phpMyDataGrid Professional Evolution - Documentación

Instalación

Incluya el archivo:

Función:

require_once('path');

Parámetros:

path	:	classes/phpMyDataGrid.class.php
		Ruta de la clase

Ejemplo de uso:

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

Defina el objeto de la clase:

Función:

datagrid('scriptName','gridID');

Parámetros:

scriptName	:	sample.php
		Nombre del archivo
gridID	:	1
		ID del Grid (Se debe usar un código diferente en cada DataGrid si desea tener varios en la misma página)

Ejemplo de uso:

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

Propiedades

ButtonWidth

Descripción:

Especifica el ancho de los iconos usados como botones en phpMyDataGrid.

Valores posibles:

ButtonWidth	:	25

Número entero.

Predeterminado: 25

EJEMPLO DE USO:

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

•

backtick

DESCRIPCIÓN:

Algunas bases de datos como MySQL permiten que el nombre de los campos , tablas o bases de datos se encuentren delimitados, por ejemplo con el caracter `

VALORES POSIBLES:

backtick : `

Caracter.

Predeterminado: `

EJEMPLO DE USO:

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

•

liquidTable

DESCRIPCIÓN:

Define si phpMyDataGrid ocupará automáticamente o no el ancho disponible en pantalla.

VALORES POSIBLES:

liquidTable : false

Booleano.

Predeterminado: false

EJEMPLO DE USO:

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

•

width

DESCRIPCIÓN:

Complementa a la propiedad liquidTable definiendo el porcentaje de pantalla que ocupará automáticamente.

VALORES POSIBLES:

width : 100%

Cadena, representando un porcentaje.

Predeterminado: 100%

EJEMPLO DE USO:

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

•

condition

DESCRIPCIÓN:

Ahora es posible condicionar la presentación de algunos registros para darles otro formato de salida, para ello basta con definir la condición que se debe cumplir.

VALORES POSIBLES:

condition : ['activo']==1

Cadena, condición.

EJEMPLO DE USO:

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

IMPORTANTE: Es posible usar cualquier campo que se encuentre en uso dentro del grid, basta con escribir el nombre del campo entre [' ... ']
NOTA: la versión evolution de phpMyDataGrid, tiene la opción de usar esta propiedad para definir un estilo CSS, o puede usar el método **addRowStyle** que le permitirá definir un conjunto de condiciones que al cumplirse formatearán los datos del grid usando classes CSS

•

conditionalStyle

DESCRIPCIÓN:

Esta propiedad complementa el uso de la propiedad condition, en esta propiedad se definen los estilos con los que se mostrarán los registros que cumplan la condición especificada.

VALORES POSIBLES:

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

Lista de Estilos CSS

EJEMPLO DE USO:

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

IMPORTANTE: Si utiliza el método **addRowStyle** para definir condicionantes, no es necesario utilizar esta propiedad

•

conditionEdit

DESCRIPCIÓN:

Si desea limitar la edición de registros a solo los que cumplan con cierta condición, utilice la siguiente propiedad con la condición que se debe cumplir.

VALORES POSIBLES:

conditionEdit : ['activo']==1

Cadena, condición.

EJEMPLO DE USO:

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

IMPORTANTE: Es posible usar cualquier campo que se encuentre en uso dentro del grid, basta con escribir el nombre del campo entre [' ... ']

conditionDelete

DESCRIPCIÓN:

Si desea limitar el borrado de registros a solo los que cumplan con cierta condición, utilice la siguiente propiedad con la condición que se debe cumplir.

VALORES POSIBLES:

conditionDelete: ['activo']==1

Cadena, condición.

EJEMPLO DE USO:

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

IMPORTANTE: Es posible usar cualquier campo que se encuentre en uso dentro del grid, basta con escribir el nombre del campo entre [' ... ']

moneySign

DESCRIPCIÓN:

Define el signo a mostrar en campos de tipo Money

VALORES POSIBLES:

moneySign : $

1 Caracter.

Predeterminado: $

EJEMPLO DE USO:

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

zebraLines

DESCRIPCIÓN:

Define la forma en que se trazará el interlineado en phpMyDataGrid

VALORES POSIBLES:

zebraLines : 1

Entero

Predeterminado: 1

EJEMPLO DE USO:

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

showToOf

DESCRIPCIÓN:

Esta propiedad activa si se muestra o no el mensaje **Mostrando registros X a Y**

VALORES POSIBLES:

showToOf : true

Booleano.

Predeterminado: true

EJEMPLO DE USO:

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

AllowChangeNumRows

DESCRIPCIÓN:

phpMyDataGrid permite al usuario cambiar la cantidad de registros que desea visualizar por página, esto puede ser cambiado definiendo esta propiedad como false

VALORES POSIBLES:

AllowChangeNumRows: true

Booleano.

Predeterminado: true

EJEMPLO DE USO:

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

charset

DESCRIPCIÓN:

Permite definir la codificación de caracteres para mostrar la página.

