Conference Paper

The Research of Automation of the Process of Indexing Tax Returns

Leonov P. Y., Ivanov N. V., and Kotelyanets O. S.

National Research Nuclear University MEPhI (Moscow Engineering Physics Institute), Kashirskoe shosse 31, Moscow, 115409, Russia

Corresponding Author:
Ivanov N. V.
ivanov.nikolay.711@gmail.com

Received: 11 December 2017
Accepted: 20 January 2018
Published: 13 February 2018

Publishing services provided by Knowledge E

Selection and Peer-review under the responsibility of the FinTech and RegTech: Possibilities, Threats and Risks of Financial Technologies Conference Committee.

Abstract

The article is devoted to the study of the automated search for information on tax declarations of different countries in public sources of various structures and the collection of information received in a single information storage. The first part of the paper describes methods of automated data collection and tasks that can be solved by these methods. The second part of the work describes the development of an algorithm for finding data on tax declarations from various sources and creating a prototype system that implements the data of the algorithm and provides access to the collected data.

Keywords: search systems, indexation of tax declarations, information retrieval system.

1. Introduction

There is an actual task of analyzing tax declarations of civil servants to identify suspicious information that may conceal some certain tax crimes. In order to effectively analyze multiple declarations, it is necessary to conduct an analysis of publicly available sources on the Internet to collect information on tax returns. If the entire set of declarations will be collected in one place, then in the future, they can be analyzed using automated means.

The purpose of this research work is to study the possibilities of automated indexing of public sources.

The tasks of this work are:

1. Study methods for automated collection of information from public sources.

2. Develop an algorithm for automated indexing of public sources based on existing methods,

How to cite this article: Leonov P. Y., Ivanov N. V., and Kotelyanets O. S., (2018), “The Research of Automation of the Process of Indexing Tax Returns” in *III Network AML/CFT Institute International Scientific and Research Conference “FinTech and RegTech: Possibilities, Threats and Risks of Financial Technologies”*, KnE Social Sciences, pages 542–551. DOI 10.18502/kss.v3i2.1588

3. Implement the designed algorithm in the form of a system that would provide a software and user interface for obtaining information about the accumulated data.

2. Methods of automated information collection from common sources

The crawler is, as indicated in [1], a system for downloading a large number of web pages. Search robots can be used for various needs, but in this case, its application in the field of Data Mining is important. The search robot process consists of the following sequence of actions, in accordance with [2]:

1. Obtaining an html-document on its IP-address or domain name.
2. Drawing up a list of references in the document.
3. Parallelizing the processing of links, if necessary.
4. Filtering links by a certain feature.
5. Elimination of references that are duplicated or have already been processed.
6. Prioritize links by a certain feature.
7. Transition to item 1.

In the case when the search robot is working to obtain certain data, it is necessary to introduce the term ”terminal reference”, the meaning of which is similar to the terminal state in the theory of automata. When a page hits a terminal link, the spider starts receiving data on the specified algorithm without continuing search for links in depth.

Thus, the task of searching data in a public source is reduced to the task of traversing an oriented state graph, avoiding cycles and considering some state nodes as terminal ones.

Scrapping (or retrieving data) is a process of programmatically retrieving data via the HTTP protocol [3]. Scrapper is a program designed to download a web page. Scrapers mimic the actions of the web browser, which is different from the search robot, whose main task is only to pass through the links on the site.

At the moment there are many freely available search robots. However, their main disadvantage is that they do not fully upload the web page, since any web page contains code written in the JavaScript language that is executed in the browser, and the download process of the web page is not available for expansion. Thus, the best way to handle a public source for searching for declarations is to use the crawler to navigate through the links, and to use the scrapper to load information and retrieve all the links on the page.

As a result, the algorithm for a complete crawl of one site with tax returns must consist of the following sequence:

1. Obtaining an html-document on its IP-address or domain name.
2. Full processing of all JavaScript scripts inside the document.
3. Drawing up a list of references in the document.
4. Filtering links by a certain feature.
5. Elimination of references that are duplicated or have already been processed.
6. Transition to item 1.

3. Development of the automated system prototype

To develop a system that could handle the largest number of publicly available sources, it is necessary to present the following requirement to the system:

- Using a combined approach to information collection.
- Actions that can be performed automatically must be taken as the basis of the algorithm, the user part should consist in indicating the features of a particular source for the correct operation of the algorithm.

