BÁO CÁO MÔN HỌC CÁC VẤN ĐỀ HIỆN ĐẠI TRONG CNTT

Tìm hiểu hệ thống mã nguồn mở crawler

Thành viên trong nhóm 3:

- Trần Minh Tuấn
- Trần Thị Thanh Huyền
- Nguyễn Hoàng Hải
- Nguyễn Thị Hồng Hải

Đặt vấn đề

Web crawler gì?

Web crawler là một chương trình hoạt động như là một tập lệnh tự động duyệt qua các trang web trên internet một cách có hệ thống. Web crawler xác định từ khóa chính trên một trang, loại nội dung của trang đó và các đường link có trong trang, trước khi trả kết quả về cho máy tìm kiếm. Tiến trình này được gọi là thu thập thông tin web.

Trang cần lấy dữ liệu được đánh chỉ mục bởi một phần mềm (web crawler). Một web crawler gom các trang của một trang web và đánh chỉ mục nó theo cách tối ưu và tự động, nhằm phục vụ cho yêu cầu của máy tìm kiếm. Crawler cũng giúp ích trong việc thẩm định mã HTML và kiểm tra các đường link có hoạt động hay không.

Vì sao Web Crawler lại quan trọng?

Dữ liệu có vai trò rất quan trọng, là trái tim của bất kỳ doanh nghiệp nào. Ngày nay, với tất cả các open standard như RSS feeds hoặc APIs chia sẻ dữ liệu qua các hệ thống đã trở nên khá dễ dàng.

Nhưng đối với dữ liệu không được cấu trúc hoặc chúng ta không có RSS feeds để dùng, vậy làm cách nào để lấy chúng đây? Lấy một ví dụ đơn giản: bạn có một trang web bán hàng trực tuyến với 1000 sản phẩm. Bạn muốn giá cả của bạn phải cạnh tranh so với các trang web bán hàng khác. Để làm được điều đó, bạn cần phải giám sát trang web của đối thủ và giá cả của các sản phẩm mà bạn cũng đang bán. Nếu có rất nhiều sản phẩm và đối thủ khác nhau, việc lấy dữ liệu trở lên vô cùng khó khăn nếu không có một tiến trình tự động. Đây chính là lý do mà Web Crawler ra đời. Web crawler có thể tự động thu thập thông tin web, tập hợp dữ liệu và giúp ích cho việc đưa ra các quyết định kinh doanh.

Công nghệ Web Crawling trở nên nổi tiếng bởi được ứng dụng vào công cụ tìm kiếm của Google. Họ đã phát hiện ra tầm quan trọng của việc thu thập và đánh chỉ mục lượng dữ liệu vô cùng lớn trên internet. Nhờ đó mà công cụ tìm kiếm của Google được biết đến là một trong những công cụ hoạt động tốt nhất hiện nay.

Các mã nguồn mở phổ biến:

Scrapy

Scrapy là một framework được viết bằng Python, nó cấp sẵn 1 cấu trúc tương đối hoàn chỉnh để thực hiện việc thu thập thông tin và trích xuất dữ liệu từ trang web một cách nhanh chóng và dễ dàng.

Scrapy cho phép thu thập thông tin trang web đồng thời mà không phải đối mặt với vấn đề về luồng, tiến trình, đồng bộ hóa, … Nó xử lý các yêu cầu không đồng bộ một cách nhanh chóng. Scrapy đã phát triển được nhiều năm dựa trên kinh nghiệm của những người thực hiện thu thập thông tin web trên quy mô lớn, vì

vậy nó giải quyết được rất nhiều thách thức mà các nhà phát triển đang phải đối mặt hàng ngày như là:

- Cung cấp cơ chế auto-throttling tự động điều chỉnh tốc độ thu thập dữ liệu dựa trên cả máy chủ web và máy tính người dùng.

- Tự động giữ lại các phiên làm việc. Nó xử lý cookies, đi qua nó một cách dễ dàng thông qua các request. Xác thực cũng không phải là trở ngại ngay cả khi mẫu đăng nhập có CSRF token.

- Nó có thể tránh các bẫy đổi hướng <noscript>

- Lọc các yêu cầu trùng lặp và cho phép tùy chỉnh hành vi lọc

Hơn nữa, kiến trúc của Scrapy được tách ra, đủ để cho phép người dùng tùy chỉnh gần như mọi thứ. Đây là một công cụ rất mạnh và linh hoạt.

