Scrapy

Settings

The Scrapy settings allows you to customize the behaviour of all Scrapy components, including the core, extensions, pipelines and spiders themselves.

The infrastructure of the settings provides a global namespace of key-value mappings that the code can use to pull configuration values from. The settings can be populated through different mechanisms, which are described below.

The settings are also the mechanism for selecting the currently active Scrapy project (in case you have many).

For a list of available built-in settings see: Built-in settings reference.

Designating the settings

When you use Scrapy, you have to tell it which settings you're using. You can do this by using an environment variable, `SCRAPY_SETTINGS_MODULE`.

The value of `SCRAPY_SETTINGS_MODULE` should be in Python path syntax, e.g. `myproject.settings`. Note that the settings module should be on the Python import search path.

Populating the settings

Settings can be populated using different mechanisms, each of which having a different precedence. Here is the list of them in decreasing order of precedence:

1. Command line options (most precedence)
2. Settings per-spider
3. Project settings module

4. Default settings per-command
5. Default global settings (less precedence)

The population of these settings sources is taken care of internally, but a manual handling is possible using API calls. See the Settings API topic for reference.

These mechanisms are described in more detail below.

1. Command line options

Arguments provided by the command line are the ones that take most precedence, overriding any other options. You can explicitly override one (or more) settings using the `-s` (or `--set`) command line option.

Example:

```
scrapy crawl myspider -s LOG_FILE=scrapy.log
```

2. Settings per-spider

Spiders (See the Spiders chapter for reference) can define their own settings that will take precedence and override the project ones. They can do so by setting their `custom_settings` attribute:

```
class MySpider(scrapy.Spider):
    name = 'myspider'

    custom_settings = {
        'SOME_SETTING': 'some value',
    }
```

3. Project settings module

The project settings module is the standard configuration file for your Scrapy project, it's where most of your custom settings will be populated. For a standard Scrapy project, this means you'll be adding or changing the settings in the `settings.py` file created for your project.

4. Default settings per-command

Each Scrapy tool command can have its own default settings, which override the global default settings. Those custom command settings are specified in the `default_settings` attribute of the command class.

5. Default global settings

The global defaults are located in the `scrapy.settings.default_settings` module and documented in the

Built-in settings reference section.

How to access settings

In a spider, the settings are available through `self.settings`:

```
class MySpider(scrapy.Spider):
    name = 'myspider'
    start_urls = ['http://example.com']

    def parse(self, response):
        print("Existing settings: %s" % self.settings.attributes.keys())
```

Note

The `settings` attribute is set in the base Spider class after the spider is initialized. If you want to use the settings before the initialization (e.g., in your spider's `__init__()` method), you'll need to override the `from_crawler()` method.

Settings can be accessed through the `scrapy.crawler.Crawler.settings` attribute of the Crawler that is passed to `from_crawler` method in extensions, middlewares and item pipelines:

```
class MyExtension(object):
    def __init__(self, log_is_enabled=False):
        if log_is_enabled:
            print("log is enabled!")

    @classmethod
    def from_crawler(cls, crawler):
        settings = crawler.settings
        return cls(settings.getbool('LOG_ENABLED'))
```

The settings object can be used like a dict (e.g., `settings['LOG_ENABLED']`), but it's usually preferred to extract the setting in the format you need it to avoid type errors, using one of the methods provided by the `Settings` API.

Rationale for setting names

Setting names are usually prefixed with the component that they configure. For example, proper setting names for a fictional robots.txt extension would be `ROBOTSTXT_ENABLED`, `ROBOTSTXT_OBEY`, `ROBOTSTXT_CACHEDIR`, etc.

Built-in settings reference

Here's a list of all available Scrapy settings, in alphabetical order, along with their default values and the scope where they apply.

The scope, where available, shows where the setting is being used, if it's tied to any particular component. In that case the module of that component will be shown, typically an extension, middleware or pipeline. It also means that the component must be enabled in order for the setting to have any effect.

