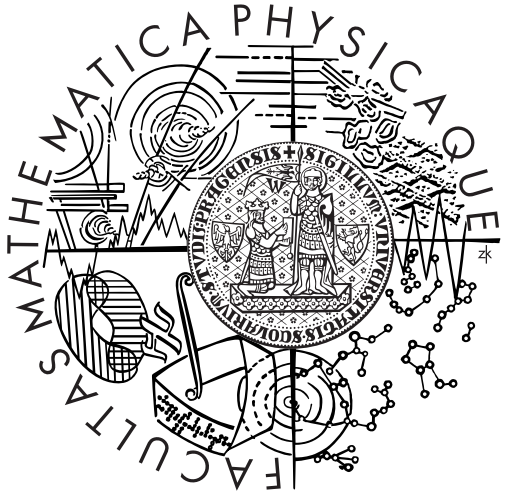
Univerzita Karlova v Praze
Matematicko-fyzikální fakulta



Bakalářská práce

Andrej Koprivňanský

Vyhledávač pro dmoz.org

Katedra softwarového inženýrství

Vedoucí bakalářské práce:

RNDr. Leo Galamboš, Ph.D.

Studijní program: Informatika, Správa počítačových systémů

2006

Na tomto mieste by som rád poďakoval vedúcemu Bakalárskej práce, pánovi RNDr. Galambošovi Ph.D., za cenné rady a podporu pri písaní práce. Tiež by som mu chcel poďakovať za poskytnutie projektu Egothor a úpravy, ktoré v ňom urobil pre moje potreby.

Prehlasujem, že som svoju bakalársku prácu napísal samostatne a výhradne s použitím citovaných prameňov. Súhlasím s požičaním práce.

V Prahe dňa 15.12.2006 Andrej Koprivňanský

Obsah

Názov práce: Vyhľadávač pre dmoz.org
Autor: Andrej Koprivňanský
Katedra (ústav): Katedra softwarového inženýrství
Vedúci bakalárskej práce: RNDr. Leo Galamboš, Ph.D.
e-mail vedúceho: leo.galambos@mff.cuni.cz

Abstrakt: dmoz.org je v súčasnosti najväčší otvorený katalóg internetových stránok spravovaný dobrovoľnými editormi. Dáta obsiahnuté v tomto katalógu sú voľne k dispozícii verejnosti na jeho internetových stránkach. Cieľom mojej práce je navrhnúť a implementovať fulltextový vyhľadávací stroj pre katalógovú databázu dmoz.org. V práci riešim problematiku spracovania súborov s dátami z tohoto katalógu do formy portálu prezeratelného internetovým prehliadačom s možnosťou vyhľadávania v kategóriach. Pretože sa jedná o rýchle meniacu sa databázu, treba tiež vyriešiť jej efektívnu aktualizáciu.

Kľúčové slová: dmoz, katalóg, vyhľadávací stroj, index

Title: Search Engine for dmoz.org
Author: Andrej Koprivňanský
Department: Department of software Engineering
Supervisor: RNDr. Leo Galamboš, Ph.D.
Supervisor's e-mail address: leo.galambos@mff.cuni.cz

Abstract: dmoz.org is the largest human-edited directory of the Web. It is possible to get database of this directory by the public from the project web page. The main purpose of my work is to design and implement fulltext search engine for the directory database dmoz.org. In my work I deal with directory database processing that will make the web portal readable by any web browser. There will be also possible to search in categories of the directory. It is necessary to implement effective updating of the portal because of often changes in the directory database.

Keywords: dmoz, directory, search engine, index

Kapitola 1

Úvod

1.1 Cieľ

Hlavným cieľom mojej práce je vytvoriť program, ktorý umožní z dát získaných z otvoreného internetového katalógu dmoz.org ([1]) vytvoriť portál, ktorý bude umožňovať vyhľadávať v kategóriach. Ďalej bude umožňovať efektívne aktualizovať tento vytvorený portál z aktuálnych dát. Pretože sa jedná o rýchle meniacu sa databázu, je nutné vyriešiť predovšetkým (re)indexačnú fázu.

1.2 Katalóg dmoz.org

Katalóg dmoz.org ([1]) je v súčasnosti najväčší otvorený katalóg internetových stránok. Jeho obsah spravujú dobrovoľní editori patriaci do komunity tohto portálu. Editori majú rozdelené jednotlivé kategórie katalógu a rozhodujú o tom, ktoré stránky v danej kategórii budú zverejnené. Katalóg je už v dnešnej dobe pomerne rozšírený. Vyhľadávajú v ňom najväčšie internetové vyhľadávače, ako sú Google, Yahoo, AOL Search. Dáta katalógu sú voľne k dispozícii na internetových stránkach [1] vo forme dvoch skomprimovaných xml súborov `structure.rdf.u8.gz` a `content.rdf.u8.gz`. Prvý obsahuje informácie o kategóriach a ich štruktúre. Druhý obsahuje informácie o stránkach patriacich k jednotlivým kategóriám.

1.3 Funkčné požiadavky

Program musí byť schopný z dát z katalógu dmoz.org vytvoriť portál, pomocou ktorého bude možné prezerať katalóg internetových stránok bežným internetovým prehliadačom. Užívateľ si bude môcť zvoliť kategórie, z ktorých sa portál vytvorí.
Vytvorený portál umožní užívateľovi vyhľadávať v kategóriach. Užívateľ bude mať 3 možnosti vyhľadávania.

1. Vyhľadávať vo všetkých kategóriach.
2. Vyhľadávať vo všetkých podkategóriach kategórie, z ktorej bol dotaz zadaný.
3. Vyhľadávať vo všetkých podkategóriach kategórie, z ktorej bol dotaz zadaný s prehľadávaním kategórií, na ktoré vedie cesta prostredníctvom symlinkov.

Program tiež musí umožniť efektívne aktualizovať existujúci portál aj s dátami, v ktorých sa bude vyhľadávať.

1.4 Alternatívne projekty

Alternatívny projekt, ktorý by pokryl všetky naše funkčné požiadavky, v čase implementácie, nebolo možné nájsť. Existujú ale projekty, ktoré implementujú aspoň ich časť.

1.4.1 JWebDirectory

V dobe našej implementácie projekt ešte nevyšiel. Na internetových stránkach projektu [8] je ale možné sa dočítať, že autori sľubujú rýchle vytváranie portálu s možnosťou filtrovania kategórií a vyhľadávanie s využitím vyhľadávacieho stroja Lucene ([18]). Neimplementuje ale vyhľadávanie v symlinkoch a aktualizáciu portálu s indexom pre vyhľadávač.

1.4.2 Dmoz Extractor

Dmoz Extractor je komerčný projekt, ktorý umožňuje z dátových súborov z katalógu dmoz.org vybrať požadované kategórie a uložiť ich v rôznom formáte. Spolu s ďalším komerčným projektom INDEXU umožňuje vytvoriť

vyhľadávanie v kategóriach. Podporuje ale iba malé množstvo kategórií a to menej ako 10 000.
Ďalšie informácie o tomto projekte je možné nájsť na jeho internetových stránkach [9].

1.4.3 ODP Data Parser

Ide o sadu skriptov napísaných v jazyku perl, ktoré uložia informácie z katalógu dmoz.org do relačnej databázy MySQL. Nad touto databázou je možné implementovať vyhľadávanie, ktoré je ale pri väčších dátach značne pomalé. Ďalšie informácie o tomto projekte je možné nájsť na internetových stránkach [10].

Kapitola 2

Analýza a návrh

2.1 Vstupné dáta

2.1.1 Charakteristika

Vstupné dáta pre program získame z internetovej stránky `http://rdf.dmoz.org/rdf`. Nachádzajú sa tam aktuálne, ale aj staršie verzie súborov, v ktorých sú uložené dáta pre katalóg dmoz.org. Nás budú zaujímať súbory `structure.u8.rdf.gz` a `content.u8.rdf.gz`.
Súbor `structure.u8.rdf.gz` je skomprimovaný GNU zipom XML súbor vo formáte RDF. Obsahuje informácie o štruktúre kategórií. Každá kategória obsahuje jednoznačný číselný identifikátor, popis, zoznam podkategórií, symlinky odkazujúce do podobných kategórií a ďalšie popisujúce údaje. Poradie, v akom sú kategórie vypísané, je odpovedajúce prechádzaniu do hĺbky stromom, ktorý vytvárajú. V súčasnej dobe obsahuje viac ako 700 000 kategórií a pribúdajú stále nové. Jeho veľkosť po dekomprimovaní je viac ako 600 MB.
Súbor `content.rdf.u8.gz` je podobne ako prvý menovaný skomprimovaný XML súbor v RDF formáte. Obsahuje zoznam adries internetových stránok patriacich ku kategóriam. Poradie kategórií v tomto súbore je rovnaké ako v súbore `structure.rdf.u8.gz`. V súčasnej dobe obsahuje spolu viac ako 4 800 000 adries na externé internetové stránky. Jeho veľkosť po dekomprimovaní je viac ako 2 GB.
Obidva súbory sa pravidelne každý mesiac aktualizujú.

2.1.2 Spracovanie

Súbory, ktoré dostane program na vstup, sú vo formáte XML, preto môžeme použiť vhodný XML parser. Do operačnej pamäte sa nám nezmestia, preto

nie je možné použiť DOM XML parser. Súbory nám bude stačiť prechádzať sekvenčne a nebudeme potrebovať do nich zapisovať. Tieto kritériá spĺňajú dva XML parsery StAX ([13]) a SAX ([14]). V projekte použijeme StAX parser, lebo jeho použitím sa sprehľadní kód a rýchlosťou je porovnateľný so SAX parserom.
Ďalej môžeme využiť toho, že súbory sú skomprimované, čím sa zníži počet I/O operácií na úkor využitia procesoru. Preto necháme na užívateľovi, aby si mohol zvoliť, či ich chce použiť skomprimované alebo dekomprimované.

2.2 Vytvorenie portálu

Portál, ktorý chceme vytvoriť, má formát katalógu, preto môžeme jeho štruktúru využiť ako štruktúru súborov a adresárov na súborovom systéme. Adresáre budú predstavovať kategórie a súbory v adresároch budú obsahovať stránku s príslušnými odkazmi pre danú kategóriu. Ako formát súborov bude vhodné použiť XML. Tento formát umožní jednoducho definovať formát, v akom sa stránka zobrazí užívateľovi pomocou technológií xslt ([16]) a css ([15]). Výhodou tohoto formátu je aj jednoduché parsovanie, čo uľahčí prácu pri aktualizácii.
Kategórie môžu byť uvedené v rôznych jazykoch, preto treba vyriešiť problém s ich pomenovaním. Názvy kategórií môžu obsahovať znaky, ktoré nepatria do ASCI. Jedna možnosť je použiť tieto znaky. Ale vzhľadom k tomu, že nie všetky súborové systémy podporujú diakritiku, bude vhodnejšie prevádzať tieto znaky do ASCI.
Ďalší problém môže byť s dĺžkou názvu kategórie. Väčšina súborových systémov podporuje dĺžku názvu do 256 znakov, preto budeme musieť dlhšie názvy skrátiť. Keby sme len vynechali posledné znaky, mohlo by dôsť ku konfliktom v názvoch v rámci adresára. Preto na tieto dlhé názvy použijeme nejakú hešovaciu funkciu, ktorú nám poskytnú knižnice použitého programovacieho jazyka. Aby sme mohli efektívne aktualizovať portál a index pre vyhľadávač, musíme byť schopní rýchlo zistiť, či boli zmenené údaje o kategórii. Jedna možnosť, ako zistiť, či bola kategória zmenená, je načítať si údaje o nej z existujúceho súboru v portáli a porovnať ich s údajmi z aktuálneho zdrojového súboru. Takýmto spôsobom by ale aktualizácia portálu trvala podstatne dlhšie, ako jeho vytvorenie, lebo by nám pribudli I/O operácie na nájdenie súboru s kategóriou a jeho načítanie. Preto si budeme ukladať údaje o každej kategórii vo forme md5 kódu v pomocnej dátovej štruktúre, ktorú budeme mať uloženú v súbore pre ďalšie použitie. Tieto súbory si pripravíme dva. Jeden bude obsahovať md5 kódy o všetkých údajoch každej kategórie. Tento súbor budeme používať pri aktualizácii portálu. V druhom súbore budeme

mať uložené iba md5 kódy údajov o kategórii, ktoré budeme používať pri jej indexovaní.

