From interface to interaction

General Web Application Architecture (GWA2)

Zhenxing Liu

wadelau@{ufqi, gmail, hotmail}.com

(Working with zoneli.com, ufqi.com)

Update Jan. 2013

(This page is intentionally left blank)

Contents

1. Overview

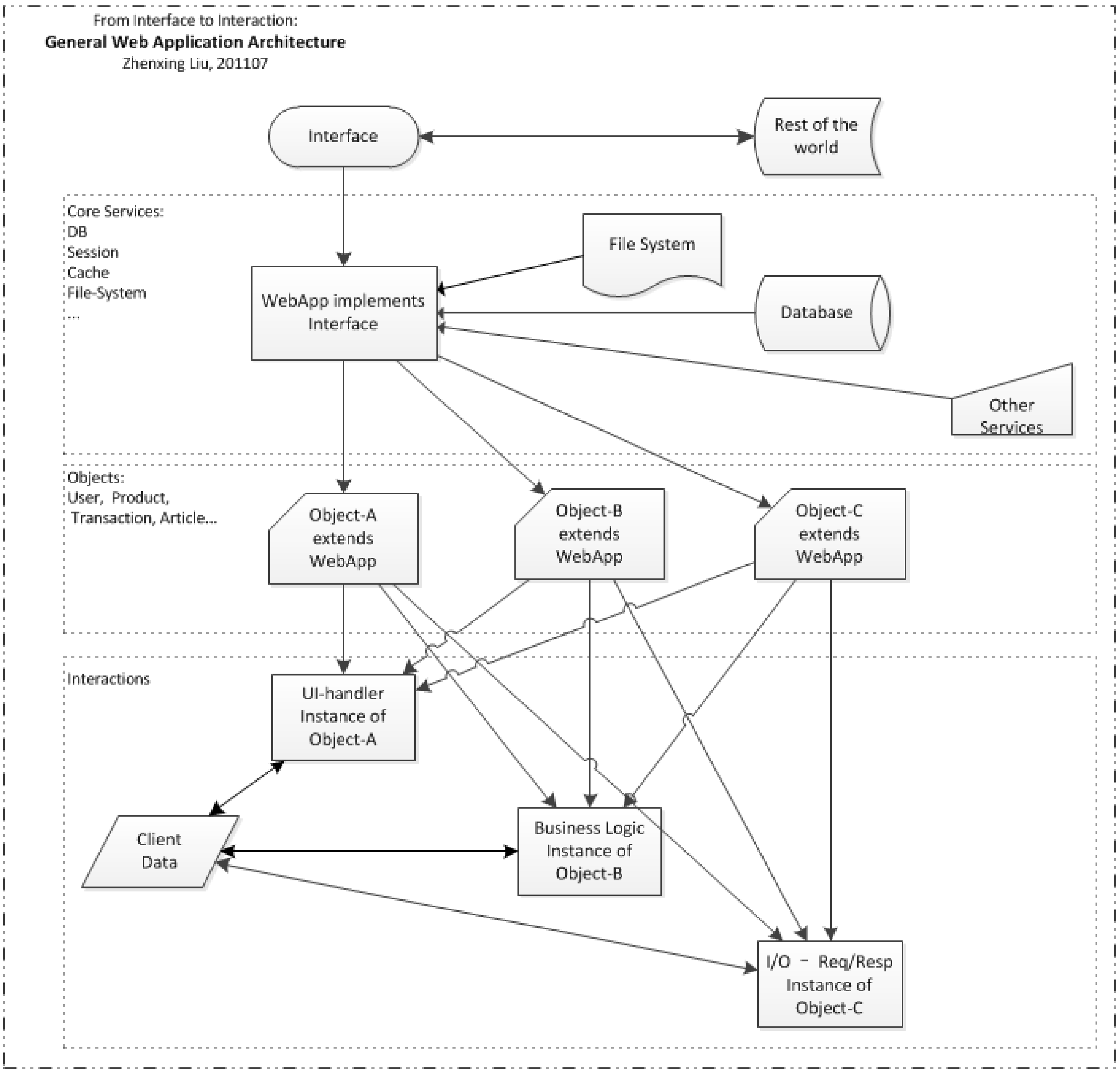
From Interface to Interaction:
General Web Application Architecture
Zhenxing Liu, 201107
Interface
Rest of the world
Core Services:
DB
Session
Cache
File-System
...
File System
WebApp implements Interface
Database
Other Services
Objects:
User, Product, Transaction, Article...
Object-A extends WebApp
Object-B extends WebApp
Object-C extends WebApp
Interactions
UI-handler Instance of Object-A
Client Data
Business Logic Instance of Object-B
I/O - Req/Resp Instance of Object-C

Figure 1 Main components in this GWA2 design

2. Interface: /inc/webapp.interface.php

```
interface WebAppInterface {
    function set($key, $value);
    function get($key);
....
```

At the top level of a web application, it needs an interface to tell the rest world what it has and how it could communicate with outside.

