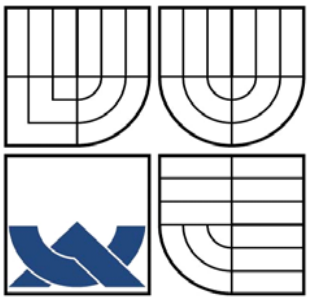
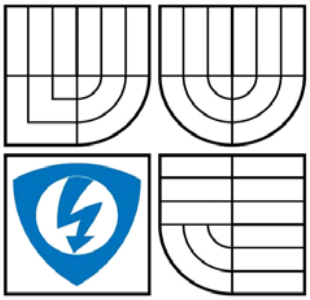


VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA ELEKTROTECHNIKY A KOMUNIKACNÍCH TECHNOLOGIÍ

ÚSTAV TELEKOMUNIKACÍ

FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION
DEPARTMENT OF TELECOMMUNICATIONS

Centralizace a správa distribuovaných informací

Centralization and administration of distributed information

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE — Bc. Richard Valčák
AUTHOR

VEDOUCÍ PRÁCE — Ing. Patrik Morávek
SUPERVISOR

BRNO 2010

ZADANIE

ANOTACE

Diplomová práca pojednáva o súčasnom stave problematiky web mining, ako sú informačné zdroje, bezobslužné metódy prístupu k týmto zdrojom, súhrn dostupných metód a nástrojov. Web data mining je veľmi užitočný nástroj ako získať požadované informácie, ktoré potrebujeme a sú pre nás dôležité pre ďalšie využitie.

Práca je zameraná na návrh systému, ktorý bude z daných informačných zdrojov získavať požadované informácie. Diplomová práca sa skladá z troch častí, ktoré využívajú nami vytvorenú knižnicu a to sú : API, ktorú využíva programátor, serverová aplikácia pre získavanie informácii v čase napr. kurz doláru a ukážku AWT aplikácie, ktorá slúži k preberaniu tabuliek, ktoré sú dostupné na internete.

Kľúčové slova: web mining, web content mining, API, AWT, Java.

.

ABSTRACT

The master's thesis deals with the web mining, information sources, unattended access methods to these sources, summary of available methods and tools. Web data mining is a very useful tool for required information acquiring, which is used for further processing.

The work is focused on the proposal of a system, which is created to gather required information from given sources. The master's thesis consists of three parts, which employ the developed library: API, which is used by programmers, server application for gathering information in time (such an exchange rate for instance) and example of AWT application, which serves for the processing of tables available on the internet.

Keywords: web mining, web content mining, API, AWT, Java.

VALČÁK, R. *Centralizace a správa distribuovaných informacíi: diplomová práce*. Brno: FEKT VUT v Brně, 2010. 56 stran, 2 přílohy. Vedoucí práce Ing. Patrik Morávek.

PREHLÁSENIE

Prehlasujem, že svoju diplomovú prácu na tému „Centralizace a správa distribuovaných informacíi“ som vypracoval samostatne pod vedením vedúceho diplomovej práce a s použitím odbornej literatúry a ďalšími informačnými zdrojmi, ktoré sú všetky citované v práci a uvedené v zozname literatúry na konci práce.

Ako autor uvedenej diplomovej práce ďalej prehlasujem, že v súvislosti s vytvorením tohto projektu som neporušil autorské práva tretích osôb, predovšetkým som nezasiahol nedovoleným spôsobom do cudzích autorských práv osobnosti a som si plne vedomí následkov porušenia ustanovenia § 11 a nasledujúceho autorského zákona č. 121/2000 Sb., vrátane možných trestnoprávnych dôsledkov vyplývajúcich z ustanovenia § 152 trestného zákona č. 140/1961 Sb.

V Brne dňa…….. …….

Podpis študenta

POĎAKOVANIE

Ďakujem vedúcemu práce Ing. Patrikovi Morávkovi za veľmi užitočnú metodickú pomoc a cenné rady pri spracovaní diplomovej práce.

V Brne dňa …… ……..
Podpis študenta

OBSAH

Zoznam obrázkov

1.Úvod

V dnešnej dobe sa internetová scéna rozrastá takým tempom, že vznikajú aspekty, ktorým sa je potreba venovať. Jedným z nich je získavanie, triedenie a následne použitie získaných informácii podstatou pre bežného používateľa Internetu ale samozrejme aj pre podnikanie.

Služba Internetu rastie obrovským tempom a v ďalších rokoch sa tento smer podľa všetkého tak skoro nezmení a tým pádom rastie obrovské množstvo informácií. Bežný užívateľ sa tak môže cítiť bezradné a v tom mu môže mu pomôcť web mining.

Web Mining môže byť široko definovaný ako objavovanie a analyzovanie užitočných informácií z WWW (World Wide Web), ako aplikácia data-miningových technológii k obrovskému skladu webových a iných dát. Práve preto sa pre získavanie požadovaných informácií venuje veľa času na vývoj aplikácií, ktoré tieto informácie vedia získať, uložiť a následne ich využiť ďalej. A to uľahčujú užívateľom množstvo práce, času ale aj peňazí.

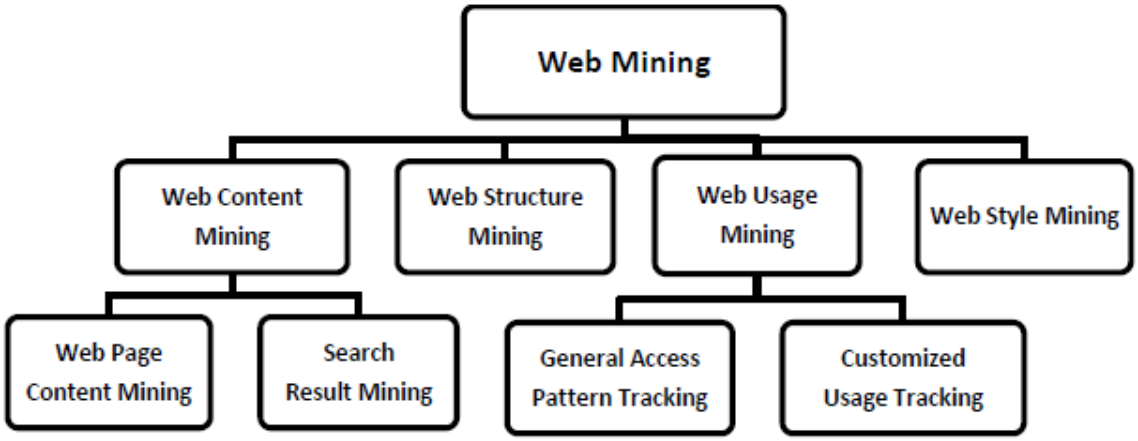
2. Web Mining

Web Mining [1] je extrakcia zaujímavých a potenciálne užitočných vzorov, zákonitosti a skrytých informácií z artefaktu alebo aktivít prevádzkovaných na WWW (Word Wide Web). Web mining je súčasťou odvetvia nazvaného Data Mining (dolovanie či vyťažovanie dát), ktoré vzniklo z nutnosti spracovávať nepreberné množstvo dát a v nich objavovať skryté, hlbšie vzťahy. K tomu využíva najmocnejších poznatkov a technológii.

Data Mining znamená veľa rôznych postupov a algoritmov, ktoré umožnia odhaliť užitočné vzťahy ukryté v dátach. Neexistuje žiadna zázračná metóda, ktorá rieši všetky úlohy s ľubovoľným typom dát. Pre rôzne úlohy a dáta sa hodia rôzne metódy. Veľmi častým prípadom je, že najlepších výsledkov dosiahneme vhodnou kombináciou rôznych metód. Typickými úlohami Data Miningu sú detekcia podvodu, profily zákazníkov, udržanie zákazníka, určenie diagnózy, analýza časových rad, analýza prehľadávania stránok na Internete. Sú približne tri až štyri oblasti odhaľovania znalosti, ktoré patria do Web Miningu a sú to tieto: Web Content Mining, Web Structure Mining, Web Usage Mining a Web Style Mining.

2.1 Metódy Web Miningu

- Web Content Mining [4] tvorí samotný obsah webových stránok, ktorý dáva informácie užívateľom napr. video, audio, grafika, text (html).
- Web Structure Mining analyzuje hyperlinkovú štruktúru webovej stránky a webu. Používa k tomu html tagy, ktoré prepájajú jednotlivé stránky navzájom.
- Web Usage Mining doluje z dát, ktoré reflektujú používanie webových stránok napr. históriu vo webových vyhľadávačoch, logy, proxy servere..
- Web Style Mining je proces zaoberajúci sa designom WWW stránok.



Obr.1 : Metódy Web Miningu

Data mining taktiež hľadá vzťahy a vzory, ktoré majú potenciál stať sa všeobecnými pravidlami a zároveň tieto vzťahy a vzory uplatňuje na dáta neznáme. Ciele data miningu a web miningu sú v tomto prípade veľmi podobné.

Web mining a data mining taktiež používajú rovnakú metodológiu, ktorá pozostáva z krokov, ktoré musia byť urobené k úspešnému dolovaniu znalostí. Základné kroky sa skladajú z výberu správnych dát na dolovanie, predspracovania dát, objavovania vzorov a následnej analýzy. Web mining si tieto fázy prispôsobil a rozšíril pre svoje potreby. Data mining a web mining sa tiež zhodujú v niektorých nástrojoch používaných k interpretácii objavených vzorov. Patria k nim vizualizácia, dotazovacie mechanizmy, OLAP [20] (Online Analytical Processing), expertné systémy a pod.. Za posledné desaťročia bol data mining úspešne použitý vo výskume vedenom v oblasti World Wide Web. Postupom času sa vyvíjali niektoré špecifické črty, čo viedlo k odčleneniu tejto disciplíny od data miningu. Preto sú metodológia, cieľ a niektoré z nástrojov oboch disciplínach veľmi podobné alebo dokonca totožné.

2.2 Web Content Mining

Jedná sa o extrahovanie užitočných informácií z obsahu, dát a dokumentov na webe. Obsah webu môže zahŕňať rozličné typy dát, ako aj techniky, ktoré sa dajú aplikovať vo web

content miningu sú rôznorodé. Obsah webu je neštruktúrovaný, prípadne pološtruktúrovaný a môže zahŕňať text, video, audio, obrázky, zoznamy a tabuľky. Na základe obsahu webu je dolovanie ešte rozdelené na text mining a multimédia mining. Multimédia podľa Giudiciho [24] (2003) vzbudzujú veľký záujem vedcov napriek tomu, že dolovanie dát z multimédií je iba v začiatkoch. Srivastava (2008) ďalej rozdeľuje web content mining na dve oblasti – vyhľadávanie informácií a spracovanie prirodzeného jazyka.

Cieľom oboch oblasti web content miningu je pomôcť užívateľom nájsť užitočné a relevantné informácie. Tieto informácie môžu byť taktiež využité pre potreby vyhľadávačov a personalizáciu webu. Cieľom spracovania prirodzeného jazyka je skúmanie textov alebo hovoreného slova, ktoré vyžaduje určitý stupeň porozumenia prirodzenému jazyku.

2.3 Web Structure Mining

Web structure mining analyzuje vzájomné prepojenie stránok, kde jednotlivé stránky predstavujú uzly a odkazy spojnice týchto stránok. Pri web structure miningu je snaha o objavenie štruktúry v oblasti odkazov (hyperlink) a dokumentov ako aj o nájdenie modelu, ktorý kopíruje štruktúru odkazov na stránke, prípadne vo vnútri stránky. Dizajnéri obyčajne navrhujú stránku používajúc pri tom rozdelenie obsahu a jeho prepojenie na základe logickej štruktúry. Použitá logická štruktúra môže analytikovi pomôcť odhaliť spojitosti medzi stránkami z pohľadu autora.

