**THESIS**

**Diplomed:**

**Mariyan Stanchev Marinov**

**Matriculation No: 063137**

**Rousse**

**2010 year**

**Table of Contents**

[***1.*** ***Abstract*** 1](#_Toc267989148)

[***2.*** ***Introduction*** 2](#_Toc267989149)

[***3.*** ***Evaluation of existing solutions.*** 4](#_Toc267989150)

[**3.1.** **Overview of existing solutions.** 4](#_Toc267989154)

[ Google 4](#_Toc267989155)

[ Yahoo! 6](#_Toc267989156)

[ MSN 6](#_Toc267989157)

[ Others 6](#_Toc267989158)

[**3.2.** **Conclusion** 7](#_Toc267989159)

[**3.3.** **Purpose and objectives.** 7](#_Toc267989160)

[***4.*** ***Design and description of the proposed solution.*** 9](#_Toc267989161)

[**4.1.** **Requirements for the software system.** 9](#_Toc267989163)

[**4.2.** **Logical model of the software** 9](#_Toc267989164)

[ Logical model of the Web Crawler 9](#_Toc267989165)

[ Logical model of the ‘Margent’ search agent 13](#_Toc267989166)

[**4.3.** **System architecture** 16](#_Toc267989167)

[**4.4.** **Selection of programming language and development environment.** 17](#_Toc267989168)

[**4.4.1.** **.NET Framework 3.5** 17](#_Toc267989169)

[**4.4.2.** **CSLA.NET** 18](#_Toc267989170)

[**4.4.3.** **MS Visual Studio 2008** 19](#_Toc267989171)

[**4.4.4.** **MS SQL Server 2008 и Microsoft SQL Server Management Studio 2008** 19](#_Toc267989172)

[**4.4.5.** **C#** 20](#_Toc267989173)

[**4.4.6.** **TortoiseSVN** 22](#_Toc267989174)

[**4.4.7.** **Easy SMTP Server** 22](#_Toc267989175)

[***5.*** ***Implementation of the software system:*** 23](#_Toc267989176)

[**5.1.** **Data structure** 23](#_Toc267989177)

[**5.2.** **Description of program modules** 25](#_Toc267989178)

[**5.2.1.** **Structure and organization of the GUI** 34](#_Toc267989179)

[**5.3.** **Instructions for using the software system:** 41](#_Toc267989180)

[**5.3.1.** **User Guide** 41](#_Toc267989181)

[**5.3.2.** **Instructions and requirements for installing the system.** 42](#_Toc267989182)

[***6.*** ***Tests and results*** 44](#_Toc267989183)

[***7.*** ***Conclusion and recommendation*** 45](#_Toc267989184)

[ Add more ‘stop’ words, because a lot of of ‘parasite’ words are indexed now; 45](#_Toc267989185)

[ Split tables – Split table Words to different tables for words with numbers, words with non-Latin letters; Split Files table to different tables for every indexed file type; 45](#_Toc267989186)

[ Update tables – do not truncate tables on every start of the crawler, but only update the data. For now only Words table is updated; 45](#_Toc267989187)

[ Develop SqlBulkCopy for copy from one DB to another 45](#_Toc267989188)

[ Improve indexing of .pdf and .mp3 files; also add indexing of .doc, .pps, .xls, other file types 45](#_Toc267989189)

[ Create better page ranking – with number of links on the page and some other criteria; 45](#_Toc267989190)

[ Add options to choose how many results to be fetched from the DB, and how many to be displayed per page; 45](#_Toc267989191)

[***8.*** ***References*** 46](#_Toc267989192)

[***9.*** ***Appendix*** 47](#_Toc267989193)

[ Source code of the Web Crawler 47](#_Toc267989194)

[ Source code of ‘Margent’ search agent 105](#_Toc267989195)

**Table of Figures**

[Figure 1 Timeline of Google's history 5](#_Toc267989196)

[Figure 2 Google search statistics [7] 6](#_Toc267989197)

[Figure 3 Use-case diagram of the SE 10](#_Toc267989198)

[Figure 4 Activity diagram of Start crawling - the essential process 11](#_Toc267989199)

[Figure 5 Overall workflow processes in the Web Crawler 12](#_Toc267989200)

[Figure 6 Use-case diagram of 'Margent' web page 14](#_Toc267989201)

[Figure 7 Activity diagram of 'Margent' search agent workflow 15](#_Toc267989202)

[Figure 8 Sequence diagram of 'Margent' web site 16](#_Toc267989203)

[Figure 9 Structure of .NET Framework 17](#_Toc267989204)

[Figure 10 Data structure diagram 23](#_Toc267989205)

[Figure 11 Class diagram of namespace MMarinov.WebCrawler.Indexer pt.1 26](#_Toc267989206)

[Figure 12 Class diagram of namespace MMarinov.WebCrawler.Indexer pt.2 27](#_Toc267989207)

[Figure 13 Class diagram of namespace MMarinov.WebCrawler.Indexer 29](#_Toc267989208)

[Figure 14 Class diagram of namespace MMarinov.WebCrawler.Library 29](#_Toc267989209)

[Figure 15 Class diagram of namespace MMarinov.WebCrawler.Report 30](#_Toc267989210)

[Figure 16 Class diagram of namespace MMarinov.WebCrawler.Stemming 31](#_Toc267989211)

[Figure 17 Class diagram of namespace MMarinov.WebCrawler.Stopper 32](#_Toc267989212)

[Figure 18 Class diagram of DataFetcher.cs 33](#_Toc267989213)

[Figure 19 Class diagram of namespace MMarinov.WebCrawler.UI 34](#_Toc267989214)

[Figure 20 Overview of the Crawling application 35](#_Toc267989215)

[Figure 21 Progress bar for long time taking processes 35](#_Toc267989216)

[Figure 22 Buttons in initial state 36](#_Toc267989217)

[Figure 23 Buttons in working process 36](#_Toc267989218)

[Figure 24 Grid with statistics of previous crawling processes 36](#_Toc267989219)

[Figure 25 Initial view of 'Margent' search agent 37](#_Toc267989220)

[Figure 26 Enter query area 37](#_Toc267989221)

[Figure 27 Drop down suggestion words list 38](#_Toc267989222)

[Figure 28 Main grid of associated words 39](#_Toc267989223)

[Figure 29 Child grid with result links 40](#_Toc267989224)

[Figure 30 No records found message 40](#_Toc267989225)

[Figure 31 Too short word message 41](#_Toc267989226)

[Figure 32 Paging function and summary 41](#_Toc267989227)

[Figure 33 No connection to the DB error message 41](#_Toc267989228)

**Table of Tables**

[Table 1 dbo.Words structure 23](#_Toc267352298)

[Table 2 dbo.Files structure 24](#_Toc267352299)

[Table 3 dbo.WordsInFiles structure 24](#_Toc267352300)

[Table 4 dbo.Statistics structure 24](#_Toc267352301)

**Glossary**

* DB – Database
* EFRE – European Fund for regional development
* SQL – Structured Query Language
* MSSQL – Microsoft SQL
* SE – Search engine
* UI – User interface
* LINQ - Language-Integrated Query
* WPF – Windows Presentation Foundation
* IDE - Integrated Development Environment
* PPC – Paid-per-click
* CPC – Cost-per-click

1. ***Abstract***

This project presents a web service for multi-facets information search that returns search-results of a query in two levels – first is list of keywords that are most common with the searched topic and second is child list of links for every word. When a user clicks on that link, he will be redirected to that web page. Current project is extension of the ‘Impact’ project, but it is developed to work independently, so it could be regarded as standalone.

IMPACT project is funded by the (EFRE) and aims to promote the economic performance of Berlin in key strategic areas by research capacity and innovation. It has several branches and one that my project refers is Competence center for knowledge visualization. [1]

That web application is carried through C# program language with .NET Framework 3.5; for IDE was used Visual Studio 2008; for data storage and management were respectively used MSSQL Server 2008 and MSSQL Management Studio 2008.

Thesis project consists of two main modules: one is Web Crawler that is deployed on a server and the other one is a web application that clients use.

The Web Crawler module – its purpose is to collect significant words from every web page in the Internet and save it into DB.

The web service module – a user enters some query in the web search portal and receives results from DB, represented in the above mentioned way. The client can follow any of the result links.

1. ***Introduction***

Nowadays, computers are integral part of each area of our daily lives, thanks to the possibilities they offer. The ability to store large amounts of information and its centralization and exceptional computing capabilities allow people to search through wide used web SEs for any information they need.

Find useful information is getting harder, because of the amount of information on the web is growing at lightning speed. Computers are not intelligent enough to know what is the exact kind of information we are looking for, therefore SEs show long result list and users must find their useful information manually. Sometimes this costs long time and gets on users’ nerves, which is a hint to developers that a new, clear and simple way of presenting the information is required. The idea here is to find the main aspects of the object of interest, those that people would be most probably interested in and give them back to the users. Then, the user chooses the subject he/she is interested in, he would receive a list of URLs related to what he/she is interested in. That would be significant help of users to find useful for them information.

A SE is a searchable online database of Internet resources. It has several components: SE software, spider software, an index (database), and a relevancy algorithm (rules for ranking). The SE software consists of a server or a collection of servers dedicated to indexing Internet webpages, storing the results, and returning lists of pages to match user queries. The spidering software constantly crawls the Web, collecting webpage data for the index. The index is a database for storing the data. [2]

There are four main types of Internet SEs by their structure/way of working:

* crawler-based (traditional or common engines);
* directories (human-edited catalogs);
* hybrid engines, which are META engines;
* those using other engines’ results, and paid listings (PPC and paid inclusion engines).

Spider software belongs to crawler-based Internet SEs. In a nutshell, their work is as following: spiders read web pages, index, and rank them. Finally, they appear on SE results pages for the words and phrases most common on the indexed webpage.

Directories work in the following way: you have to submit your pages manually to one of the existing categories, your site is visited and read by a directory editor. You must be ready for long queue process as reviewing by an editor (directories use human power for indexing) takes much longer to process all pages. Most directories do not have their own ranking mechanism; they use some obvious factor to sort the URLs such as an alphabetic sequence or Google Page Rank.

Paid inclusion engines require certain fees to list your page with some differences in the working system as re-spidering or top-ranking for keywords that you choose. Moreover, most major Internet SEs utilize such schemes as a part of their indexing and ranking system. PPC engines use an auction system where keywords and phrases are associated with a cost-per-click (CPC) fee. The fundamental principle that lies at the heart of PPC process is that the higher you bid, the higher your position will be for the particular search terms.

Spider-based SEs have made their way from simple, spam-vulnerable algorithms to complex and sophisticated mechanisms that are dangerous to play with. Also, the SE optimization industry has developed a number of black-hat techniques to abuse the automatic site indexation and ranking. These techniques are referred to as SE spamming. Nowadays, they can be considered neither legitimate nor effective.

Current developed SE is crawler-based, but it extends the normal use of such engine, because it returns as a result list of keywords (most common words) and for every word you can expand a list of websites that this word appear, ordered by weight(that is the classic SE part).

By content/topic SEs could be:

* General
* Geographical limited scope
* Business
* Enterprise
* Mobile/Handheld
* Job
* Legal
* Medical
* News
* Television
* Video Games

By information type:

* Forum
* Blog
* Multimedia
* Source code
* BitTorrent
* Email
* Maps
* Question and answer
  + Human answers
  + Automatic answers
* Natural language

By model:

* Open source search engines
* Semantic browsing engines
* Social search engines
* Metasearch engines
* Visual search engines
* Search appliances
* Desktop search engines
* Usenet [3]

1. ***Evaluation of existing solutions.***

There are lots of existing SEs. After a research wasn’t found such kind of SE with keywords as a result.

3. 1. **Overview of existing solutions.**

Here are some details of the so called Big3 (most used SEs) [4]:

* Google

These both statements are proved by the next Google definition, made by the free encyclopedia “Wikipedia”: “Google is a search engine owned by Google Inc. whose mission statement is to ‘organize the world's information and make it universally accessible and useful.’ The largest search engine on the Web, Google, receives over 200 million queries each day through its various services”. This Google definition formulates the main mission of the search engine. The efforts of Google are evident, and it is confirmed by the number of queries, over 200 million each day. Such activities illustrate this Google definition, as the most popular search engine.

When a man enters the definite keyword to the search box, Google begins to scan webpages, looking for the instances. Let’s absorb in this process. Actually, Google doesn’t scan the Web during the search request. It has the huge database, called Index, where a large amount of webpages are situated. Index is constantly increasing and gets information from “spiders.” Search engine “spiders” or “crawlers” surf the Web all the time looking for some changes there. If they find recently posted pages or updates of the existing pages, the information is evaluated. If the new page is treated to be the relevant source, it gets to the Index. All updates, qualitative and non-qualitative, will influence on the rate of the pages in results of search requests.

Google sorts billions of bits of information for its users. Here are some little-known bits of information about Google:

* + Google translates billions of HTML web pages into a display format for WAP and i-mode phones and wireless handheld devices, and has made it possible to enter a search using only one phone pad keystroke per letter, instead of multiple keystrokes.
  + Google Groups comprises more than 845 million Usenet messages, which is the world's largest collection of messages or the equivalent of more than a terabyte of human conversation.
  + The basis of Google's search technology is called PageRank™, and assigns an "importance" value to each page on the web and gives it a rank to determine how useful it is. However, that's not why it's called PageRank. It's actually named after Google co-founder Larry Page.
  + Googlers are multifaceted. One operations manager, who keeps the Google network in good health is a former neurosurgeon. One software engineer is a former rocket scientist. And the company's chef formerly prepared meals for members of The Grateful Dead and funkmeister George Clinton. [5]
  + Google’s Home Page Has 35 Validation Errors and 2 warning(s)
  + The name ‘Google’ was an accident. A spelling mistake made by the original founders who thought they were going for ‘Googol’, which is a mathematical term 1 followed by one hundred zeroes.
  + Google consists of over 450,000 servers, racked up in clusters located in data centers around the world. [6]

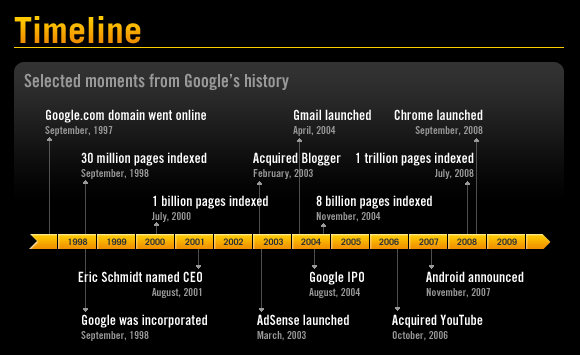


Figure Timeline of Google's history

On Figure 1 one can trace the progress of Google company. The most important here is that page indexing grows in logarithmic way.

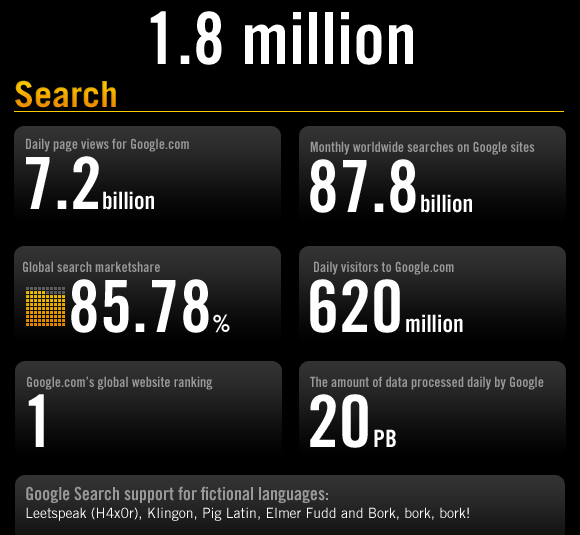


Figure Google search statistics [7]

* Yahoo!

Yahoo! is one of the three major players on the search market (the other two are Google and MSN). Being Google's main competitor, Yahoo! uses its own directory as a main source for feeding results to web surfers.

Yahoo! believes that search can be made more intelligent by aiding algorithmic results with human editors. They are trying to beat Google by improving algorithmic search results with editors. [8]

* MSN

The MSN search engine MSN is owned by Microsoft, and its history as an independent engine is rather short – MSN only recently started to use its own Web spider to compile the database of webpages. Up until this point, they used Inktomi’s database. MSN is featuring Web search and also shows news, weather, links to dozen of sites on the MSN search engine network, and offers from affiliated sites.

* Others

The search market has three definite leaders (Google, MSN, and Yahoo!), but it is not limited to these only. There are other engines possessing a significant share in the market, e.g. AltaVista, Ask, Baidu, Jeeves,Teoma, DogPile;

* Directories

The term "Directories", or "Human powered" search facilities, mainly refer to online catalogs which categorize Web sites into thematic sections. Yahoo!, along with the regular Web search, offers one of the most complete catalogs on the Web. When you submit a site to a directory, it is queued for editorial review. Usually, when you submit, you are allowed to choose the category your site will be placed under, as well as to enter the desired description and title for your site, which will show in the related category. However, the actual presence of your site is subject to the editor's decision when he or she browses your site. Directories DO NOT accept automated submissions and have special means to protect themselves from auto-submission software.

* 1. **Conclusion**

Today what people call a search engine is generally a much more complex Web search portal. It is designed as starting point for users who need to find information on the Web. However, on a search portal, you can find many different search options and services

SEs, that I have revised, show results in very similar way – page title and some matching content. Page ranking (order of the pages) mainly comes from the biggest sequence of founded query. Last years some SEs offer promotion of web sites by payment method. Most used SEs by search topic take part in general group but also have functionality a user to specify the topic. All them are crawler-based, which means that this is the most preferred structure. Therefore my SE uses the same.

There wasn’t found a SE, which works like the one, developed by me – to show results of keywords, which would be in common with the search query. Mine SE has general content, because one can search for anything. By information type it should take part in automatic answers group, because no further action from a user is needed to sort and display the results. As a crawler based engine, it has spider software, search engine software, an index database and the last important component, a relevancy algorithm. At first, spider software follows the links from the sites kept in the database for finding and changing information. That helps engine to build index of sites. After this stage, information is processed by the help of servers to calculate sites’ relevance. It is widely known that each search engine uses sophisticated and complex algorithms for this purpose.

* 1. **Purpose and objectives.**

Development of this project must be in help of knowledge visualization part of IMPACT project. It has to override the functionality of a common SE with one more level of visualization and division of the results i.e. to group link results by common keywords, which are thematically close to the searched word.

Problems, which must be resolved in process of development, are:

* collecting data from the Internet, revise it to find appropriate keywords;
* save it into a DB in an optimal for indexing way;
* copy that DB to another ‘active’ one that is used from users in WWW;
* develop fast methods for DB fetching;
* create a good stop-words list, to skip ‘parasite’ words
* compose a good ranking algorithm for sorting the results – this will provide better relative words as returned results, which is one of the most important steps;
* display the results in intuitive for users way – it will be easier for any user to see only the referenced words as an initial result and as second step list with web links to be shown for every word, on user click;

1. ***Design and description of the proposed solution.***
2. 1. **Requirements for the software system.**

The following features were given from my supervisor:

* Two-layered visualization of the results;
* Fast execution of a query;
* Easy and intuitive UI;

These are some of the problems, enumerated in 3.3 Purpose and objectives. As one can see in progress of reading, all of them were well developed and extended.

* 1. **Logical model of the software**

Communication between people and the system are presented in two separated use-case diagrams, because the solution contains two almost independent projects, and actors are pretty different. The activity diagrams are included to show the main workflow processes in abstract and clear way.

* Logical model of the Web Crawler

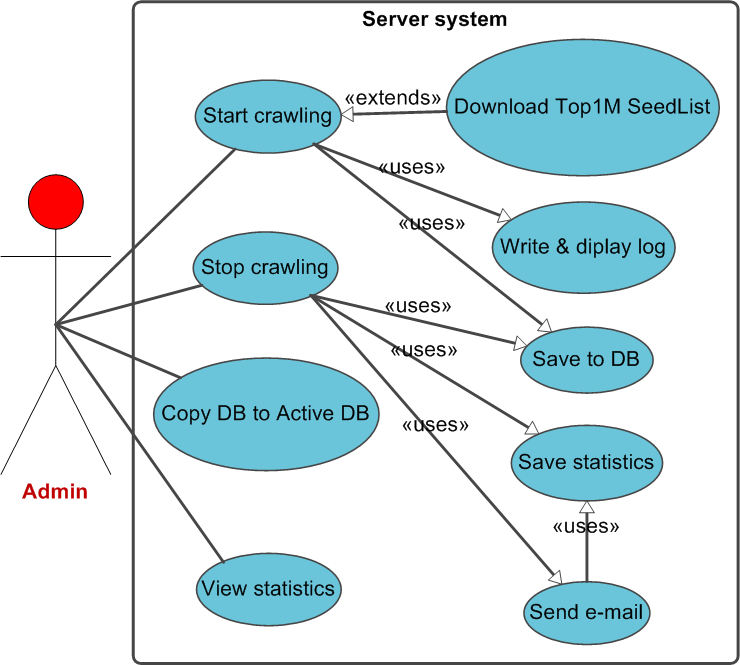


Figure Use-case diagram of the SE

The basic idea of the crawler is to be started and left to do its work at all. Because of the huge WWW and billions of web sites, I have developed it to be able to stop also by admin’s command. Available admin actions are: start crawling, stop crawling, copy DB and view statistics. Copy DB is available only when crawler is not in search progress.

