



Web site for team-sport  
clubs and leagues

# Developer manual

PhpMySport v1.2  
Released on March 8<sup>th</sup>, 2007

Software distributed under GNU/GPL licence

Copyright © 2007

# Table of contents

- Introduction ..... 2**
  - Who is this manual aimed at?.....2
  - PhpMySport .....2
  - Before beginning.....3
- Objectives, constraints and methods .....4**
  - Objective.....4
  - Constraints .....4
  - Design methods .....5
  - Languages Used.....5
- Organization of the software.....7**
  - Software architecture .....7
  - File structure.....8
- Principal techniques used..... 10**
  - Language files..... 10
  - Generic functions for database integrations ..... 11
  - URL rewriting: easily-understood URLs..... 11
- Design and templates ..... 13**
  - What is a template?..... 13
  - The template code ..... 13
  - Creating your own design ..... 14
- Personalising phpMySport with plugins ..... 16**
  - What is a plugin? ..... 16
  - File hierarchy of a plugin..... 16
  - How does one create a new plugin?..... 18
  - Example: web link management plugin ..... 19
- Common constants and functions that may be used.....24**
  - Constants ..... 24
  - Functions..... 25
- Programming Conventions and Rules ..... 29**
  - Naming Conventions..... 29
  - Code Structure..... 29
  - Other Rules ..... 30
- Credits ..... 31**
  - Authors ..... 31
  - Version..... 31
- Appendices ..... 32**
  - Physical model of the database (phpMySport v1.1) ..... 32

# Introduction

## Who is this manual aimed at?

This manual is aimed at all **editors, webmasters** and **other developers** who wish to **understand the functionality of phpMySport** in order to **personalize**, make modifications to, or add supplementary functionality to it. It allows you to find out the **development methods** chosen, to acquire the concepts used and to understand the specifics of the software. You will also have **all the keys** at hand to understand the structure of the program and adapt it to your needs. But having all this choice, do you know what phpMySport is?

## PhpMySport

**PhpMySport** is a web application designed for **sports clubs and sporting leagues** that wish to easily create their **internet site**. As well as being complete and easy to use, it can be adapted for most team sports: football, rugby, volleyball, basketball, hockey, water polo, etc.

It allows for the management of **players, matches, team composition, championships**, playing **seasons** and much more besides. The application also provides for **news management**, a section of freely modifiable **web pages** and a discussion **forum**.

**Written in PHP** and coupled with a **MySQL** database, phpMySport provides a «Content Management Systems» or **CMS**; which is a content management system allowing **easy creation and management of a web site**. It has specific sporting differences but still contains the main aspects of a CMS: **personalization** of design, a **WYSIWIG text editor**, simple and intuitive management, **multi-user management**, etc.

In other words, phpMySport provides a complete web site which is adaptable to your sport, even in the colors of your team or league!

Here are the main functions provided by phpMySport, ordered by version:

### Version 1.0

Version 1.0 of phpMySport includes the following functionality:

- News management
- Management of information web pages
- Management of team members, players, trainers and managers
- Management of clubs and venues
- Management of teams and their composition
- Management of matches, playing seasons and competitions
- Discussion forums
- Members area
- Administration area

### Version 1.1

Provides the following new functionality:

- Plugin management
- Adding a plugin: Management of web links

### Version 1.2

Provides the following new functionality:

- Ordering by points and team statistics
- Player statistics
- Improvements to competition management: tour management
- Management of files and images
- Introduction of Italian and Portuguese versions

## Before beginning

Before beginning, you will need to **download the latest version** of the application and **install it** following the instructions in the user manual. If this is your first installation, then take a moment to work through the application and find out the different functionalities. This will greatly assist you in easily understanding the way it is designed.

Have you mastered the application but you now wish to make modifications to it? Then this manual is made for you. It will explain how to create a new design, how to develop a plugin or even how the database is structured.

Do not worry, it is not necessary to be an expert to understand the functionality of phpMySport. None the less, it is preferable that you have at least some understanding of PHP/MySQL, HTML and JavaScript, languages with which phpMySport has been developed. The words «variable» and «database» are not new to you? Then begin reading this guide and discover all the secrets of phpMySport...

Happy reading!

## Objectives, constraints and methods

### Objective

The objective of PhpMySport is to allow for the creation and management of a web site for a team-sport league or club. This web application must be easy to use and at the same time be sufficiently feature rich to manage all the site content. It needs to be suitably intuitive for a competent novice to use but must also allow sufficient freedom for a committed webmaster who wishes to personalize the site.

### Constraints

PhpMySport is aimed at users whose requirements, tastes and needs are diverse. The software therefore needs to take account of these usage differences to provide a large number of requirements. If phpMySport is to satisfy users, it will need to be adaptable to existing environments and to databases currently in use. This will be needed for easy development elsewhere.

Faced with these needs, many constraints will need to be defined. For each of the entries, a solution is considered and chosen that leads to powerful software. We do not detail the choices made here; we simply detail them.

Here is a list of principal constraints and the solutions adopted:

Constraint	Chosen solution
The application needs to be editable for many languages	Use of language files (See the Language files section)
The design of the application needs to be easily modifiable	Use of a template system and CSS styles (See the Design and templates section)
The application needs to be easy to use and manage	Integration of the administration area (back office) with the general interface (front office). Use of explanatory icons and a 'What You See Is What You Get' (WYSIWIG) text editor.
The pages need to be optimized for reference by search engines	Use of valid XHTML and CSS. Use of URL rewriting (See the URL rewriting: easily-understood URLs section)
The application needs to be usable with different types of database	Use of generic functionality and separation of SQL requests into specific files (See the Generic functions for database integrations section)
The application needs to be deployable on most web servers	Use of the PHP language, in version 4.0
The software needs to be easily installed and updated	Integration of an installation and update assistant

## Design methods

Reliable and secure software cannot be developed without a solid and well thought-out concept. PhpMySport does not break this rule and is created following well known and frequently used software design methods: the MVC model for development and the Merise model for modeling the database.

