Západočeská univerzita v Plzni

Fakulta Aplikovaných věd

Katedra informatiky a výpočetní techniky

Webový sumarizátor

Semestrální Projekt 5

Leden 2008 Martin Sloup, A04372, msloup@students.zcu.cz

Obsah

1. Zadání semestrální práce....3
2. Analýza problému....3
2.1. Různé postřehy....3
2.2. Popis webové sumarizace....3
2.3. Tvorba klíčových slov....4
2.4. Hledání pomocí webového vyhledávače....4
3. Programátorská příručka....5
3.1. Popis tříd....5
3.1.1. Balíček cz.zcu.sumarizer....5
3.1.2. Balíček cz.zcu.sumarizer.exception....5
3.1.3. Balíček cz.zcu.sumarizer.jobs....5
3.1.4. Balíček cz.zcu.sumarizer.pageparser....5
3.1.5. Balíček cz.zcu.sumarizer.tests....6
3.1.6. Balíček cz.zcu.sumarizer.utils....6
3.1.7. Balíček cz.zcu.sumarizer.web....6
3.1.8. Balíček cz.zcu.sumarizer.websearches....6
4. Uživatelská příručka....6
4.1. Popis služby....6
4.2. Kompilace a instalace....6
4.3. Konfigurace....7
4.4. Použití....9
5. Problémy a řešení....9
6. Závěr....10
6.1. Další vývoj....10
7. Literatura....10

1. Zadání semestrální práce

Za pomocí Musutelsa sumarizátoru sestrojte webovou službu, která po zadání otázky najde relevantní stránky a použije je jako zdroj dat v sumarizátoru. Výsledek bude předložen uživateli. Uživatel by měl mít možnost určit, kolik slov mu bude ve výsledku předloženo.

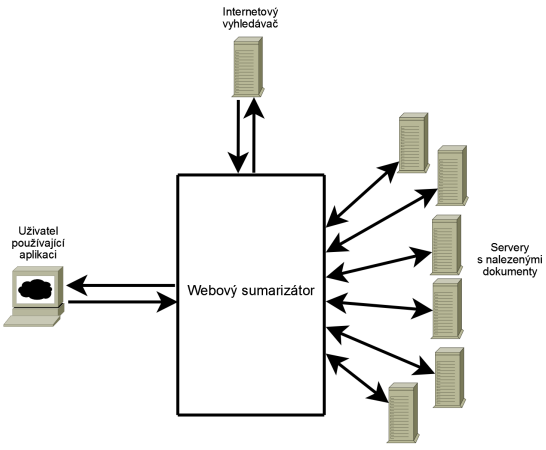
2. Analýza problému

2.1. Různé postřehy

- Při prvním zběžném ohledání **Musutelsa sumarizátoru** (viz [1]) jsem zjistil, že je napsán v programovacím jazyce **Java**. Proto jsem si řekl, že by bylo výhodné celou webovou službu naprogramovat taktéž v Javě. Výkonostně a ani z bezpečnostních důvodů by určitě nebylo dobré, aby se celý proces sumarizace spouštěl z příkazové řádky z důvodu použití jiného programovacího jazyka jako např. **PHP**, **Ruby on Rails**, či **Pythoon**.
- Řekl jsem si, že se na sumarizátor budu dívat jako na černou skříňku. Vnitřek sumarizátoru budu upravovat jen v případě, že to bude nutné pro správný chod celého sumarizačního procesu. To bylo nakonec nutné provést v několika případech.
- Původně jsem měl v úmyslu použít k vyhledávání dokumentů, které vstupují do sumarizačního procesu výsledky získané přes **Google Search API**. Ale bohužel, **Google** zrušilo vytváření nového klíče k fungování této služby v průběhu prosince 2006. Nakonec jsem si řekl, že si napíšu vlastní rozhraní, z kterého bude možné vytvářet jednoduchým způsobem vyhledávací moduly pro různé webové vyhledávače.

2.2. Popis webové sumarizace

Webový sumarizátor po obdržení otázky od uživatele provede transformaci otázky na klíčová slova. K zjištění klíčových slov jsem použil funkce **Musutelsa sumarizátoru**, které přímo k tomuto vybízeli. To se provádí tak, že načteme seznam slov. Pak postupně projdeme všechna slova, co uživatel zadal do otázky a porovnáme je se seznamem stopslov. Pokud slovo není v seznamu stop slov, tak ho přidáme mezi klíčová slova. Když už jsou k dispozici klíčová slova, odešle se požadavek na webový vyhledávač a jako parametr hledání se použijí klíčová slova



Obrázek 1: Schéma komunikace webového sumarizátoru

získaná z předchoziho kroku.