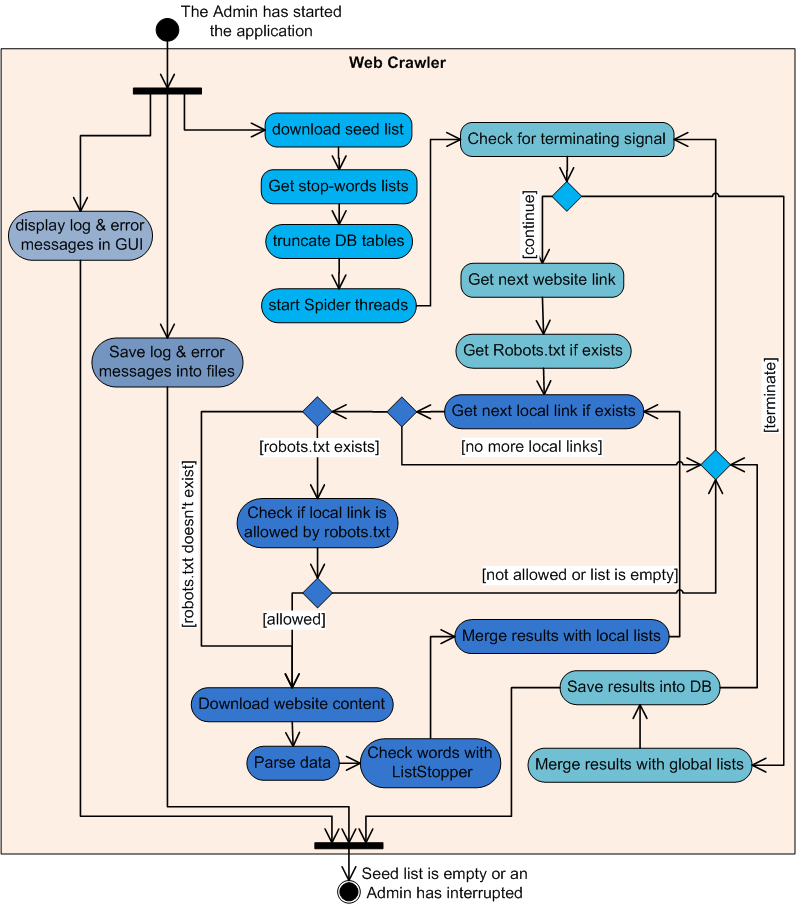


Figure Activity diagram of Start crawling - the essential process

On Figure 4 one can trace the crawling process. There are three parallel processes running during work-time:

* + One for updating the GUI, that shows messages, errors and other information about the process;
  + Second for saving some of these messages and all the errors into separated log files;
  + And the one that do all the important job with crawling the Web – here important activities are downloading seed list of web links, get ‘parasite’ words lists for different languages (used by ListStopper), truncating data in DB tables, next activities could be grouped and discussed like crawling content of a whole web site;

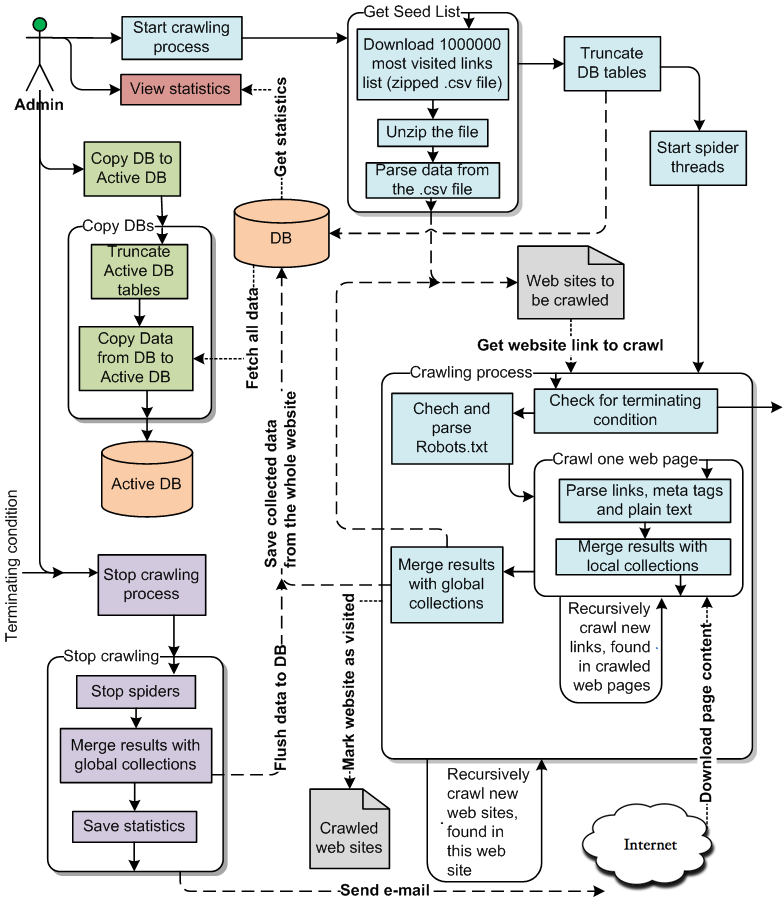


Figure Overall workflow processes in the Web Crawler

On Figure 5 I have tried to illustrate the workflow of all available actions in the Web Crawler. This includes the Admin and his actions and also activities that derive from them. Some activities are presented like grouped sub-activities for better understanding granularity of the activities. Also the two DBs are displayed, to show which one with which objects interact. The internet is also included like a cloud object. With is displayed writing into DB or any other structure. With is displaying fetching any kind of data.

First the admin causes the SE to start an action. These are actually the use-cases on Figure 3. Here are the scenarios for every case:

* + Start crawling process

Crawling process should have feed list of links to start indexing. This is provided by a list, which is downloaded from <http://alexa.com>. The list is in .csv file, which is in .zip archive. Therefore after downloading, the archive is unzipped and then the file is parsed to receive the list. Its data is stored as a GlobalURLsToVisit (it is shown as a gray page on the figure). Also lists with stop words are loaded for supported languages (EN, DE, BG).

Then tables in the DB are truncated of all previous data. Truncating is used to restart identity numeration.

After we have a start point (the seed list), the crawling process could start. The real work is done by threads a.k.a. spiders. Every spider crawls one website at a time. Crawling of a website starts with a check for robots.txt file. It contains list of relative web page that should or should not be indexed by SEs. If such exists, every web page is checked; if it doesn’t exist pages are indexed directly.

For every web page the html code is extracted and after parsing are collected: title, keywords, description, plain text. Words that are indexed are from these parts and also from the URL. Every words is checked if it’s a parasite word and if is it is skipped. Also words can be processed by PorterStemming method. When the html code is parsed, all local and external links. Results are merged with these of other iterated local pages. The web page is moved from GlobalURLsToVisit to GlobalVisitedURLs.

When all local pages are iterated, results collected for the web site are merged with ‘global’ results. Only different results are saved into DB.

* + Stop crawling process

The crawling process continues until the GlobalURLsToVisit is empty or a stop signal is received. Then spiders are stopped and results that were just collected and not saved into DB are flushed. This prevents from losing any collecting data till the moment when anyone pressed the Stop button.

Then statistics about the process are saved. Also an e-mail is send to the admin with the statistics data.

* + Copy DB to Active DB

add more here

* + View statistics

add more here

* Logical model of the ‘Margent’ search agent

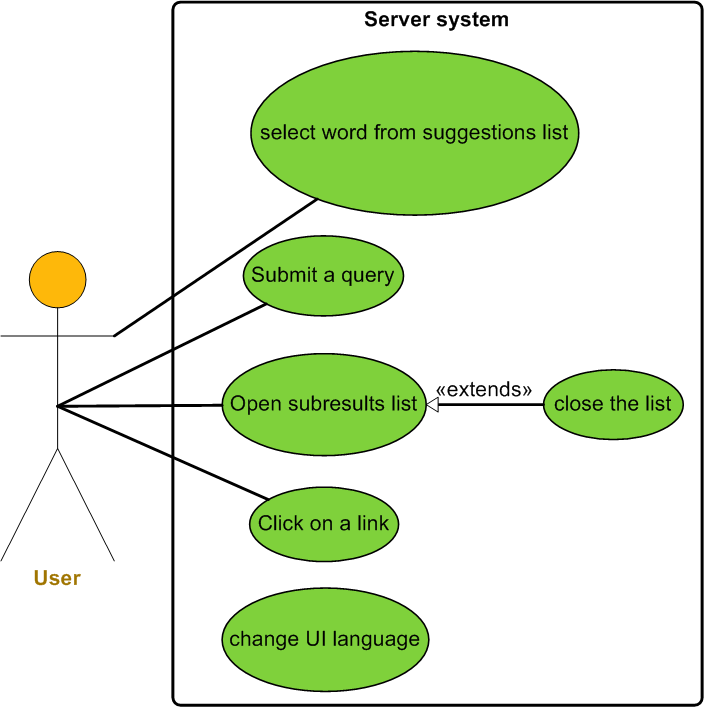
****

Figure Use-case diagram of 'Margent' web page

This diagram shows the actions that a user can do when open the ‘Margent’ search agent in a web browser. Initially he/she can enter and submit a query. As a help tool is added a suggestions list with words. Then he receives some results as a list of keywords and can click on a keyword and then a sub-list is reveled, so he can see webpages that contains the word and follow some of them. The user can also close the sub-list or open another one. He/she can change the UI language. English, German and Bulgarian are supported till now.

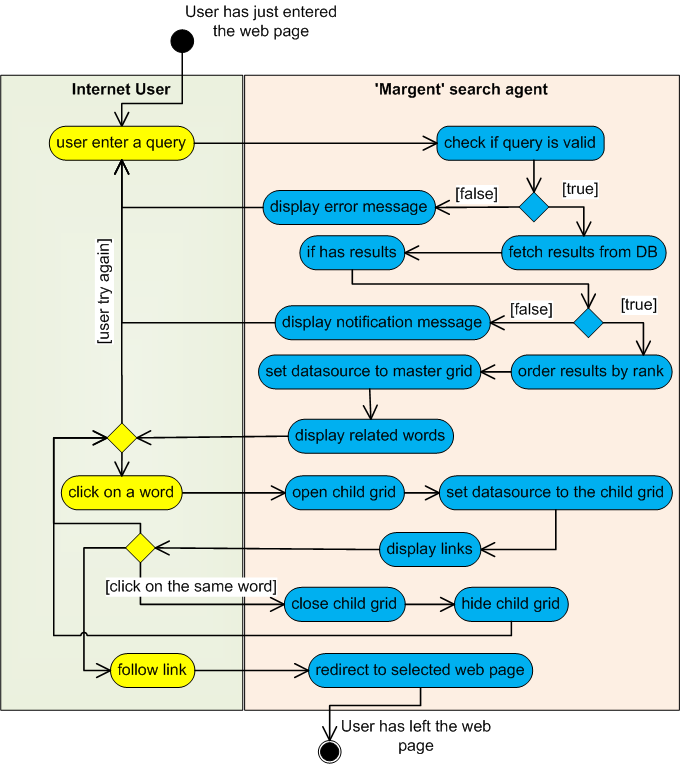


Figure Activity diagram of 'Margent' search agent workflow

The diagram of search process of the agent is shown on Figure 7. In abstract explanation the workflow is:

* + The user enters a word/words to search;
  + The system validates the query and if it’s valid, fetches the results and shows them in the page; otherwise shows appropriate message;
  + The user can now click on a word from the list and view sub-results list with links;
  + He/she can follow some link or do any of the previous steps.

Here the most important activities are: fetch results from DB, order results by rank (these must be as much as faster)

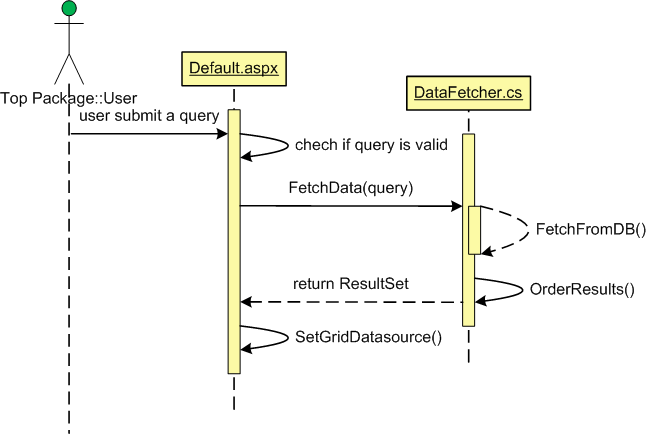


Figure Sequence diagram of 'Margent' web site

On Figure 8 one can see the interaction between the frontend page Default.aspx and the backend class, which contains methods for fetching data from DB and sorting the results. The main idea here of course is to create one and only connection to the DB for one submit of an internet user and connection’s life to be as much as minimum.

* 1. **System architecture**

The solution consists of two modules, as previously said – web crawler and the searching web-site. They work independently, but share one DB.

* Web crawler – This is a desktop application that does the following:
  + downloads seed list – list of 1000000 most visited web-sites, to start search from them
  + get stop lists
  + truncates DB tables
  + starts few processes, called spiders, that do the web crawling – collect data from the Web and save it into DB
  + copy results from working DB to actual used DB
  + saves statistics about crawling process
  + send e-mail to admin with that statistics
* Web site – Margent

Show suggestions list

* + gets the query that some user has entered
  + execute it and
  + return the result in view of nested tables – table of keywords and subtable of links to websites
  1. **Selection of programming language and development environment.**

For creation of a search engine can be use different tools, but this usually is a heavy project so it’s better to be used powerful technologies. Their selection depends on the specific tasks and the discretion of the developer. In this case, the following tools were selected:

* + 1. **.NET Framework 3.5**

The .NET Framework 3.5 is an incarnation of the mainstream Windows programming environment. Built on and extending its predecessors, its goal is to support the creation of modern applications. By building its various technologies on a common foundation, Microsoft is striving to make the whole greater than the sum of the parts, letting developers create applications that use the various parts of the .NET Framework 3.5 in a coherent way. [11]

**Base Class Library**

**Windows Communication Foundation**

**Windows Workflow Foundation**

**Windows Presentation Foundation**

**Windows**

**CardSpace**

**. . .**

**. . .**

***.NET Framework 2.0***

***.NET Framework 3.0 Additions***

***.NET Framework 3.5 Additions***

**ASP.NET**

**ASP.NET AJAX**

**. . .**

**. . .**

**. . .**

**Common Language Runtime (CLR)**

**LINQ**

Figure Structure of .NET Framework

Everything in the .NET Framework depends, as it always has, on the Common Language Runtime (CLR). A large group of classes, known as the .NET Framework class library, is built on top of the CLR. This library has expanded with each release, as the figure shows. The .NET Framework 2.0 provided the fundamentals of a modern development environment, including the base class library, ASP.NET, ADO.NET, and much more. The .NET Framework 3.0 didn’t change any of this, but it did add four important new technologies: WCF, WF, WPF, and Windows CardSpace.

The changes in the .NET Framework 3.5 affect several parts of the 3.0 release. ASP.NET gets AJAX support, while LINQ is available for use with ADO.NET and in other ways. Various additions were made to the base class library, such as addition of a type supporting sets—unordered collections of unique elements—and improved encryption support. WCF, WF, and WPF each get enhancements as well, as described later in this overview.

* LINQ

Creating a common approach for accessing diverse data isn’t an easy task. Making that approach comprehensible to developers—not bogging them down in more complexity—is even harder. To address this, LINQ relies on a common and quite

The syntax of the query is reminiscent of SQL, today’s standard language for accessing relational data. This makes sense, since SQL is a widely used language, one that many developers know. Yet it’s important to understand that even though it looks somewhat like SQL, the LINQ query shown above isn’t an embedded SQL statement. Instead, it’s pure C#, part of the language itself. This means that the query can use other program variables, be accessed in a debugger, and more. The name of this technology is “Language-Integrated Query” for a reason: the statements for querying diverse kinds of data are integrated directly into the programming language.

And despite its similarity to SQL, this example query isn’t limited to accessing only relational data. In fact, the .NET Framework 3.5 includes several different LINQ variations, all of which use the same basic syntax for queries. Those variations include the following: LINQ to ADO.NET, LINQ to Objects, LINQ to XML

* + LINQ to ADO.NET: Provides object/relational (O/R) mapping. LINQ to SQL, translates a query like the one above into a SQL query, then issues it against tables in a SQL Server database.

Like SQL, LINQ also defines other operators for queries. They include things such as OrderBy, which determines how results are ordered; GroupBy, which organizes selected data into groups; and arithmetic operators such as Sum. And once again, these can be used generally across the LINQ varieties—they’re not just for the LINQ to SQL option.

LINQ’s creators aimed at several targets, including providing O/R mapping for .NET applications, allowing a common syntax for working with different kinds of data, integrating that syntax directly into the programming language, and more. As with everything else described in this introduction, the goal is to make life better for developers working with Visual Studio 2008 and the .NET Framework 3.5.

* WPF

The goal of WPF is to address the challenges of creating user interfaces for modern applications.

* + 1. **CSLA.NET**

Rockford Lhotka’s CSLA .NET framework is an application development framework that reduces the cost of building and maintaining applications.

The framework enables developers to leverage the power of object-oriented design as the basis for creating powerful applications. Business objects based on CSLA automatically gain many advanced features that simplify the creation of WPF, ASP.NET MVC, Web Forms, WCF, WF and Web Services interfaces.

CSLA .NET allows great flexibility in object persistence, so business objects can use virtually any data sources available. The framework supports 1-, 2- and n-tier models through the concept of mobile objects. This provides the flexibility to optimize performance, scalability, security and fault tolerance with no changes to code in the UI or business objects. [18][16]

* + 1. **MS Visual Studio 2008**

Microsoft Visual Studio is an IDE from Microsoft. It can be used to develop console and graphical UI applications along with Windows Forms applications, web sites, web applications, and web services in both native code together with managed code for all platforms supported by Microsoft Windows, Windows Mobile, Windows CE, .NET Framework, .NET Compact Framework and Microsoft Silverlight.

Visual Studio includes a code editor supporting IntelliSense as well as code refactoring. The integrated debugger works both as a source-level debugger and a machine-level debugger. Other built-in tools include a forms designer for building GUI applications, web designer, class designer, and database schema designer. It accepts plug-ins that enhance the functionality at almost every level—including adding support for source-control systems (like Subversion and Visual SourceSafe) and adding new toolsets like editors and visual designers for domain-specific languages or toolsets for other aspects of the software development lifecycle (like the Team Foundation Server client: Team Explorer).

Visual Studio supports different programming languages by means of language services, which allow the code editor and debugger to support (to varying degrees) nearly any programming language, provided a language-specific service exists. Built-in languages include C/C++ (via Visual C++), VB.NET (via Visual Basic .NET), C# (via Visual C#), and F# (as of Visual Studio 2010). Support for other languages such as M, Python, and Ruby among others is available via language services installed separately. It also supports XML/XSLT, HTML/XHTML, JavaScript and CSS. Individual language-specific versions of Visual Studio also exist which provide more limited language services to the user: Microsoft Visual Basic, Visual J#, Visual C#, and Visual C++.

Visual Studio 2008 features include an XAML-based designer (codenamed Cider), workflow designer, LINQ to SQL designer (for defining the type mappings and object encapsulation for SQL Server data), XSLT debugger, JavaScript Intellisense support, JavaScript Debugging support, support for UAC manifests, a concurrent build system, among others. It ships with an enhanced set of UI widgets, both for Windows Forms and WPF. It also includes a multithreaded build engine (MSBuild) to compile multiple source files (and build the executable file) in a project across multiple threads simultaneously. It also includes support for compiling PNG compressed icon resources introduced in Windows Vista. An updated XML Schema designer will ship separately sometime after the release of Visual Studio 2008. [12]

* + 1. **MS SQL Server 2008 и Microsoft SQL Server Management Studio 2008**

Microsoft SQL Server is a relational model DB server produced by Microsoft. Its primary query languages are T-SQL and ANSI SQL.

SQL Server 2008 aims to make data management self-tuning, self-organizing, and self-maintaining with the development of SQL Server Always On technologies, to provide near-zero downtime. SQL Server 2008 also includes support for structured and semi-structured data, including digital media formats for pictures, audio, video and other multimedia data. In current versions, such multimedia data can be stored as BLOBs (binary large objects), but they are generic bitstreams. SQL Server 2008 can be a data storage backend for different varieties of data: XML, email, time/calendar, file, document, spatial, etc. as well as perform search, query, analysis, sharing, and synchronization across all data types.

Other new data types include specialized date and time types and a Spatial data type for location-dependent data. Better support for unstructured and semi-structured data is provided using the new FILESTREAM data type, which can be used to reference any file stored on the file system. Structured data and metadata about the file is stored in SQL Server database, whereas the unstructured component is stored in the file system. Such files can be accessed both via Win32 file handling APIs as well as via SQL Server using T-SQL; doing the latter accesses the file data as a BLOB. Backing up and restoring the database backs up or restores the referenced files as well. SQL Server 2008 also natively supports hierarchical data, and includes T-SQL constructs to directly deal with them, without using recursive queries.

The Full-Text Search functionality has been integrated with the database engine, which simplifies management and improves performance.

SQL Server includes better compression features, which also helps in improving scalability. It enhanced the indexing algorithms and introduced the notion of filtered indexes. It also includes Resource Governor that allows reserving resources for certain users or workflows. It also includes capabilities for transparent encryption of data (TDE) as well as compression of backups. SQL Server 2008 supports the ADO.NET Entity Framework and the reporting tools, replication, and data definition will be built around the Entity Data Model. SQL Server Reporting Services will gain charting capabilities from the integration of the data visualization products from Dundas Data Visualization Inc., which was acquired by Microsoft. On the management side, SQL Server 2008 includes the Declarative Management Framework which allows configuring policies and constraints, on the entire database or certain tables, declaratively. The version of SQL Server Management Studio included with SQL Server 2008 supports IntelliSense for SQL queries against a SQL Server 2008 Database Engine. SQL Server 2008 also makes the databases available via Windows PowerShell providers and management functionality available as Cmdlets, so that the server and all the running instances can be managed from Windows PowerShell. [13]

* + 1. **C#**

C# is a type-safe, object-oriented language that is simple yet powerful, allowing programmers to build a breadth of applications. Combined with the .NET Framework, Visual C# 2008 enables the creation of Windows applications, Web services, database tools, components, controls, and more.

As an object-oriented language, C# supports the concepts of encapsulation, inheritance, and polymorphism. All variables and methods, including the Main method, the application's entry point, are encapsulated within class definitions. A class may inherit directly from one parent class, but it may implement any number of interfaces. Methods that override virtual methods in a parent class require the override keyword as a way to avoid accidental redefinition. In C#, a struct is like a lightweight class; it is a stack-allocated type that can implement interfaces but does not support inheritance.

In addition to these basic object-oriented principles, C# makes it easy to develop software components through several innovative language constructs, including the following:

* Encapsulated method signatures called delegates, which enable type-safe event notifications.
* Properties, which serve as accessors for private member variables.
* Attributes, which provide declarative metadata about types at run time.
* Inline XML documentation comments.
* Language-Integrated Query (LINQ) which provides built-in query capabilities across a variety of data sources.