The «**Model View Controller**» (**MVC**) model allows the separation of design layers:

- The model for accessing the database
- The view for displaying the information
- The controller is the intermediate step for selecting and processing the data

This separation makes the code very readable and easy for team working. For example, the web designers can create the graphical interface for the site while the developers can busy themselves writing the PHP code for processing the information to display.

Although often associated with Object Orientated Programming (OOP), the phpMySport MVC model does not use classes or objects. OOP is in effect largely an abstract notion among web designers and perhaps not correctly appreciated other than by experimental developers. PhpMySport is to be open and easy to modify, even by novices. As a consequence an OOP developer is not required in order to understand this design.

The **Merise method** allows the application to process data into models and to define relationships that exist between them. For example one could say according to the sporting season, a player might belong to one or more clubs. This method consists of different steps that result in the creation of the database.

A schema representing the structure of the phpMySport v1.1 databases available in the appendix (note: at present, the discussion forum tables are not shown).

## Languages Used

PhpMySport uses no less than **5 languages** which all have their own characteristics. Here is a quick overview of the languages present in phpMySport.

### MySQL

MySQL is a database management system. It allows the creation and storage of data. The information is held in tables. The insertion, update, deletion and selection of data is possible using SQL requests. PhpMySport uses a MySQL database to hold information for the site.



### PHP

PHP is a programming language, currently widely-used for the creation of web sites like phpMySport. It allows calculations to be performed, the retrieval of form values and even interacting with a database. PHP also has features that allow rendering of HTML code. However, in phpMySport, the PHP and HTML code are kept separate. The application uses in effect a template system which will be detailed further on.



## HTML

The HTML or XHTML language is used to display the information transmitted. It is a language which is understood by your web browser: it will decode and interpret the downloaded pages that you see on the screen.

## CSS

The CSS language is used to format the downloaded information. It is complementary to HTML. It is also a widely-used language for web applications.

## JavaScript

JavaScript is a language that can alter HTML code. For example it is used to verify the information entered in a form or to display interactive menus. It is one of the components of AJAX technology. PhpMySport uses it principally to display form information pleasingly.

# Organization of the software

## Software architecture

PhpMySport is constructed according to its own architecture, but relies on methods widely used among the software designers.

The application is constructed around a core or **skeleton**, upon which the different modules or **organs** live. Now these organs each perform a precise function, for example «Posting and management of news items». The skeleton has no real function except to allow the organs to communicate with each other. It combines the functionality of all the organs. Without the organs, the skeleton is useless, and the organs cannot function without the skeleton.

### The skeleton

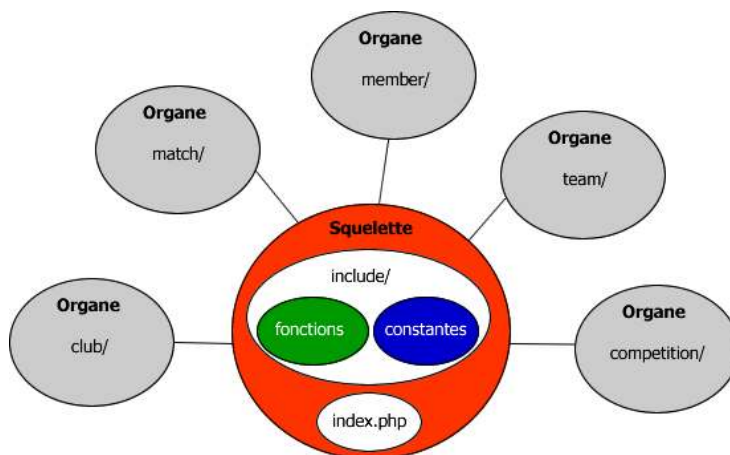
The skeleton is the central component in phpMySport. It communicates with all the organs and aggregates the functionality. The skeleton is composed of:

- The «**index.php**» **file** which is at the same time the brain and the heart of the application. This is what determines the page requested by the user, which retrieves the information and which sends it. It also controls the access to pages reserved for webmasters.
- The «**include/**» **folder** which provides the vital functions of the application. The folder contains all the information communicated to the organs, such as database access functions, form processing functions, the text used in the general interface (language files) as well as the communication constraints (configuration file). The list of communication elements is detailed in the Common constants and functions that may be used section.

### The organs

An organ corresponds to a precise function. For example, the «club» organ allows the management of sports clubs. For this, the «club» organ uses the skeleton functions to communicate with the database and retrieves the information on registered clubs.

Each organ is composed of many files contained within the same directory. The functionality of the organ is provided by the code contained in these files. For example the «club» organ will have many files which allow it to add, modify, remove and display clubs, which is everything that is required for the function.



## File structure

In version 1.2, phpMySport comprises 17 directories. 11 of these correspond to organs:

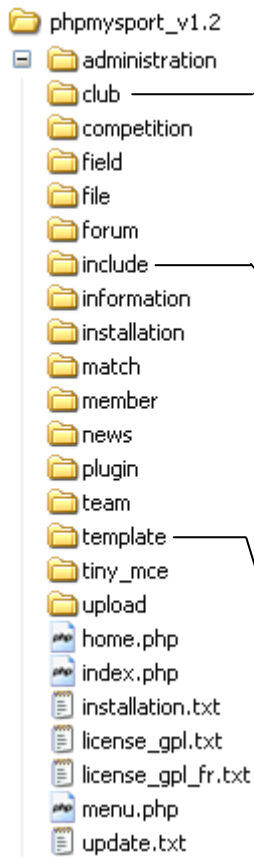
administration:	management and configuration of the site
club:	management of sports clubs
competition:	management of competitions, championships and playing seasons
field:	management of match venues (fields, halls, etc.)
file:	management of files and images uploaded to the server
forum:	management of the discussion forum
information:	management of information web pages
match:	management of matches and match statistics
member:	management of members, players and managers
news:	management of news articles
team:	management of teams and team makeup

The 6 remaining directories are:

