WEB SCRAPING WITH JAVA - JSOUP

A MINI PROJECT REPORT

Submitted by

S. RAGUL (811519104089)

V.TAMILARASAN (811519104110)

D. VIJAYKUMAR (811519104119)

B. MARIA MANOJE (811519104302)

In partial fulfilment for the award of the degree

of

BACHELOR OF ENGINEERING

in

COMPUTER SCIENCE AND ENGINEERING

K.RAMAKRISHNAN COLLEGE OF ENGINEERING
(AUTONOMOUS),
SAMAYAPURAM, TIRUCHIRAPPALLI –621 112.

JUNE 2022

BONAFIDE CERTIFICATE

Certified that this mini project report **" WEB SCRAPING WITH JAVA – JSOUP "** is the bonafide work of **"S. RAGUL, V. TAMILARASAN, D. VIJAYKUMAR, B. MARIA MANOJE,"** who carried out the mini project work under my supervision.

SIGNATURE	**SIGNATURE**
Dr.T.M.NITHYA, ME.,	Dr.R.SRIDEVI
Associate Professor	Professor
HEAD OF THE DEPARTMENT	**SUPERVISOR**
Computer Science & Engineering	Computer Science & Engineering
K.Ramakrishnan College of Engineering (Autonomous)	K.Ramakrishnan College of Engineering (Autonomous)
Trichy – 621 112	Trichy – 621 112

Submitted for the Project Viva-Voce Examination held on……

INTERNAL EXAMINER **EXTERNAL EXAMINER**

ACKNOWLEDGEMENT

We thank the almighty GOD, without whom it would not have been possible for us to complete our project.

We wish to address our profound gratitude to **Dr. K. RAMAKRISHNAN**, Chairman, K. Ramakrishnan College of Engineering, who encouraged and gave us all help throughout the course.

We express our hearty gratitude and thanks to our honorable and grateful executive director **Dr. S. KUPPUSAMY, B.sc., MBA., Ph.D.,** K. Ramakrishnan College of Engineering.

We are glad to thank our principal **Dr. D. SRINIVASAN, M.E, Ph.D., FIE., MIIW., MISTE., MISAE., C.Engg.**, for giving us permission to carry out this mini project.

We wish to convey our sincere thanks to **Dr. T. M. NITHYA, M.E., Ph.D.,** Head of the Department, Computer Science and Engineering for giving us constant encouragement and advice throughout the course.

We are graceful to **Dr.R.SRIDEVI,** Professor in the Department of Computer Science and Engineering, K.Ramakrishnan College of Engineering for her guidance and valuable suggestions during the course of study.

Finally we sincerely acknowledged in no less term for all our staff members, colleagues, our parents and friends for their co-operation and help at various stages of this mini project work.

DECLARATION

I hereby declare that the work entitled " **WEB SCRAPING WITH JAVA – JSOUP "** is submitted in partial fulfillment of the requirement for the reward of the degree in B.E.,Anna University,Chennai, is a record of our own work carried out by me during the academic year 2021-2022 under the supervision and guidance of **Dr. SRIDEVI, Professor, Department of Computer Science and Engineering, K.Ramakrishnan College of Engineering**. The extent and source of information have derived from the existing literature and have been indicated through the dissertation at the appropriate places. The matter embodied in this work is original and has not be submitted for the award of any degree or diploma, either in this or any other University.

S. RAGUL(811519104089)

I certify that the declaration made by above candidate is true.

Dr.R.Sridevi

Professor/CSE

DECLARATION

I hereby declare that the work entitled " **WEB SCRAPING WITH JAVA – JSOUP "** is submitted in partial fulfillment of the requirement for the reward of the degree in B.E.,Anna University,Chennai, is a record of our own work carried out by me during the academic year 2021-2022 under the supervision and guidance of **Dr. SRIDEVI Professor, Department of Computer Science and Engineering, K.Ramakrishnan College of Engineering**. The extent and source of information have derived from the existing literature and have been indicated through the dissertation at the appropriate places. The matter embodied in this work is original and has not be submitted for the award of any degree or diploma, either in this or any other University.

V. TAMILARASAN(811519104110)

I certify that the declaration made by above candidate is true.

Dr.R.Sridevi

Professor/CSE

DECLARATION

I hereby declare that the work entitled " **WEB SCRAPING WITH JAVA – JSOUP "** is submitted in partial fulfillment of the requirement for the reward of the degree in B.E.,Anna University,Chennai, is a record of our own work carried out by me during the academic year 2021-2022 under the supervision and guidance of **Dr. SRIDEVI Professor, Department of Computer Science and Engineering, K.Ramakrishnan College of Engineering**. The extent and source of information have derived from the existing literature and have been indicated through the dissertation at the appropriate places. The matter embodied in this work is original and has not be submitted for the award of any degree or diploma, either in this or any other University.

D.VIJAYKUMAR (811519104119)

I certify that the declaration made by above candidate is true.

Dr.R.Sridevi

Professor/CSE

DECLARATION

I hereby declare that the work entitled " **WEB SCRAPING WITH JAVA – JSOUP "** is submitted in partial fulfillment of the requirement for the reward of the degree in B.E.,Anna University,Chennai, is a record of our own work carried out by me during the academic year 2021-2022 under the supervision and guidance of **Dr. SRIDEVI Professor, Department of Computer Science and Engineering, K.Ramakrishnan College of Engineering**. The extent and source of information have derived from the existing literature and have been indicated through the dissertation at the appropriate places. The matter embodied in this work is original and has not be submitted for the award of any degree or diploma, either in this or any other University.

B.MARIA MANOJE (811519104302)

I certify that the declaration made by above candidate is true.

Dr.R.Sridevi

Professor/CSE

ABSTRACT

Web scraping, also known as web extraction or harvesting, is a technique to extract data from the World Wide Web (WWW) and save it to a file system or database for later retrieval or analysis. Commonly, web data is scrapped utilizing Hyper-text Transfer Protocol (HTTP) or through a web browser. This is accomplished either manually by a user or automatically by a bot or web crawler. Main objective of Web Scraping is to extract information from one or many websites and process it into simple structures such as spread sheets, database or CSV file. However, in addition to be a very complicated task, Web Scraping is resource and time consuming, mainly when it is carried out manually. Jsoup : A Java library that implements the WHATWG HTML5 specification, can be used to parse HTML documents, find and extract data from HTML documents, and manipulate HTML elements. Web scraping is a legal process and it is completely based on business prospectives to improve or enlightens data sharing for solding products and other needs.

TABLE OF CONTENTS

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

Web scraping refers to the process of extracting a significant amount of information from a website using scripts or programs. Such scripts or programs allow one to extract data from a website, store it and present it as designed by the creator. The data collected can also be part of a larger project that uses the extracted data as input. With web scraping, can not only automate the process but also scale the process to handle as many websites as your computing resources can allow.Data collection lives in the now. Stride at the same speed with this straightforward guide to web scraping with Java.

1.1 MOTIVATION

As opposed to the "time is money" mentality of the 20th century, now it's all about data. Particularly in the last decade, web scrapers have become extremely popular. It’s not hard to understand why - the Internet is brimming with valuable information that can make or break companies.

The chapter will provide a step-by-step tutorial on creating a simple web scraper using Java to extract data from websites and then save it locally in CSV format.

1.2 Literature review

Crime on the Internet As the Internet sets out to connect the world, more people than ever are using the Internet and the World Wide Web in their everyday lives. Estimates have found that in 2000, there were around 400 million users of the Internet, while in 2015, there were around 3.2 billion users, an increase of 2.8 billion. This is shown in

figure 1.1 along with the proportions of the populations of developed, developing and least developed countries (LDCs) who have access to the Internet.

This is shown in figure 1.1 along with the proportions of the populations of developed, developing and least developed countries (LDCs) who have access to the Internet.

