A Guide to Scrapy

Digital Technologies and Value Creation - Philippe Blaettchen

Introduction

As you know, Scrapy is a library that allows to manage large scraping and crawling projects. Here, we will go step by step through a Scrapy project to generate our very own Newsfeed.

Of course, to use scrapy, we need to install it. For example, we can use

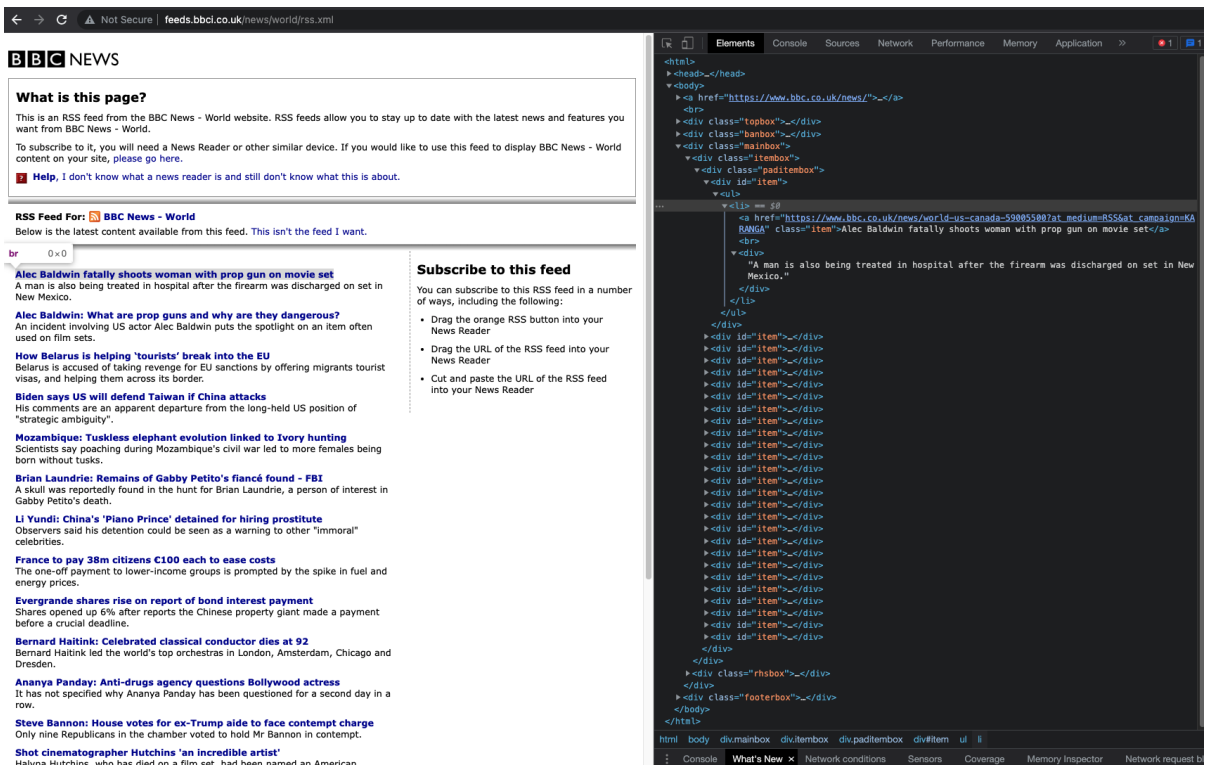
```
pip install scrapy
```

Our starting point will be the website https://www.bbc.co.uk/news/10628494, which gives links to different RSS newsfeeds (note that many other news organizations provide similar RSS feeds, and so the code can be adapted quite easily). We will be using the "World" newsfeed (http://feeds.bbci.co.uk/news/world/rss.xml). Our aim in this project will be to search through any news that contain a mention of "Covid", and to then scrape the actual piece of news on the BBC website.

What a GET request actually returns

The first thing to note is that an RSS feed is actually an XML. This has two effects: on the one hand, it means that we know how to deal with the RSS feed of any news provider relatively effectively. On the other hand, it means that what we see in the browser is not the same as what we get back when requesting the website. Note the difference below:

1. Website inspection: we see an HTML with many div-tags. Some of those have an id-attribute "item". Within each of these items, we find the actual pieces of news as a list (ul-tag for the overall list and li-tags for the individual items).



