DMetaG – Graphical Views of DMeta Data

# Introduction

DMetaG is a web-based application that can be used to query and view DMeta (Dolphin Metadata) data in hierarchical graphs. DMetaG does not have a database of its own, relying instead on the metadata definitions and associated data entries in the DMeta database. This allows it to be used for virtually any type of data, since the structure and relationship between data types is ascertained from the DMeta definitions. In this document the examples are taken from a DMeta installation with patients, samples, and associated data.

# Login

Before starting work with DMetaG a login must be done. Login is done using DMeta credentials since DMetaG has no accounting information of its own and the login is directed to the DMeta server. At the time of login all needed metadata definitions are read and cached with the client. If the definitions ever need to be refreshed logging in again will force a refresh to occur.

Graphical user interface, application

Description automatically generated

DMetaG login

# Find and Graph Displays

At the heart of DMetaG is the find command, which allows users to select entries to be displayed in a graph. When the Find command is first selected from the menu a form is displayed to ask what type of entry is being searched for. The entry types displayed are a list of all the collection types found in the Dmeta installation.

A picture containing graphical user interface

Description automatically generated

Find Entry Select List for DMeta Project

Once the wanted entry type is selected a simple form, with all fields for the entry type, is displayed.

Graphical user interface, application

Description automatically generated

Find query for Patient

To refine the selection any number of the fields can be filled in with criteria to find the wanted entry. If a single entry is found matching the specified criteria, then the graph for the entry is immediately displayed. If more than one entry matches, then a table of available choices is displayed. Choosing one of the rows displayed brings up the graph for the chosen entry.

Table

Description automatically generated

Table display for all patients with Gender M

When a graph is displayed the chosen entry is made the focus of the graph, with all parents and children of the entry displayed. Below, the graph is a form with the field data for the chosen entry. Within the graph the path to the entry is highlighted.

Diagram

Description automatically generated

Graph Display for Patient CB019

The field data for the chosen entry can be modified using the form below the graph by changing the data as wanted and choosing the submit button.

To change the focus of the graph one needs to simply click on a node in the graph to create a new graph, with the selected entry as the focus. For example, if Biosamples “CB019\_H2\_V1\_Bst” is chosen the following graph is displayed.

Graphical user interface

Description automatically generated with medium confidence

Graph Display for BioSamples CB019\_H2\_v1\_Bst

In the new graph the focus has been changed to the Biosamples entry, as shown by the highlighted path. Also Experiment Series and Experiments entries are now displayed since they are parents of Biosamples, but were not parents of the Patient.

Right-clicking on any entry in the graph brings up a context menu that allows addition of new subentries, as well as moving and deletion of the chosen entry. The entry type for each operation is automatically set as appropriate for the parent/child relationships of the chosen entry. For example, only Samples’ entries, which is a child of BIosamples, can be added to a BIosample. Similarly, a Biosamples entry can only be moved to an Experiments or Patient Visit entry since those entry types are direct parents of Biosamples.

Graphical user interface

Description automatically generated

Biosamples context menu

If move is chosen a list of places to move the sample is displayed. Each possible destination is a fully qualified label, including the primary identifying fields of all parents. Parents at the same level are separated by “||” and each generation of parents is enclosed by parenthesis, with the top-level parents listed first. Following is the list displayed when a move of the Samples entry is requested.

A picture containing diagram

Description automatically generated

Samples Move list

When delete is chosen from the context menu the chosen entry and all its children are deleted. Before the delete occurs, a table listing all the entries to be deleted is displayed, and the delete can be either cancelled or confirmed.

Diagram

Description automatically generated

Delete of Biosamples

If confirmed (delete button is selected) the delete happens immediately and is not reversable except via direct access to the DMeta database.

# Find Barcode

Find Barcode is a special purpose command to look for entries across multiple entry types. Find Barcode requests a barcode, which can be input using either a barcode scanner or typed directly.

Graphical user interface, text, application, chat or text message

Description automatically generated

Find Barcode

Upon submission of the barcode all entry types that contain a “Barcode” field are searched for the entered barcode. The barcode can be entered manually, but if unique barcodes are attached to storage containers, such as plates or tubes, the barcodes can simply be scanned to display the graph and entry for the container’s contents. Following is an example of the results of a Find Barcode command for a Biosample stored in a -80C freezer.

Graphical user interface

Description automatically generated

Find of Biosamples by Barcode

Associating both storage locations and barcodes with samples can facilitate finding sample containers. Once the storage location for a container is known, the barcodes of the containers in that location can be scanned to find the one matching the wanted sample.

# About

The About command shows the relationships between all the entry types found in the DMeta database. The relationships for the project used in the previous examples are shown in the following About display.

Diagram

Description automatically generated

About display

# Internals

DmetaG has a small server, used exclusively to deliver the single page application and associated resources to the client. As such it has two requests: “dmetag” to deliver the single page client application, and “resource” to access resource files used to support the client application. Resource files include “justcontext” css and js files, to help display the context menus, and optionally the client scala.js source files that can be helpful when debugging the client application’s javascript code. The server is based on Akka Http (<https://doc.akka.io/docs/akka-http/current/index.html>) but given its limited functionality could easily be transported to any server side infrastructure.

The source code for the single page client application is written in scala.js (<https://www.scala-js.org>) which is compiled into a single javascript file placed in the server’s resource area.

# Installation

Download the sources via

git clone <https://github.com/nnovod-umms/DmetaG.git>

To build “cd” to the top-level directory where the sources are downloaded. The file build.sbt should be located there.

DmetaG is built using the sbt build tool (<https://www.scala-sbt.org/>). To run DmetaG first build the client javascript file with the following command:

sbt "dmetagClient / fastOptJSBundle"

That builds the client javascript file, copies it into the server’s resources area as “dmetagclient.js”, and also copies the scala.js sources into a the resource’s subdirectory dmetaclientSources to allow for debugging in the client browser using the original scala.js files. Alternatively, to build a fully optimized javascript file that is both smaller and potentially has better performance, use the command:

sbt "dmetagClient / fullOptJSBundle"

To run the server give the command:

sbt "dmetagServer / run"

The server runs on localhost, port 8080 by default. Those settings are in the file Server.scala file in the directory dmetagServer/src/main/scala/edu/umassmed/dmetag.

The client attaches to Dmeta via the URL set in the file URLs.scala in the directory dmetagClient/src/main/scala/edu/umassmed/dmetag/dmeta. By default the URL for Dmeta is set to run with a local Dmeta instance: <http://localhost:4000/api/v1>

# Things to do

* Include DmetaG in Dolphin SSO
* Add javascript calls in client to allow login, find, find barcode, add, delete, and move separate from the UI. Then a different UI can be layered on top if wanted.
* Have external config files for server settings (e.g., URLs)
* Have settings command to reset client settings (e.g., DMeta URL)