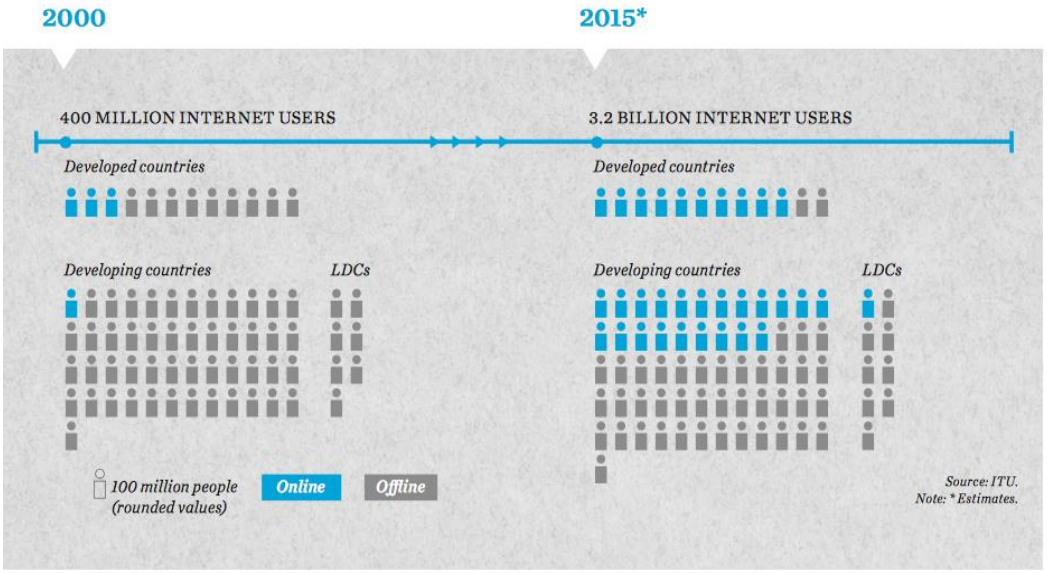

Fig 1.1 - Number of users of the Internet from 2000 to 2015

1.3 Surface Web, Deep Web and Dark Web

The web has been estimated to be as large as 6 zettabytes , constantly growing larger as more content is produced. As the web is a large collection of documents, difficulties can arise when trying to index the entire World Wide web.PHP performs system functions, i.e., from files on a system it can create, open, read, write, and close them.The surface web consists of webpages and documents that have been or can be indexed by search engines.

Although crime can be easily detected on the surface web, platforms still exist selling counterfeit goods and drugs, among others, and can be found on social media or by using a simple search on a browser.

CHAPTER 2

WEB SCRAPING

2.1 Web Scraping

Web scraping refers to the process of collecting information from specific websites with predefined and tailored automated software. As the Internet and the web continue to expand, it can become difficult to access webpages without knowing beforehand the address of the page. This is where search engines come in. Search engines use a process called web crawling, which is an algorithm designed to scan or crawl through a collection of websites, which is indexed and searched.

Fig 2.1 –Web Scraping

Web crawler scans many webpages to find links, the format of those links remains the same, however, when using a web scraper to extract

data from webpages, the format of the markup changes between different websites. Despite this, web scrapers are often used to access areas of a website that search engines cannot.

2.1.1 Understanding web scraping

What does web scraping refer to? Many sites do not provide their data under public APIs, so web scrapers extract data directly from the browser. It’s a lot like a person copying text manually, but it’s done in the blink of an eye.When you consider that better business intelligence means better decisions, this process is more valuable than it seems at first glance. Websites are producing more and more content, so doing this operation entirely by hand is not advisable anymore.

You might be wondering, "What am I going to do with this data?". Well, let's see a few of the use cases where web scraping can really come in handy:

- Lead generation: an ongoing business requires lead generation to find clients.
- Price intelligence: a company's decision to price and market its products will be informed by competitors’ prices.
- Machine learning: to make AI-powered solutions work correctly, developers need to provide training data.

IP blocking: if a website determines multiple requests are coming from the same IP address, it can block access to that website or greatly slow you down.

Honeypots: invisible links that are visible to bots but invisible to humans; once the bots fall for the trap, the website blocks their IP address.

2.1.2 Understanding the Web

To understand the Web, you need to understand Hypertext Transfer Protocol (HTTP) which explains how a server communicates with a client. There are multiple pieces of information that a message contains that describe the client and how it handles data: method, HTTP version, and headers. Web scrapers use the GET method for HTTP requests, meaning that they retrieve data from the server. Some advanced options also include the POST and the PUT methods.

Several additional details about requests and responses can be found in HTTP headers. But the ones relevant in web scraping are:

- User-Agent: indicates the application, operating system, software, and version; web scrapers rely on this header to make their requests seem more realistic.
- Host: the domain name of the server you accessed.
- Referrer: contains the source site the user visited; accordingly, the content displayed can differ, so this fact has to be considered as well.
- Cookie: keeps confidential information about a request and the server (such as authentication tokens).
- Accept: ensures the response for the server is in a specific type (ex: text/plain, application/json, etc.).

CHAPTER 3

SYSTEM SPECIFICATION

3.1 PLATFORM REQUIREMENT

3.1.1 SUPPORTIVE OPERATING SYSTEMS

The supported Operating Systems for client include:

- Windows XP onwards
- Linux any flavor.

Windows and Linux are two of the operating systems that will support comparative website. Since Linux is an open source operating system, this system which is will use in this project is developed on the Linux platform but is made compatible with windows too. The comparative Website will be tested on both Linux and windows. The supported Operating Systems for server include: The supported Operating Systems For server include Linux. Linux is used as server operating system. For web server are using apache 2.0

3.2 HARDWARE SPECIFICATION

- 1 GB Ram.
- 40 GB Hard Disk Minimum.
- Intel Core

3.3 SOFTWARE REQUIREMENT

The Software Requirements in this project include:

- Spring Boot Tool 4
- Internet Explorer, Mozilla Firefox, Google Chrome etc.

CHAPTER 4

DESIGNING OF WEB SCRAPING

4.1 DESIGN

The designing of Webscraping process is sub divided into following 5 major steps,

4.1.1 STEPS

Step 1: Set up the environment

To build our Java web scraper, we need first to make sure that have all the prerequisites:

- Java 8: even though Java 11 is the most recent version with Long-Term Support (LTS), Java 8 remains the preferred production standard among developers.
- Gradle: is a flexible open-source build automation tool with a wide range of features, including dependency management (requires Java 8 or higher);
- After installation, we should verify if we followed the official guides correctly. Open a terminal and run the following commands:

```
> java -version
> gradle -v
```

Step 2: Inspect the page you want to scrape

Cool, let's move on! Navigate to the page you want to scrape and right-click anywhere on it, then hit "Inspect element". The developer console will pop up, where you should see the HTML of the website.

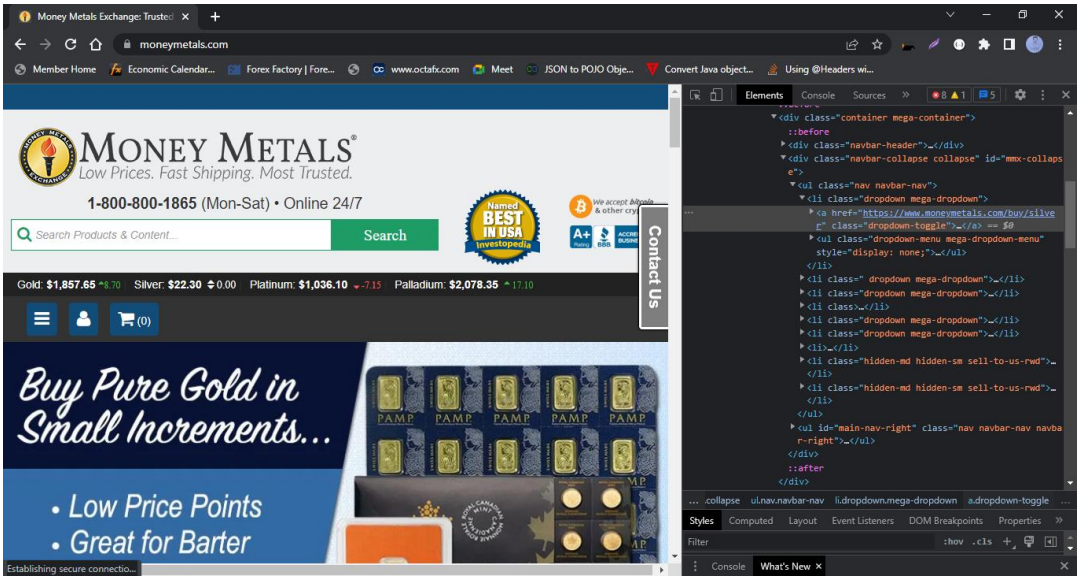

Fig 4.1 –Scraping data from website

Step 3: Send an HTTP request and scrape the HTML

Now, to get that HTML on our local machine, they send an HTTP request using HtmlUnit, that will return the document. Let's get back to the IDE, and put this idea into code.

