Coroutines

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\scrapy-master\docs\topics\[scrapy-master][docs][topics]coroutines.rst, line 5)

Unknown directive type "versionadded".

```
.. versionadded:: 2.0
```

Scrapy has :ref:`partial support <coroutine-support>` for the :ref:`coroutine syntax <async>`.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\scrapy-master\docs\topics\[scrapy-master][docs][topics]coroutines.rst, line 7); *backlink*

Unknown interpreted text role "ref".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\scrapy-master\docs\topics\[scrapy-master][docs][topics]coroutines.rst, line 7); *backlink*

Unknown interpreted text role "ref".

Supported callables

The following callables may be defined as coroutines using `async def`, and hence use coroutine syntax (e.g. `await`, `async for`, `async with`):

- :class:`~scrapy.Request` callbacks.

  System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\scrapy-master\docs\topics\[scrapy-master][docs][topics]coroutines.rst, line 18); *backlink*

  Unknown interpreted text role "class".

  Note

  The callback output is not processed until the whole callback finishes.

  As a side effect, if the callback raises an exception, none of its output is processed.

  This is a known caveat of the current implementation that we aim to address in a future version of Scrapy.

- The :meth:`process_item` method of :ref:`item pipelines <topics-item-pipeline>`.

  System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\scrapy-master\docs\topics\[scrapy-master][docs][topics]coroutines.rst, line 29); *backlink*

  Unknown interpreted text role "meth".

  System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\scrapy-master\docs\topics\[scrapy-master][docs][topics]coroutines.rst, line 29); *backlink*

  Unknown interpreted text role "ref".

- The :meth:`~scrapy.downloadermiddlewares.DownloaderMiddleware.process_request`, :meth:`~scrapy.downloadermiddlewares.DownloaderMiddleware.process_response`, and :meth:`~scrapy.downloadermiddlewares.DownloaderMiddleware.process_exception` methods of :ref:`downloader middlewares <topics-downloader-middleware-custom>`.

  System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\scrapy-master\docs\topics\[scrapy-master][docs][topics]coroutines.rst, line 32); *backlink*

Unknown interpreted text role "meth".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\scrapy-master\docs\topics\[scrapy-master][docs][topics]coroutines.rst, line 32); *backlink*

Unknown interpreted text role "meth".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\scrapy-master\docs\topics\[scrapy-master][docs][topics]coroutines.rst, line 32); *backlink*

Unknown interpreted text role "meth".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\scrapy-master\docs\topics\[scrapy-master][docs][topics]coroutines.rst, line 32); *backlink*

Unknown interpreted text role "ref".

- :ref:`Signal handlers that support deferreds <signal-deferred>`.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\scrapy-master\docs\topics\[scrapy-master][docs][topics]coroutines.rst, line 40); *backlink*

Unknown interpreted text role "ref".

Usage

There are several use cases for coroutines in Scrapy. Code that would return Deferreds when written for previous Scrapy versions, such as downloader middlewares and signal handlers, can be rewritten to be shorter and cleaner:

```
from itemadapter import ItemAdapter

class DbPipeline:
    def _update_item(self, data, item):
        adapter = ItemAdapter(item)
        adapter['field'] = data
        return item

    def process_item(self, item, spider):
        adapter = ItemAdapter(item)
        dfd = db.get_some_data(adapter['id'])
        dfd.addCallback(self._update_item, item)
        return dfd
```

becomes:

```
from itemadapter import ItemAdapter

class DbPipeline:
    async def process_item(self, item, spider):
        adapter = ItemAdapter(item)
        adapter['field'] = await db.get_some_data(adapter['id'])
        return item
```

Coroutines may be used to call asynchronous code. This includes other coroutines, functions that return Deferreds and functions that return :term:`awaitable objects <awaitable>` such as :class:`~asyncio.Future`. This means you can use many useful Python libraries providing such code:

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\scrapy-master\docs\topics\[scrapy-master][docs][topics]coroutines.rst, line 73); *backlink*

Unknown interpreted text role "term".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\scrapy-master\docs\topics\[scrapy-master][docs][topics]coroutines.rst, line 73); *backlink*

Unknown interpreted text role "class".

```
class MySpiderDeferred(Spider):
    # ...
    async def parse(self, response):
        additional_response = await treq.get('https://additional.url')
        additional_data = await treq.content(additional_response)
        # ... use response and additional_data to yield items and requests

class MySpiderAsyncio(Spider):
    # ...
    async def parse(self, response):
        async with aiohttp.ClientSession() as session:
            async with session.get('https://additional.url') as additional_response:
                additional_data = await additional_response.text()
        # ... use response and additional_data to yield items and requests
```

Note

Many libraries that use coroutines, such as aio-libs, require the :mod:`asyncio` loop and to use them you need to :doc:`enable asyncio support in Scrapy<asyncio>`.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\scrapy-master\docs\topics\[scrapy-master][docs][topics]coroutines.rst, line 93); *backlink*

Unknown interpreted text role "mod".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\scrapy-master\docs\topics\[scrapy-master][docs][topics]coroutines.rst, line 93); *backlink*

Unknown interpreted text role "doc".

Note

If you want to `await` on Deferreds while using the asyncio reactor, you need to :ref:`wrap them<asyncio-await-dfd>`.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\scrapy-master\docs\topics\[scrapy-master][docs][topics]coroutines.rst, line 97); *backlink*

Unknown interpreted text role "ref".

Common use cases for asynchronous code include:

- requesting data from websites, databases and other services (in callbacks, pipelines and middlewares);
- storing data in databases (in pipelines and middlewares);
- delaying the spider initialization until some external event (in the :signal:`spider_opened` handler);

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\scrapy-master\docs\topics\[scrapy-master][docs][topics]coroutines.rst, line 105); *backlink*

Unknown interpreted text role "signal".

- calling asynchronous Scrapy methods like `ExecutionEngine.download` (see :ref:`the screenshot pipeline example<ScreenshotPipeline>`).

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\scrapy-master\docs\topics\[scrapy-master][docs][topics]coroutines.rst, line 107); *backlink*

Unknown interpreted text role "ref".