设置

Scrapy设置允许您自定义所有Scrapy组件的行为，包括核心，扩展，管道和蜘蛛本身。

设置的基础结构提供了键值映射的全局命名空间，代码可以使用该命名空间从中提取配置值。可以通过不同的机制填充设置，如下所述。

这些设置也是选择当前活动的Scrapy项目的机制（如果你有很多）。

有关可用内置设置的列表，请参阅：内置设置参考。

指定设置

当您使用Scrapy时，您必须告诉它您正在使用哪些设置。您可以使用环境变量来完成此操作`SCRAPY_SETTINGS_MODULE`。

值`SCRAPY_SETTINGS_MODULE`应该是Python路径语法，例如`myproject.settings`。请注意，设置模块应位于Python 导入搜索路径中。

设置

可以使用不同的机制填充设置，每个机制具有不同的优先级。以下是按优先顺序递减的列表：

1. 命令行选项（最优先）
2. 每蜘蛛的设置
3. 项目设置模块
4. 每个命令的默认设置
5. 默认全局设置（优先级较低）

这些设置源的数量由内部处理，但可以使用API调用进行手动处理。请参阅 Settings API主题以供参考。

下面更详细地描述这些机制。

1.命令行选项

命令行提供的参数是最优先的参数，覆盖任何其他选项。您可以使用`-s`（或`--set`）命令行选项显式覆盖一个（或多个）设置。

例：

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

2.每蜘蛛的设置

蜘蛛（参见Spiders章节以供参考）可以定义自己的设置，这些设置优先并覆盖项目设置。他们可以通过设置`custom_settings`属性来实现：

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

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

3.项目设置模块

项目设置模块是Scrapy项目的标准配置文件，它将填充大多数自定义设置。对于标准Scrapy项目，这意味着您将添加或更改`settings.py`为项目创建的文件中的设置。

4.默认设置per-

每个Scrapy工具命令都可以有自己的默认设置，这些设置会覆盖全局默认设置。这些自定义命令设置`default_settings`在命令类的属性中指定。

5.默认全局设置

全局默认值位于`scrapy.settings.default_settings`模块中，并记录在内置设置参考部分中。

如何访问设置

在蜘蛛中，设置可通过`self.settings`以下方式获得：

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

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

注意

`settings`在初始化蜘蛛之后，在基础Spider类中设置该属性。如果要在初始化之前使用设置（例如，在spider的`__init__()`方法中），则需要覆盖该`from_crawler()`方法。

可以通过`scrapy.crawler.Crawler.settings` Crawler 的属性访问设置，该属性传递给`from_crawler`扩展，中间件和项目管道中的方法：

```
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'))
```

设置对象可以像dict一样使用（例如`settings['LOG_ENABLED']`），但通常首选使用`Settings`API提供的方法之一以您需要的格式提取设置以避免类型错误。

设置名称的基本原理

设置名称通常以它们配置的组件为前缀。例如，对于一个虚构的robots.txt扩展正确的设置名称会`ROBOTSTXT_ENABLED`，`ROBOTSTXT_OBEY`，`ROBOTSTXT_CACHEDIR`，等。

内置设置参考

以下是所有可用Scrapy设置的列表，按字母顺序排列，以及它们的默认值和应用范围。

范围（如果可用）显示设置的使用位置，如果它与任何特定组件相关联。在那种情况下，将显示该组件的模块，通常是扩展，中间件或管道。它还意味着必须启用该组件才能使设置产生任何效果。