2.2.1 Algoritmus vytvorenia portálu

Na vytvorenie portálu využijeme štruktúru súborov z katalógu dmoz.org. Kategórie v súboroch sú radené prechádzaním do hĺbky stromom, ktorý tvoria. To nám umožní prechádzať súbor sekvenčne a postupne vytvárať adresárovú štruktúru zhora nadol. Poradie kategórií v obidvoch súboroch je rovnaké, čo nám zase umožní čítať obidva súbory súbežne po kategóriach.

Kroky algoritmu:

1. Načítame kategóriu s jej údajmi zo súboru `structure.u8.rdf.gz`.
2. Vytvoríme adresárovú štruktúru podľa podkategórií načítanej kategórie.
3. Načítame ku kategórii z kroku 1 odkazy na externé stránky s ich popismi zo súboru `content.u8.rdf.gz`.
4. Vytvoríme DOM dokument z načítanej kategórie a ten transformujeme do výsledného súboru `index.xml` na cestu vytvorenú z jej plného názvu. Vytvorený súbor obsahuje informácie o kategórii a zároveň obsahuje všetky odkazy na externé stránky vzťahujúce sa k tejto kategórii.
5. Zistíme si md5 kód pre všetky údaje kategórie a uložíme do pomocnej dátovej štruktúry
6. Zistíme si md5 kód pre údaje kategórie, ktoré sa použijú pri vytváraní indexu a uložíme ho do pomocnej dátovej štruktúry.
7. Ak už žiadna ďalšia kategória neexistuje, uložíme pomocné dátové štruktúry s md5 kódmi kategórií do súborov a algoritmus ukončíme. Inak pokračujeme krokom 1.

2.3 Vyhľadávanie v portáli

Na každej stránke v portáli bude formulár na zadanie dotazu pre vyhľadávač. Užívateľ si bude môcť zvoliť, či chce vyhľadávať v celom portáli, vo všetkých podkategóriach aktuálnej kategórie, alebo či sa má vyhľadávať aj v symlinkoch z podkategórií. Vyhľadávať sa bude v názvoch kategórií a v ich

popisoch.
Na vyhľadávanie v portáli použijeme vyhľadávací stroj Egothor vytvorený pánom RNDr. Galambošom., Ph.D.. Tento nástroj nám umožní efektívne vyhľadávať podľa požadovaných kritérií. Ďalšou úlohou bude pripraviť mu dáta v potrebnom formáte.

2.3.1 Popis fungovania vyhľadávacieho stroja Egothor

Egothor je vyhľadávací stroj určený na akademické použitie. Umožňuje efektívne indexovať dokumenty, následne v nich vyhľadávať a efektívne rieši aj aktualizáciu indexu. Spĺňa všetky naše požiadavky, preto ho použijeme v projekte. Ďalšou možnosťou bolo použiť open source projekt Lucene ([18]), ktorý ale výkonom zaostáva za Egothorom, hlavne po prevedení väčšieho počtu aktualizácií.
Vstupom pre Egothor sú dokumenty s metadátami a prúdom slov, nad ktorými sa vytvára index. Metadáta dokumentu sú tvorené názvom dokumentu, jeho popisom a cestou, kde sa na súborovom systéme indexovaný dokument nachádza. Prúd indexovaných slov je možné určiť z metadát, prípadne pridať dalšie informácie o dokumente pomocou parametrov. Z prúdu slov si Egothor vytvára takzvané tokeny, ktoré určujú, ako sa má k danému slovu správať pri indexovaní a vyhľadávaní. Je možné použiť aj špeciálne typy tokenov, ako je napríklad `<BITMAP>`, ktorý zariadi uloženie slova v indexe vo forme bitmapy. To nám umožní rýchlo vyhľadávať dokumenty, ktoré obsahujú rovnaké slovo, a nezáleží na počte jeho výskytov.
Egothor tiež umožňuje aktualizovať dokumenty, nad ktorými má vytvorený index. Dá sa to zariadiť odstránením dokumentu z indexu a následným vložením s aktualizovanými metadátami.

2.3.2 Vyhľadávanie v kategóriach

Aby sme vedeli pri dotazovaní určiť, v ktorých kategóriach sa má vyhľadávať, potrebujeme nejakým spôsobom dokumenty podľa nich rozlíšiť. Ďalej budeme chcieť, aby vo výsledku pre vyhľadávanie v kategórii boli aj výsledky pre podkategórie. To by sa dalo zariadiť napríklad tak, že by sme si ku každému dokumentu ukladali plný názov kategórie aj s jej nadradenými kategóriami. Potom pri dotazovaní by stačilo hľadať všetky dokumenty, ktoré obsahujú slovo s prefixom plného názvu hľadanej kategórie. Toto riešenie by bolo ale značne pomalé. Preto použijeme špeciálny typ tokenov `<BITMAP>`, ktoré Egothor poskytuje. Pomocou týchto tokenov si Egothor uloží informácie o kategórii do bitmapy, čím sa urýchli vyhodnocovanie dotazu na výskyt dokumentu v kategórii. Ku každému dokumentu budeme pridávať tieto to-

keny pre kategóriu, v ktorej sa vyskytuje, ale aj pre jej nadradené kategórie. Tým sa zariadi, aby bol dokument nájdený, aj ked sa vyskytuje v podkategórii hľadanej kategórie.

2.3.3 Použitie vyhľadávacieho stroja Egothor v projekte

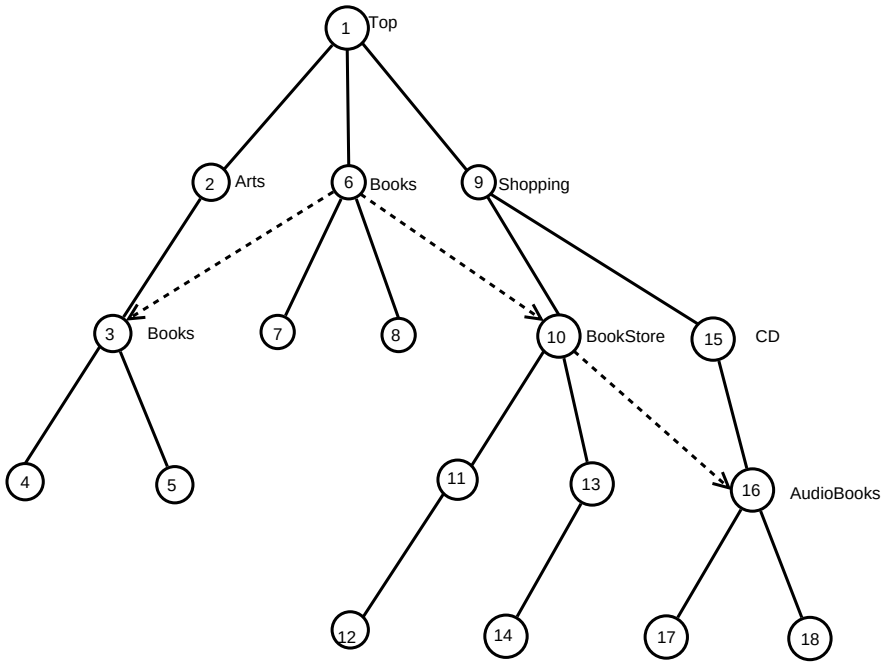
Najkôr budeme musieť pripraviť dáta, nad ktorými spraví Egothor index pre vyhľadávanie. Budeme potrebovať pre každú kategóriu uložený popis, názov a cestu, kde v portáli sa nachádza súbor so stránkou. Na tieto údaje si vystačíme s metadátami, ktoré nám ponúka Egothor. Ďalej budeme musieť určiť, z ktorých údajov sa bude vytvárať prúd slov do indexu. Bude treba vyhľadávať v názvoch a popisoch, preto z týchto údajov vytvoríme prúd slov. K týmto slovám dalej pridáme špeciálne tokeny `<BITMAP>` pre kategórie a symlinky. Takto pripravený dokument môžeme predať Egothoru na spracovanie do indexu.

2.3.4 Príprava `<BITMAP>` tokenov pre kategórie

Zoberieme si napríklad kategóriu /Shopping/CD/AudioBooks z obrázku 2.1. Táto kategória by sa mala prehľadávať na dotaz, pri ktorom sa vyhľadáva vo všetkých podkategóriach kategórie /Shopping, /Shopping/CD a /Shopping/CD/AudioBooks. Preto budeme musieť vytvoriť `<BITMAP>` tokeny `<BITMAP>shopping`, `<BITMAP>shopping.cd` a `<BITMAP>shopping.cd.audiobooks` a pridať ich do dokumentu pre kategóriu /Shopping/CD/AudioBooks. Takýmto spôsobom budeme vytvárať `<BITMAP>` tokeny pre všetky kategórie. Pri dotazovaní bude stačiť zadať dotaz na hľadané slovo a pridať k nemu token `<BITMAP>` pre hľadanú kategóriu.

2.3.5 Príprava `<BITMAP>` tokenov pre symlinky

Podobne ako pre určenie kategórií použijeme tokeny `<BITMAP>`. Ku každému dokumentu pridáme k tokenom nesúcim informáciu o jeho kategórii a nadradených kategóriach ďalšie tokeny s informáciou o všetkých symlinkoch, cez ktoré vedie k danému dokumentu cesta. Tieto tokeny budú mať špeciálny prefix symlink, podla ktorého bude možné pri dotazovaní rozlíšiť, že chceme hľadať aj v symlinkoch.
Konkrétne pre kategóriu /Shopping/CD/AudioBooks z obrázku 2.1 to znamená, že potrebujeme zistiť všetky kategórie, z ktorých do nej vedie cesta



Obr. 2.1: Príklad štruktúry kategórií

prostredníctvom symlinkov. V tomto prípade sú to kategórie /Books a /Shopping/BookStore. Preto do indexu pridám tokeny `<BITMAP>symlink.books`, `<BITMAP>symlink.shopping.bookstore` a `<BITMAP>symlink.shopping` pre vyhľadávanie v symlikoch odkazujúcich z podkategórií.

