

Modules

Our web application is consist of modules. Each module is written in separate file and it is defined as a class. This class must have at least one function (`_default`). `_default` function is first called.

In paragraph "Smarty functions" is described how to call another function.

In each function these variables are available:

```
$this->project_id
$this->project_name
$this->page_name
```

Each public function must return hash array with defined key `'template'`. This is name of smarty template which is used to generate module content. Key `'data'` is optional but if it is defined value must be an array. Elements of this array are passed to Smarty template.

Example:

```
return array('template'=>'test.tpl', 'data' => array('v1' => 'value1', 'v2' => 'value2'));
This will call template test.tpl and pass to it variables {v1} and {v2}.
```

Page

Page files are used to include module to page. Admin pages have prefix `"adminpage_"` and all other have pages have prefix `"page_"`. See `pages/` directory for details.

Default template for displaying page is `page.tpl` which is based on Laszlo's design. It has 3 columns named 'left', 'middle' and 'right'. Also, there is available `'top_left'` and `'top_right'` which are used for top navigation and logout.

Smarty functions

{wp_action}

Prints `<a>` HTML element with coded information about module and function to call.

Parameters:

Key	Type	Description
<code>_name</code>	string (optional)	<code><a name="\$_name"...</code>
<code>_id</code>	string (optional)	<code><a id="\$_id"...</code>
<code>_anchor</code>	string (optional)	<code><a href="...#\$_anchor"...</code>
<code>_function</code>	string (optional)	<code><a href="...\$_function..."...</code>
<code>_value</code>	string (optional)	<code><a ...>\$value</code>
<code>_title</code>	string (optional)	Mouse over title
<code>_class</code>	string (optional)	Class for displaying link
<code>_target</code>	string (optional)	Link target (<code>_blank</code> , <code>_self</code> etc.)
<code>_method</code>	string (optional, default: get)	Get or Post. Post is used for changing data, e.g. deletion of user.
<code><param*></code>	mixed	Variable that will be encoded in href

Example:

```
{wp_action _function="help" var="this is variable"}
```

This will produce `<A>` element with some encoded information. Following that link and generating module content on same page will call function `help($params)`. `$params` is array and one row of that array is `'var' => 'this is variable'`.

{wp_form}/{wp_form}

Prints <form></form> HTML element with coded information about module and function to call.

Parameters:

Key	Type	Description
_name	string (optional)	<form name="\$_name"...
_id	string (optional)	<form id="\$_id"...
_anchor	string (optional)	<form action="...#\$_anchor"...
_function	string (optional)	<a action="...\$_function..."...
_target	string (optional)	Target (_blank, _self etc.)
<param*>	mixed	Variable that will be encoded in href

Example:

```
{wp_form _name="Hello form"}  
  <input type="text" />  
  <input type="submit" value="Submit" />  
{/wp_form}
```

{wp_message}

Purpose of this function is to make all messages in application look in a same way. For example see mod_example3 and page example on any project.

Parameters:

Key	Type	Description
title	string (optional, default: <type>)	Message title
show_title	boolean (optional, default: true)	If false, message will be generated without title
type	string (optional, default: notice)	For each type there is a different color set. Notice (green), Warning (orange), Error (red)
value	string	Message text

Title
Value

Title
Value

Title
Value

{\$var|count} – Count modifier

Example:

```
{$my_array|@count} – Output is number of elements in array $my_array
```

Debug

PHP

You can use `dump_var` function to dump variables to screen. This function is wrapper for PHP standard function `var_dump` that puts its output in `<pre></pre>` HTML tag so it's easier to read. Variable can be array, string, integer etc.

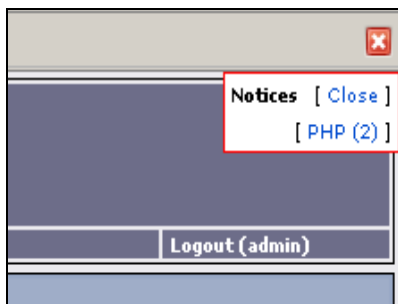
Example: `dump_var ($GLOBALS);`

If error or warning occurs during page generation, generation will be terminated and all generated content will be deleted. Information about error is displayed. Error code, Error description, GET parameters, Stack.

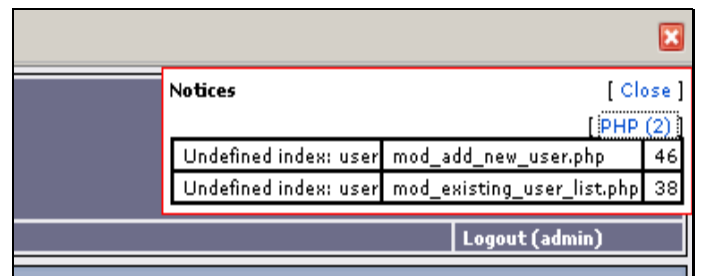


Picture 1: User Error

All notices are saved in two arrays. One array is for PHP notices and the other is for Smarty. Notices are displayed in upper right corner of the page. First column is error description, second is file name and third line number.



