道生一，一生二，二生三，三生万物。

——《道德经》

general Management Information System

*g*MIS

通用管理信息系统

Zhenxing Liu

wadelau@{ufqi, gmail, hotmail}.com

(Working with ufqi.com, ippjj.com)

Update Oct. 2014

(This page is intentionally left blank)

Contents

"In a demand-driven opinion, we faced increasing requests of creating enormous table-based management tools for operation teams in years of 2005-2010 at ChinaM, an affiliate of Telstra[1] in Beijing. These shared some common functions and most of them just needed to achieve basic goals (CURDLS) for a table. So we conducted many practices to find one to meet this kind of demand, for all, forever."

[1] http://www.telstra.com.au

1. MIS, GWA2 and gMIS

According to OCC's MIS guidelines[2], a Management Information System (MIS) is defined as "a system or process that provides the information necessary to manage an organization effectively. MIS and the information it generates are generally considered essential components of prudent and reasonable business decisions". A MIS has these goals:

- Enhance communication among employees.
- Deliver complex material throughout the institution.
- Provide an objective system for recording and aggregating information.
- Reduce expenses related to labour-intensive manual activities.
- Support the organization's strategic goals and direction.

Regarding to general purposes, especially with MySQL as back-end, we have tools like phpMyAdmin[3] and its descendant, Chive [4](Web-based MySQL Admin Interface). These systems have three common merits:

- No-targeting user data.
- On-demand and instant deployment without 2nd-round development needed.
- Basic data and structures manipulations: **c**reate, **u**pdate, **r**etrieve, **d**elete, **l**ist and **s**earch (**CURDLS**).

General Web Application Architecture[5] (GWA2) is a framework for web applications development. Its feathers include MVC-based, core-shared, clear and clean structures, quite fast new instances deployment and easy kick-start. GWA2 uses Smarty[6] as its default template engine.

*g*MIS is a new and generally-targeted kind of MIS. It is a GWA2-based web application and has following characteristics:

- For general purpose, no user data involved in codes.
- XML configurations.
- Run immediately on connection to databases.
- Ajax-enabled, i.e. GTAjax[7].
- Inline editing of user data.
- Customized input and output, e.g. WYSWYG.
- Strong searching, i.e. innovative page navigator.

We explore gMIS in next sections in detail and explain what these items represent.

2. gTbl Class

gTbl is the core component of gMIS. It locates in mod/gtbl.class.php in an instance of GWA2 point of view. gTbl stands for "general table" as this idea comes from an intention to find a tool to manage all tables without coding each function of **CURDLS** for every single table.

[2] http://www.occ.gov/publications/publications-by-type/comptrollers-handbook/mis.pdf
[3] http://www.phpmyadmin.net/home_page/index.php
[4] http://www.chive-project.com
[5] http://ufqi.com/dev/gwa2
[6] http://smarty.net
[7] http://ufqi.com/dev/gtajax

In a demand-driven opinion, we faced increasing requests of creating enormous table-based management tools for operation teams in years of 2005-2010 in ChinaM, an affiliate of Telstra in Beijing. These shared some common functions and most of them just needed to achieve basic goals (CURDLS) for a table. So we had conducted many practices to find one to meet this kind of demand, for all, forever.

We had ever tested phpMyAdmin and some other similar toolsets, but all of them failed to satisfy us. Then gTbl came out with the goal of general purpose, in the meantime, providing a customized configuration for each table. Gradually gTbl evolved itself to gMIS. That is to say, it is a pre-condition that a phpMyAdmin-like tool with more powerful and manageable functionalities.

Nowadays, there are still many engineers in various organizations to create different "management console" for their applications. It is therefore necessary to continuously improve gMIS and advocate it to public for wide usages.

