lansuite

Documentation: write your own lansuite module

1) Prologue

This documentation will explain to you how to write your own lansuite module. It describes the basic classes and functions which make it easy for you to use some often performed procedures of lansuite.

The first step will be the explanation of how to create a new module. It describes which files have to be changed, where you have to place your new files and which URL you have to enter to view them.

The second step will be the explanation of how to use lansuite's display-class, which makes it easy to implement and use the characteristic design-elements of lansuite.

Next will be the usage of multi-language support.

If you have finished writing your own module and think it could also be useful for other Lanparty-teams, we would be pleased if you'll send us your new module, so we can put it into the next release, if we think so either.

I'll hope this documentation proves useful and we'll get some great new modules in return :-)

2) Create a new module

First of all you'll have to be introduced to lansuite's directory structure.
The root-directory consists of the following, important files:

- modules: This is the directory where all the modules are placed, no matter if they are called from the navigation-panel, or some other box (like the messenger), or link (like the mastersearch – ill come to this particular module later).
  As you surely already have guessed this is the most important directory for you, since you want to write your own module.
- Design: In this directory all of lansuite's design-elements are placed. No need for you to change here anything, since all module related design elements, not already included in the display-class (ill come to that later), shall be placed in a subdirectory of your module called "templates"!
- Ext_Inc: If your module uses some files (like the pictures in the picturegalery, the tournament rules in of the tournaments, etc.) they shall be placed in a subdirectory of "ext_inc".
- Index.php: This is the main file of LanSuite. ALL pages of lansuite must be called by this file. For example the news file is called by the URL: "index.php?mod=news"
- Inc/classes: This is the directory, where all the main functions of lansuite are stored. Some of these will be explained later.

So to create your own module you'll have to switch to the directory "modules" and create a new directory inside, with the name of your new module.

2.1) modindex.php:

Inside your module directory you'll first have to place a file called "modindex_MODULENAME.php" where MODULENAME is the name of your module. The same name which you gave to your modules directory.
The information inside this file is used to specify which file shell be loaded according to the action which have been given by the URL-parameters.
Therefore this file should be read each time someone calls a page of your module.
To do so, open the file "index_module.inc.php", which is located in the root-directory of lansuite, with your favourite text editor an add the following lines in the "switch($module)"-section:

```
case "MODULE":
        include_once("modules/MODULE/language/MODULE_lang_".$language.".php");
        include("modules/MODULE/modindex_MODULE.php");
break;
```

You'll have to replace the five entries of "MODULE" with the name of your module.

Next you'll have to create the Modindex-File. Goto your module directory and create the file "modindex_MODULENAME.php".

Let's give an example for the content of this file:
The Modindex-File could look like this:
(MODULE, again, is the name of the directory of your module)

```
<?
switch($_GET["action"]) {

        case show:
```

```
            include("modules/MODULE/show.php");
        break;

        case add:
            if ($_SESSION["auth"]["type"] > 0) {
                include ("modules/MODULE/add.php");
            } else $func->error("ACCESS_DENIED","");
        break;

        default:
            include ("modules/MODULE/show.php");
        break;
}
?>
```

As you can see this file describes which file of your module should be included according to the action performed (which is submitted by the URL).

For example the URL
http://localhost/index.php?modules.php&mod=MODULE&action=show
would display the content of the file “show.php” located in your modules directory.

As you can also see in the example above, the Modindex-file is used to determinate which users shall have access to the preformed action. The variable SESSION["auth"]["type"] contains the access level of the current user. The value 0 means that he is a user having no rights. 1 stands for Admins and 2 for Operators. Whether an action is permitted only to operators, or also to Admins is the decision of the module writer. Basically a limitation only to Operators is used by actions causing the removal of some entries or other deep changes in the database.
The Variable “$_SESSION["auth"]["login"]” holds the information of the users current login state. It returns 1 if the user’s currently logged in, or 0 if he’s not.

