SEP	13
Title	Middlewares Refactoring
Author	Pablo Hoffman
Created	2009-11-14
Status	Document in progress (being written)

SEP-013 - Middlewares refactoring

This SEP proposes a refactoring of Scrapy middlewares to remove some inconsistencies and limitations.

Current flaws and inconsistencies

Even though the core works pretty well, it has some subtle inconsistencies that don't manifest in the common uses, but arise (and are quite annoying) when you try to fully exploit all Scrapy features. The currently identified flaws and inconsistencies are:

1. Request errback may not get called in all cases (more details needed on when this happens)
2. Spider middleware has a `process_spider_exception` method which catches exceptions coming out of the spider, but it doesn't have an analogous for catching exceptions coming into the spider (for example, from other downloader middlewares). This complicates supporting middlewares that extend other middlewares.
3. Downloader middleware has a `process_exception` method which catches exceptions coming out of the downloader, but it doesn't have an analogous for catching exceptions coming into the downloader (for example, from other downloader middlewares). This complicates supporting middlewares that extend other middlewares.
4. Scheduler middleware has a `enqueue_request` method but doesn't have a `enqueue_request_exception` nor `dequeue_request` nor `dequeue_request_exception` methods.

These flaws will be corrected by the changes proposed in this SEP.

Overview of changes proposed

Most of the inconsistencies come from the fact that middlewares don't follow the typical [https://twistedmatrix.com/projects/core/documentation/howto/defer.html deferred] callback/errback chaining logic. Twisted logic is fine and quite intuitive, and also fits middlewares very well. Due to some bad design choices the integration between middleware calls and deferred is far from optional. So the changes to middlewares involve mainly building deferred chains with the middleware methods and adding the missing method to each callback/errback chain. The proposed API for each middleware is described below.

See - a diagram draft for the process architecture.

Global changes to all middlewares

To be discussed:

1. should we support returning deferreds (i.e. `maybeDeferred`) in middleware methods?
2. should we pass Twisted Failures instead of exceptions to error methods?

Spider middleware changes

Current API

- `process_spider_input(response, spider)`
- `process_spider_output(response, result, spider)`
- `process_spider_exception(response, exception, spider=spider)`

Changes proposed

1. rename method: `process_spider_exception` to `process_spider_output_exception`
2. add method" `process_spider_input_exception`

New API

- `SpiderInput` deferred
  - `process_spider_input(response, spider)`
  - `process_spider_input_exception(response, exception, spider=spider)`
- `SpiderOutput` deferred
  - `process_spider_output(response, result, spider)`
  - `process_spider_output_exception(response, exception, spider=spider)`

Downloader middleware changes

Current API

- `process_request(request, spider)`
- `process_response(request, response, spider)`
- `process_exception(request, exception, spider)`

Changes proposed

1. rename method: `process_exception` to `process_response_exception`
2. add method: `process_request_exception`

New API

- `ProcessRequest` deferred - `process_request(request, spider)` - `process_request_exception(request, exception, response)`
- `ProcessResponse` deferred - `process_response(request, spider, response)` - `process_response_exception(request, exception, response)`

Scheduler middleware changes

Current API

- `enqueue_request(spider, request)` - '''TBD:''' what does it mean to return a Response object here? (the current implementation allows it)
- `open_spider(spider)`
- `close_spider(spider)`

Changes proposed

1. exchange order of method arguments '''(spider, request)''' to '''(request, spider)''' for consistency with the other middlewares
2. add methods: `dequeue_request`, `enqueue_request_exception`, `dequeue_request_exception`
3. remove methods: `open_spider`, `close_spider`. They should be replaced by using the `spider_opened`, `spider_closed` signals, but they weren't before because of a chicken-egg problem when open spiders (because of scheduler auto-open feature).

- '''TBD:''' how to get rid of chicken-egg problem, perhaps refactoring scheduler auto-open?

New API

- `EnqueueRequest` deferred
    - `enqueue_request(request, spider)`
        - Can return:
            - return Request: which is passed to next mw component
            - raise `IgnoreRequest`
            - raise any other exception (errback chain called)
    - `enqueue_request_exception(request, exception, spider)`
    - Output and errors:
        - The Request that gets returned by last enqueue_request() is the one that gets scheduled
        - If no request is returned but a Failure, the Request errback is called with that failure
            - '''TBD''': do we want to call request errback if it fails scheduling?0
- `DequeueRequest` deferred
    - `dequeue_request(request, spider)`
    - `dequeue_request_exception(exception, spider)`

Open issues (to resolve)

1. how to avoid massive `IgnoreRequest` exceptions from propagating which slows down the crawler
2. if requests change, how do we keep reference to the original one? do we need to? - opt 1: don't allow changing the original Request object - discarded - opt 2: keep reference to the original request (how it's done now) - opt 3: split SpiderRequest from DownloaderRequest
    - opt 5: keep reference only to original deferred and forget about the original request
3. scheduler auto-open chicken-egg problem
    - opt 1: drop auto-open y forbid opening spiders if concurrent is full. use SpiderScheduler instead. why is scheduler auto-open really needed?
4. call `Request.errback` if both schmw and dlmw fail? - opt 1: ignore and just propagate the error as-is - opt 2: call another method? like Request.schmw_errback / dlmw_errback? - opt 3: use an exception wrapper? SchedulerError()

DownloaderError()?