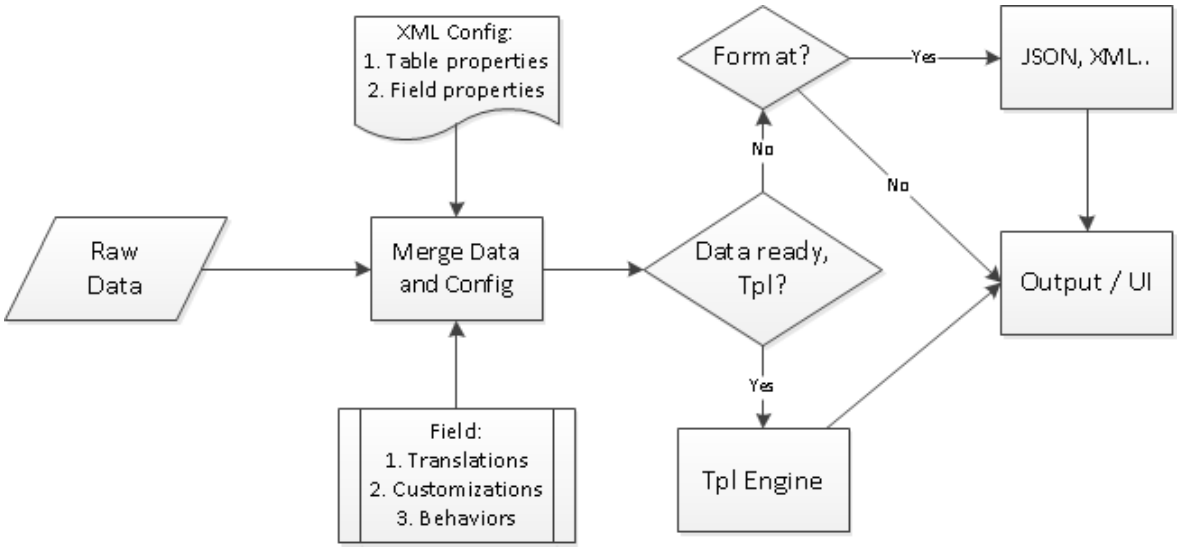


Figure 1, gMIS' flowchart

As an instance of and expanding from inc/webapp.class.php, the mission of gTbl is to read table configurations and apply those properties and behaviours to the table. Figure 1 shows the processing of gMIS. Its first step is to retrieve raw data from a database connection. Then gMIS merges data with predefined configurations. These configurations include translations, customizations, behaviours and constraints, all of which differentiate one table from the others. At its 3rd step, gMIS directs itself by whether there is a template file to be specified. If yes, it transfers the data to template engine for further output procedures; otherwise, it comes to test whether a kind of output format is needed, if one kind of data format is specified, e.g. JSON or XML, then do as request and output the formatted data.

In general, gTbl can be seen as a bridge between tables' configurations and manipulations. gTbl contains these components at present:

- **tags**
  Tags are defined to provide consistency for each property which may be used across all the application. Tags are used in describing both tables and fields. E.g. "table", "field", "chnname", "inputtype", "selectoption" and so on. All tags will be explained again in next sections.
- **set/get**

gTbl is a subclass of inc/webapp.class.php and instantiates some methods of its parent class. Most of methods are to get something from the configurations hash map which is created and filled in runtime, e.g.

a) getTblCHN, read a table's Chinese name from XML configurations via a hash map.
b) getTblCHK, read a table's validating rules.
c) getTblCharset
d) getOrderBy, read override settings for default order clauses in list view.
e) getMode, read table's access mode, usually a table can be accessed in full privileges, but in some cases, a table may be locked in read mode only.
f) getJsAction, read some JavaScript settings for a table in general.
g) getRelatedRef, read related functions and/or modules to current table.
h) getCHN, read its Chinese name of a field in current table.
i) getInputType, read a field's input type in html form.
j) getListView, read settings of whether to display current field in list view, value codes are set to be:
    i. '0': not show,
    ii. '1' or '': show,
    iii. '2': force to show.
k) getSingleRow, whether a field should be displayed in a single row.
l) getAccept, validating rules on current field. i.e. what kind of values can be input and saved in the field. This functions works with GTAjax or extra JavaScript-related html form validations.
m) setTrigger/getTrigger, set/get some trigger(s) on a field or table.
n) Other methods….

These methods can be read by source in class/gtb.class.php.

- **xml2hash**

  This function reads table configurations from an xml file and generates a key for each item of settings and save it in a hash container.

  When one table is specified in runtime, its configurations can be read from the hash container through an instance of class/gtbl.class.php.

  The hash container also includes some settings from global configurations which would be read from inc/config.class.php. The latter holds more items for the whole application.

