**OTOMOTO URL ANALYSIS**

General url: <https://www.otomoto.pl/>

General url for just cars: <https://www.otomoto.pl/osobowe>

**JavaScript addons in url:**

* Brand (or brands):
  + F.e. citroen: <https://www.otomoto.pl/osobowe/citroen>
  + F.e. citroen and cupra: <https://www.otomoto.pl/osobowe/citroen--cupra>
* Model:
  + F.e. berlingo: <https://www.otomoto.pl/osobowe/citroen/berlingo>
  + F.e. berlingo and c-crosser: <https://www.otomoto.pl/osobowe/citroen/berlingo--c-crosser>
* Example link:
  + <https://www.otomoto.pl/osobowe/citroen/berlingo?search%5Bfilter_enum_damaged%5D=0&search%5Bfilter_enum_fuel_type%5D=petrol&search%5Bfilter_enum_generation%5D=gen-i-1996-2010>
* Filtering:
  + If we have any JS filters we need to add at the end of existing default link:
    - ?search
  + Between every two filters:
    - &search
  + Damages:
    - No damages:
      * %5Bfilter\_enum\_damaged%5D=0
    - car is damaged:
      * %5Bfilter\_enum\_damaged%5D=1
  + Mileage:
    - From -> to:
      * %5Bfilter\_float\_mileage%3Afrom%5D=20000&search%5Bfilter\_float\_mileage%3Ato%5D=150000
    - Just From:
      * %5Bfilter\_float\_mileage%3Afrom%5D=20000
    - Just to:
      * %5Bfilter\_float\_mileage%3Ato%5D=175000
  + Year of production:
    - Weird things are happening here:
    - Just from:
      * To the main link we need to add “/od-2000”
      * Its even before the “?” sign so we can assume that this variable is not recognized by the webpage as a filter – just another thing just as brand
    - Just to:
      * To the filters section we need to add:
      * %5Bfilter\_float\_year%3Ato%5D=2020
    - From -> to:
      * Both things must be adden
  + Fuel type:
    - We will only think about one type of fuel all other things are going to be deleted or modified by pipelines
    - %5Bfilter\_enum\_fuel\_type%5D%5B0%5D=petrol
    - (if we want to add second fuel type we add the same to the link but a number following “B” changes +1
  + Engine power:
    - From to:
      * %5Bfilter\_float\_engine\_power%3Afrom%5D=50&search%5Bfilter\_float\_engine\_power%3Ato%5D=300
    - Just From:
      * %5Bfilter\_float\_engine\_power%3Afrom%5D=50
    - Just To:
      * %5Bfilter\_float\_engine\_power%3Ato%5D=300
  + Shift:
    - Manual shift:
      * %5Bfilter\_enum\_gearbox%5D%5B0%5D=manual
    - Automatic shift:
      * %5Bfilter\_enum\_gearbox%5D%5B0%5D=automatic

**HTML/CSS layout of the page:**

* Whole car – all important data:
  + HTML OBJECT: article
  + CSS CLASS: ooa-yca59n e1oqyyyi0
* Price:
  + HTML OBJECT: h3
  + CSS CLASS: e1oqyyyi16 ooa-1n2paoq er34gjf0
* Year: (it is not that simple) - XPATH
  + HTML OBJECT: dd
  + CSS CLASS: ooa-1omlbtp e1oqyyyi13
  + DATA-PARAMETER: year
* Mileage: -XPATH
  + HTML OBJECT: h3
  + CSS CLASS: ooa-1omlbtp e1oqyyyi13
  + DATA-PARAMETER: mileage

**Going to next page:**

#so there is no link associated with next-page arrow >  
#so we need to somehow craft it  
#we just increase the single number in page link (at the end of it)[if it is here - if not we need to add it]  
#but if we dont check if there is another page we are going to loop the program on the last page  
#so to increase the number in link we need to check if there is an next-

page arrow

First thing we do we don’t want the have a very long xpath so we create a variable whole\_list to store only the code we are interested in. Then we check if an arrow (svg element – described by xpath: whole\_list.xpath("//li[@data-testid='pagination-step-forwards']").get())

even exist if so we for sure know that the another site exist (page + 1).Then we check if we have in our link (response.url()) a part “&page=” if it exist we know that we are not on the first page, then if we indeed aren’t we just change the last letter of URL to a number + 1 and thanks to that we change our page to page + 1. If we are on the first page we just add at the end of the link “&page=2” – we go to the second page.

**To do list:**

* Write a script constructing a starting URL for program – it is complicated process and require a lot of effort to check if the proposed problem solving ideas
* Read pipelines docs, configure a second pipeline only for OTOMOTO data and change settings file in order for it to work, use DropItem function in pipeline
* Check if it works – if so:
* Go back to watching the Scrapy tutorial to learn about configuring fake user agents to make similar program for OLX
* Configure and install github repository and upload the whole project there
* Learn if we can add an user interface for this program using for example tkinter library
* Finish whole video about scrapy especially the part about online services helpful for data science and web scrapying
* START LEARNING PANDAS and use collected data to do data analysis using python – not only excel

**Issues list:**

* Carspider OTOMOTO in changing pages module I need to extract the whole number at the end of the link in order to move between pages >10 – finding on what position there is ‘=’ (WE MUST BE CHECKING IT FROM THE END {rfind() function}) if there is a “&page=” string inside link - SOLVED
* Pipeline not returning items for different spiders – SOLVED
* DropItem function – SOLVED
* Data is not on the same position as the kleineinzeigen data – SOLVED

**Pipeline construction: - DONE**

* Data structure:
  + Year: just an int so – conversion and done
  + Price: deleting whitespaces and then making an integrer – assuming that in prices there aren’t ‘.’ or ‘,’ (if there is there shouldn’t be any problems but it needs to be checked)
  + Mileage: “km” sign (maybe a miles sign too – if there is a bug we can check that), delete whitespaces and then make an int