SDH WEB SITE
MVC STRUCTURE

Version 0.0.1

1. Web site structure

root
- ckeditor
- ckfinder
- css
- datatable
- gv
- html2pdf
- htmltodocx
- hv
- icons
- images
- js
- libs
- phpexcel

index.php

login.php

forgot.php

2. Structure of MVC of SDH

We use the MVC structure to develop all the features for SDH web sites. Below is MVC structure exaple.

```
gv
|___ app
|       |__ config
|       |       |__ config.php
|       |       |__ config.yml
|       |       |__ route.php
|       |       |__ route.yml
|       |__ libs
|       |       |_________ helper
|       |__ model
|       |          |_______ base
```

```
|          |__ module
|          |          |_____ Index
|          |          |          |_____ Controller
|          |          |          |          |__  Index.php
|          |__ migrate
|          |__template
|                     |_____ layout
|                     |          |_____ default
|                     |          |          |__  header.php
|                     |          |          |__  footer.php
|                     |          |          |__  index.php
|                     |          |          |__  blank.php
|                     |_____ view
|                                |_____ default
|                                           |__index.php
|___ css
|___ icons
|___ images
|___ js
```

Folder **config** is used to store the file that configures the database connection info (user, pass, server, database), the routing configurations (routing pattern url, parameters of routing).

All the library such as TCPDF, MPDF will be stored to folder **libs**. In this folder, we have some base functions that is used such as helper for the system (get base URL, get root URL etc...)

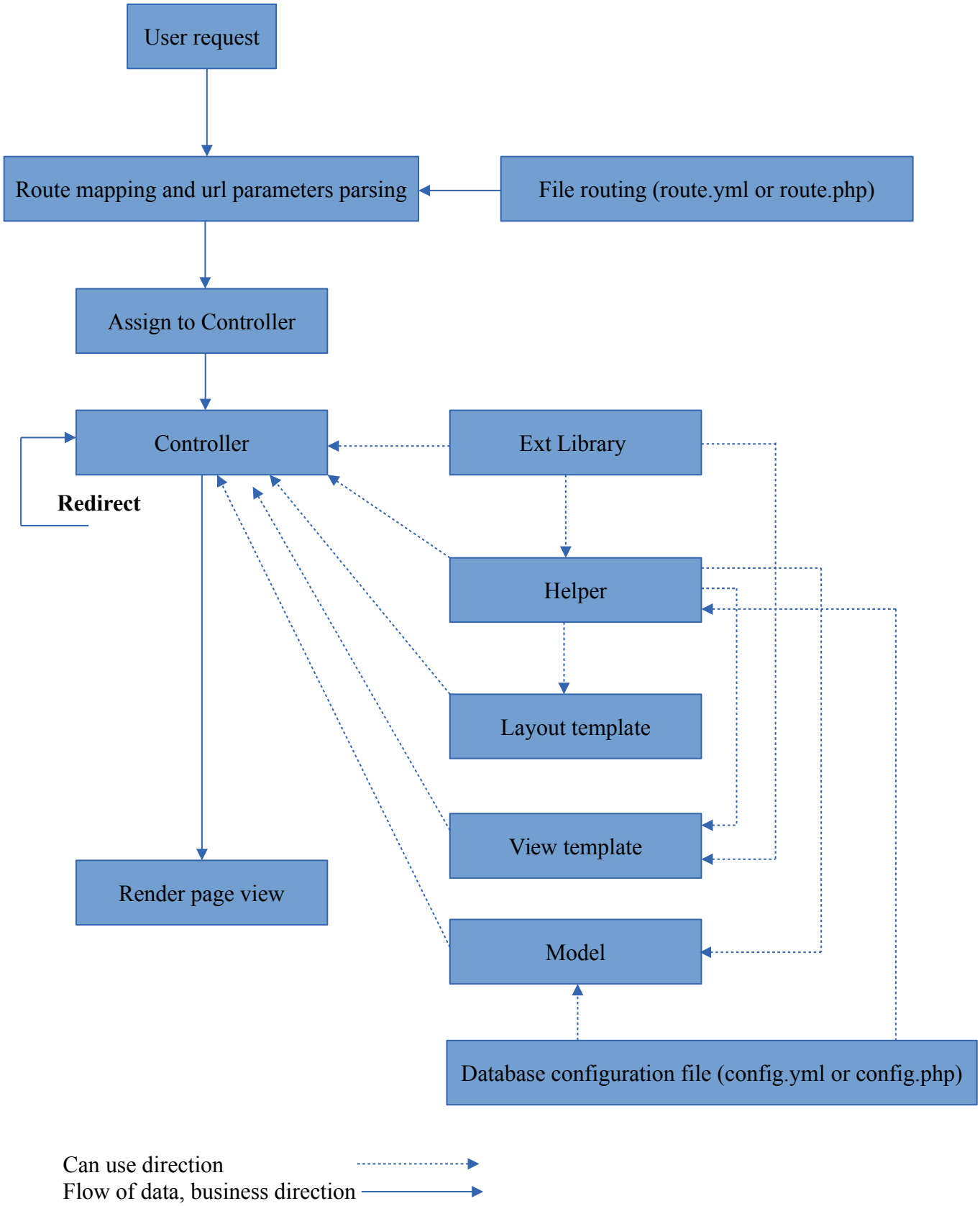
All the templates will be stored to folder **template**, the layout will be stored to the other folder **layout**.

