Univerzita Hradec Králové
Fakulta informatiky a managementu
Katedra informatiky a kvantitativních metod

Web scraping

Diplomová práce

Autor: Michal Kozderka
Studijní obor: Aplikovaná informatika

Vedoucí práce: Ing. Pavel Kříž, Ph.D.

Hradec Králové Duben 2016

Prohlášení

Prohlašuji, že jsem diplomovou práci zpracoval samostatně a s použitím uvedené literatury.

V Hradci Králové dne Michal Kozderka

Poděkování

Na tomto místě bych rád poděkoval vedoucímu práce Ing. Pavlu Křížovi, Ph.D., za poskytnutí tématu a možnost vytvářet práci pod jeho vedením.

Anotace

Diplomová práce se zabývá získáváním dat z internetu. V rámci práce bylo porovnáno několik nástrojů pro získávání dat z internetu. Po analýze těchto nástrojů byl navržen nový nástroj, který odstraňuje některé nedostatky nástrojů analyzovaných. Nástroj vychází z několika případů užití. Následně byly vybrány technologie pro jeho implementaci. Nakonec byl nástroj naimplementován a otestován na vybraných případech užití.

Annotation

Title: Web scraping

The topic of the Diploma Thesis is internet data mining. Several web scraping tools were compared and analysed. Based on the results of this process a new tool was designed. It eliminates some of their flaws and can be used in several defined areas. Technologies for its implementation were selected and the tool was afterwards implemented and tested in the defined areas.

Obsah

1 Úvod

Během studia na Univerzitě Hradec Králové mi bylo nabídnuto několik témat diplomových prací. Vybral jsem si Web Scraping. Toto téma mi je blízké, protože pracuji ve firmě, kde vyvíjíme webové aplikace, a vím, že získat potřebná a aktuální data je obtížné. Při zpracování tohoto tématu si mohu vyzkoušet získávání dat z internetových stránek a následně je uložit ve strukturované formě pro další použití.

Diplomová práce se bude zabývat získáváním dat z internetových stránek. V diplomové práci jsou porovnávána stávající řešení, je provedeno shrnutí jejich výhod a nevýhod. Dále je navržen a naprogramován nástroj, který se bude snažit využít výhody a opravit nevýhody nástrojů konkurenčních. Výstupem bude strukturovaný soubor se získanými údaji, které se následně budou používat k další analýze.

V návrhu aplikace bude kladen důraz na uživatelský komfort. Uživatel nemusí být z oboru IT a nemusí rozumět struktuře internetových stránek, musí být však poučen, jak obsluhovat daný nástroj podle návodu. Aplikace musí být spustitelná jak na uživatelském počítači, tak na serveru. Z tohoto důvodu budou zvoleny i vybrané technologie pro vývoj.

Celá aplikace se bude skládat ze dvou částí. První část bude sloužit k vytvoření konfigurace pro scrapování. Tato část bude pouze na klientské stanici. Druhá část bude sloužit k získávání dat a k jejich následnému ukládání. Tato část již bude spustitelná na serveru i na běžném počítači.

Pro vývoj bude použit jako programovací jazyk javascript. Klientská část aplikace bude naprogramována jako doplněk pro webový prohlížeč Chrome, sloužící pro nastavení scrapování. Druhá část využije Node.js a headless browser[1] pro průchod internetovými stránkami a zisk potřebných dat.

[1] Internetový prohlížeč, který nemá grafické uživatelské rozhraní. Lze ho ovládat z příkazové řádky.

2 Web scraping

Web scraping je termín pro různé metody, používané pro získání dat z webových stránek. Nástroje pro web scraping simulují chování uživatele ve webovém prohlížeči, na webových stránkách, a mají za cíl automaticky získávat data pro účely následné další analýzy. Pro web scraping existuje mnoho nástrojů, od jednoduchých jednoúčelových skriptů simulujících kopírování uživatele pomocí CTRL-C CTRL-V, až po komplexní software umožňující široké nastavení možností.[6]

Web scraping má široké využití. Využívá se k získávání dat pro marketingový nebo vědecký výzkum. Společnosti pomocí scrapingu získávají data k analýze konkurence na trhu, a mohou tak pružně reagovat na změny trhu. V marketingu se scraping používá pro získávání kontaktů na společnosti, osoby atd. [6]

Na web scraping následně navazuje datamining. Datamining slouží k analýze velkého objemu dat, z nichž se snaží získat užitečné informace a vztahy mezi nimi. [6]

V dnešní době se web scraping velmi rychle rozvíjí. Stále více se také podílí na celkovém provozu sítě. Podle The scraping threat report 2015 v roce 2015 bylo považováno 22% návštěvníků internetových stránek za scrapovací nástroje. [6]

3 Získávání dat z internetu

V dnešní době vyspělých informačních systémů by se mohlo zdát, že bude snadný přístup k libovolným informacím. Bohužel opak je pravdou. Například státní instituce, které by měly umožňovat získat data snadno a ve strukturované podobě, tak mnohdy nečiní. Často se snaží získávání dat všemožně ztížit různými omezeními nebo poskytováním dat v podobě, ze které se špatně získávají. Dalším příkladem, kdy je třeba získávat data, je konkurenční boj, v němž je důležité mít aktuální informace o konkurenci. Například pokud uživatel provozuje internetový obchod, je pro něj důležité mít informace o cenách konkurenčních internetových obchodů, aby byl cenově konkurenceschopný. Takovýchto příkladů, kdy je potřeba získávat data z internetu, je obrovské množství.

Díky výše zmíněným potřebám si musí uživatelé vystačit buďto s omezeným a mnohdy velmi nepohodlným přístupem k potřebným datům, nebo si musí pořídit specializovaný nástroj. Jednodušší nástroje se dají nalést zdarma. Pokud by byl potřeba sofistikovanější nástroj, museli by za něj uživatelé zaplatit. Některé nástroje potřebují k obsluze uživatele, který musí být expertem na daný nástroj, aby ho mohl nastavit podle potřeb.

Hlavním cílem takovýchto aplikací je umožnit stahovat data z webových stránek běžným uživatelům, kterým je poskytnut návod, podle kterého jsou schopni nástroj obsluhovat. Problém lze rozdělit na dva podproblémy. Prvním podproblémem je zachycení chování uživatele na webové stránce a výběr elementů, ze kterých se budou data získávat. Druhým podproblémem je vlastní proces získávání dat.

3.1 Nastavení scrapování

Nastavení scrapování se skládá z výběru elementů na cílové webové stránce, aniž by musel uživatel znát XPath, CSS selektory nebo jiné prostředky, pomocí kterých může vybírat jednotlivé elementy HTML dokumentu. Uživatel by měl mít možnost vidět graficky znázorněný element, který vybírá. Pokud má uživatel aplikace potřebné znalosti, měl by mít možnost si upravit výběr prvků ručně. Další funkčností je možnost nahrát chování uživatele na webových stránkách. Nahrávání musí umět zachytit nastavení formulářů a další kroky na webových stránkách. Formulářové prvky musí být konfigurovatelné tak, aby mohl být formulář vyplněn opakovaně podle předdefinovaných kombinací. Tyto kombinace umožní rychlé nastavení scrapování, aniž by uživatel musel pro každou

kombinaci nahrávat jeho chování na stránkách. Během scrapování musí být uživatel informován o stavu procesu scrapování. Ideálním řešením informování o průběhu scrapování je zobrazovat přímo průchod scrapovacího nástroje webovými stránkami.

3.2 Scrapování dat

Scrapování dat musí umět podle konfigurace přejít na počáteční stránku scrapovaného webu. Podle konfigurace musí umět nastavit formulář, provést další kroky pro přechod na cílovou stránku a získat z cílové stránky potřebná data. Data se následně ošetří podle předdefinovaných filtrů, např. získání státu, URL obrázku, odmazávání HTML entit a dalších filtrů, které daný uživatel bude potřebovat. Během scrapování stránek dochází k vypisování logů o průběhu a vytváření screenshotů právě navštívené webové stránky. Vytváření těchto logů a screenshotů umožní uživateli získat přehled o právě běžícím procesu scrapování. Vlastní proces scrapování musí být možné spustit jak na serveru, tak na lokálním počítači.

Aplikace by měla umět využívat proxy servery, aby mohla opakovaně získávat data ze serverů, kde dochází k omezení přístupu na IP adresy nebo k omezení velikosti získaných dat na jednu IP adresu. Některé servery umožňují obsloužit pouze určitý počet requestů za den. Pokud by došlo k překročení, může server zablokovat danou IP adresu. Z tohoto důvodu je potřeba kontrolovat i počet kroků.

Jelikož aplikace bude nainstalována i na serveru, je zapotřebí ji zabezpečit jednoduchým autentizačním mechanismem.

4 Analýza existujících řešení

Při rozhodování o výběru vhodného nástroje je nutné vzít v potaz uživatelské prostředí, možnosti nastavení a cenu za používání.

4.1 Import.io

Import.io je nástroj pro získávání dat z internetových stránek. Nástroj je distribuován zdarma, pro jeho používání je třeba se registrovat. Pro funkčnost je nutné založit si účet a stáhnout nástroj pro nastavení scrapování. Grafické uživatelské rozhraní je k náhledu na Obrázek 1. Účet je potřeba mít založený z důvodu získání scrapovaných dat. Po přihlášení je možné si prohlédnout všechny konfigurace scrapování, stáhnout nebo prohlédnout si získaná data nebo další informace o scrapování. Scrapování probíhá každých 12 hodin, bohužel nelze nastavit jiný časový interval. Rovněž nelze nastavit proxy servery.[9]

Obrázek 1 – Výběr elementu v Import.io

Desktopová aplikace slouží pro nastavení scrapování. Po instalaci se objeví jakýsi webový prohlížeč a nahoře lišta sloužící pro nastavení. Samotná konfigurace je jednoduchá a přehledná. Výběr elementů na stránce je realizován klikem na element, ze kterého se získají data. Aplikace umožňuje nastavit tři druhy získávání dat:

- extractor,
- connector,
- crawler.

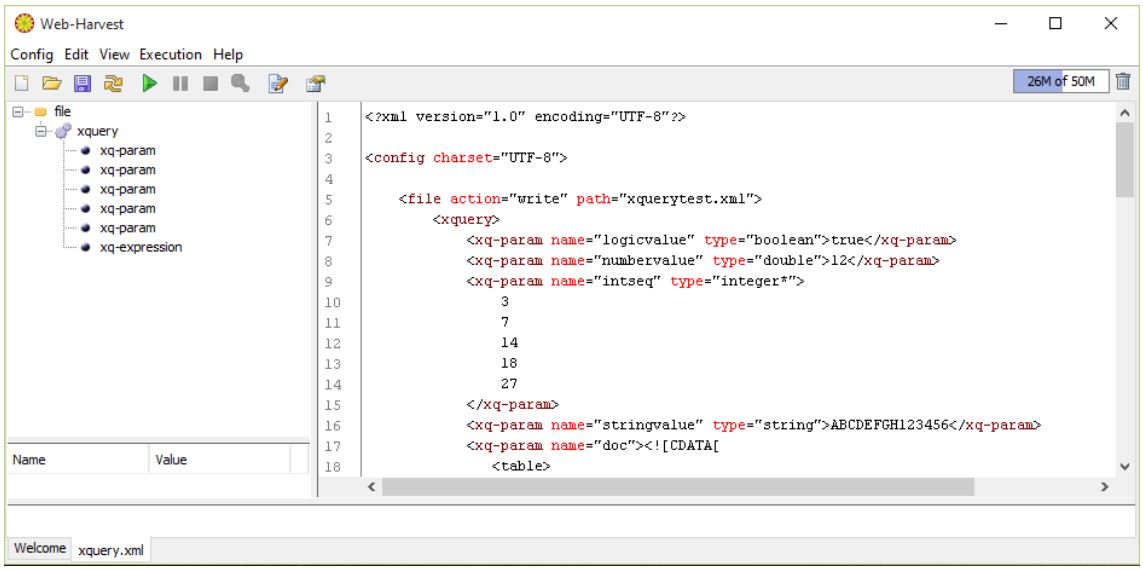
Extractor je nejjednodušší způsob získávání dat. Stahuje nadefinovaná data z jedné stránky. Jedna konfigurace se může použít pro různé weby, které mají stejnou strukturu HTML.[9]

Crawler obsahuje konfiguraci extractoru, s jejíž pomocí se vydefinují data, která jsou potřeba získat. Hlavní rozdíl oproti extractoru je, že crawler prochází celý web. Omezit prohledávání webu jde pomocí URL vzoru.[9]

Connector je nejkomplexnější nástroj. Connector umí vyplňovat formuláře a klikat na elementy na stránce. V connectoru lze nastavit i stránkování. Nastavení výběru dat je opět stejné jako v extractoru.[9]

4.2 Web-Harvest

Web-Harvest je open-source nástroj napsaný v Javě. Nástroj se ovládá z GUI zobrazeném na Obrázek 2 nebo z příkazové řádky. Pro extrakci dat je potřeba nakonfigurovat nástroj pomocí XML. V XML se uvede, odkud a jaká data se mají stahovat. Pro extrakci dat jsou použity XSLT, XQuery nebo regulární výrazy. Data se mohou extrahovat i přímo z databáze. Extrahovaná data se mohou ukládat do databáze nebo do souboru ve formátu XML nebo CSV. V jakém formátu budou data uložena, je čistě na konfiguraci. Tento nástroj se spouští z lokální stanice, ale je možné ho spustit i na serveru. Rozsah webu může být omezen podle URL nebo pomocí průchodu přes stránkování. Opakované stahování lze řešit pomocí cronu[2]. Zablokování IP adres lze řešit pomocí proxy serverů. Pro běžné uživatele, kteří nejsou z oboru, je tento nástroj nepoužitelný, protože by ho sami nedokázali nastavit.[10]

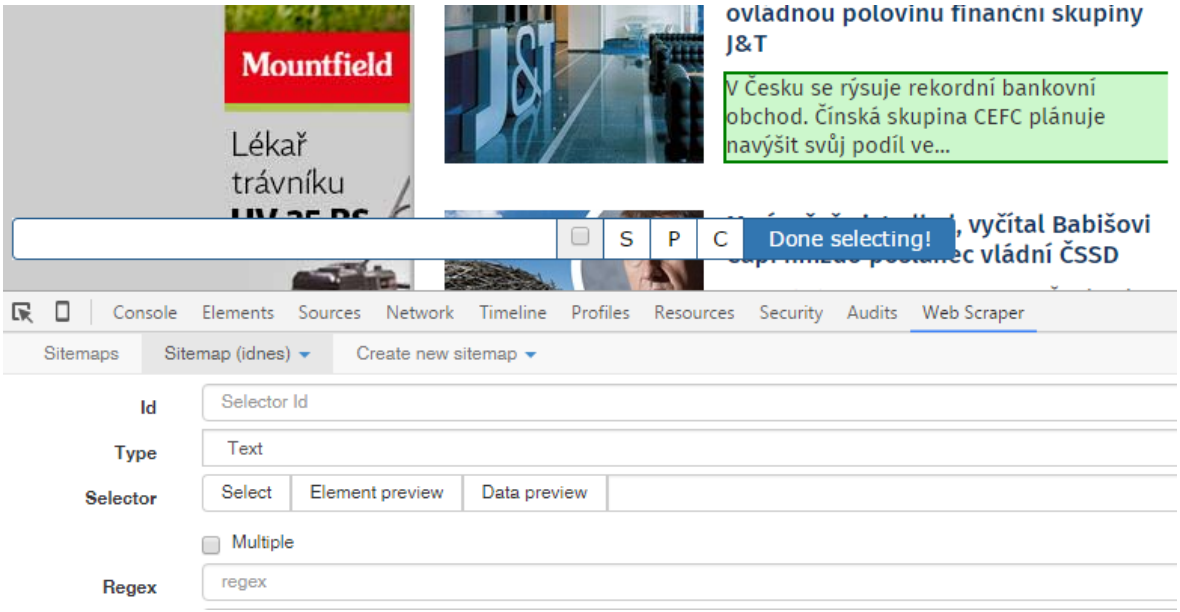


Obrázek 2 – Prostředí programu Web Harvest

[2] Systémový plánovač úloh, umožňující opakované spouštění procesů.

