Scrapy介绍三

Scrapy介绍三

命令行工具介绍和示例

命令行工具

Scrapy是通过scrapy命令行工具来控制的，当前最新版本0.10

配置设置

Scrapy将scrapy.cfg在标准位置的ini样式文件中查找配置参数：

1. 系统默认配置：`/etc/scrapy.cfg`或`c:\scrapy\scrapy.cfg`
2. 全局配置文件：`~/.config/scrapy.cfg`（`$XDG_CONFIG_HOME`）和`~/.scrapy.cfg`（`$HOME`）用于
3. 项目配置文件：`scrapy.cfg`在scrapy项目的根目录中。

来自这些文件的设置将按照所列的优先顺序进行合并：用户定义的值比系统级默认值具有更高的优先级，**项目范围的设置将在定义时覆盖所有其他设置**。
Scrapy也理解，并且可以通过配置一些环境变量。目前这些是：
SCRAPY_SETTINGS_MODULE（请参阅指定设置）
SCRAPY_PROJECT
SCRAPY_PYTHON_SHELL（见Scrapy shell）

Scrapy项目的默认结构

在深入了解命令行工具及其子命令之前，让我们先了解Scrapy项目的目录结构。
虽然可以修改，但所有Scrapy项目默认情况下具有相同的文件结构，类似于：

```
scrapy.cfg
myproject/
    __init__.py
    items.py
    pipelines.py
    settings.py
    spiders/
        __init__.py
        spider1.py
        spider2.py
        ...
```

其中，目录scrapy.cfg文件位于项目的根目录。该文件包含定义项目设置的python模块的名称。这里是一个例子：

```
[settings]
default = myproject.settings
```

使用scrapy工具

您可以从运行没有参数的Scrapy工具开始，它将打印一些使用帮助和可用的命令：
直接在项目根目录的命令行输入： scrapy

```
会得到如下提示：
Scrapy 1.3.2 - no active project
Usage:
  scrapy <command> [options] [args]
Available commands:
  bench         Run quick benchmark test
  commands
  fetch         Fetch a URL using the Scrapy downloader
  genspider     Generate new spider using pre-defined templates
  runspider     Run a self-contained spider (without creating a project)
  settings      Get settings values
  shell         Interactive scraping console
  startproject  Create new project
  version       Print Scrapy version
  view          Open URL in browser, as seen by Scrapy
  [ more ]      More commands available when run from project directory
Use "scrapy <command> -h" to see more info about a command
```

创建项目

你通常用这个scrapy工具做的第一件事是创建你的Scrapy项目：
`scrapy startproject myproject [ project_dir ]`
这将在该project_dir目录下创建一个Scrapy项目。如果project_dir没有指定，project_dir将会和myproject名称一样。
接下来，进入新的项目目录：
`cd project_dir`
您可以使用scrapy命令从那里管理和控制您的项目。

控制项目

您可以使用scrapy项目内部的工具来控制和管理它们。

大家不要着急一下子把所以东西都介绍到，具体细节后面都会写到。

例如，要创建一个新的爬虫：
`scrapy genspider mydomain mydomain.com`

通过上述命令创建了一个spider name为mydomain的爬虫，start_urls为http://www.cnblogs.com/的爬虫。

一些Scrapy命令（如crawl）必须从Scrapy项目内部运行。请参阅命令参考下文中的哪些命令必须从内部项目运行的详细信息。

还要记住，一些命令在从项目中运行时可能有稍微不同的行为。例如，user_agent如果正在获取的URL与某个特定的爬虫相关联，fetch命令将使用爬虫覆盖的行为（例如属性覆盖用户代理）。这是有意的，因为该fetch命令用于检查爬虫程序如何下载页面。

常用的工具命令

此部分包含可用内置命令的列表，其中包含描述和一些用法示例。记住，您可以随时通过运行以下命令获得有关每个命令的更多信息：

```
scrapy <command> -h
```