VALORES POSIBLES:

charset : UTF-8

Cadena

Predeterminado: ISO-8859-1

EJEMPLO DE USO:

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

sqlDataCoding

DESCRIPCIÓN:

Permite definir el tipo de codificación a usar en la base de datos

VALORES POSIBLES:

sqlDataCoding: SET NAMES 'utf8'

Cadena

EJEMPLO DE USO:

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

sqlcharset

DESCRIPCIÓN:

Define a MySQL el tipo de codificación que usará

VALORES POSIBLES:

sqlcharset : utf8

Cadena

utf8, big5, latin1, etc

EJEMPLO DE USO:

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

IMPORTANTE: Es importante definir esta propiedad como complemento a **sqlDataCoding** para un correcto funcionamiento

defaultdateformat

DESCRIPCIÓN:

Permite definir el formato predeterminado para los campos tipo fecha

VALORES POSIBLES:

defaultdateformat: ymd

Combinación de las letras dmy

Predeterminado: dmy

EJEMPLO DE USO:

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

defaultdateseparator

DESCRIPCIÓN:

Permite definir el separador para los campos tipo fecha.

VALORES POSIBLES:

defaultdateseparator: -

1 Caracter.

Predeterminado: /

EJEMPLO DE USO:

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

csvSeparator

DESCRIPCIÓN:

Caracter a usar para separar los valores al exportar en formato CSV (Valores separados por coma)

VALORES POSIBLES:

csvSeparator : ;

1 caracter

Predeterminado: ;

EJEMPLO DE USO:

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

uploadDirectory

DESCRIPCIÓN:

Esta propiedad define el directorio al cual se pueden **subir** o cargar las imágenes

VALORES POSIBLES:

uploadDirectory: /images/upload/

ruta de ubicación válida, puede ser relativa o absoluta

Predeterminado: /

EJEMPLO DE USO:

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

IMPORTANTE: La ruta especificada debe tener permisos de escritura.

show404image

DESCRIPCIÓN:

Le indica a phpMyDataGrid si debe mostrar o no una imagen representativa cuando no encuentre la imagen relacionada a un campo.

VALORES POSIBLES:

show404image: false

Booleano.

Predeterminado: false

EJEMPLO DE USO:

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

•

retcode

DESCRIPCIÓN:

De forma predeterminada, phpMyDataGrid es renderizado al terminar la ejecución del script, pero si la propiedad retcode es cambiada a true, se devolverá el código HTML para ser almacenado en una variable.

VALORES POSIBLES:

retcode : false

Booleano

Predeterminado: false

EJEMPLO DE USO:

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

•

getMyOwnButtons

DESCRIPCIÓN:

Permite al programador tener control sobre la posición de los botones Nuevo/Buscar/Exportar, al no mostrarlos directamente en el grid, sino que permite posicionarlos en cualquier sección de su sitio Web.

VALORES POSIBLES:

getMyOwnButtons: false

Booleano.

Predeterminado: false

EJEMPLO DE USO:

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

IMPORTANTE: Para su funcionamiento, la propiedad **retcode** debe estar puesta a **true**.

•

strAddBtn:

FUNCIÓN:

Esta propiedad complementa la propiedad **getMyOwnButtons**, devolviendo en ella el código HTML del botón adicionar

EJEMPLO DE USO:

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

•

strSearchBtn:

FUNCIÓN:

Esta propiedad complementa la propiedad **getMyOwnButtons**, devolviendo en ella el código HTML del botón buscar

EJEMPLO DE USO:

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

•

strExportBtn:

FUNCIÓN:

Esta propiedad complementa la propiedad **getMyOwnButtons**, devolviendo en ella el código HTML del botón exportar

EJEMPLO DE USO:

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

•

cssPrinter

DESCRIPCIÓN:

Define el nombre del archivo de estilos que será usado para impresión.

VALORES POSIBLES:

cssPrinter : /css/my_css_file.css

Ruta y nombre del archivo CSS

Predeterminado: css/b-w-print.css

EJEMPLO DE USO:

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

•

PDFfont

DESCRIPCIÓN:

Define el tipo de letra a usar al exportar a PDF

VALORES POSIBLES:

PDFfont : Times New Roman

Nombre de tipo de letra

Predeterminado: Arial

EJEMPLO DE USO:

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

PDFfontsize

DESCRIPCIÓN:

Define el tamaño tipo de letra a usar al exportar a PDF

VALORES POSIBLES:

PDFfontsize : 8

Numérico.

Predeterminado: 7

EJEMPLO DE USO:

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

PDFfill

DESCRIPCIÓN:

Define los valores RGB para el color de fondo para los títulos al exportar a PDF

VALORES POSIBLES:

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

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

EJEMPLO DE USO:

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

PDFdraw

DESCRIPCIÓN:

Define los valores RGB para el color de la letra al exportar a PDF

VALORES POSIBLES:

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

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

EJEMPLO DE USO:

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

actHeader

DESCRIPCIÓN:

Se usa para generar un encabezado en las opciones: **Nuevo**, **Editar**, **Ver**.

VALORES POSIBLES:

actHeader ['add'] : Encabezado para Nuevo

```
$objGrid -> actHeader['add'] = 'Encabezado para <strong>Nuevo</strong>';
```

actFooter

DESCRIPCIÓN:

Se usa para generar un pie de página en las opciones: **Nuevo**, **Editar**, **Ver**.

VALORES POSIBLES:

actFooter ['add'] : Pie de página para Nuevo

```
$objGrid -> actFooter['add'] = 'Pie de página para <strong>Nuevo</strong>';
```

images

DESCRIPCIÓN:

Esta propiedad es un vector que contiene los nombres de los archivos de imagen que se usarán como iconos en el Datagrid.

