
sphinx-tabs

unknown

Feb 15, 2024

CONTENTS

1	Installation	3
2	Sphinx Configuration	5
3	Basic Tabs	7
4	Nested Tabs	9
5	Group Tabs	11
6	Code Tabs	13

Create tabbed content in [Sphinx documentation](#) when building HTML.

INSTALLATION

```
pip install sphinx-tabs
```

To enable the extension in Sphinx, add the following to your `conf.py`:

```
extensions = ['sphinx_tabs.tabs']
```

If you are using [Read The Docs](#) for building your documentation, the extension must be added as a requirement. Please add *sphinx-tabs* to *requirements.txt* at the root of the project or in your docs folder.

SPHINX CONFIGURATION

If needed, there is a configuration option to allow additional builders to be considered compatible. For example, to add the *linkcheck* builder, add the following to your *conf.py*:

```
sphinx_tabs_valid_builders = ['linkcheck']
```

By default, tabs can be closed by selecting the open tab. This functionality can be disabled using the *sphinx_tabs_disable_tab_closing* configuration option:

```
sphinx_tabs_disable_tab_closing = True
```

Custom lexers that have been loaded in the sphinx *conf.py* can be used with *code-tabs*:

```
def setup(app):  
    app.add_lexer('alias', MyCustomLexer())
```

By default, the extension loads predefined CSS styles for tabs. To disable the CSS from loading, add the following to your *conf.py*:

```
sphinx_tabs_disable_css_loading = True
```


BASIC TABS

All *sphinx-tabs* use the *tabs* directive to define a tab set. Basic tabs are added using the *tab* directive, which takes the tab's label as an argument:

```
.. tabs::  
  
    .. tab:: Apples  
  
        Apples are green, or sometimes red.  
  
    .. tab:: Pears  
  
        Pears are green.  
  
    .. tab:: Oranges  
  
        Oranges are orange.
```

These will appear as:

Apples

Pears

Oranges

Apples are green, or sometimes red.

Pears are green.

Oranges are orange.

The contents of each tab can be displayed by clicking on the tab that you wish to show. Clicking on the tab that is currently open will hide the tab's content, leaving only the tab set labels visible.

Alternatively, tab sets can be focused using `Tab`. The `Left Arrow` and `Right Arrow` keys can then be used to navigate across the tab set and `Enter` can be used to select a tab.

NESTED TABS

Tabs can be nested inside one another:

```
.. tabs::  
    .. tab:: Stars  
        .. tabs::  
            .. tab:: The Sun  
                The closest star to us.  
            .. tab:: Proxima Centauri  
                The second closest star to us.  
            .. tab:: Polaris  
                The North Star.  
        .. tab:: Moons  
            .. tabs::  
                .. tab:: The Moon  
                    Orbits the Earth  
                .. tab:: Titan  
                    Orbits Jupiter
```

Nested tabs appear as:

Stars

Moons

The Sun

Proxima Centauri

Polaris

The closest star to us.

The second closest star to us.

The North Star.

The Moon

Titan

Orbits the Earth

Orbits Jupiter

GROUP TABS

When multiple tab sets contain related content, the *group-tab* directive can be used to create group tabs:

```
.. tabs::

  .. group-tab:: Linux

    Linux tab content - tab set 1

  .. group-tab:: Mac OSX

    Mac OSX tab content - tab set 1

  .. group-tab:: Windows

    Windows tab content - tab set 1

.. tabs::

  .. group-tab:: Linux

    Linux tab content - tab set 2

  .. group-tab:: Mac OSX

    Mac OSX tab content - tab set 2

  .. group-tab:: Windows

    Windows tab content - tab set 2
```

Linux

Mac OSX

Windows

Linux tab content - tab set 1

Mac OSX tab content - tab set 1

Windows tab content - tab set 1

Linux

Mac OSX

Windows

Linux tab content - tab set 2

Mac OSX tab content - tab set 2

Windows tab content - tab set 2

The tab selection in these groups is synchronised, so selecting the ‘Linux’ tab of one tab set will open the ‘Linux’ tab contents in all tab sets on the current page.

If permitted by the user’s browser, the last selected group tab will be remembered when changing page in the current session. As such, if any tabsets on the next page contain a tab with the same label it will be selected.

CODE TABS

A common use of group tabs is to show code examples in multiple programming languages. The *code-tab* directive creates a group tab and treats the tab content as a *code-block*.

The first argument to a *code-tab* is the name of the language to use for code highlighting, while the optional second argument is a custom label for the tab. By default, the tab is labelled using the lexer name. The tab label is used to group tabs, so the same custom label should be used to group related tabs.

```
.. tabs::

.. code-tab:: c

    C Main Function

.. code-tab:: c++

    C++ Main Function

.. code-tab:: py

    Python Main Function

.. code-tab:: java

    Java Main Function

.. code-tab:: julia

    Julia Main Function

.. code-tab:: fortran

    Fortran Main Function

.. code-tab:: r R

    R Main Function

.. tabs::

.. code-tab:: c
```

(continues on next page)

(continued from previous page)

```
int main(const int argc, const char **argv) {
    return 0;
}

.. code-tab:: c++

    int main(const int argc, const char **argv) {
        return 0;
    }

.. code-tab:: py

    def main():
        return

.. code-tab:: java

    class Main {
        public static void main(String[] args) {
        }
    }

.. code-tab:: julia

    function main()
    end

.. code-tab:: fortran

    PROGRAM main
    END PROGRAM main

.. code-tab:: r R

    main <- function() {
        return(0)
    }
```

C

C++

Python

Java

Julia

Fortran

R

C Main Function

C++ Main Function

```
Python Main Function
```

```
Java Main Function
```

```
Julia Main Function
```

```
Fortran Main Function
```

```
R Main Function
```

```
C
```

```
C++
```

```
Python
```

```
Java
```

```
Julia
```

```
Fortran
```

```
R
```

```
int main(const int argc, const char **argv) {
    return 0;
}
```

```
int main(const int argc, const char **argv) {
    return 0;
}
```

```
def main():
    return
```

```
class Main {
    public static void main(String[] args) {
    }
}
```

```
function main()
end
```

```
PROGRAM main
END PROGRAM main
```

```
main <- function() {
    return(0)
}
```

Code tabs support highlighting using [custom syntax highlighters](#) that have been loaded in the sphinx configuration. To use custom lexers, pass the lexers alias as the first argument of *code-tab*.

Note: Sphinx’s *include* directive does not work inside *code-tab*. Use *tab* or *group-tab* instead.