AWS_ACCESS_KEY_ID

默认：`None`

需要访问Amazon Web服务的代码使用的AWS访问密钥，例如S3 Feed存储后端。

AWS_SECRET_ACCESS_KEY

默认：`None`

需要访问Amazon Web服务的代码使用的AWS密钥，例如S3 Feed存储后端。

BOT_NAME

默认：`'scrapybot'`

此Scrapy项目实现的bot的名称（也称为项目名称）。这将默认用于构建User-Agent，也用于日志记录。

使用该 `startproject` 命令创建项目时，它会自动填充项目名称。

CONCURRENT_ITEMS

默认： `100`

在项目处理器（也称为项目管道）中并行处理的最大并发项目数（每个响应）。

CONCURRENT_REQUESTS

默认： `16`

Scrapy下载程序将执行的最大并发（即同时）请求数。

CONCURRENT_REQUESTS_PER_DOMAIN

默认： `8`

将对任何单个域执行的最大并发（即同时）请求数。

另请参阅：AutoThrottle扩展及其 `AUTOTHROTTLE_TARGET_CONCURRENCY` 选项。

CONCURRENT_REQUESTS_PER_IP

默认： `0`

将对任何单个IP执行的最大并发（即同时）请求数。如果非零， `CONCURRENT_REQUESTS_PER_DOMAIN` 则忽略该 设置，而使用此设置。换句话说，并发限制将应用于每个IP，而不是每个域。

此设置还会影响 `DOWNLOAD_DELAY` 和 AutoThrottle扩展：如果 `CONCURRENT_REQUESTS_PER_IP` 非零，则按 IP而不是每个域强制执行下载延迟。

DEFAULT_ITEM_CLASS

默认： `'scrapy.item.Item'`

将用于实例化Scrapy shell中的项的默认类。

DEFAULT_REQUEST_HEADERS

默认：

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

用于Scrapy HTTP请求的默认标头。他们居住在这里 `DefaultHeadersMiddleware` 。

DEPTH_LIMIT

默认： `0`

范围： `scrapy.spidermiddlewares.depth.DepthMiddleware`

允许为任何站点爬网的最大深度。如果为零，则不会施加任何限制。

DEPTH_PRIORITY

默认： `0`

范围： `scrapy.spidermiddlewares.depth.DepthMiddleware`

一个整数，用于根据深度调整请求优先级：

- 如果为零（默认），则不从深度进行优先级调整
- **正值将降低优先级，即稍后将处理更高深度的请求**；这在进行广度优先爬网（BFO）时常用
- 负值将增加优先级，即更快的深度请求将被更快地处理（DFO）

另请参阅：Scrapy是以广度优先还是深度优先顺序爬行？关于为BFO或DFO调整Scrapy。

> **注意**
>
> 此设置调整优先级**以相反的方式**相对于其他优先级设置 `REDIRECT_PRIORITY_ADJUST` 和 `RETRY_PRIORITY_ADJUST` 。

DEPTH_STATS

默认： `True`

范围： `scrapy.spidermiddlewares.depth.DepthMiddleware`

是否收集最大深度统计数据。

DEPTH_STATS_VERBOSE

默认： `False`

范围： `scrapy.spidermiddlewares.depth.DepthMiddleware`

是否收集详细的深度统计数据。如果启用此选项，则会在统计信息中收集每个深度的请求数。

DNSCACHE_ENABLED

默认： `True`

是否启用DNS内存缓存。

DNSCACHE_SIZE

默认： `10000`

DNS内存缓存大小。

DNS_TIMEOUT

默认： `60`

在几秒钟内处理DNS查询的超时。支持浮动。

下载

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

用于抓取的下载程序。

DOWNLOADER_HTTPCLIENTFACTORY

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

定义 `protocol.ClientFactory` 用于HTTP / 1.0连接（for `HTTP10DownloadHandler`）的Twisted 类。

> **注意**
>
> HTTP / 1.0现在很少或使用，因此您可以放心地忽略此设置，除非你使用双绞线<11.1，如果你真的想使用HTTP / 1.0，并覆盖 `DOWNLOAD_HANDLERS_BASE` 了 `http(s)` 相应的方案，即 `'scrapy.core.downloader.handlers.http.HTTP10DownloadHandler'`。

DOWNLOADER_CLIENTCONTEXTFACTORY

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

表示要使用的ContextFactory的类路径。

这里，“ContextFactory”是SSL / TLS上下文的Twisted术语，定义了要使用的TLS / SSL协议版本，是否进行证书验证，甚至启用客户端身份验证（以及其他各种事情）。

