Library Requirements and Design

Abstract

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

Introduction

Purpose of this project is to store meta information of objects. An object can be any type of document, url, project, contact etc. This information is about the object in question and is meant to give some extra meaning to it. For example when labels or keywords are used in the meta data, it is possible to group objects together when a value of the key or label is the same over all the objects of the same group. This information must be independent of the current location (if any) of an object.

A short list of possible objects:

- Files are objects with a content. E.g. text, image, xml, scripts, archives and code. For most of these type objects there are viewers and editors available. Besides its content like mp3, svg etc, a language can be specified e.g. C, Perl or Markdown. Also its purpose is important. Most of the time these can be resource files or configuration files. Examples of these are; rdf, html, vCard, calendar etc.
  The files can be found locally or on another server. While the purpose of the project is not to manage files, it must be possible to download files for offline use or archiving. The external document might be removed from servers as time passes by, so that is another reason to download. The content can be retrieved using various protocols among which http, ssh, ftp, smtp and ldap.
- Directories are containers for files which are grouped together. Meta data could describe the purpose of the container. It might be the root of a project or a directory for libraries.
- Servers are objects of which the meta data can describe its services.

Requirements

Environment

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

- LIBRARY-CONFIG. Environment variable holding the root directory location on disk

where the library program data is stored. Here the program can store pid info of background programs and log files. Also the configuration files are stored here. If the database server is local, its data should come here too. Perhaps a slave server in a replicaset.

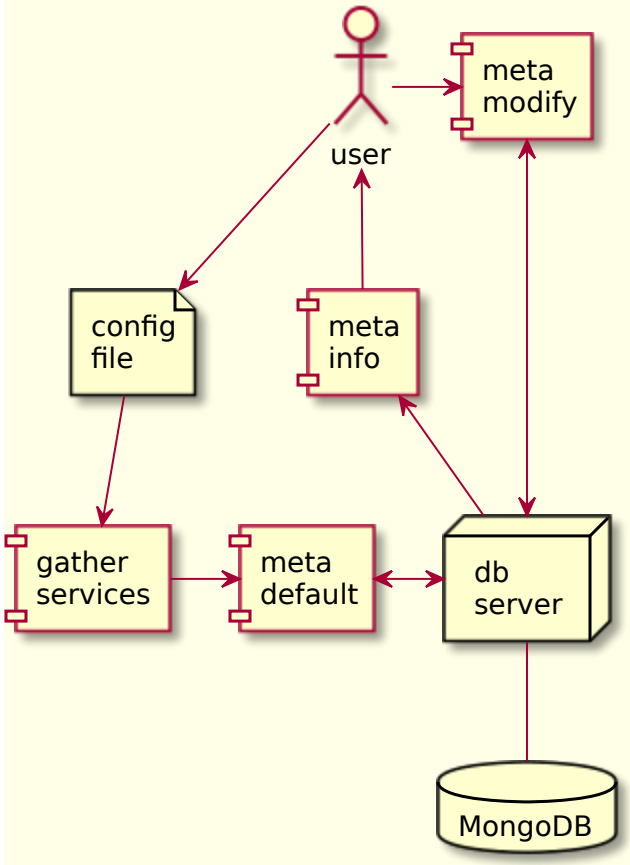
- Configuration file. This file holds information on how to connect to the mongodb database server and what database and collections to use amongst others.
- Database. The data needed to save for each object can be diverse and varying depending on its type. Sometimes, there are only a few fields, on other times a lot of extra fields are added. They might also have different field names. A choice is made, based 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 several machines.
- An important aspect of data are possible relations 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 is **Neo4j** which is a network database.