Príkladom takejto logickej štruktúry je strom a graf. Na základe analýzy odkazov je založená aj jedna z najznámejších techník hodnotenia stránok - pagerank, ktorá číselne vyhodnotí relatívnu významnosť stránky a podľa tejto hodnoty stránku radí na popredné alebo menej významné miesto vo výsledkoch vyhľadávania. Web structure mining sa tiež používa na optimalizáciu firemných stránok a na zisťovanie tematickej podobnosti.

2.4 Web Usage Mining

Cieľom web usage miningu [19] je snaha o pochopenie spôsobu akým sa užívateľ pohybuje po webových stránkach, uľahčiť mu tento pohyb a pomôcť mu s nájdením relevantných informácií.

Web usage mining využíva viacero zdrojov – najčastejšie sa jedná o logy zo strany serveru, proxy serveru alebo dáta od samotného užívateľa. Každé s týchto dát poskytujú iné informácie, a preto je vhodné voliť ich výber s ohľadom na cieľ analýzy. Dáta slúžia podľa Kosaly [21] a Blockeela [21] (2000) k dvom hlavným účelom - k tvorbe profilu užívateľa a k objavovaniu vzorov správania. Ostatné aplikácie ako personalizácia webu, dynamické stránky, zlepšenie výkonu a bezpečnosti, zlepšenie hodnotenia stránok a ich reorganizácia sa dajú odvodiť od predchádzajúcich dvoch kategórií. Napriek tomu, že existujú tri oblasti web miningu, kde každá z nich používa iný typ dát, nie sú v praxi od seba striktne oddelené, skôr sa prelínajú.

Napríklad pri optimalizácii pre vyhľadávače je potrebné optimalizovať nielen obsah stránok, ale aj ich štruktúru a odkazy medzi nimi. Optimalizácia obsahu je síce hlavnou časťou Search Engine Optimalization (SEO), ale zároveň nie je jedinou. Tým, že je optimalizovaný obsah a štruktúra zároveň, tak dochádza k prelínaniu do dvoch oblastí web miningu - web content mining a web structure mining. Podobný príklad kde sa prelína web structure mining a web usage mining je optimalizácia štruktúry na základe zistených vzorov správania z clickstream analýzy.

Z uvedených príkladov je vidieť, že na jednotlivé oblasti web miningu sa nedá pozerať ako na oddelené celky, ale je vhodné informácie z nich kombinovať tak, aby sme dostali ucelený obraz. V mnohých prípadoch je táto kombinácia nutná k správnemu interpretovaniu získaných informácií.

2.5 Web Style Mining

Web Style Mining sa zaoberá analýzou štýlu a prezentácie webových stránok. Zo štýlu dokumentov, ktoré sú zobrazené webovým prehliadačom, môžu byť vytiahnuté cenné

informácie. Táto metóda je pomerne nová a rozvíja sa predovšetkým v Azií. Web Style Mining je možné aplikovať v oblastiach: selekcia založená na charakteristických štýloch, indexovanie štýlov, generovanie štýlov a vyhľadávanie štýlov.

3. Zdroje dát pre Web Mining

Data pre Web Mining môžeme zhromažďovať z mnohá zdrojov. Jedným zdrojom dát sú obsahy webových stránok, odkiaľ môžeme získať zobrazovaný obsah, meta popis stránky, WWW odkazy, URL a jeho štruktúru, atd. . Druhým zdrojom sú záznamy a dáta o chovaní užívateľa, ktoré sa automaticky ukladá v logovacích súboroch, tieto súbory sú na strane serveru, proxy serveru alebo na strane klienta. Data z týchto zdrojov sa líšia v ich pôvodu a klasifikácií. Môžeme ich rozdeliť do štyroch skupín.

Obsah – Data, ktoré sú určene k tomu aby boli prezentovane užívateľom. Sú to dáta, ktoré sa nachádzajú na webových stránkach, skladá sa z textu a grafiky, pričom najväčší význam pre analýzu ma textová zložka. Zdrojom informácií je aj obsah hlavičky www stránky, ktorý môže obsahovať cenné informácie.

Štruktúra – Usporiadanie informácií, ktoré charakterizuje štruktúru obsahu. Medzistránková štruktúra je tvorená prostredníctvom hyperlinkov, ktoré spojujú stránku s ostatnými . Usporiadanie HTML a XML tagov tvorí vnútrostránkovú štruktúru.

Užívanie – Sú to dáta, ktoré popisujú vzory užívania webových stránok. Sú to IP adresy, dáta a časy prístupov atď. Užívateľské dáta pochádzajú z rozšírenia bežného log formátu (ECLF – Extended Common Log Format).

Užívateľský profil – Sú to dáta, ktoré poskytujú demografické informácie o užívateľoch webových stránok. Sú to registračné dáta a ďalšie informácie o užívateľoch..

3.1 Problémy a obmedzenia Web Miningu

Získavanie dát zo serveru je obmedzené vlastnosťami logovacích súborov, ktoré boli pôvodne vytvárané pre účely ladenia. Logovacie súbory obsahujú veľké množstvo neužitočných informácií a na druhu stranu v nich niektoré užitočné informácie môžu chýbať. Bežne je používaný Common Log Format, lepšie je používať formát Extended Log Format.

- Logy neukladajú informácie o požiadavkách, ktoré boli zachytené pri použití webových alebo proxy cache.
- Logovacie súbory ukladajú iba URL stránok a nie ich sémantický popis, tiež neobsahujú dáta z webových formátov.
- Problémy prenáša aj identifikácie užívateľa, pokiaľ nie je žiadna použitá, tak ako dáta môžu byť skreslene, pretože za jednou IP adresou môže byť skryto viac užívateľov alebo naopak jeden užívateľ môže vystupovať pod viac IP adresami.
- U identifikácie sadenia môže byť problém so zaisťovaním času, kde bola stránka opustená. Problémy prináša stránky, na ktorých sú použité rámce, alebo dynamické stránky.
- Získavanie dát na úrovní klienta závisí na jeho spolupráci.
- Určité problémy nastupujú aj oblasti ochrany súkromia užívateľov.
- Inými kategóriami problémov môžu byť problémy s náročnosťou prevedených výpočtov.

3.2 Techniky Web Content Mining-u

Metódy, ktoré sa pri riešení tohto typu úloh využívajú, sú najmä crawlery, indexácia a data miningové techniky, menovane klasifikácia a zhlukovanie, založené na textovej substancii webových dokumentov. Ďalej sa budeme venovať popisu zvyšných techník dolovania dát z obsahu webu.

K problematike dobývania znalostí z obsahu webu sa môžeme postaviť z dvoch základných hľadísk. Prvým je dolovanie s využitím vyhľadávacích agentov, z ktorých sú najznámejšie Crawlery, Bookmark Organizer a Shopboti a HyPursuit.

Crawlery[22], tiež známe pod názvom „webové pavúky", sú programy alebo automatizované skripty, ktoré systematicky prehľadávajú webový priestor. Celý webový priestor pritom vnímajú ako orientovaný graf a prehľadávajú ho buď do šírky (BFS - Breadth First Search, implementácia pomocou fronty), alebo do hĺbky (DFS - Depth First Search, implementácia pomocou zásobníka). Dôležité je plnenie dátovej štruktúry URL odkazmi na prehľadávané stránky, ktoré sa následne na to spracúvajú. Z toho dôvodu je presnejšie vymedzenie pojmov uvedené v nasledujúcej vete. Crawlery sú využívané hlavne k vytvoreniu kópií navštívených stránok pre ich neskoršie spracovanie vyhľadávacími strojmi. Dôležitú rolu pri tom zohráva indexácia navštívených stránok vzhľadom k budúcemu rýchlejšiemu vyhľadávaniu.

Indexácia - proces vytvárania indexov nad obsahom jednotlivých web stránok s účelom rýchlejšieho vyhľadávania. Invertované súbory patria v súčasnosti k najpoužívanejšej technike indexovania webu. Na vstupe máme kolekciu dokumentov, ku ktorým vytvoríme slovník použitých slov. Každé slovo sa označí jeho pozíciou v dokumente a záznamy výskytov sa priradia ku každému slovu v slovníku. Vyhľadávanie potom prebieha invertované na základe vytvoreného slovníka. Okrem tejto techniky sa využíva indexácia pomocou príponových stromov, ktorá je vhodná pre prácu s frázami a to najmä preto, že poskytuje rýchlejšiu odozvu. Výrazným nedostatkom je však náročná tvorba a udržiavanie konzistencie týchto stromov.

Druhý spôsob dolovania využíva multiúrovňové databázy a webové dotazovacie systémy. Multiúrovňové databázy samostatne organizujú čiastočne štruktúrované údaje na webe do štruktúrovaných zbierok zdrojov.

Bookmark Organizer kombinuje techniky hierarchického zhlukovania s interakciou používateľa. Informácie takto získané využíva na organizáciu dokumentov na webe.

HyPursuit využíva sémantické informácie v štruktúre odkazu a v obsahu dokumentov. Jeho cieľom je pochopiť vzťahy hypertextových dokumentov a štrukturovať' informačný priestor.

ShopBoti sú vyhľadávací agenti, ktorí pracujú na princípe získavania informácií o produktoch z rôznych stránok predajcov. Opierajú sa o strategické informácie v sfére produktov.

Multiúrovňové databázy využívajú databázový prístup k organizácií webových informácií. Hlavnou myšlienkou je prítomnosť čiastočne štruktúrovaných informácií na najnižšej úrovni databázy vo forme rôznorodých webových skladov, ktorými sú hypertextové dokumenty.

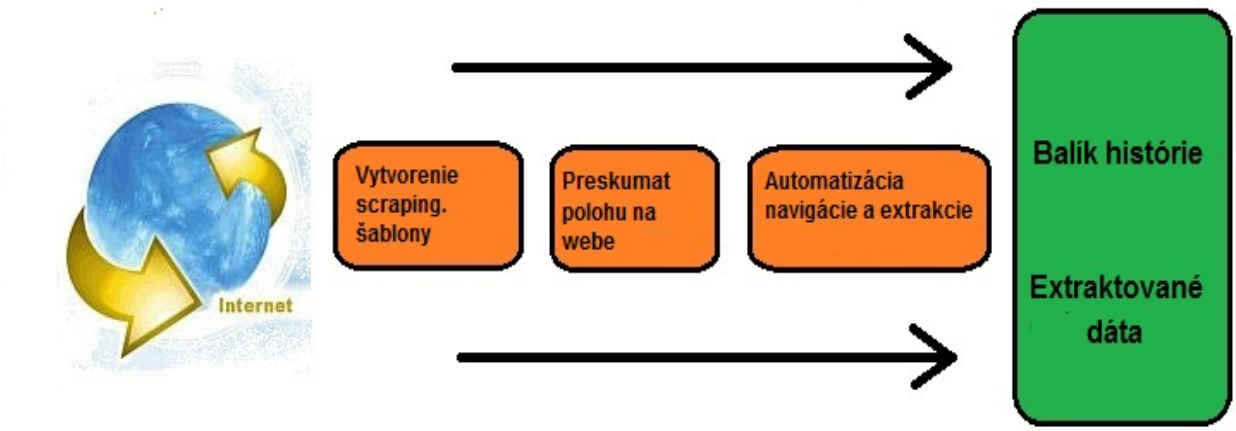
Na každej vyššej úrovni sú metadata alebo generalizácie (zobecnenie vlastností entity) extrahované z predchádzajúcich nižších úrovni. Tieto informácie sú štruktúrované do relačných, prípadne objektovo orientovaných databáz.

Webové dotazovacie systémy sú systémy, založené na funkcionalite štandardných dotazovacích jazykov (SQL), štrukturálnom usporiadaní webových dokumentov ako aj spracovaní prirodzeného jazyka. Štruktúra a funkcionalita takýchto systémov je podriadená typu vyhľadávacích dotazov používaných na webe.