If you have to interact with other Windows software such as COM objects or native Win32 DLLs, you can do this in C# through a process called "Interop." Interop enables C# programs to do almost anything that a native C++ application can do. C# even supports pointers and the concept of "unsafe" code for those cases in which direct memory access is absolutely critical. [14]

The C# build process is simple compared to C and C++ and more flexible than in Java. There are no separate header files, and no requirement that methods and types be declared in a particular order. A C# source file may define any number of classes, structs, interfaces, and events.

Some more interesting features of C# language, used in the process of the current project development:

* Lambda expressions

C# 2.0 (which shipped with VS 2005) introduced the concept of anonymous methods, which allow code blocks to be written "in-line" where delegate values are expected.

Lambda Expressions provide a more concise, functional syntax for writing anonymous methods. They end up being super useful when writing LINQ query expressions - since they provide a very compact and type-safe way to write functions that can be passed as arguments for subsequent evaluation. [15]

* Anonymous types

Anonymous types are a convenient language feature of C# and VB that enable developers to concisely define inline CLR types within code, without having to explicitly define a formal class declaration of the type.

Anonymous types are particularly useful when querying and transforming/projecting/shaping data with LINQ. [16]

* Query syntax

Query syntax is a convenient declarative shorthand for expressing queries using the standard LINQ query operators. It offers a syntax that increases the readability and clarity of expressing queries in code, and can be easy to read and write correctly. Visual Studio provides complete intellisense and compile-time checking support for query syntax.

Under the covers the C# and VB compilers take query syntax expressions and translate them into explicit method invocation code that utilizes the new Extension Method and Lambda Expression language features in "Orcas". [17]

* Object Initializers
* Collection Initializers  
  + 1. **TortoiseSVN**

TortoiseSVN is a really easy to use Revision control / version control / source control software for Windows.

Revision control or software configuration management (SCM), is the management of changes to documents, programs, and other information stored as computer files. It is most commonly used in software development, where a team of people may change the same files. Changes are usually identified by a number or letter code, termed the "revision number", "revision level", or simply "revision". Revisions can be compared, restored, and with some types of files, merged. [19]

Software tools for revision control are essential for the organization of multi-developer projects.

Traditional revision control systems use a centralized model where all the revision control functions take place on a shared server. If two developers try to change the same file at the same time, without some method of managing access the developers may end up overwriting each other's work. Centralized revision control systems solve this problem in one of two different "source management models": file locking and version merging. [20]

* + 1. **Easy SMTP Server**

Easy SMTP Server is a simple easy-to-use program working in background that lets you send e-mail messages directly from your PC bypassing ISP's SMTP servers. Using this program instead of your ISP's SMTP server you will increase your e-mail security and privacy as well as get rid of annoying change of settings for your e-mail program. Easy SMTP Server is supported by all email programs including Outlook Express and Eudora. It may serve multitude of SMTP connections concurrently to use your Internet connection up to maximum. [21]

1. ***Implementation of the software system:*** 
   1. **Data structure**

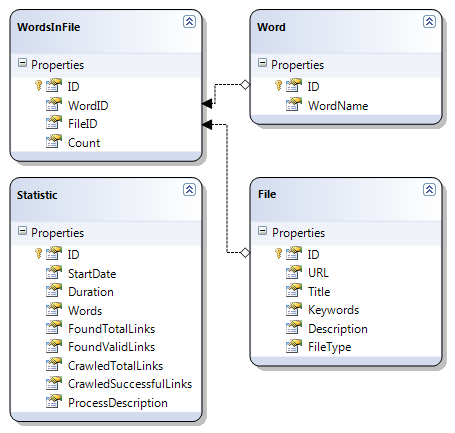
****

Figure Data structure diagram

On Figure 10 one can see the dependencies in the DB.

* Table Words

Contains words, found in crawled web pages.

Table dbo.Words structure

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Field | Type | Nullable | Key | Default | Extra |
| ID | Bigint |  | PK |  | AutoIncrement |
| WordName | Nvarchar(50) |  |  |  |  |

Fields description:

* + ID – Unique ID number for the table, that helps for better indexing
  + WordName – the word itself
* Table Files

Contains data about crawled web pages.

Table dbo.Files structure

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Field | Type | Nullable | Key | Default | Extra |
| ID | Bigint |  | PK |  | AutoIncrement |
| URL | Nvarchar(2500) |  |  |  |  |
| Title | Nvarchar(200) |  |  |  |  |
| ImportantWords | Nvarchar(500) |  |  |  |  |
| WeightedWords | Nvarchar(500) |  |  |  |  |
| FileType | Tinyint |  |  |  |  |

Fields description:

* + ID – Unique ID number for the table, that helps for better indexing
  + URL – web link to the current document
  + Title – title of the webpage,mp3,text document or other kind of file
  + Keywords – words collected from meta-tag ’keywords ’, if such exists
  + Description – content of meta-tag ’description’, if such exists
  + FileType – for now are supported web-pages, txt, mp3 and pdf files
* Table WordsInFiles

That table stores connections like: which word is situated in which web sites and how many times it occurs in each site.

Table dbo.WordsInFiles structure

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Field | Type | Nullable | Key | Default | Extra |
| ID | Bigint |  | PK |  | AutoIncrement |
| FileID | Bigint |  | FK |  |  |
| WordID | Bigint |  | FK |  |  |
| Count | int |  |  |  |  |

Fields description:

* + ID – Unique ID number for the table, that helps for better indexing
  + FileID – ID of the file that contains the word
  + WordID – ID of the word
  + Count – how many times the word occurs in each site
* Table Statistics

Saves information about every execution of the crawler.

Table dbo.Statistics structure

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Field | Type | Nullable | Key | Default | Extra |
| ID | int |  | PK |  | Auto Increment |
| StartDate | datetime |  |  |  |  |
| Duration | Varchar(50) |  |  |  |  |
| Words | Bigint |  |  |  |  |
| FoundTotalLinks | Bigint |  |  |  |  |
| FoundValidLinks | Bigint |  |  |  |  |
| CrawledTotalLinks | Bigint |  |  |  |  |
| CrawledSuccessful Links | bigint |  |  |  |  |
| ProcessDescription | Nvarchar(250) |  |  |  |  |

Fields description:

* + ID – Unique ID number for the table, that helps for better indexing
  + StartDate – time of starting the crawling process
  + Duration – time of working, presented in format dd:HH:mm(total min)
  + Words – Count of total unique words found
  + FoundTotalLinks – Count of total links found in the web pages
  + FoundValidLinks – Count of valid links that we want to follow(depends on which file types are allowed)
  + CrawledTotalLinks – Count of tried links
  + CrawkedSuccessfulLinks – Count of links that were accessed.
  + ProcessDescription – describes which properties are set.
  1. **Description of program modules**

The following modules were developed, for division of UI, DB access and other classes:

* CrawlerEngine

This is the module that does most of the work and it’s the main project for the crawling process. It truncates data from tables, downloads list with top 1000000 visited web sites and downloads and parse data from the Internet. That means keeping lists of visited and to be visited links, parsing Robot.txt files, interfaces for stemming and ignoring some words and prepare collected data for storing into DB.

Here are most of the important connections between the classes in that module, displayed in class diagrams, grouped by namespace. They are not placed in 4.2 Logical model of the software, because the idea is to be shown properties and methods in the classes and their inheritance, not the sequence of work.

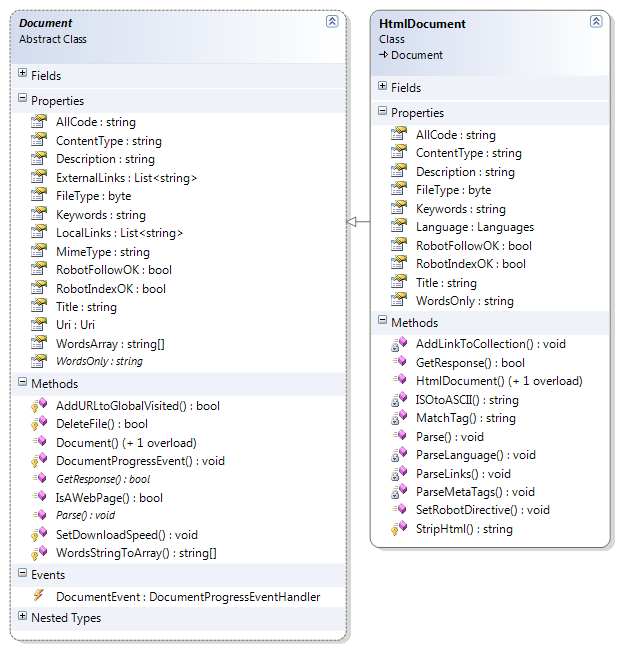


Figure Class diagram of namespace MMarinov.WebCrawler.Indexer pt.1

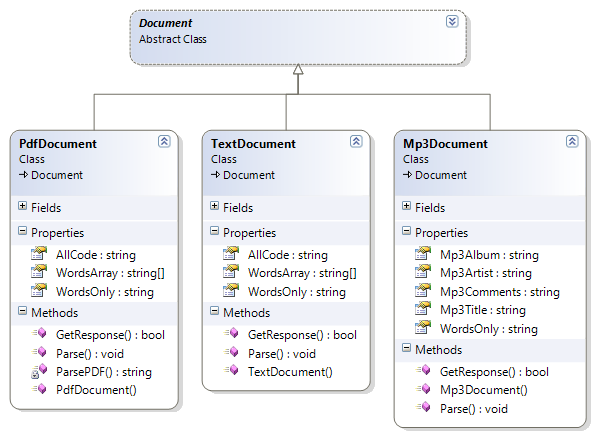
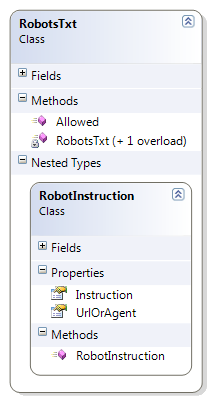
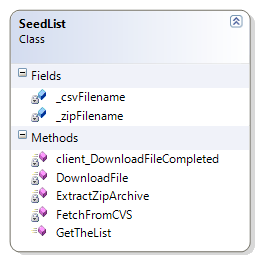
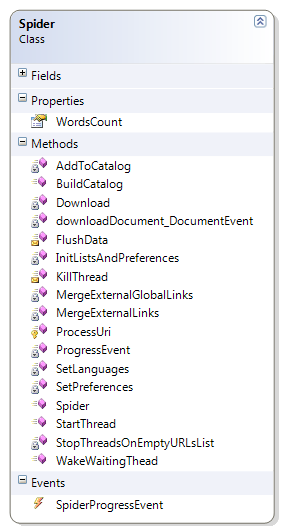
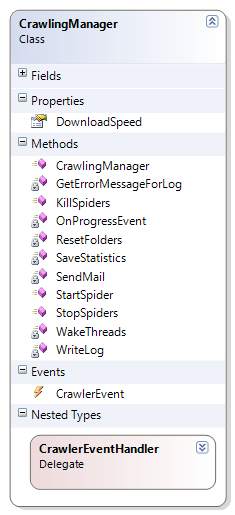


Figure Class diagram of namespace MMarinov.WebCrawler.Indexer pt.2

Document is an abstract class that represents all files and documents that the SE could work with. HTMLDocument, TextDocument, Mp3Document and PDFDocument derive from it. PDFDocument download the content in a temporary folder and when the data is parsed from it, the file is deleted. Last two classes need a bit more development improvement. PDFDocument use a method from a foreign DLL for parsing data. PDFDocument also PDFDocument use a method from a foreign DLL for getting ID3 tags content.

Downloading of the file is implemented in GetResponse() method for every kind of document. The significant data is get in the Parse() method.

­­­­­­



1

1..n

1

n

1

1

Figure Class diagram of namespace MMarinov.WebCrawler.Indexer

SeedList is a class that downloads a .zip archive, which contains a .csv file. Then extract the file from the archive and get list with html links. The idea is the following: that file contains list of so called Top1M – top 1000000 of the most visited links on the Internet. It is downloaded from alexa.com and the good part is that it is daily updated. The SE has a flag to download or not that file, so you can make your own list of initial links the crawling to start from.

As you can understand of the title of the class, CrawlingManger is the class that runs, controls, watches, stops and at all manages the crawling process e.g. spider threads. It also has EventHandler for receiving messages from the spiders and fires event for sending messages to the GUI.

Spider class starts a thread that crawls through the Internet. It does the following for every website:

* Checks from Robots.txt – if exists, checks which relative links to index;
* Starts crawling from domain of the website and then recursively crawl in all founded relative links. Crawling is implemented by width search method and also has a recursion level limit;
* Collects web links and their content – there are methods for excluding some words, depending on the language of the page, save results to collection for that website;
* When all local links for the current website are indexed, the collected data is merged with the global collection and is saving into DB. All new external links are added to ‘links to be crawled’ list. This is an optimal way to rarely access the DB.

RobotsTxt class parses the content of a Robots.Txt file. Such file contains information like which relative links a website could be indexed. It also could contain different variations for different crawlers.

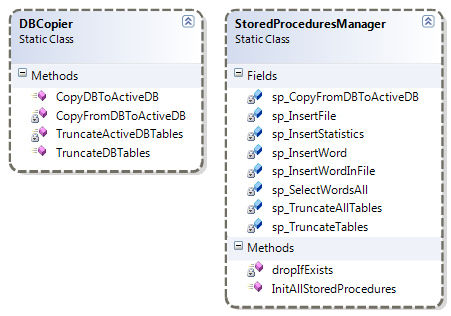


Figure Class diagram of namespace MMarinov.WebCrawler.Library

DBCopier class is used to copy all data from the DB, which the SE saves into, to the Active DB, that ‘Margent’ search agent use. Also have methods for truncating all tables for both DBs.

StoredProceduresManager is called on every start of the crawler to recreate all stored procedures. They initially were created direct on the server, but I decided that is better to be in the code, in case of using different servers. In that case they will be dynamically created.

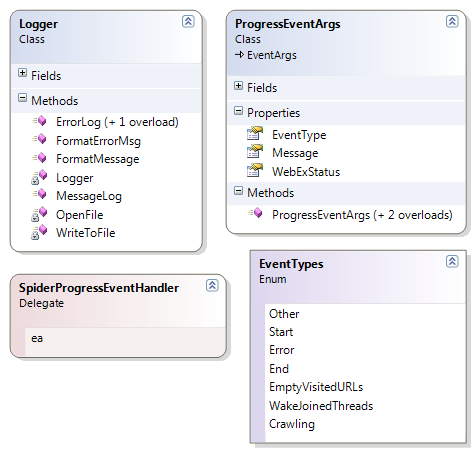
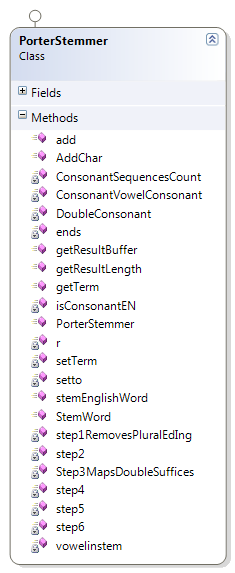
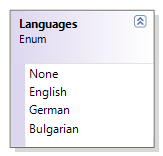
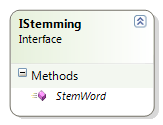
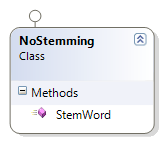


Figure Class diagram of namespace MMarinov.WebCrawler.Report

Such kind of software should have opportunity to be watch at every time, and not to stop on any error on a web page, so a lot of messages and errors are passed to the GUI and also saved into different log files. Logger class is used to format and save all the messages and errors. Spiders send any kind of messages like events.

For better vision of different kind of messages is developed ProgressEverArgs, which contain message or error data. CrawlingManager listens for events(with ProgressEverArgs parameter) from working spiders and displays and saves them.

EventTypes is an enumeration that describes what kind of messages could be passed.



1

1

Figure Class diagram of namespace MMarinov.WebCrawler.Stemming

In linguistic morphology, stemming is the process for reducing inflected (or sometimes derived) words to their stem, base or root form – generally a written word form. The stem need not be identical to the morphological root of the word; it is usually sufficient that related words map to the same stem, even if this stem is not in itself a valid root. The process of stemming, often called conflation, is useful in search engines for query expansion or indexing and other natural language processing problems.

Stemming programs are commonly referred to as stemming algorithms or stemmers. [9]

The class PorterStemmer was taken from the internet. I only make some insignificant changes. Stemming algorithm works only with English words yet. [10][11]

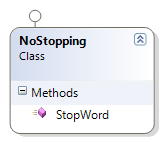
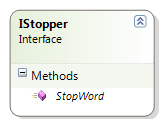
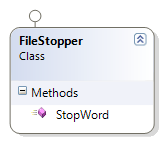
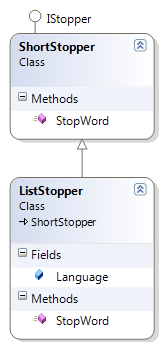


Figure Class diagram of namespace MMarinov.WebCrawler.Stopper

Stopping is method of excluding some words. There are developed two different methods of stopping words to be indexed:

The first one is realized in ShortStopper class and excludes words that have words count which is out of the range [3;50].

Say sth about fetching stop words list

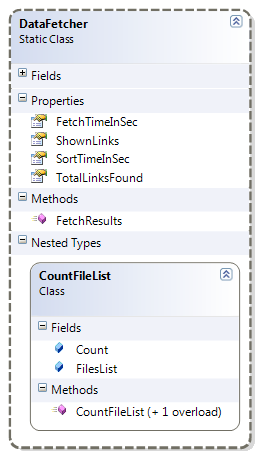
The other one is ListStopper and extends the ShortStopper class. It has list of words to exclude depending on given language. Till now it stops words that are part of Languages enumeration – Bulgarian, German, English. FileStopper class is in basic level of development and it will contain file types that must be skipped if such are indexed. NoStopping class is just used if StoppingMode is set to Off. All stopping classes implement IStopper interface and its only method – StopWord() as you can see on Figure 17.

* MMWebCrawler – web site ‘Margent’

This module is the client-side of the project. It is represented by a web page, which has a common SE design. Its work is to receive a query and return the result. Results are visualized in a gridview of relayted words, which has paging functionality and is ordered descending by count of the results. That grid contains collapsible grid for every result.

The file structure of the module is very well separated. There are individual class for the logic that process the search query – DataFetcher.cs, and another one for the interface (Default.aspx). Also StyleSheets for the design of the page is pulled out in a Page.css file and styles for the gridviews are saved in GridViews.css. A bit of JavaScript code is used for show/hide the child grid and is also divided in a .js file. It has reference to DALWebCrawlerActive module, which is responsible for the DB connection.

There is no need of site map here, because the website contains only one web page – Default.aspx. The interaction between the frontend page and the backend class DataFetcher could be seen on Figure 8.

The DataFetcher is a class (Figure 18) that is correspondent between the frontend and the DB. It receives a query of words to search and returns a result-set, a table of type [Word, CountFileList]. CountFileList is an inner class that stands for every word and contains its total count and list of links that contains it.

DataFetcher also returns summary info about fetch time from DB, sort time and links found.

Figure Class diagram of DataFetcher.cs

* DALWebCrawlerActive

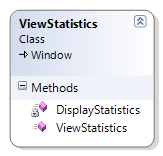
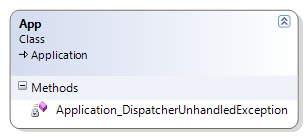
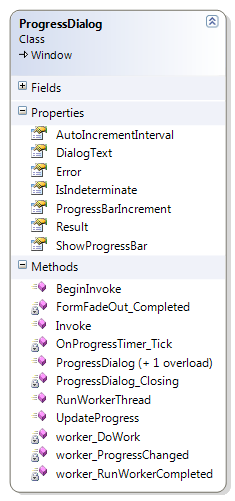
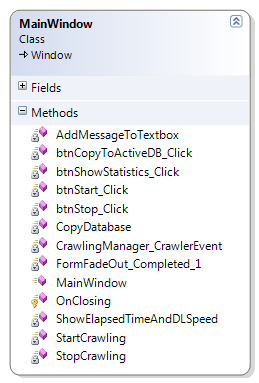
Any good software, which includes interaction with a DB, has a separate layer which is responsible for this connection. So that is the data access layer, used by the client-side, which is MMWebCrawler module. It uses LINQtoSQL connection, which provides very good business objects.

Business classes here are generated according to the attached DB, so the diagram with their connections is the same like that on Figure 10 Data structure diagram.

* MMarinovCrawler

That is a GUI application that an admin uses to operate with the SE. The admin can start or stop the process, copy the new DB to the one, that ‘Margent’ uses; watch the overall process e.g. links being crawled, errors, catch exceptions, elapsed time and more. He/she can also see statistics of previous crawling processes.

That is the class diagram that represents the interactions between the windows in the GUI module:



0..1

0..1

1

Figure Class diagram of namespace MMarinov.WebCrawler.UI

The application starts from App class, which calls the MainWindow. As the title hits, that is the main GUI of the program. An admin can control and watch the crawling process from it.

ProgressDialog window is shown when a long-time process is executed and restricts the access to the main form.

ViewStatistics window can be opened at any time. It just fetches statistics collection from the DB.

* + 1. **Structure and organization of the GUI**
* GUI of the SE



Figure Overview of the Crawling application

On Figure 20 you can see the crawling application in progress. There are the following fields:

* + Indexed links – shows which link is being indexed by which spider and current number of the link;
  + Crawled domains – displays domains/websites being indexed, also some messages like start of the spiders, when they are waiting;
  + Web, protocol and timeout exceptions – there are many pages that cannot be accessed and here are more common cases;
  + Errors – shows errors that are differ from above ones;

There are also some labels showing info about current download speed, total links found, and elapsed time.

At the bottom of the window you can see a status bar, showing current action.

D:\MMarinov_docs\Universitet\Thesis\images\crawler-progressbar.PNG

Figure Progress bar for long time taking processes

This progressbar shows over the main windows when:

* + Starting the crawling process, for the initializing process;
  + Stop the crawling – while killing the spiders and flush data to DB;
  + Copy the DB to the active DB;



Figure Buttons in initial state

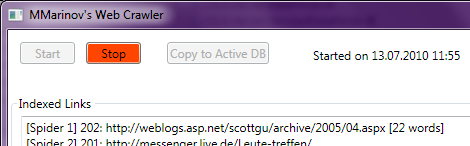


Figure Buttons in working process

On Figure 22 and Figure 23 you can see the dependency between there 3 action buttons, according to working status of the crawler.

