple test case with reference in text

.. math::
   :label: test

    v = p + \beta v

this is a reference to :eq:`test` which is the above equation
```

will look as follows in the jupyter notebook

Math

Inline maths with inline role: $x^3 + \frac{1+\sqrt{2}}{\pi}$

Inline maths using dollar signs (not supported yet): $x^3 + frac1 + sqrt2pi$ as the backslashes are removed.

$$x^3 + \frac{1+\sqrt{2}}{\pi}$$

check math with some more advanced LaTeX, previously reported as an issue.

$$\mathbb{P}\{z = v \mid x\} = \begin{cases} f_0(v) & \text{if } x = x_0, \\ f_1(v) & \text{if } x = x_1 \end{cases}$$

and labeled test cases

$$\mathbb{P}\{z = v \mid x\} = \begin{cases} f_0(v) & \text{if } x = x_0, \\ f_1(v) & \text{if } x = x_1 \end{cases} \tag{1}$$

Further Inline

A continuation Ramsey planner at $t \geq 1$ takes $(x_{t-1}, s_{t-1}) = (x_-, s_-)$ as given and before s is realized chooses $(n_t(s_t), x_t(s_t)) = (n(s), x(s))$ for $s \in \mathcal{S}$

Referenced Math

Simple test case with reference in text

$$v = p + \beta v \tag{2}$$

this is a reference to (2) which is the above equation

4.6 block-quote

The following code in the .rst file

```
Quote
-----

This is some text

    This is a quote!

and this is not

Epigraph
--------

An epigraph is a special block-quote node

.. epigraph::

    "Debugging is twice as hard as writing the code in the first place.
    Therefore, if you write the code as cleverly as possible, you are, by definition,
    not smart enough to debug it."

-- Brian Kernighan

and one that is technically malformed

.. epigraph::

    "Debugging is twice as hard as writing the code in the first place.
    Therefore, if you write the code as cleverly as possible, you are, by definition,
    not smart enough to debug it." -- Brian Kernighan

with some final text
```

will look as follows in the jupyter notebook

Quote

This is some text

> This is a quote!

and this is not

Epigraph

An epigraph is a special block-quote node

> "Debugging is twice as hard as writing the code in the first place. Therefore, if you write the code as cleverly as possible, you are, by definition, not smart enough to debug it."
>
> Brian Kernighan

and one that is technically malformed

> "Debugging is twice as hard as writing the code in the first place. Therefore, if you write the code as cleverly as possible, you are, by definition, not smart enough to debug it." – Brian Kernighan

with some final text

4.7 slides

The following code in the .rst file

```
Slide option activated
----------------------

.. jupyter::
    :slide: enable

This is a collection of different types of cells where the toolbar: Slideshow has␣
↪been activated

.. jupyter::
    :cell-break:
    :slide-type: subslide

The idea is that eventually we will assign a type (*slide*, *subslide*, *skip*,␣
↪*note*) for each one. We used our **jupyter** directive  to break the markdown cell␣
↪into two different cells.


.. code:: python3

    import numpy as np

    x = np.linspace(0, 1, 5)
    y = np.sin(4 * np.pi * x) * np.exp(-5 * x)
```

(continues on next page)

(continued from previous page)

```
    print(y)

.. code:: python3

    import numpy as np

    z = np.cos(3 * np.pi * x) * np.exp(-2 * x)
    w = z*y

    print(w)

Math
++++

The previous function was

.. math:: f(x)=\sin(4\pi x)\cos(4\pi x)e^{-7x}


.. jupyter::
    :cell-break:
    :slide-type: fragment

We can also include the figures from some folder


.. figure:: _static/hood.jpg
```

will look as follows in the jupyter notebook

Slide Type Slide

Slide option activated

This is a collection of different types of cells where the toolbar: Slideshow has been activated

Slide Type Sub-Slide

The idea is that eventually we will assign a type (*slide*, *subslide*, *skip*, *note*) for each one. We used our **jupyter** directive to break the markdown cell into two different cells.

In [ ]:

Slide Type Slide

```
import numpy as np

x = np.linspace(0, 1, 5)
y = np.sin(4 * np.pi * x) * np.exp(-5 * x)

print(y)
```

In [ ]:

Slide Type Slide

```
import numpy as np

z = np.cos(3 * np.pi * x) * np.exp(-2 * x)
w = z*y

print(w)
```

Slide Type Slide

Math

The previous function was

$$f(x) = \sin(4\pi x)\cos(4\pi x)e^{-7x}$$

Slide Type Fragment

We can also include the figures from some folder

4.8 footnotes

The following code in the .rst file

```
Rubric
======

Define the government's one-period loss function [#f1]_

.. math::
    :label: target

    r(y, u)  =  y' R y  + u' Q u


History dependence has two sources: (a) the government's ability to commit [#f2]_ to␣
↪a sequence of rules at time :math:`0`


.. rubric:: Footnotes

.. [#f1] The problem assumes that there are no cross products between states and␣
↪controls in the return function.  A simple transformation  converts a problem whose␣
↪return function has cross products into an equivalent problem that has no cross␣
↪products.

