Contents

[File Schema and functionalities 2](#_Toc320529799)

[Principal functions and classes 3](#_Toc320529800)

[Jslib.js 3](#_Toc320529801)

[Function objetoAjax() 3](#_Toc320529802)

[Function autentication() 3](#_Toc320529803)

[Function go\_to\_page(page) 3](#_Toc320529804)

[Function add\_to\_page\_said(conf,module,page,dbId,name,des,dashId) 3](#_Toc320529805)

[Function chk\_characters(checkStr) 3](#_Toc320529806)

[Function validate\_send(form,msg1,msg2) 3](#_Toc320529807)

[Mainlib.php 3](#_Toc320529808)

[class permission 3](#_Toc320529809)

[class sqlImport 3](#_Toc320529810)

[class file\_vars 3](#_Toc320529811)

[function get\_user\_id($CFG,$username) 3](#_Toc320529812)

[function show\_menu($CFG,$username) 3](#_Toc320529813)

[function make\_safe($variable) 3](#_Toc320529814)

[function getUserType($uType) 3](#_Toc320529815)

[function is\_table($tablename) 3](#_Toc320529816)

[function delete\_folder($dir, $borrarme) 3](#_Toc320529817)

[function is\_a\_ip($ipdir) 3](#_Toc320529818)

[function is\_a\_email($email) 3](#_Toc320529819)

[function return\_date($date) 3](#_Toc320529820)

[function MongoConnect($username, $password, $database, $host) 3](#_Toc320529821)

[GLASS MySQL databases 3](#_Toc320529822)

[glass\_dashboard 3](#_Toc320529823)

[glass\_ddbb 3](#_Toc320529824)

[glass\_modules 3](#_Toc320529825)

[glass\_myview 3](#_Toc320529826)

[glass\_permision 3](#_Toc320529827)

[glass\_settings 3](#_Toc320529828)

[glass\_user 3](#_Toc320529829)

[New modules development 3](#_Toc320529830)

[GLASS Scheme in MongoDB 3](#_Toc320529831)

# File Schema and functionalities

GLASS is constituted by a set of files which contains the scrip needed to run the tool. These files are dividend according to the curried out operation. It has tried to keep the most possible logical structure.

The files set are as follows:

* **Root folder:** here are the configuration files, start-up features and no very complex features. The following describes each:
  + **Login:** it includes *access.php* (which is accessed via ajax) and *index.html* files. It allowed logging into the tool, to linking with LDAP and creating user if it is the first time he makes login and starring the web installation process if it is not.
  + **Logout:** Allows the tool to disconnect the session. It is *logout.php* file.
  + **Configuration variables:** Contains configurations variables of the tool. It is *config.php* file. If there a problems in the installation process you can edit the values in this file to ensure the proper functioning of the tool. The file *install.sql* of the folder *install* contains the SQL statements needed to install the tool. For any kind of database error, take a look at this file.
  + **Documentation:** script PHP (*credits.php and documentation.php*) which links to the platform documentation and group reference.
  + **Home page:** it is the *home.php* file. This file manages the dashboard content.
  + **GLASS settings:** it is the *settings.php* file. This file links to *install* folder witch take all the configuration and installation process.
* ***Install* folder:** Contains the files needed to install the tool, both in the database as settings server variables. *Installprocess.php* file is responsible for this task. This file edits the content of *config.php* and *install.sql* addition to performing the last script in MySQL database. Also in this folder contains all the configuration process of the tool that can be viewed within the menu section of GLASS.
* ***Fview* folder:** Contains all necessary script to shown the GLASS section of my favourite views.
* ***Lib* folder:** it consist of a set of files that stored processes that can be used by developers. Later will explain the different functions and classes of some of the files. These functions in turn call other files in this folder, especially those whose HTML or JavaScript files that use Ajax. For example, there is a function that generates the menu which calls the file *menu.html*.
* ***Modules* folder:** Their files are responsible for managing the different modules and their installation on the database.
* ***Resources* folder:** Here are located the files you want to share. The new module scheme is also stored in this folder as it is discussed in the creation of new modules.
* ***Themes* folder:** Inside this folder are the folders in change or graphic themes of the platform. Although a theme selection process is not implemented, if it is added only would need to interact on one of these folders to change the theme. Thus for only see a folder called *classic* that included all the images and cascading style sheets files.
* ***User* folder:** This folder contains the necessary scripts to manage users and privileges.
* ***Visualization* folder:** here is where you have to copy the new modules as it is discussed in the creation of new modules.

