



THE UNIVERSITY
OF QUEENSLAND
AUSTRALIA

LIBRARY
We link people with information

LibStats Installation Manual

R00 - 25 October 2006

Contents

1	Introduction.....	3
1.1	License	3
2	Requirements	3
3	Installation.....	3
3.1	Unpack	3
3.2	Create the Database.....	4
3.3	Edit config.inc.php.....	4
3.4	Test.....	4
4	Customising	4
4.1	LDAP	5
4.2	Templates	5
4.3	Reports	5
4.3.1	The StatsReports Class	6
4.4	Importers	7
4.4.1	BRAN Importer	7
4.4.2	Generalised Importer	7
4.4.3	Writing an Importer	Error! Bookmark not defined.
4.4.4	CircTab	7
4.4.5	DocDel	8
4.4.6	RecordStatsController->action('import', \$p)	8
5	Deploy.....	9
6	Revision History	10

1 Introduction

LibStats was developed at the University of Queensland Library using MySQL AB's *Eventum* application as a starting point.

1.1 License

LibStats is licensed under the GNU GPL. See <http://www.gnu.org/copyleft/gpl.html> for details. In summary, the license allows redistribution of the source code and application, distribution of modified source code and application provided the modified version is labelled as a modification, the modifications retain the license and credit the original authors. The license makes it clear that this software is distributed with no warranty. There is also a restriction on patenting the program.

2 Requirements

- PHP 5
- Apache - not tested on 2.x but should work
- SQL - UQ uses MySQL 4.1

3 Installation

3.1 Unpack

The distribution comes as a tar.gz file which can be unpacked on a unix system using

```
tar -zxvf libstats-1.0.tar.gz
```

This will create a directory called 'libstats' under the current working directory. The libstats directory in turn has four subdirectories:

- db-schema – contains the mysql dump of the DB which is used to initialise the database.
- htdocs – the part of the program which should be in the webroot
- docs – user manual and a copy of this document
- uqlapi – not needed but we use this to store the config.inc.php file outside of the webroot.

Once the files have been unpacked, you need to create some extra directories and set permissions.

```
cd libstats/htdocs
mkdir templates_c
touch error_handler.log
chown apache templates_c error_handler.log

echo "<?php
include_once('/absolute/path/to/uqlapi/config.inc.php'); ?>" >
config.inc.php
```

The last command makes a config.inc.php file which includes from the uqlapi dir. All of the program files include config.inc.php from the base of the htdocs dir but we don't want to have passwords in the webroot so we in turn include from the uqlapi dir.

3.2 Create the Database

In the db-schema directory there is one file called dump.sql. If you run this in an SQL client logged in with root privileges, it will create and initialise a database called libstats. You can change the name by editing the file.

```
mysql -u root -p < dump.sql
```

Then in mysql, set the access rights to the DB

```
grant all on libstats.* to  
'libstats'@'webserver.someplace.edu.au' identified by  
'secretpassword';
```

3.3 Edit config.inc.php

```
cp htdocs/setup/config.inc.php uqlapi
```

Now edit the uqlapi/config.inc.php file. Start at line 35 to set the main permissions and paths for the application.

- APP_PATH is the full unix path to the htdocs directory
- APP_RELATIVE_URL is the relative URL to the base of the application as seen by the web client.
- APP_HOSTNAME – is the fully qualified domain name of your webserver.
- APP_START_YEAR is the first year that you will be entering stats. If you have historical stats to enter, then you can set this to the year you started recording and import the historical data.

Be sure to edit the SQL settings too.

3.4 Disable SSL

If you don't have an SSL mirror of your website, you might have trouble logging in as the login page tries to use SSL. To disable it, edit include/controllers/class.logincontroller.php line 65 and comment out the 'if' block.

3.5 Test

The application should now be ready run.

Open a web browser on the configured URL for libstats. You should see a login screen. You can login as 'admin' using the password 'admin'. From here you can check that the pages load for the user management and configuration pages.

4 Customising

To customise libstats, a person with basic PHP and HTML skills is needed. The modifications needed are small.

4.1 LDAP

There are some LDAP configuration variables in `config.inc.php`. These can be used to set the LDAP server and the user and password that can search LDAP.

The file `htdocs\include\class.uq_ad.php` needs to be customised to work with your institution's ldap server. This is used when adding contacts and users to load their email and phone numbers into the LibStats DB.