2.3.6 Algoritmus hľadania kategórií pre symlinky

Ku každej kategórii je treba určiť, z ktorých kategórií do nej vedie cesta prostredníctvom symlinkov. Preto si potrebujeme vytvoriť dátovú štruktúru, ktorá bude obsahovať zoznam kategórií a ku každej kategórií zoznam symlinkov. Z tejto dátovej štruktúry si vieme pre každú kategóriu určiť, do ktorých kategórií vedie cesta prostredníctvom symlinkov a to prehľadávaním do hĺbky, pri ktorom si budeme pamätať cez ktoré kategórie sme prechádzali. Ku každej navštívenej kategórii si budeme ukladať informáciu o tom, cez ktoré kategórie sme sa do nej dostali. Všetky tieto dátové štruktúry budú zaberať naraz veľa miesta v operačnej pamäti, preto sa budeme snažiť rozdeliť jednotlivé operácie tak, aby sme si musel pamätať minimálne množstvo údajov. Ďalšie možné ušetrenie pamäte je vo formáte uloženia symlinkov. Vo vstupnom súbore sú symlinky vypísané

prostredníctvom plného názvu kategórie s nadradenými kategóriami, na ktorú symlink odkazuje. Keďže každá kategória má svoj unikátny číselný identifikátor, môžeme ho použiť pri ďalších výpočtoch. Teraz popíšeme jednotlivé kroky algoritmu podľa obrázku 2.2.

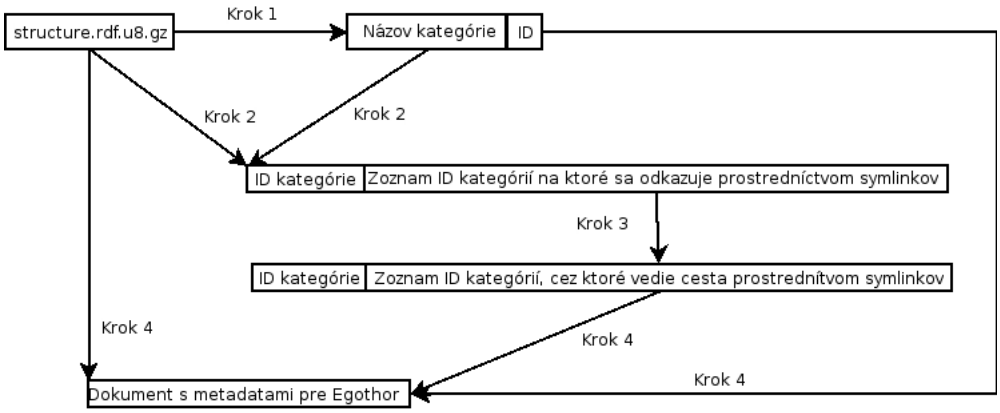
1. V prvom kroku si zo vstupného súboru načítame zo všetkých kategórií číselné identifikátory s plným názvom kategórie. Budeme si ich ukladať do dátovej štruktúry, ktorá nám umožní rýchly prístup k číselnému identifikátoru kategórie prostredníctvom jej plného názvu.

2. V druhom kroku si vytvoríme dátovú štruktúru, v ktorej si budeme ukladať identifikátory kategórií a k nim identifikátory kategórií na ktoré priamo vedie z danej kategórie symlink. Vytvoríme ju tak, že zo vstupného súboru budeme postupne čítať kategórie, pri ktorých sú uvedené symlinky plným názvom kategórie na ktorú sa odkazuje. Na prevod názvu do číselného identifikátora použijeme dátovú štruktúru z kroku 1.

3. V treťom kroku vytvoríme dátovú štruktúru obsahujúcu ako kľúč identifikátor kategórie a ako hodnotu zoznam identifikátorov kategórií, z ktorých vedie cesta do uvažovanej kategórie prostredníctvom symlinkov. V tejto fáze už ale nepotrebujeme dátovú štruktúru z kroku 1, preto si ju odložíme do súboru na ďalšie použitie. Na jej vytvorenie použijeme dátovú štruktúru z kroku 2. Postupne budeme prechádzať všetky záznamy a algoritmom prechádzania do hĺbky hľadať všetky možné cesty vedúce cez symlinky. Budeme pri tom kontrolovať, či nám v ceste nevznikajú cykly. Do novej dátovej štruktúry si budeme ukladať ku každej kategórií cez ktoré kategórie sme sa do nej dostal.

4. Vo štvrtom kroku budeme pre každú kategóriu hľadať názvy kategórií cez ktoré vedie do nej cesta prostredníctvom symlinkov. Na výpočty už nebudeme potrebovať dátovú štruktúru z kroku 2, preto ju uložíme do súboru. Ďalej budeme potrebovať z číselného identifikátoru určiť plný názov kategórie. Preto si načítame dátovú štruktúru z kroku 1, ktorá obsahuje potrebné informácie. Postupne budeme načítávať kategórie zo vstupného súboru a z dátovej štruktúry z kroku 3 budeme vedieť, cez ktoré kategórie je možné sa do uvažovanej kategórie dostať prostredníctvom symlinkov.

Napriek tomu, že sme si výpočty rozdelili do viacerých krokov, pri ktorých si nedržíme všetky dáta v operačnej pamäti, pri vytváraní indexu



Obr. 2.2: Kroky algoritmu

nad všetkými kategóriami pamäťové nároky pomerne dosť narastú. Potrebná pamäť môže prekročiť 2GB. Pri výpočtoch najviac miesta v pamäti zaberá dátová štruktúra z kroku 3. Preto by bolo možné použiť na ňu aspoň čiastočne disk. Toto by ale značne spomalilo naše výpočty. Preto dáme možnosť užívateľovi obmedziť hĺbku prehľadávania symlinkov, čo výrazne zmenší veľkosť dátovej štruktúry z kroku 3. Na použitie disku na pomocné výpočty aplikáciu pripravíme vhodnými rozhraniami v implementácii, aby toto riešenie nebolo zložité doimplementovať v prípade potreby.

2.4 Aktualizácia portálu

Po stiahnutí aktuálnych súborov z katalógu dmoz.org bude možné spraviť aktualizáciu vytvoreného portálu.

Algoritmus aktualizácie portálu:

1. Načítame si pomocnú dátovú štruktúru s md5 kódmi pripravenými z údajov o kategóriach, ktorú sme si uložili pri vytváraní portálu.
2. Načítame kategóriu s informáciami o nej zo súboru `structure.u8.rdf.gz`.
3. Načítame ku kategórii z kroku 2 odkazy na externé stránky s ich popismi zo súboru `content.u8.rdf.gz`.

4. Pripravíme si md5 kód z údajov o načítanej kategórii.

5. Pripravený kód porovnáme s kódom z predchádzajúcej verzie. Ak sú kódy rovnaké pokračujeme krokom 2. Inak pokračujeme krokom 6.

6. Vytvoríme z načítanej kategórie DOM dokument.

7. Pozrieme sa na cestu, na ktorú by sa pri vytváraní portálu ukladal súbor index.xml s touto kategóriou, a načítame ho do DOM dokumentu. Následne porovnáme nový a starý DOM dokument a zistíme, či v tejto kategórii pribudli alebo boli odstránené nejaké podkategórie. Pre nové podkategórie vytvoríme adresáre a pre odstránené podkategórie odstránime adresáre z adresárovej štruktúry. Vytvoríme nový index.xml súbor.

8. Ak existuje ďalšia kategória, prejdeme na krok 2. Inak uložíme pomocnú dátovú štruktúru s aktuálnymi md5 kódmi do súboru pre ďalšiu aktualizáciu a algoritmus ukončíme.

2.5 Aktualizácia indexu

Aktualizácia indexu prostredníctvom Egothoru je implementovaná, tak že pre zmenu dokumentu treba dokument najskôr vymazať a následne ho pridať s aktuálnymi dátami. Preto ju môžeme rozdeliť na dve operácie a to mazanie dokumentu z indexu a pridanie dokumentu do indexu. Pre mazanie potrebujeme určiť, ktoré kategórie boli odstránené alebo zmenené. Pre pridanie potrebujeme zistiť, ktoré kategórie boli pridané alebo zmenené.
Najskôr si nájdeme zmenené kategórie. Načítame si dátovú štruktúru s md5 kódmi vytvorenými z údajov o každej kategórii, ktoré sa využívajú pri vytváraní indexu. Tú sme si pripravili pri vytváraní portálu. Podobne ako pri aktulizácii portálu budeme načítavať jednotlivé kategórie, počítať md5 kódy z indexovaných údajov a porovnávať ich s kódmi z aktualizovanej verzie. Ak sú kódy rôzne, kategóriu v indexe aktualizujeme.
V ďalšom kroku budeme riešiť aktualizácie symlinkov a to nasledujúcim spôsobom. Pri aktualizácii indexu si vytvoríme rovnakú štruktúru symlinkov ako pri jeho vytváraní v kroku 2 a načítame si rovnakú dátovú štruktúru z vytvárania indexu predchádzajúcej verzie. V ďalšom kroku vyhľadáme všetky zmenené symlinky vo všetkých kategóriach a nájdeme k nim podstromy kategórii, na ktoré sa zo zmeneného symlinku odkazovalo, a tie v indexe zaktualizujeme.
Pridané a odstránené kategórie nájdeme jednoducho porovnaním dátových

štruktúr so symlinkami.
Teraz môžeme vymazať všetky kategórie, ktoré boli odstránené a kategórie, ktoré treba aktualizovať. Následne môžeme do indexu pridať kategórie s aktualizovanými metadátami a nové kategórie.

2.6 Webová aplikácia

Samotný portál bude uložený v XML súboroch. To umožní užívateľovi pomocou technológie xslt určiť jeho výsledný formát. Jednou z možností je vytvoriť pomocou xslt šablóny previazané HTML stránky pomocou hypertextových odkazov. K týmto HTML stránkam je tiež možné definovať výsledný vzhľad využitím kaskádových štýlov. Pri použití týchto technológií nie je problém v xslt šablóne pridať do stránky formulár na vyhľadávanie. Ďalej treba vyriešiť dotazovanie a vypisovanie výsledkov dotazov. Na to použijeme projekt E2Web. Ide o webovú aplikáciu, ktoré má implementovanú knižnicu jsp tagov pracujúcich priamo s Egothorom. E2Web nám s malými úpravami umožní dotazovať sa a vypisovať výsledky hľadania po stránkach. Bude ale treba do neho dorobiť podporu pre vyhľadávanie v kategóriach a v symlinkoch. To bude spočívať v pridávaní `<BITMAP>` tokenov podľa kategórie, z ktorej sa užívateľ dotazoval.

Kapitola 3

Programátorská dokumentácia

3.1 Programovací jazyk

Na implementáciu použijeme programovací jazyk Java vo verzii J2SE 5.0 ([3]). Java má veľmi dobrú podporu technológie XML, jednoduchú tvorbu a správu kódu a veľké množstvo užitočných knižníc, ktoré je možné jednoduchým spôsobom využiť v našom kóde.

3.2 Vývojové prostredie

Ako vývojové prostredie sme si zvolili Eclipse s rozšírením o plugin webtools, ktorý je možné nájsť na internetových stránkach [11]. Toto prostredie ponúka podporu pre vývoj webových aplikácií v jazyku Java.

