

Explain **LinkExtractor**, **Paging** and **Form Request**.

Objective

In this tutorial, we will extract game data from <http://store.steampowered.com>. The starting point is this link http://store.steampowered.com/search/?sort_by=Released_DESC, which contain collection of more than 30000 games title.

	Naklua VR Demo	4 Nov, 2017	Free Demo
	Prison Chainball Massacre	Nov 2017	
	Bryan Audley's Numbers	Nov 2017	
	Artifact Quest 2	4 Nov, 2017	-40% \$4.99 \$2.99
	The King's Heroes - Official Guide	4 Nov, 2017	\$4.99

showing 1 - 25 of 33308

1 2 3 ... 1333 >

From here we could extract detail information for each game

All Games > Casual Games > Artifact Quest 2

Artifact Quest 2

Community Hub



Travel together with Jess to a tropical island and find ancient treasures!

ALL REVIEWS: 1 user reviews

RELEASE DATE: 4 Nov, 2017

DEVELOPER: Best Friend Games

PUBLISHER: Best Friend Games

Popular user-defined tags for this product:

Indie Casual +



Extract Game Links from First Page

Start a new project call **steam**

```
scrapy startproject steam
```

Go inside **steam** folder and create crawl spider with name **game**

```
scrapy genspider -t crawl game steampowered.com
```

Change `start_urls` to http://store.steampowered.com/search/?sort_by=Released_DESC.

Open a detail game page, for example http://store.steampowered.com/app/718080/Artifact_Quest_2/.

You will see that every game url contain "/app/". So let change the `allow` parameter of `LinkExtractor`. Another thing with `LinkExtractor` is we do not want to follow link because this is detail game page so we want to get data only, not follow other link. So, we have spider.

```
# -*- coding: utf-8 -*-
import scrapy
from scrapy.linkextractors import LinkExtractor
from scrapy.spiders import CrawlSpider, Rule

class GameSpider(CrawlSpider):
    name = 'game'
    allowed_domains = ['steampowered.com']
    start_urls = ['http://store.steampowered.com/search/?sort_by=Released_DESC']

    rules = (
        Rule(LinkExtractor(allow=r'/app/(.+)'), callback='parse_item', follow=False),
    )

    def parse_item(self, response):
        #print url to see if spider is request game link or not
        print response.url
```

Try to crawl with above spider with **--nolog**

```
scrapy crawl game --nolog
```

and we have following result, seem are links we want. Some game need us to verify age before access, so it redirect to verify age form which include **"agecheck"**. We will deal with **"agecheck"** form later.

```
(C:\Users\TAN\Anaconda2) C:\scrapy\try\steam>scrapy crawl game --nolog
http://store.steampowered.com/app/705410/World_War_Party_Game_Of_Trump/?snr=1_7_7_230_150_1
http://store.steampowered.com/app/732550/Woody_Blox/?snr=1_7_7_230_150_1
http://store.steampowered.com/app/610310/Star_Story_The_Horizon_Escape/?snr=1_7_7_230_150_1
http://store.steampowered.com/app/707230/Prison_Chainball_Massacre/?snr=1_7_7_230_150_1
http://store.steampowered.com/app/353370/?snr=1_7_7_230_12
http://store.steampowered.com/app/726170/Hardy_Only_One/?snr=1_7_7_230_150_1
http://store.steampowered.com/app/705210/Cube_Racer/?snr=1_7_7_230_150_1
http://store.steampowered.com/app/722590/Audio_Factory/?snr=1_7_7_230_150_1
http://store.steampowered.com/app/722410/Fallen_Kingdom/?snr=1_7_7_230_150_1
http://store.steampowered.com/app/710550/Cubion/?snr=1_7_7_230_150_1
```

Extract Game Links from All Pages

In above session, we only request for 1 page, but overall we have 1333 pages to deal with.

showing 1 - 25 of 33308

1 2 3 ... 1333



Now we need a way to travel between page. Click to page 2 and we see following url

