UNIVERSITETI I PRISHTINËS
Fakulteti i Inxhinierisë Elektrike dhe Kompjuterike

Lënda: Siguria në Internet

Projekti 4: Zhvillimi i aplikacionit që mundëson URL Crawling

Studentët:
Arbena Musa 170714100093
Blerta Jashari 170714100002
Medina Krelani 170714100040

Profesor:
Dr. techn. Blerim Rexha
Asistent:
M. Sc. Arbnor Halili

Prishtinë, Janar 2019

Përmbajtja

Figurat

Hyrje

Në këtë projekt është dokumentuar zhvillimi i aplikacionit që mundëson URL Crawling. Si parakusht për krijimin e një aplikacioni të tillë është kuptimi i rëndësisë së prezencës së URL-ëve në web dhe arsyeja e përdorimit të tyre. Poashtu, është me rëndësi të kuptohet se çfarë është Web Crawling dhe si gjen zbatim esencial në web.

Çfarë është URL?

URL është shkurtesë për Uniform Resource Locator. URL merr një format të caktuar përmes të cilit krijon një lidhje dhe mundëson çasjen në informacione duke përdorur protokolle të ndryshme. Një URL është vendndodhja (adresa globale) e një burimi të caktuar në internet, që zakonisht i referohet një faqeje në internet, por mund t'i referohet edhe çdo forme tjeter si dokumenteve, imazheve, etj.

Duke qenë se URL formojnë bazën e Web-it, dhënia e rëndësisë së duhur të URL dhe rolit të tyre në mundësimin e punës në Web, parandalon shumë probleme të mundshme të përdorimit dhe të SEO (Search Engine Optimization). URL e bëjnë internetin të funksionojë.

Formati i URL

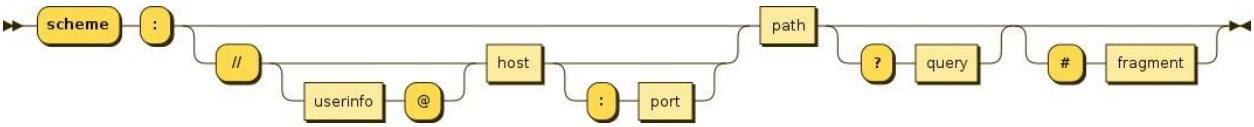


Figura 1 Skema e formatit që mund të ketë një URL

protocol://domainOrIPAddress:port/path/filename

protocol - tregon se cili protokoll i komunikimit është përdorur për t'iu çasur URL-it përkatës. Disa protokolle të zakonshme janë: HTTP (Hiper Text Transfer Protocol) për transferimi e përmbajtjes së web-it dhe FTP (File Transfer Protocol) për transferimin e file-ave përmes një lidhjeje në distancë. Protokolli pasohet nga '://'.

domain or IP address - specifikon IP adresën apo domenin në të cilin është i vendosur burimi.

port - specifikon portin që përdoret për lidhjen. Për protokolle të ndryshme përdoren porte të paracaktuar, psh. porti 80 për HTTP apo porti 21 për FTP.

path - specifikon vendndodhjen e burimit në server. Nëse path nuk ceket atëherë burimi kërkohet në root directory të serverit.

filename - emri aktual i burimit. Nëse emri i file-it nuk ceket, atëherë përdoret një emër i paracaktuar siç është index.html për HTTP lidhjet.

Vendosja e një hierarkie të duhur të përmbajtjes që pasqyrohet nga URL-i i një faqeje siguron dukshmëri më të madhe ndaj search engines, gjë që ndihmon që përmbajtja të renditet më mire në rezultatet e kërkimit.

Çfarë është Web Crawler?

Aspekt interesant i URL-ve është se crawling dhe indexing i përmbajtjes së faqeve në web i aplikuar nga search engines kryhet përmes URL-ëve. Çdo gjë që web-based search engines bëjnë është e ndërtuar në aspektin themelor të web-it, URL që lidhen me URL të tjera. Crawling është teknologjia bazë e search engines. Gjithçka që bën një crawler i një search engine përqendrohet në URL dhe gjetjen e URL-ëve të reja përmes saj.