4.3 Web Scraper

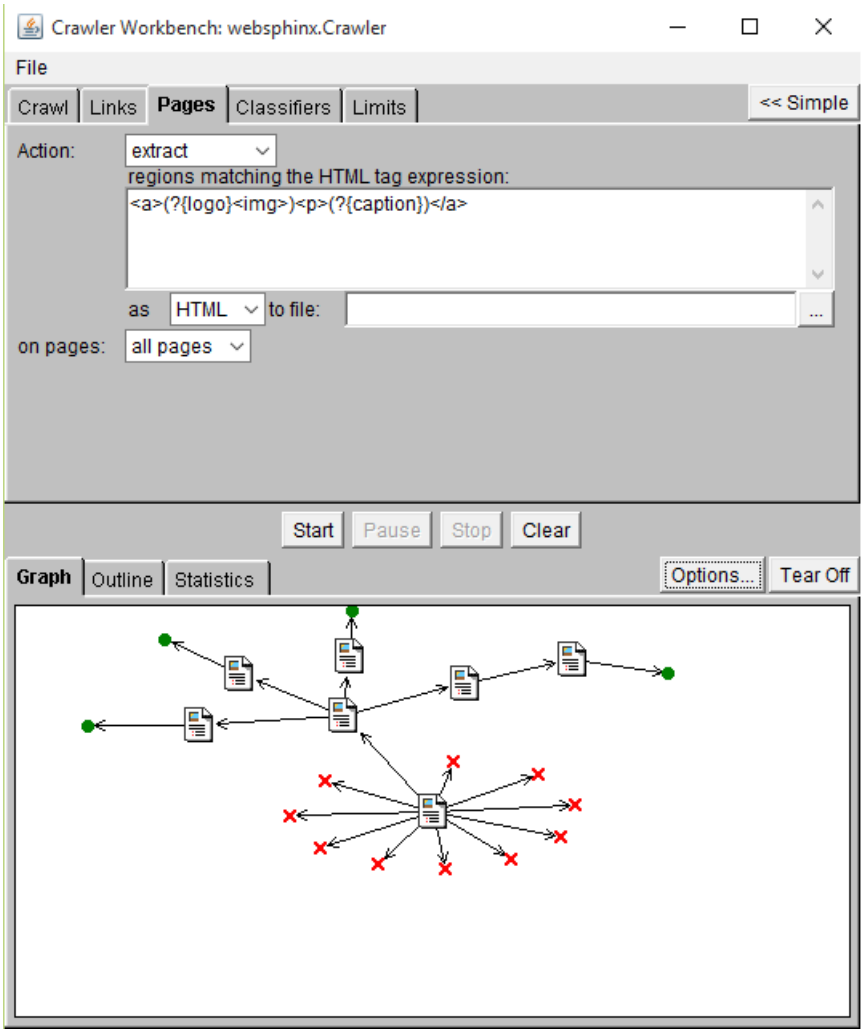
Web Sraper je rozšíření pro prohlížeč Chrome. Jednoduchý scrapovací nástroj, umožňující pomocí naklikání určit elementy na stránce. Umožňuje určit element i pomocí CSS selektoru. Vše se nastavuje ve vývojářské konzoli. Omezení prohledávání je možné jen na úrovni selektorů. Získaná data jsou uložena v CSV souboru. Po spuštění prohledávání je v okně vidět, které stránky jsou navštíveny. Opakované pravidelné prohledávání se zde neřeší. Zablokování IP adresy lze obejít pomocí proxy serveru. Tento nástroj je uživatelsky přívětivý, avšak na rozdíl od Import.io obsahuje malé možnosti konfigurace. Náhled obrazovky pro výběr elementu na stránce je na Obrázek 3.[11]



Obrázek 3 – Ukázka použití Web Scraper

4.4 Web SPHINX

Web SPHINX je open-source nástroj napsaný v Javě. Ovládá se z GUI. Pro extrakci je třeba vytvořit předpis, podle kterého se určí, jaké prvky se mají vybrat. Získaná data se ukládají do souborů. Rozsah webu umožňuje omezit jen na určité URL. Spuštění je z lokální stanice. Zablokování IP adresy se dá obejít přes proxy server. Opakované stahování lze řešit pomocí cronu. Web SPHINX není uživatelsky příliš přívětivý, i když používá GUI. Pro nastavení scrapování je třeba znát CSS selektory. V porovnání s Import.io má velmi malé možnosti jakéhokoliv nastavení. Vzhled GUI je ukázán na Obrázek 4.[13]



Obrázek 4 – Ukázka prostředí WebSPHINX

4.5 Výsledky analýzy

V analýze stávajících řešení nebyly popsány další nástroje, jako je třeba WebHarvy, z důvodu velké podobnosti s již popsanými nástroji. Uvedený WebHarvy je velmi podobný nástroji Import.io. Analýzou stávajících řešení bylo zjištěno, že žádný z nalezených nástrojů nesplňuje všechna očekávání.

Aplikace, které jsou zdarma, neposkytují takový komfort a možnosti nastavení. V některých musí mít uživatel pokročilé znalosti, aby mohl nástroj správně nastavit a používat. Placené aplikace poskytují uživatelům větší komfort a dokážou je obsluhovat poučení uživatelé. Možnosti nastavení jsou rovněž široké. I u placených aplikací se ale setkáme s určitými omezeními, která můžou být velmi nepříjemná. Nejlepší volbou z analyzovaných nástrojů je Import.io. Tento nástroj má však zásadní omezení a to nemožnost nastavit proxy server, a pokud by došlo k blokaci IP adres serveru Import.io,

pak by již pomocí tohoto nástroje nešlo získávat data ze serveru, který IP adresu zablokoval. Další nepříjemností může být absence možnosti naplánovat časy automatického pravidelného spouštění na serveru.

Protože žádný nástroj nesplňuje očekávání, je rozhodnuto vytvořit scrapovací nástroj, který by vyřešil nedostatky stávajících nástrojů.

5 Analýza a návrh implementace

Než se začne vytvářet samotný nástroj, je potřeba zaznamenat případy užití a provést analýzu technologií, která umožní použít nástroj ve všech případech užití.

5.1 Případy užití

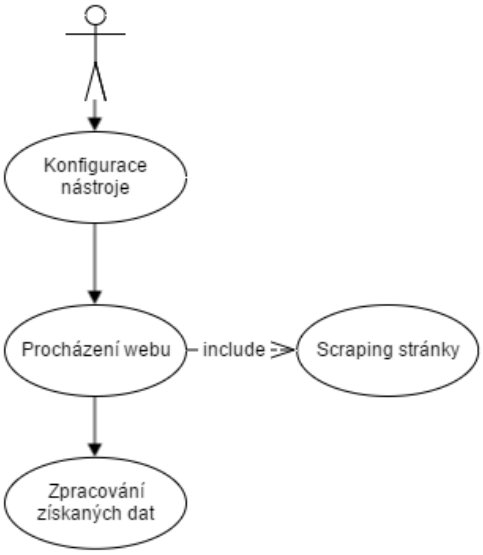
Během testování různých nástrojů pro web scraping a analýzy řešeného problému vzniklo několik scénářů pro získávání dat z webových stránek.

V prvním scénáři se bude procházet web pomocí nahraných kroků až na požadovanou stránku. Tento případ použití je graficky znázorněn na Obrázek 5. Nejprve se otevře web na úvodní stránce a postupně se, podle konfigurace, nastavují formulářové prvky a kliká se na stránku, dokud se neprojdou veškeré nahrané kroky. Po posledním kroku se vyhodnotí, zda jsou na stránce požadovaná data. Celý tento proces se může opakovat dle nastaveného konfiguračního souboru, ve kterém se nacházejí různé kombinace nastavení formulářových prvků. Tento postup se aplikuje na získání dat, například ze stránky Justice.cz.

Obrázek 5 – Přechod na požadovanou stránku pomocí nahraných kroků

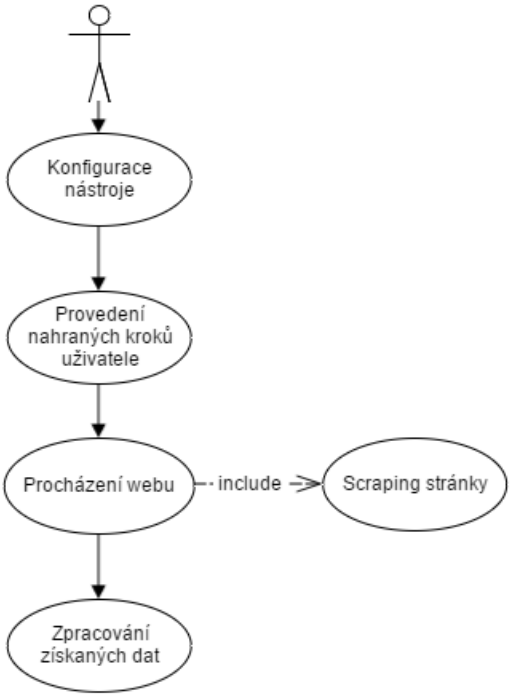
Další případ použití využívají i různí roboti z vyhledávačů, kteří postupně procházejí celý web a jednotlivé stránky indexují. V tomto případě se přejde na počáteční stránku a vyberou se z ní všechny odkazy, které se dají do fronty a postupně se procházejí. Na každé procházené stránce se vyberou všechny odkazy a přidají se znovu do fronty. Do

fronty se přidávají pouze unikátní odkazy, aby nedocházelo k několikanásobnému procházení stejné stránky. Na každé stránce dochází k vyhodnocení, zda stránka obsahuje potřebná data. Tento způsob lze využít například při získávání dat produktů na e-shopu. Tento případ užití je graficky znázorněn na Obrázek 6.



Obrázek 6 – Průchod e-shopu

Poslední způsob znázorněný na Obrázek 7, je kombinací předchozích dvou způsobů. V tomto způsobu je potřeba se nejdříve přihlásit do systému, aby bylo možné se dostat do ostatních sekcí webu, a následně se spustí procházení celého webu.



Obrázek 7 – Průchod webu s přihlášením

5.2 Návrh použitých technologií

Podle požadavků bylo nutné zvolit takové technologie, které by fungovaly na různých operačních systémech, s co možná nejmenšími režijními pracemi.

5.2.1 Výběr programovacího jazyka

Microsoft .NET nebyl použit z důvodu limitované podpory pro OS Linux. I když .NET pro Linux již Microsoft zveřejnil a umožňuje v .NET programovat na Linuxu, stále se dle oficiální podpory jedná o preview verzi, která obsahuje určité limitace.[7]

Java funguje dobře na různých OS, avšak režie spojená s instalací tomcatu na serveru, jeho nastavením a následnou možnou customizací aplikace je veliká. Tyto náklady můžou případný další vývoj prodražit.

PHP nemá tak velké režijní náklady jako Java. Jde spustit na různých OS. Vývoj aplikace pro nahrávání kroků na webové stránce nebo možnosti určit, z jakého elementu se budou data získávat, není ideální. Při použití PHP by se musel využít další programovací jazyk, a to javascript.

Jako ideální se jeví použití javascriptu, který jde v dnešní době již spouštět i na serveru pomocí Node.js a je možné psát v tomto jazyce různá rozšíření pro aplikace. Node.js funguje stejně na serveru jako na klientovi. Režie spojené s instalací a následnou správou Node.js jsou minimální. Dále díky použití javascriptu je možné využít internetový prohlížeč Chrome jako aplikaci sloužící pro nahrávání kroků na stránce, výběr elementů a dalšího nastavení scrapování. Aby Chrome umožnil veškerá tato nastavení, je zapotřebí vytvořit rozšíření pro Chrome. Výhoda využití internetového prohlížeče spočívá ve velké komunitě lidí, kteří ho používají, a není nutné si instalovat novou aplikaci určenou pro scrapování webových stránek.

5.2.2 Javascript

Javascript je interpretovaný, objektově orientovaný programovací jazyk. Nejčastěji se používá pro webové aplikace, kde slouží zejména pro obsluhu událostí nebo pro načítání dat pomocí AJAX (Asynchronous JavaScript and XML).[4]

V dnešní době se stále více začíná využívat i pro programování desktopových, mobilních, ale i serverových aplikací. Příkladem desktopových aplikací může být podpora HTML5

aplikací ve Windows. V mobilním světě je dobrým příkladem Firefox OS, pro který se píší aplikace pomocí javascriptu, nebo React Native od Facebooku, pomocí kterého se píší aplikace v javascriptu pro různé mobilní platformy. Na serveru se javascript spouští pomocí Node.js. V dnešní době je spousta aplikací běžících na Node.js.