- **I/O**

  Fields input and output are defined in xml configurations, e.g.

```
<field name="brandid">
 <chnname>商品品牌</chnname>
 <selectoption>fromtable::brandtbl::chnname</selectoption> <!—options read from another table -->
</field>
<field name="img">
 <chnname>配图 1</chnname>
 <inputtype>file</inputtype> <!—upload an file(image) -->
</field>
<field name="beauty_words">
 <chnname>美丽心语</chnname>
 <inputtype>textarea</inputtype> <!—input many paragraphs a time, e.g. an article -->
 <singlerow>1</singlerow>
 <listview>0</listview>
```

</field>

Accordingly, output of these fields will be showed in different style, e.g. a selector keep the same way of input and output; an image will show itself directly in browsers; a text area also lists all of its contents after same security validations.

More customized output tags will be needed with specified demands increasing, e.g. WYSIWYG html editor, multiple checkbox, embedded audio/video and so on.

- **ido.php/jdo.php**

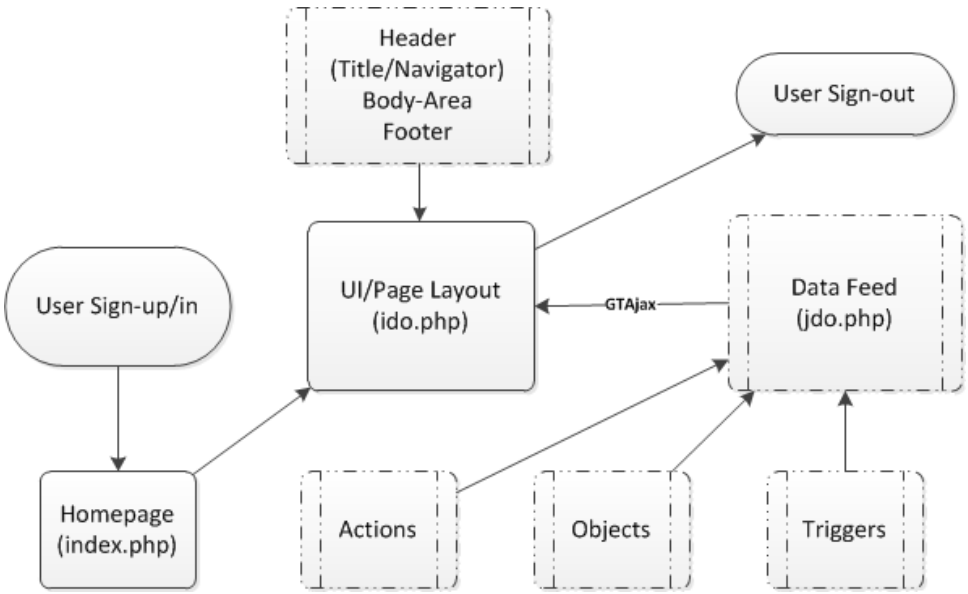


Figure 2, UI/Page Layout

gMIS has two main scripts: ido.php is designed for page layout and jdo.php is employed to manipulate data in background. These two parts are bridged with GTAjax, which triggered by user's mouse click or keyboard events.

As Figure 2 shows, the procedure can be seen that a user will be provided with a sign-in or sign-up page if he/she has not logged in. After the sign-in got done, a page or an application environment is given. All of tasks will be done within this page of the application. Behind it Data-Feed (jdo.php) continuously read or write data from/to backend services.

It is clear that gMIS just has only a PAGE in which most of other frames or popup windows/dialogs would be called for different requests and interactions.

3. Extra IO

In practice, the main purpose of a MIS is to input some and get others from the system. gMIS supports some basic input and output as text, select, file, textarea. For other uncommon field type, they might need more technical codes to be embedded. The following are some ideas on various extra input and output.

Extra IO is located in /extra/.

