Západočeská univerzita v Plzni

Fakulta aplikovaných věd

Katedra informatiky a výpočetní techniky

Bakalářská práce

Import dat ze služby Google Scholar do formátu XML

Plzeň, 2012 Tomáš Hanke

Prohlášení

Prohlašuji, že jsem bakalářskou práci vypracoval samostatně a výhradně s použitím citovaných pramenů.

V Plzni dne 3. května 2012.

Tomáš Hanke

Abstract

The Bachelor thesis deals with transforming data from Google Scholar to XML. It analyses methods and describes conditions for making this transformation possible and useful. At the beginning the Google Scholar and its main features are introduced. It also deals with problem how to cover up robotic accessing to the Google servers.

Thesis also examines HTTP protocol which is one of the main parts of communication through the internet network and describes its characteristic features.

The realization of the application is described in detail at the end and also workflow and online communication are outlined.

Poděkování

Tímto děkuji Ing. Daliboru Fialovi, Ph.D. za vedení mé bakalářské práce, zvláště pak za podnětné rady, připomínky a čas věnovaný mi při konzultacích.

Obsah

1 Úvod

Cílem projektu je vytvořit program, který umí získat informace o vědeckých publikacích ze služby Google Scholar a uložit je ve formátu XML nebo alespoň zjistit, zda a v jaké míře je toto možné. Aplikace by měla umět podle zadaných podmínek o autorovi, publikaci, případně podle data vydání vyhledat příslušné záznamy, data zpracovat a uložit ve formátu XML. Navíc je nutné, aby na serverech Googlu negenerovala příliš mnoho častých přístupů, což by vedlo k prozrazení robotického prohledávání jejich obsahu. Dalším úkolem tedy je zjistit, kolikrát a v jakém časovém horizontu je možno na servery Googlu přistoupit bez toho, aby vzniklo podezření na robotický přístup. Z toho plyne, že součástí programu bude i možnost pozastavení jeho činnosti na předem zvolený časový interval.

Google Scholar je webová služba, takže veškerá data předává ve formátu HTML. Jako nejlepší řešení pro získávání těchto dat se tudíž jeví HTML parser. K datům by šlo sice přistupovat jako k obyčejnému souboru uloženému v paměti, případně na pevném disku a obsahujícímu zdrojový kód webové stránky, ale zpracovávání takového souboru by bylo velice složité a extrémně neefektivní. Na druhou stranu vytvořit efektivní HTML parser také není jednoduché. Parser musí celý kód stránky projít, zanalyzovat a správně rozsekat na jednotlivé elementy, atributy, komentáře a obyčejný text. V ideálním případě by se měl být schopen vypořádat i s různými chybami a anomáliemi v kódu, jakými mohou být například překřížené nebo chybějící HTML tagy, případně zakázané atributy. V Javě již pro parsování HTML kódu existuje předpřipravená třída *javax.swing.text.html.HTMLEditorKit.ParserCallback*. Ta tvoří velmi slušný základ pro vytvoření vlastního HTML parseru; stačí překrýt některé její callback metody. Z názvu vyplývá další výhoda a to ta, že vlastní parsování HTML kódu probíhá ve vlastním vlákně nezávisle na zbytku programu. Zachytí-li vlákno, zpracovávající HTML stránku, nějaký prvek HTML kódu (např. počáteční tag), zavolá pouze příslušnou callback metodu, běžící ve vlastním vlákně, která se postará o další zpracování nalezeného prvku, zatímco parser současně pokračuje v prohledávání HTML dokumentu.

2 Google Scholar

Google Scholar je služba pro vyhledávání odborné literatury. Sdružuje informace o vědeckých knihách, článcích, dizertačních pracích od akademických nakladatelství po vědecké konference. Mezi hlavní funkce patří vyhledávání zdrojů, článků, abstraktů a citací, nalezení celého článku v knihovně či na webu, poskytnutí informace o klíčových článcích v jakémkoli vědním oboru, či dokonce stažení plného textu vybraných článků. Články jsou hodnoceny a uživateli nabízeny podle jejich relevantnosti [Osl12]. Vyhledávat je možné podle klíčových slov v názvu publikace nebo v celém popisku díla. Další možností je vyhledávání článků podle jména autora, názvu publikace a nalezené výsledky je možné filtrovat podle roku vydání. Anglická verze (stačí změnit jazykové rozhraní pod odkazem „Předvolby služby Scholar“) umožňuje vyhledávání omezit pouze na určené vědní obory.

Scholar je často srovnáván s weby jako Web of Science, Scopus nebo ACM Digital Library. Stejně jako tyto služby se snaží uživateli poskytnout pouze odborné články na určité téma a neobtěžovat ho nevědeckými informacemi. Databáze ale obsahuje i studentské práce, takže ne všechny články na Scholaru mají striktně vědecký charakter. Velkou výhodou je rozsah databáze. Scholar se nezaměřuje jen na některé vědní obory, ale shromažďuje články o jakýchkoli tématech a v jakýchkoli jazycích z celého světa [Nás12]. Není tedy problém najít článek psaný například japonsky nebo rusky.

Chceme-li zjistit, jestli někdo citoval některý z našich článků, Scholar nám to umožní pomocí vyhledávání mezi citacemi. Tím můžeme zjistit kdo, kdy a v jakém článku citoval naši práci. Každý uživatel si navíc může na Scholaru založit svůj vlastní účet, což mu umožní spravovat svoje práce, jejich citace a vykreslovat si citační statistiky svých článků.

Dalším plusem pro Google Scholar je fakt, že jeho použití je jednoduché a bezplatné. Odkazy na plné texty vyhledaných článků sice mohou vést do placených databází či uzavřených katalogů knihoven a univerzit, ale základní informace jsou k dispozici každému. Navíc je vyhledávání v databázi Scholaru téměř totožné s klasickým prohledáváním webu pomocí intuitivního vyhledavače Google, takže většina uživatelů se nemusí učit nic nového.

2.1 Předvolby

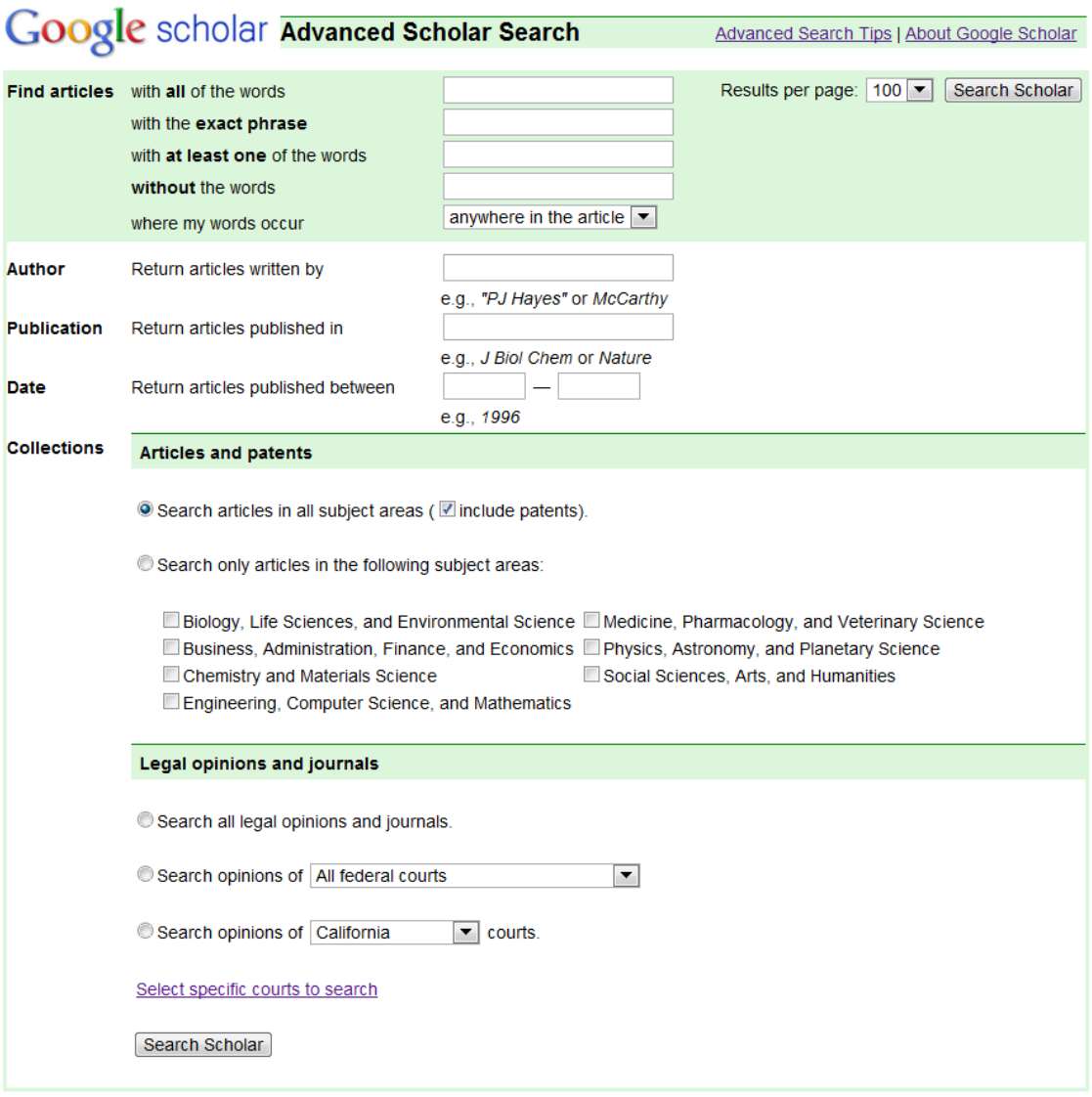
V nastavení předvoleb si uživatel může vybrat jazykovou mutaci Scholaru a jazyk, ve kterém musí být nalezené články napsány. Vybrat si může z třinácti nabízených světových jazyků, přičemž může zvolit jeden z nich, více jazyků najednou anebo nemusí jazyky omezovat vůbec, což je i výchozí a Googlem doporučená hodnota. Další nepovinnou možností je volba knihovny. Je možné vyhledat si kromě jiného i univerzitní knihovnu, v jejíž databázi se poté budou automaticky hledat publikace, které uživatel vyhledává pomocí služby Scholar. Mezi výsledky vyhledávání se poté budou zobrazovat i přímé odkazy na položky zvolených knihoven. Poslední důležitou funkcí je možnost nechat si zobrazovat odkazy na informace o nalezených publikacích ve formě určené pro import do jednoho z pěti programů určených ke správě citací v textu – BibTeX, EndNote, RefMan, RefWorks, WenXianWang.

2.2 Pravidla vyhledávání

Vyhledávací prvky Scholaru jsou z velké části podobné klasickému vyhledávání na Googlu. Ve vyhledávacím dotazu jsou vždy ignorovány rozdíly mezi malými a velkými písmeny. Ve většině případů není brán ohled ani na interpunkci včetně znaků „@#$%^&*()=+[]\“. Zde ale existují výjimky. Interpunkce mající konkrétní význam nebo vyskytující se v názvech ignorována není. Například C# nebo C++. Dále není ignorován znak dolaru ($), pokud je použit pro označení ceny. Informace v této podkapitole jsou podloženy studiem zdrojů [Tip12], [Zák12] a [Dal12].

Základním pravidlem je použití co nejmenšího počtu klíčových slov, které co nejpřesněji vystihují hledaný článek. Čím více klíčových slov bude použito v dotazu, tím méně článků bude vyhledáno, ale většinou budou i přesněji odpovídat hledanému tématu. Je tedy potřeba najít vhodný kompromis mezi stručností a konkrétností každého dotazu.

Vyhledávání službou Scholar lze jednoduše zpřesnit pomocí možnosti „Rozšířené vyhledávání služby Scholar“ (viz Obrázek 2-1), případně použitím doprovodných operátorů, přidaných ke klíčovým slovům v dotazu. Tyto operátory je možné v dotazu různě kombinovat.



Obrázek 2-1: Rozšířené vyhledávání služby Scholar

- **Hledání přesné fráze, uzavření klíčových slov do uvozovek**

Slova, uzavřená v uvozovkách, jsou vyhledávána jako fráze. To znamená, že ve výsledném textu se musí vyskytovat všechna slova uzavřená v uvozovkách a to přesně v tom pořadí, v jakém jsou zapsána. Navíc nesmí být dodatečně skloňována. Například dotazu [*havran polní*] budou vyhovovat všechny články, obsahující jak slovo „*havran*“, tak i slovo „*polní*“ v jakémkoli pořadí a pádu. Tedy například věta „Polního havrana můžeme spatřit...“ je pro tento typ dotazu zcela vyhovující. (Dokonce by tomuto dotazu vyhovovala i věta „Havran letěl přes řeku, zatímco hraboš polní...“) Avšak dotazu [„havran polní“] by tato věta již nevyhovovala a daný článek by byl z výsledků hledání vynechán. Dokonce mu nevyhovují ani věty „Havrana polního můžeme spatřit...“, kde je zachován alespoň slovosled (ale ne pád), ani „Polní havran může být spatřen...“, zachovávající pro změnu skloňování (nikoli však slovosled).

- **Hledání alespoň jednoho klíčového slova, použití operátoru „*OR*“**

Máme-li více klíčových slov, ale stačí nám, aby se ve výsledném článku objevilo vždy alespoň jedno z nich, poslouží nám operátor „*OR*“. Použijeme-li předchozí příklad s havranem polním, upravíme dotaz na [havran OR polní]. Takovému dotazu budou kromě článku „Havran polní“ vyhovovat i články s názvy „Polní zelinářství“, „Hraboš polní“ nebo články napsané panem Havranem.