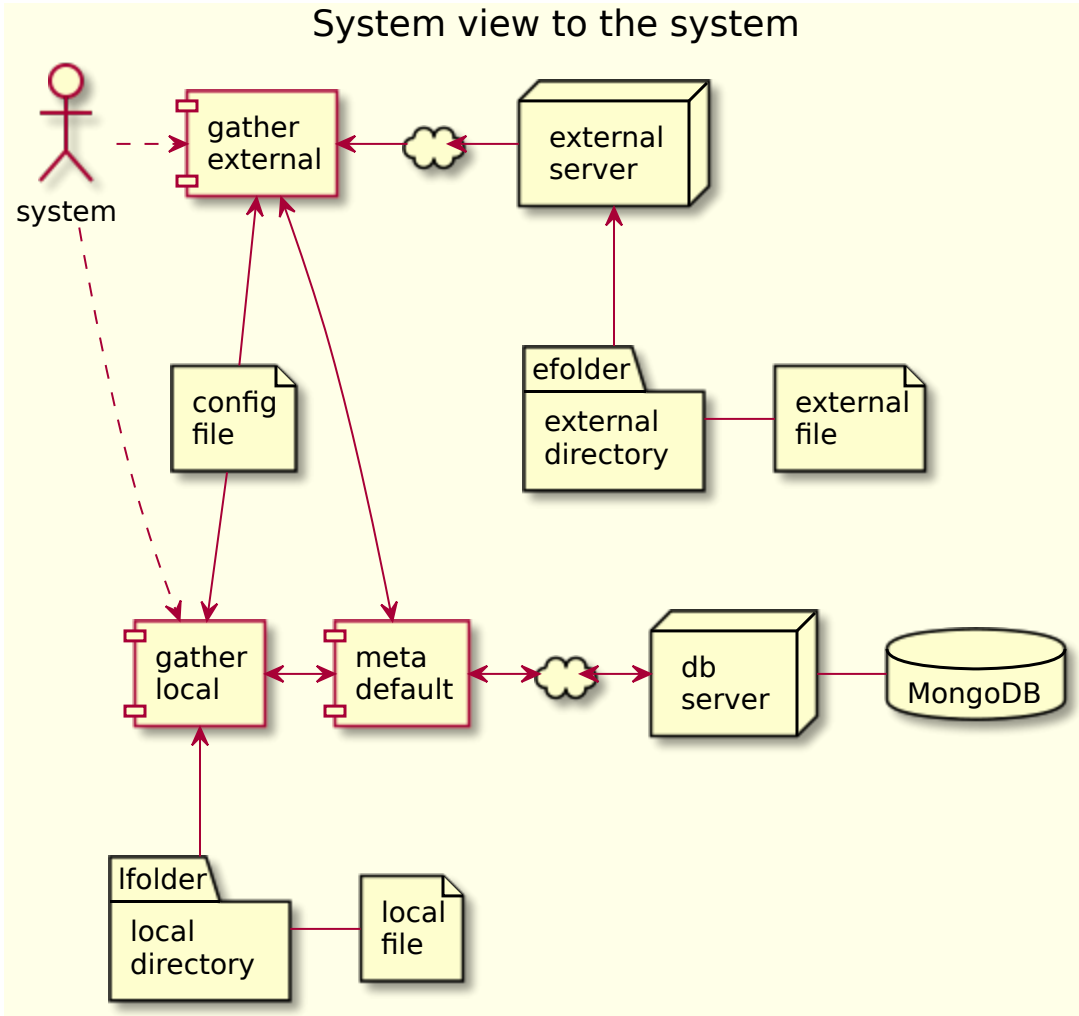
Simplified view from user

Two diagrams to show how the user interacts with the system. The important points are

- the user changes the configuration by editor or using other components of the system.
- the system starts the gathering processes.
- the gathering processes use the configuration to know what to search for. Then the data is examined to set default meta information and sends it to the database. These processes are also checking for possible changes in the file system like renamed or moved files.
- the user is able to modify the meta data by adding keys or modify those keys.

User view to the system





General specifications of an object

Objects found anywhere are only described by system. Its content will never be changed. External objects may be copied to the local system. 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
    - Automatically found data such as ownership, path to document and volume name in case of documents
    - Sha1 generated numbers based on the location and content of the object for searching and comparing. This can help to find e.g. a renamed file and attach the meta information already in the database. Only the general info need to be modified. Actions which could take place are

            - Move a file from one place to another on the file system.
            - Rename a file
            - Rename and move in one operation
            - Modify its contents
            - Modify ownership or access rights
            - Remove the file
            - Modify URI
        - 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.

Document objects

- This system will not manage documents. It will manage information 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.

- 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 web servers.
- As a side effect of locating documents on e.g. external servers, these documents can be stored on disk for offline use.

URL Website information

A file on a disk is pointed to by a name and path and a alse a drive when working on windows. There are other ways to get to a document like using a unified resource locator (**URL**).

Protocols are used to get to the document before processing it.
E.g. the **http** protocol is used to get a web page from a site on the network and **file** is used often to get a document from the local file system. 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 web page documents from a web server.
- ftp: File transfer protocol.

Also here, as a side effect of locating documents on external servers, these documents can be stored on disk for offline use.