Selenium

Do các framework thu thập dữ liệu dưới dạng HTML nên với các xử lý về JS bên trong, chúng ta thường phải làm trình tự các bước giống như trong hàm JS. Nếu ta gặp phải hàm JS khởi tạo đến hàng nghìn dòng lệnh thì dùng các framework thu thập dữ liệu thông thường là không thể. Selenium webdriver là một công cụ rất hay có thể xử lý được những rắc rối này.

Selenium Webdriver là một công cụ tự động hóa các thao tác của một người dùng bình thường trên browser như: truy cập vào máy chủ, click link, điền thông tin, gửi form,.. Selenium giống như một người dùng, nó yêu cầu trang web tải toàn bộ HTML, JS, hình ảnh,.. Do đó, sử dụng selenium có thể khiến tốc độ xử lý chậm hơn và tốn bộ nhớ hơn.

Nutch

Apache Nutch là một framework mã nguồn mở được viết bằng Java. Đây là một dự án phổ biến sử dụng Apache Lucene. Mục tiêu chính của framework này là cào các dữ liệu phi cấu trúc từ các tài nguyên khác nhau như RSS, HTML, CSV, PDF, và kết cấu nó cho quá trình tìm kiếm. Apache Nutch có thể quản lý thu thập dữ liệu hiệu quả. Apache Lucene đóng một vai trò rất quan trọng trong việc giúp Nutch lập chỉ mục và tìm kiếm.

Chức năng của Apache Nutch cũng tương tự như các crawlers khác. Các khác biệt chính của Apache Nutch bao gồm:

- Khả năng mở rộng: giúp mở rộng chức năng tùy biến của người dùng với sự trợ giúp của một số giao diện như Parse, Index và ScoringFilter.

- Pluggale: Cấu hình Apache Nutch dựa trên kiểu plug và chạy, giúp thêm hoặc loại bỏ các chức năng bắt buộc từ cấu hình.

- Tuân thủ các quy tắc robots.txt: Lấy nội dung từ các trang web có tác nhân-người dùng thích hợp cho robots.txt. Nutch sẽ không cào nội dung từ các trang web bị hạn chế.

Crawler4j

Crawler4j cung cấp một giao diện đơn giản để thu thập thông tin Web.

a) Ưu điểm:

- Giao diện đơn giản, dễ dùng

- Dễ dàng scale đến 20M trang

- Rất nhanh (Ví dụ: Đã thu thập và xử lý toàn bộ Wikipedia tiếng Anh trong 10 giờ kể cả thời gian giải nén và lưu trữ cấu trúc liên kết và text của các bài viết)

b) Nhược điểm:

- Không tôn trọng những hạn chế của robots.txt

- Không giới hạn số lượng yêu cầu gửi đến host (Ví dụ: Chính sách của Wikipedia không cho phép các chương trình gửi requests nhanh hơn 1 request/giây. Crawler4j có lịch sử gửi 200 requests/giây)

- Chỉ thu thập nội dung văn bản-text (không hình ảnh hay bất cứ nội dung khác)

- Chỉ với các trang có định dạng UTF-8

Chi tiết về Scrapy

Kiến trúc Scrapy

TODO: update hình

Scrapy Architecture (source: scrapy.org)

Thành phần

- Scheduler: bộ lập lịch thứ tự các URL download.
- Downloader: thực hiện tải dữ liệu. Quản lý các lỗi khi download. Chống trùng.
- Spiders: bóc tách dữ liệu thành các items và requests
- Item Pipeline: xử lý dữ liệu bóc tách được và lưu vào cơ sở dữ liệu.
- Scrapy Engine: quản lý các thành phần trên.

Luồng dữ liệu

Bước 1: Cung cấp URL xuất phát (start*url), được tạo thành một Request lưu trong Scheduler.*

Bước 2 - 3: Scheduler lần lượt lấy các Requests gửi đến Downloader.

Bước 4 - 5: Downloader tải dữ liệu từ internet, được Responses gửi đến Spiders.

