Západočeská univerzita v Plzni
Fakulta aplikovaných věd
Katedra informatiky a výpočetní techniky

Bakalářská práce

Vytvoření kolekce obsahů informatických předmětů

Plzeň 2015 Marek Šimůnek

Prohlášení

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

V Plzni dne 7. května 2015

Marek Šimůnek

Poděkování

Tímto bych chtěl poděkovat svému vedoucímu bakalářské práce panu Prof. Ing. Karlu Ježkovi CSc., za odborné vedení, podnětné návrhy a vstřícný přístup.

Abstract

The purpose of this bachelor thesis is to analyze web pages of the czech universities with the informatics fields. The work includes to design suitable structure to save contents of subjects and choose a system for downloading pages. This system is used for automatic crawling web pages of the chosen universites and download all courses in informatics field. After that we process these downloaded courses to the form which is usable for queries.

Abstrakt

Cílem této práce je analyzovat webové stránky českých univerzit s informatickými obory. Dále navrhnout vhodnou strukturu pro uložení obsahů předmětů a zvolit systém pro stahování stránek. Ten pak následně automatizovaně projde stránky vybraných českých univerzit a stáhne všechny předměty vyučované v informatických oborech. Stažené předměty budou zpracovány do formy, která umožní dotazování.

Obsah

1 Úvod

1.1 Motiv práce

V současné době univerzity v České republice nabízejí stále více oborů a zaměření. Rostoucím trendem je pak změna oboru či dokonce univerzity v průběhu studia. Aby budoucí student mohl správně rozhodnout, měl by si prostudovat skladbu předmětů ze všech oborů, o které má zájem. Vyhledávání požadovaných informací o jednotlivých předmětech z různých fakult a univerzit není vůbec jednoduché. Aby si budoucí student udělal jakousi ucelenou představu o možnostech a obsahové náplni jednotlivých studijních oborů, potřebuje spoustu času k vyhledávání požadovaných informací. Nabízí se vytvořit systém, který nabídne hledané informace o předmětech ze všech univerzit ve stručném přehledu. Takový systém budoucím zájemcům o studium usnadní výběr univerzity a oboru, jakož i zároveň ušetří drahocenný čas.

1.2 Cíl práce

Cílem této práce je automatizovaně projít stránky vybraných českých univerzit a stáhnout všechny předměty vyučované v informatických oborech. Stažené kolekce obsahů předmětů budou zpracovány do vhodného datového formátu, ve kterém bude možné informace o předmětech vyhledávat. Zároveň bude provedením minimálních změn nastavení možné stáhnout předměty z jiných fakult nebo oborů.

Ve výsledné aplikaci bude umožněno nastavit stahování z jednotlivých univerzit. Aplikace bude poskytovat funkce pro přehledné zobrazení vytvořené kolekce. Zároveň je připravena na budoucí rozšíření.

2 Webové stránky českých univerzit

2.1 Úvod do webových stránek

Nejprve představím dva nejzákladnější jazyky, kterými se řídí struktura webové stránky.

2.1.1 HTML

HyperText Markup Language (HTML) je značkovací jazyk popisující strukturu webových stránek. Je jedním z hlavních jazyků používaných k publikování dokumentů v systému *World Wide Web*.

HTML umožňuje[1]:

- Vytváření online dokumentů s nadpisy, textem, tabulkami, seznamy, fotkami atd.
- Získávání online informace přes hypertextové odkazy
- Návrh formulářů pro provedení komunikace se vzdálenými službami, pro vyhledávání informací, pro rezervování, pro objednávání atd.
- Vložení tabulkových procesorů, videí, písniček a jiných aplikací přímo v dokumentu

HTML dokument se skládá z HTML elementů. Element začíná otevíracím tagem a končí zavíracím tagem. Mezi těmito tagy je obsah elementu.

Od roku 1997 je *HTML* jazyk standardizován mezinárodním konsorciem *W3C*, jehož snahou je sjednotit specifikace na výklad jazyka a vývoj dalších protokolů a směrnic. To má zajistit kompatibilitu na různých webových prohlížečích a platformách.

Verze *HTML5* vydaná v říjnu 2014 přináší velké změny od své předchozí verze. Mezi hlavní novinky patří přidání nové syntaktické značky (<*video*>, <*audio*>) a podpora pro aplikace (specifikací rozhraní).

2.1.2 CSS

Cascading Style Sheets je jazyk popisující prezentaci webové stránky včetně barev, rozvržení a písma. Umožňuje přizpůsobit zobrazení na různých typech zařízení (velké obrazovky, mobily, tiskárny). *CSS* je nezávislé na *HTML* a může být použito v jakémkoliv značkovacím jazyce založeném na XML. Oddělením HTML od CSS získáme další výhody, které usnadní udržování stránek i samotné stylování více stránek se stejnými elementy. [2]

Důležitým pojmem jsou **CSS selektory**. Určují, na které elementy se vztahují pravidla (obvykle stylování). Lze je použít i při procházení stránky internetovým robotem (viz kapitola 4).

Tabulka 2.1: Stručná ukázka několika pravidel CSS selektorů [3]

selektor	**příklad**	**popis**
element	div	vybere všechny elementy
[atribut]	[href]	vybere všechny elementy, které mají atribut href
[atribut=hodnota]*	[href*=predmet]	vybere všechny elementy, u kterých atribut href obsahuje hodnotu predmet

Takže pokud chceme vybrat všechny odkazy z dokumentu, které v adrese obsahují slovo *predmet*. CSS selektor bude vypadat: *a[href*=predmet]*

2.2 Analýza stránek českých univerzit

V následující kapitole se zaměřím na formu a obsah předmětů na webových stránkách vysokých škol a přehlednost jejich navigace.

Z prohlížení univerzitních webů jsem zjistil, že struktura jejich domovských stránek se podobá. Pro uchazeče zpravidla nabízejí po úvodním výběru fakulty popis oborů, důvody proč studovat právě u nich a podání přihlášky. Skladbu předmětů jednotlivých oborů však bývá obtížnější najít (existují i výjimky). Ve všech případech je buď uživatel přesměrován na informační systém univerzity nebo je obsah zobrazeného předmětu systémem vygenerován. Z tohoto důvod je proto je lepší rovnou najít informační systém univerzity (např. u Západočeské univerzity se jmenuje *portál*) a postupně procházet fakulty, obory a předměty.

Webové stránky vysokých škol většinou nepoužívají dynamické prvky,

které umožňují změny obsahu webu. Pro internetového robota je tudíž snazší získat potřebné informace, aniž by musel simulovat činnost člověka. Dynamické prvky jsou zde převážně použity pro měření dat nebo různých bannerů.

Většina univerzit dále nabízí *ECTS*[1] katalog předmětů. Předmět v katalogu by měl obsahovat údaje podle ECTS návodu[2] jako jsou název, semestr, obsah, studijní zatížení, literatura atd.

Výběr univerzit pro vytvoření kolekce probíhal podle několika faktorů. Nejprve byly vybrány fakulty, které ve svých programech vyučují informační technologie. Poté jsem použil hodnocení škol z několika různých zdrojů a vybral podle nich 10 nejlepších fakult. Čerpal jsem z každoročního vyhodnocení Hospodářských novin [5], hodnocení kvality webu z Webometrics [6] a z hodnocení všech univerzit na světě U-Multirank[7].

Tabulka 2.2: Vybrané univerzity s verzí HTML a počtem chyb na stránce

Název univerzity	**verze HTML**	**validní**	**počet chyb**
Masarykova univerzita Brno	XHTML 1.0	ne	7
Univerzita Karlova	HTML 5	ne	29
České vysoké učení technické	HTML 5	ne	76
Vysoké učení technické	HTML 5	ne	8
Západočeská univerzita	No doctype	ne	40
Univerzita Pardubice	HTML 5	ne	9
Technická univerzita Ostrava	XHTML 1.0	ne	1
Univerzita Tomáše Bati	HTML 5	ne	2
Vysoká škola ekonomická	XHTML 1.0 Transitional	ano	0
Technická univerzita Liberec	HTML 5	ne	23

2.2.1 Podobné rysy

Struktura zobrazení stránek univerzitních oborů je velice podobná. Největší problémem je duplicita předmětů nebo oborů při procházení stránek

[1] European Credit Transfer and Accumulation System (ECTS) je standard pro porovnávání práce studentů na vysokých školách po celé Evropské unii. 1 ECTS kredit odpovídá 25-30 hodinám studia průměrného studenta.

[2] více na `http://ec.europa.eu/education/tools/docs/ects-guide_en.pdf`

robotem. Většina univerzit totiž nabízí k nahlížení stejné studijní plány s aktuální, ale i se starou akreditací. Ukládání stejných předmětů můžeme zamezit vytvořením seznamu, do kterého budeme ukládat adresu a název předmětu. Navštívení další stránky je pak podmíněno tím, že adresa stránky není obsažena v tomto seznamu. Usnadňujícím prvkem je odkaz na každý detail předmětu obsahující v adrese řetězec *predmet*.

