蜘蛛中间件

蜘蛛中间件是一个钩入Scrapy蜘蛛处理机制的框架，您可以在其中插入自定义功能来处理发送给Spiders的响应，以便处理和处理从蜘蛛生成的请求和项目。

激活蜘蛛中间件

要激活蜘蛛中间件组件，请将其添加到 `SPIDER_MIDDLEWARES` 设置中，该设置是一个dict，其键是中间件类路径，它们的值是中间件命令。

这是一个例子：

```
SPIDER_MIDDLEWARES = {
    'myproject.middlewares.CustomSpiderMiddleware': 543,
}
```

该 `SPIDER_MIDDLEWARES` 设置与 `SPIDER_MIDDLEWARES_BASE` Scrapy中定义的设置合并（并不意味着被覆盖），然后按顺序排序，以获得已启用的中间件的最终排序列表：第一个中间件是靠近引擎的中间件，最后一个是更接近引擎的中间件对蜘蛛。换句话说，`process_spider_input()` 将以增加的中间件顺序（100,200,300，...）`process_spider_output()` 调用每个中间件的方法，并且将按递减顺序调用每个中间件的 方法。

要确定分配给中间件的顺序，请参阅 `SPIDER_MIDDLEWARES_BASE` 设置并根据要插入中间件的位置选择值。订单很重要，因为每个中间件执行不同的操作，您的中间件可能依赖于应用的某些先前（或后续）中间件。

如果要禁用内置中间件（ `SPIDER_MIDDLEWARES_BASE` 默认情况下定义的中间件 ），则必须在项目 `SPIDER_MIDDLEWARES` 设置中定义它并将*None*指定为其值。例如，如果要禁用异地中间件：

```
SPIDER_MIDDLEWARES = {
    'myproject.middlewares.CustomSpiderMiddleware': 543,
    'scrapy.spidermiddlewares.offsite.OffsiteMiddleware': None,
}
```

最后，请记住，可能需要通过特定设置启用某些中间件。有关详细信息，请参阅每个中间件文档。

编写自己的蜘蛛中间件

每个中间件组件都是一个Python类，它定义了以下一个或多个方法：

类 scrapy.spidermiddlewares.SpiderMiddleware

process_spider_input（*响应*，*蜘蛛*）

对于通过蜘蛛中间件并进入蜘蛛的每个响应，都会调用此方法进行处理。

`process_spider_input()` 应该返回 `None` 或提出异常。

如果它返回 `None` ，Scrapy将继续处理此响应，执行所有其他中间件，直到最后，响应被交给蜘蛛进行处理。

如果它引发异常，Scrapy将不会打扰任何其他蜘蛛中间件 `process_spider_input()` ，并将调用请求errback。errback的输出被链接回另一个方向 `process_spider_output()` 以便处理它，或者 `process_spider_exception()` 它是否引发异常。

参数：
- **response**（`Response` object）- 正在处理的响应
- **spider**（`Spider` object）- 此响应所针对的蜘蛛

process_spider_output（*响应*，*结果*，*蜘蛛*）

在处理响应之后，使用Spider返回的结果调用此方法。

`process_spider_output()` 必须返回一个可迭代的 `Request` ，dict或 `Item` 对象。

参数：
- **response**（`Response` object）- 从蜘蛛生成此输出的响应
- **结果**（可迭代的 `Request` ，dict或 `Item` 对象）- 蜘蛛返回的结果
- **spider**（`Spider` object）- 正在处理其结果的蜘蛛

process_spider_exception（*响应*，*异常*，*蜘蛛*）

当蜘蛛或 `process_spider_input()` 方法（来自其他蜘蛛中间件）引发异常时，将调用此方法。

`process_spider_exception()` 应返回 `None` 或者是可迭代的 `Request` ，dict或 `Item` 对象。

如果它返回 `None` ，Scrapy将继续处理此异常，执行 `process_spider_exception()` 以下中间件组件中的任何其他组件，直到没有剩余中间件组件并且异常到达引擎（它被记录并丢弃）。