AWS_ACCESS_KEY_ID

Default: `None`

The AWS access key used by code that requires access to Amazon Web services, such as the S3 feed storage backend.

AWS_SECRET_ACCESS_KEY

Default: `None`

The AWS secret key used by code that requires access to Amazon Web services, such as the S3 feed storage backend.

BOT_NAME

Default: `'scrapybot'`

The name of the bot implemented by this Scrapy project (also known as the project name). This will be used to construct the User-Agent by default, and also for logging.

It's automatically populated with your project name when you create your project with the `startproject` command.

CONCURRENT_ITEMS

Default: `100`

Maximum number of concurrent items (per response) to process in parallel in the Item Processor (also known as the Item Pipeline).

CONCURRENT_REQUESTS

Default: `16`

The maximum number of concurrent (ie. simultaneous) requests that will be performed by the Scrapy downloader.

CONCURRENT_REQUESTS_PER_DOMAIN

Default: `8`

The maximum number of concurrent (ie. simultaneous) requests that will be performed to any single domain.

See also: AutoThrottle extension and its `AUTOTHROTTLE_TARGET_CONCURRENCY` option.

CONCURRENT_REQUESTS_PER_IP

Default: `0`

The maximum number of concurrent (ie. simultaneous) requests that will be performed to any single IP. If non-zero, the `CONCURRENT_REQUESTS_PER_DOMAIN` setting is ignored, and this one is used instead. In other words, concurrency limits will be applied per IP, not per domain.

This setting also affects `DOWNLOAD_DELAY` and AutoThrottle extension: if `CONCURRENT_REQUESTS_PER_IP` is non-zero, download delay is enforced per IP, not per domain.

DEFAULT_ITEM_CLASS

Default: `'scrapy.item.Item'`

The default class that will be used for instantiating items in the the Scrapy shell.

DEFAULT_REQUEST_HEADERS

Default:

```
{
    'Accept': 'text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8',
    'Accept-Language': 'en',
```

```
}
```

The default headers used for Scrapy HTTP Requests. They're populated in the `DefaultHeadersMiddleware`.

DEPTH_LIMIT

Default: `0`

Scope: `scrapy.spidermiddlewares.depth.DepthMiddleware`

The maximum depth that will be allowed to crawl for any site. If zero, no limit will be imposed.

DEPTH_PRIORITY

Default: `0`

Scope: `scrapy.spidermiddlewares.depth.DepthMiddleware`

An integer that is used to adjust the request priority based on its depth:

- if zero (default), no priority adjustment is made from depth
- **a positive value will decrease the priority, i.e. higher depth requests will be processed later** ; this is commonly used when doing breadth-first crawls (BFO)
- a negative value will increase priority, i.e., higher depth requests will be processed sooner (DFO)

See also: Does Scrapy crawl in breadth-first or depth-first order? about tuning Scrapy for BFO or DFO.

Note

This setting adjusts priority **in the opposite way** compared to other priority settings `REDIRECT_PRIORITY_ADJUST` and `RETRY_PRIORITY_ADJUST`.

DEPTH_STATS

Default: `True`

Scope: `scrapy.spidermiddlewares.depth.DepthMiddleware`

Whether to collect maximum depth stats.

DEPTH_STATS_VERBOSE

Default: `False`

Scope: `scrapy.spidermiddlewares.depth.DepthMiddleware`

Whether to collect verbose depth stats. If this is enabled, the number of requests for each depth is collected in the stats.

DNSCACHE_ENABLED

Default: `True`

Whether to enable DNS in-memory cache.

DNSCACHE_SIZE

Default: `10000`

DNS in-memory cache size.

DNS_TIMEOUT

Default: `60`

Timeout for processing of DNS queries in seconds. Float is supported.

DOWNLOADER

Default: `'scrapy.core.downloader.Downloader'`

The downloader to use for crawling.