Bước 6 - 7: Spiders thực hiện:

• Bóc tách dữ liệu, thu được Item, gửi đến Item Pipeline.

• Tách được URLs, tạo các Requests gửi đến Scheduler.

Bước 8: Item Pipeline thực hiện xử lý dữ liệu bóc tách được. Đơn giản nhất là thực hiện lưu dữ liệu vào database.

Bước 9: kiểm tra Scheduler còn Request?

• Đúng: quay lại Bước 2.

• Sai: kết thúc.

Kiến trúc Scrapy

TODO: update hình

Scrapy Architecture (source: scrapy.org)

Thành phần

- Scheduler: bộ lập lịch thứ tự các URL download.
- Downloader: thực hiện tải dữ liệu. Quản lý các lỗi khi download. Chống trùng.
- Spiders: bóc tách dữ liệu thành các items và requests

- Item Pipeline: xử lý dữ liệu bóc tách được và lưu vào cơ sở dữ liệu.
- Scrapy Engine: quản lý các thành phần trên.

Luồng dữ liệu

Bước 1: Cung cấp URL xuất phát (start*url), được tạo thành một Request lưu trong Scheduler.*

Bước 2 - 3: Scheduler lần lượt lấy các Requests gửi đến Downloader.

Bước 4 - 5: Downloader tải dữ liệu từ internet, được Responses gửi đến Spiders.

Bước 6 - 7: Spiders thực hiện:

• Bóc tách dữ liệu, thu được Item, gửi đến Item Pipeline.

• Tách được URLs, tạo các Requests gửi đến Scheduler.

Bước 8: Item Pipeline thực hiện xử lý dữ liệu bóc tách được. Đơn giản nhất là thực hiện lưu dữ liệu vào database.

Bước 9: kiểm tra Scheduler còn Request?

• Đúng: quay lại Bước 2.

• Sai: kết thúc.

Cài đặt scrapy

Scrapy chạy trên Python 2.7 và Python 3.3 trở lên

Cài đặt Scarpy:

- Cách cài đặt qua pip:

  pip install Scrapy
- Cách anaconda:

conda install -c conda-forge scrapy

Ví dụ về thiết lập Scrapy:

Tạo một Scrapy project mới:

scrapy startproject tutorial

Danh sách mã nguồn:

├── **init**.py
    ├── items.py : định nghĩa cấu trúc dữ liệu sẽ bóc tách.
    ├── pipelines.py : định nghĩa hàm thực hiện việc chèn dữ liệu vào database.
    ├── settings.py : cài đặt cấu hình.
    └── spiders
    ├── **init**.py
    └── vietnamnet*vn.py : định nghĩa hàm bóc tách dữ liệu*

Viết một spider

Spider là class chúng ta định nghĩa và được scrapy sử dụng để scrape thông tin từ một domain (hoặc một nhóm domain)

Chúng ta định nghĩa một danh sách khởi tạo của URLs để download, cách follow links, và cách parse nội dung của pages để trích xuất items.

Để tạo một spider, chúng ta tạo một subclass scrapy.Spider và định nghĩa một số thuộc tính:

- name: định danh spider và nó là duy nhất
- start*urls: một danh sách urls cho spider bắt đầu thực hiện crawl. Các trang được download đầu tiên sẽ bắt đầu từ đây, còn lại sẽ được tạo từ dữ liệu đã được lấy về*
- parse(): một phương thức sẽ được gọi tới để giải quyết một phản hồi đã được download của mỗi start urls. Phản hồi sẽ được truyền tới phương thức như là tham số đầu tiên và duy nhất của phương thức. Phương thức này có trách nhiệm phân tích dữ liệu phản hồ, trích xuất dữ liệu đã được thu thập, tìm kiếm các url mới và tạo các yêu cầu mới trên các url đó.

Chạy spider

Tới thư mục gốc của project và chạy lệnh:

scrapy crawl spider*name*

Dòng lệnh sẽ gửi một số request tới miền spider*name.toscrape.com*

Quá trình thực hiện:

- Scrapy tạo scrapy.Request và gán chúng cho mỗi URL trong danh sách start*urls của spider, phương thức parse được gọi bởi hàm callback*.
- Các Request được lập lịch rồi thực thi và trả về đối tượng scrapy.http.Response, sau đó được đưa trở lại spider thông qua phương thức parse().

