记录

注意

`scrapy.log`已被弃用于其函数，支持显式调用Python标准日志记录。继续阅读以了解有关新记录系统的更多信息。

Scrapy使用Python的内置日志记录系统进行事件记录。我们将提供一些简单的示例来帮助您入门，但对于更高级的用例，强烈建议您仔细阅读其文档。

日志记录开箱即用，并且可以在某种程度上使用记录设置中列出的Scrapy设置进行配置。

Scrapy调用`scrapy.utils.log.configure_logging()`设置一些合理的默认值并在运行命令时在日志记录设置中处理这些设置，因此如果您从脚本中运行Scrapy中所述的脚本运行Scrapy，建议手动调用它。

日志级别

Python的内置日志记录定义了5个不同的级别来指示给定日志消息的严重性。以下是标准的，按降序排列：

1. `logging.CRITICAL` - 严重错误（严重程度最高）
2. `logging.ERROR` - 经常出错
3. `logging.WARNING` - 用于警告信息
4. `logging.INFO` - 用于提供信息
5. `logging.DEBUG` - 用于调试消息（最低严重性）

如何记录消息

以下是如何使用`logging.WARNING`级别记录消息的快速示例：

```
import logging
logging.warning("This is a warning")
```

在任何标准5级别上都有用于发出日志消息的快捷方式，并且还有`logging.log`一种将给定级别作为参数的通用方法。如果需要，最后一个示例可以重写为：

```
import logging
logging.log(logging.WARNING, "This is a warning")
```

最重要的是，您可以创建不同的“记录器”来封装消息。（例如，通常的做法是为每个模块创建不同的记录器）。这些记录器可以独立配置，并允许分层构造。

前面的示例在幕后使用根记录器，这是一个顶级记录器，所有消息都传播到该记录器（除非另有说明）。使用 `logging` 帮助程序只是显式获取根记录器的快捷方式，因此这也是最后一个代码段的等效内容：

```
import logging
logger = logging.getLogger()
logger.warning("This is a warning")
```

您只需使用 `logging.getLogger` 函数获取其名称即可使用其他记录器 ：

```
import logging
logger = logging.getLogger('mycustomlogger')
logger.warning("This is a warning")
```

最后，您可以使用 `__name__` 变量填充当前模块的路径，确保为您正在处理的任何模块设置自定义记录器：

```
import logging
logger = logging.getLogger(__name__)
logger.warning("This is a warning")
```

也可以看看

模块记录，HowTo

基本日志教程

模块记录，记录器

有关记录器的更多文档

从蜘蛛记录

Scrapy `logger` 在每个Spider实例中提供了一个，可以像这样访问和使用：

```
import scrapy

class MySpider(scrapy.Spider):

    name = 'myspider'
    start_urls = ['https://scrapinghub.com']

    def parse(self, response):
        self.logger.info('Parse function called on %s', response.url)
```

该记录器是使用Spider的名称创建的，但您可以使用任何您想要的自定义Python记录器。例如：

```
import logging
import scrapy

logger = logging.getLogger('mycustomlogger')

class MySpider(scrapy.Spider):

    name = 'myspider'
    start_urls = ['https://scrapinghub.com']

    def parse(self, response):
        logger.info('Parse function called on %s', response.url)
```

记录配置

记录器本身不管理如何显示通过它们发送的消息。对于此任务，可以将不同的“处理程序”附加到任何记录器实例，并将这些消息重定向到适当的目标，例如标准输出，文件，电子邮件等。

默认情况下，Scrapy根据以下设置为根记录器设置和配置处理程序。

记录设置

这些设置可用于配置日志记录：

- `LOG_FILE`
- `LOG_ENABLED`
- `LOG_ENCODING`
- `LOG_LEVEL`
- `LOG_FORMAT`
- `LOG_DATEFORMAT`
- `LOG_STDOUT`
- `LOG_SHORT_NAMES`

前几个设置定义了日志消息的目标。如果 `LOG_FILE` 设置，则通过根记录器发送的消息将重定向到 `LOG_FILE` 以encoding编号 命名的文件 `LOG_ENCODING` 。如果未设置且 `LOG_ENABLED` 是 `True` ，则会在标准错误上显示日志消息。最后，如果 `LOG_ENABLED` 是 `False` ，则不会有任何可见的日志输出。