# Principal functions and classes

GLASS mainly uses two files to manage the functions and classes. On one it is *mainlib.php* where is located the PHP functions and classes and on the other hand it is jslib.js which meets global JavaScript functions. JavaScript functions often use Ajax so it usually uses complementary PHP files.

## Jslib.js

The functions included are the followings:

### Function objetoAjax()

Create an Ajax objet.

Incoming variable:

* No

Returns:

* The created objet.

### Function autentication()

It manages the authentication process via Ajax. It uses *access*.*php* file to manage authentication, check if the tool is installed, creating a new user profiles, etc. It shows the give back message in a div whose id was error. It needs two variables: *user* and *pass*. These variables must be in a form called login.

Incoming variable:

* No.

Returns:

* No.

### Function go\_to\_page(page)

Load the selected page.

Incoming variable:

* Page: string which save the page to load.

Returns:

* No.

### Function add\_to\_page\_said(conf,module,page,dbId,name,des,dashId)

Add an element to the dashboard or to my favourite views. It shows the give back message in a div whose id was msg-info.

Incoming variable:

* conf: Json string with the configuration of the view.
* module: name of the folder where are the module files. Only the folder name that we found in the *visualization* folder.
* page: page where the *dashboar.php* or *myview.php* is located.
* dbId: Id of mongo database.
* name: the name that we set. It is only used to my favourite view not to the dashboard. For this last case set it to null.
* des: description of the favourite view, it is not used to the dashboard. For this last case set it to null.
* dashId: dashboard id or my favourite view id for update it. If you want to add a view to the dashboard or to your favourite view you can set this value to null.

Returns:

* No.

### Function chk\_characters(checkStr)

Cheek if you are using characters allowed.

Incoming variable:

* checkStr: string which the information to check.

Returns:

* true: if it is ok.
* false: if it is not ok

### Function validate\_send(form,msg1,msg2)

Cheek the text field of an input.

Incoming variable:

* form: the object which represent the form.
* msg1: message to show if the field is empty.
* msg2: message to show if some char is not correct.

Returns:

* true: if it is ok.
* false: if it is not ok.

## Mainlib.php

The class included are the followings:

### class permission

Class which save or load the user permits in the database.

Attributes

* $userName: the name of the user.
* $userType: user model.
* $userViewLevel: user level.
* $userModifyPermision: user permit modification
* $userTypeChange: permission to change a user type
* $moduleInstall: permission to install modules
* $importView: permission to import view
* $varSettings: permission to change the configuration variables.
* $addBBDDCAM: permission to add a new database
* $download: permission no download
* $viewUser: permission to view users
* $viewSuggest: permission to suggest metadata