3.) Often used Functions, especially for the output of information.

Another thing, you may have recognized in the example of the modinedx-file is the usage of the "$func->error("","");"-Function.

3.1) $func->error($error, $link_back);
This displays the Error-Message $error. The optional parameter $link_back may contain an URL which will be executed by clicking the Back-Button in the error-message.

But before ill describe some more functions here, lets create our first own page.
Create a file named "show.php" in your modules directory and write the following lines inside:

```
<?
$dsp->NewContent("My first lanuite page", "This is an example of a simple page");
$dsp->AddSingleRow("Huh, that was easy :-)");
$dsp->AddContent();
?>
```

Now you should be able to access this file by calling the URL
http://localhost/index.php?modules.php&mod=MODULE&action=show

3.2) $dsp->NewContent ($caption, $sub_caption);
This function initiates a new output. It should be called before any other function of the display class, but only once on a page. The Parameters $caption and $sub_caption display the title of the page

3.3) $dsp->AddContent();
This function should be called after all other functions of the display class, but also only once on a page. It writes all the display-data to the screen. If this function is not executed, there will be no output.

3.4) $dsp->AddSingleRow($content)
This displays a single row containing the Text $content. In the example above this displays the text "Huh, that was easy :-)".

3.5) $dsp->AddDoubleRow($content1, $content2)

This is quite the same, as AddSingleRow, with the only difference, that now there are to columns in the output, each containing the corresponding parameter.
This is mostly used to display a key and its value.

3.6) Use Forms

To create a form there are at least 2 more functions to be called:

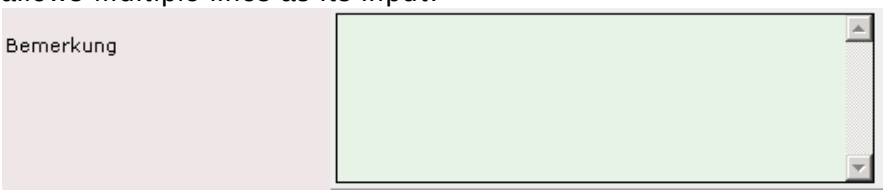
1) $dsp->SetForm($form_url, $form_name = NULL, $form_method = NULL)
   This function initiates the form. $form_url is the URL, which will be launched after the Submit-Button was pressed. The other two, optional parameters define the name of the form (for example if you use JavaScript an want to address a child value of this form) and the method which shall be used to transmit the data. Default is POST.
2) $dsp->AddFormSubmitRow($button, $helplet_id = NULL)
   This will add a Submit-Button and close the form.
   The optional parameter $helplet_id can contain the name of a helplet file. If this parameter is given, then a help-button will be displayed, linking to “doc/online/$helplet_id.htm”.

In between these two functions there may be used any of the following:

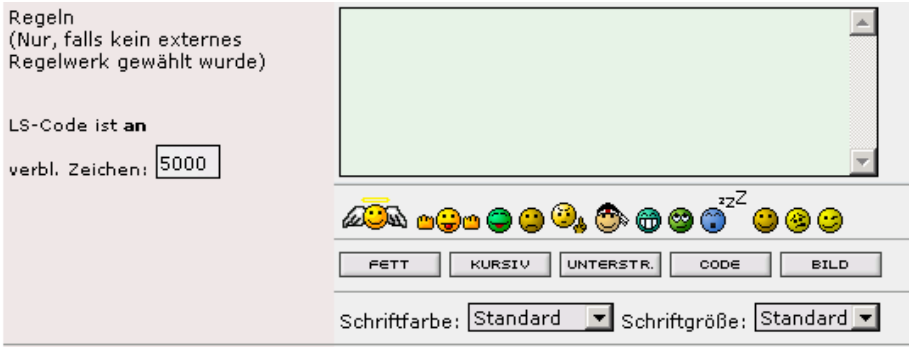
- **Textfield**:
  $dsp->AddTextFieldRow($name, $key, $value, $errortext, $size = NULL, $optional = NULL)
  This function creates a standard-single-line-text-input, where $name is its name in the form, $key the description and $value the standard input-value. $errortext can beused to display errors. For example if no data has been submitted, in spite of its necessity. With $size a size for the textbox can be defines. If the parameter $optional is set to 1, then lansuite will use a style sheet which symbolises that this input is just optional.

