Scrapy壳

Scrapy shell是一个交互式shell，您可以非常快速地尝试调试您的抓取代码，而无需运行蜘蛛。它用于测试数据提取代码，但您实际上可以使用它来测试任何类型的代码，因为它也是常规的Python shell。

shell用于测试XPath或CSS表达式，看看它们是如何工作的，以及它们从你试图抓取的网页中提取的数据。它允许您在编写蜘蛛时以交互方式测试表达式，而无需运行蜘蛛来测试每个更改。

一旦熟悉了Scrapy shell，您就会发现它是开发和调试蜘蛛的宝贵工具。

配置

如果安装了IPython，Scrapy shell将使用它（而不是标准的Python控制台）。该IPython的控制台功能更强大，并提供智能自动完成和彩色输出，等等。

我们强烈建议您安装IPython，特别是如果您正在使用Unix系统（IPython擅长）。有关 详细信息，请参阅IPython安装指南。

Scrapy也支持bpython，并且会尝试在IPython 不可用的地方使用它。

通过scrapy的设置，您可以配置为使用中的任何一个 `ipython` ， `bpython` 或标准的 `python` 外壳，安装不管是哪个。这是通过设置 `SCRAPY_PYTHON_SHELL` 环境变量来完成的; 或者在scrapy.cfg中定义它：

```
[settings]
shell = bpython
```

启动

要启动Scrapy shell，您可以使用如下 `shell` 命令：

```
scrapy shell <url>
```

在哪里 `<url>` 是您要抓取的URL。

`shell`也适用于本地文件。如果您想要使用网页的本地副本，这可能很方便。`shell`了解本地文件的以下语法：

```
# UNIX-style
scrapy shell ./path/to/file.html
scrapy shell ../other/path/to/file.html
scrapy shell /absolute/path/to/file.html

# File URI
scrapy shell file:///absolute/path/to/file.html
```

注意

使用相对文件路径时，请明确并在其前面添加`./`（或`../`在相关时）。 不会像人们预期的那样工作（这是设计，而不是错误）。`scrapy shell index.html`

因为`shell`优先于文件URI上的HTTP URL，并且`index.html`在语法上类似于`example.com`，所以`shell`将`index.html`视为域名并触发DNS查找错误：

```
$ scrapy shell index.html
[ ... scrapy shell starts ... ]
[ ... traceback ... ]
twisted.internet.error.DNSLookupError: DNS lookup failed:
address 'index.html' not found: [Errno -5] No address associated with hostname.
```

`shell`如果`index.html`当前目录中存在调用的文件，则不会事先测试。再次，明确。

使用

Scrapy shell只是一个常规的Python控制台（如果有的话，它可以是IPython控制台），它提供了一些额外的快捷功能以方便使用。

可用的快捷方式

- `shelp()` - 使用可用对象和快捷方式列表打印帮助
- `fetch(url[, redirect=True])` - 从给定的URL获取新响应并相应地更新所有相关对象。您可以选择要求HTTP 3xx重定向，然后不要传递`redirect=False`
- `fetch(request)` - 从给定请求中获取新响应并相应地更新所有相关对象。
- `view(response)` - 在本地Web浏览器中打开给定的响应，以进行检查。这将向响应主体添加<base>标记，以便正确显示外部链接（如图像和样式表）。但请注意，这将在您的计算机中创建一个临时文件，该文件不会自动删除。

可用的Scrapy对象

Scrapy shell自动从下载的页面创建一些方便的对象，如 `Response` 对象和 `Selector` 对象（对于HTML和XML内容）。

那些对象是：

- `crawler` - 当前 `Crawler` 对象。
- `spider` - 已知处理URL的Spider，或者 `Spider` 当前URL没有找到蜘蛛时的 对象
- `request` - `Request` 最后一个获取页面的对象。您可以 `replace()` 使用 `fetch` 快捷方式使用或获取新请求（不离开shell）来修改此请求。
- `response` - `Response` 包含最后一个提取页面的对象
- `settings` - 目前的Scrapy设置

shell会话的例子

下面是一个典型shell会话的示例，我们首先抓取https://scrapy.org页面，然后继续刮取 https://reddit.com 页面。最后，我们将（Reddit）请求方法修改为POST并重新获取它以获得错误。我们通过在Windows中键入Ctrl-D（在Unix系统中）或Ctrl-Z来结束会话。

