phpMyDataGrid Professional Evolution - Documentation

Installation

Include file:

Function name:

require_once('path');

Parameters:

path : classes/phpMyDataGrid.class.php

Path to class

Example of use:

```
require_once('classes/phpMyDataGrid.class.php');
```

Define class object container:

Function name:

datagrid('scriptName','gridID');

Parameters:

scriptName : sample.php

File name

gridID : 1

Grid ID (You would use a diferent code for each DataGrid if you like to have several grids on the same page)

Example of use:

```
$objGrid = new datagrid('sample.php','1');
```

Properties

ButtonWidth

Description:

Specifies the width of the icons used as buttons on phpMyDataGrid.

Allowed values:

ButtonWidth : 25

Integer.

Default: 25

EXAMPLE OF USE:

```
$objGrid -> ButtonWidth = '25';
```

•

backtick

DESCRIPTION:

Some databases like MySQL allow to enclose the name of the fields, table names or database names, by using a special character, for example ` the character

ALLOWED VALUES:

backtick : `

Character.

Default: `

EXAMPLE OF USE:

```
$objGrid -> backtick = '`';
```

•

liquidTable

DESCRIPTION:

Defines whether or not phpMyDataGrid will fit automatically the width available on the screen.

ALLOWED VALUES:

liquidTable : false

Boolean.

Default: false

EXAMPLE OF USE:

```
$objGrid -> liquidTable = false;
```

•

width

DESCRIPTION:

It complements the property liquidTable defining the percentage of which will fit the screen automatically.

ALLOWED VALUES:

width : 100%

String, Defining a percentage.

Default: 100%

EXAMPLE OF USE:

```
$objGrid -> width = '100%';
```

•

condition

DESCRIPTION:

It is now possible to condition the style of certain records giving them another output format, as easy as define the conditions that must be fulfilled.

ALLOWED VALUES:

condition : ['activo']==1

String, conditional statement.

EXAMPLE OF USE:

```
$objGrid -> condition = "['activo']==1";
```

ATENTION: It is possible to use any field used within the grid, simply type the field name between [' ... ']
NOTE: phpMyDataGrid evolution, allows to you to use this property in order to define a CSS style, as well as you can use the **addRowStyle** method which allows to define a set of conditions in order to get as result a CSS class to format the row output

•

conditionalStyle

DESCRIPTION:

This property complements the use of condition property, in this property are defined the styles to display the records that meet the specified condition.

ALLOWED VALUES:

conditionalStyle: style='background:#600;color:#FFF;'

CSS style list

EXAMPLE OF USE:

```
$objGrid -> conditionalStyle = "style='background:#600;color:#FFF;'";
```

ATENTION: If you use the method **addRowStyle** to set conditions, it is not necessary to use this property

•

conditionEdit

DESCRIPTION:

If you want to limit the editing records only those who meet certain conditions, use the following property to define the condition that must be fulfilled.

ALLOWED VALUES:

conditionEdit : ['activo']==1

String, conditional statement.

EXAMPLE OF USE:

```
$objGrid -> conditionEdit = "['activo']==1";
```

ATENTION: It is possible to use any field used within the grid, simply type the field name between [' ... ']

conditionDelete

DESCRIPTION:

If you want to limit the deleting records only those who meet certain conditions, use the following property to define the condition that must be fulfilled.

ALLOWED VALUES:

conditionDelete: ['activo']==1

EXAMPLE OF USE:

```
$objGrid -> conditionDelete = "['activo']==1";
```

ATENTION: It is possible to use any field used within the grid, simply type the field name between [' ... ']

moneySign

DESCRIPTION:

Define the sign to display in Money field types

ALLOWED VALUES:

moneySign : $

1 character.

Default: $

EXAMPLE OF USE:

```
$objGrid -> moneySign = '$';
```

•

zebraLines

DESCRIPTION:

Defenes the way in which the interlined will be displayed

ALLOWED VALUES:

zebraLines : 1

Integer

Default: 1

EXAMPLE OF USE:

```
$objGrid -> zebraLines = '1';
```

•

showToOf

DESCRIPTION:

This property defines if the message **Displaying records X to Y** is displayed or not

ALLOWED VALUES:

showToOf : true

Boolean.

Default: true

EXAMPLE OF USE:

```
$objGrid -> showToOf = true;
```

•

AllowChangeNumRows

DESCRIPTION:

phpMyDataGrid allows to end user to modify the amount of records per page to display, if you don't like to allow the user to do that, set this property to false

ALLOWED VALUES:

AllowChangeNumRows: true

Boolean.

Default: true

EXAMPLE OF USE:

```
$objGrid -> AllowChangeNumRows = true;
```

•

charset

DESCRIPTION:

Allow to define the page coding in which the page will be rendered.

ALLOWED VALUES:

charset : UTF-8

String

Default: ISO-8859-1

EXAMPLE OF USE:

```
$objGrid -> charset = 'UTF-8';
```

•

sqlDataCoding

DESCRIPTION:

Defines the database encoding

ALLOWED VALUES:

sqlDataCoding: SET NAMES 'utf8'

String

EXAMPLE OF USE:

```
$objGrid -> sqlDataCoding = "SET NAMES 'utf8'";
```

•

sqlcharset

DESCRIPTION:

Defines the encoding to use by MySQL

ALLOWED VALUES:

sqlcharset : utf8

String

utf8, big5, latin1, etc

EXAMPLE OF USE:

```
$objGrid -> sqlcharset = "utf8";
```

ATENTION: It is important to define this property as a complement to **sqlDataCoding** for a right working

•

defaultdateformat

DESCRIPTION:

Defines the default format to date type fields

ALLOWED VALUES:

defaultdateformat: ymd

Combination of dmy characters

Default: dmy

EXAMPLE OF USE:

```
$objGrid -> defaultdateformat = 'ymd';
```

•

defaultdateseparator

DESCRIPTION:

Defines the date separator char.