2.2.2 Rozbor univerzit

Než začnu s rozborem, vysvětlím, jak chápu termín *hledaný výraz*. Myslím tím textový řetězec, který je obvykle obsažen v samostatném HTML elementu a specifikuje druh informace o předmětu. Pokud se dále nespecifikuje, tak se předpokládá, že za tímto elementem následuje hledaný údaj. Pokud chci například znát předpoklady předmětu, ale na stránce jsou označovány jako prerekvizity, pak hledaný výraz pro získání předpokladů bude *prerekvizity*.

Informační systém **Masarykovy univerzity Brno** se nachází na adrese `is.muni.cz`. Informace o předmětu má jako jediná univerzita uložené v elementech *dl* (description list). Získání požadovaných údajů je proto jednodušší. Stačí najít pouze hledaný výraz. Výjimkou je název předmětu, který se nachází v nadpisu, a semestr, který je nutné speciálně parsovat. Zároveň je Masarykova univerzita unikátní v označení jednotlivých semestrů. Zatímco ostatní univerzity uvádí letní a zimní semestr na brněnské univerzitě je označují jarní a podzimní semestr. Cesta k předmětu vede přes vyplněný formulář s fakultou a oborem a následným výběrem odkazu předmětu. To nám umožňuje snadné vybrání všech odkazů vedoucích k detailním informacím o předmětech.

Univerzita Karlova má svůj informační systém na adrese `is.cuni.cz/studium`. Na rozdíl od ostatních škol nemá v detailu předmětu informaci, ve kterých oborech se předmět vyskytuje. Je tedy třeba počítat s tím, že při procházení je nutné si pamatovat obor, přes který jsem začal vyhledávat. Informace o vyučujícím také chybí, a proto jsem jej nahradil jménem garanta, jenž uvedený je. Dále chybí údaj o nutných předpokladech k absolvování předmětu. Ostatní informace se pak nacházejí v elementech, které lze jednoznačně identifikovat třídami a hledaným názvem. K samotnému předmětu lze pak dojít přes formulář s vyplněnou fakultou, výběrem oboru přes odkaz a poté opět klikneme na odkaz s předmětem.

České učení technické v Praze (ČVUT) nemá svůj informační sys-

tém veřejně přístupný. Nicméně předměty můžeme získat ze stránek fakulty informačních technologií. Přehled oborů se nalézá na `bk.fit.cvut.cz`. Nejjednodušší cesta k předmětům je přes sekci všechny předměty a navštívením odkazů s předměty. Většina obsahu stránky je strukturována do odstavců. Údaje o názvu předmětu, vyučujícím, semestru a počtu kreditů se nacházejí v tabulce. Název předmětu lze získat pomocí selektoru (viz kapitola 2.1) v posledním elementu v hlavičce tabulky (*th:last-of-type*). Obsah, cíl, literatura a předpoklady předmětu pak dostáváme hledáním výrazu v odstavcích. Je potřeba si dávat pozor na měnící se HTML strukturu stránky v závislosti na vyplněném obsahu. Jako řešení se nabízí zaměření pouze na obsahovou část a tím extrahovat pouze text nacházející se mezi dvěma hledanými výrazy.

Vysoké učení technické Brno (VUT) má svůj informační systém zakomponovaný v hlavní stránce `www.vutbr.cz` a z vybraných univerzit se hledal katalog předmětů nejsnáze (přes dva odkazy studium a studijní programy). Informace o předmětu pak vytěžíme podobně jako u ČVUT univerzity, jen s tím rozdílem, že název předmětu je v nadpisu a sylabus předmětu není v odstavcích, ale v buňkách tabulky(td).

Vysoká škola báňská — Technická univerzita Ostrava nemá veřejně přístupný informační systém. K přehledu předmětů lze dojít jako u VUT z hlavní stránky `www.vsb.cz`. Detail předmětu pak obsahuje všechny údaje, ke kterým se dostaneme přes rodiče elementu nalezeného výrazu.

Západočeská univerzita Plzeň, **Univerzita Pardubice** , **Univerzita Tomáše Bati** a **Technická univerzita v Liberci** mají jedno společné. Všechny univerzity používají informační systém studijní agendy (IS/STAG) vyvinutý na ZČU. Ten generuje na *ects* doméně třetího řádu studijní katalog (u ZČU je to adresa `ects.zcu.cz`). Jelikož používají stejný systém, jsou šablony a průchod robotem (viz kapitola 4.6) u všech univerzit stejný. Procházení probíhá přes odkazy v posloupnosti podle jména fakulty, studijního programu, studijního plánu a pak samotného předmětu. Údaje o vyučujícím, semestru, počtu kreditů se získá z hledaného výrazu, a zbytek informací jako u předchozí univerzity přes rodiče elementu nalezeného výrazu.

Vysoká škola ekonomická Praha — její informační systém je velmi nepřehledný. Použil jsem raději ECTS katalog na `www.vse.cz/ects`. U detailních informací o předmětu jsou všechna klíčová slova v elementu *small*. Stačí jej najít a k požadovanému textu se dostat přes rodiče vybraného elementu. O něco komplikovanější je cesta k magisterským předmětům, které se nacházejí v jiné záložce. Po získání bakalářských předmětů se celý proces

restartuje a začne u odkazu s magisterskými obory.

3 XML

Stažené předměty se budou ukládat do formátu XML. Tento typ souboru byl zvolen hned z několika důvodů. **Extensible Markup Language** (XML) je velmi jednoduchý, flexibilní značkovací jazyk. Je standardizován konsorciem W3C do formátu, který je čitelný jak pro člověka, tak i pro počítač. XML je široce využíváno pro výměnu různých typů dat. Mezi hlavní výhody tohoto jazyka patří přenositelnost, která má za následek podporu mnoha programovacích jazyků, nástrojů a nezávislost na operačním systému. XML podporuje univerzální jazykovou sadu, takže dokument může obsahovat více jazyků najednou, mezi kterými je možné přepínat. Dalším pozitivem jazyka je snadná konverze do jiných formátů, automatická kontrola struktury dokumentu a možnost vytvářet odkazy a hypertext.

3.1 Správně strukturovaný dokument

Jedná se o nejnižší úroveň kontroly, která se provádí při každém zpracování XML dokumentu. Aby byl XML soubor správně strukturovaný (*well-formed*), musí splňovat následující pravidla [4]:

- Musí existovat právě jeden kořenový element.
- Každá otevírací značka má svoji odpovídající ukončovací značku.
- Elementy se nesmějí navzájem křížit nebo překrývat.
- Hodnoty atributů musí být uzavřeny v uvozovkách nebo apostrofech.
- V elementech nesmějí být stejně pojmenované atributy.
- Komentáře nesmějí být vnořené a ani uvnitř značek.
- Ve znakových datech nejsou znaky < a &.

3.2 Schéma

XML schéma(XSD) je dalším důležitým pojmem při pracování s XML dokumenty. Jeho cílem je definovat strukturu XML dokumentu a tím získat nástroj pro snadné ověření správnosti nově vytvořené kolekce. XML schéma definuje:

- elementy a atributy, které se mohou objevit v dokumentu
- rodičovské elementy a počet a pořadí jejich potomků
- jednoduché a komplexní datové typy elementů a atributů
- různá omezení (např. minimální počet výskytu, může být element prázdný, maximální délku, výčet přípustných hodnot atd.)
- jmenné prostory, referenční integritu (podobné relacím v databázi)
- jedinečnost hodnot (zabránění výskytu duplicit)

Pro můj typ úkolu jsem si zavedl dva jednoduché datové typy, pro které v případě potřeby můžeme přidat nějaká omezení. Prvním je textový řetězec, který bude ve většině elementech a atributech.

```
<xs:simpleType name="stringType">
  <xs:restriction base="xs:string"/>
</xs:simpleType>
```

Druhým jednoduchým datovým typem je kladné celé číslo, které se bude používat v elementu pro počet kreditů.

```
<xs:simpleType name="intType">
  <xs:restriction base="xs:positiveInteger"/>
</xs:simpleType>
```

Komplexním datovým typem bude předmět. Přehled všech elementů obsažených v tomto typu můžete vidět na další stránce. Jedná se o výběr nejdůležitějších informací o předmětu, které byly dostupné na všech stránkách univerzit, a ty, které by budoucí zájemce o studium mohl chtít znát. První element *all* je pro účely vyhledávání. Samotný předmět obsahuje údaje: obsah, počet kreditů, obory, cíl, literaturu, předpoklady, semestr a odkaz na předmět.