3.1. Inline editing

Normal listing

14 / 45284	0	呼和浩特房地产资讯	news.nm.fang.com	资讯站	自动抓取	2014-10-20 09:02:11
15 / 45283	0	中国警察网消防频道	xf119.cpd.com.cn	资讯站	手工	2014-10-20 08:58:13
16 / 45282	0	长江商学院	www.ckgsb.edu.cn	资讯站	自动抓取	2014-10-20 08:41:10
17 / 45281	厦门网- (7130)	厦门人居网	rjz.xmnn.cn	资讯站	手工	2014-10-20 08:41:09
18 / 45280	0	松原新闻网	www.0438news.com	资讯站	手工	2014-10-20 08:38:53
19 / 45279	机构或官网- (24369)	博罗县文体旅游局	wtlyj.boluo.gov.cn	资讯站	手工	2014-10-20 08:31:17
20 / 45278	0	江西消防在线	xf.jxnews.com.cn	资讯站	自动抓取	2014-10-20 08:25:51
21 / 45277	南阳网- (6768)	宛城网	wancheng.01ny.cn	资讯站	手工	2014-10-20 08:23:00
22 / 45276	0	西城区人民政府网站		资讯站	自动抓取	2014-10-20 08:17:05
23 / 45275	0	厦门业主网	yezhu.xmnn.cn	资讯站	手工	2014-10-20 08:16:52

Double click to make it editable.

10 / 45288	0	松原在线	www.songyuan.ccoo.cn	资讯站	自动抓取	2014-10-20 09:10:55
11 / 45287	0	新建在线	www.xinjian.ccoo.cn	资讯站	自动抓取	2014-10-20 09:09:29
12 / 45286	0	新建在线	www.0791d.com	资讯站	手工	2014-10-20 09:09:23
13 / 45285	0	张家港电视台		资讯站	自动抓取	2014-10-20 09:03:13
14 / 45284	0	呼和浩特房地产资讯	news.nm.fang.com	资讯站	自动抓取	2014-10-20 09:02:11
15 / 45283	0	中国警察网消防频道	xf119.cpd.com.cn	+报纸(1)	手工	2014-10-20 08:58:13
16 / 45282	0	长江商学院	www.ckgsb.edu.cn		自动抓取	2014-10-20 08:41:10
17 / 45281	厦门网- (7130)	厦门人居网	rjz.xmnn.cn		手工	2014-10-20 08:41:09
18 / 45280	0	松原新闻网	www.0438news.com		手工	2014-10-20 08:38:53
19 / 45279	机构或官网- (24369)	博罗县文体旅游局	wtlyj.boluo.gov.cn		手工	2014-10-20 08:31:17
20 / 45278	0	江西消防在线	xf.jxnews.com.cn	资讯站	自动抓取	2014-10-20 08:25:51
21 / 45277	南阳网- (6768)	宛城网	wancheng.01ny.cn	资讯站	手工	2014-10-20 08:23:00

-选择-
资讯站(0)
+报纸(1)
-杂志(2)
×电视(3)
÷广播(4)
通讯社(7)
机构及官网(5)
其他(6)

3.2. Input2Select

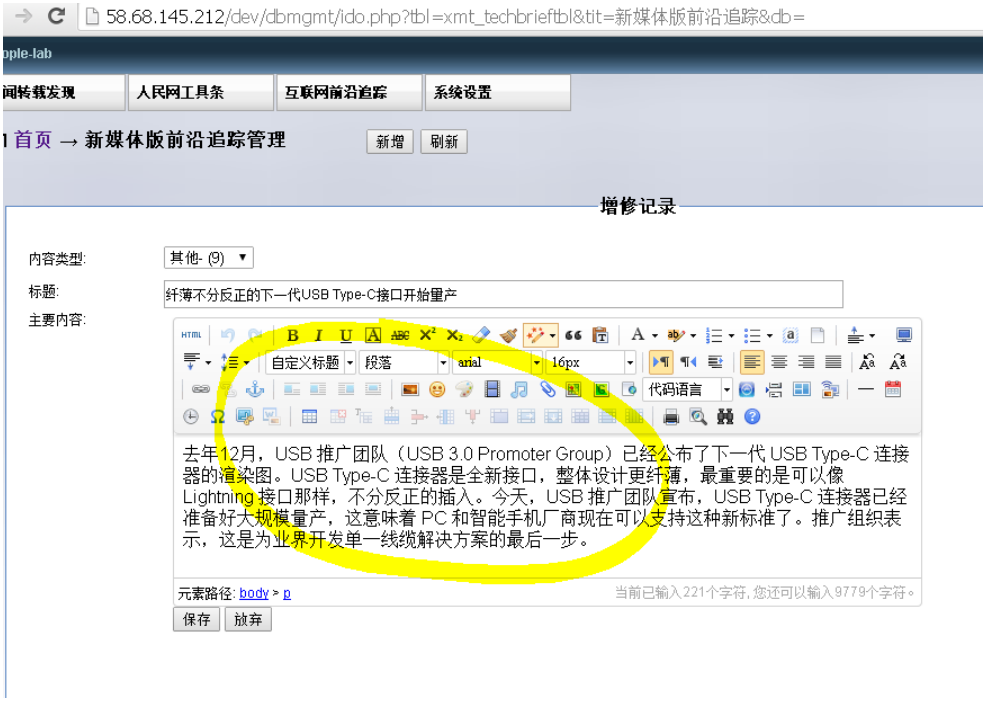
489 / 34279	0		auto.beelink.com	资讯站	手
490 / 34277	0		news.beelink.com	资讯站	手
491 / 34276	0	中国投影网		资讯站	自
492 / 34275	大众	大众网威海频道		资讯站	自
493 / 34274				资讯站	自
494 / 34273			学校中心校	资讯站	自
495 / 34272			报	资讯站	自
496 / 34271				资讯站	自
497 / 34270			业报导》	资讯站	自
498 / 34269			学校中心校	资讯站	自
499 / 34268				资讯站	自
500 / 34267				资讯站	自

