自然中没有任何偶然的东西。There is no accident in nature. –Baruch Spinoza《伦理学》

From interface to interaction

General Web Application Architecture

GWA2

通用网络应用架构系统

Zhenxing Liu

wadelau@{ufqi, gmail, hotmail}.com

(Working with zoneli.com, ufqi.com)

Update Oct. 2016

太阳底下没有新鲜的事。There is nothing new under the sun. – Bible 《圣经》

(This page is intentionally left blank)

(This document contains three sections marked as A, B, C)

（本手册分为 A、B、C 三个部分）

Contents/目录

A. 框架系统开发说明/Developing Sections

1. GWA2

GWA2，通用网络应用架构（General Web Application Architecture）系统，是 一套软件开发框架系统（Framework），主要面向于网络应用软件及系统的开发，应用于网络应用软件及系统的开发，包括但不限于各种 B/S（浏览器/服务器）、C/S(客户端/服务器)架构的软件系统。它提供应用软件通用的、基础性的结构设计、组件及类库的封装。基于GWA2，软件开发人员可以快速地构建各种网络应用软件系统，开发人员由于无需关心结构设计、基础及通用组件及类库，因此可以将更多的精力放在对具体业务需要的解析和实现上，从而提高了生产效率。

IT 语境中的框架（Framework），是指为解决一个开放性问题而设计的具有一定约束性的支撑结构。在此结构上可以根据具体问题扩展、安插更多的组成部分，从而更迅速和方便地构建完整的解决问题的方案。框架本质上是整个或部分系统的可重用设计，表现为一组抽象构件及构件实例间交互的方法。GWA2 作为一种软件开发框架，也具有如下特征：1）框架本身一般不完整到可以解决特定问题；2）框架天生就是为扩展而设计的；3）框架里面可以为后续扩展的组件提供很多辅助性、支撑性的方便易用的实用工具（Utilities），也就是说框架时常配套了一些帮助解决某类问题的库（Libraries）或工具（Tools）。

软件架构系统与建筑设计设定建筑项目的设计原则和目标，作为绘图员画图的基础一样，一个软件架构或者系统架构陈述软件架构以作为满足不同客户需求的实际系统设计方案的基础。从和目的、主题、材料和结构的联系上来说，软件架构可以和建筑物的架构相比拟。一个软件架构需要有广泛的软件理论知识和相应的经验来实施和管理软件产品。软件架构系统定义和设计软件的模块化，模块之间的交互，用户界面风格，对外接口方法，创新的设计特性，以及高层事物的对象操作、逻辑和流程。

GWA2 目前提供了 –PHP 和 –Java 两种编程语言的实现版本，开发人员可以根据项目需要选用。在 –PHP 方面，与之对标的或者类似的开发框架包括但不限于 Zend Framework、Yii、Code Igniter、Lavarel、Symfony2 等等。

在 -Java 方面，与 GWA2 对标或者类似的开发框架包括但不限于 Structs、JSF、Spring MVC、Wicket、Stripes、Seam 等等。

GWA2 的核心思想是，实现较基本的的类库封装、较直白的路由及运行时环境，在便捷性和用户学习成本之间寻找和选择某种平衡。GWA2 最早来源 Tom.com 的业务系统的一些思考、ChinaM.com 的业务系统的一些小结及心得，也有部分来自 People.cn 软件设计思路的参考，在多年实践的基础上，随着面向对象编程思想的成熟，GWA2 也逐渐显现其雏形。

基于这些设计思想，还可以进一步地移植到 ASP.NET 及其他相关开发语言，实现相应版本的开发框架系统。

便捷性是指 GWA2 实现了一些基础类库和组件；用户学习成本较低是指这些设计和类库的实现，采用了较直白和简洁的做法，一目了然，一看即知、即会。没有过多的封装和嵌套，也不会为了面向对象而迂回曲折。

GWA2 的口号是：E.A.S.Y， Easy Along, Swift Yield，轻松启动，快速产出。

GWA2 的标志是：

Figure 1. GWA2 标志

GWA2 标记寓意是一篇绿叶，象征生命、活力和环保。“G”是通用（general），也是绿色（green）。

GWA2 还有个俗称，或者说是英文简写的音译为“吉娃兔”，使用自然域名 –吉娃兔 可以访问到 –GWA2 的官方网页，-GWA2 也是该框架官网地址的自然域名表述。

-GWA2 的官网首页附图如下。

Figure 2 GWA2 官网

2. GWA2 系统下载与安装

2.1. 下载

GWA2的软件程序可以通过如下两种方式获得。

一是在软件源代码托管网站 –GitHub 上直接下载，-GWA2 在 –GitHub 上的地址为：

https://github.com/wadelau/GWA2

安装程序程序的脚本 install.php 的具体方位路径（以 –PHP 为实现为例）：

https://github.com/wadelau/GWA2/blob/master/php/install.php

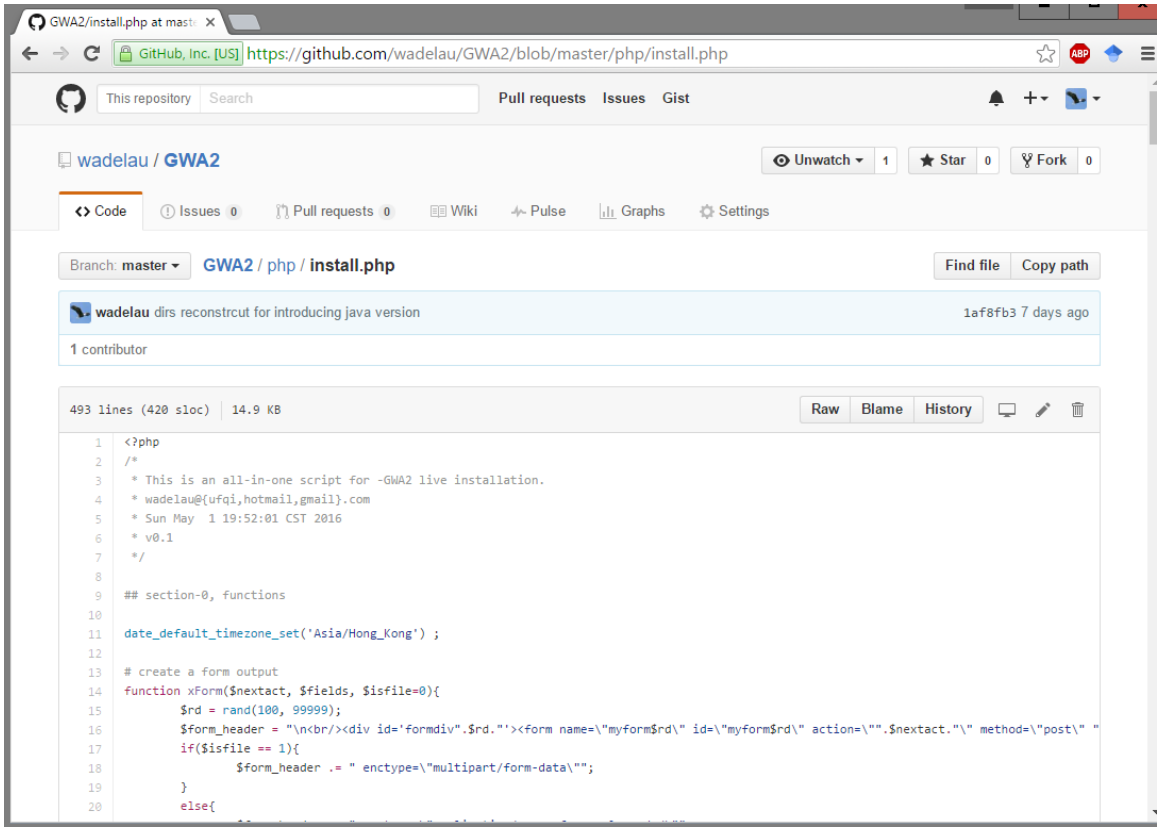


Figure 3. 安装程序在 -GitHub 上的代码托管

获取 GWA2 的另外一种方式是通过发送电子邮件给软件开发团队联络人（电子邮件：wadelau@ufqi.com）来获得完整的 GWA2 的软件程序代码。

开发人员在下载 GWA2 时需要区分所需要的软件开发环境是 –PHP 或者 –Java，目前 GWA2 对这两种主流开发语言均提供了相应的实现版本。

2.2. 安装

GWA2 提供了自动在线安装程序，将 install.php 脚本程序下载并放置到目标安装目录下，通过浏览器访问该脚本，运行 install.php 即启动安装程序。

Figure 4. GWA2 installation (1)

安装程序首先检查当前安装环境，目前的可读及可写；然后进行源代码的下载，源代码从 –Github-Wadelau 上读取，下载到 Zip 压缩包后，使用系统自带的相关程序进行解压，然后进行数据库的配置、样例模块 Hello 的初始化设置。

安装程序最后会引导到项目的初始页面。install.php 详细构成部件及构成请参考下图。这一过程是完全自动化的，类似我们常规软件安装程序一样，一直点击下一步、下一步即可完成所有步骤的安装、设置。

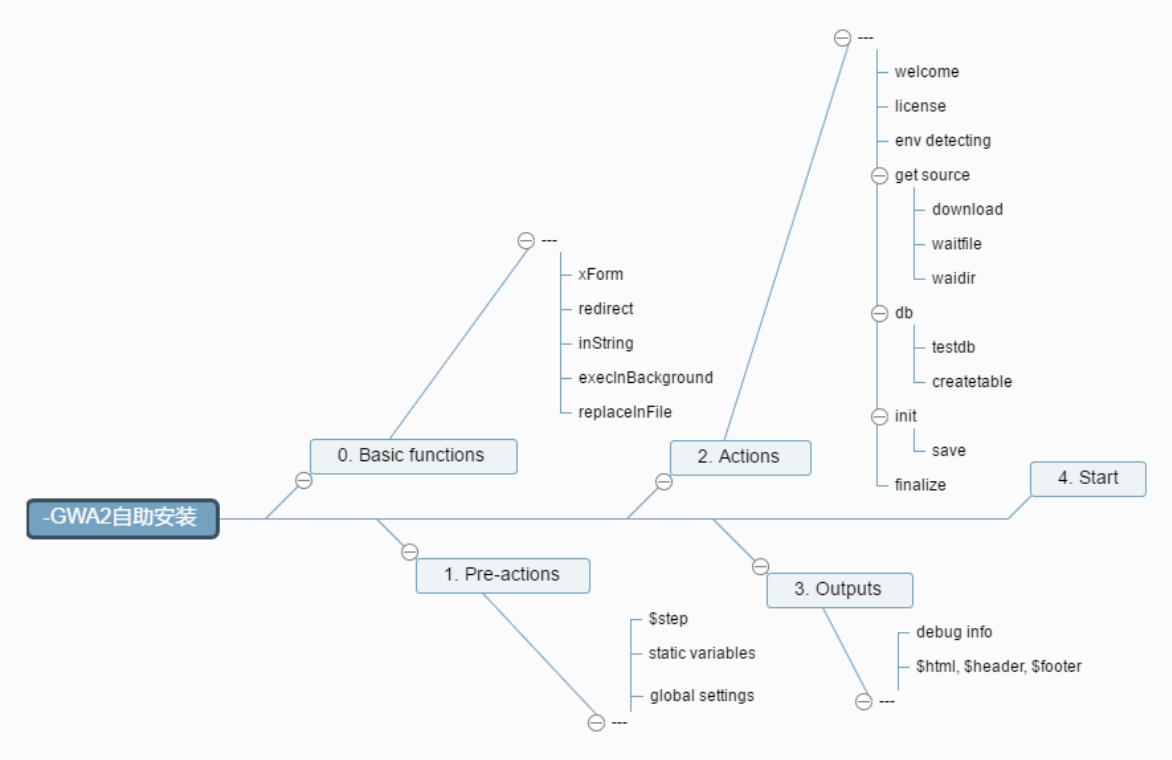


Figure 5. GWA2 installation (2)

安装程序依赖操作系统所提供的一些操作命令：

- rm
- mv
- wget
- unzip

安装过程大致分为五个部分：

首先，在 GWA2 的安装程序中定义了一些基本和公用的函数，这些函数包括连接数据库、文件读写、可执行程序的调用等；

第二部分是与真实安装相关的一些设置项，如接受用户使用协议，定义安装所需的静态变量和全局变量等；

第三部分是 GWA2 安装的各个子步骤，其中又可以细分为：

- 安装环境初始化、读写环境探测等；
- 主体程序文件的下载、解压与部署等；
- 连接数据库所需的信息录入、初始及样例数据表的创建等；
- 初始化运行环境的设置及其保存；
- 完成安装、引导用户至项目初始页面。

第四部分是与安装过程中的输出设置相关的，如进度条、页面头部、尾部及过程页面的布局及内容显示设置等；

第五部分是项目初始页面的引导设置。

安装完成后，用户可以根据引导进入一个 GWA2 的样例系统。文档后续将从各个特定的角度，阐述 GWA2 的一些特色功能及其使用、用法。

3. 核心：五层架构设计

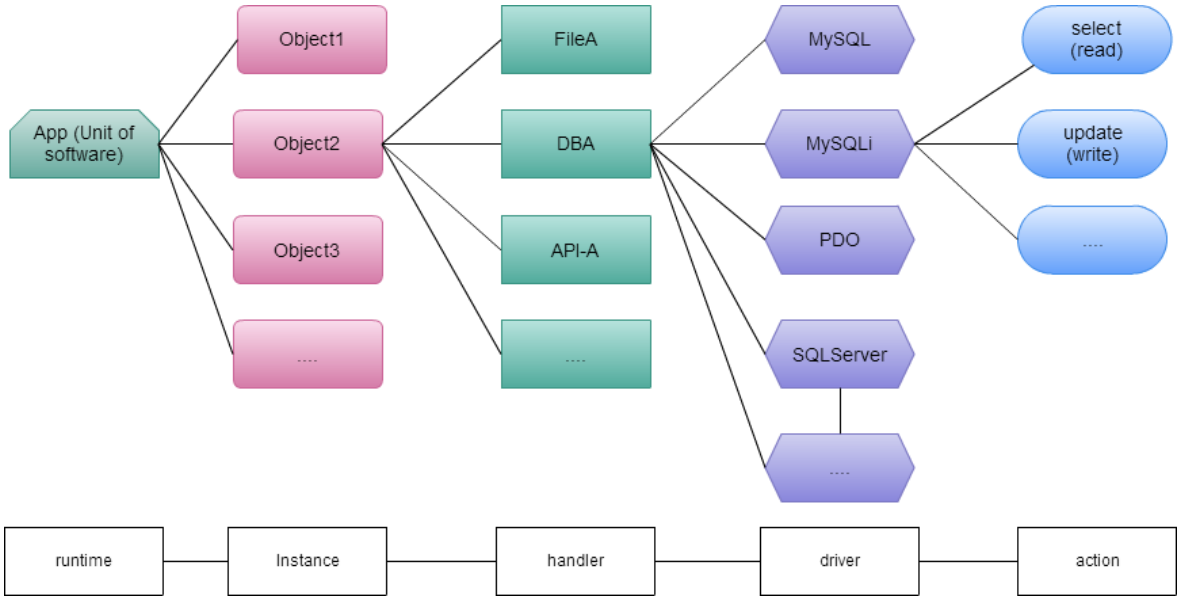


Figure 6 GWA2 的核心五层设计

App（Application，软件）是对一个应用程序的统称，从更广义的视角考察软件，可以从五个层面来认识一个软件，即：运行时（Runtime）、实例（Instance）、处置器（Handler）、驱动器（Driver）和行为动作（Action）。以下以 GWA2 的多个数据库支持的思想来说明，应用软件五层设计说涉及到的逐个层面。

第一层面是 App，也即在客观世界中，一个具有各个对象（Object）的有机整体，互相之间协作，共同完成某一个或者多个功能，满足某一类或者多类需求。

第二个层面是对象（Object），也即一个软件中具体的业务实例，可能包括用户、文章、商品、交易、订单、支付、快递等等。

第三个层面是组成对象或者服务于对象的具体处置器，如一个用户对象，可能需要读写文件系统、读写数据库系统、读写内外部的各种 API 等，进行数据交互或者状态转移。

第四个层面是进一步的细化到服务于处置器的驱动部件，这些部件是在某一类处置器之下，负责具体的实现，如以读取数据库为例，对象读取数据库的请求统一发给 DBA，然后 DBA 进一步地分析，是连接哪一种数据库来实现对数据读写，可能备选的包括 MySQL、MySQLi、PDO、SQLServer、Oracle、Postgres 等等。