> **注意**
>
> Scrapy默认上下文工厂**不执行远程服务器证书验证**。这通常适用于网页抓取。
>
> 如果确实需要启用远程服务器证书验证，Scrapy还有另一个可以设置的上下文工厂类 `'scrapy.core.downloader.contextfactory.BrowserLikeContextFactory'`，它使用平台的证书来验证远程端点。 **仅当您使用Twisted> = 14.0时才可用。**

如果您确实使用自定义ContextFactory，请确保它 `method` 在init 接受参数（这是 `OpenSSL.SSL` 方法映射 `DOWNLOADER_CLIENT_TLS_METHOD` ）。

DOWNLOADER_CLIENT_TLS_METHOD

默认： `'TLS'`

使用此设置可自定义默认HTTP / 1.1下载程序使用的TLS / SSL方法。

此设置必须是以下字符串值之一：

- `'TLS'`：映射到OpenSSL `TLS_method()`（又名 `SSLv23_method()`），它允许协议协商，从平台支持的最高点开始; **默认，推荐**
- `'TLSv1.0'`：此值强制HTTPS连接使用TLS 1.0版; 如果你想要Scrapy <1.1的行为，请设置此项
- `'TLSv1.1'`：强制TLS版本1.1
- `'TLSv1.2'`：强制TLS版本1.2
- `'SSLv3'`：强制SSL版本3（**不推荐**）

> **注意**
>
> 我们建议您使用PyOpenSSL> = 0.13和Twisted> = 0.13或更高（如果可以，Twisted> = 14.0）。

DOWNLOADER_MIDDLEWARES

默认：： `{}`

包含项目中启用的下载器中间件及其订单的dict。有关更多信息，请参阅激活下载中间件。

DOWNLOADER_MIDDLEWARES_BASE

默认：

```
{
    '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,
}
```

包含Scrapy中默认启用的下载器中间件的dict。低订单更接近引擎，高订单更接近下载。您永远不应该在项目中修改此设置，`DOWNLOADER_MIDDLEWARES`而是修改 。有关更多信息，请参阅 激活下载中间件。

DOWNLOADER_STATS

默认： `True`

是否启用下载统计信息收集。

DOWNLOAD_DELAY

默认： `0`

下载器从同一网站下载连续页面之前应等待的时间（以秒为单位）。这可以用来限制爬行速度，以避免过于严重地击中服务器。支持十进制数。例：

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

此设置也受`RANDOMIZE_DOWNLOAD_DELAY` 设置（默认情况下启用）的影响。默认情况下，Scrapy不会在请求之间等待一段固定的时间，而是使用0.5 * `DOWNLOAD_DELAY`和1.5 * 之间的随机间隔`DOWNLOAD_DELAY` 。

当`CONCURRENT_REQUESTS_PER_IP`非零时，每个IP地址而不是每个域强制执行延迟。

您还可以通过设置`download_delay` spider属性来更改每个蜘蛛的此设置。

DOWNLOAD_HANDLERS

默认： `{}`

包含项目中启用的请求下载程序处理程序的dict。请参阅 `DOWNLOAD_HANDLERS_BASE` 格式。

DOWNLOAD_HANDLERS_BASE

默认：

```
{
    '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',
}
```

包含Scrapy中默认启用的请求下载处理程序的dict。您永远不应该在项目中修改此设置， `DOWNLOAD_HANDLERS` 而是修改 。

您可以通过分配 `None` 其URI方案来禁用任何这些下载处理程序 `DOWNLOAD_HANDLERS` 。例如，要禁用内置的FTP处理程序（无需替换），请将其放在 `settings.py` ：

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

DOWNLOAD_TIMEOUT

默认： `180`

下载器在超时之前等待的时间（以秒为单位）。

> **注意**
>
> 可以使用 `download_timeout` spider属性为每个蜘蛛设置此超时，使用 `download_timeout` Request.meta键为每个请求设置此超时。

DOWNLOAD_MAXSIZE

默认值：*1073741824*（1024MB）

下载程序将下载的最大响应大小（以字节为单位）。

如果要禁用它，请将其设置为0。

> **注意**
>
> 可以使用`download_maxsize` spider属性为每个蜘蛛设置此大小，使用`download_maxsize` Request.meta键为每个请求设置此大小。
>
> 此功能需要Twisted> = 11.1。