For a website with online service, it (/inc/webapp.interface.php) defines the following methods (behaviours):

```
function set($key, $value);
function get($key);
function setTbl($tbl);
function getTbl();
function setId($id);
function getId();
function setBy($fields, $conditions);
function getBy($fields, $conditions);
function rmBy($conditions);
…
```

In a web application driven by databases, this interface provides two more pairs of methods(.setId/getId, .setTbl/.getTbl) which assume that every table in backend databases have a table name and an id field.

.set/get methods save running time values temporarily in a pre-defined container (e.g. an array/hash table). .setBy/getBy is to write/read data to/from persistent storage device (e.g. database, file system, remote network). .rmBy will be invoking when deletion is requesting to an object.

3. Parent class: /inc/webapp.class.php

```
class WebApp implements WebAppInterface {
    var $dba = null;
    var $hm = array();
    var $hmf = array(); # container for the Object which extends this class
    var $isdbg = 1;
....
```

In order to make it come true, a top parent class (/inc/webapp.class.php) is defined to implement the interface above. This class realizes (put the real codes there and take actual actions to make something changed) all methods which have been listed in the interface.

$dba here is designed to connect to different database service, e.g. MySQL, Oracle, SQL Server.

$hmf is the so-called container mentioned in above section, which temporarily holds running time values and waits for further action with these values: writing to persistent storage or discarding at the end of this script.

.setBy/getBy/rmBy collect values in $hmf and construct new sql sentence then send the sql to $dba, after that waiting for the result and forward $dba result to its caller. In other scenarios, these methods might connect to other services or write to local disk instead that communicate with databases service.

This design of temporary container and persistent storage is much like what it can be seen from operating systems where they have RAM for temporary access and local disk for persistent storage.

4. Actual Object: /mod/user.class.php

```
class User extends WebApp {
	var $sep = "|";
	var $eml = "email";
	function User() {
		$this->dba = new DBA();
		$this->setTbl("zoneli_usertbl");
	}
	function setEmail($email) {
		$this->set($this->$email);
	}
....
```

Due to this actual object User is created by extending the top parent class WebApp, it has and holds all the properties and methods from its parent. Therefore it needs not to do works like connecting databases, handling I/O and so on, what it needs to do are the specific properties and methods which distinguish itself from the common class, i.e. the parent class or the interface itself.

These specific properties of the object User here include Email, Nickname, Password etc., and accordingly they needs some methods(behaviours) to change their status or values, that is, setEmail, getEmail, setPassword, getNickname and so on.

Till now, the design looks much like other projects which declares they deploy PHP web application using object-oriented architecture. After the actual class has been created, it is easy to create a front page which initializes an instance of this class and access its properties and execute its methods.

5. Front-page, view, ctrl

Take /user/index.php as an example.

```
require_once("../inc/header.inc");

$user = new User();

$out = str_replace('TITLE','User  Center', $out);

$a = $_REQUEST['a'];

if($user->isLogin()){

	if($a == 'logout'){
```

```
                # actions
        }else{
                # actions
        }
    }else{
        if($a == ''){ $a = 'logi';}
        if($a == 'logi') {
            $out .= $logiform;
            $actresult = true;
        }else if($a == 'regi') {
            $out .= $regiform;
            $actresult = true;
        }else if($a == 'regi.do'){
    ....
```

It is show time!

/inc/header.inc is appointed to some basic work for creating a dynamic page to user (request). This included page might act like:

1) Initialize a page object and prepare an output holder, $out
2) Initialize a user object and identify user according to the environment parameters, e.g., IP, SessionId, and CookieId and request URL....
3) Some other business logic

Then, it is the turn for the front page to run, adding something more the $out and work with the object $user, or most, other objects, e.g. $article, $product, $transaction and so on.

