Library Requirements Document

Marcel Timmerman <mt1957@gmail.com>

Abstract

This is a small document of requirements which I think is needed for this project. It describes the needed functionality, programs and structures.

The latest version of this document is generated on date 2017-01-10

Table of Contents

1. Introduction

Purpose of this project is to store meta information of objects which can be any type of documents, on disks or located on websites. This information is about the object in question and is meant to give some extra meaning to it. For example to use labels or keywords it is possible to group these objects together independent of the current location of these objects. As mentioned, possible objects can be anything, a short list:

- *File:* Files are objects with a content. E.g. text, image, xml, scripts, archives and code. For these type objects there are viewers and editors available.
- *Device:* These are objects with a special purpose and should not be manipulated. We can always say something about them so it might be worthwhile to add metadata for it.
- *Servers:* Objects with a content found on other servers. There should be a difference between local servers where the data can read via mountable filesystems and external servers where content can only be retrieved using protocols like http, ssh, ftp smtp, ldap etc. The data from external servers are copied to the local disk because this type of data also tend to be more fleeting in nature.
- *Special:* Objects can have a special meaning. Most of the time these can be resource files or configuration files. Examples of these are RDF, Html, SVG, vCard, calendar etc.

This document is an investigation of requirements needed for this program. It does not mention if this will be feasable or not. That will come in a later part.

2. Requirements

First some specifications to define the environment and products needed to run the programs.

- *Configuration*. This is a location where the program can find a config file. This file has e.g. info how to connect to the mongodb database server and what database and collections to use. On this location the program can store pid info of background programs and logfiles.
- *LIBRARY-CONFIG*. Environment variable holding the location on disk of the library configuration.
- *Database*. The data needed to save per object can be diverse and varying depending on its type. Sometime there are only a few field on other times a lot of extra fields are added, also which have different field names. A choice is made on this phenomenon to use a document based NoSQL database MongoDB. While developing, a standalone server is setup but can later become a replica server on serveral machines.

  An important aspect of data are its relations there might exist between objects. Here things like Rdf and Tupple comes into play. To describe those in a database another NoSQL typed database should be used also. An example of this Neo4j which is a network database.

2.1. General specifications of any object

Objects found anywhere are only descibed in this system. Its content will never be stored. Every object should always have a type. Some can be automatically assigned e.g. for files and directories. Others need help from the user of the system. E.g. a project description is not an object found on disk but is mostly a group of files together with other objects such as a server, websites or devices.

- There are three types of information to be stored, A) automatically found data such as ownership, path to document and volumename in case of documents, B) sha1 generated numbers based on the location and content of the object for searching and comparing, and C) explicitly provided information like keywords, name and address of owner, project name etcetera.
- Store, update or delete meta information in the database. This is for automatically retrieved or explicitly provided information.
- Search meta information using exact match or regular expressions on any part of the meta information.
- Displaying output from searches by commandline program or by webpage in a browser.
- Actions can be started using mimetypes also stored as metadata.
- Mimetypes are an important type of description method to show what can be done with the document. The list can also be used to start native applications to process a particular document. According to their mimetype of the document it mostly has also a proper suffix such as *.txt* or *.html* . See also [MIMETYPES] .

  A few examples are
  - *text/plain:* This is simple text format mostly created with simple text editors.
  - *audio/mpeg3:* A type of audio file with document suffix of *mp3* .
- General meta information to store. Besides the list below, users must be capable to add new metadata attributes.
  - *name:* Name of the object
  - *description:* Description of the object
  - *author:* A set of data like name and surname, address and email etc.
  - *datetime:* Date and time of retrieval, date and time of modification or current date and time.
  - *object-type:* Type of object such as document, directory or url.
  - *keys:* A list of keywords under which the object can be catagorized.
-
-

2.2. Document objects

1. This system will not manage documents. It will manage metadata about the documents. Location of the document is stored as part of this information.It is however nice when it can detect duplicates when another document is entered by the user. This duplication can be caused by backups or archives.
2. A document can be found on the local disk, externally connected disk, other computers and network devices such as network attached storage (NAS), media stores or on webservers.