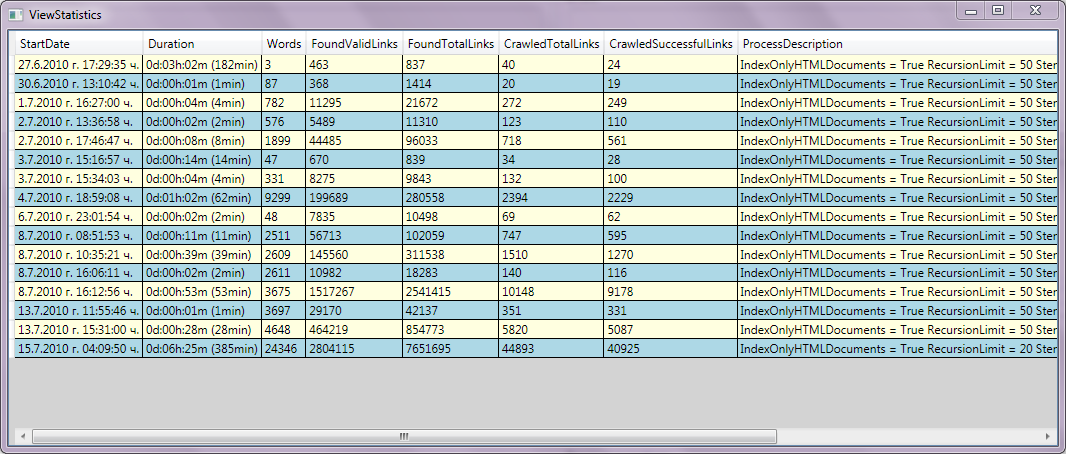


Figure Grid with statistics of previous crawling processes

This is a window that is opened on button Statistics click. It displays a grid with full info of all previous crawling processes.

* GUI of ‘Margent’ web page



Figure Initial view of 'Margent' search agent

On Figure 25 you can see the initial view of the web page. It is only one text box for entering a query and a submit button.

****

Figure Enter query area

When the text box is on focus (Figure 26), its style is changed for higher contrast of the action.



Figure Drop down suggestion words list

When more than two letters are entered, a drop down pop up with suggestion words list. It is updated on every key-press. The idea is only to help the user with some words (Figure 27).

****

Figure Main grid of associated words

The grid contains two columns:

* + Word – related words;
  + Rank – used to order the words. It comes from total number of appearance of the word in all web pages;

****

Figure Child grid with result links

For every word could be open a child grid with links, containing the word. Every record of it contains:

* + Title of the page
  + Description
  + Web address of the page

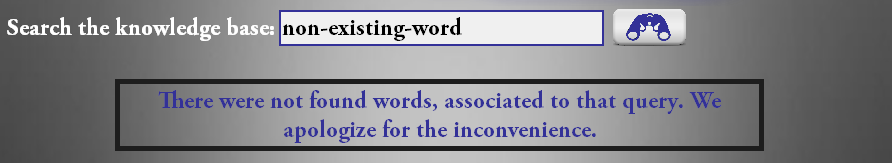
****

Figure No records found message

On Figure 30 you can see the message that is displayed when there is no coincidence in the DB with the search word.

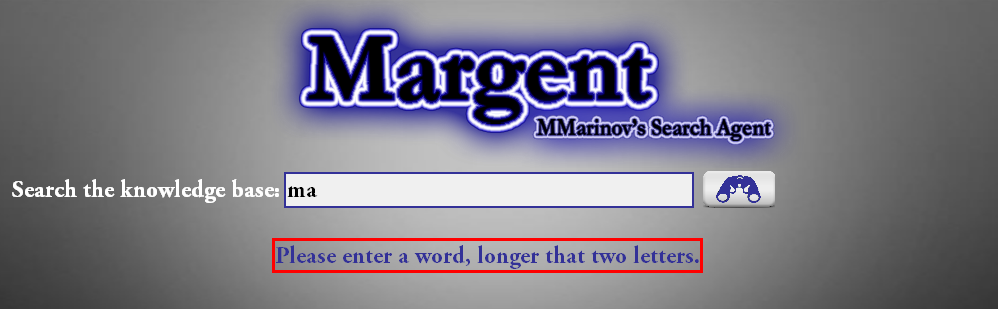


Figure Too short word message

There is a functionality that prevent searching a word that is shorter that three letters, because these kind of words are not indexed. That’s why the message on Figure 31 is shown.

****

Figure Paging function and summary

There is a paging functionality with a fancy slider. You can also see some summary under the main grid (bottom of Figure 30).

****

Figure No connection to the DB error message

This (Figure 33) is how the error message for no connection to the DB looks like.

* 1. **Instructions for using the software system:** 
     1. **User Guide**
* Instructions for using the crawling SE

When an admin starts the application, he/she can do the following actions:

* + Start the crawling process

This happens on button Start click. Then a progress-bar shows while initializing the process (Figure 21) and crawling starts after a while. Button Stop comes enabled but buttons Start and Copy to active DB become inactive (Figure 22 becomes Figure 23).

On pressing the Stop button, the progress-bar shows again while flushing some results to DB. Then visibility of these three buttons become like in the beginning.

* + Copy DB to the DB that ‘Margent’ search agent uses

The admin may decide to use the new DB, so he/she can copy it to the one that ‘Margent’ search agent uses. The progress-bar is shown also here while the copying is processing.

* + View statistics of previous crawling processes

The admin can open statistics windows at any time from button Statistics (Figure 24).

* + Can watch all log and error messages that are passed by the crawling spiders. Also can watch the download speed and number of indexed links.
* Instructions for using ‘Margent’ search agent

When a user enter the web-site, he/she can only enter a word (or more) to search. The query has e restriction of searching words, shorter than three letters.

When the query is executed, the first level results are displayed in a grid-view. The user can click on a word (a row from the grid) and an inner grid is displayed. It contains links, which contain selected word. He/she can follow a link or open another child grid, or close the current.

There is also a fancy paging functionality with a slider and some summary info at the bottom of the page. One fetching the results, every action is performed without reloading the page.

* + 1. **Instructions and requirements for installing the system.**

You need the minimum of the following to run the SE:

* Hardware requirements - Server system with at least:
  + Memory: 4GB RAM
  + Processor: Quad-core
  + 1TB free disk space
* Software requirements:
  + Windows XP or above 64 bit version
  + .NET Framework 3.5
  + MS SQL Server 2008 Professional

The admin that uses the application should know all the properties in the configuration file, so he/she could set it to work properly in any cases. The configuration file MMarinovCrawler.exe.config is in XML format and is located in the same directory like the MMarinovCrawler.exe file. It must be with the same name, in case of renaming the .exe. All settings are situated in the <appSettings> tag and are in format <add key="titleOfTheSetting" value="valueOfTheSetting"/>. Here are the instructions needed to configure the crawler:

IndexOnlyHTMLDocuments – if the value is true will index only standard web page files like html,aspx,jsp … otherwise will also index and txt,pdf and mp3 files

ThreadsCount – how many spider threads to be started;

RequestTimeout - Seconds to wait for a page to respond, before giving up;

RecursionLimit - Limit to the number of 'levels' of links to follow;

SpeedLimit - Request another page after waiting x seconds;

StemmingModeEnabled - Whether to use stemming (English only) - false/true;

StoppingType - Whether to use stop words (English, German,Bulgarian), and what mode [ Off | Short | List ]. Recommended to be List mode to remove ’spam’ words.

UserAgent – name of user agent to use

WorkingPath – directory with read and write access to use for log and some temp files

DefaultLanguage – default is en-US

SeedURLsSource - <http://s3.amazonaws.com/alexa-static/top-1m.csv.zip>

URL with a zip archive with the top 1000000 websites from Alexa; leave it blank if you want to create your own list.

DB:WebCrawler – connection string to a server to use

ConnectionStringActive – connection string to server that the search agent uses.

For the ‘Margent’ web-site you need Internet connection and a browser that support ActiveX.

Any user can access the ‘Margent’ search agent at address <http://i.must.deploy.it.on.the.server.PLS>. Then he/she sees the initial view (Figure 25).

The only one setting that and admin needs to do here is to set the connection string to the DB. The configuration file here is web.config. The structure is the same like of MMarinovCrawler.exe.config. The setting is situated in the <appSettings> tag and the key must be ConnectionStringActive.

1. ***Tests and results***

Hardware configuration, used for testing:

* Processor: [Dual Xeon E5520](http://ark.intel.com/Product.aspx?id=40200)
* Memory: 24GB DDR3 tri-channel 1066MHz ECC SDRAM
* HDD: Dual 1,5TB in Raid 0
* Dual 2408WFP UltraSharp 24″ Widescreen Flat Panel LCD Monitor(Analog&DVI)
* Dual 512MB PCIe x16 nVidia Quadro FX580, Quad Monitor DisplayPort,DVI

Tests, implemented with the SE:

There is a table Statistics in the DB that stores information about every crawling process. It comes in very use here for testing comparison. On Figure 24 you can see and compare some previous results like how many links have been indexed, how many total links have been found, words count, etc.

Tests, implemented with ‘Margent’ search agent:

A successful test with the Margent search agent you can see for example on Figure 28 and Figure 29. The test is processed with the word ‘live’.

An example of unsuccessful search with the word ’non-existing-word’ is shown on Figure 30.

1. ***Conclusion and recommendation***

Here are some recommendations for the SE:

* Add more ‘stop’ words, because a lot of of ‘parasite’ words are indexed now;
* Split tables – Split table Words to different tables for words with numbers, words with non-Latin letters; Split Files table to different tables for every indexed file type;
* Update tables – do not truncate tables on every start of the crawler, but only update the data. For now only Words table is updated;
* Develop SqlBulkCopy for copy from one DB to another
* Improve indexing of .pdf and .mp3 files; also add indexing of .doc, .pps, .xls, other file types

Recommendations for ‘Margent’ web crawler:

* Create better page ranking – with number of links on the page and some other criteria;
* Add options to choose how many results to be fetched from the DB, and how many to be displayed per page;

**Conclusion .. ??**

1. ***References***
2. <http://impact.htw-berlin.de/>
3. <http://www.webproguide.com/seo-articles-index/What-is-a-Search-Engine/index.php>
4. <http://en.wikipedia.org/wiki/List_of_search_engines>
5. <http://www.webproguide.com/seo-articles-index/The-Search-Engines-in-Detail/>
6. <http://www.google.com/press/funfacts.html>
7. <http://deanhunt.com/13-amazing-google-facts-you-dont-know/>
8. <http://royal.pingdom.com/2010/02/24/google-facts-and-figures-massive-infographic/>

1. <http://www.webproguide.com/articles/Yahoo-Content-Acquisition-Program/>

1. [http://en.wikipedia.org/wiki/Stemming](http://en.wikipedia.org/wiki/Stemming - 4.5.2)

1. <http://tartarus.org/~martin/PorterStemmer/>

1. <http://www.google.com/url?sa=t&source=web&cd=1&ved=0CBQQFjAA&url=http%3A%2F%2Fdownload.microsoft.com%2Fdownload%2Ff%2F3%2F2%2Ff32ff4c6-174f-4a2f-a58f-ed28437d7b1e%2FIntroducing_NET_Framework_35_v1.doc&ei=AKIbTI3HEMSPOJm09JEK&usg=AFQjCNHmRrlEPWuQa2ehJDL8zRoexv2uTg&sig2=W2KnRn3SkhwjWMqne9iJOQ>

1. <http://en.wikipedia.org/wiki/Microsoft_Visual_Studio>

1. <http://en.wikipedia.org/wiki/Microsoft_SQL_Server>
2. <http://msdn.microsoft.com/library/z1zx9t92%28VS.100%29.aspx>
3. <http://weblogs.asp.net/scottgu/archive/2007/04/08/new-orcas-language-feature-lambda-expressions.aspx>
4. <http://weblogs.asp.net/scottgu/archive/2007/05/15/new-orcas-language-feature-anonymous-types.aspx?CommentPosted=true>
5. <http://weblogs.asp.net/scottgu/archive/2007/04/21/new-orcas-language-feature-query-syntax.aspx>
6. <http://www.lhotka.net/cslanet/>
7. <http://tortoisesvn.tigris.org/>
8. <http://en.wikipedia.org/wiki/Revision_control>
9. <http://www.softfolder.com/easy_smtp_server.html>
10. ***Appendix***

Source code

Here are some ‘slices’ the whole source code, because the full volume is about 9000 lines or about 200 pages. I have tried to sift out the more significant classes and files.

The whole source code could be found on the accompanied CD.

* Source code of the Web Crawler

Most interesting classes here, that one should survey are: CrawlerManager.cs, Spider.cs, HTMLDocument.cs, Document.cs, DocumentFactory.cs, SeedList.cs, MMWebCrawler.config.

DBCopier.cs

using System;

using System.Data;

using System.Data.SqlClient;

namespace MMarinov.WebCrawler.Library

{

public static class DBCopier

{

public static void TruncateDBTables()

{

SqlConnection cn = new SqlConnection(Preferences.ConnectionString);

SqlCommand cm = new SqlCommand();

cn.Open();

try

{

cm.Connection = cn;

cm.CommandType = CommandType.StoredProcedure;

cm.CommandText = "sp\_TruncateTables";

cm.ExecuteNonQuery();

}

finally

{

cn.Close();

}

}

private static void TruncateActiveDBTables()

{

using (DALWebCrawlerActive.WebCrawlerActiveDataContext dataContext = new DALWebCrawlerActive.WebCrawlerActiveDataContext(Preferences.ConnectionStringActive))

{

dataContext.sp\_TruncateAllTables();

}

}

public static void CopyDBToActiveDB()

{

TruncateActiveDBTables();

CopyFromDBToActiveDB();

}

private static void CopyFromDBToActiveDB()

{

SqlConnection cn = new SqlConnection(Preferences.ConnectionString);

SqlCommand cm = new SqlCommand();

SqlTransaction tr;

cn.Open();

try

{

tr = cn.BeginTransaction(IsolationLevel.Serializable);

try

{

cm.Transaction = tr;

cm.Connection = cn;

cm.CommandType = CommandType.StoredProcedure;

cm.CommandText = "sp\_CopyFromDBToActiveDB";

cm.ExecuteNonQuery();

tr.Commit();

}

catch (Exception ex)

{

tr.Rollback();//important here

throw (ex);

}

}

finally

{

cn.Close();

}

}

}

**Word**.cs

using System;

using System.Data;

using System.Data.SqlClient;

using CSLA;

using CSLA.Data;

namespace MMarinov.WebCrawler.Library

{

public class Word : CSLA.BusinessBase

{

#region Class Level Private Variables

private long \_id = 0; //\*\*PK

private string \_wordName = "";

#endregion //Class Level Private Variables

#region Constructors

private Word()

{

MarkAsChild();

}

#endregion //Constructors

#region Business Properties and Methods

public long ID

{

get { return \_id; }

}

public string WordName

{

get { return \_wordName; }

set

{

if (value != \_wordName)

{

\_wordName = value;

MarkDirty();

}

}

}

public bool IsSaveable

{

//Since you cannot bind a control to multiple properties you need to create a property that combines the ones you need

//In this case, bind the UI Save button Enabled property to IsSaveable. (Why save an object that has not changed?)

get

{

return IsValid && IsDirty;

}

}

#endregion //Business Properties and Methods

#region System.Object Overrides

public override string ToString()

{

return "Word" + "/" + \_id.ToString();

}

public bool Equals(Word file)

{

return \_id.Equals(file.ID);

}

public override int GetHashCode()

{

return ("Word" + "/" + \_id.ToString()).GetHashCode();

}

#endregion //System.Object Overrides

#region Criteria (identifies the Individual Object/ Primary Key)

[Serializable]

private class Criteria

{

public int ID = 0;

public Criteria()

{

}

public Criteria(int id)

{

this.ID = id;

}

}

#endregion //Criteria

#region Static Methods

public static Word NewWord()

{

return (Word)DataPortal.Create(new Criteria());

}

public static Word NewWord(string wordName)

{

Word w = (Word)DataPortal.Create(new Criteria());

w.WordName = wordName;

return w;

}

public static Word FetchWord(SafeDataReader dr)

{

// Load an Existing Object from Data Reader

Word child = NewWord();

child.Fetch(dr);

return child;

}

/// <summary>

/// Called by DataPortal to load data from the database

/// </summary>

/// <param name="dr"></param>

public void Fetch(SafeDataReader dr)

{

// Retrieve the data from the passed in data reader,

// which may or may not have a transaction associated with it

\_id = dr.GetInt64("ID");

\_wordName = dr.GetString("WordName");

MarkOld();

}

#endregion //Static Methods

#region Data Access

//Called by DataPortal so that we can set defaults as needed

protected override void DataPortal\_Create(object criteria)

{

}

internal void Update(SqlTransaction tr)

{

if (!IsDirty)

{

return;

}

// save data into db

SqlConnection cn = tr.Connection;

SqlCommand cm = new SqlCommand();

try

{

cm.Connection = cn;

cm.Transaction = tr;

cm.CommandType = CommandType.StoredProcedure;

// is not deleted object, check if this is an update or insert

if (this.IsNew)

{

//perform an insert, object has not been persisted

cm.CommandText = @"sp\_InsertWord";

}

else

{

//check

}

cm.Parameters.AddWithValue("@WordName", \_wordName);

\_id = Convert.ToInt32(cm.ExecuteScalar());

// mark the object as old (persisted)

MarkOld();

}

catch (Exception ex)

{

throw (ex);

}

}

protected override void DataPortal\_Update()

{

// save data into db

SqlConnection cn = new SqlConnection(DB("WebCrawler"));

SqlCommand cm = new SqlCommand();

SqlTransaction tr;

cn.Open();

try

{

tr = cn.BeginTransaction(IsolationLevel.Serializable);

try

{

cm.Connection = cn;

cm.Transaction = tr;

cm.CommandType = CommandType.StoredProcedure;

// is not deleted object, check if this is an update or insert

if (this.IsNew)

{

//perform an insert, object has not been persisted

cm.CommandText = @"sp\_InsertWord";

}

else

{

//check

}

cm.Parameters.AddWithValue("@WordName", \_wordName);

cm.Parameters.AddWithValue("@ID", \_id).Direction = ParameterDirection.Output;

\_id = Convert.ToInt32(cm.ExecuteScalar());

// mark the object as old (persisted)

MarkOld();

tr.Commit();

}

catch (Exception ex)

{

tr.Rollback();

throw (ex);

}

}

finally

{

cn.Close();

}

}

#endregion //Data Access

internal void SaveOne()

{

DataPortal\_Update();

}

}

}

WordCollection.cs

using System;

using System.Data;

using System.Data.SqlClient;

using CSLA;

using CSLA.Data;

namespace MMarinov.WebCrawler.Library

{

[Serializable]

public class WordCollection : CSLA.BusinessCollectionBase

{

#region Business Properties and Methods

public Word this[int index]

{

get { return (Word)List[index]; }

}

public void Add(Word item)

{

if (!Contains(item))

{

List.Add(item);

}

else

throw new Exception("Word '" + item.ToString() + "' already exist.");

}

/// <summary>

/// Adds a new word if not exists

/// </summary>

/// <param name="wordName"></param>

/// <returns></returns>

public Word AddWord(string wordName)

{

Word w = GetWord(wordName);

if (w == null)

{

w = Word.NewWord(wordName);

w.SaveOne();

List.Add(w);

}

return w;

}

public Word GetWord(string wordName)

{

foreach (Word child in List)

{

if (child == null)

{

}

if (child.WordName.Equals(wordName))

return child;

}

return null;

}

protected override object OnAddNew()

{

Word project\_det = Word.NewWord();

InnerList.Add(project\_det);

return project\_det;

}

public bool IsSaveable

{

//Since you cannot bind a control to multiple properties you need to create a property that combines the ones you need

//In this case, bind the UI Save button Enabled property to IsSaveable. (Why save an object that has not changed?)

get { return IsValid && IsDirty; }

}

#endregion //Business Properties and Methods

#region Contains

public bool Contains(Word item)

{

return List.Contains(item);

}

public bool Contains(string wordName)

{

foreach (Word child in List)

{

if (child.WordName.Equals(wordName))

return true;

}

return false;

}

public bool Contains(int id)

{

foreach (Word child in List)

{

if (child.ID.Equals(id))

return true;

}

return false;

}

#endregion //Contains

#region Constructor

private WordCollection()

{

//prevent direct creation

AllowSort = true;

AllowFind = true;

AllowNew = true;

}

#endregion //Constructor

#region Static Methods

public static WordCollection NewWordCollection()

{

return (WordCollection)DataPortal.Create(new Criteria());

}

public static WordCollection GetWordCollection()

{

return (WordCollection)DataPortal.Fetch(new Criteria());

}

#endregion //Static Methods

#region Criteria (identifies the Individual Object/ Primary Key)

[Serializable]

private class Criteria

{

public int ID = 0;

public Criteria()

{

}

public Criteria(int id)

{

this.ID = id;

}

}

#endregion //Criteria

#region Data Access

//Called by DataPortal so that we can set defaults as needed

protected override void DataPortal\_Create(object criteria)

{

}

//Called by DataPortal to load data from the database

protected override void DataPortal\_Fetch(object criteria)

{

//retrieve data from database

SqlConnection cn = new SqlConnection(DB("WebCrawler"));

SqlCommand cm = new SqlCommand();

SqlTransaction tr;

cn.Open();

try

{

tr = cn.BeginTransaction(IsolationLevel.ReadCommitted);

try

{

cm.Connection = cn;

cm.Transaction = tr;

cm.CommandType = CommandType.StoredProcedure;

cm.CommandText = @"sp\_SelectWordsAll";

SafeDataReader dr = new SafeDataReader(cm.ExecuteReader());

try

{

while (dr.Read())

{

List.Add(Word.FetchWord(dr));

}

}

finally

{

dr.Close();

}

tr.Commit();

}

catch (Exception ex)

{

tr.Rollback();

throw (ex);

}

}

catch (Exception ex)

{

throw (ex);

}

finally

{

cn.Close();

}

}

protected override void DataPortal\_Update()