DOWNLOADER_HTTPCLIENTFACTORY

Default: `'scrapy.core.downloader.webclient.ScrapyHTTPClientFactory'`

Defines a Twisted `protocol.ClientFactory` class to use for HTTP/1.0 connections (for

`HTTP10DownloadHandler`).

> **Note**
>
> HTTP/1.0 is rarely used nowadays so you can safely ignore this setting, unless you use Twisted<11.1, or if you really want to use HTTP/1.0 and override `DOWNLOAD_HANDLERS_BASE` for `http(s)` scheme accordingly, i.e. to `'scrapy.core.downloader.handlers.http.HTTP10DownloadHandler'`.

DOWNLOADER_CLIENTCONTEXTFACTORY

Default: `'scrapy.core.downloader.contextfactory.ScrapyClientContextFactory'`

Represents the classpath to the ContextFactory to use.

Here, "ContextFactory" is a Twisted term for SSL/TLS contexts, defining the TLS/SSL protocol version to use, whether to do certificate verification, or even enable client-side authentication (and various other things).

> **Note**
>
> Scrapy default context factory **does NOT perform remote server certificate verification**. This is usually fine for web scraping.
>
> If you do need remote server certificate verification enabled, Scrapy also has another context factory class that you can set, `'scrapy.core.downloader.contextfactory.BrowserLikeContextFactory'`, which uses the platform's certificates to validate remote endpoints. **This is only available if you use Twisted>=14.0.**

If you do use a custom ContextFactory, make sure it accepts a `method` parameter at init (this is the `OpenSSL.SSL` method mapping `DOWNLOADER_CLIENT_TLS_METHOD`).

DOWNLOADER_CLIENT_TLS_METHOD

Default: `'TLS'`

Use this setting to customize the TLS/SSL method used by the default HTTP/1.1 downloader.

This setting must be one of these string values:

- `'TLS'`: maps to OpenSSL's `TLS_method()` (a.k.a `SSLv23_method()`), which allows protocol negotiation,

starting from the highest supported by the platform; **default, recommended**

- `'TLSv1.0'`: this value forces HTTPS connections to use TLS version 1.0 ; set this if you want the behavior of Scrapy<1.1
- `'TLSv1.1'`: forces TLS version 1.1
- `'TLSv1.2'`: forces TLS version 1.2
- `'SSLv3'`: forces SSL version 3 (**not recommended**)

Note

We recommend that you use PyOpenSSL>=0.13 and Twisted>=0.13 or above (Twisted>=14.0 if you can).

DOWNLOADER_MIDDLEWARES

Default:: `{}`

A dict containing the downloader middlewares enabled in your project, and their orders. For more info see Activating a downloader middleware.

DOWNLOADER_MIDDLEWARES_BASE

Default:

```
{
    'scrapy.downloadermiddlewares.robotstxt.RobotsTxtMiddleware': 100,
    'scrapy.downloadermiddlewares.httpauth.HttpAuthMiddleware': 300,
    'scrapy.downloadermiddlewares.downloadtimeout.DownloadTimeoutMiddleware': 350,
    'scrapy.downloadermiddlewares.defaultheaders.DefaultHeadersMiddleware': 400,
    'scrapy.downloadermiddlewares.useragent.UserAgentMiddleware': 500,
    'scrapy.downloadermiddlewares.retry.RetryMiddleware': 550,
    'scrapy.downloadermiddlewares.ajaxcrawl.AjaxCrawlMiddleware': 560,
    'scrapy.downloadermiddlewares.redirect.MetaRefreshMiddleware': 580,
    'scrapy.downloadermiddlewares.httpcompression.HttpCompressionMiddleware': 590,
    'scrapy.downloadermiddlewares.redirect.RedirectMiddleware': 600,
    'scrapy.downloadermiddlewares.cookies.CookiesMiddleware': 700,
    'scrapy.downloadermiddlewares.httpproxy.HttpProxyMiddleware': 750,
    'scrapy.downloadermiddlewares.stats.DownloaderStats': 850,
    'scrapy.downloadermiddlewares.httpcache.HttpCacheMiddleware': 900,
}
```

A dict containing the downloader middlewares enabled by default in Scrapy. Low orders are closer to the engine, high orders are closer to the downloader. You should never modify this setting in your project, modify `DOWNLOADER_MIDDLEWARES` instead. For more info see Activating a downloader middleware.