Z výsledku, který nám vyhledávač zašle zpět, se vytáhnou odkazy na dokumenty. Obsah dokumentů na těchto odkazech se postupně stáhne a prožene parserem, který nám vrátí obsahy dokumentů. Tento parser se řídí definičním souborem, který obsahuje **XQuery** dotazy pro vytažení hlavního textu a titulku pro každý zdrojový server zvlášť. To umožňuje přesné vytažení důležitého obsahu z každého nadefinovaného serveru. Avšak pokud není server nadefinovaný, použijí se výchozí **XQuery** dotazy, které vrátí jako obsah textu celý dokument.

Získaný obsah dokumentu se rozdělí na věty a věty ze všech dokumentů vstupují do Musutelsa sumarizátoru. Ten nalezne k zadané otázce relevantní věty a vratí je jako výsledek. Nakonec se provede složení vět do jednoho souvislého textu a ten se vrátí uživateli.

Délka počtu slov výsledného textu, která se vrací uživateli, záleží na počátečním nastavení. V konečné podobě se ale může lišit o pár slov, jelikož sumarizátor vrací další relevantní větu, pokud nedosahne celkový počet slov určenému limitu.

2.3. Tvorba klíčových slov

Jelikož bylo nutné ze zadané otázky získat klíčová slova, musel jsem nastudovat z **Musutelsa sumarizátoru**, jak probíhá proces získávání klíčových slov.

K tvorbě klíčových slov jsou potřebná tzv. stop slova. To jsou slova, která nenesou žádný význam. Mezi tyto slova patří třeba zájmena, předložky a spojky (je jich daleko víc, ale pro představu předkládám alespoň tři nejpočetnější skupiny). Pokud se tedy najde v otázce slovo, které se nachází mezi stop slovy, vyřadí se. Dostaneme tak několik slov, které nosí nějakou informaci.

Dále by se ještě mělo provést převedení těchto slov do základního tvaru, tzv. proces lemmatizace. K tomu je potřeba slovník, kde je uvedena dvojce původni slovo a slovo v základním tvaru. Problematika je bohužel trochu složitější. V českém jazyce se vyskytují slova, z kterých lze z původního slova získat více různých slov v základním tvaru. Pro příklad uvedu slovo tancích (německých tancích). Toto slovo může být buď slovo tank nebo tanec. Rozhodl jsem se tedy tento problém přenechat webovému vyhledávači, protože nejdůležitější je získat co nejlepší vstupní data.

2.4. Hledání pomocí webového vyhledávače

Ve webovém sumarizátoru jsem implementoval dva vyhledávací moduly.

První z nich je vyhledávání prostřednictvím **Google**. Ten vrací slušné výsledky při hledání. Ja jsem ale potřeboval hledat pouze na určité doméně. Tedy použil jsem známý modifikátor při hledání „site:domena.cz“. Bohužel problém tohoto modifikátoru je ten, že neumožňuje určit více domén najednou. Jediné řešení je tedy v budoucnu použít vícenásobné hledání a sloučit výsledky hledání.

Protože jsem potřeboval lepší výsledky hledání a vyhledával jsem dokumenty jen pro doménu Novinky.cz, použil jsem vyhledávání přímo na stránkách **Novinky.cz**. Tento vyhledávač je postavený na vyhledávání od **Seznamu**, takže je částečně zaručena dobrá kvalita vyhledávání.

U obou těchto vyhledávacích modulů pak ze stránky výsledku hledání získám odkazy na nalezené dokumenty. Získávání těchto odkazů provádím prostřednictvím regulárních výrazů. Použil jsem je z toho důvodu, že se pro ně píše jednoduše předpis a jsou docela rychlé, oproti

použití třeba XPath dotazů.

3. Programátorská příručka

Webový službu jsem tvořil iterativní metodou. Tím jsem zajistil, že můžu postupně svému zadavateli ukazovat funkční kód a získávat další připomínky k vývoji. Momentálně se webový sumarizátor nachází ve třetí iteraci.