Then initialize a WebClient and send an HTTP request to the website that will return a HtmlPage. It's important to remember to close the connection after receiving the response, as the process will continue running.

Step 4: Extracting specific sections

Have an HTML document, but want data, which means that have should parse the previous response into human-readable information.

Starting with baby steps, let's extract the title of the website. They can do this with the help of the built-in method getTitleText:

```
<a title="Linguine alla Parmigiana" class="card-link" href="/recipes/linguine-alla-parmigiana/" target="_self">…</a> == $0
```

Step 5: Export the data to CSV

It will create a CSV file, as it can be easily read by another application and opened with Excel for further processing. First, just one more import:

```
import java.io.FileWriter;
```

Then initialize our FileWriter that will create the CSV in “append” mode:

```
FileWriter recipesFile = new FileWriter("recipes.csv
recipesFile.write("id,name,link\n");
```

They are done writing to the file, so now it’s time to close it:

```
recipesFile.close();
```

CHAPTER 5

SYSTEM ANALYSIS

5.1 System Architecture

Architecture diagrams can help system designers and developers visualize the high-level, overall structure of their system or application for the purpose of ensuring the system meets their users' needs. They can also be used to describe patterns that are used throughout the design.

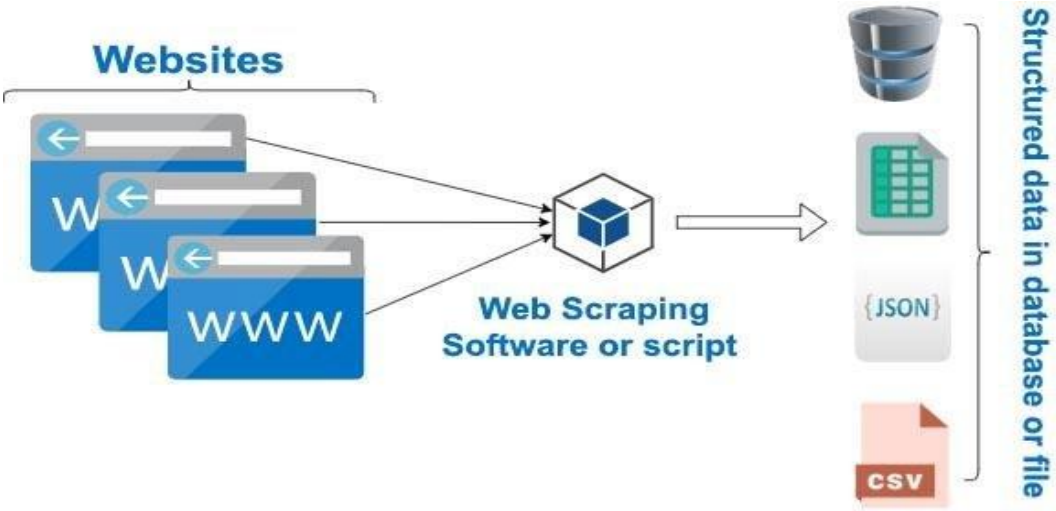

Fig 5.1 - System Architecture

5.2 Flowchart

A flowchart is simply a graphical representation of steps. It shows steps in sequential order and is widely used in presenting the flow of algorithms, workflow or processes. Typically, a flowchart shows the steps as boxes of various kinds, and their order by connecting them with arrows.

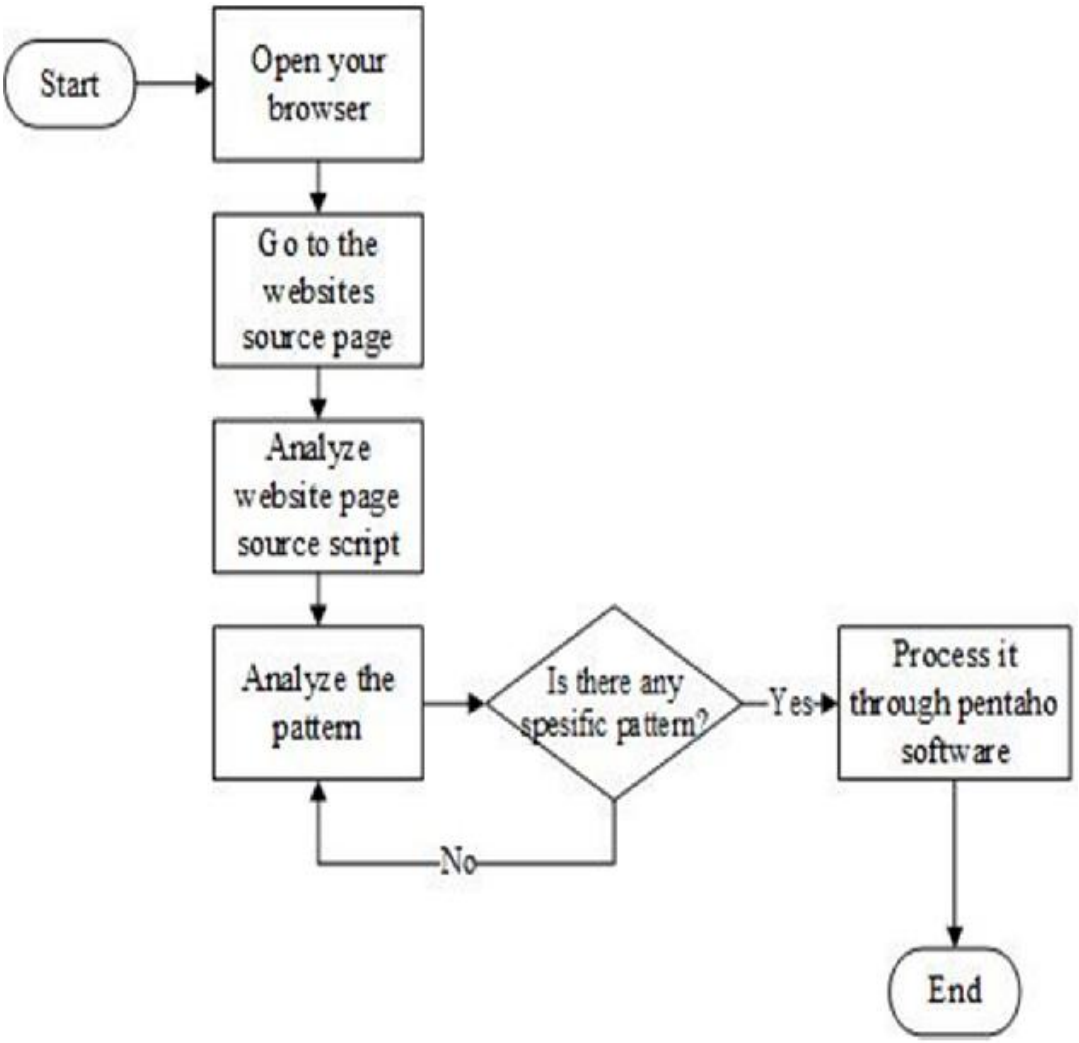

Fig 5.2 - Flowchart

5.3 Usecase Diagram

A use case is a methodology used in system analysis to identify, clarify and organize system requirements. A use case document can help the development team identify and understand where errors may occur during a transaction so they can resolve them.