第五个层面是描述驱动部件中的具体实现，如读的行为，或者写的行为。

结合到 GWA2 的代码层面的思想为如下思路：某个 Object 使用 getBy/setBy 的方法进行信息的请求，getBy/setBy 根据设置，分发不同的请求给不同的处置器，处置器根据用户的指派，进一步地选择相应的驱动器，由驱动器实现具体的读写动作。

如下是一个实现缓存服务的设计及其实现。

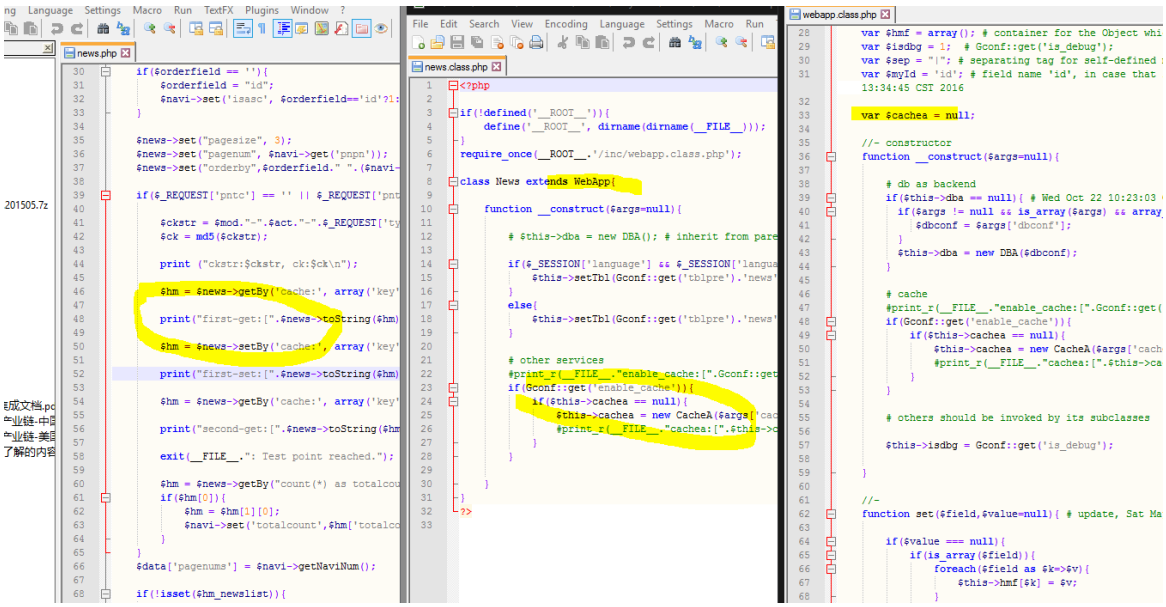


Figure 7 GWA2 五层设计的 Cache 服务实现（1）——左：对象控制器，中：对象模块类，右：模块类的父类

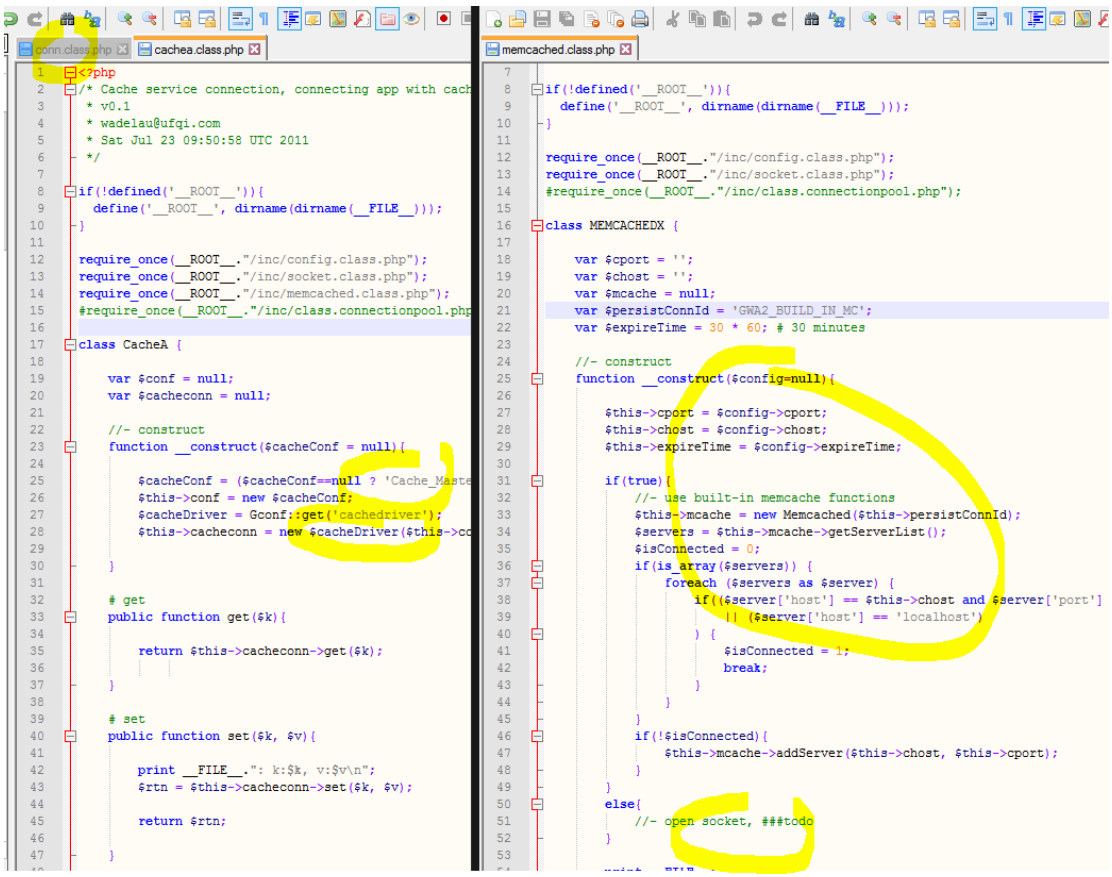


Figure 8 GWA2 五层设计的 Cache 服务实现（2）——左：缓存处置器，右：缓存驱动器

在第一张图中，对象 $news 在控制器中通过 getBy('cache:', $args) 的方法调用缓存（左），该请求进一步地递交给 $news 的模块类（中）， $new 模块类将请求前转给其父类 $webapp，在$webapp 中声明了缓存的处置器 $cachea（右），父类$webapp，进一步地将请求转交给缓存处置器 $cachea；

在第二张图中，$cachea 接到请求后（左），根据配置信息，初始化相应的缓存驱动器 $memcached（右），缓存驱动器 $memcached 被调用后，具体负责对请求的处理，也即对数据的读写。

梳理相关流程即为：

$news.getBy → $webapp.getBy → $cachea.get→ $memcached.get

对应程序代码中的截图为：

模块控制器（ctrl/news.php）→ 模块类（mod/news.class.php）→ 模块类父类 (inc/webapp.class.php) → 缓存处置器(inc/cachea.class.php) → 缓存驱动器 (inc/memcached.class.php)

4. 使用：数据从后台到前端

如下演示开发一个基于 GWA2 框架系统的模块的主要步骤，这些步骤概括为：创建面向对象的数据表；根据数据表，创建继承父类 webapp.class 的子类模块；在路由及控制器目录创建请求处理程序脚本；在控制器脚本中实现对子类的实例化，并实现读写数据；最后将数据输出到模板引擎，由模板引擎调用终端呈现页面返回给调用者。如下是一个具体的实现实例，供参考。

在以数据库为驱动后台的 WebApp 中，以展示广告为例，来演示数据从数据库后台到用户前端的过程，这些过程在 GWA2 中的流转如下。

1） 首先，在数据库中创建广告数据存储表，这一步通常还需要在以 –gMIS 为管理后台的界面中增加对广告数据的可视化管理界面；

2） 其次，创建 mod/ads.class.php 模块；

3） 然后，在控制器目录主文件 ctrl/index.php 中将对应的 mod 模块 include 进来，这里主要考虑到广告展示会显示在所有页面，所以会将该模块在主控文件中加载，若不需要在所有页面展示广告，则该模块不需要在主控文件中加载；

include($appdir."/class/ads.class.php");

4） 实例化广告 mod/ads.class，并读取；

```
……
$ads = new ADS();
$hm_homepage_adlist = $ads->getBy('*',
"adplace='homepage' and state=1");
$data['hm_homepage_adlist'] = $hm_homepage_adlist;

……
```

5） 在前端使用模板引擎（Smarty）语句循环 $hm_homepage_adlist。

```
……
<ul>
      {foreach $hm_homepage_adlist as $k=>$v}
            <li><a href="{$v['link']}"><span
class="img_1"><img src="{$v['imagepath']}"
/></span></a></li>
      {/foreach}
      <li><span class="img_2"><img
src="images/photo/banner-2.jpg" /></span></li>
      <li><span class="img_3"><img
src="images/photo/banner-3.jpg" /></span></li>
</ul>

……
```

5. 使用：请求处理流程

GWA2 集成了路由功能，用户的请求统一发送至入口程序 index.php 或其基于服务器端的软连接上，经由该程序对请求进行初始化处理，根据 mod 和 act 参数进行路由定位，然后加载相应的模块和动作进行用户请求的处理，整体看来，（以用户模块为例）用户处理流程如下。

1）用户请求统一入口地址

index.php?mod=xxxx&act=yyyy

在启用 RESTful 风格的 URL 中，该地址或是

index.php/mod/xxxx/act/yyyy

i/mod/xxxx/act/yyyy

其中 “i” 是 index.php 在服务器端的软连接；

2）请求经过 index.php 分发给 ctrl/ 目录下的对应 mod 控制器，如 ctrl/user.php

```
# the application entry...

# main logic
$mod = $_REQUEST['mod']; # which mod is requested?
$act = $_REQUEST['act']; # what act needs to do?

if($mod == ""){
  $mod = "index";
}

# header.inc file
include("./comm/header.inc");

$data['mod'] = $mod;
$data['act'] = $act;
$data['baseurl'] = $baseurl;

if(file_exists("./ctrl/".$mod.".php")){
      include("./ctrl/".$mod.".php");
}
else{
      print "ERROR.";
      error_log(__FILE__.": found error mod:[$mod]");
      exit(0);
}
……
```

3）控制器 ctrl/user.php 中调用模块 model 的类，如 class/user.class.php 或者 mod/user.class.php, 实现改变对象的状态等动作；参考 A 部分的第 16 节的代码示意；

4）在控制器 ctrl/user.php 中完成业务逻辑的处理，设置输出的 $data 数组 和 模板文件 如 view/user.html；

5）模板引擎 （如 Smarty）将 $data 和 view/user.html 模板文件做拼合 merge 操作，最终生成用户端的 html 页面；

6）关键步骤，第 3） 步，控制器里加载模型对象。

6. 核心： getBy/setBy

GWA2 的核心有四个方法，分表是 set/get, setBy/getBy, 其中前一对设计用来处理运行时变量，后一对设计用来对数据进行持久化操作的。

根据需求驱动，-GWA2 扩展增加 read/writeObject 到 setBy/getBy 中，对数据持久化层的访问增加了对缓存系统、文件系统和远程地址接口的支持。

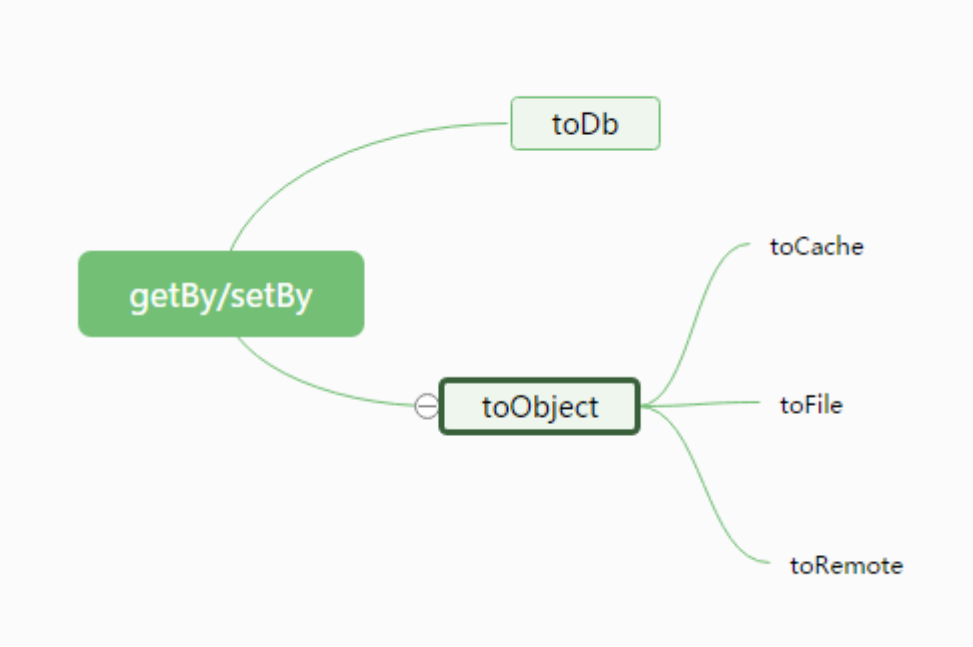


Figure 9 getBy/setBy

在技术实现上， -GWA2 的设计中，此前将所有对对象的改变，默认持续化存储是 数据库，没有深入考虑或者实现其扩展将持久化存储到其他方式方法，比如网络、文件系统等，本周在研发中遇到实例，就顺便将改变状态的主方法 getBy/setBy 做了扩展，使之能够支持读写文件和网络访问。也即，增加了 readObject/writeObject。 其技术细节如下：

1. readObject

   在 getBy 中判断 getBy 的第一个参数 $fields 中是否包括 ":", 如果包括，则视为读取文件系统或者网络系统，而不是数据库系统。

   进一步地，当判断成从非数据库读取时，则分别判断是：

   a) 'file:' ， 这个关键词表示，从文件系统读取;

   b) 'url:' , 这个关键词表示，从网络远程通过 URL 读取；

   c) 'cache:', 这个关键词表示，从缓存中读取。

2. writeObject

   同样地， 在 -GWA2 中，在 setBy 方法中，判断其第一个参数 $fields, 如果保护有 ':' 关键词，则视为存储持久化到非数据库系统，由此将处置权交给 writeObject. 在 writeObject 中进一步地判断:

   a) 若存储为 'file:' 关键词，则调用文件写操作系统；

   b) 若存储为 'url:' 关键词，则存储为进一步调用远端接口，此时为触发 HTTP POST/GET 方法，封装细节见程序代码；

   c) 'cache:' , 这个关键词表示，从缓存中读取。

3. 调用样例

   调用上述写入或者读取非数据库的持久化趁，使用:

   $obj = $webappInstance->getBy('url:', $args);

   $obj = $webappInstance->getBy('file:', $args);

   $obj = $webappInstance->setBy('url:', $args);

   $obj = $webappInstance->setBy('file:', $args);

   $obj = $webappInstance->setBy('cache:', $args);

   $obj = $webappInstance->setBy('cache:', $args);

   其中，当 'file:' 时，

   $args = array('target'=>'xxx', 'content'=>'xxx', 'islock'=>1, 'isappend'=>1);

   当 'url:' 时，

   $args = array('target'=>'xxx', 'method'=>'xxx',
   'header'=>array('headerxxx'=>'headerxxx_value'),
   'parameter'=>array('parameter_1′=>'parameter_1_value'));

   当 'cache:' 时，

   $args = array('key=>'xxx', 'value'=>'xxx');

   上述方法的返回值，遵循 getBy / setBy 的定义，统一规范为

   return array(boolean <true|false>, array('content'=>$returnObject,

'errorcode'=>'xxxx', 'errordesc'=>'xxxx'));

其中， 成功 true 时，只有 content 的键值; 失败 false 时，只有 errorcode 和 errordesc 。

如下是几个实例:

```
# read from an external src
# $adoffer is a sub class extending WebApp
$myObj = $adoffer->getBy('url:', array(
	'target'=>HASOFFERS_API_URL, 'parameter'=>array(
		'NetworkId' => 'xxxxx',
		'Target' => 'Affiliate_Offer',
		'offer_id' => $k, //-
		'params' => array(
			'sub_id' => '{channel}',
			'info'=>'{info}'
			),
		'options' => array(
			'tiny_url' => 0
			)
		)
	));
if($myObj[0]){
	$myObj = $myObj[1]['content'];
}

# save to file
# $adoffer is a sub class extending WebApp
$file = '/tmp/offer_'.$k.'/'.$k.'.txt';
print "\nsave to file:[$file]";
$writeResult = $adoffer->setBy("file:", array(
	'target'=>$file,
	'content'=>
"click:[$click]\nname/title:[".$nameArr[0]."]\ncountry:[".$nameA
rr[3]."]\ndesc:[".$ioffer['description']."]",
	'islock'=>1,
	'isappend'=>1,
	''=>''
	));
print "\n";
print_r($writeResult);

# read from file
print "\nread from file:[$file]";
$readResult = $adoffer->getBy('file:', array(
	'target'=>$file
	));
print "\n";
print_r($readResult);
```

Figure 10 setBy/getBy 扩展调用实例

其中，errorcode 是 -GWA2 特有的一种标记系统，系全局所有，用于全局定位。 默认采用 timestamp 的 YYmmddHHMM 十位数字系统。

当编程语言不再是羁绊时，思想和文化变作为主因，比如上面修改中，为何要将 read/writeObject 融入到 getBy/setBy 中，而不是在 WebApp 中直接调用?