Trích xuất dữ liệu

Cách tốt nhất để trích xuất dữ liệu là thử các selector sử dụng Scrapy shell. Chạy lệnh:

scrapy shell 'http://pider*name.toscrape.com/page/1/*'

- Sử dụng cơ chế dựa trên Xpath hoặc biểu thức CSS gọi là Scrapy Selector.

Selector bằng XPath mạnh mẽ hơn CSS

- Scrapy cung cấp class Selector và một số quy ước, shortcut để làm việc với biểu thức xpath và css.
- Đối tượng Selector đại diện các nodes ở trong một văn bản có cấu trúc. Vì thế đầu tiên khởi tạo một selector gắn với node gốc hoặc toàn bộ tài liệu.
- Selector có 4 phương thức cơ bản:
    - xpath(): trả về danh sách các selectors, mỗi cái đại diện cho một node đã được chọn bằng tham số biểu thức xpath truyền vào.
        - css(): trả về danh sách các selectors, mỗi cái đại diện cho một node đã được chọn bằng tham số biểu thức css truyền vào.
        - extract(): trả về một list unicode string với dữ liệu được chọn -> có thể dùng extract*first() để lấy 1 phần tử đầu tiên*
        - re(): trả về danh sách unicode string đã được trích xuất bằng applying tham số biểu thức chính quy truyền vào.

Đối tượng response có thuộc tính selector là thực thể của lớp Selector. Chúng ta có thể truy vấn bằng cách: response.selector.xpath() hoặc response.selector.css() hoặc sử dụng shortcut: response.xpath() hoặc response.css()

Sử dụng item:

Có thể truy cập vào các trường bằng cách:

item = DmozItem() //DmozItem là tên class định nghĩa item

item['title'] = 'Example title'

Sử dụng item trong phương thức parse() (đối tượng yield Item)

yield là gì : http://phocode.com/python/python-iterator-va-generator/

Các link kéo theo:

url = response.urljoin(href.extract())

yield scrapy.Request(url, callback=self.parse*dir*contents)

Lưu trữ dữ liệu đã thu thập được.

Cách đơn giản để lưu trữ dữ liệu thu thập được là sử dụng Feed exports, sử dụng câu lệnh:

scrapy crawl spider*name -o quotes.json*

Cài đặt scrapy

Scrapy chạy trên Python 2.7 và Python 3.3 trở lên

Cài đặt Scarpy:

- Cách cài đặt qua pip:

  pip install Scrapy
- Cách anaconda:

  conda install -c conda-forge scrapy

Ví dụ về thiết lập Scrapy:

Tạo một Scrapy project mới:

scrapy startproject tutorial

Danh sách mã nguồn:

├── **init**.py
    ├── items.py : định nghĩa cấu trúc dữ liệu sẽ bóc tách.
    ├── pipelines.py : định nghĩa hàm thực hiện việc chèn dữ liệu vào database.
    ├── settings.py : cài đặt cấu hình.
    └── spiders
    ├── **init**.py
    └── vietnamnet*vn.py : định nghĩa hàm bóc tách dữ liệu*

Viết một spider

Spider là class chúng ta định nghĩa và được scrapy sử dụng để scrape thông tin từ một domain (hoặc một nhóm domain)

Chúng ta định nghĩa một danh sách khởi tạo của URLs để download, cách follow links, và cách parse nội dung của pages để trích xuất items.

Để tạo một spider, chúng ta tạo một subclass scrapy.Spider và định nghĩa một số thuộc tính:

- name: định danh spider và nó là duy nhất
- start*urls: một danh sách urls cho spider bắt đầu thực hiện crawl. Các trang được download đầu tiên sẽ bắt đầu từ đây, còn lại sẽ được tạo từ dữ liệu đã được lấy về*

- parse(): một phương thức sẽ được gọi tới để giải quyết một phản hồi đã được download của mỗi start urls. Phản hồi sẽ được truyền tới phương thức như là tham số đầu tiên và duy nhất của phương thức. Phương thức này có trách nhiệm phân tích dữ liệu phản hồ, trích xuất dữ liệu đã được thu thập, tìm kiếm các url mới và tạo các yêu cầu mới trên các url đó.

