调试蜘蛛

本文档介绍了调试蜘蛛的最常用技术。考虑下面的scrapy蜘蛛：

```
import scrapy
from myproject.items import MyItem

class MySpider(scrapy.Spider):
    name = 'myspider'
    start_urls = (
        'http://example.com/page1',
        'http://example.com/page2',
        )

    def parse(self, response):
        # collect `item_urls`
        for item_url in item_urls:
            yield scrapy.Request(item_url, self.parse_item)

    def parse_item(self, response):
        item = MyItem()
        # populate `item` fields
        # and extract item_details_url
        yield scrapy.Request(item_details_url, self.parse_details, meta={'item': item})

    def parse_details(self, response):
        item = response.meta['item']
        # populate more `item` fields
        return item
```

基本上这是一个简单的蜘蛛，它解析两页的项目（start_urls）。项目还具有与附加信息的详细信息页面，所以我们使用 `meta` 的功能， `Request` 通过一个部分填充的项目。

解析命令

检查蜘蛛输出的最基本方法是使用该 `parse` 命令。它允许在方法级别检查蜘蛛的不同部分的行为。它具有灵活且易于使用的优点，但不允许在方法内调试代码。

要查看从特定网址抓取的项目，请执行以下操作：

```
$ scrapy parse --spider=myspider -c parse_item -d 2 <item_url>
[ ... scrapy log lines crawling example.com spider ... ]

>>> STATUS DEPTH LEVEL 2 <<<
# Scraped Items  ------------------------------------------------------------
[{'url': <item_url>}]

# Requests  -----------------------------------------------------------------
[]
```

使用 `--verbose` 或 `-v` 选项我们可以看到每个深度级别的状态：

```
$ scrapy parse --spider=myspider -c parse_item -d 2 -v <item_url>
[ ... scrapy log lines crawling example.com spider ... ]

>>> DEPTH LEVEL: 1 <<<
# Scraped Items  ------------------------------------------------------------
[]

# Requests  -----------------------------------------------------------------
[<GET item_details_url>]


>>> DEPTH LEVEL: 2 <<<
# Scraped Items  ------------------------------------------------------------
[{'url': <item_url>}]

# Requests  -----------------------------------------------------------------
[]
```

检查从单个start_url中删除的项目，也可以使用以下方法轻松实现：

```
$ scrapy parse --spider=myspider -d 3 'http://example.com/page1'
```

Scrapy壳牌

虽然该 `parse` 命令对于检查蜘蛛的行为非常有用，但除了显示收到的响应和输出之外，检查回调内部发生的事情也没什么帮助。如何在 `parse_details` 有时没有收到项目时调试情况 ？

幸运的 `shell` 是，在这种情况下，这是你的面包和黄油（请参阅 从蜘蛛调用shell来检查响应 ）：

```
from scrapy.shell import inspect_response

def parse_details(self, response):
    item = response.meta.get('item', None)
    if item:
        # populate more `item` fields
        return item
    else:
        inspect_response(response, self)
```

另请参阅：从spiders调用shell以检查响应。

在浏览器中打开

有时您只想查看某个响应在浏览器中的外观，您可以使用该 `open_in_browser` 功能。以下是如何使用它的示例：

```
from scrapy.utils.response import open_in_browser

def parse_details(self, response):
    if "item name" not in response.body:
        open_in_browser(response)
```

`open_in_browser`将打开一个浏览器，其中包含Scrapy在此时收到的响应，调整基本标签以便正确显示图像和样式。

记录

记录是获取有关蜘蛛运行的信息的另一个有用选项。虽然不方便，但它具有以下优点：如果日志将来再次需要，它们将在以后的所有运行中可用：

```
def parse_details(self, response):
    item = response.meta.get('item', None)
    if item:
        # populate more `item` fields
        return item
    else:
        self.logger.warning('No item received for %s', response.url)
```

有关更多信息，请查看“ 日志记录”部分。