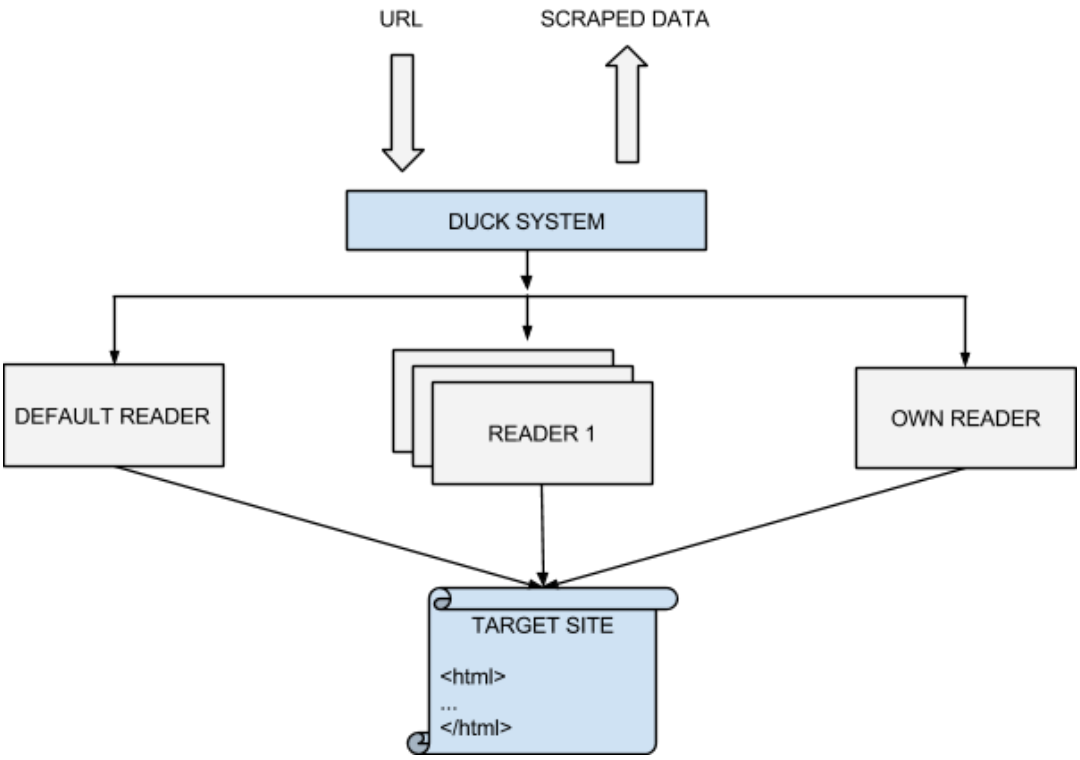

Fig 5.3 -Usecase diagram

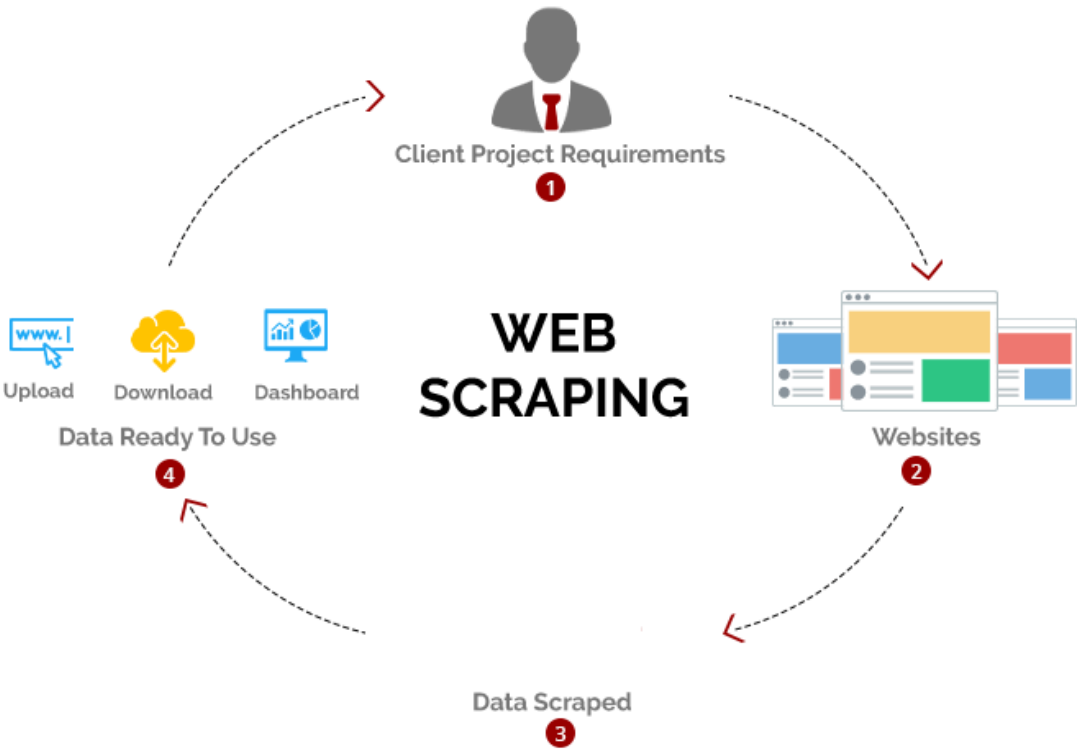

Fig 5.4 -Web Scraping in website

CHAPTER 6

CONCLUSION & FUTURE ENHANCEMENT

6.1 CONCLUSION

The " WEB SCRAPING WITH JAVA – JSOUP " has been developed to satisfy all proposed requirements. When have learned about web scraping using the Java language and built a functional scraper using the simple but powerful JSoup library. So now that have the scraper and the data, what next? There is more to web scraping than what have covered. Practice is as important as it is helpful, so build more scrapers covering new ground of complexity with each new one and even with different libraries to widen your knowledge.

6.2 FUTURE ENHANCEMENT

In future, the mini project can be enhanced by,

- The Future of web scraping is indeed bright and it will become more and more essential for every business with the passage of time.
- Web scraping services are considered as one of the most practiced activities done by most of the IT companies and Ecommerce Stores that operate across the globe.
- Use R- language , in the web scraping then provide and show graphical representation.

APPENDICES

APPENDIX I - SAMPLE CODE

WebscrapApplication.java :

```
package com.example.webscrap;
import org.springframework.boot.CommandLineRunner;
import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.context.annotation.Bean;
@SpringBootApplication
public class WebsiteScrapeApplication implements CommandLineRunner{
	public static void main(String[] args) {
		SpringApplication.run(WebsiteScrapeApplication.class, args);
	}
	@Override
	public void run(String... args) throws Exception {
		// TODO Auto-generated method stub
		System.out.println("hello");
	}
}
```

SERVICES :