ALLOWED VALUES:

defaultdateseparator: -

1 Character.

Default: /

EXAMPLE OF USE:

```
$objGrid -> defaultdateseparator = '-';
```

•

csvSeparator

DESCRIPTION:

Character used to separate the values exported in CSV (Comma Separated Values)

ALLOWED VALUES:

csvSeparator : ;

1 character

Default: ;

EXAMPLE OF USE:

```
$objGrid -> csvSeparator = ';';
```

uploadDirectory

DESCRIPTION:

This property is used to define the folder where images will be uploaded

ALLOWED VALUES:

uploadDirectory: /images/upload/

path to images folder, it may be relative or absolute

Default: /

EXAMPLE OF USE:

```
$objGrid -> uploadDirectory = '/images/upload/';
```

ATENTION: Specified path must have write permissions.

show404image

DESCRIPTION:

Defines if phpMyDataGrid would display a 404 image when the related image is not found.

ALLOWED VALUES:

show404image: false

Boolean.

Default: false

EXAMPLE OF USE:

```
$objGrid -> show404image = false;
```

retcode

DESCRIPTION:

By default, phpMyDataGrid is rendered at the end of the script execution, but if this property is changed to true, then the HTML code will be returned as value to be stored in a variable.

ALLOWED VALUES:

retcode : false

Boolean

Default: false

EXAMPLE OF USE:

```
$objGrid -> retcode = false;
```

•

getMyOwnButtons

DESCRIPTION:

It allows the developer to control the position of the buttons New / Search / Export, by not displaying them directly on the grid, but allows to be positioned in any section of your website.

ALLOWED VALUES:

getMyOwnButtons: false

Boolean.

Default: false

EXAMPLE OF USE:

```
$objGrid -> getMyOwnButtons = false;
```

ATENTION: Note: To function, property **retcode** must be set to **true**.

•

strAddBtn:

FUNCTION NAME:

This property complements **getMyOwnButtons** property, returning the HTML for add button

EXAMPLE OF USE:

```
$strReturn = $objGrid -> strAddBtn;
```

•

strSearchBtn:

FUNCTION NAME:

This property complements **getMyOwnButtons** property, returning the HTML for search button

EXAMPLE OF USE:

```
$strReturn = $objGrid -> strSearchBtn;
```

strExportBtn:

FUNCTION NAME:

This property complements **getMyOwnButtons** property, returning the HTML for export button

EXAMPLE OF USE:

```
$strReturn = $objGrid -> strExportBtn;
```

cssPrinter

DESCRIPTION:

Define the CSS styles filename that will be used for printing.

ALLOWED VALUES:

cssPrinter : /css/my_css_file.css

Path and CSS filename

Default: css/b-w-print.css

EXAMPLE OF USE:

```
$objGrid -> cssPrinter = '/css/my_css_file.css';
```

PDFfont

DESCRIPTION:

Set the font to use when exporting to PDF

ALLOWED VALUES:

PDFfont : Times New Roman

FontName

Default: Arial

EXAMPLE OF USE:

```
$objGrid -> PDFfont = 'Times New Roman';
```

PDFfontsize

DESCRIPTION:

Set the font size to use when exporting to PDF

ALLOWED VALUES:

PDFfontsize : 8

Numeric.

Default: 7

EXAMPLE OF USE:

```
$objGrid -> PDFfontsize = '8';
```

PDFfill

DESCRIPTION:

It defines the RGB values for the background color for titles when exporting to PDF

ALLOWED VALUES:

PDFfill : array("R"=>0,"G"=>192,"B"=>192)

Default: array("R"=>192,"G"=>192,"B"=>192);

EXAMPLE OF USE:

```
$objGrid -> PDFfill = array("R"=>0,"G"=>192,"B"=>192);
```

PDFdraw

DESCRIPTION:

It defines the RGB values for the color of the letter to export to PDF

ALLOWED VALUES:

PDFdraw : array("R"=>0,"G"=>192,"B"=>192)

Default: array("R"=>0,"G"=>0,"B"=>0);

EXAMPLE OF USE:

```
$objGrid -> PDFdraw = array("R"=>0,"G"=>192,"B"=>192);
```

actHeader

DESCRIPTION:

It is used to generate a header in the options: **New**, **Edit**, **View**.

ALLOWED VALUES:

actHeader ['add'] : Header for New

```
$objGrid -> actHeader['add'] = 'Header for <strong>New</strong>';
```

actFooter

DESCRIPTION:

It is used to generate a footer in the options: **New**, **Edit**, **View**.

ALLOWED VALUES:

actFooter ['add'] : Footer for New

```
$objGrid -> actFooter['add'] = 'Footer for <strong>New</strong>';
```

images

DESCRIPTION:

This property is a vector containing the names of the image files to be used as icons in the Datagrid.

ALLOWED VALUES:

images ['add'] : add.gif

```
$objGrid -> images['add'] = 'add.gif';
```

ATENTION: If you change something in the name of image, make sure that the image exists in the folder of images

message

DESCRIPTION:

This property is a vector that contains the descriptive texts to be used in the Datagrid, changing their values, you'll customize the datagrid to use your own texts

ALLOWED VALUES:

message ['cancel' : Cancel
']

```
$objGrid -> message['cancel'] = 'Cancel';
```

•

debug

DESCRIPTION:

Generates messages to debug the results of phpMyDataGrid

ALLOWED VALUES:

debug : false

Boolean.