7. 格式化输出

如在 B 部分第 18 部分描述的那样，Figure 19 Output in GWA2 所描述的，GWA2 支持多种输出格式，既有基于模板的，也有纯文本输出后者其他格式的输出。根据该图的描述，GWA2 的格式化输出的控制代码如下所示。

```
……
if($smttpl != ''){

	$data['smttpl'] = $smttpl;
	$data['viewdir'] = $rtvviewdir; # where and why?
	$data['rtvdir'] = $rtvdir;
	foreach($data as $k=>$v){
		$smt->assign($k, $v);
	}

	if(1){
		$markfile = $appdir."/tmp/tpl_last_modified.tmp";
		# todo: cache the modified tpl file
	}
	# for conflicts between smarty {} and javascript {},
using {literal}{/literal}
	if($display_style == $_CONFIG['display_style_index']){

		$smttpl = $smttpl.".tmp";
		$smt->assign('smttpl', $smttpl);
		$smt->display('index.html.tmp');
	}
	else if($display_style ==
$_CONFIG['display_style_smttpl']){

		$smt->display($smttpl.".tmp"); # use template file
only

	}
	else{
		error_log(__FILE__.": Something wrong with display
style and smttpl:$smttpl .");
	}
}
else{

	if(isset($fmt)){
		if($fmt == 'json'){
			header("Content-type:
application/json;charset=utf-8");
			print json_encode($data['respobj']);
		}
		else{
			print "Unknown fmt:[$fmt] in output.";
		}
	}
	else{
		print $out;
	}
}

……
```

GWA2 的输出是分层的，首层是判断是否有指定的模板，如果有指定的模板文件，则按模板文件输出；如果没有模板，则进一步地判断是有指定的输出内容格式，如 JSON、XML 等，如果有指定的内容输出格式，则按相应的格式对内容进行格式化操作后，然后输出。

如果没有指定输出模板文件，也没有指定输出内容格式，GWA2 则会默认输出 Plain Text 纯文本格式的内容，如程序所示，输出程序所拼接的内容字符串 $out。

8. 多语言版本实现

如前所述，根据国际化要求，通常多数网站需要进行多语言支持，目前采用的方法如下图所示。GWA2 中也已经实现了对 i18n 的支持。

Figure 11 i18n in GWA2

I18n的主要组成部分和实现流程如下。

1) 在lang这个文件夹下，存储各种语言的propertis文件，像 en_US_Message.propertis

zh_CN_Message.propertis，根据实际需要增加任何一种想要的语言文件。将页面上不能从后台取出却要进行语言转换的词语在这里进行翻译。示例如下：

```
v.common.weibo = Weibo
v.common.home = Home
v.common.aboutus = About As
v.common.businessfield = Business field
v.common.product = Product
v.common.contact= Contact
v.common.news = News
v.common.newsmore = More
v.common.salesoffice = Sales Office
v.common.webservice = Web Service
v.common.jiahuabairui = JiaHua BaiRui
v.common.fdknews = News
v.common.fdkjobs = Job
v.common.fdkcompanyname = Beijing Findik Foods Co. Ltd
v.common.contentmore = More
v.common.protypes = Product
v.common.salesadd = Sales Office
v.common.links = Links
v.common.address = ADD
```

```
v.common.weibo = 官方微博
v.common.home = 首页
v.common.aboutus = 关于芬帝
v.common.businessfield = 业务范围
v.common.product = 芬帝产品
v.common.contact= 联系芬帝
v.common.news = 最新资讯
v.common.newsmore = 更多新闻资讯
v.common.salesoffice = 销售地址
v.common.webservice = 品牌建站
v.common.fdknews = 芬帝资讯
v.common.fdkjobs = 芬帝招聘
v.common.fdkcompanyname = 北京芬帝食品有限公司
v.common.jiahuabairui = 嘉华伯瑞
v.common.contentmore = 更多内容
v.common.protypes = 产品系列
v.common.salesadd = 销售中心
v.common.links = 友情链接
v.common.address = 地址
```

Figure 12 i18n in GWA2 (2)

2) 入口文件index.php会在comm/header.inc引用language.php文件，language.php文件的作用主要有如下：

- 获取页面当前的语言
- 根据不同的语言选择从不同的 propertis 文件中取数据
- 如果网站内容特别多，可以将将当前网站内容分成多个module，比方说common等

等，目前内容不多，就都存储到了 common 这个 module 中了

- 定时将当前语言存储到 cookie 中，以供点击页面各个链接时，保留当前语言不被修改。

3) language.php 会调用 langlist.php 中的函数，langlist.php 目前只提供了一个函数，就是将不同页面分别放到不同的 module 下，目前内容少的情况，可以暂时不考虑。

4) 网站多语言支持流程图

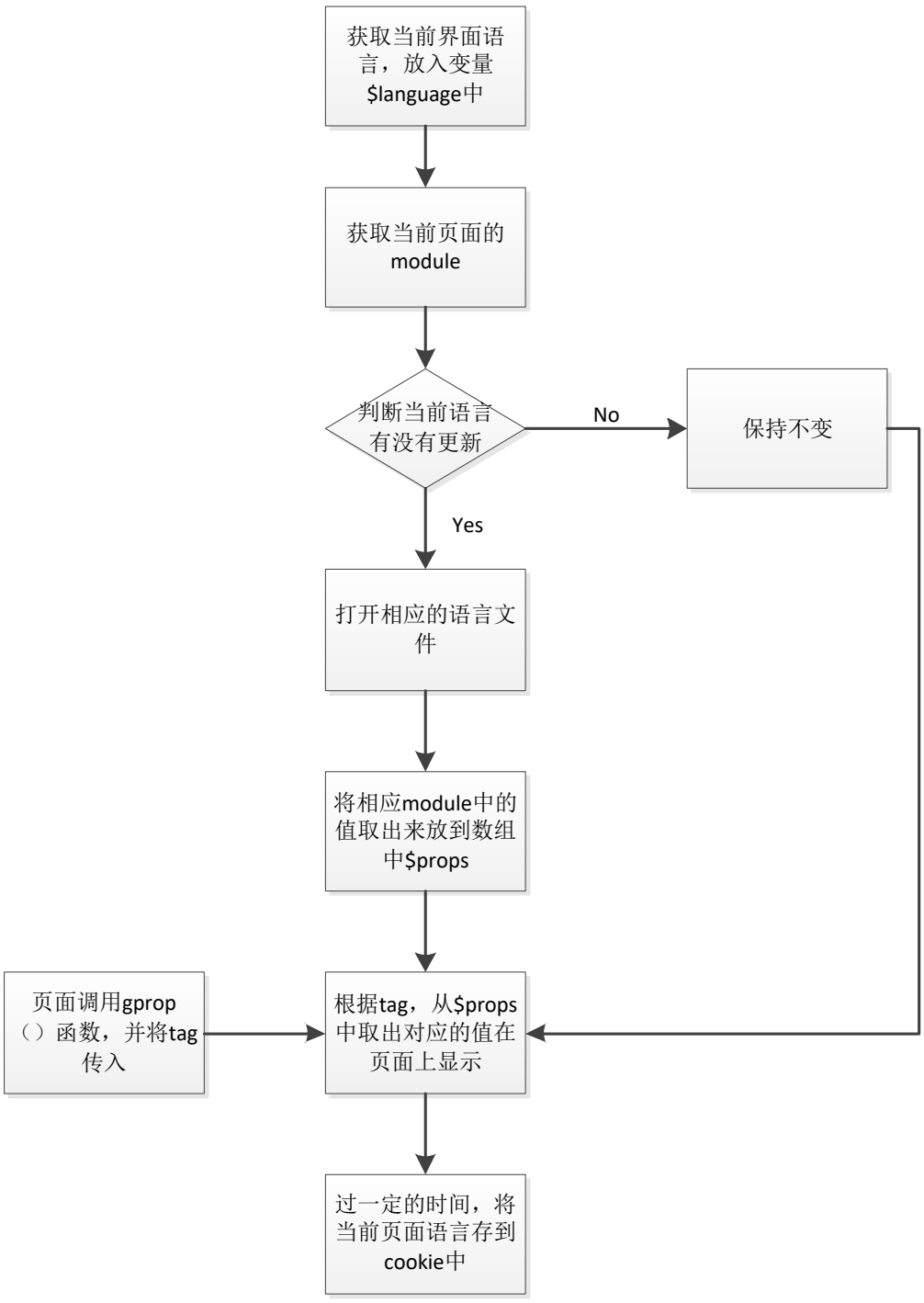


Figure 13 i18n in GWA2 (3)

a. 当进行中英文切换时，在 URL 上增加一个标识，如下：

```
……
<li><a href="{$url}&language=zh_CN">中文</a></li>
<li><a href="{$url}&language=en_US">English</a></li>

……
```

这样切换语言时，language.php 中将会获取到当前语言，并将当前语言存储到 $language 中。同时会定时从页面中获取当前语言存储到 cookie 中。因为当点击其他链接时，url 中将不会有 language 标识，这时就从 cookie 中获取当前语言。

b. 获取当前页面的 module 信息，目前没有多少内容，没有进行 module 的划分，统一从 common 中获取内容。

c. 判断当前语言有没有更新，如果当前语言进行了更新，比方说更新为 zh_CN，则打开 zh_CN_Message.propertis 这个文件，然后根据 module，把当前 module 中的内容取出来存到一个数组中$props。示例如下

```
Array (
	[common] => Array (
		[weibo] => 官方微博
		[home] => 首页
		[aboutus] => 关于芬帝
		[businessfield] => 业务范围
	)
)
```

这里面的 common 即为 module 名称，weibo 等即为 tag

如果语言没有更新，则保持不变仍然从之前的$props 中取值。

d. 页面上需要进行语言切换的所有内容，将通过 gprop(),这个函数来进行获取，例如

```
……
<ul>
	<li><a {if $mod eq "index"} class="active"{/if}
href="{$url}">{gprop("home")}</a></li>
	<li><a {if $mod eq "aboutfdk"}
class="active"{/if}
href="{$url}&mod=aboutfdk">{gprop("aboutus")}</a></li>
	<li><a {if $mod eq "business"}
class="active"{/if}
href="{$url}&mod=business">{gprop("businessfield")}</a>
</li>
	<li><a {if $mod eq "products"}
class="active"{/if}
href="{$url}&mod=products">{gprop("product")}</a></li>
	<li class=" last"><a {if $mod eq "contact"}
class="active"{/if}
href="{$url}&mod=contact">{gprop("contact")}</a></li>
</ul>

……
```

gprop 函数传递的参数为 tag，language.php 中的 gprop()函数将会根据这个 tag 从 $props 这个数组中取出相应的内容。例如 tag 为 home，则英文取出的内容为 Home，中文则为”首页”。

9. RESTful 地址风格实现

表示状态转移（Representational State Transfer，简称 REST）是一种万维网软件架构风格。由于 HTTP 连接是无状态的（也就是不记录每个连接的信息），而 REST 传输会包含应用的所有状态信息，因此可以大幅降低对 HTTP 连接的重复请求资源消耗（**-R/12V7** ）。

符合 REST 设计风格的 Web API 称为 RESTful API。它从以下三个方面资源进行定义：

1) 直观简短的资源地址：URI，比如：http://example.com/resources/;
2) 传输的资源：Web 服务接受与返回的互联网媒体类型，比如：JSON，XML，YAML 等;
3) 对资源的操作：Web 服务在该资源上所支持的一系列请求方法（比如：POST，GET，PUT 或 DELETE）。

基于这些考虑，RESTful 设计的优点也显而易见。

- 可更高效利用缓存来提高响应速度
- 通讯本身的无状态性可以让不同的服务器的处理一系列请求中的不同请求，提高服务器的扩展性
- 浏览器即可作为客户端，简化软件需求
- 相对于其他叠加在 HTTP 协议之上的机制，REST 的软件依赖性更小
- 不需要额外的资源发现机制
- 在软件技术演进中的长期的兼容性更好

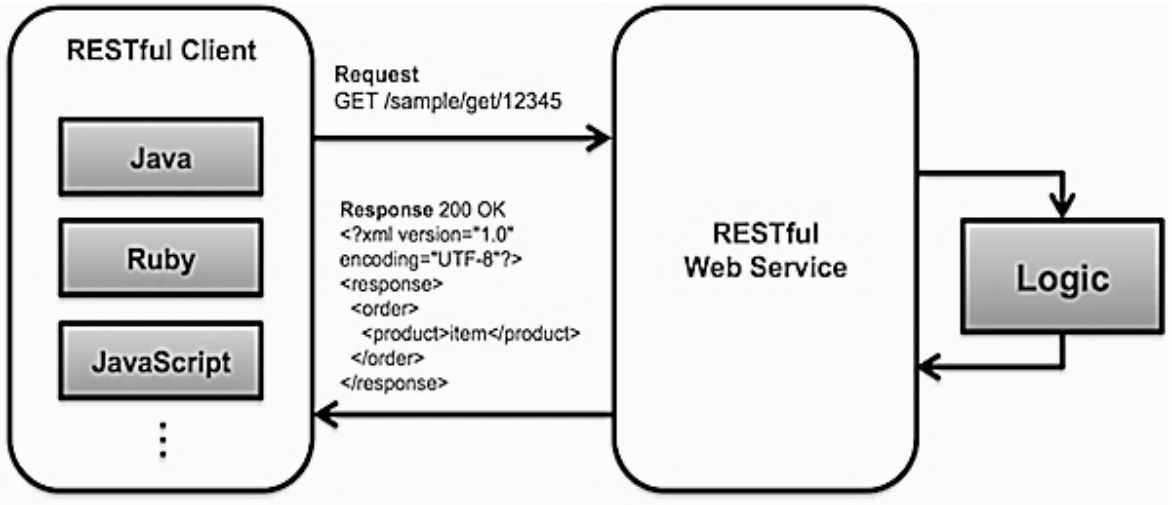


Figure 14 RESTful in GWA2 (-R/72Ti)

GWA2 目前提供了对 RESTful 风格 URL 的支持。

################### 关于 RESTful URL 地址风格的实现, Mon Oct 15 22:15:17 CST 2012, update Sun Jan 24 14:43:59 CST 2016

1) 资源访问路径中的 ? 去掉

此前的

http://ufqi.com/dev/xxx/index.php?mod=web&act=preview&id=1234

RESTful 的

http://ufqi.com/dev/xxx/index.php/mod/web/act/preview/id/1234

或 RESTful 的

http://ufqi.com/dev/xxx/i/mod/web/act/preview/id/1234

use i as a soft link to index.php in server side

2) 规则：”?” 后面的参数，总是成对出现，奇数位的是参数名称，偶数位是参数值；
3) 程序实现：

在后台程序中，使用 $url.”/para/value” 的样式拼合；

在 Smarty 模板中，使用那个 {$url}/para/value 的样式拼合；

在入口程序中，./index.php (i) 对 / 分割的参数重新转为 $_REQUEST 变量，同时重写如下全局变量：

$_REQUEST['para'] = value;

$_SERVER['REQUEST_URI'];

$_SERVER['QUERY_STRING'];