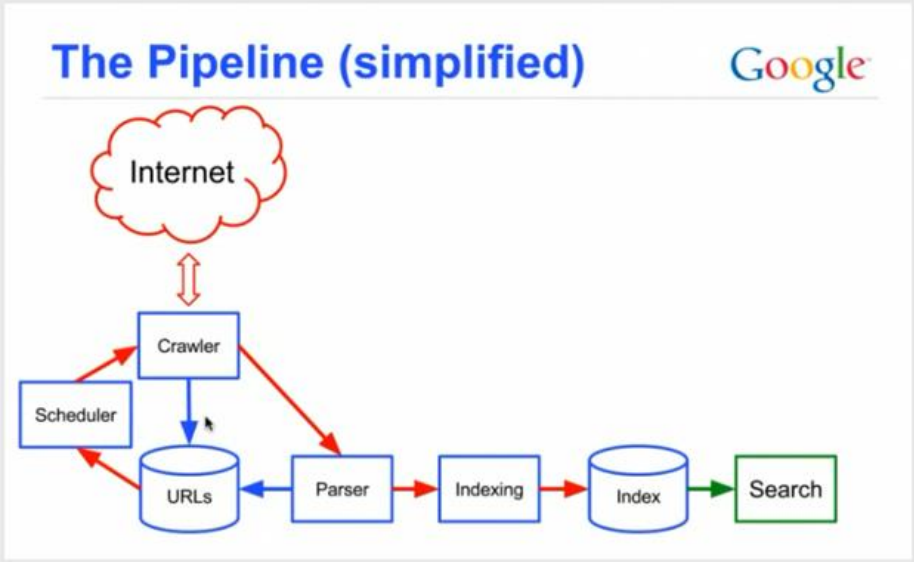


Figura 2 Procesi i kërkimit të Google search engine

" Një web crawler (i njohur edhe si web spider, spider bot, web bot, apo thjesht crawler) është një program softuerik që përdoret nga një search engine për të indeksuar web faqe dhe përmbajtje në World Wide Web. "

Kur një përdorues bën kërkime në një search engine, para se të shfaqen rezultatet ato paraprakisht duhet të gjenden. Për të gjetur informacione mbi qindra miliona faqe që ekzistojnë në internet, një search engine përdorë programe softuerike që quhen spiders, të cilat ndërtojnë listat e fjalëve që gjenden në web faqe. Kur një spider ndërton listat e kërkimit, procesi quhet Web crawling.

Një web crawler fillon kërkimin nga një listë e URL-ëve të quajtur seeds. Ndërsa crawler viziton këto URL, ai identifikon të gjitha lidhjet me faqet e tjera dhe i shton ato në listën e URL-ëve që duhet t'i vizitoj që quhet crawl frontier. Crawler zakonisht bën një lloj normalizimi të URL-ëve për të shmangur përdorimin e të njejtit burim shumë herë.

Teknologjia e përdorur

Google Crawler ka qenë fillimisht i zhvilluar në Python, andaj edhe për zhvillimin e programit tonë kemi përdorur po këtë gjuhë programuese, përkatësisht një prej framework-ëve të saj Flask.

Implementimi

MyApp është klasë e cila si instancë të saj do të ketë një web application. Inicializohet përmes konstruktorit që ka si parameterër një array të linqeve.

MyApp.py

```
class MyApp:
  def __init__(self, links):
    self.links = links
```

Klasa UrlCrawler inicializohet përmes konstruktorit të saj i cili pranon dy parametra, *starting_url* që është URL-i i parë prej të cilit niset kërkimi i URL-ëve të tjerë dhe *depth* që përcakton se deri në cilën shkallë do të vazhdojë kërkimi.