DOWNLOAD_WARNSIZE

默认值：*33554432*（32MB）

下载程序将开始发出警告的响应大小（以字节为单位）。

如果要禁用它，请将其设置为0。

> **注意**
>
> 可以使用`download_warnsize` spider属性为每个蜘蛛设置此大小，使用`download_warnsize` Request.meta键为每个请求设置此大小。
>
> 此功能需要Twisted> = 11.1。

DOWNLOAD_FAIL_ON_DATALOSS

默认：`True`

是否在响应中断时失败，即声明`Content-Length`与服务器发送的内容不匹配或者分块响应未正确完成。如果`True`，这些响应会引发`ResponseFailed([_DataLoss])`错误。如果`False`，这些响应被传递并且标志`dataloss`被添加到响应中，即：是。`'dataloss' in response.flags` `True`

（可选）可以使用`download_fail_on_dataloss` Request.meta键为每个请求设置`False`。

> **注意**
>
> 在从服务器配置错误到网络错误再到数据损坏的几种情况下，可能会发生损坏的响应或数据丢失错误。由用户决定处理损坏的响应是否有意义，因为它们可能包含部分或不完整的内容。如果`RETRY_ENABLED`是`True`并且此设置设置为`True`，`ResponseFailed([_DataLoss])`则将像往常一样重试失败。

DUPEFILTER_CLASS

默认：`'scrapy.dupefilters.RFPDupeFilter'`

用于检测和过滤重复请求的类。

默认（`RFPDupeFilter`）过滤器使用该`scrapy.utils.request.request_fingerprint`功能基于请求指纹。为了更改检查重复项的方式，您可以继承`RFPDupeFilter`并覆盖其`request_fingerprint`方法。此方法应接受scrapy `Request`对象并返回其指纹（字符串）。

您可以通过设置`DUPEFILTER_CLASS`为禁用对重复请求的过滤`'scrapy.dupefilters.BaseDupeFilter'`。但是要非常小心，因为你可以进入爬行循环。通常最好将`dont_filter`参数设置为不应过滤`True`的特定参数`Request`。

DUPEFILTER_DEBUG

默认：`False`

默认情况下，`RFPDupeFilter`仅记录第一个重复请求。设置`DUPEFILTER_DEBUG`为`True`将使其记录所有重复的请求。

编辑

默认值:(`vi`在Unix系统上）或IDLE编辑器（在Windows上）

用于使用`edit`命令编辑蜘蛛的编辑器。此外，如果`EDITOR`设置了环境变量，则`edit`命令将优先于默认设置。

扩展

默认：：`{}`

包含项目中启用的扩展名及其订单的dict。

EXTENSIONS_BASE

默认：

```
{
    '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,
}
```

包含Scrapy中默认可用扩展名的dict及其订单。此设置包含所有稳定的内置扩展。请记住，其中一些需要通过设置启用。

有关详细信息，请参阅扩展程序用户指南 和可用扩展程序列表。

FEED_TEMPDIR

Feed Temp dir允许您在使用FTP Feed存储和 Amazon S3上载之前设置自定义文件夹以保存搜寻器临时文件。

FTP_PASSIVE_MODE

默认： `True`

启动FTP传输时是否使用被动模式。

FTP_PASSWORD

默认： `"guest"`

该密码才能使用FTP连接时，有没有`"ftp_password"` 在`Request`元。

> **注意**
>
> 解释RFC 1635，虽然通常使用密码“guest”或匿名FTP的一个电子邮件地址，但某些FTP服务器明确要求用户的电子邮件地址，并且不允许使用“访客”密码登录。