4) 在其他程序中，与普通动态地址一样使用；
5) TODO: 需要对 value 中的 “/” 做转义或者编码；
6) 默认情况启用，since Sun Jan 24 14:00:51 CST 2016 。

10. 对象、单表与多表

在 GWA2 中，对象（Object，位于 module 或者 class 目录下）与表（数据库中的数据表）大致对等，但并不是一一对应关系。一个对象可能需要两个以上的数据表来存储相应的数据，同样的，两个对象，也可能存储于同一张数据表中。通常情况下，前者较为常见，而后者可能是父子对象共宿一表。

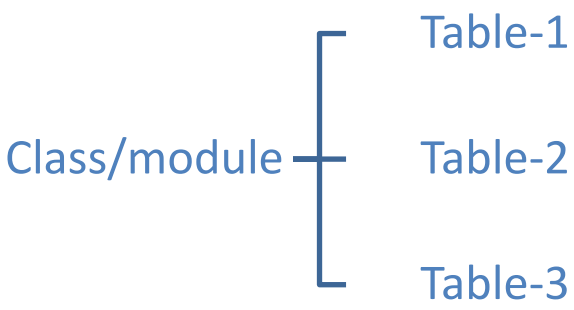


Figure 15 module and tables

在某些场景下，为了读取数据，通常不需要一个单独、明确的对象，或者此时对象不明确，比如读取一张字典表等，可以考虑使用如下方式进行。

以读取数据表 abtctbl 中的数据为例：

```
......
$webapp = new WebApp();
$webapp->setTbl("abctbl");
$hm = $webapp->getBy("*", $condition);
......
```

附件

开发手记：数据表如何联查两个不相关联表的数据？ 多表,联表

（2012 年 08 月 31 日）在数据表操作中，要在两个以上的数据表 tbl, table 中联查，一般使用：

1） select t1.a, t2.b from t1, t2 where t1.c = t2.d;

或者使用 join 关键词

*2）SELECT * FROM t1 LEFT JOIN (t2, t3, t4) ON (t2.a=t1.a AND t3.b=t1.b AND t4.c=t1.c);*

这两种模式都要求，两个表之间有某种关联关系，t1 的某个字段是 t2 中的某个字段，反之亦然。

如果是两个不相关联的表，则不能这样做，否则出来的结果就比较凌乱，取而代之的是 union 这个关键词，用法是：

select a,b,c from t1 union all select e,f,g from t2;

union 也可以写成 union all，不去重， union 要求两个 sql 所取的字段数是一样的，而且结果集里的 meta 信息只有 第一个 子 sql 的。 来个例子：

t1, 学生表
id,name, age, sex
t2, 教师表
id,name, age, sex

我想同时在这两个表里查找 name 里有 “国强”的人，通过 union 可以写成：

select id, name from t1 where name like ‘%国强%‘ union all select id, name from t2 where name like ‘%国强%’;

结果集可能会是这样：

id name
1 张国强
2 李国强
3 刘国强
….

怎么能知道 id=2 的记录是 t1 的还是 t1 的呢？

问题再放大些，如果是 3 个表，t3, 怎么知道当前这个 union 之后的结果集的记录来自哪个数据表呢？

经过一番探索形成如下解决方法：

select concat('aaa',id) as id, name from t1 where name like '%国强%' union all select concat('bbb',id) as id, name from t2 where name like '%国强%';

执行之后的结果集可能是：

id name
aaa1 张国强
bbb2 李国强
aaa3 刘国强
….

这样就知道， aaa1 是来自 t1 是个学生，bbb2 是来自 t2 是个老师.

也即在结果集的关键字段上打上数据表的标签。

一种更通用的多表联查的实现，源码见下。

```
class Search extends WebApp{

	var $mainTables = array("livetbl"=>array("id"=>"id",
"title"=>"name", "searchkey"=>"title"),
		"producttbl"=>array("id"=>"id","pname"=>"name",
"searchkey"=>"pname"),
		"wanttbl"=>array("id"=>"id", "wname"=>"name",
"searchkey"=>"wname"),
		"usertbl"=>array("id"=>"id", "nickname"=>"name",
"searchkey"=>"nickname")
		);

	//-
	function __construct(){
		$this->dba = new DBA();
		# no tables set

	}

	//-
	function getList($kw, $scope=''){
		$sql = ""; $pagesize = 500;
		foreach($this->mainTables as $k=>$v){
			#print __FILE__.": k:[$k] , v:[$v].";
			if($scope != ''){
				if(strpos($k, $scope) ===
false){ continue; }
			}
			$sql .= "select ";
			foreach($v as $k2=>$v2){
				if($k2 == 'searchkey'){ continue; }
				else if($v2 == 'name'){
					$sql .= "concat('".$k."_',$k2) as
$v2,";
				}
				else{
					$sql .= "$k2 as $v2,";
				}
			}
			$sql = substr($sql, 0, strlen($sql)-1)." from
".Gconf::get('tblpre').$k." ";
			$sql .= "where ".$v['searchkey']." like '%$kw%'
";
			$sql .= "union all ";
		}
		$sql = substr($sql, 0, strlen($sql)-strlen("union all
"))." ";
		$sql .= " order by id desc limit 0, $pagesize";
		#error_log(__FILE__.": sql: $sql , kw:$kw ,
sql:$sql .");
		#print(__FILE__.": sql: $sql , kw:$kw , sql:$sql .");
		$hm = $this->execBy($sql);
		#print_r($hm);

		return $hm;

	}

}
```

对应的基于 –GWA2 的控制器的做法是在 ctrl 目录下生成 ctrl/search.php 的模块，然后在其中应用上述类，同时设定传输参数：

1）搜索关键词输入表单；

2）递交关键词及搜索范围参数（默认是全局）；

3）搜索并返回搜索结果并在前端呈现。

```
……
$search = new Search();

$searchList = array();
$pagesize = 12;

# actions
if($act == 'do'){
	$kw = $_REQUEST['kw'];
	$scope = $_REQUEST['scope'];

	$searchList = $search->getList($kw, $scope);
	if($searchList[0]){
		$data['searchlist'] = $searchList[1];
	}
	else{
		$data['searchlist'] = array();
	}

	if($scope == 'pair'){
		if($kw == ''){ $smttpl = 'fs-search.html'; }
		else{ $smttpl = 'fs-new-search.html'; }
	}
	else if($scope == 'user'){
		# something to do

	}

}

……
```

对应的在 GWA2 的 view 部分，可以战列显示搜索关键词及搜索范围的 HTML Form，如下图所示，该表单显示了搜索关键词为 kw，搜索范围为 product 的递交表单。

```
……
<div class="fsSearch" style="margin-left:45px;">
      <form name="productForm"
action="{$url}&mod=search&act=do&scope=product"
method="get">
            <input type="text" name="kw" value="{$kw}"
placeholder="{$kw}" />
                  <input type="submit" style="opacity:0" />
            <input type="hidden" name="mod"
value="search"/>
            <input type="hidden" name="act" value="do"/>
            <input type="hidden" name="scope"
value="product"/>
            </form>
    </div>

……
```

11.数据表使用建议[1]

在长期的使用 MySQL 过程中，积累了一些东西，往往都是东一榔头，西一棒槌的学来，测试出和总结到的，在没有 ufqi.com 之前，还没个地方梳理。 今天在和 Zoneli.com 的朋友分享代码时，又遇到类似的问题，MySQL table 里的 fileld 类型 和 长度 等细节，于是尝试回忆并将可能的技巧收录于此，随着时间的推移，或者可以形成一个系列。

1) **Don't join** . 尽量不用 jion 或者 jion-like 的多表联查。联查时，数据库引擎需要处理的数项是 2 个表，多个表记录的记录项的乘积。而且这个待扫描的临时结果集是没有直接索引可用的。 测试参考 MySQL 的 join 和 no-join 对比性能测试. 实际案例：http://ufqi.com/exp/x326.html?title=mysql 下慎用 like 与多表连查同时共用 .

2) **不用 NULL 值** 建表的字段，如果没有使用 default 设定默认值的情况下，MySQL 会设置一个 NULL，这个很讨厌，具体案例印象不深刻，可以通过 default 0 对数值，default '' 对字符及字符串. 2011.08.13 补充： 从 MySQL 的文档看其定义，Null 表示的“情况”是，不知道这个字段的“情况”，在“有没有”值的问题上悬疑，不确定。 而 0 或者 ""(空值)，则明确表示，当前字段有值，其值是 0 或者 “”。 这显然是两个语义范畴，“null”是不知道有没有，而其余的是知道了，就是 0 或者 "" .

3) **不用 varchar 类型** 硬盘空间已经不需要用牺牲性能来换取。总是使用 char 这样固定长度的字符类型，使用空间来换取性能。

[1] 原记录于 –ufqi-exp，整理自原文的连续三篇。

4) **字段长度是 4 的倍数** 这个可能测试或者涉及到的内容较深，内存或者计算机的处理的字节，如果是以 4 字节为最小单位的话，那么写成 1/2/3/4 Bytes 计算机都需要进行一次读写操作。

5) **多量少次 地执行操作** 如果有 10 条语句需要执行，那么将 10 条语句合成 1 句来一次递交执行要比 10 次执行快。 具体案例 http://ufqi.com/exp/x857.html?title=少量多次 vs.多量少次的 sql 执行 .

6) **统一到 Unicode: UTF-8 上来** MySQL 本身对字符集及多语言的支持可以细化到四个层面：Server Database Table Connection 也就是说，关于字符集的设置，可以在这四个方面各有不同，可以想见的是，对于非统一的设置会给应用程序和后期维护带来难以逆转的复杂和麻烦。 因此从一开始就全部统一到 UTF-8 上来，为多语言及国际化做好准备。 详细参见： http://ufqi.com/exp/x890.html?title=[konder]MySQL4 的字符编码集配置加载顺序及 UTF-8 环境设置

(接续，to be continued)

7) **MyISAM or InnoDB** 关于数据表的引擎之说，流传的说法是在一些关键应用场景，比如 google 在使用 MySQL 部署服务时，采用的数据库引擎就是 InnoDB, 因为其支持事务，而这通常被认为企业级数据库所必备的。 更多的测试在网上能找到，普遍的测试认为：MyISAM 是 MySQL 默认的引擎，其偏重中 select 查询较多的场合； 而 InnoDB 则具有较多的功能，而且对并发支持较好，同时偏重于频繁 update 操作的应用场景。 如果不确定，所属情况下，MyISAM 都能胜任，这也是 MyISAM 被设置为 默认 引擎的原因之一吧。 (2011.08.07: 新版 5.5/5.6 默认的存储引擎已经变为 InnoDB , 可能 InnoDB 已经成熟到可以担此重任的时候了。) 参考： http://www.mysqlperformanceblog.com/2007/01/08/innodb-vs-myisam-vs-falcon-benchmarks-part-1/http://developer99.blogspot.com/2011/07/mysql-innodb-vs-myisam.html

8) **indexing the searched fields, 在用作查询条件的字段上做索引** 这个很容易理解，但却容易忽视。其区别却是显而易见的，在做了索引的字段上查询，可以一步到位的检索出结果，而没有做索引的字段，数据引擎不得不从头至尾的扫描一遍。 比如，在一个应用中，经常用到的查询如下： select id,email from usertbl where id=? or email=?; 这样的话，就需要在 id 和 email 上分别做相应的索引才会是优化状态。

9) **忽视语法规则的命名法,No grammar abiding** 语法的形成帮助人们更好的表达意思，在程序或者数据库里设计表名或者字段名，甚至数据库名时，存在着是否遵守语法规则的选

择，比如： Messagestbl 还是 messagetbl ? 前者大写首字母，并加了复数形式，可能要表达这个表要存储多条 message 的意思（不过只存一条的表似乎也没存在的现实意义）。再比如： DateCreated 还是 createdate ? 同样前者遵循语法规则，后置了修饰词，并给了过去式的分词做修饰，而后者空口白话的 createdate ，在没有语境的情况下，可能会理解成 创建一个 date，而不是当前记录的 being created 的 date . 但是在简洁一致的要求下，数据库管理中的命名，以忽视语法为简洁，上面的均采用 messagetbl createdate 不会有歧义，前端开发人员在使用的时候也无需关心语法问题。

接续， *[非技术评述, 之后是 hint-10]* 在 MySQL 被原先的 SUN 买了过去后，MySQL 社区兴奋了一阵子，不久， Oracle 这个“黄雀在后”的出现了，买了 SUN ， 于是 MySQL 就陪嫁过去了给了 Oracle . 天下所有悲凉的事，如果排个行的话，对于被竞争对手买过去的事，一定很靠前。 许多业内观察人士认为， 在 Oracle 的麾下， MySQL 的前景就是没有前景。 我也这么认为，甚至一度对自己熟悉的 MySQL 感到惋惜，而且再重新熟悉另外一个产品，需要额外的时间成本。 而且是，开源的可以替代的还没显现，于是世界就悲凉下来了。

直到最近，看到 另外一篇评论 MySQL 说， MySQL 只能在 Oracle 的麾下，棋子般的活着，而不是死去或者脱出 Oracle 的控制之外。如果 MySQL 死去，开源社区不久就会出现个 MySQL2 或者 YourSQL 来。 而如果 Oracle 继续持有 MySQL 并将其按商业利益阉割成鸡肋，那是最符合 Oracle 的规划的-- 社区没想法弄 YourSQL，而现有的 MySQL 也在股掌之中. 虽然很尴尬，这个情况，但可见的未来，MySQL 在中等规模企业层面，还在继续更新，算是略略宽心一些。 ---- ---- ---- ---- MySQL 技巧和注意事项，续，

10) **float vs. int** 应该尽量避免使用 float, 尽量使用 int 来代替 float，原因是 float 的运算比较耗费 CPU . 能够检索到的例子：

And, here is aome test result

```
mysql> select benchmark(100000000,101&lt;101);
+------------------------------+
| benchmark(100000000,101&lt;101) |
+------------------------------+
| 0 |
+------------------------------+
1 row in set (4.25 sec)
mysql> select benchmark(100000000,1.01&lt;1.01);
+--------------------------------+
| benchmark(100000000,1.01&lt;1.01) |
+--------------------------------+
| 0 |
```

+------------------------------+

1 row in set (9.84 sec)

Integer comparison takes less than half the time.

（refer:http://forums.mysql.com/read.php?21,111272,111783#msg-111783）

这一点，程序员应该理解更多感触一些。

11) **公用字段的确定及命名，id/inserttime/updatetime/status** 如果操作的数据表足够多，可能会注意到，很多表都有一些对当前记录本身的描述，比如 id, 当前记录的自增索引号 inserttime, 当前记录的插入/生成时间，createtime updatetime, 当前记录的修改时间 status, 当前记录的状态，有效/失效... 对这些多数表都可能存在的字段，从一开始就设定好，并在后来的开发中加以利用，对整个系统来说，大有裨益.

12) **避免使用 by rand()** 。这个是刚才在网络上看到的，结合之前在工作环境中的一个实现，所以印象很深刻，就写到 No.12 . 用 rand() 取得随机一条或者多条是个令人心动的选择，但 rand()函数却有性能问题， 避免的做法，就是将随机的功能拿到数据库外面做，比如已经知道结果集的规模后，随机一个结果集范围内的数传递给 sql .

13) 数据表单表创建参考

数据库建表注意事项， 以 MySQL 为例

a) 不使用中文字符做 comment;

b) 不使用 varchar， 使用 char;

c) 不允许 null， 每个字段都指定 default 默认值;

d) 每个表至少有一个 primary key 和 unique index key;

e) 公共字段：

id int(11) auto_increment,

state tinyint(1) not null default 1,

updatetime datetime not null default 0,

operator char(32) not null default ''

f) 表名, 前缀 如 lsh_ , 后缀， tbl , 如 lsh_categorytbl，不使用复数形式;

g) 字段名，两个以上单词构成的，不需要下划线连接，除非有明显的歧义;

h) 表名和字段名中相邻的重复字符不省略，如 inserttime , 不能写成 insertime;

i) 使用 -gMIS 作为管理后台的话，需要有些基础设置表;

j) 不使用 name, value, key 等这些容易与保留关键词冲突，或者容易 -GTAjax 中 eval 出错的词语作为 字段名或者表名; 推荐的命名方法，可以使用 “i+实际名称”, 如 iname, ivalue, iage, ikey, idesc, isomethingelse 等

Sun Apr 19 09:14:53 CST 2015

k) 字段的长度, 以一个字节 8 为基本单位， 如 char(50) 应改为 char(48)， char 以外的多数靠名称决定长度;

INT 是靠名称决定, 如下，但有时写成 int(11) 意思是 在客户端显示的时候用 11 位（字节），也即 01234567890

TINYINT = 1 byte (8 bit)

SMALLINT = 2 bytes (16 bit)

MEDIUMINT = 3 bytes (24 bit)

INT = 4 bytes (32 bit)

BIGINT = 8 bytes (64 bit).