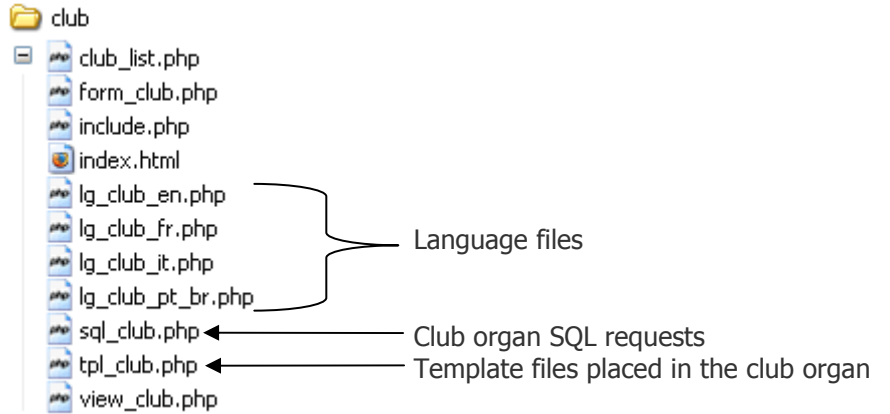
include:	contains the configuration files and the functions (skeleton)
installation:	helps with installation and updates; for security reasons, these files will be removed at the end of the installation
plugin:	directory to which plugins are to be copied
template:	directory in which templates are placed
tiny_mce:	HTML text editor used by forms
upload:	directory to which files and images are uploaded (via file management)

Here is a global representation of the file structure with details of the «include», «template» and «club» directories:

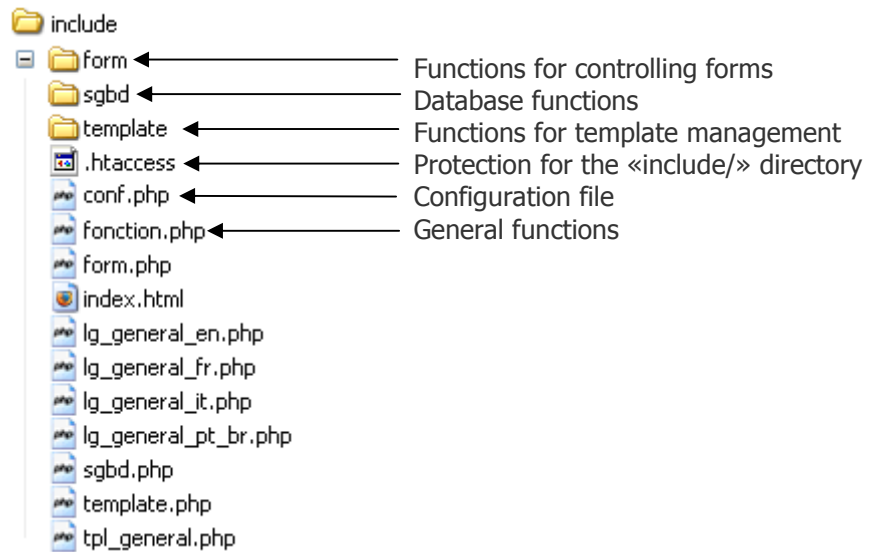
**General structure**



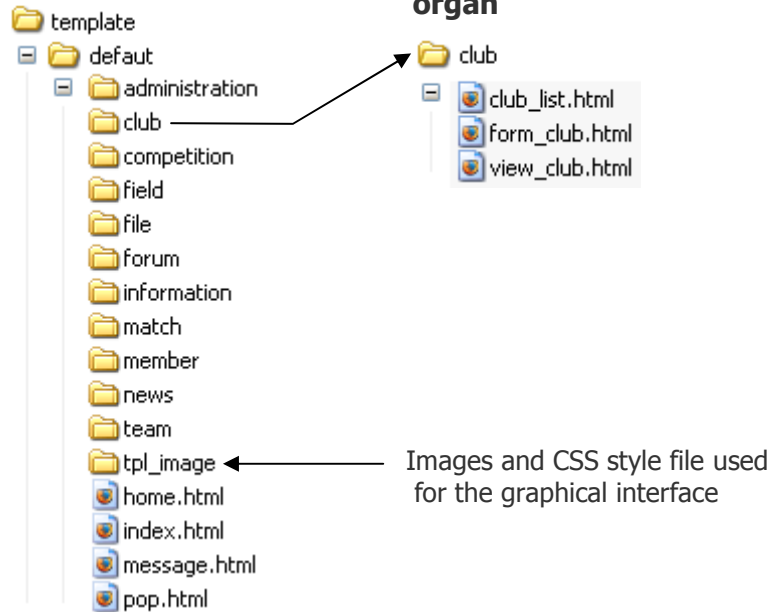
**«club» organ**



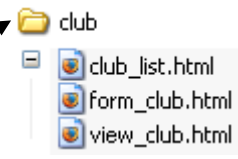
**Skeleton**



**Template**



**Template for the «club» organ**



Images and CSS style file used for the graphical interface

## Principal techniques used

As we have seen in the Objectives, constraints and methods section, the constraints imposed for the creation of phpMySport required the use of adapted technical solutions. The following details three of these: language files, generic database access functions and complex URL rewriting.

### Language files

What is the best way that phpMySport can be easily edited in multiple languages? In other words, in what way can the interface text files be made easy to translate?

The solution chosen is the use of language files. These are PHP files that contain the series of words and phrases used in general interface of the application. The names of the files respect the following convention:

«lg\_XXXX\_yy.php» where:

- «lg» indicates that it is a language file,
- «XXXX» is the name of the directory that the file relates to
- «yy» is the two-letter abbreviation of the translation language

For example, «lg\_match\_fr.php» is the language file for the match organ/directory translated into French (fr).

Each file contains the \$lang variable which is in fact an associative table holding the values used. Here is an example:

```
$lang['match']['match']='Match';  
$lang['match']['match_list']='Listes des matchs';  
$lang['match']['add_match']='Ajouter un match';
```

Here, three language variables have been defined for the «match» organ. In this example, their values correspond to text in French.

The «lg\_match\_en.php» file contains each of the same variables but with their values in English («en» for English):

```
$lang['match']['match']='Match';  
$lang['match']['match_list']='Matches list';  
$lang['match']['add_match']='Add a match';
```

As a result of the language files chosen and defined at the time of installation, the corresponding language files will be applied to the application.