3.3 Web scraping

Web scraping označuje akýkoľvek spôsob extrakcie internetovej stránky za účelom jeho ďalšieho spracovania v inom rozložení či inom formáte. Web scraping sa využíva od budovania vlastných databáz informácií získaných na Internete až po automatizované získavanie čerstvých dát a ich okamžité využívanie napr. pohyby cien komodít ako sú ropa, zlato ale aj o stave kurzu meny napr. eura, doláru a podobne. Získanie týchto dát pomáha byť o krok napred a tým pádom mať stále aktuálne informácie, s ktorými je možno ďalej pracovať. Web scraping pracuje priamo so zdrojovým kódom webovej stránky. Pracuje so statickými webovými stránkami a každú z nich spracováva len raz. Typickým príkladom web scrapingu je web crawler, ktorý kopíruje obsah jednej ale viac webových stránok a za rôznym

´čelom generuje scraper site. Web scraping, ktorý z webových stánok vyťahuje odkazy na ďalšie stránky, ktoré nasleduje a tiež spracováva sa nazýva web harvesting. Takéto programy sa označujú termínom webbot, crawler, spider.



Obr. 2: Princíp Web scrapingu

Indexácia - proces vytvárania indexov nad obsahom jednotlivých web stránok s účelom rýchlejšieho vyhľadávania. Invertované súbory patria v súčasnosti k najpoužívanejšej technike indexovania webu. Na vstupe máme kolekciu dokumentov, ku ktorým vytvoríme slovník použitých slov. Každé slovo sa označí jeho pozíciou v dokumente a záznamy výskytov sa priradia ku každému slovu v slovníku. Vyhľadávanie potom prebieha invertované na základe vytvoreného slovníka. Okrem tejto techniky sa využíva indexácia pomocou príponových stromov, ktorá je vhodná pre prácu s frázami a to najmä preto, že poskytuje rýchlejšiu odozvu. Výrazným nedostatkom je však náročná tvorba a udržiavanie konzistencie týchto stromov.

Možnosti webbotov využíva napríklad vyhľadávač Google, ktorý z ich pomocou zisťuje, na ktorej webovej stránke je najviac smerovaných odkazov a podľa toho im potom priraďuje váhu, ktorá ovplyvňuje poradie vo vyhľadávači.

4. Koncepcia systému

Zmyslom nasledujúcej kapitoly je popis implementácie systému, ktorý bude vytvárať prostredie pre implementáciu web data miningu. Prvá aplikácia získava informácie z internetu, predovšetkým webových stránok. Tieto informácie v priebehu času obnovuje a ukladá ich aktualizácie. Druhým produktom nie je aplikácie ako taká, ale jedna sa o API sadu tried umožňujúce jednoduché získavanie obsahu z celých webových prezentácií. API bude poskytovať prostredie, ktoré umožní zaistiť prácu s HTTP a HTML bez ich konkrétnej znalosti. Posledným programovým dielom je AWT aplikácia, ktorá umožní jednoducho a interaktívne prejsť celý proces web dat miningu od fázy získania obsahu až do fázy vizuálnej prezentácie.

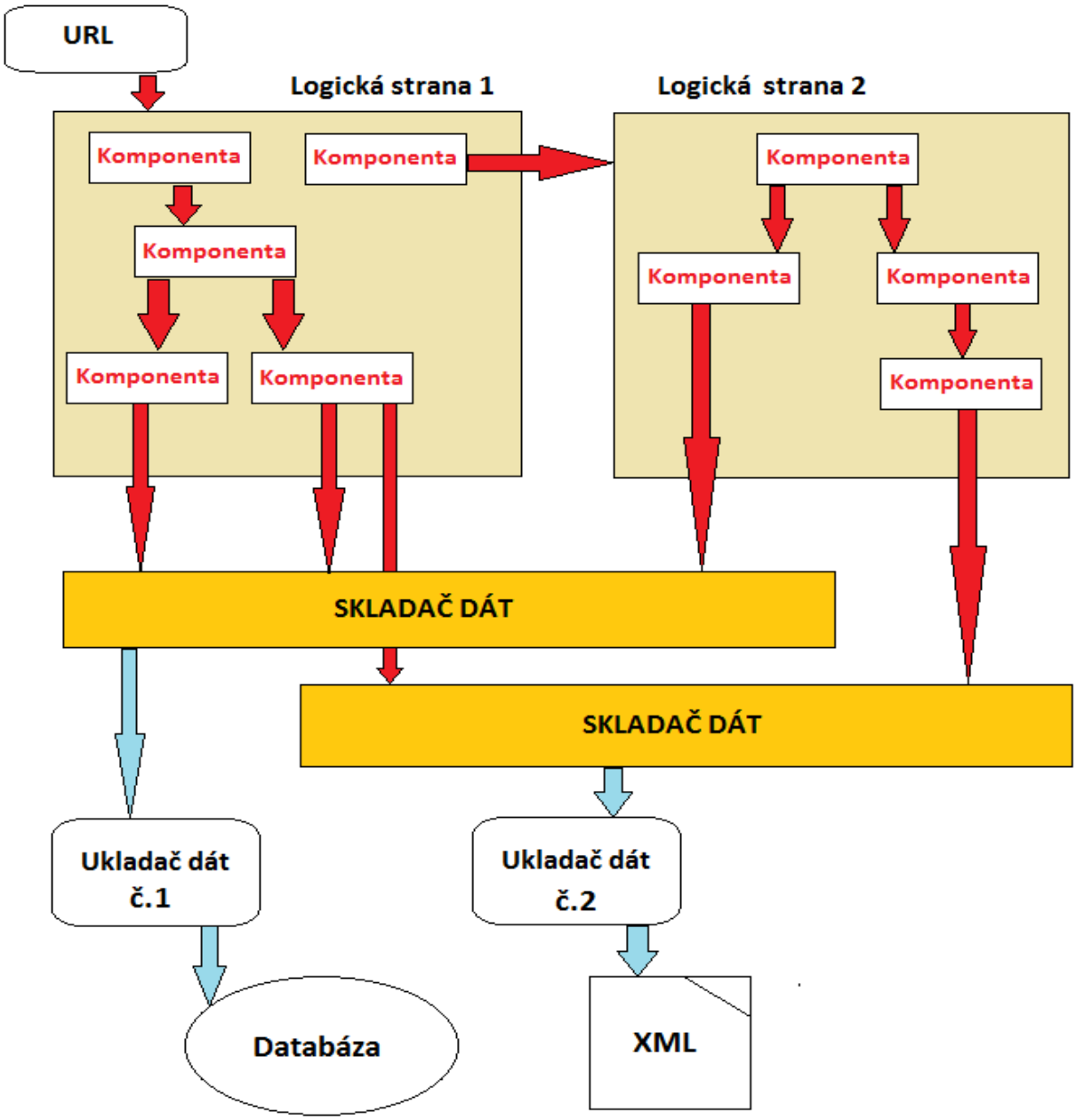
Aby nedošlo k tomu, že aplikácia bude získavať dáta bez akýchkoľvek ďalších nástrojov a nedochádzalo tak k veľkej množine úloh, ktoré sa neustále opakujú bolo usúdeno, že dobrým prístupom k riešeniu problému bude vytvorenie komponent, ktoré budú tieto opakujúce sa úlohy vykonávať. Prepojovaním týchto komponent tak vznikne celkový popis úlohy, ktorá má byť na webovej stránke, v celom webe prevedená. Komponentou sa rozumie akákoľvek časť nami vytvorených prostriedkov (AWT aplikácie, API..) Tieto komponenty môžu byť nezávisle kombinované a upravované tak, aby bolo možné dosiahnuť požadované výsledky. Medzi tieto komponenty môžu vstupovať aj komponenty tretích strán, ako napríklad databázy, webové servery, prostriedky systému (Cron..).

Príkladom môže byť komponenta, ktorá na vstupe získa HTML stránku zo zoznamom žiakov a prehľadom známok. Na výstupe takejto komponenty bude dátová množina priebehu známok jednotlivých žiakov. Popísať úlohu by malo byť ľahké a ich zápis prehľadný a dobre upravovateľný, pretože sa stránky menia a požiadavky, čo z nich získavame tiež.

Ďalej je vhodne, aby užívateľom bola práca z veľkej časti obmedzená na deklaratívne popisovanie úlohy, teda na prepojovanie jednotlivých komponent. Ako vhodný kandidát pre takýto zápis je XML, vďaka svojej širokej podpore a možnosti nad ním vybudovať editačné nástroje. Môže sa stať, že komponenta bude vedieť z veľkej časti to čo je potreba, ale bude po nej vyžadované trochu iné chovanie. V takomto prípade je zbytočne vytvárať novu komponentu podobne k tej existujúcej, ktorá by mala takmer identický kód, ale iba určíte časti by boli odlišné.

Kvôli tomu bude možne do komponent na tieto miesta, ktoré budú pravdepodobne často menená pridať udalosti a na ne programovo reagovať v priloženom zdrojovom súbore, poprípade v AWT aplikácií. Pokiaľ je dopredu známa i množina zmien, ktoré môže užívateľ vyžadovať, je možné chovanie komponenty ovplyvňovať vlastnosťami.

V systéme budú zabudované štandardné komponenty, ktoré budú využívať pre spracovanie HTML stránky dopredu definovane sady príkazov.



Obr. 3: Koncepcia systému

Celkový popis úlohy by mohol byť popísaný jedným veľkým prepojením komponent, to by však bolo neprehľadné. Preto bude tento popis rozdelený na tzv. logické stránky, ktoré budú združením komponent, ktoré sa budú aplikovať na určitú HTML stránku. Ďalej bude navigácia spočívať v nasledovaní odkazu získaných z obsahu a ich spracovaním logickými stránkami. Štruktúra webových stránok môže byť často podobná len sa odlišujú prvky na stránke. Z pozorovania sa napríklad ukázalo, že mnoho stránok, ktoré vyzerajú úplné inak, ale majú dáta zobrazene v nejakom zozname. Ďalej je potreba vyriešiť zloženie z komponent získaných dát, pretože informácie o nejakej veci môže byť roztrúsená po viacerých stránkach.

Takto roztrúsenú informáciu je nutné zložiť, aby šla uložiť ako jeden záznam, ako už riadok tabuľky alebo čokoľvek iného. Na to bude špeciálna komponenta, ktorá bude výhradne v réžii užívateľa / programátora, ktorá bude najčastejšie používaná ako posledná v reťazci volania komponent. Data si bude priebežne ukladať do nejakej cache a až bude mať všetko potrebné, záznam zašle k uloženiu. Data, ktoré už boli poskladané, je nutné niekam uložiť. K tomuto účelu bude v systéme rozhranie, ktoré si môže užívateľ systému zahrnúť do nejakej svojej triedy alebo využiť existujúci implementáciu, ktorá bude ukladať dáta do pamäti. S nimi bude možné po dokončení sťahovania pracovať. Je možné, že bude potreba dáta rovno v priebehu sťahovania ukladať do finálneho dátového úložiska, napríklad u dôvodu nedostatku pamäti pre veľkú úlohu. V takomto prípade si užívateľ vytvorí vlastnú implementáciu triedu pre ukladanie dát.

Ďalším požiadavkom na systém je rýchlosť, ktoré sa najlepšie dosiahne využitím paralismu. K tomu je potrebná celkovú úlohu rozdeliť na menšie moduly, ktoré môžu byť vykonávané paralelne. Najpomalejším procesom v systéme je komunikácia s HTTP serverom, preto je najlepšie úlohy deliť práve tu. To už ale bolo urobené rozdelením prepojením komponent do logických stránok, ktoré sa v tomto kontextu dajú brať ako úlohy, ktoré budú spúšťané na rôznych vláknach.

4.1 API - Application programming interface

Každá aplikácia pre svoje správne pracovanie využíva podporné prostriedky, ktoré jej ponúkajú aplikačné prostredie v ktorom je vytvorené. V priebehu vývoja našej aplikačnej základne pre WebDataMinning sme využívali celého spektra externých API od rôznych dodávateľov.

Rovnako tak ako iný programátori, tak aj my sme dospeli do štádia, kde naša práca nesie účelovú sadu metód a prístupu, ktoré sa často opakujú. Nastala preto najlepšia chvíľa pre návrh vlastného podporného systému. Systém je využívateľný aj pre ďalšiu tvorbu web dataminingových aplikácií.