{

// save data into db

SqlConnection cn = new SqlConnection(DB("WebCrawler"));

SqlCommand cm = new SqlCommand();

SqlTransaction tr;

cn.Open();

try

{

tr = cn.BeginTransaction(IsolationLevel.Serializable);

try

{

// loop through each non-deleted child object and call its Update() method

foreach (Word child in List)

child.Update(tr);

tr.Commit();

}

catch (Exception ex)

{

tr.Rollback();

throw (ex);

}

}

finally

{

cn.Close();

}

}

#endregion //Data Access

internal void AddRange(WordCollection newWordsColl)

{

InnerList.AddRange(newWordsColl);

}

}

}

Document.cs

using System;

namespace MMarinov.WebCrawler.Indexer

{

public abstract class Document

{

public enum DocumentTypes

{

HTML = 1,

Text = 2,

Mp3 = 3,

PDF = 4

}

private Uri \_Uri;

private string \_allCode;

private string \_ContentType;

private string \_MimeType = "";

private string \_Title;

private string \_Description = "";

private byte \_fileType =0;

private string \_keywords ="";

public static Int64 FoundValidLinks = 0;

public static Int64 FoundTotalLinks = 0;

public static double DownloadSpeed = 0;

protected void SetDownloadSpeed(double speed)

{

DownloadSpeed = speed;

}

private System.Collections.Generic.List<string> \_localLinks;

private System.Collections.Generic.List<string> \_externalLinks;

public abstract bool GetResponse(System.Net.HttpWebResponse webresponse);

public abstract void Parse();

public delegate void DocumentProgressEventHandler(Report.ProgressEventArgs pea);

public event DocumentProgressEventHandler DocumentEvent;

protected void DocumentProgressEvent(Report.ProgressEventArgs pea)

{

if (this.DocumentEvent != null)

{

DocumentEvent(pea);

}

}

public System.Collections.Generic.List<string> LocalLinks

{

get

{

return \_localLinks;

}

set

{

\_localLinks = value;

}

}

public System.Collections.Generic.List<string> ExternalLinks

{

get

{

return \_externalLinks;

}

set

{

\_externalLinks = value;

}

}

public virtual string AllCode

{

get { return \_allCode; }

set { \_allCode = value; }

}

public virtual string ContentType

{

get { return \_ContentType; }

set { \_ContentType = value; }

}

public virtual string MimeType

{

get { return \_MimeType; }

set { \_MimeType = value; }

}

public virtual string Keywords

{

get { return \_keywords; }

set { \_keywords = value; }

}

public virtual byte FileType

{

get { return \_fileType; }

set { \_fileType = value; }

}

public abstract string WordsOnly { get; }

public virtual string Title

{

get { return \_Title; }

set { \_Title = value; }

}

public virtual string Description

{

get { return \_Description; }

set { \_Description = value; }

}

/// <summary>

/// http://www.ietf.org/rfc/rfc2396.txt

/// </summary>

public virtual Uri Uri

{

get { return \_Uri; }

set { \_Uri = value; }

}

public virtual string[] WordsArray

{

get { return this.WordsStringToArray(WordsOnly); }

}

/// <summary>

/// Most document types don't have embedded robot information

/// so they'll always be allowed to be followed

/// (assuming there are links to follow)

/// </summary>

public virtual bool RobotFollowOK

{

get { return true; }

}

/// <summary>

/// Most document types don't have embedded robot information

/// so they'll always be allowed to be indexed

/// (assuming there is content to index)

/// </summary>

public virtual bool RobotIndexOK

{

get { return true; }

}

/// <summary>

/// Constructor for any document requires the Uri be specified

/// </summary>

public Document(Uri uri)

{

\_Uri = uri;

}

/// <summary>

/// Constructor for any document requires the Uri and MimeType to be specified

/// </summary>

public Document(Uri uri, string mimeType)

{

\_Uri = uri;

\_MimeType = mimeType;

}

/// <summary>

/// COMPRESS ALL WHITESPACE into a single space, seperating words

/// </summary>

/// <param name="words"></param>

/// <returns></returns>

protected string[] WordsStringToArray(string words)

{

if (words.Length > 0)

{

return System.Text.RegularExpressions.Regex.Replace(words, Common.MatchEmptySpacesPattern, " ").Split(Common.Separators, StringSplitOptions.RemoveEmptyEntries);

}

else

{

return new string[0];

}

}

/// <summary>

/// Is the value of the href pointing to a web page?

/// </summary>

/// <param name="foundHref">The value of the href that needs to be interogated.</param>

/// <returns>Boolen </returns>

public static bool IsAWebPage(string foundHref)

{

if (foundHref.Length < 2 || foundHref.IndexOf("javascript:") > -1 || foundHref.IndexOf("mailto:") > -1 || foundHref.StartsWith("#") || foundHref.StartsWith("file://") || foundHref.StartsWith(@"\\") || foundHref.StartsWith("ftp://"))

{

return false;

}

if (!Uri.IsWellFormedUriString(foundHref, UriKind.RelativeOrAbsolute))

{

return false;

}

string extension = "";

try

{

Uri uri = new Uri(foundHref);

if (uri.Segments.Length == 0)

{

}

string lastSegment = uri.Segments[uri.Segments.Length - 1];

if (lastSegment.Contains("."))

{

extension = lastSegment.Substring(lastSegment.LastIndexOf(".") + 1, lastSegment.Length - lastSegment.LastIndexOf(".") - 1);

}

else

{

return true;

}

}

catch//relative url

{

if (foundHref.Contains("."))

{

extension = foundHref.Substring(foundHref.LastIndexOf(".") + 1, foundHref.Length - foundHref.LastIndexOf("..") - 1);

}

else

{

return true;

}

}

switch (extension)

{

case "htm":

case "html":

case "shtml":

case "dhtml":

case "xhtml":

case "asp":

case "aspx":

case "cgi":

case "php":

case "jsf":

case "jsp":

case "pl":

case "txt":

return true;

case "mp3":

case "pdf":

return !Preferences.IndexOnlyHTMLDocuments;

default:

return false;

}

}

/// <summary>

/// False if the link exists, so the website must be skipped

/// </summary>

/// <param name="uri"></param>

/// <returns>False if the link exists</returns>

protected bool AddURLtoGlobalVisited(Uri uri)

{

lock (Spider.GlobalVisitedURLs)

{

string link = Common.GetHttpAuthority(uri);

if (!Spider.GlobalVisitedURLs.Contains(link))

{

Spider.GlobalVisitedURLs.Add(link);

DocumentProgressEvent(new Report.ProgressEventArgs(Report.EventTypes.Start, "Crawling website(redirected):" + link));

return true;

}

else

{

return false;

}

}

}

protected bool DeleteFile(string filename)

{

// delete file

try

{

new System.IO.FileInfo(filename).Delete();

}

catch (Exception e)

{

Report.Logger.ErrorLog(e);

return false;

}

return true;

}

}

}

**DocumentFactory.cs**

using System;

namespace MMarinov.WebCrawler.Indexer

{

public static class DocumentFactory

{

public static Document New(Uri uri, System.Net.HttpWebResponse contentType)

{

Document newDoc = null;

string mimeType = ParseMimeType(contentType.ContentType.ToString()).ToLower();

string encoding = ParseEncoding(contentType.ToString()).ToLower();

switch (mimeType)

{

case "text/css":

case "text/xml":

case "application/x-msdownload":

case "application/octet-stream":

case "application/xml":

case "application/rss+xml":

case "application/rdf+xml":

case "application/atom+xml":

case "application/xhtml+xml":

break;

case "application/vnd.ms-powerpoint":

case "application/msword":

//TODO: parse !

break;

case "application/pdf":

if (!Preferences.IndexOnlyHTMLDocuments)

{

newDoc = new PdfDocument(uri);

}

break;

case "text/plain":

newDoc = new TextDocument(uri);

break;

case "audio/mpeg":

if (!Preferences.IndexOnlyHTMLDocuments)

{

newDoc = new Mp3Document(uri);

}

break;

default://case "text/html":

//newDoc = new HtmlDocument(uri);

if (mimeType.IndexOf("text") > -1)

{ // If we got 'text' data (not images)

newDoc = new HtmlDocument(uri, mimeType);

}

break;

} // switch

return newDoc;

}

private static string ParseMimeType(string contentType)

{

string mimeType = "";

string[] contentTypeArray = contentType.Split(';');

// Set MimeType if it's blank

if (mimeType == "" && contentTypeArray.Length >= 1)

{

mimeType = contentTypeArray[0];

}

return mimeType;

}

private static string ParseEncoding(string contentType)

{

string encoding = "";

string[] contentTypeArray = contentType.Split(';');

// Set Encoding if it's blank

if (encoding == "" && contentTypeArray.Length >= 2)

{

int charsetpos = contentTypeArray[1].IndexOf("charset");

if (charsetpos > 0)

{

encoding = contentTypeArray[1].Substring(charsetpos + 8, contentTypeArray[1].Length - charsetpos - 8);

}

}

return encoding;

}

}

}

**HtmlDocument.cs**

using System;

using System.Text.RegularExpressions;

namespace MMarinov.WebCrawler.Indexer

{

/// <summary>

/// Storage for parsed HTML data returned by ParsedHtmlData();

/// </summary>

/// <remarks>

/// Arbitrary class to encapsulate just the properties we need

/// to index Html pages (Title, Meta tags, Keywords, etc).

/// </remarks>

public class HtmlDocument : Document

{

#region Private fields: \_Uri, \_ContentType, \_RobotIndexOK, \_RobotFollowOK

private string \_htmlCode = "";

private String \_ContentType;

private bool \_RobotIndexOK = true;

private bool \_RobotFollowOK = true;

private string \_WordsOnly = "";

/// <summary>MimeType so we know whether to try and parse the contents, eg. "text/html", "text/plain", etc</summary>

private string \_MimeType = "";

/// <summary>Html &lt;title&gt; tag</summary>

private String \_Title = "";

/// <summary>Html &lt;meta http-equiv='description'&gt; tag</summary>

private string \_Description = "";

private string \_keywords = "";

private MMarinov.WebCrawler.Stemming.Languages \_language = MMarinov.WebCrawler.Stemming.Languages.None;

private System.Collections.Generic.List<string> linksLocal = new System.Collections.Generic.List<string>();

private System.Collections.Generic.List<string> linksExternal = new System.Collections.Generic.List<string>();

#endregion

#region Constructor requires Uri

public HtmlDocument(Uri location)

: base(location)

{

this.Uri = location;

}

public HtmlDocument(Uri location, string mimeType)

: base(location, mimeType)

{

this.Uri = location;

\_MimeType = mimeType;

}

#endregion

#region Public Properties: Uri, RobotIndexOK

/// <summary>

/// Whether a robot should index the text

/// found on this page, or just ignore it

/// </summary>

/// <remarks>

/// Set when page META tags are parsed - no 'set' property

/// More info:

/// http://www.robotstxt.org/

/// </remarks>

public override bool RobotIndexOK

{

get { return \_RobotIndexOK; }

}

/// <summary>

/// Whether a robot should follow any links

/// found on this page, or just ignore them

/// </summary>

/// <remarks>

/// Set when page META tags are parsed - no 'set' property

/// More info:

/// http://www.robotstxt.org/

/// </remarks>

public override bool RobotFollowOK

{

get { return \_RobotFollowOK; }

}

public override string ContentType

{

get

{

return \_ContentType;

}

set

{

\_ContentType = value.ToString();

string[] contentTypeArray = \_ContentType.Split(';');

// Set MimeType if it's blank

if (\_MimeType == "" && contentTypeArray.Length >= 1)

{

\_MimeType = contentTypeArray[0];

}

// Set Encoding if it's blank

if (Encoding == null && contentTypeArray.Length >= 2)

{

int charsetpos = contentTypeArray[1].IndexOf("charset");

if (charsetpos > 0)

{

Encoding = System.Text.Encoding.GetEncoding(contentTypeArray[1].Substring(charsetpos + 8, contentTypeArray[1].Length - charsetpos - 8));

}

}

}

}

public MMarinov.WebCrawler.Stemming.Languages Language

{

get { return \_language; }

}

#endregion

#region Public fields: Encoding, Keywords, All

/// <summary>

/// Encoding eg. "utf-8", "Shift\_JIS", "iso-8859-1", "gb2312", etc

/// </summary>

public System.Text.Encoding Encoding;

/// <summary>

/// Html &lt;meta http-equiv='keywords'&gt; tag

/// </summary>

public override string Keywords

{

get

{

return \_keywords;

}

set

{

\_keywords = value.Substring(0, 500);

}

}

public override byte FileType

{

get

{

return (byte)DocumentTypes.HTML;

}

}

public override string Title

{

get

{

return \_Title;

}

set

{

\_Title = value;

}

}

/// <summary>

/// Raw content of page, as downloaded from the server

/// Html stripped to make up the 'wordsonly'

/// </summary>

public override string AllCode

{

get { return \_htmlCode; }

set

{

\_htmlCode = value;

\_WordsOnly = StripHtml(\_htmlCode);

}

}

public override string WordsOnly

{

get { return \_Title + " " + this.\_keywords + " " + this.\_Description + " " + Common.GetAuthority(Uri)+" " + this.\_WordsOnly; }

}

public override string Description

{

get

{

return \_Description;

}

set

{

\_Description = value.Substring(0, 500);

}

}

#endregion

/// <summary>

/// Pass in a ROBOTS meta tag found while parsing,

/// and set HtmlDocument property/ies appropriately

/// </summary>

/// <remarks>

/// More info:

/// \* Robots Exclusion Protocol \*

/// - for META tags http://www.robotstxt.org/wc/meta-user.html

/// - for ROBOTS.TXT in the siteroot http://www.robotstxt.org/wc/norobots.html

/// </remarks>

public void SetRobotDirective(string robotMetaContent)

{

robotMetaContent = robotMetaContent.ToLower();

if (robotMetaContent.IndexOf("none") >= 0)

{

// 'none' means you can't Index or Follow!

\_RobotIndexOK = false;

\_RobotFollowOK = false;

}

else

{

if (robotMetaContent.IndexOf("noindex") >= 0) { \_RobotIndexOK = false; }

if (robotMetaContent.IndexOf("nofollow") >= 0) { \_RobotFollowOK = false; }

}

}

#region Parsing

/// <summary>

///

/// </summary>

/// <remarks> Regex on this blog will parse ALL attributes from within tags...

/// IMPORTANT when they're out of order, spaced out or over multiple lines

/// http://blogs.worldnomads.com.au/matthewb/archive/2003/10/24/158.aspx

/// http://blogs.worldnomads.com.au/matthewb/archive/2004/04/06/215.aspx

/// </remarks>

public override void Parse()

{

if (string.IsNullOrEmpty(this.\_Title))

{

this.\_Title = Regex.Match(\_htmlCode, @"(?<=<title[^\>]\*>).\*?(?=</title>)", RegexOptions.IgnoreCase | RegexOptions.ExplicitCapture).Value;

}

ParseLanguage();

ParseMetaTags();

ParseLinks();

this.LocalLinks = linksLocal;

this.ExternalLinks = linksExternal;

}// Parse

private void ParseLinks()

{

// see the full version on the CD

}// Parse Links

/// <summary>

/// Gets the content of meta tags and set keywords, description and robot directives

/// </summary>

private void ParseMetaTags()

{

// see the full version on the CD

}//Parse MetaTags

/// <summary>

/// Parse HTML tag to look for lang or xml:lang tag.

/// </summary>

private void ParseLanguage()

{

Match htmlMatch = Regex.Match(\_htmlCode, @"<html\b(?>\s+(?:alt=""([^""]\*)""|lang=""([^""]\*)""|xml:lang=""([^""]\*)"")|[^\s>]+|\s+)\*>", RegexOptions.IgnoreCase | RegexOptions.ExplicitCapture);

// Loop through the attribute/value pairs inside the tag

foreach (Match submetamatch in Regex.Matches(htmlMatch.Value, @"(?<name>\b(\w|-)+\b)\s\*=\s\*(""(?<value>[^""]\*)""|'(?<value>[^']\*)'|(?<value>[^""'<> ]+)\s\*)+", RegexOptions.IgnoreCase | RegexOptions.ExplicitCapture))

{

if (submetamatch.Groups[1].Value.ToLower() == "lang")

{

switch (submetamatch.Groups[2].Value.ToLower())

{

case "en":

\_language = Stemming.Languages.English;

break;

case "de":

\_language = Stemming.Languages.German;

break;

case "bg":

\_language = Stemming.Languages.Bulgarian;

break;

default: break;

}

}

else if (submetamatch.Groups[1].Value.ToLower() == "xml:lang" && \_language == Stemming.Languages.None)

{

switch (submetamatch.Groups[2].Value.ToLower())

{

case "en":

\_language = Stemming.Languages.English;

break;

case "de":

\_language = Stemming.Languages.German;

break;

case "bg":

\_language = Stemming.Languages.Bulgarian;

break;

default: break;

}

}

}

}// ParseLanguage

/// <summary>

/// Checks link and adds it to external/local links collection

/// </summary>

/// <param name="link"></param>

private void AddLinkToCollection(string link)

{

// some checks here

// see the full version on the CD

}

#endregion

public override bool GetResponse(System.Net.HttpWebResponse webResponse)

{

if (webResponse.ContentEncoding != "")

{

// Use the HttpHeader Content-Type in preference to the one set in META

this.Encoding = System.Text.Encoding.GetEncoding(webResponse.ContentEncoding);

}

else if (this.Encoding == null)

{

this.Encoding = System.Text.Encoding.UTF8; // default

}

System.IO.StreamReader stream = null;

try

{

DateTime startDLtime = DateTime.Now;

stream = new System.IO.StreamReader(webResponse.GetResponseStream(), this.Encoding);

if (webResponse.ResponseUri != this.Uri)

{

this.Uri = webResponse.ResponseUri; // we \*may\* have been redirected... and we want the \*final\* URL

if (!base.AddURLtoGlobalVisited(this.Uri))

{

return false;

}

}

this.AllCode = stream.ReadToEnd();

if (this.AllCode != "")

{

base.SetDownloadSpeed(this.AllCode.Length / (DateTime.Now - startDLtime).TotalSeconds / 1000);

}

}

catch (Exception e)

{

this.DocumentProgressEvent(new Report.ProgressEventArgs(new Exception(Uri.AbsoluteUri, e)));

return false;

}

finally

{

if (stream != null)

{

stream.Close();

stream.Dispose();

}

}

return true; //success

}//GetResponse

/// <summary>

/// Stripping HTML

/// http://www.4guysfromrolla.com/webtech/042501-1.shtml

/// </summary>

/// <remarks>

/// Using regex to find tags without a trailing slash

/// http://concepts.waetech.com/unclosed\_tags/index.cfm

///

/// Replace html comment tags

/// http://www.faqts.com/knowledge\_base/view.phtml/aid/21761/fid/53

/// </remarks>

protected string StripHtml(string htmlCode)

{

const string matchCommentPattern = @"(\<![ \r\n\t]\*(--([^\-]|[\r\n]|-[^\-])\*--[ \r\n\t]\*)\>)";

htmlCode = Regex.Replace(htmlCode, "&amp;", "&", RegexOptions.IgnoreCase | RegexOptions.Singleline);

//Strips the <script> and <noscript> tags, comments <!-- --> , <style>

htmlCode = Regex.Replace(htmlCode, MatchTag("script") + "|" + MatchTag("noscript") + "|" + matchCommentPattern + "|" + MatchTag("style"), " ", RegexOptions.Singleline | RegexOptions.IgnoreCase);

htmlCode = ISOtoASCII(htmlCode);

//removes tags, new lines and multiple spaces

htmlCode = Regex.Replace(htmlCode, "<(.|\n)\*?>", " ");

// new lines and multiple spaces

htmlCode = Regex.Replace(htmlCode, "(&(.|\n)+?;)|(\r?\n?)", "");

htmlCode = Regex.Replace(htmlCode, Common.MatchEmptySpacesPattern, " ");

return htmlCode;

}//stripHtml

private string MatchTag(string tagName)

{

return @"(\<[ \r\n\t]\*" + tagName + @"([ \r\n\t\>]|\>){1,}([ \r\n\t]|.)\*</[ \r\n\t]\*" + tagName + @"[ \r\n\t]\*\>)";

}

}

}

**PdfDocument.cs**

using System;

namespace MMarinov.WebCrawler.Indexer

{

public class PdfDocument : Document

{

private string \_All;

private string \_WordsOnly;

public PdfDocument(Uri location)

: base(location)

{ }

/// <summary>

/// Set 'all' and 'words only' to the same value (no parsing)

/// </summary>

public override string AllCode

{

get { return \_All; }

set

{

\_All = value;

\_WordsOnly = \_All;

}

}

public override string WordsOnly

{

get { return \_WordsOnly; }

}

public override string[] WordsArray

{

get { return base.WordsStringToArray(WordsOnly); }

}

/// <summary>

///

/// </summary>

public override void Parse()

{

System.Text.RegularExpressions.MatchCollection matchLinks = System.Text.RegularExpressions.Regex.Matches(\_All, @"http(s)?://([\w+?\.\w+])+([a-zA-Z0-9\~\!\@\#\$\%\^\&amp;\\*\(\)\_\-\=\+\\\/\?\.\:\;\'\,]\*)?", System.Text.RegularExpressions.RegexOptions.IgnoreCase);

System.Text.RegularExpressions.MatchCollection matchLinks2 = System.Text.RegularExpressions.Regex.Matches(\_All, "(http|https)://([a-zA-Z0-9\\~\\!\\@\\#\\$\\%\\^\\&amp;\\\*\\(\\)\_\\-\\=\\+\\\\\\/\\?\\.\\:\\;\\'\\,]\*)?", System.Text.RegularExpressions.RegexOptions.IgnoreCase);

}

public override bool GetResponse(System.Net.HttpWebResponse webResponse)

{

string filename = System.IO.Path.Combine(Preferences.TempPath, (System.IO.Path.GetFileName(this.Uri.LocalPath)));

this.Title = System.IO.Path.GetFileNameWithoutExtension(filename);

System.IO.BinaryReader binaryReader = null;

System.IO.FileStream iofilestream = null;

try

{

binaryReader = new System.IO.BinaryReader(webResponse.GetResponseStream());

iofilestream = new System.IO.FileStream(filename, System.IO.FileMode.Create);

const int BUFFER\_SIZE = 8192;

byte[] buf = new byte[BUFFER\_SIZE];

int n = binaryReader.Read(buf, 0, BUFFER\_SIZE);

while (n > 0)