All migration files store in folder **migrate**.

This system also enable to define the object model that have functions can CRUD (create, read, update, delete) the tables in database.

One more thing is you can copy this MVC structure by coping all the folder **app** to the other web site. By the simple way like that, you can make your web site that support MVC structure.

3. Working flow



4. Set the database configuration parameters

This system uses Oracle database, to config the user/pass/host/port we have to open the file **app/config/conf.yml**

```
host: 172.28.40.254
port: 1521
sid:  SDHbk
user: csdl
pass: xxxxxx
charset:  UTF8
```

Or file **app/config/conf.php**.

```
<?php
$config = array(
'host' => '172.28.40.254',
'port' => 1521,
'sid' => 'SDHbk',
'user' => 'csdl',
'pass' => 'xxxxxx',
'charset' => 'UTF8'
);
```

5. How to create a controller

Controller is always belonged to a module. So we need to create a module class first.

The module is created in folder module with name must have the capital on the first character.

Example we create module name Simpletest, just create a folder Simpletest in folder module (path: **app/module/Simpletest**). Then create a folder name Controller in folder Simpletest. The folder structure as below:

```
app
  |__ module
        |__Simpletest
              |____ Controller
```

Next, we need to create a file that will be defined the action. This file always is be capital on the first character.

Example, we create a controller file name **Hello.php** in folder **Simpletest/Controller**.

```
app
  |__ module
        |__Simpletest
              |____ Controller
                    |___ Hello.php
```

In file Hello.php, we define a class that extend class FrontController

```
<?php
/**
 *
 */
class ModuleSimpletestControllerHello extends FrontController {
```

```
	function __construct() {

	}
…...
}
```

Next, we create an action that is a function in class **ModuleSimpletestControllerHello**. We example that is action **begin**.

```
<?php
/**
 *
 */
class ModuleSimpletestControllerHello extends FrontController {

	function __construct() {

	}
	function beginAction(){
		echo "The year param value is '".$this->getParam('year',0)."'";
		echo "<br>";
		echo "This is the example for begin action";

	}
}
```

After done, we still aren't finish. We have to define the routing name that enable the system can parse and assign to this controller.

For example, we have an URL http://sdh.localhost.com/front.php/tkb/phanbo/test/param1/2013?hisid=s3368aidfqph6eh78ifto1c2u5#

In this URL, we see:

- **tkb**: is the name that is mapped with a controller in file route.yml
- **phanbo**: is mapped to controller file name. Note that the first character is lower that is difference with the name of controller file name.
- **test:** is action name in controller file Phanbo.php
- **param1**: name of the first parameter
- **2013**: is value of the first parameter

Now, we open the file **route.yml**, we will see the define as below:

```
tkb:
…
  Thoikhoabieu:
    Phanbo:
      …
      test:
        param1: 0
```

Or open file route.php, we also see the define such as:

```
<?php
$config = array(
```

```
      …...
      'tkb' => array(
                  'Thoikhoabieu' => array(
                                          'Phanbo' => array(
                                                'test' => array(
                                                            'param1' => 0
                                                            )
                                                )
                  )
      )

      …...
);
```

Why we have the file route.php. The root cause is that in some system, we can't install YAML lib, so that PHP cannot parse the file route.yml to array object. The file route.php is the second way that to make sure system can parse the URL and map to the right controller.