## Generic functions for database integrations

PhpMySport can be installed on different platform types and is capable of communicating with many types of database. The application can be easily adapted to MySQL, PostgreSQL or Oracle. To permit this requirement, generic database management functions are implemented. They are used in the application suite. For example, the `sql_connect()` function allows connection to the database. Whether connected to a MySQL or Oracle database, according to the application configuration, this function connects to the database. The list of generic functions is presented in the Database Functions section.

*Note: in version 1.2, only functions for MySQL are available. Future versions of the application should be compatible with other databases.*

Throughout, the database table names are held as PHP constants. If needed, it is possible to rename them. It is sufficient to simply modify the name in the constants for all the requests using the tables to be modified. The constants are defined in the `«/include/sqldb/sqldb_tables.php»` file. Here is an example that shows how the matches table is defined:

```
define("T_match", "`match`");
```

Finally, the SQL code is separated from the PHP code. In the same way as with the language files, the SQL files contain the sets of requests used by the database module. The `«sql_match.php»` file for example contains the requests used for inserting, modifying, removing and selecting matches. Each file contains the `$sql` variable. Here is an example:

```
$sql['match']['select_match']="SELECT * FROM ".T_match." ";
```

This shows the request allowing selection of all the matches in the `«match»` table. The `«T_match»` constant corresponds to the name of the match table.

## URL rewriting: easily-understood URLs

As with all web applications, navigation to phpMySport pages is made by way of internet links (hyperlinks). In phpMySport, these links consist of information that indicates which page in the site the application should display. The information passed for navigation to the page is based on the following convention:

<b>r</b>	root of the site	<i>this is the module to be called</i>
<b>v1</b>	first variable	<i>this is the page to be called</i>
<b>v2</b>	second variable	<i>other variables used by the page called</i>

Example:

```
index.php?r=match&v1=match_list
```

In this example, it requests the download of the match list (`v1=match_list`) from the match module (`r=match`).

These URLs pose some problems: they are difficult to remember by users and are not easily understood by search engines. Therefore, phpMySport has adopted a technique for rewriting the URLs (URL rewriting) which allows simplification of complex web addresses. This technique is not possible in the application itself and is installed on an

Apache server which has mod «Rewrite» activated. When installing phpMySport, the user is asked whether this option is required. If so, the application creates a file named .htaccess which allows Apache servers to understand simplified addresses.

During development, all the links must respect the convention and will be constructed with a function that allows conversion if the application is configured for URL rewriting. This function is:

```
convert_url($url)
```

Here is an example of its use:

```
convert_url("index.php?r=match&v1=match_list");
```

returns:

```
/match/match_list.html
```

## Design and templates

### What is a template?

In phpMySport, the PHP development is separated from the HTML design. Thus, the application can be personalized according to the colors and layout desired by the user. This is possible thanks to a «template» system. There are a set of files containing PHP code and other «template» files which are related to HTML pages containing template code. Having loaded a web page, the application then reads the HTML files and replaces the template code with the corresponding PHP variable values.

The different designs are situated in the «template» directory. Each design is composed of a directory of HTML files, ordered by module. An initial design is provided with phpMySport, named the «default» template.

Here are the details of the template code.

### The template code

Three types of information can be displayed on a page: static information like text, cyclic information as in lists and option information.

Three template codes are thus required. Each is associated with a PHP variable type: text (string) for static data, tables (array) for lists and a boolean of options.

The gathering of information for display is held in a single and unique variable: \$page. This is a two-dimensional table that contains the set of text, tables and booleans needed by a page.

A PHP function is used in the «index.php» file to set this \$page variable and the template file defined to create the final HTML code seen by the user.

#### Basic code

The basic code is used to display a PHP variable of type «text».

The name of the variable is placed within a pair of «{» and «}» symbols.

Template code (HTML file)	PHP code (PHP file)	Displayed on the screen
{my_text}	\$page['my_text'] = "My text"	My text

#### List code

Lists are used to display a PHP variable of «array» type. The key word «LOOP» is used to signify that a variable corresponds to a set. All the HTML code held within «LOOP» and «END LOOP» will be repeated as many times as the table has lines.

Template code	PHP code	Displayed on the screen
<pre>&lt;!-- LOOP my_list --&gt; &lt;p&gt;{name} {firstname}&lt;/p&gt; &lt;!-- END LOOP my_list --&gt;</pre>	<pre>\$page['my_list']=array( array('name'=&gt;"Durant", 'firstn ame'=&gt;"Martin"), array('name'=&gt;"Dupond", 'firstn ame'=&gt;"Marie"));</pre>	<p>Durant Martin Dupond Marie</p>

### Option code

The option code is used for boolean variables or more precisely «true / false» values. If the variable is empty, then the HTML code within the two «OPTION» tags will not be visible. Otherwise it will be displayed.

Template code	PHP code	Displayed on the screen
<pre>&lt;!-- OPTION var1 --&gt; &lt;p&gt;{text1}&lt;/p&gt; &lt;!-- FIN OPTION var1 --&gt;  &lt;!-- OPTION var2 --&gt; &lt;p&gt;{text2}&lt;/p&gt; &lt;!-- END OPTION var2 --&gt;</pre>	<pre>\$page['text1']="Text 1"; \$page['text2']="Text 2"; \$page['var1']=""; \$page['var2']="1";</pre>	<p>Text 2</p>

## Creating your own design

To personalize the design of phpMySport and display it in your club’s colors, you will need to create your own template. This is relatively quick, but it is preferable to already have an understanding of the CSS and HTML languages. Here are the different steps to follow:

- Open the repository where the phpMySport files are held (with the aid of you FTP client if necessary).
- Open the «template» repository situated in the root of your site. A subdirectory named «défaut» will be present. The latter contains the default phpMySport design. **To avoid any risk of damaging the scripts, it is strongly recommended that you do not delete or modify this directory or the files that it contains!**
- Copy and paste the «default» directory. Rename it (text without spaces) to a name of your choice (eg: «mondesign») and place this in the «template» directory. From now on, this will contain two subdirectories: «default» and «mondesign».
- Open the directory that you have just copied. There you will find many folders as well as a number of HTML files. Rest assured, it is not necessary to modify all of these to change the design of your site! Only two elements are important:
  - The «/tpl\_image/» directory: this contains all the images which comprise the graphical interface for the site.