大众网- (681)
大众证券报- (2431)
大众日报- (2494)
华声在线-大众卫生报- (3207)
农村大众- (4527)
大众医药网- (4826)
大众健康之窗- (4868)
大众科技报- (5542)
《大众日报》- (6652)
农村大众报- (6839)
大众网淄博综合- (19833)
大众网- (20893)
大众网体育- (21137)
大众网-滨州- (21249)
大众网--鲁南商报- (21268)

-R/l2Sn, -R/r2Sl

3.3. Wysiwyg

By iframe and other JavaScript-based tips to embedded a live html editor in the system.



3.4. Dialogs

(TODO)

4. Tags

Tags are used for two purposes: 1) to define a function across the whole application; 2) to be used as keys in xml2hash.

Tags are also xml node names and carry information from xml files to hash containers via xml2hash in class/gtbl.class.php, e.g.

```
<field name="sex">
    <chnname>适用人群</chnname> <!—“sex” will be showed as “适宜人群” in front page.  -->
    <selectoption>0:女性|1:男性|2:Both</selectoption> <!—the input type would be set as “select” with three options. -->
</field>
```

“field”, “chnname”, “selectoption” are all tags, which will be referred in gtbl class as:

```
public function getCHN($field){
    $tmpstr = $this->hmconf[$this->taglist['field'].$this->sep.$field.$this->sep.$this->taglist['chnname']];
    return $tmpstr = $tmpstr==null?$field:$tmpstr;
  }
```

There are already more than 30 tags defined by now in class/gtbl.class.php. Here is the full list.

```
1) 'table'=>'table',  # to mark a node of a table
2) 'field'=>'field', # to mark a node of a field
3) 'chnname'=>'chnname', #  Chinese name of a field
4) 'inputtype'=>'inputtype', # input type of a field, available for text | textarea | select | file
5) 'selectoption'=>'selectoption', # select options, two types:
```

- <selectoption>0:女性|1:男性|2:Both</selectoption> # definite options
- <selectoption>fromtable::brandtbl::chnname</selectoption> # dynamically read from another table

6) 'selectmultiple' => 'selectmultiple', # is multiple of a select or not
7) 'extrainput'=>'extrainput', # other input types which cannot be showed as common, usually
   - read/write to another table via extra/linktbl.php or extra/OTHER
   - other special I/O control, e.g. multiple checkbox
8) 'memo'=>'memo', # memo message for a field or table
9) 'charset'=>'charset', # character charset
10) 'dbname'=>'dbname', # database name
11) 'relatedref'=>'relatedref', # related functions of a table
12) 'listfieldcount'=>'listfieldcount', # how many fields to display in list view within a single row
13) 'listview'=>'listview', # hide in list view or not, '0': not disp, '1' or '': disp, '2': force to disp
14) 'singlerow'=>'singlerow', # display in a single row
15) 'printref'=>'printref', # for a table in print layout
16) 'reftable'=>'reftable', # related tables to current table
17) 'jsaction'=>'jsaction', # JavaScript binding to a table or field
18) 'delayjsaction'=>'delayjsaction', # JavaScript for a table or field, delay its start time in runtime
19) 'check'=>'check', # process business check logic...
20) 'orderby'=>'orderby', # explicitly specify a field for ordering...
21) 'defaultvalue'=>'defaultvalue', # default value during add/modify…
22) 'managemode'=>'managemode', # managemode for a table, r(read),w(write),d(delete)
23) 'accept'=>'accept', # front-end validator, e.g. "lt=100,gt=1000"
    This requires GTAjax loaded firstly.
24) 'trigger'=>'trigger', # trigger sth when meeting some requesters
    The tag makes it come true that a static project goes alive and health with the concept of work flow.
25) 'readonly' => 'readonly', # there are two kinds of read-only data:
   - some fields do not need input by users, but by programs;
   - for privileges reason, some fields are denied to modify by some users.
26) 'href' => 'href', # href of a field
27) 'hidesk' => 'hidesk', # default search key, e.g. environment variables like time, IP, user ID and so on, and with these clauses the query can be limited in a small scope.
28) 'css' => 'css', # css of a field or table
29) 'superaccess' => 'superaccess', # access control over system settings
30) 'stat' => 'stat', # methods used when making statistics, sum | count | average