B. 架构设计说明/Design Sections

12.Overview

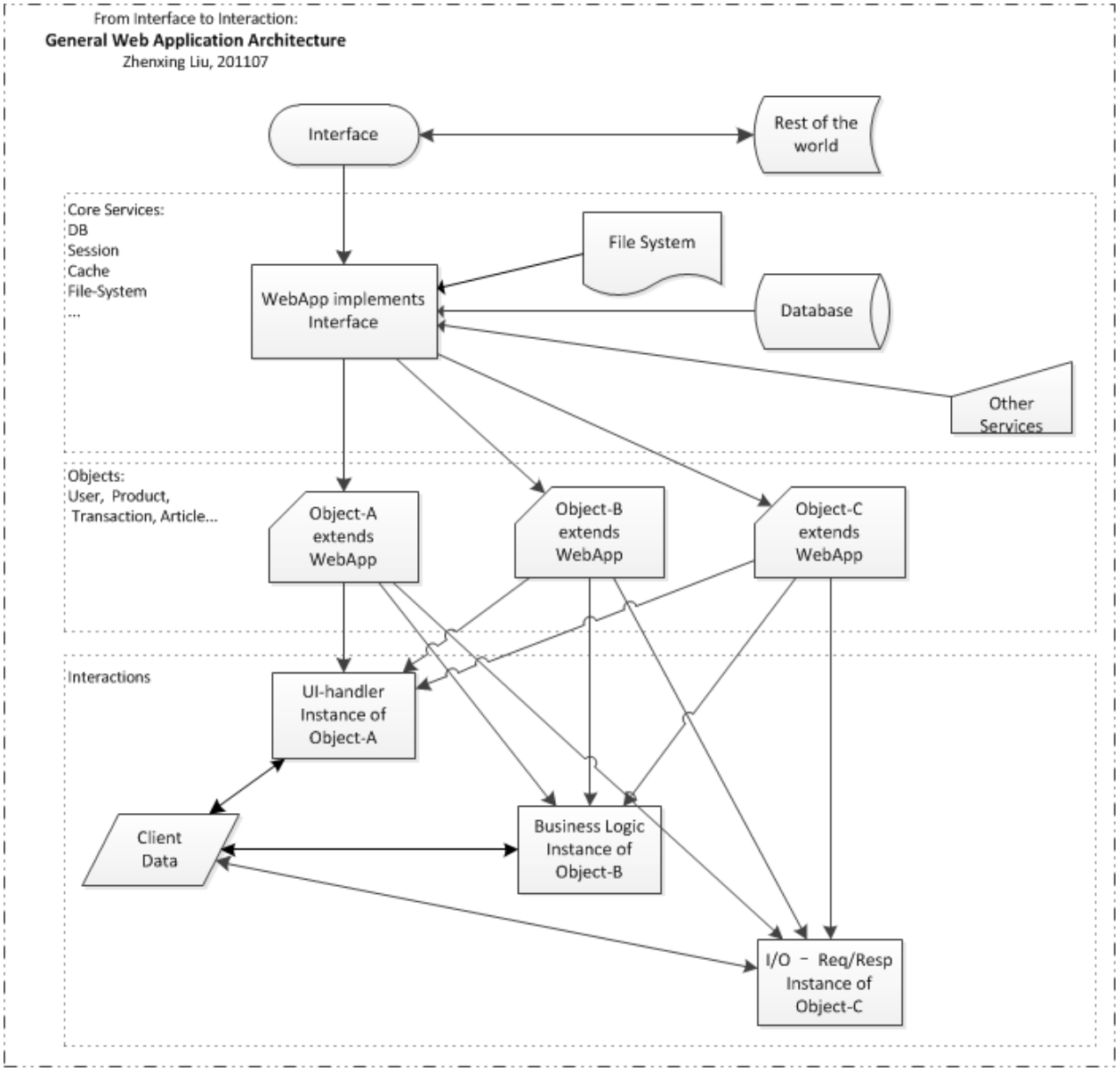


Figure 16 Main components in GWA2

From a point of factory view, we see these three parts/tiers as listed in the following diagram.

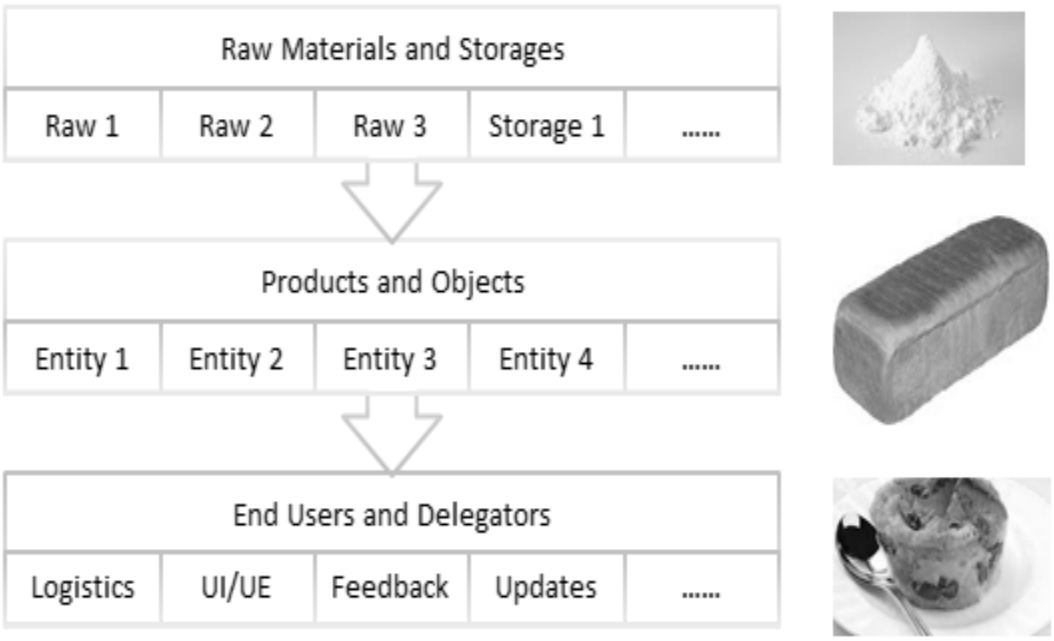


Figure 17 Factory view in GW2

Core service can be regarded as the raw materials and/or storages, products are objects and interactions are some components in end-user and delegator section.

GWA2 looks similar with abstract factory pattern (**-R/12V8**). There are the interface webapp.interface, the parent and/or abstract class webapp.class which implements the interface, and the extended classes in mod subdirectory.

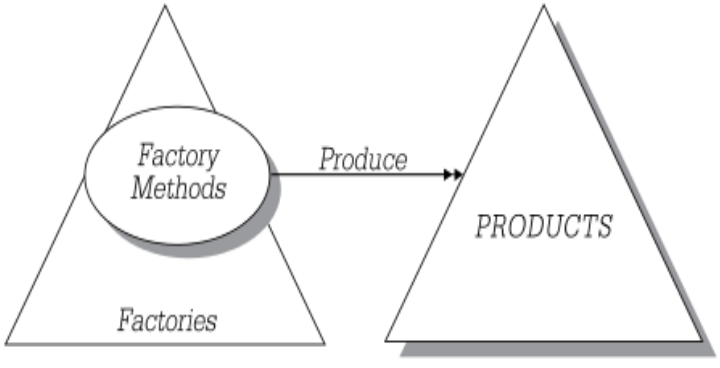


Figure 18 Abstract Factory in GWA2

The abstract factory pattern provides a way to encapsulate a group of individual factories that have a common theme without specifying their concrete classes[2]. Abstract Factory patterns work around a super-factory which creates other factories. This factory is also called as factory of

[2] http://shop.oreilly.com/product/9780596007126.do

factories. This type of design pattern comes under creational pattern as this pattern provides one of the best ways to create an object[3].

In Abstract Factory pattern an interface is responsible for creating a factory of related objects without explicitly specifying their classes. Each generated factory can give the objects as per the Factory pattern.

Here is an example (**-R/92TS**) for this design pattern to demonstrate what GWA2 has done to organize the codes and those components.

1) The interface for PersonInterface

```
package com.ufqi.webapp;

import java.util.Calendar;

public interface PersonInterface {

	public static final String ROLE =
PersonInterface.class.getName();

	public abstract String getId() throws Exception;

	public abstract void setId(String id) throws
Exception;

	public abstract String getName() throws Exception;

	public abstract void setName(String name) throws
Exception;

	public abstract Calendar getBirthday() throws
Exception;

	public abstract void setBirthday(Calendar birthday)
throws Exception;

}
```

2) The parent and/or abstract class PersonInterfaceImpl

[3] http://www.tutorialspoint.com/design_pattern/abstract_factory_pattern.htm

```java
package com.ufqi.webapp;

import java.util.Calendar;

import com.ufqi.webapp;.PersonInterface;

public abstract class PersonInterfaceImpl implements
PersonInterface {

	private String id = null;
	private String name = null;
	private Calendar birthday = null;

	public String getId() {
		return id;
	}

	public void setId(String id) {
		this.id = id;
	}

	public String getName() {
		return name;
	}

	public void setName(String name) {
		this.name = name;
	}

	public Calendar getBirthday() {
		return birthday;
	}

	public void setBirthday(Calendar birthday) {
		this.birthday = birthday;
	}

}
```

3) The concrete class Adult

```java
package tw.idv.javax.demo.Extends;

import com.ufqi.webapp.PersonInterfaceImpl;

public class Adult extends PersonInterfaceImpl {

	private int income = 0;

	public int getIncome() {
		return income;
	}

	public void setIncome(int income) {
		this.income = income;
	}

}
```

We would follow this road to illustrate all of GWA2.

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

```
/* WebApp Interface definition for all of the
implement classes
 * v0.1,
 * wadelau@ufqi.com, 2011-07-10 15:27
 * remedy by wadelau@ufqi.com, 10:12 01 May 2016
 */

interface WebAppInterface{

	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);

	.....
```

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 or entity, 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, $conditions);
	function execBy($fields, $conditions);
	function rmBy($conditions);

	function toString($object);

	.....
```

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 has a table name and an id field as default.

.set/get methods save running time values temporarily in a pre-defined container (e.g. an array/hash table/map). .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.

These actions cover most of transactions between the WebApp and the rest of world, i.e. something input and something output.

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

```
class WebApp implements WebAppInterface{

	var $dba = null;
	var $hm = array();
	var $hmf = array();
	var $isdbg = 1;  # Gconf::get('is_debug');
	var $sep = "|";

	//- constructor
	function __construct(){

	if($this->dba == null){
		$this->dba = new DBA();
	  }
		$this->isdbg = Gconf::get('is_debug');
	}

	//-
	function set($field,$value=null){

		if($value === null){
			if(is_array($field)){
				foreach($field as $k=>$v){
					$this->hmf[$k] = $v;
				}
			}
			else{
				$this->hmf[$field] = '';
			}
		}
		else{
			$this->hmf[$field] = $value;
		}
	}

	//-
	function get($field){

		if(array_key_exists($field,$this->hmf)){
			return $this->hmf[$field];
		}
		else if($field != 'id' & $field != 'tbl' &&
$field != 'er'){
			if($this->get('er') != 1){
	……
```

In order to make the interface 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, which is de facto backend for most of WebApp.

$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 execution cycle.

.setBy/getBy/rmBy collect values in $hmf and construct new sql sentence and/or storage commands then send the sql/command to $dba or other storage engines, after that waiting for the result and forward returning 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.

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

```
class User extends WebApp{

	var $sep = "|";
	var $eml = "email";

	//-
	function User(){
		//-
		$this->dba = new DBA();

	$this->setTbl(Gconf::get('tblpre').'info_siteuser
tbl');
	}

	//-
	function setEmail($email){
		$this->set($this->eml,$email);
	}

	//-
	function getEmail(){
		return $this->get($this->eml);
	}

	//-
	function isLogin(){
		return $this->getId() != '';
	}

	……
```

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 other common class, i.e. the parent class or the interface itself and some other objects.

These specific properties of the object User here include e-mail, 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.

16.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'){
		# actions
	}
	......

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

It is show time!

/inc/header.inc is appointed to do some basic work for creating a dynamic page to user (request). This included script 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 to the $out and work with the object $user, or most of all, other objects, e.g. $article, $product, $transaction and so on.

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

Moreover, there is a parameter “a” (act) 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[4]: Practice of Java in JSP.

17.Return values

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).

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

For instance, in statically typed languages (i.e. Java[5]),