5.2.3 Node.js

Node.js je platforma postavená nad javascriptovým runtime V8 vyvinutým původně pro prohlížeč Chrome. Node.js využívá event-driven a non-blocking I/O model. Event-driven model je způsob zpracování programu, kdy existuje seznam eventů, které když nastanou, zavolají obsluhující funkci, tzv. callback. Non-blocking I/O model znamená, že node.js může I/O operace zpracovávat asynchronně. Node.js je vhodný pro realtime aplikace, standardní webové aplikace, ale také pro vývoj desktopových aplikací. Pro Node.js se píší aplikace v javascriptu. Node.js používá balíčkovací systém NPM, pomocí kterého lze jednoduše stáhnout potřebné knihovny pro naši aplikaci.[1]

5.2.4 PhantomJS

PhantomJS je headless prohlížeč bez grafického uživatelského prostředí. Je založen na jádru WebKit. Hlavní výhodou je poskytnutí API pro možnost vytvoření javascriptových skriptů, které umožňují pracovat s PhantomJS jako s normálním internetovým prohlížečem. PhantomJS je vhodný pro automatické testování webových aplikací před spuštěním na produkčním serveru, pro tvorbu screenshotů, získávání dat z webových stránek atd. Alternativou může být SlimerJS s jádrem Gecko.[2]

Díky poskytnutému API lze s PhantomJS pracovat na uživatelském počítači i na serveru. Toho je využito při scrapování stránek, kdy se s prohlížečem pracuje stále stejným způsobem, bez ohledu na to, kde je daný prohlížeč nainstalován. Tento prohlížeč je nejdůležitější součástí scrapovacího nástroje, bez něj by nebylo možné spustit proces scrapování. Tohoto prohlížeče využívá knihovna CasperJS. Díky tomuto headless prohlížeči lze spouštět proces scrapování jak na serveru, tak na desktopu, nezávisle na nainstalovaném internetovém prohlížeči.[2]

5.2.5 CasperJS

CasperJS je open-source navigační a testovací utilita pro PhantomJS nebo SlimerJS napsaná v javascriptu. Ulehčuje plnohodnotnou práci s headless prohlížečem, klikání na jednotlivé elementy, vyplňování formulářových polí, stahování různých příloh nebo umožňuje i web scraping. Tato utilita byla zvolena právě pro svou vlastnost scrapovat web.[8][3]

5.3 Použité open-source nástroje

Pro rychlejší vývoj byly použity již hotové open-source nástroje. Tyto nástroje byly dále upraveny pro potřeby navrhovaného web scrapingového nástroje.

5.3.1 Web Scraper

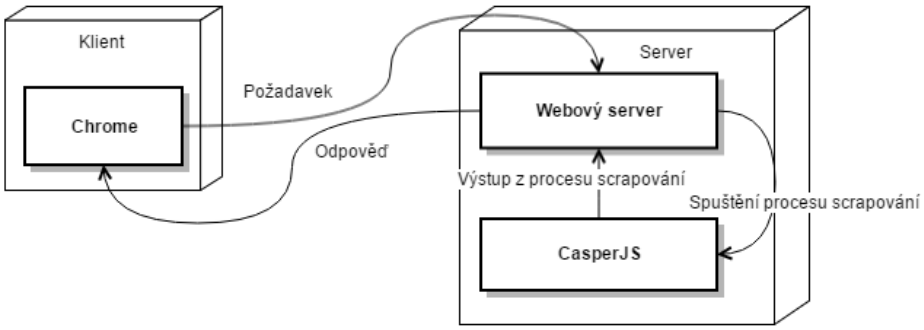
Více informací o tomto nástroji je uvedeno v části Analýza existujících řešení - Web Scraper. Do doplňku bude přidána funkčnost pro nahrávání kroků uživatele a odebrána funkčnost pro vlastní scrapování. Po úpravách bude sloužit pro získání nastavení scrapování.

5.3.2 Resurrectio

Resurrectio je doplněk pro prohlížeč Chrome, který je určen k nahrávání uživatelských kroků na webových stránkách. Tento doplněk následně vygeneruje kód pro CasperJS, který slouží pro testování UI. Funkcionalita pro nahrávání pohybu uživatele na webové stránce bude použita pro nastavení scrapování.[12]

5.4 Architektura aplikace

Aplikace využívá architektury klient-server, kde klient běží na lokální stanici uživatele a přes síť komunikuje se serverem. Tato architektura umožňuje spouštět scrapování na lokálním počítači, který bude zároveň klientem a serverem, a na serveru. Klient je tvořen doplňkem pro Chrome. Doplněk následně komunikuje se serverem, který spouští samotný proces scrapování webu. Celá architektura je znázorněna na Obrázek 8.



Obrázek 8 – Architektura aplikace klient-server

5.5 Vývoj rozšíření pro Chrome

Rozšíření pro Chrome umožňuje obohatit prohlížeč o další funkcionalitu, a ulehčit tak práci s prohlížečem. Pro vývoj rozšíření se využívá javascript. Každé rozšíření se skládá z následujících souborů:

- manifest,
- html soubory,
- javascriptové soubory,
- další soubory, které jsou použité v rozšíření.

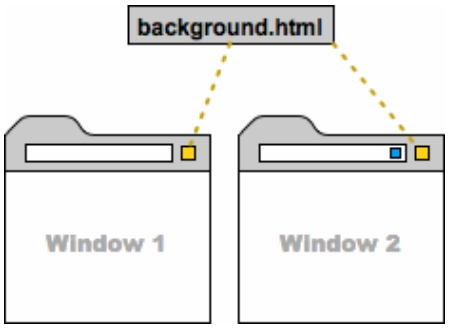
Všechny výše zmíněné soubory jsou zabaleny do speciálního zipu .crx. Jednotlivé soubory lze načíst přes relativní URL. Soubor manifest obsahuje informace o rozšíření, např. kdo je vyrobil, popis rozšíření, verzi, název, různá povolení atd. Pro vývoj rozšíření pro Chrome je dobré použít MVC javascriptový framework, který ulehčí následný vývoj. Chrome dále poskytuje vlastní API, které umožňuje využít funkčnost prohlížeče.[14]

5.5.1 Architektura

Rozšíření se skládá ze tří částí:

- Background page
- UI pages
- Content scripts

Background page umožňuje obsluhovat akci prohlížeče (Obrázek 9Obrázek 10 modrý čtvereček) nebo akci stránky (Obrázek 9Obrázek 10 žlutý čtvereček).[14]

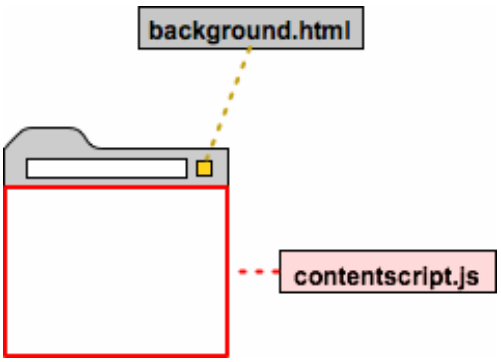


Obrázek 9 – Okno prohlížeče

Obě události obsahují stránku background.html a javascriptový soubor, který obslouží dané události. Existují dva typy background page: persistent background page nebo event page. Persistent background page jsou obvykle otevřeny po celou dobu, na rozdíl od event page, které se otevírají pouze pokud je potřeba, takže tím nezatěžují paměť a další systémové zdroje. Z tohoto důvodu je doporučeno používat (je-li to možné) event page místo persistent background page.[14]

UI pages slouží pro zobrazení UI rozšíření. Obvykle tyto stránky slouží k nastavení rozšíření. Dále umožňují přepsat stránky, které Chrome obsahuje (záložky, historii nebo nový tab). Nakonec lze využít vytvoření nové záložky nebo otevření okna k zobrazení dalších HTML stránek, které rozšíření obsahuje.[14]

Content script se spouští uvnitř webové stránky a umožňuje dále pracovat s touto stránkou. Content scripty jsou spouštěny po každém reloadu stránky. Rozdělení okna prohlížeče je znázorněno na Obrázek 10.[14]



Obrázek 10 – Rozdělení okna prohlížeče

Pro komunikaci mezi rozšířením a content skriptem se využívá mechanismu zasílání zpráv, kdy Chrome poskytuje metody pro zaslání zprávy a nastavení listeneru pro příjem zprávy.

Pro odeslání pouze jednoho požadavku z content skriptu do rozšíření slouží metoda *Chrome.runtime.sendMessage*, a naopak z rozšíření do content skriptu metoda *Chrome.tabs.sendMessage*. Pokud bychom chtěli udržovat spojení delší dobu, nejprve získáme port pomocí metody *Chrome.runtime.connection*, následně získaný port použijeme pro zaslání zprávy pomocí metody *postMessage*. Chrome dále poskytuje API pro komunikaci mezi jednotlivými rozšířeními, zasílání zpráv z webových stránek a pro zasílání zpráv nativním aplikacím.[14]

5.6 Vývoj skriptů pro CasperJS

Pro vývoj v CasperJS se používá javascript. Jedním ze základních prvků je zásobník, do kterého se vkládají navigační kroky. Navigační krok je javascriptová funkce, která může dělat dvě věci:

- čeká na provedení předchozího kroku,
- čeká na načtení požadované stránky.

Díky tomuto zásobníku je možné ve skriptu nadefinovat přesně po sobě jdoucí kroky akcí.

CasperJS pro vývoj poskytuje několik modulů. Casper modul obsahuje funkce pro nastavení CasperJS, vložení navigačního kroku do zásobníku, pohyb po webové stránce a možnost nastavení callbacků pro různé eventy. Modul ClientUtils vkládá do webové stránky podpůrné funkce pro práci s DOM. Modul Colorize umožňuje formátovat barevně výstup na konzoli. Modul Mouse simuluje práci s myší. Modul Tester poskytuje funkce pro testování webových aplikací. A konečně modul Utils poskytuje různé podpůrné funkce, jako například test, zda je proměnná pole nebo číslo, výpis proměnné na konzoli a mnoho dalších.[3][8]

6 Implementace

Implementace nástroje pro scrapování je logicky rozdělena na dvě části:

- implementace klienta,
- implementace serverové části.

Klient je naprogramován jako rozšíření pro internetový prohlížeč Chrome. Serverová část využívá Node.js, CasperJS a PhantomJS.

6.1 Implementace klientské části

Doplněk vychází ze dvou existujících rozšíření, která byla v rámci této diplomové práce spojena do jednoho a dále významně upravena a rozšířena. Díky již hotovému řešení se využijí postupy a prostředky, kterými jsou tato řešení naprogramována. Implementace využívá návrhového vzoru MVC a jednoduchého templatovacího systému. Pro rychlejší úpravu vzhledu byl použit css framework Bootstrap.

6.1.1 Práce se šablonami

Pro práci se šablonami byla použita knihovna ICanHaz.js. Knihovna umožňuje načítat do stránky různé šablony a ulehčuje práci s nimi.

V šabloně se vypisují hodnoty pomocí složených závorek.

```
{{title}}
```

Podmínka *if* a *for* cyklus mají totožný zápis. Šablona pozná, zda se jedná o pole, a pokud se detekuje pole, dojde k procházení jednotlivých prvků. V příkladu obsahuje *selector.multiple* hodnotu true nebo false a podle ní se vypíše text mezi ohraničením.

```
<input
    type="checkbox"
    name="multiple"
    {{#selector.multiple}}
        checked="checked"
    {{/selector.multiple}}> Multiple
```

Pokud je proměnná pole a obsahuje pouze primitivní datové typy, pak se hodnota vypisuje pomocí *{{.}}*.

```
<select multiple class="form-control" id="parentSelectors"
        name="parentSelectors">
    {{#selectorIds}}
        <option value="{{.}}">{{.}}</option>
    {{/selectorIds}}
</select>
```

Pokud se jedná o pole objektů, pak se k jednotlivým vlastnostem třídy přistupuje pomocí jejich jména.

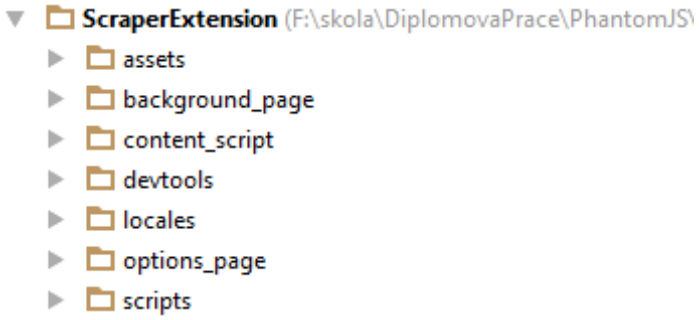
```
<select class="form-control" multiple id="filters" name="filters">
    {{#selectorFilters}}
        <option value="{{filter}}"
                data-i18n="{{filter}}">{{title}}</option>
    {{/selectorFilters}}
</select>
```

Vykreslení a předání dat do šablony probíhá tak, že se nejdříve zavolá funkce pro vykreslení, která má stejné jméno jako klíč šablony a jako parametr funkce přijímá objekt s daty pro vykreslení v šabloně. Funkce vrátí vykreslenou šablonu, která se následně pomocí jQuery vloží na dané místo. V následujícím zdrojovém kódu je ukázka vložení vykreslené šablony do stránky SitemapEditMetadata.html.

```
var sitemap = this.state.currentSitemap;
var $sitemapMetadataForm = ich.SitemapEditMetadata(sitemap);
```

6.1.2 Adresářová struktura

Zdrojové kódy jsou rozděleny do několika složek viz Obrázek 11. Složka *assets* obsahuje různé knihovny, použité v této aplikaci. Složka *background_page* obsahuje logiku pro ukládání nastavení a komunikaci s content skriptem. Ve složce *content_script* jsou obsažené soubory, které slouží pro nahrávání chování uživatele na webové stránce. Dále obsahuje logiku pro výběr elementu s požadovanými daty. Složka *devtools* obsahuje veškeré obrazovky. Jazykové překlady jsou umístěny ve složce *locales*. Složka *option_page* obsahuje formulář pro nastavení rozšíření. Business logika je obsažena v souborech adresáře *scripts*.