Some of tags share a few default settings, e.g.

0 or ‘’: always means “No”, “Not”, “Negative”, “Empty”;

1: stands for “True”, “Positive”, and “Not Empty”.

A brief of a configuration xml looks like:

```
<?xml version="1.0" standalone="yes"?>

<tablecfg>

    <table name="producttbl">

      <chnname>商品详情表</chnname>

      <listfieldcount>8</listfieldcount> <!--max_disp_cols-->
```

```
            <since>20121015</since>
            <creator>Wadelau</creator>
            <superaccess>inherit::id=USER_ID::rw</superaccess>
          </table>
          <field name="productno"><chnname>商品编码</chnname></field>
          <field name="chnname">
           <chnname>中文名</chnname>
          </field>
 ....
</tablecfg>
```

5. Actions

Actions include standard tasks of what have been discussed above, i.e. CURDLS, and non-standard jobs which generated by specific work of a project. Each of standard actions needs at least an html form and a corresponding do-form action. For non-standard jobs, there are some tailored pairs.

There are therefore the following mandatory actions located in act/:

1) addmodi.php, to present creating form and/or updating form
2) doaddmodi.php, to receive and save data filled by user in addmodi.php
3) dodelete.php, to function a backend service for dropping records
4) jdo.php(in /), to function as a combination of LIST and SEARCH
   Page navigator[8] in section 9 of GWA2describes the powerful SEARCH in great detail.
5) log.php(writelog.php), to log each step in backend, provide traceable history
6) print.php, to display an html form in printable style
7) view.php, to present a retrieving form with stored data
8) updatefield.php, to edit any single field of a row separately, i.e. inline editing of user data

Other non-standard jobs include (in act/):

9) checkaccess.php(and tblcheck.php), for security-check jobs
10) sendbulkmail.php, for mails sending
11) toexcel.php, for downloading data in Microsoft Office Excel compatible format
12) trigger.php, for dynamically invoking other actions, which make work flow possible

These actions listed below are used to carry out dynamic modules:

13) synctblfield.php
14) synctblindexkey.php
15) updateobjectfieldtbl.php
16) updateobjecttbl.php

[8] http://ufqi.com/dev/gwa2/General.Web.Application.Architecture.201301.v4.pdf

Actions can be launched by either users' clicks or being triggered by other actions.

All actions are handling in jdo.php(see Figure 2), and they are provided in the parameters of each request as:

http://ufqi.com/dev/proj-a/console/?tbl=tbl-a&act=view&...

For various actions, jdo.php includes specified act scripts to make transactions. At present, there are a few of pre-defined acts in paraments:

&act=add | modify | list | list-addform | list-dodelete | view | print | updatefield

"list" is the default action. "updatefield" is worth being explained in more detail. As a distinctive point of Chive, "Inline editing of data" provide users a quiet convenient way of working with data input and output, so does gMIS. "updatefield" is designed to edit any single field of a row separately, e.g.

Figure 3, Inline editing of data in gMIS.

6. Triggers

Actions are routine jobs and triggers are dynamically-set work which keep the target project "alive and health", e.g. procedures automation, data consistence, message notifications, auditing and monitoring and traceable history (log).

In the general view of MIS, triggers are set by the concept of these principles showed in the following figure (Figure 4).

