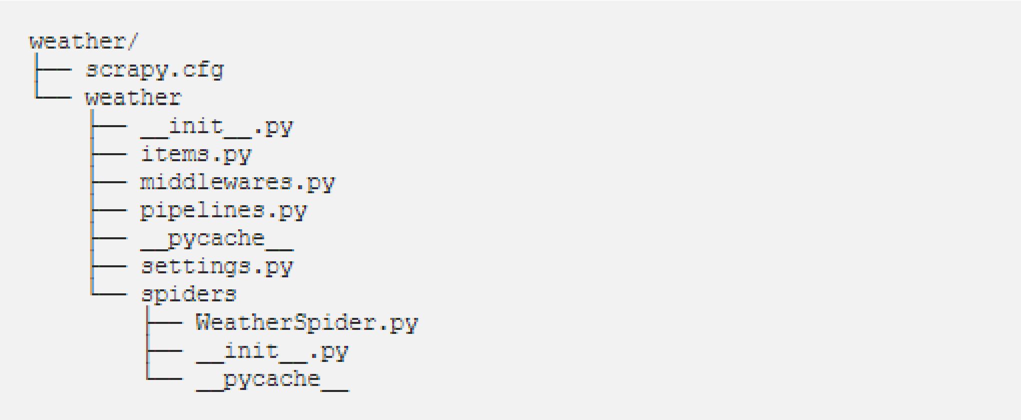
Tool	Tool Description	Tested Version
Python	Python (Tested version with below tools : 3.7.8)	3.7.8
Elasticsearch	Distributed, search and analytics engine	7.9.3
Kibana	Elasticsearch Index visualization	7.9.3
Longstash ***optional**	Load Bulk data into elasticsearch	
Scrapy	Scrape framework	2.4.1
ScrapyElasticSearch	Scrapy pipeline which allows you to store scrapy items in Elastic Search.	0.9.2

Documentation for Scrapy Spiders :

- https://towardsdatascience.com/web-scraping-with-scrapy-practical-understanding-2fbdae337a3b
- https://devhints.io/xpath
- https://www.simplified.guide/scrapy/scrape-table

Scrapy is an application framework for crawling web sites and extracting structured data which can be used for a wide range of useful applications, like data mining, information processing or historical archival.

Scrapy framework – Project build Path:



```
weather/
├── scrapy.cfg
└── weather
    ├── __init__.py
    ├── items.py
    ├── middlewares.py
    ├── pipelines.py
    ├── __pycache__
    ├── settings.py
    └── spiders
        ├── WeatherSpider.py
        ├── __init__.py
        └── __pycache__
```

The main components of Scrapy framework are:

- Items.py
- settings.py
- Spiders.*myspider.py*

The item.py is holds the description of the item which we work on. Generally it is a python class and contains the structure of the object at field level. A typical example is showed below with the definition of each necessary field of our project:

```
class MyItem(scrapy.Item):
id           = scrapy.Field()
source       = scrapy.Field()
time         = scrapy.Field()
timecrawl    = scrapy.Field()
temperature  = scrapy.Field()
humidity     = scrapy.Field()
wind         = scrapy.Field()
barometer    = scrapy.Field()
yetos        = scrapy.Field()
direction    = scrapy.Field()
city         = scrapy.Field()
```

The settings.py file keeps the configuration of the Scrapy project such as the encoding definition, the settings for pipelines or middlewares. The purpose of pipelines and middlewares is actually the next steps when Scrapy engine retrieve the data from a web server. A typical example may be the data store into a repository (database, Elasticsearch, etc.). Below you can find the important points of settings.py file for store data into Elasticsearch.