Předtím než zmíním výčet nezahrnutých údajů musíme si uvědomit, že kolekce je primárně určena pro studenta přihlašujícího se na univerzitu. Takový student nemá většinou znalosti průběhu výuky a studijního systému. K předmětu nebyla zahrnuta informace o metodě výuky. Většinou obsahovala kombinace přednášek a cvičení, což pro studenta není moc důležitá informace. Dalším vynechaným údajem je způsob hodnocení, u kterého nevidím důvod, proč by mohl rozhodovat při výběru předmětu. Třetí neuvedený údaj je rok studia. Ten může být leckdy zavádějící, protože pro jiný obor může mít předmět v jiném studijním roce. Mezi další vynechané údaje jsem zařadil vyučovací jazyk, typ studia (dá se zjisti z oboru), typ předmětu a studijní praxe. Všechny tyto zmíněné informace mají zároveň společné to, že nebyly k dispozici na všech univerzitních stránkách.

```
<xs:complexType name="courseType">
  <xs:sequence>
    <xs:element name="all" type="stringType" />
    <xs:element name="content" type="stringType" />
    <xs:element name="credits" type="intType" />
    <xs:element name="fields" type="fieldsType" />
    <xs:element name="goal" type="stringType" />
    <xs:element name="literature" type="stringType" />
    <xs:element name="prerequisites" type="stringType" />
    <xs:element name="semester" type="stringType" />
    <xs:element name="url" type="stringType" />
  </xs:sequence>
  <xs:attribute name="name" type="stringType"/>
</xs:complexType>
```

U elementu *fieldsType* si můžeme všimnout dalšího neznámého datového typu. Bude se jednat o seznam oborů, ve kterém se daný předmět vyučuje.

```
<xs:complexType name="fieldsType">
<xs:sequence>
  <xs:element name="item" type="stringType" />
</xs:sequence>
</xs:complexType>
```

Předměty pak budou vnořené do elementu *university*, kde bude v atributech její jméno a jméno fakulty. A rodičem všech univerzit pak bude element informatické předměty, který bude obsahovat celou vytvořenou kolekci.

3.3 Ukázka

Ukázka předmětu základy kybernetiky pro informatiku na Západočeské univerzitě bude v kolekci vypadat takto :

```
<informaticCourses>
    <university name="Západočeská univerzita" faculty="Fakulta
        aplikovaných věd">
        <course name="Základy kybernetiky pro informatiky">
            <all>základ kybernetik informatik finančn
                informatik statistik cíl seznámn základn přístup
                princip kybernetik předmět .. </all>
            <content>1. Úvodní poznámky o kybernetice. 2.
                Základní pojmy a poznatky teorie informace.
                Entropie, komunikační kanál, kódy...</content>
            <credits>3</credits>
            <studyFields>
                <item>Informační systémy</item>
                <item>Výpočetní technika</item>
                <item>Informatika</item>
                ..
            </studyFields>
            <goal>Cílem je seznámení se základními přístupy a
                principy kybernetiky. Předmět má ...</goal>
            <literature>Kotek, Z., Vysoký, P., Zdráhal, Z.
                Kybernetika. Praha, 1990. Mařík, Vladimír a kol.
                Umělá inteligence (2). Academia, Praha,
                1997..</literature>
            <prerequisites>Nejsou předepsány žádné specifické
                předpoklady.</prerequisites>
            <semester>Zimní</semester>
            <url>http://ects.zcu.cz/predmet/KKY...</url>
        </course>
    </university>
</informaticCourses>
```

3.4 JAXB

Abychom mohli kolekci ukládat a dobře s ní manipulovat, potřebujeme prostředky pro zpracování XML dokumentu. Základní přístupy jsou proudové čtení a reprezentace dokumentu do stromu. Mezi nejznámější zástupce těchto dvou způsobů patří SAX a DOM.

Výhodou proudové čtení je velká rychlost a nízké paměťové nároky. Zato stromová reprezentace má celý dokument dostupný v paměti, což nám umožňuje se k jednotlivým údajům vracet nebo je upravovat.

JAXB (Java architecture for XML Binding) představuje zástupce stromové reprezentace, který provádí mapování na objekty. Jednotlivé objekty jsou převedeny z elementů v XML souboru. Třídy reprezentující objekty mohou vzniknout automatickým generováním XSD. JAXB je vhodné použít v situacích, kdy je známé schéma dopředu, obsahuje opakující se informace a potřebujeme XML načítat a ve velkém rozsahu upravovat. [4]

4 Internetový bot

Internetový bot nebo také webový robot je program, který automatizovaně vykonává činnost při procházení Internetu. Jedná se většinou o jednoduché opakované úkony, které jsou prováděny daleko rychleji než by dokázal člověk. Největší využití botů se skládá z indexování pro vyhledávače, analyzování obsahu stránek nebo sbírání dat.

Internetový bot je také znám pod dalšími názvy: *spider*, *crawler* nebo *robot*. Terminologie v tomto není moc jednotná, a proto zde může docházet k nedorozuměním. Například Google považuje tyto termíny za rovnocenné [8]. Na druhou stranu jinde vysvětlují rozdíl, že spider je program, který v určitý čas čte web stránky jako uživatel (tudíž je něco jako prohlížeč). Crawlerem je myšlen sémantický algoritmus, který říká spiderovi kdy, kde a jak hluboko danou stránku procházet. [9]

4.1 Druhy robotů

Vyhledávací roboti jsou využívány všemi internetovými vyhledávači. Používají své algoritmy, které stanoví jaké stránky navštívit, jak často a kolik z jedné adresy. Každou navštívenou stránku analyzuje a provede indexaci[1] všech slov a jejich umístění na stránce.

Batch crawlers pokračují v prohledávání dokud není splněna nějaká podmínka. Obvykle se jedná o zastavovací kritérium typu [11]:

- dokud nevyprší určitý čas
- nějaký počet stránek byl navštíven
- paměťový limit byl překročen

Incremental crawlers neustále znovu procházejí již dříve navštívené

[1] Indexace proces vyjádření obsahu dokumentu pomocí prvků selekčního jazyka, obvykle s cílem umožnit zpětné vyhledávání. (BALÍKOVÁ, 2003, *Česká terminologická databáze knihovnictví a informační vědy*)

stránky a starají se tak o aktuálnost obsahu. Web je ze své podstaty velmi dynamický, protože obsah přibývá, aktualizuje se a maže se. K udržení aktuálního obsahu se nejvíce používají ohodnocující funkce nazvané *freshness* a *age*. [12]. Freshness funkce vrací 1, pokud je lokální kopie přesná, a 0 pokud není. Age funkce měří, jak je lokální kopie zastaralá.

Focused crawlers projde určitou množinu stránek týkajících se jednoho tématu a snaží se minimalizovat počet stránek, které s požadovaným tématem nemají nic společného.

Boti pro správu a údržbu se používají hlavně v rozsáhlých webových informačních systémech, kde zpravidla vytvářejí nová přesměrování, kontrolují validní HTML, odstraňují nefunkční odkazy nebo cyklické směrování.

Spamboti jsou typem automatizovaného programu, který pomáhá šířit nevyžádané zprávy (spam). Toho docílí několika způsoby: sbírají e-mailové adresy nalezené na stránkách, přidávají příspěvky na blogy, fóra a do komentářů, ať už za účelem šíření reklamy nebo zvýšením počtu zpětných odkazů na svou stránku. Nejznámější obranou proti poslednímu zmiňovanému problému je technika CAPTCHA[1]. Jedná se o Turingův test, kde opsáním pro člověka čitelného textu zamezíte botům v přístupu na určitou část webové aplikace.

4.2 Konvence internetového bota

Martijn Koster zformuloval zásady pro vytváření internetového bota. Zde jsou nejdůležitější body [10]:

4.2.1 Buď odpovědný

Identifikace robota — HTTP protokol má v hlavičce pole *User-agent* pro identifikování WWW prohlížeč. Jelikož robot je něco jako prohlížeč, může použít "mujRobot/verze", a tím se pro správce serveru odlišit od běžných prohlížečů.

[1] Completely Automated Public Turing test to tell Computers and Humans Apart

Identifikace tvůrce — HTTP podporuje pole *From*, které identifikuje uživatele pouštějící WWW prohlížeč. Toto pole se používá pro uvedení e-mailové adresy, aby ho správce serveru mohl kontaktovat v případě potíží.

Informuj — Často správce zajímá, proč je navštíven jejich server. Lze je informovat polem *Referer*, který slouží k určení, z jaké stránky uživatel přišel, nebo v případě bota se může jednat o adresu, popisující jeho funkce a účel.

Buď přítomen — Doporučuje se pro tvůrce nenechat bota běžet v době dlouhodobé nepřítomnosti. Bot může začít působit škody a pak nelze kontaktovat jeho tvůrce, aby jej vypnul.

4.2.2 Nepřivlastňuj si zdroje

Testuj lokálně — Nespouštěj opakované testy na vzdálené servery. Místo toho vyzkoušej na několika lokálních serverech a analyzuj výkon, výsledky a zatížení, a až poté pust' do reálného provozu.

Raději pomalu než rychle— Internetový bot velice snadno zahltí server stovkami požadavků za minutu, a proto je důležité, aby dodržoval nějaký časový odstup od každého nového požadavku. Koneckonců by mohl server botovi zamezit přístup úplně kvůli neúnosnému zatěžování a nezískali bychom žádné informace.

Použij HEAD požadavek — Kde je možné, použij méně náročnější požadavek *HEAD* místo *GET*.

Řekni, co chceš — V hlavičce HTTP je pole *Accept*, které určuje, jaké typy dat dokáže prohlížeč zpracovat. Pro robota procházejícího pouze text není třeba, aby server posílal obrázky, média, archivní souboru, atd.

Neopakuj se — Neprocházej stejnou adresu vícekrát. Je dobré si vést list navštívených adres, aby nedošlo k opětovnému navrácení. Tento bod zároveň eliminuje zacyklení.