- The «`/tpl_image/styles.css`» file: this contains all the information to format and display page text, menus, headings, tables, etc
- To modify the design, you will need to integrate your own images and modify the content of the CSS file.
- Here is an example if you wish to insert your own logo in place of the phpMySport one:
  - Place the image corresponding to your logo (in .jpg, .gif, .png or other format) in the «`/tpl_image/`» folder of the repository that you copied («`mondesign`»)
  - Open the «`/tpl_image/styles.css`» CSS file
  - Replace the following line:

```
div#header { height:90px; width:100%; margin:0 auto; padding:0; }
```
  - With:

```
div#header { height:90px; width:100%; margin:0 auto; padding:0; background:url(mon_logo.jpg) no-repeat; } #header img { display:none };
```
  - The «`mon_logo.jpg`» file corresponds with the image that you have placed in the «`/tpl_image/`» folder. If necessary, modify the height «`90px`» as required for your image.
- You can of course personalize the other elements of the interface: menu, title, page footer, font colors, height and color of text, titles, etc.
- To change the design from the default and apply the modification that you have made:
  - Go to the administration area, heading «`site configuration`»
  - In the corresponding list displayed, choose the design that you created and select the choice
  - The new design of your site will now display.

Please do not hesitate to make known your artistic qualities and share your graphic endeavors with other phpMySport users. To do this, please send your images and personalized CSS files to us!

## Personalising phpMySport with plugins

### What is a plugin?

A plugin is a programme that interacts with software, in this case phpMySport, so as to increase that software's functionality. By creating your own plugin, or by installing an existing plugin, you can considerably increase phpMySport's capabilities. For example, you can integrate photo galleries, RSS feeds or even go over the news management already included in phpMySport and make some improvements.

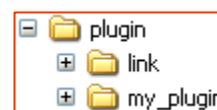
Do you wish to create your own plugin? If so, please read the following instructions carefully. After studying the file hierarchy of a plugin, we'll go step-by-step through the creation process before wrapping up with a concrete example.

### File hierarchy of a plugin

#### phpMySport's « plugin » directory

phpMySport has a directory « plugin/ » in which downloaded or created plugins must be placed.

In the figure we see the plugin directory tree containing 2 plugins labelled « link » and « my\_plugin ».



#### Structure of a plugin

A plugin comprises 3 file types.

The first are vital and constitute the heart of the plugin. They are involved with the comprehension, functioning and integration of the plugin within phpMySport. The second are optional and depend solely on the needs of the plugin. Lastly, the third type deals with introducing the new functionality proposed by the plugin.

#### Necessary files

- conf.php : plugin configuration file
- include.php : file that links the plugin and phpMySport
- install.php : plugin installation assistant
- lg\_xxxx\_yy.php : language file for plugin xxxx
- tpl\_xxxx.php : file listing the template pages used
- read\_me.txt : description of the plugin
- index.html : empty file to secure the plugin

We'll now go over in detail « conf.php », « install.php » and « include.php ».

#### The file « conf.php »

As indicated by its name, « conf.php » contains information on the plugin configuration. It also acts as a link between phpMySport and the plugin. The variables it contains are used to detect and integrate it into the software.

**Be careful** then and **do not modify the names of the variables used**, otherwise phpMySport will be unable to detect your plugin!

Here is the basic information that « conf.php » must contain:

Variable	Example value(s)	Description
<code>\$plugin_name</code>	My plugin	Plugin name
<code>\$plugin_idurl</code>	Myplugin	ID used for the URL. Must be a word with no spaces or special characters.
<code>\$plugin_root</code>	ROOT."/plugin/my_plugin"	Path to the plugin directory. Last character must not be a forward slash « / ».
<code>\$plugin_install</code>	0 or 1	Indicates whether the plugin has been installed (1) or not (0).
<code>\$plugin_active</code>	0 or 1	Indicates whether the plugin is active (1) or not (0).
<code>\$plugin_lang</code>	array("fr","en")	Array containing the list of languages into which the plugin has been translated. A language is indicated by a two-letter abbreviation.
<code>\$plugin_page_admin</code>	array("install","form")	Array containing the list of pages that are protected and reserved for the site administrator. Often this involves the installation file as well as forms for the addition and/or modification of data.

The file « install.php »

The file « install.php » is an installation file and is required for the easy installation of the plugin by future users. Most notably, it is used to:

- Execute SQL requests to modify the database, if necessary.
- Modify the plugin configuration file « conf.php » so as to activate it. During this stage, the variables `$plugin_install` and `$plugin_active` are set to value 1.

The file is accompanied by a template file, the latter producing a visual assistant for the user during plugin installation.

*Note: future versions will most likely require an uninstall file.*

The file « `include.php` »

The file « `include.php` » plays two roles:

- It makes use of the plugin common files: the language file, SQL requests, the list of template pages used and, if required, the functions and classes used.
- It allows inclusion of the `.php` file corresponding to the page called. The latter is stored in the variable `$_GET['v1']`.

Optional files

- `sql_xxxx.php` : all SQL requests used by the plugin
- `bdd.txt` : SQL requests necessary for plugin installation

Other files that introduce new functionality

There can be one or more of them. Their names are essentially unconstrained but should reflect the action they perform. For example, a file named « `item_list.php` » would correspond to a list of « `item` » elements.

Each such file is coupled to a template file of the same name but with extension `.html`. It is recommended that the template file and images used by the plugin be placed in a separate folder called « `tpl` ». The list of template files must be indexed in « `tpl_xxxx.php` » files (where `xxxx` corresponds to the plugin name).

## How does one create a new plugin?

Here are the various steps required to create a new plugin:

### Step 1

In phpMySport's « `plugin/` » folder, create a new directory with the name of your plugin. We will refer to this as the root folder of your plugin. In this new folder, create another folder called « `tpl` » where you will place the template pages of your plugin.

### Step 2

As described previously, create all the files necessary to the correct working of your plugin (see the section File hierarchy of a plugin). Put them in the root folder of your plugin.

### Step 3

If information has to be stored or archived, you may need to make use of phpMySport's database and make certain modifications. To this end, think about your data structure and create a file `.txt` containing the (those) SQL request(s) for table creation and/or modification. This file will be necessary for the plugin installation. Note that you may also make use of one or more XML files in place of the database.