To authenticate users against the LDAP, you may need to edit the file `include/class.auth.php` at line 498. Note that this function always returns true while `APP_TESTING` is on so you can turn off the `APP_TESTING` define in `config.inc.php` to make this function actually do the ldap authentication.

Also set `APP_USE_LDAP` to true in `config.inc.php` to turn LDAP checking on. Be sure to add an LDAP user with Administrator permissions before turning on LDAP.

4.2 Templates

The two files useful for branding LibStats are `templates/en/banner.tpl.html`, `templates/en/app_info.tpl.html` and `templates/en/footer.tpl.html`.

The contents of `banner.tpl.html` will be put at the top of each LibStats page.

The contents of `footer.tpl.html` will be put at the bottom of each LibStats page.

The contents of `app_info.tpl.html` is also at the bottom of most pages just above the footer.

These templates use a templating system called smarty <http://smarty.php.net/manual/en/>. The pages are similar to HTML with some variables obtained from the PHP main software through '{ }' statements. The variables made available to smarty templates must be assigned in the application using `$tpl->assign('variable_name', $value);` statements. These can be located on every page by putting them in `includes/class.template.php` function `processTemplate` on line 180.

4.3 Reports

The reporting system has the ability to derive columns, rows and stats groups from the stats data. This is how we get totals, percentages and summary stats groups.

- The derived values are not configurable through the menus. The code must be altered in order to derive new values.
- If the database ids of the stats groups, rows and columns change, any reports that use them will not work properly.
- If the default Stats Groups supplied with the software are used, no modifications are needed. This section only applies if there is a need to customise the reports.

4.3.1 The StatsReports Class

To customise the derived values in the reports, edit the file `include/class.statsreports.php`.

At the top of this file is a list of 'handlers' for stats groups (`sg_handlers`). The numbers on the left are the database Ids (`sg_id`) of the StatsGroups in the db table `ls_stats_group`. The negative numbers in the `sg_handlers` array are for derived stats groups which are described below.

The names on the right of the `sg_handlers` table are names of methods to call (listed at the end of the file) to add the derived values to the stats group.

The method `getStatsGroups` at about line 115 has a list of derived tables. The derived tables are numbered starting at -1 and counting backwards. The `$table_search` array on line 140 is used to setup where the derived tables will be placed in the list of stats groups. The number on the left are the `sg_id` numbers in the stats group table of the stats group that we should insert the derived stats group after.

The first of the custom table handlers is found at line 448. The handlers setup an array of new column definitions and sometimes new row definitions. There are also generic handlers which are used if no custom handler is defined.

The columns in the new columns array rows are:

- Column Name
- Column type: 0 = count, 1=time, 2=percent, 3=decimal number
- isTotal: 1 If the column is a total, 0 if it is not
- function – the function to call to derive the columns - more on this below.
- Input_columns – an array of `sc_id` numbers from the `ls_stats_column` table in the db. These are the values that will be fed into the function to get a derived value – e.g. you can add two columns together. If you are using a derived column to derive another column, then use a -ve number where the first derived column in the list is -1.
- Position – the position at which this column will be inserted in the stats group. 0 is the start of the group. Note that you should allow for preceding column insertions when working out this number. If you don't want the column to actually be in the stats group (like if it is an intermediate column for use in another calculation) then put -1.

The columns are derived by getting the values of the columns for each row and then putting them in an array and passing them to the configured function.

The function which is called to generate the column can be:

- `array_sum` – this is a built in php function which takes an array of values and adds them together.
- `calcPercent` – configured as `array(get_class($this), 'calcPercent')`. This will accept two columns and will do a `col1 / col2 * 100`.
- `calcSubtract` - `array(get_class($this), 'calcSubtract')`. Results in `col1 - col2`.

Note that `handleStatsGroup2ai` calculates the percentage columns after the rows have been calculated. This is so that the percentages of the derived totals rows will be calculated correctly. The nulls in the columns tables are used so that the references to the other columns work properly. Also, the second call to the derived columns actually passes the values in the columns instead of the names of them. Follow this example when making tables that have percentages.

4.4 Importers

Like the reporting code, the importers rely on hard coded database ids to bring stats from other files and databases into LibStats.

There is an exception to the importers which is that the method called 'Import From BRAN' uses a totally different framework to the generalised importers called 'Import Circ' and 'Import DocDel'.