- **Vyloučení klíčového slova, použití operátoru „-“**

Při použití operátoru mínus před některým klíčovým slovem v dotazu způsobí, že všechny články, obsahující toto slovo budou z výsledků vyhledávání vyřazeny. To se hodí v situaci, kdy vyhledáváme podle slova, které může označovat více věcí. Například dotazu [polní -havran] budou opět vyhovovat články „Polní zelinářství“ nebo „Hraboš polní“, ale článek „Havran polní“ bude z výsledků vyhledávání vyřazen. Dotaz [havran –author:havran] zase vyhledá články, obsahující slovo „havran“ a zároveň vynechá ty články, které byly napsány panem Havranem (o operátoru *author* viz níže).

- **Vyhledávání běžně ignorovaných znaků, použití operátoru „*+*“**

V úvodu této podkapitoly je zmíněno, že určité znaky jsou ve vyhledávácím dotazu obvykle ignorovány. Pro zobrazení i těch článků, které obsahují tato běžně ignorovaná slova, písmena a čísla, slouží operátor „*+*“, který vyhledávací algoritmus donutí brát zřetel na znaky po něm bezprostředně následující. Například dotaz [czech & slovak] zobrazí i články, které neobsahují znak „&“, kdežto dotazem [czech +& slovak] dáváme vyhledávači najevo, že o články, znak „&“ neobsahující, nemáme zájem.

- **Vyhledávání podle autora, použití operátoru „author“**

Pro vyhledávání článků podle jména jejich autora stačí do vyhledávacího pole zadat příjmení autora, například [knuth]. Chceme-li vyhledávat i podle křestního jména, doporučuje se uvést celé jméno do uvozovek, například [„donald knuth“]. Většinou jsou ale místo celého křestního jména indexovány pouze iniciály, proto je dobré je uvádět i v dotazu. Místo dotazu [„donald knuth“] je tedy lepší uvést [„d knuth“]. Zároveň není dobré se křestního jména vzdávat úplně,

protože je to údaj, zpřesňující výsledky vyhledávání. Z toho vyplývá, že nejlepší variantou budou dotazy ve tvaru [„d knuth“], [„de knuth“] nebo [„donald e knuth“].

Problém může nastat v situaci, kdy jméno autora tvoří nějaké běžné podstatné jméno. Použijme opět příklad s klíčovým slovem „havran“. V takovém případě by vyhledávač nevěděl, zda má uživatel na mysli zvíře nebo ho zajímají články napsané autorem s příjmením Havran. Pro tuto situaci existuje operátor „author“, který říká, že bezprostředně následující řetězec znaků, určuje jméno autora článku. Velmi důležité je, že mezi operátorem a vlastním vyhledávaným výrazem nesmí být mezera. Zajímají-li nás tedy články napsané panem Havranem, nejlepší verze dotazu je [author:havran]. Chceme-li uvést i křestní jméno, musíme již celý řetězec, představující jméno autora, uzavřít do uvozovek. Tedy pro články pana Vlastimila Havrana bude dotaz vypadat takto: [author:“v havran“].

Chceme-li hledat články podle více autorů, stačí v dotazu uvést operátor „*author*“ vícekrát. Například při hledání článků, napsaných současně pány Knuthem a Grahamem, dostaneme vyhledávací dotaz [author:knuth author:graham].

- **Hledání v názvu článku, použití operátoru „*intitle*“**

Potřebujeme-li vyhledávat určitá slova pouze v názvu článku, použijeme operátor „*intitle*“. Například dotaz [intitle:havran], zobrazí jen ty články, které mají slovo „*havran*“ v názvu. Pokud se dané slovo v článku vyskytuje, ale není uvedeno v jeho titulku, bude článek ignorován. Tento operátor akceptuje opět pouze slovo bezprostředně následující, takže mezi nimi nesmí být mezera a potřebujeme-li v titulku článku vyhledávat více slov, uzavřeme je do uvozovek, například [intitle:“havran polní“]. Může se stát, že potřebujeme vyhledávat články podle slov, které spolu nesouvisí a nevyskytují se tedy v článku jako fráze. V takovém případě je možné použít operátor „*intitle*“ vícekrát. Tedy i dotaz [intitle:havran intitle:kavka] je zcela v pořádku a bude hledat pouze články obsahující v názvu slova „*havran*“ i „*kavka*“ zároveň.

- **Omezení publikačních zdrojů**

Scholar umožňuje vyhledávat články s ohledem na to, kde byly publikovány. Vyhledávání lze omezit pouze na určený časopis nebo konferenci (či spíše sborníky konferencí). Pro tuto možnost neexistuje žádný operátor a je tedy přístupná pouze přes možnost „*Rozšířené vyhledávání služby Scholar*“. Na této stránce je políčko s názvem „*Publikace*“, do něhož je třeba uvést název časopisu, v němž byl námi hledaný článek publikován (například „European Nuclear Young Generation Forum“ nebo „Kybernetika“). Do klasického vyhledávacího pole zadáme klíčová slova článku a kliknutím na tlačítko „*Prohledat Scholar*“ zobrazíme výsledky hledání. V případě, kdy zadáme pouze název časopisu bez dalších klíčových slov, zobrazí se všechny články, publikované v daném časopisu a zaindexované Scholarem. Při přidání omezení datem lze vyhledávání omezit kromě časopisu ještě časovým intervalem publikování článku (omezení datem viz níže).

- **Omezení datem**

Dále Scholar umožňuje vyhledávat články s ohledem na to, kdy byly publikovány. Vyhledávání lze omezit časovým intervalem v rámci let. Pro tuto možnost neexistuje žádný operátor a je tedy přístupná pouze přes možnost „*Rozšířené vyhledávání služby Scholar*“. Na této stránce

jsou dvě políčka s názvem „*Datum*“, do nichž je třeba uvést roky (případně interval), během nichž byl námi hledaný článek publikován. Zde máme na výběr několik možností.

Chceme-li vyhledávání omezit jen spodní časovou hranicí (například pro zobrazení nejnovějších článků v daném oboru), zadáme rok (např. „2004“) pouze do prvního pole, čímž říkáme, že nás zajímají jen články, vydané od roku 2004 později. Při zadání roku jen do druhého políčka, budou vyhledávány články, vydané před zadaným rokem. Pokud vyplníme obě políčka, nastane jedna ze dvou situací. Zadáme-li do obou políček shodný rok, zobrazí se pouze články vydané v daném roce. V případě, že byl zadán časový interval (například 2004-2009), budou vyhledány články z tohoto intervalu. Mezi zobrazenými články budou vždy i ty, vydané v mezích zadaného intervalu (tedy články vydané v roce 2004 včetně až 2009 včetně).

2.3 Výsledky vyhledávání

U každého nalezeného článku je uveden jeho typ, název práce, jméno autora (případně jména několika autorů, nemusí být vyjmenováni všichni), rok publikování, nakladatelství, krátký popis práce (někdy se jedná o vybraný kus textu) a několik odkazů, viz Obrázek 2-2.

- **Počet citací tohoto článku: X – (Cited by: X)**

Ihned pod popisem publikace je uveden údaj o počtu článků, které danou publikaci citují a které jsou zároveň uvedeny v databázi Scholaru. Po rozkliknutí odkazu se tyto články zobrazí ve stejné formě, tedy včetně popisu, detailů a příslušných odkazů. Tímto způsobem lze do libovolné hloubky vyhledávat citace citací daného článku.

- **Související články – (Related articles)**

Pro každý článek se Google Scholar snaží automaticky najít práce, které s daným článkem tematicky souvisí [Náp12]. Tato funkce je užitečná v tom případě, že se chceme o daném tématu dozvědět více informací. Související práce jsou seřazeny především podle jejich podobnosti s původním článkem, ale i podle relevance každé práce [Náp12].

- **Archiv – Cached**

V případě, že je článek volně vystavený na internetu, může být k dispozici jeho předchozí verze zobrazená tak, jak si ji ve svém archivu zaindexoval Google při procházení webu. Tento odkaz tedy vede právě na archivovanou verzi článku a uživatel je informován o tom, že prohlíží pouze otisk dané stránky. Může se samozřejmě stát, že článek byl pozměněn nebo odstraněn. Archivovaná verze článku proto nemusí být aktuální.

- **Všechny verze (počet: Y) – (All Y versions)**
- **Zobrazit jako HTML – View as HTML**

V některých případech, je-li článek k dispozici ve formě PDF, převede ho Google do formátu HTML a zobrazí jako běžnou webovou stránku. Tato funkce ale není automaticky přístupná u všech PDF článků.

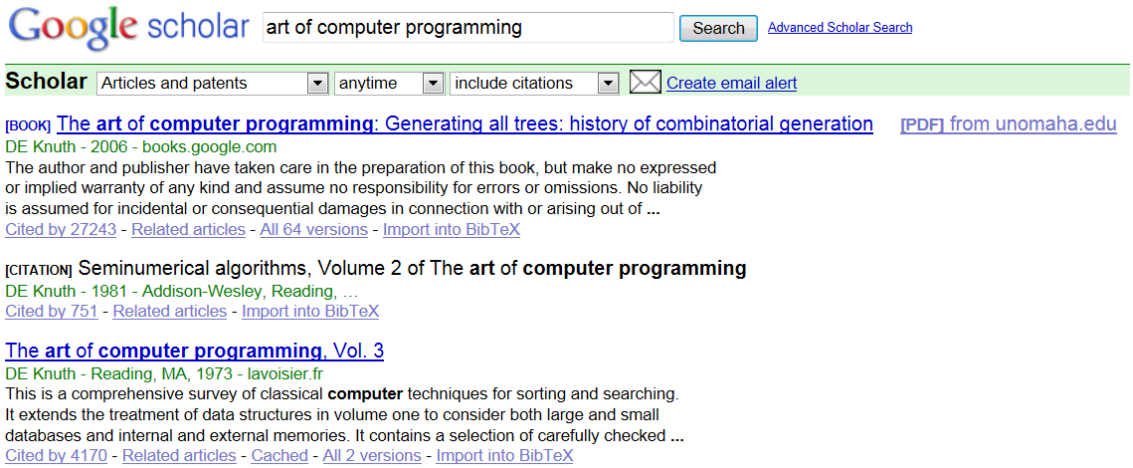
- **Import do programu BibTeX – (Import into BibTeX)**

Informace, shromážděné Scholarem o daném článku lze převést do formátu vhodného pro import do jednoho z pěti programů, určených pro správu citací (viz kapitola *Předvolby*). Tuto funkci je nutno aktivovat v nastavení Scholaru.

Ne vždy musí být všechny tyto odkazy dostupné. Někdy se stává, že Scholar nemá o článku dostatek informací, takže příslušnou funkci nezpřístupní. Například nemá-li informace o počtu citací, nezobrazí se odkaz „Počet citací tohoto článku“. Kromě výše zmíněných funkcí je u článku obvykle uveden ještě odkaz na jeho plný text, je-li k dispozici, nebo na placenou databázi, kde je článek ke stažení. V předvolbách si lze zapnout i zobrazování odkazů do databází některých knihoven.

Poslední důležitou položkou je určení typu vyhledaného záznamu. Ten je uveden v hranatých závorkách před názvem dané práce. Pro některé záznamy však informace o typu chybí. Oficiální popis jednotlivých typů článků se nepodařilo nikde najít, proto jsou dále popsána spíše vlastní pozorování a logická odvození.

- *KNIHA/BOOK* – jde o papírovou knihu, dostupnou v tištěné podobě v některé z knihoven;
- *CITACE/CITATION* – představuje pouze citaci článku, není ale k dispozici ani část jeho textu;
- *PDF* – článek nebo jeho část lze zobrazit ve formě PDF;
- *HTML* – text článku je publikován na webové stránce.



Obrázek 2-2: Ukázka výsledků hledání

2.4 Překážky získávání dat

Google si svá data pečlivě chrání a neumožňuje k nim snadný přístup. Pro službu Google Scholar bohužel neexistuje žádné oficiální API, které by získávání příslušných dat usnadnilo a standardizovalo. To ale není jediná překážka v cestě. Servery Googlu se brání robotickému procházení jejich obsahu a při podezření uživateli zabrání v přístupu. V první fázi jde o Captcha kódy, ve druhé již o chybovou hlášku *503 Service Unavailable* o dočasné nedostupnosti služby.

3 Hypertext Transfer Protocol

Celá aplikace využívá pro přístup k datům na serverech Scholaru protokol HTTP. Jedná se o bezestavový protokol určený pro komunikaci mezi klientem a serverem za účelem výměny informací. Klient (obvykle prohlížeč) se připojí k serveru (standardně na portu 80, což lze v nastavení serveru změnit) a odešle mu požadavek. Server ho zpracuje a pošle klientovi odpověď. Formát komunikace udává specifikace protokolu a liší se podle verze protokolu.