Obrázek 11 – Adresářová struktura rozšíření pro Chrome

6.1.3 Inicializace

Při inicializaci celé aplikace je třeba nejdříve zaregistrovat šablony. Registrace šablon do templatovacího systému je provedena ve Zdrojový kód 1.

```
var cbLoaded = function (templateId, template) {
    templatesLoaded++;
    ich.addTemplate(templateId, template);
    if (templatesLoaded === templateIds.length) {
        cbAllTemplatesLoaded();
    }
}
templateIds.forEach(function (templateId) {
    $.get(this.templateDir + templateId + '.html',
        cbLoaded.bind(this, templateId));
}.bind(this));
```

Zdrojový kód 1 – Inicializace šablon

Pole *teplateIds* obsahuje názvy souborů obsahujících danou šablonu pro vykreslení. Pomocí *forEach* se projdou postupně všechny položky v poli a pro každou se načte šablona a zaregistruje se. Registrace probíhá pomocí metody *addTemplate*, kde parametr *templateId* je identifikátor šablony a *template* je šablona.

Následně se zaregistrují události[3]. Po odchycení události dojde v závislosti na ní k zavolání příslušné funkce. Mapování události na příslušnou funkci je provedeno v metodě *control*. Tato metoda se volá při inicializaci aplikace. Vstupním parametrem je objekt obsahující mapování.

Struktura tohoto parametru je ukázána níže.

```
{
    '#sitemaps-nav-button': {
        click: this.showSitemaps
    },
...
}
```

[3] Událost je akce vyvolaná chováním uživatele na webové stránce nebo vyvolaná softwerově.

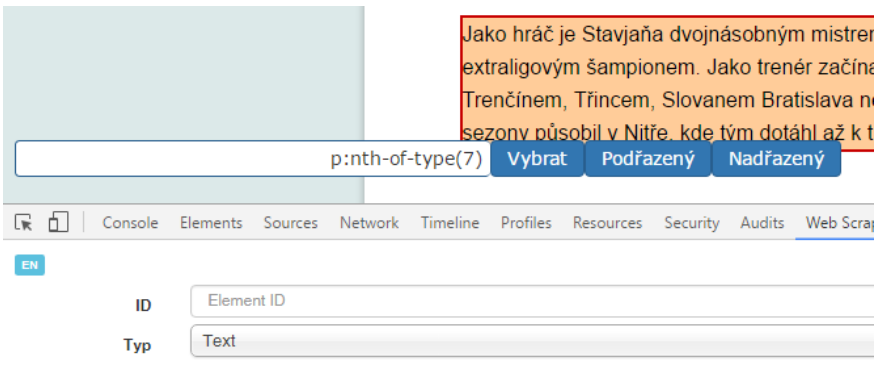
Z kódu je zřejmé, že název objektu obsahuje selektor příslušného elementu a jako hodnota je další objekt, kde název je událost a hodnota je ukazatel na danou funkci. Následně se pomocí dvou cyklů projdou všechny vlastnosti a pomocí jQuery se nastaví odchytávání jednotlivých událostí a jejich callback funkce. Kód pro průchod pomocí dvou cyklů je ukázán ve Zdrojový kód 2.

```
var controller = this;
for (var selector in controls) {
    for (var event in controls[selector]) {
        $(document).on(event, selector, (function (selector, event) {
            return function () {
                var continueBubbling =
                    controls[selector][event].call(controller, this);
                if (continueBubbling !== true) {
                    return false;
                }
            }
        })(selector, event));
    }
}
```

Zdrojový kód 2 – Nastvení callback funkcí

6.1.4 Výběr elementu

Výběr elementu probíhá tak, že v editaci elementu se klikne na tlačítko vybrat, to vyvolá událost click, kterou odchytí controller. V controlleru je vyvolána metoda *selectSelector*. Metoda připraví data a pomocí *Chrome.runtime.sendMessage* odešle zprávu s připravenými daty do background scriptu. Background skript pomocí nastaveného listeneru zachytí vysílanou zprávu a dále ji přepošle do aktivní záložky webového prohlížeče Chrome. Tato záložka zprávu zachytí v content skriptu a zaregistruje handlery pro událost click, mouseenter, mouseleave a keydown. Pro účel registrace handlerů slouží metody *bindKeyboardSelectionManipulation*, *bindKeyboardSelectionManipulation a bindElementSelection*.

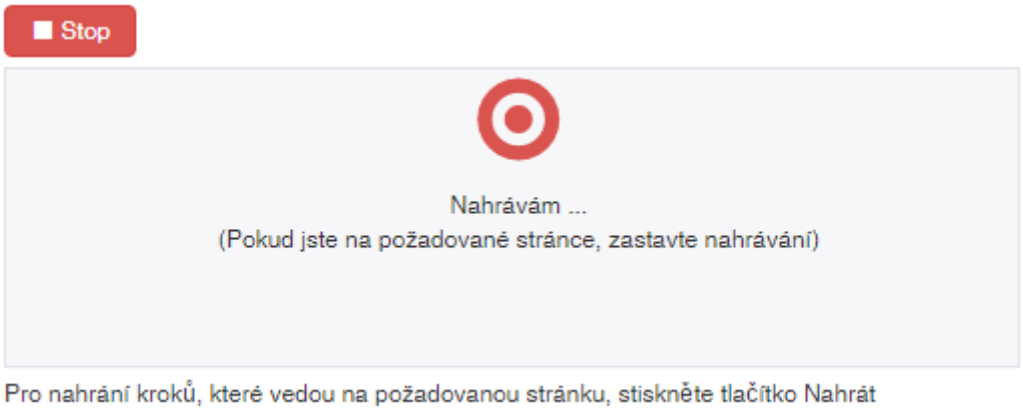


Obrázek 12 – Ukázka výběru elementu

Handler pro událost click přidá vybraný element do pole *selectedElements*, element ve stránce zvýrazní červeným rámečkem s červeným pozadím a pomocí třídy *CssSelector* zjistí selektor na daný element a vypíše ho do pole vedle tlačítek pro výběr. Handler pro událost mouseenter přidává na element, nad kterým je myš, třídu pro zvýraznění elementu zeleným rámečkem se zeleným pozadím a mouseleave tuto třídu naopak odebírá. Handler pro keydown umožňuje pohybovat se po rodičích nebo potomcích. U vybraného elementu lze dále ještě zkonkretizovat výběr pomocí výběru rodičovského elementu nebo potomka. Pokud je výběr v pořádku, klikne se na tlačítko vybrat. Vyvolaný event odebere handlery pro click, mouseenter, mouseleave a keydown. Následně získá pomocí třídy *CssSelector* selektor na daný element a poté odešle tento selektor přes background skript až do callback funkce implementované v controlleru při odesílání zprávy.

6.1.5 Nahrávání uživatelského chování

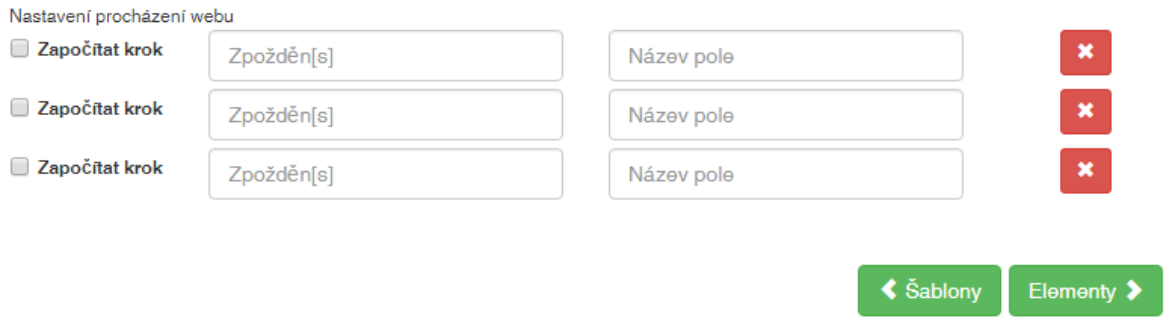
Po kliknutí na tlačítko Nahrát se vyvolá událost, kterou zachytí controller a následně zpracuje metoda *start*. Tato metoda vytvoří dlouho trvající připojení s background skriptem a následně pošle zprávu, aby se spustilo nahrávání. Background skript tuto zprávu přijme a přepošle ji do content skriptu, aktuálně otevřené záložky. Content skript tuto zprávu zpracuje a nad instancí třídy *Recorder* se zavolá metoda *start*. Tato metoda nastaví odchytávání veškerých událostí metodami této třídy.



Obrázek 13 – Indikace nahrávání

Po odchycení události dojde k vytvoření instance třídy *Event*. Třída *Event* poskytuje rozhraní pro získání informací o události. Poté je dle druhu události zavolána metoda *peek*. Metoda *peek* vrátí poslední vložený element. *Peek* je volána pouze u událostí onpageload a onkeypress. Po vrácení elementu metodou *peek* se porovná poslední vložený element

s aktuálně vkládaným elementem. Pokud jsou porovnávané elementy shodné, volá se metoda *poke* jinak *append*. Metoda *poke* přepíše naposled vkládaný element aktuálně vkládaným elementem. Metoda *append* přidá aktuálně vkládaný element na konec pole. Metody *append* a *poke* nakonec zašlou zprávu zpět do background skriptu s parametry informací o vložení elementu a samotný vkládaný element.



Obrázek 14 – Nastavení formulářových prvků

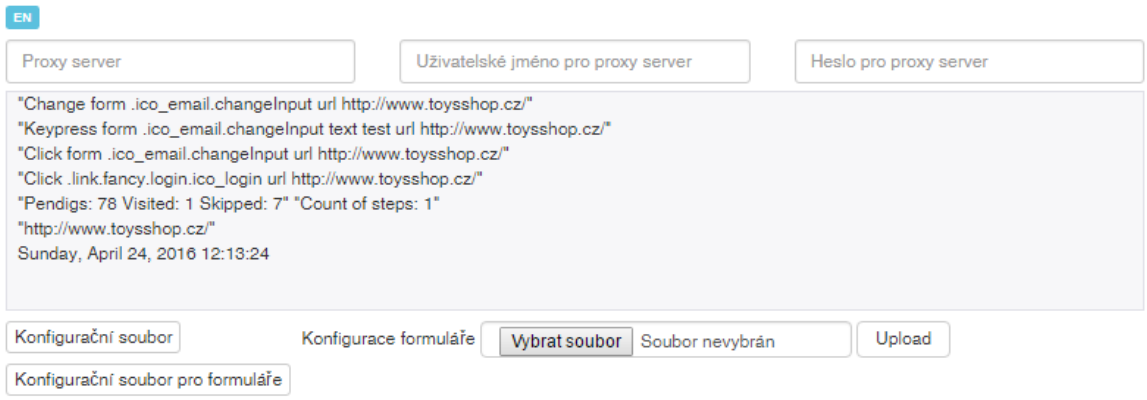
Background skript přijme zprávu z content skriptu o vložení nového elementu a vloží tento element do pole *testcase_items*. Podle toho, zda se jedná o zprávu z *append* nebo *poke*, prvek buď přidá na konec pole, nebo přepíše posledně přidaný prvek. Pokud background skript zjistí, že se jedná o zprávu z *append*, pak zašle zprávu do controlleru. Nastavený listener tuto zprávu příjme a následně vyhodnotí, jestli se jedná o textový input nebo kliknutí. Pokud vyhodnotí, že se jedná o jednu z těchto možností, přidá do UI pole k nastavení jména, zpoždění a započtení do limitu kroků. Jméno lze dále použít pro nastavení různých kombinací formuláře. Zpožděním se nastaví doba, po kterou má scraper čekat po kliknutí na element. Započítat krok je nastavení, které připočte tento krok k celkovému limitu kroků.

Po kliknutí na tlačítko stop se vyvolá událost, kterou obslouží metoda *stop*. Metoda zašle zprávu do background skriptu. Odtud se pošle zpráva do content skriptu a zde dojde k uvolnění handlerů pro události a vypnutí nahrávání. Následně se v background skriptu do sitemapy připojí nahrané kroky, tato sitemapa se odešle zpět do controlleru a tam se dále zpracuje. K uložení nahrávání a nastavení formulářových prvků a doby zpoždění dojde při přechodu na další stránku.

6.1.6 Spuštění scrapování

Interakce mezi klientem a serverem je vedena pomocí websocketu a umoňuje zobrazování screenshotů, zachycených při scrapování uživateli. Při každé komunikaci musí být na

server zasláno uživatelské jméno i heslo pro ověření totožnosti. Obrázek 15 znázorňuje uživatelské rozhraní pro komunikaci s uživatelem. Po spuštění je na server poslán request na inicializaci a ověření přístupu uživatele. Pokud server vrátí návratový kód 200, je vše v pořádku a klient může navázat spojení s websocket serverem pro získání informací o průběhu scrapování. Pokud se nepovedlo přihlásit, server vrátí návratový kód 403 a uživateli je vypsáno chybové hlášení.



Obrázek 15 – UI pro výpis textového stavu scrapování

O průběhu scrapování je uživatel informován pomocí screenshotů, pořízených během scrapování. Tyto screenshoty ulehčí orientaci, v jaké fázi se scrapování nachází. Pro získání screenshotu je v pravidelných intervalech pomocí javascriptu zasílán request na získání aktuálního screenshotu.

6.1.7 Lokalizace

Pro větší komfort uživatelů během nastavování nástroje byly přidány jazykové mutace - konkrétně české a anglické překlady. O překlady se stará javascriptová knihovna i18next. Tato knihovna využívá data atributu na elementu pro získání identifikátoru překladu.

```
<label for="selector" class="col-lg-1 control-label" data-i18n="element">
    Element
</label>
```

Do data atributu se zapíše klíč k překladu a tentýž klíč se přidá i do souboru s překlady. Soubory s překlady jsou obyčejné soubory ve formátu JSON. Díky různým zápisům hodnot v data atributu lze cílit překlad např. do placeholderu u inputu, titulku u odkazu

a mnoha dalších nastavení. Více o nastaveních a možnostech i18next naleznete na oficiálním webu[4].

6.2 Implementace serverové části

Implementace serverové části je dále rozdělena na implementaci webového a websocket serveru a na implementaci řešení pro scrapování.