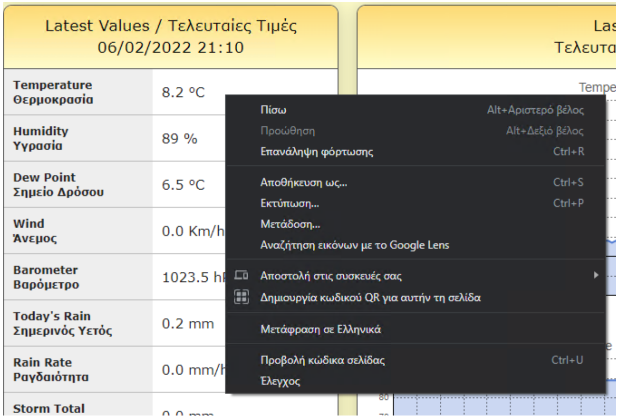
```
# Scrapy settings for my_project project

FEED_EXPORT_ENCODING = 'utf-8'


ITEM_PIPELINES = {
    'my_porject.scrapyelasticsearch.ElasticSearchPipeline': 500
}
ELASTICSEARCH_SERVERS = ['localhost']
ELASTICSEARCH_INDEX = 'weather'
ELASTICSEARCH_INDEX_DATE_FORMAT = ''
ELASTICSEARCH_TYPE = 'items'
ELASTICSEARCH_UNIQ_KEY = 'id'   # Custom unique key
```

We should describe the Elasticsearch engine connection – configuration and we use the `ElasticSearchPipeline` method from `scrapyelasticsearch.py` file which is the default code for sending data into Elasticsearch.

Finally the spiders.myspider.py keeps all the code for the transformation of the web site data (field – information) into our (pre)defined item (item.py). You can find an example at next segment. What i would like to note here is the html element transformation process. Each web site keeps data in different way such as html tables etc. No matter what html element is, we need the (html) Xpath information. An easy way to find the Xpath is to mark the field that you interested in (let's say the temperature) and with right click choose the «'Ελεγχος» option.

