1. **Setting Environment**

Library dependencies :

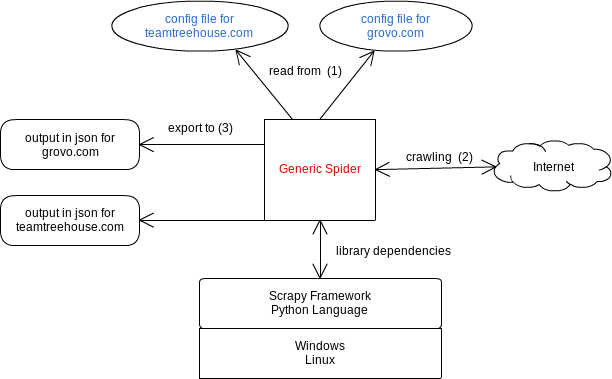
* Python version 2.7.3
* Scrapy 0.24.2

Module dependencies:

sudo apt-get install python-dev

* sudo pip install demjson
* sudo pip install python-dateutil
* sudo pip install service\_identity
* sudo pip install selenium
* sudo pip install html2text
* sudo pip install langdetect
* sud pip install w3lib
* sudo pip install cssselect
* sudo pip install queuelib
* sudo pip install tinys3
* sudo pip install twisted
* sudo pip install cryptography
* sudo pip install PyPDF2

1. **Architect of Generic Library (Generic Spider ):**



1. **Terms :**

* **baseUrl** : the url of website that will be crawled.
* **levels** : Each course need access some urls to get informations. In general, these urls has a form of tree. Here is a config for this part :

Example :

“levels” : {

“1” : config\_for\_level\_1, // detail see 3 How to config to evaluate a value:

“1\_1”:config\_for\_level\_1\_1 // detail see 3 How to config to evaluate a value:

}

* **level “0”** : the first page that display information to extract course links the first page that
* **level “1”** : the first page that display information for a course.
* **level “1.1” or “1.2”** : the page that its url got from the level “1” . the results can be a list of HtmlReponse
  + source information can be a HtmlResponse (a full web page for a url)
  + source information can be a part of HtmlResponse (extracted from a web page)
* **xpath :** see [selectors](http://doc.scrapy.org/en/latest/topics/selectors.html)

Example :

“xpath” : “//a”

“xpath” : [‘//a’, ’//div’] // stop and return when got a result not **NULL**

“xpath” : {

“value” : [‘//a’,’//div’],

“operator”:’or’ // stop and return when got a result no **NULL**

}

“xpath” : {

“value” : [‘//a’,’//div’],

“operator”: “and” // this will concatenate the result evaluated by each xpath

}

* **re :** see [selectors](http://doc.scrapy.org/en/latest/topics/selectors.html)

Example :

“re” : “\\d+.?\\d\*”

* **python**

Example :

“python” : “convert\_date(data)"

**Note :**

* to evaluate a field, the process will start by evaluate on source information by the following order : ***“xpath” -> “re” -> “python”***
* if the site of is difficult or complex to get data (example : complexe click selenium , get data form javascript or multi html structure) we can write a python code for this web site and place it into the following folder : **scraper-framework/scrapy\_balloons/supportclients**
* **output\_config** : definition of the structure of product json for output.
* **other\_fields** : // detail see 3 How to config to evaluate a value:
* **selenium\_config:** for some web site has ajax loading or scroll down, we can use selenium to get data.

“selenium\_config” : {

“driver” : “chrome” // values possible are : “phantomjs | chrome | firefox”

"wait\_until\_available": {

"xpath":"//div[contains(@class,'drvisible')]",

"sleep":2

}

}

* other events config see file scrapy\_balloons/selenium\_api.py
* a selenium function can be called from the callback function (extractor Rules) or in python function
* **extractor\_rules :** see in detail[link\_extractors](http://doc.scrapy.org/en/latest/topics/link-extractors.html)

"extractor\_rules": {

"rules": [

"Rule(lxml(allow=(‘courses’')),callback=slm.wait\_until\_available)",

"Rule(lrgl(allow=('courses')),callback='parse\_product')"

]

}

* **pre\_run** : the scrape execute this step firstly before gather other data for courses. We will use this to avoid to scrap multiple times for the same data, for example :
* website have courses that the price is paid by month (so the price is the same for every course)
* website have all product events that come from url (each course will search its product\_event corresponding ).