ProductService.java :

```
package com.example.webscrap.service;
import java.io.File;
import java.io.FileWriter;
```

```
import java.io.Writer;
import java.util.ArrayList;
import java.util.Arrays;
import java.util.List;
import javax.servlet.http.HttpServletResponse;
import org.apache.commons.csv.CSVFormat;
import org.apache.commons.csv.CSVPrinter;
import org.apache.commons.lang3.ArrayUtils;
import org.json.JSONArray;
import org.json.JSONObject;
import org.jsoup.Jsoup;
import org.jsoup.nodes.Document;
import org.jsoup.nodes.Element;
import org.jsoup.select.Elements;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.context.annotation.Configuration;
import org.springframework.stereotype.Service;
import com.example.webscrap.model.AllProducts;
import com.example.webscrap.model.Product;
import com.example.webscrap.model.Specification;
import com.example.webscrap.repository.ProductRepository;
import com.example.webscrap.repository.SpecificationRepository;
import com.example.webscrap.service.FileLoad;
import com.example.webscrap.service.ProductUrlService;
import com.fasterxml.jackson.databind.DeserializationFeature;
import com.fasterxml.jackson.databind.MapperFeature;
import com.fasterxml.jackson.databind.ObjectMapper;
import com.opencsv.CSVWriter;

@Service
```

```
public class ProductService {
	@Autowired
	ProductRepository productrepository;

	@Autowired
	SpecificationRepository specrepository;

	Specification specification=new Specification();
	List<Specification> specificationItems = new ArrayList<Specification>();
	List<AllProducts> Allproductitems = new ArrayList<AllProducts>();
	int id = 1;
	FileLoad files = new FileLoad();
	String mintFecility = "", denomination = "", date_year = "", quality = "",
purity = "",
			metalWeight = "", diameter = "", edge = "", thickness = "",
obverse = "", reverse = "";

	// To get grid coinurls
	public void getUrl(JSONArray  CategoryBase,String csvname) {
		// String baseUrl =
"https://www.moneymetals.com/buy/silver/coins";
		String categoryValue = "";
		// JSONArray CategoryBase = ProductUrlService.getcoinUrls();
		List<String> coinurl = new ArrayList<>();
		ArrayList<String> productUrls = new ArrayList<>();
		List<AllProducts> scrappedData = new ArrayList<AllProducts>();
		Allproductitems.clear();
		// testing
		/*JSONArray CategoryBase = new JSONArray();
		for (int i = 0; i < 1; i++) {
```

```
                JSONObject categoryJsonobj = new JSONObject();
                categoryJsonobj.put("categoryUrl",
"https://www.moneymetals.com/buy/silver/bars");
                categoryJsonobj.put("category", "Silver bars");
                CategoryBase.put(categoryJsonobj);
        }

         for (int i = 0; i < 1; i++) { JSONObject categoryJsonobj = new
JSONObject();
         categoryJsonobj.put("categoryUrl",
         "https://www.moneymetals.com/buy/gold/pre-1933");
         categoryJsonobj.put("category", "Pre-1933");
         CategoryBase.put(categoryJsonobj); }*/

        try {
                // getting category value and baseurl depends category
                int j=CategoryBase.length();
                for (int i = 0; i < CategoryBase.length(); i++) {
                        JSONObject jsonObject1 =
CategoryBase.getJSONObject(i);
                        String baseUrl =
jsonObject1.optString("categoryUrl");
                        categoryValue = jsonObject1.optString("category");

                        Document Baseurldocument =
Jsoup.connect(baseUrl).get();
                        Elements baseProduct =
Baseurldocument.select("div.wrap div.categories div.row");
                        Elements rows = baseProduct.select("div.ilb-products
div.row div.ilb-product div.mmx-product-title");
                        for (Element values : rows) {
                                coinurl.add(values.select("a").attr("href"));
                        }
```

```
                    /*coinurl.clear();
                    coinurl.add("https://www.moneymetals.com/2022-1-oz-american-gold-eagle-coin-in-mintid-22k-purity/762");

        coinurl.add("https://www.moneymetals.com/american-silver-eagle-random-year/262");
                    categoryValue="Gold Coins";*/
                    scrappedData=scrapeData(coinurl, categoryValue);
                    System.out.println("scrappedData" + scrappedData);

                    coinurl.clear();
                }
                System.out.println("---writer started----");
                csvwriter(Allproductitems, categoryValue,csvname);
                System.out.println("----writer ended---");

            } catch (Exception e) {
                e.printStackTrace();
            }
        }

        // To get scraped data
        private List<AllProducts> scrapeData(List<String> urls, String category) throws Exception {
            List<Product> items = new ArrayList<Product>();

            List<Specification> specitems = new ArrayList<Specification>();
            List<String> namesList = files.filedata();
            JSONObject productJsonvalue = new JSONObject();
            JSONObject SpecJsonvalue = new JSONObject();

            String Namedetails = "", imgUrls = "";
```

```
String[] arr = new String[10];
int index = 0;
Boolean flag;
// loop through url's
try {
        for (String url : urls) {

            final Document document = Jsoup.connect(url).get();
            Elements mmx_items = document.select("div.mmx-
item");
            Elements row_data = mmx_items.select("div.row");
            // getting fileurl for condition
            flag = namesList.contains(url);

            if (!flag) {

                // getting name
                Namedetails = row_data.select("h1").text();
                productJsonvalue.put("Name",Namedetails);
                // lowest_price
                String lowest_price =
row_data.select("div.top-lowest-price strong").text();
                productJsonvalue.put("price",lowest_price);
                // price actions
                String price_elements =
row_data.select("div.products-right div.price-actions button").text();
                Boolean stock;
                if (price_elements.trim().equals("ADD TO
CART")) {
                    stock = true;
                } else {
                    stock = false;
```

```
                                        }
                                        // getting bannerimage item-thumb
                                        String banner_image =
row_data.select("div.item-thumb img").attr("src");

        productJsonvalue.put("BannerImageUrl",banner_image);
                                        // all image urls ProductCategory
                                        Elements ul_list = row_data.select("div.row
div.item ul.item-preview-list");

                                        for (Element row : ul_list.select("li")) {
                                                String result = row.select("img.thumb-
nubs").attr("src");
                                                imgUrls = imgUrls + result + ";";
                                        }
                                        productJsonvalue.put("ImageUrls",imgUrls);

                                        //category
                                        productJsonvalue.put("ProductCategory",
category);

                                        // getting specifications

                                        /*  Elements spec =
row_data.select("div.item-tabs div.tab-content div#prod_specs table");

                                          Elements tbody = spec.select("tbody");
                                          for (Element row : tbody.select("tr"))
                                          {
                                                String td_val =
row.select("td").text();
                                                String[] arrOfStr =  td_val.split(":",
2);
```

```
                                        SpecJsonvalue.put(arrOfStr[0],
arrOfStr[1]);
                                    }


                                     productJsonvalue.put("specification",
SpecJsonvalue);
                                    System.out.println("specstring"
productJsonvalue);
                                    //object mapper
                                    ObjectMapper mapper=new ObjectMapper();
                                    Product
productItems=mapper.readValue(productJsonvalue.toString(), Product.class);

mapper.configure(DeserializationFeature.FAIL_ON_UNKNOWN_PROPERTIES,
false);
                                //
mapper.configure(MapperFeature.DEFAULT_VIEW_INCLUSION, true);
                                     productrepository.save(productItems);*/
                                    //String
specification=getSpecification(row_data);


                                    String[] arrOfStr;
                                    Elements spec = row_data.select("div.item-
tabs div.tab-content div#prod_specs table");


                                    Elements tbody = spec.select("tbody");
                                    System.out.println("berfore
insert"+SpecJsonvalue);
                                    for (Element row : tbody.select("tr")) {
                                        String td_val = row.select("td").text();
                                         arrOfStr = td_val.split(":", 2);
                                        SpecJsonvalue.put(arrOfStr[0],
arrOfStr[1]);
                                        switch (arrOfStr[0]) {
```

```
                case "Denomination":

                        if(arrOfStr[1]=="") {
                                denomination="";
                        }else {
                                denomination =
arrOfStr[1];

                        }
                        continue;
                case "Quality / Type":
                        if(arrOfStr[1]=="") {
                                quality="";
                        }else {
                                quality = arrOfStr[1];
                        }
                        continue;
                case "Mint":
                case "Mint Facility":
                        if(arrOfStr[1]=="") {
                                mintFecility="";
                        }else {
                                mintFecility =
arrOfStr[1];

                        }
                continue;
                case "Year":
                case "Date":
                        if(arrOfStr[1]=="") {
                                date_year="";
                        }else {
```

```
                    date_year =
arrOfStr[1];
                }
                continue;
            case "Purity":
                if(arrOfStr[1]=="") {
                    purity="";
                }else {
                    purity = arrOfStr[1];
                }
                continue;
            case "Weight":
            case "Metal Weight":
                if(arrOfStr[1]=="") {
                    metalWeight="";
                }else {
                    metalWeight =
arrOfStr[1];
                }
                continue;
            case "Edge":
                if(arrOfStr[1]=="") {
                    edge="";
                }else {
                    edge = arrOfStr[1];
                }
                continue;
            case "Obverse":
                if(arrOfStr[1]=="") {
                    obverse="";
                }else {
```

```
                    obverse = arrOfStr[1];
                }
                continue;
            case "Reverse":
            if(arrOfStr[1]=="") {
                reverse="";
            }else {
                reverse = arrOfStr[1];
            }
                continue;
            case "Diameter":
                if(arrOfStr[1]=="") {
                    diameter="";
                }else {
                    diameter = arrOfStr[1];
                }
                continue;
            case "Thickness":
                if(arrOfStr[1]=="") {
                    thickness="";
                }else {
                    thickness =
arrOfStr[1];
                }
                continue;

            }

        }
```

```
                                        Allproductitems.add(new
AllProducts(id,Namedetails, category,
                                                        lowest_price, banner_image,
imgUrls,denomination,quality,mintFecility,date_year,

        purity,metalWeight,edge,obverse,reverse,diameter,thickness));
                                        imgUrls="";
                                        for (Product data : items) {
                                                productrepository.save(data);
                                        }
                                id++;
                                }
                        }

                } catch (Exception ex) {
                        ex.printStackTrace();
                }
                return Allproductitems;


        }


//getting specification
        private String getSpecification(Elements rowdata) {
                String[] arrOfStr;
                JSONObject SpecJsonvalue = new JSONObject();
                Elements spec = rowdata.select("div.item-tabs div.tab-content
div#prod_specs table");


                Elements tbody = spec.select("tbody");
                System.out.println("berfore insert"+SpecJsonvalue);
                for (Element row : tbody.select("tr")) {
                        String td_val = row.select("td").text();
```

```
 arrOfStr = td_val.split(":", 2);
SpecJsonvalue.put(arrOfStr[0], arrOfStr[1]);
switch (arrOfStr[0]) {
case "Denomination":

        if(arrOfStr[1]=="") {
                denomination="";
        }else {
                denomination = arrOfStr[1];
        }
        continue;
case "Quality / Type":
        if(arrOfStr[1]=="") {
                quality="";
        }else {
                quality = arrOfStr[1];
        }
        continue;
case "Mint":
case "Mint Facility":
        if(arrOfStr[1]=="") {
                mintFecility="";
        }else {
                mintFecility = arrOfStr[1];
        }
continue;
/*case "Year":
case "Date":
        date_year = arrOfStr[1];
        continue;
```

```
                case "Purity":
                        purity = arrOfStr[1];
                        continue;
                case "Metal Weight":
                        metalWeight = arrOfStr[1];
                        continue;
                case "Edge":
                        edge = arrOfStr[1];
                        continue;
                case "Obverse":
                        obverse = arrOfStr[1];
                        continue;
                case "Reverse":
                        reverse = arrOfStr[1];
                        continue;
                case "Diameter":
                        diameter = arrOfStr[1];
                        continue;
                case "Thickness":
                        thickness = arrOfStr[1];
                        continue;*/

                }

        }

        arrOfStr=ArrayUtils.EMPTY_STRING_ARRAY;
        specificationItems.add(new
Specification(denomination,quality,"",mintFecility));
        System.out.println("specificationItems"+specificationItems);
        return SpecJsonvalue.toString();
```

```
}

// To write the data to csv
private void csvwriter(List<AllProducts> items, String
categoryValue,String csvname) {
    // csv writer
    String exportDir = "D:/" + csvname + ".csv";
    File file = new File(exportDir);
    try {
        FileWriter outputfile = new FileWriter(file);
        CSVWriter writer = new CSVWriter(outputfile);
        String[] header = { "Id", "Name", "Category", "Price",
"ThumbUrl", "ImageUrl","Denomination","Quality",

"Mint","DateYear","Purity","metalWeight","Edge","Obverse","Reverse","
Diameter","Thickness"};
        // List<Product> datas=items;
        writer.writeNext(header);
        // writer.writeAll(datas);
        for (AllProducts product : items) {
            String[] data = { String.valueOf(product.getId()),
product.getName(), product.getProductCategory(),
                    product.getPrice(),
product.getBannerImageUrl(), product.getImageUrls(),product.getDenomination(),

product.getQuality(),product.getMint(),product.getDateYear(),product.getP
urity(),product.getMetalWeight(),

product.getEdge(),product.getObverse(),product.getReverse(),product.getDi
ameter(),product.getThickness()};
            writer.writeNext(data);
        }
        writer.close();
```

```
                } catch (Exception e) {
                        System.out.println("Error While writing CSV " + e);
                }

        }

}
```

