Debugging memory leaks

In Scrapy, objects such as requests, responses and items have a finite lifetime: they are created, used for a while, and finally destroyed.

From all those objects, the Request is probably the one with the longest lifetime, as it stays waiting in the Scheduler queue until it's time to process it. For more info see :ref:`topics-architecture`.

> **System Message: ERROR/3** (`D:\onboarding-resources\sample-onboarding-resources\scrapy-master\docs\topics\(scrapy-master)(docs)(topics)leaks.rst`, line 10); *backlink*
>
> Unknown interpreted text role "ref".

As these Scrapy objects have a (rather long) lifetime, there is always the risk of accumulating them in memory without releasing them properly and thus causing what is known as a "memory leak".

To help debugging memory leaks, Scrapy provides a built-in mechanism for tracking objects references called :ref:`trackref <topics-leaks-trackrefs>`, and you can also use a third-party library called :ref:`muppy <topics-leaks-muppy>` for more advanced memory debugging (see below for more info). Both mechanisms must be used from the :ref:`Telnet Console <topics-telnetconsole>`.

> **System Message: ERROR/3** (`D:\onboarding-resources\sample-onboarding-resources\scrapy-master\docs\topics\(scrapy-master)(docs)(topics)leaks.rst`, line 18); *backlink*
>
> Unknown interpreted text role "ref".

> **System Message: ERROR/3** (`D:\onboarding-resources\sample-onboarding-resources\scrapy-master\docs\topics\(scrapy-master)(docs)(topics)leaks.rst`, line 18); *backlink*
>
> Unknown interpreted text role "ref".

> **System Message: ERROR/3** (`D:\onboarding-resources\sample-onboarding-resources\scrapy-master\docs\topics\(scrapy-master)(docs)(topics)leaks.rst`, line 18); *backlink*
>
> Unknown interpreted text role "ref".

Common causes of memory leaks

It happens quite often (sometimes by accident, sometimes on purpose) that the Scrapy developer passes objects referenced in Requests (for example, using the :attr:`~scrapy.Request.cb_kwargs` or :attr:`~scrapy.Request.meta` attributes or the request callback function) and that effectively bounds the lifetime of those referenced objects to the lifetime of the Request. This is, by far, the most common cause of memory leaks in Scrapy projects, and a quite difficult one to debug for newcomers.

> **System Message: ERROR/3** (`D:\onboarding-resources\sample-onboarding-resources\scrapy-master\docs\topics\(scrapy-master)(docs)(topics)leaks.rst`, line 28); *backlink*
>
> Unknown interpreted text role "attr".

> **System Message: ERROR/3** (`D:\onboarding-resources\sample-onboarding-resources\scrapy-master\docs\topics\(scrapy-master)(docs)(topics)leaks.rst`, line 28); *backlink*
>
> Unknown interpreted text role "attr".

In big projects, the spiders are typically written by different people and some of those spiders could be "leaking" and thus affecting the rest of the other (well-written) spiders when they get to run concurrently, which, in turn, affects the whole crawling process.

The leak could also come from a custom middleware, pipeline or extension that you have written, if you are not releasing the (previously allocated) resources properly. For example, allocating resources on :signal:`spider_opened` but not releasing them on :signal:`spider_closed` may cause problems if you're running :ref:`multiple spiders per process <run-multiple-spiders>`.

> **System Message: ERROR/3** (`D:\onboarding-resources\sample-onboarding-resources\scrapy-master\docs\topics\(scrapy-master)(docs)(topics)leaks.rst`, line 41); *backlink*
>
> Unknown interpreted text role "signal".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\scrapy-master\docs\topics\(scrapy-master)(docs)(topics)leaks.rst, line 41); *backlink*

Unknown interpreted text role "signal".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\scrapy-master\docs\topics\(scrapy-master)(docs)(topics)leaks.rst, line 41); *backlink*

Unknown interpreted text role "ref".

Too Many Requests?

By default Scrapy keeps the request queue in memory; it includes :class:`~scrapy.Request` objects and all objects referenced in Request attributes (e.g. in :attr:`~scrapy.Request.cb_kwargs` and :attr:`~scrapy.Request.meta`). While not necessarily a leak, this can take a lot of memory. Enabling :ref:`persistent job queue <topics-jobs>` could help keeping memory usage in control.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\scrapy-master\docs\topics\(scrapy-master)(docs)(topics)leaks.rst, line 50); *backlink*

Unknown interpreted text role "class".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\scrapy-master\docs\topics\(scrapy-master)(docs)(topics)leaks.rst, line 50); *backlink*

Unknown interpreted text role "attr".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\scrapy-master\docs\topics\(scrapy-master)(docs)(topics)leaks.rst, line 50); *backlink*

Unknown interpreted text role "attr".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\scrapy-master\docs\topics\(scrapy-master)(docs)(topics)leaks.rst, line 50); *backlink*

Unknown interpreted text role "ref".