DOWNLOADER_STATS

Default: `True`

Whether to enable downloader stats collection.

DOWNLOAD_DELAY

Default: `0`

The amount of time (in secs) that the downloader should wait before downloading consecutive pages from the same website. This can be used to throttle the crawling speed to avoid hitting servers too hard. Decimal numbers are supported. Example:

```
DOWNLOAD_DELAY = 0.25    # 250 ms of delay
```

This setting is also affected by the `RANDOMIZE_DOWNLOAD_DELAY` setting (which is enabled by default). By default, Scrapy doesn't wait a fixed amount of time between requests, but uses a random interval between 0.5 * `DOWNLOAD_DELAY` and 1.5 * `DOWNLOAD_DELAY`.

When `CONCURRENT_REQUESTS_PER_IP` is non-zero, delays are enforced per ip address instead of per domain.

You can also change this setting per spider by setting `download_delay` spider attribute.

DOWNLOAD_HANDLERS

Default: `{}`

A dict containing the request downloader handlers enabled in your project. See `DOWNLOAD_HANDLERS_BASE` for example format.

DOWNLOAD_HANDLERS_BASE

Default:

```
{
    'file': 'scrapy.core.downloader.handlers.file.FileDownloadHandler',
    'http': 'scrapy.core.downloader.handlers.http.HTTPDownloadHandler',
    'https': 'scrapy.core.downloader.handlers.http.HTTPDownloadHandler',
    's3': 'scrapy.core.downloader.handlers.s3.S3DownloadHandler',
    'ftp': 'scrapy.core.downloader.handlers.ftp.FTPDownloadHandler',
```

```
}
```

A dict containing the request download handlers enabled by default in Scrapy. You should never modify this setting in your project, modify `DOWNLOAD_HANDLERS` instead.

You can disable any of these download handlers by assigning `None` to their URI scheme in `DOWNLOAD_HANDLERS`. E.g., to disable the built-in FTP handler (without replacement), place this in your `settings.py`:

```
DOWNLOAD_HANDLERS = {
    'ftp': None,
}
```

DOWNLOAD_TIMEOUT

Default: `180`

The amount of time (in secs) that the downloader will wait before timing out.

Note

This timeout can be set per spider using `download_timeout` spider attribute and per-request using `download_timeout` Request.meta key.

DOWNLOAD_MAXSIZE

Default: 1073741824 (1024MB)

The maximum response size (in bytes) that downloader will download.

If you want to disable it set to 0.

Note

This size can be set per spider using `download_maxsize` spider attribute and per-request using `download_maxsize` Request.meta key.

This feature needs Twisted >= 11.1.

DOWNLOAD_WARNSIZE

Default: 33554432 (32MB)

The response size (in bytes) that downloader will start to warn.

If you want to disable it set to 0.

Note

This size can be set per spider using `download_warnsize` spider attribute and per-request using `download_warnsize` Request.meta key.

This feature needs Twisted >= 11.1.

DOWNLOAD_FAIL_ON_DATALOSS

Default: `True`

Whether or not to fail on broken responses, that is, declared `Content-Length` does not match content sent by the server or chunked response was not properly finish. If `True`, these responses raise a `ResponseFailed([_DataLoss])` error. If `False`, these responses are passed through and the flag `dataloss` is added to the response, i.e.: `'dataloss' in response.flags` is `True`.

Optionally, this can be set per-request basis by using the `download_fail_on_dataloss` Request.meta key to `False`.

Note

A broken response, or data loss error, may happen under several circumstances, from server misconfiguration to network errors to data corruption. It is up to the user to decide if it makes sense to process broken responses considering they may contain partial or incomplete content. If `RETRY_ENABLED` is `True` and this setting is set to `True`, the `ResponseFailed([_DataLoss])` failure will be retried as usual.