Methods

* \_\_construct: needs glass configurations variables and the user name and establish the values of the attributes.
* set\_DB\_userViewLevel: save the value that indicates its name in the database. Needs glass configurations variables.
* set\_DB\_userModifyPermision: like the previous.
* set\_DB\_userModifyPermision: like the previous.
* set\_DB\_moduleInstall: like the previous.
* set\_DB\_importView: like the previous.
* set\_DB\_varSettings: like the previous.
* set\_DB\_addBBDDCAM: like the previous.
* set\_DB\_download: like the previous.
* set\_DB\_viewUser: like the previous.
* set\_DB\_viewSuggest: like the previous.
* set\_userType: set this attribute.
* set\_userViewLevel: set this attribute.
* set\_userModifyPermision: set this attribute.
* set\_userTypeChange: set this attribute.
* set\_moduleInstall: set this attribute.
* set\_importView: set this attribute.
* set\_varSettings: set this attribute.
* set\_addBBDDCAM: set this attribute.
* set\_download: set this attribute.
* set\_viewUser: set this attribute.
* set\_viewSuggest: set this attribute.
* get\_userType: get this attribute.
* get\_userViewLevel: get this attribute.
* get\_userModifyPermision: get this attribute.
* get\_userTypeChange: get this attribute.
* get\_moduleInstall: get this attribute.
* get\_importView: get this attribute.
* get\_varSettings: get this attribute.
* get\_addBBDDCAM: get this attribute.
* get\_download: get this attribute.
* get\_viewUser: get this attribute.
* get\_viewSuggest: get this attribute.

### class sqlImport

Load a SQL file to install the necessary tables and values.

Attributes

* $ErrorDetected: Check if there is an error.
* $CodigoError: save the error code.
* $TextoError: save the string error message.

Methods

* sqlImport: constructor. Need user, password, host and the SQL file. Retrieving the vars. suprime los comentarios procedentes del archivo en formato texto and lo devuelve
* is\_comment: difference if comments. Needs the file in text format and return it without comments.
* dbConnect: establish the connect with the database.
* import: import process.
* ShowErr: print errors that may have taken place

### class file\_vars

Change the variable value of *config.php and install.sql*.

Attributes

* $data: content of the file.
* $file: file name.

Methods

* construct: reads the file then it needs it.
* replace\_PHP\_valuere: replace the values of a PHP file. Needs the variable name and the new variable value.
* replace\_SQL\_value: replace the values of a SQL file. Needs the word to replace and the new word.
* update\_file: update the file with the new value set.
* set\_file: set the name of the file.

The functions included are the followings:

### function get\_user\_id($CFG,$username)

Get the user id.

Incoming variable:

* $CFG: object with the configurations variables of the platform. These variables are included in *config.php*.
* $username: name of the user to find.

Returns:

* Id of the user.

### function show\_menu($CFG,$username)

Prints the menu

Incoming variable:

* $CFG: object with the configurations variables of the platform. These variables are included in *config.php*.
* $username: name of the user.

Returns:

* No

### function make\_safe($variable)

Quote string with slashes and strip whitespace (or other characters) from the beginning and end of a string

* $variable: variable to analyse.

Returns:

* The same variable with changes.

### function getUserType($uType)

Convert user type specified as a number to a text

Incoming variable:

* $uType: number which represents the user type.

Returns:

* String which represents the user type.

### function is\_table($tablename)

Look for if a table exists in the database

Incoming variable:

* $tablename: MySQL table to find.

Returns:

* true: if the table exist.
* false: if the table not exit.

### function delete\_folder($dir, $borrarme)

Delete a folder and its files.

Incoming variable:

* $dir: folder to delete.
* $borrarme: set true if you want to delete the folder and false if you only want to delete its files.

Returns:

* No.

### function is\_a\_ip($ipdir)

Check if $ipdir is an IP direction.

* $ipdir: IP direction to check.

Returns:

* true: if it is an IP direction.
* false: if it is not an IP direction.

### function is\_a\_email($email)

Check if $email is an email direction.

* $email: email direction to check.

Returns:

* true: if it is an email direction.
* false: if it is not an email direction.

### function return\_date($date)

Check if $date is a correct date string.

* $date: date string to check.

Returns:

* true: if it is a date string.
* false: if it is a date string.

### function MongoConnect($username, $password, $database, $host)

It establishes a connection with a mongo database. If an error occurs return the following string: “error: impossible to connect!”

* $username: Mongo database user name.
* $password: Mongo database user password.
* $database: Mongo database name.
* $host: Mongo host.

Returns:

* Object connection.