Chạy spider

Tới thư mục gốc của project và chạy lệnh:

scrapy crawl spider*name*

Dòng lệnh sẽ gửi một số request tới miền spider*name.toscrape.com*

Quá trình thực hiện:

- Scrapy tạo scrapy.Request và gán chúng cho mỗi URL trong danh sách start*urls của spider, phương thức parse được gọi bởi hàm callback*.
- Các Request được lập lịch rồi thực thi và trả về đối tượng scrapy.http.Response, sau đó được đưa trở lại spider thông qua phương thức parse().

Trích xuất dữ liệu

Cách tốt nhất để trích xuất dữ liệu là thử các selector sử dụng Scrapy shell. Chạy lệnh:

scrapy shell 'http://pider*name.toscrape.com/page/1/*'

- Sử dụng cơ chế dựa trên Xpath hoặc biểu thức CSS gọi là Scrapy Selector.

Selector bằng XPath mạnh mẽ hơn CSS

- Scrapy cung cấp class Selector và một số quy ước, shortcut để làm việc với biểu thức xpath và css.
- Đối tượng Selector đại diện các nodes ở trong một văn bản có cấu trúc. Vì thế đầu tiên khởi tạo một selector gắn với node gốc hoặc toàn bộ tài liệu.
- Selector có 4 phương thức cơ bản:
    - xpath(): trả về danh sách các selectors, mỗi cái đại diện cho một node đã được chọn bằng tham số biểu thức xpath truyền vào.
        - css(): trả về danh sách các selectors, mỗi cái đại diện cho một node đã được chọn bằng tham số biểu thức css truyền vào.
        - extract(): trả về một list unicode string với dữ liệu được chọn -> có thể dùng extract*first() để lấy 1 phần tử đầu tiên*
        - re(): trả về danh sách unicode string đã được trích xuất bằng applying tham số biểu thức chính quy truyền vào.

Đối tượng response có thuộc tính selector là thực thể của lớp Selector. Chúng ta có thể truy vấn bằng cách: response.selector.xpath() hoặc response.selector.css() hoặc sử dụng shortcut: response.xpath() hoặc response.css()

Sử dụng item:

Có thể truy cập vào các trường bằng cách:

item = DmozItem() //DmozItem là tên class định nghĩa item

item'title' = 'Example title'

Sử dụng item trong phương thức parse() (đối tượng yield Item)

yield là gì : http://phocode.com/python/python-iterator-va-generator/

Các link kéo theo:

url = response.urljoin(href.extract())

yield scrapy.Request(url, callback=self.parse*dir*contents)

Lưu trữ dữ liệu đã thu thập được.

Cách đơn giản để lưu trữ dữ liệu thu thập được là sử dụng Feed exports, sử dụng câu lệnh:

scrapy crawl spider*name -o quotes.json*

Các vấn đề cần giải quyết scrapy

Scrapy (todo update tiêu đề)

- re-extractor (ví dụ như dùng caching)
- download continue
- re-visit for update.
- Xử lí vấn đề caching. Thử nghiệm.

Monitoring scrapy, status, log:

Tìm cách lấy các thông tin về tình trạng crawler, như Scrapy stats nhưng ở dạng re-altime. Mục đích là để biết tình trạng crawler như thế nào.

Using scrapy with Docker: Đóng gói scrapy vào docker. Chạy trong docker

Lưu dữ liệu vào cơ sở dữ liệu

Lưu dữ liệu vào cơ sở dữ liệu

MongoDB: là kiểu noSQL

Tải docker image của mongodb về và chạy.

mongodb

https://hub.docker.com//*mongo/*

Tạo file docker-compose.yml có nội dung:

version: "2"

services:

mongodb:

image: mongo:3.2

ports:

- "27017:27017"

volumes:

- ./mongodb-data/:/data/db

hostname: mongodb

domainname: coclab.-lan

cpus*hares: 512 # 0.5 CPU*

mem*limit: 536870912 # 512 MB RAM*

privileged: true

restart: always

stdin*open: true*

tty: true

Tham chiếu thuật ngữ tương đương giữa MongoDB và MySQL

Database == Database

Collection == Table

Document == Row

Query Mongo : https://docs.mongodb.org/getting-started/python/query/

https://blog.scrapinghub.com/2013/05/13/mongo-bad-for-scraped-data/

đề cập các vấn đề gặp phải khi sử dụng mongodb

1. Locking
2. Poor space efficiency
3. Too Many Databases
4. Ordered data
5. Skip + Limit Queries are slow
6. Restrictions
7. Impossible to keep the working set in memory
8. Data that should be good, ends up bad!