Example :

"pre\_run": {

"all\_events": {

"start\_url": "https://f5.com/education/training/schedule",

"extractor\_rules": {

"rules": [

"Rule(lxml(allow=('.\*'), restrict\_xpaths=\"//span[@id='dnn\_ContentPane']//ul//li/a\"), follow=True)",

"Rule(lxml(allow=('schedule-plain',), tags=('iframe'), attrs=('src'), ), callback=balloon\_spider.pre\_run\_service.parse)"

]

}

},

"price\_info": {

"xpath": “//div”,

”start\_url": "https://teamtreehouse.com/subscribe/plans?trial=yes"

}

}

* In the above example, we have two configs:
* configs for “all\_events” need a “start\_url” and “extractor\_rules” : The response for key “all\_events” will be a list of HtmlResponse
* config for “price\_info” need a “start\_url” and a xpath: The response for this key will be a Selector.
* to access the response for a key in “pre\_run” config:

in config file :

“level” : “price\_info”

or

“level” : “all\_events”

in python code :

from scrapy\_balloons.spiders.balloon import balloon\_spider

response = balloon\_spider.pre\_run\_service.responses\_received\_by\_key['all\_events']

* **settings :** override settings scrapy from config file. Here is a example:

"settings": {

"DOWNLOAD\_DELAY": 3,

"DOWNLOADER\_MIDDLEWARES": {

"scrapy\_balloons.useragent.RandomUserAgentMiddleware":500,

"handle\_httpstatus\_list": [500,300]

},

}

* **pre\_filters :** filter or filters\_not responses before create course product

- filters : take only response, that passe all python function

- filters\_not : ignore response, that passe at least one python function

Example :

"pre\_filters": {

"filters\_not": [

{

"python": "'taiwan.html' in response.url",

"level": "1"

},

{

"python": "'tw.html' in response.url",

"level": "1"

}

]

},

variable available :

response : response corresponding with level passed in config

source : dictionary contains responses of all level for one course

* **post\_filters :** filter or filters\_not after create course product

- filters : take only course, that passe all python function

- filters\_not : ignore course, that passe at least one python function

variable available :

item : type is Product

Example :

"post\_filters": {

"filters\_not": [

{

"python": "'http://www.tradingacademy.com/education/' == item['product\_url']"

}

]

}

* **post\_interceptors :** allow modify the course

variable available :

item : type is Product

Example :

"post\_interceptors": [

{

"python": "item['product\_events'] = [event for event in item['product\_events'] if event['location\_display']]"

}

1. **How to config for a field :**

* **“field\_name” : constant**  //The value is a constant, str | int | bool and etc ..

It mean the value will be the same in the output json file

* **“field\_name”** :

{

**“xpath” :** “…” // optional

**“re”** : “...” // optional

**“python”** : “ ..” // optional

**“type”** : “....” //optional, type will be a object, list of object.

**“level”** : “...” // optional, value choose from any key in the levels config or pre\_run config.

if missing. If not specified the value will be the first level of product, means **“0” or “1”**