### Step 4

One-by-one create those files that introduce the new functionality of your plugin. In these PHP files, it is strongly recommended to use phpMySport's variables, constants and functions. If needed, you can implement your own functions. These files should be sufficiently commented and respect phpMySport's development methodology. Remember to use the `convert_url()` function, the `$page` variable and to indicate the

template file via which they are associated. Don't forget to create the corresponding template files in the « tpl/ » folder, and to index and declare them in your page's code.

#### Step 5

So you're in the midst of implementing your plugin? You probably want to check the result of your code ... for that:

- Execute your SQL request to modify the database (if necessary)
- Activate your plugin by setting the « \$plugin\_install » and « \$plugin\_active » variables to 1.

Open your web browser and enter the URL of your phpMySport site. The name of your plugin will appear in the main menu of the software.

#### Step 6

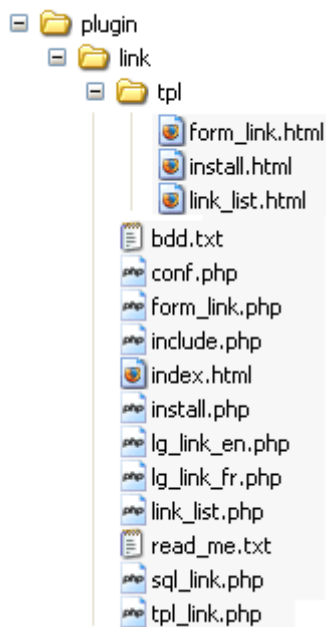
That's it; your plugin is finally working? All that remains is for you to make it known and to diffuse it so that other phpMySport users can install it. To that end, go to the official phpMySport site, then category « extension » where you'll find practical information on how to distribute your plugin.

## Example: web link management plugin

Have you understood the structure and steps to create a plugin? In that case, there's nothing like an example to put into practice that which you've learnt. We're going to examine how to create a simplified web link management: the « link » plugin.

#### File hierarchy

First off, let's have a look at the file hierarchy of our plugin :



We have created and placed a folder called « link » in the « plugin » directory of phpMySport. The former has a sub-directory called « tpl » and, within it, a set of files.

Within the « link » folder we can see our indispensable files, optional files and files that introduce our new functionality:

- « form\_link.php » : for adding or modifying a web link. Coupled to the template file called « form\_link.html », the latter being an HTML form.
- « list\_link.php » : for accessing the list of saved web links. Coupled to the template file « link\_list.html ».

## The Database

For our web link management we need to be able to store information on each web link: name, URL and description. To that end, we will use the database and create a new table. Here is the corresponding SQL request:

```
CREATE TABLE `link` (
  `link_id` int(11) NOT NULL auto_increment,      # Unique link identifier
  `link_name` text NOT NULL,                      # Web link name
  `link_url` text NOT NULL,                       # Link address (URL)
  `link_description` text NOT NULL,              # Description of the link
  PRIMARY KEY (`link_id`)
);
```

We'll place this request in the text file « bdd.txt ».

## The file « conf.php »

```
<?php
# plugin configuration
$plugin_name="Link";
$plugin_idurl="link";
$plugin_root=ROOT."/plugin/link";
$plugin_install="0";
$plugin_active="0";
$plugin_lang=array("fr","en");
$plugin_page_admin=array("install","form_link");
?>
```

We can see above that the variables « \$plugin\_install » and « \$plugin\_active » have value 0. If a user downloads this plugin, she could install and activate it through the administration area.

We've configured the plugin so that it's available in French (fr) and English (en). Indeed, looking at our plugin files we can see that there are two language files « lg\_link\_fr.php » and « lg\_link\_en.php ».

Access to the « install » installation pages and the form for adding a link, « form\_link », must be set to administrator-only. This information is enshrined in the array « \$plugin\_page\_admin » which ensures that these pages may not be opened by a simple web surfer.

## The file « include.php »

```

<?php
# Include those files required by the plugin
include_once(ROOT."/plugin/link/tpl_link.php");
include_once(ROOT."/plugin/link/lg_link_".LANG.".php");
include_once(ROOT."/plugin/link/sql_link.php");
include_once(ROOT."/plugin/link/conf.php");

# Check that the plugin is correctly installed
if($plugin_install!=1) {
    include(ROOT."/plugin/link/install.php");
}
else {
    # Display the list of links by default
    if(!isset($_GET['v1'])) {
        include(ROOT."/plugin/link/link_list.php");
    }
    else {
        # Depending on the page requested, deal with the various PHP files
        switch($_GET['v1']) {
            case "form_link" : include(ROOT."/plugin/link/form_link.php");
break;
            case "link_list" : include(ROOT."/plugin/link/link_list.php");
break;
            default : include(ROOT."/plugin/link/link_list.php");
        }
    }
}
?>

```

In the above, we include the template files, language files, SQL requests and plugin configuration. To this end we make use of phpMySport's ROOT and LANG constants.

We then look to see which of the plugin files must be invoked. Several possibilities:

- We wish to install the plugin: we invoke « install.php »
- We wish to add/modify a plugin: we invoke « form\_link.php »
- We wish to display the list of plugins: we invoke « link\_list.php »

### The file « install.php »

As the code in the installation file is quite long, we'll only describe how it works. You may however visualise the code as the file is, by default, provided with phpMySport.

Plugin installation is in 2 steps:

- Firstly a page introduces the plugin functionality to the user and a form requests that installation be confirmed or not.
- If she confirms, the installation procedure is launched: the new table « link » is created in the database and the file « conf.php » is updated with the fact that the plugin has been installed and is active.

The files « link\_list.php » and « link\_list.html »

The code in file « link\_list.php » allows us to collect the list of links stored in the database table « link ». It's also used to delete a link.