V první iteraci jsem provedl rozbor zadání. Navrhl jsem postup implementace. Jako cíl jsem zvolil konzolovou aplikaci, které se zadá otázka a aplikace odpoví naleznýma dokumentama.

V druhé iteraci jsem doimplementoval použití procesu sumarizace. Odpověď aplikace již nyní byl výsledný text sumarizačního procesu.

Ve třetí iteraci jsem napsal webovou část celé služby. Vytvořil navrh designu pro stránky dotazu a výsledku, následně napsal k nim html kód a kaskádové styly. Odladil poslední chyby.

3.1. Popis tříd

3.1.1. Balíček cz.zcu.sumarizer

- **ConsoleSumarizer** - Hlavní třída konzolové aplikace
- **Setting** – Nastvení pro konzolovou aplikace
- **SumarizerData** – Třída pracující s xml souborem sumarizátoru, do kterého se ukládají data k procesu sumarizace
- **SumarizerSource** – Třída pro samostaný dokument

3.1.2. Balíček cz.zcu.sumarizer.exception

- **EmptyException** – Třída pro vyjjímku pro prázdný text

3.1.3. Balíček cz.zcu.sumarizer.jobs

- **Job** – Interface pro úlohy
- **Jobs** – Třída spouštějící postupně všechny nadefinované úlohy
- **MusutelsaSumarizerJob** – Třída úlohy spouštějící Musutelsa sumarizátor
- **TextAbstracterJob** – Třída úlohy vytvářející výsledný celkový text z ukazatelů na věty dokumentů

3.1.4. Balíček cz.zcu.sumarizer.pageparser

- **PageDefinition** – Interface pro práci s hodnotami v definičním souboru
- **PageParser** – Třída parsuje vstupní HTML kód a vrací nadefinované části
- **PageParserDefinition** – Načítá a vrací definice k parsování HTML kódu pro zadaný server

3.1.5. Balíček cz.zcu.sumarizer.tests

Třídy pro různé testování funkcí služby

3.1.6. Balíček cz.zcu.sumarizer.utils

- **ArrayUtils** – Různé funkce pro práci s polem
- **MusutelsaSumarizerUtils** – Fukce pro práci s Musutelsa sumarizátorem
- **PageDownloader** – Třída na stahování obsahu dokumentů
- **StringUtils** – Funkce pro práci se Stringem

3.1.7. Balíček cz.zcu.sumarizer.web

- **WebSumarizer** – Hlavní třída webové služby, která sbíra požadavky od uživatele provádí celý proces webové sumarizace a vrací výsledné HTML stránky

3.1.8. Balíček cz.zcu.sumarizer.websearches

- **GoogleSearch** – Třída implementující hledání pro Google.cz
- **NovinkySearch** – Třída implementující hledání pro Novinky.cz
- **Search** – Základní třída provádějící hledání prostřednictvím webových vyhledávačů. Pro specifický vyhledávač je nutné do implementovat funkci doSearch()
- **SearchResult** – Drží pohromadě výsledek hledání

4. Uživatelská příručka

4.1. Popis služby

Aplikace slouží k získání odpovědi na otázku, kterou zadá uživatel. Forma odpovědi představuje shluk vět, jenž vrátí sumarizér ze zdrojů, které jsou relevantní k této otázce. Zdroj odpovědi se získá z výsledku hledání na webovém vyhledávači. K tomu použijeme klíčová slova, co se vytvořila ze zadané otázky uživatele. Jako uživatelské rozhraní poslouží webová stránka, kde uživatel přehledně vypní všechny požadované údaje. Na další stránce se mu v přehledné formě vrátí odpověď včetně odkazů na použité zdroje, jenž by chtěl dále uživatel využít.

4.2. Kompilace a instalace

Celý webový sumarizátor lze lehce zkompilovat pomocí **ANTového** skriptu. Nejprve je nutné otevřít soubor „build.properties“ a nastavit do „catalina.home“ cestu ke kořeni **Apache Tomcat 5.5**[1]. Následně stačí spustit **ANT** s parametrem „dist“[2]:

```
> ant dist
Buildfile: build.xml
```

1 Webový sumarizátor funguje i na Apache Tomcat 6.0, ale kompilační skript funguje pouze s Tomcat verze 5.5. Tomcat 6.0 má truchu jinak řešenou adresářovou strukturu.