Picture 2: Two PHP Notices



Picture 3: Expanded PHP Notices

Smarty

For debugging in Smarty you can use `{debug}` function.

Global Variables

Global Variables

For accessing global variables from function use global keyword before variable is used (`global $variable_name;`) or access variable through array `$GLOBALS['variable_name']`

\$href_* (string)

Smarty: `{$href_base}`, `{$href_all}` etc.

Variable	Description	Example
<code>\$href_base</code>	Base URL	<code>/~kkroflin/webproject</code>
<code>\$href_application</code>	Web Application	<code>/~kkroflin/webproject/index.php</code>
<code>\$href_projects_root</code>	Listing all enrolled projects	<code>/~kkroflin/webproject/index.php/projects</code>
<code>\$href_admin_root</code>	Administrator home page	<code>/~kkroflin/webproject/index.php/admin</code>
<code>\$href_project</code>	Home page of project	<code>/~kkroflin/webproject/index.php/projects/<project_name></code>
<code>\$href_page</code>	Page currently displayed	<code>/~kkr....php/projects/<project_name>/<page_name></code>
<code>\$href_all</code>	Same URL as calling	<code>/~kkr....php/projects/<project_name>/<page_name>/...</code>

\$_info (array)

Smarty: `{$_info.project}`, `{$_info.page}` etc.

Key	Description
<code>IS_ADMIN</code>	True if user is Administrator
<code>IS_MANAGER</code>	True if user is Manager
<code>IS_VICEMANAGER</code>	True if user is Vice Manager
<code>IS_MEMBER</code>	True if user is Member
<code>project</code>	Project name
<code>project_id</code>	Project ID
<code>page</code>	Page name
<code>lang</code>	Language
<code>user</code>	Information about user (array) nick, title, first_name, last_name, email, design, language, administrator, enabled
<code>level</code>	Level can be: admin, projects

Additional Smarty variables

Variable	Description	Example
<code>_anchor</code>	Anchor name for linking to position at current module	<code>#mod_login</code>
<code>_module_name</code>	Module name	<code>mod_login</code>
<code>_function</code>	Module function (method)	<code>_default, login</code>
<code>IS_ADMIN</code>	True if user is Administrator	
<code>IS_MANAGER</code>	True if user is Manager	
<code>IS_VICEMANAGER</code>	True if user is Vice Manager	
<code>IS_MEMBER</code>	True if user is Member	
<code>IS_LOGGED</code>	True if user is logged on	

Language definitions (Smarty)

Language definitions that are specified in `lang/<code>/default.lang.php` are available through variable `{$_langdef}`.

Example: `{$_langdef.accept}`

Definition that are in `lang/<code>/<module_name>.lang.php` are available through variable `{$_lang}`

Useful PHP Functions

Function	Description
<code>is_admin()</code>	Returns true if user is Administrator
<code>is_manager()</code>	Returns true if user is Manager
<code>is_vicemanager()</code>	Returns true if user is Vice Manager
<code>is_member()</code>	Returns true if user is Member
<code>is_logged()</code>	Returns true if user is logged in
<code>dump_var()</code>	Dumps a string representation of variable to output surrounded by <code><pre></pre></code> tags
<code>to_bool()</code>	Return true if variable seems to be true. Useful for DB queries if parameter is boolean type. <i>Example:</i> <pre>\$row = \$db->getRow("SELECT * WHERE nick = ?", \$nick); \$is_administrator = to_bool(\$row['administrator']);</pre>
<code>trigger_error()</code>	Throws error. Don't use this function if you want to inform user about error in a nice way and give him a chance to correct error. This produces "ugly" development error description page. First parameter is text message, and second <code>E_USER_ERROR</code> or <code>E_USER_WARNING</code>

Database

It is very important to escape string variables. Also it is very important to check data that you wish to embed in SQL query.

Watch out of SQL injection.

There are 2 ways to include variable in query.

Through parameter

This requires that variables are the same type as those in database (integer, string, boolean)

Example:

```
$db->getAll("SELECT * FROM users WHERE nick = ? AND pass = ?", array($nick, $pass));
```

Escaping variables

Example:

```
$db->getAll("SELECT * FROM users WHERE nick = `"  
    . $db->quoteSmart($nick)  
    . "` AND pass = `"  
    . $db->quoteSmart($pass)  
    . "`");
```