Here's a snippet showing the code for gathering the list of links:

```
# SQL request for selecting all links
$sql_link=sql_replace($sql['link']['select_link_condition'],$var);

$sgbd = sql_connect(); # database connection
$res_link = sql_query($sql_link); # execute the request
$nb_ligne=sql_num_rows($res_link); # count the number of results
if($nb_ligne=="0")
{
    # if length 0, display an error message
    $page['L_message']=$lang['link']['E_link_not_found'];
}
else
{
    # Accumulate one by one the results in variable $page['link']
    $i="0";
    while($ligne = sql_fetch_array($res_link))
    {
        $page['link'][$i]['id']=$ligne['link_id'];
        $page['link'][$i]['name']=$ligne['link_name'];
        $page['link'][$i]['url']=$ligne['link_url'];
        $page['link'][$i]['description']=$ligne['link_description'];
        $page['link'][$i]['L_show_view']=$lang['link']['show_view'];
        $i++;
    }
}
sql_free_result($res_link);
sql_close($sgbd); # close the database connection

# Page title
$page['L_title']=$lang['link']['link_list'];

# Corresponding template file
$page['template']=$tpl['link']['link_list'];
```

The list of links is stored in variable \$page['link']. The latter is an array of all the information concerning the links and is exploited by file « link\_list.html ».

Here's the HTML code of that file and its corresponding visual aspects:

HTML code of template « link_list.html »	Web browser appearance	Example with web links
<pre> &lt;h1&gt;{L_title}&lt;/h1&gt;  &lt;!-- LOOP link --&gt; &lt;div&gt; &lt;h2&gt;{name}&lt;/h2&gt; &lt;p&gt;{description} &lt;br /&gt; &lt;a href="{url}"&gt;{url}&lt;/a&gt; &lt;/p&gt; &lt;/div&gt; &lt;!-- END LOOP link --&gt;                     </pre>	<p><b>{L_title}</b></p> <p><b>{name}</b></p> <p>{description}</p> <p><a href="#">{url}</a></p>	<p><b><u>WEBSITES LIST</u></b></p> <p><b><u>My link</u></b></p> <p>Description of the link  <a href="http://www.mylink.com">http://www.mylink.com</a></p> <p><b><u>PhpMySport</u></b></p> <p>PhpMysport official website  <a href="http://phpmysport.sourceforge.net">http://phpmysport.sourceforge.net</a></p>

## Common constants and functions that may be used

### Constants

Here is a list of constants used by the software. They are defined in the configuration files « include/conf.php » and « index.php ».

Name of constant	Type	Example	Description
ROOT	text	/var/www/site	Root directory of the site. To be used for including PHP files or classes. Last character must not be a « / ».
ROOT_URL	text	http://www.mysite.com	Web address (URL) of the site. Last character must not be a « / ».
SITE_TITLE	text		Title of the site
CLUB	integer	0	Default club identifier (equal to 0 if phpMySport is configured in league mode)
SGBD_HOST	text	localhost	Address of the Apache server
SGBD_USER	text	root	Server user name
SGBD_PWD	text	password	Server password
SGBD_NAME	text	mybase	Database name
VERSION	text	1.1	Version of phpMySport used
MAX_FILE_SIZE	integer	2000000	Maximum size allowed (in octets) for files that are uploaded
URL_REWRITE	boolean	0	Activate (1) or not (0) rewriting of complex URLs
SENDER_EMAIL	text	contact@mysite.com	Email address for sending mail
SENDER_NAME	text	Martin	Name of expeditor
NB_MAX_PLAYER	integer	11	Maximum number of players per team
SITE_OPEN	text	1	Defines if the site is open (1) or under construction (0)
LANG	text	en	Defines site language
MEMBER	boolean	1	Defines if the user is a connected member (1) or not (0)
MEMBER_ID	integer	12	Connected member's identifier.

			If it's just a guest visitor, then it takes value 0.
ADMIN	boolean	1	Defines if the user has administrator status (1) or not (0)

## Functions

Here is the list of functions and procedures defined in phpMySport.

### General functions

The following functions are defined in the file « include/fonction.php »

#### **convert\_url(\$s,\$c)**

Description: converts a PHP URL to an HTML URL

Params: \$s: the URL to convert, \$c: a boolean. When set to 0, prevents rewriting.

Returns: the converted URL

Example:

```
convert_url("index.php?&r=member&v1=sign_up");
// Returns: /member/sign_up.html
```

#### **convert\_date\_sql(\$date)**

Description: converts a date to the SQL format yyyy-mm-dd

Params: \$date: date in format dd-mm-yyyy

Returns: date in format yyyy-mm-dd

Example:

```
convert_date_sql("08-07-2007");
// Returns: 2007-07-08
```

#### **date\_day(\$format)**

Description: returns today's date in the indicated format

Params: \$format is a date format (see PHP's date() function)

Returns: today's date, formatted

#### **generate\_pagination(\$url, \$nb\_page,\$page\_num,[\$end\_url])**

Description: generates a page listing

Params: \$url: URL of the PHP page  
 \$nb\_page: total number of pages  
 \$nb\_max: number of elements per page  
 \$page\_num: number of the current page  
 \$end\_url: any end of URL variables

Returns: an array containing a list of pages matched to their URL

#### **html2txt(\$text)**

Description: converts a character chain to a raw text containing no HTML tags or punctuation. White space is preserved.

Params: \$text is the character chain to convert to raw text

Returns: raw text without HTML code or punctuation

Example:

```
html2txt("<img src=image.jpg> Here is an image !");
```

```
// Returns: here is an image
```

### **unhtmlentities(\$chaineHtml)**

Description: replaces HTML characters with their text equivalents

### **text\_replace(\$text,\$var)**

Description: replaces the element {xxx} in \$text by the value of the corresponding variable \$var['x']

Params: \$text is the character chain containing the elements to replace  
\$var is an array containing the replacement values

Returns: the text with elements replaced

Example:

```
$var["example"]="Example of text";
text_replace("Here is the code to replace : {example}",$var);
// Returns: Here is the code to replace: Example of text
```

### **text\_tronquer(\$text,\$nb[\$b])**

Description: truncates a text after the given number of characters

Params: \$text is the text to truncate  
\$nb is the number of characters at which to truncate  
If \$b is 1, word slicing is forced. By default \$b is 0.