你可以看到所有可用的命令：
`scrapy -h`

上面两种命令，它们只能在Scrapy项目内部工作，也可以全局命令的情况下工作（但它们可能会被项目内的配置覆盖）。

- 全局命令：
- [startproject]
- [genspider]
- [settings]
- [runspider]
- [shell]
- [fetch]
- [view]
- [version]
- 仅项目命令：
- [crawl]
- [check]
- [list]
- [edit]
- [parse]

- [bench]

startproject

语法：
`scrapy startproject <project_name> [project_dir]`

在目录project_name下创建一个名为的Scrapy项目project_dir 。如果project_dir没有指定，project_dir将会和myproject名称一样。

用法示例：
`$ scrapy startproject myproject`

genspider

语法：
`scrapy genspider [-t template] <name> <domain>`

在当前文件夹或当前项目的spiders文件夹中创建一个新的爬虫，如果从项目中调用。该<name>参数设置为爬虫的name，而<domain>用于生成allowed_domains和start_urls爬虫的属性。

用法示例：

```
$ scrapy genspider -l
Available templates:
  basic
  crawl
  csvfeed
  xmlfeed
$ scrapy genspider example example.com
Created spider 'example' using template 'basic'
$ scrapy genspider -t crawl scrapyorg scrapy.org
Created spider 'scrapyorg' using template 'crawl'
```

这只是一个方便的快捷命令，用于创建基于预定义模板的爬虫，但当然不是唯一的方式来创建爬虫。您可以自己创建爬虫源代码文件，而不是使用此命令。

crawl

语法：
`scrapy crawl <spider>`

使用爬虫开始爬行。

用法示例：

```
$ scrapy crawl myspider
[ ... myspider starts crawling ... ]
```

check

语法：
`scrapy check [-l] <spider>`

用法示例：

```
$ scrapy check -l
first_spider
  * parse
  * parse_item
second_spider
  * parse
  * parse_item
$ scrapy check
[FAILED] first_spider:parse_item
>>> 'RetailPricex' field is missing
[FAILED] first_spider:parse
>>> Returned 92 requests, expected 0..4
```

list

语法：
`scrapy list`

列出当前项目中的所有可用爬虫。每行输出一个爬虫。

用法示例：

```
$ scrapy列表
spider1
spider2
```

edit

语法：
`scrapy edit <spider>`

此命令仅作为最常见情况的方便快捷方式提供，开发人员当然可以选择任何工具或IDE来编写和调试他的爬虫。

用法示例：

```
$ scrapy edit spider1
```

fetch

语法：
`scrapy fetch <url>`

使用Scrapy下载器下载给定的URL，并将内容写入标准输出。

这个命令的有趣的事情是它获取爬虫下载它的页面。例如，如果爬虫有一个USER_AGENT 属性覆盖用户代理，它将使用那个。

所以这个命令可以用来“看”你的爬虫如何获取一个页面。

如果在项目外部使用，将不应用特定的每个爬虫行为，它将只使用默认的Scrapy下载器设置。

支持的选项：

--spider=SPIDER：绕过爬虫自动检测和强制使用特定的爬虫
--headers：打印响应的HTTP头，而不是响应的正文
--no-redirect：不遵循HTTP 3xx重定向（默认是遵循它们）

用法示例：

```
$ scrapy fetch --nolog http://www.example.com/some/page.html
[ ... html content here ... ]
$ scrapy fetch --nolog --headers http://www.example.com/
{'Accept-Ranges': ['bytes'],
 'Age': ['1263   '],
 'Connection': ['close     '],
 'Content-Length': ['596'],
 'Content-Type': ['text/html; charset=UTF-8'],
 'Date': ['Wed, 18 Aug 2010 23:59:46 GMT'],
 'Etag': ['"573c1-254-48c9c87349680"'],
 'Last-Modified': ['Fri, 30 Jul 2010 15:30:18 GMT'],
 'Server': ['Apache/2.2.3 (CentOS)']}
```

view

语法:
`scrapy view <url>`

在浏览器中打开给定的URL，因为您的Scrapy爬虫会“看到”它。有时，爬虫会看到与普通用户不同的网页，因此可以用来检查爬虫“看到了什么”并确认它是您期望的。

支持的选项:

--spider=SPIDER：绕过爬虫自动检测和强制使用特定的爬虫
--no-redirect：不遵循HTTP 3xx重定向（默认是遵循它们）
用法示例:

```
$ scrapy view http://www.example.com/some/page.html
[ ... browser starts ... ]
```

shell

语法:
`scrapy shell [url]`
启动给定URL（如果给定）的Scrapy shell，如果没有给出URL，则为空。还支持UNIX样式的本地文件路径，相对于 ./或../前缀或绝对文件路径。有关详细信息，请参阅Scrapy shell。

支持的选项:

--spider=SPIDER：绕过爬虫自动检测和强制使用特定的爬虫
-c code：评估shell中的代码，打印结果并退出
--no-redirect：不遵循HTTP 3xx重定向（默认是遵循它们）；这只影响你可以在命令行上作为参数传递的URL; 一旦你在shell中，fetch(url)默认情况下仍然会遵循HTTP重定向。
用法示例:

```
$ scrapy shell http://www.example.com/some/page.html
[ ... scrapy shell starts ... ]
$ scrapy shell --nolog http://www.example.com/ -c '(response.status,
response.url)'
(200, 'http://www.example.com/')
# shell follows HTTP redirects by default
$ scrapy shell --nolog http://httpbin.org/redirect-to?
url=http://example.com/ -c '(response.status, response.url)'
(200, 'http://example.com/')
# you can disable this with --no-redirect
# (only for the URL passed as command line argument)
```

```
$ scrapy shell --no-redirect --nolog http://httpbin.org/redirect-to?
url=http://example.com/ -c '(response.status, response.url)'
(302, 'http://httpbin.org/redirect-to?url=http://example.com/')
```

parse

语法：
`scrapy parse <url> [options]`

获取给定的URL并使用处理它的爬虫解析它，使用通过--callback选项传递的方法，或者parse如果没有给出。

支持的选项：

--spider=SPIDER：绕过爬虫自动检测和强制使用特定的爬虫
--a NAME=VALUE：set spider argument（可以重复）
--callback或者-c：spider方法用作回调来解析响应
--pipelines：通过管道处理项目
--rules或者-r：使用CrawlSpider 规则来发现用于解析响应的回调（即，spider方法）
--noitems：不显示已抓取的项目
--nolinks：不显示提取的链接
--nocolour：避免使用pygments来着色输出
--depth或-d：请求应递归跟踪的深度级别（默认值：1）
--verbose或-v：显示每个深度级别的信息
用法示例：

```
$ scrapy parse http://www.example.com/ -c parse_item
[ ... scrapy log lines crawling example.com spider ... ]
>>> STATUS DEPTH LEVEL 1 <<<
# Scraped Items  ------------------------------------------------------------
---
[{'name': u'Example item',
 'category': u'Furniture',
 'length': u'12 cm'}]
# Requests  -----------------------------------------------------------------
---
[]
```

settings

语法：
`scrapy settings [options]`

获取Scrapy设置的值。

如果在项目中使用，它将显示项目设置值，否则将显示该设置的默认Scrapy值。

用法示例：

```
$ scrapy settings --get BOT_NAME
scrapybot
$ scrapy settings --get DOWNLOAD_DELAY
0
```

runspider

语法：
`scrapy runspider <spider_file.py>`

运行一个自包含在Python文件中的爬虫，而不必创建一个项目。

用法示例：

```
$ scrapy runspider myspider.py
[...爬虫开始爬行...]
```

version

语法：
`scrapy version [-v]`

打印Scrapy版本。如果使用-v它也打印Python，Twisted和平台信息，这是有用的错误报告。

bench

新版本0.17。
语法：

运行快速基准测试。基准

自定义项目命令

您还可以使用COMMANDS_MODULE设置添加自定义项目命令 。有关如何实现命令的示例，请参阅scrapy/commands中的Scrapy命令。