Requirements for the information system:

- It is necessary to store the data.
- Provide a programming interface for reading data.
- Provide the ability to export data in the specified format.
- Provide a user interface for reading data and for managing the system.

4. Choice of development technologies

To develop the algorithm for collecting information, the Scala language and the scala-scraper library were chosen. Scala is a JVM-compatible functional programming language, scala-scraper provides the possibilty to get a complete web page by reference and DSL for parsing the resulting page [4].

In addition to the module for parsing sites in the program, there are also data storage modules, indexing in in-memory storage, export, API and web applications. These modules were developed in the Java language using the Spring framework. This combination provides an opportunity to create extensible corporate systems of any complexity, providing multi-threaded system management, extensibility and configuration.

5. Development of an algorithm for collecting information

The above was a sequence of actions for finding declarations on one site. But since there are several sites, a general algorithm has been developed for passing through a multitude of sites.

The crawler loads the web page without executing JavaScript scripts at the speed of HTTP data packets, but in the case of sharing web scrapping, the time increases due to the need to process all the scripts present on the page. Thus, processing of even one site turns into a long operation, therefore it was decided that the general algorithm of data processing will be performed in the background multithreaded mode.

Additionally, the schedule for indexing was introduced into the algorithm, because executing several indexations in the same time period is a resource intensive operation, therefore, it is necessary to separate indexations of different sites in time. The choice of the site to be indexed is done by sorting the indexed sites according to the time of the last indexation in descending order - the site that was not indexed the longest, appears in the head of the list. In addition, it should be noted that the administrator of the system can forcefully index the site of a particular country, therefore, the tasks for indexing from the administrator are viewed first. After selecting a site, its indexing starts. Users of the system see the indexed results in real time, however, during the indexing, a warning is displayed to the user.

To prevent tracking of the crawler, the ability to download web pages via the Proxy server has been added.

As a result, the following indexing settings are provided to the user:

- The interval between downloading the next web page (some sites can block IP, that sends requests very often).
- The interval between the indexing of declarations of one country (in days).
- Schedule indexer scheduler (in the form of cron-expression, used in Linux operating systems to form the schedules of processes).
- Proxy server settings.

It was decided to check the efficiency of the algorithm on the website of the tax returns in Bulgaria, as its structure is simpler than that of similar sites in other countries. The diagram of the classes of the realized information subsystem is shown in Fig. 1:

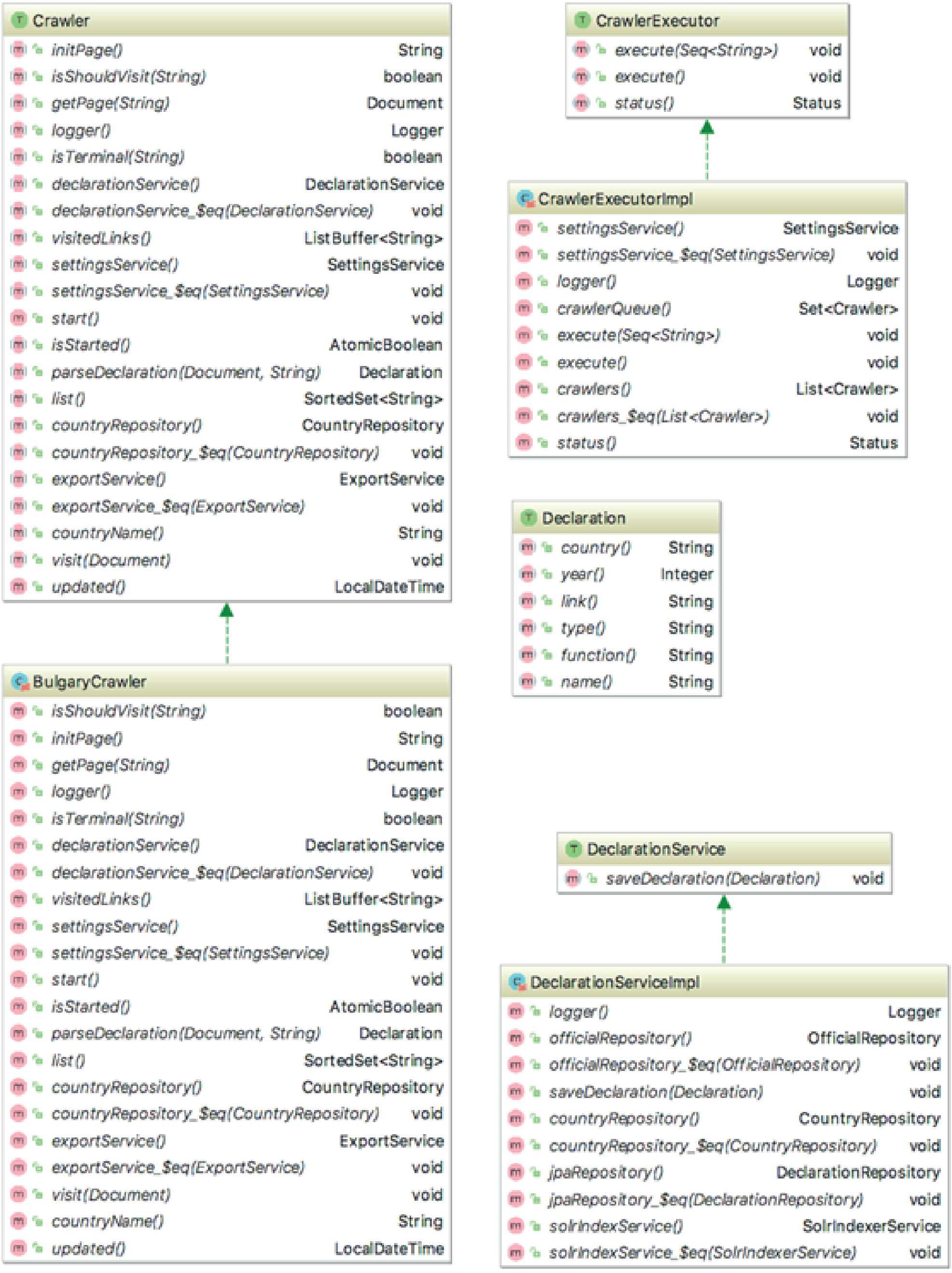

Crawler
initPage() String
isShouldVisit(String) boolean
getPage(String) Document
logger() Logger
isTerminal(String) boolean
declarationService() DeclarationService
declarationService_$eq(DeclarationService) void
visitedLinks() ListBuffer<String>
settingsService() SettingsService
settingsService_$eq(SettingsService) void
start() void
isStarted() AtomicBoolean
parseDeclaration(Document, String) Declaration
list() SortedSet<String>
countryRepository() CountryRepository
countryRepository_$eq(CountryRepository) void
exportService() ExportService
exportService_$eq(ExportService) void
countryName() String
visit(Document) void
updated() LocalDateTime
CrawlerExecutor
execute(Seq<String>) void
execute() void
status() Status
CrawlerExecutorImpl
settingsService() SettingsService
settingsService_$eq(SettingsService) void
logger() Logger
crawlerQueue() Set<Crawler>
execute(Seq<String>) void
execute() void
crawlers() List<Crawler>
crawlers_$eq(List<Crawler>) void
status() Status
Declaration
country() String
year() Integer
link() String
type() String
function() String
name() String
BulgaryCrawler
isShouldVisit(String) boolean
initPage() String
getPage(String) Document
logger() Logger
isTerminal(String) boolean
declarationService() DeclarationService
declarationService_$eq(DeclarationService) void
visitedLinks() ListBuffer<String>
settingsService() SettingsService
settingsService_$eq(SettingsService) void
start() void
isStarted() AtomicBoolean
parseDeclaration(Document, String) Declaration
list() SortedSet<String>
countryRepository() CountryRepository
countryRepository_$eq(CountryRepository) void
exportService() ExportService
exportService_$eq(ExportService) void
visit(Document) void
countryName() String
updated() LocalDateTime
DeclarationService
saveDeclaration(Declaration) void
DeclarationServiceImpl
logger() Logger
officialRepository() OfficialRepository
officialRepository_$eq(OfficialRepository) void
saveDeclaration(Declaration) void
countryRepository() CountryRepository
countryRepository_$eq(CountryRepository) void
jpaRepository() DeclarationRepository
jpaRepository_$eq(DeclarationRepository) void
solrIndexService() SolrIndexerService
solrIndexService_$eq(SolrIndexerService) void

Figure 1: Class diagram of the information assembly module.