请记住，此处提取的数据在您尝试时可能不一样，因为这些页面不是静态的，并且在您测试时可能已更改。此示例的唯一目的是让您熟悉Scrapy shell的工作原理。

首先，我们启动shell：

```
scrapy shell 'https://scrapy.org' --nolog
```

然后，shell获取URL（使用Scrapy下载程序）并打印可用对象列表和有用的快捷方式（您会注意到这些行都以 `[s]` 前缀开头）：

```
[s] Available Scrapy objects:
[s]   scrapy     scrapy module (contains scrapy.Request, scrapy.Selector, etc)
[s]   crawler    <scrapy.crawler.Crawler object at 0x7f07395dd690>
[s]   item       {}
[s]   request    <GET https://scrapy.org>
[s]   response   <200 https://scrapy.org/>
[s]   settings   <scrapy.settings.Settings object at 0x7f07395dd710>
[s]   spider     <DefaultSpider 'default' at 0x7f0735891690>
[s] Useful shortcuts:
[s]   fetch(url[, redirect=True]) Fetch URL and update local objects (by default, redirects are
followed)
[s]   fetch(req)                  Fetch a scrapy.Request and update local objects
[s]   shelp()           Shell help (print this help)
[s]   view(response)    View response in a browser

>>>
```

之后，我们可以开始玩对象了：

```
>>> response.xpath('//title/text()').extract_first()
'Scrapy | A Fast and Powerful Scraping and Web Crawling Framework'

>>> fetch("https://reddit.com")

>>> response.xpath('//title/text()').extract()
['reddit: the front page of the internet']

>>> request = request.replace(method="POST")

>>> fetch(request)

>>> response.status
404

>>> from pprint import pprint

>>> pprint(response.headers)
{'Accept-Ranges': ['bytes'],
 'Cache-Control': ['max-age=0, must-revalidate'],
 'Content-Type': ['text/html; charset=UTF-8'],
 'Date': ['Thu, 08 Dec 2016 16:21:19 GMT'],
 'Server': ['snooserv'],
 'Set-Cookie': ['loid=KqNLou0V9SKMX4qb4n; Domain=reddit.com; Max-Age=63071999; Path=/;
expires=Sat, 08-Dec-2018 16:21:19 GMT; secure',
                'loidcreated=2016-12-08T16%3A21%3A19.445Z; Domain=reddit.com; Max-Age=63071999;
Path=/; expires=Sat, 08-Dec-2018 16:21:19 GMT; secure',
                'loid=vi0ZVe4NkxNWdlH7r7; Domain=reddit.com; Max-Age=63071999; Path=/;
expires=Sat, 08-Dec-2018 16:21:19 GMT; secure',
                'loidcreated=2016-12-08T16%3A21%3A19.459Z; Domain=reddit.com; Max-Age=63071999;
Path=/; expires=Sat, 08-Dec-2018 16:21:19 GMT; secure'],
 'Vary': ['accept-encoding'],
 'Via': ['1.1 varnish'],
 'X-Cache': ['MISS'],
 'X-Cache-Hits': ['0'],
 'X-Content-Type-Options': ['nosniff'],
 'X-Frame-Options': ['SAMEORIGIN'],
 'X-Moose': ['majestic'],
 'X-Served-By': ['cache-cdg8730-CDG'],
 'X-Timer': ['S1481214079.394283,VS0,VE159'],
 'X-Ua-Compatible': ['IE=edge'],
 'X-Xss-Protection': ['1; mode=block']}
>>>
```

从蜘蛛调用shell来检查响应

有时您想要检查蜘蛛的某个特定点正在处理的响应，如果只是为了检查您期望的响应是否到达那里。

这可以通过使用该 `scrapy.shell.inspect_response` 功能来实现。

这是一个如何从蜘蛛中调用它的示例：

```
import scrapy


class MySpider(scrapy.Spider):
    name = "myspider"
    start_urls = [
        "http://example.com",
        "http://example.org",
        "http://example.net",
    ]

    def parse(self, response):
        # We want to inspect one specific response.
        if ".org" in response.url:
            from scrapy.shell import inspect_response
            inspect_response(response, self)

        # Rest of parsing code.
```

当你运行蜘蛛时，你会得到类似的东西：

```
2014-01-23 17:48:31-0400 [scrapy.core.engine] DEBUG: Crawled (200) <GET http://example.com>
(referer: None)
2014-01-23 17:48:31-0400 [scrapy.core.engine] DEBUG: Crawled (200) <GET http://example.org>
(referer: None)
[s] Available Scrapy objects:
[s]   crawler    <scrapy.crawler.Crawler object at 0x1e16b50>
...

>>> response.url
'http://example.org'
```

然后，您可以检查提取代码是否正常工作：

```
>>> response.xpath('//h1[@class="fn"]')
[]
```

不，它没有。因此，您可以在Web浏览器中打开响应，看看它是否是您期望的响应：

```
>>> view(response)
True
```

最后，按Ctrl-D（或Windows中的Ctrl-Z）退出shell并继续爬行：

```
>>> ^D
2014-01-23 17:50:03-0400 [scrapy.core.engine] DEBUG: Crawled (200) <GET http://example.net>
(referer: None)
...
```

请注意， `fetch` 由于Scrapy引擎被shell阻止，因此您无法使用此处的快捷方式。但是，在您离开外壳后，蜘蛛将继续在停止的位置爬行，如上所示。