VALORES POSIBLES:

images [' add '] : add.gif

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

IMPORTANTE: Si cambia algún valor de nombre de imagen, asegúrese que la imagen exista en la carpeta de imágenes

•

message

DESCRIPCIÓN:

Esta propiedad es un vector que contiene los textos descriptivos que se usarán en el Datagrid, cambiando sus valores, personalizará el datagrid para que utilice sus propios textos

VALORES POSIBLES:

message [' cancel '] : Cancel

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

•

debug

DESCRIPCIÓN:

Genera mensajes para depurar los resultados de phpMyDataGrid

VALORES POSIBLES:

debug : false

Booleano.

Predeterminado: false

EJEMPLO DE USO:

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

•

logSQLError

DESCRIPCIÓN:

Define si se registrará o no los errores SQL en un archivo de registro

VALORES POSIBLES:

logSQLError : false

Booleano.

Predeterminado: true

EJEMPLO DE USO:

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

•

logfile

DESCRIPCIÓN:

Define el nombre de archivo y la ruta para registrar errores SQL

VALORES POSIBLES:

logfile : phpMyDGlogError.log

Ruta y nombre del archivo

EJEMPLO DE USO:

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

IMPORTANTE: El archivo en la ruta especificada debe tener permisos de escritura

•

processData

DESCRIPCIÓN:

Se utiliza para definir el nombre de una función que pre-procesará la información de la base de datos antes de ser mostrada en el grid, esta propiedad es útil (por ejemplo) si requiere alterar el valor de una fecha, o anteponer un prefijo a un número de factura, etc

VALORES POSIBLES:

processData : functionName

Nombre de la funcion sin paréntesis o parámetros

EJEMPLO DE USO:

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

IMPORTANTE: La función a llamar debe recibir como parametro un array con todos los registros a mostrar, y deberá retornar de nuevo un array con los elementos modificados

•

nocenter

DESCRIPCIÓN:

Si esta propiedad es definida, las ventanas de acción (adicionar, editar, ver, buscar, exportar) no serán centradas en la ventana, sino que permanecerán en la esquina superior izquierda de la pantalla

VALORES POSIBLES:

nocenter : false

Booleano.

Predeterminado: false

EJEMPLO DE USO:

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

•

saveaddnew

DESCRIPCIÓN:

Define si al adicionar/editar se muestra el boton 'Grabar & nuevo'

VALORES POSIBLES:

saveaddnew : false

Booleano.

Predeterminado: false

EJEMPLO DE USO:

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

•

height

DESCRIPCIÓN:

Define una altura fija para el Grid sin importar cuantos registros contenga, en caso de sobrepasar el la altura, se mostrará una barra de desplazamiento.

VALORES POSIBLES:

height : 300px

Alto del Grid

EJEMPLO DE USO:

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

•

btnOrder

DESCRIPCIÓN:

Define el orden en que se presentarán los botones de mantenimiento por cada registro.

VALORES POSIBLES:

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

Cualquier combinación de las siguientes definiciones: [E][V][D][Up][Dn]

EJEMPLO DE USO:

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

IMPORTANTE: A continuación explicamos la representación de cada botón:

[E] = Botón Editar

[V] = Botón Ver

[D] = Botón Borrar

[Up]= Botón Subir Renglón

[Dn]= Botón Bajar Renglón

nowindow

DESCRIPCIÓN:

Al definir esta propiedad como verdadera, para los procesos de mantenimiento no se creara una ventana superpuesta, en cambio se generará una ventana en reemplazo del DataGrid

VALORES POSIBLES:

nowindow : false

Booleano.

Predeterminado: false

EJEMPLO DE USO:

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

toolbar

DESCRIPCIÓN:

Cambia la presentación tradicional del grid, organizando los iconos en una barra de botones.

VALORES POSIBLES:

toolbar : false

Booleano.

Predeterminado: false

EJEMPLO DE USO:

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

•

delchkbtn

DESCRIPCIÓN:

Trabaja en conjunto con la propiedad Toolbar, al definirla como verdadera adiciona a la barra de herramientas un botón que permite borrar masivamente los registros seleccionados.

VALORES POSIBLES:

delchkbtn : false

Booleano.

Predeterminado: false

EJEMPLO DE USO:

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

IMPORTANTE: Esta propiedad solo debe activarse si la propiedad toolbar esta puesta a 'true' de lo contrario puede obtener resultados inesperados

•

strExportInline

DESCRIPCIÓN:

Trabaja en conjunto con la propiedad Toolbar, al definirla como verdadera integra la ventana de exportación de datos a la barra de botones evitando generar ventanas emergentes.

VALORES POSIBLES:

strExportInline: false

Booleano.

Predeterminado: false

EJEMPLO DE USO:

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

IMPORTANTE: Esta propiedad solo debe activarse si la propiedad toolbar esta puesta a 'true' de lo contrario puede obtener resultados inesperados

•

strSearchInline

DESCRIPCIÓN:

Trabaja en conjunto con la propiedad Toolbar, al definirla como verdadera integra la ventana de busqueda a la barra de botones evitando generar ventanas emergentes.

VALORES POSIBLES:

strSearchInline: false

Booleano.

Predeterminado: false

EJEMPLO DE USO:

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

IMPORTANTE: Esta propiedad solo debe activarse si la propiedad toolbar esta puesta a 'true' de lo contrario puede obtener resultados inesperados

reload

DESCRIPCIÓN:

Trabaja en conjunto con la propiedad Toolbar, al definirla como verdadera adiciona a la barra de herramientas un botón que permite actualizar/recargar el grid.

VALORES POSIBLES:

reload : false

Booleano.