1. Downloading the XML: back on the website https://www.bbc.co.uk/news/10628494, right-click on the "World" newsfeed and choose "Download linked file" (this doesn't work in all Browsers. For example, in Firefox, you first need to go through these steps https://support.mozilla.org/en-US/questions/930224 and then Option/Alt-Click the link). You can open the downloaded XML file with any text-editor. Here, the news appear within item-tags that contain a title, a description, a link, a more compact link, and a publication date.


```
rss.xml — Edited
<?xml version="1.0" encoding="UTF-8"?>
<?xml-stylesheet title="XSL_formatting" type="text/xsl" href="/shared/bsp/xsl/rss/nolsol.xsl"?>
<rss xmlns:dc="http://purl.org/dc/elements/1.1/" xmlns:content="http://purl.org/rss/1.0/modules/content/" xmlns:atom="http://www.w3.org/2005/Atom"
version="2.0" xmlns:media="http://search.yahoo.com/mrss/">
    <channel>
        <title><![CDATA[BBC News - World]]></title>
        <description><![CDATA[BBC News - World]]></description>
        <link>https://www.bbc.co.uk/news/</link>
        <image>
            <url>https://news.bbcimg.co.uk/nol/shared/img/bbc_news_120x60.gif</url>
            <title>BBC News - World</title>
            <link>https://www.bbc.co.uk/news/</link>
        </image>
        <generator>RSS for Node</generator>
        <lastBuildDate>Fri, 22 Oct 2021 12:17:32 GMT</lastBuildDate>
        <copyright><![CDATA[Copyright: (C) British Broadcasting Corporation, see http://news.bbc.co.uk/2/hi/help/rss/4498287.stm for terms and
conditions of reuse.]]></copyright>
        <language><![CDATA[en-gb]]></language>
        <ttl>15</ttl>
        <item>
            <title><![CDATA[Alec Baldwin fatally shoots woman with prop gun on movie set]]></title>
            <description><![CDATA[A man is also being treated in hospital after the firearm was discharged on set in New Mexico.]]></description>
            <link>https://www.bbc.co.uk/news/world-us-canada-59005500?at_medium=RSS&amp;at_campaign=KARANGA</link>
            <guid isPermaLink="false">https://www.bbc.co.uk/news/world-us-canada-59005500</guid>
            <pubDate>Fri, 22 Oct 2021 09:26:03 GMT</pubDate>
        </item>
        <item>
            <title><![CDATA[Alec Baldwin: What are prop guns and why are they dangerous?]]></title>
            <description><![CDATA[An incident involving US actor Alec Baldwin puts the spotlight on an item often used on film sets.]]></description>
            <link>https://www.bbc.co.uk/news/world-us-canada-59006905?at_medium=RSS&amp;at_campaign=KARANGA</link>
            <guid isPermaLink="false">https://www.bbc.co.uk/news/world-us-canada-59006905</guid>
            <pubDate>Fri, 22 Oct 2021 11:34:39 GMT</pubDate>
        </item>
        <item>
            <title><![CDATA[How Belarus is helping 'tourists' break into the EU]]></title>
            <description><![CDATA[Belarus is accused of taking revenge for EU sanctions by offering migrants tourist visas, and helping them across its
border.]]></description>
            <link>https://www.bbc.co.uk/news/world-58952867?at_medium=RSS&amp;at_campaign=KARANGA</link>
            <guid isPermaLink="false">https://www.bbc.co.uk/news/world-58952867</guid>
            <pubDate>Thu, 21 Oct 2021 23:51:33 GMT</pubDate>
        </item>
        <item>
            <title><![CDATA[Biden says US will defend Taiwan if China attacks]]></title>
            <description><![CDATA[His comments are an apparent departure from the long-held US position of "strategic ambiguity".]]></description>
            <link>https://www.bbc.co.uk/news/world-asia-59005300?at_medium=RSS&amp;at_campaign=KARANGA</link>
            <guid isPermaLink="false">https://www.bbc.co.uk/news/world-asia-59005300</guid>
            <pubDate>Fri, 22 Oct 2021 08:27:22 GMT</pubDate>
        </item>
        <item>
            <title><![CDATA[Mozambique: Tuskless elephant evolution linked to Ivory hunting]]></title>
            <description><![CDATA[Scientists say poaching during Mozambique's civil war led to more females being born without tusks.]]></description>
            <link>https://www.bbc.co.uk/news/world-africa-59008037?at_medium=RSS&amp;at_campaign=KARANGA</link>
            <guid isPermaLink="false">https://www.bbc.co.uk/news/world-africa-59008037</guid>
            <pubDate>Fri, 22 Oct 2021 11:23:39 GMT</pubDate>
        </item>
        <item>
            <title><![CDATA[Brian Laundrie: Remains of Gabby Petito's fiancé found - FBI]]></title>
            <description><![CDATA[A skull was reportedly found in the hunt for Brian Laundrie, a person of interest in Gabby Petito's death.]]></
description>
            <link>https://www.bbc.co.uk/news/world-us-canada-59004831?at_medium=RSS&amp;at_campaign=KARANGA</link>
            <guid isPermaLink="false">https://www.bbc.co.uk/news/world-us-canada-59004831</guid>
            <pubDate>Fri, 22 Oct 2021 08:52:49 GMT</pubDate>
        </item>
        <item>
            <title><![CDATA[Li Yundi: China's 'Piano Prince' detained for hiring prostitute]]></title>
            <description><![CDATA[Observers said his detention could be seen as a warning to other "immoral" celebrities.]]></description>
            <link>https://www.bbc.co.uk/news/world-asia-china-59005937?at_medium=RSS&amp;at_campaign=KARANGA</link>
            <guid isPermaLink="false">https://www.bbc.co.uk/news/world-asia-china-59005937</guid>
            <pubDate>Fri, 22 Oct 2021 05:32:44 GMT</pubDate>
        </item>
        <item>
```


The content is the same, but your browser has already converted the XML into an HTML. When you request the page (say with Requests, or in Scrapy), however, you will get back the original XML!

The Scrapy shell

Let's get started in Scrapy. It's always a good idea to explore a website inside the Scrapy shell, before even setting up a project. Open your terminal, navigate to your virtual environment where Scrapy is installed (if you are using one), and type `scrapy shell`