Vyber pravý čas — Většina serverů mívá během určité části dne menší zatížení, a proto je vhodné vybrat takový čas, kdy běžný uživatel stránky spí.

4.2.3 Výsledky bota

Loguj — Vedení různých statistik typu počet úspěšných/neúspěšných požadavků, velikost stránky, navštívené adresy, atd. pomáhá odhalit některé chyby.

Ukládej výsledky — Je dobré si uchovávat tolik dokumentů, kolik to jen jde. Dokumenty se mohou případně znovu prozkoumat jinou metodou a získat nové užitečné výsledky.

4.3 Parsování HTML

Ačkoliv má HTML předepsaný standard, téměř každá stránka obsahuje chyby (tzv. není validní). Mezi nejčastější chyby patří chybějící tagy, neuvedená deklarace DOCTYPE nebo kódování a špatné pořadí elementů. Důležitou vlastností HTML parseru je se vypořádat s těmito chybami.

Pro parsování celých HTML dokumentů se ještě nabízí použití regulárních výrazů (*regex*). To by však nebyla vhodná volba, protože HTML je gramatika typu 2 (bezkontextová gramatika) podle Chomského rozdělení a regulární výrazy rozpoznají pouze gramatiku typu 3 (regulární gramatika). Z toho plyne, že regulární výraz nemůže zpracovat celý HTML dokument, jelikož závisí na existenci otevíracích a uzavíracích tagů. Na druhou stranu regex je velmi vhodný při získávání určité informace z obsahu nějakého elementu.

Další problém může vzniknout, když robot navštíví web s rámy (frames). Prohlížeč je rozdělen na několik vymezených rámů, kde v každém rámu je samostatná HTML stránka. Tudíž k sestavení celé stránky je třeba načíst každý rám. Naštěstí se od rámů upustilo z důvodů špatné udržovatelnosti, grafické vizualizaci, ukládání, ovládání a pomalejší rychlosti načítání.

Posledním problémem jsou dynamické prvky na stránce, které jsou přístupné až po určitě akci. Typickým příkladem je spuštění funkce javascriptu kliknutím na tlačítko. Pokud robot chce získat všechna data, měl by být schopen simulovat tuto činnost. Ovšem leckdy je složité provést přesnou posloupnost akcí k dosažení požadované informace. Usnadňujícím faktorem může být znalost webové stránky, kde se pak dají akce provádět pro konkrétní případ.

4.4 Výběr programovacího jazyka

Mým prvním úkolem je vytvořit webového robota, který projde stránky univerzit a stáhne ty, na nichž se nachází seznam předmětů.

Na tento problém lze použít celou řadu programovacích jazyků. Zkusím vybrat vhodné pro tento účel a zhodnotit jejich využití.

- Skriptovací jazyk **PHP** je především určený pro vývoj dynamických webových aplikací. Není však vyhovující pro tento typ úkolu. Důvodů je několik: PHP samo o sobě nepodporuje vícevláknové programy, špatně se udržuje rozsáhlá architektura a existující knihovny či framework jsou velmi slabé.
- **Python** se jeví jako velmi vhodný. Je snadný na rychlé naučení. Knihovny jako *Beautiful Soup* nebo framework *scrapy* se jeví jako snadný start. Na druhou stranu jazyk neoplývá žádnou speciální vlastností, která by jej zvýhodňovala při procházení a stahování stránek.
- **Clojure** tzv. moderní Lisp zaměřený na zjednodušení vývoje vícevláknových aplikací se stal neskutečně silným a produktivním nástrojem v této oblasti. Jeho hlavní nevýhodou je velká časová náročnost na osvojení jazyka.
- **Javascript** je dalším skriptovacím jazykem a vhodným řešením pro tento problém. Obsahuje mnoho knihoven (např *pjscrape*) poskytujících snadnou implementaci a dostatečnou funkčnost.
- **Java** nejrozšířenější objektově orientovaný jazyk je na tom podobně jako zmíněné jazyky *javascript* či *python*. Opět obsahuje velké množství knihoven řešících daný problém a nabízí snadnou implementaci.

Rozhodl jsem se využít programovací jazyk *Java* a to hned z několika důvodů. *Javu* ovládám z programovacích jazyků nejlépe. V prostředí naší univerzity je také hojně používaná, takže případná návaznost na moji práci by byla snazší. Počet použitelných knihoven je vyšší než u ostatních jazyků. Dále při rozsáhlejším projektu nebo při přidávání dalších funkčních modulů se *Java* snáze udržuje.

Následující HTML parsery jsou napsané v Javě a všechny mají licenci Open source.

4.5 Porovnání HTML parserů

NekoHTML — *http://nekohtml.sourceforge.net*

Jednoduchý HTML skener a doplňovač chybějících tag elementů umožňuje programátorovi naparsovat HTML dokumenty a přistupovat k nim pomocí XML rozhraní. Opravuje běžně se vyskytující chyby, přidá nebo opraví chybějící elementy a poradí si se špatně vloženými tagy. Je založen na populárním XML parseru Xerces.

Přistupování ke stránce pomocí rozhraní XML mi nepřipadá efektivní a průhledné. Plusem může být přístupná dokumentace, kde lze vyčíst, že celková funkčnost není oslnivá, a v porovnání s ostatními nabízí pouze nejnutnější prvky.

HTML Parser — *http://htmlparser.sourceforge.net*

„Rychlý real-time parser pro real-world HTML“, jak o sobě tvrdí HTML parser. Hlavním lákadlem je jednoduchý design, rychlost a vypořádání se s online streamováním HTML. Je zejména vhodný pro dva scénáře: extrakci dat a transformaci. Získávání dat zahrnuje potřebné metody jako vytěžení textu, odkazů, různých souborů (obrázky, hudba) ze stránky. Další zajímavou vlastností HTML parseru je kontrola validních odkazů a sledování stránek (kontrola aktualizace obsahu). Transformací se pak rozumí upravování odkazů, cenzura (odstraňování sprostých slov), validace HTML, odstraňování reklam a konverze stránky do XML.

Sestavování filtrů pro parsování HTML se jeví krkolomné a chybí mi tu několik důležitých filtrů (jako jsou CSS selektory, filtry na tabulky atd.). Dále postrádám ukázkové příklady a nesmím opominout, že knihovna je z roku 2006 a poslední update vyšel před více než rokem.

Jericho HTML Parser — *http://jericho.htmlparser.net*

Jericho HTML Parser umí analyzovat a upravovat HTML dokument včetně tagů na straně serveru(PHP, JSP, ASP). Poradí si se špatně formátovaným HTML dokumentem. Má snadno přístupné jednotlivé elementy, které lze upravovat, aniž by se musel vytvářet znovu celý dokument z DOM stromu (nevyužívá parsování do stromu ani pomocí událostí jako DOM nebo SAX). Jedná se o kombinaci textového vyhledávání a tag rozpoznávání.

Přehledná dokumentace, obsáhlé vyjmenování vlastností a názorné ukázky jsou hlavní předností této knihovny. Pozici na hlavního kandidáta pro použití pouze brání absence CSS selektorů a neaktuální podpora (poslední update vyšel před dvěma lety).

JTidy — *http://jtidy.sourceforge.net*

Knihovna převážně funguje jako kontrolor syntaxe s jejím následným opravením. Může tedy být použita pro poškozené či špatně strukturované HTML dokumenty. K elementům přistupuje přes strom DOMu.

Pro moje účely to není příliš vhodná knihovna svojí funkčností. Dále při opravování HTML vyčistí vše, co nezná, což nemusí být vždy přínosné. Zároveň nesmím opomenout, že poslední aktualizace proběhla v roce 2009.

HTML cleaner — *http://htmlcleaner.sourceforge.net*

S podobnou funkčností jako předchozí knihovna přichází i HTML cleaner. Vytvoří well-formed XML a k elementům přistupuje pouze přes DOM.

Na rozdíl od JTidy je knihovna stále udržovaná, ale soustřeďuje se především na čisté HTML než na snadný přístup k elementům.

Jsoup — *http://jsoup.org*

Jsoup umí pracovat s real-world HTML (dokáže opravit špatnou strukturu). Poskytuje snadné API pro stahování a manipulaci s daty za použití toho nejlepšího z DOMu, CSS a metod podobných jquery. Dále implementuje HTML5 standard a parsuje HTML do stejného DOMu jako moderní prohlížeče.

Knihovna obsahuje názorné ukázky použití mezi něž patří: parsování dokumentu z textového řetězce, načtení dokumentu z url/souboru, dolování dat pomocí různých metod, úprava elementů a vyčištění HTML kódu od škodlivého kódu. Následující příklad demonstruje snadné získání všech elementů s odkazem ze stránky Západočeské univerzity.

```
Document doc = Jsoup.connect("http://www.zcu.cz").get();
Elements linkElements = doc.select("a");
```

Knihovna pod MIT je průběžně vylepšována (poslední update 13.4.2015) a má poměrně velkou komunitu a také kvalitní podporu.