Nejstarší verze je 0.9 publikovaná v roce 1991 [The12]. Požadavek obsahuje pouze metodu GET následovanou relativní cestou k požadovanému dokumentu. Pro zobrazení stránky *http://www.mujportal.cz/uzivatel/profil.html* pomocí protokolu HTTP/0.9 je třeba se připojit na server *www.mujportal.cz* a zaslat požadavek ve tvaru „*GET /uzivatel/profil.html*“. V případě existence daného souboru server odpoví zasláním jeho obsahu. V opačném případě odešle výpis souborů v daném adresáři (je-li to v nastavení serveru povoleno). Požadavek musí vždy kromě funkce *GET* obsahovat cestu ve formě alespoň lomítka („*GET /*“). V takovém případě server prohledá kořenový adresář na přítomnost souboru s výchozím jménem (podle nastavení web serveru, typicky *index.html* nebo *index.php*). Nalezne-li ho, odešle klientovi informaci o jeho přesném umístění, aby o něj mohl znovu požádat. K ukončení spojení dochází automaticky po přenesení celého dokumentu. V rámci jednoho spojení je tedy možné obsloužit pouze jeden požadavek.

V roce 1996 byla publikována druhá verze protokolu, pod označením 1.0, která zavedla HTTP hlavičky a nové metody *HEAD* a *POST*. Metoda *HEAD* je identická s metodou *GET* s tou výjimkou, že server nesmí klientovi odeslat obsah souboru, ale pouze hlavičky obsahující metainformace (např. datum poslední změny). Tyto informace by měly být totožné s informacemi odeslanými metodou *GET* [Hyp19]. Metoda *POST* se používá pro předání dat z formuláře serveru.

Zatím poslední je verze 1.1, která umožňuje udržovat TCP spojení a tím odeslat serveru více požadavků v rámci jednoho spojení. To může být ukončeno jak serverem, tak i klientem. Stačí do požadavku přidat hlavičku „*Connection: close*“. Protokol také definuje nové metody *OPTIONS* (informace o nastavení spojení), *PUT* (uložení souboru na zadanou URL adresu), *DELETE* (smazání souboru ze serveru), *TRACE* (odeslání kopie dotazu zpět klientovi) a *CONNECT* (spojení se serverem při použití proxy s cílem vytvořit tunel, např. SSL spojení) [Hyp26]. Tvar HTTP požadavku ve verzi 1.0 a 1.1:

<<metoda>> <<URL adresa>> <<verze protokolu HTTP>>
<<hlavičky>>
<<prázdná řádka>>

Každá hlavička je na nové řádce ve tvaru <<klíč>>: <<hodnota>>. V případě, že se požadavek týká metody POST (tudíž přenášíme data z formuláře), následují za prázdnou řádkou ještě formulářová data opět ve tvaru <<klíč>>: <<hodnota>>. Jedinou povinnou hlavičkou je *Host*, která nese doménový název serveru, pro který je dotaz určen. Tato potřeba vznikla kvůli hostování více virtuálních serverů na jednom fyzickém, disponujícím pouze jednou IP adresou. Tuto hlavičku zavádí až protokol HTTP/1.1. Předtím bylo nutné mít pro každý

virtuální server vlastní IP adresu, což vedlo ke zbytečnému plýtvání adresami a znesnadňovalo přidávání nových virtuálních serverů.

Formát odpovědi je v HTTP/1.0 a HTTP/1.1 o něco složitější než ve verzi HTTP/0.9:

<<protokol>> <<stavový kód>> <stavové hlášení>>
<<hlavičky>>
<<prázdná řádka>>
<<obsah odpovědi>>

Protokol udává verzi použitého HTTP protokolu. *Stavový kód* a *stavové hlášení* popisují výsledek zpracování požadavku. *Kód* ve formě trojmístného čísla a *hlášení* ve formě slovního popisu. Hlaviček je v HTTP/1.1 definováno 47.

3.1 Stavové kódy protokolu HTTP/1.1

Stavové kódy slouží klientovi k bližší identifikaci stavu zpracování jeho požadavku serverem. Na základě toho, jaký kód klient od serveru dostane v odpovědi, se musí zachovat a přizpůsobit tomu svoje další požadavky. Kódy se dělí do skupin podle jejich charakteru na informační, úspěšné vyřízení požadavku, přesměrování, chyba na straně klienta a chyba na straně serveru.

Tato podkapitola byla sestavena na základě zdrojů [Kos99], [Hyp26] a [Sta12].

3.1.1 1xx Informační kódy

Tato skupina sdružuje kódy sestávající pouze z prvního řádku odpovědi a volitelných hlaviček. Předchozí verze protokolu HTTP nepodporují tyto stavové kódy, takže servery nesmí při komunikaci s klienty, používajícími starší verze protokolu, tyto hlavičky posílat.

- *100 Continue* – Počáteční část dotazu byla serverem zpracována a klient může pokračovat v zasílání požadavku.
- *101 Switching Protocols* – Server mění na žádost klienta komunikační protokol.

3.1.2 2xx Úspěšné vyřízení požadavku

Kódy ve druhé skupině indikují, že požadavek klienta byl doručen a úspěšně zpracován.

- *200 OK* – Požadavek byl úspěšně zpracován. Obsah odpovědi se odvíjí od použité metody. V případě metody GET odpovídá entitě požadované v dotazu. U metody HEAD jde pouze o hlavičky požadované entity. Byla-li použita metoda POST, obsahuje odpověď popis výsledku žádané akce a v případě metody TRACE je vrácen původní požadavek, přičemž odesílatelem je cílový server požadavku.
- *201 Created* – Úspěšné vytvoření nového objektu, přístupného přes URI, vrácené v odpovědi.
- *202 Accepted* – Požadavek byl přijat, ale jeho zpracování ještě není hotové. Návratová entita by měla obsahovat informaci o aktuálním stavu zpracování a předpokládaném času vyřízení.

- *203 Non-Authoritative Information* – Požadavek byl úspěšně zpracován, ale návratová informace pochází z jiného zdroje. Použití tohoto kódu není striktně vyžadováno a místo něj může být použit kód 200.
- *204 No Content* – Požadavek byl úspěšně zpracován, ale neexistuje žádný obsah, který by měl být klientovi odeslán. Odpověď může obsahovat aktualizované metainformace.
- *205 Reset Content* – Požadavek byl zpracován a klient by měl obnovit dokument, který vyvolal odeslání požadavku.
- *206 Partial Content* – Server odesílá pouze část objektu, požadovaného metodou GET, protože byla použita hlavička Range. Požadavek by v tomto případě měl obsahovat i hlavičku If-Range.

3.1.3 3xx Přesměrování

Tyto kódy klienta informují o tom, že pro vyřízení jeho požadavku je třeba provést další kroky. Požadované akce mohou být klientem provedeny bez vědomí uživatele, ale pouze tehdy, pokud je v následujícím dotazu použita metoda GET nebo HEAD. Klient by měl být schopen rozpoznat nekonečnou smyčku přesměrování. Předchozí verze specifikace doporučovaly jako maximum pět přesměrování.

- *300 Multiple Choices* – Požadovaný objekt odpovídá více dostupným objektům. Klient si může vybrat, který chce, a specifikovat ho v dalším požadavku.
- *301 Moved Permanently* – Požadovaný objekt byl natrvalo přesunut a je k dispozici na vrácených adresách.
- *302 Found* – Požadovaný objekt je dočasně k dispozici na vráceném URI.
- *303 See Other* – Požadovaný objekt je k dispozici na jiném URI pomocí metody GET. Používá se především pro přesměrování po použití metody POST.
- *304 Not Modified* – Odpověď na podmíněný dotaz, např. hlavičkou If-Modified-Since. Požadovaný dokument nebyl změněn, takže ho není třeba přenášet znovu. Odpověď nesmí obsahovat tělo, takže je vždy ukončena prázdným řádkem.
- *305 Use Proxy* – Požadavek musí být odeslán znovu pomocí proxy, jejíž URI je uvedeno v odpovědi.
- *306* (nepoužito) – Tento kód byl používán v předchozích verzích. Již se nepoužívá a je rezervován.
- *307 Temporary redirect* – Požadovaný zdroj byl dočasně přesunut na vrácené URI.

3.1.4 4xx Chyba klienta

Tato skupina kódů se používá v případě, kdy došlo k chybě u klienta. Server by měl do odpovědi zahrnout vysvětlení, proč nemohl vyhovět klientově žádosti, a určit zda se jedná o dočasný nebo permanentní stav.

- *400 Bad Request* – Požadavek nemůže být serverem zpracován, protože má chybnou syntaxi. Klient by neměl požadavek opakovat, pokud nedojde k jeho změně.
- *401 Unauthorized* – K vyřízení požadavku je potřeba autorizace klienta. Požadavek musí obsahovat hlavičku *HTTP-Authenticate*. Ta buď v požadavku chybí nebo byly poskytnuté údaje serverem odmítnuty.
- *402 Payment Required* – Rezervováno pro budoucí použití.

- *403 Forbidden* – Server rozumí požadavku, ale odmítá ho provést. Tato chyba nemá nic společného s autentifikací a požadavek by neměl být opakován. Server může odeslat důvod zamítnutí požadavku. Pokud ho nechce zveřejnit, může být místo tohoto kódu použit kód 404 Not Found.
- *404 Not Found* – Požadovaný objekt není dostupný. Neříká nic o tom, zda dočasně nebo permanentně. Byl-li objekt permanentně odstraněn, měl by být použit kód 410 Gone. Používá se v případě, kdy server nechce prozradit příčinu zamítnutí požadavku nebo když nelze aplikovat jiný stavový kód.
- *405 Method Not Allowed* – Pro požadovaný objekt (v hlavičce *Request-URI*) je volaná metoda (v hlavičce *Request-Line*) nepřístupná. Odpověď musí obsahovat hlavičku *Allow*, obsahující metody, přípustné pro daný objekt.
- *406 Not Acceptable* – Server může vygenerovat pouze odpověď, která je v rozporu s hlavičkami uvedenými v dotazu.
- *407 Proxy Authentication Required* – Klient musí být před provedením požadavku autentizován proxy serverem. Proxy musí vrátit hlavičku Proxy-Authenticate, s níž může klient požadavek opakovat.
- *408 Request Time-out* – Vypršel čas na odeslání požadavku klientem. Požadavek může být beze změn odeslán znovu.
- *409 Conflict* – Požadavek nemůže být proveden, protože příslušný zdroj není dostupný. Tento kód je povolen pouze v situacích, kdy se předpokládá, že klient bude schopen problém vyřešit a odeslat požadavek znovu. Odpověď by měla obsahovat informace vedoucí k rozpoznání konfliktu, nejlépe k jeho vyřešení.
- *410 Gone* – Požadovaný zdroj již není na serveru dostupný a neexistuje pro něj žádná náhradní adresa. Tento stav je považován za permanentní. Nelze-li rozhodnout, zda se nejedná o dočasný stav, měl by být použit kód 404 Not Found.
- *411 Length Required* – Server odmítl zpracovat požadavek, protože nebyla uvedena validní hlavička *Content-Length*, specifikující délku zprávy. Klient může opakovat požadavek po jejím přidání.
- *412 Precondition Failed* – Podmínka, předaná v požadavku, byla serverem vyhodnocena jako chybná.
- *413 Request Entity Too Large* – Server odmítl zpracovat požadavek, protože je příliš velký. Server může uzavřít spojení s klientem, aby mu zabránil v opakování požadavku. Je-li tento stav dočasný, odpověď by měla obsahovat hlavičku *Retry-After*, indikující, kdy se klient může pokusit požadavek opakovat.
- *414 Request-URI Too Large* – Server odmítl zpracovat požadavek, protože URI požadovaného zdroje je příliš dlouhé. Vyskytuje se například při chybné konverzi požadavku z metody POST na metodu GET.
- *415 Unsupported Media Type* – Server odmítl zpracovat požadavek, protože požadovaný zdroj není danou metodou podporován.
- *416 Requested range not satisfiable* – V požadavku je uvedena hlavička *Range* s nevalidní hodnotou (klient požaduje část objektu, která není k dispozici) a chybí hlavička *If-Range*.
- *417 Expectation Failed* – Server nemůže splnit předpoklad, uvedený v hlavičce *Expect*.

3.1.5 5xx Chyba serveru

Poslední skupina obsahuje kódy signalizující, že server při zpracovávání požadavku pochybil nebo není schopen daný požadavek vykonat. Stejně jako u kódů čtvrté skupiny by měl server do odpovědi zahrnout vysvětlení, proč nemohl vyhovět klientově žádosti, a určit zda se jedná o dočasný nebo permanentní stav.

- *500 Internal Server Error* – Server narazil na neočekávanou chybu, která mu znemožnila provedení požadavku.
- *501 Not Implemented* – Server nepodporuje metodu potřebnou k vykonání požadavku nebo ji nerozeznal.
- *502 Bad Gateway* – Server v roli brány (gateway) nebo proxy dostal neplatnou odpověď od nadřazeného serveru.
- *503 Service Unavailable* – Server aktuálně nemůže zpracovat požadavek. Nejčastěji zapříčiněno přetížením nebo údržbou serveru.
- *504 Gateway Time-out* – Server v roli brány nebo proxy neobdržel od nadřazeného serveru včasnou odpověď.
- *505 HTTP Version not supported* – Server nepodporuje verzi HTTP protokolu, specifikovanou v požadavku. Odpověď by měla obsahovat informaci o tom, proč není daná verze protokolu podporována a jaké další protokoly server podporuje.

4 Realizační část

Aplikace je napsána v programovacím jazyku Java ve verzi 1.6.0_24. K vývoji bylo použito vývojové prostředí *eclipse Helios* verze 3.6.2. Program byl vyvíjen pod operačním systémem Microsoft Windows 7 Professional 64bit a testován i na operačním systému Microsoft Windows XP Professional 32bit.