The output should look something like this:

```
(dtvc_env) philippe@192 ~ % scrapy shell
2021-10-22 14:03:17 [scrapy.utils.log] INFO: Scrapy 2.5.0 started (bot: scrapybot)
2021-10-22 14:03:17 [scrapy.utils.log] INFO: Versions: lxml 4.6.3.0, libxml2 2.9.10, cssselect 1.1.0, parsel 1.6.0, w3lib 1.22.0, Twisted 21.7.0, Python 3.7.7 (defaul
t, Mar 26 2020, 10:32:53) - [Clang 4.0.1 (tags/RELEASE_401/final)], pyOpenSSL 20.0.1 (OpenSSL 1.1.1l  24 Aug 2021), cryptography 3.4.8, Platform Darwin-20.6.0-x86_64-
i386-64bit
2021-10-22 14:03:17 [scrapy.utils.log] DEBUG: Using reactor: twisted.internet.selectreactor.SelectReactor
2021-10-22 14:03:17 [scrapy.crawler] INFO: Overridden settings:
{'DUPEFILTER_CLASS': 'scrapy.dupefilters.BaseDupeFilter',
 'LOGSTATS_INTERVAL': 0}
2021-10-22 14:03:17 [scrapy.extensions.telnet] INFO: Telnet Password: e2cd766a4fe84f3d
2021-10-22 14:03:17 [scrapy.middleware] INFO: Enabled extensions:
['scrapy.extensions.corestats.CoreStats',
 'scrapy.extensions.telnet.TelnetConsole',
 'scrapy.extensions.memusage.MemoryUsage']
2021-10-22 14:03:17 [scrapy.middleware] INFO: Enabled downloader middlewares:
['scrapy.downloadermiddlewares.httpauth.HttpAuthMiddleware',
 'scrapy.downloadermiddlewares.downloadtimeout.DownloadTimeoutMiddleware',
 'scrapy.downloadermiddlewares.defaultheaders.DefaultHeadersMiddleware',
 'scrapy.downloadermiddlewares.useragent.UserAgentMiddleware',
 'scrapy.downloadermiddlewares.retry.RetryMiddleware',
 'scrapy.downloadermiddlewares.redirect.MetaRefreshMiddleware',
 'scrapy.downloadermiddlewares.httpcompression.HttpCompressionMiddleware',
 'scrapy.downloadermiddlewares.redirect.RedirectMiddleware',
 'scrapy.downloadermiddlewares.cookies.CookiesMiddleware',
 'scrapy.downloadermiddlewares.httpproxy.HttpProxyMiddleware',
 'scrapy.downloadermiddlewares.stats.DownloaderStats']
2021-10-22 14:03:17 [scrapy.middleware] INFO: Enabled spider middlewares:
['scrapy.spidermiddlewares.httperror.HttpErrorMiddleware',
 'scrapy.spidermiddlewares.offsite.OffsiteMiddleware',
 'scrapy.spidermiddlewares.referer.RefererMiddleware',
 'scrapy.spidermiddlewares.urllength.UrlLengthMiddleware',
 'scrapy.spidermiddlewares.depth.DepthMiddleware']
2021-10-22 14:03:17 [scrapy.middleware] INFO: Enabled item pipelines:
[]
2021-10-22 14:03:17 [scrapy.extensions.telnet] INFO: Telnet console listening on 127.0.0.1:6023
2021-10-22 14:03:17 [asyncio] DEBUG: Using selector: KqueueSelector
[s] Available Scrapy objects:
[s]   scrapy     scrapy module (contains scrapy.Request, scrapy.Selector, etc)
[s]   crawler    <scrapy.crawler.Crawler object at 0x7f88e29dced0>
[s]   item       {}
[s]   settings   <scrapy.settings.Settings object at 0x7f88e2ab8cd0>
[s] Useful shortcuts:
[s]   fetch(url[, redirect=True]) Fetch URL and update local objects (by default, redirects are followed)
[s]   fetch(req)                  Fetch a scrapy.Request and update local objects
[s]   shelp()           Shell help (print this help)
[s]   view(response)    View response in a browser
2021-10-22 14:03:18 [asyncio] DEBUG: Using selector: KqueueSelector
In [1]:
```

You can now issue a GET request, by typing `fetch("http://feeds.bbci.co.uk/news/world/rss.xml")`

Hopefully, the website is properly obtained, which is signified as follows:

```
In [1]: fetch("http://feeds.bbci.co.uk/news/world/rss.xml")
2021-10-22 14:04:49 [scrapy.core.engine] INFO: Spider opened
2021-10-22 14:04:49 [scrapy.core.engine] DEBUG: Crawled (200) <GET http://feeds.bbci.co.uk/news/world/rss.xml> (referer: None)
```

The return value (the XML document) can now be accessed through the variable `response` :

```
In [2]: response
Out[2]: <200 http://feeds.bbci.co.uk/news/world/rss.xml>
```

With this, we can start to get our hands dirty. We know that all pieces of news are within item-tags. So let's begin by searching for the first item-tag in the XML:

```
In [3]: response.xpath('//item').get()
Out[3]: '<item xmlns:dc="http://purl.org/dc/elements/1.1/" xmlns:content="http://purl.org/rss/1.0/modules/content/" xmlns:atom="http://www.w3.org/2005/Atom" xmlns:med
ia="http://search.yahoo.com/mrss/">\n            <title>Alec Baldwin fatally shoots woman with prop gun on movie set</title>\n            <description>A man is also b
eing treated in hospital after the firearm was discharged on set in New Mexico.</description>\n            <link>https://www.bbc.co.uk/news/world-us-canada-59005500?a
t_medium=RSS&amp;at_campaign=KARANGA</link>\n            <guid isPermaLink="false">https://www.bbc.co.uk/news/world-us-canada-59005500</guid>\n            <pubDate>Fr
i, 22 Oct 2021 09:26:03 GMT</pubDate>\n        </item>'
```

Here, we used XPath, to find any item-tag in the document. `//` selects nodes anywhere in the document, if you just type `response.xpath('//item')` you get back all item-tag nodes (or objects representing them). Using `.get()` returns the first of these nodes as a string. Using instead `.getall()` returns all nodes as a string.

A final note: if you use `/` instead of `//` , it only selects top-level nodes. Try out `/item` and `/rss` to see this.

The string we got back by using `.get()` is a bit messy. Instead, we want to use the nice structure of the XML to navigate through the news item. So let's drop the `.get()` for now. Instead, we simply use `response.xpath('//item')[0]` .

What is the `[0]` doing here? Remember that `response.xpath('//item')` returns **all** item-tags. But we are only concerend about the first piece of news for now.

```
In [9]: news_item = response.xpath('//item')[0]
```

To verify that this worked, you can try out `news_item.get()`. Will there be any difference between `news_item.get()` and `news_item.getall()`? Why not?

Our `news_item` is an object within the XML tree. Hence, we can use XPath to search within the news item (in the tags within this specific item-tag). Let's start by looking at the title of the news (we are using `.get()` to see the information as a String instead of just an object):

```
In [12]: news_item.xpath('.//title').get()
Out[12]: '<title xmlns:dc="http://purl.org/dc/elements/1.1/" xmlns:content="http://purl.org/rss/1.0/modules/content/" xmlns:atom="http://www.w3.org/2005/Atom" xmlns:m
edia="http://search.yahoo.com/mrss/">Alec Baldwin fatally shoots woman with prop gun on movie set</title>'
```

One thing looks new here: the `.//`. This means we are looking for a title-tag, but only within the children of our item-tag. If we just write `//`, we will get all the title-tags in the XML (and only by choosing the first one with `.get()` we coincidentially end up on the same one).