4.5.1 Shrnutí

Pro účely stahování stránek jsem si vybral knihovnu *Jsoup*. K rozhodnutí přispěly důvody jako podpora HTML 5, snadná extrakce dat pomocí CSS selektoru nebo DOMu. Podporuje i manipulaci HTML elementů, která může být nápomocná při vyplňování různých formulářů, vyhledávacích boxů a různých selektorů/dropdownů. Dalším plusem je přístup a parsování HTML přes URL, soubor nebo i String.

Neméně důležitými aspekty pro použití této knihovny se stala MIT licence, dobře popsaná dokumentace s ukázkovými příklady a již zmíněná aktuální udržovatelnost projektu, velká komunita uživatelů a kvalitní podpora.

Tabulka 4.1: Srovnání vybraných HTML parserů

Parser	License	Latest update	HTML Parsing	CSS selector	HTML 5 support
NekoHTML	Apache Software	2014-06-02	Ne	Ne	Ne
HTML Parser	General Public	2006-09-23	Ano	Ne	Ne
Jericho HTML Parser	General Public	2012-10-30	Ne	Ne	Ne
JTidy	JTidy License	2009-12-01	Ano	Ne	Ne
HTML cleaner	BSD License	2014-10-31	Ne	Ne	Ne
JSOup	MIT license	2015-04-13	Ano	Ano	Ano

4.6 Způsob procházení a získávání dat

Na dolování dat z webových stránek proběhlo několik výzkumů. V současné době dominuje technika dolování nazývaná *wrapper induction*. Wrapper je procedura pro získání obsahu z určitého zdroje informací a jejich uložení do relačního modelu. Tato technika je založena na kontrolovaném učení procedur. Procedura se na začátku natrénuje na ručně označeném obsahu (*manual labeling*), který chceme stáhnout. Tím si vytvoří množinu pravidel (šablonu) pro získávání dalších dat bez znalosti struktury zdrojů. Nevýhoda je časově náročné manuální označování. Vytvořená pravidla z natrénování nefungují na velké množství stránek a je potřeba vytvořit novou vzorovou množinu[13].

Podobným způsob na dolování dat se může jevit *instance-based learning*.

Ten nejprve ve své klasické podobě uloží označené instance. Pak při zkoumání nové instance porovnává s jinými již uloženými a vyhodnotí výsledek. K porovnání se nejčastěji používá algoritmus k-nejbližších sousedů (k-nearest neighbors) nebo případové usuzování (Case-Based Reasoning) [14]. Jedná se o běžný způsob klasifikace. Můžeme si všimnout podobného problému jako u předchozí techniky, a to je vytvoření ručně označené množiny pro další porovnání. Existuje metoda, která eliminuje časově náročné manuální označovaní požadovaného obsahu. Uživatel vyznačí informace, které ho zajímají, pouze u jedné stránky. To stačí pro další běh programu. Přesné získání chtěných informací tkví v určení podobnosti mezi uloženým obsahem a novým cílovým [15].

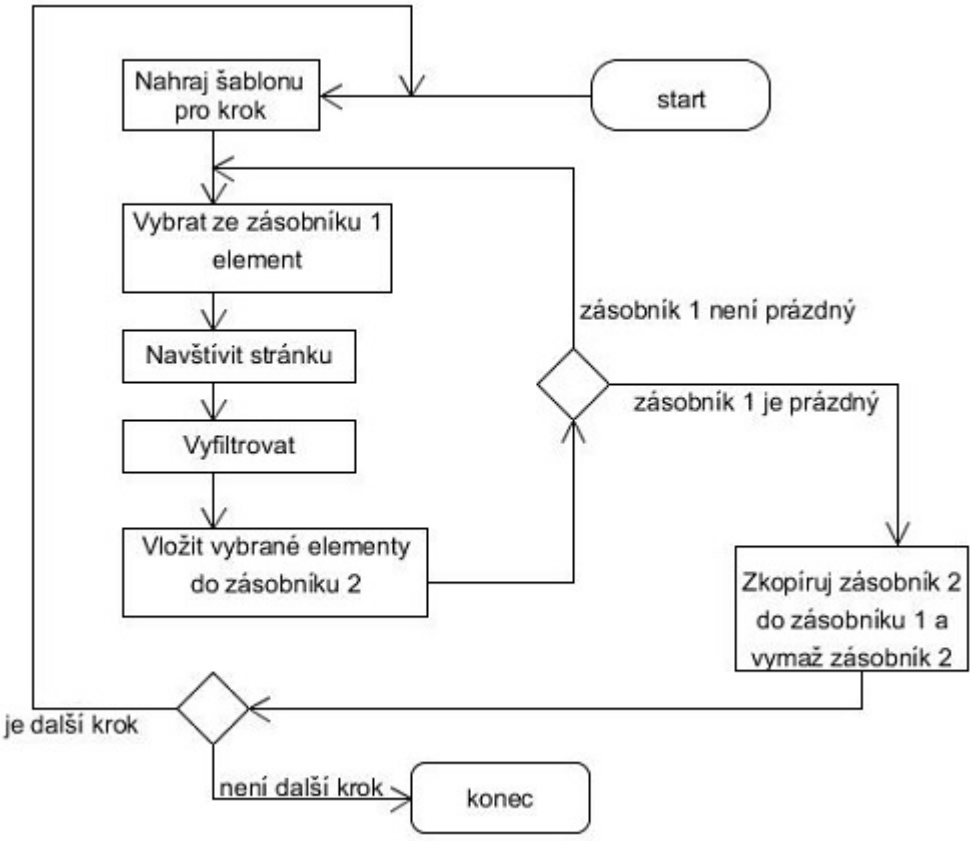
Vhodnějším řešením bude napsat šablony na procházení jednotlivých stránek. Hlavními nevýhodami bude větší časová náročnost a nutná znalost struktury webové stránky. Bude se jednat o implementačně daleko jednodušší variantu, protože seznam stránek není velký. Dále lze předpokládat, že vysoké školy nebudou strukturu svých webů tak často měnit. Odměnou budou stažená data, u nichž budeme vědět, kam přesně je zařadit. Pro budoucí vyhledávání tak budeme porovnávat významově stejná data. Například na jedné univerzitě mají u předmětu uvedeno *Výstupy z učení* a na druhé *Cíle předmětu*. Při použití šablony budeme vědět, že se jedná o totéž a uložíme výsledek ke stejnému klíči.

Ze sekce 4.2.2 víme, že nemáme zatěžovat server velkým počtem požadavků za krátký čas. Velikost intervalu na vykonání dalšího požadavku necháme na volbě uživatele v aplikaci. Vzniká však problém prodloužení čekací doby na stažení všech předmětů. To lze vyřešit paralelismem. Při paralelním běhu bota ve více vláknech si musíme dávat pozor na opakované navštívení stránky. V našem případě je řešení jednoduché. Každému crawlerovi přiřadíme jednu adresu univerzity.

Chování našeho bota vypadá následovně (obrázek 4.1). Postup bota po stránce se skládá z kroků. U každého kroku je obecně určeno, na které elementy a textové řetězce se zaměřovat (důležité pro fázi filtrování elementů).

Následující úryvek metody se používá ve stavu navštívení stránky (Obrázek 4.1). Vyhodnocuje, zda element je odkaz, odesílací tlačítko (input) nebo výběr z možností (option)

```
Elements analyze(Element el, String searchedText, String url){
       if (el.tagName().equalsIgnoreCase("a")) {
              url = el.attr("abs:href");
              return visitNextPage(url, searchedText);
..
```



Obrázek 4.1: Vývojový diagram pro internetového bota

5 Zpracování vytvořené kolekce

5.1 Způsob uložení

Ze stažených stránek můžeme získat různé informace a nabízí se otázka, zda je uložit nebo vyfiltrovat k potřebným účelům a zbytek zahodit. Úkolem bylo vytvořit kolekci informatických předmětů, kde formát každého předmětu bude předem daný, a tak je lepší se soustředit pouze na určité druhy informací (viz sekce 3.2). Prakticky tedy není důvod se ke stránce vracet, natož ji ukládat. Navíc si uchováváme odkaz na stránku s předmětem a v případě potřeby jej můžeme znova navštívit.

Nicméně „vyfiltrovaný“ obsah je třeba nějakým způsobem uložit, aby se dal využít na následné vyhledávání, k čemuž je přímo určen.

5.1.1 Ukládání do souborů

Uložený soubor odpovídá kolekci uložených předmětů

Tato varianta ukládání má své výhody a nevýhody. Výhodou je rychlý sekvenční přístup a implementačně nejjednodušší způsob ukládání. Jako největší nevýhoda se jeví detekce duplicitních předmětů. Procházet celý soubor nebo jenom jeho část a zjišťovat, zda předmět již nebyl zapsaný, by bylo velmi časově náročné. Duplicitu by bylo nejvhodnější ošetřit již před zapsáním (tedy za běhu). Musíme brát v potaz, že se nám musí vejít do paměti RAM seznam předmětů, kde může dojít k duplicitě.