DUPEFILTER_CLASS

Default: `'scrapy.dupefilters.RFPDupeFilter'`

The class used to detect and filter duplicate requests.

The default (`RFPDupeFilter`) filters based on request fingerprint using the `scrapy.utils.request.request_fingerprint` function. In order to change the way duplicates are checked you could subclass `RFPDupeFilter` and override its `request_fingerprint` method. This method should accept scrapy `Request` object and return its fingerprint (a string).

You can disable filtering of duplicate requests by setting `DUPEFILTER_CLASS` to `'scrapy.dupefilters.BaseDupeFilter'`. Be very careful about this however, because you can get into crawling loops. It's usually a better idea to set the `dont_filter` parameter to `True` on the specific `Request` that should not be filtered.

DUPEFILTER_DEBUG

Default: `False`

By default, `RFPDupeFilter` only logs the first duplicate request. Setting `DUPEFILTER_DEBUG` to `True` will make it log all duplicate requests.

EDITOR

Default: `vi` (on Unix systems) or the IDLE editor (on Windows)

The editor to use for editing spiders with the `edit` command. Additionally, if the `EDITOR` environment variable is set, the `edit` command will prefer it over the default setting.

EXTENSIONS

Default:: `{}`

A dict containing the extensions enabled in your project, and their orders.

EXTENSIONS_BASE

Default:

```
{
    'scrapy.extensions.corestats.CoreStats': 0,
    'scrapy.extensions.telnet.TelnetConsole': 0,
    'scrapy.extensions.memusage.MemoryUsage': 0,
    'scrapy.extensions.memdebug.MemoryDebugger': 0,
    'scrapy.extensions.closespider.CloseSpider': 0,
```

```
    'scrapy.extensions.feedexport.FeedExporter': 0,
    'scrapy.extensions.logstats.LogStats': 0,
    'scrapy.extensions.spiderstate.SpiderState': 0,
    'scrapy.extensions.throttle.AutoThrottle': 0,
}
```

A dict containing the extensions available by default in Scrapy, and their orders. This setting contains all stable built-in extensions. Keep in mind that some of them need to be enabled through a setting.

For more information See the extensions user guide and the list of available extensions.

FEED_TEMPDIR

The Feed Temp dir allows you to set a custom folder to save crawler temporary files before uploading with FTP feed storage and Amazon S3.

FTP_PASSIVE_MODE

Default: `True`

Whether or not to use passive mode when initiating FTP transfers.

FTP_PASSWORD

Default: `"guest"`

The password to use for FTP connections when there is no `"ftp_password"` in `Request` meta.

Note

Paraphrasing RFC 1635, although it is common to use either the password "guest" or one's e-mail address for anonymous FTP, some FTP servers explicitly ask for the user's e-mail address and will not allow login with the "guest" password.

FTP_USER

Default: `"anonymous"`

The username to use for FTP connections when there is no `"ftp_user"` in `Request` meta.

ITEM_PIPELINES

Default: `{}`

A dict containing the item pipelines to use, and their orders. Order values are arbitrary, but it is customary to define them in the 0-1000 range. Lower orders process before higher orders.

Example:

```
ITEM_PIPELINES = {
    'mybot.pipelines.validate.ValidateMyItem': 300,
    'mybot.pipelines.validate.StoreMyItem': 800,
}
```

ITEM_PIPELINES_BASE

Default: `{}`

A dict containing the pipelines enabled by default in Scrapy. You should never modify this setting in your project, modify `ITEM_PIPELINES` instead.

LOG_ENABLED

Default: `True`

Whether to enable logging.

LOG_ENCODING

Default: `'utf-8'`

The encoding to use for logging.

LOG_FILE

Default: `None`

File name to use for logging output. If `None`, standard error will be used.

LOG_FORMAT

Default: `'%(asctime)s [%(name)s] %(levelname)s: %(message)s'`

String for formatting log messsages. Refer to the Python logging documentation for the whole list of available placeholders.