FTP_USER

默认： `"anonymous"`

用户名使用的FTP连接时，有没有`"ftp_user"` 在`Request`元。

ITEM_PIPELINES

默认： `{}`

包含要使用的项目管道的dict及其订单。订单值是任意的，但通常在0-1000范围内定义它们。在更高订单之前降低订单处理。

例：

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

ITEM_PIPELINES_BASE

默认： `{}`

包含Scrapy中默认启用的管道的dict。您永远不应该在项目中修改此设置， `ITEM_PIPELINES` 而是修改。

LOG_ENABLED

默认： `True`

是否启用日志记录。

LOG_ENCODING

默认： `'utf-8'`

用于记录的编码。

LOG_FILE

默认： `None`

用于记录输出的文件名。如果 `None` ，将使用标准错误。

LOG_FORMAT

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

用于格式化日志消息的字符串。有关可用占位符的完整列表，请参阅Python日志记录文档。

LOG_DATEFORMAT

默认： `'%Y-%m-%d %H:%M:%S'`

用于格式化日期/时间的字符串，用于扩展 `%(asctime)s` 占位符 `LOG_FORMAT` 。有关可用指令的完整列表，请参阅Python datetime文档。

LOG_LEVEL

默认： `'DEBUG'`

记录的最低级别。可用级别为：CRITICAL，ERROR，WARNING，INFO，DEBUG。有关更多信息，请参阅记录。

LOG_STDOUT

默认： `False`

如果 `True` ，您的进程的所有标准输出（和错误）将被重定向到日志。例如，如果它将出现在Scrapy日志中。 `print 'hello'`

LOG_SHORT_NAMES

默认： `False`

如果 `True` ，日志将只包含根路径。如果设置为 `False` 则显示负责日志输出的组件

MEMDEBUG_ENABLED

默认： `False`

是否启用内存调试。

MEMDEBUG_NOTIFY

默认： `[]`

启用内存调试时，如果此设置不为空，则会将内存报告发送到指定的地址，否则报告将写入日志。

例：

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

MEMUSAGE_ENABLED

默认： `True`

范围： `scrapy.extensions.memusage`

是否启用内存使用扩展。此扩展程序跟踪进程使用的峰值内存（将其写入统计信息）。它还可以选择在超出内存限制时关闭Scrapy进程（请参阅参考资料`MEMUSAGE_LIMIT_MB`），并在发生时通过电子邮件通知（请参阅参考资料`MEMUSAGE_NOTIFY_MAIL`）。

请参阅内存使用扩展。

MEMUSAGE_LIMIT_MB

默认： `0`

范围： `scrapy.extensions.memusage`

关闭Scrapy之前允许的最大内存量（以兆字节为单位）（如果MEMUSAGE_ENABLED为True）。如果为零，则不执行检查。

请参阅内存使用扩展。

MEMUSAGE_CHECK_INTERVAL_SECONDS

版本1.1中的新功能。

默认： `60.0`

范围： `scrapy.extensions.memusage`

的内存使用扩展 会检查当前存储器使用，相对于限制由设置 `MEMUSAGE_LIMIT_MB` 和 `MEMUSAGE_WARNING_MB`，以固定时间间隔。

这将设置这些间隔的长度，以秒为单位。

请参阅内存使用扩展。

MEMUSAGE_NOTIFY_MAIL

默认： `False`

范围： `scrapy.extensions.memusage`

如果已达到内存限制，则通知的电子邮件列表。

例：

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

请参阅内存使用扩展。

MEMUSAGE_WARNING_MB

默认： `0`

范围： `scrapy.extensions.memusage`

发送警告电子邮件通知之前允许的最大内存量（以兆字节为单位）。如果为零，则不会产生警告。

NEWSPIDER_MODULE

默认： `''`

使用该 `genspider` 命令模块在哪里创建新的蜘蛛。

例：

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

RANDOMIZE_DOWNLOAD_DELAY

默认： `True`

如果启用，Scrapy将在从同一网站获取请求时等待一段随机时间（介于0.5 * `DOWNLOAD_DELAY` 和1.5 *之间 `DOWNLOAD_DELAY` ）。

这种随机化降低了分析请求在其请求之间寻找统计上显着相似性的请求被检测（并随后被阻止）的机会。

随机化策略与wget `--random-wait` 选项使用的相同。

如果 `DOWNLOAD_DELAY` 为零（默认），则此选项无效。

REACTOR_THREADPOOL_MAXSIZE

默认： `10`

Twisted Reactor线程池大小的最大限制。这是各种Scrapy组件使用的常见多用途线程池。螺纹DNS解析器，BlockingFeedStorage，S3FilesStore仅举几例。如果遇到阻塞IO不足的问题，请增加此值。