```
……
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]
}

……
```

[5] GWA2 is designed for general programming languages, not specifically for any one of them, but for all of them. Many examples in this manual book are written in PHP, however, they will run as normal when being written in other languages, e.g. Java, Perl, C++ and so on.

Compared this with PHP built-in function, mysql_query[6],

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

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.

18.Output, response, view

From what has been described above, the output of a request has not been explained clearly. In GWA2 there always is a global variable to hold all the data to be displayed to users, i.e., $out. With MVC mode enabled, $tpl and $data are defined.

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

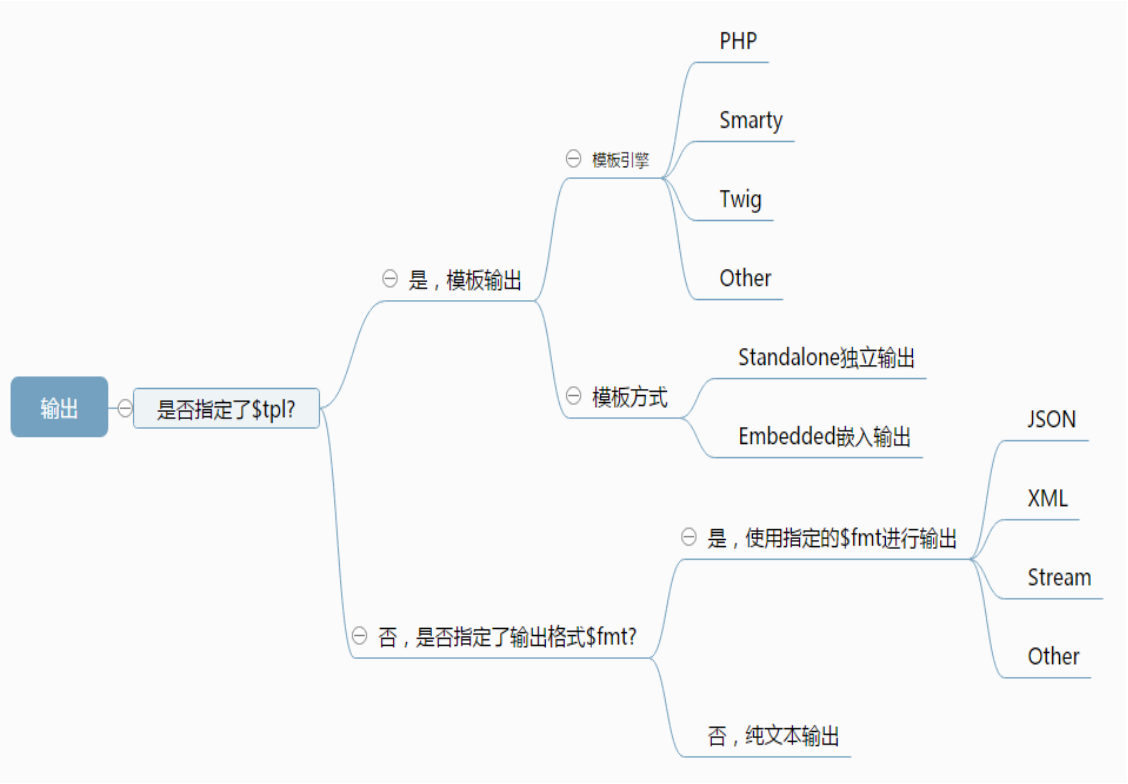


Figure 19 Output in GWA2

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 embedded.

  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 but 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.

19.MVC, Module-View-Controller

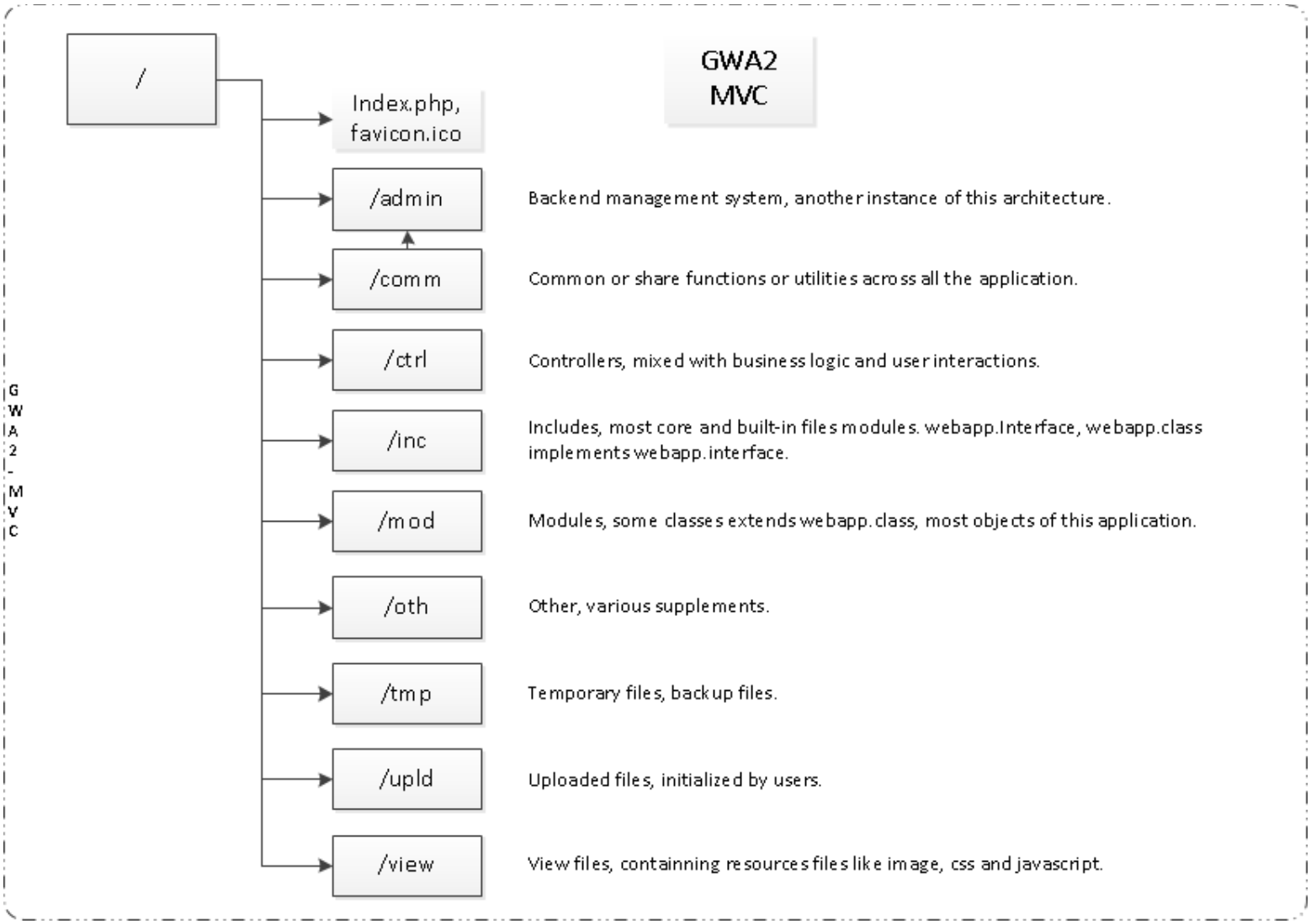


Figure 20 directories of an application

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[7]. 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…

  In a RESTful URL style enabled environment, the entry URL looks like as below.

  index.php/para1/value1/para2/value2
  i/mod/product/act/preview/id/2 # "I" is a soft link to index.php in server side

  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 serves for this purpose.
- /comm

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

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

  HTML, image, css, JavaScript, media…

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

A standard flowchart of any given request:

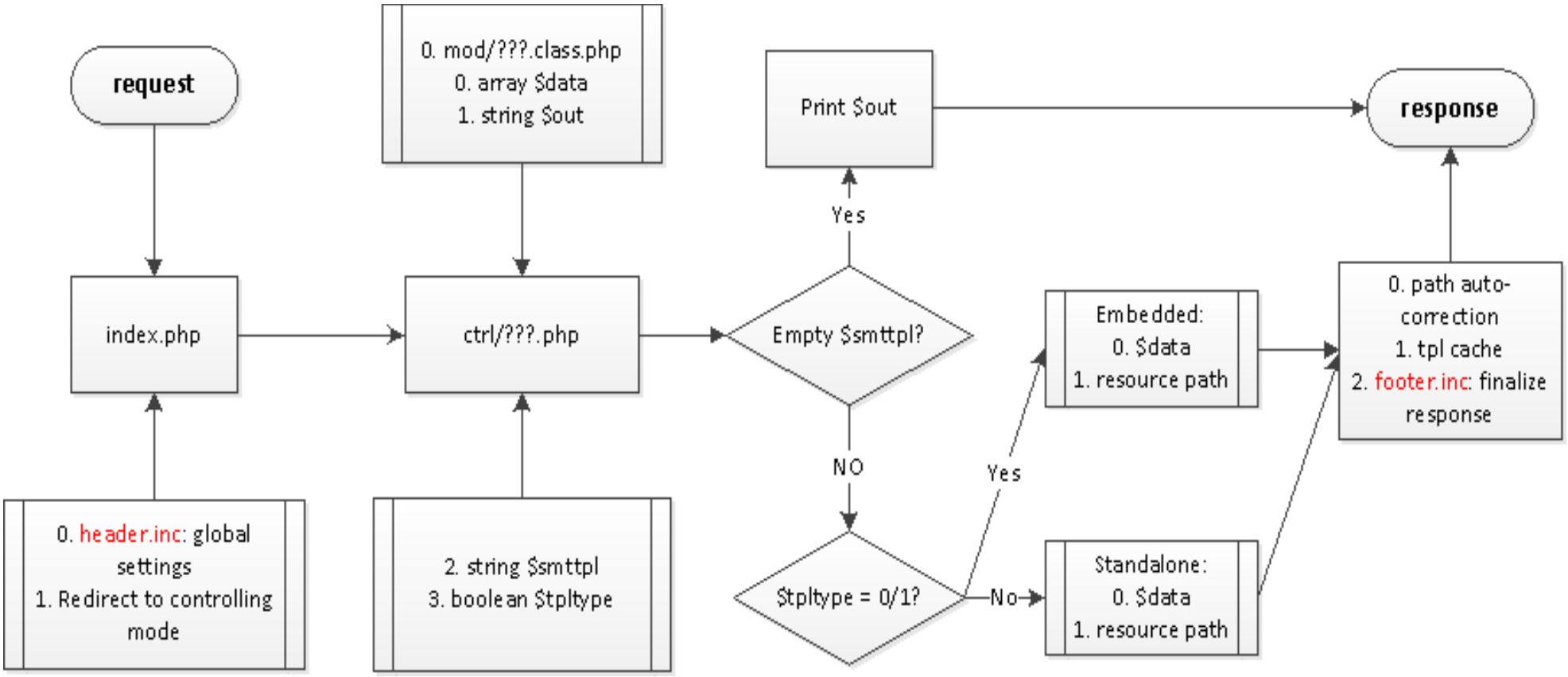


Figure 21 Flowchart of a request-response cycle

20.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.

20.1. 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. There may be a list of search keys used in a single request.
6) oppnsk, operator of page navigator search key, the operators include:
    - for number
        - =, equal to
        - !=, not equal to
        - >, greater than
        - >=,
        - <, less than
        - <=,
        - inlist, 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
        - =, equal to
        - !=, not equal to
        - 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,

Corresponding to search keys, there may be a list of oppnsk used in a single request.

7) pnsm, page navigator search mode, ‘or|and’, used when many keys are involved
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.
   This is a must-be field following the field pnsc.

20.2. 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. i.e., given a searching criteria, the total count matching the conditions are only needed to be sum once.
3) pnsk and oppnsk are used in a parallel mode. E.g.,
   index.php?mod=user&act=list&pnskname=james&oppnsk=contains…[8]
   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 contain 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’.

[8] In RESTful URL enabled mode, the URL may vary in some parameters.

4) pnsc is an extension to the page navigator. 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 the 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. This is to verify the authentication of SQL and prevent the server from SQL injection.

21.Other issues

21.1. 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.

```
……
//-- multiple tables operation

$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
}

……
```

21.2. 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 and UTF-16 can be found at Wikipedia[9].

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

[9] http://en.wikipedia.org/wiki/UTF-8

    - Field-level
- Services settings, default charset
- Front-end, page code, charset
    - Html page code, charset
    - HTTP charset

22. To-do

22.1. In LAMP, MySQLi[10] or PHP Data Objects (PDO)[11]?
(Partially done, Oct, 20016)

22.2. Adding rollback() in DBA?

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

22.4. Cache/Queue/Session/File service need to be improved.
(Partially done, Oct, 2016)

23. Design 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	

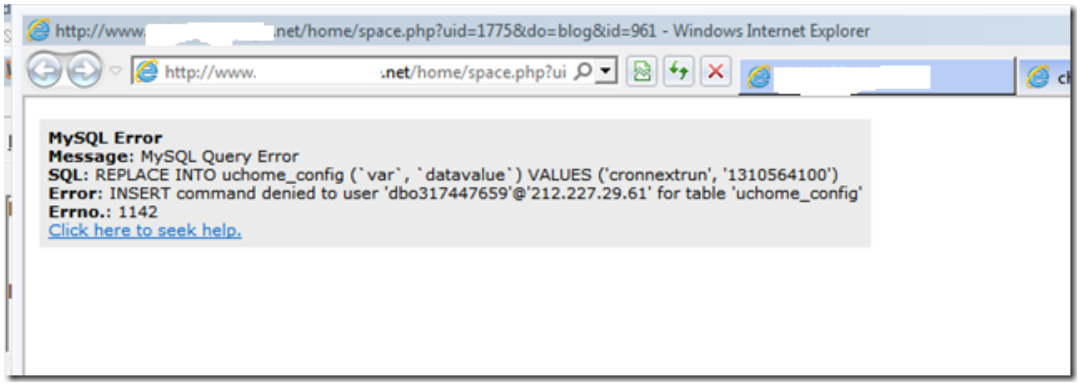
[10] http://cn2.php.net/manual/en/mysqli.query.php
[11] http://cn2.php.net/manual/en/pdo.query.php

No.	Version	Updates	Date	Author(s)	Inspector(s)
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	
9	V0.5	Extended into three sections: design, developing and update.	2016-04	Zhenxing Liu	
10	V0.6	Refined	2016-10	Zhenxing Liu	

C. 升级与维护/Updates and Maintenances

24. 关于 PHP , MySQL Error Handler 错误处理机制的新探索

（2011-07-22）



上图某应用的 PHP 针对数据库的报错信息。

Discuz! info: MySQL Query Error

Time: 2011-5-6 11:43am
Script: /index.php

SQL: SELECT s.sid, s.styleid, s.groupid='6' AS ipbanned, s.pageviews AS spageviews, s.lastolupdate, s.seccode, m.uid AS discuz_uid, m.username AS discuz_user, m.password AS discuz_pw, m.secques AS discuz_secques, m.adminid, m.groupid, m.groupexpiry, m.extgroupids, m.email, m.timeoffset, m.tpp, m.ppp, m.posts, m.threads, m.digestposts, m.oltime, m.pageviews, m.credits, m.extcredits1, m.extcredits2, m.extcredits3, m.extcredits4, m.extcredits5, m.extcredits6, m.extcredits7, m.extcredits8, m.timeformat, m.dateformat, m.pmsound, m.sigstatus, m.invisible, m.lastvisit, m.lastactivity, m.lastpost, m.prompt, m.accessmasks, m.editormode, m.customshow, m.customaddfeed, m.newbietaskid FROM [Table]sessions s, [Table]members m WHERE m.uid=s.uid AND s.sid='Tlt66l' AND CONCAT_WS('.',s.ip1,s.ip2,s.ip3,s.ip4)='1[redacted]' AND m.uid='1' AND m.password='[redacted]' AND m.secques=''
Error: Can't find file: './[redacted]/[Table]sessions.frm' (errno: 13)
Errno.: 1017

到 http://faq.comsenz.com 搜索此错误的解决方案

上图某应用的 PHP 针对 SQL 查询的报错。

当 PHP 在处理 MySQL 的相关操作时，遇到错误，多数开发人员会选择如上的做法，将信息完整的呈现给调用者（终端用户）。 实际上，这样是不安全的和用户 UI 不友好的。
本文就解决上面的问题，提出替代现行的 die 做法，而引入 error_log 来实现更加安全和友好的错误处理机制。

最近在写一个全新的项目，在梳理之前的代码时，有新的思考和实现，其中关于 PHP，MySQL 相结合的部分，在对待错误处理方面，有不少可以改进的地方，比如报错信息的打印，程序流程的导向，甚至函数返回值的定义与解读等。
在接下来会就 PHP 的错误处理机制进行一次新的探索，在另外一篇 blog 中，会就 PHP 的函数/方法返回值的思考总结。

促使写关于 PHP 错误处理机制的文章，主要 2 件事的诱因。其一，之前的代码中能，关于报错信息的处理，居然用了 2 个不同的方法，也即一个方法报错，显示给调用者，另外一个方法是不报错，出错了，就一片空白的给调用者。
另外一个是最近遇到的页面报错，PHP 所写，也与 MySQL 相关，如文章开头的部分的截图所示。
说这样不安全，是因为将代码信息和数据库信息透露给调用者（探测者），这为攻击者洞开了大门。

说这样不友好，是因为，对终端用户来说，这些信息没有任何意义，用户只会不知所措。

因此，对这样的做法应该可以有纠正和改进的空间。

基于上面的认识，加上对之前代码的修正的需求，驱动我考虑是否可以将 Java 的做法引入到 PHP 里？ 出错了，将错误信息打印到后台，而前端可以继续显示一些自定义的信息，友善的提醒用户等等。

这样做的好处，明显地就解决了上面提到的将错误信息输出到后台，便于开发人员分析，同时避免程序和数据信息外泄，而且显示自定义的信息给用户，也更加 UI 的友好。

新的探索内容如下：

1.

之前的做法：

```
$result = mysql_query($sql) or die("error_no:[".mysql_errno."] error_info:[".mysql_error."] sql:[".$sql."]");
```

上面 die 里面的消息体，可以自由定义，程序在 mysql_query 返回 false 时，执行 die 操作，就是退出当前程序，并显示 die 里面的消息体参数 。

这样的写法和操作，就如同上面文章开头的截图一样。类似的，我们之前遇到的，需要根据调用者的不同（不同身份，不同来源等），决定是否显示错误消息到前端，于是就做了不同的区别对待的 方法体，而这样做仅仅是期望是否打印消息 —— 这显然需要改进。

2.

改进后的做法：

```
......
$result = mysql_query($sql) or $this->handleError($sql);

......

function handleError($sql){

	if($isdebug || $isadmin){
		//- print msg

	}
	else{
		//- log in background
		error_log("ERROR_LOC: ERROR_MSG");
		//- notify admin
		SEND_EMAIL_OR_SMS;
	}

	return false;

}
......
```

上文程序中，如果是测试模式，或者管理员身份调用，直接打印如上文开头的截图部分，其余的，都使用 error_log 将错误记入日志，便于进一步的分析. 同时，启动通知机制，触发警报。

在函数体后面，

```
return false;
```

进一步的通知后面的程序，当前执行遇到故障，后面的程序体，可以根据当前的返回值，做进一步的处理，如：

```
//- after mysql_query
```

```
……
if(!$query_result){

   print "Sorry, something wrong...";
   //- exit now or not?