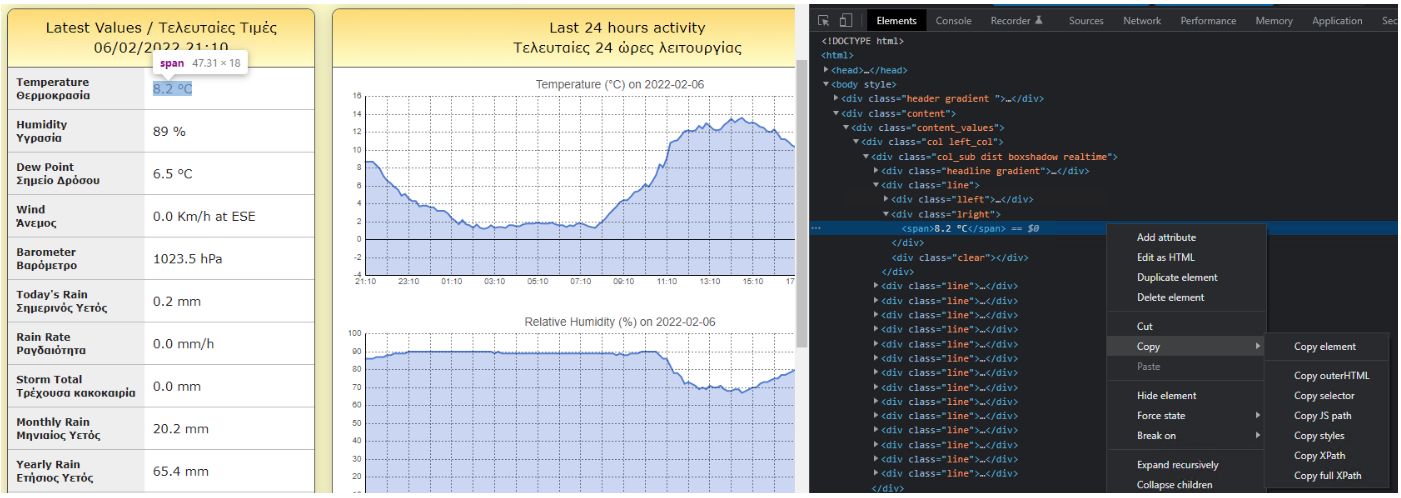


Latest Values / Τελευταίες Τιμές
06/02/2022 21:10

Temperature Θερμοκρασία	8.2 °C
Humidity Υγρασία	89 %
Dew Point Σημείο Δρόσου	6.5 °C
Wind Άνεμος	0.0 Km/h
Barometer Βαρόμετρο	1023.5 hP
Today's Rain Σημερινός Υετός	0.2 mm
Rain Rate Ραγδαιότητα	0.0 mm/h
Storm Total	0.0 mm

Las
Τελευτα
Tempe
Πίσω Alt+Αριστερό βέλος
Προώθηση Alt+Δεξιό βέλος
Επανάληψη φόρτωσης Ctrl+R
Αποθήκευση ως... Ctrl+S
Εκτύπωση... Ctrl+P
Μετάδοση...
Αναζήτηση εικόνων με το Google Lens
Αποστολή στις συσκευές σας
Δημιουργία κωδικού QR για αυτήν τη σελίδα
Μετάφραση σε Ελληνικά
Προβολή κώδικα σελίδας Ctrl+U
Έλεγχος
80

Then at the right side of the screen we can see the html element information. With right click, we can copy the Xpath value of the element.



Latest Values / Τελευταίες Τιμές
06/02/2022 21:10

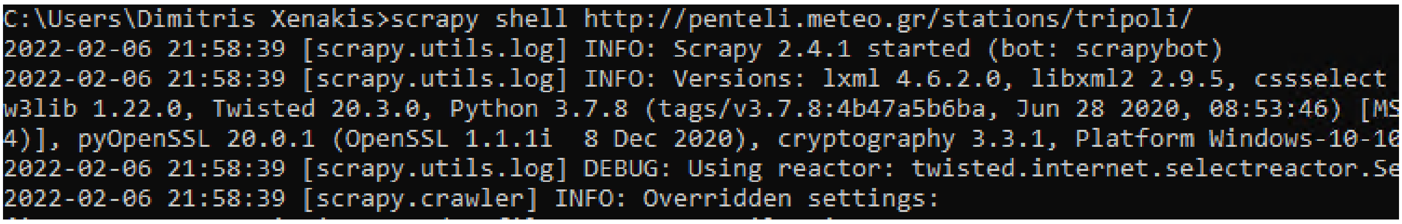
Temperature Θερμοκρασία	8.2 °C
Humidity Υγρασία	89 %
Dew Point Σημείο Δρόσου	6.5 °C
Wind Άνεμος	0.0 Km/h at ESE
Barometer Βαρόμετρο	1023.5 hPa
Today's Rain Σημερινός Υετός	0.2 mm
Rain Rate Ραγδαιότητα	0.0 mm/h
Storm Total Τρέχουσα κακοκαιρία	0.0 mm
Monthly Rain Μηνιαίος Υετός	20.2 mm
Yearly Rain Ετήσιος Υετός	65.4 mm

```
<!DOCTYPE html>
<html>
▶<head>…</head>
▼<body style>
  ▶<div class="header gradient ">…</div>
  ▼<div class="content">
    ▼<div class="content_values">
      ▼<div class="col left_col">
        ▼<div class="col_sub dist boxshadow realtime">
          ▶<div class="headline gradient">…</div>
          ▼<div class="line">
            ▶<div class="lleft">…</div>
            ▼<div class="lright">
...             <span>8.2 °C</span> == $0
              </div>
              <div class="clear"></div>
            </div>
          ▶<div class="line">…</div>
          ▶<div class="line">…</div>
          ▶<div class="line">…</div>
          ▶<div class="line">…</div>
          ▶<div class="line">…</div>
          ▶<div class="line">…</div>
          ▶<div class="line">…</div>
          ▶<div class="line">…</div>
          ▶<div class="line">…</div>
          ▶<div class="line">…</div>
          ▶<div class="line">…</div>
          ▶<div class="line">…</div>
          ▶<div class="line">…</div>
          ▶<div class="line">…</div>
        </div>
```