Funksioni *get_app_from_link()* pranon si parametër një link prej të cilit supoohet të niset kërkimi. Funksioni *requests.get(link)* dërgon GET kërkesë në linkun përkatës dhe përgjigjen e ruan si response object. Prej linkut përkatës *html.fromstring(start_page.text)* nxjerr HTML përmbajtjen e linkut e cila më pas është e gatshme të analizohet. Metoda *xpath()* përmes query-ve të cilave iu nënshtrohet i tërë dokumenti arrin që të nxjerr informata përmes mënyrës se si është i strukturuar HTML dokumenti. Në këtë rast nxjerren të gjitha linqet brenda faqes dhe ruhen në array-in *linka*. Kontrollohet nëse ndojë prej elemeteve të ruajtura është i zbrazët, nëse jo vazhdohet me verifikim të mëtejshëm. Si përfundim krijohet instanca *app* e klasës *MyApp* konstruktori i të cilës merr si parametrës array-in me linqe dhe funksioni e kthen instancën.

Funksioni *crawl()* ka për detyrë të bëjë rrugëtimimn nga një URL si burim kryesor, në burimet e tjera që sigurohen. Fillimisht përmes funksionit *get_app_from_link()* krijohet instanca app me listën e linqeve përkatëse. Instanca e krijuar shtohet në listën e *apps*, ndërsa linqet e instancës shtohen në listën e *depth_links*. Nëse shkalla aktuale e thellësisë ende nuk ka arritur shkallën e thellësisë së synuar për kërkimin e linqeve, atëherë për secilën shkallë të linqeve të deritanishme instancohet një object *MyApp* i ri me parametra po ato inqe. Ky kërkim rekurzim ndalon në momentin që shkalla e thellësisë së kërkimit ka arritur shkallën e paracaktuar.

Funksioni get_related_links kthen listën me të gjitha linqet e nxjerra përmes URL crawling.

UrlCrawler.py

```
from lxml import html
import requests

from MyApp import MyApp


class URLCrawler:
    def __init__(self, starting_url, depth):
        self.starting_url = starting_url
        self.depth = depth
        self.current_depth = 0
        self.depth_links = []
        self.apps = []

    def get_app_from_link(self, link):
        start_page = requests.get(link)
        tree = html.fromstring(start_page.text)
        linka = tree.xpath('//*/a/@href')
        links = []
        for li in linka:
            if len(li) != 0:
                if li[0] == '/':
                    li = link + li
                if li not in links and li[0] != '#' and li[0] != '?':
                    links.append(li)

        app = MyApp(links)
        return app

    def crawl(self):
        app = self.get_app_from_link(self.starting_url)
        self.apps.append(app)
        self.depth_links.append(app.links)

        while self.current_depth < self.depth:
            current_links = []
            for link in self.depth_links[self.current_depth]:
                current_app = self.get_app_from_link(link)
                current_links.extend(current_app.links)
                self.apps.append(current_app)
            self.current_depth += 1
            self.depth_links.append(current_links)
```