Vlastní program sestává z osmi tříd. Jsou to *BibtexParser*, *Farmar*, *Hlavni*, *HtmlParser*, *HttpClient*, *Okno*, *StAX* a *Zaznam*. Již jejich názvy napovídají, jakou činnost zhruba provádí. Jejich popis bude rozebrán níže.

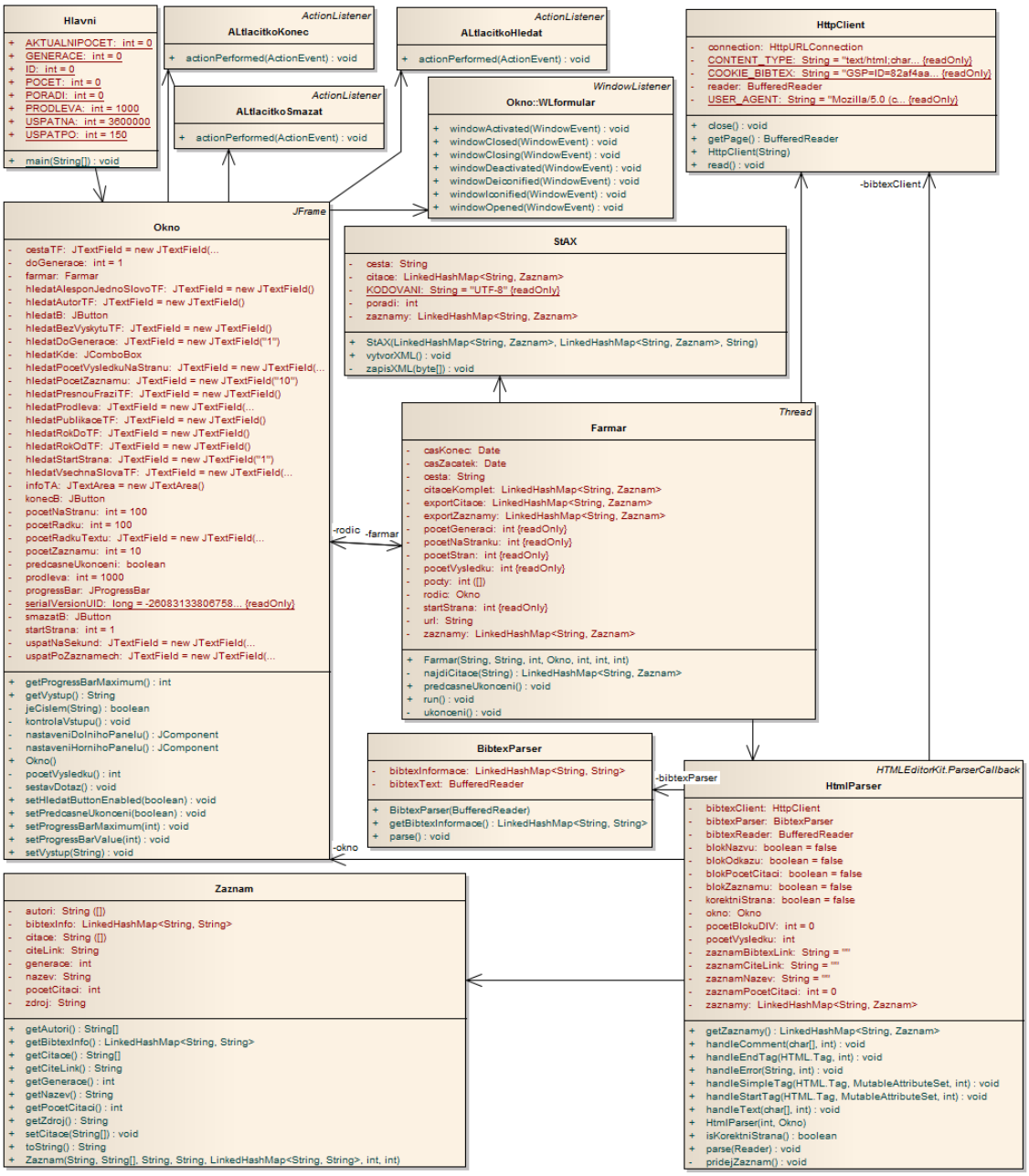
4.1 Stručně a souhrnně

Třída *Hlavni* pouze vytvoří nový objekt třídy *Okno*, čímž dojde k sestavení a zobrazení grafického prostředí aplikace. K vytvoření grafického rozhraní bylo použito knihoven Swingu. Po zadání kritérií vyhledávání a stisku tlačítka *Hledat* dojde k předání řízení třídě *Farmar*. Tato třída se stará o rozdělování práce mezi třídy *HttpClient* a *HtmlParser* a o kolekce, sbírající nalezené a zpracované záznamy.

Tímto je aplikace teoreticky připravena pro přechod na vícevláknovou architekturu. Toho by se dalo využít zejména v tom případě, když by se na grabování výsledků hledání podílelo více počítačů s rozdílnými veřejnými IP adresami. Za dobu, než by servery Googlu zjistily, že jde o strojové prohledávání jejich obsahu a zabránily aplikaci v přístupu, by každý počítač nasbíral přibližně stejné množství dat. Ta se předá třídě *Farmar*, který je shromáždí a provede s nimi další akce. V současné podobě nejsou vlákna do programu implementována, ale program je připraven na jejich použití.

Třída *HttpClient* má za úkol stáhnout požadovanou HTML stránku a její kód předat třídě *HtmlParser*, jež se postará o její parsování a tím nám získá kýžené informace. Třída *HtmlParser* ke své práci využívá třídu *BibtexParser*, parsující informace o článcích v podobě dat pro Bibtex. Tato data mají specifický formát a nelze pro ně použít klasický HTML parser. Nakonec se získané informace předají třídě *StAX*, starající se o jejich zapsání do XML souboru.

Vztahy mezi jednotlivými třídami znázorňuje diagram tříd na obrázku 4-1. Přibližný průběh vykonávání programu je zobrazen v příloze B. Kvůli velkým rozměrům obou obrázků je pro jejich detailní prohlížení nutné zobrazit si elektronickou verzi, dostupnou na přiloženém CD.



Obrázek 4-1: Diagram tříd

4.2 Struktura výsledného XML souboru

Všechna důležitá data o zpracovaných publikacích jsou uložena do XML souboru, jehož struktura je navržena a přizpůsobena dalším potřebám Katedry informatiky a výpočetní techniky, která bude vygenerovaný XML soubor používat ve svých dalších projektech. Jednotlivé publikace jsou v souboru seřazeny podle generací a v rámci každé generace podle toho, jak byly postupně nacházeny.

Následuje výčet elementů a atributů XML souboru se stručným popiskem:

- *<?xml>* – obvyklá hlavička XML dokumentu, obsahuje atributy *version*, *encoding* a *standalone*.
- *<publications>* – kořenový element XML souboru. Sdružuje záznamy o všech nalezených publikacích. Obsahuje podelement *<publication>* a nemá žádné atributy.
  - *<publication>* – element XML souboru, představující jednu nalezenou publikaci a sdružující veškeré dostupné informace o daném záznamu. Obsahuje podelementy *<title>*, *<authors>*, *<source>* a *<citations>* a atributy *id*, *number* a *generation*.
    - *<title>* – element, obsahující úplný název dané publikace/článku/knihy apod. Nemá žádné další podelementy ani atributy.
    - *<authors>* – element, obsahující kompletní výčet jmen všech autorů, podílejících se na tvorbě dané publikace. Každé jméno je uzavřeno podelementem *<author>*.
      - *<author>* – element, obsahující jméno autora dané publikace ve formátu „*Příjmení, J.*“ Nemá žádné další podelementy ani atributy.
    - *<source>* – element, sdružující veškeré podrobnosti o dané publikaci jako například rok vydání, nakladatelství, název a číslo časopisu, v němž článek vyšel, svazek knihy (například sborníky z konferencí), v níž se článek nachází, a výčet stran, na kterých je vytištěn. Tento element je velice proměnný, jeho elementy závisí na tom, kolik informací má o dané publikaci Scholar nashromážděných. Obsahuje atributy *type* a *complete* a mezi jeho nejčastější podelementy patří *<journal>*, *<volume>*, *<number>*, *<booktitle>*, *<pages>*, *<year>* a *<publisher>*.
      - *<journal>* – element, obsahující název časopisu, v němž byl daný článek publikován. Nemá žádné další podelementy ani atributy.
      - *<volume>* – element, obsahující číslo časopisu, v němž byl daný článek publikován. Nemá žádné další podelementy ani atributy.
      - *<number>* – element, obsahující číslo svazku knihy. U časopisu může určovat jeho ročník. Nemá žádné další podelementy ani atributy.
      - *<booktitle>* – element, obsahující název části knihy, obsahující daný článek. Obvykle se jedná o sborníky z konferencí nebo knihy, vycházející na pokračování. Nemá žádné další podelementy ani atributy.
      - *<pages>* – element, obsahující rozmezí stránek, kde se daný článek nachází. Typické pro časopisy a sborníky. Nemá žádné další podelementy ani atributy.
      - *<year>* – element, obsahující rok vydání dané publikace. Nemá žádné další podelementy ani atributy.
      - *<publisher>* – element, obsahující název vydavatele dané publikace, případně časopisu, v němž článek vyšel nebo konference, na níž byl publikován. Nemá žádné další podelementy ani atributy.
      - *type* – atribut, určující typ článku tak, jak ho ohodnotil Google Scholar. Může obsahovat jednu z následujících hodnot: *book* (jedná se o celou knihu, případně o její jeden celý svazek), *article* (jde o článek v časopisu, ne o celou knihu), *inproceedings* (vše ostatní včetně konferencí).
      - *complete* – atribut, obsahující informace ze všech podelementů elementu <source> oddělených znakem „;“ (středník).

- *<citations>* – element, sdružující informace o článcích, které danou publikaci citují. Jednotlivá ID citujících článků jsou uzavřena v podelementech *<citedby>*. Element obsahuje atributy *count* a *link*. Položky poslední prohledávané generace již tento element neobsahují, protože dané informace pro ně chybí (citující publikace již nebyly hledány).
  - *<citedby>* – element, obsahující identifikátor záznamu, který danou publikaci cituje. Nemá žádné další podelementy ani atributy.
  - *count* – atribut, obsahující celkový počet záznamů, citujících danou publikaci podle Scholaru. Toto číslo tedy neodpovídá počtu stažených citujících publikací, ale počtu všech citujících publikací, které má ve svých záznamech Google Scholar.
  - *link* – atribut, nesoucí přímý odkaz na stránku s publikacemi, které citují aktuální záznam.
- *id* – atribut, nesoucí informaci o jednoznačném identifikátoru daného záznamu. Ty jsou převzaty ze služby Google Scholar, případně, pokud Scholar pro některou publikaci identifikátor nezveřejní (to se stává v tom případě, že Scholar nemá k dané publikaci žádné informace o publikacích citujících), je programem vygenerován vlastní identifikátor. Elementy *<citedby>* obsahují právě tyto číselné identifikátory.
- *number* – atribut, určující pořadové číslo publikace. Číslo v tomto atributu u posledního záznamu XML souboru určuje celkový počet nalezených publikací.
- *generation* – atribut, určující číslo generace, v níž byla daná publikace nalezena.

Ukázka vygenerovaného XML souboru o jednom záznamu nulté generace se třemi citujícími položkami:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<publications>
 <publication id="12187675114908023502" number="1" generation="0">
  <title>The art of computer programming, Vol. 3</title>
  <authors>
   <author>Knuth, D.E.</author>
  </authors>
  <source type="article" complete="Reading, MA; 1973">
   <journal>Reading, MA</journal>
   <year>1973</year>
  </source>
  <citations count="4194">
   <citedby>16016193676806417363</citedby>
   <citedby>12292274492368157079</citedby>
   <citedby>17157916999812790334</citedby>
  </citations>
 </publication>
</publications>
```

4.3 Třída BibtexParser

Vstupem třídě *BibtexParser* je BufferedReader obsahující kód stránky s informacemi o záznamu pro BibTeX. Metoda *parse* postupně projde všechny řádky BufferedReaderu a vezme si z nich požadované informace. Z prvního řádku se vyextrahuje informace o typu nalezeného záznamu (book/article/inproceedings). Všechny další pro nás zajímavé řádky jsou ve tvaru „*název_položky={obsah},*“ (viz Obrázek 4-2). Tyto položky BibtexParser zpracuje a uloží do kolekce *HashMap<String, String>*, přičemž jako klíč použije *název_položky* a jako hodnotu *obsah*. Po projití všech řádků vrací volajícímu (třída *HtmlParser*) LinkedHashMapu, naplněnou informacemi o daném záznamu. Díky použití této kolekce je ve výsledném XML souboru zaručeno stejné pořadí jednotlivých položek jako na webu.