http://store.steampowered.com/search/?sort_by=Released_DESC&page=2

So seem to travel between page, I just need to allow spider crawl url with contain "page". That it, let modify our spider by adding one more extract link rule. And because the second Rule just allow spider to travel between page, we set `follow=True` and do not need to specify a callback function.

```
# -*- coding: utf-8 -*-
import scrapy
from scrapy.linkextractors import LinkExtractor
from scrapy.spiders import CrawlSpider, Rule

class GameSpider(CrawlSpider):
    name = 'game'
    allowed_domains = ['steampowered.com']
    start_urls = ['http://store.steampowered.com/search/?sort_by=Released_DESC']

    rules = (
        Rule(LinkExtractor(allow=r'/app/'), callback='parse_item', follow=False),
        Rule(LinkExtractor(allow=r'page',), follow=True),
    )

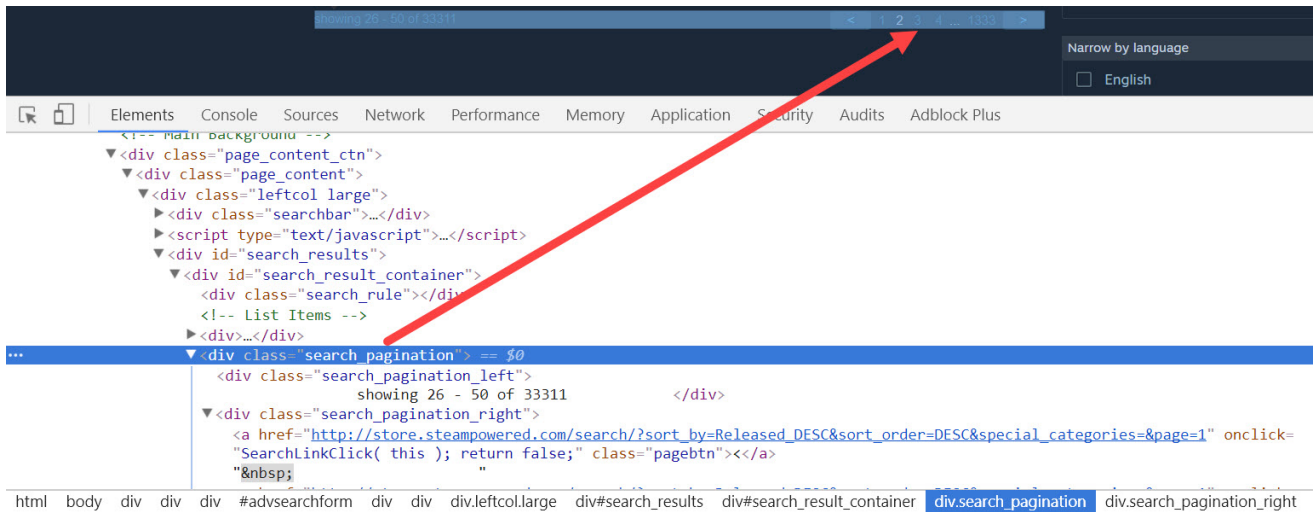
    def parse_item(self, response):
        print response.url
```

Let try to run crawl again and, seem the unlimited of detail url show up as we expected.

```
scrapy crawl game --nolog
```

Need for Speed ?

With above spider, we already could travel between pages, but seem quite slow. Reason is spider find for url contain "page" in whole source page. From Chrome developer tool, try to inspection, we will see that link for page only local inside a `div` tag which have class is `search_pagination`



So one way to speed up the spider is limit finding area with parameter `restrict_css`. Now we have following spider

```
# -*- coding: utf-8 -*-
import scrapy
from scrapy.linkextractors import LinkExtractor
from scrapy.spiders import CrawlSpider, Rule

class GameSpider(CrawlSpider):

    name = 'game'
    allowed_domains = ['steampowered.com']
    start_urls = ['http://store.steampowered.com/search/?sort_by=Released_DESC']

    rules = (

        Rule(LinkExtractor(allow=r'/app/'), callback='parse_item', follow=False),
        # restrict area before search for page link
        Rule(LinkExtractor(allow=r'page', restrict_css='.search_pagination_right')),

    )

    def parse_item(self, response):
        print response.url
```

Try crawl now and you could see how it supper fast now.

Want more speed ? Let's add `restrict_css` when searching for game also

```
# -*- coding: utf-8 -*-
import scrapy
from scrapy.linkextractors import LinkExtractor
from scrapy.spiders import CrawlSpider, Rule

class GameSpider(CrawlSpider):

    name = 'game'
    allowed_domains = ['steampowered.com']
    start_urls = ['http://store.steampowered.com/search/?sort_by=Released_DESC']

    rules = (
        # limit area before search for app link
        Rule(LinkExtractor(allow=r'/app/', restrict_css='#search_result_container'),
        callback='parse_item', follow=False),
        # limit area before search for page link
        Rule(LinkExtractor(allow=r'page', restrict_css='.search_pagination_right')),
    )

    def parse_item(self, response):
        print response.url
```

Dealing with Age Form

Some of above links show up with **agecheck** like this one

http://store.steampowered.com/agecheck/app/249082/?snr=177230150_1158 . Because this type of game make sure you above some age before allow you to access.