Other information

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

- Contact information can be imported from vcard files. This data can also be linked to other meta items.
- Relations between objects are stored in the database using directions of Topic Maps ($abbrev[TM]). Import and export are done via
- XML as XTM or encapsulated in RDF.
- 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.

Summary of fields

Generic meta data fields

- 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.

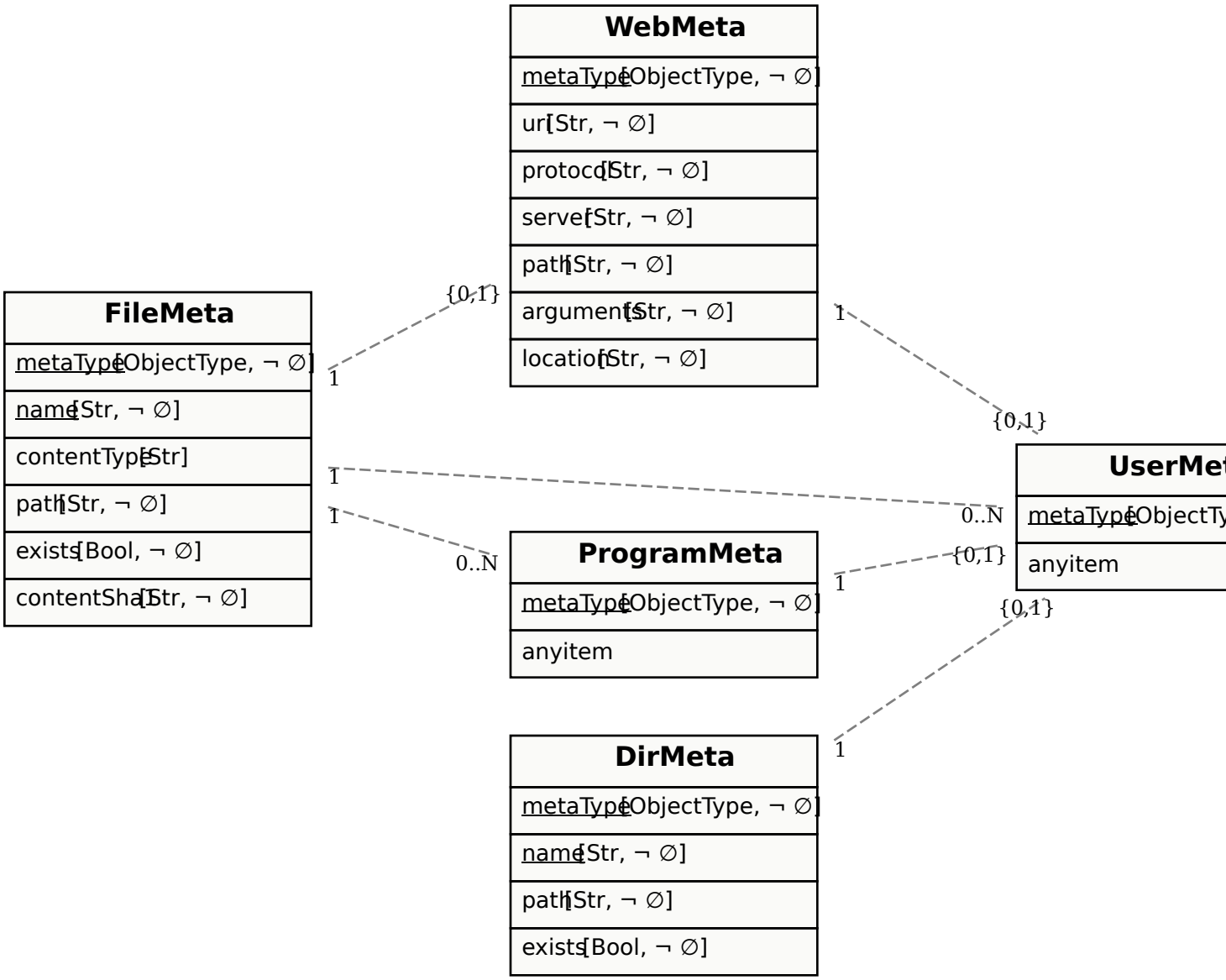
Document meta data fields derived from files and directories

- 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.
- location: Place where document is downloaded

Web meta information to store.

- uri: Url, uri or iri
- protocol: Name of used protocol
- server: Name of server
- path: Path of document
- arguments: key-value pairs found on the url
- location: Place where document is downloaded

Collections and sub documents. Types are perl6 types



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

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

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.

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 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.

Priorities

Design

Overview