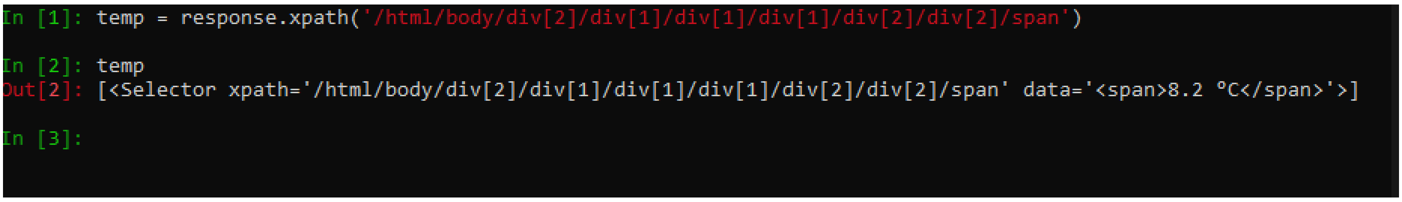
When we have our xpath value we can retrieve the information data (temperature). We can check that we receive the right information through a command line running the commands below.

```
scrapy shell http://penteli.meteo.gr/stations/tripoli/
temp = response.xpath('/html/body/div[2]/div[1]/div[1]/div[1]/div[2]/div[2]/span')
```

At response.xpath method we paste the value we copy from web browser.



```
C:\Users\Dimitris Xenakis>scrapy shell http://penteli.meteo.gr/stations/tripoli/
2022-02-06 21:58:39 [scrapy.utils.log] INFO: Scrapy 2.4.1 started (bot: scrapybot)
2022-02-06 21:58:39 [scrapy.utils.log] INFO: Versions: lxml 4.6.2.0, libxml2 2.9.5, cssselect
w3lib 1.22.0, Twisted 20.3.0, Python 3.7.8 (tags/v3.7.8:4b47a5b6ba, Jun 28 2020, 08:53:46) [MS
4)], pyOpenSSL 20.0.1 (OpenSSL 1.1.1i  8 Dec 2020), cryptography 3.3.1, Platform Windows-10-10
2022-02-06 21:58:39 [scrapy.utils.log] DEBUG: Using reactor: twisted.internet.selectreactor.Se
2022-02-06 21:58:39 [scrapy.crawler] INFO: Overridden settings:
```



```
In [1]: temp = response.xpath('/html/body/div[2]/div[1]/div[1]/div[1]/div[2]/div[2]/span')

In [2]: temp
Out[2]: [<Selector xpath='/html/body/div[2]/div[1]/div[1]/div[1]/div[2]/div[2]/span' data='<span>8.2 °C</span>'>]

In [3]:
```

At next page you can find a real example.

You can find also plenty real examples here.

Let’s say we need to create a new scrapy project for weathersite.gr. Then we should follow the steps below:

- Create the buildpath with the command: `scrapy startproject weathersite`
  Now we have create the above buildpath of our project

- Inside spiders folder generate the Spider with the command :
  `scrapy genspider -t basic weathersite_spider weathersite.com`

- The configuration is ready, we can run the spider with the command (under spider folder):
  `scrapy crawl weather_spider`

Example:

Let’s say we need to scrape data for meteo.gr (weather station) and load them into an elasticsearch index (we assume that we have already create an index at elasticsearch with name “weather”).