```
@inproceedings{knuth2007computer,
  title={Computer programming as an art},
  author={Knuth, D.E.},
  booktitle={ACM Turing award lectures},
  pages={1974},
  year={2007},
  organization={ACM}
}
```

Obrázek 4-2: Stránka s informacemi pro BibTeX

4.4 Třída Farmar

Třída *Farmar* očekává na vstupu několik proměnných, ale za zmínku stojí především tři z nich. První je URL adresa dotazu, sestavená třídou *Okno*, druhou cesta k souboru pro uložení výsledného XML a třetí odkaz na nadřazený objekt, tedy třídu *Okno*. Tento odkaz je důležitý pro řízení grafického prostředí aplikace v průběhu vyhledávání. Pomocí něj jsou na textový výstup aplikace posílány zprávy o činnosti programu a upravována hodnota progressbaru tak, aby co nejvíce odpovídala skutečnému počtu již nalezených položek a zároveň dávala uživateli najevo, že „se něco děje“. *Farmar* se stará o rozdělování práce mezi ostatní třídy programu. Na začátku třídě *HttpClient* předá získanou URL adresu s dotazem a metodou *read* si vyžádá HTML kód příslušné stránky, jejž poté předá metodě *parse* třídy *HtmlParser*. Ta se stará o parsování HTML kódu webové stránky, čímž získává informace o vyhledaných záznamech. Tyto si *Farmar* od *HtmlParser*u vyžádá metodou *getZaznamy*. K jejich uložení slouží třída *Zaznam*, navržená tak, aby byla schopna rychle a přehledně uchovávat potřebné informace. Ta je použita pro vytvoření kolekce *LinkedHashMap<String, Zaznam>*, pojmenované *zaznamy*, která nalezené záznamy ukládá jako dvojici klíč-hodnota. Jako klíč je zde použito unikátní ID záznamu, převzaté ze služby Google Scholar (pokud ID pro nějaký záznam ve Scholaru neexistuje, je mu programem vygenerováno) a jako hodnota jsou použity všechny informace předané *HtmlParser*em. V tomto bodě má *Farmar* k dispozici výsledky hledání s informacemi o jejich názvu, autorech, roce vydání, nakladatelství, počtu stran, odkazu na vyhledání citujících článků, případně dalšími dostupnými informacemi.

Dále je třeba ke každému z těchto záznamů najít citující materiály a informace o nich. K tomu poslouží námi dříve získaný odkaz, uložený ve struktuře *Zaznam* a pojmenovaný jako *citeLink*. Skrývá se pod ním obyčejná URL adresa, která je opět předána třídě *HttpClient* a výše popsaný sled událostí se opakuje pro každý nalezený záznam. Mění se tedy jen URL adresa a

získávané výsledky *HtmlParser*u, které teď představují citující články. Nalezené záznamy jsou ukládány do další *LinkedHashMap*y, pojmenované *citace*. Identifikátory citujících záznamů jsou poté, jako pole *String*ů, přidruženy k příslušnému citovanému záznamu.

Pro maximalizaci času, po který budou servery Googlu akceptovat přístupy programu než je vyhodnotí jako robotické, je mezi hledání citujících záznamů každého citovaného vložena časová prodleva o délce specifikované veřejnou statickou proměnnou *PRODLEVA* ze třídy *Hlavni*.

Poslední úkol třídy *Farmar* je, aby po shromáždění všech citovaných i citujících záznamů, předal tyto dvě *LinkedHashMapy* spolu s cestou ke XML souboru třídě *StAX*, která se postará o vytvoření správného XML souboru a zápis dat v příslušném formátu.

Na závěr je spočítáno, kolik záznamů bylo nalezeno v jednotlivých generacích a vypsána informace o času začátku a konce hledání a celkové době běhu programu.

4.5 Třída Hlavni

Třída *Hlavni* pouze vytvoří nový objekt třídy *Okno*, čímž zapříčiní zobrazení grafického rozhraní programu.

Kromě toho tato třída obsahuje několik veřejných statických proměnných, které jsou provázány s velkou částí programu a proto by zde měly být popsány. Všechny jsou datového typu integer. Jsou to:

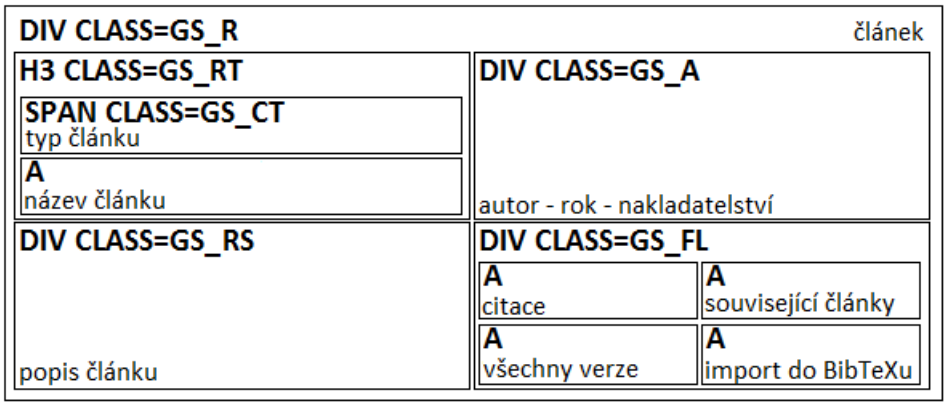
- PRODLEVA – doba v milisekundách, na kterou se program uspí před každým přístupem na servery Scholaru. Tím se aplikace snaží maximálně prodloužit dobu, než budou její přístupy na servery Googlu vyhodnoceny jako robotické a její dotazy budou ze strany serverů blokovány. Implicitně je nastavena na 1000 milisekund, ale pomocí grafického rozhraní aplikace je možné ji změnit.
- ID – udržuje informaci o posledním vygenerovaném identifikátoru pro záznamy, pro které není možné ze Scholaru stáhnout Goolem vygenerovaný identifikátor. Tento problém se vyskytuje u těch publikací, které nejsou citovány žádnými dalšími články. Při každém dalším takovém nalezeném záznamu je proměnná ID zvětšena o jedničku a její nová hodnota je použita jako identifikátor právě nalezeného záznamu. Implicitní hodnota je nastavena na nulu, ale první použitý identifikátor bude jedna.
- GENERACE – reprezentuje číslo aktuálně zpracovávané generace záznamů. Implicitní hodnota je nastavena na nulu, další hodnoty nejsou získávány inkrementací této proměnné, ale přiřazením hodnoty z jiné proměnné, se kterou se pracuje v hlavním cyklu třídy *Farmar*.
- PORADI – její hodnota se rovná celkovému počtu nalezených záznamů od začátku běhu programu. Hodnoty této proměnné se mohou v průběhu programu skokově měnit i o desítky záznamů, což je dáno kontrolou duplicit, popsanou v jiné části dokumentu. Tato proměnná tudíž není určena pro zjišťování přesného počtu nalezených záznamů. Implicitní hodnota je nastavena na nulu.

- AKTUALNIPOCET – udržuje informaci o počtu nalezených potomků (citujících článků) pro jeden, aktuálně zpracovávaný, záznam. Z její hodnoty je zjišťováno, kolik citujících publikací již bylo pro daný záznam nalezeno a porovnáním s hodnotou proměnné, určující uživatelem požadovaný počet záznamů, se rozhoduje, zdali se má v hledání pokračovat. Implicitní hodnota je nastavena na nulu.
- USPATPO – určuje, po kolika přístupech na servery Googlu se program uspí. V programu je hodnota této proměnné porovnávána s hodnotou proměnné *POCET*. Implicitní hodnota je nastavena na 150 přístupů, ale je možné ji pomocí grafického rozhraní programu změnit.
- USPATNA – je úzce spjata s proměnnou *USPATPO* a určuje dobu v milisekundách, na kterou se program uspí po dosažení hodnoty *USPATPO*. Implicitní hodnota je nastavena na 3 600 000 milisekund, což odpovídá jedné hodině. I tuto proměnnou je možné změnit pomocí grafického rozhraní aplikace.
- POCET – udržuje informaci o počtu přístupů na servery Scholaru, uskutečněných od spuštění programu. Po dosažení libovolného celého násobku hodnoty, uložené v proměnné *USPATPO*, se aplikace uspí na dobu stanovenou v proměnné *USPATNA*. Toto je další ochrana aplikace proti zablokování přístupu ze strany Googlu. Implicitní hodnota je nastavena na nulu a inkrementována při každém přístupu na servery Scholaru.

4.6 Třída HtmlParser

Klíčovým prvkem třídy *HtmlParser*, dědící od třídy *HTMLEditorKit.ParserCallback*, je metoda *parse*. Ta na vstupu očekává kód HTML stránky, který bude parsovat. Třída implementuje několik povinných metod pro zachytávání různě strukturovaných HTML tagů, jejich atributů, komentářů, chyb apod.

Na obrázku 4-3 je ukázán zdrojový kód pro jeden záznam na stránce výsledků. Tagy *DIV* a *SPAN* jsou určeny pro ohraničení delšího textu. *H3* je nadpis třetí úrovně a tag *A* definuje odkaz na jinou stránku.



Obrázek 4-3: Struktura článku v HTML kódu

4.6.1 Metoda handleText

Metoda *handleText* předává libovolný text v kódu stránky, který není tagem ani atributem. Jejími parametry jsou nalezený text jako pole charů a pozice začátku textu v souboru.

V této metodě je zjišťován celkový počet citací daného záznamu podle služby Google Scholar. Také zde jsou zachytávány názvy nalezených článků. Ty jsou totiž na stránce vždy obaleny tagy *H3*.

4.6.2 Metoda handleComment

HandleComment je metoda volaná při zachycení HTML komentáře. Jejími atributy jsou opět pole charů, reprezentující nalezený komentář, a pozice začátku komentáře ve zdrojovém kódu. Tato metoda není v programu používána.

4.6.3 Metoda handleEndOfLineString

V programu tato metoda není použita, ale je součástí třídy *HTMLEditorKit.ParserCallback* a je volána při dosažení konce řádku v HTML kódu.

4.6.4 Metoda handleStartTag

Při zachycení počátečního párového HTML tagu je volána metoda handleStartTag, jejímiž argumenty jsou tag (datového typu *HTML.Tag*), jeho atributy (datový typ *MutableAttributeSet*) a pozice začátku tagu v HTML kódu.

Převážně z předávaných atributů se v této metodě získávají informace, jako jsou odkazy na vyhledávání citujících článků a informací pro Bibtex. Také se zjišťuje, zda jsme v souboru na správné pozici pro zachytávání názvu, roku publikování nebo bližšího popisu článku.

4.6.5 Metoda handleEndTag

Tato metoda je volána při zachycení koncového párového tagu. Jejími argumenty jsou tag (datového typu *HTML.Tag*) a pozice začátku tagu v HTML kódu.

Každý nalezený záznam je „zabalen“ do několika bloků tagu *DIV*. Při nalezení koncového páru tohoto tagu dojde ke snížení příslušného čítače. Dosáhne-li čítač nuly, dostali jsme se ve čtení souboru na konec daného záznamu. V tom případě dojde k volání metody *pridejZaznam*, která všechny sebrané informace dále zpracuje.

4.6.6 Metoda handleSimpleTag

Na metodu *handleSimpleTag* přichází řada ve chvíli, kdy je v kódu nalezen „jednoduchý“, nepárový tag. Jejími argumenty jsou tag (datového typu *HTML.Tag*), jeho atributy (datový typ *MutableAttributeSet*) a pozice začátku tagu ve zdrojovém kódu. Tato metoda není v programu využívána.

4.6.7 Metoda handleError

HandleError je metoda volaná při zachycení chyby v HTML kódu. Jako parametry má řetězec nalezené chyby a její pozici v HTML kódu. Tato metoda není v programu použita.

4.6.8 Metoda pridejZaznam

Tato metoda není jako jediná z této třídy překrytá. I přesto je ale velice důležitá, protože zpracovává získané informace o záznamu. Na začátku je vytvořen nový objekt třídy *HttpClient* a

BibtexParser pro získání informací, uložených ve tvaru dat pro BibTeX. *HttpClient* opět stáhne celou stránku a její kód předá třídě *BibtexParser*, která se postará o jeho parsování. Její přesná činnost je popsána výše. Návratem třídy *BibtexParser* jsou všechny dostupné informace o článku, jež jsou dále zpracovány. Nejvíce nás zajímá kompletní seznam autorů. Jen málokdy jsou u článku uvedeny všechny informace, často jsou k dispozici jen dva tři údaje. Jde například o název knihy/článku, název časopisu/konference, číslo části (*volume*), pořadové číslo (*number*), čísla stran, na nichž se citace vyskytují, organizace, nakladatelství a rok vydání.

Tato metoda je v podstatě nejčastěji volanou částí celé aplikace, takže se krásně hodí pro kontrolu počtů přístupů na servery Scholaru. Na závěr své činnosti metoda vždy porovná stavy proměnných *POCET* a *USPATPO* a pokud je to nutné, činnost programu se na tomto místě pozastaví na čas určený proměnnou *USPATNA*. Přesný význam těchto konstant je popsán v kapitole „*Třída Hlavni*“.

4.7 Třída HttpClient

Tato třída slouží k přístupu ke konkrétní webové stránce a získání jejího HTML kódu. Vzhledem k ochraně serverů Googlu před robotickým prohledáváním jejich obsahu je nutné implementovat určitá opatření pro obejití jejich ochran. Každému dotazu musí být nastaveny následující vlastnosti:

- ***User-Agent***

Nastaven na „*Mozilla/5.0 (compatible; MSIE 9.0; Windows NT 6.1; Trident/5.0)*“, program se tedy hlásí jako prohlížeč IE9 v kompatibilním módu.

- ***Content-type***

Obsah „*text/html;charset=UTF-8*“ říká, že jde o HTML kód a kódování UTF-8.

- ***Cookie***

Tato položka je nejdůležitější a zároveň nejkomplikovanější. Pomocí cookies se nastavuje zobrazování odkazu na import dat pro BibTeX (přepínač CF=4). Další důležitou vlastností je ID, které slouží k identifikaci pro servery Googlu. V případě podezření na robotické prohledávání dojde k přesměrování na Captcha kód. Po jeho správném zadání se obsah cookies opět změní. Například je přidána IP adresa počítače, ze kterého se ke službě přistupuje.

Všechny tyto vlastnosti se objektu třídy *HttpURLConnection* nastavují pomocí metody *setRequestProperty* jako dvojice stringů *klíč* a *hodnota*.

Ukázka možných cookies:

- „*GSP=ID=82af4aaa552af366:CF=4*“ – v programu aktuálně používaná a zároveň nejjednodušší cookies, obsahující jen povinný řetězec ID, složený ze šestnácti znaků v šestnáctkové soustavě a přepínače CF=4, který zajišťuje zobrazení odkazu na import dat

pro BibTeX. Takto jednoduchá cookies prohlížeč sám nikdy nedostane, ale pro chod aplikace jsou zcela postačující.

- *„PREF=ID=82af4aaa552af366:LD=en:NR=100:CR=2:TM=1320506269:LM=1320506767:S=ecx-XQ9JnNN5-_cL;GSP=ID=82af4aaa552af366:IN=ee4d35360801e2e5+31689a4f8f07755d:CF=4*“ – reálná cookies, která dostane prohlížeč. Obsahují spoustu dat, která nejsou pro chod programu nijak důležitá, takže mohou být vynechána.

4.8 Třída Okno

K vytvoření grafického rozhraní slouží třída *Okno*. Rozhraní je tvořeno dvěma komponentami typu *JPanel* a jednou komponentou typu *ProgressBar*, umístěnou ve spodní části programu. Horní část slouží k nastavení parametrů vyhledávání, časových intervalů pro uspávání programu a lokace výsledného XML souboru. Prostřední část, sloužící jako konzole, interaguje s uživatelem formou výpisu činnosti.

Základní layout použitý pro správné rozložení zmíněných dvou *JPanel*ů a *PrograssBar*u je *BorderLayout* (použity jsou pozice *NORTH*, *CENTER* a *SOUTH*). Pro rozložení jednotlivých komponent (*JLabel* a *JTextField*) na horním panelu byl použit *GridLayout*, zvolený pro jeho jednoduché a přitom efektivní použití.

Po zadání kritérií hledání dojde nejprve ke kontrole vstupů a poté k převedení vstupních řetězců na tvar použitelný v URL adrese (odstranění diakritiky a jiných nebezpečných znaků, nahrazení mezer). Následně se sestaví URL adresa, která je spolu s cestou pro uložení výsledného XML souboru předána třídě *Farmar*. Pokud je při kontrole některý vstup shledán chybným, je nahrazen výchozí hodnotou, uživatel o daném faktu informován textovým upozorněním a program spuštěn.

4.8.1 Skládání URL adresy

URL adresa obsahuje mnoho parametrů, podle kterých lze ve Službě Scholar vyhledávat, a má svůj specifický tvar, který je nutné dodržet. Všechny parametry jsou od sebe navzájem odděleny znakem „*&*“ (ampersand). Za názvem parametru (ten je klíčový a nesmí být změněn, protože ho přebírají skripty Scholaru a dále s ním pracují) je vždy znak „*=*“ (rovnítko) následovaný hodnotou parametru. URL adresa vždy začíná řetězcem „*http://scholar.google.com/scholar?*“, poté jsou vyjmenovány všechny parametry se svými hodnotami. Pokud hodnota parametru nebyla v programu zadána (to znamená, že daný parametr nemá na výsledky hledání žádný vliv), je sice v URL adrese uveden jeho název, ale nemá žádnou hodnotu. Následuje výčet jednotlivých parametrů i s popisem:

- *as_q* – každý vyhledaný záznam musí obsahovat všechna slova uvedená v tomto parametru, ale nemusí být přesně v uvedeném pořadí.
- *as_epq* – každý vyhledaný záznam musí obsahovat všechna slova uvedená v tomto parametru jako přesnou frázi (ekvivalentem je ruční zapsání hledaného textu do vyhledavače v uvozovkách, například: „The art of computer programming“).
- *as_oq* – každý vyhledaný záznam musí obsahovat alespoň jedno slovo uvedené v tomto parametru.

- *as_eq* – žádný z vyhledaných záznamů nesmí obsahovat ani jedno slovo uvedené v tomto parametru.
- *as_sauthors* – tento parametr udává jméno autora, jehož články se budou vyhledávat
- *as_publication* – tento parametr udává název časopisu nebo sborníku, mezi jehož články se má hledat.
- *as_ylo* – určuje spodní hranici období vydání hledaného článku (rok).
- *as_yhi* – určuje horní hranici období vydání hledaného článku (rok).
- *as_occt* – určuje, zda budou klíčová slova hledána jen v názvu článku nebo kdekoli v jeho těle (tělo zahrnuje stručný popis publikace).
- *num* – určuje počet výsledků zobrazených na jedné straně. Lze nastavit hodnoty z intervalu 1-100. Googlem doporučovaná hodnota je deset, ale pro naše potřeby se více hodí horní hodnota intervalu.