Prvotnou úlohou bolo klasifikovať štruktúru webových prezentácií. Na základe tejto štruktúry navrhnúť model prístupu k získavaniu informácií. Veľmi často využívaná architektúra webových prezentácií, je architektúra stromového typu. Kde primárnym miestom a základným rozcestníkom je index webovej stránky (úvodná stránka). Odkazy na neho uvedené nás najčastejšie presmerovavajú na jednotlivé kategórie. Každá kategória obsahuje penzum odkazu na finálne html dokumenty, ktoré ukrývajú nami požadované informácie.

{OBRAZOK STROMOVEJ STRUKTURY KMEN = INDEX, VETVA = KATEGORIE}

Na základe tohto modelu došlo k návrhu architektúry vytvoreného API. Z pohľadu programátora musí byť práca s API pokiaľ je to možné čo najjednoduchšia a najkompaktnejšia. Z toho dôvodu sme zvolili prístup na báze projektového modelu.

Spracovaná webová prezentácia je zaistená všeobecne majúcim projektom. Ten obsahuje všetky spracované HTML stránky a získané informácie. Projekt ďalej ponúka široké spektrum služieb(metód) pre prechádzanie, filtrovanie webovej stránky a parserovanie obsahu.

4.1.1 API z pohľadu programátora – príručka

Po importovaní nášho API a všetkých potrebných knižníc, môže začať budovanie WebDataMinning aplikácií.

Nastavíme si modelovú situáciu. Web na adrese http:/www.web.cz ma na úvodnej stránke celú radu odkazov. Časť odkazov smeruje do kategórií o ktoré sa zaujímame. Iné odkazy smerujú na obsah, pre nás nezaujímavý. Samotné kategórie prezentuje penzum odkazu na informačné zdroje, ktoré sú pre nás dôležité. Samotný informačný zdroj, prezentuje štandardnú HTML stránku obsahujúcu nami hľadanú informáciu.

Rozbor webovej prezentácie. Pri rozbore webovej prezentácie musíme urobiť nasledujúce kroky:

1. Identifikovať vstupný bod.
2. Definovať predpis a charakteristiku stránky, ktorá obsahuje získavané dáta.
3. Na základe architektúry webu definovať cestu k požadovanému obsahu.
4. Identifikovať presnú pozíciu hľadaných dát v rámci HTML stránky.

- Vstupným bodom do celej webovej prezentácie je url adresa http://www.web.cz/.
- Požadované informácie sa nachádzajú na stránkach kde ich url adresa je v nasledujúcom formáte: http://www.web.cz/hdp-stat-id.html Táto adresa je klasickým použitím SEO adresy. Skladá sa z troch častí oddelených pomlčkami:
    - hdp – skratka slova hrubý domáci produkt.
    - Textové označenie štátu.
    - id – identifikátor daného štátu používaného interne v rámci webovej prezentácie.
- K požadovanému obsahu (súbor s HDP pre jednotlivé štáty) musíme hľadať cestu z východzieho bodu.
    a. Začneme načítaním úvodnej stránky a získaním zoznamu odkazov.

b. Vyfiltrujeme len tie odkazy, ktoré vedú na kategórie obsahujúce odkazy na stránky s HDP. Predpis ich url je: http://www.web.cz/kontinent-nazovKontinentu-id.

c. Načítame tieto kategórie a získame odkazy. Vyfiltrujeme odkazy tak, aby nám zostali len odkazy vedúce na stránky obsahujúce HDP jednotlivých štátov.

d. Načítame stránky obsahujúce informácie o štátoch a ich HDP.

- Z danej stránky budeme získavať tri hodnoty (názov štátu, HDP, letopočet). Pre každú hodnotu urobíme rozbor HTML dokumentu a identifikujeme ich presnú pozíciu.

Na základe predošlého rozboru môžeme pristúpiť k riešeniu načrtnutej situácii za pomocou API. Prvým krokom bude inicializácia projektu a nastavenie kódovania.

```
HtmlProject htmlProject = new HtmlProject();

htmlProject.setCharset("utf-8");
```

Do projektu musíme zaniesť bod jedna nášho rozboru. Vstupný bod predstavuje štandardnú url adresu. Tu do projektu zanesieme pomocou volania getMainUrlStack(). Táto metóda vracia objekt UrlStack obsahujúci zoznam spracovaných url adries. Pomocou metódy addUrl volanej na objekte typu UrlStack, pridáme prvú spracovanú url adresu.

```
      htmlProject.getMainUrlStack().addUrl("http://www.web.cz/");
```

Teraz musíme stránku spracovať alebo stiahnuť ju a získať zoznam url adries, ktoré obsahuje.

```
      htmlProject.download();

      htmlProject.collectAllNewUrl();
```

Podľa nášho zadania a rozboru vieme, že úvodná stránka obsahuje dva druhy odkazov.

- Hľadáme odkazy na kategórie.
- Ostatné odkazy, ktoré pre nás nemajú zmysel.

Našou úlohou je urobiť filtrovanie odkazov tak, aby boli odstránené všetky odkazy spadajúce do druhej kategórie.

```
UrlStack urlList = htmlProject.getMainUrlStack().select("kontinent-");

// Vloži vyfiltrovaný zoznam url adries spat do projektu

htmlProject.setMainUrlStack(urlList);
```

Následné potom čo sme zaktualizovali zoznam url adries len na tie, ktoré pre nás majú zmysel, pristúpime k odstráneniu starých stránok z pamäti projektu. Obsah stránky bude odstránený, jedine čo sa v projekte zachová, je jeho url adresa. Ta je po stiahnutí zapísaná do zásobníka adries stiahnutých stránok. Tento zásobník je dôležitý predovšetkým z nasledujúceho dôvodu. Každá novo nájdená adresa je overovaná so zoznamom už stiahnutých stránok. Tým zabránime duplicitnému sťahovaniu obsahu a blúdeniu v kruhu. Ďalej nasleduje stiahnutie všetkých nových stránok a získanie všetkých ďalších url adries.

```
      htmlProject.clearPages();

      htmlProject.download();
```

4.1.2 Finálne filtrovanie adries

Načítame kategórie obsahujúce celu radu odkazov. Pomocou nasledujúceho kódu dôjde k vyfiltrovaniu iba tých url adries, ktoré vedú na stránky s hľadanými dátami. Požadované stránky následne stiahneme.

```
      urlList = htmlProject.getMainUrlStack().select("hdp-");

      htmlProject.setMainUrlStack(urlList);

      htmlProject.clearPages();

      htmlProject.download();
```

V tejto chvíli projekt obsahuje len tie stránky obsahujúce nami hľadané dáta. Teraz pristúpime k hľadaniu obsahu jednotlivých informácií.

Hľadaná informácia: názov štátu. Na základe rozboru html dokumentu vieme, že názov štátu je vypísaný v tele HTML tagu SPAN s id = "štát". Vďaka tejto informácií môžeme previesť jednoznačne k odstráneniu všetkého čo je mimo tag. K odstráneniu všetkého čo je pred hľadaným obsahom použijeme funkciu cutBefore("co") volané priamo na projekte. Táto funkcia uskutoční odstránenie obsahu vo všetkých stránkach, ktorá projekt obsahuje. Obdobne funguje funkcia cutAfter("co"), pre odstránenie všetkého obsahu za hľadanou informáciou. Po odstránení obsahu pred a za hľadanou informáciou danej stránky, obsahuje len tu časť svojho pôvodného obsahu, ktorá nás zaujíma. Uložením tohto obsahu dosiahneme pomocou volania funkcie storeContent("nazov"). StoreContent uloží obsah webových stránok do HashMapy a ako kľúč použije reťazec "nazov".

Pred samotným hľadaním je potrebné previesť jeden dôležitý krok a to zálohu obsahu. Záloha obsahu sa robí z dôvodu nezvratných zmien v telách stiahnutých stránok (cutAfter, cutBefore). Zálohu prevedieme pomocou metódy createBackup(). Tá obsah zálohuje pre neskoršou obnovou. Neskoršiu obnovu týchto stránok využijeme v prípade, že z obsahu stránky získavame viac ako jednu informáciu.

```
// Vytvorenie virtualnej zalohy obsahu
      htmlProject.createBackup();

      // Odstranenie obsahu všetkých stránok pred a za daným reťazcom -
Názov dielu
      htmlProject.cutBefore("<span id =\"stat\">");
      htmlProject.cutAfter("</span>");
      htmlProject.storeContent("nazevStau");

      // Nahranie zálohy a získavanie ďalšieho obsahu
      htmlProject.loadBackup();

      htmlProject.cutBefore("<span id =\"information\">");
      htmlProject.cutAfter("</span>");
      htmlProject.storeContent("hdp");
```

4.1.3 Export dát do databáze

Pre persistenciu získavaných informácií obecne používané databázové systémy. My sme zvolili SQL server, distribúcie MySQL. Volaním metód createSQL(“tableName“) zaistíme vygenerovanie SQL príkazu, ktoré vloží zistené dáta do databázového systému.

```
// Generovani SQL dotazu.
      htmlProject.createSQL("movie_epizoda");
```

Tým sme pomocou niekoľko riadkov kódu pracujúcich s našimi API dokázali získať množstvo špecifických informácií, bez nutnej znalosti práce http clienta, parserovania html obsahu a ďalších technológií.

4.1.4 Celková ukážka kódu

Nasledujúca ukážka kódu demonštruje komplexní použitie nášho API. Jeho implementácia po programátorovi vyžaduje iba základnú znalosť programovania a orientáciu v oblasti web data miningu. Kód je doplnený stručnými komentármi.

```
// Aplikáciu sme umiestili do balíčkku cz.newpackage.ukazka
      package cz.newpackage.ukazka;

// Import tried potrebných pre využívanie nášho API
      import cz.newpackage.HtmlProject;
      import cz.newpackage.UrlStack;

// Deklarácia verejnej a spustiteľnej triedy – slúžia iba pre obal kódu v
statickej metóde
      public class SpustitelnaUkazka {

      // Spustiteľná metóda main
      public static void main(String args[]) {

         // Vytvorenie triedy projektu pre web data minning
        HtmlProject htmlProject = new HtmlProject();
        //   Nastavenie   kódovania,   používaného   na   stránkach   ktoré
spracovávame.
        htmlProject.setCharset("utf-8");
```

```
        // Pridanie prvej a základnej url – index webu
       htmlProject.getMainUrlStack().addUrl("http://www.online-
futurama.cz/");

        // Stiahnutie úvodnej – index stránky
        htmlProject.download();

        // Získanie zoznamu všetkých obsadených  v prvej strane url
        htmlProject.collectAllNewUrl();

        // Prevedie filtrovanie pre url vedúci k dátovým zdrojom
        UrlStack urlList = htmlProject.getMainUrlStack().select("?p=");
// Zmaže stiahnuté stránky zo zásobníka
        htmlProject.clearPages();

        // Vloží vyfiltrovaný zoznam url adres
        htmlProject.setMainUrlStack(urlList);

        //Stiahne všetky nové pridane url adresy
        htmlProject.download();
        // Vytvorenie virtuálnej zálohy obsahu
        htmlProject.createBackup();

// Odstránenie obsahu všetkých stránok pred a za daným reťazcom - Názov
dielu
        htmlProject.cutBefore("<h2>");
        htmlProject.cutBefore(">");
        htmlProject.cutAfter("</a>");

        // Uloženie ostaného obsahu pod daným nazvem
        htmlProject.storeContent("nazev");

        // Nahranie zálohy a získavanie ďalšieho obsahu
        htmlProject.loadBackup();
        htmlProject.cutBefore("</script></p>");
        htmlProject.cutBefore("<p>");
        htmlProject.cutAfter("</p");
        htmlProject.cutAfter("<strong>");
        // Ulozeni zbyleho obsahu pod danym nazvem
        htmlProject.storeContent("html");
```

```
        // Výstup premenných
        htmlProject.printVariables();

        // Generovanie SQL dotazu.
        htmlProject.createSQL("movie_epizoda");

    }

}
```

Táto ukážka kódu je funkčným využitím web data miningu. Konkrétne spracováva webovú prezentáciu na adrese http://www.online-futurama.cz/ .Daná webová prezentácia obsahuje takzvané embed kódy (fragmenty html stránok prehrávajúce audio-video) obsahujúce jednotlivé diely seriálu Futurama. Pomocou našej aplikácie, tieto kódy získame a dokážeme ich exportovať napríklad do MySQL.