如果它返回一个iterable，则 `process_spider_output()` 管道启动，并且不会 `process_spider_exception()` 调用其他管道。

参数：
- **response**（`Response` object）- 引发异常时正在处理的响应
- **exception**（异常对象）- 引发异常
- **spider**（`Spider` object）- 引发异常的蜘蛛

process_start_requests(*start_requests*, *spider*)

0.15版本的新功能。

使用spider的启动请求调用此方法，并且该`process_spider_output()`方法与该方法的工作方式类似，不同之处在于它没有关联的响应，并且必须仅返回请求（而不是项目）。

它接收一个iterable（在`start_requests`参数中）并且必须返回另一个可迭代的`Request`对象。

注意

在您的蜘蛛中间件中实现此方法时，您应该始终返回一个iterable（在输入之后）并且不消耗所有`start_requests`迭代器，因为它可能非常大（甚至无限制）并导致内存溢出。Scrapy引擎设计用于在有能力处理它们的情况下提取启动请求，因此启动请求迭代器可以在有一些其他条件停止蜘蛛（如时间限制或项目/页数）时有效。

参数：
- **start_requests**（可迭代`Request`） - 开始请求
- **spider**（`Spider` object） - 启动请求所属的蜘蛛

from_crawler(*cls*, *crawler*)

如果存在，则调用此类方法以从a创建中间件实例`Crawler`。它必须返回一个新的中间件实例。Crawler对象提供对所有Scrapy核心组件的访问，如设置和信号; 它是中间件访问它们并将其功能挂钩到Scrapy的一种方式。

参数： **crawler**（`Crawler` object） - 使用此中间件的爬网程序

内置蜘蛛中间件参考

此页面描述了Scrapy附带的所有蜘蛛中间件组件。有关如何使用它们以及如何编写自己的蜘蛛中间件的信息，请参阅蜘蛛中间件使用指南。

有关默认启用的组件列表（及其订单），请参阅`SPIDER_MIDDLEWARES_BASE`设置。

DepthMiddleware

类 **scrapy.spidermiddlewares.depth.DepthMiddleware**

DepthMiddleware用于跟踪被抓取站点内每个请求的深度。只要没有先前设置的值（通常只是第一个请求）并将其递增1，它就可以通过设置*request.meta ['depth'] = 0*来工作。

它可用于限制刮擦的最大深度，根据深度控制请求优先级等。

该`DepthMiddleware`可通过以下设置进行配置（详情参见设置文档）：

- `DEPTH_LIMIT` - 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_STATS` - 是否收集深度统计数据。
- `DEPTH_PRIORITY` - 是否根据深度确定请求的优先级。

HttpErrorMiddleware

类 **scrapy.spidermiddlewares.httperror.HttpErrorMiddleware**

过滤掉不成功（错误）的HTTP响应，以便蜘蛛不必处理它们（大多数时候）会产生开销，消耗更多资源，并使蜘蛛逻辑更复杂。

根据HTTP标准，成功的响应是那些状态代码在200-300范围内的响应。

如果您仍希望处理该范围之外的响应代码，则可以使用`handle_httpstatus_list`spider属性或`HTTPERROR_ALLOWED_CODES`设置指定spider能够处理的响应代码 。

例如，如果您希望蜘蛛处理404响应，您可以执行以下操作：

```
class MySpider(CrawlSpider):
    handle_httpstatus_list = [404]
```

所述`handle_httpstatus_list`的键`Request.meta`也可以被用于指定的响应代码，以允许在每个请求基础。您还可以设置meta键`handle_httpstatus_all`来`True`，如果你想以允许请求的任何响应代码。

但请记住，处理非200响应通常是一个坏主意，除非你真的知道你在做什么。

有关更多信息，请参阅：HTTP状态代码定义。

HttpErrorMiddleware设置

HTTPERROR_ALLOWED_CODES

默认：`[]`

传递包含在此列表中的非200状态代码的所有响应。

HTTPERROR_ALLOW_ALL

默认：`False`

传递所有响应，无论其状态代码如何。

OffsiteMiddleware

类 **scrapy.spidermiddlewares.offsite.OffsiteMiddleware**

过滤掉蜘蛛所涵盖域外的URL请求。

此中间件过滤掉主机名不在spider `allowed_domains` 属性中的每个请求。列表中任何域的所有子域也是允许的。例如，规则 `www.example.org` 也允许 `bob.www.example.org` ，但不能 `www2.example.com` 也不 `example.com` 。