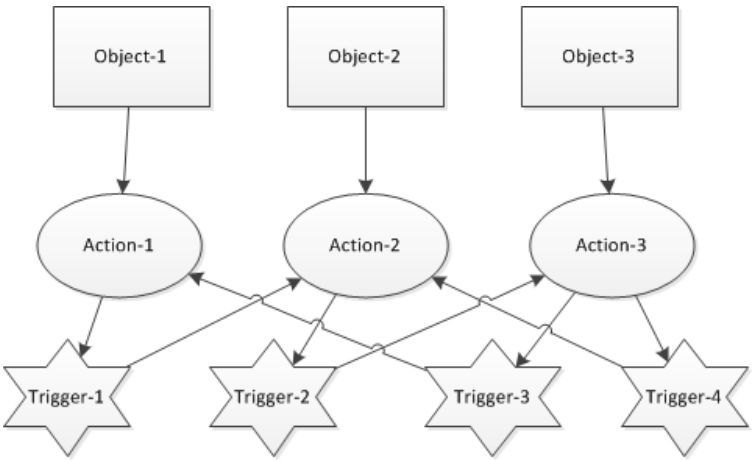


Figure 4, Triggers in gMIS

Suppose object-1 has an action-1. When a user launches the action-1, the action would do another action-2 by trigger-1. That is to say, a user's request results in two actions at a single time.

In the same way, object-2 gets two actions from action-2 with an accompanying action-3 via trigger-2. Object-3 gets three actions done in a single request with two triggers: trigger-3 for action-1 and trigger-4 for action-2.

An example demonstrates this clearer.

In an office daily work, a clerk raises an application for leave. When the clerk clicks the submit button on an application form after filling some information, gMIS will do at least two jobs: 1) one is to save the data of this application form; 2)forward the application to the clerk's manager to approve this application. The second action is set by a trigger looking as:

```
<field name="state">
        <selectoption>0:applying|1:manager-auditing|2:passed|3:done</selectoption>
        <trigger>0::notify::manager_todo_tbl::....
                |2::notify::clerk_leave_tbl::...
        </trigger>
        ....
</field>
```

It is easy to see that triggers simulate the mode of work flow in the scenario. In general, gMIS splits each step of a complex job into many pieces of work. For each piece of work, a trigger and a corresponding action are defined and every one of them can be invoked or triggered in a loop-like application environment. Another benefit from this design is that each action can be reused again for other objects. It is therefore to realize that write once for all, forever.

Triggers are interpreted in act/trigger.php. Triggers are bound to updating actions, e.g. act/doaddmod.php, act/dodelete.php.

7. Dynamic Modules

Adding or removing modules or objects could be done dynamically in gMIS because that gMIS inherits some gene of Chive and/or phpMyAdmin. Just like manipulating another new table in Chive, managing a new object/module is easy to achieve by creating an xml file for it.

Through an xml configuration is optional, it is recommended to generate a customized xml configuration file for each table. The xml file usually contains lots of information on what the table/object/module looks like and how the data of the module will be transacted.

Generally, adding a module is to create a link to a URL like:

```
./?tbl=tbl-new&act=list&...
```

Then, a web-based management console is showed in a modern browser. It is just a finger-snap. But there is a long way to improve the basic CURDLS function into a customized MIS. To map the module into a position (link) in the menu of this MIS is the first step. This step can be done in default gMIS menu bar and its path is Settings → Menu Adjustment.

The second step it to create the module in the path of Settings → Modules. This function looks much like phpMyAdmin or Chive. These kinds of tools provide users lots of convenience that instead of creating tables in command-line console via a remote connection to server, such web-based admin interfaces allow user to manipulate tables directly, e.g. create a table, drop a table, add/remove/update table fields, add/remove/modify table indexes and keys. All in one, this function is a web-based console for tables' management for a database.

For an instance, in a running application administrators want to add a module to support a new kind of product which has its own properties and actions to do.

Table 1, ways of adding new module

		Traditional MIS	**gMIS Dynamic Module**
Developing	Basic CURDLS	Dir-a/, add-a.php/add-a.html, do-add-a.php, edit-a.html/edit-a.php doedit-a.php list-a.php/list-a.html search-a.php/search-a.html do-search.php dodelete.php	(No script files needed, add new module in web-based console).
	Business-based	OTHER-BUSINESS_LOGIC.php ….	OTHER_BUSINESS_LOGIC.php ….
Running		modify menu, add new link manually in script files; Train operators for new functions.	Add new link in web-based console. Notify operators.
Time cost		3 plus days.	3 minus hours.