4.2 JavaDoc

DownloadCall

Interface DownloadCall pre predávanie správ medzi triedami api. Predávanie správy je založene na implementácií jednou triedou a vlastnením odkazu na objekt inou triedou.

Metódy:

`Void downloaded();`

Metóda predá informácie o stiahnutí daného HTML objektu.

HtmlPage

Trieda predstavuje virtuálny HTML dokument. Je vždy obrazom jednej konkrétnej spracovanej stránky.

Premenné:

`private String url;`

Textová reprezentácia url adresy spracovanej stránky.

```
private UrlStack pageUrl = new UrlStack();
```

Zásobník odkazu, ktoré stránka obsahuje.

```
private String page = null;
```

HTML kód stránky, poprípade textový odkaz danej url adresy.

```
private String pageBackup = null;
```

Záloha HTML kódu stránky, poprípade textového odkazu danej adresy.

```
private HttpClient client = null;
```

Objekt http klienta vytvoreného konzorciom Apache Founadtion. Klient obstaráva všetku prácu s vrstvou http.

```
private HashMap<String, String> values = new HashMap<String, String>();
```

Mapa získaných hodnôt z danej stránky.

```
private String charset;
```

Textová informácia o kódovaní.

```
Private boolean ok = false;
```

Akonáhle je všetko spracované a v priebehu spracovania sa nenastali vážnejšie komplikácie je nastavená hodnota true.

```
DownoaldCall call =  null;
```

Odkaz na triedu primajúca informáciu o dokončení.

Konštruktor:

```
public HtmlPage(String url, String charset)
```

Najprv ukladá informácie o url adrese stránky a kódovaní obsahu stránky a následne spúšťa spracovanie.

`String url` – Url adresa stránky.

`String chrset` – Kódovanie stránky.

Metódy:

Pokiaľ v obsahu pre daný kľúč nájde aspoň jeden výskyt reťazca content, odstráni ho zo zoznamu premenných. Zo zoznamu premenných ho odstráni tiež v prípade, keď je obsah k danému kľúču rovný NULL. Metóda nachádza využitie predovšetkým v oblastiach kde chceme filtrovať už získaný obsah.

`Strin name` – Kľúč k obsahu.

`String content` – reťazec testovaný na výskyt obsahu.

```
public void createBackup()
```

Vytvorí zálohu aktuálneho obsahu. Metóda sa používa pred volaním funkcií orezávajúce obsah.

```
public void loadBackup()
```

Nahraje zálohovaný obsah stránok. Reverzná metóda k metóde createBackup.

```
public void cutAfter(String find)
```

Nájde v obsahu stránky prvý výskyt reťazca a odstráni všetko čo je za prvou pozíciou výskytu reťazca.

`String find` – hľadaný reťazec

```
public void cutBefore(String find)
```

Nájde v obsahu stránky prvý výskyt reťazca a odstráni všetko čo je pred poslednou pozíciou výskytu reťazca.

`String find` – hľadaný reťazec

```
public String getContent()
```

Vráti textový obsah danej url adresy. Alebo null v prípade neúspešného načítania obsahu.

```
public UrlStack getPageUrl()
```

Vráti zoznam všetkých url získaných z aktuálnej stránky. Bližší popis viz. JavaDoc k UrlStack.

`public String getUrl()` – get metóda pre premennou url.

`public HashMap<String, String> getValues()` – get metóda pre vratenie mapy získaných hodnôt.

`boolean isOk()` – get metóda pre premennú ok.

`public String openUrl(String url)`

Metóda zaistí načítanie danej url stránky. Obecne je táto metóda volaná interne ale pomocou definície prístupnosti public má programátor možnosť samostatne vyvolať spracovanie inej url než predané konštruktoru.

`private void processNode(Node node)`

Interná metóda slúžiaca k parserovaniu HTML obsahu a hľadanie odkazov.

`public void run()` – prepis metódy run používane v oblasti viacvláknového spracovania. Metóda riadi získavanie obsadených url adries.

`public void storeContent(String name)`

Aktuálny obsah v premennej page(obsah stránky) uloží do hash mapy pod kľúčom name.

`String name` – kľuč pre uloženie obsahu.

`public void storeContentSQL(String name)`

Vygeneruje SQL dotaz typu insert. SQL dotaz obsahuje upravené hodnoty získane na stránke. Insert sa vkladá do tabuľky špecifikované parametrom name.

`String name` – názov tabuľky.

HtmlProjekt

Najdôležitejšia trieda v celom API. Komplexne zaisťuje pracovné prostredie a obstaráva väčšinu interácie.

Konštruktor:

`HtmlProject()`

Slúži iba k získavaniu inštancie.

Premenné:

```
private String charset
```

Premenná obsahuje kódovanie daného projektu. Defaultne má nastavenú hodnotu kódovania utf-8.

```
private UrlStack main UrlStack
```

Url stack je zásobníkom získaných url adries. Tento hlavný zásobník obsahuje zoznam url v celom projekte.

```
private HashMap<String, HtmlPage>pages
```

Táto hash mapa ako kľuč používa textový reťazec – url obsadenej stránky. Na kľúč je naviazaný objekt typu HtmlPage. V premennej sú uložené všetky spracované (stiahnuté) obsahy url adries. Tieto objekty sú tu uložené po celú dobu života triedy HtmlProject, pokiaľ nie sú zámerne vymazané.

```
public void clearBadVariable(String name, String content)
```

Metóda zaistí volanie rovnomennej metódy clearBadVariable implementované v triede HtmlPage. Táto metóda je zavolaná na všetkých objektoch typu HtmlPage uložených v hash mape pages. Zmysel tejto metódy je aplikovať rovnaké pravidla/operácie na celý obsah spracovaného webu.

```
public void clearPages()
```

Metóda zaistí odstránenie všetkých získaných stránok. Ich url adresy zostanú, aby nedošlo k duplicitnému spracovaniu.

```
public void collectAllNewUrl()
```

Zaistí získanie všetkých nových url adries, zo všetkých stiahnutých stránok. Vkladané adresy sú kontrolované na duplicitný obsah. Tým je zaistené, že získané adresy sú originálne a žiadnu časť webu nespracováva dvakrát.

```
public void createBackup()
```

Metóda prevedie volanie createBackup() na všetkých objektoch HtmlPage uložených v projekte. Tým prevedieme zálohu celého projektu a pripravíme si priestor pre spracovanie webu a získavanie obsahu.

```
public void createSQL(String table)
```

Zavolá rovnomennú metódu na všetkých stránkach uložených v projekte. Jednotlivé časti výsledkov spoji do jedného celku a ten vypíše do štandardného výstupu. Pomocou tejto metódy získame kód pre uloženie získaných informácií do databáze SQL.

```
public void cutAfter(String find)
```

Metóda volá metódu cut After(String find) na všetkých objektoch HtmlPage obsadených v projekte.

```
public void cutBefore(String find)
```

Metóda volá metódu cut Before(string find) na všetkých objektoch HtmlPage obsadených v projekte.

```
public void download()
```

Metóda získa zoznam všetkých novo pridaných a nezapracovaných url v projekte (mainUrlStack). Pre každú url adresu získa novú inštanciu triedy HtmlPage a následne stiahne obsah danej url adresy. Po stiahnutí obsahu prevedie analýzu nových odkazov.

```
private String getCode(String codeJs)
```

Interne používaná metóda. Niektoré webové stránky majú svoj obsah skrytý v JavaScripte. Táto metóda dokáže pomocou pridaných knižníc vykonávať JavaScript kód. Daná metóda prevažne experimentálna a nie je vhodné ju využívať.

```
public UrlStack getMainUrlStack()
```

Metóda vráti hlavný zásobník url adries.

```
public HashMap<String, HtmlPage> getPages()
```

Vracia hash mapu, kde je pod reťazcom obsahujúcim url adresu uložený objekt HtmlPage reprezentujúci stiahnutý obsah.

```
public String getSQL(String table)
```

Zavolá rovnomennú metódu na všetkých stránkach uložených v projekte. Jednotlivé časti výsledkov spoji do jedného celku a ten vráti. Pomocou tejto metódy získame kód pre aktualizáciu, už získaných hodnôt v databáze SQL.

```
public String getUpdate()
```

Zavolá rovnomennú metódu na všetkých stránkach uložených v projekte. Jednotlivé časti výsledkov spoji do jedného celku a ten vráti. Pomocou tejto metódy získame kód pre aktualizáciu, už získaných hodnôt v databáze SQL.

```
public HashMap<String, HashMap<String,String>> getVariables()
```

Vráti kompletný dátový set všetkých získaných hodnôt pre celý web. Dátová štruktúra sa skladá zo sady hash map. Prvá hash mapa obsahuje textový reťazec v podobe url adresy. Pomocou tohto kľúča je vrátená uložená hash mapa obsahujúca informácie k danej url adrese(stránke). Týmto spôsobom môžeme manipulovať s konkrétnym informačným obsahom a nie s celým webom naraz.

```
public void loadBackup()
```

Na všetkých spracovaných stránkach volá rovnomennú metódu, ktorá vracia zálohovaný obsah.

```
public void printVariables()
```

Do štandardného výstupu pre všetky stránky kompletný výpis získaných informácií. Metóda ma zmysel predovšetkým v oblasti ladenia programu.

```
public void procesJsContent(String name)
```

Metóda spracováva obsadený JavaScript kód. Je konkrétne zameraná na spracovanie webovej stránky, ktoré svoj obsah chránia za použitia javascriptu. Ako ostatné metódy pracujúce s javascriptom je ich použitie exponenciálne.

```
public void replaceContent(String name, String find, String
replace)
```

Táto metóda prepisuje obsah informácie získané zo stiahnutého dokumentu. Platnosť metódy sa vzťahuje na všetky stránky daného projektu. Prepisovanie nie je prevedené na všetkých

získaných informáciách, ale len na informácií, kde jej názov odpovedá hodnote parametru name.

```
public void setCharset(String charset)
```

Nastaví názov kódovania pre celý projekt.

```
public void setContent(String name, String value)
```

Nastaví obsah uložený v parametri value do objektu triedy HtmlAdress. Tento objekt pod url adresou definované parametrom name.

```
public void setContentUrl(String name)
```

Táto metóda pridá všetkým spracovaným stránkam novú položku do získaných informácií. Kľúč získavanej informácie je uložený v parametre name. Hodnota tejto informácie odpovedá url adrese danej stránky.

```
public void setMainUrlStack(UrlStack mainUrlStack)
```

Uloží zásobník adries spracovaných projektom. Metóda slúži k aktualizácií zásobnika na ktorom boli prevedené zmeny.

```
public void storeContent(String name)
```

Funkcie na všetkých stránkach projektu volá rovnomennou metódou. Metóda vloží do zoznamu informáciu, novú položku s kľúčom odpovedajúcim obsahu parametru name. Vkladaná hodnota predstavuje otrimovaný aktuálny obsah stránky.

```
public void storeContentSQL(String name)
```

Funkcie na všetkých stránkach projektu volá rovnomennou metódou. Metóda vloží do zoznamu informáciu, novú položku s kľúčom odpovedajúcim obsahu parametru name. Vkladaná hodnota predstavuje otrimovaný aktuálny obsah stránky upravený pre vloženie do databáze SQL.

```
public void trimContent(String name)
```

Metóda slúži k upraveniu získaného obsahu na všetkých stránkach. Tento textový obsah je otrimovaný. Operácia otrimovania sa robí iba na obsahu, kde jeho názov odpovedá hodnote parametru name.