.. [#f2] The government would make different choices were it to choose sequentially,␣
↪that is,  were it to select its time :math:`t` action at time :math:`t`.
```

will look as follows in the jupyter notebook

Rubric

Define the government's one-period loss function [1]

$$r(y, u) = y' R y + u' Q u \tag{1}$$

History dependence has two sources: (a) the government's ability to commit [2] to a sequence of rules at time 0

Footnotes

[1] The problem assumes that there are no cross products between states and controls in the return function. A simple transformation converts a problem whose return function has cross products into an equivalent problem that has no cross products.

[2] The government would make different choices were it to choose sequentially, that is, were it to select its time t action at time t.

4.9 solutions

The following code in the .rst file

```
Notebook without solutions
==========================
```

(continues on next page)

(continued from previous page)

```
The idea is with the use of classes, we can  decide whether to show or not the␣
↪solutions
of a particular lecture, creating two different types of jupyter notebooks. For now␣
↪it only
works with *code blocks*, you have to include **:class: solution**, and set in  the␣
↪conf.py file
*jupyter_drop_solutions=True*.


Here is a small example

Question 1
----------

Plot the area under the curve

.. math::

    f(x)=\sin(4\pi x) exp(-5x)

when :math:`x \in [0,1]`

.. code-block:: python3
    :class: solution

    import numpy as np
    import matplotlib.pyplot as plt

    x = np.linspace(0, 1, 500)
    y = np.sin(4 * np.pi * x) * np.exp(-5 * x)

    fig, ax = plt.subplots()

    ax.fill(x, y, zorder=10)
    ax.grid(True, zorder=5)
    plt.show()
```

will look as follows in the jupyter notebook

Notebook without solutions

The idea is with the use of classes, we can decide whether to show or not the solutions of a particular lecture, creating two different types of jupyter notebooks. For now it only works with *code blocks*, you have to include **:class: solution**, and set in the conf.py file *jupyter_drop_solutions=True*.

Here is a small example

Question 1

Plot the area under the curve

$$f(x) = \sin(4\pi x)exp(-5x)$$

when $x \in [0, 1]$

Todo: Currently generating the two sets of notebooks requires two separate runs of sphinx which is incovenient. It

would be better to develop a set of notebooks without solutions (as Default) and a set of notebooks with solutions in a subdir.

4.10 tables

Basic table support is provided by this extension.

Note: Complex tables are **not** currently supported. See Issue [#54](https://github.com/QuantEcon/sphinxcontrib-jupyter/issues/54)

The following code in the .rst file

```
Table
=====

These tables are from the `RST specification <http://docutils.sourceforge.net/docs/
↪ref/rst/restructuredtext.html#grid-tables>`__:

Grid Tables
-----------

A simple rst table with header

+------+------+
| C1   | C2   |
+======+======+
| a    | b    |
+------+------+
| c    | d    |
+------+------+

**Note:** Tables without a header are currently not supported as markdown does
not support tables without headers.


Simple Tables
-------------

=====  =====  =======
A      B    A and B
=====  =====  =======
False  False  False
True   False  False
False  True   False
True   True   True
=====  =====  =======

Directive Table Types
---------------------

These table types are provided by `sphinx docs <http://www.sphinx-doc.org/en/master/
↪rest.html#directives>`__
```

(continues on next page)

(continued from previous page)

```
List Table directive
~~~~~~~~~~~~~~~~~~~~

.. list-table:: Frozen Delights!
:widths: 15 10 30
:header-rows: 1

* - Treat
    - Quantity
    - Description
* - Albatross
    - 2.99
    - On a stick!
* - Crunchy Frog
    - 1.49
    - If we took the bones out, it wouldn't be crunchy, now would it?
* - Gannet Ripple
    - 1.99
    - On a stick!
```

will look as follows in the jupyter notebook

Table

These tables are from the RST specification:

Grid Tables

A simple rst table with header

C1	C2
a	b
c	d

Note: Tables without a header are currently not supported as markdown does not support tables without headers.

Simple Tables

A	B	A and B
False	False	False
True	False	False
False	True	False
True	True	True

Directive Table Types

These table types are provided by sphinx docs

List Table directive

Frozen Delights!