By saving time and work cost, dynamic modules make gMIS self-improve and self-evolve to meet new changes with business growing. It decrease the amount of coding work to a very lower level by reuse basic CURDLS functions and also decreases the work of training operators for new modules due to that modules added in this way share the same UI and logic.

8. Security, Firewall-style Auditing

It is no more careful than us to strengthen the securities with web-based MIS since our servers ever caused severe data leak caused by one issue with phpMyAdmin in around 2000 when we provided maintenances for those servers.

Based on a patent solution[9] and the theory of access control[10], we design a new mode of access control mechanism. The design works like firewall for network packets flowing.

System securities can be set and adjusted in the path of Settings → Privileges Management.

[9] http://www.cnpatent.com/list_zhuanli.asp?id=200810171394&zt=
[10] http://infosec.pku.edu.cn/~wyz/course/8.pdf

The mode has three parts: objects and objects group, users and users group, access methods and their combinations. Matching any three of these parts can make a rule of access control, e.g.

User-a → object-a → read & write

User-group-b → object-b → read only

User-c → object-group-c → read only

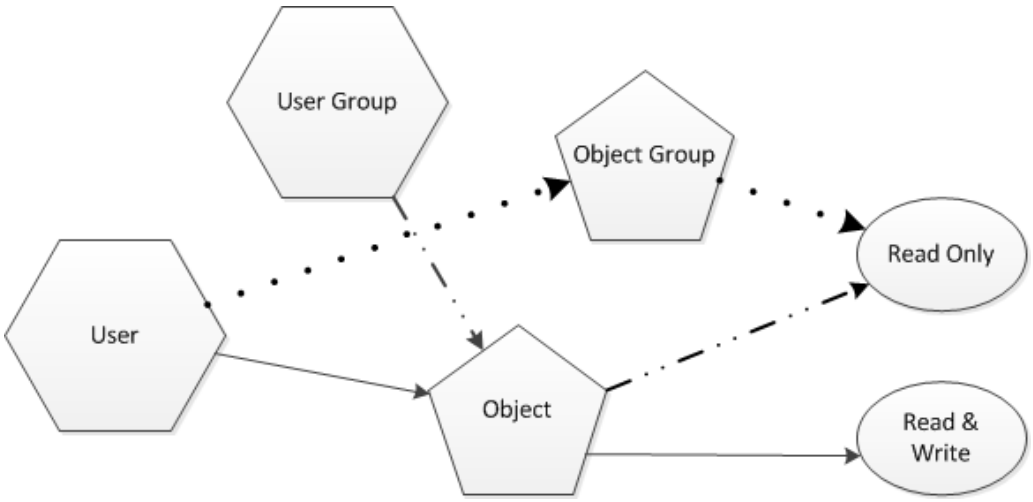


Figure 5, Securities Mapping in gMIS

Unlimited rules of access control could be made as wanted except that there are sometimes conflicts between two of these rules. For instance that, user-a could read object-b according to rule-a but could not read it by rule-b. Therefore one more field of priority is provided to make sure that when conflicts aroused, the rule with higher priority will win.

In gMIS, the access control goes further for a table-based record. It provides a tool that allows administrators to hide one column of a record but show other columns of that table, e.g. in table with four fields:

Field-a, field-b, field-c, field-d

A rule says user-a could read field-a, field-b and field-c but not field-d.

In runtime, for every request, there is a full-match checking by users' inputs, e.g. objects, actions and so on. If no access control rules are matched, it goes to the default. That is to say, one can set a system with open-door or close-door in default.

The procedure of checking access is at act/checkaccess.php which is included in comm/header.inc.

Figure 6, rules of access control in gMIS

9. To-do

9.1. Automatic Modules and Tags Setup

Instead of composing XML configurations manually, an automatic toolset will be made to provide this procedure of manipulating modules and tags in a web-based manners.

9.2. API

For a future deployment, API is selected for both mobile Internet and desktop applications.

10. Document history

Table 2 document history

No.	Version	Updates	Date	Author(s)	Inspector(s)
2	V0.1	(Revised)	2013-03-09	Zhenxing Liu	
1	v0.1	Initial draft	2013-01-23	Zhenxing Liu	