The module consists of the following programming interfaces:

1. Crawler - interface for passing through the site;

2. Declaration - interface for obtaining declaration data;
3. DeclarationService - interface for processing the declaration;
4. CrawlerExecutor - the interface for running Crawler on a schedule.

For the DeclarationService interface, a default implementation was created. The remaining interfaces are implemented depending on which sites need to be processed. It is assumed that the system administrator provides one implementation of the Crawler and Declaration interfaces for each site. The implementation of the Crawler interface is to specify the following parameters:

- Name of the country.
- The root link from which to start indexing.
- A symptom by which the terminal reference is determined.
- Algorithm for extracting data about the declaration from the web page to which the terminal link leads.

6. Storage Module

The system stores data about the following objects:

1. Countries
2. Declarations
3. Civil servants
4. Users of the system
5. System Settings

The database schema is shown in Fig. 2:

Data management in the system is carried out through ORM, therefore any DBMS can be used as a DBMS. For carrying out automated testing of the system, the DBMS H2 is used, for startup - PostgreSQL. You can change the data about the relational database in the project configuration file.

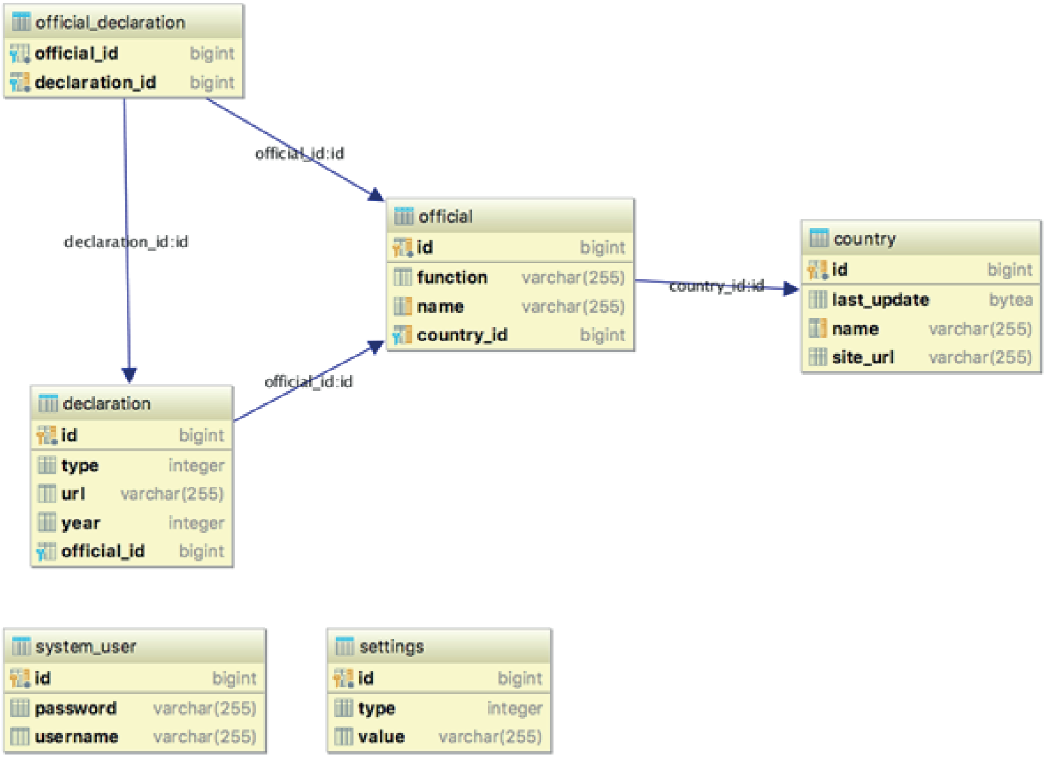


Figure 2: System database schema.

7. Data export module

The data export module uses the system settings - what type of export to use, in which directory to write. The module provides a service for indexing data for a specific country.