REDIRECT_MAX_TIMES

默认： `20`

定义可以重定向请求的最大次数。在此最大值之后，请求的响应将按原样返回。我们使用Firefox默认值来执行相同的任务。

REDIRECT_PRIORITY_ADJUST

默认： `+2`

范围： `scrapy.downloadermiddlewares.redirect.RedirectMiddleware`

相对于原始请求调整重定向请求优先级：

- **正优先级调整（默认）意味着更高的优先级。**
- 负优先级调整意味着较低的优先级。

RETRY_PRIORITY_ADJUST

默认： `-1`

范围： `scrapy.downloadermiddlewares.retry.RetryMiddleware`

相对于原始请求调整重试请求优先级：

- 正优先级调整意味着更高的优先级。
- **负优先级调整（默认）表示优先级较低。**

ROBOTSTXT_OBEY

默认： `False`

范围： `scrapy.downloadermiddlewares.robotstxt`

如果启用，Scrapy将尊重robots.txt政策。有关更多信息，请参阅 RobotsTxtMiddleware。

> **注意**
>
> 虽然默认值是 `False` 出于历史原因，但默认情况下会在命令生成的settings.py文件中启用此选项。 `scrapy startproject`

调度器

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

用于爬网的调度程序。

SCHEDULER_DEBUG

默认： `False`

设置为 `True` 将记录有关请求调度程序的调试信息。如果请求无法序列化到磁盘，则此当前日志（仅一次）。Stats counter（ `scheduler/unserializable` ）跟踪发生这种情况的次数。

日志中的示例条目：

```
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

默认： `'scrapy.squeues.PickleLifoDiskQueue'`

调度程序将使用的磁盘队列类型。其他可用的类型有 `scrapy.squeues.PickleFifoDiskQueue` ， `scrapy.squeues.MarshalFifoDiskQueue` ， `scrapy.squeues.MarshalLifoDiskQueue` 。

SCHEDULER_MEMORY_QUEUE

默认： `'scrapy.squeues.LifoMemoryQueue'`

调度程序使用的内存中队列的类型。其他可用的类型是： `scrapy.squeues.FifoMemoryQueue` 。

SCHEDULER_PRIORITY_QUEUE

默认： `'queuelib.PriorityQueue'`

调度程序使用的优先级队列的类型。

SPIDER_CONTRACTS

默认：： `{}`

包含在项目中启用的蜘蛛合同的dict，用于测试蜘蛛。有关更多信息，请参阅蜘蛛合同。

SPIDER_CONTRACTS_BASE

默认：

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

包含Scrapy默认情况下启用的scrapy合约的dict。您永远不应该在项目中修改此设置， `SPIDER_CONTRACTS` 而是修改。有关更多信息，请参阅蜘蛛合同。

您可以通过分配 `None` 其类路径来禁用任何这些合同 `SPIDER_CONTRACTS` 。例如，要禁用内置功能 `ScrapesContract` ，请将其放入 `settings.py` ：

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

SPIDER_LOADER_CLASS

默认： `'scrapy.spiderloader.SpiderLoader'`

将用于加载蜘蛛的类，必须实现 SpiderLoader API。

SPIDER_LOADER_WARN_ONLY

版本1.3.3中的新功能。

默认： `False`

默认情况下，当scrapy尝试从中导入蜘蛛类时 `SPIDER_MODULES` ，如果有任何 `ImportError` 异常，它将大声失败。但您可以选择将此异常静音并通过设置将其转换为简单警告。 `SPIDER_LOADER_WARN_ONLY = True`

> **注意**
>
> 有些scrapy命令使用此设置运行 `True` 已经（即他们只会发出警告并不会失败），因为他们实际上并不需要加载蜘蛛类的工作： ， ， ， 。 `scrapy runspider` `scrapy settings` `scrapy startproject` `scrapy version`

SPIDER_MIDDLEWARES

默认：： `{}`

包含项目中启用的蜘蛛中间件的dict及其命令。有关更多信息，请参阅激活蜘蛛中间件。

SPIDER_MIDDLEWARES_BASE

默认：