Another thing to note is the formating. We have the entire node here, with all sorts of meta-information. We don't care about this, we just want the text between the node tags. This we can get using `/text()`:

```
In [13]: news_item.xpath('.//title/text()').get()
Out[13]: 'Alec Baldwin fatally shoots woman with prop gun on movie set'
```

Not too bad, right? We can proceed identically for the other tags we may care about. Remember, we found a title, a description, a link, a more compact link, and a publication date. Let's get all of them (or more specifically, the text within them):

```
In [18]: news_item.xpath('.//title/text()').get()
Out[18]: 'Alec Baldwin fatally shoots woman with prop gun on movie set'

In [19]: news_item.xpath('.//description/text()').get()
Out[19]: 'A man is also being treated in hospital after the firearm was discharged on set in New Mexico.'

In [20]: news_item.xpath('.//link/text()').get()
Out[20]: 'https://www.bbc.co.uk/news/world-us-canada-59005500?at_medium=RSS&at_campaign=KARANGA'

In [21]: news_item.xpath('.//guid/text()').get()
Out[21]: 'https://www.bbc.co.uk/news/world-us-canada-59005500'

In [22]: news_item.xpath('.//pubDate/text()').get()
Out[22]: 'Fri, 22 Oct 2021 09:26:03 GMT'
```

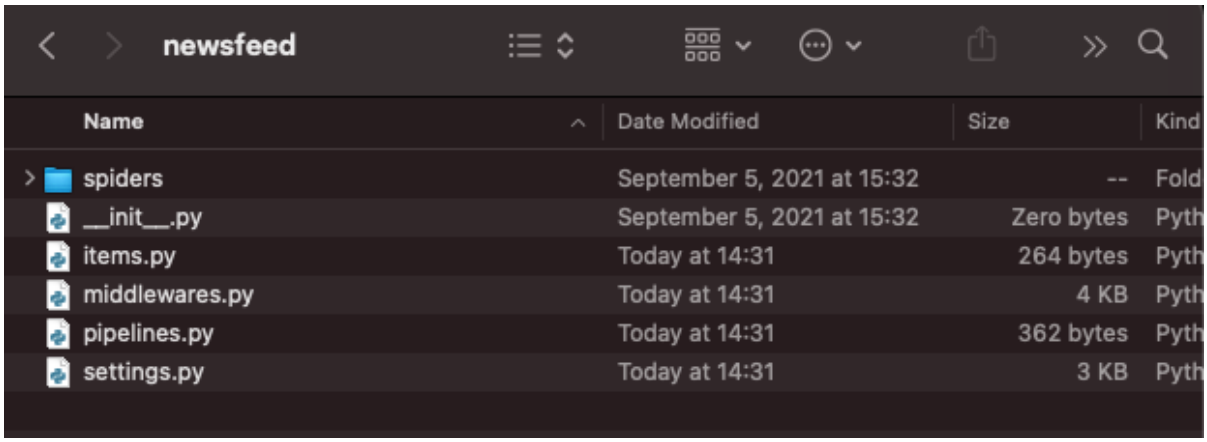
With all this information, we are now ready to use standard Python code to work with the data. So let's start creating a Spider.

Building a spider

It's time to actually start a Scrapy project. To do so, we leave the shell with `exit()`. We are then back in our terminal. To create a new Scrapy project, we can run `scrapy startproject name directory`, for example:

```
(dtvc_env) philippe@192 ~ % scrapy startproject newsfeed /Users/philippe/Documents
New Scrapy project 'newsfeed', using template directory '/Users/philippe/anaconda3/envs/dtvc_env/lib/python3.7/site-packages/scrapy/templates/project', created in:
    /Users/philippe/Documents

You can start your first spider with:
    cd /Users/philippe/Documents
    scrapy genspider example example.com
```

If we navigate into the project folder, we see something like this:



The subfolder Spider will eventually contain our spider. How to get started? Luckily, Scrapy already tells us:

```
 cd "Folder with scrapy project"
scrapy genspider spidername startdomain
```

We could use the following parameters (note that you have to use quotation marks if you path contains spaces):

```
(dtvc_env) philippe@192 ~ % cd /Users/philippe/Documents
(dtvc_env) philippe@192 Documents % scrapy genspider news feeds.bbci.co.uk
Created spider 'news' using template 'basic' in module:
  newsfeed.spiders.news
```

Inside the Spiders subfolder, we now find a file `news.py`. We can open this with any editor of our choice (I use Visual Studio Code, but even the standard text editor does the trick).

When opening the spider, you'll see this:

```
import scrapy


class NewsSpider(scrapy.Spider):
    name = 'news'
    allowed_domains = ['feeds.bbci.co.uk']
    start_urls = ['http://feeds.bbci.co.uk/']

    def parse(self, response):
        pass

```

A few things to note here:

- the `name` variable tells us how scrapy will reference this spider
- the `allowed_domains` variable is not strictly necessary, but helps avoiding to build web crawlers that run off into corners of the internet that they shouldn't go to.
- the variable `start_urls` contains a list of websites on which we want to start scraping. This is based on the Domain we gave, but we need to slightly adjust it, so we get to the right feed (change to "http://feeds.bbci.co.uk/news/world/rss.xml" instead)

- you don't have to have a list of starting urls. Instead, you can also use a `start_requests` function, which is essentially an initialization function. We will see this when we deal with Splash.
- any spider has a `parse(self, response)` function. This is the function that parses the websites that our spider finds. Remember how we got a `response` variable containing the website when fetching it within the Scrapy shell? That's exactly the `response` input to this function.

Currently, the `parse` function doesn't do anything. We will change this, to instead go through the XML and return the relevant data.