Default: false

EXAMPLE OF USE:

```
$objGrid -> debug = false;
```

•

logSQLError

DESCRIPTION:

Defines whether or not the SQL errors will be recorded in a log file

ALLOWED VALUES:

logSQLError : false

Boolean.

Default: true

EXAMPLE OF USE:

```
$objGrid -> logSQLError = false;
```

•

logfile

DESCRIPTION:

Defines the filename and path to log SQL errors

ALLOWED VALUES:

logfile : phpMyDGlogError.log

path and filename

EXAMPLE OF USE:

```
$objGrid -> logfile = 'phpMyDGlogError.log';
```

ATENTION: Filename must have write permissions

•

processData

DESCRIPTION:

It is used to define the name of a function that pre-process the information from the database before being displayed in the grid, this property is useful (for example) if it requires changing the value of a date or put a prefix to an invoice number, etc.

ALLOWED VALUES:

processData : functionName

Function name (without parenthesis or parameters)

EXAMPLE OF USE:

```
$objGrid -> processData = "functionName";
```

ATENTION: The function should receive an array as a parameter containing all the records display, and must return a new array with the elements modified

•

nocenter

DESCRIPTION:

If this property is defined, the windows of action (add, edit, view, search, export) will not be centered in the window, but would remain in the upper left corner of the screen

ALLOWED VALUES:

nocenter : false

Boolean.

Default: false

EXAMPLE OF USE:

```
$objGrid -> nocenter = false;
```

•

saveaddnew

DESCRIPTION:

Defines whether to add / edit displays the 'Save & New' button

ALLOWED VALUES:

saveaddnew : false

Boolean.

Default: false

EXAMPLE OF USE:

```
$objGrid -> saveaddnew = false;
```

•

height

DESCRIPTION:

Define a fixed height for the grid, if the data is larger than grid height then a scrollbar will be displayed.

ALLOWED VALUES:

height : 300px

Grid Height

EXAMPLE OF USE:

```
$objGrid -> height = '300px';
```

•

btnOrder

DESCRIPTION:

Define the order to display the record buttons

ALLOWED VALUES:

btnOrder : [E][V][D][Up][Dn]

Any possible conbination of the following definitions: [E][V][D][Up][Dn]

EXAMPLE OF USE:

```
$objGrid -> btnOrder = '[E][V][D][Up][Dn]';
```

ATENTION: Next we will explain each button representation:

[E] = Adir Button

[V] = View Button

[D] = Delete Button

[Up]= Row Up Button

[Dn]= Row Down Button

nowindow

DESCRIPTION:

Set this property to true if you do not want to have 'pop up' layers for data maintenance

ALLOWED VALUES:

nowindow : false

Boolean.

Default: false

EXAMPLE OF USE:

```
$objGrid -> nowindow = false;
```

toolbar

DESCRIPTION:

Modify the grid interfase by adding a toolbar for icons.

ALLOWED VALUES:

toolbar : false

Boolean.

Default: false

EXAMPLE OF USE:

```
$objGrid -> toolbar = false;
```

delchkbtn

DESCRIPTION:

This property works together with the toolbar property, set to true to add a button to the toolbar that allow to delete multiple checked rows.

ALLOWED VALUES:

delchkbtn : false

Boolean.

Default: false

EXAMPLE OF USE:

```
$objGrid -> delchkbtn = false;
```

ATENTION: This property must be activated only if the toolbar property is set to 'true', otherwise you may get unexpected results

strExportInline

DESCRIPTION:

This property works together with the toolbar property, set to true to integrate the export window into the toolbar avoiding the use of popup layers.

ALLOWED VALUES:

strExportInline: false

Boolean.

Default: false

EXAMPLE OF USE:

```
$objGrid -> strExportInline = false;
```

ATENTION: This property must be activated only if the toolbar property is set to 'true', otherwise you may get unexpected results

strSearchInline

DESCRIPTION:

This property works together with the toolbar property, set to true to integrate the search window into the toolbar avoiding the use of popup layers.

ALLOWED VALUES:

strSearchInline: false

Boolean.

Default: false

EXAMPLE OF USE:

```
$objGrid -> strSearchInline = false;
```

ATENTION: This property must be activated only if the toolbar property is set to 'true', otherwise you may get unexpected results

reload

DESCRIPTION:

This property works together with the toolbar property, set to true to add into the toolbar a button for refresh/reload the DataGrid.

ALLOWED VALUES:

reload : false

Boolean.

Default: false

EXAMPLE OF USE:

```
$objGrid -> reload = false;
```

ATENTION: This property must be activated only if the toolbar property is set to 'true', otherwise you may get unexpected results

poweredby

DESCRIPTION:

Defines whether the phpMyDataGrid logo is displayed

ALLOWED VALUES:

poweredby : false

Boolean.

Default: false

EXAMPLE OF USE:

```
$objGrid -> poweredby = false;
```

METHODS

Define the name of form:

FUNCTION NAME:

Form ('formName',doForm);

PARAMETERS:

formName : employees

Name of the form

doForm : true

EXAMPLE OF USE:

```
$objGrid -> Form ('employees',true);
```

Define the method of the form:

FUNCTION NAME:

methodForm ('strMethod');

PARAMETERS:

strMethod : post

You can use either of the methods available form, **GET** or **POST**

EXAMPLE OF USE:

```
$objGrid -> methodForm ('post');
```

ATENTION: If you are making the call to phpMyDataGrid from a form defined by you, you must use the same method used in form.