Finally with the ending page /inc/footer.inc, $out is ready to print out.

Moreover, there is a parameter "a" being employed to decide which action or request is made currently.

With "?a=logi", the connection is to make a request for login form page;

With "?a=logi.do", the connection is to make a request for real check-in action with username and password;

....

Otherwise, there are lots of pages to be created and lots of redirects to go through among these pages. For further discussion about this sub-mode, please see the page[1]: Practice of Java in JSP.

6. Return values

[1] http://ufqi.com/exp/x1499.html?title=Java,JSP 应用开发实践中的新模式探索.

Whatever it is, a function or a method, a return value is necessary except that a void type is used explicitly. It does not need pay more attention to it if the return type and value is defined as Boolean type, i.e. true or false. Additionally it is also easy to handle a return value as basic arithmetic, e.g.

```
        int function addTwice(int x){ return (x + 2x); }
```

However, in more “advanced” program languages, or in object-oriented languages, functions or methods of an object are not always to compute some basic calculation. Sometime it can never be more complicated or sophisticated than a cat’s eating or a bike wheel rotating.

The expecting results might be at least two parts: result type (succ|fail) and result description/status/data (what|how).

For instance, in statically typed languages,

```
        public Hashmap eat(int foodcount) {
                Hashmap result = new Hashmap();
                boolean issucc = true;
                if(cat.isFull()){
                        result.put(“result”, new Boolean(false));
                        result.put(“desc”,”cat is full.”); issucc = false;
                } else {
                        for(int i = 0; i < foodcount; i++) {
                        If(i > cat.STOMACH_CAPACITY) {
                                result.put(“result”, new Boolean(false)); result.put(“desc”,”cat cannot
    eat more than [“+i+”].”);
                                issucc = false;
                                break;
                        }
                    }
                }
                if(issucc){
                        result.put(“result”, new Boolean(true));
                        result.put(“desc”, “cat ate [“+foodcount+”]”);
                }
                return result;
        }
```

Taking this into consideration, .setBy/.getBy mentioned above defines their return types as array/hashmap. That is, in PHP, or other dynamically typed languages,

```
    $result = user.setBy(“nickname,updatime”, $this->hmf);
    If($result[0] || $result[“result”]) {
        //- succ, display msg from $result[1]
    } else {
        //- fail, display err msg from $result[1]
    }
```

Compared this with PHP built-in function, mysql_query[2],

For SELECT, SHOW, DESCRIBE, EXPLAIN and other statements returning resultset, mysql_query() returns a resource on success, or FALSE on error.

[2] http://cn2.php.net/manual/en/function.mysql-query.php

For other type of SQL statements, INSERT, UPDATE, DELETE, DROP, etc., mysql_query() returns TRUE on success or FALSE on error.

This hashmap return value customizes the message body on success and failure. It provides "insertid" or "affectedrows" on success, and self-defined error message on failure by analysing other PHP built-in functions in the 2nd scenario. And similarly in the 1st case, it also provides custom error message on failure and structured data on success.

Actions like these may be called further-packaging sometimes.

7. Output, response, view

From what has been described above, the output of a request has not been explained clearly. There always is a global variable to hold all the data to be displayed to users, i.e., $out. With MVC mode, $tpl and $data are defined.

In general, GWA2 supports two kinds of data output:

- **Non-template-based layouts.**
  It just adds all output data into a variable named $out as usual and print the $out directly at the end of a request session.
  This has been used in the example code above as seen in front page.
  It mixes all of html, css and JavaScript together and construct a single long string, $out.
  Any content to be pushed to the client should be put in this variable, except some HTTP headers information, e.g. Content-Type, Redirect, Cookie, etc.
  This layout suits for small projects which do not need professional UI designers to be engaged in these projects. Programmers do all the work: coding, business logic and layouts sketching. The output may not have a nice appearance, but tidy, clean and lightweight. For instance, some APIs work between two more nodes.

