Part 4 of scrapy walkthrough :

1. To create a new generic spider, simply run the **genspider** command inside the spider folder:

*# syntax is --> scrapy genspider <name\_of\_spider> <website>*   
$ scrapy genspider (immospider) (immoweb)

1. **Note:** If you would like to use IPython as your Scrapy shell (much more powerful and provides smart auto-completion and colorized output), then make sure you have ipython installed:

pip3 install ipython

1. And then edit your scrapy.cfg file like so:

*## scrapy.cfg*  
[settings]  
default =xxx.settings  
shell = ipython

1. In your immospider file, set the start url as : ["https://www.immoweb.be/en/search/house-and-property/for-sale?countries=BE"]
2. To open Scrapy shell use this command:

$ scrapy shell

The first thing we want to do is fetch the main products page in Scrapy shell.

$ fetch ('url of page where you look for data’)

As we can see, we successful retrieve the page from 'https://www.immoweb.be/en', and Scrapy shell has automatically saved the HTML response in the response variable.

$ response

(You should get a 200 response)

1. Then you can choose an element from the HTML inspect page to scrape.

Example : the property element :

$ response.css('article.card.card--result.card--xl')

-🡪 Returns all the properties on the page.

1. Then we can use the get function to return the first result of the page.

$ response.css('a.card\_\_title-link').get()

1. We can also create a variable and put the response in it.

$ properties = response.css('article.card.card--result.card--xl')

$ property = properties[0]

🡪 output = the first property

1. How many are there ?

$ len(properties)

🡪 output = the number of properties on the first page

1. Show the URL :

property.css('h2 a').attrib['href']

* scrapy crawl immospider : start the scraping

class ImmospiderSpider(scrapy.Spider):

    name = "immospider"

    allowed\_domains = ["www.immoweb.be"]

    start\_urls = ["https://www.immoweb.be/en/search/house-and-apartment/for-sale?countries=BE&page=1&orderBy=relevance"]

    def parse(self, response):

        properties = response.css('article.card.card--result.card--xl')

        for property in properties:

            yield {

                            'name': property.css('h2 a::text').get(),

                            'url': property.css('h2 a').attrib['href']

                        }

        pass

If the selector `response.css('li.pagination\_\_item a.pagination\_\_link.pagination\_\_link--next.button.button--text.button--size-small::attr(href)').get()` does not return any output, it is possible that the element you are trying to select is not present in the HTML response or is loaded dynamically through JavaScript.

You can try to use your browser's developer tools to inspect the HTML code and verify that the element you are trying to select is present in the response. Also, you can check if the HTML response is loaded dynamically using JavaScript by disabling JavaScript in your browser and reloading the page to see if the element is still present.

If the element is loaded dynamically, you may need to use Scrapy's built-in support for handling JavaScript through the `Splash` or `Puppeteer` middleware. You can find more information about these middleware and how to use them in the Scrapy documentation.