{

iofilestream.Write(buf, 0, n);

n = binaryReader.Read(buf, 0, BUFFER\_SIZE);

}

if (webResponse.ResponseUri != this.Uri)

{

this.Uri = webResponse.ResponseUri; // we \*may\* have been redirected... and we want the \*final\* URL

base.AddURLtoGlobalVisited(this.Uri);

}

\_All = ParsePDF(filename);

}

catch (Exception e)

{

base.DocumentProgressEvent(new Report.ProgressEventArgs(new Exception(Uri.AbsoluteUri, e)));

return false;

}

finally

{

if (binaryReader != null)

{

binaryReader.Close();

}

if (iofilestream != null)

{

iofilestream.Close();

iofilestream.Dispose();

}

}

return !string.IsNullOrEmpty(this.AllCode);

}

private static string ParsePDF(string input)

{

org.pdfbox.pdmodel.PDDocument doc = org.pdfbox.pdmodel.PDDocument.load(input);

return new org.pdfbox.util.PDFTextStripper().getText(doc);

}

}

}

**Mp3Document.cs**

using System;

namespace MMarinov.WebCrawler.Indexer

{

public class Mp3Document : Document

{

private string \_wordsOnly = "";

private string mp3Title = "";

private string mp3Artist = "";

private string mp3Album = "";

private string mp3Comments = "";

public string Mp3Title

{

get { return mp3Title; }

}

public string Mp3Artist

{

get { return mp3Artist; }

}

public string Mp3Album

{

get { return mp3Album; }

}

public string Mp3Comments

{

get { return mp3Comments; }

}

public Mp3Document(Uri location)

: base(location)

{

}

public override bool GetResponse(System.Net.HttpWebResponse webResponse)

{

string filename = System.IO.Path.Combine(Preferences.TempPath, (System.IO.Path.GetFileName(this.Uri.LocalPath)));

System.IO.BinaryReader binaryReader = null;

System.IO.FileStream iofilestream = null;

try

{

binaryReader = new System.IO.BinaryReader(webResponse.GetResponseStream());

iofilestream = new System.IO.FileStream(filename, System.IO.FileMode.Create);

const int BUFFER\_SIZE = 8192;

byte[] buf = new byte[BUFFER\_SIZE];

int n = binaryReader.Read(buf, 0, BUFFER\_SIZE);

while (n > 0)

{

iofilestream.Write(buf, 0, n);

n = binaryReader.Read(buf, 0, BUFFER\_SIZE);

}

if (webResponse.ResponseUri != this.Uri)

{

this.Uri = webResponse.ResponseUri; // we \*may\* have been redirected... and we want the \*final\* URL

base.AddURLtoGlobalVisited(this.Uri);

}

}

catch (Exception e)

{

base.DocumentProgressEvent(new Report.ProgressEventArgs(new Exception(Uri.AbsoluteUri, e)));

return false;

}

finally

{

if (binaryReader != null)

{

binaryReader.Close();

}

if (iofilestream != null)

{

iofilestream.Close();

iofilestream.Dispose();

}

}

HundredMilesSoftware.UltraID3Lib.UltraID3 mp3 = new HundredMilesSoftware.UltraID3Lib.UltraID3();

try

{

mp3.Read(Uri.AbsoluteUri);

mp3Title = mp3.Title;

mp3Artist = mp3.Artist;

mp3Album = mp3.Album;

mp3Comments = mp3.Comments;

this.Title = (mp3Title + " " + mp3Artist).Trim();

this.Description = mp3Album + " " + mp3Comments;

return !string.IsNullOrEmpty(this.Title);

}

catch (Exception e)

{

base.DocumentProgressEvent(new Report.ProgressEventArgs(e));

return false;

}

}

public override void Parse()

{

System.Text.StringBuilder strBldr = new System.Text.StringBuilder();

Array.ForEach<string>(

base.WordsStringToArray(mp3Title + " " + mp3Artist + " " + mp3Album + " " + mp3Comments),

word => strBldr.Append(word).Append(" "));

\_wordsOnly = strBldr.ToString();

}

public override string WordsOnly

{

get

{

return \_wordsOnly;

}

}

}

}

CrawlingManager.cs

using System;

namespace MMarinov.WebCrawler.Indexer

{

/// <summary>

/// Summary description for CrawlingManager

/// </summary>

public class CrawlingManager

{

public static volatile bool ShouldStopThreads = false;

public static volatile int WaitingThreadsCount = 0;

private Spider[] spiderArray;

private System.Threading.Timer timer;

private static string errorMessage = "";

private static string logMessage = "";

private static string indexedLinksMessage = "";

private static string errorWebMessage = "";

private static string errorWebProtocolMessage = "";

private static string errorWebTimeoutMessage = "";

private DateTime startDate;

/// <summary>

/// Declaring the Event Handler delegate

/// </summary>

public delegate void CrawlerEventHandler(Report.ProgressEventArgs pea);

public static event CrawlerEventHandler CrawlerEvent;

public CrawlingManager()

{

ResetFolders();

}

/// <summary>

/// Number of spider threads to start

/// </summary>

/// <param name="p"></param>

public void StartSpider()

{

SeedList.GetTheList();

timer = new System.Threading.Timer(new System.Threading.TimerCallback(WriteLog), null, 200, 3000);

startDate = DateTime.Now;

spiderArray = new Spider[Preferences.ThreadsCount];

try

{

for (int i = 0; i < Preferences.ThreadsCount; i++)

{

spiderArray[i] = new Spider(i);

spiderArray[i].SpiderProgressEvent += new MMarinov.WebCrawler.Report.SpiderProgressEventHandler(OnProgressEvent);

spiderArray[i].StartThread();

}

}

catch (System.Exception e)

{

GetErrorMessageForLog(new System.Exception("Error while creating Threads: ", e));

}

}

private static void GetErrorMessageForLog(System.Exception e)

{

if (CrawlerEvent != null)

{

Report.ProgressEventArgs pea = new Report.ProgressEventArgs(e);

CrawlerEvent(pea);

errorMessage += pea.Message;

}

}

private void SendMail(string bodyMsg)

{

try

{

System.Net.Mail.MailMessage mail = new System.Net.Mail.MailMessage();

mail.To.Add("m.marinov.de@gmail.com");

mail.From = new System.Net.Mail.MailAddress("WebCrawler@MMarinovService.com", "WebCrawler Service");

mail.Subject = "Crawling Report";

mail.Body = bodyMsg;

//mail.Attachments.Add(new System.Net.Mail.Attachment(""));

System.Net.Mail.SmtpClient sc = new System.Net.Mail.SmtpClient("localhost");

sc.Send(mail);

}

catch (System.Exception ex)

{

GetErrorMessageForLog(new System.Exception("Email Notifier", ex));

}

}

private void WakeThreads()

{

if (WaitingThreadsCount > 0)

{

foreach (Spider spider in spiderArray)

{

spider.WakeWaitingThead();

}

WaitingThreadsCount = 0;

}

}

public void StopSpiders()

{

ShouldStopThreads = true;

timer.Dispose();

System.Threading.Thread.Sleep(100);

KillSpiders();

for (int i = 0; i < Preferences.ThreadsCount; i++)

{

spiderArray[i].FlushData();

System.Threading.Thread.Sleep(50);

}

SendMail(SaveStatistics());

}

/// <summary>

/// Only terminates threads, used also for closing the programm without any saving

/// </summary>

public void KillSpiders()

{

for (int i = 0; i < Preferences.ThreadsCount; i++)

{

spiderArray[i].KillThread();

System.Threading.Thread.Sleep(200);

}

}

public double DownloadSpeed

{

get

{

return Document.DownloadSpeed;

}

}

}

}

**Spider.cs**

using System;

using MMarinov.WebCrawler.Report;

using MMarinov.WebCrawler.Library;

namespace MMarinov.WebCrawler.Indexer

{

/// <summary>

/// The Spider that crawls a website, link by link.

/// </summary>

public class Spider

{

internal static ThreadedGenerics.TList<string> GlobalURLsToVisit = new ThreadedGenerics.TList<string>();

internal static ThreadedGenerics.TList<string> GlobalVisitedURLs = new ThreadedGenerics.TList<string>();

#region Private fields

private object \_GlobalURLsToVisitSyncObj = new object();

private object \_GlobalVisitedURLsSyncObj = new object();

private int \_spiderIndex = 0;

private System.Threading.Thread thread;

// private System.Threading.EventWaitHandle \_threadWaitHandle = new System.Threading.AutoResetEvent(false);

private ThreadedGenerics.TList<string> \_visitedLinks = new ThreadedGenerics.TList<string>();

private ThreadedGenerics.TList<string> \_externalLinks = new ThreadedGenerics.TList<string>();

public static Int64 CrawledTotalLinks = 0;

public static Int64 CrawledSuccessfulLinks = 0;

private int \_localPagesCount = 0;

/// <summary></summary>

private FileCollection \_fileColl;

private static WordCollection \_wordsCollGlobal = WordCollection.GetWordCollection();

/// <summary>Stemmer to use</summary>

private static Stemming.IStemming \_Stemmer;

/// <summary>Stemmer to use</summary>

private static Stopper.IStopper \_Stopper;

/// <summary>Loads and acts as 'authorisation' for robot-excluded Urls</summary>

private RobotsTxt \_Robot;

#endregion

#region Public events/handlers: SpiderProgressEvent

/// <summary>

/// Event Handler to communicate progress and errors back to the calling code

/// </summary>

public event SpiderProgressEventHandler SpiderProgressEvent;

/// <summary>

/// Only trigger the event if a Handler has been attached.

/// </summary>

private void ProgressEvent(ProgressEventArgs pea)

{

if (this.SpiderProgressEvent != null)

{

SpiderProgressEvent(pea);

}

}

#endregion

public Spider(int spiderIndex)

{

\_spiderIndex = spiderIndex;

}

#region Thread methods

public void StartThread()

{

try

{

thread = new System.Threading.Thread(new System.Threading.ParameterizedThreadStart(this.BuildCatalog));

thread.Name = "[Spider " + \_spiderIndex + "]";

thread.Start((object)\_spiderIndex);

ProgressEvent(new ProgressEventArgs(EventTypes.Start, thread.Name + " started in " + DateTime.Now));

}

catch (System.Exception e)

{

ProgressEvent(new ProgressEventArgs(new System.Exception("Error while creating " + thread.Name, e)));

}

}

public void WakeWaitingThead()

{

if (thread.ThreadState == System.Threading.ThreadState.WaitSleepJoin)

{

thread.Interrupt();

}

}

/// <summary>

/// Aborts thread and flushs/saves catalogued results.

/// </summary>

internal void KillThread()

{

try

{

thread.Abort();

}

catch (System.Threading.ThreadStateException e)

{

ProgressEvent(new ProgressEventArgs(e));

}

System.Threading.Thread.Sleep(100);

}

#endregion

/// <summary>

/// Takes a single Uri (Url) and returns the catalog that is generated

/// by following all the links from that point.

/// </summary>

/// <remarks>

///This is the MAIN method of the indexing system.

/// </remarks>

public void BuildCatalog(object threadID)

{

Uri startPageUri;

try

{

while (!CrawlingManager.ShouldStopThreads)

{

startPageUri = null;

lock (GlobalURLsToVisit)

{

lock (GlobalVisitedURLs)

{

if (GlobalURLsToVisit.Count == 0)

{

if (!StopThreadsOnEmptyURLsList())

{

System.Threading.Thread.Sleep(100);

continue;

}

}

else

{

GlobalVisitedURLs.Add(GlobalURLsToVisit[0]);

startPageUri = new Uri(GlobalURLsToVisit[0]);

GlobalURLsToVisit.RemoveAt(0);

}

}

}

if (startPageUri != null)

{

ProgressEvent(new ProgressEventArgs(EventTypes.Start, thread.Name + ": Crawling website: " + startPageUri.AbsoluteUri));

InitListsAndPreferences();

\_Robot = new RobotsTxt(startPageUri, Preferences.RobotUserAgent);

ProcessUri(startPageUri, 0);

MergeExternalGlobalLinks();

\_fileColl.Save();

}

else

{

ProgressEvent(new ProgressEventArgs(EventTypes.End, thread.Name + ": !!! No more tasks - wait for a signal "));

}

}

ProgressEvent(new ProgressEventArgs(EventTypes.End, thread.Name + ": !!! Crawling finished at: " + DateTime.Now));

}

catch (Exception e)

{

}

}

private void InitListsAndPreferences()

{

\_fileColl = FileCollection.NewFileCollection();

\_visitedLinks = new MMarinov.ThreadedGenerics.TList<string>();

\_externalLinks = new MMarinov.ThreadedGenerics.TList<string>();

\_localPagesCount = 0;

// Setup Stop, Go, Stemming

SetPreferences();

}

/// <summary>

/// Checks if all threads are not working and kills them.

/// </summary>

/// <returns>True if Should Stop Threads</returns>

private bool StopThreadsOnEmptyURLsList()

{

CrawlingManager.WaitingThreadsCount++;

ProgressEvent(new ProgressEventArgs(EventTypes.EmptyVisitedURLs, thread.Name + " : !!! Empty GlobalURLsToVisit at: " + DateTime.Now));

if (CrawlingManager.WaitingThreadsCount == Preferences.ThreadsCount)

{

CrawlingManager.ShouldStopThreads = true;

}

return CrawlingManager.ShouldStopThreads;

}

private void MergeExternalGlobalLinks()

{

foreach (string link in \_externalLinks)

{

if (!GlobalVisitedURLs.Contains(link) && !GlobalURLsToVisit.Contains(link))

{

GlobalURLsToVisit.Add(link);

}

}

}

/// <summary>

/// Merges the external links for the current website with these, which are found in the local pages

/// </summary>

/// <param name="sourceList">The list whose values need to be merged.</param>

private void MergeExternalLinks(System.Collections.Generic.List<string> sourceList)

{

foreach (string str in sourceList)

{

if (!\_externalLinks.Contains(str))

{

\_externalLinks.Add(str);

}

}

}

/// <summary>

/// Setup Stop and Stemming

/// </summary>

private void SetPreferences()

{

if (Preferences.StemmingModeEnabled)

{

\_Stemmer = new Stemming.PorterStemmer(); //Stemming enabled.

}

else

{

\_Stemmer = new Stemming.NoStemming();//Stemming DISabled.

}

switch (Preferences.StoppingMode)

{

case Stopper.StoppingModes.Short:

\_Stopper = new Stopper.ShortStopper();//Stop words shorter than 3 chars.

break;

case Stopper.StoppingModes.List:

\_Stopper = new Stopper.ListStopper();//Stop words from list.

break;

case Stopper.StoppingModes.Off:

\_Stopper = new Stopper.NoStopping();//Stopping DISabled.

break;

default:

\_Stopper = new Stopper.ShortStopper();

break;

}

}

/// <summary>

///GETS THE FIRST DOCUMENT, AND STARTS THE SPIDER!

// RECURSIVE CALL

/// </summary>

protected int ProcessUri(Uri uri, int level)

{

if (level > Preferences.RecursionLimit || \_localPagesCount > 10000 || CrawlingManager.ShouldStopThreads)

{

return Preferences.RecursionLimit;

}

int wordcount = 0;

string url = uri.AbsoluteUri;

if (\_Robot.Allowed(uri) && !\_visitedLinks.Contains(url))

{

\_visitedLinks.Add(url);

CrawledTotalLinks++;

Document downloadDocument = Download(uri);

if (null != downloadDocument)

{

downloadDocument.DocumentEvent += new Document.DocumentProgressEventHandler(downloadDocument\_DocumentEvent);

downloadDocument.Parse();

MergeExternalLinks(downloadDocument.ExternalLinks);// Adds external links from every page from the site

if (downloadDocument.RobotIndexOK)

{

if (downloadDocument is HtmlDocument)

{

SetLanguages(((HtmlDocument)downloadDocument).Language);

}

wordcount = AddToCatalog(downloadDocument);

ProgressEvent(new ProgressEventArgs(EventTypes.Crawling, thread.Name + " " + \_localPagesCount++));

ProgressEvent(new ProgressEventArgs(EventTypes.Crawling, thread.Name + " " + ++CrawledSuccessfulLinks + ": " + url + " [" + wordcount + " words]"));

}

}

// ### Loop through the 'local' links in the document and parse each of them recursively ###

if (null != downloadDocument && null != downloadDocument.LocalLinks && downloadDocument.RobotFollowOK)

{ // only if the Robot meta says it's OK

foreach (string link in downloadDocument.LocalLinks)

{

try

{

ProcessUri(new Uri(downloadDocument.Uri, link), level + 1); // calls THIS method!

}

catch (Exception ex)

{

ProgressEvent(new ProgressEventArgs(new Exception(" new Uri(" + downloadDocument.Uri + ", " + link + ") invalid : ", ex)));

}

}

} // process local links

} // robot allowed and not visited

return level;

}// ProcessUri

private void downloadDocument\_DocumentEvent(ProgressEventArgs pea)

{

ProgressEvent(pea);

}

/// <summary>

/// Set current language of the HTML page to Stemmer and Stopper

/// </summary>

/// <param name="lang"></param>

private void SetLanguages(Stemming.Languages lang)

{

if (\_Stopper is Stopper.ListStopper)

{

((Stopper.ListStopper)\_Stopper).Language = lang;

}

if (\_Stemmer is Stemming.PorterStemmer)

{

((Stemming.PorterStemmer)\_Stemmer).Language = lang;

}

}

/// <summary>

/// Attempts to download the Uri and (based on it's MimeType) use the DocumentFactory

/// to get a Document subclass object that is able to parse the downloaded data.

/// </summary>

private Document Download(Uri uri)

{

System.Net.ServicePointManager.DefaultConnectionLimit = 200;

System.Net.ServicePointManager.MaxServicePointIdleTime = 2000;

// Open the requested URL

System.Net.HttpWebRequest webRequest = (System.Net.HttpWebRequest)System.Net.HttpWebRequest.Create(uri.AbsoluteUri);

webRequest.UnsafeAuthenticatedConnectionSharing = true;

webRequest.AllowAutoRedirect = true;

webRequest.MaximumAutomaticRedirections = 3;

webRequest.UserAgent = Preferences.UserAgent; //"Mozilla/6.0 (MSIE 6.0; Windows NT 5.1; MMarinov.NET)";

webRequest.KeepAlive = true;

webRequest.Method = System.Net.WebRequestMethods.Http.Get;

webRequest.Timeout = Preferences.RequestTimeout \* 1000;

// Get the stream from the returned web response

System.Net.HttpWebResponse webResponse = null;

Document htmldoc = null;

try

{

webResponse = (System.Net.HttpWebResponse)webRequest.GetResponse();

}

catch (System.Net.WebException ex)

{ //remote url not found, 404; remote url forbidden, 403

ProgressEvent(new ProgressEventArgs(new Exception(uri.AbsoluteUri, ex), ex.Status));

}

finally

{

if (webResponse != null)

{

htmldoc = DocumentFactory.New(uri, webResponse);

if (htmldoc != null)

{

htmldoc.GetResponse(webResponse);

}

webResponse.Close();

}

}

return htmldoc;

}

/// <summary>

///

/// </summary>

/// <return>Number of words catalogued</return>

private int AddToCatalog(Document downloadDocument)

{

int wordsCount = 0;

string wordName = ""; // temp variable

WordCollection newWordsColl = WordCollection.NewWordCollection();

System.Collections.Generic.List<string> formatedNewWords = new System.Collections.Generic.List<string>();

File infile = File.NewFile();

infile.Title = downloadDocument.Title;

infile.FileType = downloadDocument.FileType;

infile.Keywords = downloadDocument.Keywords;

infile.URL = Common.GetHttpAuthority(downloadDocument.Uri) + downloadDocument.Uri.AbsolutePath;

infile.Description = downloadDocument.Description;

foreach (string rawWord in downloadDocument.WordsArray)

{

wordName = rawWord.ToLower();

// Apply Stemming and stopping (set by preferences)

wordName = \_Stemmer.StemWord(wordName);

wordName = \_Stopper.StopWord(wordName);

if (wordName != "")

{

wordsCount++;

lock (\_wordsCollGlobal)

{

Word w = \_wordsCollGlobal.GetWord(wordName);

if (w == null)

{

if (!newWordsColl.Contains(wordName))

{

newWordsColl.Add(Word.NewWord(wordName));

}

formatedNewWords.Add(wordName);

}

else

{

if (w.ID == 0)

{

}

infile.WordsInFileColl.AddOrIncrease(w.ID);

}

}

}

} // foreach

newWordsColl.Save();

lock (\_wordsCollGlobal)

{

\_wordsCollGlobal.AddRange(newWordsColl);

}

foreach (string formWord in formatedNewWords)

{

Word wrd = newWordsColl.GetWord(formWord);

if (wrd == null)

{

}

infile.WordsInFileColl.AddOrIncrease(wrd.ID);

}

\_fileColl.Add(infile);

return wordsCount;

}

public static long WordsCount

{

get

{

return \_wordsCollGlobal.Count;

}

}

internal void FlushData()

{

try

{

\_fileColl.Save();

}

catch (Exception e)

{

}

}

}

}

SeedList.cs

using System;

using System.Linq;

namespace MMarinov.WebCrawler.Indexer

{

public class SeedList

{

private static string \_zipFilename = "";

private static string \_csvFilename = "top-1m.csv";

public static void GetTheList()

{

if (DownloadFile())

{

ExtractZipArchive();

}

FetchFromCVS();

}

/// <summary>

/// Gets links from CVS file and feed the SeedList.