UrlStack

Trieda UrlStack je potomkom triedy LinkedList<String>. Tento rodič je zásobníkom (kontajnerom) rady položiek bez vzájomnej logickej väzby. V našom prípade je doplnený o radu funkcií využívaných WebDataMinning API v oblasti správy zoznamu dostupných url adries.

Premenné:

Všetky potrebné premenné sú už definované v rodičovskej triede.

Konštruktor:

```
UrlStack()
```

Vráti inštanciu triedy, táto inštancia neobsahuje žiadne položky.

Metódy:

```
public void addUrl(String url)
```

Pridá url do zoznamu. Pridávaná url adresa je testovaná na duplicitný výskyt. Pokiaľ je v zozname originálny, je pridaná. V prípade, že pridávaná hodnota vytvára duplicitu, tak nie je pridávaná.

```
public boolean contains(String value)
```

Reťazec obsadený v parametre value (url adresa) je testovaná na výskyt v udržovaných hodnotách. V prípade nálezu zhodného reťazca (adresy) je vrátený primitívni typ boolean obsahujúci hodnotu treu. V prípade, že testovaná hodnota v zozname nie je, je vrátená hodnota false.

```
private UrlStack getCopy()
```

Metóda vytvára novú inštanciu triedy UrlStack a naplňuje ju svojim obsahom.

```
public void print()
```

Metóda na štandardný výstup vypíše zoznam obsadených url adries vo formáte adresa a zalomenie na nový riadok.

```
public UrlStack remove(String r)
```

Metóda vracia novú inštanciu triedy UrlStack. Táto inštancia neobsahuje url adresy, ktoré majú vo svojom obsahu aspoň jeden výskyt obsahu parametru r.

```
public UrlStack select(String r)
```

Metóda vracia novú inštanciu triedy UrlStack. Táto inštancia obsahuje url adresy, ktoré majú vo svojom obsahu aspoň jeden výskyt obsahu parametru r.

UrlTool

Táto trieda obsahuje len jednu staticku metódu. Slúži k úprave url adries v rámci projektu.

```
public static String getValidUrl(String baseUrl, String href)
```

Metóda na základe znalosti validnej a plnohodnotnej url adresy, definovanej parametrom baseUrl dokáže upraviť hodnotu parametru href na plnohodnotnú url adresu. Metóda prevádza niektoré operácie ako napríklad.:

- doplnenie protokolu,
- doplnenie domény,
- doplnenie adresárovej štruktúry z relatívnej na absolútnu,
- zmenu adresára na základe ../.

Touto metódou dokážeme zvalidovať všetky druhy adries používaných na webových prezentáciách.

5. Získavanie informácií v čase

Cieľom je návrh systému, ktorý zo zadaných informačných zdrojov (webové servery, databázy) získavať požadované informácie a udržovať ich vo svojej databázy pre ďalšie spracovanie.

Zmysel aplikácie je v prostredí Internetu získavať informácie nie len iba dáta, ale konkrétne údaje, ktoré sú zachytene v reálnom čase. Príkladom môže byť sledovanie

kurzového lístku. Konkrétne v tejto aplikácie sú informácie získavane zo stránky http://kurzy.divoch.net/. Slúži to ako demonštratívna ukážka funkčnosti. Získanie informácii sa dá prispôsobiť na akúkoľvek inú stránku. Aplikácia je napísaná v jazyku JAVA, ktorý je na to vhodný vďaka tomu, že je to objektovo orientovaný programovací jazyk.

Klasické systémy založené na full-texte nechápu dáta z prostredia ako informácie. Vidia ich iba ako množiny znakov s obmedzenými vzťahmi. Tento systém nezískava objemné množstvo dát ale informácie.

Získavanie dát je uskutočnené individuálne. Každý zdroj dát je v systéme realizovaný pomocou tzv. pluginov. Tieto drobné časti kódu hovoria, kde a v akej podobe sa dané informácie nachádzajú. Všetko je realizované tak, aby sme neboli v ničom obmedzovaní. V prípade potreby môžeme použiť akýkoľvek komunikačný protokol, v akomkoľvek prostredí (RPC, XML-RPC, HTTP, HTTTPS, POP3, Telnet, Jabber, SNMP etc.) Táto nezávislosť na platforme a komunikačných protokoloch nám umožňuje plne sa sústrediť na samostatné získavanie dát – informácií. Po procese získavania informácií dochádza k dôležitej časti a to finalizácií procesu a to persistenciou. Všetky tieto podporné operácie zaisťuje táto aplikácia. Tá ja akým si životným prostredím (runtime) pre pluginy.

5.1 Obecný popis tried programu

Spustiteľná trieda

Trieda Main sa nachádza v základnom balíku. Táto trieda obsahuje spustiteľnú metódu main. Vyvolaním tejto metódy spôsobíme spustenie celej aplikácie.

Potrebné knižnice

commons-codec-1.3.jar

commons-httpclient-3.1.jar

commons-logging-1.1.1.jar

jdom.jar

log4j-1.2.15.jar

mysql-connector-java-5.0.6-bin.jar

Konfigurácia aplikácie

Chod aplikácie sa riadi pomocou niekoľkých konfiguračných súborov. Tie nastavujú všetky dôležité vlastnosti, potrebné pre bezchybný beh systému.

Logovanie

Aplikácia využíva služieb dnes už klasického rozhrania pre správu logovania Log4j. To je pri spustení nastavené hodnotami v konfiguračnom súbore log4j.xml. Súbor log4j.xml je umiestnený v root aplikácie.

Konfigurácie pripojenia k databázy a konfigurácia aplikácie

V roote aplikácie sa nachádza ďalší konfiguračný súbor setting.xml. Tento súbor obsahuje všetky konfiguračné hodnoty potrebné pre beh aplikácie.

Typ aplikácie

Celý systém je napísaný ako konzolová aplikácia. Neobsahuje žiadne GUI prvky. Pre túto architektúru som sa rozhodol z dôvodu splnenia nasledujúcich téz.

- Systém je určený do neinteraktívneho prostredia
- Musí byť schopný samostatného fungovania.

5.1.1 Popis jednotlivých tried a metód

Nasledujúce triedy sú obsadené v hlavnom balíku aplikácie.

Main

Trieda obsahujúca metódu main, spúšťa celú aplikáciu.

Application

Táto trieda zaisťuje celú aplikáciu. Drží si odkazy na objekty ako je napr. Manager, Persistence alebo MySql.

Konštruktor Application volaný z triedy Main ma za úlohu zaistiť inicializáciu prvkov aplikácie a následne predanie riadenia chodu aplikácie triede Manager.

Najdôležitejší metóda **ini** obsahuje kód nutný pre inicializáciu všetkých modelov aplikácií.

```
public class Application {

    MySql mySql = null;

    Manager manager = null;

    Persistence persistence = null;

    public Application() {

                ini();

    }

    private void ini() {

        mySql = new MySql();

        manager = new Manager();

        manager.setMySql(mySql);

        persistence = new Persistence();

        persistence.setMySql(mySql);

        manager.setPersistence(persistence);

        manager.processPlugin();

    }

}
```

Manager

Je jadrom celej aplikácie. Riadi prevedenie jednotlivých operácií, ktoré získavajú informácie z ľubovoľného prostredia. Konštruktory triedy volá metódou **ini**, tá načíta z konfigurácie zoznam pluginov a ukladá si ju k ďalšiemu použitiu.

Metóda **processPlugin** sa stará o spustenie všetkých aktívnych pluginov. Je volaná z objektu application, ktorý túto triedu inicializoval.

Persistence

Tento modul aplikácie nemá žiadnu zvláštnu väzbu na ďalšie prostriedky aplikácie, okrem triedy umožňujúce spoluprácu s databázou. Jeho účelom je poskytovať pluginom jednotne a na implementácií nenáročne prostredie persistenciu dát.

```
      public class Persistence {

          private Mysql mySql = null;

          public Persistence() {

          }

          public MySql getMySql() {

              return mySql;

          }

          public void setMySql(MySql mySql) {

              this.mySql = mySql;

          }

          public void persist(int idInfo, String value) {

              try {

                  // Ulozi do databaze. iDInfo -> id typu informace.

                  String query = "Insert into Data (`idTask`, `valueR`,
`date`) values ('" + idInfo + "', '" + value + "', DATE(NOW()));";

                  System.out.println(query);

                  getMySql().getStatement().execute(query);

              } catch (SQLException ex) {

Logger.getLogger(Persistence.class.getName()).log(Level.SEVERE, null, ex);

              }

          }
```

Balík setting

Tento balík obsahuje dve triedy, ktoré umožňujú ľahký prístup ku konfiguračným hodnotám.

Constants

Trieda určená ku statickému importu. Obsahuje konštanty konfigurácie.

SettingXML

Trieda sprístupňujúca XML konfiguračný súbor. Názvy kľúčov sa skladajú ako cesta oddelená znakom bodky.

Balík tool, obsahuje triedy, kde ich povaha skôr odpovedá všeobecne použiteľným nástrojom nezávislým na danom zmysle aplikácie.

MySQL

Nástroj umožňujúci vykonávanie SQL dotazov oproti MySQL serveru.

Web

Sada statických metód využívajúcich api HttpClient. Základná metóda **Strin getPage(String url)** vracia reťazec obsahujúci vrátený obsah danej url adresy.

Balík plugin, tento balík obsahuje jednotlivé triedy implementujúci kód, ktorý umožní získavať informácie z vnútorného prostredia.

```
public class Web {

    public static String getPage(String url) {

        try {
            HttpClient client = new HttpClient();
            PostMethod method = new PostMethod(url);
            HttpMethodParams HMP = new HttpClientParams();

client.getParams().setCookiePolicy(CookiePolicy.BROWSER_COMPATIBILITY);

            //NameValuePair p2 = new NameValuePair("username", user);
```

```
            //method.setRequestBody(new NameValuePair[]{p1});


            method.setParams(HMP);
            // set per default
            client.getParams().setParameter(HttpMethodParams.RETRY_HANDLER,
new DefaultHttpMethodRetryHandler());


            client.executeMethod(method);
            byte[] responseBody = method.getResponseBody();
            method.releaseConnection();


            return new String(responseBody);
        } catch (IOException ex) {
            System.out.println(ex);
        }
        return null;
    }
}
```

PluginInterface

Tento interface obsahuje nutný predpis metód, ktoré implementuje každé rozhranie. So všetkými pluginmi v celom systéme sa pracuje z pohľadu metód definovaných v PluginInterface.

```
public interface PluginInterface {

    void setManager(Manager manager);

    void startProcess();

    void processData();

    void finishProcess();

    int getStatus();

}
```

Pop3Plugin, WebPlugin

Tieto triedy implementujú PluginInterface, niektoré jeho metódy priamo doplňujú kódom, iné nechávajú až samostatným pluginom. Každá z týchto tried je koncipovaná pre iné prostredie (web, pop3). Implementujú sa práve tie metódy, ktoré sú tento prostredím spoločné.

5.2 Príklad použitia

Pomocou programu PuTTY [18] (klient protokolov SSH, Telnet, rlogin a holého TCP) sme sa prihlásili pomocou vyplnených prístupových údajov a na serveri sme spustili samotnú aplikáciu. Výsledok je vidno nižšie.

```
rychard@server:~$ java –jar WebDataGetter.jar

log4j:WARN No appenders could be found for logger
(cz.g2w.webdatagetter.settings.SettingsXML)

log4j:WARN Please initialize the log4j system properly.

URL : jdbc:mysql://localhost/
Database connection established
Insert into Data (`idTask`, `valueR`, `date`) values ('1', '25.740',
     DATE(NOW()));
 Insert into Data (`idTask`, `valueR`, `date`) values ('2', '17.575',
     DATE(NOW()));
```

1) java -jar WebDataGetter.jar - Spusti aplikáciu

2) log4j:WARN No appenders could be found for logger (cz.g2w.webdatagetter.settings.SettingsXML).
   - Pokúsi sa inicializovať logovanie na základe log4j.xml

3) log4j:WARN Please initialize the log4j systém properly.
   - Informácia o logovaní

4) URL : jdbc:mysql://localhost/
   Url pre pripojenie k MySQL

5) Database connection established

- Informácia o úspešnom pripojení k databáze

-Tu je všetko inicializovane a aplikácia začína vykonávať svoju prácu.