- *start* – určuje číslo vyhledaného záznamu, který má být na stránce zobrazen jako první. Všechny předchozí záznamy se přeskočí. Tím umožňuje přecházet mezi jednotlivými stránkami s výsledky hledání. Je důležité, aby tento parametr byl vždy na konci URL adresy, protože je v programu často měněn a jeho hodnota je nahrazována podle potřeby procházení mezi jednotlivými stránkami. Neuvedení tohoto parametru, nebo jeho uvedení na jiném než posledním místě URL adresy, by mohlo vést k nefunkčnosti programu.

Příklad URL adresy vygenerované programem na dotaz „the *art of computer programming*“, kdy požadujeme, aby vyhledané záznamy obsahovaly všechna slova hledaného řetězce („*as_q=the+art+of+computer+programming*“) a aby bylo vyhledáváno i v popisku publikací („*as_occt=any*“). Požadujeme sto záznamů na stránku („*num=100*“) a chceme zobrazit první stranu výsledků („*start=0*“):

„http://scholar.google.com/scholar?as_q=the+art+of+computer+programming&as_epq=&as_oq=&as_eq=&as_sauthors=&as_publication=&as_ylo=&as_yhi=&as_occt=any&num=100&start=0“

4.9 Třída StAX

StAX je jeden z několika standardů pro vytváření XML souborů. Jak název třídy napovídá, byl použit v programu pro uložení získaných dat. Práce s XML soubory není v Javě složitá. Stačí nám objekt typu *XMLStreamWriter* a jeho metody. Mezi ně patří:

- *writeStartDocument* pro zapsání hlavičky XML souboru, obsahující především kódování a verzi,
- *writeStartElement* pro zápis počátečního párového tagu,
- *writeAttribute* pro zápis atributů tagu,
- *writeCharacters* pro zápis textu mezi počáteční a koncový tag,
- *writeEndElement* pro zápis koncového párového tagu,
- *writeEndDocument* pro zapsání tagu, uzavírajícího celý XML soubor.

Tato třída prochází kolekce sdružující všechny nalezené informace o všech publikacích a vytváří z nich XML dokument dané struktury (viz kapitola 4.2).

4.10 Třída Zaznam

Jedná se pouze o strukturu pro uložení informací o záznamech. Je používána v kolekci *LinkedHashMap*. Skládá se z těchto vlastností:

- *nazev* – uchovává název knihy/článku/abstraktu,
- *autori* – pole pro uložení všech autorů,
- *citeLink* – URL adresa na vyhledání citujících článků,
- *bibtexInfo* – všechny ostatní informace o článku jako nakladatelství, rok vydání, počet stran, číslo, název časopisu nebo konference,
- *citace* – pole identifikátorů citujících záznamů,
- *generace* – číslo generace, do níž daný záznam patří,
- *pocetCitaci* – celkový počet citujících článků daného záznamu podle Scholaru.

5 Závěr

Bakalářská práce byla realizována s cílem zjistit, zda je možné vytvořit počítačový program, umožňující robotické stahování dat ze služby Google Scholar a jejich ukládání ve formátu XML. V první části je proto popsána služba samotná a rozebrány její přednosti, včetně možných přístupů ke zpracování dat. Během práce bylo zjištěno, že pro přístup ke Scholaru neexistuje žádné oficiální API a na žádných oficiálních místech se zatím nemluví ani o možnosti jeho budoucího vydání. Z tohoto důvodu bylo nutné navrhnout vlastní způsob získávání příslušných dat. Jako nejvhodnějším a v podstatě i jediným možným řešením (s ohledem na výkon aplikace a složitost realizace) se ukázalo být použití HTML parseru.

Nevýhodou aplikace je fakt, že je závislá na struktuře HTML kódu, který zpracovává. Z toho vyplývá, že změní-li se zásadnějším způsobem kód stránek, generovaných Scholarem, bude s velkou pravděpodobností nutné upravit i zdrojový kód programu. Toto je ale obecný a těžko řešitelný problém podobných specializačních aplikací. Mělo by vždy jít pouze o drobné změny v rámci části kódu, starající se o parsování HTML. Ke změně struktury HTML stránek došlo již během vývoje aplikace. Tehdy šlo o změnu tagu, na jehož základě je získávána informace o názvu publikace.

Z důvodu zákazu robotického prohledávání obsahu serverů služby Google Scholar bylo nutné implementovat patřičná opatření, chránící aplikaci před zakázáním přístupu ze strany Googlu. Proto jsou mezi jednotlivé požadavky, posílané serverům, vkládány časové mezery. Ty jsou ve výchozím stavu nastaveny na 1 000 milisekund. Dalším nezbytným opatřením je pozastavit činnost programu na delší časový úsek (v řádu hodin) po určitém soustavném počtu přístupů k serverům. Experimentálně bylo zjištěno, že ideální je pozastavit program na jednu hodinu po každých 150 uskutečněných dotazech. Je ovšem možné, že po větším počtu přístupů z jedné IP adresy, bude nutné limity zpřísnit. Při běhu programu s výchozím nastavením limitů došlo po zhruba třiceti hodinách k zablokování přístupu k serverům. Z důvodu sdílené veřejné IP adresy však nebylo možné zjistit, jestli z dané adresy k serverům Googlu přistupovala opravdu pouze aplikace nebo i jiní klienti.

Z dosažených výsledků lze říci, že projekt byl úspěšný ve všech bodech zadání. Praktickým vylepšením programu by mohlo být například rozšíření o schopnost komunikovat s okolními počítači v síti, kde by jeden fungoval pouze jako „úschovna“ dat a dělil by práci mezi ostatní. Disponoval-li by každý stroj vlastní veřejnou IP adresou, dokázaly by za krátký časový úsek nashromáždit obrovské množství informací.

Program by měl být v budoucnu použit jako součást projektu s cílem vytvářet citační sítě z bibliografických dat. Za tímto účelem byl také navržen použitý formát XML souboru, umožňující snadnou orientaci mezi navzájem se citujícími publikacemi.

Literatura

[Osl12] O službě Google Scholar, nahlíženo 21. 1. 2012,
http://scholar.google.cz/intl/cs/scholar/about.html

[Jav12] Java SE Downloads, nahlíženo 21. 1. 2012,
http://www.oracle.com/technetwork/java/javase/downloads/index.html

[Tip12] Google Scholar, Tipy pro rozšířené vyhledávání služby Scholar, nahlíženo 26. 4. 2012,
http://scholar.google.cz/intl/cs/scholar/refinesearch.html

[Zák12] Google Scholar, Základní informace o vyhledávání, nahlíženo 26. 4. 2012,
http://support.google.com/websearch/bin/answer.py?hl=cs&answer=134479

[Dal12] Google Scholar, Další informace o vyhledávání, nahlíženo 26. 4. 2012,
http://support.google.com/websearch/bin/answer.py?hl=cs&answer=136861#exceptions

[Náp12] Google Scholar, Nápověda služby Google Scholar, nahlíženo 29. 4. 2012,
http://scholar.google.cz/intl/cs/scholar/help.html

[Nás12] ÚVT MU, Nástroje Google. 2. Google Scholar, nahlíženo 29. 4. 2012,
http://www.ics.muni.cz/bulletin/articles/602.html

[Kos99] Kosek Jiří, PHP – tvorba interaktivních internetových aplikací, Grada, 1999,
ISBN 80-7169-373-1

[The12] W3C, The HTTP Protocol As Implemented In W3, nahlíženo 30. 4. 2012,
http://www.w3.org/Protocols/HTTP/AsImplemented.html

[Hyp19] RFC 1945, Hypertext Transfer Protocol – HTTP/1.0, nahlíženo 30. 4. 2012,
http://tools.ietf.org/html/rfc1945

[Hyp26] RFC 2616, Hypertext Transfer Protocol – HTTP/1.1, nahlíženo 30. 4. 2012,
http://tools.ietf.org/html/rfc2616

[Sta12] W3C, HTTP/1.1: Status Code Definitions, nahlíženo 1. 5. 2012,
http://www.w3.org/Protocols/rfc2616/rfc2616-sec10.html

Přílohy

A Uživatelská dokumentace

Ovládání programu je velice jednoduché. Po spuštění se zobrazí úvodní a zároveň jediné okno aplikace, které se skládá ze dvou částí. Horní část slouží k nastavení parametrů pro hledání, spodní část k vypisování informací o výsledcích činnosti programu.

Program umožňuje vyhledávat záznamy obsahující všechna zadaná slova, přesnou frázi, alespoň jedno ze zadaných slov, případně záznamy, v nichž se zadaná slova nevyskytují. Dovoluje také vyhledávání článků podle jména autora, názvu nakladatelství, roku vydání a hledání lze omezit na titulek nebo celý článek. Poslední dvě položky určují počet výsledků na stránku a umístění výsledného XML souboru. Po stisknutí tlačítka *Hledat* se spustí vyhledávání podle zadaných parametrů. Po jeho dokončení jsou do zadaného souboru uloženy zpracované výsledky. Nakonec aplikace informuje výpisem do textového pole ve spodní části o sestavené URL adrese a času začátku, konce a celkové době prohledávání.

A.1 Spuštění

Program lze spustit z příkazové řádky Windows příkazem *java –jar scholar.jar*. Pokud je v systému nastavena asociace *.jar* souborů, stačí poklepat na soubor *scholar.jar*.

Pro správný běh aplikace je potřeba mít na počítači nainstalováno virtuální prostředí Javy JVM (Java Virtual Machine). To lze bezplatně stáhnout ze stránek výrobce [Jav12].

A.2 Grafické rozhraní

Grafické rozhraní je zobrazeno na obrázku A-1 a tvoří ho jediné okno, složené z několika částí. Čísla 1-4 označují komponenty pro zadání klíčových slov hledání (slova se oddělují pouze mezerou, stejně jako tomu je u klasického vyhledávání na stránkách Googlu). Políčka s čísly 5-9 jsou určeny pro omezující podmínky výsledků hledání, vlastnosti 10-18 určují chování programu a tlačítka 19-21 slouží k ovládání aplikace. Nyní jejich bližší popis:

(1) *Hledat všechna slova* – každý nalezený záznam musí obsahovat všechna slova uvedená v tomto poli, ale nemusí být v přesně uvedeném pořadí a mohou být skloňována.
(2) *Hledat přesnou frázi* – každý nalezený záznam musí obsahovat slova uvedená v tomto poli jako přesnou frázi bez skloňování a jiných odchylek.
(3) *Hledat alespoň jedno slovo* – každý nalezený záznam musí obsahovat alespoň jedno slovo ze zde uvedených.
(4) *Hledat beze slov* – žádný nalezený záznam nesmí obsahovat ani jedno slovo ze zde uvedených.
(5) *Kde hledat* – určuje, zda se budou klíčová slova vyhledávat jen titulku záznamu nebo v celém těle včetně popisku.
(6) *Hledat podle autora* – jméno autora, jehož články chceme vyhledávat. Jméno se doporučuje zadávat ve tvaru „*J Příjmení*“, například „*A Einstein*“.
(7) *Hledat články publikované v* – název časopisu, v němž byl námi požadovaný článek vydán nebo konference, na níž byl publikován.

(8) *Hledat články publikované od roku* – počáteční rok intervalu, v němž byl článek publikován.
(9) *Hledat články publikované do roku* – koncový rok intervalu, v němž byl článek publikován. Chceme-li vyhledávat články pouze z jednoho roku, bude tato hodnota totožná s hodnotou políčka (8).
(10) *Počet výsledků na stránku* – udává, kolik záznamů má být obsaženo na jedné straně výsledků hledání. Povolené hodnoty jsou 1-100. Čím více výsledků je umístěno na jednu stranu, tím méně přístupů na servery Googlu program provede, protože nebude muset stále žádat o další výsledky hledání.
(11) *Umístění výsledného XML souboru* – udává cestu k souboru, do nějž mají být uloženy výsledky transformace dat. Výchozí hodnota je nastavena na soubor „*vystup.xml*“, umístěný ve složce, z níž je program spuštěn.
(12) *Počet výsledků na generaci a záznam* – udává maximální počet citujících publikací (potomků), hledaných pro každou již nalezenou položku. Pokud některá publikace na Scholaru tolik citujících záznamů nemá, vezmou se všechny dostupné. Zároveň určuje počet výsledků v nulté generaci.
(13) *Do generace číslo* – číslo generace, která má být prohledávána jako poslední (včetně). Pozor na fakt, že generace nejsou číslovány od jedné, ale od nuly. To znamená, že pokud zadáme jako číslo generace jedničku, ve skutečnosti budou procházeny generace dvě. Naopak, chceme-li najít jen jednu generaci publikací (nezajímají nás citace), uvedeme do tohoto políčka nulu.
(14) *Začít na straně číslo (neplatí pro citace)* – určuje číslo strany s výsledky, od které se bude prohledávat (platí jen pro nultou generaci). Souvisí s možností (10). Pokud například zvolíme deset výsledků na stránku a začínat budeme na třetí straně, program přeskočí prvních třicet záznamů a začne zpracovávat záznamy až od čísla 31 dál. To se hodí například v případě, že máme prohledaných prvních 30 záznamů a zjistili jsme, že jich potřebujeme ještě dalších 20. Takto můžeme těch 30 již zpracovaných záznamů přeskočit a rovnou hledat nové. To šetří dotazy na Google a tím pádem i čas, potřebný ke stažení požadovaných informací.
(15) *Prodleva mezi záznamy* – čas v milisekundách, na který se program uspí po každém dotazu na servery Googlu. Doporučená hodnota je 1000 milisekund. Kratší doba může vést k rychlejšímu zakázání přístupu na servery. Nastavení delšího intervalu se neukázalo jako přínosné.
(16) *Počet řádků textu* – počet řádků, zobrazovaných v konzoli (22). Pokud je řádků k zobrazení více než požadovaný počet, je zpráva zepředu ořezána na nastavený počet řádků.
(17) *Uspat po záznamech* – určuje, po kolika nalezených záznamech se má program uspat. Toto je další opatření před zamezením přístupu k serverům Googlu kvůli robotickému prohledávání jejich obsahu. Doporučená hodnota je 150, vyšší je zbytečná, při nižší hodnotě riskuje uživatel brzké zabránění přístupu k serverům Googlu.
(18) *Uspat na* – čas v milisekundách, na který se program uspí po nalezení tolika záznamů, kolik je definováno v poli (17). Doporučená hodnota je 3 600 000 milisekund, což odpovídá jedné hodině. V kombinaci s použitím doporučené hodnoty u možnosti (17) nebyly zaznamenány žádné komplikace s přístupy na servery.

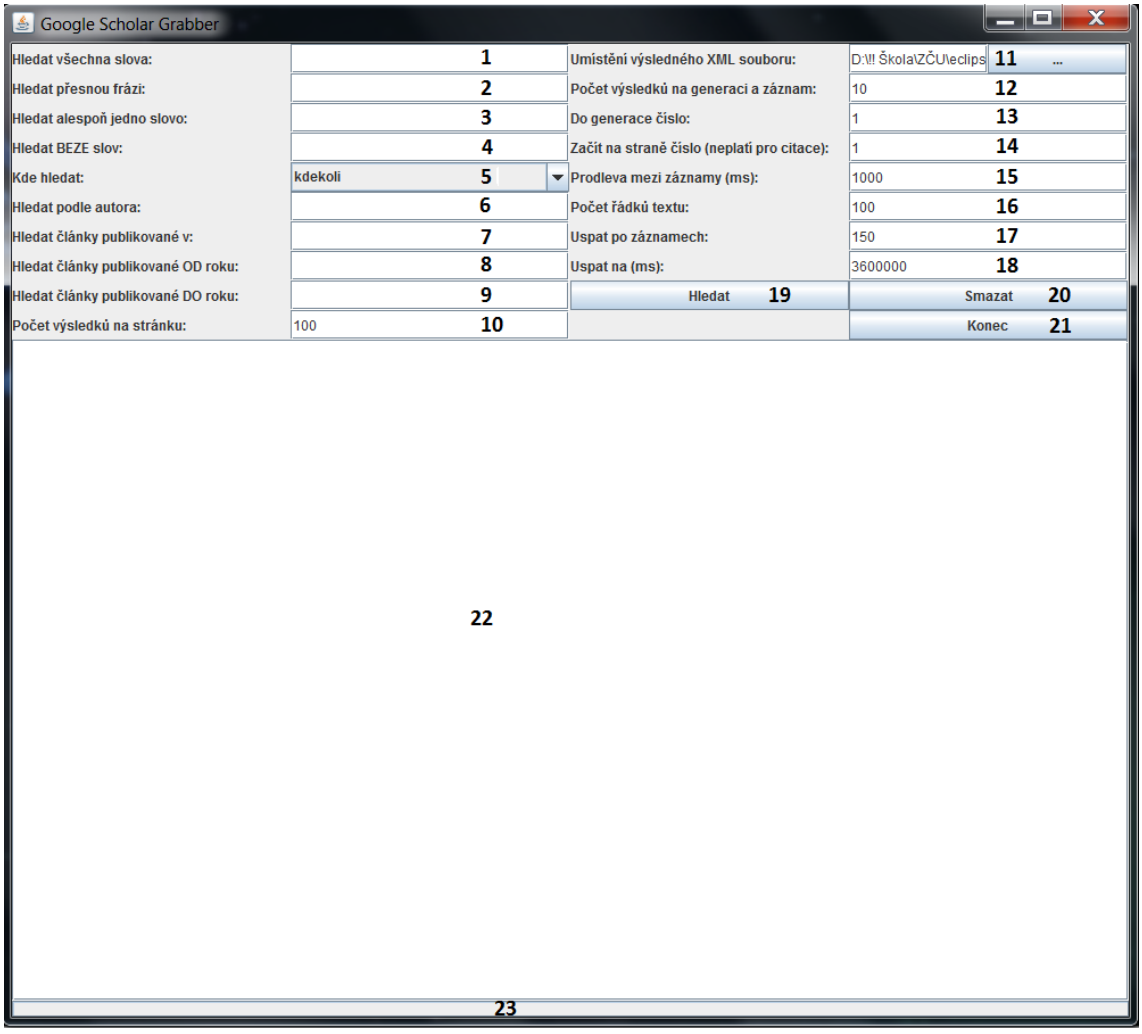
(19) *Hledat* – tlačítko pro zahájení činnosti programu. Nejprve dojde ke kontrole validity vstupních dat a poté je zahájeno vyhledávání. V průběhu vyhledávání je toto tlačítko neaktivní.

(20) *Smazat* – tlačítko pro smazání obsahu konzole (22). Lze ho použít kdykoli v průběhu programu. Veškerý obsah konzole bude nenávratně ztracen!

(21) *Konec* – tlačítko pro ukončení programu. Jeho použití je ekvivalentní s ukončením programu pomocí křížku v pravém horním rohu okna.

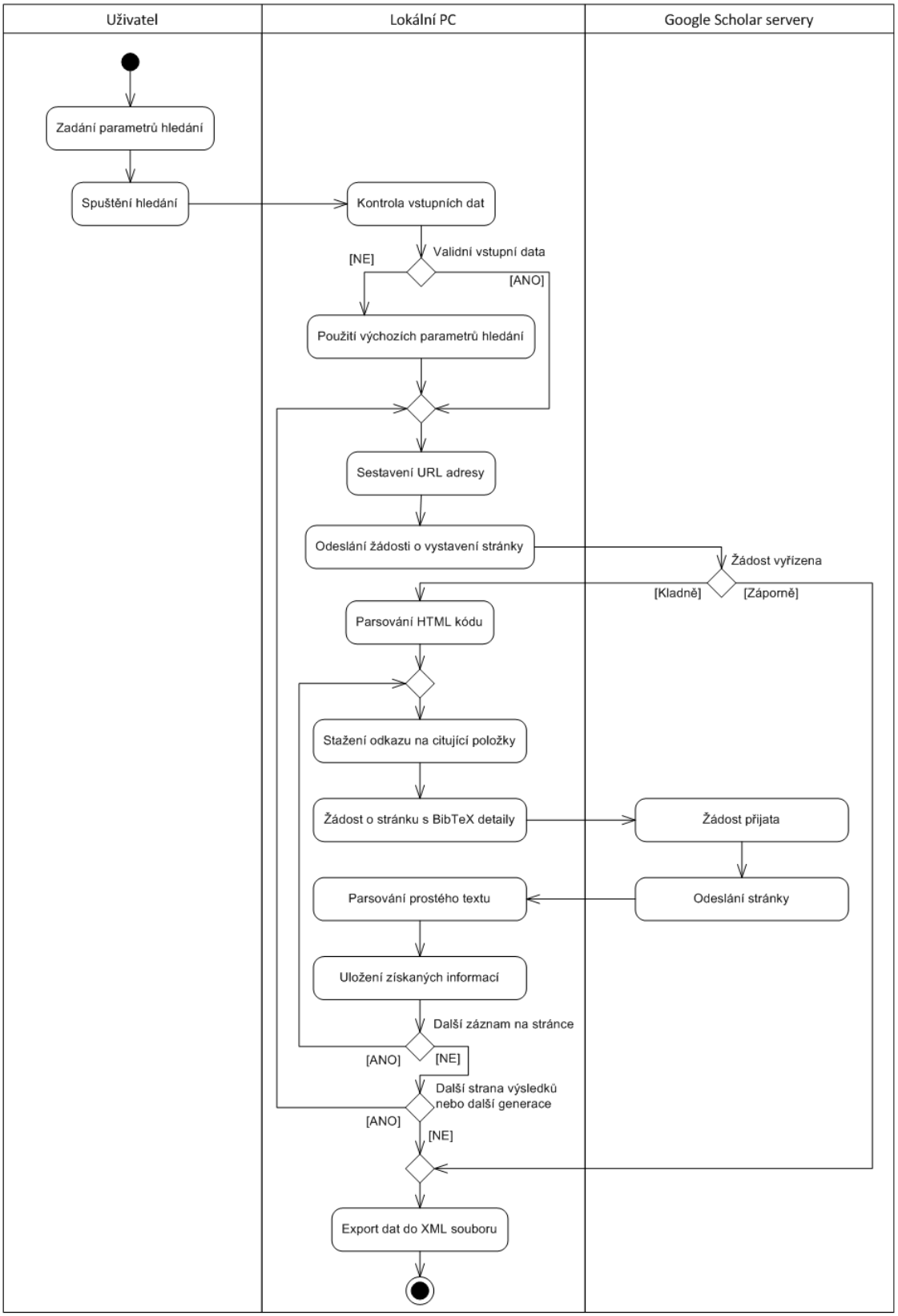
(22) *Programová konzole/stavový řádek* – prostor pro výpisy, informující o běhu programu, a další důležité informace, jako počet nalezených citací jednotlivých záznamů, volané URL adresy a souhrnné statistiky na konci vyhledávání.

(23) *ProgressBar* – graficky informuje o přibližném průběhu vyhledávání. Zobrazuje počet již nalezených záznamů. V některých chvílích neuvažuje duplicity, takže jeho hodnoty se mohou skokově měnit.



Obrázek A-1: Grafické rozhraní

B Diagram aktivit



Obrázek B-1: Diagram aktivit

C Vygenerovaný XML soubor

Kompletní XML soubor, vygenerovaný programem na dotaz „the art of computer programming“ se dvěma záznamy na generaci a prohledáváním do generace číslo 2.

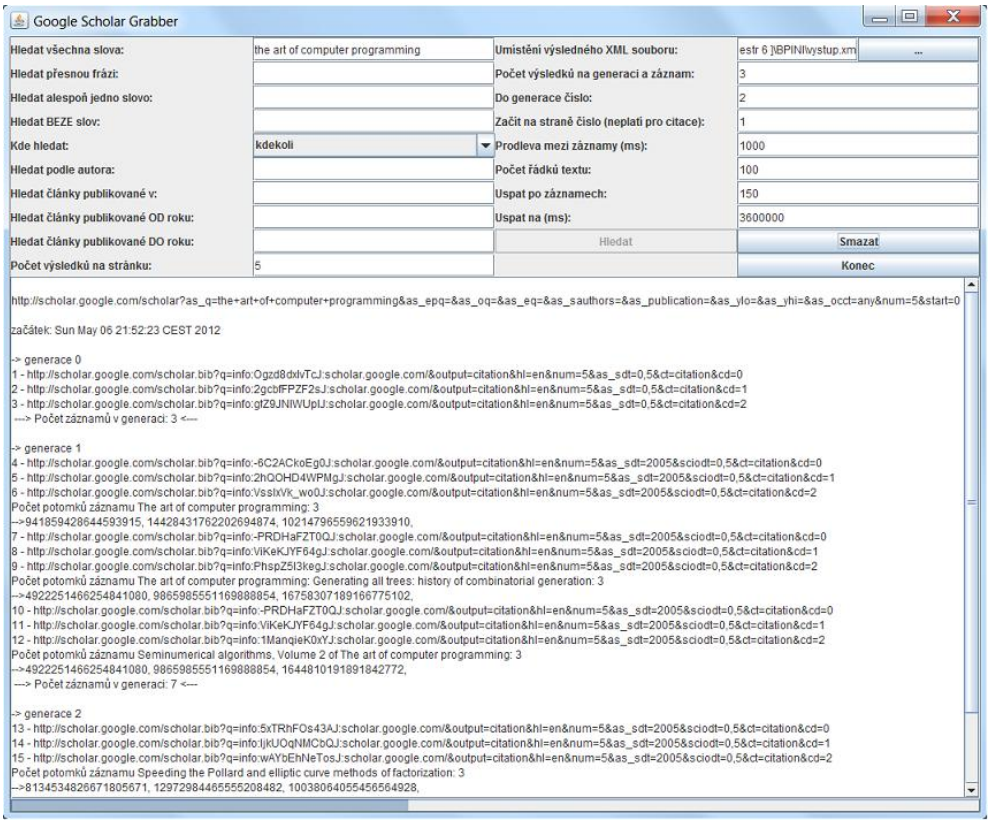
```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<publications>
 <publication id="4016478442309356602" number="1" generation="0">
  <title>The art of computer programming</title>
  <authors>
   <author>Donald, E.K.</author>
  </authors>
  <source type="article" complete="Massachusetts: Addison-Wesley; 1973">
   <journal>Massachusetts: Addison-Wesley</journal>
   <year>1973</year>
  </source>
  <citations count="77">
   <citedby>941859428644593915</citedby>
   <citedby>14428431762202694874</citedby>
  </citations>
 </publication>
 <publication id="7716875439110293466" number="2" generation="0">
  <title>The art of computer programming: Generating all trees: history of combinatorial
generation</title>
  <authors>
   <author>Knuth, D.E.</author>
  </authors>
  <source type="book" complete="vol. 4; addison-Wesley; 2006">
   <volume>4</volume>
   <year>2006</year>
   <publisher>addison-Wesley</publisher>
  </source>
  <citations count="27270">
   <citedby>4922251466254841080</citedby>
   <citedby>98659855551169888854</citedby>
  </citations>
 </publication>
 <publication id="941859428644593915" number="3" generation="1">
  <title>Speeding the Pollard and elliptic curve methods of factorization</title>
  <authors>
   <author>Montgomery, P.L.</author>
  </authors>
  <source type="article" complete="Mathematics of computation; vol. 48; no. 177; p. 243-264; 1987">
   <journal>Mathematics of computation</journal>
   <volume>48</volume>
   <number>177</number>
   <pages>243-264</pages>
   <year>1987</year>
```