The diagram of the classes of the data export module is shown in Fig. 3:

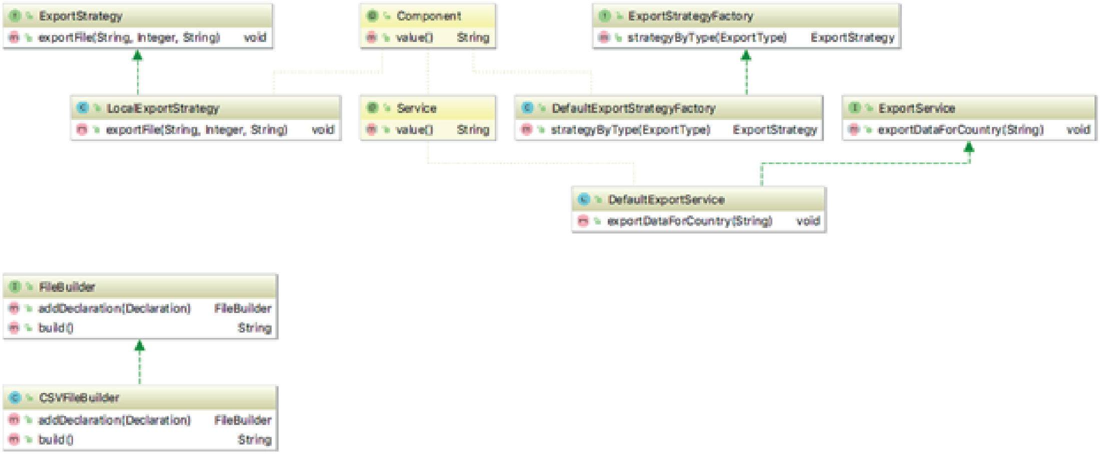

ExportStrategy
exportFile(String, Integer, String) void
Component
value() String
ExportStrategyFactory
strategyByType(ExportType) ExportStrategy
LocalExportStrategy
exportFile(String, Integer, String) void
Service
value() String
DefaultExportStrategyFactory
strategyByType(ExportType) ExportStrategy
ExportService
exportDataForCountry(String) void
DefaultExportService
exportDataForCountry(String) void
FileBuilder
addDeclaration(Declaration) FileBuilder
build() String
CSVFileBuilder
addDeclaration(Declaration) FileBuilder
build() String

Figure 3: Class diagram of the data export module.

8. Data Indexing Module

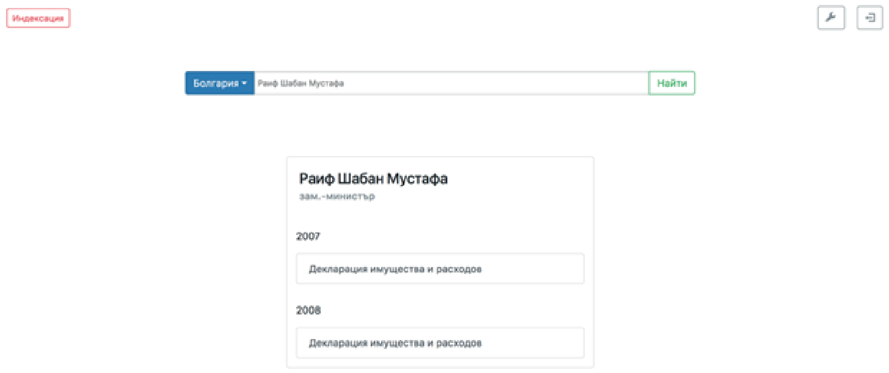
When using the program interface and simultaneous indexing of sites, a high load on the database is possible, so it was decided to duplicate the received declarations in the data store, which would be located not in the hard disk, but in RAM. This reduced the load on the database, accelerated the receipt of data from the database and provided an opportunity for full-text search of civil servants, which include declarations. As a software tool, it was decided to use ElasticSearch, which in addition provides the ability to scale storage to multiple machines, and also supports the removal of storage to the cloud.

The indexing of declarations in memory occurs as the declarations are stored in the database. When the system is turned off, the data is stored on the hard disk, when it is turned back on, it is returned to the main memory.

