.vscode/settings.json

{ "python.pythonPath": ".venv/bin/python3.7"

"python.pythonPath": ".venv/bin/python"

}

housingWebScraper/housingWebScraper/spiders/trulia\_spider.py

|  |
| --- |
| import scrapy |
|  |  |  |
|  |  | class StatesSpider(scrapy.Spider): |
|  |  | def extract\_with\_css(response, query): |
|  |  | return response.css(query).get(default='').strip() |
|  |  |  |
|  |  | class TruliaSpider(scrapy.Spider): |
|  |  | name = 'trulia' |
|  |  | start\_urls = [ |
|  |  | 'https://www.trulia.com/property-sitemap/', |
|  |  | ] |
|  |  |  |
|  |  | def parse(self, response): |
|  |  | for state in response.css('ul.all-states > li'): |
|  |  | counties\_page\_url = extract\_with\_css(state, 'a.clickable::attr("href")') |
|  |  | if counties\_page\_url is not None: |
|  |  | yield response.follow(counties\_page\_url, self.parse\_counties) |
|  |  |  |
|  |  | def parse\_counties(self, response): |
|  |  | for county in response.css('ul.all-counties > li'): |
|  |  | zipcodes\_page\_url = extract\_with\_css(county, 'a.clickable::attr("href")') |
|  |  | if zipcodes\_page\_url is not None: |
|  |  | yield response.follow(zipcodes\_page\_url, self.parse\_zipcodes) |
|  |  |  |
|  |  | def parse\_zipcodes(self, response): |
|  |  | for zipcode in response.css('ul.all-zip-codes > li'): |
|  |  | streets\_page\_url = extract\_with\_css(zipcode, 'a.clickable::attr("href")') |
|  |  | if streets\_page\_url is not None: |
|  |  | yield response.follow(streets\_page\_url, self.parse\_streets) |
|  |  |  |
|  |  | def parse\_streets(self, response): |
|  |  | for street in response.css('ul.all-streets > li'): |
|  |  | properties\_page\_url = extract\_with\_css(street, 'a.clickable::attr("href")') |
|  |  | if properties\_page\_url is not None: |
|  |  | # If there is only a single property record for a street, the property url is in href instead of the properties sitemap url |
|  |  | if "trulia.com/p/" not in properties\_page\_url: |
|  |  | yield response.follow(properties\_page\_url, self.parse\_properties) |
|  |  | else: |
|  |  | #yield response.follow(properties\_page\_url, self.parse\_property) |
|  |  | yield { |
|  |  | 'url': properties\_page\_url |
|  |  | } |
|  |  |  |
|  |  | def parse\_properties(self, response): |
|  |  | for property in response.css('ul.all-properties > li'): |
|  |  | property\_page\_url = extract\_with\_css(property, 'a.clickable::attr("href")') |
|  |  | yield { |
|  |  | 'name': state.css('a.clickable::text').get(), |
|  |  | 'url': state.css('a.clickable::attr("href")').get(), |
|  |  | 'url': property\_page\_url |
|  |  | } |
|  |  |  |
|  |  | ''' |
|  |  | next\_page = response.css('li.next a::attr("href")').get() |
|  |  | if next\_page is not None: |
|  |  | yield response.follow(next\_page, self.parse) |
|  |  | ''' |
|  |  | def parse\_property(self, response): |
|  |  | yield { |
|  |  | 'name': extract\_with\_css(response, 'a.clickable::text'), |
|  |  | 'url': extract\_with\_css(response, 'a.clickable::attr("href")'), |
|  |  | } |
|  |  |  |
|  |  |  |