```
  </source>
  <citations count="639">
   <citedby>8134534826671805671</citedby>
   <citedby>12972984465555208482</citedby>
  </citations>
 </publication>
 <publication id="14428431762202694874" number="4" generation="1">
  <title>Reorthogonalization and stable algorithms for updating the Gram-Schmidt QR
factorization</title>
  <authors>
   <author>Daniel, J.</author>
   <author>Gragg, WB</author>
   <author>Kaufman, L.</author>
   <author>Stewart, GW</author>
  </authors>
  <source type="article" complete="Math. Comp; vol. 30; no. 136; p. 772-795; 1976">
   <journal>Math. Comp</journal>
   <volume>30</volume>
   <number>136</number>
   <pages>772-795</pages>
   <year>1976</year>
  </source>
  <citations count="358">
   <citedby>11610274364157909144</citedby>
   <citedby>13416794202114718285</citedby>
  </citations>
 </publication>
 <publication id="4922251466254841080" number="5" generation="1">
  <title>Introduction to algorithms</title>
  <authors>
   <author>Cormen, T.H.</author>
  </authors>
  <source type="book" complete="The MIT press; 2001">
   <year>2001</year>
   <publisher>The MIT press</publisher>
  </source>
  <citations count="28558">
   <citedby>16016193676806417363</citedby>
   <citedby>17157916999812790334</citedby>
  </citations>
 </publication>
 <publication id="9865985551169888854" number="6" generation="1">
  <title>Numerical recipes in FORTRAN: the art of scientific computing</title>
  <authors>
   <author>Press, W.H.</author>
  </authors>
  <source type="book" complete="vol. 1; Cambridge Univ Pr; 1992">
   <volume>1</volume>
   <year>1992</year>
   <publisher>Cambridge Univ Pr</publisher>
```

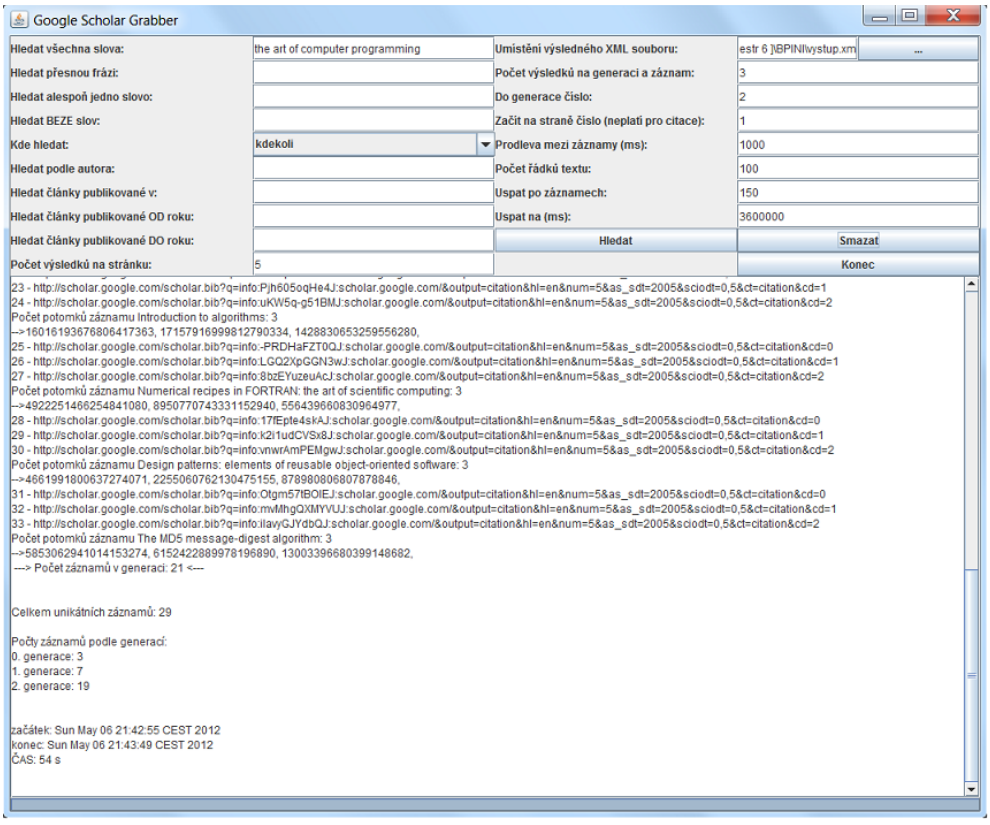
```
  </source>
  <citations count="62072">
   <citedby>4922251466254841080</citedby>
   <citedby>8950770743331152940</citedby>
  </citations>
 </publication>
 <publication id="8134534826671805671" number="7" generation="2">
  <title>Elliptic curves in cryptography</title>
  <authors>
   <author>Blake, I.F.</author>
   <author>Seroussi, G.</author>
   <author>Smart, N.P.</author>
  </authors>
  <source type="book" complete="vol. 265; Cambridge Univ Pr; 1999">
   <volume>265</volume>
   <year>1999</year>
   <publisher>Cambridge Univ Pr</publisher>
  </source>
 </publication>
 <publication id="12972984465555208482" number="8" generation="2">
  <title>Elliptic curve public key cryptosystems</title>
  <authors>
   <author>Menezes, A.J.</author>
  </authors>
  <source type="book" complete="vol. 234; Springer; 1993">
   <volume>234</volume>
   <year>1993</year>
   <publisher>Springer</publisher>
  </source>
 </publication>
 <publication id="11610274364157909144" number="9" generation="2">
  <title>Practical optimization</title>
  <authors>
   <author>Gill, P.E.</author>
   <author>Murray, W.</author>
   <author>Wright, M.H.</author>
  </authors>
  <source type="book" complete="vol. 1; no. 2; Academic press; 1981">
   <volume>1</volume>
   <number>2</number>
   <year>1981</year>
   <publisher>Academic press</publisher>
  </source>
 </publication>
 <publication id="13416794202114718285" number="10" generation="2">
  <title>The symmetric eigenvalue problem</title>
  <authors>
   <author>Parlett, B.N.</author>
  </authors>
  <source type="book" complete="vol. 20; Society for Industrial Mathematics; 1998">
```