2 Další nejpoužívanější parametry jsou clean, javadoc a compile. První z nich vyčistí veškeré odkládací adresáře, které vznikly při kompilaci a tvorbě distribučního balíčku. Druhý parametr vygeneruje javadoc dokumentaci a poslední zkompiluje zdrojové soubory.

```
prepare:
    [mkdir] Created dir: build
    [mkdir] Created dir: build/WEB-INF
    [mkdir] Created dir: build/WEB-INF/classes
     [copy] Copying 43 files to build

compile:

javadoc:
    [mkdir] Created dir: dist/docs/api
  [javadoc] Generating Javadoc
        ...

dist:
      [jar] Building jar: dist/sumarizer.war

BUILD SUCCESSFUL
```

V adresáři „dist“ se nyní nachází soubor „sumarizer.war“. Ten stačí zkopírovat do adresáře „webapp“ v instalaci **Tomcatu**. Dále je nutné nastavit Tomcatu maximální velikost heap memory pomocí parametru -Xmx500m na minimální hodnotu alespoň 500MB z důvodu načtení slovníků do paměti. Nakonec je nutné nastavit práva pro zápis do adresáře „WEB-INF/output“ pro uživatele, pod kterým běží **Tomcat**.

4.3. Konfigurace

Konfigurace celého webového sumarizátoru se provadí pomocí dvou souborů. První z nich „setting.xml“ nastavuje samotný sumarizátor. Ten se nachází v adresáři WEB-INF/data. Ukáži zde použité výchozí nastavení s jednoduchými popiskami. Podrobnější informace viz [1].

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE settings SYSTEM "settings.dtd">


<settings>
  <parameters>
    <!-- jaky generator se ma pouzit, mozno musutelsa | random -->
    <parameter name="extractor">musutelsa</parameter>

    <!-- id clusteru, ktery chci zpracovat, mozno pouzit all pro zpracovani vsech clusteru -->
    <parameter name="cid">all</parameter>
    <!-- id dotazu, ktery se ma pouzit -->
    <parameter name="qid">1</parameter>
    <!-- vaha slov, jez se vyskytuji v dotazu -->
    <parameter name="weight">2</parameter>
    <!-- vaha slov, ktera se objevuji ve vete vicekrat, 0 -> vicecetnost se neuvazuje-->
    <parameter name="multiple">0</parameter>
    <!-- metoda orezu slov, 0 -> nepouzije se, 1 a vetsi udava delku slova, ktere jeste bude
oznaceno za stop slovo -->
    <parameter name="cutting">2</parameter>

    <!-- prednastavena hodnota prahu lsi -->
    <parameter name="lsi">5</parameter>
    <!-- prednastavena hodnota prahu cos -->
    <parameter name="cos">60</parameter>
    <!-- prednastavena hodnota prahu sen -->
    <parameter name="sen">0.5</parameter>
    <!-- prednastavena hodnota mocniny matice sigma po SVD rozkladu, bezne 2 -->
    <parameter name="sigmapow">2</parameter>
    <!-- cislo, jim se deli uhel pro kosinovou podobnost, pokud neni ektrakt naplnen a jsou k
dispozici dalsi vety, kttere vsak nesponuji kosinovou podobnost musi byt mensi nez 1 -->
    <parameter name="cosreduce">0.8</parameter>

    <!-- jednotka extraktu, zatim umim jen w jako word -->
    <parameter name="lunit">w</parameter>
    <!-- jendotka extraktu, zatim umim jen s jako sentence -->
    <parameter name="extunit">s</parameter>
    <!-- delka extraktu v danych jednotkach -->
    <parameter name="length">250</parameter>

    <!-- jmeno vystupniho souboru, kam se ulozi zmenene xml -->
```

```
    <parameter name="outputfile">sumarizer_out.xml</parameter>
  </parameters>