- **Checkbox**:
  $dsp->AddCheckBoxRow($name, $key, $text, $errortext, $optional = NULL, $checked = NULL, $disabled = NULL)
  This function creates a checkbox.
  All parameters are similar to the ones from AddTextFieldRow.
  If $checked is set to 1, then the checkbox will be checked by default.
  The parameter $disabled can be used to mark the checkbox read-only.

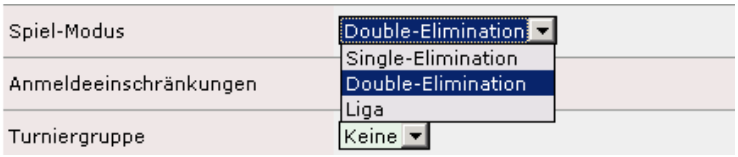
- **Textarea**:
  $dsp->AddTextAreaRow($name, $key, $value, $errortext, $cols = NULL, $rows = NULL, $optional = NULL)
  AddTextAreaRow is quite the same, as AddTextFieldRow, with the only difference, that it allows multiple lines as its input.



- **Textarea** (with JavaScript-Buttons to insert emoticons, and text formats):
  $dsp->AddTextAreaPlusRow($name, $key, $value, $errortext, $cols = NULL, $rows = NULL, $optional = NULL)
  This is the extended version of AddTextAreaRow.
  Below the input, there are some buttons and emoticons, which help the user to input some code (for example [b] to display bold text.), by using JavaScript.
  To show this code correctly, you shall later, when printing this, make use of the function $func->text2html($text), which replaces code like [b], with its according html-code (<b>).

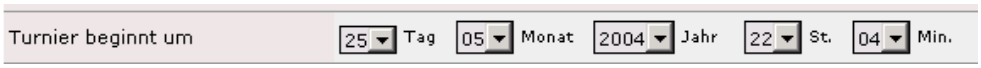


- **Dropdown**:
  $dsp->AddDropDownFieldRow($name, $key, $option_array, $errortext, $optional = NULL)
  This function Displays a dropdown-menu.
  $option_array contains all the keys and values, which the dropdown shall contain.



- **Dateselect**:
  $dsp->AddDateTimeRow($name, $key, $time, $errortext, $values = NULL, $disabled = NULL)
  Use this function to create a date-input.
  This input contains of 5 dropdowns (Year, Month, Day of month, hour, minute).
  You can set the default date either by $time, or $values.
  This is expect to be a timestamp, an $values an array, which looks like the following:

```
$values["day"] = 25;
$values["month"] = 5;
$values["year"] = 2003;
$values["hour"] = 21;
$values["min"] = 42;
```



- **Pictureselect** (Dropdown):
  $dsp->AddPictureDropDownRow($name, $key, $path, $errortext, $optional = NULL, $selected = NULL)
  This function reads all Pictures (jpeg, jpg, png, gif) from the directory $path and lists there names in a dropdown-menu. By selecting one of its entries a java script displays a thumbnail of the selected entry.
  If GD-Lib is enabled and the path $path is writeable, lansuite will generate the thumbnails and store them in this path, so they have to be generated only once.

- **Pictureselect** (List):
  $dsp->AddPictureSelectRow($key, $path, $pics_per_row = NULL, $max_rows = NULL, $optional = NULL, $checked = NULL, $pic_def_width = NULL, $pic_def_height = NULL)
  This function reads all Pictures (jpeg, jpg, png, gif) from the directory $path and lists them as thumbnails. Each row contains $pics_per_row. After $max_rows the output stops.
  If GD-Lib is enabled and the path $path is writeable, lansuite will generate the thumbnails and store them in this path, so they have to be generated only once.