LOG_DATEFORMAT

Default: `'%Y-%m-%d %H:%M:%S'`

String for formatting date/time, expansion of the `%(asctime)s` placeholder in `LOG_FORMAT`. Refer to the Python datetime documentation for the whole list of available directives.

LOG_LEVEL

Default: `'DEBUG'`

Minimum level to log. Available levels are: CRITICAL, ERROR, WARNING, INFO, DEBUG. For more info see Logging.

LOG_STDOUT

Default: `False`

If `True`, all standard output (and error) of your process will be redirected to the log. For example if you `print 'hello'` it will appear in the Scrapy log.

LOG_SHORT_NAMES

Default: `False`

If `True`, the logs will just contain the root path. If it is set to `False` then it displays the component responsible for the log output

MEMDEBUG_ENABLED

Default: `False`

Whether to enable memory debugging.

MEMDEBUG_NOTIFY

Default: `[]`

When memory debugging is enabled a memory report will be sent to the specified addresses if this setting is not empty, otherwise the report will be written to the log.

Example:

```
MEMDEBUG_NOTIFY = ['user@example.com']
```

MEMUSAGE_ENABLED

Default: `True`

Scope: `scrapy.extensions.memusage`

Whether to enable the memory usage extension. This extension keeps track of a peak memory used by the process (it writes it to stats). It can also optionally shutdown the Scrapy process when it exceeds a memory limit (see `MEMUSAGE_LIMIT_MB`), and notify by email when that happened (see `MEMUSAGE_NOTIFY_MAIL`).

See Memory usage extension.

MEMUSAGE_LIMIT_MB

Default: `0`

Scope: `scrapy.extensions.memusage`

The maximum amount of memory to allow (in megabytes) before shutting down Scrapy (if MEMUSAGE_ENABLED is True). If zero, no check will be performed.

See Memory usage extension.

MEMUSAGE_CHECK_INTERVAL_SECONDS

New in version 1.1.

Default: `60.0`

Scope: `scrapy.extensions.memusage`

The Memory usage extension checks the current memory usage, versus the limits set by `MEMUSAGE_LIMIT_MB` and `MEMUSAGE_WARNING_MB`, at fixed time intervals.

This sets the length of these intervals, in seconds.

See Memory usage extension.

MEMUSAGE_NOTIFY_MAIL

Default: `False`

Scope: `scrapy.extensions.memusage`

A list of emails to notify if the memory limit has been reached.

Example:

```
MEMUSAGE_NOTIFY_MAIL = ['user@example.com']
```

See Memory usage extension.

MEMUSAGE_WARNING_MB

Default: `0`

Scope: `scrapy.extensions.memusage`

The maximum amount of memory to allow (in megabytes) before sending a warning email notifying about it. If zero, no warning will be produced.

NEWSPIDER_MODULE

Default: `''`

Module where to create new spiders using the `genspider` command.

Example:

```
NEWSPIDER_MODULE = 'mybot.spiders_dev'
```

RANDOMIZE_DOWNLOAD_DELAY

Default: `True`

If enabled, Scrapy will wait a random amount of time (between 0.5 * `DOWNLOAD_DELAY` and 1.5 * `DOWNLOAD_DELAY`) while fetching requests from the same website.

This randomization decreases the chance of the crawler being detected (and subsequently blocked) by sites which analyze requests looking for statistically significant similarities in the time between their requests.

The randomization policy is the same used by wget `--random-wait` option.

If `DOWNLOAD_DELAY` is zero (default) this option has no effect.

REACTOR_THREADPOOL_MAXSIZE

Default: `10`

The maximum limit for Twisted Reactor thread pool size. This is common multi-purpose thread pool used by various Scrapy components. Threaded DNS Resolver, BlockingFeedStorage, S3FilesStore just to name a few. Increase this value if you're experiencing problems with insufficient blocking IO.