# GLASS MySQL databases

GLASS uses a MySQL database to operate itself. So it can save the setting, save user preferences, manage modules, etc. The database entity relationship diagram is the following:



The following will detail the structure for each table.

## glass\_dashboard

This table save the necessary information to print each widget in the dashboard.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Field** | **Type** | **Null** | **Default** | **Comments** |
| id | int(11) | No |  | Auto-incremental. |
| userId | int(11) | No |  | Id in glass\_user. |
| moduleId | int(11) | No |  | Id in glass\_modules. |
| widgetconf | varchar(500) | Yes | NULL | Here goes the necessary information to print the widget. It should be a Json string. |
| bdCAMid | int(11) | No |  | Id in glass\_ddbb. |
| pos | int(11) | No |  | It is the position in the dashboard. |

## glass\_ddbb

This table save the different Mongo database that GLASS will analyse.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Field** | **Type** | **Null** | **Default** | **Comments** |
| id | int(11) | No |  | Auto-incremental. |
| host | varchar(30) | No |  | Host direction. |
| name | varchar(30) | No |  | Database name. |
| user | varchar(30) | No |  | User name . |
| pass | varchar(30) | No |  | Database password. |
| filters | varchar(500) | No |  | Json string with the selected filters. |
| description | varchar(500) | Yes | *NULL* | Database description. |

## glass\_modules

This table save the necessary information that a module must have.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Field** | **Type** | **Null** | **Default** | **Comments** |
| id | int(11) | No |  | Auto-incremental. |
| name | varchar(30) | No |  | Module name. |
| folder | varchar(30) | No |  | Module folder in *visualization* folder. |
| description | varchar(200) | Yes | *NULL* | Module description. |

## glass\_myview

This table save the necessary information to can save a view in my favourite views.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Field** | **Type** | **Null** | **Default** | **Comments** |
| id | int(11) | No |  | Auto-incremental. |
| name | varchar(60) | No |  | My favourite view name. |
| userid | int(11) | No |  | Id in glass\_user. |
| moduleId | int(11) | No |  | Id in glass\_modules. |
| bdCAMid | int(11) | No |  | Id in glass\_ddbb. |
| pos | int(11) | No |  | It is the position in the my favourite view list |
| widgetconf | varchar(500) | No |  | Here goes the necessary information to print the widget. It should be a Json string. |
| description | varchar(500) | Yes | *NULL* | My favourite view description. |

## glass\_permision

This table save GLASS permission.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Field** | | **Type** | **Null** | **Default** | **Comments** |
| userType | | varchar(30) | No |  | User model |
| userViewLevel | | int(11) | No |  | User level |
| userModifyPermision | | Binar | No |  | User permit modification |
| userTypeChange | | Binar | No |  | Permission to change a user type |
| moduleInstall | | Binar | No |  | Permission to install modules |
| importView | | Binar | No |  | Permission to import view |
| varSettings | | Binar | No |  | Permission to change the configuration variables. |
| addBBDDCAM | | Binar | No |  | Permission to add a new database |
| download | | Binar | No |  | Permission to download |
| viewUser | | Binar | No |  | Permission to view user |
| viewSuggest | Binar | No |  | Permission to suggest metadata |

## glass\_settings

This table save the necessary information to print each widget in the dashboard.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Field** | **Type** | **Null** | **Default** | **Comments** |
| id | int(11) | No |  | Auto-incremental. |
| ddbbId | int(11) | No |  | Id in glass\_ddbb. |
| userId | int(11) | No |  | Id in glass\_user. |
| dbcol | int(11) | No |  | Number of columns that user wants in his dashboard. |

## glass\_user

This table save users information.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Field** | **Type** | **Null** | **Default** | **Comments** |
| userId | int(11) | No |  | Auto-incremental. |
| name | varchar(30) | No |  | User name. |
| userType | varchar(30) | No |  | User type in glass. It is used to assign permissions. |

# New modules development

A basic structure of a module can be found inside the resources GLASS folder. To begin designing a new module only need make a copy of this folder, put a name that is unique, i.e. it is not repeated with other folder listed in *visualizations*, open all module files to change the name to those variables that tell us so and copy it in the GLASS *visualization* folder. GLASS automatically detects the presence of new modules and will be ready to install in management modules option of the tool.