Treat	Quantity	Description
Albatross	2.99	On a stick!
Crunchy Frog	1.49	If we took the bones out, it wouldn't be crunchy, now would it?
Gannet Ripple	1.99	On a stick!

4.11 tests

The following code in the .rst file

```
Notebook without Tests
======================

This is an almost exact analogue to the solutions class. The idea is that we can␣
↪include test blocks using **:class: test** that we can toggle on or off with␣
↪*jupyter_drop_tests = True*. A primary use case is for regression testing for the 0.
↪6 => 1.0 port, which we will not want to show to the end user.

Here is a small example:
```

(continues on next page)

(continued from previous page)

```
Question 1
------------

.. code-block:: julia

    x = 3
    foo = n -> (x -> x + n)

.. code-block:: julia
    :class: test

    import Test
    @test x == 3
    @test foo(3) isa Function
    @test foo(3)(4) == 7
```

will look as follows in the jupyter notebook

Notebook without Tests

This is an almost exact analogue to the solutions class. The idea is that we can include test blocks using **:class: test** that we can toggle on or off with *jupyter_drop_tests* = *True*. A primary use case is for regression testing for the 0.6 => 1.0 port, which we will not want to show to the end user.

Here is a small example:

Question 1

```
x = 3
foo = n -> (x -> x + n)
```

Note: inclusion of tests in the generated notebook can be controlled in the `conf.py` file using `jupyter_drop_tests = False`. This is useful when using the `coverage` build pathway.

CHAPTER 5

Example *conf.py* file

After running a sphinx-quickstart you can add the *jupyter* options needed for your project in a similar fashion to what is shown belows.

```
# ---------------------------------------------
# sphinxcontrib-jupyter Configuration Settings
# ---------------------------------------------

# Conversion Mode Settings
# If "all", convert codes and texts into jupyter notebook
# If "code", convert code-blocks only
jupyter_conversion_mode = "all"

# Write notebook creation metadata to the top of the notebook
jupyter_write_metadata = True

# Location for _static folder
jupyter_static_file_path = ["_static"]

# Configure Jupyter Kernels
jupyter_kernels = {
    "python3": {
        "kernelspec": {
            "display_name": "Python",
            "language": "python3",
            "name": "python3"
            },
        "file_extension": ".py",
    },
}

# Configure default language for Jupyter notebooks
# Can be changed in each notebook thanks to the ..highlight:: directive
jupyter_default_lang = "python3"
```

(continues on next page)

(continued from previous page)

```
# Prepend a Welcome Message to Each Notebook
jupyter_welcome_block = "welcome.rst"

# Solutions Configuration
jupyter_drop_solutions = True

# Tests configurations
jupyter_drop_tests = True

# Add Ipython as Synonym for tests
jupyter_lang_synonyms = ["ipython"]
```

CHAPTER 6

Managing Large Projects

Large projects may require different build pathways due to the time required for execution of embedded code. This can be done by modifying the `Makefile` to accomodate multiple build pathways.

You may, for example, wish to leave `make jupyter` simply building notebooks while setting up an alternative `make` command to target a full `website` build.

In the `Makefile` you can add an alternative build target such as:

```
BUILDWEBSITE  = _build/website
```

and then you can modify options (set in the `conf.py` file) using the *-D* flag.

```
website:
    @$(SPHINXBUILD) -M jupyter "$(SOURCEDIR)" "$(BUILDWEBSITE)" $(SPHINXOPTS) $(O) -D␣
↪jupyter_make_site=1 -D jupyter_generate_html=1 -D jupyter_download_nb=1 -D jupyter_
↪execute_notebooks=1 -D jupyter_target_html=1 -D jupyter_images_markdown=0 -D␣
↪jupyter_html_template="theme/templates/lectures-nbconvert.tpl" -D jupyter_download_
↪nb_urlpath="https://lectures.quantecon.org/"
```

this will setup a new folder `_build/website` for the new build pathway to store resultant files from the options selected.

Note: this method also preserves the `sphinx` cache mechanism for each build pathway.

Warning: Issue #199 will alter this approach to include all *configuration* settings in the `conf.py` file and then the different pipelines can be switched off in the Makefile which will be less error prone.

CHAPTER 7

Credits

This project is supported by QuantEcon

Many thanks to the lead developers of this project.

- @AakashGfude
- @mmcky
- @NickSifniotis
- @myuuuuun

Contributors

- FelipeMaldonado

CHAPTER 8

Projects using Extension

1. QuantEcon Lectures

If you find this extension useful please let us know at contact@quantecon.org

CHAPTER 9

LICENSE

CHAPTER 10

Indices and tables

- genindex
- modindex
- search