```
    <volume>20</volume>
    <year>1998</year>
    <publisher>Society for Industrial Mathematics</publisher>
   </source>
  </publication>
  <publication id="16016193676806417363" number="11" generation="2">
   <title>Pattern Classification and Scene Analysis 2nd ed.</title>
   <authors>
    <author>Duda, R.O.</author>
    <author>Hart, P.E.</author>
    <author>Stork, D.G.</author>
   </authors>
   <source type="article" complete="1995">
    <year>1995</year>
   </source>
  </publication>
  <publication id="17157916999812790334" number="12" generation="2">
   <title>Modern information retrieval</title>
   <authors>
    <author>Baeza-Yates, R.</author>
    <author>Ribeiro-Neto, B.</author>
    <author>others</author>
   </authors>
   <source type="book" complete="vol. 82; Addison-Wesley New York; 1999">
    <volume>82</volume>
    <year>1999</year>
    <publisher>Addison-Wesley New York</publisher>
   </source>
  </publication>
  <publication id="8950770743331152940" number="13" generation="2">
   <title>The Architectural and Economic Impact of the Integration of SONET and DWDM
Platforms</title>
   <authors>
    <author>Han, S.</author>
   </authors>
   <source type="inproceedings" complete="National Fiber Optic Engineers Conference; Optical Society
of America; 2006">
    <booktitle>National Fiber Optic Engineers Conference</booktitle>
    <year>2006</year>
    <organization>Optical Society of America</organization>
   </source>
  </publication>
</publications>
```

D Náhledy programu



Obrázek D-1: Program v průběhu stahování dat



Obrázek D-2: Program po ukončení činnosti