3.7) Some more display related functions

Here are some more display related functions, which may also be used outside a form-tag:

- **BackButton**:
  $dsp->AddBackButton($back_link, $helplet_id = NULL)
  This adds a back button, where $back_link is the target link of the button and $helplet_id the path to he online-help to this page. The online-help-file will be searched in the following path: doc/online/$helplet_id.htm

- **Horizontal line**:
  $dsp->AddHRuleRow()
  This adds a horizontal line. That's good to optically separate some content.

- **Sub-Navigation**;
  $dsp->AddHeaderMenu($names, $link, $active = NULL)

- **Information box**:
  $func->information($text, $link_target)
  This function should be preferred to $func-error(), if it is no error causing the system to stop its functionality.

- **Confirmation box**:
  $func->confirmation($text, $link_target)
  This function can be used to confirm an action, which was performed successfully.

- **Yes-No Question box**:
  $func->question($text,$link_target_yes,$link_target_no)

- **Question box**:
  $func->multiquestion($questionarray, $linkarray, $text)

- **Dialog box**:
  $func->dialog($dialogarray, $linkarray, $picarray)

3.8) Add your own template

If all these functions, described above, do not deliver the output you need, lansuite gives you the ability to write your own template for your mod.
This is done, by using the following function:

```
$dsp->AddModTpl($mod, $name)
```

Where $mod is the name of your module and $name the name of the template.
This template needs to be placed in the following place:
“modules/$mod/templates/$name.htm”

```
If ( foo == bar) {
	$sample = $dsp->FetchModTpl("c2","main");
} else {
	$sample = $dsp->FetchModTpl("c2","main_2");
}
$dsp->AddSingleRow("Beispiel", $sample);
```

4) Often used Functions (mysql)

Let's start with a simple example:
This will display the username of the current user.

```
$users = $db->query("Select username FROM {$config["tables"]["user"]} WHERE userid = '{$_SESSION["auth"]["userid"]}'");
while($user = $db->fetch_array($users)) {
	echo $user['username'];
}
$db->free_result($user);
```

As you can see, the "$db->query"-function is used to get the results of a mysql-query.
You may also have recognized that all the table names in lansuite are stored in variables. (for example: {$config["tables"]["user"]}). These variables are defined in the file "inc/base/lansuite.conf" in the [modules]-section.

The function "$db->free_result();" releases the query.

Now that the result of the query in this example delivers only one row, there is a much easier way to show the username of the current user:
The "$db->query_first()"-function returns the first row, of the result:

```
$user = $db->query_first("Select username FROM {$config["tables"]["user"]} WHERE userid = '{$_SESSION["auth"]["userid"]}'");
echo $user['username'];
```

5) Multilanguage system

To add Multilanguage-support to your module, you have to create a directory named language, containing a file named MODNAME_lang_LANGUAGE.php (for example: news_lang_de.php).

Inside this file you have to define all Texts which your module uses.
Here an example from the german language file of the newsmodule:

```
$lang["news"]["add_caption"]      = "News hinzufügen / ändern";
$lang["news"]["add_subcaption"] = "Mit Hilfe des folgenden Formulars können Sie eine Newsmeldung hinzufügen / ändern.";
$lang["news"]["add_headline"]     = "&Uuml;berschrift";
```

So whenever you want to write the Text "News hinzufügen / ändern" in your module you have to put there the variable $lang["news"]["add_caption"].

To read these variables from your languagefile, you have to include your languagefile in the file index_module.inc.php.
This can be done, by adding the following line at the line, where switch($mod) = “YOUR_MODULE”:

```
include_once("modules/news/language/news_lang_".$language.".php");
```