6.2.1 Webový a websocket server

Pro spuštění aplikace je třeba mít spuštěný webový server. O spuštění webového serveru se stará soubor server.js. Tento soubor se spouští pomocí Node.js a má za úkol zprostředkovávat komunikaci mezi klientem a serverem. Soubor se spouští pomocí následujícího příkazu.

```
node cesta/k/souboru/server.js
```

Spuštěním tohoto souboru se spustí webový server a připraví se i websocket server. Pomocí parametru *server-port* jde nastavit port, na kterém webový server poběží. Defaultně server běží na portu 16002. Pro uživatele Windows je připraven soubor scraper.bat, pomocí něhož se zavolá příkaz pro spuštění serveru s defaultním portem, a tak si uživatel nemusí pamatovat tento příkaz. U uživatelů Linuxu je počítáno s většími znalostmi používání PC, a proto není potřeba vytvářet spouštěcí skript i na operační systém Linux.

Protože je požadavek na možnost spouštění aplikace na serveru, bylo zapotřebí vytvořit mechanismus pro autentizaci uživatele. Autentizace probíhá tak, že při každém požadavku se na server posílají spolu s daty i uživatelské jméno a heslo. Heslo je zahashované pomocí md5. Po přijetí požadavku webovým serverem a websocket serverem je spuštěno ověření uživatele. Ověření probíhá proti seznamu uživatelů v souboru auth.csv. Formát CSV byl zvolen pro snadnou úpravu a přidání nových uživatelů. První sloupec souboru obsahuje uživatelské jméno, druhý sloupec zahashované heslo.

[4] Oficiální dokumentace pro knihovnu i18next http://i18next.com

Server přijímá pouze tyto požadavky:

- /status – zjištění, zda jsou přihlašovací údaje správné,
- /init – slouží k nastavení nástroje,
- /upload-config-form – nahraje soubor pro konfiguraci formulářů,
- /download – slouží ke stažení výsledků,
- /screenshot – slouží k získání screenshotu pořízeného během scrapování.

Pokud se bude někdo snažit přistoupit na jinou url, server vrátí odpověď 404, pokud je uživatel nepřihlášený, vrací server 403. Pokud dojde k chybě, vrátí server 500.

Po úspěšném přihlášení může uživatel spustit scrapování pomocí websocket serveru. Po příchozí zprávě pomocí websocketu z klientské části aplikace se spustí proces scrapování. Výstup z tohoto procesu je odchytáván a posílán zpět na klientskou stanici.

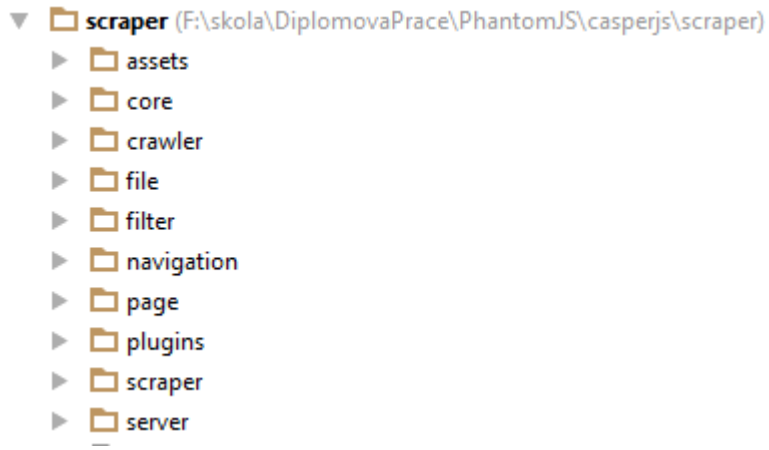
6.2.2 Implementace scrapování

Část pro scrapování využívá API pro CasperJS a PhantomJS. Vedle těchto dvou nástrojů se v serverové části využívá databáze seznamu států a různé moduly pro Node.js.

6.2.2.1 Adresářová struktura

Jednotlivé konfigurace pro procházené webové stránky jsou uloženy ve složce *plugins*, v níž jsou následně složky s různými konfiguracemi. Každá tato složka obsahuje konfigurační soubor pro vyplňování formulářových prvků, konfiguraci scrapování, poslední výsledky scrapování a helper javascriptový soubor s metodami umožňujícími ovlivňovat běh scrapování.

Složka *core* zahrnuje základní třídy pro inicializaci aplikace. Složka *crawler* obsahuje funkčnost pro crawling webu. Složka *filter* slouží pro třídy, které umožňují filtrovat a dále zpracovávat získaná data scrapováním. Složka *file* obsahuje třídy pro práci se soubory. Složka *assets* obsahuje další knihovny nebo podpůrné soubory. Složka *navigation* obsahuje třídy sloužící pro navigaci po webových stránkách. Ve složce *page* se nachází třídy sloužící pro získání dat z webové stránky. Ve složce *scraper* je inicializace a přenos dat z webové stránky do aplikace. Složka *server* obsahuje webový a websocket server pro komunikaci s klientem.



Obrázek 16 – Adresářová struktura serverové části

6.2.2.2 Inicializace aplikace

CasperJS umožňuje spustit skript tak, aby pro přístup na webové stránky používal proxy server, tedy nepřicházel na web z vlastní IP adresy, a mohl tak po zablokování IP adresy proxy serveru tento server zaměnit za jiný a vyhnout se blokaci. Pro nastavení proxy serveru se využívá parametr *--proxy=ip_adresa:port*. Pokud je proxy server zabezpečený, přidá se i druhý parametr *--proxy-auth=uživatelské_jméno:heslo*.

Výchozím spouštěcím souborem je *scraper.js* v kořenovém adresáři aplikace. V tomto souboru se připraví konstanty cest a pomocí třídy *Core* se načtou a inicializují veškeré části aplikace. Pro inicializaci a načtení všech částí aplikace slouží metody *loadConfiguration*, *loadFormValues*, *initCasper* a *initLibraries*.

Nejdůležitější metodou inicializace aplikace je *initCasper*. Ostatní metody pouze načítají javascriptové soubory nebo vytvářejí potřebné instance. Metoda *initCasper* nastavuje callback funkce, pokud je vyvolána událost *page.error* nebo *error.message*. Více informací o těchto eventech lze získat v oficiální dokumentaci[5]. Obě tyto callback funkce umožňují pouze to, že vytvoří zprávu o eventu a vloží ji k ostatním chybovým hlášením.

Po načtení všech knihoven a jejich nakonfigurování se spustí metoda *run*. Tato metoda spustí celý proces scrapování. Po dokončení scrapování se v tomto souboru zavolají metody pro uložení získaných dat. Nakonec se proces ukončí.

[5] Dokumentace k eventám nástroje CasperJS http://casperjs.readthedocs.org/en/latest/events-filters.html

6.2.2.3 Crawling webu

O celý průběh průchodu webem se stará metoda *run* třídy *Crawler*, která je zobrazena ve Zdrojový kód 3. Tato metoda je volána i v případě, kdy je průchod celým webem vypnutý. V takovémto případě se volá pouze jednou.

```
Crawler.prototype.run = function (url) {
    this.visitedUrls.push(url);
    var crawler = this;
    casper.then(function () {
        casper.open(url).then(function () {
            utils.dump(url);
            this.emit('create.screenshot');
            crawler.actualCountOfSteps++;

            crawler.navigate();
            this.emit('create.screenshot');

            var link = crawler.getLink(url);
            var baseUrl = this.getGlobal('location').origin;
            var localLinks = crawler.getLinks();

            crawler.dataObj.links.push(link);
            for (key in localLinks) {
                crawler.prepareLink(localLinks[key], baseUrl);
            }

            if (crawler.pendingUrls.length > 0 &&
                (!crawler.countOfStepsLimit ||
                crawler.actualCountOfSteps< crawler.countOfStepsLimit) &&
                crawler.core.configuration.crawlWeb) {
                    crawler.showInfo(crawler);
                    var nextUrl = crawler.pendingUrls.shift();
                    crawler.dataObj.linkCount++;
                    crawler.run(nextUrl);
            } else {
                crawler.showInfo(crawler);            }
        });
    });
```

Zdrojový kód 3 – Průběh scrapování v metodě run

Parametrem metody je URL stránky pro navštívení. Tato URL se přidá do pole již navštívených stránek a otevře se v headless browseru. Po načtení URL je zavolána funkce *navigate*, která provede nahrané kroky uživatele a z cílové stránky získá požadovaná data. Následně se vezme aktuální link, získá se základní URL a z aktuální stránky se načtou všechna URL. Poté se pomocí for cyklu projdou všechna URL a pomocí metody *prepareLink* se zjistí, jestli URL je již navštívená nebo čeká na navštívení. Pokud čeká na navštívení, musí se zjistit, zda již není v poli mezi čekajícími. Pokud tam není, přidá se, jinak se pokračuje na další URL.

Pokud není pole čekajících URL na navštívení prázdné a počet kroků nepřesáhl povolený limit a zároveň je nastaveno procházení celého webu, pak se vezme první čekající URL

a zavolá se znovu metoda *run*, s touto URL. Pokud je jedna z podmínek vyhodnocena negativně, pak procházení končí.

6.2.2.4 Průchod pomocí nahraných kroků

O průchod pomocí nahraných kroků se stará třída *Navigate* a její metoda *navigate* viz Zdrojový kód 4. Pokud je nastaveno procházení celého webu, pak se tato metoda spustí pouze jednou, při prvním zavolání. Nejdříve se vezme aktuální krok a vytvoří se instance třídy *NavigationElement*, parametrem konstruktoru bude aktuální krok z konfigurace. Třída *NavigationElement* poskytuje metody pro snazší práci s konfigurací. Následně je pomocí javascriptové funkce *eval* zavolána metoda, jejíž název získáme zavoláním metody *getMethodName* nad instancí třídy *NavigationElement*. Nakonec se ukazatel přesune na následující krok.

```
Navigation.prototype.navigate = function (crawler) {
    var navigationElement =
      new NavigationElement(this.core.configuration.recorder[this.step]);
    if (navigationElement.getMethodName() &&
        typeof this[navigationElement.getMethodName()] == 'function') {
        eval('this.' + navigationElement.getMethodName())
            (navigationElement, this.core.configuration,
             this.core.formValues[this.currentFormValues], this.step);
    }

    var delay = this.getDelay(this.core.configuration.delay, this.step);
    if (delay) {
        utils.dump('Waiting for ' + delay + 's');
        casper.then(function () {
            casper.wait(delay * 1000);
        });
    }

    if (this.isStep(this.core.configuration.countStep, this.step)) {
        crawler.actualCountOfSteps++;
    }

    this.step++;
}
```

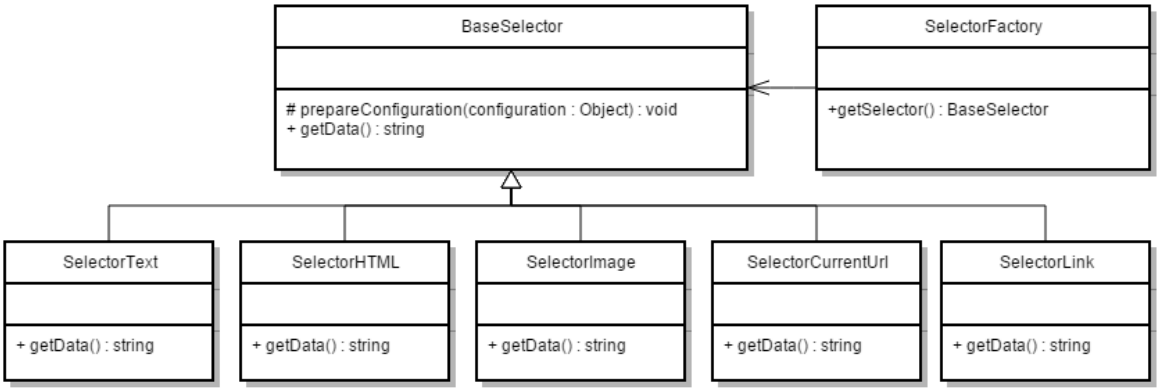
Zdrojový kód 4 – Průchod pomocí nahraných kroků

V některých případech se data donačítají pomocí ajaxu a jejich načtení může trvat déle, než je povolená doba pro čekání na konkrétní element. Z tohoto důvodu je v nastavení zpoždění. Zpoždění se spustí vždy po příslušné akci, ke které bylo nastaveno.

6.2.2.5 Scraping a filtrace dat

O scraping se stará třída *Scraper* a její metoda *getData*. Pomocí for cyklu se postupně začnou procházet jednotlivé selektory elementů, ze kterých se mají získat data. Pro každý

selektor se volá metoda *evaluate*. *Evaluate* na webové stránce nainicializuje *SelectorFactory*. Pomocí této *SelectorFactory* se získá instance konkrétní třídy pro získání dat. Následně se pomocí této instance získají konkrétní data. Pro získání instance konkrétní třídy selektoru je použit návrhový vzor Abstract Factory.



Obrázek 17 – Class diagram elementů

Pro lepší porozumění, k čemu slouží třídy získané pomocí *SelectorFactory,* slouží následující ukázka. V ukázce je příklad HTML dokumentu a použití jednotlivých tříd s vysvětlenou funkčností. Předpokládá se, že v každé třídě je použit selektor, který odpovídá výběru správného elementu na cílové stránce. Získaná data budou zobrazena zvýrazněním.

Níže je uveden HTML dokument, na který budou použity jednotlivé třídy.

```
<div>
    <a href="url/adresa.html"><img src="cesta/k/obrazku.jpg" />
        Text odkazu
    </a>
</div>
```

SelectorImage získá z webové stránky URL adresu obrázku.

```
<div>
    <a href="url/adresa.html"><img src="cesta/k/obrazku.jpg" />
        Text odkazu
    </a>
</div>
```

SelectorLink získá z webové stránky URL adresu v odkazu.

```
<div>
    <a href="url/adresa.html"><img src="cesta/k/obrazku.jpg" />
        Text odkazu
    </a>
</div>
```

SelectorText z HTML dokumentu získá pouze prostý text, bez všech HTML entit.

```
<div>
    <a href="url/adresa.html"><img src="cesta/k/obrazku.jpg" />
        Text odkazu
    </a>
</div>
```

SelectorHtml získá celý HTML dokument.

```
<div>
    <a href="url/adresa.html"><img src="cesta/k/obrazku.jpg" />
        Text odkazu
    </a>
</div>
```

SelectorCurrentUrl se liší od ostatních tříd tím, že nezískává data z HTML dokumentu, ale přímo si načte URL adresu dané stránky a tu vrátí jako výsledek. Nacházíli se nástroj na stránce www.seznam.cz, selektor vrátí www.seznam.cz.