ProductUrlService.java :

```
package com.example.webscrap.service;

import java.util.ArrayList;
import java.util.List;

import org.json.JSONArray;
import org.json.JSONObject;
import org.jsoup.Jsoup;
import org.jsoup.nodes.Document;
import org.jsoup.nodes.Element;
import org.jsoup.select.Elements;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;
import com.example.webscrap.service.ProductService;
@Service
public class ProductUrlService {
        @Autowired
        private ProductService productService;
```

```
public void getcoinUrls() {
        final String baseUrl = "https://www.moneymetals.com";
        ArrayList<String> silverUrls = new ArrayList<>();
        ArrayList<String> goldUrls = new ArrayList<>();
        ArrayList<String> productUrls = new ArrayList<>();
        String silverUrl="",goldUrl="",coinurl="",category="",csvname="";

        JSONArray  categoryArry = new JSONArray();
        JSONObject urlJsonobj = new JSONObject();
        JSONArray  urlArray = new JSONArray();
        try {


            Document Baseurldocument = Jsoup.connect(baseUrl).get();
            Elements baseProduct = Baseurldocument.select("body
div.wrap div#ilb-menu div#affix-menu div.menu-wrap div#mmx-collapse");
            Elements rows = baseProduct.select("ul.nav.navbar-nav
li.dropdown.mega-dropdown");
            int index=0;
            for (Element a : rows) {
                if(index<1) {
                silverUrl = a.select("a").attr("href");


                System.out.println("silver"+silverUrl);


                //geting all topicurls
                Document urldocument =
Jsoup.connect(silverUrl).get();


                Elements Product =
urldocument.select("section.categories");
```

```
                    //For silver
                    Elements row = Product.select("ul#silver_side li");
                    int silvercount=0;
                    for (Element element : row) {
                            csvname="Silver";
                            if(silvercount<2) {
                            JSONObject categoryJsonobj = new
JSONObject();
                            coinurl = element.select("a").attr("href");
                            String coinval=element.select("a").text();
                            categoryJsonobj.put("categoryUrl",coinurl);
                            categoryJsonobj.put("category",coinval);
                            categoryArry.put(categoryJsonobj);
                            }
                            silvercount++;
                    }
                    productService.getUrl(categoryArry,csvname);

                    categoryArry.clear();
                    //for gold
                    int goldcount=0;
                    Elements rowdata = Product.select("ul#gold_side
li");
                    for (Element element : rowdata) {
                            csvname="Gold";
                            if(goldcount<2) {
                            JSONObject categoryJsonobj = new
JSONObject();
                            coinurl = element.select("a").attr("href");
                            String coinval=element.select("a").text();
                            categoryJsonobj.put("categoryUrl",coinurl);
```

```
                                categoryJsonobj.put("category",coinval);
                                categoryArry.put(categoryJsonobj);
                                }
                                goldcount++;
                        }
                        productService.getUrl(categoryArry,csvname);

                }
                        index++;
                }

                System.out.println("urlArray"+categoryArry);
                } catch (Exception e) {
                e.printStackTrace();
        }

    }
}
```