- **Template-based layouts.**
  It deploys itself as an M-V-C backend, generating $data and a variable of template file.
  In this scenario, two more kinds of sub layouts are defined:
  - Template-file standalone.
    The template-file display itself as standalone mode without any further decoration, e.g. header, footer or other part of an html page. For instance,
    $template-engine->display($template-file);
  - Template-file embedding.
    This mode allows the template file to be embedded in a pre-defined layout and be a part of that page. That is to say, the template-file is a partial area of an html page not containing any other part, e.g. header, footer and so on. For instance,
    template-file-whole.html:
    <html><head></head><body>
    {include file="template-file-partial.html"}
    </body></html>
    The displaying procedure will be:
    $template-file = "template-file-partial.html";

```
$template-engine->display("template-file-whole.html");
```

However, these two layouts, template or non-template, are not conflictive, can work together smoothly catering for various scenarios. In the same way, behaviours of template-file can also be standalone mode or embedding.

Hints for template engine Smarty-specific:

- Automatically convey elements of $data to $smarty->assign($k, $v).
  ```
  foreach($data as $k=>$v){
     $smt->assign($k, $v);
   }
  ```
- Convert resources path to relative directory.
- Using include directive for share modules.
  ```
  …
  {include file="left-menu.html"}
  …
  ```
- Using {literal} to avoid the conflict from the delimiter "{" symbol.

8. MVC, Module-View-Controller

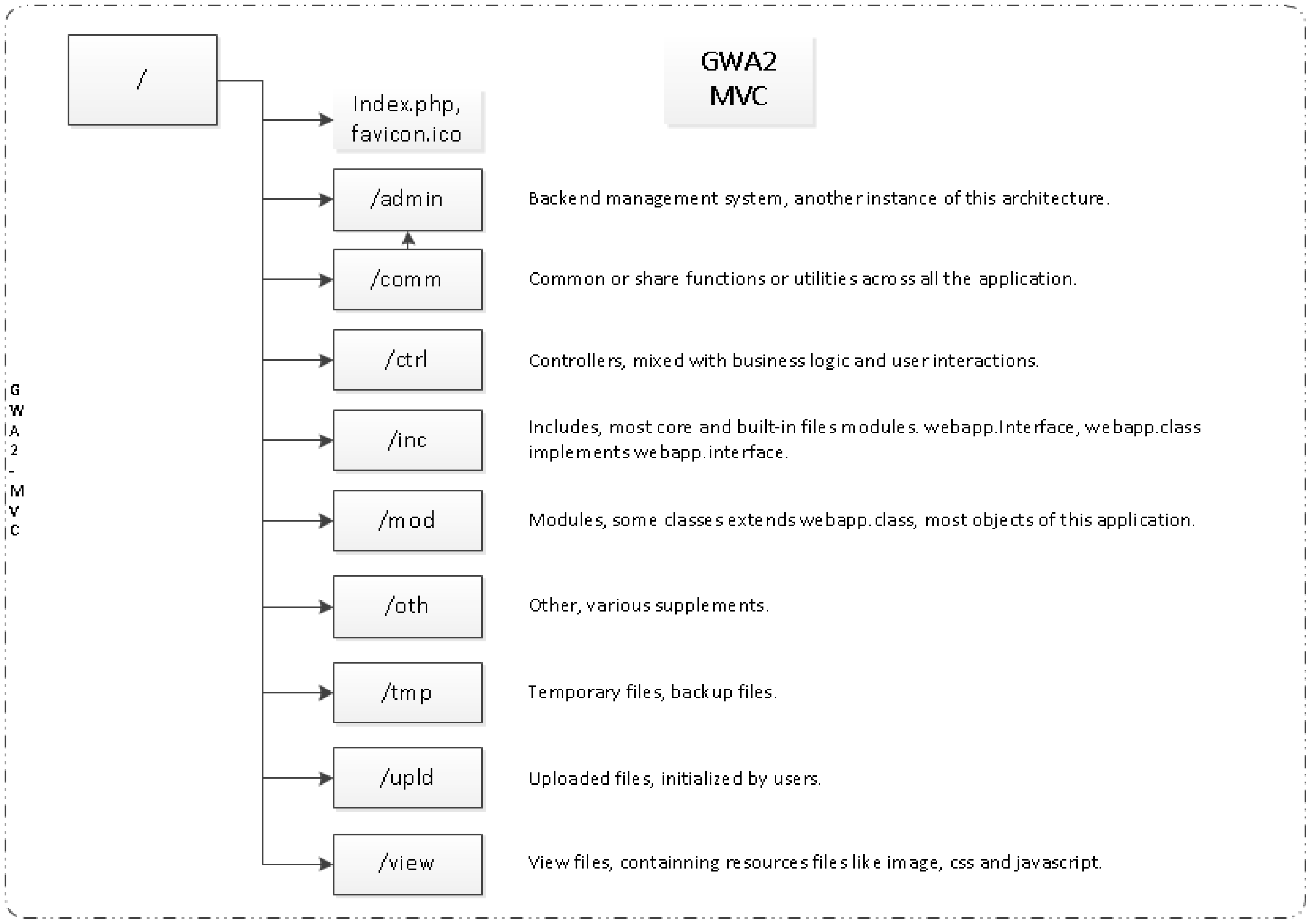
GWA2 MVC
GWA2-MVC
/
Index.php, favicon.ico
/admin
/comm
/ctrl
/inc
/mod
/oth
/tmp
/upld
/view
Backend management system, another instance of this architecture.
Common or share functions or utilities across all the application.
Controllers, mixed with business logic and user interactions.
Includes, most core and built-in files modules. webapp.Interface, webapp.class implements webapp.interface.
Modules, some classes extends webapp.class, most objects of this application.
Other, various supplements.
Temporary files, backup files.
Uploaded files, initialized by users.
View files, containning resources files like image, css and javascript.