`LOG_LEVEL` 确定要显示的最低严重性级别，将过滤掉那些严重性较低的消息。它的范围是日志级别中列出的可能级别。

`LOG_FORMAT` 并 `LOG_DATEFORMAT` 指定用作所有消息的布局的格式字符串。这些字符串可以包含日志记录的logrecord属性docs和 datetime的strftime和strptime指令中 分别列出的任何占位符。

如果 `LOG_SHORT_NAMES` 设置，则日志不会显示打印日志的scrapy组件。默认情况下未设置，因此日志包含负责该日志输出的scrapy组件。

命令行选项

有可用于所有命令的命令行参数，可用于覆盖有关日志记录的某些Scrapy设置。

- `--logfile FILE`

  覆盖 `LOG_FILE`

- `--loglevel/-L LEVEL`

  覆盖 `LOG_LEVEL`

- `--nolog`

  设置 `LOG_ENABLED` 为 `False`

也可以看看

模块logging.handlers

有关可用处理程序的进一步文档

高级定制

由于Scrapy使用stdlib日志记录模块，因此您可以使用stdlib日志记录的所有功能自定义日志记录。

例如，假设您正在抓取一个返回许多HTTP 404和500响应的网站，并且您希望隐藏所有这样的消息：

```
2016-12-16 22:00:06 [scrapy.spidermiddlewares.httperror] INFO: Ignoring
response <500 http://quotes.toscrape.com/page/1-34/>: HTTP status code
is not handled or not allowed
```

首先要注意的是记录器名称 - 它在括号中：`[scrapy.spidermiddlewares.httperror]`。如果你得到的`[scrapy]`话`LOG_SHORT_NAMES`可能会设置为True; 将其设置为False并重新运行爬网。

接下来，我们可以看到消息具有INFO级别。要隐藏它，我们应该将日志级别设置为`scrapy.spidermiddlewares.httperror`高于INFO; INFO之后的下一级是警告。它可以用蜘蛛的`__init__`方法完成：

```
import logging
import scrapy


class MySpider(scrapy.Spider):
    # ...
    def __init__(self, *args, **kwargs):
        logger = logging.getLogger('scrapy.spidermiddlewares.httperror')
        logger.setLevel(logging.WARNING)
        super().__init__(*args, **kwargs)
```

如果再次运行此蜘蛛，则`scrapy.spidermiddlewares.httperror`记录器中的INFO消息 将消失。

scrapy.utils.log模块

scrapy.utils.log.configure_logging (*settings = None*，*install_root_handler = True*)

初始化Scrapy的日志记录默认值。

参数：

- **settings**（dict，`Settings` object或`None`） - 用于为根记录器创建和配置处理程序的设置（默认值：None）。
- **install_root_handler**（*bool*） - 是否安装root日志记录处理程序（默认值：True）

这个功能可以：

- 通过Python标准日志记录路由警告和扭曲日志记录
- 分别为Scrapy和Twisted记录器分配DEBUG和ERROR级别
- 如果LOG_STDOUT设置为True，则将stdout路由到日志

如果`install_root_handler`为True（默认），则此函数还会根据给定的设置为根记录器创建处理程序（请参阅记录设置）。您可以使用`settings`参数覆盖默认选项。如果`settings`为空或无，则使用默认值。

`configure_logging`在使用Scrapy命令时会自动调用，但在运行自定义脚本时需要显式调用。在这种情况下，不需要使用它，但建议使用它。

如果您打算自己配置处理程序，仍然建议您调用此函数，并传递*install_root_handler = False*。请记住，在这种情况下默认情况下不会设置任何日志输出。

要开始手动配置日志记录的输出，可以使用 logging.basicConfig（ ）来设置基本的根处理程序。这是关于如何将 `INFO` 消息重定向或更高的消息到文件的示例：

```
import logging
from scrapy.utils.log import configure_logging

configure_logging(install_root_handler=False)
logging.basicConfig(
    filename='log.txt',
    format='%(levelname)s: %(message)s',
    level=logging.INFO
)
```

有关以这种方式使用Scrapy的更多详细信息，请参阅脚本中的Run Scrapy。