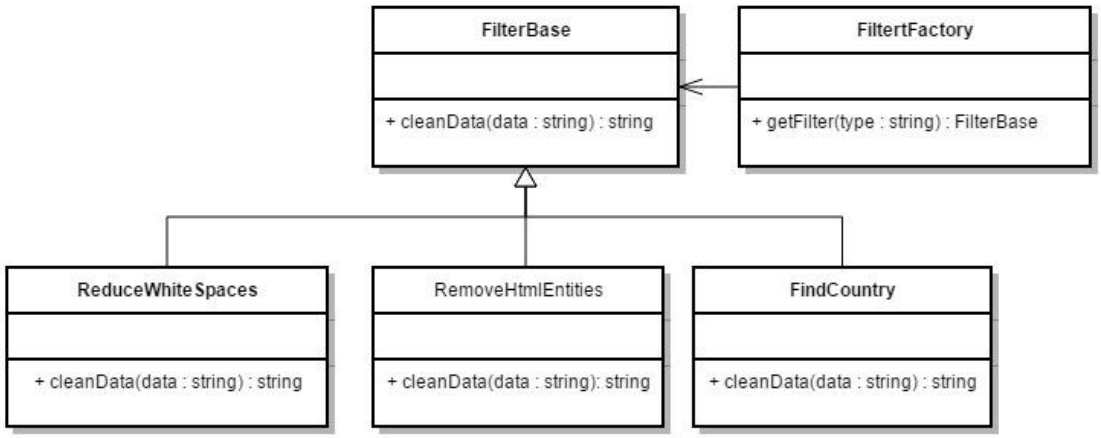
Dále se zkontroluje, zda je element povinný. Pokud je element povinný a scraper nevrátil žádná data, pak se scrapování ukončí a pokračuje se dál v celém procesu. Pokud výše zmíněná podmínka neplatí, pokračuje se filtrací dat a vložením dat mezi ostatní data získaná nástrojem. Ukázka kódu této funkčnosti je ve Zdrojový kód 5.

```
Scraper.prototype.getData = function() {
    var data = [];
    for (var key in this.configuration.selectors) {
        var selector = this.configuration.selectors[key];
        var result = this.evaluate(selector);
        if (selector.required && !result) {
            return null;
        }
        var scrapedData = {'name' : selector.id,
            'data' : this.filter.cleanData(result, selector)};
        data.push(scrapedData);
    }
    return data;
}

Scraper.prototype.evaluate = function(selector) {
    return casper.evaluate(function(selector) {
        var factory = new SelectorFactory();
        var specificSelector = factory.getSelector(selector);
        var page = new Page();
        return page.getData(pageElement);
    }, selector);
}
```

Zdrojový kód 5 – Získání dat ze stránky

Pro filtry byl využit návrhový vzor Abstract Factory, který vytváří jednotlivé třídy. Pro získání instance třídy filtru se nad objektem *FilterFactory* zavolá metoda *getFilter*, která jako parametr přebírá z nastavení druh filtru. Po získání instance filtru se zavolá metoda

cleanData, které se předají scrapovaná data a ta se vyčistí. Ve filtrech je důležité počítat s tím, že nemusí přijít pouze primitivní datový typ, ale i serializované pole v JSON. Celý návrh je znázorněn na Obrázek 18.



Obrázek 18 – Class diagram filtrů

Pro snazší pochopení, jaká data jsou odstraněna jednotlivými filtry, slouží následující ukázka. Scraper získal tento kus html dokumentu:

```
<div><h1>Nadpis s     hodně mezerami slovensko</h1></div>
```

Aplikováním filtru ReduceWhiteSpaces se mezery jdoucí za sebou sloučí do jedné.

```
<div><h1>Nadpis s hodně mezerami</h1></div>
```

Filtr RemoveHtmlEntities ze získaného řetězce odebere všechny HTML entity.

```
Nadpis s     hodně mezerami
```

Filtr FindCountry se pokusí najít v řetězci buďto název státu nebo kód státu a tento výsledek vrátí.

```
slovensko
```

Získaná data se na konci celého procesu scrapování ukládají do dvou základních souborů *results.json* a *results.csv*. Další možné formáty lze vytvořit pomocí helper metody daného pluginu. Tato metoda je volána před uložením dat do dvou základních souborů.

6.3 Přidání nových filtrací

Přidání nových filtrů je velmi jednoduché. Stačí přidat v doplňku pro Chrome, v souboru *controller.js* do objektu *selectorFilters*, název třídy, která se bude volat během scrapování, a název pro UI.

```
selectorFilters: [
    {
        filter: 'reduceWhiteSpaces',
        title: 'Reduce white spaces'
    } ,
    {
        filter: 'RemoveHTMLEntities',
        title: 'Remove HTML entities'
    } ,
    {
        filter: 'Country',
        title: 'Select country'
    },
]
```

Následně se vytvoří nová třída ve složce *filter*. Tato třída dědí *BaseFilter*. Název souboru musí být stejný jako název třídy. U názvu souboru jsou všechna písmena malá. Tato třída musí implementovat metodu *cleanData*. Metoda se stará o filtraci dat. Musí počítat i s tím, že může příjmout pole hodnot, které jsou serializované do JSONu. V takovémto případě musí JSON dekódovat a následně projít a profiltrovat jednotlivé položky pole. Ukázka implementace třídy pro odstranění mezer je ve Zdrojový kód 6.

```
function ReduceWhiteSpace() {}

ReduceWhiteSpace.prototype = new BaseFilter();

ReduceWhiteSpace.prototype.cleanData = function(data) {
    if (!data) {
        return data;
    }

    if(this.helper.isJSON(data)){
        var jsonObject = JSON.parse(data);
        for(var i = 0; i < jsonObject.length; i++){
            jsonObject[i] = jsonObject[i].trim();
            jsonObject[i] = jsonObject[i].replace(/\s+/g, ' ');
        }

        return JSON.stringify(jsonObject);
    }

    data.trim();

    return data.replace(/\s+/g, ' ');
}
```

Zdrojový kód 6 – Filtr ReduceWhiteSpace pro odstranění přebytečných mezer

6.4 Přidání nových elementů

Podobně jako filtry jsou naprogramovány třídy pro získávání dat ze stránky. Pro přidání nového typu elementu stačí přidat v doplňku pro Chrome, v souboru controller.js do objektu *selectorTypes* název třídy, která se bude volat během scrapování, a název pro UI.

```
selectorTypes: [
    {
        type: 'SelectorText',
        title: 'Text'
    } ,
    {
        type: 'SelectorLink',
        title: 'Url'
    },
    {
        type: 'SelectorCurrentUrl',
        title: 'Current URL'
    },
    {
        type: 'SelectorImage',
        title: 'Image'
    },
    {
        type: 'SelectorHTML',
        title: 'HTML'
    }
],
```

Poté je třeba vytvořit třídu pro získání nastavení v doplňku pro Chrome. Ve složce *scripts* a v ní ve složce *Selector* se vytvoří nový soubor, který je pojmenován stejně jako třída, která se bude vytvářet. Třída bude obsahovat stejné metody, jako znázorněná třída ve Zdrojový kód 3. Třída obsahuje pouze metodu *getData*. Metoda *getData* slouží pro získání dat ze stránky. Tato data jsou následně použita k náhledu na seznamu elementů nebo v detailu elementu. Níže je ukázka kódu pro třídu v doplňku Chrome získavající aktuální url (Zdrojový kód 7).

```
var SelectorCurrentUrl = {
    getData: function (parentElement) {
        var result = [];
        var data = {};

        data[this.id] = window.location.href;

        result.push(data);

        return result;
    }
};
```

Zdrojový kód 7 – Třída v doplňku pro Chrome sloužící k získání selectoru na element

Nakonec se vytvoří nová třída ve složce *page* v serverové části scrapovacího nástroje, ukázkový kód je ve Zdrojový kód 8. Tato třída dědí *BaseSelector*. Název souboru musí být stejný jako název třídy. U názvu souboru jsou všechna písmena malá. Tato třída musí implementovat rovněž metodu *getData*. Metoda získá aktuální URL ze scrapované stránky.

```
function SelectorCurrentUrl(data) {
    this.prepareData(data);
}

SelectorCurrentUrl.prototype = new BaseSelector();

SelectorCurrentUrl.prototype.getData = function() {
    var text = document.querySelector(this.selector).innerText;

    if(text == ''){
        return;
    }

    return text;
}
```

Zdrojový kód 8 – Třída pro získání dat ze stránky

6.5 Možnost úpravy konfigurace a formulářových dat za běhu

Scraper umožňuje změnit konfiguraci a pročistit data pomocí helperu pro daný plugin. Tento helper obsahuje metody, které se volají před načtením úvodní stránky, po načtení konfigurace pro formulář, před filtrací získaných dat a před uložením dat.

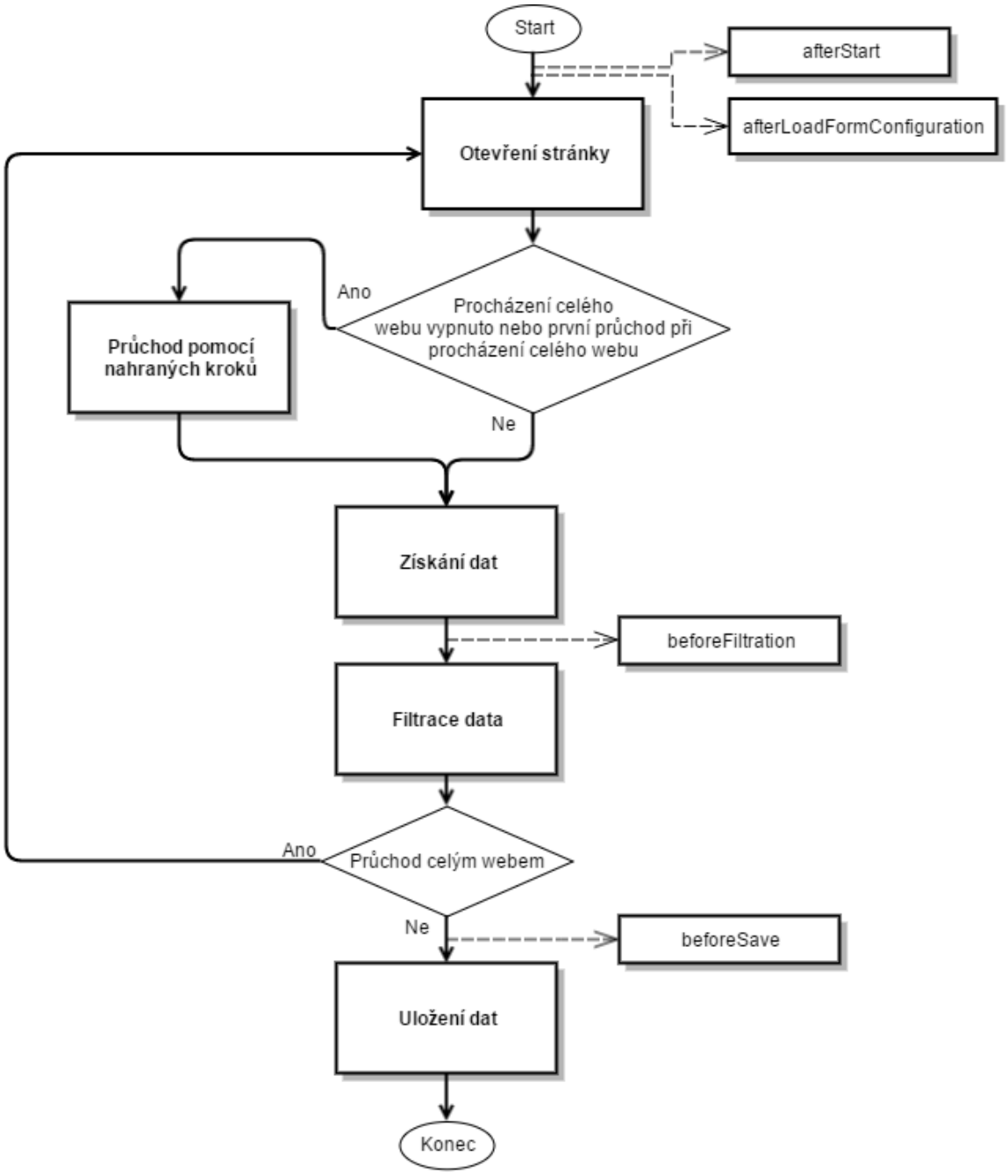
Při volání hned po startu se jako parametr metody *afterStart* použije samotná konfigurace. Po načtení konfigurace formuláře se volá metoda *afterLoadFormConfiguration* s parametrem obsahujícím konfiguraci formuláře. Před uložením bude metoda *beforeSave* volána s parametrem, který obsahuje vlastní výsledky scrapování. Nakonec před filtrací je jako parametr metody *beforeFiltration* string se získanými daty. Každá metoda musí vracet data, která přijala jako parametr, ale již budou upravená.

Helper nemusí obsahovat všechny metody, ale jen ty, které jsou zapotřebí pro daný plugin. Ve Zdrojový kód 9 je znázorněn helper pro web Justice.cz, který generuje identifikační číslo osoby, známé pod zkratkou IČ. Seznam vygenerovaných IČ bude použit při vyplňování formuláře pro nalezení konkrétní firmy. Během testování docházelo ke generování velkého množství IČ, proto je doporučeno tento skript rozdělit a spouštět scrapování paralelně jen pro generování určitého množství dat.

```
function PluginHelper() {
}

PluginHelper.prototype.afterLoadFormConfiguration =
    function (formConfiguration) {
    for (var ic = 0; ic < 99999994; ic++) {
        ic = ic.toString();

        for (var length = ic.length; length <= 8; length++) {
            ic = '0' + ic;
        }

        if (this.isIC(ic)) {
            formConfiguration.push({'IC': ic});
        }
    }
    return formConfiguration;
}

PluginHelper.prototype.isIC = function (x) {
    try {
        var a = 0;

        if (x.length != 8) {
            throw 1;
        }

        var b = x.split('');
        var c = 0;

        for (var i = 0; i < 7; i++) {
            a += (parseInt(b[i]) * (8 - i));
        }

        a = a % 11;
        c = 11 - a;

        if (a == 1) {
            c = 0;
        }

        if (a == 0) {
            c = 1;
        }

        if (a == 10) {
            c = 1;
        }
        if (parseInt(b[7]) != c) {
            throw(1);
        }
    }
    catch (e) {
        return false;
    }
    return true;
}

module.exports = PluginHelper;
```

Zdrojový kód 9 – Helper pro web Justice.cz

Na Obrázek 19 je znázorněno, kdy se jednotlivé metody volají. Díky těmto metodám lze ovlivnit nastavení, pročistit data nebo uložit data do jiného formátu.