3.3 Štruktúra workspace

Workspace, alebo teda pracovná plocha v prostredí Eclipse, obsahuje 4 projekty:

- DMOZ_LAUNCH - Projekt obsahuje inštaláciu webového servera Apache Tomcat 5.5 nakonfigurovanú na použitie priamo z workspace. Obsahuje jednu triedu `TomcatLaunch`, ktorá v main metóde spúšťa webový server. Pre vývoj netreba už tento server inštalovať ani konfigurovať, stačí ho iba spustiť.
- DMOZ_PROCESSOR - Projekt obsahuje implementáciu programu.

- EGOTHOR - Projekt obsahuje zdrojový kód projektu Egothor, pre jednoduchší debuging a prípadné zmeny potrebné k implementácii.
- DMOZ_UNIT - Projekt je určený pre JUnit testy.

Workspace je možné nájsť na priloženom DVD v adresári `Workspace`.

3.4 Vytvorenie inštalačného balíku

Na vytvorenie inštalačného balíku použijeme nástroj Apache Ant ([17]). V projekte DMOZ_PROCESSOR je súbor so skriptom `build.xml`, pomocou ktorého je možné skompilovať zdrojový kód, vytvoriť webovú aplikáciu pre Apache Tomcat 5.5 a vytvoriť inštalačný balík so všetkými potrebnými súbormi pre použitie programu.
Skript `build.xml` je možné spustiť priamo z vývojového prostredia Eclipse, alebo pomocou príkazového riadku. V tom prípade treba mať nadstavenú systémovú premennú `ANT_HOME` na cestu k inštalácii Apache Ant a v premennej `PATH` pridanú cestu `$ANT_HOME/bin`.

Funkcie skriptu:

- `clear` - Premaže adresáre, ktoré sa vytvárajú pri kompilácii a vytváraní balíkov.
- `compile` - Skompiluje zdrojový kód do adreára temp.
- `jar` - Vytvorí jar archív `dmoz_processor.jar` a skopíruje ho do adresára dist.
- `war` - Vytvorí webový archív `dmoz.war` a skopíruje ho do adresára dist.
- `package` - Vytvorí inštalačný balík a skoprimuje ho do archívu `dmoz.tar.gz` v adresári dist.
- `run-all` - Spustí všetky funkcie skriptu.

Príklad spustenia skriptu z príkazového riadku:
`ant run-all`

3.5 Popis tried, rozhraní a balíkov

Programátorská dokumentácia k balíkom, triedam a rozhraniam je vytvorená vo forme HTML stránok pomocou nástroja javadoc ([4]), ktorý je súčasťou J2SE 5.0 a je k dispozícii na priloženom DVD v adresári `Documents/Javadoc`. Je písaná v angličtine, pre prípad zverejnenia projektu na niektorom z open source serverov.
Aplikácia je rozdelená do dvoch hlavných balíkov dmoz a e2web. Balík dmoz obsahuje balíky s triedami a rozhraniami s našou implementáciou. Balík e2web obsahuje balíky z projektu E2Web s našou modifikáciou.

3.6 Použité knižnice

3.6.1 Egothor

Vyhľadávací stroj Egothor. V projekte je použitý na implementáciu vyhľadávania v katalógu dmoz.org. Ide o open source projekt a je možné ho nájsť na internetových stránkach [2].

3.6.2 E2Web

E2Web je webová aplikácia, ktorá obsahuje implementáciu jsp tagov pracujúcich s vyhľadávacím strojom Egothor. Je súčasťou projektu Egothor a je ju možné nájsť na internetových stránkach [2].

3.6.3 Woodstox

Je rýchla a spoľahlivá implementácia StAX parsera XML súborov. Je to open source projekt a je ho možné nájsť na internetových stránkach [6].

3.6.4 Trove

Trove je implementácia dátových štruktúr optimalizovaných na rýchlosť a veľkosť využitej operačnej pamäte pre primitívne dátové typy v jazyku Java. Pri ich použití vo výpočtoch pri hľadaní symlinkov nám znížili pamäťové nároky o viac ako 50%. Ide o open source projekt a jeho ho možné nájsť na internetových stránkach [7].

3.6.5 Log4j

Log4j je knižnica, ktorá umožňuje jednoduchým spôsobom zaviesť a následne nadstaviť logovanie aplikácie. Dovoľuje rozdeliť logovacie hlášky do viacerých úrovní podľa typu a dôležitosti informácie. V konfiguračnom súbore potom stačí iba uviesť, akú úroveň logovania chce užívateľ vidieť. Môžeme si tiež nadstaviť, kam chceme, aby sa nám logy zobrazovali, v akom formáte a ktoré konkrétne balíky chceme logovať. Ide o open source projekt a jeho ho možné nájsť na internetových stránkach [5].

3.7 Webové rozhranie

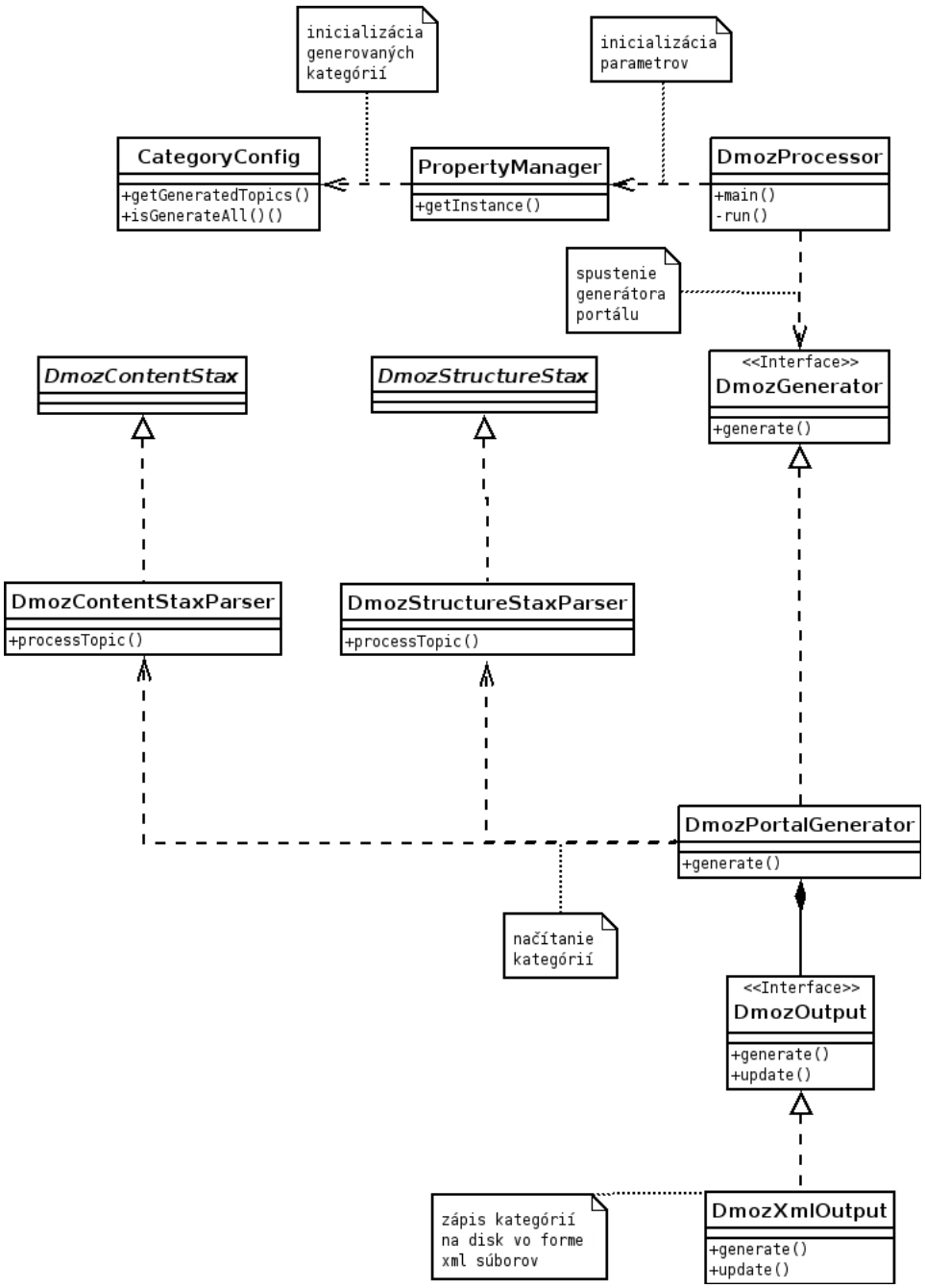
Aby sme mohol pristupovať k indexu vyhľadávača, potrebujeme použiť webový server s podporou Java Server Pages na spustenie webovej aplikácie E2Web, ktorá je súčasťou projektu Egothor. Táto aplikácia vyžaduje Apache Tomcat 5.5.

3.8 Rozšíritelnosť

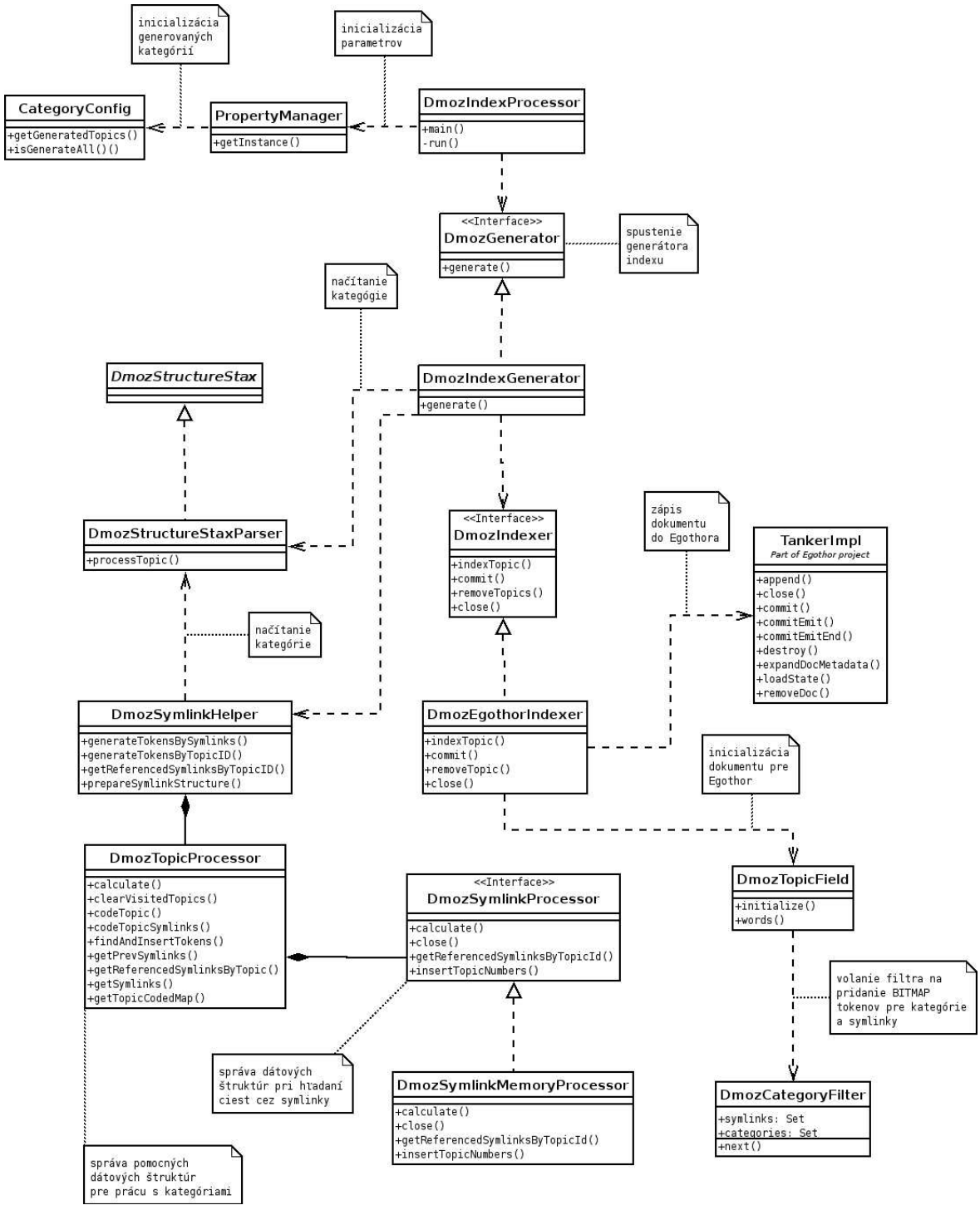
Na miestach, kde by malo zmysel použiť inú implementáciu, sme použili rozhrania, ktoré zabezpečia jej jednoduchý prechod. Ide hlavne o spracovanie vstupu a výstupu, použitie vyhľadávacieho stroja a kritické miesta pri výpočtoch.