In my example, we have to create action begin. We map **Hello.php** controller file with mapping name is **firstexample.** The parameter that we want to create name is **year**. To do that, we follow the steps below:

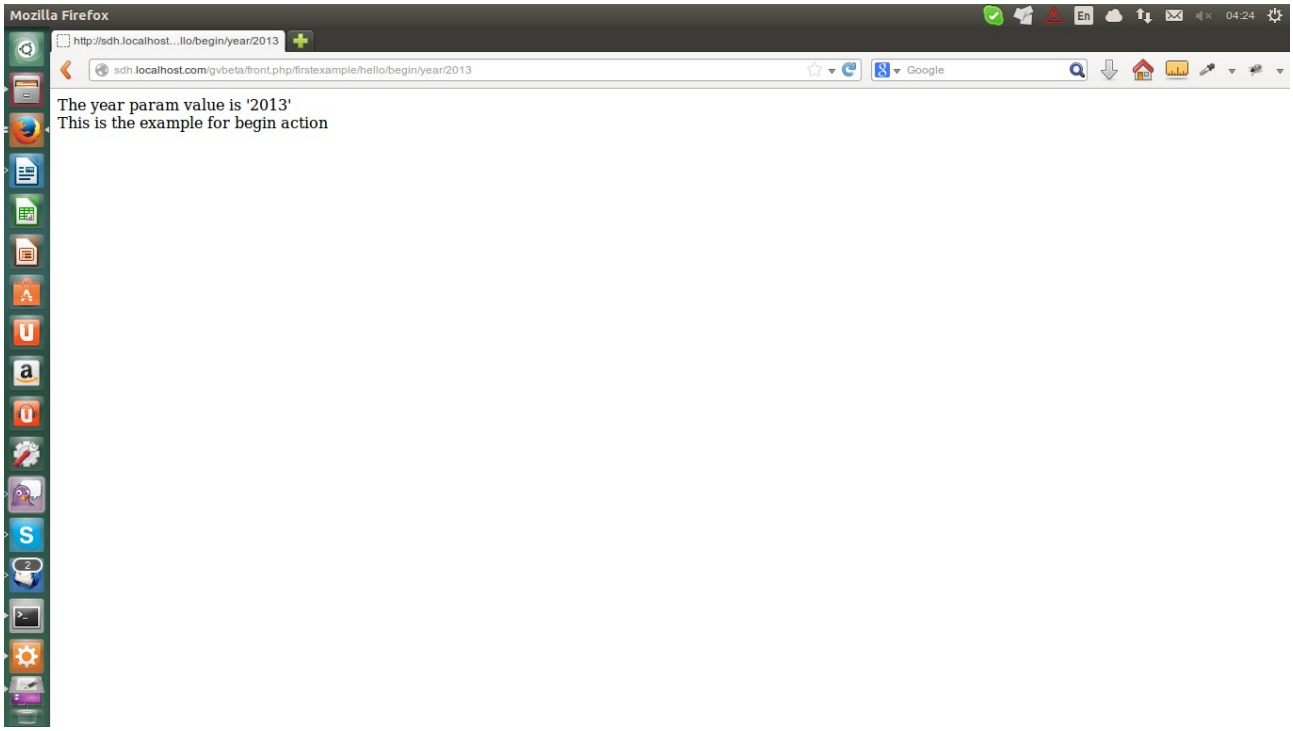
- Open file : app/config/route.yml
- Put the text as below to file

```
firstexample:
  Simpletest:
    Hello:
      begin:
        year: 0
```

- Open the file route.php, we have to put as below:

```
<?php
$config = array(
      'firstexample' => array(
                  'Simpletest' => array(
                                          'Hello' => array(
                                                'begin' => array(
                                                            'year' => 0
                                                            )
                                                )
                  )
      )
);
```

After this step, we open the browser and put the link http://sdh.localhost.com/gvbeta/front.php/firstexample/hello/begin/year/2013. We will see the page as below.



6. How to create model

Model is an definition that is mapped with a table in database. We can use this model can CRUD (create, read, update, delete) row in database.

Example we need to create a model name **TableTest**. This model maps to table as the same name. Below are my steps.