If you want to propagate your own GET or POST parameters, pass the data to the script, so it continue transferring the parameters in the dataGrid, for this, use:

FUNCTION NAME:

linkparam("parameters");

PARAMETERS:

parameters : &session={$session}&userid={$userid}&option=4

You must set the parameter list in the same way that parameters of the type **GET**

EXAMPLE OF USE:

```
$objGrid -> linkparam("&session={$session}&userid={$userid}&option=4");
```

ATENTION:

The variables to be included in the list of parameters must have been previously caught, for instance, to capture the variable **session**:

If method = GET:
$session = $_GET['session'];

If method = POST:
$session = $_POST['session'];

If you do not know the request method or it can be mixed (GET and/or POST):
$session = (isset($_GET['session'])?$_GET['session']:(isset($_POST['session'])?$_POST['session']:''));

NOTE THAT THE LIST OF PARAMETERS ARE DELIMITED BY DOUBLE QUOTES AND THE NAME OF THE VARIABLES ENCLOSED BETWEEN BRACKETS {}, SO, THE VARIABLES ARE REPLACED BY ITS RECORD VALUE.

If you want to have a context menu with options for maintenance, use the following line:

FUNCTION NAME:

useRightClickMenu('classpath');

PARAMETERS:

classpath : phpMyMenu.inc.php

write the full path including filename for phpMyMenu class

EXAMPLE OF USE:

```
$objGrid -> useRightClickMenu('phpMyMenu.inc.php');
```

To define the way it generates the HTML:

FUNCTION NAME:

friendlyHTML(bolStat);

PARAMETERS:

bolStat	:	true

true = for readable friendly HTML output

false = For obfuscated code

EXAMPLE OF USE:

```
$objGrid -> friendlyHTML(true);
```

For compatibility with XHTML use this feature:

FUNCTION NAME:

closeTags(bolStat);

PARAMETERS:

bolStat	:	false

true = XHTML compatible code

false = HTML only code

EXAMPLE OF USE:

```
$objGrid -> closeTags(false);
```

To set the path where images are located:

FUNCTION NAME:

pathtoimages('strPath');

PARAMETERS:

strPath	:	/images/

Path to images

EXAMPLE OF USE:

```
$objGrid -> pathtoimages('/images/');
```

It is necessary to establish a connection to the database server to do that use the following function:

FUNCTION NAME:

conectadb('strServer', 'strUsername', 'strPassword', 'strDatabase', 'useADOdb', 'strType', intPort);

PARAMETERS:

strServer	:	localhost

Server Name or Server IP

strUsername : root

DataBase Username

strPassword : %my12PaSS%

Password

strDatabase : testdb

Database Name

useADOdb : false

true = Will use ADOdb library for connection

false = Will use MySQL native drivers on php

strType : mysql

For connetions made with ADOdb, you can define the database Type to use

intPort : 3306

DataBase port

EXAMPLE OF USE:

```
$objGrid -> conectadb('localhost', 'root', '%my12PaSS%', 'testdb', 'false', 'mysql', 3306);
```

ATENTION: If you choose to use ADOdb library to make the connection, you must download it from ADOdb website, and must include the necesary files at the beginning of the script

The default language for messages phpMyDataGrid is English, if you want to change the language, use this function:

FUNCTION NAME:

language('strLang');

PARAMETERS:

strLang : es

Enter the two-character ISO code language

EXAMPLE OF USE:

```
$objGrid -> language('es');
```

ATENTION: Available languages on phpMyDataGrid Professional are Spanish and English, if you want to add another language, you can just create the file in the **languages** folder.

To enable or disable maintenance icons, use:

FUNCTION NAME:

buttons(bolAdd, bolUpd, bolDel, bolChk, intColumn, 'strColumnName');

PARAMETERS:

bolAdd : true

true = Enable the option to add

false = Disable the option to add option

bolUpd : true

true = Enable the option to Edit

false = Disable the option to Edit Option

bolDel : true

true = Enable the option to Delete

false = Disable the option to Delete Option

bolChk : true

true = Enable the option to View

false = Disable the option to View option

intColumn : -1

Define the column in which you want to display the maintenance icons (-1 indicates at the right side of the grid)

strColumnName: Column Title

Complete this field, whether to show a title in the column of icons

EXAMPLE OF USE:

```
$objGrid -> buttons(true, true, true, true, -1, 'Column Title');
```

•

To enable or disable the options for export, use the following function:

FUNCTION NAME:

export(bolExportsheet, bolExportCSV, bolExportXML, bolPrinter, bolExportPDF, 'pdfOrientation');

PARAMETERS:

bolExportsheet: true

true = Enable the option to export to worksheet (XLS)

false = Disable the option to export to worksheet (XLS)

bolExportCSV : true

true = Enable the option to export to CSV

false = Disable the option to export to CSV

bolExportXML : true

true = Enable the option to export to XML

false = Disable the option to export to XML

bolPrinter : true

true = Enable the option to print

false = Disable the option to print

bolExportPDF : true

true = Enable the option to export to PDF

false = Disable the option to export to PDF

pdfOrientation : P

Select the page orientation to export to PDF

(P) = Portrait

(L) = Landscape

EXAMPLE OF USE:

```
$objGrid -> export(true, true, true, true, true, 'P');
```

•

If you want to have a column with checkboxes for multiple-choice, use the following function:

FUNCTION NAME:

checkable(status);

PARAMETERS:

status : false

true = Enable checkboxes column

false = Disable checkboxes column

EXAMPLE OF USE:

```
$objGrid -> checkable(false);
```

•

To define a title in the grid, use:

FUNCTION NAME:

TituloGrid('strTitle');

PARAMETERS:

strTitle : Grid Title

EXAMPLE OF USE:

```
$objGrid -> TituloGrid('Grid Title');
```

•

To define a footnote in the grid, use:

FUNCTION NAME:

FooterGrid('strFooter');

PARAMETERS:

strFooter	:	Grid footnote, © 2008 Gurú Sistemas

EXAMPLE OF USE:

```
$objGrid -> FooterGrid('Grid footnote, © 2008 Gurú Sistemas');
```

To control the number of rows to display on each page:

FUNCTION NAME:

datarows(intLines);

PARAMETERS:

intLines	:	15

Number of entries per page

EXAMPLE OF USE:

```
$objGrid -> datarows(15);
```

If you want to set the number of pages that appear before summarize with ... change the amount with this function:

FUNCTION NAME:

linksperpage('amount');

PARAMETERS:

amount	:	4

Enter the number of links that you want to display

EXAMPLE OF USE:

```
$objGrid -> linksperpage('4');
```

ATENTION: Do not define very large quantities because they can distort the structure of the Grid

By setting linksperpage = 5, you'll look something like this:

1 2 3 4 5 ... 15 16 17 18 19 **20** 21 22 23 24 25 ... 45 46 47 48 49

To change the pagination format, use:

FUNCTION NAME:

paginationmode('pgm',inTable);

PARAMETERS:

pgm	:	mixed Define the type of paging, there are 3 available values: **links** = Generates a list page numbers indicating the page number to which the user want to go (Recommended tables with no more than 20 pages) **select** = Generates a menu that lets you choose the number of page you want to go **mixed** = Generates a list combining the above methods (default) **input** = Set a input for the user type the page numbers
inTable	:	false This parameter is not used anymore, left for backward compatibility

EXAMPLE OF USE:

```
$objGrid -> paginationmode('mixed',false);
```

•

You can define a security code (Magic word) which will help to phpMyDataGrid to be safer:

FUNCTION NAME:

salt('code');

PARAMETERS:

code	:	salt&pepper

EXAMPLE OF USE:

```
$objGrid -> salt('salt&pepper');
```

•

Set the number of decimal digits to be displayed in numeric fields:

FUNCTION NAME:

decimalDigits('amount');

PARAMETERS:

amount	:	2

EXAMPLE OF USE:

```
$objGrid -> decimalDigits('2');
```

•

You can define the character you want to use as a decimal point:

FUNCTION NAME:

decimalPoint('char');

PARAMETERS:

char	:	.

EXAMPLE OF USE:

```
$objGrid -> decimalPoint('.');
```

The online edition can be: off, activated with a single click and save when leave the field, or ask for confirmation to save. To define this, use:

FUNCTION NAME:

ajax('style',clicks);

PARAMETERS:

style	:	none

none = Disable online edition

default = Enables online edition with confirmation recording

silent = Enables online edition with automatic recording

clicks	:	1

Define the amount of clicks needed to activate the data for online edition.

1 - Single click

2 - Double click

EXAMPLE OF USE:

```
$objGrid -> ajax('none',1);
```

If you want to differentiate on the screen the values that have been modified via AJAX, you can use this function:

FUNCTION NAME:

AjaxChanged('strColor');

PARAMETERS:

strColor	:	#900

It must be a valid hex color

EXAMPLE OF USE:

```
$objGrid -> AjaxChanged('#900');
```

If for some reason needed to check whether a call is being made to AJAX page can use this function:

FUNCTION NAME:

isAjaxRequest();

EXAMPLE OF USE:

```
$objGrid -> isAjaxRequest();
```

ATENTION:

```
if ($objGrid -> isAjaxRequest()){
    echo 'This is an AJAX request';

} else {
    echo 'This is a direct request';

}
```

You can customize the actions to do when a maintenance button is clicked:

FUNCTION NAME:

setAction('button', 'event');

PARAMETERS:

button	:	add
event	:	javascript_function()

EXAMPLE OF USE:

```
$objGrid -> setAction('add', 'javascript_function()');
```

ATENTION: Considerations to have in mind to set new processes buttons:

Add button = None

Sample: $objGrid -> setAction('add','new_adicionar()');

Edit button = Function must have 2 parameters, delimited in the following way: (\"%s\",\"%s\");

Sample: $objGrid -> setAction('edit','new_editrow(\"%s\",\"%s\")');

Delete Button = Function must have 2 parameters, delimited in the following way: (\"%s\",\"%s\");

Sample: $objGrid -> setAction('delete','new_deleterow(\"%s\",\"%s\")');

Search Button = None

Sample: $objGrid -> setAction('search','new_search()');

View Button = Function must have 2 parameters, delimited in the following way: (\"%s\",\"%s\");

Sample: $objGrid -> setAction('view','new_viewrow(\"%s\",\"%s\")');

It is necessary to define the table on which the grid will work:

FUNCTION NAME:

tabla('strTable');

PARAMETERS:

strTable	:	employees

EXAMPLE OF USE:

```
$objGrid -> tabla('employees');
```

You can define a WHERE condition to filter and display only the records that meet the condition:

FUNCTION NAME:

where('strWhere');

PARAMETERS:

strWhere	:	active = 1

EXAMPLE OF USE:

```
$objGrid -> where('active = 1');
```

You can define a HAVING condition to filter and display only the records that meet the condition:

FUNCTION NAME:

having('strHaving');

PARAMETERS:

strHaving : active = 1

EXAMPLE OF USE:

```
$objGrid -> having('active = 1');
```

If you need to group records by some fields, or fields, you can use this function:

FUNCTION NAME:

groupby('strGroup');

PARAMETERS:

strGroup : status

EXAMPLE OF USE:

```
$objGrid -> groupby('status');
```

ATENTION: Note that when grouping data, the functionality of maintenance (Add, Edit, Delete, online edition), not function properly, therefore it is recommended to disable, or if necessary, implement your own maintenance processes.