  <dictionaries>
    <!-- jmeno souboru se slovnikem pro lemmatizaci-->
    <dictionary type="lemma">voc.ldict</dictionary>
    <!-- jmeno souboru se slovnikem stop slov -->
    <dictionary type="stop">stoplist.cz</dictionary>
  </dictionaries>
</settings>
```

Duhým souborem „web.xml“ se nastaví webový sumarizátor a zároveň slouží jako nastavení **Apache Tomcat**. Tento soubor lze nalézt v adresáři WEB-INF.

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app version="2.4"
    xmlns="http://java.sun.com/xml/ns/j2ee"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://java.sun.com/xml/ns/j2ee
    http://java.sun.com/xml/ns/j2ee/web-app_2_4.xsd">
  <servlet>
    <servlet-name>index</servlet-name>
    <servlet-class>cz.zcu.sumarizer.web.WebSumarizer</servlet-class>
    <load-on-startup>1</load-on-startup>
  </servlet>
  <servlet-mapping>
    <servlet-name>index</servlet-name>
    <url-pattern>/sumarize</url-pattern>
  </servlet-mapping>

  <welcome-file-list>
    <welcome-file>index.jsp</welcome-file>

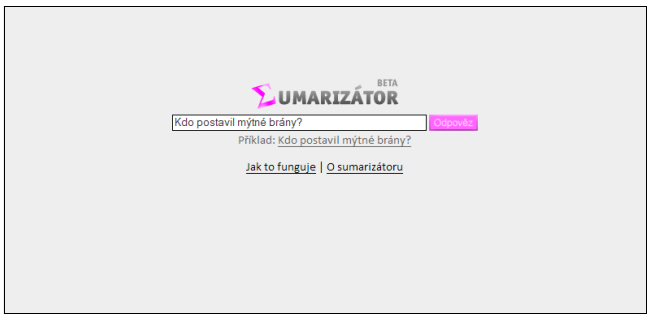
  </welcome-file-list>

  <!-- Soubor se slovnikem pro lemmatizaci -->
  <context-param>
    <param-name>lemmafile</param-name>
    <param-value>WEB-INF/data/voc-min.ldict</param-value>
  </context-param>
  <!-- Soubor se stopslovy -->
  <context-param>
    <param-name>stopwordsfile</param-name>
    <param-value>WEB-INF/data/stoplist.cz</param-value>
  </context-param>
  <!-- Soubor s nastavenim -->
  <context-param>
    <param-name>settingsfile</param-name>
    <param-value>WEB-INF/data/settings.xml</param-value>
  </context-param>
  <!-- Soubor definici pro parsovani dokumentu -->
  <context-param>
    <param-name>pageparserfile</param-name>
    <param-value>WEB-INF/data/pageparser.def</param-value>
  </context-param>
</web-app>
```

U tohoto souboru nás především zajímají nastavení uvedené v tagu „context-param“. Jedná se tedy o čtyři hodnoty: lemmafile, stopwordsfile, settingsfile, pageparserfile. První z nich, lemmafile, nastavuje slovník použitý pro lemmatizaci v sumarizátoru **Musutelsa**. Druhý, stopwordsfile, slouží k určení slovníku se stopslovy, které slouží k získání slov a používají se i při samotném procesu sumarizace. Nastavení slovníků se tedy provádí zde. Veškerá nastavení ohledně slovníků provedená v souboru „settings.xml“ se ignorují. Třetí položka nastavení je pro určení právě souboru „settings.xml“, kde se nachází výchozí nastavení sumarizátoru. Poslední, tedy čtvrtá položka, slouží k nastavení souboru s definicí **XQuery** dotazů pro parsování dokumentů, z kterých se získává obsah pro sumarizátor. Další hodnoty nastavení naleznete v dokumentaci k Java Servlet Specification, viz [2].

4.4. Použití

K ovládání celé webové aplikace slouží vstupní formulář, viz *obrázek 2*.



Obrázek 2: Dotazovací formulář webové aplikace

Zadáme tedy otázku, například: „*Kdo postavil mýtné brány?*“ Následně odešleme dotazovací formulář pomocí tlačítka „*Odpověz*“. Nyní probíhá samotný proces hledání zdroje a nasledné sumarizace. Po dokončení tohoto procesu nám stránka vrátí výsledek hledání (*Obrázek 3*):



Obrázek 3: Výsledek sumarizace

5. Problémy a řešení

Během vývoje služby jsem narazil na pár problémů, které jsem musel řešit.

V úvodu jsem již zmínil, že jsem původně chtěl použít **Google Search API**, které bylo nahrazeno svou Javascriptovou variantou. Celou věc jsem vyřešil tedy vlastní implementací, jenž mi umožnila napsat celé vyhledávání tak, aby bylo možné později libovolně upravovat server, ze kterého se bude provádět vyhledávání v dokumentech.