We start with essentially the same code that we used in the Scrapy shell (remember that we also had the same `response` object there!):

```
import scrapy


class NewsSpider(scrapy.Spider):
    name = 'news'
    allowed_domains = ['feeds.bbci.co.uk']
    start_urls = ['http://feeds.bbci.co.uk/news/world/rss.xml']

    def parse(self, response):
        news_item = response.xpath('//item')[0]

        print(news_item.xpath('.//title/text()').get())
        print(news_item.xpath('.//description/text()').get())
        print(news_item.xpath('.//link/text()').get())
        print(news_item.xpath('.//guid/text()').get())
        print(news_item.xpath('.//pubDate/text()').get())
```

The only difference is that we are explicitly printing out the items.

It's now time to run our first spider. For this, we return to the console, make sure we are in the folder where we created the project (or in the project), and run `scrapy crawl news`. You'll see a lot of text, but most importantly is this part:

```
2021-10-22 14:47:14 [scrapy.core.engine] INFO: Spider opened
2021-10-22 14:47:14 [scrapy.extensions.logstats] INFO: Crawled 0 pages (at 0 pages/min), scraped 0 items (at 0 items/min)
2021-10-22 14:47:14 [scrapy.extensions.telnet] INFO: Telnet console listening on 127.0.0.1:6023
2021-10-22 14:47:14 [scrapy.core.engine] DEBUG: Crawled (200) <GET http://feeds.bbci.co.uk/robots.txt> (referer: None)
2021-10-22 14:47:14 [scrapy.core.engine] DEBUG: Crawled (200) <GET http://feeds.bbci.co.uk/news/world/rss.xml> (referer: None)
Alec Baldwin fatally shoots woman with prop gun on movie set
A man is also being treated in hospital after the firearm was discharged on set in New Mexico.
https://www.bbc.co.uk/news/world-us-canada-59005500?at_medium=RSS&at_campaign=KARANGA
https://www.bbc.co.uk/news/world-us-canada-59005500
Fri, 22 Oct 2021 09:26:03 GMT
2021-10-22 14:47:14 [scrapy.core.engine] INFO: Closing spider (finished)
```

What happened? We see that Scrapy first went through the robots.txt file of our domain (it will always try to do that and follow the guidelines here, unless we specifically turn this behavior off). Then, it crawled our starting url and parsed it (we can see that it parsed it, because exactly the print commands that we set were returned).

Now that we know how our spider works, we can make it a bit more complicated: We will not just go through the first piece of news, but through all news items in a `for` loop. Instead of

printing out the news, we will search for any hint of a mention of Covid in the title and description. If Covid is mentioned, we print out the (compact) url:

```
import scrapy


class NewsSpider(scrapy.Spider):
    name = 'news'
    allowed_domains = ['feeds.bbci.co.uk']
    start_urls = ['http://feeds.bbci.co.uk/news/world/rss.xml']

    def parse(self, response):
        news_items = response.xpath('//item')
        for news_item in news_items:
            title = news_item.xpath('.//title/text()').get()
            description = news_item.xpath('.//description/text()').get()
            link = news_item.xpath('.//link/text()').get()
            compact_link = news_item.xpath('.//guid/text()').get()
            date = news_item.xpath('.//pubDate/text()').get()

            if 'covid' in title.lower() or 'covid' in description.lower():
                print("Article with Covid news: " + compact_link)
```

Note here that we use `.lower()` on both the title and the description strings to avoid having matching issues, just because Covid may be spelled in different ways.

We run our spider once more (with `scrapy crawl news`):

```
2021-10-22 14:57:05 [scrapy.core.engine] DEBUG: Crawled (200) <GET http://feeds.bbci.co.uk/robots.txt> (referer: None)
2021-10-22 14:57:05 [scrapy.core.engine] DEBUG: Crawled (200) <GET http://feeds.bbci.co.uk/news/world/rss.xml> (referer: None)
Article with Covid news: https://www.bbc.co.uk/news/world-europe-58998366
Article with Covid news: https://www.bbc.co.uk/news/world-us-canada-58989555
2021-10-22 14:57:05 [scrapy.core.engine] INFO: Closing spider (finished)
```

It seems that there were two news articles. Next, we adjust our crawler to only consider news items that are less than a five days old. For this, we need to convert the publication date string into an actual date object.

Recall the format of publication dates: "Wed, 20 Oct 2021 10:43:03 GMT". We convert this, using the datetime package (finding the right conversion can sometimes be a bit tedious - but Google generally does the trick). Here is an example:

In [24]:

```
from datetime import datetime

date = "Wed, 20 Oct 2021 10:43:03 GMT"
converted_date = datetime.strptime(date, '%a, %d %b %Y %H:%M:%S %Z')
converted_date
```

Out[24]:

```
datetime.datetime(2021, 10, 20, 10, 43, 3)
```

We can then measure the time difference between now and that date:

In [25]:

```
time_delta = datetime.today() - converted_date
time_delta
```

Out[25]:

```
datetime.timedelta(days=2, seconds=16172, microseconds=32030)
```

The `time_delta` can easily be converted into days:

In [26]:
```
time_delta.days
```

Out[26]: 2

Let's transfer all of this into our spider:

```
import scrapy
from datetime import datetime

class NewsSpider(scrapy.Spider):
    name = 'news'
    allowed_domains = ['feeds.bbci.co.uk']
    start_urls = ['http://feeds.bbci.co.uk/news/world/rss.xml']

    def parse(self, response):
        news_items = response.xpath('//item')
        for news_item in news_items:
            title = news_item.xpath('.//title/text()').get()
            description = news_item.xpath('.//description/text()').get()
            link = news_item.xpath('.//link/text()').get()
            compact_link = news_item.xpath('.//guid/text()').get()
            date = news_item.xpath('.//pubDate/text()').get()

            if 'covid' in title.lower() or 'covid' in description.lower():
                converted_date = datetime.strptime(date, '%a, %d %b %Y %H:%M:%S %Z')
                time_delta = datetime.today() - converted_date
                if time_delta.days < 5:
                    print("Article with Covid news from the last 5 days: " + compact_link)
```

Don't forget the import here!