Figure 2 directories of an application

As it has been said in last section, if one deployment of template-engine scheme is chosen, MVC architecture will be followed. Mush has been discussed on MVC[3]. There are some similar derivatives during the long evolution of MVC concepts. Though changing gradually, there are two core points, born with MVC, which any implementation of MVC architecture should bear in mind.

- **Separation of concerns.**
  MVC help separate concerns from each other among the anticipators of software engineering which include Project manager, Architecture, coder, UI designer, DBA, tester, etc. Under MVC, it is easy to see that UI designers work on V, DBA works with M, and Project manager and coder work on C.
- **Code reusability.**
  Code reusability contains two parts: 1) some common and share objects or functions can be used across all the application even other similar applications; 2) one module can serve various views with different tailored styles, e.g. one backend server replies requests from different clients, PC, pad, and handsets.
  Even more powerful it is if matching MVC with object-oriented programming environments.

For this architecture, the figure above shows the directories working with an application.

- Index.php, favicon.ico
  These two files are two mandatory for an application. Index.php is the entry of this application; each request should be access via the URL like
  /index.php?para1=value1&para2=value2…
  Favicon.ico is an icon for the application or a website.
- /admin
  This is the backend console system for the application. Whatever size or type an application/website is, it should have a backend management system. /admin serve this purpose.
- /comm
  Header.inc, footer.inc, tools.function…
- /ctrl
  Including mod/xxx.class, redirecting….
- /inc
  Webapp.interface
  Webapp.class implements webapp.interface.
  Core files, core services, DB, Cache, Session, etc.
- /mod
  a.class extends webapp.class, b.class, c.class….
  This sub-directory also contains some 3rd-party components. E.g. in a widely-used deployment, Smarty classes would be placed under this directory.
- /oth
- /tmp
- /upld
- /view

[3] http://st-www.cs.illinois.edu/users/smarch/st-docs/mvc.html

Html, image, css, JavaScript, media…

For each instance, these directories may vary according to application-specific environments.

A standard flowchart of any given request:

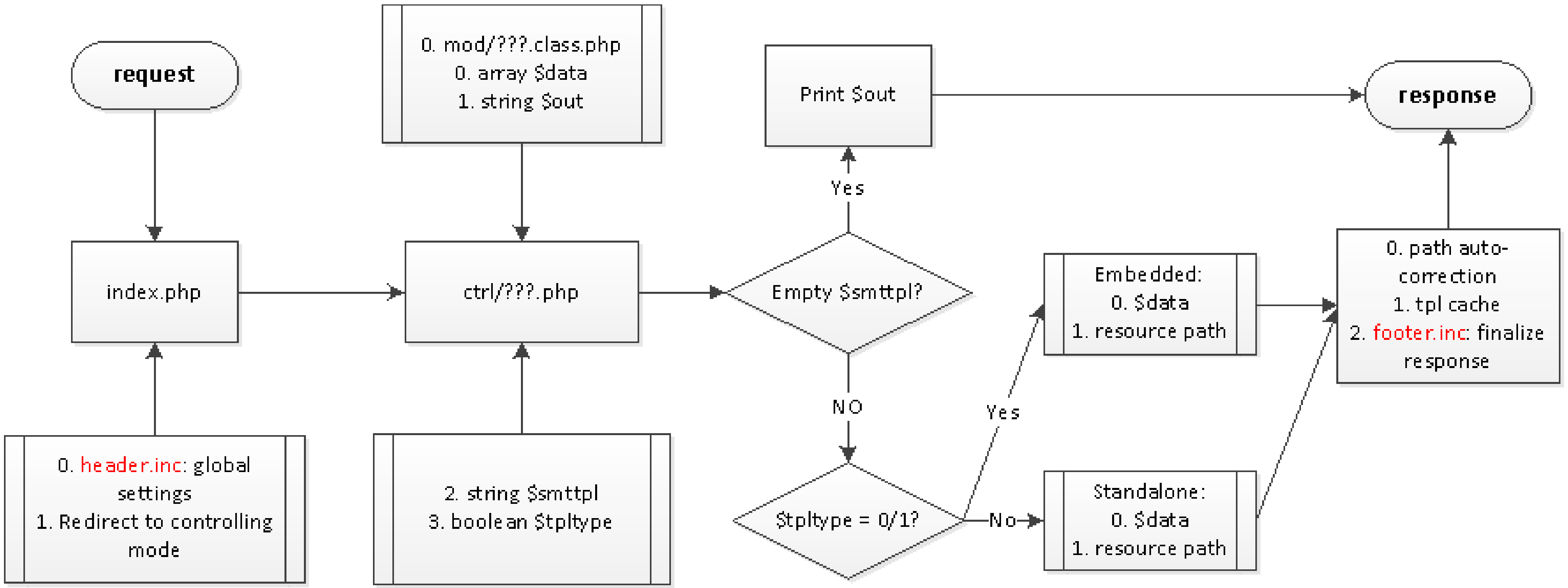

Figure 3, Flowchart of a request-response, a general request URL will consist of index.php?mod=xxxx&act=yyyy….

9. Page navigator

In all cases where GWA2 have been employed, there are many functions which need page navigators to list items more than one page. Page navigator works in almost every place online, therefore there are many ways to so do. The commonest method is a google-like page navigator, which lists a few last pages and next pages near the current page. This is also the default style GWA2 uses.

Page navigator is located in mod/pagenavi.class.php.

Beyond the default page explorer, GWA2 also provides some specific functions in order to make the developing work go fast. Here are some keywords GWA2 employs to join search and page navigator together. Some examples are also listed after these keywords to demonstrate how to use these keywords in actual developing environments.

9.1. Keywords used in the Page navigator

1) pn, page navigator
2) pnpn, page navigator page number
3) pnps, page navigator page size
4) pntc, page navigator total count, number of items
5) pnsk, page navigator search key, this key will be the field of target being searched table.
6) oppnsk, operator of page navigator search key, the operators include:
    - for number
        - =, equal to
        - >, greater than
        - >=,
        - <, less than
        - <=,
        - in list, equal to one of a list, e.g., '1,2,3,5'
        - inrange, greater than the min, and less than the max, e.g., '89, 156'
    - for string
        - contains, substring in any place of the target string,
        - notcontains,
        - inlist, equal to one of the given list, e.g., 'ab, cd, e, fgh'
        - startsWith,
        - endsWith,
7) pnsm, page navigator search mode, 'or|and'
8) pnsc, page navigator search condition, some pre-composed sql-like string
9) pnsck, page navigator search condition key, a key will be used to validate the requested pnsc is legal or not, signed by some encryption method.

9.2. Rules and examples

1) pnpn and pnps consists of the basic function of page navigator, though pnpn and pnps are given default values, they are recommended to be used as mandatory in a web application.