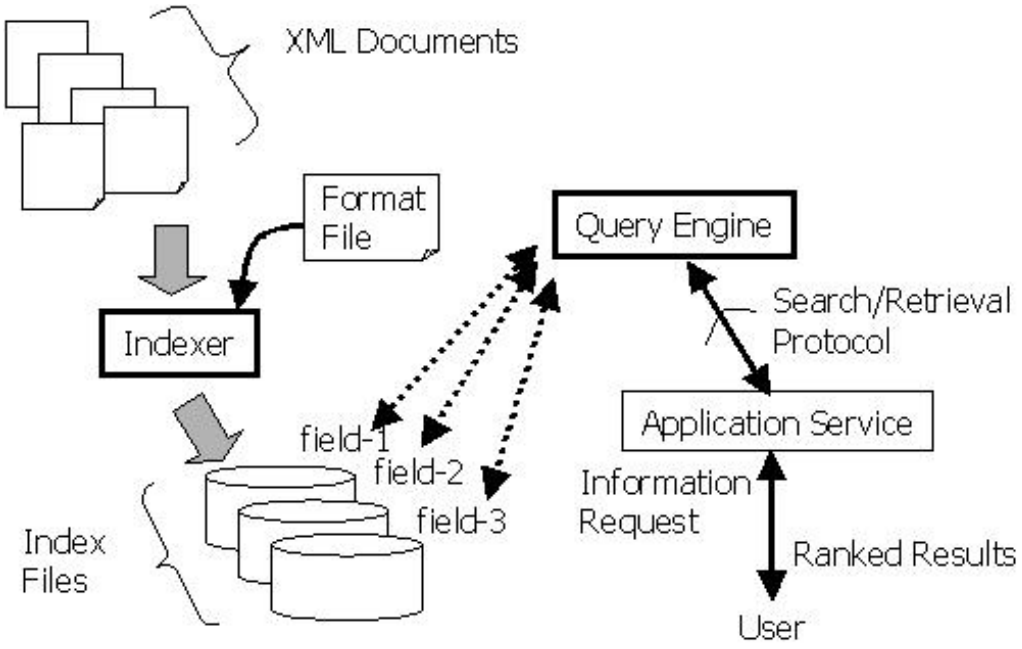
Uložený soubor odpovídá jednomu předmětu

Uložením předmětu po jednom usnadní problém řešení duplicit. Určili bychom si jednotný formát názvů souborů, a tím se vyvarovali kolizím i další kopiím. Na druhou stranu si přitížíme podstatným zhoršením rychlosti čtení a redundancí dat (některé vlastnosti budou společné a v samostatném souboru budou muset být znovu uvedeny).

5.1.2 Ukládání do databáze

Ukládání předmětů do relační databáze bude mít složitější implementaci. Zato nám poskytne celý aparát, který se za ní skrývá. Myslím zejména použití libovolného jazyka na přístup do databáze, zachování integrity dat, vzdálený přístup a možnost distribuovat systém. Dalším plusem by bylo snadné vyvarování duplicit a jednoduché vyhledávání pomocí dotazovacích jazyků.

Alternativním způsobem může být uložení do nativní XML databáze, kde by byl snadný přechod ze souborového způsobu ukládání. Pravděpodobně v budoucnu budeme potřebovat poskytnout službu, která by umožnila vyhledávání online. Předměty před vložením do XML databáze musíme nejdříve převést do správného formátu a poté zaindexovat. Nad tím vytvořit dotazovací službu (*query engine*), kterou poté můžeme volat libovolnou aplikační službou přes zvolené rozhraní. Návrh by mohl vypadat následovně (viz 5.1):



Obrázek 5.1: Návrh architektury vyhledávání v XML databázích [16]

5.1.3 Shrnutí

Úkolem bakalářské práce je vytvořit XML kolekci informatických předmětů. Byl zvolen způsob ukládání do souboru, kde jeden soubor odpovídá celé kolekci. Výhodou je snadnější implementace. Dále k tomuto řešení přispěl i fakt, že objem dat nebude velký (1 univerzita má velikost souboru cca 2 MB). Zpracování XML souboru pro následné prohledávání nebude časově náročné. V případě enormního zvětšení dat není díky zvolenému XML formátu problém přejít na jiný způsob uložení (například na zmíněnou XML databázi).

5.2 Lemmatizace

Pro dosažení cíle relevantní odpovědi při vyhledávání je potřeba indexovat jednotlivá slova v základním tvaru. Vyhledávací algoritmus nemusí rozumět českému jazyku, a tak například při požadavku na „levné peněženky“ vyhodnotí výraz jako jiný než „levná peněženka“. Proto je nutné jednotlivá slova vyskytující se v různých morfologických tvarech (pád, osoba, číslo atd.) převést na základní tvar tzv. lemmu. Lemmatizace je proces převádění slov na lemmy. Předpokládá se, že základní tvar bude mít stejný význam ve všech jeho tvarech. Úskalí nastává v jazycích, které obsahují slova mnohoznačná. [20]

5.2.1 Způsoby lemmatizace

Lemmatizaci lze provést několika způsoby:

- *slovník kořenů* — Pokud je k dispozici slovník kořenů (základních tvarů) s příslušným skloňováním, je možné jej využít pro vytvoření lemmy. Výhodou je minimální chybovost metody. Celková úspěšnost pak záleží na rozsáhlosti slovníku a omezení použití na specifický obor, kde se hojně používají odborné termíny. Hlavní nevýhodou je skutečnost, že slovník je v podstatě vždy neúplný. Ve slovníku se nemůžou nacházet všechna vlastní jména, názvy nebo odborné termíny. Neúplnost slovníku také znamená problém, že pokud požadované slovo nenajdeme ve slovníku, lemmatizace nám neposkytne ani přibližný odhad kořene slova.

- *odstranění předpon a přípon* — Základní tvar se získá odstraněním přípon a předpon na základě slovníku afixů[1] nebo pomocí definovaných pravidel. Tato metoda velice dobře funguje u jazyků, kde se nemění kořen slova a hlásky při spojování slov. Příkladem je anglický jazyk, u kterého se nejčastěji používá Porterův algoritmus založený právě na odstraňování přípon a dalších morfologických[2] pravidel angličtiny.
- *statistická metoda* — Převod probíhá na základě statistického přístupu a strojového učení. Frekvence jednotlivých skupin písmen určuje, zda se jedná o příponu nebo předponu. Výhodou je nezávislost na konkrétním jazyku.

5.2.2 Použité řešení

Na český jazyk už bylo vytvořeno několik lemmatizátorů. *CLEF2007* se zaměřuje na slovní druhy nejvíce vystihující význam věty. Jedná se o podstatná a přídavná jména. Byl tak vytvořen soubor pravidel k získání kmenu slova.

Dalším je program *Morce* na morfologickou analýzu českého jazyka. Jádrem lemmatizátoru je neuronová síť, která se učí na slovníkových datech. Převod probíhá na základě spojení slovníkového a statistického způsobu lemmatizace [17].

Posledním ve výčtu jsou algoritmy navržené v programovacím jazyce *Snowball*. Jedná se o jazyk speciálně vytvořený pro účely popisu algoritmů lemmatizace nebo stemmingu (převod na kmen slova). Z nejzajímavějších algoritmů zmíním zlepšený stemmer, který vznikl na Západočeské univerzitě [18] a stemmer založený na gramatických pravidlech.[19]

Většina ze zmíněných algoritmů je implementována v projektu *Apache Lucene*. Lucene je open source knihovna určená pro získávání a vyhledávání informací. Pro účely práce byla použita pouze část knihovny umožňující lemmatizaci. Z výběru jejích lemmatizačních algoritmů jsem zvolil standardní

[1] Afix je jednotlivá část slova, která se připojuje k základnímu tvaru slova (kořenu nebo kmeni).
[2] Morfologie neboli tvarosloví je nauka o struktuře a tvaru slov (včetně skloňování, časování, derivace atd.)

metodu (oproti agresivnější). Výhodou tohoto algoritmu je rychlost, protože pracuje na základě gramatických pravidel. K dalšímu důvodu, proč použít právě tento způsob, byla snadná implementace a poměrně velká přesnost.

K porovnání jsem ještě implementoval vlastní lemmatizační algoritmus, kde jsem využil dodaného slovníku kořenů, který obsahuje přes 2 miliony tvarů. Tento postup se samozřejmě potýkal s problémy slovníkového způsobu. Největší potíží byla dlouhá doba načtení slovníku do paměti při lemmatizaci vytvořené kolekce. Dále pak absence převodu názvů a vlastních jmen.

Z těchto dvou naprogramovaných řešení jsem vyhodnotil jako lepší variantu pro použití lemmatizační algoritmus od knihovny Lucene. V osnovách informatických předmětů sice není velký rozsah slovní zásoby, zato se vyskytuje velký výčet zkratek a vlastních jmen v odborných názvech (např. Petriho sítě). A právě v těchto případech slovníkový přístup zaostává.