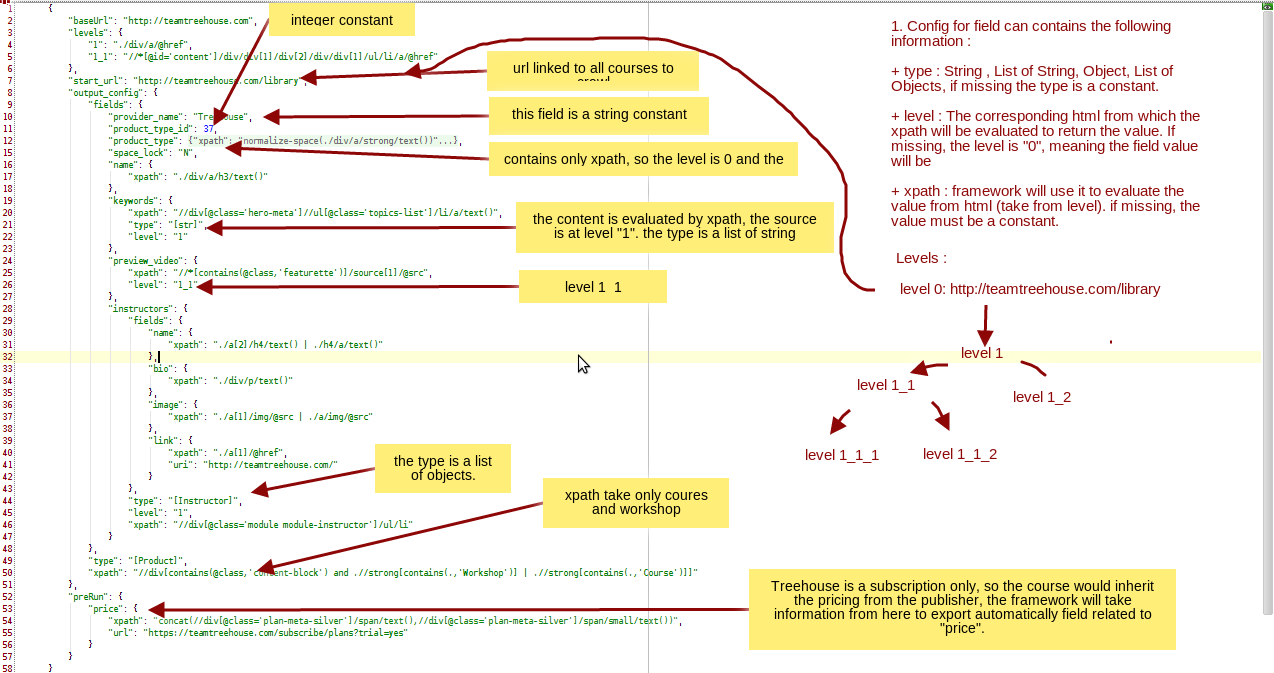
}

* **“field\_name” : [ {sub\_1} , {sub\_2} ] //** the value can be a list of sub config. The spider will iterate each sub\_config in the list and process to extract value. If it found the result not Null, it will stop and return immediately.

**Note : at least one of following key “xpath” | “python” | “re” must be present**

1. **Example :**

**Here is a basic example :**



**6. How to debug in an interactive mode**

C*ommand line :*

*scrapy crawl web\_scraper -a config\_file={{config\_file\_name}} -a limit={{limit}} -o {{output\_data\_name}} -a debug={{field\_name,set\_trace}}*

**field\_name=\*** : The spider print the info of all field that there value is null for each course.

**field\_name=description** : the spider will print the info for only field ‘description’ that its value is null for each course

**set\_trace= true | false :** the spider will run the pdb.set\_trace() or not when a null condition satisfied

Example :

* *scrapy crawl web\_scraper -a config\_file=config/treehouse\_config.json --nolog -a limit=5 -o output/treehouse\_output.json -a debug=\*,true*
* *scrapy crawl web\_scraper -a config\_file=config/treehouse\_config.json --nolog -a limit=5 -o output/treehouse\_output.json -a debug=duration\_filter,true*
* *{{debug}} : the value has a structure field\_name,enable\_pdb\_settrace*

**7. How** t**o run a library**

C*ommand line :*

*scrapy crawl web\_scraper -a config\_file={{config\_file\_name}} -a limit={{limit}} -o {{output\_data\_name}}*

* {{config\_file\_name}} : path to config file
* {{output\_data\_name}} : filename of output data.
* {{limit}} : limit no of products in output.

Command :

* *scrapy crawl web\_scraper -a config\_file=config/treehouse\_config.json -o treehouse\_output.json*

**8. How to run multiple library :**

There is a python script will find all config files in the folder config and schedule to run n config files in a same time

go to folder : **scraper-framework/tools**

Command : *run\_all.py [-h] [-i CONFIG\_PATH] [-o OUTPUT\_PATH] [-l LIMIT] [-n PROCESS\_NUMBER] [-s3 S3\_PATH] [-a ALLOW] [-d DENY] [-e EMAIL]*

*optional arguments:*

-**h**  show this help message and exit

**-i** folder path where the config files are [Required]

**-o** output folder where the output file will be stored. Without it,current epoch time as folder name will be created [Optional]

**-l** limit the courses number to be scraped for each provider [Optional]

**-n** number of providers to run in parallel [Optional]

**-a** path to text file. Only providers has config file names in this text file will be scraped [Optional]

**-d** path to text file. All providers has config file names in this text will be ignored to be scraped. [Optional]

-**s3** : s3 path. When a provider is finished, the scraper will compress the output json and upload the compressed file to s3. The credential information to connect to s3 is saved in : scrapy\_balloons/settings.py [Optional]

- **e** : Email address that will receive notifications and report. [Optional]

**Command :**

sudo python tools/run\_all.py -i config -n 2 -l 2 -o output -e xyz@gmail.com -s3 test -products/test\_upload

Output :

A folder with the name **output** will be created at current path.

in the folder **output** there are the following files :

* folder **output :** json output of all providers
* file summary **summary**.csv : Report file in csv format
* file summary **summary**.json : Report file in json format
* folder **logs** : log files of all providers.

skilledup server : 54.174.14.128

go to :

/opt1/scraper-framework/05\_03

apache2 server : http://54.174.14.128/scraper/