Scrapy

目錄

1. 介紹
2. 从这儿开始
    i. Scrapy 概览
    ii. 安装
    iii. Scrapy 初步
    iv. 示例
3. 基本概念
    i. 命令
    ii. 蜘蛛
    iii. 选择器
    iv. Shell
    v. Items
    vi. Item Loaders
    vii. 管道
    viii. Feed Exports
    ix. 请求 和 响应
    x. 链接提取器
    xi. 配置
    xii. 异常

scrapy-chinese

这是 scrapy 官方文档的中文翻译。我们希望本文档一直跟随 scrapy的每次发布而更新，为国内同行提供便利。然而一个人的精力总是有限的，如果你也在用 scrapy，并愿意抽空给我们翻译一段，我们将非常高兴！

概要

Scrapy 是一个快速、高级的 web 抓取和提取框架，用于抓取网站、从页面中提取结构化数据。它的用途非常广泛，从数据挖掘到监控和自动化测试。

关于 scrapy 的更多特性，请移步到主页http://scrapy.org.

依赖

- Python2.7
- 在Linux, Windows, Mac OSX, BSD平台可用

安装

快速方法：

```
pip install scrapy
```

更多关于安装的信息，请看 安装

版本发布

你可以从这里下载最新的稳定和开发版本。

文档

官方在线文档: 官方文档

建议：官方文档只有英文版的，本文档是最新的中文翻译。

社区

社区链接

贡献

贡献你的力量

Scrapy 一瞥

scrapy 是一个用于抓取网站、提取结构化数据的应用框架。广泛应用于数据挖掘、信息处理和历史归档等应用中。

虽然原本本设计用来做数据抓取，它也可以用来通过 API 提取数据(例如 Amazon Associates Web Services) 或者一般目的的爬虫。

来看一个简单爬虫

为了让你看看 Scrapy 能干什么，我们通过一个能运行的简单蜘蛛来了解一下。

下面是一个蜘蛛的代码，它的作用是跟踪StackOverflow上最热门的问题的链接，并从每个页面中提取一些数据：

```
import scrapy


class StackOverflowSpider(scrapy.Spider):
    name = 'stackoverflow'
    start_urls = ['http://stackoverflow.com/questions?sort=votes']

    def parse(self, response):
        for href in response.css('.question-summary h3 a::attr(href)'):
            full_url = response.urljoin(href.extract())
            yield scrapy.Request(full_url, callback=self.parse_question)

    def parse_question(self, response):
        yield {
            'title': response.css('h1 a::text').extract()[0],
            'votes': response.css('.question .vote-count-post::text').extract()[0],
            'body': response.css('.question .post-text').extract()[0],
            'tags': response.css('.question .post-tag::text').extract(),
            'link': response.url,
        }
```

把这段代码保存到 `stackoverflow_spider.py` 中，用 `runspider` 命令运行一下：

```
scrapy runspider stackoverflow_spider.py -o top-stackoverflow-questions.json
```

运行结束时会生成 `top-stackoverflow-questions.json` 文件。以 json 格式保存了StackOverflow上最热门的问题，一个 tags 列表和 HTML 格式的问题内容，就是这个（为了方便阅读整理了格式）：

```
[{
    "body": "... LONG HTML HERE ...",
    "link": "http://stackoverflow.com/questions/11227809/why-is-processing-a-sorted-array-faster-than-an-unsorted-array
    "tags": ["java", "c++", "performance", "optimization"],
    "title": "Why is processing a sorted array faster than an unsorted array?",
    "votes": "9924"
},
{
    "body": "... LONG HTML HERE ...",
    "link": "http://stackoverflow.com/questions/1260748/how-do-i-remove-a-git-submodule",
    "tags": ["git", "git-submodules"],
    "title": "How do I remove a Git submodule?",
    "votes": "1764"
},
...]
```

发生了什么？

当你运行命令 `scrapy runspider somefile.py` 的时候，Scrapy在文件中查找蜘蛛定义(Spider definition)并且通过爬虫引擎运行了它。

抓取从定义在 `start_urls` 属性中的 URL 生成请求开始，由默认的回调方法 `parse` 调用（传入响应对象）。在 `parse` 方法中，我们用 CSS 选择器从问题页中提取数据，接着我们又抛出几个待发送的请求，把 `parse_question` 作为这几个新请求的回调方法，直到每个请求都完成。

这时你注意到一个 scrapy 的主要优势：请求是异步地被调度和处理的。这意味着 Scrapy 不需要等一个请求返回并处理完成，它就可以同时发送另一个请求，或者做些其他事情。这也意味着即使一些请求失败了，或者处理过程中出错了，其他请求可以继续处理。

在给你快速抓取能力（以fault-tolerant方式可以并发发送多个请求）的同时，你可以通过一些设置控制它。你可以设置每个请求之间的下载延时，限制每个域或者 IP 的并发数，甚至通过auto-throttling扩展尝试自动限制并发数。

最后，`parse_question` 抓取问题数据从每一个页面，返回一个 dict，然后收集起来，写入命令行指定的文件中。

```
注意

这里使用了 `feed exports`来生成 JSON 文件，你也可以轻易地改变导出格式(XML 或者 CSV)，改变存储后端(FTP 或者 Amazon S3).你也可以写一个
```

还有什么？

你已经知道如何用 scrapy从网站 提取数据，把它们存储起来，但这还只是表面的东西。 SCrapy 提供了许多强大的功能，以使抓取变得容易、有效。例如：

- 通过扩展 CSS 选择器和 XPATH 表达式来选择和提取需要的数据和一些用正则表达式提取的帮助方法。
- 提供了一个交互式的 shell 控制台(支持 IPython)。测试 能提取数据的CSS 和 XPATH 表达式，当你在编码或者调试蜘蛛的时候非常有用。
- 内建支持多种格式的 `feed exports` (JSON, CSV, XML),可以保存在多种存储后端(FTP, S3, 本地文件系统)
- 广泛的编码支持和自动检测来处理外部的、非标准的、错误的编码声明。
- 强大的扩展支持。通过信号和良好定义的 API（中间件、扩展、管道）使你能够自由地配置自己编写的功能。
- 大量的内置扩展和中间件：
  - cookies and session 处理
  - HTTP 特性，如压缩，认证，缓存
  - user-agent欺骗
  - robots.txt
  - 限制抓取深度
  - 更多
- Telnet 控制台：可以在运行过程中进入 Python 控制台来内省、调试你的爬虫
- 还有很多好东西，像通过Sitemap和XML/CSV feeds抓取的可重用蜘蛛(reusable spiders), 自动下载图片或其他多媒体的多媒体管道(media pipeline),可以缓存的 DNS 解析器等。

接下来

下一步你可以通过安装 Scrapy, Scrapy 入门学习如何在 scrpay 项目中组织代码，你也可以加入社区。感谢你的阅读！