- Create a file **TableTest.php** in folder model.

```
app
  |__ model
        |__TableTest.php
```

- In that file, we have to define a class extends from **BaseTable**

```php
<?php
/**
 *
 */
class TableTestModel extends BaseTable {
	public $connection;

	function __construct() {
		parent::init("table_test");
		$this->connection = DbFactory::getInstance();
	}

	function __destruct() {
		parent::__destruct();
```

```
	}
	function test() {
		$sql = "select * from ".$this->tableName."";

		$check = $this->getQuery($sql)->execute(false, array());

		$ret = array();

		if($check->itemsCount > 0){
			$ret = $check->result;
		}
		return $ret;
	}
}
```

- We add an action in controller **Hello.php** with name **testmodel** (define function, route)

```
<?php
/**
 *
 */
class ModuleSimpletestControllerHello extends FrontController {
	function __construct() {

	}
	function beginAction(){
		echo "The year param value is '".$this->getParam('year',0)."'";
		echo "<br>";
		echo "This is the example for begin action";
	}
	function testmodelAction(){
		$model = new TableTestModel();
		$count = $model->test();
		print_r($count);
	}
}
```

7. How to create a view template

Sometime we have some pages have the html is the same, we can create a view that can reuse.

In this system, the view template we can set some parameters that can bind value from controller action.

First, we have to create folder and files for an example template as structure below:

```
app
  |__ template
        |__view
              |__simple
                    |___index.php
```

The view file **index.php** content:

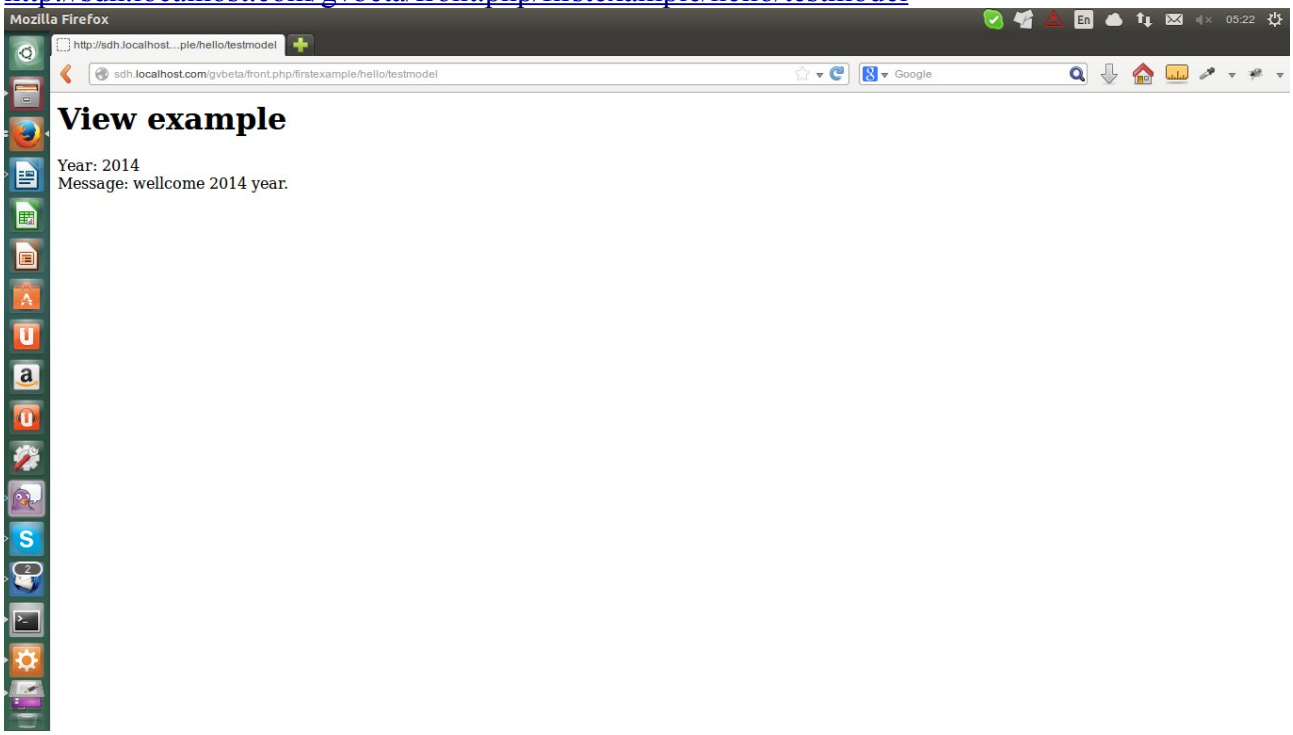
```
<h1>View example</h1>
Year: <?php echo $year; ?><br>
Message: <?php echo $message; ?><br>
```