Predeterminado: false

EJEMPLO DE USO:

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

IMPORTANTE: Esta propiedad solo debe activarse si la propiedad toolbar esta puesta a 'true' de lo contrario puede obtener resultados inesperados

poweredby

DESCRIPCIÓN:

Define si el logo de phpMyDataGrid es mostrado

VALORES POSIBLES:

poweredby : false

Booleano.

Predeterminado: false

EJEMPLO DE USO:

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

M&ÉTODOS

Defina el nombre de formulario:

FUNCIÓN:

Form('formName', doForm);

PARÁMETROS:

formName : employees

Nombre del formulario

doForm : true

true = phpMyDatagrid generará automáticamente el formulario (Debe usar esta opción si desea que phpMyDataGrid cargue imágenes al servidor)

false = phpMyDataGrid usará un formulario creado por el programador

EJEMPLO DE USO:

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

Defina el método del formulario:

FUNCIÓN:

methodForm('strMethod');

PARÁMETROS:

strMethod : post

Puede usar cualquiera de los métodos de formulario disponibles, **GET** o **POST**

EJEMPLO DE USO:

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

IMPORTANTE: Si está haciendo el llamado a phpMyDataGrid desde un formulario definido por el programador, deberá usar el mismo método usado en el formulario.

Si usted traslada a la página del script sus propios parámetros, debería continuar trasladándolos al script del dataGrid, para esto, utilice:

FUNCIÓN:

linkparam("parameters");

PARÁMETROS:

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

Debe definir la lista de parámetros de la misma forma que se crean parámetros del tipo**GET**

EJEMPLO DE USO:

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

IMPORTANTE:

Las variables que se han incluido en la lista de parámetros deben haber sido previamente capturadas, por ejemplo, para capturar la variable **session**:

Si el envío es con GET:
$session = $_GET['session'];

Si el envío es con POST:
$session = $_POST['session'];

Si no conoce el origen del envío, o el envío puede ser mixto (GET y POST):
$session = (isset($_GET['session'])?$_GET['session']:(isset($_POST['session'])?$_POST['session']:''));

NÓTESE QUE LA LISTA DE PARÁMETROS SE ENCUENTRA DELIMITADA POR COMILLAS DOBLES Y EL NOMBRE DE LAS VARIABLES ENCERRADO ENTRE LLAVES {}, ASÍ LAS VARIABLES SON REEMPLAZADAS POR SU VALOR.

Si desea tener un menú contextual con las opciones de mantenimiento, utilice la siguiente línea:

FUNCIÓN:

useRightClickMenu('ruta');

PARÁMETROS:

ruta : phpMyMenu.inc.php

escriba la ruta completa, incluyendo el nombre del archivo de la clase phpMyMenu.

EJEMPLO DE USO:

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

Para definir el modo en que se genera el código HTML:

FUNCIÓN:

friendlyHTML(bolStat);

PARÁMETROS:

bolStat	:	true

true = si desea que el código HTML generado sea legible

false = Si desea que el código fuente generado esté 'obfuscado'

EJEMPLO DE USO:

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

Para compatibilidad con XHTML utilice esta función:

FUNCIÓN:

closeTags(bolStat);

PARÁMETROS:

bolStat : false

true = si desea que el HTML generado sea correcto para la declaración XHTML

false = Si desea que el código generado sea compatible con HTML

EJEMPLO DE USO:

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

Para definir la ruta (camino) en el cual se encuentran los iconos del datagrid:

FUNCIÓN:

pathtoimages('strPath');

PARÁMETROS:

strPath : /images/

Escriba la ruta en la cual se encuentran ubicados los archivos de imagen

EJEMPLO DE USO:

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

Es necesario establecer una conexión con el servidor de bases de datos, para esto utilice la siguiente función:

FUNCIÓN:

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

PARÁMETROS:

strServer : localhost

Nombre o dirección IP del servidor

strUsername : root

Nombre de usuario de la base de datos

strPassword : %my12PaSS%

Contraseña de usuario de la base de datos

strDatabase : testdb

Nombre de la base de datos

useADOdb : false

true = Ralizará la conexión a la base de datos usando la liberia ADOdb

false = Utilizará los drivers nativos de php para conectarse a MySQL

strType : mysql

Tipo de servidor de base de datos (aplica solamente para conexiones realizadas con ADOdb)

intPort : 3306

Puerto de escucha de la base de datos

EJEMPLO DE USO:

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

IMPORTANTE: Si decide utilizar la librería ADOdb para realizar la conexión, deberá descargarla desde su sitio web, asi mismo, deberá incluirla al inicio del script

El idioma predeterminado para los mensajes de phpMyDataGrid es inglés, si desea cambiar el idioma, utilice esta función:

FUNCIÓN:

language('strLang');

PARÁMETROS:

strLang : es

Escriba el código ISO de dos caracteres del idioma

EJEMPLO DE USO:

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

IMPORTANTE: phpMyDataGrid Professional tiene como idiomas base el español **es** y el inglés **en**, si desea adicionar otro idioma, basta con que lo cree en la carpeta **languages**

Para habilitar o desabilitar los iconos de mantenimiento, utilice la siguiente función:

FUNCIÓN:

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

PARÁMETROS:

bolAdd : true

true = Habilita el sistema de adición de registros del grid

false = Desabilita el sistema de adición de registros del grid

bolUpd : true

true = Habilita el sistema de actualización de registros del grid

false = Desabilita el sistema de actualización de registros del grid

bolDel : true

true = Habilita el sistema de borrado de registros del grid

false = Desabilita el sistema de borrado de registros del grid

bolChk : true

true = Habilita el sistema de visualización de registros del grid

false = Desabilita el sistema de visualización de registros del grid

intColumn : -1

Define la posición (columna) en la cual se desea mostrar los iconos de mantenimiento (-1 indica al final del grid)

strColumnName: Column Title

Llene este campo, si desea mostrar un título en la columna de iconos

EJEMPLO DE USO:

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

Para habilitar o desabilitar las opciones de exportar, utilice la siguiente función:

FUNCIÓN:

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

PARÁMETROS:

bolExportsheet: true

true = Habilita la opción de exportar a Hoja de cálculo (XLS)

false = Desabilita la opción de exportar a Hoja de cálculo (XLS)

bolExportCSV : true

true = Habilita la opción de exportar a archivo separado por comas (CSV)

false = Desabilita la opción de exportar a archivo separado por comas (CSV)

bolExportXML : true

true = Habilita la opción de exportar a XML

false = Desabilita la opción de exportar a XML

bolPrinter : true

true = Habilita la opción de imprimir

false = Desabilita la opción de imprimir

bolExportPDF : true

true = Habilita la opción de exportar a PDF

false = Desabilita la opción de exportar a PDF

pdfOrientation : P

Selecciona la orientación de la página al exportar a PDF

(P) = Vertical

(L) = Horizontal

EJEMPLO DE USO:

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

Si desea tener una columna con casillas de verificación para selección múltiple, utilice la siguiente función:

FUNCIÓN:

checkable(status);

PARÁMETROS:

status : false

true = Habilita la columna de casillas de verificación

false = Desabilita la columna de casillas de verificación

EJEMPLO DE USO:

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

Para definir un título en el grid, utilice:

FUNCIÓN:

TituloGrid('strTitle');

PARÁMETROS:

strTitle : Titulo del grid

EJEMPLO DE USO:

```
$objGrid -> TituloGrid('Titulo del grid');
```

Para definir un pie de página en el grid, utilice:

FUNCIÓN:

FooterGrid('strFooter');

PARÁMETROS:

strFooter : Pie de página del Grid, © 2008 Gurú Sistemas

EJEMPLO DE USO:

```
$objGrid -> FooterGrid('Pie de página del Grid, © 2008 Gurú Sistemas');
```

Para controlar la cantidad de registros a visualizar por cada página:

FUNCIÓN:

datarows(intLines);

PARÁMETROS:

intLines : 15

Cantidad de registros por página

EJEMPLO DE USO:

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

Si desea definir la cantidad de páginas que se mostrarán antes de resumir con ... puede hacerlo cambiando la cantidad con esta funcion:

FUNCIÓN:

linksperpage('amount');

PARÁMETROS:

amount : 4

Escriba la cantidad de links que desea visualizar

EJEMPLO DE USO:

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

IMPORTANTE: No defina cantidades muy grandes ya que pueden distorsionar la estructura del Grid

Si define linksperpage = 5, verá algo como:
1 2 3 4 5 ... 15 16 17 18 19 **20** 21 22 23 24 25 ... 45 46 47 48 49

Para cambiar el formato de paginación, utilice:

FUNCIÓN:

paginationmode('pgm',inTable);

PARÁMETROS:

pgm	:	mixed

Defina el tipo de paginación, existen 3 valores disponibles:

links = Genera un a lista de números de página que indican el número de página al que se desea ir (Recomendado tablas con no mas de 20 páginas)

select = Genera un menú desplegable que permite elegir el número de página a la que desea ir

mixed = Genera un listado combinando los métodos links y select (valor predeterminado)

input = Crea un cuadro de texto para que el usuario escriba el número de página

inTable	:	false

este parámetro ya no tiene ninguna funcion, esta aqui por compatibilidad con versiones anteriores

EJEMPLO DE USO:

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

Puede definir un código de seguridad (palabra mágica) que ayudará a phpMyDataGrid a ser más seguro:

FUNCIÓN:

salt('code');

PARÁMETROS:

code	:	salt&pepper

EJEMPLO DE USO:

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

Defina la cantidad de dígitos decimales que desea mostrar en los campos de tipo numérico:

FUNCIÓN:

decimalDigits('amount');

PARÁMETROS:

amount	:	2

EJEMPLO DE USO:

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

Puede definir el caracter que desea utilizar como separador decimal:

FUNCIÓN:

decimalPoint('char');

PARÁMETROS:

char	:	.

EJEMPLO DE USO:

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

La edición en línea puede estar: desactivada, activarse con un solo click y grabar al salir del campo, o solicitar confirmación de grabación, esto puede ser definido desde la función:

FUNCIÓN:

ajax('style', clicks);

PARÁMETROS:

style	:	none

none = Deshabilita la edición en línea

default = Habilita la edición en línea con confirmación de grabación

silent = Habilita la edición en línea con grabación automática

clicks	:	1

Define la cantidad de clicks necesarios para activar la edición en línea.

