通用实践

本节介绍使用Scrapy时的常见做法。这些内容涉及许多主题，并且通常不属于任何其他特定部分。

从脚本运行Scrapy

您可以使用API从脚本运行Scrapy，而不是运行Scrapy的典型方法。`scrapy crawl`

请记住，Scrapy是基于Twisted异步网络库构建的，因此您需要在Twisted reactor中运行它。

您可以用来运行蜘蛛的第一个实用程序是`scrapy.crawler.CrawlerProcess`。该类将为您启动Twisted reactor，配置日志记录并设置关闭处理程序。此类是所有Scrapy命令使用的类。

这是一个示例，展示如何使用它运行单个蜘蛛。

```
import scrapy
from scrapy.crawler import CrawlerProcess

class MySpider(scrapy.Spider):
    # Your spider definition
    ...

process = CrawlerProcess({
    'USER_AGENT': 'Mozilla/4.0 (compatible; MSIE 7.0; Windows NT 5.1)'
})

process.crawl(MySpider)
process.start() # the script will block here until the crawling is finished
```

请务必查看`CrawlerProcess`文档以熟悉其使用详细信息。

如果您在Scrapy项目中，则可以使用一些其他帮助程序在项目中导入这些组件。您可以自动导入传递其名称的蜘蛛`CrawlerProcess`，并用于`get_project_settings`获取`Settings`项目设置的实例。

以下是使用testspiders 项目作为示例的如何执行此操作的工作示例。

```
from scrapy.crawler import CrawlerProcess
from scrapy.utils.project import get_project_settings

process = CrawlerProcess(get_project_settings())

# 'followall' is the name of one of the spiders of the project.
process.crawl('followall', domain='scrapinghub.com')
process.start() # the script will block here until the crawling is finished
```

还有另一个Scrapy实用程序可以更好地控制爬网过程：`scrapy.crawler.CrawlerRunner`。此类是一个瘦包装程序，它封装了一些简单的帮助程序来运行多个爬网程序，但它不会以任何方式启动或干扰现有的反应堆。

使用此类，应在安排蜘蛛后显式运行reactor。如果您的应用程序已经在使用Twisted并且您想在同一个反应器中运行Scrapy，建议您使用`CrawlerRunner`而不是`CrawlerProcess`。

请注意，在蜘蛛完成后您还必须自己关闭Twisted反应器。这可以通过向`CrawlerRunner.crawl`方法返回的延迟添加回调来实现。

以下是其使用示例，以及在*MySpider*完成运行后手动停止反应器的回调。

```
from twisted.internet import reactor
import scrapy
from scrapy.crawler import CrawlerRunner
from scrapy.utils.log import configure_logging

class MySpider(scrapy.Spider):
    # Your spider definition
    ...

configure_logging({'LOG_FORMAT': '%(levelname)s: %(message)s'})
runner = CrawlerRunner()

d = runner.crawl(MySpider)
d.addBoth(lambda _: reactor.stop())
reactor.run() # the script will block here until the crawling is finished
```

也可以看看

扭曲反应堆概述。

在同一个进程中运行多个蜘蛛

默认情况下，Scrapy在您运行时为每个进程运行一个蜘蛛。但是，Scrapy支持使用内部API为每个进程运行多个蜘蛛。`scrapy crawl`

这是一个同时运行多个蜘蛛的示例：

```
import scrapy
from scrapy.crawler import CrawlerProcess

class MySpider1(scrapy.Spider):
    # Your first spider definition
    ...

class MySpider2(scrapy.Spider):
    # Your second spider definition
    ...

process = CrawlerProcess()
process.crawl(MySpider1)
process.crawl(MySpider2)
process.start() # the script will block here until all crawling jobs are finished
```

使用相同的示例`CrawlerRunner`：

```
import scrapy
from twisted.internet import reactor
from scrapy.crawler import CrawlerRunner
from scrapy.utils.log import configure_logging

class MySpider1(scrapy.Spider):
    # Your first spider definition
    ...

class MySpider2(scrapy.Spider):
    # Your second spider definition
    ...

configure_logging()
runner = CrawlerRunner()
runner.crawl(MySpider1)
runner.crawl(MySpider2)
d = runner.join()
d.addBoth(lambda _: reactor.stop())

reactor.run() # the script will block here until all crawling jobs are finished
```

相同的例子，但通过链接延迟顺序运行蜘蛛：

```
from twisted.internet import reactor, defer
from scrapy.crawler import CrawlerRunner
from scrapy.utils.log import configure_logging

class MySpider1(scrapy.Spider):
    # Your first spider definition
    ...

class MySpider2(scrapy.Spider):
    # Your second spider definition
    ...

configure_logging()
runner = CrawlerRunner()

@defer.inlineCallbacks
def crawl():
    yield runner.crawl(MySpider1)
    yield runner.crawl(MySpider2)
    reactor.stop()

crawl()
reactor.run() # the script will block here until the last crawl call is finished
```

也可以看看

从脚本运行Scrapy。

分布式抓取

Scrapy不提供以分布式（多服务器）方式运行爬网的任何内置工具。但是，有一些方法可以分发爬网，这取决于您计划如何分发爬网。

如果你有很多蜘蛛，分配负载的明显方法是设置许多Scrapyd实例并在其中分配蜘蛛运行。

如果你想通过许多机器运行单个（大）蜘蛛，你通常做的是分区网址以便抓取并将它们发送到每个单独的蜘蛛。这是一个具体的例子：

首先，准备要抓取的网址列表，并将它们放入单独的文件/网址中：

```
http://somedomain.com/urls-to-crawl/spider1/part1.list
http://somedomain.com/urls-to-crawl/spider1/part2.list
http://somedomain.com/urls-to-crawl/spider1/part3.list
```

然后你在3个不同的Scrapyd服务器上发射蜘蛛。蜘蛛会收到一个（蜘蛛）参数 `part` ，其中包含要爬网的分区的编号：

```
curl http://scrapy1.mycompany.com:6800/schedule.json -d project=myproject -d spider=spider1 -d
part=1
curl http://scrapy2.mycompany.com:6800/schedule.json -d project=myproject -d spider=spider1 -d
part=2
curl http://scrapy3.mycompany.com:6800/schedule.json -d project=myproject -d spider=spider1 -d
part=3
```

避免被禁止

一些网站实施某些措施来防止机器人抓取它们，具有不同程度的复杂性。绕过这些措施可能既困难又棘手，有时可能需要特殊的基础设施。如有疑问，请考虑联系商业支持。

以下是处理这些类型网站时要记住的一些提示：

- 从浏览器中的众所周知的池中轮换您的用户代理（谷歌周围获取它们的列表）
- 禁用cookie（请参阅参考资料 `COOKIES_ENABLED` ），因为有些网站可能会使用cookie来发现僵尸程
- 使用下载延迟（2或更高）。见 `DOWNLOAD_DELAY` 设定。
- 如果可能，请使用Google缓存来抓取网页，而不是直接点击网站

- 使用一组旋转IP。例如，免费的Tor项目或像ProxyMesh这样的付费服务。一个开源替代品是scrapoxy，一个超级代理，你可以附加自己的代理。
- 使用一个高度分布的下载器，可以在内部绕过禁令，因此您可以专注于解析干净的页面。这种下载器的一个例子是 Crawlera

如果您仍无法阻止机器人被禁止，请考虑联系 商业支持。