As both articles found earlier were from the last five days, the result of our crawling effort is unchanged:

```
2021-10-22 15:15:29 [scrapy.core.engine] DEBUG: Crawled (200) <GET http://feeds.bbci.co.uk/robots.txt> (referer: None)
2021-10-22 15:15:29 [scrapy.core.engine] DEBUG: Crawled (200) <GET http://feeds.bbci.co.uk/news/world/rss.xml> (referer: None)
Article with Covid news from the last 5 days: https://www.bbc.co.uk/news/world-europe-58998366
Article with Covid news from the last 5 days: https://www.bbc.co.uk/news/world-us-canada-58989555
2021-10-22 15:15:30 [scrapy.core.engine] INFO: Closing spider (finished)
```

Congratulations, you are now able to find news that are relevant to you!

Let's get crawling

The key idea of crawling is that we (automatically) find new websites to parse. Hence, instead of just printing out the links to the websites, we will actually parse them. We can do this, by creating a new request to the links we identified. For example,

```python
import scrapy
from datetime import datetime

class NewsSpider(scrapy.Spider):
    name = 'news'
    allowed_domains = ['feeds.bbci.co.uk']
    start_urls = ['http://feeds.bbci.co.uk/news/world/rss.xml']

    def parse(self, response):
        news_items = response.xpath('//item')
        for news_item in news_items:
            title = news_item.xpath('.//title/text()').get()
            description = news_item.xpath('.//description/text()').get()
            link = news_item.xpath('.//link/text()').get()
            compact_link = news_item.xpath('.//guid/text()').get()
            date = news_item.xpath('.//pubDate/text()').get()

            if 'covid' in title.lower() or 'covid' in description.lower():
                converted_date = datetime.strptime(date, '%a, %d %b %Y %H:%M:%S %Z')
                time_delta = datetime.today() - converted_date
                if time_delta.days < 5:
                    page_to_scrape = response.urljoin(compact_link)
                    yield scrapy.Request(page_to_scrape)
```

To do so, we use the function `scrapy.Request(url)`. However, we have to first ensure that we have a "proper" url. Here, `response.urljoin(url)` helps to ensure that we are good to go even if the link we obtained is only relative.

There is an issue when running our spider, however:

```
2021-10-22 15:20:48 [scrapy.core.engine] DEBUG: Crawled (200) <GET http://feeds.bbci.co.uk/robots.txt> (referer: None)
2021-10-22 15:20:48 [scrapy.core.engine] DEBUG: Crawled (200) <GET http://feeds.bbci.co.uk/news/world/rss.xml> (referer: None)
2021-10-22 15:20:48 [scrapy.spidermiddlewares.offsite] DEBUG: Filtered offsite request to 'www.bbc.co.uk': <GET https://www.bbc.co.uk/news/world-europe-58998366>
2021-10-22 15:20:48 [scrapy.core.engine] INFO: Closing spider (finished)
```

What happened? The actual BBC News are not on the domain "feeds.bbci.co.uk". Instead, they are on the domain "www.bbc.co.uk". But this is not currently an allowed domain in our spider. Let's add it, and try again:

```python
import scrapy
from datetime import datetime

class NewsSpider(scrapy.Spider):
    name = 'news'
    allowed_domains = ['feeds.bbci.co.uk','www.bbc.co.uk']
    start_urls = ['http://feeds.bbci.co.uk/news/world/rss.xml']

    def parse(self, response):
        news_items = response.xpath('//item')
        for news_item in news_items:
            title = news_item.xpath('.//title/text()').get()
            description = news_item.xpath('.//description/text()').get()
            link = news_item.xpath('.//link/text()').get()
            compact_link = news_item.xpath('.//guid/text()').get()
            date = news_item.xpath('.//pubDate/text()').get()

            if 'covid' in title.lower() or 'covid' in description.lower():
                converted_date = datetime.strptime(date, '%a, %d %b %Y %H:%M:%S %Z')
                time_delta = datetime.today() - converted_date
                if time_delta.days < 5:
                    page_to_scrape = response.urljoin(compact_link)
                    yield scrapy.Request(page_to_scrape)
```

```
2021-10-22 15:24:43 [scrapy.core.engine] DEBUG: Crawled (200) <GET http://feeds.bbci.co.uk/robots.txt> (referer: None)
2021-10-22 15:24:43 [scrapy.core.engine] DEBUG: Crawled (200) <GET http://feeds.bbci.co.uk/news/world/rss.xml> (referer: None)
2021-10-22 15:24:43 [scrapy.core.engine] DEBUG: Crawled (200) <GET https://www.bbc.co.uk/robots.txt> (referer: None)
2021-10-22 15:24:43 [scrapy.core.engine] DEBUG: Crawled (200) <GET https://www.bbc.co.uk/news/world-us-canada-58989555> (referer: http://feeds.bbci.co.uk/news/world/r
ss.xml)
2021-10-22 15:24:43 [scrapy.core.engine] DEBUG: Crawled (200) <GET https://www.bbc.co.uk/news/world-europe-58998366> (referer: http://feeds.bbci.co.uk/news/world/rss.
xml)
2021-10-22 15:24:43 [scrapy.core.engine] INFO: Closing spider (finished)
```

This time, we successfully crawled also the two news sites.

But why did nothing happen? Because the function called when parsing the news sites is exactly the function we have here. But the news sites have a completely different structure (they are not XMLs, but actual HTMLs), so nothing happens.

To analyze the actual news, we thus create a second parse method, and slightly adjust our new request:

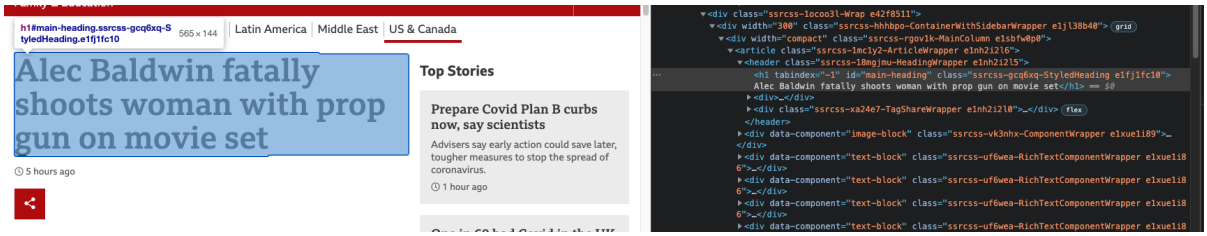
```python
import scrapy
from datetime import datetime

class NewsSpider(scrapy.Spider):
    name = 'news'
    allowed_domains = ['feeds.bbci.co.uk','www.bbc.co.uk']
    start_urls = ['http://feeds.bbci.co.uk/news/world/rss.xml']

    def parse(self, response):
        news_items = response.xpath('//item')
        for news_item in news_items:
            title = news_item.xpath('.//title/text()').get()
            description = news_item.xpath('.//description/text()').get()
            link = news_item.xpath('.//link/text()').get()
            compact_link = news_item.xpath('.//guid/text()').get()
            date = news_item.xpath('.//pubDate/text()').get()

            if 'covid' in title.lower() or 'covid' in description.lower():
                converted_date = datetime.strptime(date, '%a, %d %b %Y %H:%M:%S %Z')
                time_delta = datetime.today() - converted_date
                if time_delta.days < 5:
                    page_to_scrape = response.urljoin(compact_link)
                    yield scrapy.Request(page_to_scrape, callback=self.parse_news)

    def parse_news(self,response):
        print(" A news site was successfully parsed ")
```

We can run our spider again to see that it worked:

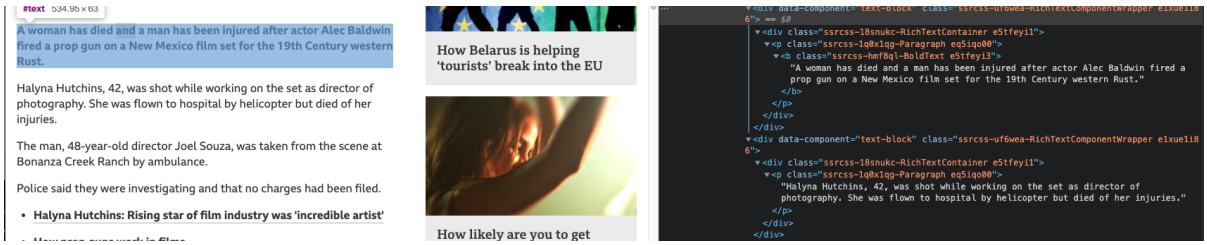
```
2021-10-22 15:27:34 [scrapy.core.engine] DEBUG: Crawled (200) <GET http://feeds.bbci.co.uk/robots.txt> (referer: None)
2021-10-22 15:27:34 [scrapy.core.engine] DEBUG: Crawled (200) <GET http://feeds.bbci.co.uk/news/world/rss.xml> (referer: None)
2021-10-22 15:27:34 [scrapy.core.engine] DEBUG: Crawled (200) <GET https://www.bbc.co.uk/robots.txt> (referer: None)
2021-10-22 15:27:34 [scrapy.core.engine] DEBUG: Crawled (200) <GET https://www.bbc.co.uk/news/world-us-canada-58989555> (referer: http://feeds.bbci.co.uk/news/world/r
ss.xml)
2021-10-22 15:27:34 [scrapy.core.engine] DEBUG: Crawled (200) <GET https://www.bbc.co.uk/news/world-europe-58998366> (referer: http://feeds.bbci.co.uk/news/world/rss.
xml)
 A news site was successfully parsed
 A news site was successfully parsed
2021-10-22 15:27:35 [scrapy.core.engine] INFO: Closing spider (finished)
```

Let's now actually analyze the news websites. We can again use the Scrapy Shell together with the inspection tool (remember, we have an HTML this time). Going this way, we identify a few key elements about a news page:

We first find the title:



We can then see that the article text is within div-tags with the attribute data-component="text-block". There is a slight hitch, however: In some cases, the div-tag contains another div-tag, that contains a p-tag, that contains the text (see second example below). In other cases, the p-tag contains a b-tag with all the text (see first example below).



Using the Scrpay shell, we can play around with this, then convert what we find into our parse function:

```
25      def parse_news(self,response):
26          title = response.xpath('//h1[@id="main-heading"]/text()').get()
27          text = ""
28          paragraphs = response.xpath('//div[@data-component="text-block"]/div')
29          for paragraph in paragraphs:
30              if len(paragraph.xpath('.//p/text()')) > 0:
31                  text = text + '\n' + paragraph.xpath('.//p/text()').get()
32              elif len(paragraph.xpath('.//p/b/text()')) > 0:
33                  text = text + '\n' + paragraph.xpath('.//p/b/text()').get()
34
35          print("Full text scraped: ")
36          print(text)
```

Let's analyze this:

- we obtain the title directly from the XPath, using the fact that it comes in an h1-tag with id="main-heading"
- we then obtain all paragraphs (div-tags with data-component="text-block"), and the second-layer div-tag within
- we then loop through the paragraphs. If we find there is text within the p-tag, we use that. Else, if we find there is text within a b-tag within the p-tag, we take that.
- we add all the paragraphs into a long text variable. The `\n` adds a line break into our string.
- finally, we print out the complete text (with a small introduction).

When running our spider again, we see that it worked:

```
2021-10-22 15:39:38 [scrapy.core.engine] DEBUG: Crawled (200) <GET http://feeds.bbci.co.uk/robots.txt> (referer: None)
2021-10-22 15:39:38 [scrapy.core.engine] DEBUG: Crawled (200) <GET http://feeds.bbci.co.uk/news/world/rss.xml> (referer: None)
2021-10-22 15:39:38 [scrapy.core.engine] DEBUG: Crawled (200) <GET https://www.bbc.co.uk/robots.txt> (referer: None)
2021-10-22 15:39:38 [scrapy.core.engine] DEBUG: Crawled (200) <GET https://www.bbc.co.uk/news/world-us-canada-58989555> (referer: http://feeds.bbci.co.uk/news/world/r
ss.xml)
2021-10-22 15:39:38 [scrapy.core.engine] DEBUG: Crawled (200) <GET https://www.bbc.co.uk/news/world-europe-58998366> (referer: http://feeds.bbci.co.uk/news/world/rss.
xml)
Full text scraped:

A standoff in Chicago, where the head of the city's largest police union is urging officers to defy a vaccine mandate, is the latest battle line being drawn in a nati
onwide fight over Covid-19 jabs.
Chicago, a city of nearly three million people, has seen more than 1,600 sexual assaults, nearly 3,000 shootings and 649 murders this year – a 14% increase over last.

Just as violent crimes have risen, though, thousands of the city's police force may not show up to work.
Officers are weighing whether to resist a mayoral mandate requiring all public employees to report their vaccine status. City employees must now show proof of vaccina
tion or submit to bi-weekly testing, unless approved for a religious or medical exemption. By the end of this year, all employees must be vaccinated.
Nearly one-third of Chicago's almost 13,000-member police department have so far refused to register their vaccination status, putting them on track for dismissal.
Twenty-one have been officially removed from active duty so far, but some officials have warned that the mandate could leave Chicago's police force dangerously deplet
ed.
```