3.9 UML

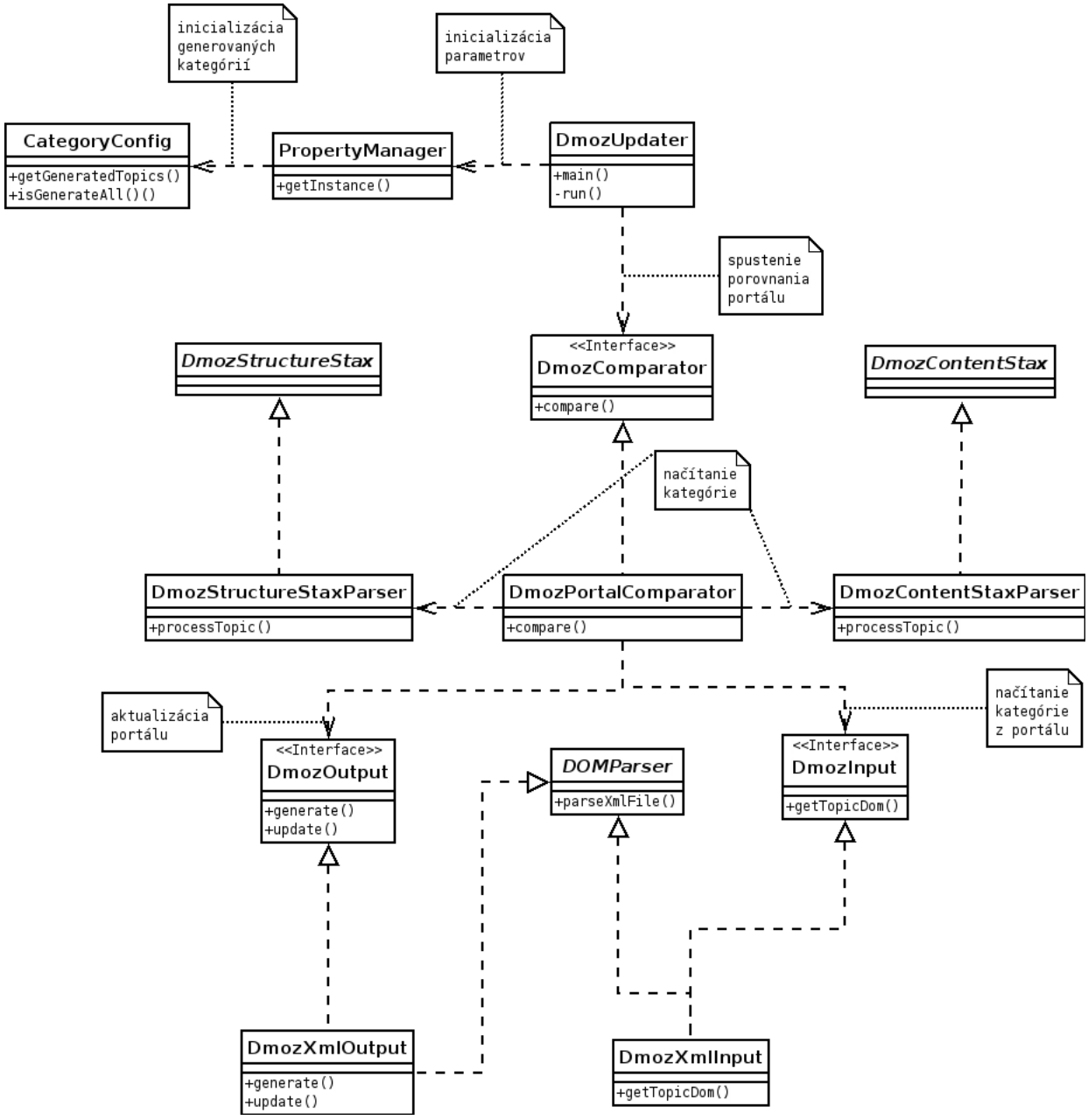
Pre znázornenie závislostí medzi triedami sme zvolili diagramy tried v UML notácii. Triedy v diagramoch obsahujú pre prehľadnosť len public metódy. Na obrázku 3.1 môžeme vidieť diagram tried potrebných pre vytvorenie portálu pre katalóg dmoz.org. Obrázok 3.2 znázorňuje diagram tried potrebných pre vytvorenie indexu. Obrázky 3.3 a 3.4 znázorňujú diagramy tried potrebných pre aktualizáciu portálu a indexu.



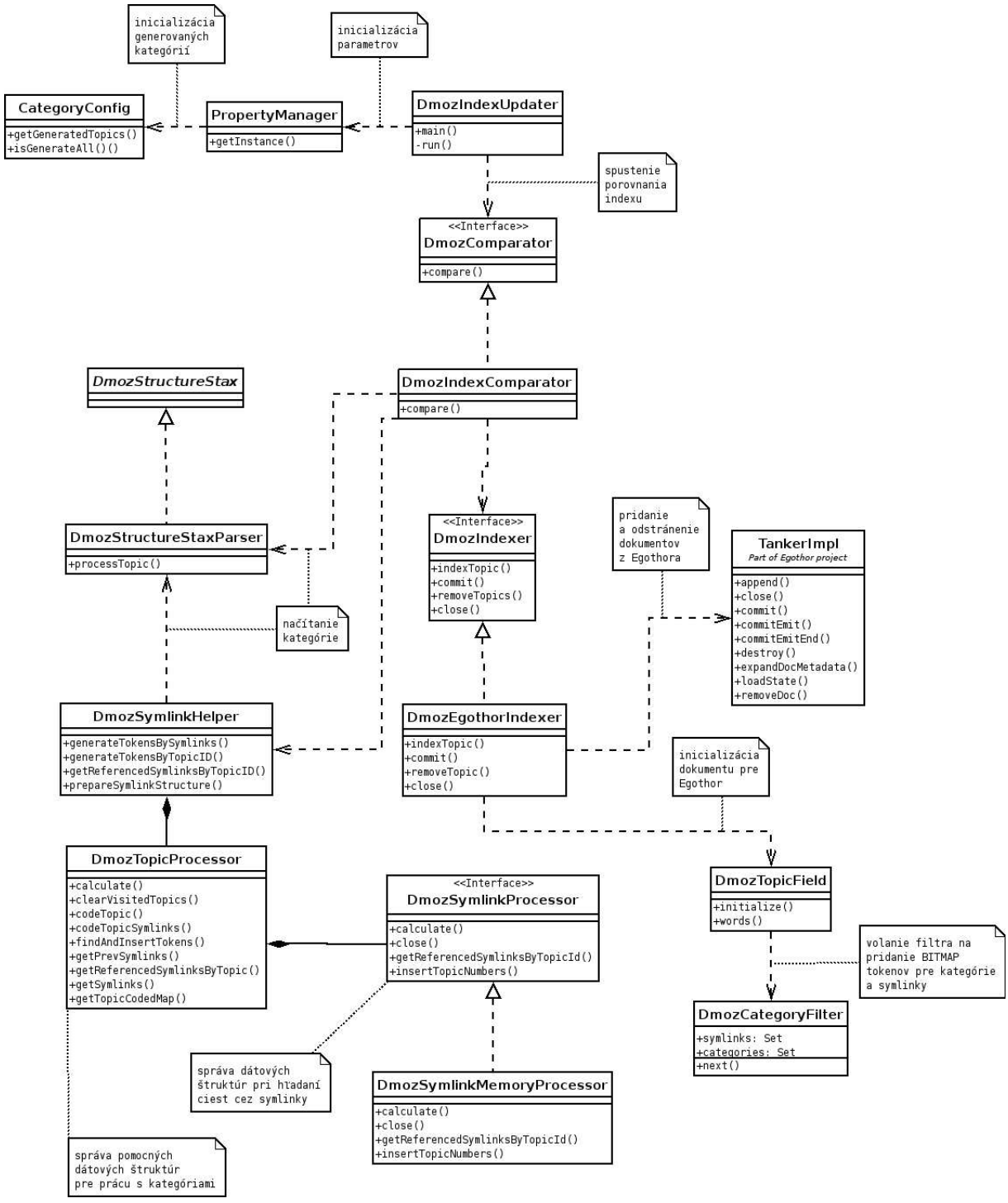
Obr. 3.1: Diagram tried pre vytvorenie portálu



Obr. 3.2: Diagram tried vytvorenia indexu



Obr. 3.3: Diagram tried aktualizácie portálu



Obr. 3.4: Diagram tried pre aktulizáciu indexu

Kapitola 4

Testovanie

4.1 Testovacie prostredie

4.1.1 Konfigurácia hardware

CPU	: AMD Athlon(tm) 64 X2 Dual Core Processor 3800+
Operačná pamäť	: 2 GB
Disk	: 98 GB, 7200 ot/s

4.1.2 Konfigurácia software

OS	: Gentoo Linux, kernel 2.6.18
Súborový systém	: Ext2
JDK	: Sun Microsystems JDK 1.5.0.08

4.2 Výsledky testov

4.2.1 Vytvorenie portálu

Vytvorenie portálu sme testovali nad celým obsahom katalógu dmoz.org.

Výsledky testu	
Počet aktualizovaných kategórií	: 718842
Čas trvania testu	: 35m 24s

Ďalší test sme vykonali s kategóriou /Arts.

Výsledky testu	
Počet aktualizovaných kategórií	: 47003
Čas trvania testu	: 4m 18s

4.2.2 Vytvorenie indexu

Vytvorenie indexu sme testovali nad celým obsahom katalógu dmoz.org. Hĺbku prehľadávania symlinkov sme zvolili 10.

Výsledky testu	
Počet aktualizovaných kategórií	: 718842
Čas trvania testu	: 3h 12m 6s

Druhý test sme spravili s kategóriou /Arts.