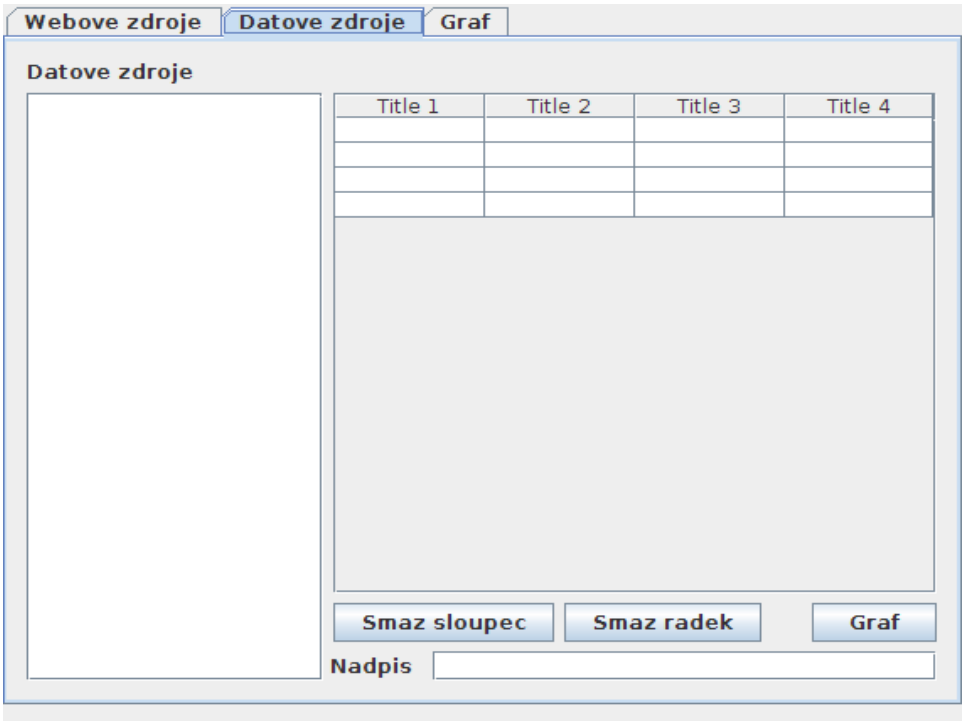
6) Insert into Data (`idTask`, `valueR`, `date`) values ('1', '25.740', DATE(NOW()));

7) Insert into Data (`idTask`, `valueR`, `date`) values ('2', '17.575', DATE(NOW()));

Kroky 6 a 7 sú výsledky činnosti pluginu. Každý plugin stiahne obsah url adresy. Vyparseruje hľadanú hodnotu a výsledok predá modulu pre perzistenciu dát. Ten vytvorí SQL insert a vloží ho do databázy plus vypíše na obrazovku (štandardný výstup)

6. Aplikácie pre priamy web data minning

Súčasťou mojej práce je aj skúšobná aplikácia pre web data mining. Zmyslom tejto aplikácie bolo vytvoriť skúšobné prostredie a overiť tak metódy prístupu zberu informácií, ich úpravy a prezentácia.



Obr. 4: AWT aplikácia priamy web data minning

Aplikácia je vytvorená v programovacom jazyku Java. Vyžíva štandardných knižníc pre prácu s grafickým užívateľským rozhraním (GUI). Celá ja založená na takzvaných lepších komponentoch, nevyužíva klasický balíček java.awt ale javax.swing.

Členenie aplikácie je prevedené do troch častí. Každá časť sa logický stará o komplexnú skupinu funkciu ore web data mining. Práca a ovládanie aplikácie je navrhnuté tak, aby ju mohol ovládať priemerný užívateľ. Pre jej používanie nemusíme byť programátor a nemusíme mať konkrétne znalosti z oblasti web miningu.

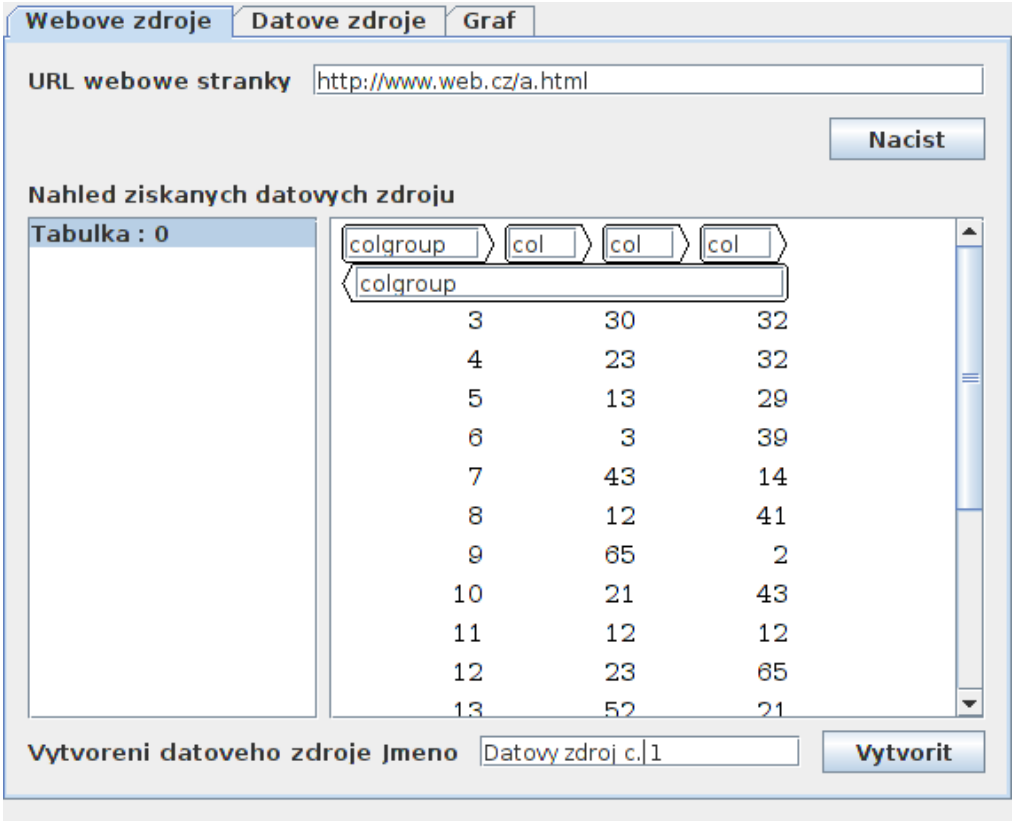
6.1 Prvá časť aplikácie – Dátový most

Prvá časť vytvára dátový most medzi našou aplikáciou a prostredím webových služieb. Koncepcia rozdeľujúca aplikáciu do viac častí bola zvolená predovšetkým z dôvodu jednoduchej zrozumiteľnosti a upraviteľnosti aplikácie. Každá časť má jasne definované vstupy a výstupy. Preto nie je problém upraviť aplikáciu bez ohrozenia funkčnosti iných celkov. Zmyslom tejto časti je prezentovať užívateľovi webovú stránku ako množinu použiteľných hodnôt, alebo dátový zdroj. Toto veľmi dôležité mapovanie prebieha v niekoľkých krokoch. Ich výsledkom je prevedenie prostej HTML stánky na skupinu informácií. K tomuto prevodu využívame zatiaľ pomerne jednoduchých metód. Vychádzame z nasledujúcich skúsenosti týkajúcich sa organizácie obsahu webových stránok. Najčastejšie využívaným prvkom pre organizáciu informácií na webovej stránke je tabuľka. HTML tabuľka je párový prvok jazyka HTML.

Pomocou ďalších prvok jazyka umožňuje organizovať dáta do dvojrozmerných štruktúr. My na webovej stránke tieto štruktúry hľadáme a interaktívne ich užívateľovi ponúkame. Tento proces prebieha vo viacerých krokoch. Najprv je nutné v HTML kóde tieto tabuľky nájsť (vyfiltrovať). Následne je využito bohatých schopností wsing komponent. Tieto komponenty dokážu renderovať fragmenty HTML kódu. Spojením týchto dvoch postupov dostáva užívateľ možnosť prezrieť si získané informácie a samotne sa rozhodnúť či majú pre neho význam.

Postup získania dátového zdroja

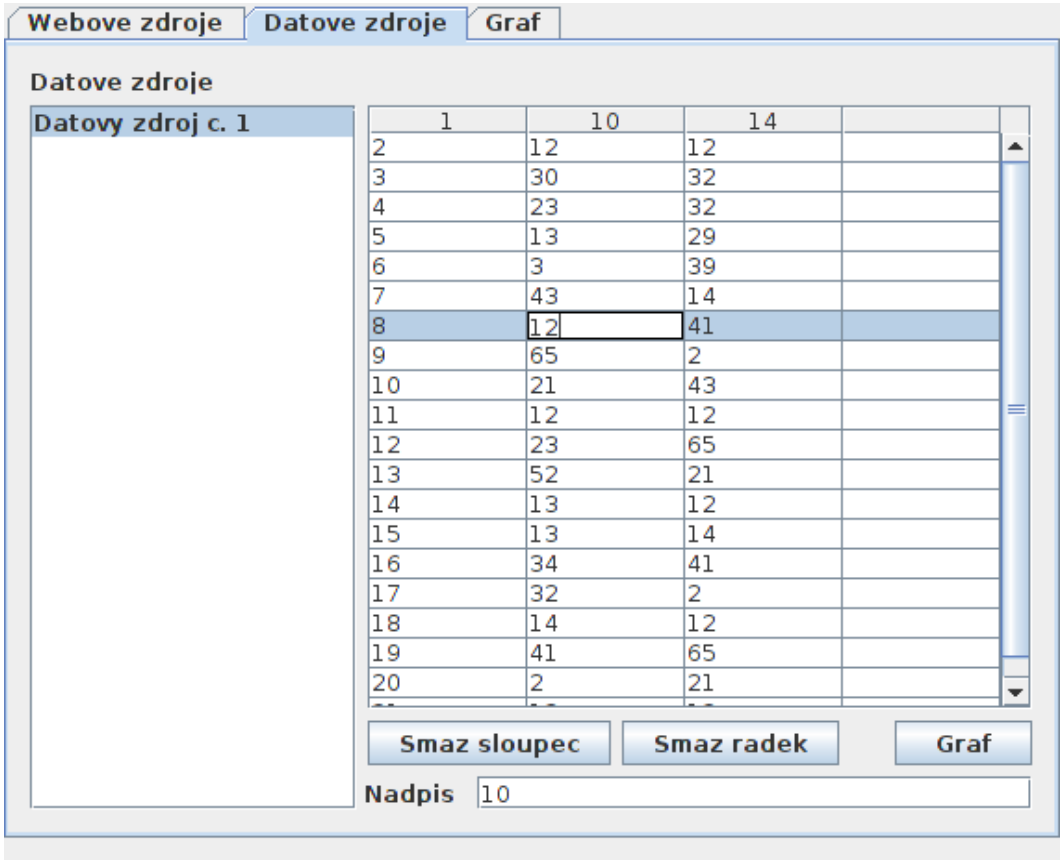
1. Vyplnenie URL adresy vedúcej k stránke s informáciami.
2. Stiahnutie požadovanej stránky.
3. Výber vhodnej tabuľky, ktorá obsahuje hľadané informácie.
4. Vytvorenie dátového zdroja.