Instead of just printing out the text, let's save it to a file. We use a json file, as it is very easy to interact with json files in Python (they correspond to lists and dictionaries nested into each other).

There are many ways to do this, but the simplest is to add an output file to our spider to which the parse functions transmit their output:

```python
import scrapy
from datetime import datetime

class NewsSpider(scrapy.Spider):
    name = 'news'
    allowed_domains = ['feeds.bbci.co.uk','www.bbc.co.uk']
    start_urls = ['http://feeds.bbci.co.uk/news/world/rss.xml']

    #location of json file
    custom_settings = {
        'FEED_URI': 'news.json',
        'FEED_FORMAT': 'json',
        'FEED_EXPORTERS': {
            'json': 'scrapy.exporters.JsonItemExporter',
        },
        'FEED_EXPORT_ENCODING': 'utf-8',
    }

    def parse(self, response):
        news_items = response.xpath('//item')
        for news_item in news_items:
            title = news_item.xpath('.//title/text()').get()
            description = news_item.xpath('.//description/text()').get()
            link = news_item.xpath('.//link/text()').get()
            compact_link = news_item.xpath('.//guid/text()').get()
            date = news_item.xpath('.//pubDate/text()').get()

            if 'covid' in title.lower() or 'covid' in description.lower():
                converted_date = datetime.strptime(date, '%a, %d %b %Y %H:%M:%S %Z')
                time_delta = datetime.today() - converted_date
                if time_delta.days < 5:
                    page_to_scrape = response.urljoin(compact_link)
                    yield scrapy.Request(page_to_scrape, callback=self.parse_news)

    def parse_news(self,response):
        title = response.xpath('//h1[@id="main-heading"]/text()').get()
        text = ""
        paragraphs = response.xpath('//div[@data-component="text-block"]/div')
        for paragraph in paragraphs:
            if len(paragraph.xpath('.//p/text()')) > 0:
                text = text + '\n' + paragraph.xpath('.//p/text()').get()
            elif len(paragraph.xpath('.//p/b/text()')) > 0:
                text = text + '\n' + paragraph.xpath('.//p/b/text()').get()

        scraped_info = {
            'title' : title,
            'text' : text
        }

        yield scraped_info
```

Note the two changes:

1. We added a json-file output for the spider. This part of the code can simply be ported, so don't worry too much about the `'FEED_EXPORTERS'`. The main thing is the `'FEED_URI'`, which specifies the name of our output file.
2. We created a dictionary of our data (with an entry title and an entry text), we then `yield` this dictionary to a higher level. Our spider will automatically catch what is yielded and put it into the the output strem (our json-file).

Let's run this again. There is a lot happening, but the key information is at the end of the log:

```
2021-10-22 15:48:31 [scrapy.core.engine] INFO: Closing spider (finished)
2021-10-22 15:48:31 [scrapy.extensions.feedexport] INFO: Stored json feed (2 items) in: news.json
```

The json file ("news.json") will be inside the folder from which you called `scrapy crawl news`, unless otherwise specified. Opening this (for example in Firefox for easier visibility), we see that our news-scraping efforts were successful:


```
JSON   Raw Data   Headers
Save   Copy   Collapse All   Headers   d All   Filter JSON
▼ 0:
  ▼ title:   "Russia's Covid nightmare driven by vaccine rejection"
  ▼ text:    "\nNear the darkened entrance of Hospital Number One in the city of Vologda in
             them had not been vaccinated.\nLives are being lost here in this region every
             deaths so far. These are record numbers for Russia, making it the worst-hit in
             cases and deaths is a lack of trust in the Sputnik V and other Russian-made co
             Vologda hospital as elsewhere in the country, there is a real concern about th
             patients we have.\" \nAt one of the vaccine centres set up in a large sports h
             criticised those opposed to the vaccine who have taken to social media to spre
             shocked by the scale of anti-vaccination sentiment in the region, highlighting
             from many sources including the authorities in the region. People were afraid
             main cemetery there continues to be a massive increase in the number of burial
▼ 1:
  ▼ title:   "Crime-hit Chicago grapples with police vaccine revolt"
  ▼ text:    "\nA standoff in Chicago, where the head of the city's largest police union is
             last. \nJust as violent crimes have risen, though, thousands of the city's pol
             exemption. By the end of this year, all employees must be vaccinated. \nNearly
             Chicago's police force dangerously depleted. \nDuring a CNN town hall on Thurs
             leading cause of death for police in the US, according to the Officer Down Mem
             cause combined. \n\"This virus is no different than the gunfire we take as cop
             public.\n\"It's hard to understand,\" said Art Acevedo, who has served as chie
             been vaccinated,\" he said. \"In my business, we call that a clue.\" \nMr Acev
             members to defy the city's requirement to report their vaccination status. \n\
             has suggested the vaccine requirements are an illegal violation of privacy. \n
             order. \nAccording to Mr Catanzara, as many as 50% of Chicago's 13,000 officer
             Villaneuva, who oversees the largest sheriff's department in the country, has
```


Of course, we can enrich our spider by collecting more (or more specific) data. But this should give you a good start for programming a Scrapy spider.