当您的蜘蛛返回不属于蜘蛛所涵盖的域的请求时，此中间件将记录类似于此的调试消息：

```
DEBUG: Filtered offsite request to 'www.othersite.com': <GET
http://www.othersite.com/some/page.html>
```

为了避免在日志中填充太多噪音，它只会为每个过滤的新域打印其中一条消息。因此，例如，如果 `www.othersite.com` 过滤了另一个请求，则不会打印任何日志消息。但是，如果请求 `someothersite.com` 被过滤，则将打印一条消息（但仅针对第一个请求进行过滤）。

如果蜘蛛没有定义 `allowed_domains` 属性，或者属性为空，则异地中间件将允许所有请求。

如果请求具有 `dont_filter` 设置的属性，则异地中间件将允许该请求，即使其域未在允许的域中列出。

RefererMiddleware

类 **scrapy.spidermiddlewares.referer.RefererMiddleware**

`Referer` 根据生成它的响应的URL 填充请求标头。

RefererMiddleware设置

REFERER_ENABLED

0.15版本的新功能。

默认： `True`

是否启用referer中间件。

REFERRER_POLICY

版本1.4中的新功能。

默认： `'scrapy.spidermiddlewares.referer.DefaultReferrerPolicy'`

在填充请求“Referer”标头时应用的Referrer策略。

注意

您还可以使用特殊的 `"referrer_policy"` Request.meta键为每个请求设置Referrer Policy ，其 `REFERRER_POLICY` 设置的可接受值相同。

REFERRER_POLICY的可接受值

- 要么是 `scrapy.spidermiddlewares.referer.ReferrerPolicy` 子类的路径- 自定义策略，要么是内置策略之一（参见下面的类），
- 或者标准的W3C定义的字符串值之一，
- 还是特别的 `"scrapy-default"` 。

字符串值	类名（作为字符串）
`"scrapy-default"` （默认）	`scrapy.spidermiddlewares.referer.DefaultReferrerPolicy`
“无引荐”	`scrapy.spidermiddlewares.referer.NoReferrerPolicy`
“无引荐，当降级”	`scrapy.spidermiddlewares.referer.NoReferrerWhenDowngradePolicy`
“同源”	`scrapy.spidermiddlewares.referer.SameOriginPolicy`
“起源”	`scrapy.spidermiddlewares.referer.OriginPolicy`
“严格的原产地”	`scrapy.spidermiddlewares.referer.StrictOriginPolicy`
“源 - 当交起源”	`scrapy.spidermiddlewares.referer.OriginWhenCrossOriginPolicy`
“严格的原产地，当交起源”	`scrapy.spidermiddlewares.referer.StrictOriginWhenCrossOriginPoli`
“不安全的URL”	`scrapy.spidermiddlewares.referer.UnsafeUrlPolicy`

类 **scrapy.spidermiddlewares.referer.DefaultReferrerPolicy**

“no-referrer-when-downgrade”的变体，如果父请求正在使用 `file://` 或者 `s3://` 计划，则不会发送“Referer” 。

警告

Scrapy的默认引荐策略 - 就像“no-referrer-when-downgrade”，W3C推荐的浏览器值 - 将从 `http(s)://` 任何 `https://` URL 向任何URL 发送非空的“Referer”头，即使域不同。

如果要删除跨域请求的引用者信息，“同源”可能是更好的选择。

类 **scrapy.spidermiddlewares.referer.NoReferrerPolicy**

https://www.w3.org/TR/referrer-policy/#referrer-policy-no-referrer

最简单的策略是“no-referrer”，它指定不将引用者信息与从特定请求客户端发出的请求一起发送到任何源。标题将完全省略。

类 scrapy.spidermiddlewares.referer.NoReferrerWhenDowngradePolicy

https://www.w3.org/TR/referrer-policy/#referrer-policy-no-referrer-when-downgrade