```
{
    '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,
}
```

包含Scrapy中默认启用的蜘蛛中间件的dict及其命令。低订单更接近引擎，高订单更接近蜘蛛。有关更多信息，请参阅激活蜘蛛中间件。

SPIDER_MODULES

默认： `[]`

Scrapy将寻找蜘蛛的模块列表。

例：

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

STATS_CLASS

默认： `'scrapy.statscollectors.MemoryStatsCollector'`

用于收集统计信息的类，必须实现 Stats Collector API。

STATS_DUMP

默认： `True`

蜘蛛完成后，将Scrapy统计数据（转到Scrapy日志）转储。

有关详细信息，请参阅：统计数据集。

STATSMAILER_RCPTS

默认值:(`[]` 空列表)

在蜘蛛完成刮擦后发送Scrapy统计数据。有关详情 `StatsMailer` ，请参阅 。

TELNETCONSOLE_ENABLED

默认： `True`

一个布尔值，指定是否 启用telnet控制台（如果其扩展名也已启用）。

TELNETCONSOLE_PORT

默认： `[6023, 6073]`

用于telnet控制台的端口范围。如果设置为 `None` 或 `0` ，则使用动态分配的端口。有关详细信息，请参阅 Telnet控制台。

TEMPLATES_DIR

默认值： `templates` dir在scrapy模块中

使用 `startproject` 命令创建新项目时使用命令查找模板的目录以及使用命令创建 新蜘蛛的目录 `genspider` 。

项目名称不得与 `project` 子目录中的自定义文件或目录的名称冲突。

URLLENGTH_LIMIT

默认： `2083`

范围： `spidermiddlewares.urllength`

允许抓取的网址的最大网址长度。有关此设置的默认值的详细信息，请参阅：
https：//boutell.com/newfaq/misc/urllength.html

USER_AGENT

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

爬网时使用的默认User-Agent，除非被覆盖。

其他地方记录的设置：

其他地方记录了以下设置，请检查每个特定情况以了解如何启用和使用它们。

- 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
- FEED_STORE_EMPTY
- FEED_URI
- FILES_EXPIRES
- FILES_RESULT_FIELD
- FILES_STORE
- FILES_STORE_S3_ACL
- FILES_URLS_FIELD
- GCS_PROJECT_ID
- HTTPCACHE_ALWAYS_STORE
- HTTPCACHE_DBM_MODULE
- HTTPCACHE_DIR
- HTTPCACHE_ENABLED
- HTTPCACHE_EXPIRATION_SECS
- HTTPCACHE_GZIP
- HTTPCACHE_IGNORE_HTTP_CODES
- HTTPCACHE_IGNORE_MISSING
- HTTPCACHE_IGNORE_RESPONSE_CACHE_CONTROLS
- HTTPCACHE_IGNORE_SCHEMES
- HTTPCACHE_POLICY
- HTTPCACHE_STORAGE
- HTTPERROR_ALLOWED_CODES
- HTTPERROR_ALLOW_ALL
- HTTPPROXY_AUTH_ENCODING
- HTTPPROXY_ENABLED
- IMAGES_EXPIRES
- IMAGES_MIN_HEIGHT
- IMAGES_MIN_WIDTH

- IMAGES_RESULT_FIELD
- IMAGES_STORE
- IMAGES_STORE_S3_ACL
- IMAGES_THUMBS
- IMAGES_URLS_FIELD
- MAIL_FROM
- MAIL_HOST
- MAIL_PASS
- MAIL_PORT
- MAIL_SSL
- MAIL_TLS
- MAIL_USER
- MEDIA_ALLOW_REDIRECTS
- METAREFRESH_ENABLED
- METAREFRESH_MAXDELAY
- REDIRECT_ENABLED
- REDIRECT_MAX_TIMES
- REFERER_ENABLED
- REFERRER_POLICY
- RETRY_ENABLED
- RETRY_HTTP_CODES
- RETRY_TIMES
- TELNETCONSOLE_HOST
- TELNETCONSOLE_PORT