1 - Click sencillo

2 - Doble click

EJEMPLO DE USO:

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

Si desea diferenciar en pantalla los valores que han sido modificados via AJAX, puede usar esta función:

FUNCIÓN:

AjaxChanged('strColor');

PARÁMETROS:

strColor	:	#900

Debe ser un color hexadecimal válido

EJEMPLO DE USO:

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

Si por algún motivo requiere controlar si se está realizando una llamada AJAX a la página puede usar esta función:

FUNCIÓN:

isAjaxRequest();

EJEMPLO DE USO:

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

IMPORTANTE: Puede usar la función para determinar si la solicitud a la página fue una llamada AJAX

```
if ($objGrid -> isAjaxRequest()){
    echo 'esto se hace en el proceso AJAX';

} else {
    echo 'esta es una solicitud directa';

}
```

Usted puede personalizar las acciones que realizarán los botones de mantenimiento:

FUNCIÓN:

setAction('button', 'event');

PARÁMETROS:

button	:	add
event	:	javascript_function()

EJEMPLO DE USO:

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

IMPORTANTE: Consideraciones a tener en cuenta a la hora de asignar nuevas acciones a los botones:

Botón Adicionar = Ninguna en especial

Ejemplo: $objGrid -> setAction('add','nuevo_adicionar()');

Botón Editar = la función deberá contener dos parámetros, los cuales deben ir delimitados asi: (\"%s\",\"%s\");

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

Botón Borrar = la función deberá contener dos parámetros, los cuales deben ir delimitados asi: (\"%s\",\"%s\");

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

Botón Buscar = Ninguna en especial

Ejemplo: $objGrid -> setAction('search','nuevo_buscar()');

Botón Ver = la función deberá contener dos parámetros, los cuales deben ir delimitados asi: (\"%s\",\"%s\");

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

Es necesario definir la tabla sobre la cual trabajará el grid:

FUNCIÓN:

tabla('strTable');

PARÁMETROS:

strTable	:	employees

EJEMPLO DE USO:

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

Puede definir una condición del tipo WHERE para filtrar y mostrar solo los registros que cumplan con una condición:

FUNCIÓN:

where('strWhere');

PARÁMETROS:

strWhere	:	active = 1

EJEMPLO DE USO:

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

Si necesita agrupar registros por algún campo, o campos, puede usar la función:

FUNCIÓN:

groupby('strGroup');

PARÁMETROS:

strGroup	:	status

EJEMPLO DE USO:

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

IMPORTANTE: Tenga en cuenta que al agrupar datos, las funcionalidades de mantenimiento (Adicionar, Editar, Borrar, edición en línea), no funcionarán correctamente, por lo tanto se recomienda deshabilitarlas, o si es necesario, aplicar sus propios procesos de mantenimiento.

Puede definir una condición del tipo HAVING para filtrar y mostrar solo los registros que cumplan con una condición:

FUNCIÓN:

having('strHaving');

PARÁMETROS:

strHaving	:	active = 1

EJEMPLO DE USO:

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

Defina los campos por los cuales desea ordenar los registros:

FUNCIÓN:

orderby('fields','style');

PARÁMETROS:

fields	:	name,lastname

Lista de campos por los que desea ordenar la salida, separada por comas

style : asc,desc

Defina el tipo de ordenamiento para cada campo **ASC** o **DESC**, en caso de dejar en blanco, se usará automáticamente **ASC**

EJEMPLO DE USO:

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

La versión professional cuenta con una característica de ordenamiento manual, la cual sirve, por ejemplo para definir el orden en el que aparecerán los productos de una página:

FUNCIÓN:

setorderarrows('field');

PARÁMETROS:

field : consecutivo

El campo de consecutivo NO debería ser el ID

EJEMPLO DE USO:

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

IMPORTANTE: El campo de ordenamiento debe ser consecutivo y no debe ser autoincrementable, para los ejemplos prácticos que hemos desarrollado, siempre el consecutivo es inicialmente igual al ID autoincrementable del registro.

Por defecto, phpMyDataGrid habilita las flechas de ordenamiento en los títulos de todas las columnas, si desea desactivar esta característica utilice la siguiente función:

FUNCIÓN:

noorderarrows();

EJEMPLO DE USO:

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

IMPORTANTE: Si desea deshabilitar el ordenamiento de solo unas cuantas columnas del grid, NO utilice esta función, en su lugar utilice **chField** especificando los parámetros necesarios.

phpMyDataGrid Genera automáticamente las consultas SQL basado en la información suministrada, pero en algunas ocasiones es necesario generar consultas avanzadas que requieren que el programador las defina manualmente, para eso puede usar esta función:

FUNCIÓN:

sqlstatement('strSQL','strCount');

PARÁMETROS:

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

EJEMPLO DE USO:

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

IMPORTANTE: Las consultas SQL manuales no deben incluir las sentencias WHERE, GROUP u ORDER, estas deben ser definidas directamente desde las funciones disponibles.
También debe tener en cuenta que es muy importante que todos los nombres de campos que utilice en el grid, se encuentren definidos en la consulta SQL

Para el programador es indispensable poder hacer seguimiento al comportamiento de los scripts, esta función permite al programador recibir correos electrónicos con 'eventuales' errores SQL que se puedan generar:

FUNCIÓN:

reportSQLErrorsTo('strMail', 'bolShow');

PARÁMETROS:

strMail	:	developer@thecompanymail.com
		Escriba la dirección de e-mail del programador
bolShow	:	false
		true = Muestra los errores SQL en pantalla durante la ejecución (Recomendado en desarrollo)
		false = Oculta cualquier error SQL generado (Recomendado en entornos productivos)

EJEMPLO DE USO:

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

IMPORTANTE: Esta función hace uso de la sentencia **mail()** de php, por lo tanto para que funcione correctamente, deberá estar configurada y activa

Para operaciones de mantenimiento es necesario definir un campo clave, utilice la función:

FUNCIÓN:

keyfield('strField');

PARÁMETROS:

strField : id

EJEMPLO DE USO:

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

phpMyDataGrid le permite hacer búsquedas, puede definir los campos por los que desea buscar información usando la siguiente función:

FUNCIÓN:

searchby('listoffields');

PARÁMETROS:

listoffields : firstname,lastname,date:select

Puede adicionar la instrucción **:SELECT** al nombre del campo para mostrar un menú desplegable con los posibles valores de búsqueda.