9. Web application module

For interaction of the user with the system, an API was developed, consisting of a set of http-requests that allow obtaining information on declarations, system settings, scanning status and indexed civil servants.

To manage the system, a web interface was developed. One need to be authorized to access. Figure 4 shows the main interface with search results by the name of a civil servant.



Раиф Шабан Мустафа

Раиф Шабан Мустафа
зам.-министър

2007

Декларация имущества и расходов

2008

Декларация имущества и расходов

Figure 4: System administrator user interface.

The general diagram of the interaction of the administrator with the system is shown in Fig. 5:

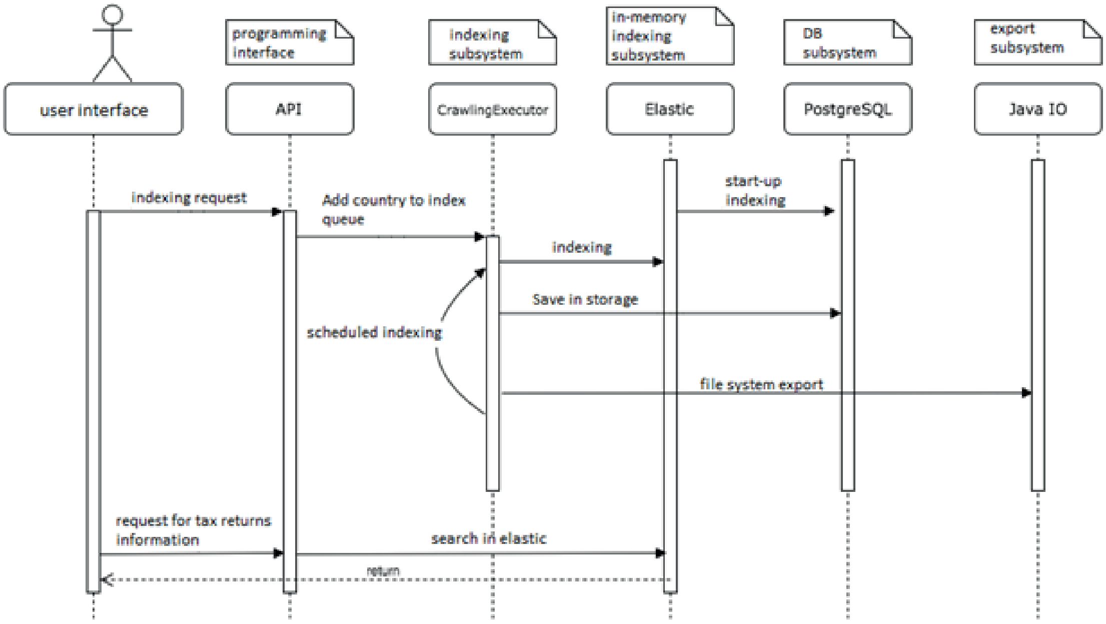


Figure 5: Diagram of the interaction sequence of the system modules.

10. Conclusion

As a result the possibilities of automated tools for searching and accumulating information were studied, an algorithm was developed for automated indexing of tax declarations using a search robot and web-scrapping. The designed algorithm was implemented as part of an information system that indexes declarations in the background and provides the system administrator with a software and user interface for obtaining indexed data. The developed prototype is used in one of the graduate qualification works of the students of the Institute of Financial and Economic Security of the NRNU MEPhI. This prototype can be extended and refined for practical use and integration with data analysis systems to identify tax returns with suspicious data.

Acknowledgements

This work was supported by Competitiveness Growth Program of the Federal Autonomous Educational Institution of Higher Education National Research Nuclear University MEPhI (Moscow Engineering Physics Institute).

References

[1] Patil, Yugandhara и Patil, Sonal. www.ijarcce.com. [Internet] 16 January 2016 http://www.ijarcce.com/upload/2016/january-16/IJARCCE%2052.pdf

[2] Cambridge University Press. [Internet] 1 April 2009 https://nlp.stanford.edu/IR-book/pdf/20crawl.pdf

[3] Sushitha, S, etc. Patents and Publications Web Scraping. International Journal of Computer Science and Network. 2016, V. 5, 2.

[4] Scala Scraper. *GitHub.com.* [Internet] 2016 https://github.com/ruippeixotog/scala-scraper.