/// Its stucture is: in first column > cell(index, link)

/// </summary>

private static void FetchFromCVS()

{

using (System.IO.StreamReader readFile = new System.IO.StreamReader(Preferences.WorkingPath + "\\" + \_csvFilename))

{

string fileContent = System.Text.RegularExpressions.Regex.Replace(readFile.ReadToEnd(), @"\d+,", "");

string[] links = fileContent.Split(Environment.NewLine.ToCharArray(), StringSplitOptions.RemoveEmptyEntries);

System.Collections.Generic.List<string> linkList = links.ToList<string>();

linkList.ForEach(url =>

{

try

{

Spider.GlobalURLsToVisit.Add(Common.GetHttpAuthority(new System.Uri(Common.HTTP + url)));

}

catch

{

linkList.Remove(url);

}

});

}

}

private static void ExtractZipArchive()

{

using (Ionic.Zip.ZipFile zipArchiveNew = Ionic.Zip.ZipFile.Read(\_zipFilename))

{

Ionic.Zip.ZipEntry zipFileNew = zipArchiveNew[0];

\_csvFilename = zipFileNew.FileName;

zipFileNew.Extract(Preferences.WorkingPath, Ionic.Zip.ExtractExistingFileAction.OverwriteSilently);

MMarinov.WebCrawler.Report.Logger.MessageLog("Zip file " + Preferences.WorkingPath + "\\" + \_csvFilename + " extracted at " + System.DateTime.Now.ToString(Common.DateFormat), Report.EventTypes.Other);

}

}

private static bool DownloadFile()

{

if (Preferences.SeedURLsSource == "")

{

return false;

}

try

{

System.Uri uriZip = new System.Uri(Preferences.SeedURLsSource);

\_zipFilename = Preferences.WorkingPath + "\\SeedList\_" + uriZip.Segments[2];

System.Net.WebClient client = new System.Net.WebClient();

client.DownloadFileCompleted += new System.ComponentModel.AsyncCompletedEventHandler(client\_DownloadFileCompleted);

client.DownloadFile(uriZip, \_zipFilename);

MMarinov.WebCrawler.Report.Logger.MessageLog("Zip file " + \_zipFilename + " downloaded at " + System.DateTime.Now.ToString(Common.DateFormat), Report.EventTypes.Other);

return true;

}

catch (System.Exception e)

{

MMarinov.WebCrawler.Report.Logger.ErrorLog(e);

}

return false;

}

static void client\_DownloadFileCompleted(object sender, System.ComponentModel.AsyncCompletedEventArgs e)

{

throw new System.NotImplementedException();

}

}

}

RobotsTxt.cs

using System;

using System.Text;

namespace MMarinov.WebCrawler.Indexer

{

/// <summary>

/// Represents the rules for a specific domain for a specific host

/// (ie it aggregates all the rules that match the UserAgent, plus the special \* rules)

///

/// http://www.robotstxt.org/

/// </summary>

public class RobotsTxt

{

#region Private Fields: \_FileContents, \_UserAgent, \_Server, \_DenyUrls, \_LogString

private string \_FileContents;

private string \_UserAgent;

private string \_Server;

/// <summary>lowercase string array of url fragments that are 'denied' to the UserAgent for this RobotsTxt instance</summary>

private System.Collections.Generic.List<string> \_DenyUrls = new System.Collections.Generic.List<string>();

#endregion

#region Constructors: require starting Url and UserAgent to create an object

private RobotsTxt()

{ }

public RobotsTxt(Uri startPageUri, string userAgent)

{

\_UserAgent = userAgent;

\_Server = startPageUri.Host;

System.Net.HttpWebRequest req = (System.Net.HttpWebRequest)System.Net.WebRequest.Create("http://" + startPageUri.Authority + "/robots.txt");

try

{

System.Net.HttpWebResponse webresponse = (System.Net.HttpWebResponse)req.GetResponse();

using (System.IO.StreamReader stream = new System.IO.StreamReader(webresponse.GetResponseStream(), Encoding.ASCII))

{

\_FileContents = stream.ReadToEnd();

} // stream.Close();

string[] fileLines = \_FileContents.Split(Environment.NewLine.ToCharArray(), StringSplitOptions.RemoveEmptyEntries);

bool rulesApply = false;

foreach (string line in fileLines)

{

RobotInstruction ri = new RobotInstruction(line);

if (ri.Instruction == "")

{

continue;

}

switch (ri.Instruction[0])

{

case '#': //then comment - ignore

break;

case 'u': // User-Agent

rulesApply = (ri.UrlOrAgent.IndexOf("\*") >= 0) || (ri.UrlOrAgent.IndexOf(\_UserAgent) >= 0);

break;

case 'd': // Disallow

if (rulesApply)

{

\_DenyUrls.Add(ri.UrlOrAgent.ToLower());

}

break;

default:

// empty/unknown/error/allow

break;

}

}

}

catch (System.Net.WebException)

{

\_FileContents = "";

}

}

#endregion

#region Methods: Allow

/// <summary>

/// Does the parsed robots.txt file allow this Uri to be spidered for this user-agent?

/// </summary>

/// <remarks>

/// This method does all its "matching" in lowercase - it expects the \_DenyUrl

/// elements to be ToLower() and it calls ToLower on the passed-in Uri...

/// </remarks>

public bool Allowed(Uri uri)

{

if (\_DenyUrls.Count == 0) return true;

string url = uri.AbsolutePath.ToLower();

foreach (string denyUrlFragment in \_DenyUrls)

{

if (url.Length >= denyUrlFragment.Length && url.Substring(0, denyUrlFragment.Length) == denyUrlFragment)

{

return false;

}

// else url is shorter than fragment, therefore cannot be a 'match' // else not a match

}

if (url == "/robots.txt")

{

return false;

}

// no disallows were found, so allow

return true;

}

#endregion

/// <summary>

/// Use this class to read/parse the robots.txt file

/// </summary>

/// <remarks>

/// Types of data coming into this class

/// User-agent: \* ==> \_Instruction='User-agent', \_Url='\*'

/// Disallow: /cgi-bin/ ==> \_Instruction='Disallow', \_Url='/cgi-bin/'

/// Disallow: /tmp/ ==> \_Instruction='Disallow', \_Url='/tmp/'

/// Disallow: /~joe/ ==> \_Instruction='Disallow', \_Url='/~joe/'

/// </remarks>

private class RobotInstruction

{

private string \_Instruction = "";

private string \_Url = "";

/// <summary>

/// Constructor requires a line, hopefully in the format [instuction]:[url]

/// </summary>

public RobotInstruction(string line)

{

string instructionLine = line.Trim();

int commentPosition = instructionLine.IndexOf('#');

if (commentPosition == 0)

{

\_Instruction = "#";

}

if (commentPosition >= 0)

{ // comment somewhere on the line, trim it off

instructionLine = instructionLine.Substring(0, commentPosition);

}

if (instructionLine.Length > 0)

{ // wasn't just a comment line (which should have been filtered out before this anyway

string[] lineArray = instructionLine.Split(':');

\_Instruction = lineArray[0].Trim().ToLower();

if (lineArray.Length > 1)

{

\_Url = lineArray[1].Trim();

}

}

}

/// <summary>

/// Lower-case part of robots.txt line, before the colon (:)

/// </summary>

public string Instruction

{

get { return \_Instruction; }

}

/// <summary>

/// Lower-case part of robots.txt line, after the colon (:)

/// </summary>

public string UrlOrAgent

{

get { return \_Url; }

}

}

}

}

App.config

<?xml version="1.0" encoding="utf-8" ?>

<configuration>

<appSettings>

<!-- If False will also index PDFs and Mp3s -->

<add key="IndexOnlyHTMLDocuments" value="true"/>

<add key="ThreadsCount" value="3"/>

<!-- Seconds to wait for a page to respond, before giving up -->

<add key="RequestTimeout" value="3"/>

<!-- Limit to the number of 'levels' of links to follow -->

<add key="RecursionLimit" value="20"/>

<!-- Request another page after waiting x seconds; use zero ONLY on your own/internal sites -->

<add key="SpeedLimit" value="1"/>

<!-- Whether to use stemming (English only) false/true -->

<add key="StemmingModeEnabled" value="False"/>

<!-- Whether to use stop words (English, German,Bulgarian), and if so, what mode [ Off | Short | List ] -->

<add key="StoppingType" value="List"/>

<add key="UserAgent" value="Mozilla/6.0 (MSIE 6.0; Windows NT 5.1; MMarinov.NET; robot)"/>

<add key="WorkingPath" value="D:\MMarinov\_docs\Universitet\Thesis\\_CrawlerWorkingFolder"/>

<!--<add key="WorkingPath" value="C:\MMarinov\projects\\_CrawlerWorkingFolder"/>-->

<!-- Language to use when none is supplied (or supplied language is not available) -->

<add key="DefaultLanguage" value="en-US"/>

<!--URL with a zip archive with the top 1000000 websites from Alexa; leave it blank if you want to be indepentent.-->

<!--<add key="SeedURLsSource" value="http://s3.amazonaws.com/alexa-static/top-1m.csv.zip"/>-->

<add key="DB:WebCrawler" value="Data Source=MMarinov-PC;Initial Catalog=WebCrawler;Persist Security Info=True;User ID=MMarinov;Password=IamNotSoStupid"/>

<add key="ConnectionStringActive" value="Data Source=MMarinov-PC;Initial Catalog=WebCrawlerActive;Persist Security Info=True;User ID=MMarinov;Password=OrMaybeThereIsNoPass"/>

</appSettings>

<system.net>

<connectionManagement>

<add address="\*" maxconnection="65000" />

</connectionManagement>

<defaultProxy enabled="false">

<proxy/>

<bypasslist/>

<module/>

</defaultProxy>

</system.net>

</configuration>

MainWindow.xaml

<Window x:Class="MMarinov.WebCrawler.UI.MainWindow"

xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"

xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"

Title="MMarinov's Web Crawler" Height="604" Width="1298" WindowState="Maximized" Name="theWin">

<Window.Triggers>

<EventTrigger RoutedEvent="Window.Loaded">

<BeginStoryboard>

<Storyboard>

<DoubleAnimation Storyboard.TargetProperty="Opacity" Storyboard.TargetName="theWin" From="0.2" To="1" Duration="0:0:0.8" />

</Storyboard>

</BeginStoryboard>

</EventTrigger>

<EventTrigger RoutedEvent="Window.Unloaded">

<BeginStoryboard>

<Storyboard Name="FormFadeOut" Completed="FormFadeOut\_Completed\_1">

<DoubleAnimation Storyboard.TargetProperty="Opacity" Storyboard.TargetName="theWin" From="1.0" To="0.0" Duration="0:0:1" />

</Storyboard>

</BeginStoryboard>

</EventTrigger>

</Window.Triggers>

<Grid>

<Grid.ColumnDefinitions>

<ColumnDefinition Width="456\*" />

<ColumnDefinition Width="467\*" />

</Grid.ColumnDefinitions>

<Grid.RowDefinitions>

<RowDefinition Height="44\*"></RowDefinition>

<RowDefinition Height="325\*"></RowDefinition>

<RowDefinition Height="116.873\*"></RowDefinition>

<RowDefinition Height="25"></RowDefinition>

</Grid.RowDefinitions>

<Button HorizontalAlignment="Left" Margin="12,12,0,9" Name="btnStart" Width="55" IsCancel="False" IsDefault="False" Click="btnStart\_Click" Visibility="Visible" Background="LightGreen" VerticalAlignment="Top">Start</Button>

<Button HorizontalAlignment="Left" Margin="78,12,0,9" Name="btnStop" IsCancel="False" OverridesDefaultStyle="False" Click="btnStop\_Click" Background="OrangeRed" Width="55" VerticalAlignment="Top">Stop</Button>

<Button HorizontalAlignment="Left" Margin="159,12,191,9" Name="btnSaveToDB" Click="btnCopyToActiveDB\_Click" Background="Yellow" VerticalAlignment="Top">Copy to Active DB</Button>

<GroupBox Grid.Row="1" Grid.Column="0" Header="Indexed Links" Margin="2,2,2,2">

<TextBox IsReadOnly="True" Margin="5,5,5,5" Name="tbIndexedLinks" Grid.Row="1" TextDecorations="None" Visibility="Visible" IsEnabled="True" TextWrapping="Wrap" VerticalScrollBarVisibility="Auto" HorizontalScrollBarVisibility ="Visible" />

</GroupBox>

<GroupBox Grid.Column="0" Grid.Row="2" Header="Crawled Domains" Margin="2,2,2,2">

<TextBox IsReadOnly="True" Name="tbCrawlerDomains" Grid.Row="2" VerticalScrollBarVisibility="Auto" Margin="5,5,5,5" HorizontalScrollBarVisibility="Disabled"></TextBox>

</GroupBox>

<Label Margin="300,12,180,14.168" Name="lblStartTime">Start time:</Label>

<Label HorizontalAlignment="Right" Margin="0,12,8.613,14.168" Name="lblTimeElapsed" Width="131.387">Elapsed:</Label>

<GroupBox Grid.Column="1" Grid.Row="2" Header="Timeout exceptions" Margin="2,2,2,2">

<TextBox IsReadOnly="True" HorizontalScrollBarVisibility="Visible" Margin="5,5,5,5" Name="tbTimeoutEx" TextWrapping="Wrap" VerticalScrollBarVisibility="Auto" Grid.Column="1" Grid.Row="2" />

</GroupBox>

<Grid Grid.Column="1" Grid.Row="1">

<Grid.ColumnDefinitions>

<ColumnDefinition Width="48\*"></ColumnDefinition>

</Grid.ColumnDefinitions>

<Grid.RowDefinitions>

<RowDefinition></RowDefinition>

<RowDefinition></RowDefinition>

<RowDefinition></RowDefinition>

</Grid.RowDefinitions>

<GroupBox Grid.Column="0" Grid.Row="0" Header="Errors" Margin="2,2,2,2">

<TextBox IsReadOnly="True" Name="tbErrors" TextWrapping="Wrap" VerticalScrollBarVisibility="Auto" HorizontalScrollBarVisibility ="Visible" Margin="5,5,5,5" />

</GroupBox>

<GroupBox Grid.Column="0" Grid.Row="1" Header="Web exceptions" Margin="2,2,2,2">

<TextBox IsReadOnly="True" HorizontalScrollBarVisibility="Visible" Margin="5,5,5,5" Name="tbWebEx" TextWrapping="Wrap" VerticalScrollBarVisibility="Auto" Grid.Row="1" Grid.ColumnSpan="2" />

</GroupBox>

<GroupBox Grid.Column="0" Grid.Row="2" Header="Protocol exceptions" Margin="2,2,2,2">

<TextBox IsReadOnly="True" HorizontalScrollBarVisibility="Visible" Margin="5,5,5,5" Name="tbProtocolEx" TextWrapping="Wrap" VerticalScrollBarVisibility="Auto" Grid.Row="2" Grid.ColumnSpan="2" />

</GroupBox>

</Grid>

<Label HorizontalAlignment="Left" Grid.Column="1" Margin="51.103,9,0,17.168" Name="lblDlSpeed" Width="166">Download speed:</Label>

<Label Grid.Column="1" HorizontalAlignment="Right" Margin="0,9,300,6" Name="lblTotalFoundLinks" Width="Auto" VerticalAlignment="Top">Total links found:</Label>

<Label HorizontalAlignment="Right" Margin="0,9,100,6" Name="lblTotalValidLinksFound" Width="Auto" Grid.Column="1" VerticalAlignment="Top">Total valid links found:</Label>

<StatusBar Grid.Row="3" Grid.Column="0" VerticalAlignment="Bottom" >

<StatusBarItem>

<TextBlock> Status: </TextBlock>

</StatusBarItem>

<StatusBarItem>

<TextBlock Name="lblStatus"></TextBlock>

</StatusBarItem>

</StatusBar>

<Button Grid.Column="1" Grid.Row="3" HorizontalAlignment="Right" VerticalAlignment="Top" Margin="0,0,10,3" Name="btnShowStatistics" Width="106" VerticalContentAlignment="Center" Click="btnShowStatistics\_Click">Show statistics</Button>

</Grid>

</Window>

**MainWindow.cs**

using System;

using System.Windows;

using System.Windows.Controls;

using System.Windows.Input;

namespace MMarinov.WebCrawler.UI

{

/// <summary>

/// Interaction logic for MainWindow.xaml

/// </summary>

public partial class MainWindow : Window

{

private MMarinov.WebCrawler.Indexer.CrawlingManager manager = null;

private System.Threading.Timer timer;

private Int64 elapsedSec = 0;

public MainWindow()

{

InitializeComponent();

btnStop.IsEnabled = false;

}

private void btnStart\_Click(object sender, RoutedEventArgs e)

{

Cursor = Cursors.Wait;

tbCrawlerDomains.Text = "";

tbErrors.Text = "";

tbIndexedLinks.Text = "";

tbProtocolEx.Text = "";

tbTimeoutEx.Text = "";

tbWebEx.Text = "";

btnStop.IsEnabled = true;

btnStart.IsEnabled = !btnStop.IsEnabled;

btnSaveToDB.IsEnabled = !btnStop.IsEnabled;

ProgressDialog dlg = new ProgressDialog("This procees will override the database! Truncating tables .. ", this, true);

dlg.RunWorkerThread(StartCrawling);

elapsedSec = 0;

timer = new System.Threading.Timer(new System.Threading.TimerCallback(ShowElapsedTimeAndDLSpeed), null, 0, 1000);

lblStatus.Text = "Crawling...";

lblStartTime.Content = "Started on " + DateTime.Now.ToString(Common.DateFormat);

Cursor = Cursors.Arrow;

}

/// <summary>

/// This method will be invoked on a background thread by the ProgressDialog control.

/// </summary>

/// <param name="sender">The background worker component</param>

/// <param name="e">Provides the start value as the argument.</param>

private void StartCrawling(object sender, System.ComponentModel.DoWorkEventArgs e)

{

//the sender property is a reference to the dialog's BackgroundWorker component

System.ComponentModel.BackgroundWorker worker = (System.ComponentModel.BackgroundWorker)sender;

System.Threading.Thread.Sleep(600);

DBLibrary.StoredProceduresManager.InitAllStoredProcedures();

Library.DBCopier.TruncateDBTables();

worker.ReportProgress(30, "Initializing crawling process...");

manager = new MMarinov.WebCrawler.Indexer.CrawlingManager();

Indexer.CrawlingManager.CrawlerEvent += new Indexer.CrawlingManager.CrawlerEventHandler(CrawlingManager\_CrawlerEvent);

manager.StartSpider();

worker.ReportProgress(70, "Loading the seed list...");

System.Threading.Thread.Sleep(800);

worker.ReportProgress(100);

}

private void ShowElapsedTimeAndDLSpeed(object o)

{

lblTimeElapsed.Dispatcher.BeginInvoke(

System.Windows.Threading.DispatcherPriority.Normal,

(Action)(() =>

{ lblTimeElapsed.Content = "Elapsed: " + TimeSpan.FromSeconds(++elapsedSec); }));

lblDlSpeed.Dispatcher.BeginInvoke(

System.Windows.Threading.DispatcherPriority.Normal,

(Action)(() =>

{ lblDlSpeed.Content = "Download speed: " + manager.DownloadSpeed.ToString("######0.00") + " KB/s"; }));

}

private void AddMessageToTextbox(Report.ProgressEventArgs pea, TextBox tbx)

{

tbx.Dispatcher.BeginInvoke(

System.Windows.Threading.DispatcherPriority.Normal,

(Action)(() =>

{

if (tbx.LineCount > 500)

{

tbx.Text = "";

}

tbx.Text = pea.Message + Environment.NewLine + tbx.Text;

}));

}

private void btnStop\_Click(object sender, RoutedEventArgs e)

{

Cursor = Cursors.Wait;

ProgressDialog dlg = new ProgressDialog("Finalizing crawling process.. ", this, true);

dlg.RunWorkerThread(StopCrawling);

btnStop.IsEnabled = false;

btnStart.IsEnabled = !btnStop.IsEnabled;

btnSaveToDB.IsEnabled = !btnStop.IsEnabled;

lblStatus.Text = "Stopped.";

Cursor = Cursors.Arrow;

}

private void StopCrawling(object sender, System.ComponentModel.DoWorkEventArgs e)

{

//the sender property is a reference to the dialog's BackgroundWorker component

System.ComponentModel.BackgroundWorker worker = (System.ComponentModel.BackgroundWorker)sender;

manager.StopSpiders();

timer.Dispose();

worker.ReportProgress(70, "Saving statistics ...");

System.Threading.Thread.Sleep(1000);

worker.ReportProgress(100);

}

private void btnCopyToActiveDB\_Click(object sender, RoutedEventArgs e)

{

Cursor = Cursors.Wait;

lblStatus.Text = "Coping DB to active DB, used of the web client...";

ProgressDialog dlg = new ProgressDialog("Coping DB to active DB, used of the web client...", this, true);

dlg.RunWorkerThread(CopyDatabase);

lblStatus.Text = "Saved to Active Database";

Cursor = Cursors.Arrow;

}

private void CopyDatabase(object sender, System.ComponentModel.DoWorkEventArgs e)

{

//the sender property is a reference to the dialog's BackgroundWorker component

System.ComponentModel.BackgroundWorker worker = (System.ComponentModel.BackgroundWorker)sender;

Library.DBCopier.CopyDBToActiveDB();

worker.ReportProgress(100);

}

#region Closing

protected override void OnClosing(System.ComponentModel.CancelEventArgs e)

{

if (!btnStart.IsEnabled)

{

if (!closeCompleted && MessageBox.Show("Crawling process in progress. Are you sure you want to terminate it?", "Exit application?", MessageBoxButton.YesNo, MessageBoxImage.Asterisk) == MessageBoxResult.Yes)

{

if (!closeCompleted)

{

manager.KillSpiders();

e.Cancel = true;

FormFadeOut.Begin();

}

}

}

else if (!closeCompleted)

{

e.Cancel = true;

FormFadeOut.Begin();

}

}

private bool closeCompleted = false;

private void FormFadeOut\_Completed\_1(object sender, EventArgs e)

{

closeCompleted = true;

Application.Current.Shutdown();//killin' me softly

}

#endregion

private void btnShowStatistics\_Click(object sender, RoutedEventArgs e)

{

ViewStatistics stats = new ViewStatistics();

stats.WindowState = WindowState.Maximized;

stats.Show();

}

}

}

* Source code of ‘Margent’ search agent

Default.aspx

<%@ Page Language="C#" AutoEventWireup="true" CodeFile="Default.aspx.cs" Inherits="Margent.\_Default" %>

<%@ Register Assembly="AjaxControlToolkit" Namespace="AjaxControlToolkit" TagPrefix="ajaxToolkit" %>

<%@ Register Assembly="System.Web.DynamicData, Version=3.5.0.0, Culture=neutral, PublicKeyToken=31bf3856ad364e35"

Namespace="System.Web.DynamicData" TagPrefix="cc1" %>

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<html xmlns="http://www.w3.org/1999/xhtml">