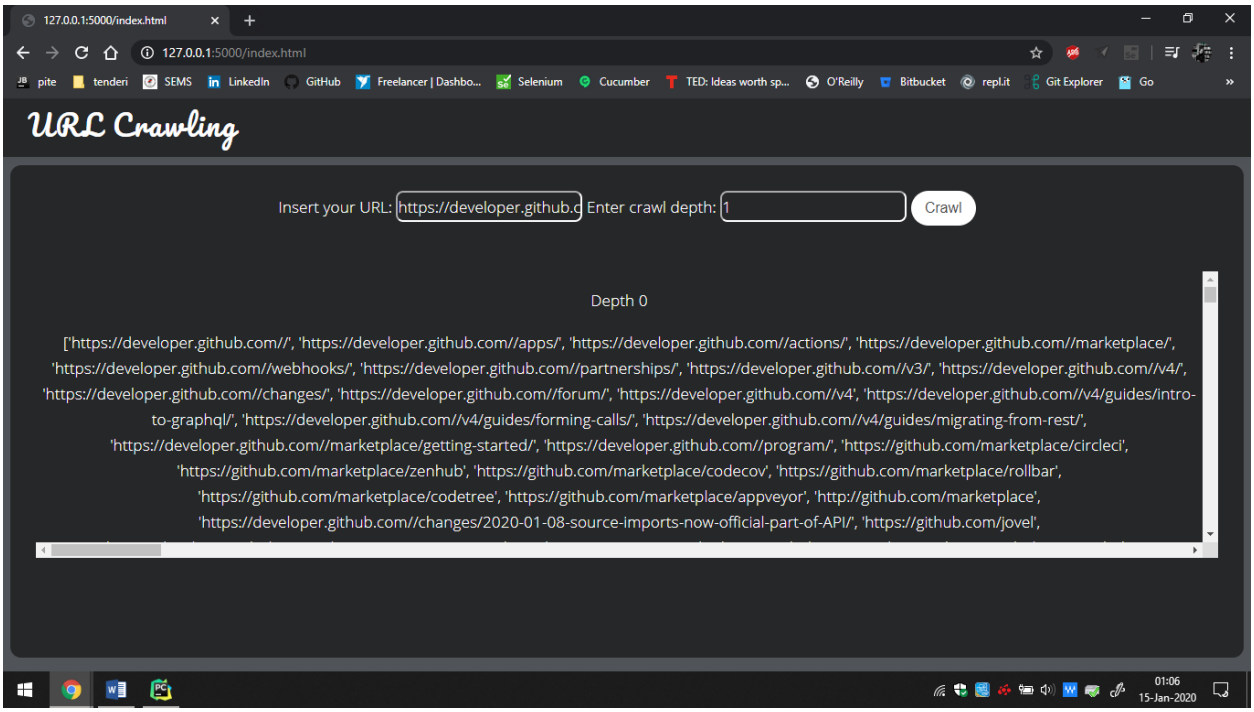
```
    def get_related_links(self):
        return self.depth_links
```

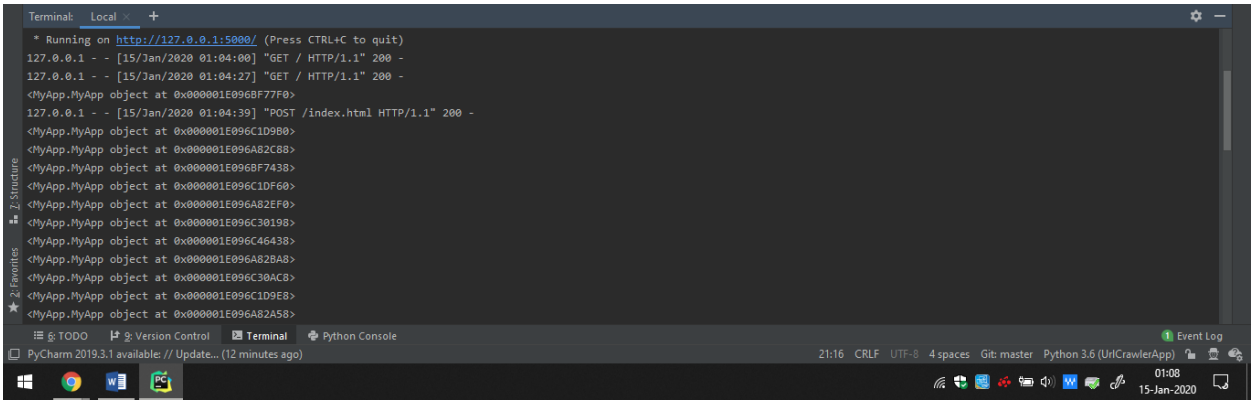
Klasa *app.py* është klasa kryesore e programit e cila edhe bën lidhjen e formës e cila plotësohet nga përdoruesi me funksionalitetin e programit. Kur forma bëhet submit nga ama e përdoruesit të dhënat nga fusha e url dhe depth ruhen në dy variablat *givenUrl* dhe *givenDepth.* Këto dy të dhëna pasohen si argumente në instancilin e klasës *URLCrawler*. Për të kryer përgjigjet tek përdoruesi thirren funksionet *crawl()* dhe *get_related_links()* të cilat mundësojnë gjetjen e linqeve të dëshiruar të cilat edhe i shfaqen më pas shfrytëzuesit.