Debugging memory leaks with `trackref`

:mod:`trackref` is a module provided by Scrapy to debug the most common cases of memory leaks. It basically tracks the references to all live Request, Response, Item, Spider and Selector objects.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\scrapy-master\docs\topics\(scrapy-master)(docs)(topics)leaks.rst, line 63); *backlink*

Unknown interpreted text role "mod".

You can enter the telnet console and inspect how many objects (of the classes mentioned above) are currently alive using the `prefs()` function which is an alias to the :func:`~scrapy.utils.trackref.print_live_refs` function:

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\scrapy-master\docs\topics\(scrapy-master)(docs)(topics)leaks.rst, line 67); *backlink*

Unknown interpreted text role "func".

```
telnet localhost 6023

>>> prefs()
Live References

ExampleSpider                       1   oldest: 15s ago
HtmlResponse                       10   oldest: 1s ago
Selector                            2   oldest: 0s ago
FormRequest                       878   oldest: 7s ago
```

As you can see, that report also shows the "age" of the oldest object in each class. If you're running multiple spiders per process chances are you can figure out which spider is leaking by looking at the oldest request or response. You can get the oldest object of each class using the :func:`~scrapy.utils.trackref.get_oldest` function (from the telnet console).

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\scrapy-master\docs\topics\(scrapy-master)(docs)(topics)leaks.rst, line 81); *backlink*

Unknown interpreted text role "func".

Which objects are tracked?

The objects tracked by `trackrefs` are all from these classes (and all its subclasses):

- :class:`scrapy.Request`

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

  Unknown interpreted text role "class".

- :class:`scrapy.http.Response`

  System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\scrapy-master\docs\topics\(scrapy-master)(docs)(topics)leaks.rst, line 94); *backlink*

  Unknown interpreted text role "class".

- :class:`scrapy.Item`

  System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\scrapy-master\docs\topics\(scrapy-master)(docs)(topics)leaks.rst, line 95); *backlink*

  Unknown interpreted text role "class".

- :class:`scrapy.Selector`

  System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\scrapy-master\docs\topics\(scrapy-master)(docs)(topics)leaks.rst, line 96); *backlink*

  Unknown interpreted text role "class".

- :class:`scrapy.Spider`

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

  Unknown interpreted text role "class".

A real example

Let's see a concrete example of a hypothetical case of memory leaks. Suppose we have some spider with a line similar to this one:

```
return Request(f"http://www.somenastyspider.com/product.php?pid={product_id}",
               callback=self.parse, cb_kwargs={'referer': response})
```

That line is passing a response reference inside a request which effectively ties the response lifetime to the requests' one, and that would definitely cause memory leaks.

Let's see how we can discover the cause (without knowing it a priori, of course) by using the `trackref` tool.

After the crawler is running for a few minutes and we notice its memory usage has grown a lot, we can enter its telnet console and check the live references:

```
>>> prefs()
Live References

SomenastySpider                     1   oldest: 15s ago
HtmlResponse                     3890   oldest: 265s ago
Selector                            2   oldest: 0s ago
```

```
Request                         3878   oldest: 250s ago
```

The fact that there are so many live responses (and that they're so old) is definitely suspicious, as responses should have a relatively short lifetime compared to Requests. The number of responses is similar to the number of requests, so it looks like they are tied in a some way. We can now go and check the code of the spider to discover the nasty line that is generating the leaks (passing response references inside requests).

Sometimes extra information about live objects can be helpful. Let's check the oldest response:

```
>>> from scrapy.utils.trackref import get_oldest
>>> r = get_oldest('HtmlResponse')
>>> r.url
'http://www.somenastyspider.com/product.php?pid=123'
```

If you want to iterate over all objects, instead of getting the oldest one, you can use the :func:`scrapy.utils.trackref.iter_all` function:

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\scrapy-master\docs\topics\(scrapy-master)(docs)(topics)leaks.rst, line 142); *backlink*

Unknown interpreted text role "func".

```
>>> from scrapy.utils.trackref import iter_all
>>> [r.url for r in iter_all('HtmlResponse')]
['http://www.somenastyspider.com/product.php?pid=123',
 'http://www.somenastyspider.com/product.php?pid=584',
...]
```

Too many spiders?

If your project has too many spiders executed in parallel, the output of :func:`prefs()` can be difficult to read. For this reason, that function has a `ignore` argument which can be used to ignore a particular class (and all its subclases). For example, this won't show any live references to spiders:

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\scrapy-master\docs\topics\(scrapy-master)(docs)(topics)leaks.rst, line 154); *backlink*

Unknown interpreted text role "func".

```
>>> from scrapy.spiders import Spider
>>> prefs(ignore=Spider)
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\scrapy-master\docs\topics\(scrapy-master)(docs)(topics)leaks.rst, line 163)

Unknown directive type "module".