EJEMPLO DE USO:

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

Si desea generar un enlace para controlar el proceso de 'desfiltrar' después de realizar búsquedas, puede usar esta función:

FUNCIÓN:

getResetSearch(icon);

PARÁMETROS:

icon : false

Define si se devuelve solo texto o una imagen representativa, valor predeterminado: false

EJEMPLO DE USO:

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

Una de las funciones principales es FormatColumn, con esta función usted definirá las características de cada uno de los campos que mostrará en grid:

FUNCIÓN:

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

PARÁMETROS:

strfieldName : lastname

Nombre del campo en la tabla

strHeader : Apellido

Título de la columna

fieldWidth : 0

Solo usado en campos de tipo **textarea**, identifica la cantidad de líneas que contendrá el textarea

maxlength : 20

Longitúd máxima de caracteres a aceptar en el campo

inputtype : 0

Tipo de campo

0 = Campo Normal

1 = Campo de solo lectura

2 = Campo Oculto

3 = Imágen, cálculo o enlace sin relación con campo en la tabla

4 = Imágen, cálculo o enlace relacionado con un campo en la tabla

columnwidth : 80

Ancho de la columna (En píxeles)

align : center

Alineación del texto en la columna

center = Centrado (Valor predeterminado)

left = Ajuste del texto a la Izquierda

right = Ajuste del campo a la derecha

Mask : text

Enmascaramiento para el campo

text = Campo normal de texto (Valor predeterminado)

textarea = Región de edición de texto (puede tener un mayor área que los campos tipo **text**)

image = Muestra una imágen, puede ser relacionada con un campo o fija. (ver ejemplos)

imagelink = Muestra una imágen con enlace, puede ser relacionada a un campo o fija. (ver ejemplos)

number = Campo numérico

money = Campo numérico con formato de moneda, forma de uso: money:signo, ejemplo **money:$ money:£**

date = Campo tipo fecha, forma de uso: date:formato:separador, ejemplo **date:dmy:/ date:ymd:-**

datetime = Campo tipo fecha y hora, forma de uso: datetime:formatofecha:separador:formatohora,separador, ejemplo **datetime:datetime:mdy:/:His,: o datetime:mdy:/:hisa,:**

link = Campo con enlace. ver ejemplos

password = Campo tipo contraseña (Protegido con asteriscos)

calc = Campo calculado. ver ejemplos

scalc = Campo calculado que almacena el valor del cálculo, ver ejemplos

bool = Campo Booleano, genera una casilla de verificación y almacena 0 si no esta checkeada y 1 si esta chequeada

check = Igual al campo tipo **bool**

select = Campo con menú de opciones, las opciones pueden ser manualmente definidas, o dinámicamente desde otra tabla de la base de datos.

0 = Campo numérico sin decimales

1 = Campo numérico con 1 decimales

2 = Campo numérico con 2 decimales

3 = Campo numérico con 3 decimales

4 = Campo numérico con 4 decimales

integer = Campo numérico sin decimales

related = Busca un registro coincidente en otra tabla y muestra la relación

array() - conditional = Muestra una respuesta analizando una serie de condiciones dadas. (ver ejemplos)

default :

Valor predeterminado del campo (se usa solo en la opción de adicionar nuevos registros)

cutChar :

Util en campos de tipo **textarea** que contengan mucha información, con esta opción solo mostrará los primeros **X** caracteres mientras visualice la información en el grid, para ver la información completa puede usar la opcion **Ver registro**

EJEMPLO DE USO:

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

Puede definir características adicionales a cada una de las columnas, para esto basta con usar:

FUNCIÓN:

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

PARÁMETROS:

strfieldName : nombrecampo

Nombre del campo

permissions : N+

N+ = Visualizar campo al adicionar registro

N- = Ocultar campo al adicionar registro

E+ = Visualizar campo al editar registros

E- = Ocultar campo al editar registros

V+ = Visualizar campo al ver registro

V- = Ocultarcampo al ver registro

R = Quitar flechas de ordenamiento a este campo

U = Si es un campo de tipo **image** permita cargar fotos en el campo

M = Habilita la opción de cargar imágenes sobre imágenes ya exitentes

overwrite : false

Define si las definiciones realizadas anteriormente al campo deberán ser borradas o acumulativas

EJEMPLO DE USO:

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

IMPORTANTE: Tenga en cuenta que si desea asignar varios modificadores al mismo campo, debe hacerlo en la misma solicitud, por ejemplo:

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

Define el ancho de los campos de texto al adicionar/editar:

FUNCIÓN:

setInputWidth('field','width');

PARÁMETROS:

field	:	Nombre campo
width	:	300px

EJEMPLO DE USO:

```
$objGrid -> setInputWidth('Nombre campo','300px');
```

Si desea realizar validaciones javascript sobre el campo, utilice la siguiente función:

FUNCIÓN:

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

PARÁMETROS:

strField	:	firstname Nombre del campo
strValidation	:	this.value.length>=8 Código JavaScript para validar el campo, (Puede tambien llamar una función JS)
strErrorMessage	:	El nombre debe contener más de 7 caracteres Mensaje de error cuando no se cumpla la condición
strDisplayMessage:		Por favor escriba el nombre del cliente Mensaje descriptivo de la información que deberá digitar el usuario

EJEMPLO DE USO:

```
$objGrid -> jsValidate('firstname', 'this.value.length>=8', 'El nombre debe contener más de 7 caracteres', 'Por favor escriba el nombre del cliente');
```

IMPORTANTE: La validación Javascript se usará en los procesos **Nuevo** y **Edición**, y también en el proceso de edición en línea, cabe tener en cuenta que las validaciones solo se realizan sobre el campo activo, no se pueden realizar operaciones con los otros campos de la tabla.

