Scrapy Tutorial #5: How To Create Simple Scrapy Spider

Last updated on Feb 25 2019 by Michael Yin

Introduction:

This is the #5 post of my Scrapy Tutorial Series, in this Scrapy tutorial, I will talk about how to create a Scrapy project and a Scrapy spider, in addition, I will show you how to use some basic scrapy commands.

You can get the source code of this project at the end of this tutorial.

Scrapy Commands

First, we can take a short view about the Scrapy commands and have an impression, and later we can learn more about them. Type `scrapy` in the terminal, below is the output.

```
$ scrapy
Scrapy 1.4.0 - no active project

Usage:
  scrapy <command> [options] [args]

Available commands:
  bench         Run quick benchmark test
  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
```

As you can see, here is a short list of scrapy commands, if you want to check the detail about any scrapy commands, just use `scrapy <command> -h`. In this tutorial, we whould use `startproject` and `genspider` to help us create project and spider file, and then I would introduce you how to use `shell` and `fetch` commands to test your code.

Create Simple Scrapy Project

Now we start to create a new scrapy project from scratch.

```
$ scrapy startproject scrapy_spider
```

Now a project named `scrapy_spider` has been created, we can follow the output to use `genspider` to generate one scrapy spider for us

```
You can start your first spider with:
    cd scrapy_spider
    scrapy genspider example example.com
```

Now you have a scrapy project which contains a spider named `example`. Let's take a look at the project directory.

```
├── scrapy.cfg                # deploy configuration file
└── scrapy_spider             # project's Python module, you'll import your code from he
    ├── __init__.py
    ├── items.py              # project items definition file
    ├── middlewares.py        # project middlewares file
    ├── pipelines.py          # project pipeline file
    ├── settings.py           # project settings file
    └── spiders               # a directory where spiders are located
        ├── __init__.py
        └── example.py        # spider we just created
```

Our first Scrapy spider

Now we start to create a Scrapy spider to scrap something for us. Our target website is Quotes to Scrape and we plan to crawl all the quotes of the page for us, but first, we need to create a new Scrapy spider.

```
# Usage: scrapy genspider [options] <name> <domain>

$ scrapy genspider quotes_spider quotes.toscrape.com
```

We just use `genspider` to create a spider, the parameters you passed in would be seen in the generated python file, let's see what is in the `scrapy_spider/spiders/quotes_spider.py`

```
# -*- coding: utf-8 -*-
import scrapy

class QuotesSpiderSpider(scrapy.Spider):
    name = 'quotes_spider'
    allowed_domains = ['quotes.toscrape.com']
    start_urls = ['http://quotes.toscrape.com/']

    def parse(self, response):
        pass
```

As you can see, the spider file has contains some parameters.

1. `name`: identifies the Spider. It must be unique within a project
2. `start_urls`: The list of the feed URLs, the spider would start by crawling the feed URLs.
3. `allowed_domains`: This setting is useful for broad crawls, if the domain of the URL is not in this setting, then the URL would be ignored.
4. `parse`: a method that will be called to handle the response downloaded for each of the requests made.

Now we start to write some code in `parse` to crawl something for us.

```
# -*- coding: utf-8 -*-
import scrapy

class QuotesSpiderSpider(scrapy.Spider):
    name = 'quotes_spider'
    allowed_domains = ['quotes.toscrape.com']
    start_urls = ['http://quotes.toscrape.com/']

    def parse(self, response):
        quotes = response.xpath("//div[@class='quote']//span[@class='text']/text()").ext
        yield {'quotes': quotes}
```

We use XPath expression to extract the data which will be talked about more in detail. After the spider is done, we can now run the spider to crawl the data.

```
$ scrapy crawl quotes_spider

# Then you can see the valid data in the terminal output
[scrapy.core.scraper] DEBUG: Scraped from <200 http://quotes.toscrape.com/>
```

Conclusion

In this scrapy tutorial, we successfully create a Scrapy project and a Scrapy spider using some Scrapy commands, and we have a spider which can crawl data for us now.

To help user focus on the key part, I only paste part of the source code instead of the whole file in this tutorial, If you want source code which can run in your local env directly, just

```
git clone git@github.com:michael-yin/scrapy-spider-example.git

cd scrapy-spider-example

git checkout ffc4cc3

# If you can star my project, I would appreciate that
```