Define the fields for which you want to sort the records:

FUNCTION NAME:

orderby('fields','style');

PARAMETERS:

fields : name,lastname

List of fields to which you want to sort the output, separated by commas

style : asc,desc

Define the type ordering type for each field ASC or DESC, if blank, **ASC** will be used automatically

EXAMPLE OF USE:

```
$objGrid -> orderby('name,lastname','asc,desc');
```

The professional version has a feature which will allow you to order manually a set of records, which is useful for example to define the order in which products will appear on a page:

FUNCTION NAME:

setorderarrows('field');

PARAMETERS:

field	:	consecutivo

The consecutive field should not be the table ID

EXAMPLE OF USE:

```
$objGrid -> setorderarrows('consecutivo');
```

ATENTION: The ordering field should be consecutive and not be autoincrementable, for the examples that we have developed, its value is always the same as the autoincrementable ID value

By default, phpMyDataGrid enables the ordering arrows in the titles of all the columns, if you want to disable this feature use the following function:

FUNCTION NAME:

noorderarrows();

EXAMPLE OF USE:

```
$objGrid -> noorderarrows();
```

ATENTION: If you want to disable the ordering of a few columns only, DO NOT use this function, instead use chField specifying the necessary parameters.

phpMyDataGrid automatically generates SQL queries based on the information provided, but sometimes it is necessary to generate advanced queries that require the programmer defined them manually, to do this, use:

FUNCTION NAME:

sqlstatement('strSQL','strCount');

PARAMETERS:

strSQL	:	SELECT *, now() as fecha from employees
strCount	:	select count(*) from employees

EXAMPLE OF USE:

```
$objGrid -> sqlstatement('SELECT *, now() as fecha from employees','select count(*) from employees');
```

ATENTION: User defined SQL queries should not include sentences WHERE, GROUP or ORDER, these must be defined directly from the class functions.
You should have in mind that it is very important that all the names of fields used in the grid must be defined in the SQL query

For the developer it is essential to track the performance of the scripts, this feature allows to the programmer to receive emails with 'any' SQL errors that may be generated:

FUNCTION NAME:

reportSQLErrorsTo('strMail', 'bolShow');

PARAMETERS:

strMail	:	developer@thecompanymail.com Developer e-mail
bolShow	:	false **true** = Display SQL errors on the screen to end user (Recommended while developing)) **false** = Hides any generated SQL errors (Recommended for production environments)

EXAMPLE OF USE:

```
$objGrid -> reportSQLErrorsTo('developer@thecompanymail.com', 'false');
```

ATENTION: This function uses the **mail()** php function, therefore to work properly, it must be configured and working

For maintenance options it is necesary to define a key fields, to do this, use:

FUNCTION NAME:

keyfield('strField');

PARAMETERS:

strField	:	id

EXAMPLE OF USE:

```
$objGrid -> keyfield('id');
```

To allow search data in phpMyDataGrid, simply define the fields which will be allowed to search for:

FUNCTION NAME:

searchby('listoffields');

PARAMETERS:

listoffields	:	firstname,lastname,date:select You can add the instruction **:SELECT** to field name to display a drop-down menu with the resume of all possible values to search.

EXAMPLE OF USE:

```
$objGrid -> searchby('firstname,lastname,date:select');
```

If you like to set the 'Reset Search' button outside the grid, use this function:

FUNCTION NAME:

getResetSearch(icon);

PARAMETERS:

icon	:	false Defines if the return data is just text or a icon, default: false

EXAMPLE OF USE:

```
$objGrid -> getResetSearch(false);
```

One of the main functions is FormatColumn, with this function you define the characteristics of each one of the fields displayed on grid:

FUNCTION NAME:

FormatColumn('strfieldName','strHeader','fieldWidth','maxlength','inputtype','columnwidth','align','Mask','default','cutChar');

PARAMETERS:

strfieldName : lastname

Field name

strHeader : Lastname

Column Title

fieldWidth : 0

Used only in **textarea** fields, it is used to set the textarea rows

maxlength : 20

Max length allowed to be typed by user

inputtype : 0

Field type

0 = Normal Field

1 = Read only field

2 = Hidden field

3 = Image, calc, link or imagelink without relation to a field

4 = Image, calc or link related with a field

columnwidth : 80

Column width (in pixels)

align : center

Column Text align

center = (default)

left

right

Mask : text

Masking to field

text = Normal field text

textarea = Region for text editing (may have a larger area that the fields type **text**)

image = Display an image, may be field related or static. (see samples)

imagelink = Display a clickable image, may be field related or static. (see samples)

number = Numeric field

money = Numeric money sign field, format: money:sign, sample**money:$ money:£**

date = Date type field, format: date:format:separator, sample **date:dmy:/ date:ymd:-**

datetime = Date and time type field, format: datetime:dateformat:separator:timeformat,separator, sample**datetime:datetime:mdy:/:His,: or datetime:mdy:/:hisa,:**

link = Link field. view samples

password = Password type field

calc = Calculated field. view samples

scalc = Calculated field which stores the result in a field, view samples

bool = Boolean field, create a checkbox and store 0 when unchecked and 1 when checked

check = same as **bool** type field

select = Field with a dropdown menu, the options can be manually defined or dynamically loaded from another table in the database.