Update code for action **testmodel**

```
	function testmodelAction(){
```

```
            $template = new BaseTemplate("simple/index","default/blank");
            $template->year =  2014;
            $template->message = 'wellcome '.$template->year.' year.';
            $template->renderTemplate();
      }
```

In browser, we will see the page after refresh URL
http://sdh.localhost.com/gvbeta/front.php/firstexample/hello/testmodel



8. How to create a layout template

Template layouts is the same template of view. It is meaning a group of many views of template. Example, we can have layout one that has the header, footer . The other layout can have more with the right menu.
In this system, when you want to render the layout, you have to set the parameters by another way that is difference with view of template.

```
$nhanSu = new NhanSuModel();
if ($this->checkLogin() && $nhanSu->checkPhanBoCanBo() ){
      $template = new BaseTemplate("tkb/khoa/phanbocbgd","default/index");
      unset($nhanSu);
      $tkb = new ThoiKhoaBieuModel();
      $template->listMonHoc =$tkb->getListHocKy();
      unset($tkb);
      $config = new ConfigModel();
      $template->dothoc =$config->getPhanBoCbgdDotHoc();
      unset($config);
      $templateContent = $template->contentTemplate();
      $template->renderLayout(array('title' => '','content' =>
$templateContent));
}
```

The red highlight is the way how to bind the values to view of template. And the yellow highlight is the way how to bind values to layout of template. A layout template need to create in folder **app/template/layout**.

9. How to create helper library

Helper is some class object that can help developer can reuse many functions that repeat many time in source code such as: get root URL, get the root folder path etc...
Developer can define function and call it in controller, model, view and layout.
To create a helper class, first we have to create the structure folder as below:

```
app
  |__  libs
        |__helper
              |__Simplehelper
                    |___Common.php
```

As the structure upper, we have to define a class in file Common.php with content:

```php
<?php

/**
 *
 */
class HelperSimplehelperCommon {

	function __construct() {

	}

	function testHelper() {
		echo “Your helper”;
	}

}
```

You can call the helper and function with :

```php
$helper = Helper::getHelper('simplehelper/common');
$t = $helper->testHelper();
```

10. How to add an extension library

One more thing is very important in this system is that it can include and use many of open source of PHP library such as: tcpdf, mpdf etc...

How we can do that, just very simple, we copy the library to folder **app/libs**. Then we declare as the guiding line in library. Almost them have the example index.php file. It will let you know to can import and use them.

Example in this system is TCPDF library. After copy all source code to folder libs, I just put a include function in file **front.php**.

```php
<?php
date_default_timezone_set('Asia/Ho_Chi_Minh');
```

```php
// Include the main TCPDF library (search for installation path).
require_once('./app/libs/tcpdf/examples/tcpdf_include.php');
// Include mPDF library.
require_once("./app/libs/mpdf57/mpdf.php");
//Add auto loader
require_once ('./app/libs/res/auto_loader.php');
//Add route map
include './app/libs/res/route.php';
//Add front end class
include './app/libs/res/front.php';
//Add template object
include './app/template/index.php';
//Add base table
include './app/model/base/basetable.php';
//Add helper static class
include './app/libs/helper/helper.php';
$route = new Route();
```

Now, you can use TCPDF library every where.

11. How to use migartion

This system have a function to create some task file that can modify database (CRUD row). This feature is really helpful, it help developer can do deployment easily in many system and make them are the same the structure.

Ex: in system 1, you created a new table but the system 2 hasn't. To do that, developer can make a migration file that will create new table in system 2. So that the system 2 is same system 2. We call that is same migration version.

Below are steps to use migration feature:

1. Get the help information by:
    1. Go to folder root/app
    2. run command: **php migration.php -h**

```
hpnguyen@hpnguyen:~/Working/svn_repository_source/gvbeta/app$ php migrate.php -hUsage:
php migrate.php [options] <name|version>
[-h]          Get help.
[-v]          Get current migration version.
[-c]  <name>    Create migration file. Name must be <xxxx_yyyy_zzzz> and unique.
[-u]  <version> Upgrate current to option version.
[-u]          Upgrate current up to one new version.
[-ua] <version> Upgrate from version to the lastest version.
[-ua]         Upgrate from current version to the lastest version.
[-d]  <version> Downgrate from current version to option version.
[-d]          Downgrate from current to one older version.
[-da] <version> Downgrate from option version to the first version.
[-da]         Downgrate from current version to the first version.
```

As the guiding information, you can create, execute to upgrade or execute to downgrade the migration version.