}

……
```

如此完美解决了 记录日志，触发通知， 和 友好显示的目标。

25. GWA2 核心部件升级

（2013 年 9 月 2 日）核心部分，

1) inc/webapp.class.php 中 setBy 方法

if($this->getId() == ”)

判断修改为

if($this->getId() == ” && ($conditions == null ||$conditions == ”))

增加

$v = trim($v);

2) class|mod/mailer.class.php 中

FALSE –> false;

ereg –> preg_match;

ereg_replace –> preg_replace;

非 bug 修正，之前测试已经正常的功能不受影响。

26. GWA2 更新：SQL 语句写入数字型字符串

（2014 年 1 月 7 日）在 GWA2 的核心处理 DB 连接及查询处理时，会对用户的输入做安全性过滤，如在 inc/dba.class.php 中对用户输入进行封装，并将结果交给指定的 inc/mysql.class.php 去处理。

在 inc/mysql.class.php 中，会将用户的输入去除转义字符，在非数字的字符串上加上引号。其判断逻辑是：

```
……
            if ($this->ismagicquote){
                  $value = stripslashes($value);
            }

            if (!is_numeric($value)) {

                  if($this->mode == 'mysqli'){
                        $value =
"'".$this->m_link->real_escape_string($value)."'";
                  }
                  else{
                        $value =
"'".mysql_real_escape_string($value, $this->m_link)."'";
                  }
            # in some case, e.g. $value = '010003', which
is expected to be a string, but is_numeric return true.
            # this should be handled by $webapp->execBy
with manual sql components...
            }
            else{

                  if($defaultValue == ''){
                        if($this->mode == 'mysqli'){
                              $value =
"'".$this->m_link->real_escape_string($value)."'";
                        }
                        else{
                              $value =
"'".mysql_real_escape_string($value, $this->m_link)."'";
                        }
                  }
            }

……
```

通常这样做是没有问题的，但如果想在数据表中存入数字型字符串，比如在一个应用中的目录代码是 01010003 , 在使用 is_numeric 时，返回的就会是 true:

is_numeric(’1010003’); # true

is_numeric(’010100003’); # true

遇到这种情况，就不能使用 GWA2 的安全套件，如$obj->setBy, $obj->getBy 等，但可以使用 $obj->execBy, 这个方法允许用户自行组装 SQL 语句，从而避开原有的检查程序的缺陷。

关于这个问题，在 -gMIS 中找到一种实现方法， -R/z2TO ，依赖对数据表配置信息的获取来判断字段类型。

反复思考，在没能获得数据表字段配置信息的情况下，还是这个法子好。不然，无论是采取多增加配置文件，或者读取表属性的方法来判断某个字段是否有 number as string 的需求，都有点一个人（字段）有病，所有人（字段）吃药的感觉。

27.一例网络应用开发的 bug 分析

(2014 年 7 月 3 日) 作为 -GWA2 的一个实例， -gMIS 一直运行良好，今天在部署到一个新项目测试运行中，发现一个令人不安的 bug，虽很快找到并修复，但就像 OpenSSL 的 heartbleed 一样，需要事后反思。

事情缘起这样，新项目部署后，其中一个数据表的记录被不断的更新，数据被清空。由于是部署在一个“伪私有主机”上，无法登陆到服务器控制台进行查询，所以加大了判断和 troubleshooting 的难度，好在对于 -gMIS 相当熟稔，同时对业务逻辑也很明细，再同时 -gMIS 里能够看到详细的 operatelog , 这样定位起来，即便在“虚拟主机”，看不到访问日志的情况下，基本推测出出错的流程。

错误点在于，由于未知的原因，远程访问的请求拼出类似下面这样的地址：

admin/jdo.php?act=list-addform&id=…

常规情况下，这一请求会被

admin/comm/header.inc

里的判断逻辑所阻隔，并跳转到登录页面，所使用的语句是

header(“Location: LOGIN_PAGE”);

问题恰出在此处，虽然这里设置了 header , 也即在 HTTP 的 response 头部增加了 302 的跳转导向，但！是！ header 之后的语句逻辑会照样被执行，也即符合相关条件的记录仍会被更新，由于这个请求是被恶意拼接的，所以相应的数据根本就是错误的，也因此出现了开头的一幕，数据被不断的重置为空，但记录数并没有变，只是项的值被不断的填入为“”。

经上分析测试，在测服务器上快速部署了代码

exit(0);

在发现未登录用户的情况下，随即结束当前页面的业务逻辑，终止进一步操作并将页面处理权由 PHP 交回到 Apache 那里去。

虽然是个小 bug，并且发现及时，但仍事后感到心有余悸。为何之前没有预料到这样的错误呢？如果没有

exit(0);

执行语句，程序页面在测试时并无任何异常，逻辑仍是按照预期的路线跳转到登录页，只是这种依赖终端浏览器的测试，不能察觉出，实际的 mod 和 act 仍旧被执行了。这里只测试了浮表，并未能使用 code review 的方式进行深入分析每一块逻辑和每一行语句。

软件仍是在不断使用和测试中成长的。

28.-gwa2 安全更新

（2014 年 10 月 30 日）

```
……
$data['mod'] = $mod;
$data['act'] = $act;
$data['baseurl'] = $baseurl;

if(file_exists("./ctrl/".$mod.".php")){
	include("./ctrl/".$mod.".php");
}
else{
	print "ERROR.";
	error_log(__FILE__.": found error
mod:[$mod]");
	exit(0);
}

……
```

更新主要出于安全考虑，在使用命令 include 相关模块之前，先探测相关模块文件是否存在，以此来规避可能存在的 Script 攻击行为。

当发现对不存在的 mod 进行调用时，直接终止并退出对当前 request 的处理。

29.一种 debug 方法的实现

（2015 年 10 月 19 日）据说程序的开发成本占 20%，维护成本占 80%，而维护的主要方式之一依靠的是 log 输出。

输出 log 有很多方式，各种开发语言都有不同的设施来满足这一功能。

在 PHP 中，可以这么实现一个 debug 功能。其主要功能是输出什么时间在什么位置（程序名、函数名）输出了标记为 xxx 的某个对象$obj 的打印形式的描述信息。

```
function debug($obj, $tag=null, $output=null);
```

$obj 是待输出和跟踪的对象，可能是 string，也可能是 array、hash 等不同的数据结构；

$tag 是用于标记待输出对象的字符串，一般用于过滤相关输出内容；默认 $tag 为 null;

$output 类似 loglevel, 但又不同，通常 $output==null, 表示在后天输出相关日志，与 error_log 函数类似； $output==1 时，在后台和前端同事输出相关测试信息；$output==2 时，根据 PHP 中的 backtrace 信息，输出更多内容。

调用例子:

debug($aHash); # 在后台输出一个 hash 变量的内容；使用 -GWA2 的 WebApp->toString 方法或者 -PHP 内置的 serialize ;

debug($aHash, ‘aHash’); # 在输出中增加识别标记字符 ‘aHash’;

debug($aHash, ‘aHash’, 2); # 同时在前端和后台输出相关内容，然后附带 -PHP 的 backtrace 信息.

源码 Source，位于 comm/tools.function 下面，可供全局调用。

```
# write log in a simple approach
# by wadelau@ufqi.com, Sat Oct 17 17:38:26 CST 2015
# e.g.
# debug($user);
# debug($user, 'userinfo');  # with tag 'userinfo'
# debug($user, 'userinfo', 1); # with tag 'userinfo' and in
backend and frontend
function debug($obj, $tag='', $output=null){
	$caller = debug_backtrace();
	if(is_array($obj) || is_object($obj)){
		if(isset($user)){
			$s .= $user->toString($obj);
		}
		else{
			$s .= serialize($obj);
		}
	}
	else{
		$s .= $obj;
	}

	if($tag != ''){
		$s = " $tag:[$s]";
	}

	$callidx = count($caller) - 2;
	$s .= ' func:['.$caller[$callidx]['function'].']
file:['.$caller[$callidx]['file'].']';

	if($output != null){
		if($output == 0){ # in backend only
			error_log($s);
		}
		else if($output == 1){ # in backend and frontend
			error_log($s);
			print $s;
		}
		else if($output == 2){ # in backend and frontend
with backtrace
			$s .= " backtrace:[".serialize($caller)."]";
			error_log($s);
			print $s;
		}
	}
	else{
		error_log($s); # default mode
	}

}
```

30.GWA2 更新：SQL 语句写入数字型字符串(2)

（2016 年 3 月 6 日）关于在 MySQL 的使用中，插入数字型字符串时，在贴文:

“GWA2 更新：SQL 语句写入数字型字符串”（ -R/gwa2-update ）

做过分析，遇到这样的情况，需要特殊对待，避开 $obj->setBy , 而是使用 $obj->execBy 来自行拼凑 SQL 语句。

然而在 -gMIS 这样的使用情景中，却不能自行拆解 $obj->setBy , 所以需要进一步的分析寻找解决方法，思路如下：

1) -gMIS 环境中，有 tblconf.inc 通过他能够读取到字段的默认值，这也是在

“-gMIS 更新兼容 Strict SQL Mode”（ -R/e2TX ）

讨论过，设想如果知道一个字段的默认值，可以据此判断它的值的表现形式；这是下面的基础；

2) 在 webapp.class 中增加 $hmfieldinfo 用于保存当前操作表的结构信息，主要包括字段默认值等信息；从这里也可以读取到字段类型，比如 int，char 等；后续可以继续利用这些属性；
3) 进一步地，在 gtbl.class extends webapp.class 中，在设置 setFieldList 时，填充 $hmfieldinfo ;
4) 在 dba.class 中，进一步的透传 $hmfieldinfo 到 mysql.class 中，然后在 query 执行过程中，根据 $hmfieldinfo 中的相关属性进行判断：当 field 是 char 或者默认值是 ” 时，即便是 0102 这样的值，也当作字符串进行处理。

如此即可。

31.-GWA2 core updates: 容错处理,孤儿占位符

（2016 年 4 月 3 日）-GWA2 的核心升级了一处容错处理，问题描述为：

如果某个 WebApp 的实例$obj, 在进行 SQL 组装时，如果$obj->set 的变量数据与 SQL 中的占位符(?)不一致时，特别是占位符(?)所指代的变量名未出现时，在 SQL 拼装时，容易产生将 idxArr[0]的值赋予给这个缺失的变量值。这样就导致占位符序列和变量序列错位现象，从而可能到 SQL 在写入时的错误。这种错误可称之为“孤儿占位符”。

问题的原因是，调用者在使用$obj->set 时，在 SQL 中写入了某个占位符，但并未在$obj-hmf 中给定相应的值。一个错误的调用如：

```
$obj->set('b', $aVal);

$obj->set('c', $bVal);
```

$result = $obj->setBy('a, b, c', null);

此时的 a，没有通过$obj->set 赋值，会在后续拼装 SQL 时出错。

解决的办法是，在拼装匹配占位符和变量时，做变量名和变量值的检查。涉及到的代码在 inc/mysql.class , 如下:

```
……
      if(strpos($t, $idxarr[$i]) === false){
            # in case that, field was not set by $obj->set but
written in sql with '?', Sat Apr  2 23:54:48 CST 2016
            debug(__FILE__.": found unmatched field:[$t].");
            $sql = substr($sql,$a+1);
            $a = strpos($sql,"?");
            $newsql .= str_replace("?", '\'\'', $t);
      }
      else{
            $sql = substr($sql,$a+1);
            $a = strpos($sql,"?");
            $newsql .=
str_replace("?",$this->_QuoteSafe($hmvars[$idxarr[$i]]),$t);
            $i++;
      }

……
```

同时也将修改同步到基于 -GWA2 的 -gMIS 中。

32. MySQLi 支持及 xForm

(2016 年 05 月 02 日)

32.1. MySQLi 及多数据库的支持

在较新的 -PHP 环境中，运行 mysql_connect 时，会报错：

Deprecated: mysql_connect(): The mysql extension is deprecated and will be removed in the future: use mysqli or PDO instead in xxxx file.

MySQLi 名称是取自 MySQL Improved 的缩写，它是 MySQL 针对 PHP 所设计的一个扩充模组。虽然 PHP 原本就有能够存取 MySQL 的函式库，但是在 MySQL 4.1.3 版之后，PHP 官方强烈推荐使用 MySQLi。

mysqli 扩展，有时称之为 MySQL 增强扩展，可以用于使用 MySQL4.1.3 或更新版本中新的高级特性。mysqli 扩展在 PHP 5 及以后版本中包含。mysqli 扩展有一系列的优势，相对于 mysql 扩展的提升主要有：

- 面向对象接口
- prepared 语句支持（关于 prepare 请参阅 mysql 相关文档）
- 多语句执行支持
- 事务支持
- 增强的调试能力
- 嵌入式服务支持

在提供了面向对象接口的同时也提供了一个面向过程的接口。mysqli 扩展是使用 PHP 扩展框架构建的，它的源代码在 PHP 源码目录下的 ext/mysqli 中。

与 PDO （PHP 数据对象）不同，后者提供了一个 数据访问 抽象层，这意味着，不管使用哪种数据库，都可以用相同的函数（方法）来查询和获取数据。 PDO 不提供 数据库 抽象层；它不会重写 SQL，也不会模拟缺失的特性。如果需要的话，应该使用一个成熟的抽象层。

由于在 -GWA2 的设计中，已经将存储驱动单独隔离开来，如数据库的驱动，经由 inc/dba.class 来实现，MySQL 的连接，默认使用 inc/mysql.class 来实现，因此改进架构对 MySQLi 的支持，只要修改数据库连接驱动，inc/mysql.class 文件中相应的文件即可。

同时，考虑前向兼容没有 MySQLi 的运行环境，需要在过度阶段对 MySQL 的运行模式进行判断，也即优先判断当前环境是否支持 MySQLi — 如果支持，则启用功能更强大的 MySQLi 增强扩展, 如果不支持，则仍回归使用旧式的 MySQL 扩展。

```
……
        //-
        function _initconnection(){
              if ($this->m_link==0){
                    $real_host =
$this->m_host.":".$this->m_port;
                    $this->mode =
function_exists('mysqli_connect') ? 'mysqli' : 'mysql';
                    if($this->mode == 'mysqli'){
                          $this->m_link = new
mysqli($this->m_host, $this->m_user, $this->m_password,
$this->m_name, $this->m_port);
                    }
                    else{
                          $this->m_link =
mysql_connect($real_host,$this->m_user,$this->m_password) or
die($this->Err("mysql connect"));
                    }

                    if ($this->mode == 'mysql' && "" !=
$this->m_name){
                          mysql_select_db($this->m_name,
$this->m_link) or die($this->Err("use ".$this->m_name));
                    }

        if(get_magic_quotes_gpc()){ $this->ismagicquote = 1; }
              }
        }

……
```

在发起连接之后，其他相关的函数调用或者对象及方法的使用，也要根据此处的 $this->mode 做相应的调整。

32.2. 一种快捷的 HTML Form 生成方法

HTML Form 在 WebApp 中应用非常广泛，是进行数据交互的基本形式之一，构造一个 HTML Form 有很多种方法，如下是在 PHP 中，快速生成一个 Form 表单的方法之一。

```
# create a form output
function xForm($nextact, $fields, $isfile=0){
	$rd = rand(100, 99999);
	$form_header = "\n<br/><div id='formdiv".$rd."'><form
name=\"myform$rd\" id=\"myform$rd\" action=\"".$nextact."\"
method=\"post\" ";
	if($isfile == 1){
		$form_header .= " enctype=\"multipart/form-data\"";

	}
	else{
		$form_header .= " enctype=\"application/x-www-form-
urlencoded\"";
	}
	$form_header .= ">";
	$form_footer = "<br/><br/><input type='submit'
name='mysubmit' value='确定&下一步' />";
	$form_footer .= "  <input name='myback'
type='button' value='取消&返回'
onclick='javascript:window.history.back(-1);' />";
	$form_footer .= "</form></div>";

	$form_field = '';
	$xform = '';
	if(is_array($fields)){
		foreach($fields as $k=>$v){
			$dispname = '';
			$form_field = '<input name="'.$k.'"
id="'.$k.'"';
			if(is_array($v)){
				foreach($v as $k1=>$v1){
					if($k1 == 'dispname'){ $dispname =
$v1; }
					else{ $form_field .= "
$k1='$v1'"; }
				}
			}
			else{
				print __FILE__.": illegal setting.
1604161129.";
			}
			$form_field = "\n<br/><br/>".$dispname.":
".$form_field." />";
			$xform .= $form_field;
		}
	}
	else if($fields == null){
		# no extra field.
	}
	else{
		print __FILE__.": illegal setting. 1604161130.";

	}

	#return $form_header.$form_field.$form_footer;
	return $form_header.$xform.$form_footer;