0 = Integer, no decimals

1 = Float 1 decimal

2 = Float 2 decimals

3 = Float 3 decimals

4 = float 4 decimals

integer = Integer, no decimals

related = Find a correspondence in other table based in the field value

array() - conditional = Display a value based on a set of conditions that meet the field value. (see samples)

default :

Default value for field (Used in add option)

cutChar :

Used generally in **textarea** fields which are too long, by using this parameter the grid will only display the amount of **X** characters

EXAMPLE OF USE:

```
$objGrid -> FormatColumn('lastname','Lastname','0','20','0','80','center','text','','');
```

•

You can define aditional atributes for each column:

FUNCTION NAME:

chField('strfieldName', 'permissions', overwrite)

PARAMETERS:

strfieldName : nombrecampo

Field Name

permissions : N+

N+ = Display field while adding

N- = Hide field while adding

E+ = Display field while editing

E- = Hide field while editing

V+ = Display field while viewing records

V- = Hide field while viewing

R = Remove order arrows for field

U = (only for imagetype fields) allow new images to be uploaded to the field

M = (only for imagetype fields) Allow images to be uploaded, no matter if field already has an image on it

X+ = To export hidden fields in grid

X- = To not export visible fields

overwrite : false

Define whether the definitions made earlier to the field must be erased or cumulative

EXAMPLE OF USE:

```
$objGrid -> chField('nombrecampo', 'N+', false)
```

ATENTION: You would combine several atributes in the same request, sample:

```
$objGrid -> chField('firstname','N-E+V+R');
```

Define the size for the input boxes when editing/adding:

FUNCTION NAME:

setInputWidth('field','width');

PARAMETERS:

field	:	field Name
width	:	300px

EXAMPLE OF USE:

```
$objGrid -> setInputWidth('field Name','300px');
```

If you want to validate user input data via Javascript use:

FUNCTION NAME:

jsValidate('strField', 'strValidation', 'strErrorMessage', 'strDisplayMessage');

PARAMETERS:

strField	:	firstname FieldName
strValidation	:	this.value.length>=8 JavaScript code to validate, (You can call a JS function as well)
strErrorMessage	:	Name must have at least 7 chars long Error Message
strDisplayMessage:		Write customer name Input instructions message

EXAMPLE OF USE:

```
$objGrid -> jsValidate('firstname', 'this.value.length>=8', 'Name must have at least 7 chars long', 'Write customer name');
```

ATENTION: Javascript validations will be executed on **New** and **Edit** options, as well in online edition, please keep in mind you can validate only the actual field (this.) validations can not be done combined with other fields

To Just display an instructions message to user, use:

FUNCTION NAME:

fldComment('strField', 'strDisplayMessage');

PARAMETERS:

strField	:	lastname Field name
strDisplayMessage:		Write customer lastname Input instructions message

EXAMPLE OF USE:

```
$objGrid -> fldComment('lastname', 'Write customer lastname');
```

ATENTION: Those messages will be available in **Add** and **Edit** options only.

If you like to use a datepicker, you should use this function:

FUNCTION NAME:

useCalendar(bolCalendar);

PARAMETERS:

bolCalendar	:	true

EXAMPLE OF USE:

```
$objGrid -> useCalendar(true);
```

To totalize columns, use:

FUNCTION NAME:

total('fields');

PARAMETERS:

fields	:	salary,days,total List of fields to totalize, comma separated

EXAMPLE OF USE:

```
$objGrid -> total('salary,days,total');
```

ATENTION: Remember, now you can totalize calculated colums as well

Define the size (width, height) to display the pictures in a field:

FUNCTION NAME:

setImageSize('field','width','height');

PARAMETERS:

field	:	fieldName
		Field Name
width	:	95
		Width in px (just the integer value)
height	:	127
		Height in px (just the integer value)

EXAMPLE OF USE:

```
$objGrid -> setImageSize('fieldName','95','127');
```

ATENTION: This function do not resize the stored images, just resize the output image in browser.

Lets you set a conditional format to CELLS that meet a condition:

FUNCTION NAME:

addCellStyle('field','condition', 'style');

PARAMETERS:

field	:	fieldName
		Field Name
condition	:	[\"activo\"]==1
		String, condition.
style	:	CSSClassName
		Name of the CSS class to use

EXAMPLE OF USE:

```
$objGrid -> addCellStyle('fieldName','["activo"]==1', 'CSSClassName');
```

ATENTION: It is possible to use any field used within the grid, simply type the field name between [' ... '].

Lets you set a conditional format to ROWS that meet a condition:

FUNCTION NAME:

addRowStyle('condition', 'style');

PARAMETERS:

condition	:	[\"activo\"]==1 String, condition.
style	:	CSSClassName Name of the CSS class to use

EXAMPLE OF USE:

```
$objGrid -> addRowStyle('["activo"]==1', 'CSSClassName');
```

ATENTION: It is possible to use any field used within the grid, simply type the field name between [' ... '].