Returns: the truncated text

Example:

```
$text="Example of text too long to truncate";
text_tronquer($text,20);
// Returns: Example of text too...
```

### **return\_bytes(\$val)**

Description: converts file size to octets

Params: a number in GB, MB or KB

Returns: number in octets

Example:

```
return_bytes("2 mo");
// Returns: 2048
```

### **filesize\_format (\$bytes, [\$decimal, \$spacer, \$lowercase])**

Description: modifies the display format of a file size

### **ecartDate(\$date1,\$date2)**

Description: returns the separation in seconds between two dates whose format is yyyy-mm-dd hh:mm:ss

## Database Functions

The following functions are defined in « include/sghd/mysql/mysql\_fonctions.php ».

### **sql\_replace(\$sql,\$var)**

Description: identical to the function text\_replace()

### **sql\_connect()**

Description: connects to the server and selects the database

Params: none

Returns: connection identifier

**sql\_close(\$id\_connection, \$result = false)**

Description: closes the database and SQL server connections

Params: \$id\_connection is the connection identifier to close

Returns: none

**sql\_query(\$sql, \$result = false)**

Description: executes an SQL request

Params: \$sql is the SQL request to execute

Returns:

**sql\_num\_rows(\$result)**

Description: counts the number of lines resulting from a request

Params: \$result is the result returned by sql\_query()

Returns: number of results

**sql\_result(\$result)**

Description: returns the result of a request

**sql\_fetch\_array(\$result)**

Description: returns an array containing the values of a row

**sql\_insert\_id(\$sgbd)**

Description: returns the identifier of the last row inserted

**sql\_free\_result(\$result)**

Description: frees the results of a request

**sql\_error()**

Description: returns an error message in case of a problem with an SQL request

### Functions dealing with form data

The following functions are defined in « include/form/fonctions.php »

**format\_txt(\$txt)**

Description: formats the text by replacing special characters with their HTML equivalent

**check\_text(\$text)**

Description: checks that the text contains no numbers

**check\_integer(\$nb)**

Description: checks that the number contains no text characters

**check\_email(\$email)**

Description: checks that the email syntax is correct

**check\_date(\$date)**

Description: checks that \$date is in one of the formats: jj-mm-yyyy, jj/mm/yyyy, jj.mm.yyyy or yyyy/mm/jj

**check\_hour(\$hour)**

Description: checks that the time is of the form hh:mm:ss

**check\_login(\$text)**

Description: checks the login format (no white space, no accents, between 4 and 10 characters)

**check\_file\_name(\$text)**

Description: checks file name format (no white space, no accents, between 1 and 100 characters)

**check\_url(\$url)**

Description: checks that the URL syntax is correct

**check\_formula(\$formula)**

Description: checks that the syntax of a (statistical) formula is correct

### Template Functions

The following functions are defined in « include/template/fonctions.php »

**parse\_template(\$tpl,\$var)**

Description: grabs the content of a template file and replaces the template code elements with their corresponding values

**parse\_html(\$code,\$var)**

Description: replaces the template code elements with their corresponding values

## Programming Conventions and Rules

These rules are, for the most part, taken from the « phpBB Coding Standard Guidelines » proposed by the phpbb group, creators of the famous forum.

### Naming Conventions

English is the language used for the software. It is recommended to use it for file names, variables, constants and functions.

#### Variables

Are lower case, words separated by an underscore «\_», using alphanumeric characters (without accents). Numbers are permitted, but discouraged. Short names are preferred but should remain sufficiently comprehensible.

Incorrect:     `$membername, $MemberName, $name_of_member`

Correct:       `$member_name`

#### Constants

Same convention as for variables but using upper case.

Incorrect:     `define("Site_url");`

Correct:       `define("SITE_URL");`

#### Loop indices

This is the only situation for which variables can take a character: `$i`, `$j` or `$k`

#### Functions: name and parameters

Same convention as for variables.

## Code Structure

#### Indentation and white space

Code indentation should use spaces and not tabulations. No unnecessary white space: use spaces between the different entities, but not between common structures:

Incorrect:     `if ( $tmp == 1 && $k != true )`

Correct:       `if($tmp==1 && $k!=true)`

#### Curly Brackets

Always use curly brackets, even if they are not obligatory to obtain a working script. They should be positioned as follows:

##### Incorrect

```
if (condition)    action();
if (condition)
    action();
while (condition)
    action();
```

##### Correct

```
if (condition)
{
    action();
}
while (condition)
```

```

for ($i = 0; $i < size; $i++)
    action($i);

```

```

{
    action();
}
for ($i = 0; $i < size; $i++)
{
    action();
}

```

## Other Rules

### Use single quotes

*Incorrect:*

```

$member_name="Name of member ";
action("$member_name");
$_POST[member_name];

```

*Correct:*

```

$member_name='Name of member';
action($member_name);
$_POST['member_name'];

```

### Concatenate text and variables

*Incorrect:*    \$text="Start of \$variable and end of text";

*Correct:*     \$text="Start of ".\$variable." and end of text ";

### Do not use non-initialised variables

*Incorrect:*    if(\$nom\_membre)

*Correct:*      if(isset(\$nom\_membre))

### Comment your code

Code is easier to understand if it is correctly commented. Users who wish to examine phpMySport's code will surely have a certain knowledge of PHP, or at least of computing. It is thus not useful to over-comment the code, explaining everything point-by-point. On the other hand, the general workings of the programme, as well as the meaning of new variables, must be correctly explained. Comments must be placed between « /\* » and « \*/ » or after the character « # ».

## Credits

### Authors

The phpMySport software was conceived by Jérôme PLACE, bio-informatician by training and a sport amateur. He is also webmaster of the site Cité Sport ([www.citesport.com](http://www.citesport.com)) which aims to promote amateur sports clubs, and sport in general. In creating this software, he hopes to enable all sporting associations and sport lovers to profit from his experience of web site creation.

### Version

<b>Software Version:</b>	1.2
<b>Software release date:</b>	March 8 <sup>th</sup> , 2007
<b>Manual published:</b>	November 20 <sup>th</sup> , 2007
<b>Manual written by:</b>	Jérôme PLACE
<b>Manual translation:</b>	Harry Glasgow, Brinick Simmons

# Appendices

## Physical model of the database (phpMySport v1.1)