MODELS :

AllProducts.java :

```
package com.example.webscrap.model;
import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
@Entity
```

```
public class AllProducts {
	@Id
   @GeneratedValue(strategy = GenerationType.AUTO)
	private Integer Id;
	private String Name;
	private String ProductCategory;
	private String price;
	private String BannerImageUrl;
	private String ImageUrls;

	private String Denomination;
	private String Quality;
	private String Mint;
	private String DateYear;
	private String Purity;
	private String metalWeight;
	private String Edge;
	private String Obverse;
	private String Reverse;
	private String Diameter;
	private String Thickness;
	public AllProducts(Integer id, String name, String productCategory, String price, String bannerImageUrl,
			String imageUrls, String denomination, String quality, String mint, String dateYear, String purity,
			String metalWeight, String edge, String obverse, String reverse, String diameter, String thickness) {
		super();
		Id = id;
		Name = name;
		ProductCategory = productCategory;
```

```
        this.price = price;
        BannerImageUrl = bannerImageUrl;
        ImageUrls = imageUrls;
        Denomination = denomination;
        Quality = quality;
        Mint = mint;
        DateYear = dateYear;
        Purity = purity;
        this.metalWeight = metalWeight;
        Edge = edge;
        Obverse = obverse;
        Reverse = reverse;
        Diameter = diameter;
        Thickness = thickness;
}
public Integer getId() {
        return Id;
}
public void setId(Integer id) {
        Id = id;
}
public String getName() {
        return Name;
}
public void setName(String name) {
        Name = name;
}
public String getProductCategory() {
        return ProductCategory;
}
```

```
public void setProductCategory(String productCategory) {
        ProductCategory = productCategory;
}
public String getPrice() {
        return price;
}
public void setPrice(String price) {
        this.price = price;
}
public String getBannerImageUrl() {
        return BannerImageUrl;
}
public void setBannerImageUrl(String bannerImageUrl) {
        BannerImageUrl = bannerImageUrl;
}
public String getImageUrls() {
        return ImageUrls;
}
public void setImageUrls(String imageUrls) {
        ImageUrls = imageUrls;
}
public String getDenomination() {
        return Denomination;
}
public void setDenomination(String denomination) {
        Denomination = denomination;
}
public String getQuality() {
        return Quality;
}
```

```
public void setQuality(String quality) {
        Quality = quality;
}
public String getMint() {
        return Mint;
}
public void setMint(String mint) {
        Mint = mint;
}
public String getDateYear() {
        return DateYear;
}
public void setDateYear(String dateYear) {
        DateYear = dateYear;
}
public String getPurity() {
        return Purity;
}
public void setPurity(String purity) {
        Purity = purity;
}
public String getMetalWeight() {
        return metalWeight;
}
public void setMetalWeight(String metalWeight) {
        this.metalWeight = metalWeight;
}
public String getEdge() {
        return Edge;
}
```

```
	public void setEdge(String edge) {
		Edge = edge;
	}
	public String getObverse() {
		return Obverse;
	}
	public void setObverse(String obverse) {
		Obverse = obverse;
	}
	public String getReverse() {
		return Reverse;
	}
	public void setReverse(String reverse) {
		Reverse = reverse;
	}
	public String getDiameter() {
		return Diameter;
	}
	public void setDiameter(String diameter) {
		Diameter = diameter;
	}
	public String getThickness() {
		return Thickness;
	}
	public void setThickness(String thickness) {
		Thickness = thickness;
	}
	@Override
	public String toString() {
		return "AllProducts [Id=" + Id + ", Name=" + Name + ",
ProductCategory=" + ProductCategory + ", price=" + price
```

```
                + ", BannerImageUrl=" + BannerImageUrl + ", ImageUrls=" + ImageUrls + ", Denomination=" + Denomination
                + ", Quality=" + Quality + ", Mint=" + Mint + ", DateYear=" + DateYear + ", Purity=" + Purity
                + ", metalWeight=" + metalWeight + ", Edge=" + Edge + ", Obverse=" + Obverse + ", Reverse=" + Reverse
                + ", Diameter=" + Diameter + ", Thickness=" + Thickness + "]";
        }
        public AllProducts() {
                super();
                // TODO Auto-generated constructor stub
        }
}
```

CONTROLLER :

WebscrapController.java :

```
package com.example.webscrap.controller;
import java.io.IOException;
import java.io.Writer;
import java.util.List;
import javax.servlet.http.HttpServletResponse;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.web.bind.annotation.*;

import com.example.webscrap.model.Product;
import com.example.webscrap.service.FileLoad;
import com.example.webscrap.service.ProductService;
import com.example.webscrap.service.ProductUrlService;
```

```
@RestController
public class WebscrapController {

	@Autowired
	private ProductService productService;

	@Autowired
	private FileLoad filload;

	@Autowired
	private ProductUrlService producturlservice;



	/*@RequestMapping(path="/hello",method = RequestMethod.GET)
	public void hello() {
		System.out.println("hellllooooo");

	}*/


	@RequestMapping(path="/data",method = RequestMethod.GET)
	public void getUrl() throws IOException, Exception  {

		producturlservice.getcoinUrls();
	}

	/*@RequestMapping(path="/url",method = RequestMethod.GET)
	public void getcoinUrls(HttpServletResponse servletResponse) throws
IOException {
```

```
            producturlservice.getcoinUrls();
        }

        @RequestMapping("/file")
        public List<String> getfile() throws Exception  {

            return filload.filedata();
        }*/
}
```