2) pntc is used to avoid the second time to calculate total count of items matching some searching criteria. This is an enhanced point over some other page navigators.
3) pnsk and oppnsk are used in a parallel mode. E.g.,
index.php?mod=user&act=list&pnskname=james&oppnsk=contains…
will search the user table where the field name like '%james%'. If oppnsk is not given, the default value is '=', meaning 'equal to'.
A request URL could contains more than two pairs of pnsk and oppnsk.
index.php?mod=user&act=list&pnskname=james&oppnsk=contains&pnskage=20,40&oppnskage=inrange…
will yield a sql like "select * from usertbl where name like '%james%' or age>=20 and age<=40".
If pnsm is not specified, the default value is 'or', so the above URL could be improved as:
index.php?mod=user&act=list&pnskname=james&oppnsk=contains&pnskage=20,40&oppnskage=inrange&pnsm=1…
its searching conditions will be joined by 'and'.
4) pnsc is an extension to the page nagivator. In most cases, pnsc will not be used due to that pnsk and oppnsk could meet almost every query demand. However, in some circumstances, e.g.,
*"select * from usertbl where (sex=1 and age<20) or (sex=0 and age>20)"*
It is very hard to pre-compose the request URL by this page navigator because there no keywords designated to describe the logic of brackets.
Here comes pnsc. The query conditions can be append in URL directly with the name 'pnsc' like this one
index.php?mod=user&act=list&pnsc=URLENCODE('(sex=1 and age<20) or (sex=0 and age>20)')&pnsck=ENCRYPT(pnsc)….
The request will yield the SQL as
*"select * from usertbl where (sex=1 and age<20) or (sex=0 and age>20)"*
That is to say, pnsc will be treated as the pre-composed where party of the SQL and pnsck is used to make sure that this pnsc is issued and signed by a trusted source.

10.Other issues

10.1. Query with multiple tables

The main concern about this design falls on SQL query with multiple tables after the draft of this design has been shared with colleagues online. It is reluctant to handle this sort of problem. There are two points against further thinking in this way.

- In object-oriented environments, joining two objects will result in creating the third new object, which has no definition or only temporarily existing for a short while.
- SQL is not C. Therefore any attempting to sophisticate the SQL sentence will lag down the speed of query in backend. And join-like queries put more weight on the search engine.

So there is less work about this kind of queries. But it is also on demand that sometimes a join-like query will be needed regardless of any consideration on performance. Here is an example to achieve a query with two more tables in a single SQL sentence.

```
$obj = new WebApp(); # strange? That is top parent object.

$obj->setTbl("usertbl  as A, wishtbl as B"); # two tables
$obj->set("A.id",$user->getId());  # default is “id”, not “A.id”, so this step is needed.
$hm = $obj->getBy("A.id,B.title",   "A.id=B.userid  and A.id=?");
If($hm[0]) {
            //- succ...
            $hm = $hm[1];
} else {
            //- fail...
}
```

10.2. i18n, Unicode, UTF-8

For any website it has been taken for granted that it is a global website, not a local site. Its audience come from all around the world and it is highly-expected to provide mother-tongues for different people. Therefore the web application will have to support internationalization.

Taking this into consideration, there is no better option than Unicode/UTF-8 so far. UTF-8 has been widely-used almost in every major software or system and been spread in a prevalent way. Some comparisons between UTF-8 and other formats like single-byte encodings, multi-byte encodings, UTF-16 can be found at Wikipedia[4].

It can be concluded that deploying UTF-8 in both front-end and backend is a must-be precondition for a global web application with capacity for multi-language support. At least, these settings should be implemented in the following areas:

- Operating system
  - File system, default encode
  - User environment, LC_ALL
- Database
  - Server-wide, compiling with charset
  - Database-level, default charset
  - Table-level
  - Field-level
- Services settings, default charset
- Front-end, page code, charset
  - Html page code, charset
  - HTTP charset

11. To-do

11.1. In LAMP, MySQLi[5] or PHP Data Objects (PDO)[6]?

11.2. Adding rollback() in DBA?

[4] http://en.wikipedia.org/wiki/UTF-8
[5] http://cn2.php.net/manual/en/mysqli.query.php
[6] http://cn2.php.net/manual/en/pdo.query.php

11.3. Persistent connection with backend services: to Databases, to Session Servers, to Cache Servers, connection pool.

12. Document history

Table 1 document history

No.	Version	Updates	Date	Author(s)	Inspector(s)
1	v0.1	Initial draft	2011-07-25	Zhenxing Liu	
2	V0.1	Issues on Query with multiple tables	2011-07-28	Zhenxing Liu	Hao
3	V0.1	Adding section 6.2. i18n	2011-08-01	Zhenxing Liu	Jason
4	V0.2	Adding section “Output, response, view”	2012-10-21	Zhenxing Liu	
5	V0.2	Adding section “MVC”	2012-11-04	Zhenxing Liu	
6	V0.3	Adding page navigator	2012-12-02	Zhenxing Liu	
7	V0.4	Class dir renamed to mod	2013-01-06	Zhenxing Liu	
8	V0.4	Flowchart upgrade to v2	2013-01-20	Zhenxing Liu	