```
.. module:: scrapy.utils.trackref
   :synopsis: Track references of live objects
```

scrapy.utils.trackref module

Here are the functions available in the :mod:`~scrapy.utils.trackref` module.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\scrapy-master\docs\topics\(scrapy-master)(docs)(topics)leaks.rst, line 169); *backlink*

Unknown interpreted text role "mod".

Inherit from this class if you want to track live instances with the `trackref` module.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\scrapy-master\docs\topics\(scrapy-master)(docs)(topics)leaks.rst, line 176)

Unknown directive type "function".

```
.. function:: print_live_refs(class_name, ignore=NoneType)

    Print a report of live references, grouped by class name.

    :param ignore: if given, all objects from the specified class (or tuple of
        classes) will be ignored.
    :type ignore: type or tuple
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\scrapy-master\docs\topics\(scrapy-master)(docs)(topics)leaks.rst, line 184)

Unknown directive type "function".

```
.. function:: get_oldest(class_name)

    Return the oldest object alive with the given class name, or ``None`` if
    none is found. Use :func:`print_live_refs` first to get a list of all
    tracked live objects per class name.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\scrapy-master\docs\topics\(scrapy-master)(docs)(topics)leaks.rst, line 190)

Unknown directive type "function".

```
.. function:: iter_all(class_name)

    Return an iterator over all objects alive with the given class name, or
    ``None`` if none is found. Use :func:`print_live_refs` first to get a list
    of all tracked live objects per class name.
```

Debugging memory leaks with muppy

`trackref` provides a very convenient mechanism for tracking down memory leaks, but it only keeps track of the objects that are more likely to cause memory leaks. However, there are other cases where the memory leaks could come from other (more or less obscure) objects. If this is your case, and you can't find your leaks using `trackref`, you still have another resource: the muppy library.

You can use muppy from Pympler.

If you use `pip`, you can install muppy with the following command:

```
pip install Pympler
```

Here's an example to view all Python objects available in the heap using muppy:

```
>>> from pympler import muppy
>>> all_objects = muppy.get_objects()
>>> len(all_objects)
28667
>>> from pympler import summary
>>> suml = summary.summarize(all_objects)
>>> summary.print_(suml)
                                 types |   # objects |   total size
====================================== | =========== | ============
                         <class 'str |        9822 |      1.10 MB
                        <class 'dict |        1658 |    856.62 KB
                        <class 'type |         436 |    443.60 KB
                        <class 'code |        2974 |    419.56 KB
          <class '_io.BufferedWriter |           2 |    256.34 KB
                         <class 'set |         420 |    159.88 KB
          <class '_io.BufferedReader |           1 |    128.17 KB
          <class 'wrapper_descriptor |        1130 |     88.28 KB
                      <class 'tuple |        1304 |     86.57 KB
                     <class 'weakref |        1013 |     79.14 KB
  <class 'builtin_function_or_method |         958 |     67.36 KB
           <class 'method_descriptor |         865 |     60.82 KB
                 <class 'abc.ABCMeta |          62 |     59.96 KB
                        <class 'list |         446 |     58.52 KB
                         <class 'int |        1425 |     43.20 KB
```

For more info about muppy, refer to the muppy documentation.

Leaks without leaks

Sometimes, you may notice that the memory usage of your Scrapy process will only increase, but never decrease. Unfortunately, this could happen even though neither Scrapy nor your project are leaking memory. This is due to a (not so well) known problem of Python, which may not return released memory to the operating system in some cases. For more information on this issue see:

- Python Memory Management
- Python Memory Management Part 2

- Python Memory Management Part 3

The improvements proposed by Evan Jones, which are detailed in this paper, got merged in Python 2.5, but this only reduces the problem, it doesn't fix it completely. To quote the paper:

> *Unfortunately, this patch can only free an arena if there are no more objects allocated in it anymore. This means that fragmentation is a large issue. An application could have many megabytes of free memory, scattered throughout all the arenas, but it will be unable to free any of it. This is a problem experienced by all memory allocators. The only way to solve it is to move to a compacting garbage collector, which is able to move objects in memory. This would require significant changes to the Python interpreter.*

To keep memory consumption reasonable you can split the job into several smaller jobs or enable :ref:`persistent job queue <topics-jobs>` and stop/start spider from time to time.

> **System Message: ERROR/3** (`D:\onboarding-resources\sample-onboarding-resources\scrapy-master\docs\topics\(scrapy-master)(docs)(topics)leaks.rst`, **line 277**); ***backlink***
>
> Unknown interpreted text role "ref".