Pipeline

Một item sau khi đã được cào bởi spider sẽ được chuyển đến Item Pipeline để xử lí thông qua một số thành phần được thực hiện tuần tự.

Mỗi thành phần item pipeline là một lớp Python thực hiện một phương thức đơn giản. Các lớp này nhận item và thực hiện các hành động đối với item và đồng thời quyết định xem item đó có tiếp thục pipeline hay bị bỏ hay không còn được xử lý.

Các ứng dụng tiêu biểu của item pipeline:

- Dọn dẹp dữ liệu HTML
- Xác nhận dữ liệu được cào (kiểm tra các items chứa các trường nhất định)
- Kiểm tra (và bỏ) các bản sao
- Lưu trữ các item được cào trong cơ sở dữ liệu

Viết item pipeline

Mỗi thành phần item pipeline là một lớp Python thực hiện các phương thức:

`process_item(item, spider)`

Phương thức này được gọi cho mỗi thành phần item pipeline và phải trả về một đối tượng `Item` (hoặc bất kỳ lớp con nào) hoặc đưa ra một ngoại lệ `DropItem`. Các item bị loại bỏ không còn được xử lý bởi pipeline nữa.

Các tham số:

- **item** (`item`) - item được cào
- **spider** (`BaseSpider`) - spider cào item

`open_spider(spider)`

Phương thức này được gọi khi spider được mở

Các tham số:

spider (`BaseSpider`) - spider đã được mở

`close_spider(spider)`

Phương thức này được gọi khi spider bị đóng

Các tham số:

spider (`BaseSpider`) - spider đã bị đóng

Ví dụ về item pipeline

Xác nhận giá cả và bỏ các mặt hàng không có giá

Giả sử điều chỉnh thuộc tính `price` các mặt hàng không bao gồm thuế VAT (thuộc tính `price_excludes_vat`), và bỏ các items không có giá:

```
from scrapy.exceptions import DropItem
class PricePipeline(object):
vatfactor = 1.15
def processitem(self, item, spider):
if item'price':
if itempriceexcludesvat':
item'price' = item'price'  self.vatfactor
return item
```

```
else:
raise DropItem("Missing price in %s" % item)
```

Viết item vào tệp JSON

Pipeline sau đây lưu trữ tất cả các items đã cào (từ tất cả spiders) vào một tệp `items.jl`, chứa mỗi item trên mỗi dòng, được tuần thự theo định dạng JSON:

```
import json
class JsonWriterPipeline(object):
def init(self):
self.file = open('items.jl', 'wb')
def processitem(self, item, spider):
line = json.dumps(dict(item)) + "n"
self.file.write(line)
return item
```

Ghi các item vào MongoDB

Ghi các items vào MongoDB(https://www.mongodb.com/) bằng cách dùng py-mongo(https://api.mongodb.com/python/current/). Địa chỉ MongoDB và tên cơ sở dữ liệu được chỉ định trong phần Cài đặt scrapy.

Điểm chính của ví dụ này là chỉ ra cách sử dụng phương thức `from_crawler()` và cách làm sạch các tài nguyên đúng cách:

```
import pymongo
```

```
 class MongoPipeline(object):
collectionname = 'scrapyitems'
def init(self, mongouri, mongodb):
self.mongouri = mongouri
self.mongodb = mongodb
@classmethod
def fromcrawler(cls, crawler):
return cls(
mongouri=crawler.settings.get('MONGOURI'),
mongodb=crawler.settings.get('MONGODATABASE', 'items')
)
def openspider(self, spider):
self.client = pymongo.MongoClient(self.mongouri)
self.db = self.clientself.mongodb
def closespider(self, spider):
self.client.close()
def processitem(self, item, spider):
self.dbself.collectionname.insertone(dict(item))
return item
```

Spider

Spider là lớp định nghĩa cách cào một hay nhiều trang, bao gồm cách thực hiện thu thập thông tin và trích xuất dữ liệu có cấu trúc. Nói cách khác, Spider là nơi xác định hành vi tùy chỉnh để thu thập dữ liệu và phân tích cú pháp các trang cho một trang web cụ thể.

