Cheatsheet Scrapy & Selenium

Wichtige Befehle auf einen Blick! (Dokumentation: https://docs.scrapy.org/en/latest)

Institut für Wirtschaftsinformatik
Universität St.Gallen

Scrapy Projekt starten

1. Über das Terminal in den Ordner navigieren, wo das Scrapy Projekt abgelegt werden soll (s. Cheatsheet Terminal/Cmd)
2. Über das Terminal ein Scrapy Projekt starten:
   «scrapy startproject *projektname*»
3. Über das Terminal in den Projektordner gehen und dort eine Spider generieren:
   «scrapy genspider *spidername zieldomain*»
4. Auf dem lokalen in den Projektordner gehen, dort den Ordner Spiders auswählen und die Datei *spidername*.py mit Sublime öffnen
5. In *spidername*.py sollte ein Spider-Template im nebenstehenden Format angelegt sein:

Beispiel (www. bionetz.ch) - Scrapy ohne Selenium

```
import scrapy

class Biopider(scrapy.Spider):
    name = 'bio'
    allowed_domains = ['bionetz.ch']
    start_urls = ['https://bionetz.ch/adressen/detailhandel/bio-fachgeschaefte.html]

    def parse(self, response):
                    single_etikette = response.xpath('//*[@class="listing-summary col-xs-12 col-sm-6"]')
                    for etikette in single_etikette:
                                    unternehmens_name = etikette.xpath('.//*[@itemprop="name"]/text()').extract()
                                    unternehmens_adresse = etikette.xpath('.//*[@class="address"]/text()').extract_first()

                                    yield {'Name': unternehmens_name,
                                           'Adresse': unternehmens_adresse,
                                           'Postleitzahl' : unternehmens_plz}
```

Scrapy Übersicht Allgemeine Befehle für das Terminal und die Settings.py

1. «scrapy -help» zeigt welche Commands aktuell zur Verfügung stehen
2. «scrapy startproject *projektname*» startet ein neues Scrapy Projekt
3. «scrapy genspider *spidername zieldomain*» generiert eine Spider im Basic-Template
4. «scrapy crawl *spidername* –o *filename.csv*» startet einen Crawler und definiert das Output-Format
5. Settings.py «ROBOTSTXT_OBEY = False» umgeht die robots.txt einer Zieldomain
6. Settings.py «DOWNLOAD_DELAY = 5» Reduziert die Geschwindelt des Downloads und damit die Wahrscheinlichkeit als Crawler erkannt zu werden

Selenium Übersicht Allgemeine Befehle

1. «import selenium from webdriver» importiert Selenium zum Projekt
2. «from time import sleep» importiert die Systemzeit zum Projekt
3. «self.driver = webdriver.Chrome('*C:\webdrivers\chromedriver.exe*')» Integration des Chrome Webdrivers
4. «self.driver.get(url)» Webseite wird aufgerufen
5. «sel = Selector(text=self.driver.page_source)» Webseitdencode wird an den Selector uebergeben
6. «self.driver.find_element_by_id('*below-content*')»
   «find_element_by_xpath»
   «find_element_by_class_name»
   [...] Lokalisiert Elemente auf der Seite

Beispiel (www. bionetz.ch) - Scrapy mit Selenium

```
import scrapy
from scrapy.selector import Selector
from selenium import webdriver
from time import sleep

class BioseleniumSpider(scrapy.Spider):
    name = 'bioselenium'
    allowed_domains = ['www.bionetz.ch']
    start_urls = ['http://www.bionetz.ch/']

    def parse(self, response):
        url = 'https://bionetz.ch/adressen/detailhandel/bio-fachgeschaefte.html'
        self.driver = webdriver.Chrome('C:\webdrivers\chromedriver.exe')
        self.driver.get(url)
        while self.driver.find_elements_by_xpath('//*[@title="Weiter"]'):
                        sel = Selector(text=self.driver.page_source)
                        single_etikette = sel.xpath('//*[@class="listing-summary col-xs-12 col-sm-6"]')
                        for etikette in single_etikette:
                            unternehmens_name = etikette.xpath('.//*[@itemprop="name"]/text()').extract()
                            unternehmens_adresse = etikette.xpath('.//*[@class="address"]/text()').extract_first()
                            yield {'Name': unternehmens_name,
                                   'Adresse': unternehmens_adresse}
                        element = self.driver.find_element_by_id('below-content')
                        self.driver.execute_script("arguments[0].scrollIntoView(0, document.documentElement.scrollHeight-5);", element)
                        sel = Selector(text=self.driver.page_source)
                        sleep(3)
                        self.driver.find_element_by_xpath('//*[@title="Weiter"]').click()
    self.driver.close()
```