3. As a side effect of locating documents on e.g. external servers, these documents can be stored on disk for offline use.
4. Document meta information to store.
   - *full-name:* Complete and absolute path to the document
   - *file-name:* Name of document object
   - *extension:* Extention of the document. This is empty for directory documents.
   - *accessed:* Date and time of last access.
   - *modified:* Date and time of last modification.
   - *changed:* Date and time of last change.
   - *size:* Size of document.
   - *:*
   - *:*
   - *:*

2.3. Website information

1. A file on a disk is pointed to by a name and path and a drive when working on windows. There are other ways to get to a document like using a unified resource locator (URL).
2. Protocols are used to get to the document before processing it. E.g. the *http* protocol is used to get a webpage from a site on the network and *file* is used often to get a document from the local filesystem. See also [MIMETYPES] .

   The following list is a series of protocols which might be supported.
   - *file:* Protocol to get documents from a filesystem.
   - *http* and *https:* Protocols to get webpage documents from a web server.
   - *ftp:* File transfer protocol.
3. Web meta information to store.
   - *uri:* Url, uri or iri

2.4. Other information

Other information besides meta information can be imported such as agendas and contact information.

1. Contact information can be imported from vcard files. This data can also be linked to other meta items.
2. Relations between objects are stored in the database using directions of Topic Maps (TM). Inport and export are done via XML as XTM or encapsulated in RDF .
3. Web Ontology Language (OWL). Relations defined above with TM can be tested using a reasoner reading this ontology information. The rules for this language can be imported and exported as OWL/XML documents or as RDF.

3. Implementation

This software package should come with several modules and programs to suit several ways of accessing the data. There is also an issue of making the software platform independent so everyone can be happy with it.

- *The programming language* The first item to think about is the choice of programming language. A scripting language would be a proper choice because these languages have a higher level of coding and will access the underlaying system in a platform independent way. The language I want to choose is perl6. Yes, the still unfinished perl version. I am very confident that the language gets its first release this year(2015) and wanted to learn about the language by doing this project.

  The second approach is to use a browser to do the work. There we can use *html5* , *css3* and *javascript* and libraries. There is also a server side scripting which can be any of *perl6* , *perl 5* or javascript by means of *(nodejs)* . There are also a great many javascript modules which can be used.

- *The storage method* Because the information items on one object can be different than on the other a hiërargycal database would be the choice. MongoDB is a dayabase for which there is support from javascript as well as perl6.
- *Storage* The name of the database and the names of the collections

4. Dependencies

The program will be depending on several modules and programs. That is only logical because we do not want to reinvent the wheel(s) again do we? We only try not to select those software which will bind it to some platform as explained above.

- *perl6:* The followup version of perl 5.*. The program is not yet completely finished but will be soon (2015-01). This program is a interpreter/compiler which can compile the script into some intermediary
- *perl6:* The followup version of perl 5.*. The program is not yet completely finished but will be soon (2015-01).

5. State of affairs

A list of programs and web pages created and made available for use. While the project is still in a pristene state there presumable are several bugs left behind. Also things in the database and programs might change when other ideas arrive. Below there is a list of what has been made. For documentation see library [file:///home/marcel/Languages/Perl6/Projects/Library/doc/library.pdf].

1. The mongo database is *Library:* with several collections.

   - *object_metadata:* Collection to store meta information of any object found.

   - *mimetypes:* Collection to store mimetype information. This can be connected to the object-type.

   install-mimetypes.pl6 is program to install mimetype information from a certain website [http://www.freeformatter.com/mime-types-list.html].

   store-file-metadata.pl6 is a program to insert or modify metadata of files and directories in the database.

5.1. Priorities

Bibliography

[MIMETYPES] *A list of all mimetypes*. http://www.sitepoint.com/web-foundations/mime-types-complete-list/