Obr. 5: Aplikácia: Časť prvá

6.2 Druhá časť aplikácie – Dátový sklad

Druhá časť aplikácie predstavuje dátový sklad. Jednotlivé dátové zdroje odkazujúce na množiny informácií prezentované na najrôznejších stránkach sú tu užívateľovi predstavené za pomocou komponenty typu Jlist – zoznam. Z tohto zoznamu užívateľ vyberá medzi jednotlivými dátovými zdrojmi a následne ich upravuje pre samotnú prezentáciu v časti tri.



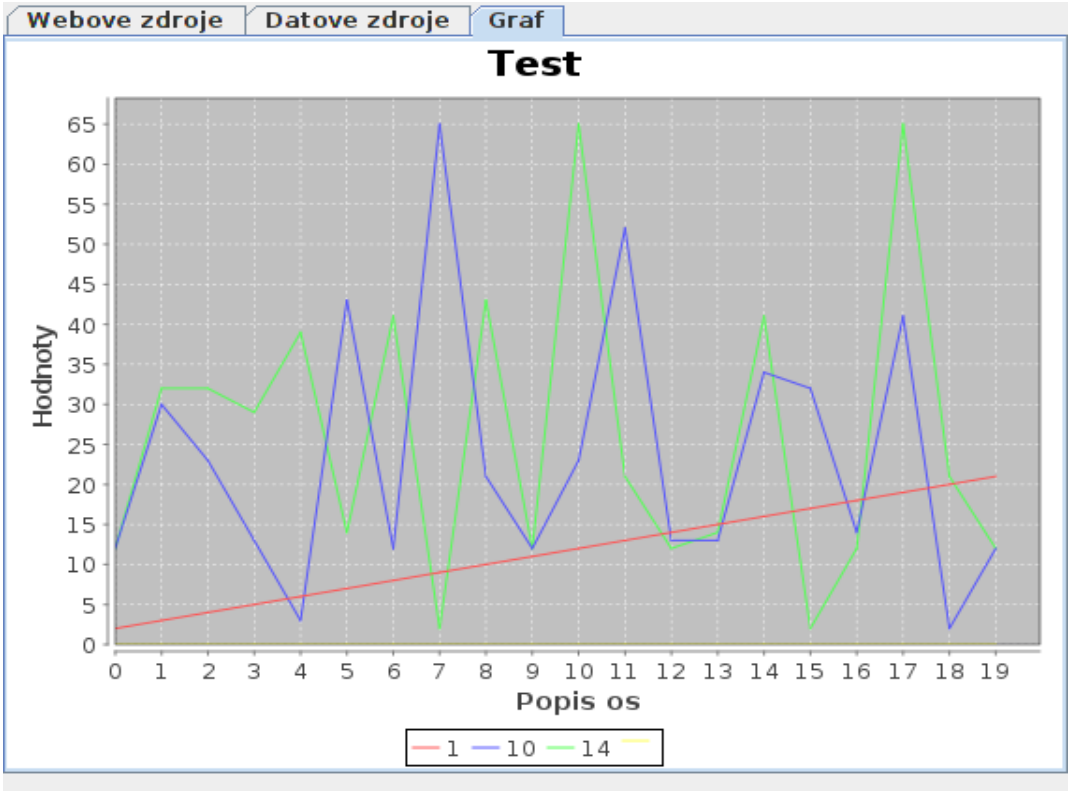
Obr. 6:Aplikácia: Časť druhá

Úprava dátového zdroja je veľmi dôležitá, predovšetkým z dôvodu zrozumiteľnosti prezentácie. Problémy s prezentáciou nastávajú predovšetkým z dôvodu zlúčenia jednotlivých informácií a informácií riadiacich ich prezentáciu. V tabuľkách sa naraz nachádzajú ako samotné číselné informácie tak aj informácie riadiace ich zobrazenie (tučné písmo, odkazy, obrázky, formátovanie ..). Všetky tieto dodatočné informácie musia byť samozrejme pred grafickým spracovaním odstránené. Aplikácia dokáže veľké množstvo týchto rušivých elementov odstrániť, samozrejme však nie je všemocná a mnohokrát musí zasiahnuť užívateľ.

Tu prichádza na radu XML, XSL, ktoré dokáže mnohé s týchto problémov riešiť. Je to bohužiaľ medzi tvorcami webových stránok neobľúbené pre svoju náročnosť. Základné operácie, ktoré nám nástroj poskytuje, sú spracovanie textu obsadeného v jednotlivých bunkách, mazanie celých stĺpcov tabuľky a mazanie celých riadkov tabuľky.

6.3 Tretia časť : grafické komponenty

Tretia časť zaisťuje prezentáciu pomocou grafickej komponenty. Táto komponenta grafický znázorňuje získané dáta ako dvojrozmerný spojnicový graf. Vďaka tejto komponente máme možnosť získať názorný prehľad o nami vytvorenom informačnom zdroji. Komponenta je interaktívna a umožňuje užívateľovi vytvárať výrezy a sledovať tak drobné zmeny grafu, ktoré by pri celkovom pohľade boli nepatrné a prehliadnuteľné.



Obr. 7: Aplikácia: Tretia časť, grafické komponenty

6.4 Zhodnotenie aplikácie

Aplikácia slúži k experimentálnym účelom. Jej zmyslom bolo overenie navrhnutých postupov. Práca a aplikáciami je jednoduchá a výsledky grafickej prezentácie dát sú presvedčivé. Potenciál je predovšetkým v dvoch smeroch.

Prvý smer vidíme vo vylepšovaní schopností aplikácie a to predovšetkým v oblasti spracovania a upravovania dát. Delenie aplikácie do viacerých častí umožňuje jednoduché implementovanie ďalších rozširujúcich funkcií a nenáročnú úpravu funkcií stavajúcich.

Druhy smer je pre nás veľmi zaujímavý, lebo naša práca vytvorila tri softwarové celky:

- AWT aplikáciu pre web data mining.
- API pre web data mining.
- Serverovú aplikáciu pre perzistenciu získaných informácií v čase.

Tieto tri celky ponúkajú odlišné uhly pohľadu na celú problematiku. Ich spojením ale môžeme získať nástroje, ktoré budú dostatočne variabilné a zároveň robustné pre skutočné operácie v oblasti získavania informačného obsahu a jeho spracovanie, perzistenciu a prezentáciu.

7. Záver

Cieľom práce bolo navrhnúť systém, ktorý bude zo zadaných informačných zdrojov, či už by to boli webové stránky alebo databázy získavať požadované verejne informácie a udržovať ich vo svojej databáze pre ďalšie využitie.

Práca sa na začiatku zaoberá problematikou web miningu, teoretickou časťou sú rozobrané metódy, problémy, zdroje dát ale taktiež techniky web content minigu, ktoré sme využívali. Praktická časť pozostávala z návrhu systému v programovacom jazyku Java. Systém je otvorený pre ďalší vývoj. Jeho pole pôsobnosti je veľmi široké a je prispôsobené, ako konkrétnemu použitiu, tak aj úplne voľnému prístupu v rámci metodológie web content minningu. Grafická AWT aplikácia je užívateľsky nenáročná a s príjemným prostredím, ktoré umožňuje získavať okamžité vizuálne prezentácie nad informačnými množinami. Programové API prostredie prevedie programátora začiatočníka technologicky nenáročnou implementáciou protokolu HTTP a časti HTML bez akýchkoľvek ich znalosti. Tretia časť programového výstupu je hotový systém umožňujúci zber hodnôt (informácií) v čase.

Najdôležitejšími prvkami našej práce je komponentové orientovaný návrh a otvorená štruktúra. Tento prístup z nášho softwarového diela vytvára produkt, ktorý môže byť ďalej jednoducho vyvíjaný a upravovaný k potrebe každého užívateľa.

Použitá literatúra

[1] Data Warehousing Review <http://www.dwreview.com/Data_mining/index.html>.

[2] Chakrabarty S., Mining the Web: Analysis of Hypertext and Semi Structured Data, Morgan Kauffman, 2002. P. 344, ISBN-10: 1558607544

[3] Čenovský, L., 2003. Web Usage Mining on is.muni.cz. <http://is.muni.cz/clanky/2002_web_usage_thesis.pdf>

[4] Dvořák, J., 2007. Web Usage Mining. Diplomová práce. Vysoká škola ekonomická v Praze, Fakulta managementu v Jindřichově Hradci.

[5] Berka, Petr. Dobývání znalostí z databází. Praha: Academia, 2003. 366 s. ISBN 80-200-1062-9.

[6] Galeas, P. 2008. What is Web Mining? <http://www.galeas.de/webmining.html> .

[7] Tanasa, D. 2005. Web Usage Mining: Contributions to Intersites Logs Preprocessing and Sequential Pattern Extraction with Low Support.

<http://www-sop.inria.fr/axis/personnel/Doru.Tanasa/these_TANASA.pdf>

[8] ADS Lab 2007, An Introduction to Web Mining, Advanced Data System Laboratory.

<http://dblab.csie.ncku.edu.tw/~tsengsm/COURSE/WebDB/Web-Mining.ppt>

[9] Borges, J, L, C 2000, A Data Mining Model to Capture UserWeb Navigation Patterns, Department of Computer Science University College London. <http://paginas.fe.up.pt/~jlborges/publications/BorgesPhDthesis.pdf.zip>.

[10] Han J. – Kamber M.: Data Mining – Concepts and Techniques, Second Edition, Morgan, Kaufmann 2006.

[11] Tang Z. – MacLennan J. – Data Mining with SQL Server 2005, Wiley Publishing, Inc.2005.

[12] Chris Bizer,Richard Cyganiak,Tom Heath: How to publish Linked Data on the Web <http://sites.wiwiss.fu-berlin.de/suhl/bizer/pub/LinkedDataTutorial/>.

[13] Liu B., Web Data Mining-Exploring Hyperlinks, Contents, and Usage Data, Springer Series on Data-Centric Systems and Applications, 2007. ISBN-10: 3540378812

[14] W3C: RDF/XML Syntax Specification,

<http://www.w3.org/TR/rdf-syntax-grammar/.

[15] Java DOM Tutorial < http://www.roseindia.net/xml/dom/>.

[16] Open Source HTML Parsers in Java

<http://java-source.net/open-source/html-parsers/>.

[17] Java: Html parser http://www.zaachi.com/cs/items/java-html-parser.html/.

[18] PuTTY< http://cs.wikipedia.org/wiki/PuTTY>.

[19] Bamshad Mobasher , Web Usage Mining

< http://maya.cs.depaul.edu/~mobasher/webminer/survey/node6.html>.

[20] OLAP (Online Analytical Processing) < http://cs.wikipedia.org/wiki/OLAP>.

[21] Raymond Kosala - Hendrik Blockeel, Web Mining Research: A Survey, 2000 <http://www.sigkdd.org/explorations/issue2-1/kosala.pdf >.

[22] Web trawler <http://en.wikipedia.org/wiki/Web_crawler> .

[23] Cooley, R., Mobasher, B., Srivastava, J: Web Mining: Information and Pattern Discovery on the World Wide Web.
<http://maya.cs.depaul.edu/~mobasher/papers/webminer-tai97.pdf >.

[24] Paolo Giudici, Silvia Figini - Applied Data Mining for Business and Industry,

ISBN**:** 9780470058862.

[25] Java 6 – Výukový kurzy, Sharon Zakhour, Scott Hommel, Jacob Royal, Isaac Rabinovitch, Tom Risser, Mark Hoeber, 536s, ISBN: 978-80-251-1575-6

[26] Pavel Herout , Java a XML 2007, ISBN: 9788072323074

Zoznam skratiek

API - Application programming interface

WWW - Word Wide Web

OLAP - Online Analytical Processing

SEO - Search Engine Optimalization

ECLF – Extended Common Log Format

BFS - Breadth First Search

SQL - Structured Query Language

HTML - HyperText Markup Language

HDP - Hrubý domáci produkt

AWT - Abstract Window Toolkit

HTTP- Hypertext Transfer Protocol