app.py

```
from flask import Flask, render_template, request, redirect, url_for

from UrlCrawler import URLCrawler

app = Flask(__name__)


@app.route('/')
def get_crawl_page():
    return render_template('index.html', title='URL-Crawling')


@app.route('/index.html', methods=['GET', 'POST'])
def getcrawl():
    error = ""
    if request.method == 'POST':
        givenUrl = request.form['givenUrl']
        givenDepth = int(request.form['givenDepth'])

    crawler = URLCrawler(givenUrl, givenDepth)

    crawler.crawl()
    links = crawler.get_related_links()
    for app in crawler.apps:
        print(app)
    return render_template('index.html', len = len(links), links = links)


if __name__ == '__main__':
    app.run()
```

Ekzekutimi


127.0.0.1:5000/index.html
URL Crawling
Insert your URL: https://developer.github.c
Enter crawl depth: 1
Crawl
Depth 0
['https://developer.github.com//', 'https://developer.github.com//apps/', 'https://developer.github.com//actions/', 'https://developer.github.com//marketplace/', 'https://developer.github.com//webhooks/', 'https://developer.github.com//partnerships/', 'https://developer.github.com//v3/', 'https://developer.github.com//v4/', 'https://developer.github.com//changes/', 'https://developer.github.com//forum/', 'https://developer.github.com//v4', 'https://developer.github.com//v4/guides/intro-to-graphql/', 'https://developer.github.com//v4/guides/forming-calls/', 'https://developer.github.com//v4/guides/migrating-from-rest/', 'https://developer.github.com//marketplace/getting-started/', 'https://developer.github.com//program/', 'https://github.com/marketplace/circleci', 'https://github.com/marketplace/zenhub', 'https://github.com/marketplace/codecov', 'https://github.com/marketplace/rollbar', 'https://github.com/marketplace/codetree', 'https://github.com/marketplace/appveyor', 'http://github.com/marketplace', 'https://developer.github.com//changes/2020-01-08-source-imports-now-official-part-of-API/', 'https://github.com/jovel',
01:06
15-Jan-2020


Terminal: Local
* Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
127.0.0.1 - - [15/Jan/2020 01:04:00] "GET / HTTP/1.1" 200 -
127.0.0.1 - - [15/Jan/2020 01:04:27] "GET / HTTP/1.1" 200 -
<MyApp.MyApp object at 0x000001E096BF77F0>
127.0.0.1 - - [15/Jan/2020 01:04:39] "POST /index.html HTTP/1.1" 200 -
<MyApp.MyApp object at 0x000001E096C1D9B0>
<MyApp.MyApp object at 0x000001E096AB2C88>
<MyApp.MyApp object at 0x000001E096BF7438>
<MyApp.MyApp object at 0x000001E096C1DF60>
<MyApp.MyApp object at 0x000001E096AB2EF0>
<MyApp.MyApp object at 0x000001E096C30198>
<MyApp.MyApp object at 0x000001E096C46438>
<MyApp.MyApp object at 0x000001E096AB2BA8>
<MyApp.MyApp object at 0x000001E096C30AC8>
<MyApp.MyApp object at 0x000001E096C1D9E8>
<MyApp.MyApp object at 0x000001E096AB2A58>
6: TODO
9: Version Control
Terminal
Python Console
Event Log
PyCharm 2019.3.1 available: // Update... (12 minutes ago)
21:16 CRLF UTF-8 4 spaces Git: master Python 3.6 (UrlCrawlerApp)
01:08
15-Jan-2020

Referencat

[1] URLs, Crawling, and PageRank; Fundamentals of SEO
[2] URL Format
[3] What Is a Web Crawler and How Does It Work?
[4] How Internet Search Engines Work