Výsledky testu	
Počet aktualizovaných kategórií	: 47003
Čas trvania testu	: 7m 42s

4.2.3 Aktualizácia portálu

Aktualizáciu portálu sme testovali nad celým obsahom katalógu dmoz.org. Najskôr sme si vytvorili portál z predposlednej verzie zdrojových súborov `structure.rdf.u8.gz` a `content.rdf.u8.gz`. Nad týmto portálom sme spustili aktualizáciu podľa zdrojových súborov poslednej verzie.

Výsledky testu	
Počet zmenených kategórií	: 19130
Počet nových kategórií	: 719
Počet odstránených kategórií	: 118
Čas trvania testu	: 11m 54s

4.2.4 Aktualizácia indexu

Aktualizáciu indexu sme testovali nad rovnakými dátami ako aktualizáciu portálu.

Výsledky testu	
Počet kategórií so zmenenými indexovanými údajmi	: 1098
Počet nových kategórií	: 719
Počet odstránených kategórií	: 118
Čas trvania testu	: 5m 19s

Aktualizáciu indexu sme testovali na vytvorenom indexe z 719443 kategórií. Nasimulovali sme rôzny počet zmien v kategóriach, ktoré sme v tomto indexe následne aktualizovali.

Výsledky sú zobrazené v následujúcej tabuľke:

Počet aktualizovaných kategórií (%)	Čas
1	3m 57s
5	6m 12s
10	10m 24s
20	20m 11s

Na výsledkoch testu je vidieť, že čas potrebný na aktualizáciu indexu závisí od počtu kategórií, ktoré treba aktualizovať.

4.2.5 Vyhľadávanie

Vyhľadávanie sme testovali s indexom vytvoreným nad celým obsahom katelógu dmoz.org. Index sme postupne aktualizovali simulovanými zmenami v kategóriach a následne testovali, ako sa zmenil čas potrebný na vyhľadanie výsledku. Ako testovacie dáta sme zobrali zoznam často používaných slov a zoznam kategórií do druhej úrovne, z ktorých sme postupne vytvárali dotazy. Počet kategórií, ktoré bolo treba aktualizovať pri teste aktualizácie indexu, vychádza na 0,2% z celkového počtu kategórií. Aby sme si otestovali vlastnosti vyhľadávača pri horších podmienkach, budeme simulovať 1% zmenených kategórií.

Výsledky testu pre aktualizácie 1% kategórií:

Počet aktualizácií	Priemerný čas vyhľadania výsledku
0	28 ms
1	37 ms
2	33 ms
3	33 ms
4	32 ms
5	37 ms
6	30 ms
7	36 ms
8	33 ms
9	37 ms
10	34 ms

Z výsledku testu je možné vidieť, že aktualizácia kategórií v indexe minimálne ovplyvní dobu potrebnú na vyhľadanie výsledku.

Kapitola 5

Užívateľská dokumentácia

5.1 Systémové požiadavky

5.1.1 Operačný systém

Program na vytvorenie a akutalizáciu portálu s vyhľadávaním je možné spustiť na každom operačnom systéme, na ktorom je možné nainštalovať Java Runtime Environment vo verzii 5.0. Zároveň sa ale musí použiť súborový systém, ktorý podporuje veľkosť názvu súborov a adresárov najmenej 255 znakov a musí podporovať dostatočne veľké súbory v závislosti od počtu vytváraných kategórií.
Aplikácia je testovaná v rámci možností na OS Linux a Windows XP. Na Linuxe bol použitý súborový systém Ext2 a ReiserFS. Doporučujeme použiť reiserfs, ktorý je optimalizovaný na prácu s malými súbormi. Vytvorenie portálu s použitím tohoto súborového systému môže aplikáciu urýchliť až dvojnásobne oproti Ext2. Na OS Windows XP je aplikácia testovaná so súborovým systémom NTFS.

5.1.2 J2SE Development Kit 5.0

J2SE Development Kit vo verzii 5.0 (ďalej JDK 5) obsahuje sadu programov pre vývojárov, ako je napríklad kompilátor, a prostredie na spustenie Java programov Java Runtime Environment. Na spustenie programu na vytvorenie a aktualizáciu portálu s vyhľadávaním stačí použiť Java Runtime Environment vo verzii 5.0. Pri použití webového servera Apache Tomcat 5.5, ktorý vyžaduje naša webová aplikácia, treba už mať nainštalovaný JDK 5.

5.1.3 Apache Tomcat 5.5

Apache Tomcat vo verzii 5.5 ([12]) je webový server, ktorý umožňuje použitie servletov a jsp stránok. Je nevyhnutný pri použití webovej aplikácie s DMOZ portálom s vyhľadávaním, ktorá je súčasťou inštalácie.

5.2 Požiadavky na hardware

5.2.1 Disk

Veľkosť potrebného voľné miesta na disku závisí od počtu kategórií, pre ktoré sa bude vytvárať portál a index pre vyhľadávač. Veľkosť indexu je ešte ovplyvnená hĺbkou prehľadávania symlinkov pri jeho vytváraní.

- Veľkosť vstupných súborov:
  - `content.rdf.u8.gz` - 312 MB, pri dekomprimácii 2 GB
  - `structure.rdf.u8.gz` - 68 MB, pri dekomprimácii 638 MB

  Veľkosť vstupných súborov bola meraná v čase implementácie projektu. S pribúdajúcimi kategóriami sa ich veľkosť bude postupne zväčšovať.
- Veľkosť portálu - 6.2 GB pri vytvorení zo všetkých kategórií.
- Veľkosť indexu:
  - S nadstavením hĺbky prehľadávania symlinkov na 0 - 3 GB
  - S nadstavením hĺbky prehľadávania symlinkov na 10 - 4.6 GB
- Veľkosť pomocných dát vytváraných pri výpočtoch - 153 MB

Pre bezproblémový beh programu je dobré vyčleniť aspoň 11.5 GB voľného miesta na disku, pri vytváraní portálu s vyhľadávaním pre všetky kategórie.

5.2.2 Operačná pamäť

Veľkosť potrebnej operačnej pamäti zavisí na počte kategórií a na nadstavení hĺbky prehľadávania symlinkov.
Potrebná pamäť na vytvorenie indexu a jeho aktualizáciu nad všetkými kategóriami:

- S nadstavením hĺbky prehľadávania symlinkov na 0 - 512 MB

- S nadstavením hĺbky prehľadávania symlinkov na 10 - 1 GB

Na vytvorenie portálu a jeho aktualizáciu si vystačíme s 512 MB operačnej pamäte.

5.2.3 Procesor

Požiadavky na procesor sú ekvivalentné s požiadavkami pre inštaláciu JDK 5.

5.3 Inštalácia

Inštaláciu balíku s programom budeme popisovať na OS Linux, na ktorý je primárne určená. Je ju možné použiť aj na ostatných operačných systémoch spĺňajúcich podmienky popísané v kapitole 5.1.1. Pre nich si ale užívateľ musí upraviť spúšťacie skripty podľa ich potrieb. Všetky súbory potrebné na spustenie sú obsiahnuté v balíku `dmoz.tar.gz`, ktorý je možné nájsť na priloženom DVD v adresári `Install`. Súbor si skopírujeme do adresára, z ktorého budeme aplikáciu spúšťať.
Po rozbalení archívu sa vytvorí adresárová štruktúra:

```
[urtax:~/dmoz/install]$ tar xvfz dmoz.tar.gz
config/
data/
lib/
war/
config/dmoz.properties
config/generate.xml
config/log4j.properties
generate_index.sh
generate_portal.sh
lib/dmoz_processor.jar
lib/egothor2.jar
lib/jsr173_api.jar
lib/log4j-1.2.13.jar
lib/trove.jar
lib/wstx-lgpl-2.0.3.jar
update_index.sh
update_portal.sh
war/dmoz.war
```

Popis súborov a adresárov v inštalácii:

- data - Prázdny adresár určený na súbory z katalógu dmoz.org.
- config - Adresár obsahujúci konfiguračné súbory.
- war - Adresár obsahujúci webový archív dmoz.war našej webovej aplikácie.
- lib - Adresár obsahujúci jar súbory k použitým knižniciam.

V hlavnom adresári sú skripty na spustenie jednotlivých procesov:

- generate_portal.sh - Skript na vytvorenie portálu.
- generate_index.sh - Skript na vytvorenie indexu k portálu.
- update_portal.sh - Skript na aktualizáciu portálu.
- update_index.sh - Skript na aktualizáciu index.

Aby bolo možné tieto súbory spustiť, treba im pridať právo na spustenie. To je možné urobiť príkazom `chmod u+x *.sh` v adresári, kde sú súbory uložené.

5.4 Konfigurácia

Konfiguračné súbory k aplikácii sa nachádzajú v adresári config. Nájdeme tam súbory `generate.xml`, `dmoz.properties` a `log4j.properties`.

5.4.1 Konfiguračný súbor `generate.xml`

Ide o súbor v XML formáte, ktorý obsahuje zoznam kategórií, z ktorých budeme vytvárať portál a index pre vyhľadávač.

```
<?xml version="1.0" encoding="UTF-8"?>
<categories all="false">
  <category>Arts</category>
  <category>World/Česky</category>
</categories>
```

Výpis kódu 5.1: Príklad generate.xml súboru

Je dôležité správne definovať kódovanie znakovej sady v hlavičke XML atribútom encoding na UTF-8, lebo názvy kategórií môžu byť definované v rôznych jazykoch. Ďalej máme možnosť definovať, či chceme použiť všetky kategórie. To je možné zariadiť v koreňovom elemente categories dosadením hodnoty true atribútu all. V tom prípade sa už nebudú brať do úvahy vymenované kategórie, ale použijú sa všetky. V opačnom prípade máme možnosť

definovať kategórie elementom category, ktorého textová hodnota označuje plný názov kategórie. Názov kategórie je nutné zadávať aj s nadradenými kategóriami a oddeliť ich znakom /, ako je to v ukážkovom `generate.xml` súbore pri kategórii World/Česky. V tomto prípade sa budú brať do úvahy iba kategórie obsiahnuté v Česky, ktorá má nadradenú kategóriu World. Aktuálny zoznam všetkých kategórií je možné nájsť na stránke katalógu dmoz.org na adrese `http://rdf.dmoz.org/rdf/categories.txt`.