4.4.1 BRAN Importer

The BRAN importer is described in the User Manual and shouldn't need modification. It is pretty generalised. If you have a spreadsheet of data that has a row for each month and the data in columns, then you can save it as an XML Spreadsheet and import it into LibStats.

4.4.2 Generalised Importer

To customise the generalised importer or write a new one, a more experienced programmer is needed. The examples for `CircTab` and `DocDel` can be followed for inserting the data but being able to read the format of the import files will require some programming knowledge.

The generalised importer allows a subclass to be made to import data from a file or database into predetermined `StatsGroup Columns` (each class has hard coded `sg_id` numbers and `sc_id` numbers).

The importers can be found in the `include/importers` directory.

- `class.importer.php` is the base class for importing. It shouldn't need to be modified.
- `class.circstab.php` and `class.docdel.php` are importer subclasses.

In order to customise importers, you can start by understanding the two examples provided. The `CircTab` importer imports from a file, the `DocDel` importer imports from another database. The overview is that you first read the data from the source and then import it into LibStats using the `RecordStatsController` class and calling the 'import' action on it.

4.4.3 CircTab

The `CircTab` importer reads a file that is extracted from UQL's Millennium LMS by the information access department. This file contains counts of various borrowing activities through the system against a number of codes in the system.

The CircTab subclass just implements the 'import' method. This method is called after posting the initial import form which is defined in templates/en/importers/circstab.tpl.html. The \$p variable contains the POST variables.

The first 'while' loop reads the tab/pipe/comma delimited file into an array. The delimiter seems to change randomly with each version of millennium.

We then read the month and year that this file is importing to (entered on the web form in LibStats).

What follows is a hard coded mapping of ls_branch br_id numbers to millennium codes. We then map the ls_column sc_id numbers to the columns in the millennium file using a another set of arrays.

Finally we call another method for each group to do that actual importing.

Up until this point, we have just been assembling the data into chunks that correspond to how they would be entered in the system manually. We then use the same object that is used to enter data through LibStats entry forms. This is called the RecordStatsController.

The RecordStatsController has a method called action which lets us submit data to it as if we were submitting a form (in fact this is how forms are submitted too). In this case however, we have a slightly special case so we use the 'import' action and supply it with an array of input values which are assembled as if they were a PHP \$_POST array. These values can be seen on lines 155-162.

The 'si_values' must also be an array of the form 'sc_id' => value.

4.4.4 DocDel

The DocDel class has a similar layout to CircTab. It implements the 'import' method and starts by reading the month and year to import into. There are a few mappings of docdel database values to libstats database values.

After the mappings, is the code to connect to the docdel database. Then there is a bit of code to assemble the data into an array ready to be imported.

The important part is using the RecordStatsController with the 'import' action and having the params setup as per the example given on lines 157-163.

4.4.5 RecordStatsController->action('import', \$p)

This is the code that does the actual importing:

```
$params = array(
    'year' => $year,
    'month' => $month,
    'Date_Day' => 1,
```

```
'sg_id' => $sg_id,  
'br_id' => $br_id,  
'usr_id' => Auth::getUserID(),  
'si_values' => $col_values  
);  
  
$res = $rc->action('import', $params);
```

- year – the year that the statistic should be imported for
- month – the month that the statistic should be imported for
- Date_Day – this can be used if you will importing more than once a month. It is not needed though. Usually just set it to 1.
- sg_id – this is the id of the stats group in ls_stats_group that identifies the destination stats group of the import
- br_id – this is the branch that the data belongs to (br_id in the ls_branch table)
- usr_id is the current logged in person doing the importing.
- si_values – this is an array of (sc_id => value) which corresponds to the columns in the stats group. The sc_ids are in the ls_stats_column table.

When calling the import action on the RecordStatsController, you can only import one month of data for one branch for one stats group at a time. A separate call is need for each branch and stats group and month being imported.

5 Deploy

Be sure to turn off APP_TESTING in config.inc.php, otherwise password authentication will not be enforced. During testing it is handy to be able to log in as a number of different people with different permissions set to make sure that the users are set up correctly.

Maybe turn off the error reporting at the top of config.inc.php.

6 Revision History

Authors:

- MSS: Matthew Smith, Senior Systems Programmer, LTS.

Locations:

- SVN Path: “/libstats/trunk/docs/ libstats installation manual.doc”

Rev	Date	Author	Description
00	25 Oct 2006	MSS	Initial