“no-referrer-when-downgrade”策略将完整URL以及来自受TLS保护的环境设置对象的请求发送到可能值得信赖的URL，以及来自未受TLS保护的客户端的请求。

另一方面，从受TLS保护的客户端到非潜在可信赖URL的请求将不包含引用者信息。不会发送Referer HTTP标头。

如果未指定任何策略，则这是用户代理的默认行为。

注意

“no-referrer-when-downgrade”策略是W3C推荐的默认值，并由主要Web浏览器使用。

但是，它不是Scrapy的默认引荐策略（请参阅参考资料 `DefaultReferrerPolicy` ）。

类 scrapy.spidermiddlewares.referer.SameOriginPolicy

https://www.w3.org/TR/referrer-policy/#referrer-policy-same-origin

“同源”策略指定在从特定请求客户端发出同源请求时，将作为引用者剥离的完整URL作为引用者信息发送。

另一方面，跨源请求不包含引用者信息。不会发送Referer HTTP标头。

类 scrapy.spidermiddlewares.referer.OriginPolicy

https://www.w3.org/TR/referrer-policy/#referrer-policy-origin

“origin”策略指定在发出来自特定请求客户端的同源请求和跨源请求时，仅将请求客户端的源的ASCII序列化作为引用者信息发送。

类 scrapy.spidermiddlewares.referer.StrictOriginPolicy

https://www.w3.org/TR/referrer-policy/#referrer-policy-strict-origin

“严格来源”策略在发出请求时发送请求客户端源的ASCII序列化： - 从受TLS保护的环境设置对象到可能值得信赖的URL，以及 - 从非TLS保护的环境设置对象到任何起源。

另一方面，来自受TLS保护的请求客户端对不可信赖的URL的请求将不包含引用者信息。不会发送Referer HTTP标头。

类 scrapy.spidermiddlewares.referer.OriginWhenCrossOriginPolicy

https://www.w3.org/TR/referrer-policy/#referrer-policy-origin-when-cross-origin

“origin-when-cross-origin”策略指定在从特定请求客户端发出同源请求时，将作为引用者剥离的完整URL作为引用者信息发送，并且仅对源的ASCII序列化发送当从特定请求客户端发出跨源请求时，请求客户端作为引用者信息被发送。

类 scrapy.spidermiddlewares.referer.StrictOriginWhenCrossOriginPolicy

https://www.w3.org/TR/referrer-policy/#referrer-policy-strict-origin-when-cross-origin

“strict-origin-when-cross-origin”策略指定在从特定请求客户端发出同源请求时，将作为引用者剥离的完整URL作为引用者信息发送，并且只有ASCII序列化进行跨源请求时请求客户端的来源：

- 从受TLS保护的环境设置对象到可能值得信赖的URL，以及
- 从非TLS保护的环境设置对象到任何来源。

另一方面，来自受TLS保护的客户端对不可信赖的URL的请求将不包含引用者信息。不会发送Referer HTTP标头。

类 scrapy.spidermiddlewares.referer.UnsafeUrlPolicy

https://www.w3.org/TR/referrer-policy/#referrer-policy-unsafe-url

“unsafe-url”策略指定剥离用作引用者的完整URL以及来自特定请求客户端的跨源请求和同源请求。

注意：政策名称不是谎言; 这是不安全的。此政策将泄露从受TLS保护的资源到不安全来源的起源和路径。仔细考虑为可能敏感的文档设置此类策略的影响。

警告

不建议使用“unsafe-url”政策。

UrlLengthMiddleware

类 scrapy.spidermiddlewares.urllength.UrlLengthMiddleware

使用长于URLLENGTH_LIMIT的网址过滤掉请求

该 `UrlLengthMiddleware` 可通过以下设置进行配置（详情参见设置文档）：

- `URLLENGTH_LIMIT` - 允许抓取的URL的最大URL长度。