5.4.2 Konfiguračný súbor `dmoz.properties`

V tomto súbore sú definované základné nadstavenia programu. Obsahuje následujúce údaje:

Názov	**Typ**	**Popis**
deploy.dir	String	Cesta, na ktorej sa budú nachádzať súbory vytvoreného portálu v rámci webovej aplikácie. Hodnota tejto premennej sa použije ako prefix cesty v URL. Pre hodnotu /dmoz/xml to znamená, že koreňovú kategóriu nájdeme na URL `http://localhost:8080/dmoz/xml/index.xml`. Pri použití webovej aplikácie z balíku je treba použiť hodnotu /dmoz/xml.
category.file	String	Cesta, na ktorej sa nachádza súbor generate.xml popísaný v kapitole 5.4.1.
result.dir	String	Cesta, na ktorú sa vytvorí portál.
structure.file	String	Cesta k súboru `structure.rdf.u8.gz` stiahnutého zo stránky katalógu dmoz.org. Ak súbor končí príponou gz, program s ním počíta ako s archívom vytvoreným GNU zipom. V opačnom prípadne s ním pracuje ako so XML súborom.
content.file	String	Cesta k súboru `content.rdf.u8.gz` stiahnutého zo stránky katalógu dmoz.org. Podobne ako u `structure.rdf.u8.gz`, ak súbor končí príponou gz, program s ním počíta ako s archívom vytvoreným GNU zipom. V opačnom prípadne s ním pracuje ako so XML súborom.

xsl.file	String	Cesta, na ktorej sa bude nachádzať xslt šablóna pre XML súbory z portálu v rámci webovej aplikácie. Pre hodnotu /dmoz/dmoz.xsl to znamená, že šablónu nájdeme na URL `http://localhost:8080/dmoz/dmoz.xsl`. Pri použití webovej aplikácie z balíku, treba použiť hodnotu /dmoz/dmoz.xsl.
topic.index.dir	String	Cesta, na ktorú sa vytvorí index pre vyhľadávač.
db.dir	String	Cesta, na ktorú sa budú ukladať pomocné súbory použité pri vytváraní indexu pre vyhľadávač.
db.prev.dir	String	Cesta k adresáru k premennej db.dir z prechádzajúcej verzie katalógu dmoz.org. Táto premenná je dôležitá pri aktualizácii indexu a portálu.
symlinks.maxdepth	Integer	Pomocou tejto premennej je možné určiť maximálnu hĺbku prehľadávania symlinkov. Týmto parametrom je možné výrazne ušetriť veľkosť potrebnej operačnej pamäte k príprave indexu.
log4j.config	String	Cesta ku konfiguračnému súboru pre log4j knižnicu.

Príklad konfiguračného súboru dmoz.properties:

```
deploy.dir=/dmoz/xml
category.file=config/generate.xml
result.dir=dmoz/output/xml
structure.file=data/structure.rdf.u8.gz
content.file=data/content.rdf.u8.gz
xsl.file=/dmoz/dmoz.xsl
topic.index.dir=output/index
db.dir=output/db
db.prev.dir=output/db_old
symlinks.maxdepth=10
log4j.config=config/log4j.properties
```

5.4.3 Konfiguračný súbor `log4j.properties`

Ide o konfiguračný súbor knižnice Log4j, v ktorej je možné definovať formát logu, jeho výstup a úrovne, ktoré sa majú logovať. Presnú syntax tohto súboru je možné nájsť na internetovej stránke projektu Log4j [5].

Príklad konfiguračného súboru `log4j.properties`:

```
log4j.rootLogger=INFO, stdout
```

```
log4j.rootLogger=DEBUG, R
log4j.appender.stdout=org.apache.log4j.ConsoleAppender
log4j.appender.stdout.layout=org.apache.log4j.PatternLayout
log4j.appender.stdout.layout.ConversionPattern=%d %5p [%t] - %m%n
log4j.appender.R=org.apache.log4j.RollingFileAppender
log4j.appender.R.File=dmoz.log
log4j.appender.R.MaxFileSize=100KB
log4j.appender.R.MaxBackupIndex=5
log4j.appender.R.layout=org.apache.log4j.PatternLayout
log4j.appender.R.layout.ConversionPattern=%d %5p [%t] - %m%n
```

V príklade je nadstavené, aby sa log vypisoval na štandardný výstup na úrovni INFO a zároveň do súboru na úrovni DEBUG. Súbor má definovanú maximálnu veľkosť 100KB. Ak túto veľkosť prekročí, vytvorí sa z neho záloha, vymaže sa a začne sa do neho zapisovať odznova. Maximálny počet záloh tohoto súboru je v príklade definovaný na 5.

5.5 Vytvorenie portálu

- Stiahneme si aktuálnu verziu zdrojových súborov katalógu dmoz.org `structure.rdf.u8.gz` a `content.rdf.u8.gz` zo stránky `http://rdf.dmoz.org/rdf` a uložíme ich do adresára data. V prípade, že máme pc s pomalým procesorom a máme dostatočne veľa voľného miesta na disku, je dobré tieto súbory dekomprimovať.

- Skontrolujeme nadstavenia v XML súbore `config/generate.xml`, v ktorom sú definované kategórie, ktoré sa majú generovať podľa pokynov v kapitole 5.4.1.

- Skotrolujeme nadstavenia v konfiguračnom súbore `config/dmoz.properties` podľa pokynov v kapitole 5.4.2.
  Dôležité nadstavenia:

  - `category.file` - Cesta k súboru `generate.xml` s definíciou kategórií.
  - `result.dir` - Cesta pre vytvorenie portálu.
  - `structure.file` - Cesta k súboru `structure.rdf.u8.gz` z katalógu dmoz.org.
  - `content.file` - Cesta k súboru `content.rdf.u8.gz` z katalógu dmoz.org.

  - `xsl.file` - Meniť iba v prípade, ak nechceme použiť webovú aplikáciu, ktorá je súčasťou balíku.
  - `deploy.dir` - Meniť iba v prípade, ak nechceme použiť webovú aplikáciu, ktorá je súčasťou balíku.
  - `db.dir` - Cesta, na ktorú sa budú ukladať pomocné súbory.

- Skontrolujeme nadstavenia systémových premenných v skripte `generate_portal.sh`. Premenná `JAVA_HOME` musí obsahovať cestu do adresára s inštaláciou JDK 5. Premenná `DMOZ_PROPERTIES` musí obsahovať cestu ku konfiguračnému súboru dmoz.properties.

- Samotné vytvorenie portálu spustíme skriptom `generate_portal.sh`. Po ukončení programu nájdeme vytvorený portál na ceste definovanej v súbore `dmoz.properties` premennou `result.dir`.

Aby bolo možné použiť webovú aplikáciu z balíku, treba ešte vytvoriť index pre vyhľadávač.

5.6 Vytvorenie indexu pre vyhľadávač

- Skontrolujeme definované kategórie v súbore config/generate.xml podobne ako pri vytváraní portálu. Skontrolujeme nadstavenia premenných v konfiguračnom súbore dmoz.properties podľa kapitoly 5.4.2. Hodnoty premenných nadstavených pri vytváraní portálu by mali zostať nezmenené. Treba ale naviac skotrolovať premenné:

  - `topic.index.dir` - Cesta, na ktorú sa vytvorí index pre vyhľadávač.
  - `db.dir` - Cesta, na ktorú sa budú ukladať pomocné súbory použité pri vytváraní indexu.
  - `symlinks.maxdepth` - Určíme maximálnu hĺbku prehľadávania symlinkov.

- Skontrolujeme nadstavenia systémových premenných v skripte `generate_index.sh`. Premenná `JAVA_HOME` musí obsahovať cestu do adresára s inštaláciou JDK 5. Premenná `DMOZ_PROPERTIES` musí obsahovať cestu k súboru `dmoz.properties`. Nadstavíme dostatočnú veľkosť operačnej pamäte, ktorú si bude môcť java virtual machine alokovať. To sa nadstaví v premennej `JAVA_OPTS` napríklad hodnotou -Xmx512m, pri použití 512 MB operačnej pamäte. Veľkosť potrebnej

pamäte zavisí na počtu vytváraných kategórií a nadstavení hĺbky prehľadávania symlinkov.

- Samotné vytvorenie indexu pre vyhľadávač spustíme skriptom `generate_index.sh`. Po ukončení programu nájdeme vytvorený index na ceste definovanej v súbore `dmoz.properties` premennou `topic.index.dir`.

5.7 Inštalácia webovej aplikácie s vytvoreným portálom s vyhľadávaním

- Skotrolujeme nadstavenia systémových premenných `JAVA_HOME`, kde by mala byť cesta k JDK 5 a `CATALINA_HOME`, kde by mala byť cesta k inštalácii webového servera Apache Tomcat 5.5.

- V adresári s inštaláciou Apache Tomcat 5.5 (nadstavenom v premennej `CATALINA_HOME`) si nájdeme adresár webapps a skopírujeme do neho webový archív z inštalačného balíku, ktorý je na ceste `war/dmoz.war`. Po spustení webového servera príkazom `$CATALINA_HOME/bin/startup.sh` sa `dmoz.war` dekomprimuje do adresára `$CATALINA_HOME/webapps/dmoz`.

- Ďalej je treba pridať do webovej aplikácie pripravený portál. Ten nájdeme na ceste definovanej v premennej `result.dir` v konfiguračnom súbore `config/dmoz.properties` v adresári, v ktorom sme portál vytvárali. Celý ho presunieme alebo prekopírujeme do adresára `$CATALINA_HOME/webapps/dmoz`. Portál je pripravený, potrebujeme ešte správne nadstaviť cestu k vytvorenému indexu. To spravíme nadstavením kontextového parametru index v súbore `$CATALINA_HOME/webapps/dmoz/WEB-INF/web.xml`. Nájdeme si parameter s hodnotou elementu `param-name` index a nadstavíme následujúcemu elementu `param-value` hodnotu na cestu k adresáru s indexom.
  Príklad:

```
<context-param>
    <description>Index location (filepath)</description>
    <param-name>index</param-name>
     <param-value>/home/andrej/dmoz/output/index</param-value>
</context-param>
```

Výpis kódu 5.2: Časť web.xml súboru

- Aby správne fungovalo vyhľadávanie aj v cudzojazyčných kategóriach, musíme nakonfigurovať `$CATALINA_HOME/conf/server.xml`. Nájdeme si element Connector definovaný pre Service s názvom Catalina a pridáme mu atribút `URIEncoding="UTF-8"`. Aby sa zmeny prejavili, je treba reštartovať webový server. To spravíme príkazmi `$CATALINA_HOME/bin/shutdown.sh` a následne `$CATALINA_HOME/bin/startup.sh`.
  Teraz si môžeme overiť, či aplikácia funguje zadaním URL `http://localhost:8080/dmoz/xml/index.xml` do webového prehliadača.

5.8 Aktualizácia portálu

- Stiahneme si aktuálnu verziu zdrojových súborov katalógu dmoz.org `structure.rdf.u8.gz` a `content.rdf.u8.gz` zo stránky `http://rdf.dmoz.org/rdf` a nahradíme nimi existujúce súbory z prechádzajúcej verzie v adresári data. V prípade, že máme pc s pomalým procesorom a dostatočne veľa volného miesta na disku, je dobré tieto súbory dekomprimovať.

- Skontrolujeme nadstavenia v XML súbore `config/generate.xml`, v ktorom sú definované kategórie, ktoré sa majú generovať podľa pokynov z kapitoly 5.4.1.

- Skotrolujeme nadstavenia v konfiguračnom súbore `config/dmoz.properties` podľa kapitoly 5.4.2.
  Dôležité nadstavenia:

  - `category.file` - Cesta k súboru s definícou kategórií.
  - `result.dir` - Cesta, kde sa nachádza portál na aktualizáciu.
  - `structure.file` - Cesta k aktuálnemu súboru `structure.rdf.u8.gz` z katalógu dmoz.org.
  - `content.file` - Cesta k aktuálnemu súboru `content.rdf.u8.gz` z katalógu dmoz.org.
  - `xsl.file` - Hodnotu zmeníme iba v prípade, ak nechceme použiť webovú aplikáciu, ktorá je súčasťou balíku.
  - `deploy.dir` - Hodnotu zmeníme iba v prípade, ak nechceme použiť webovú aplikáciu, ktorá je súčasťou balíku.
  - `db.dir` - Cesta, na ktorú sa budú ukladať pomocné súbory použité pri aktualizovaní portálu.