Define the path for skin related images:

FUNCTION NAME:

skinimages('skin', 'path');

PARAMETERS:

skin	:	SkinName Skin name
path	:	skins/%s/icons/ Path to images

EXAMPLE OF USE:

```
$objGrid -> SkinNameimages('SkinName', 'skins/%s/icons/');
```

ATENTION: The %s in path will be replaced by skin name, if this %s does not exist, the path will be used as is.

Changes the value to store instead of the specified in the online edition:

FUNCTION NAME:

setNewInlineData('newData');

PARAMETERS:

newData	:	newData

New data to store when editing online

EXAMPLE OF USE:

```
$objGrid -> setNewInlineData('newData');
```

Define a JS function to run after updating online, may have the parameters (idkey,field,newtext,oldtext):

FUNCTION NAME:

onAjaxUpdate('js');

PARAMETERS:

js	:	javascriptFunction(idkey,field,newtext,oldtext)

JavaScript function name to call when the Online edition finishes

EXAMPLE OF USE:

```
$objGrid -> onAjaxUpdate('javascriptFunction(idkey,field,newtext,oldtext)');
```

Returns the value of the AJAX action being executed:

FUNCTION NAME:

getAjaxID();

EXAMPLE OF USE:

```
$objGrid -> getAjaxID();
```

Returns codes of the selected checkboxes:

FUNCTION NAME:

getCheckedBoxes();

EXAMPLE OF USE:

```
$objGrid -> getCheckedBoxes();
```

Returns true if the AJAX action is Adding data:

FUNCTION NAME:

isadding();

EXAMPLE OF USE:

```
$objGrid -> isadding();
```

Returns true if the request is an online edition:

FUNCTION NAME:

isOnlineEdition();

EXAMPLE OF USE:

```
$objGrid -> isOnlineEdition();
```

Returns the data is being edited online:

FUNCTION NAME:

getEditedData();

EXAMPLE OF USE:

```
$objGrid -> getEditedData();
```

Set an image by aplying a new parameter:

FUNCTION NAME:

changeImage();

EXAMPLE OF USE:

```
$objGrid -> changeImage();
```

Add a separator in the toolbar:

FUNCTION NAME:

addSeparator();

EXAMPLE OF USE:

```
$objGrid -> addSeparator();
```

ATENTION: This feature is only valid if property **toolbar** is true

Add Buttons to the toolbar:

FUNCTION NAME:

addButton('img', 'action', 'message');

PARAMETERS:

img	:	/path/to/image.png Path and image name of the icon to display
action	:	javascriptFunction() Javascript funcion to call when the button is selected
message	:	Message In Toolbar Message to display at side of the icon

EXAMPLE OF USE:

```
$objGrid -> addButton('/path/to/image.png', 'javascriptFunction()', 'Message In Toolbar');
```

ATENTION: This feature is only valid if property **toolbar** is true

Add a select menu to toolbar:

FUNCTION NAME:

addSelect('arrData', 'action', 'message');

PARAMETERS:

arrData	:	array('toDisplay1','key2'=>'toDisplay2','key3'=>'toDisplay3','keyN'=>'toDisplayN')' onblur='ut("$objGrid","58","addSelect(¬c¬arrData¬c¬, ¬c¬action¬c¬, ¬c¬message¬c¬);","ta58","arrData,action,message",false)' />

action : javascriptFunction()

Name of the JavaScript function to execute

message : Message In Toolbar

Message to display at side of the icon

EXAMPLE OF USE:

```
$objGrid -> addSelect('array('key1'=>'toDisplay1','key2'=>'toDisplay2','key3'=>'toDisplay3','keyN'=>'toDisplayN')', 'javascriptFunction()', 'Message In Toolbar');
```

ATENTION: This feature is only valid if property **toolbar** is true

This function is available for backward compatibility, it is not recommended to be used, instead you would insert the JS and CSS files by yourself or use the function set_DG_Header:

FUNCTION NAME:

setHeader('phpScriptFile', 'jsFile', 'cssFile', 'jsCalFile', 'cssCalFile', 'jsmmenu');

PARAMETERS:

phpScriptFile : sample.php

php script name

jsFile : js/dgscripts.js

path and name to datagrid JS file

cssFile : css/dgstyle.css

path and name to datagrid CSS file

jsCalFile : js/dgcalendar.js

path and name to calendar JS file

cssCalFile : css/dgcalendar.css

path and name to calendar CSS file

jsmmenu : mmscripts.js

path and name to right click menu

EXAMPLE OF USE:

```
$objGrid -> setHeader('sample.php', 'js/dgscripts.js', 'css/dgstyle.css', 'js/dgcalendar.js', 'css/dgcalendar.css',  'mmscripts.js');
```

Finally, call this function which will render the grid based on all settings previously provided:

FUNCTION NAME:

grid();

EXAMPLE OF USE:

```
$objGrid -> grid();
```

If you need to pass parameters to a query is recommended to use this function which will analyze the contents of the variable, and amended if necessary to prevent SQL injections.:

FUNCTION NAME:

magic_quote($variable);

EXAMPLE OF USE:

```
magic_quote($variable);
```

ATENTION: Note that this function is not part of the DataGrid class, and therefore should not be called as an object but as independent function!

It is recommended to include JS and CSS files, by using this function, see examples for details:

FUNCTION NAME:

set_DG_Header('path/to/JS/', 'path/to/CSS/', 'closetagChar', 'skinName')

EXAMPLE OF USE:

```
set_DG_Header('path/to/JS/', 'path/to/CSS/', 'closetagChar', 'skinName')
```

ATENTION: Note that this function is not part of the DataGrid class, and therefore should not be called as an object but as independent function!