REDIRECT_MAX_TIMES

Default: `20`

Defines the maximum times a request can be redirected. After this maximum the request's response is returned as is. We used Firefox default value for the same task.

REDIRECT_PRIORITY_ADJUST

Default: `+2`

Scope: `scrapy.downloadermiddlewares.redirect.RedirectMiddleware`

Adjust redirect request priority relative to original request:

- **a positive priority adjust (default) means higher priority.**
- a negative priority adjust means lower priority.

RETRY_PRIORITY_ADJUST

Default: `-1`

Scope: `scrapy.downloadermiddlewares.retry.RetryMiddleware`

Adjust retry request priority relative to original request:

- a positive priority adjust means higher priority.
- **a negative priority adjust (default) means lower priority.**

ROBOTSTXT_OBEY

Default: `False`

Scope: `scrapy.downloadermiddlewares.robotstxt`

If enabled, Scrapy will respect robots.txt policies. For more information see RobotsTxtMiddleware.

Note

While the default value is `False` for historical reasons, this option is enabled by default in settings.py file generated by `scrapy startproject` command.

SCHEDULER

Default: `'scrapy.core.scheduler.Scheduler'`

The scheduler to use for crawling.

SCHEDULER_DEBUG

Default: `False`

Setting to `True` will log debug information about the requests scheduler. This currently logs (only once) if the requests cannot be serialized to disk. Stats counter (`scheduler/unserializable`) tracks the number of times this happens.

Example entry in logs:

```
1956-01-31 00:00:00+0800 [scrapy.core.scheduler] ERROR: Unable to serialize request:
<GET http://example.com> - reason: cannot serialize <Request at 0x9a7c7ec>
(type Request)> - no more unserializable requests will be logged
(see 'scheduler/unserializable' stats counter)
```

SCHEDULER_DISK_QUEUE

Default: `'scrapy.squeues.PickleLifoDiskQueue'`

Type of disk queue that will be used by scheduler. Other available types are `scrapy.squeues.PickleFifoDiskQueue`, `scrapy.squeues.MarshalFifoDiskQueue`, `scrapy.squeues.MarshalLifoDiskQueue`.

SCHEDULER_MEMORY_QUEUE

Default: `'scrapy.squeues.LifoMemoryQueue'`

Type of in-memory queue used by scheduler. Other available type is: `scrapy.squeues.FifoMemoryQueue`.

SCHEDULER_PRIORITY_QUEUE

Default: `'queuelib.PriorityQueue'`

Type of priority queue used by scheduler.

SPIDER_CONTRACTS

Default:: `{}`

A dict containing the spider contracts enabled in your project, used for testing spiders. For more info see Spiders Contracts.

SPIDER_CONTRACTS_BASE

Default:

```
{
    'scrapy.contracts.default.UrlContract' : 1,
    'scrapy.contracts.default.ReturnsContract': 2,
    'scrapy.contracts.default.ScrapesContract': 3,
}
```

A dict containing the scrapy contracts enabled by default in Scrapy. You should never modify this setting in your project, modify `SPIDER_CONTRACTS` instead. For more info see Spiders Contracts.

You can disable any of these contracts by assigning `None` to their class path in `SPIDER_CONTRACTS`. E.g., to disable the built-in `ScrapesContract`, place this in your `settings.py`:

```
SPIDER_CONTRACTS = {
    'scrapy.contracts.default.ScrapesContract': None,
}
```

SPIDER_LOADER_CLASS

Default: `'scrapy.spiderloader.SpiderLoader'`

The class that will be used for loading spiders, which must implement the SpiderLoader API.

SPIDER_LOADER_WARN_ONLY

New in version 1.3.3.

Default: `False`

By default, when scrapy tries to import spider classes from `SPIDER_MODULES`, it will fail loudly if there is any `ImportError` exception. But you can choose to silence this exception and turn it into a simple warning by setting `SPIDER_LOADER_WARN_ONLY = True`.