Chu kỳ cào dữ liệu đối với Spider như sau:

1. Bắt đầu bằng cách tạo ra request (yêu cầu) ban đầu để thu thập thông tin các URL đầu tiên và chỉ định hàm callback để được gọi với các phản hồi đã tải về từ các requests. Các requests đầu tiên được thực hiện bằng cách gọi phương thức `start_request()`, phương thức mặc định này tạo `Request` cho các URL được chỉ định trong phương thức `start_urls` và `parse` như hàm callback.
2. Trong hàm callback, phân tích phản hồi và trả về các đối tượng `Item`, `Request`. Những requests này cũng sẽ bao gồm một callback (có thể giống nhau) và sau đó sẽ được tải xuống bởi scrapy và sau đó phản hồi sẽ được xử lý bởi các callback được chỉ định.
3. Trong các hàm callback, phân tích nội dung trang, thường dử dụng Selectors(https://doc.scrapy.org/en/0.16/topics/selectors.html#topics-selectors "Selectors") (cũng có thể dùng BeautifulSoup, lxml, v.v..) và tạo ra các items có dữ liệu đã được phân tích.
4. Cuối cùng, các items được trả về từ spider sẽ được duy trì trong cơ sở dữ liệu (trong một số item pipeline) hoặc được ghi vào một tập tin sử dụng Feed exports(https://doc.scrapy.org/en/0.16/topics/feed-exports.html#topics-feed-exports).

Mặc dù chu kỳ này được áp dụng cho bất kỳ loại spider nào nhưng có nhiều loại spider mặc định khác nhau trong Scrapy cho các mục đích khác nhau:

crapy.Spider

`class scrapy.spider.Spider`

Đây là spider đơn giản nhất và tất cả các spider khác đều phải kế thừa. Nó không cung cấp bất kì một chức năng đặc biệt nào, chỉ cung cấp `start_menu()` gửi các request từ thuộc tính `start_urls` spider và gọi các phương thức `parse` của spider cho mỗi kết quả phản hồi.

Spider arguments

Spider có thể nhận được các đối số (argument) sửa đổi hành vi của chúng. Một số cách sử dụng phổ biến cho đối số spider là xác định URL bắt đầu hoặc để hạn chế thu thập thông tin đến các phần nhất định của trang web nhưng chúng có thể được dùng để cấu hình bất kì chức năng nào của spider.

Các đối số spider có thể được truyền thông qua lệnh `crawl` dùng tùy chọn `-a`. Ví dụ:

```
scrapy crawl myspider -a category=electronics
```

Spider có thể truy cập các đối số trong phương thức *_init*:

```
import scrapy
class MySpider(scrapy.Spider):
name = 'myspider'
def init(self, category=None, args, kwargs):
```

super(MySpider, self).**init**(*args, **kwargs)*

self.start*urls* = *'http://www.example.com/categories/%s' % category*

•••

Phương thức mặc định *_init* sẽ lấy bất kỳ đối số spider nào và sao chép chúng vào spider như các thuộc tính. Ví dụ trên cũng có thể được viết như sau:

```
import scrapy
class MySpider(scrapy.Spider):
name = 'myspider'
def start_requests(self):
yield scrapy.Request('http://www.example.com/categories/%s' % self.category)
```

Các spider phổ biến

Scrapy đi kèm với một số spdier phổ biến hữu ích, có thể dùng để phân lớp spider. Mục tiêu của chúng là cung cấp chức năng tiện lợi cho một số trường hợp cào thông thường như theo tất cả các liên kết trên một trang web dựa trên các quy tắc nhất định, thu thập thông tin từ Sitemaps(https://www.sitemaps.org/index.html) hoặc phân tích nguồn cấp dữ liệu XML/CSV.