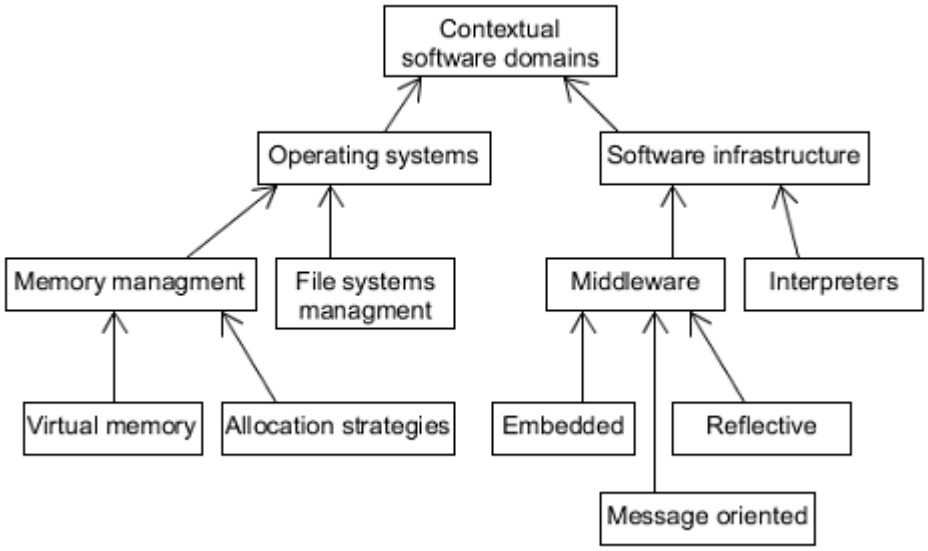
5.3 Ontologie

Ontologie má mnoho definic. Výraz pramení z filozofie a z pohledu informatiky se přirovnává k taxonomii. Vyjadřuje vztahy mezi určitými pojmy. Je to jednoduché zachycení nadtříd a podtříd objektů v naší realitě. Rozšířená definice od T. Grubera zní: „Ontologie je formální, explicitní specifikace sdílené konceptualizace“.

5.3.1 Doménová ontologie

Doménová ontologie popisuje vztahy v nějaké pojmové oblasti. Slouží k pochopení struktury informací mezi lidmi a počítači. Můžeme pomocí ní dělat předpoklady. V našem případě poslouží pro zlepšení vyhledávání v informatické kolekci. Například budeme chtít vědět, v kterých předmětech se probírá „virtuální paměť“. Pomocí doménové ontologie můžeme usoudit, že předmět může také obsahovat termín nadřazený. V našem případě (viz Obrázek 5.2) můžeme zahrnout do vyhledávání termín „správa paměti“ a pokud chceme hledat ještě obecněji zahrneme výraz „operační systém“.

The ACM[1] Computing Classification System [21] byl vytvořen



Obrázek 5.2: Ukázka části doménové ontologie ACM

v roce 2012, a jak název napovídá, jedná se o doménovou ontologii v informatice. Obsahuje přes 2500 pojmů a dodržuje specifikaci RDF definující reprezentaci dat ve formě (objekt—atribut—hodnota).

Ontologie je napsaná v anglickém jazyce. Vzhledem k tomu, že stažené předměty v informatické kolekci jsou v češtině, bylo třeba ontologii přeložit. Bohužel jsem nenašel žádný přesný anglicko-český terminologický slovník, který by byl dostupný online. Zvolil jsem k přeložení některých termínů internetovou encyklopedii Wikipedia. Termíny, které neměly českou stránku na Wikipedii, jsem poté přeložil pomocí aplikačního rozhraní od Bing překladače.

[1]ACM — Association for Computing Machinery je mezinárodní organizace podporující rozvoj a studium informatiky

Tabulka 5.1: Počet informatických oborů na fakultách univerzit s počtem stažených předmětů

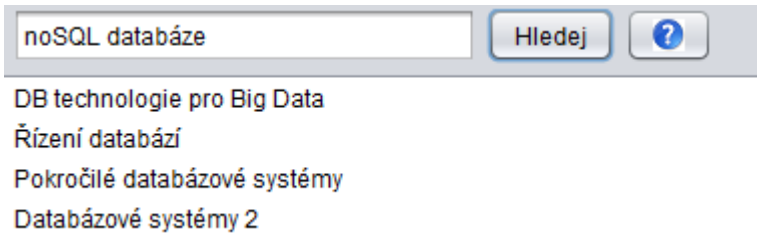
Název univerzity	**Počet oborů**	**Počet předmětů**
Masarykova univerzita Brno	30	162
Univerzita Karlova	40	244
České vysoké učení technické v Praze	27	315
Vysoké učení technické v Brně	20	311
Západočeská univerzita	27	348
Univerzita Pardubice	12	194
Technická univerzita Ostrava	16	504
Univerzita Tomáše Bati ve Zlíně	15	403
Vysoká škola ekonomická v praze	11	58
Technická univerzita Liberec	11	202

5.4 Zhodnocení

Pro vytvoření kolekce jsem vybral všechny studijní obory, které byly na seznamu akreditovaných programů Ministerstva školství ČR. Vybrané studijní programy musely obsahovat v názvu jeden z těchto řetězců: informatika, programování, softwarové, informační. Internetový robot poté prošel všechny tyto vybrané obory a stáhl jejich předměty ve studijním plánu. Výsledkem bylo 2743 stažených předmětů. Na prvním místě s největším počtem nabízených předmětů se umístila Vysoká škola báňská. Druhá skončila univerzita Tomáše Bati ve Zlíně a třetí je Západočeská univerzita s 348 předměty. Nicméně počet předmětů není důležitým faktorem, protože je ovlivněn nabídkou volitelných předmětů uvedených u studijního plánu. Tento závěr dosvědčuje nejmenší počet předmětů u Vysoké škole ekonomické. VŠE ve svých studijních plánech nabízí pouze přehled povinných a povinně volitelných předmětů. Nabídka volitelných předmětů je zobrazena odděleně.

5.4.1 Ověření funkčnosti

Pro ověření funkčnosti dotazování na obsah výuky jsem implementoval jednoduchý vyhledávací algoritmus. Ten vrací seznam všech předmětů obsahujících dvě a více slov v hledaném výrazu. Pro svou ukázku jsem vybral termín *noSQL databáze*. Tento databázový koncept ukládání dat v poslední době vzrůstá na popularitě zejména v oblastech Big Data a real-time webů. Jedná se o vzorovou situaci použití, která by studentovi pomohla se rozhodnout při výběru oboru nebo zápisu daného předmětu. Výsledek hledání (Obrázek 5.3) na odpovídající dotaz zobrazil 4 předměty: DB technologie pro Big Data vyučovaný na ČVUT, Řízení databází z univerzity v Liberci, Pokročilé databázové systémy na VUT a databázové systémy 2 na Západočeské univerzitě.



Obrázek 5.3: Výsledek hledání v kolekci

5.4.2 Doba běhu

Procházení stránek, zpracování a uložení kolekce probíhalo 33 minut na počítači s procesorem i5-3210M a 4 GB RAM. Navštěvování stránek bylo prováděno ve více vláknech se zvoleným parametrem 3 vteřiny zpoždění na další požadavek pro stránku. Při zahrnutí dosažených výsledků můžeme snadno zjistit, že samotné čekání na získání další stránky trvalo drtivou většinu času. U Vysoké školy báňské to bylo odhadem přes 600 zobrazených stránek (z toho 506 stránek s předměty), což činí čekací dobu 30 minut.

Můžeme ještě porovnat dobu trvání lemmatizace slovníkem proti knihovně Lucene. Lemmatizace knihovnou Lucene běžela 5 sekund oproti slovníkové metodě, která trvala 25 vteřin.

Načtení kolekce pro vizualizaci trvá v jednotkách sekund a to samé platí pro vyhledávání.

6 Závěr

Nejprve byla prozkoumána struktura a forma webových stránek českých univerzit. Na základě analýzy stránek se určily informace o předmětech, které dostatečně informují studenta o jeho obsahu. Jako společné údaje se vybraly: název, obsah, obory, cíl, předpoklady, vyučující, literatura, předpoklady, semestr, počet kreditů a odkaz na předmět. Podle těchto informací byla navržena struktura XML souboru pro jejich uložení. K ověření správné struktury XML bylo specifikováno XML schéma.

Dále byly prostudovány technologie a možnosti stahování webový stránek. Při výběru byl kladen důraz na jednoduchost, aktuálnost a podporu moderních technologií. Pro účel této práce byla použita knihovna *jsoup* napsaná v jazyce Java. Z analýzy stránek byl také zvolen způsob procházení internetovým botem. Z důvodu přesného získání informací internetový bot navštěvoval stránky pomocí šablon.

Posledním úkolem bylo stažené předměty zpracovat pro účely dotazování. Pro dosažení relevantní odpovědi byla stažená kolekce lemmatizována. Byly vyzkoušeny dva způsoby lemmatizace. Vhodnější metodu lemmatizace obsahuje knihovna Lucene. Budoucí možnosti vyhledávání se ještě zlepšily přidáním přeložené doménové ontologie.

Výsledkem práce je funkční aplikace. Díky této aplikaci se podařilo stáhnout informace o předmětech z 10 vybraných univerzit a z nich vytvořit kolekci dat obsahujících 2743 předmětů. Pro ověření funkčnosti dotazování byl implementován prototyp vyhledávání.

6.1 Navrhovaná vylepšení

Základním cílem práce bylo vytvořit kolekci předmětů. Nabízí se pokračovat v zlepšení vyhledávacího algoritmu. Navázat by se dalo vytvořením služby, která by studentům usnadnila výběr oborů podle zadaných kritérií. Snadné by bylo rozšíření na stahování všech ostatních předmětů, které nejsou jenom ve studijním plánu informatických oborů.