APPENDIX II – SCREENSHOTS

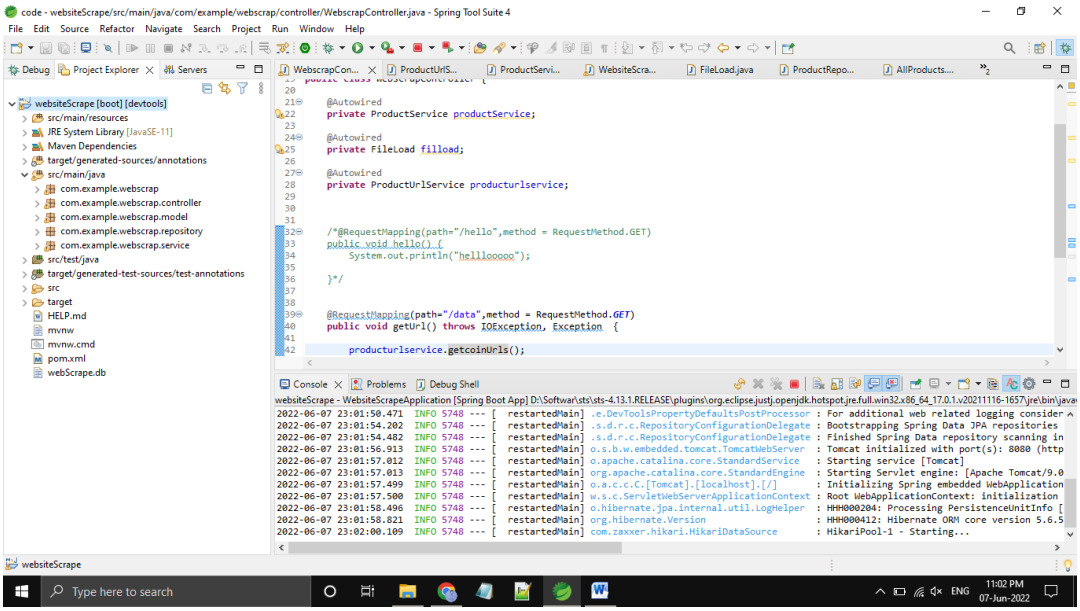

Fig 8.1 - Executing the code

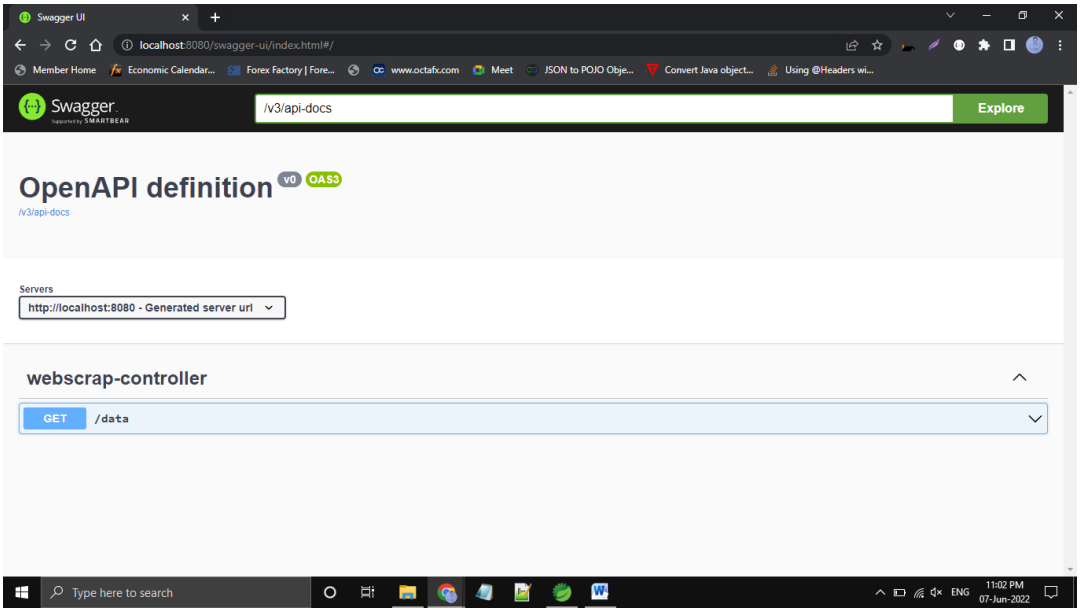

Fig 8.2 - Hit localhost in chrome

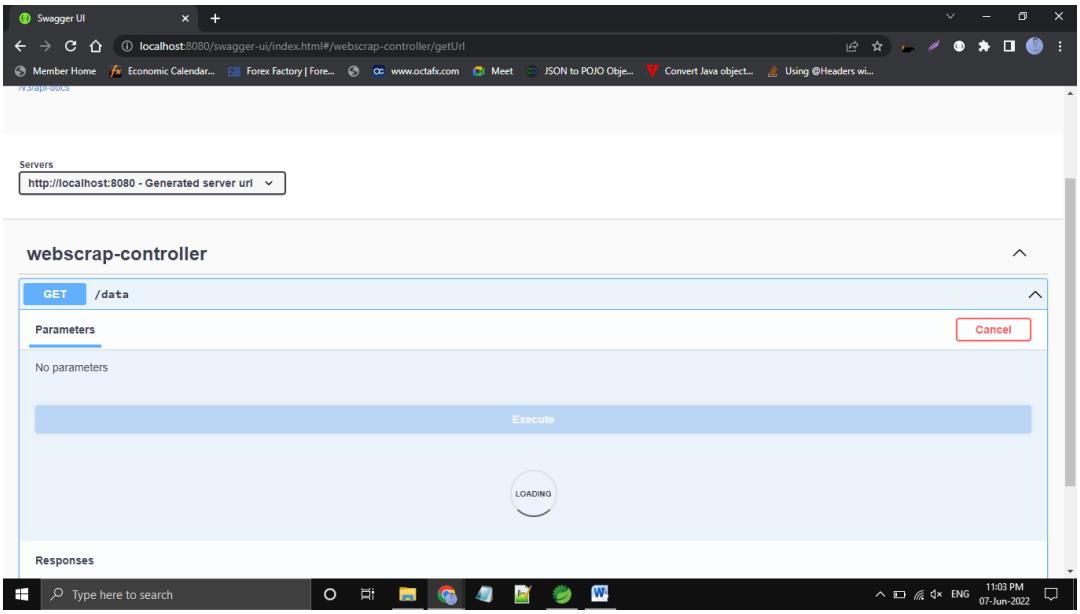

Fig 8.3 - Execute the function in localhost

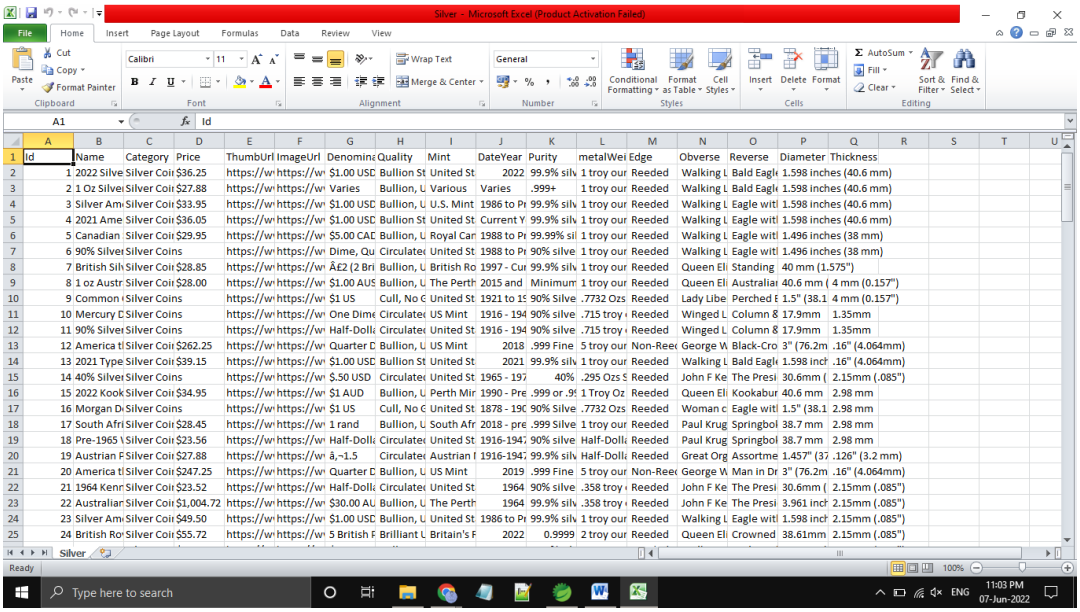

Fig 8.4 - Get website data in CSV file for destination path

REFERENCES

BIBLIOGRAPHY

1. "Search Engine History.com". Search Engine History. Retrieved November 26, 2019.
2. Semantic annotation based web scraping
3. Roush, Wade (2012-07-25). "Diffbot Is Using Computer Vision to Reinvent the Semantic Web". www.xconomy.com. Retrieved 2013-03-15.
4. "FAQ about linking – Are website terms of use binding contracts?". www.chillingeffects.org. 2007-08-20. Archived from the original on 2002-03-08. Retrieved 2007-08-20.
5. Kenneth, Hirschey, Jeffrey (2014-01-01). "Symbiotic Relationships: Pragmatic Acceptance of Data Scraping". Berkeley Technology Law Journal. 29 (4). doi:10.15779/Z38B39B. ISSN 1086-3818.
6. "What are the "trespass to chattels" claims some companies or website owners have brought?". www.chillingeffects.org. 2007-08-20. Archived from the original on 2002-03-08. Retrieved 2007-08-20.
7. "Ticketmaster Corp. v. Tickets.com, Inc". 2007-08-20. Retrieved 2007-08-20.

WEB REFERENCES

1) https://www.geeksforgeeks.org/what-is-web-scraping-and-how-to-use-it/
2) https://en.wikipedia.org/wiki/Web_scraping
3) https://www.tutorialspoint.com/jsoup/index.htm
4) https://aziryasin.medium.com/web-scraping-made-easier-with-jsoup-4c07734ec600
5) https://www.webscrapingapi.com/java-web-scraping/