Each module consists of a set of files which have unique functionality. Comments are included in each of these to know the characteristics and peculiarities of each. It is very important to follow the tips contained and change the name listed by the name of the module. A module consists of following files and folders:

* **File *index.php*:** Contains all necessary PHP code to make the visualizations. The *mainlib.php* library is included. It has all global PHP class and function that can be useful. Also jslib.js is included which contains the JavaScript global functions.
* **File *index.html*:** Contains all necessary HTML code to make the visualizations. This is where necessary code must be entered for a new visualization. This file is called from *index.php.*
* **File *info.txt*:** It is very important to complete this file before installing the module in GLASS because here is where the necessary information is included in a new module that demonstrates information for someone wanting to install the module to be developed.
* **File *share.php*:** Included code to share our favourite module in my views option of GLASS.
* **File *widget.html***: Contains all the necessary HTML code to print the form widgets. This file is called form *widget.php*.
* **File *widget.php*:** Contains all the necessary PHP code to print the form widgets.
* **Folder *lang*:** It is included all files that define strings that can be shown in the module. Each file listed in the folder corresponding to one language. It should be defined in English and Spanish.
* **Folder *css*:** include files needed to define the cascade style of our module. *style.css* has been defined as first file and it is called form *index.php*. Can be included as many files as you like.

Because it is a development version of the tool, you may be improvements occur, you find errors, etc… For this or for any question or problems please contact with the GLASS development group.

# GLASS Scheme in MongoDB

GLASS defined two collections (JSON structures) to interpret data, one for events and other for user. On the one hand, events are stored as a Mongo document according to the JSON structure, which can be seen below. It is composed of 3 main required fields that must be maintained above nomenclature, and many optional fields as needed:

* **datetime:** date and hour of the event.
* **name:** event type captured.
* **user:** entity referred to in the event. Because of that it may be possible that an event is triggered by several users this field is defined as an array. Each element of the array must necessarily contain an id and many optional fields as desired. Optional fields can have a desired key and only one value.
* **Optional fields:** they follow key-value scheme and the key name must be defined by the user.

events={

\_id:(autom\_generated),

datetime:’\_\_\_\_\_\_\_’,

name:’\_\_\_\_\_\_\_’,

user:[

{\_id:’\_\_\_\_’,optional\_A:’\_\_\_\_’, optional\_B:’\_\_\_\_’,…},

{\_id:’\_\_\_\_’,optional\_A:’\_\_\_\_’, optional\_B:’\_\_\_\_’,…},

…

],

optional\_1:’\_\_\_\_\_\_\_’,

optional\_2:’\_\_\_\_\_\_\_’,

…

}

On the other hand, users are also stored as a mongoDB document following the JSON structure that you can see below. User have only a required field which is the \_id (automatically generated), which must maintain this key that will be used to link with the user id in the event collection. In addition, you can add as many optional fields as desired, keeping the structure key-value. Key name must be defined by the user.

users={

\_id:(autom\_generated),

optional\_I:’\_\_\_\_\_\_\_’,

optional\_II:’\_\_\_\_\_\_\_’,

…

}

To install a new MongoDB database in GLASS, just need to specify the next fields to can use the function MongoConnect:

* **Username:** username with access to MongoDB.
* **Pasword:** password of the user.
* **Host:** host where is located MongoDB.
* **Database:** name of the Mongo database.

MongoConnect function would be something like this:

function MongoConnect($username, $password, $database, $host) {

// Connect to Mongo Server

$con = new Mongo("mongodb://{$username}:{$password}@{$host}");

// Connect to Database

  $db = $con->selectDB($database);

return db;

}