Obrázek 19 – Diagram volání eventů

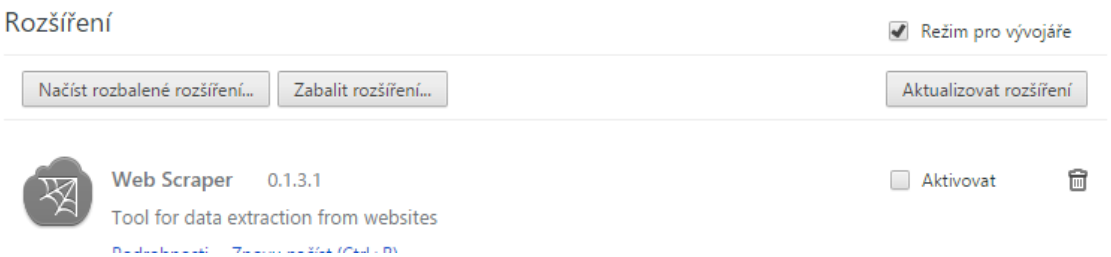
7 Instalace

Pro ulehčení instalace aplikace byl připraven instalační balíček. Nejprve je nutné spustit instalační soubor pro instalaci Node.js. Poté je třeba spustit instalační soubor WebScraper. Po spuštění obou instalačních soubrů je možné aplikaci pouze spustit a začít používat. Instalační soubor pro WebScraper byl vytvořen pomocí nástroje pro tvorbu balíků Wix Toolset. Wix Toolset je poskytován zdarma. Nástroj umožňuje vytvořit instalační soubor pro Windows.

Konfigurace nástroje probíhá pomocí XML dokumentu, kde pomocí elementů a jejich atributů se vydefinují jednotlivé soubory, které má instalátor nainstalovat, podoba instalátoru a jednotlivé kroky instalátoru. Podrobná dokumentace k tomuto nástroji se nachází na oficiálních stránkách[6]. Konfigurační XML soubor je přiložen na CD.

Po instalaci je potřeba ještě nainstalovat doplňek pro nastavení scrapování do Chromu. V nabídce se vyberou *Další nástroje* a následně *Rozšíření*. Na obrazovce s rozšířením se zaškrtne políčko *Režim pro vývojáře*. Zaškrtnutím políčka se rozbalí další nabídka, která umožní nainstalovat doplňek.

Následně se klikne na *Načíst rozbalené rozšíření* a vybere se složka, v níž je toto rozšíření nainstalované. Tato složka se nazývá ScraperExtension a je umístěna ve složce, kam byla aplikace nainstalována. Vybráním a potvrzením výběru se rozšíření nainstaluje. Aktivace rozšíření se provede zaškrtnutím políčka *Aktivovat*, které je u daného rozšíření.



Obrázek 20 – Obrazovka s rozšířením pro prohlížeč Chrome

Pro uživatele operačního systému Linux, OS X a dalších nebyl vytvořen instalační balíček, protože se počítá s tím, že tito uživatelé mají hlubší znalosti používání PC.

[6] Oficiální dokumentace nástroje Wix Toolset http://wixtoolset.org/documentation/

Nejdříve je nutné na operační systém nainstalovat tyto programy:

- PhantomJS[7],
- CasperJS[8],
- Node.js[9].

Podrobné návody a různé způsoby instalace těchto souborů jsou k dispozici na oficiálních stránkách těchto nástrojů.

Instalace nástroje se provede rozbalením přiloženého zip souboru do složky na disku a následným spuštěním příkazu *npm install*. Tento příkaz je třeba spouštět ze složky *složka_s_nastrojem/WebScraper/casperjs/scraper*. Instalace doplňku do Chromu je stejná jako v případě operačního systému Windows.

[7] Oficiální stránky pro PhantomJS http://phantomjs.org/
[8] Oficiální stránky pro CasperJS http://casperjs.org/
[9] Oficiální stránky pro Node.js https://nodejs.org/en/

8 Nastavení nástroje

Pro scrapování je potřeba mít spuštěnou serverovou část aplikace. Spuštění se provede kliknutím na ikonku nebo v adresáři, kam byl nástroj nainstalován, kliknutím na soubor scraper.bat. Tento postup spuštění je platný pro operační systém Windows. Pro operační systém Linux a další je potřeba spustit tento příkaz v příkazové řadce.

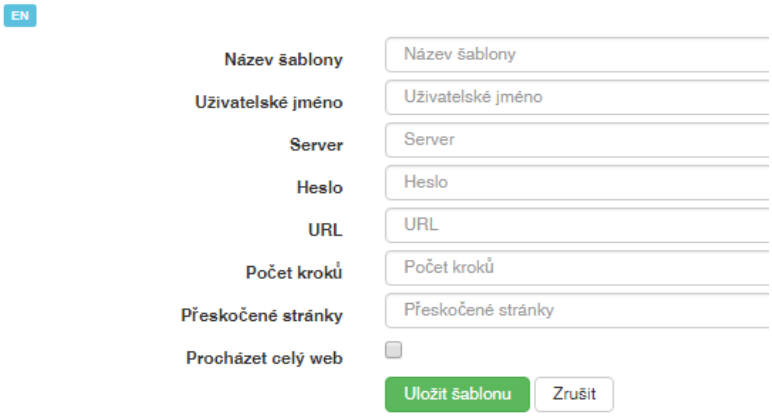
node cesta/do/adresare/aplikace/slozky/server/server.js

Nastavování scrapovacího nástroje lze provádět v českém nebo anglickém jazyce. Jazyk se přepíná kliknutím na kód jazyka v levém horním rohu. U nastavení Justice.cz budou zmíněna veškerá nastavení a bude objasněno, k čemu slouží. U následujících ukázek nastavení budou již zmíněna pouze pole, která se budou nastavovat.

8.1 Justice.cz

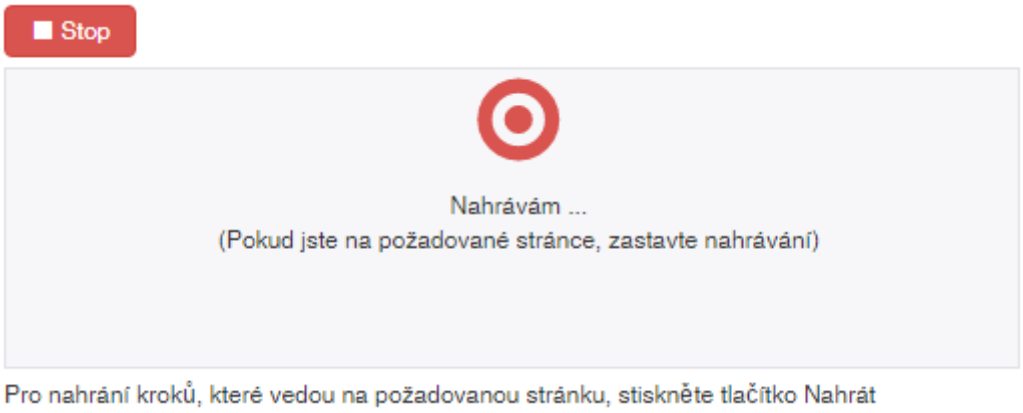
Nejdříve je potřeba vytvořit šablonu s nastavením. Na úvodní obrazovce se v pravém horním rohu klikne na tlačítko *Vytvořit šablonu*. Zobrazí se obrazovka s formulářem obsahujícím prvky pro nastavení scrapování. Na této obrazovce se jednotlivá políčka vyplní následovně:

Do *Názvu šablony* se vloží název, který bude odpovídat tomu, pro jaký web je tato šablona určená. Do políček *Uživatelské jméno*, *Server* a *Heslo* se vyplní přístupové údaje k serveru, kde běží serverová část aplikace. Do pole *URL* se vloží výchozí url, ze které bude začínat scrapovací proces. URL je třeba uvádět i s http:// nebo https://. Pokud je potřeba aplikaci omezit na počet provedených kroků, vyplní se tento počet do políčka *Počet kroků*. V daném případě bude třeba počet kroků omezit, aby scrapovací nástroj na webu Justice.cz nebyl vyhodnocen jako robot, proto se do tohoto políčka vyplní 3000. *Přeskočené stránky* se nevyplňují. Pole slouží pro vepsání regulárního výrazu. Stránky odpovídající tomuto výrazu budou přeskočeny. *Procházení celého webu* se ponechá nezaškrtnuté. *Procházení celého webu* slouží pro zapnutí možnosti, kdy scraper bude automaticky procházet celý web. Nakonec se klikne na *Uložit šablonu*.



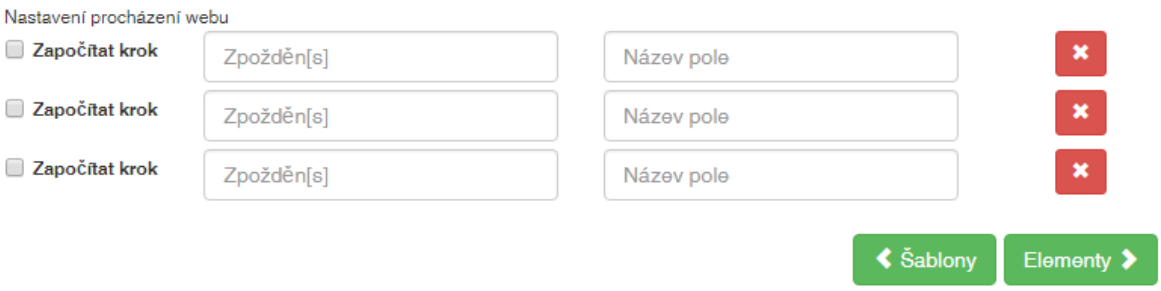
Obrázek 21 – Formulář pro nastavení šablony

Po uložení šablony se zobrazí obrazovka s nahráváním kroků uživatele na webové stránce. Na webu Justice.cz je potřeba nastavit formulářové prvky a následné kroky pro zobrazení výpisu informací o společnosti. Nahrávání těchto kroků se spustí pomocí tlačítka *Nahrát*. Následně se provedou požadované úkony na webové stránce, které vedou k výpisu informací o společnosti.



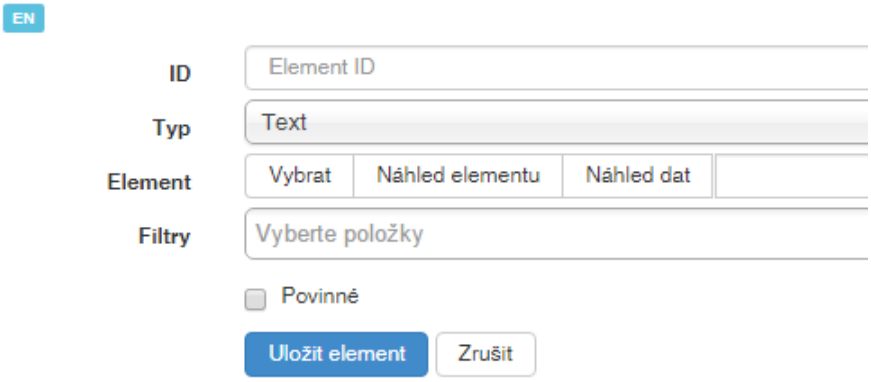
Obrázek 22 – Indikace spuštěného nahrávání

Během nahrávání kroků je pro jednotlivé kroky zobrazováno další nastavení *Započítat krok*, *Zpoždění* a *Název*. *Započítat krok* se zaškrtne pro kroky: klinutí na tlačítko vyhledat a pro kliknutí na odkaz výpis platných. Tímto se určí, že dané kroky budou započítány do limitu. *Název* se vyplní pro formulářové prvky. Tento název bude následně použit v konfiguračním souboru pro nastavení různých kombinací formuláře.



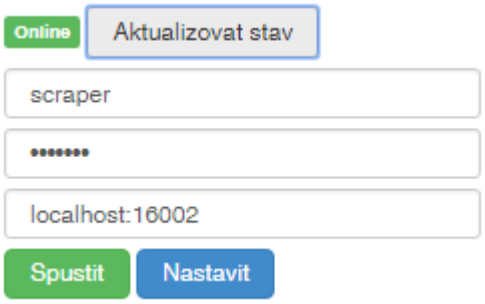
Obrázek 23 – Nastavení jednotlivých kroků

Po příchodu na požadovanou stránku nahrávání vypneme tlačítkem *Stop*. Poté se přejde na obrazovku s elementy, které budou vybrány. Nový element se vytvoří kliknutím na tlačítko *Nový element*. Na obrazovce se vyplní *ID*, což je název elementu. Následně se vybere *Typ*. *Typ* určuje, jaká data se mají zpracovávat. Poté se klikne na tlačítko *Vybrat* a kliknutím ve stránce na požadovaný element se získá jeho identifikátor. Potvrzení výběru se provede pomocí tlačítka *Vybrat*, které se nachází nad konzolí prohlížeče. Nakonec se vyberou *Filtry*, pomocí kterých dojde k odstranění nežádoucích informací. Pokud je potřeba určit, že daný element je na stránce povinný, zaškrtne se *Povinné*. Pole *Povinné* funguje tak, že pokud se na dané stránce element nenachází, pak se zahodí doposud získaná data ze stránky a ta je vyhodnocena jako stránka, která neobsahuje žádné informace. Pro uložení elementu se klikne na *Uložit element*. Pokud je potřeba přidat další elementy, postup se bude opakovat.