  - `prev.db.dir` - Cesta z premennej db.dir z predchádzajúcej verzie katalógu dmoz.org.

- Skontrolujeme nadstavenia systémových premenných v skripte `update_portal.sh`. Premenná `JAVA_HOME` musí obsahovať cestu do adresára s inštaláciou JDK 5. Premenná `DMOZ_PROPERTIES` musí obsahovať cestu k súboru `dmoz.properties`.

- Samotné vytvorenie portálu spustíme skriptom `update_portal.sh`. Po ukončení programu nájdeme aktualizovaný portál na ceste definovanej v súbore `dmoz.properties` premennou `result.dir`.

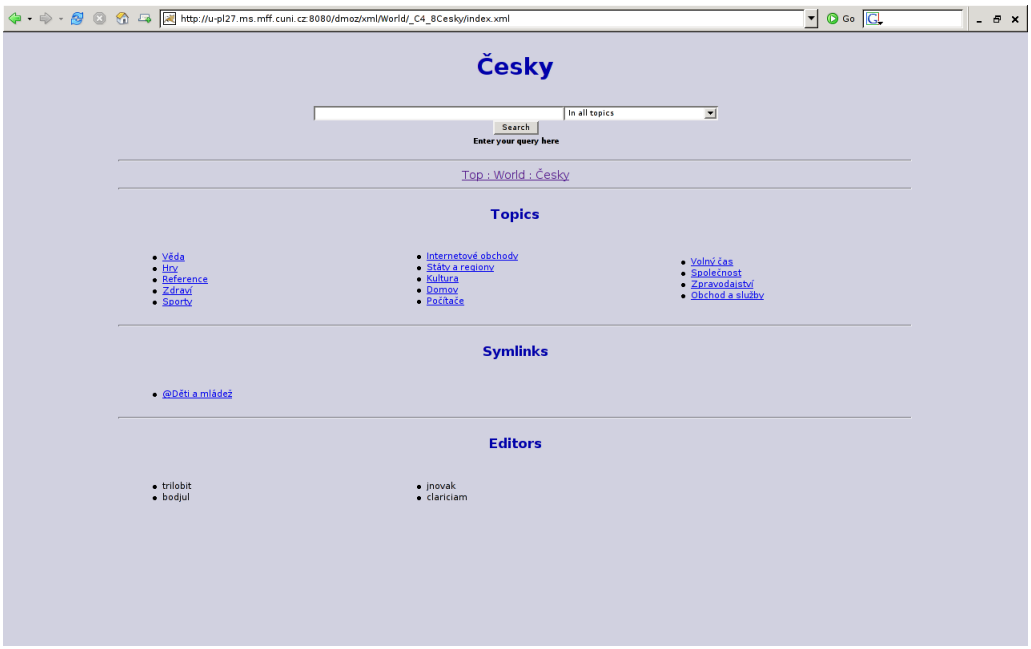
5.9 Aktualizácia indexu pre vyhľadávač

- Skontrolujeme definované kategórie v súbore `config/generate.xml` podobne ako pri aktualizácii portálu.

- Skontrolujeme nadstavenia premenných v konfiguračnom súbore `config/dmoz.properties`. Hodnoty premenných nadstavených pri aktualizovaní portálu by mali zostať nezmenené.
  Treba ale naviac skontrolovať premenné:

  - `topic.index.dir` - Cesta, na ktorej sa nachádza index na aktualizovanie.
  - `db.dir` - Cesta, na ktorú sa budú ukladať pomocné súbory použité pri aktualizovaní indexu.
  - `db.prev.dir` - Cesta z premennej db.dir z predchádzajúcej verzie indexu.
  - `symlinks.maxdepth` - Pri aktualizácii treba nechať hodnotu, aká bola použitá pri vytváraní indexu. Ak by sme rozhodli túto hodnotu zmeniť, je preto nutné vytvoriť celý index znova.

- Skontrolujeme nadstavenia systémových premenných v skripte `update_index.sh`.
  Premenná `JAVA_HOME` musí obsahovať cestu do adresára s inštaláciou JDK 5. Premenná `DMOZ_PROPERTIES` musí obsahovať cestu k súboru `dmoz.properties`. Nadstavíme dostatočnú veľkosť operačnej pamäte, ktorú si bude môcť java virtual machine alokovať. To sa nadstaví v premennej `JAVA_OPTS` napríklad hodnotou -Xmx512m pri použití 512 MB operačnej pamäte.



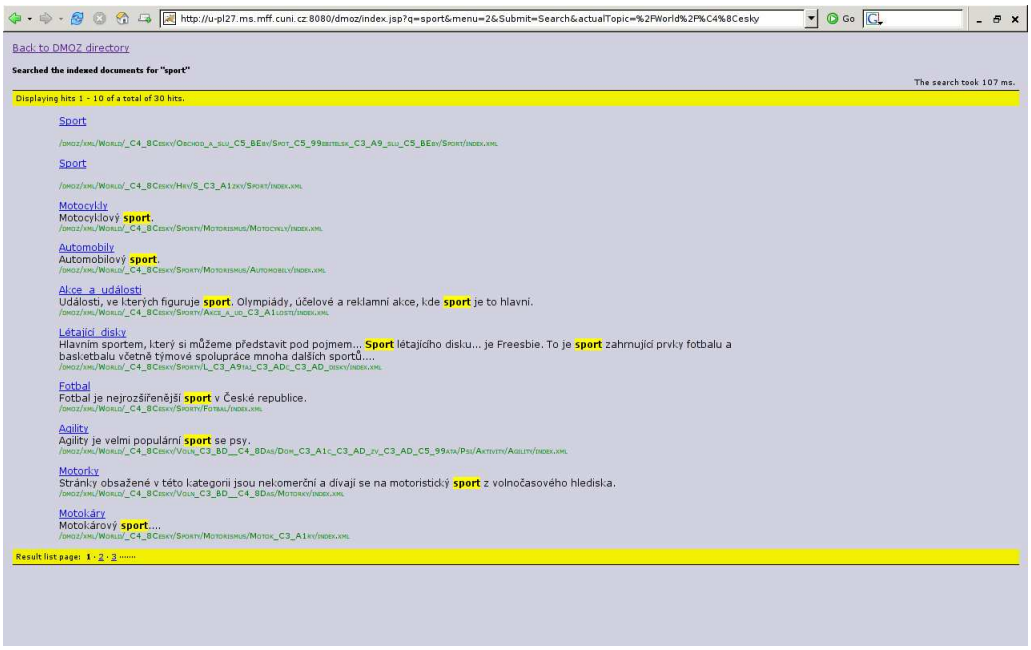
Obr. 5.1: Príklad kategórie

- Aktualizáciu indexu pre vyhľadávač spustíme skriptom `update_index.sh`. Po ukončení programu nájdeme aktualizovaný index na ceste definovanej v súbore `dmoz.properties` premennou `topic.index.dir`.

5.10 Ukážky výsledného portálu

Na obrázku 5.1 môžeme vidieť príklad stránky s kategóriou s názvom Česky vo vytvorenom portále. Môžeme si všimnúť odkazy na ďalšie podkategórie, symlinky a editorov spravujúcich túto kategóriu.

Na obrázku 5.2 môžeme vidieť výsledok vyhľadania dotazu na slovo sport v rámci kategórie s názvom Česky. Výsledok obsahuje 30 kategórií, ktoré v názve alebo popise toto slovo obsahujú. Nájdené kategórie sú rozdelené do troch stránok po 10, aby sa v nich užívateľ jednoduchšie orientoval.



Obr. 5.2: Príklad výsledku vyhľadávania

Kapitola 6

Záver

6.1 Porovnanie s alternatívnymi projektami

Z alternatívnych projektov, ktoré sme v dobe implementácie našli, umožňuje vytvoriť vyhľadávanie pre celý katalóg dmoz.org iba JWebDirectory.
Ostatné projekty umožňujú vytvoriť vyhľadávanie v kategóriach, ale sú veľmi limitované počtom kategórií. Aktualizáciu portálu a indexu, ani vyhľadávanie s prechádzaním symlinkov neimplementuje žiadny zo spomínaných projektov.

6.2 Prínos projektu

Všetky alternatívne projekty implementujú len časť z našich funkčných požiadaviek. Ani jeden z projektov neumožňuje aktualizáciu portálu a dát pre vyhľadávač, ani vyhľadávanie s prechádzaním symlinkov. Preto jedným z prínosov projektu je implementácia týchto chýbajúcich funkčností.
Ďalším prínosom je použitie naimplementovaného kódu pri podobných katalógoch. Každý katalóg má svoje špecifické dáta, preto by bolo treba spraviť určité úpravy, hlavne čo sa týka ich spracovania.

6.3 Splnenie cieľa

Výsledkom bakalárskej práce je program, ktorý dokáže vytvoriť a aktualizovať portál s vyhľadávaním z dát získaných z otvoreného katalógu internetových stránok [1].
Na aktualizáciu portálu s vyhľadávaním o jednu verziu nám pri testovaní stačilo 18,2% z času potrebného na jeho vytvorenie. Pri každej aktualizácii indexu sa čas potrebný na vyhľadanie dotazu mení len minimálne. Môžeme

teda považovať aktualizáciu portálu a indexu za efektívnu.

6.4 Možnosti ďalšieho vývoja

Programu chýba grafické rozhranie na konfiguráciu a spustenie možných operácií, ktoré by bolo možné doimplementovať.
Pri implementácii sme sa nezaoberali paralelným spracovaním jednotlivých výpočtov, ktoré by ich mohlo značne urýchliť hlavne na strojoch s viacerými procesormi. Tiež sme netestovali podporu multithreadingu, ktorú vyhľadávací stroj Egothor implementuje.

Literatúra

[1] DMoz.org: *http://www.dmoz.org*

[2] RNDr. Leo Galamboš, Ph.D.: Egothor, *http://www.egothor.org*

[3] Sun Microsystems (2004): Java 2 Platform Standard Edition 5.0 API Specification, *http://java.sun.com/j2se/1.5.0/docs/api*

[4] Javadoc Tool Home Page: *http://java.sun.com/j2se/javadoc*

[5] Log4j project: *http://logging.apache.org/log4j/docs/index.html*

[6] Woodstox: *http://woodstox.codehaus.org*

[7] Trove4j: *http://trove4j.sourceforge.net*

[8] JWebDirectory: *http://jwdd.sourceforge.net*

[9] Extreme Dmoz Extractor: *http://www.dmozextractor.com*

[10] ODP Data Parser: *http://www.ohardt.com/computer/dev/java*

[11] Eclipse Web Tools Platform Project: *http://www.eclipse.org/webtools*

[12] Apache Tomcat: *http://tomcat.apache.org*

[13] Elliotte Rusty Harold (2003): An Introduction to StAX, *http://www.xml.com/pub/a/2003/09/17/stax.html*

[14] SAX: *http://www.saxproject.org*

[15] Cascading Style Sheets: *http://www.w3.org/Style/CSS*

[16] XSL Transformations: *http://www.w3.org/TR/xslt*

[17] Apache Ant: *http://ant.apache.org*

[18] Apache Lucene: *http://lucene.apache.org*