}
```

使用范例：

$out .= xForm($file.'&step=dolicense', array('hasagree'=>array('type'=>'checkbox', 'dispname'=>'我已阅读使用协议并同意')));

将会生成一个完整的，还有一个填写字段的 HTML Form 表单。

33. -GWA2 新增命令行运行及 NO_SQL_CHECK 支持

使用中的框架才会被进化。

-GWA2 在实际部署和应用中遇到新需求，于是就增加新功能，如下两个：1）新增命令行运行模式支持；2）新增对 NO_SQL_CHECK 的支持，分述如下，备忘。

1. GWA2 新增对命令行运行模式的支持
如果使用-GWA2 的 PHP 实现，可以在某些情况下，需要在 PHP 的命令行（command line）模式下执行某些任务程序，如下是需求场景。
有 /?mod=abc&act=read 的程序，Web 访问是

http://domain/path/?mod=abc&act=read

这样可以获得程序的执行，取得预期的结果返回。这一情况是远程的，需要 Web Server 接转，然后交给 Mod_PHP 或者 FastCGI 给 PHP 解释器去执行，然后再将结果返回给 Web Server，Web Server 进一步地返回给用户。这显然有点绕，如果 PHP 处理的业务逻辑是在后台的某项计算。应景的，就是让 PHP 的脚本直接在后台执行，例如，
/usr/local/php/bin/php -c /www/bin/php/php.ini "/path/index.php" "?mod=abc&act=read&fmt=json"

就可能直接和快很多，而且有些时候，Web Server 和（或）Mod_PHP 或 FastCGI 都有执行时长的限制，这对某些需要耗时的程序也不利，于是就想法设法，让基于 -GWA2 写的脚本程序，既可以在浏览器中执行，也可以通过 PHP 脚本执行，事情就这样成了。

主要改动是在入口程序，对于 PHP 命令行的调用，构建一些基本的$_SERVER 变量出来。如此以来，一段脚本程序，既可以在 web 上被访问，也能够在本地通过命令行来执行，也即,

http://domain/path-to-webapp/index.php?mod=abc&act=efg

shell> php "/path-to-webapp/index.php" "?mod=acd&act=efg"

2) 对 NO_SQL_CHECK 的支持

-GWA2 在设计时，考虑了对 SQL Injection 的规避，所有对每次 SQL 的拼接和执行，都做安全性检测，但有时候，限于 Java 或者 PHP 程序本身的问题，如

```
<?php

$idArr = ”1, 2, 3, 4”;

$db = new MySQLi();
$bad_stmt = $db->prepare(SELECT `idAsLetters` FROM `tbl` WHERE `id` IN(?));
$bad_stmt->bind_param(“s”, $idArr);
$bad_stmt->bind_result($ias);
$bad_stmt->execute();
?>
```

就不能参数预期结果，取而代之的是会将整个 $idArr 视为一个 String，而不是一个 list.

反向的，如果我们不这样做，而是将 id list 直接放入 SQL 中，不使用 bind_param 进行变量的代入，在一些情况下也能够通过。

然而，如果 where 语句进一步的复杂成 where url in (URL_LIST)时，URL 中可能包括?，这个字符在-GWA2 的 SQL 检查中会被视为是待替换变量，会由此在运行时变量容器 hmf 中进行查找对应的参数，显然找不到，于是就会触发错误。

有鉴于此，我们在 $_CONFIG 中增加了:

```
$conf['no_sql_check'] = ‘omit_sql_security_check’; # keep original form of sql in some cases, 14:24 Friday, May 20, 2016, refer: http://php.net/manual/en/mysqli-stmt.bind-param.php, “2 asb(.d o,t )han(a t)n i h e i(d.o_t)dk ¶”
```

当访问时，设置某个$obj 为临时不检查 SQL 时（SQL 检查前置），这不对 SQL 做进一步的检查，从而略过这一步骤，如

```
$obj->set($_CONFIG['no_sql_check'], true); # omit sql check
$sql = ‘select * from ‘.$obj->getTbl().’ where url in (‘.$tmpStr.’)';
$result = $obj->execBy($sql, null);
$obj->set($_CONFIG['no_sql_check'], false); # restore it as check
```

34. -GWA2 Java 版本的 i18n/中文编码/乱码问题

（2016 年 7 月 30 日） 本篇问题域被定义为 -GWA2 （-吉娃兔）的 -Java 版本的多语言的编码/乱码问题，也包括中文的编码和乱码问题。当然，也具有普遍意义，包括所有 Java/JSP

应用的中文编码、乱码问题。

这次距离上次写 Blog 间隔的时间较长，主要是由于发布 -GWA2 Java 版本占用了大量的业余时间（包括熬夜），所以拖了一些时间，中间的探索过程(如对反射、序列化的相关思考和实验)还是有值得记录的地方，今后慢慢整理。其中之一就是 Java 的字符编码，或者再收窄成 Java 应用的中文字符乱码问题。

网上搜索 Java 中文乱码的帖子有很多，包括我自己几年前也写作过，如在 2009-12-10 17:05:14 发布的“**javac 编译 java 文件乱码的解决方案**”贴文和 2014 年 11 月 21 日的“**中文编码杂谈**”，谈了在一个 Java 实现的游戏项目中的中文编码问题及相关方面的知识和问题，这次之所以重新谈，是立场发送了改变，不只是解决问题，而是以更好的方式解决问题——**极力避免将宝贵的计算资源用在字符转码上，尽量不在应用代码成使用字符转码的方法，不做字符转码的事情**。

要这样做就是创造统一的运行时环境，包括输入与输出，Java 源码的 i18n 的编码是 unicode，我们选择的统一的运行时字符编码是 UTF-8. 要实现这一目标设计到 Webserver（App container）、Java 类、JSP、HTTP 和数据库后台、HTML 前端等，下面分别记录，以形成可供参考的文档。

0. Java 虚拟机启动时（Webserver、App container 等）

在命令行带入指定字符编码的启动参数

-Dfile.encoding=UTF8

1. JSP 文件页面头部时

设置 JSP 页面的语言及编码

<%@page language=”java” pageEncoding=”UTF-8″%>

很多时候，要求这个语句放 JSP 文件的第一行.

2. JSP 初始化运行时

加入对语言环境变量的指定，调用设置语句

System.setProperty(“sun.jnu.encoding”, “UTF-8”);

System.setProperty(“file.encoding”, “UTF-8”);

request.setCharacterEncoding(“UTF-8”);

3. JSP 对输出内容做字符编码控制

response.setCharacterEncoding(“utf-8”);

response.setContentType(“text/html;charset=utf-8”);

4. 输出内容编码的指定

如果输出的是 HTML 或者 XML 需要在相应的文档里显示地声明字符编码是 UTF-8, 如在 HTML 中

<meta http-equiv=”Content-Type” content=”text/html; charset=utf-8″/>

或者在 XML 中

```
<?xml version="1.0" encoding="UTF-8"?>
```

经过上述 7 个步骤的设置，基本的 UTF-8 环境应该已经统一起来了。然而，如果内容来自数据库（以 MySQL 为例），字符编码仍然是个不小的问题。如果是这样，需要在 JAVA 连接数据库时和进行读写时进行相应的操作。当然前提是，数据库及数据表也是 UTF-8 的编码。

基本上可以确定，读写数据库，如果不进行字符转码且操作正常，需要在如下四个方面取得高度地一致。

Server characterset: utf8

Db characterset: utf8

Client characterset: utf8

Conn. characterset: utf8

对应地在 JSP 中进行补救的相应措施包括:

5. 连接数据库时

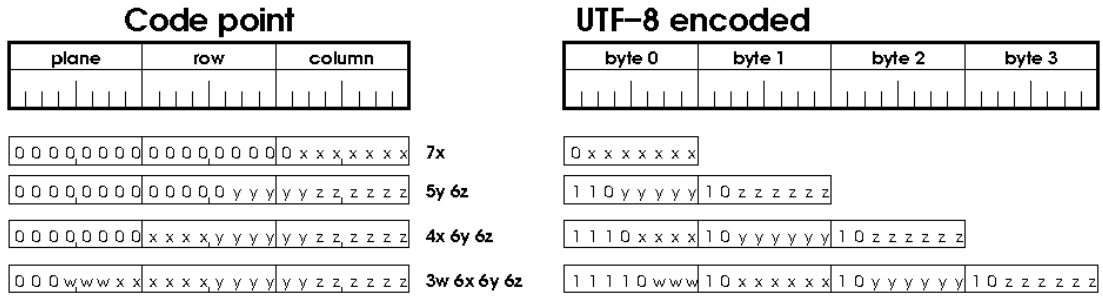
在 JDBC 等驱动加载时，增加字符编码指定参数

setCharacterEncoding=utf8

在进行创建连接进行查询时，首先执行 SQL 语句

set names ‘utf8’

经此 6 个方面的设置，大致成了。-GWA2 是 General Web Application Architecture， 通用网络应用（软件程序）架构，也提供了 -PHP 对应的版本。



附记及扩展:

What Every Programmer Absolutely, Positively Needs To Know About Encodings And Character Sets To Work With Text / 每个程序员都需要绝对、积极地知道一些文本编码和字符集的事情

35. GWA2 写日志的两种方法 setBy 和 debug

（2016 年 8 月 6 日） 在新近的一次 GWA2（吉娃兔）的部署中，被问到在 GWA2 中是否有类似 Logger 的类用来写日志的。答案是有的。
我看了一下现有的代码，大概有两种方法可以实现写日志。

1. setBy $hm_result = $webapp_instance->setBy('file:', $arrayArgs); setBy 进一步地根据 file: 设置将处理转交给 WebApp::writeObject, writeObject 实现对文件的操作。 $arrayArgs 的写法为: $arrayArgs = array('target'=>'/path/to/file', 'islock'=>false, 'isappend'=>true, 'content'=>"what needs to be written to file or log."); 返回值为 HashMap，成功或者失败+原因.
2. debug debug 是全局可访问方法，debug 本身也具有写入文件的功能, 用法为: debug($message, 'something_tag', 'file:/path/to/file'); 这样就将$message 的内容写入指定的日志文件中去，同时在 file 的文件名部分自动追加日期，在 message 消息体自动追加时间戳和日志标记（如 something_tag).
相比较而言，debug 更适合用来写日志。 debug 的设计之初也是用来调试的。debug 还支持其他丰富的调用。在输出时，不但支持指定文件 file: , 还支持在后台、在页面前端等各种方式。对日志内容描述也有更丰富的信息，如调用 debug_backtrace, 将 HashMap 对象遍历，将其他 Object 进行序列化等。

36. -GWA2 更新缓存调用 built-in cache 方法

（2016 年 10 月 15 日） -GWA2 更新增加对缓存的默认集成方法，使用额外指定缓存 Key 的方式可以激活缓存服务。

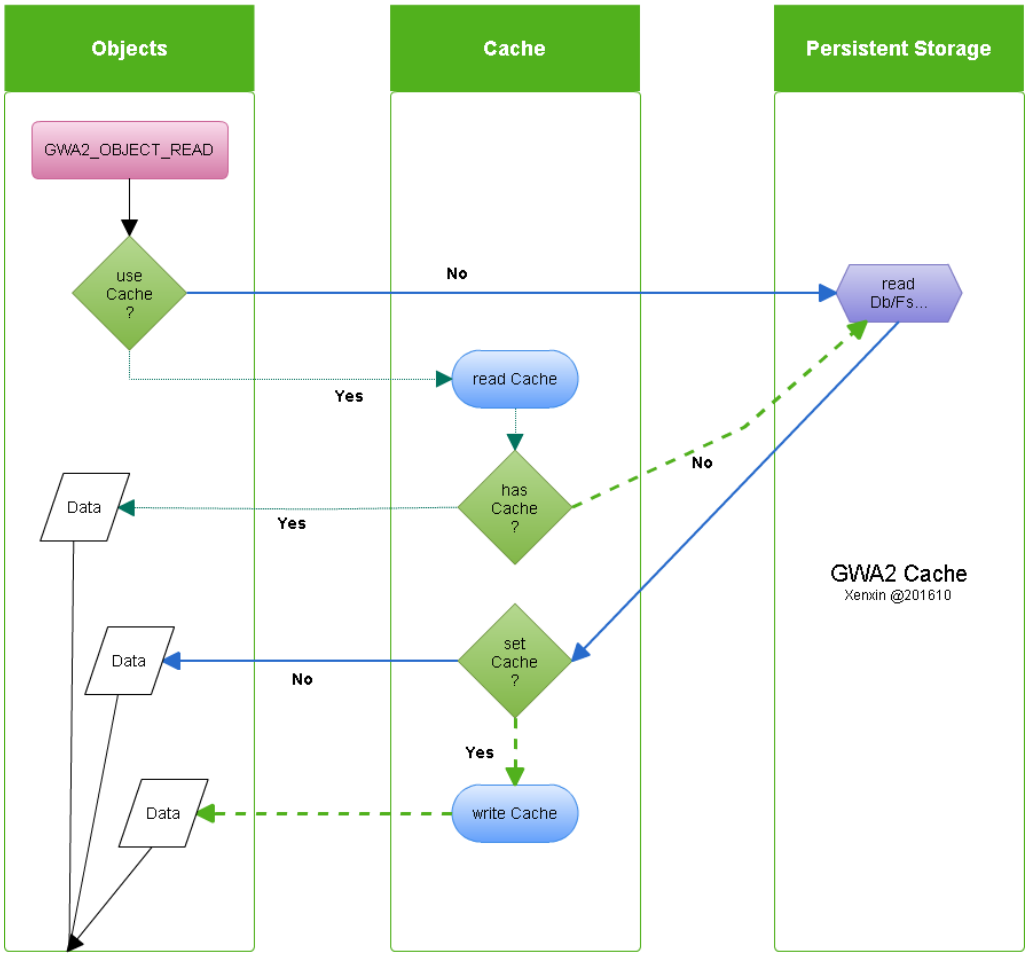
$cacheKey = 'my-cache-key';
$data = $gwa2->getBy("field1, field2, ...", "field1 like ?",
$withCache=array('key'=>$cacheKey));
或者
$data = $gwa2->execBy("select * from tbl_a", "field1 like ?",
$withCache=array('key'=>$cacheKey));

如此，则读取服务会首先根据指定的 $cacheKey 去读取缓存服务，若获得数据，则直接返回缓存中的服务；若没有获得数据，则进一步地调用读取外部数据如 Database 或者 Filesystem, 也即

$data = $gwa2->getBy(“field1, field2, ...”, “field1 like ?”);
或者
$data = $gwa2->execBy(“select * from tbl_a”, “field1 like ?”);

经此被调用的读取方法，会在成功获得外部数据后，调用缓存的写方法，将数据写入到缓存中，已备下次相同请求数据 Key 的读取能够直接从缓存中获得。如此则实现了对缓存的默认集成，built-in cache.

大致流程及图例如下。



之所以使用集成的 Built-in 的方式，是因为如果启用缓存，在多个地方需要重复对 GWA2::getBy/GWA2::execBy 的反复调用，先使用缓存调用一遍，然后判断返回结果，若失败则再使用 Db/Fs 的读取一遍，再判断返回结果，如果是成功的则设置写入到缓存。大致流程如下。

```
$cacheKey = “my-cache-key”;
$data = $gwa2->getBy(‘cache:’, $args=array(‘key’=>$cacheKey));
if($data[0]){
   # read cache succ
}
else{

   # read cache fail
$data = $gwa2->getBy(‘field1, field2, …’, “field1 like ?”);
if($data[0]){

   # read db succ
$gwa2->setBy(‘cache:’, $args=>array(‘key’=>$cacheKey, ‘value’=>$data[1]));

}
else{
   # read db fail
}

}
```

上面的集成（Built-in)本质上就是对这些重复代码进行封装的操作，如此则让开发更快捷。随着实践的增多，或许稍后调用时，$withCache 的参数由一个带有 cacheKey 的数组变为一个 boolean 值， 而 cacheKey 的生成则自动完成。也即,

```
$data = $gwa2->getBy(‘field1, field2, …’, ‘field1 like ?’, $useCache=true);
```

实现上面的操作，离不开 -GWA2 之前的更新操作，如

gwa2 更新多数据库驱动支持

受益于此，基于 -GWA2 的 -gMIS 将首先使用该缓存服务。

-GWA2 是一套通用网络应用（软件程序）架构系统，基于 -GWA2 可以轻便构建各种网络应用程序, 包括复杂的在线购物商城、 旅游交易平台、社群或者社交网站和新闻资讯网站等， 也包括各种企事业单位网上门户，在线交互及服务作业系统等. 还可以包括为 NativeApp 做服务器端支持, 甚至是 WebApp 的全部。

逻辑图的制作，使用 -cacoo 在线完成.