We create a scrapy project with command: `scrapy startproject meteo`
We move into spiders folder and run : `scrapy genspider -t basic meteo_spider meteo.gr`

First of all we need to describe the fields of the item we scrape. So we have to configure the **items.py**

```
import scrapy
class MeteoItem(scrapy.Item):
    id             = scrapy.Field()
    source         = scrapy.Field()
    time           = scrapy.Field()
    timecrawl      = scrapy.Field()
    temperature    = scrapy.Field()
    humidity       = scrapy.Field()
    wind           = scrapy.Field()
    barometer      = scrapy.Field()
    yetos          = scrapy.Field()
    direction      = scrapy.Field()
    city           = scrapy.Field()
```

Then we should configure the **setting.py**

```
ITEM_PIPELINES = {
    'meteo.scrapyelasticsearch.ElasticSearchPipeline': 500
}
ELASTICSEARCH_SERVERS = ['localhost']
ELASTICSEARCH_INDEX = 'weather'
ELASTICSEARCH_INDEX_DATE_FORMAT = ''
ELASTICSEARCH_TYPE = 'items'
ELASTICSEARCH_UNIQ_KEY = 'id'  # Custom unique key
```

The `scrapyelasticsearch.ElasticSearchPipeline` is a class which needs the classes-files below (you should keep them at the same folder with settings.py) :

- scrapyelasticsearch.py
- transportNTLM.py

The final step is to configure the **meteo_spider.py** under spiders folder (the name of the file comes from the name we state at :
`scrapy genspider -t basic meteo_spider meteo.gr` at previous step.

meteo_spider.py file:

```
import scrapy
import re
import scrapy.spiders
from ..items import MeteoItem
import datetime

class MeteoSpiderSpider(scrapy.Spider):
    name = 'meteo_spider'
    allowed_domains = ['http://penteli.meteo.gr/stations/tripoli/']
    start_urls = ['http://penteli.meteo.gr/stations/tripoli/']

    def parse(self, response):
        i=0
        table = response.xpath('//*[@id="table1"]')
        rows = table.xpath('//tr')
        source       = 'meteo.gr'
        city         = 'Tripoli'
        crawldate    =  datetime.datetime.now()
        timestr      = rows[2].xpath('td//text()')[3].extract()
        datepart     = timestr[-9:].strip()
        timepart     = timestr[2:-9].strip()
        datetimep    = datepart+' '+timepart
        time         = datetime.datetime(int('20'+datetimep[6:8]), int(datetimep[3:5]),
int(datetimep[0:2]),int(datetimep[-5:-3]),int(datetimep[-2:])) -   datetime.timedelta(hours=3,
minutes=0)
        temperature  = float(rows[3].xpath('td//text()')[4].extract()[0:-2])
        humidity     = float(rows[4].xpath('td//text()')[4].extract()[:-1])
        windends     = (rows[6].xpath('td//text()')[4].extract()).find(" ")
        winddire     = (rows[6].xpath('td//text()')[4].extract()).find("at")
        wind         = float(rows[6].xpath('td//text()')[4].extract()[0:windends])
        barends      = rows[7].xpath('td//text()')[4].extract().find(" ")
        barometer    = float(rows[7].xpath('td//text()')[4].extract()[:barends])
        yetos        = float(rows[8].xpath('td//text()')[3].extract()[:-3])
        direction    = rows[6].xpath('td//text()')[4].extract()[winddire+3:]
        id           = source+' '+datetimep
        item = MeteoItem()
        item["id"]          = id
        item["source"]      = source
        item["time"]        = time
        item["timecrawl"]   = crawldate
        item["temperature"] = temperature
        item["humidity"]    = humidity
        item["wind"]        = wind
        item["barometer"]   = barometer
        item["yetos"]       = yetos
        item["direction"]   = direction
        item["city"]        = city
        yield item


    def start_requests(self):
        yield scrapy.Request('http://penteli.meteo.gr/stations/tripoli/', self.parse)
```

Now if we run the command : `scrapy crawl meteo_spider` the crawl data transform into items and the items move into our index

Dimitris Xenakis
Feel free to contact at
d.xenakis@outlook.com