Literatura

[1] *What is HTML*. [online]. [cit. 2015-04-10]. Dostupné z: http://www.w3.org/TR/html401/intro/intro.html#h-2.2

[2] *CSS* [online]. [cit. 2015-04-10]. Dostupné z: http://www.w3.org/standards/webdesign/htmlcss

[3] *CSS selectors* [online]. [cit. 2015-04-10]. Dostupné z: http://www.w3schools.com/cssref/css_selectors.asp

[4] HEROUT, Pavel. *Java a XML*. 1. vyd. České Budějovice: Kopp, 2007. 313 s. ISBN 978-80-7232-307-4.

[5] Hospodářské noviny. *Žebříčky VŠ* [online]. [cit. 2014-04-15]. Dostupné z: http://archiv.ihned.cz/c1-63417690-informatika-se-nejlepe-vyucuje-na-cvut

[6] Webometrics. *Ranking web of universities Žebříčky VŠ* [online]. [cit. 2014-04-15]. Dostupné z: http://www.webometrics.info/en/Europe/Czech%20Republic

[7] U-Multirank. *Universities compared.*[online]. [cit. 2014-04-15]. Dostupné z: http://www.umultirank.org

[8] Google crawlers [online]. [cit. 2015-04-15]. Dostupné z: https://support.google.com/webmasters/answer/1061943?hl=en

[9] Difference between Spider,Crawler and Robot [online]. 7.4.2009, [cit. 2015-04-15]. Dostupné z : http://forums.seochat.com/search-engine-spiders-27/difference-between-spider-crawler-robot-244471.html

[10] KOSTER, M. Guidelines for Robots Writers [online]. 1993, [cit. 2015-04-15]. Dostupné z : http://www.robotstxt.org/guidelines.html

[11] Ronny Lempel, Type of crawlers [online]. 2006, [cit. 2015-04-15]. Dostupné z : http://webcourse.cs.technion.ac.il/236620/Winter2006-2007/ho/WCFiles/lec14-crawlers.PDF

[12] Junghoo Cho, Synchronizing a database to Improve Freshness, [online]. 1999-09-25, [cit. 2015-04-16]. Dostupné z : http://oak.cs.ucla.edu/~cho/papers/cho-synch.pdf

[13] KUSHMERICK, Nicholas, *Wrapper induction for Information Extraction*. [online]. [cit. 2015-04-20]. Dostupné z : http://www.isi.edu/info-agents/courses/iiweb/kushmerick-ijcai97.pdf

[14] MITCHELL, Tom. *Machine Learning*. McGraw-Hill 1997, ISBN 0070428077

[15] ZHAI, Yanhong. *Extracting Web Data Using Instance-Based Learning* [online]. [cit. 2015-05-01]. Dostupné z : http://www.dbai.tuwien.ac.at/education/wie/SS06/papers/WISE05-instance-extract.pdf

[16] HAYASHI, Yoshihiko. *Searching Text-rich XML Documents with Relevance Ranking* [online]. 2000-06-14, [cit. 2015-05-01]. Dostupné z https://www.research.ibm.com/haifa/sigir00-xml/final-papers/Hayashi/hayashi.html

[17] PYTELKA, Petr. *Jak kvalita lemmatizace ovlivňuje výsledky vyhledávání dokumentů v českém jazyce*. [online]. [cit. 2015-05-01]. Dostupné z : https://www.vse.cz/vskp/34929_jak_kvalita_lemmatizace_ovlivnuje_vysledky_vyhledavani_dokumentu_v%C2%A0ceskem_jazyce

[18] BRUCHÝN, Tomáš. *High Precision Stemmer* [online]. [cit. 2015-05-01]. Dostupné z : http://liks.fav.zcu.cz/HPS/article.pdf

[19] HELLERBRAND, David. Nalezení slovních kořenů v češtině. [online]. 2010, [cit. 2015-04-26]. Dostupné z : http://ceur-ws.org/Vol-802/paper6.pdf

[20] PÁRALIČ, Ján. *Dolovanie znalostí z textov*. Technická univerzita v Košiciach. 2010. ISBN 978-80-89284-62-7

[21] ACM Computing Classification System. [online]. [cit. 2015-04-26]. Dostupné z : http://www.acm.org/about/class/2012

A Uživatelská dokumentace

A.1 Sestavení a spuštění

Před spuštěním programu zkontrolujte, zda máte nainstalovanou Javu. Pro samotné spuštění napište do příkazové řádky v adresáři, kde se soubor nachází

```
java -jar A11B0504P_bakalarska_prace.jar
```

nebo soubor můžete otevřít dvojklikem.

A.2 Stahování

Na obrázku A.1 vidíme tlačítko se jménem univerzity reprezentující šablonu, podle které se stahují předměty. Editaci šablony otevřeme kliknutím na tlačítko a po provedených změnách stiskneme uložit. Pro stahování můžeme nastavit čas, po kterém robot čeká na další navštívení stránky na jedné doméně. Druhým možným nastavením je doba čekání na přijetí odpovědi. Pokud nepřijde od stránky odpověď do stanovené doby, stránka se vyhodnotí jako nenavštívená. Základní hodnoty jsou 3 a 10 sekund. Zahájení stahování se provede stisknutím tlačítko stáhnout vše.

Stisknutím tlačítka *Nový výběr univerzit* můžeme změnit standardní výběr univerzit. Označíme šablony, které chceme přidat (v případě označení více předmětů držíme CTRL), a výběr potvrdíme tlačítkem *Open*.

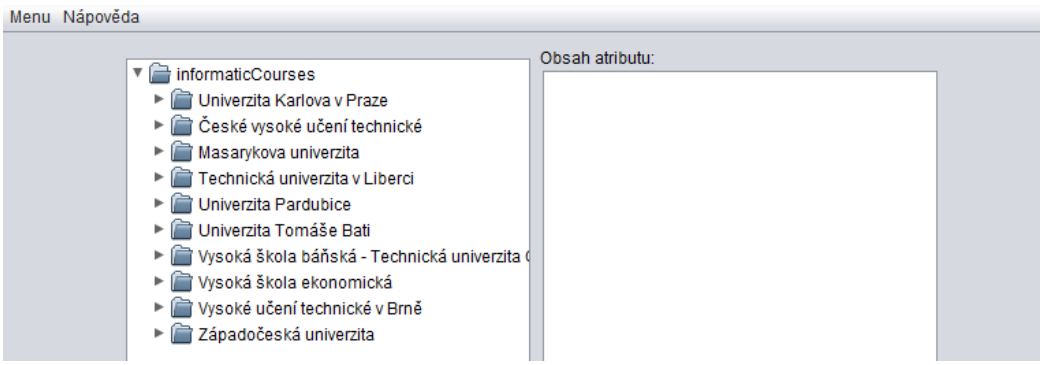
Po dokončení stahování budete informování dialogovým oknem. Stažená kolekce se uloží do souboru *file.txt*.

Obrázek A.1: Úvodní obrazovka poskytuje nastavení pro stažení předmětů

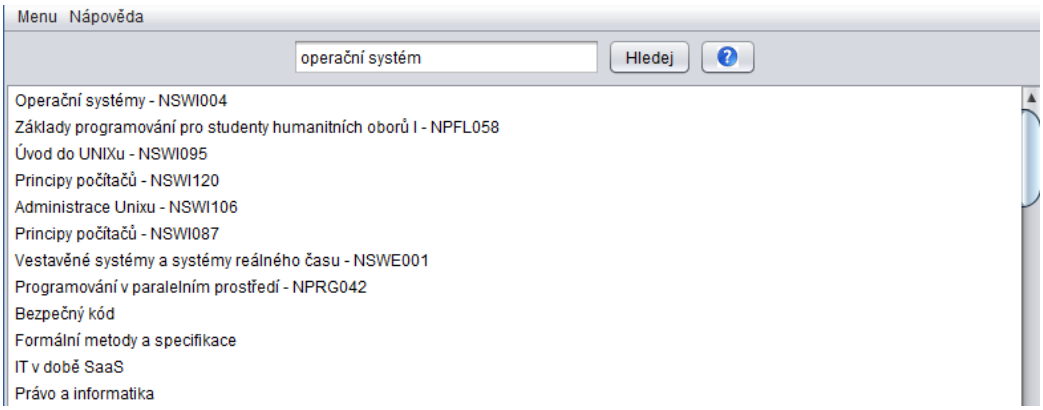
A.3 Procházení a vyhledávání

Po zvolení XML soubor v nabídce menu se zobrazí Obrázek A.2. XML soubor můžete procházet jako stromovou strukturu a po kliknutí na jednotlivé elementy jeho obsah v levém panelu

Po zvolení Hledat předmět v nabídce menu se zobrazí Obrázek A.3. V této ukázce se zadalo do vyhledávání operační systém a můžeme vidět výsledky dotazu. Pokud chceme vědět o předmětu více dvojklikem se otevře v prohlížeči stránka s detailními informacemi o předmětu.



Obrázek A.2: Procházení kolekce



Obrázek A.3: Vyhledávání předmětů v kolekci