Obrázek 24 – Formulář pro nastavení elementu

Nakonec se přejde na obrazovku scraping, kde se spustí samotný proces scrapování. Pokud se jedná o první spuštění, je potřeba kliknout na tlačítko *Nastavit,* a tím se provede nastavení scrapování podle nakonfiguraované šablony.



Obrázek 25 – Spuštění a nastavení scrapování

Pro justici.org je potřeba ještě nastavit různé kombinace pro formulář. Toto se provede tak, že nejdříve se stáhne konfigurační soubor pomocí tlačítka *Konfigurační soubor pro formuláře*. Poté se do souboru přidají jednotlivé kombinace nastavení formuláře tak, že řádek odpovídá jednomu celému nahranému průchodu webu a sloupce jsou jednotlivé formulářové prvky. Po nastavení a uložení se tento soubor nahraje na server pomocí formuláře pro nahrání souboru, který se nachází pod výpisem informací o průběhu scrapování. Nyní je vše nastavené a může se pustit scrapování tlačítkem *Spustit*.

Obrázek 26 – Výstup z procesu scrapování, nastavení formuláře a proxy serveru

Výsledky scrapování lze stáhnout pomocí tlačítek nazvaných *Výsledky*. Podle formátu souboru lze výsledky stáhnout buďto v souboru CSV nebo JSON.

8.2 Internetový obchod

Nastavení pro získání produktů z internetového obchodu se provede následovně. Vytvoření šablony je podobné jako pro web Justice.cz s několika drobnými rozdíly. *Počet kroků* nebude vyplněn, což zaručí procházení celého webu. Pole *Přeskočené stránky* bude rovněž nevyplněno, protože je požadováno procházení celého webu. A zaškrtne se *Procházení celého webu*.

Po uložení je zobrazena stránka pro nahrání kroků uživatele, vedoucích k cílové stránce. Toto nastavení se může vynechat, protože bude navštívena každá stránka na webu a přeskočí se na nastavení elementů.

Nastavení elementů probíhá stejně jako u Justice.cz. Na cílové stránce se vybere, které elementy je potřeba ze stránky získat.

Po nastavení elementů se přejde na obrazovku pro scrapování. Zde stačí pouze nastavit scrapování pomocí tlačítka *Nastavit*. V tomto případě není potřeba nastavovat konfigurační soubor pro formuláře, jelikož nebylo provedeno nahrávání úkonů na webu. Po nastavení lze spustit scraping pomocí tlačítka *Spustit*.

8.3 Web s nutností přihlášení

Příkladem může být internetový obchod, který přihlášeným zákazníkům zobrazí produkty s nižší cenou. Nastavení šablony je stejné jako v předešlé kapitole o nastavení scrapování pro internetový obchod. Jedinou změnou je nastavení pole *Přeskočené stránky,* kde je potřeba zadat regulární výraz pro URL na odhlášení, aby během scrapování nedošlo k odhlášení uživatele, a tím ke zkreslení cen.

V následujícím kroku se pomocí nahrávání zachytí přihlášení uživatele do internetového obchodu. Nahrávání je popsané v kapitole Justice.cz. V dalším kroku se nastaví elementy, pomocí kterých se získají informace ze stránky.

Na poslední stránce, kde se spouští scrapování, je potřeba spustit inicializaci nástroje. V tomto případě není potřeba nastavovat konfigurační soubor pro formuláře, jelikož ke spuštění nahraných kroků dojde pouze jednou, a to na začátku celého scrapování. Po inicializaci se již pouze pustí scrapování pomocí tlačítka *Spustit*.

8.4 Opakované spuštění na serveru

Pro možnost opakovaného spouštění je třeba nejprve provést kroky vedoucí k vytvoření konfigurace a následně nahrát konfiguraci na server. Poté je již možno spouštět scrapování opakovaně pomocí nastavení cronu. Do cronu se přidá následující příkaz, který umožňuje spouštět scrapování z příkazové řádky:

```
casperjs cesta/k/scraper.js --load-plugin=nazev --ssl-protocol=any
     --proxy=proxy_server --proxy-auth=uzivatelske_jmeno:heslo
```

V předchozím příkazu jednotlivé parametry znamenají následující údaje:

V parametru *load-plugin* se místo *nazev* vloží název pluginu, který se má spustit. Parametr *ssl-protocol* slouží k tomu, aby nástroj mohl přistupovat ke stránkám se ssl certifikátem. Zbylé dva parametry *proxy* a *proxy-auth* slouží k připojení se k proxy serveru. Za *proxy_server* se vyplní adresa proxy serveru i s portem, za *uzivatelske_jmeno* uživatelské jméno pro přihlášení k proxy serveru a za *heslo* heslo pro přihlášení.

9 Získané výsledky během scrapování

Testování probíhalo na běžném počítači. Naměřené časy se mohou lišit z důvodu různě výkonných počítačů, rychlosti internetu a dalších faktorů ovlivňujících scrapování. Během testování scrapování bylo zajímavé porovnat rychlosti scrapování, pokud se stránky načítaly s obrázky a pokud bylo načítání obrázků vypnuté. Dosažené výsledky lze porovnat v Tabulka 1. Porovnávání bylo testováno ve všech případech užití. Pro každý případ bylo provedeno 5 pokusů a výsledný čas je průměrem časů těchto pokusů.

Tabulka 1 – Porovnání časů scrapování s načtenými obrázky a bez obrázků

Případ užití	**S načítáním obrázků**	**Bez načítání obrázků**
Justice.cz (Průchod pomocí nahraných kroků, vyhledání pěti firem)	18s	18s
E-shop (Procházení celého webu, s omezením počtu kroků na 10)	37s	45s
E-shop s přihlášením (Kombinace procházení celého webu a nahraných kroků, s omezením počtu kroků na 10)	59s	79s

Z naměřených časů vyplívá, že načítání obrázků zásadně ovlivňuje rychlost scrapování. Pro web Justice.cz jsou časy podobné (liší se až v milisekundách) z důvodu malého množství obrázků. Pro internetový obchod je však rozdíl velký. Jelikož v dnešní době lze nalést jen malé množství webů, které neobsahují obrázky, je načítání obrázků vypnuté.

V několika případech se stalo, že scrapování se nepovedlo z důvodu vytížení serveru a velmi pomalé odezvy. Z tohoto důvodu docházelo k tomu, že se nestáhla všechna požadovaná data.

10 Závěr

Ze zadání diplomové práce vyplynulo, že je zapotřebí nalézt nebo vyvinout nástroj, který by umožňoval získávání dat z internetových stránek. Během porovnávání různých nástrojů byly vyhodnoceny podle požadavků jako nejlepší nástroje placené. Tyto nástroje obsahovaly velké možnosti nastavení, ale měly i vlastnosti, které mohly scrapování znesnadnit. Příkladem byla nemožnost nastavit si proxy server. Z těchto důvodů jsem se rozhodl využít výhody jednotlivých nástrojů a navrhnout vlastní řešení.

Po zvážení všech možných programovacích jazyků byl zvolen javascript. V dnešní době lze javascript spouštět na serveru díky Node.js, i na klientské stanici. Díky použití javascriptu bylo možné využít internetový prohlížeč Chrome, pro který jdou psát doplňky v javascriptu, a následně je publikovat v obchodě s aplikacemi pro Chrome.

Celý nástroj byl rozdělen na dvě části tak, aby mohl běžet na počítači klienta i na serveru. K tomuto rozdělení byla využita architektura klient-server. Část pro nastavení scrapování běží na straně klienta. Tato část byla vytvořena jako doplněk pro internetový prohlížeč Chrome. Serverová část byla rozdělena na další dvě části. Část webového serveru a aplikace pro CasperJS, kde webový server komunikuje s klientskou částí a následně spouští proces scrapování.

Nástroj plně umožňuje scrapovat data podle případů užití zmíněných v kapitole Případy užití. Pomocí základních filtrací a typů scrapovaných dat lze z webové stránky jednoduše získat požadovaný výstup.

Do nástroje se povedlo naimplementovat chybějící funkčnosti u analyzovaných nástrojů. Nástroj je uživatelsky příjemný a umožňuje volit elementy pouze výběrem ze stránky. Pro pokročilé uživatele umožňuje vybrat elementy zápisem konkrétního selektoru. Díky možnosti spouštění scrapování na serveru lze spouštět scrapování automaticky v pravidelných intervalech.

Pro rozsáhlou funkčnost se nepovedlo naimplementovat veškeré možnosti nastavení, které jsou u komerčních aplikací. Bylo by dobré doimplementovat *white list*, umožňující podle regulárního výrazu spouštět scrapování pouze na stránce odpovídající regulárnímu výrazu. Dále by bylo dobré umožnit ručně donastavit selektor i pro nahrané kroky. Velkým přínosem by bylo doimplementovat možnost obcházení zabezpečení pomocí *captchy*.

Obcházení zabezpečení by mohlo být implementováno tak, že by zaslalo obrázek captchy ke zpracování nějakému systému třetích stran, a ten by následně vrátil text zobrazovaný na obrázku captchy.

Vedlejším využitím tohoto nástroje je plnění dat, kdy díky možnosti vyplňovat formulářové prvky, nastavení vyplňovaných dat a následnému odeslání formuláře lze data uložit. Takto byl nástroj použit pro naplnění produktů do internetového obchodu a ušetřil tak rutinní práci.

Celkově hodnotím nástroj jako dobře použitelný jak běžným, tak pokročilým uživatelem. Pro běžného uživatele přináší výhody v podobě snadného získávání dat z internetových stránek. Pokročilému uživateli přináší i možnost ručně si doupravit výběr prvků či zasáhnout do běhu scrapování pomocí metod helperu daného pluginu.

Seznam použité literatury

[1] TEIXEIRA, Pedro Dennis. *Professional node.js: building javascript based scalable software*. 1st ed. Indianapolis, IN: Wiley Pub., Inc., 2012. ISBN 11-181-8546-3.

[2] BELTRAN, Aries. *Getting started with PhantomJS*. Birmingham: Packt Publishing, 2013. ISBN 978-1-78216-423-4.

[3] BRÉHAULT, Éric. *Instant testing with CasperJS*. Birmingham, England: Packt Publishing, 2014. ISBN 978-1-78328-944-8.

[4] WILTON, Paul. *Beginning JavaScript*. 2nd ed. Indianapolis, IN: Wiley Pub., c2004. ISBN 07-645-5587-1.

[5] *THE SCRAPING THREAT REPORT 2015* [online]. Boston: ScrapeSentry, 2015 [cit. 2016-03-22]. Dostupné z: https://www.scrapesentry.com/wp-content/uploads/2015/06/2015_The_Scraping_Threat_Report.pdf

[6] *ScrapeSentry* [online]. San Francisco, 2016 [cit. 2016-04-13]. Dostupné z: https://www.scrapesentry.com

[7] *ASP.NET 5 Documentation* [online]. https://public.readthedocs.com/aspnet-aspnet/en/latest/: Microsoft, 2015 [cit. 2016-03-22]. Dostupné z: https://public.readthedocs.com/aspnet-aspnet/en/latest/

[8] *CasperJS* [online]. 2016 [cit. 2016-03-22]. Dostupné z: http://casperjs.org/

[9] *Import.io* [online]. San Francisco, 2015 [cit. 2016-03-22]. Dostupné z: https://www.import.io/

[10] *Web-Harvest* [online]. 2015 [cit. 2016-03-22]. Dostupné z: http://web-harvest.sourceforge.net/

[11] *Web Scraper* [online]. 2015 [cit. 2016-03-22]. Dostupné z: http://webscraper.io/

[12] *Resurrectio* [online]. 2015 [cit. 2016-03-22]. Dostupné z: https://github.com/ebrehault/resurrectio/blob/master/README.rst

[13] *WebSPHINX: A Personal, Customizable Web Crawler* [online]. Pittsburgh: Carnegie Mellon University, 2002 [cit. 2016-04-13]. Dostupné z: https://www.cs.cmu.edu/~rcm/websphinx/

[14] *Chrome* [online]. Kalifornie: Google, 2016 [cit. 2016-04-13]. Dostupné z: https://developer.chrome.com/extensions

Seznam obrázků

Seznam tabulek

Seznam zdrojových kódů

Seznam použitých zkratek

AJAX – Asynchronous JavaScript and XML

API – Application Programming Interface

CSS – Cascading Style Sheets

CSV – Comma-Separated Values

DOM – Document Object Model

GUI – Graphical User Interface

HTML – HyperText Markup Language

IČ – Identifikační číslo osob

I/O – Input/Output

IP adresa – Internet Protocol address

JSON – JavaScript Object Notation

MVC – Model-View-Controller

OS – Operating System

PHP – PHP: Hypertext Preprocessor

URL – Uniform Resource Locator

UI – User Interface

XML – Extensible Markup Language

XPath – XML Path Language

XSLT – Extensible Stylesheet Language Transformations

Přílohy

Obsah přiloženého CD

Na přiloženém CD se v kořenovém adresáři nachází tato diplomová práce ve formátu diplomova_prace.pdf. V adresáři source jsou přiloženy zdrojové kódy této diplomové práce, v adresáři images se nacházejí obrázky použité v této práci a v adresáři instalace se nacházejí instalační soubory.

FIM UHK
UNIVERZITA HRADEC KRÁLOVÉ
Fakulta informatiky a managementu
Rokitanského 62, 500 03 Hradec Králové, tel: 493 331 111, fax: 493 332 235

Zadání k závěrečné práci

Jméno a příjmení studenta: **Michal Kozderka**

Obor studia: Aplikovaná informatika (2)
Jméno a příjmení vedoucího práce: **Pavel Kříž**

Název práce:
Nástroj pro web-scraping

Název práce v AJ:
Web-scraping tool

Podtitul práce:

Podtitul práce v AJ:

Cíl práce: Navrhnout a implementovat intuitivní nástroj pro web-scarping formou rozšíření pro webový prohlížeč. Proces scrapingu bude možné spouštět na desktopu nebo serveru.

Osnova práce:
1. Úvod
2. Analýza existujících řešení
3. Návrh a implementace vlastního řešení
    1. Rozšíření pro webový prohlížeč
    2. Scraper pro desktop
    3. Scraper pro server
4. Testování a výsledky
5. Závěr

Projednáno dne: 10. 10. 14

Podpis studenta

Podpis vedoucího práce