U **Musutelsa sumarizátoru** jsem narazil na limity. Předně bylo nutné vyřešit problém s rozdělováním vět na slova. Sumarizátor neakceptoval českou diakritiku, proto jsem upravil regulární výraz starající se o toto rozdělování. Taktéž bylo nutné zviditelnit některé metody samotného sumarizátoru, abych mohl využít samotné **API** sumarizátoru.

Musutelsa sumarizátor bohužel neřeší problém, kdy při procesu lemmatizace slova obsahuje dvě či více lemmat pro danné slovo. Například pro slovo „*brány*“ jsou lemmatizovaná slova „*brát*“ a „*brána*“. Bylo by teda nutné hodně zasáhnout do kódu **Musutelsa sumarizátoru**. To jsem ale nechtěl, a tak použiji jen první lemmatizované slovo. Bohužel při tom utrpí kvalita

výsledku.

6. Závěr

Webový sumarizátor by měl potenciálnímu uživateli přinést jednoduchý nástroj k získání odpovědi na zadanou otázku. Tato aplikace předvádí, jak by to mohlo fungovat. K perfektnímu výsledku avšak vede dlouhá cesta.

Pomocí tohoto projektu jsem se seznámil se základy sumarizace textů a naučil jsem se vytvářet servletové aplikace postavené v javě na projektu **Apache Tomcat**. Osvojil jsem si tvorbu klíčových slov. Při parsování stránek jsem použil **XQuery** a **XPath**, jenž se bylo potřeba taktéž naučit.

6.1. Další vývoj

Ve vývoji této webové aplikace bych rád v budoucnu rád pokračoval v rámci bakalářské práce, později i třeba různých projektů a diplomové práce.

Momentálně webová aplikace umí používat pouze jeden vyhledávač a hledat dokumenty na jednom serveru. To bych chtěl v dalším vývoji změnit. Vyhledávač by měl jít vybrat při hledání. Pro zjednodušenou webovou sumarizaci bude vybrán na pevno vyhledávač, jako je to doteď, ale při rozšířeném módu by mělo jít ho vybrat. Taktéž rozšířený mód by měl poskytnout větší možnosti nastavení webového sumarizátoru.

Taktéž i stahování dokumentů, které aktuálně stahuje dokumenty postupně bych rád nahradil stahováním ve více vláknech, což by mohlo zvýšit rychlost aplikace. Samotná třída strající se o stahování má občas problémy s kódováním češtiny a to v případě, když kódování dokumentu není uvedeno **HTTP** hlavičce, ale jen v samotném dokumentu. I toto by mělo být odstraněno.

Stahovaný dokument (případně článek, pokud provádíme sumarizaci na článcích s internetových deníků) často obsahuje **relevantní články**, které by mohli v určitých případech říct více informací k zadané otázce. Webový sumarizátor v této verzi na relevantní články nehledí, což bych rád v budoucnu změnil. **Relevantní članky** by se tedy přidaly do seznamu dokumetů, na kterých se provede sumarizace.

Po sumarizaci se provádí sloučení vět do jednoho celku. Dle *obrázku 3* je vidět, že to není až tak příliš dokonalé. Často věty mezi sebou postrádají význam a nejsou dle času správně chronologicky seřazeny. A i přehlednost výsledku je špatná, i když jsem použil větší řádkování z důvodu čitelnosti. Bylo by tedy vhodné napsat modul, který by provedl vhodnou úpravu pořadí vět, případně rozdělení do odstavců.

V dalším vývoji by měl být nasazen nový sumarizátor, jenž vytváří můj zadavatel projektu, *Ing. Josef Steinberger*. Webová služba by pak obsahovala sumarizátory dva a bylo by jen na volbě uživatele, který by se použil.

7. Literatura

[1] Martin Křišťan. *Vícedokumentový sumarizátor textů založen na latentní sémantické analýze*, diplomová práce, ZČU Plzeň, 2007.

[2] Danny Coward, Yutaka Yoshida. *Java Servlet Specification – Version 2.4*, Sun Microsystems, Inc., url: http://java.sun.com/products/servlet/download.html

[3] Herout Pavel. *Učebnice jazyka Java*, Kopp 2000.