Note

Some scrapy commands run with this setting to `True` already (i.e. they will only issue a warning and will not fail) since they do not actually need to load spider classes to work: `scrapy runspider`,

`scrapy settings`, `scrapy startproject`, `scrapy version`.

SPIDER_MIDDLEWARES

Default:: `{}`

A dict containing the spider middlewares enabled in your project, and their orders. For more info see Activating a spider middleware.

SPIDER_MIDDLEWARES_BASE

Default:

```
{
    'scrapy.spidermiddlewares.httperror.HttpErrorMiddleware': 50,
    'scrapy.spidermiddlewares.offsite.OffsiteMiddleware': 500,
    'scrapy.spidermiddlewares.referer.RefererMiddleware': 700,
    'scrapy.spidermiddlewares.urllength.UrlLengthMiddleware': 800,
    'scrapy.spidermiddlewares.depth.DepthMiddleware': 900,
}
```

A dict containing the spider middlewares enabled by default in Scrapy, and their orders. Low orders are closer to the engine, high orders are closer to the spider. For more info see Activating a spider middleware.

SPIDER_MODULES

Default: `[]`

A list of modules where Scrapy will look for spiders.

Example:

```
SPIDER_MODULES = ['mybot.spiders_prod', 'mybot.spiders_dev']
```

STATS_CLASS

Default: `'scrapy.statscollectors.MemoryStatsCollector'`

The class to use for collecting stats, who must implement the Stats Collector API.

STATS_DUMP

Default: `True`

Dump the Scrapy stats (to the Scrapy log) once the spider finishes.

For more info see: Stats Collection.

STATSMAILER_RCPTS

Default: `[]` (empty list)

Send Scrapy stats after spiders finish scraping. See `StatsMailer` for more info.

TELNETCONSOLE_ENABLED

Default: `True`

A boolean which specifies if the telnet console will be enabled (provided its extension is also enabled).

TELNETCONSOLE_PORT

Default: `[6023, 6073]`

The port range to use for the telnet console. If set to `None` or `0`, a dynamically assigned port is used. For more info see Telnet Console.

TEMPLATES_DIR

Default: `templates` dir inside scrapy module

The directory where to look for templates when creating new projects with `startproject` command and new spiders with `genspider` command.

The project name must not conflict with the name of custom files or directories in the `project` subdirectory.

URLLENGTH_LIMIT

Default: `2083`

Scope: `spidermiddlewares.urllength`

The maximum URL length to allow for crawled URLs. For more information about the default value for this setting see: https://boutell.com/newfaq/misc/urllength.html

USER_AGENT

Default: `"Scrapy/VERSION (+https://scrapy.org)"`

The default User-Agent to use when crawling, unless overridden.

Settings documented elsewhere:

The following settings are documented elsewhere, please check each specific case to see how to enable and use them.

- AJAXCRAWL_ENABLED
- AUTOTHROTTLE_DEBUG
- AUTOTHROTTLE_ENABLED
- AUTOTHROTTLE_MAX_DELAY
- AUTOTHROTTLE_START_DELAY
- AUTOTHROTTLE_TARGET_CONCURRENCY
- CLOSESPIDER_ERRORCOUNT
- CLOSESPIDER_ITEMCOUNT
- CLOSESPIDER_PAGECOUNT
- CLOSESPIDER_TIMEOUT
- COMMANDS_MODULE
- COMPRESSION_ENABLED
- COOKIES_DEBUG
- COOKIES_ENABLED
- FEED_EXPORTERS
- FEED_EXPORTERS_BASE
- FEED_EXPORT_ENCODING
- FEED_EXPORT_FIELDS
- FEED_EXPORT_INDENT
- FEED_FORMAT
- FEED_STORAGES
- FEED_STORAGES_BASE

- REDIRECT_MAX_TIMES
- REFERER_ENABLED
- REFERRER_POLICY
- RETRY_ENABLED
- RETRY_HTTP_CODES
- RETRY_TIMES
- TELNETCONSOLE_HOST
- TELNETCONSOLE_PORT

« Previous

Next »