The screenshot shows a Steam game page for 'Company of Heroes 2'. A purple 'DLC' badge is visible on the left. Below the game title, there is a row of five small icons representing different game modes or features. A red box highlights a pop-up form that says 'Please enter your birth date to continue:'. The form contains three dropdown menus for day, month, and year, with the values '1', 'January', and '2017' respectively, followed by an 'Enter' button. At the bottom of the page, a small line of text reads: 'This data is for verification purposes only and will not be stored.'

Now to deal with this situation, Scrapy has some thing call `FormRequest` . With `FormRequest` , we need to put in data so we simulate a POST method to server with defined data.



```
▼<form action="http://store.steampowered.com/agecheck/app/249082/" method="post" style="margin:0;padding:0;" id="agecheck_form">
  <input type="hidden" name="snr" value="1_agecheck_agecheck_age-gate">
  
  <h2>Please enter your birth date to continue:</h2>
  ▶<select name="ageDay">...</select>
  ▶<select name="ageMonth">...</select>
  ▶<select name="ageYear" id="ageYear">...</select>
  ▶<a class="btnv6_blue_hoverfade btn_small" href="#" onclick="DoAgeGateSubmit(); return false;">...</a>
</form>
```

So we change `parse_item` as following

```
def parse_item(self, response):

    if '/agecheck/app' in response.url:

        # print response.url

        yield FormRequest(
            url=response.url,
            method='POST',
            formdata={
                'snr': '1_agecheck_agecheck__age-gate',
                'ageDay': '1',
                'ageMonth': '1',
                'ageYear': '1955'
            },
            callback=self.parse_item
        )

    else:
        print response.url
```

Try to run and we will not see "agecheck" link any more.

Understand Game Detail Page

Now we already has response for all detail page. Let's do some investigate to extract detail data for each game. Enter the shell and try to fetch one game page.

```
scrapy shell
```

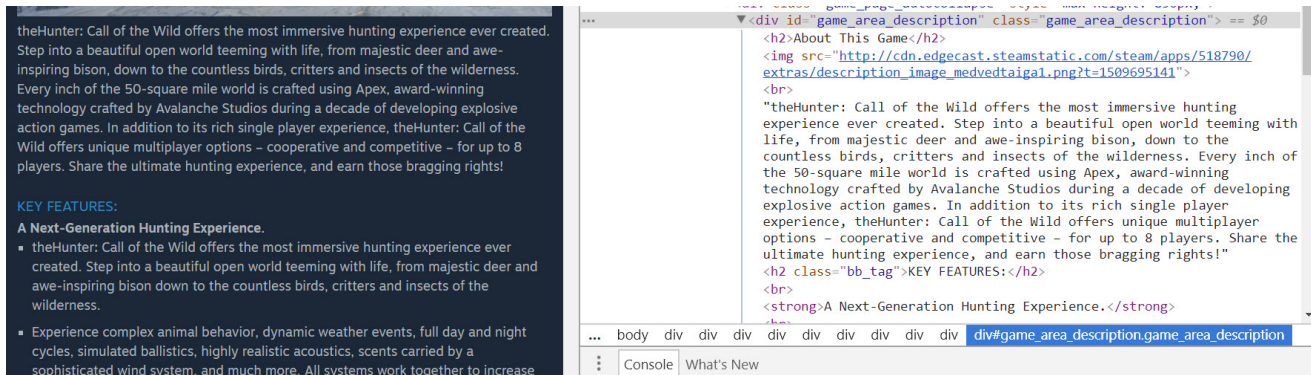
```
fetch('http://store.steampowered.com/app/518790/theHunter_Call_of_the_Wild/')
```

For game title, following selector could extract

```
response.css('.apphub_AppName::text').extract()
```

For game summary, use following selector


```
response.css('#game_area_description::text').extract()
```



Enjoy Your Game Data

That it, let modify our parse item, so we could get some data.

```
def parse_item(self, response):

    if '/agecheck/app' in response.url:

        # print response.url

        yield FormRequest(
            url=response.url,
            method='POST',
            formdata={
                'snr': '1_agecheck_agecheck__age-gate',
                'ageDay': '1',
                'ageMonth': '1',
                'ageYear': '1955'
            },
            callback=self.parse_item
        )

    else:
        title = response.css('.apphub_AppName::text').extract()
        desc = response.css('#game_area_description::text').extract()
        # print title
        # print desc
        yield {'title':title,'desc':desc}
```

That it, enjoy your steam data.