sphinxcontrib-asyncio

Release 0.3.0

Andrew Svetlov

October 24, 2020
CONTENTS

1 Installation 3
2 Usage In Documents 5
3 Usage in `sphinx.ext.autodoc` extension 7
4 Discussion list 9
5 Authors and License 11
Index 13
Add coroutine markup support to sphinx-based docs.
1. Install from PyPI:

```
$ pip install sphinxcontrib-asyncio
```

2. Enable `sphinxcontrib-asyncio` extension in your `conf.py`:

```python
extensions = ['sphinxcontrib.asyncio']
```
Usage in Documents

Use `cofunction` instead of function:

```python
.. cofunction:: coro(a, b)

    Simple coroutine function.
```

```python
coroutine coro(a, b)

    Simple coroutine function.
```

and `comethod` instead of `method`:

```python
.. class:: A

    .. comethod:: meth(self, param)

    Coroutine method.
```

```python
class A

    coroutine meth(self, param)

    Coroutine method.
```

For more complex markup use `directive options`, e.g. `async-with` for `asynchronous context managers factories`:

```python
.. cofunction:: open_url(param)

    async-with:

    A function that returns asynchronous context manager.
```

```python
async-with open_url(param)

    A function that returns asynchronous context manager.

    That means `open_url` can be used as:

    ```python
    async with open_url(arg) as cm:
        pass
    ```
```

`async-for` may be used for `asynchronous generator markup`:

```python
.. cofunction:: iter_vals(arg)

    async-for:

    A function the returns asynchronous generator.
```
async-for iter_vals \texttt{(arg)}
A function the returns asynchronous generator.

\begin{verbatim}
async for item in iter_args(arg):
    pass
\end{verbatim}

By default async-for and async-with suppresses coroutine.

If both \texttt{await func()} and \texttt{async with func():} are allowed (\texttt{func} is both \texttt{coroutine} and \texttt{asynchronous context manager}) explicit \texttt{coroutine} flag:

\begin{verbatim}
.. cofunction:: get(url)
    :async-with:
    :coroutine:

    A function can be used in ``async with`` and ``await`` context.
\end{verbatim}

coroutine async-with get \texttt{(url)}
A function can be used in async with and await context.

\texttt{comethod} also may be used with \texttt{staticmethod} and \texttt{classmethod} optional specifiers, e.g.:

\begin{verbatim}
.. class:: A

    .. comethod:: f(cls, arg)
        :classmethod:

        This \texttt{is} \texttt{classmethod}
\end{verbatim}

class A

\begin{verbatim}
    classmethod coroutine f(cls, arg)
    This is classmethod
\end{verbatim}
sphinxcontrib-asyncio add special documenters for autodocs, which will use \textit{cofunction} and \textit{comethod} directives if the function is an \texttt{async def} or is marked with \texttt{coroutine} decorator.

For example this source:

```python
import asyncio
class MyClass:
    def my_func(self):
        """ Normal function ""

@asyncio.coroutine
def my_coro(self):
    """ This is my coroutine ""

@asyncio.coroutine
def coro(param):
    """ Module level async function ""
```

Using this simple configuration in your .rst file:

```rst
.. autocofunction:: coro
.. autoclass:: MyClass
    :members:
```

Will yield next documentation:

\textbf{coroutine} \texttt{coro} (\texttt{param})

Module level async function

\textbf{class} \texttt{MyClass}

\begin{verbatim}
    my_func()
    Normal function

    coroutine my_coro()
    This is my coroutine
\end{verbatim}

You can set directive options by adding it to \texttt{autocofunction} and \texttt{autocomethod} directives:
Module level async function

You can also force `coroutine` prefix on not-coroutine method by overriding it as `autocomethod` directive:

```bash
.. autoclass:: MyClass
    :members:
    :exclude-members: my_func

.. autocomethod:: my_func()
```

```python
class MyClass

    coroutine my_func()
        Normal function

    coroutine my_coro()
        This is my coroutine
```

```rust
coroutine async-for coro(param)
    Module level async function
```

Chapter 3. Usage in `sphinx.ext.autodoc` extension
aio-libs google group: https://groups.google.com/forum/#!forum/aio-libs
Please post your questions and ideas here.
The `sphinxcontrib-asyncio` package is written by Andrew Svetlov. It’s Apache 2 licensed and freely available. Feel free to improve this package and send a pull request to GitHub.
INDEX

B
built-in function
  open_url(), 5

F
f() (A class method), 6

M
meth() (A method), 5
my_func() (MyClass method), 8

O
open_url()
  built-in function, 5