<head runat="server">

<link rel="Stylesheet" href="Styles/Page.css" type="text/css" />

<link rel="Stylesheet" href="Styles/GridViews.css" type="text/css" />

<script type="text/javascript" src="JavaScripts/GridActions.js" language="javascript"></script>

<script type="text/javascript" src="JavaScripts/JQuery.js" language="javascript"></script>

<!--[if IE 6]>

<style type="text/css">

html { overflow-y: hidden; }

body { overflow-y: auto; }

#bg { position:absolute; z-index:-1; }

#content { position:static; }

</style>

<![endif]-->

<title>Margent - MMarinov's Search Agent</title>

</head>

<body>

<form id="form1" runat="server">

<asp:ScriptManager ID="ScriptManager1" runat="server" EnablePageMethods="true">

</asp:ScriptManager>

<img src="Images/backgroundMain.jpg" alt="background image" id="bg" />

<div id="content">

<div id="divLoading">

<asp:UpdateProgress ID="UpdateProgress1" runat="server" AssociatedUpdatePanelID="updPanel">

<ProgressTemplate>

<asp:Image runat="server" ImageUrl="~/Images/LoadingAnimation.gif" Width="150px" />

</ProgressTemplate>

</asp:UpdateProgress>

</div>

<div class="Title">

<img src="Images/MargentTitle.PNG" alt="MMairnov Search Agent" />

</div>

<table width="100%">

<tr>

<td class="CellLabelSearch" width="35%">

<asp:Label ID="lblSearch" runat="server" Text="Search the knowledge base:"></asp:Label>

</td>

<td width="30%">

<asp:TextBox ID="tbSearchQuery" runat="server" CssClass="SearchQuery" autocomplete="off"

AutoCompleteType="None"></asp:TextBox>

<ajaxToolkit:AutoCompleteExtender EnableCaching="true" MinimumPrefixLength="3" CompletionSetCount="10"

runat="server" ServicePath="MMWebService.asmx" ServiceMethod="GetSuggestions"

TargetControlID="tbSearchQuery" CompletionListCssClass="autocomplete\_completionListElement"

CompletionListItemCssClass="autocomplete\_listItem" CompletionListHighlightedItemCssClass="autocomplete\_highlightedListItem"

CompletionInterval="300">

</ajaxToolkit:AutoCompleteExtender>

</td>

<td width="35%" style="padding-left: 10px;">

<asp:ImageButton ID="btnDoSearch" runat="server" Height="37px" ImageUrl="~/Images/binoc.png"

onmouseover="this.src='Images/binoc\_hover.png';" onmouseout="this.src='Images/binoc.png';" />

</td>

</tr>

</table>

<div class="PositionGrids">

<div class="CenterObject">

<asp:Label runat="server" ID="lblError" CssClass="ErrorMessage"> error message</asp:Label>

</div>

<asp:UpdatePanel ID="updPanel" runat="server" UpdateMode="Conditional">

<Triggers>

<asp:AsyncPostBackTrigger ControlID="gvKeywords" EventName="SelectedIndexChanged" />

</Triggers>

<ContentTemplate>

<asp:GridView ID="gvKeywords" runat="server" AutoGenerateColumns="False" AllowSorting="True"

CssClass="GridKeywords" AllowPaging="True" ShowHeader="true" GridLines="None"

CellPadding="2">

<RowStyle CssClass="RowStyle" />

<PagerStyle CssClass="PagerStyle" />

<EmptyDataRowStyle CssClass="NoResults" />

<AlternatingRowStyle CssClass="AltRowStyle" />

<HeaderStyle CssClass="HeaderStyle" />

<FooterStyle />

<Columns>

<asp:TemplateField>

<HeaderTemplate>

<asp:Label runat="server" CssClass="HeaderCol1" ID="lblHeaderWord">Word</asp:Label>

<asp:Label runat="server" CssClass="HeaderCol2" ID="lblHeaderCount">Rank</asp:Label>

</HeaderTemplate>

<ItemTemplate>

<div class="Keywords">

<asp:ImageButton ID="btnToggle" runat="server" ImageUrl="~/Images/Plus.gif" />

<asp:LinkButton runat="server" ID="lnkKeyword">keyword</asp:LinkButton>

<asp:Label runat="server" ID="lblTotalCount" CssClass="TotalCount">total count</asp:Label>

</div>

<!-- ===============nested view ================= -->

<asp:Literal ID="litGridLinks" runat="server"></asp:Literal>

<asp:Literal runat="server" ID="lit1" Text="<div id='trCollapseGrid' class='RowStyle' style='display:none' >" />

<asp:UpdateProgress ID="UpdateProgress2" runat="server" AssociatedUpdatePanelID="updPanelLinks">

<ProgressTemplate>

<asp:Image ID="Image1" runat="server" ImageUrl="~/Images/LoadingAnimation.gif" Width="150px" />

</ProgressTemplate>

</asp:UpdateProgress>

<asp:UpdatePanel ID="updPanelLinks" runat="server">

<ContentTemplate>

<asp:GridView ID="gvLinks" runat="server" AutoGenerateColumns="false" ShowHeader="false"

EnableViewState="False" CssClass="GridLinks" AllowPaging="True" GridLines="None"

OnRowCommand="gvLinks\_RowCommand" OnPageIndexChanging="gvLinks\_PageIndexChanging">

<PagerStyle CssClass="PagerStyle" />

<RowStyle CssClass="RowStyleLinks" />

<EmptyDataRowStyle CssClass="NoResults" />

<AlternatingRowStyle CssClass="AltRowStyleLinks" />

<Columns>

<asp:TemplateField>

<ItemTemplate>

<div class="LinkTitle">

<asp:HyperLink runat="server" ID="lnkLinkTitle">title - follow me</asp:HyperLink>

</div>

<div class="LinkDescription">

<asp:Label runat="server" ID="lblLinkDescription"></asp:Label>

</div>

<div class="LinkLink">

<asp:HyperLink runat="server" ID="lnkLink">link follow me</asp:HyperLink>

</div>

</ItemTemplate>

</asp:TemplateField>

</Columns>

<PagerTemplate>

<div style="height: 20px;">

<div class="command">

<asp:ImageButton ID="btnFirstLinks" runat="server" CommandName="FirstLinks" ImageUrl="~/Images/btnFirst.PNG"

AlternateText="First Page" ToolTip="First Page" />

<asp:ImageButton ID="btnPreviousLinks" runat="server" CommandName="PreviousLinks"

ImageUrl="~/Images/btnPrev.PNG" AlternateText="Previous Page" ToolTip="Previous Page" />

</div>

<div class="command">

<asp:TextBox ID="txtSlideLinks" runat="server" AutoPostBack="true" OnTextChanged="txtSlideLinks\_Changed" />

<ajaxToolkit:SliderExtender ID="ajaxSliderLinks" runat="server" TargetControlID="txtSlideLinks"

RaiseChangeOnlyOnMouseUp="true" Orientation="Horizontal" Minimum="1" />

</div>

<div class="command">

<asp:ImageButton ID="btnNextLinks" runat="server" CommandName="NextLinks" ImageUrl="~/Images/btnNext.PNG"

AlternateText="Next Page" ToolTip="Next Page" />

<asp:ImageButton ID="btnLastLinks" runat="server" CommandName="LastLinks" ImageUrl="~/Images/btnLast.PNG"

AlternateText="Last Page" ToolTip="Last Page" />

</div>

<div class="PagerInfo">

<asp:Label ID="lblPageLinks" CssClass="PagerInfo" runat="server" />

</div>

</div>

</PagerTemplate>

</asp:GridView>

</ContentTemplate>

</asp:UpdatePanel>

<asp:Literal runat="server" ID="lit2" Text="</div>" />

<!-- ===============nested view end ============== -->

</ItemTemplate>

</asp:TemplateField>

</Columns>

<PagerTemplate>

<div style="height: 30px;">

<div class="command">

<asp:ImageButton ID="btnFirst" runat="server" CommandName="First" ImageUrl="~/Images/btnFirst.PNG"

AlternateText="First Page" ToolTip="First Page" />

<asp:ImageButton ID="btnPrevious" runat="server" CommandName="Previous" ImageUrl="~/Images/btnPrev.PNG"

AlternateText="Previous Page" ToolTip="Previous Page" />

</div>

<div class="command">

<asp:TextBox ID="txtSlide" runat="server" Text='<%# gvKeywords.PageIndex + 1 %>'

AutoPostBack="true" OnTextChanged="txtSlide\_Changed" />

<ajaxToolkit:SliderExtender ID="ajaxSlider" runat="server" TargetControlID="txtSlide"

RaiseChangeOnlyOnMouseUp="true" Orientation="Horizontal" Minimum="1" Steps='<%# gvKeywords.PageCount %>'

Maximum='<%# ((GridView)Container.NamingContainer).PageCount %>' />

</div>

<div class="command">

<asp:ImageButton ID="btnNext" runat="server" CommandName="Next" ImageUrl="~/Images/btnNext.PNG"

AlternateText="Next Page" ToolTip="Next Page" />

<asp:ImageButton ID="btnLast" runat="server" CommandName="Last" ImageUrl="~/Images/btnLast.PNG"

AlternateText="Last Page" ToolTip="Last Page" />

</div>

<div class="PagerInfo">

<asp:Label ID="lblPage" CssClass="PagerInfo" runat="server" Text='<%# "Page " + (gvKeywords.PageIndex + 1) + " of " + gvKeywords.PageCount %>' />

</div>

</div>

</PagerTemplate>

</asp:GridView>

</ContentTemplate>

</asp:UpdatePanel>

<div class="FetchInfo">

<asp:Label runat="server" ID="lblSummary"></asp:Label>

</div>

</div>

</div>

</form>

</body>

</html>

Default.cs

using System;

using System.Collections.Generic;

using System.Text;

using System.Web.UI;

using System.Web.UI.WebControls;

using System.Linq;

namespace Margent

{

public partial class \_Default : System.Web.UI.Page

{

private static string \_connectionError = "An error occured while trying to fetch data from the DB.";

private static string \_emptyDataText = "There were not found words, associated to that query. We apologize for the inconvenience.";

private static string \_tooShortQuery = "Please enter a word, longer that two letters.";

private static Dictionary<DALWebCrawlerActive.Word, DataFetcher.CountFileList> \_resultsList = null;

protected void Page\_Load(object sender, EventArgs e)

{

//Set language

if (Request.QueryString["Language"] == "DE")

{

System.Threading.Thread.CurrentThread.CurrentUICulture = new System.Globalization.CultureInfo("de");

}

else if (Request.QueryString["Language"] == "BG")

{

System.Threading.Thread.CurrentThread.CurrentUICulture = new System.Globalization.CultureInfo("bg");

}

else

{

System.Threading.Thread.CurrentThread.CurrentUICulture = new System.Globalization.CultureInfo("en");

}

gvKeywords.RowDataBound += new GridViewRowEventHandler(gvKeywords\_RowDataBound);

gvKeywords.PageIndexChanging += new GridViewPageEventHandler(gvKeywords\_PageIndexChanging);

gvKeywords.RowCommand += new GridViewCommandEventHandler(gvKeywords\_RowCommand);

gvKeywords.EmptyDataText = \_emptyDataText;

btnDoSearch.Click += new ImageClickEventHandler(btnDoSearch\_Click);

lblSearch.Focus();

lblError.Visible = false;

lblError.Text = \_connectionError;

}

void btnDoSearch\_Click(object sender, ImageClickEventArgs e)

{

string query = tbSearchQuery.Text.Trim();

FetchData(query.ToLower());

}

void gvKeywords\_RowDataBound(object sender, GridViewRowEventArgs e)

{

if (e.Row.RowType == DataControlRowType.DataRow)

{

System.Collections.Generic.KeyValuePair<DALWebCrawlerActive.Word, DataFetcher.CountFileList> currentWord = (System.Collections.Generic.KeyValuePair<DALWebCrawlerActive.Word, DataFetcher.CountFileList>)e.Row.DataItem;

LinkButton lnkKeyword = (LinkButton)e.Row.FindControl("lnkKeyword");

ImageButton btnToggle = (ImageButton)e.Row.FindControl("btnToggle");

Label lblTotalCount = (Label)e.Row.FindControl("lblTotalCount");

lnkKeyword.Text = currentWord.Key.WordName;

lblTotalCount.Text = "[" + currentWord.Value.Count + "]";

Literal ltrl = (Literal)e.Row.FindControl("lit1");

ltrl.Text = ltrl.Text.Replace("trCollapseGrid", "trCollapseGrid" + e.Row.RowIndex);

string str = "trCollapseGrid" + e.Row.RowIndex;

lnkKeyword.Attributes["OnClick"] = "return OpenTable('" + str + "','" + btnToggle.ClientID + "');";

btnToggle.Attributes["OnClick"] = "return OpenTable('" + str + "','" + btnToggle.ClientID + "');";

GridView gvLinks = (GridView)e.Row.FindControl("gvLinks");

gvLinks.RowDataBound += new GridViewRowEventHandler(gvLinks\_RowDataBound);

gvLinks.PageIndexChanging += new GridViewPageEventHandler(gvLinks\_PageIndexChanging);

gvLinks.RowCommand += new GridViewCommandEventHandler(gvLinks\_RowCommand);

gvLinks.DataSource = currentWord.Value.FilesList;

gvLinks.EmptyDataText = \_emptyDataText;

gvLinks.DataBind();

}

}

void gvLinks\_RowDataBound(object sender, GridViewRowEventArgs e)

{

if (e.Row.RowType == DataControlRowType.DataRow)

{

DALWebCrawlerActive.File file = (DALWebCrawlerActive.File)e.Row.DataItem;

HyperLink lnkWebLink = (HyperLink)e.Row.FindControl("lnkLinkTitle");

Label lblLinkDescription = (Label)e.Row.FindControl("lblLinkDescription");

HyperLink lnkLink = (HyperLink)e.Row.FindControl("lnkLink");

lnkWebLink.NavigateUrl = file.URL;

if (file.Title != "")

{

lnkWebLink.Text = file.Title;

}

else

{

lnkWebLink.Text = file.URL;

}

if (file.Description != "/")

{

lblLinkDescription.Text = file.Description;

}

lnkLink.NavigateUrl = file.URL;

lnkLink.Text = file.URL;

}

else if (e.Row.RowType == DataControlRowType.Pager)

{

TextBox txtSlideLinks = (TextBox)e.Row.FindControl("txtSlideLinks");

txtSlideLinks.Text = (gvLinks.PageIndex + 1).ToString();

AjaxControlToolkit.SliderExtender ajaxSliderLinks = (AjaxControlToolkit.SliderExtender)e.Row.FindControl("ajaxSliderLinks");

ajaxSliderLinks.Steps = gvLinks.PageCount;

ajaxSliderLinks.Maximum = gvLinks.PageCount;

}

}

private void FetchData(string query)

{

if (query == "")

{

gvKeywords.Visible = false;

return;

}

if (query.Length < 3)

{

lblError.Text = \_tooShortQuery;

lblError.Visible = true;

gvKeywords.Visible = false;

return;

}

\_resultsList = DataFetcher.FetchResults(query);

if (\_resultsList != null)

{

if (\_resultsList.Count > 0)

{

lblSummary.Text = "fetch[" + DataFetcher.FetchTimeInSec.ToString("###0.00") + " sec]; sort[" +

DataFetcher.SortTimeInSec.ToString("###0.00") + " sec]; shown links[" + DataFetcher.ShownLinks + "]";

}

else

{

lblSummary.Text = "";

}

gvKeywords.PageIndex = 0;

gvKeywords.Visible = true;

lblError.Visible = false;

SetDataSource();

}

else

{

lblError.Visible = true;

gvKeywords.Visible = false;

}

}

private void SetDataSource()

{

gvKeywords.DataSource = \_resultsList;

gvKeywords.DataBind();

}

#region gvKeywords paging

protected void txtSlide\_Changed(object sender, EventArgs e)

{

TextBox txtSliderExt = (TextBox)gvKeywords.BottomPagerRow.Cells[0].FindControl("txtSlide");

gvKeywords.PageIndex = Int32.Parse(txtSliderExt.Text) - 1;

SetDataSource();

}

void gvKeywords\_PageIndexChanging(object sender, GridViewPageEventArgs e)

{

gvKeywords.PageIndex = e.NewPageIndex;

}

void gvKeywords\_RowCommand(object sender, GridViewCommandEventArgs e)

{

TextBox txtSliderExt = (TextBox)gvKeywords.BottomPagerRow.Cells[0].FindControl("txtSlide");

int pageIndex = Int32.Parse(txtSliderExt.Text);

switch (e.CommandName)

{

case "Next":

if (gvKeywords.PageCount > pageIndex)

{

txtSliderExt.Text = (pageIndex + 1).ToString();

gvKeywords.PageIndex = pageIndex;

SetDataSource();

}

break;

case "Previous":

if (pageIndex > 1)

{

txtSliderExt.Text = (--pageIndex).ToString();

gvKeywords.PageIndex = pageIndex - 1;

SetDataSource();

}

break;

case "Last":

if (gvKeywords.PageCount > pageIndex)

{

txtSliderExt.Text = gvKeywords.PageCount.ToString();

gvKeywords.PageIndex = gvKeywords.PageCount - 1;

SetDataSource();

}

break;

case "First":

default:

if (pageIndex > 1)

{

txtSliderExt.Text = "1";

gvKeywords.PageIndex = 0;

SetDataSource();

}

break;

}

}

#endregion

protected void gvLinks\_PageIndexChanging(object sender, GridViewPageEventArgs e)

{

((GridView)sender).PageIndex = e.NewPageIndex;

}

protected void gvLinks\_RowCommand(object sender, GridViewCommandEventArgs e)

{

GridView gv = ((GridView)sender);

TextBox txtSliderExt = (TextBox)gv.FindControl("txtSlideLinks");

int pageIndex = Int32.Parse(txtSliderExt.Text);

switch (e.CommandName)

{

case "Next":

if (gvKeywords.PageCount - 1 > pageIndex)

{

txtSliderExt.Text = (pageIndex + 1).ToString();

gvKeywords.PageIndex = pageIndex;

SetDataSource();

}

break;

case "Previous":

if (pageIndex > 1)

{

txtSliderExt.Text = (--pageIndex).ToString();

gvKeywords.PageIndex = pageIndex - 1;

SetDataSource();

}

break;

case "Last":

txtSliderExt.Text = gvKeywords.PageCount.ToString();

gvKeywords.PageIndex = gvKeywords.PageCount - 1;

SetDataSource();

break;

case "First":

default:

txtSliderExt.Text = "1";

gvKeywords.PageIndex = 0;

SetDataSource();

break;

}

}

protected void txtSlideLinks\_Changed(object sender, EventArgs e)

{

TextBox txtSliderExt = (TextBox)gvLinks.BottomPagerRow.Cells[0].FindControl("txtSlide");

gvLinks.PageIndex = Int32.Parse(txtSliderExt.Text) - 1;

SetDataSource();

}

}

}

DataFetcher.cs

using System.Linq;

using System.Collections.Generic;

using System.Collections;

using DALWebCrawlerActive;

using System;

namespace Margent

{

/// <summary>

/// Summary description for DataFetcher

/// </summary>

public static class DataFetcher

{

public static string ConnectionString = System.Configuration.ConfigurationManager.AppSettings["ConnectionStringActive"].ToString();

private static long tick1;

private static long tick2;

private static long tick3;

private static int \_totalLinksFound;

private static long \_shownLinks;

public class CountFileList

{

public int Count = 0;

public List<File> FilesList = new List<File>();

public CountFileList(int count, File fileToAdd)

{

Count = count;

FilesList.Add(fileToAdd);

}

public CountFileList(int count, List<File> fileList)

{

Count = count;

FilesList = fileList;

}

}

public static Dictionary<Word, CountFileList> FetchResults(string query)

{

IQueryable<DALWebCrawlerActive.WordsInFile> wordsInFiles;

Dictionary<Word, CountFileList> results = new Dictionary<Word, CountFileList>();

tick1 = DateTime.Now.Ticks;

\_shownLinks = 0;

try

{

string[] queryWords = query.Split(new char[] { ' ' }, System.StringSplitOptions.RemoveEmptyEntries);

using (DALWebCrawlerActive.WebCrawlerActiveDataContext dataContext = new DALWebCrawlerActive.WebCrawlerActiveDataContext(ConnectionString))

{

dataContext.CommandTimeout = 5;

wordsInFiles = from wif in

dataContext.WordsInFiles.Where(wif2 => (from wif in dataContext.WordsInFiles.Where(wif3 => queryWords.Contains(wif3.Word.WordName))

orderby wif.Count descending

select wif.FileID).Take(50).Contains(wif2.FileID)).Take(1000)

select wif;

\_totalLinksFound = wordsInFiles.Count();

tick2 = DateTime.Now.Ticks;

foreach (WordsInFile wif in wordsInFiles)

{

if (queryWords.Contains(wif.Word.WordName))

{

continue; // need improvment

}

if (results.Keys.Contains(wif.Word))

{

CountFileList cfl = results[wif.Word];

cfl.Count += wif.Count;

cfl.FilesList.Add(wif.File);

}

else

{

results.Add(wif.Word, new CountFileList(wif.Count, wif.File));

}

}

}

long tick4 = DateTime.Now.Ticks;

double a4 = TimeSpan.FromTicks(tick4 - tick2).TotalSeconds;

var orderedResults = from r in results.Take(100)

orderby r.Value.Count descending

select r;

Dictionary<Word, CountFileList> orderedResultsList = new Dictionary<Word, CountFileList>(100);

foreach (KeyValuePair<Word, CountFileList> kvp in orderedResults)

{

kvp.Value.FilesList = kvp.Value.FilesList.Take(100).ToList();

\_shownLinks += kvp.Value.FilesList.Count;

orderedResultsList.Add(kvp.Key, kvp.Value);

}

tick3 = DateTime.Now.Ticks;

return orderedResultsList;

}

catch

{

return null;

}

}

public static double FetchTimeInSec

{

get

{

return TimeSpan.FromTicks(tick2 - tick1).TotalSeconds;

}

}

public static double SortTimeInSec

{

get

{

return TimeSpan.FromTicks(tick3 - tick2).TotalSeconds;

}

}

public static int TotalLinksFound

{

get

{

return \_totalLinksFound;

}

}

public static long ShownLinks

{

get

{

return \_shownLinks;

}

}

}

}