CrawlSpider

`class scrapy.spiders.CrawlSpider`

Đây là spider phổ biến nhất được dùng để thu thập dữ liệu các trang web thông thường. Vì nó cung cấp cơ chế thuận tiện cho việc liên kết sau bằng cách định nghĩa một bộ quy tắc. Nó có thể không phù hợp nhất cho các trang web hoặc dự án cụ thể của người dùng, vì thế người dùng có thể bắt đầu từ CrawlSpider và ghi đè theo theo nhu cầu cho nhiều chức năng tùy chỉnh hoặc chỉ thực hiện spider riêng của mình.

XMLFeedSpider

`class scrapy.spiders.XMLFeedSpider`

XMLFeedSpider được thiết kế để phân tích các nguồn cung cấp dữ liệu XML bằng cách lặp lại chúng thông qua tên nút nhất định. Trình lặp có thể được chọn từ *iternodes*, *xml* và *html*.

CSVFeedSpider

`class scrapy.spiders.CSVFeedSpider`

Giống với XMLFeedSpider, CSVFeedSpider lặp qua các hàng thay vì các nút như XMLFeedSpider. Phương thức được gọi trong mỗi lần lặp là `parse_row()`.

SitemapSpider

`class scrapy.spiders.SitemapSpider`

SitemapSpider cho phép thu thập thông tin trang web bằng cách tìm kiếm các URL sử dụng Sitemaps(https://www.sitemaps.org/index.html).

Spider này hỗ trợ sơ đồ trang web lồng nhau và tìm kiếm sơ đồ trang web URL từ robots.txt(http://www.robotstxt.org/).

Cài đặt Scrapy

Tùy chọn dòng lệnh

Các tham số được cung cấp bởi dòng lệnh là những giá trị ưu tiên nhất trong các tùy chọn khác. Có thể ghi đè một hoặc nhiều các cài đặt bằng cách sử dụng dòng lệnh `-s` hoặc `--set`.

Ví dụ:

```
scrapy crawl myspider -s LOG_FILE=scrapy.log
```

Cài đặt cho mỗi spider

Spider có thể xác định các thiết lập riêng sẽ được ưu tiên và ghi đè lên các project (dự án) bằng cách thiết lập thuộc tính `custom_settings`:

class MySpider(scrapy.Spider):

name = 'myspider'

custom*settings* =

'SOME*SETTING*': *'some value'*,

}

Mô-đun cài đặt project

Mô-đun cài đặt là tệp cấu hình chuẩn cho dự án scrapy, là nơi mà hầu hết các cài đặt tùy chỉnh sẽ được phổ biến. Đối với một project Scrapy chuẩn, điều này có nghĩa là sẽ thêm hoặc thay đổi cài đặt trong tệp settings.py được tạo trong project.

Cài đặt mặc định cho mỗi lệnh

Mội lệnh công cụ Scrapy có thể có các cài đặt mặc định riêng, ghi đè cài đặt mặc định chung. Những cài đặt lệnh tùy chỉnh được chỉ định rõ trong thuộc tính `default_settings` của lớp lệnh.

Cài đặt mặc định chung

Các mặc định chung được đặt trong `scrapy.settings.default_settings`.

Cách truy cập cài đặt

Trong một spider, các cài đặt có sẵn thông qua `self.settings`:

```
class MySpider(scrapy.Spider):
name = 'myspider'
start_urls = 'http://example.com'
def parse(self, response):
print("Existing settings: %s" % self.settings.attributes.keys())
```

Lưu ý: Thuộc tính `settings` được đặt trong lớp spider cơ bản sau khi spider được khởi tạo. Nếu muốn sử dụng trước khi khởi tạo (ví dụ: trong phương thức `__init__()` của spider), cần phải ghi đè phương thức `from_crawler()` .

Cài đặt có thể được truy cập thông qua thuộc tính `scrapy.crawler.Crawler.settings` của Crawler:

```
class MyExtension(object):
def __init__(self, log_is_enabled=False):
if log_is_enabled:
print("log is enabled!")
@classmethod
def from_crawler(cls, crawler):
settings = crawler.settings
return cls(settings.getbool('LOG_ENABLED'))
```

Tên cài đặt

Tên cài đặt thường được đặt trước với thành phần mà chúng cấu hình. Ví dụ: tên cài đặt đúng cho tiện ích giả tưởng robots.txt sẽ là ROBOTSTXT_ENABLED, ROBOTSTXT_OBEY, ROBOTSTXT_CACHEDIR, *vv*.