Si no desea validar la entrada de datos, pero desea dar indicaciones al usuario sobre la información que deberá digitar, utilice:

FUNCIÓN:

fldComment('strField', 'strDisplayMessage');

PARÁMETROS:

strField	:	lastname
		Nombre del campo
strDisplayMessage:		Escriba el apellido del cliente
		Mensaje de ayuda al usuario

EJEMPLO DE USO:

```
$objGrid -> fldComment('lastname', 'Escriba el apellido del cliente');
```

IMPORTANTE: Los mensajes de ayuda no serán mostrados durante el proceso de edición en línea, sin embargo, se visualizarán en los procesos normales de crear y editar registros.

Si en su lista de campos existe alguno de tipo "date" y desea que phpMyDataGrid utilice un calendario (datepicker) para la selección de fechas, utilice:

FUNCIÓN:

useCalendar(bolCalendar);

PARÁMETROS:

bolCalendar	:	true

EJEMPLO DE USO:

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

Para totalizar columnas, utilice:

FUNCIÓN:

total('fields');

PARÁMETROS:

fields	:	salary,days,total
		Lista de campos a totalizar separados por coma

EJEMPLO DE USO:

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

IMPORTANTE: Recuerde que ahora también puede totalizar columnas calculadas

Define el tamaño (ancho y alto) para mostrar imágenes en un campo:

FUNCIÓN:

setImageSize('campo','ancho','alto');

PARÁMETROS:

campo	:	NombreCampo Nombre del campo
ancho	:	95 Ancho en px (Solo el valor numérico)
alto	:	127 Height en px (Solo el valor numérico

EJEMPLO DE USO:

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

IMPORTANTE: Esta función no cambiará el tamaño de las imágenes almacenadas, sólo cambiará el tamaño de la imagen en la salida del navegador.

Permite definir un formato condicional para CELDAS que cumplan una condición:

FUNCIÓN:

addCellStyle('campo','condicion', 'estilo');

PARÁMETROS:

campo	:	NombreCampo Nombre del campo
condicion	:	[\"activo\"]==1 Cadena, condición.
estilo	:	nombreClaseCSS Nombre de la clase CSS a ejecutar

EJEMPLO DE USO:

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

IMPORTANTE: Es posible usar cualquier campo que se encuentre en uso dentro del grid, basta con escribir el nombre del campo entre [' ... '].

Permite definir un formato condicional para FILAS que cumplan una condición:

FUNCIÓN:

addRowStyle('condicion', 'estilo');

PARÁMETROS:

condicion	:	[\"activo\"]==1
		Cadena, condición.
estilo	:	nombreClaseCSS
		Nombre de la clase CSS a ejecutar

EJEMPLO DE USO:

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

IMPORTANTE: Es posible usar cualquier campo que se encuentre en uso dentro del grid, basta con escribir el nombre del campo entre [' ... '].

Define la ruta para imágenes relacionadas a un skin:

FUNCIÓN:

skinimages('skin', 'ruta');

PARÁMETROS:

skin	:	NombreSkin
		Nombre del skin
ruta	:	skins/%s/icons/
		Ruta de las imágenes

EJEMPLO DE USO:

```
$objGrid -> NombreSkinimages('NombreSkin', 'skins/%s/icons/');
```

IMPORTANTE: El %s en la ruta será reemplazado con el nombre del skin, si el %s no existe, la ruta sera tomada tal como esté.

Modifica el valor a almacenar especificado en la edición en línea:

FUNCIÓN:

setNewInlineData('newData');

PARÁMETROS:

newData : nuevosDatos

Nuevos datos al momento de guardar una edición en línea

EJEMPLO DE USO:

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

Define una función JS a ejecutar despues de actualizar en línea, puede tener los parámetros (idkey,field,newtext,oldtext):

FUNCIÓN:

onAjaxUpdate('js');

PARÁMETROS:

js	:	javascriptFunction(idkey,field,newtext,oldtext) Nombre de la función javascript que desea ejecutar al finalizar la grabación inline

EJEMPLO DE USO:

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

Devuelve el valor de la acción AJAX que se está ejecutando:

FUNCIÓN:

getAjaxID();

EJEMPLO DE USO:

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

Devuelve los códigos de las casillas de verificación seleccionadas:

FUNCIÓN:

getCheckedBoxes();

EJEMPLO DE USO:

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

Devuelve verdadero si la acción AJAX que se esta realizando es Adición de datos:

FUNCIÓN:

isadding();

EJEMPLO DE USO:

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

•

Devuelve verdadero si se esta grabando una edición en línea:

FUNCIÓN:

isOnlineEdition();

EJEMPLO DE USO:

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

•

Devuelve la información que está siendo editada en línea:

FUNCIÓN:

getEditedData();

EJEMPLO DE USO:

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

•

Muestra una nueva imagen basándose en los parametros recibidos:

FUNCIÓN:

changeImage();

EJEMPLO DE USO:

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

•

Adiciona un separador a la barra de botones:

FUNCIÓN:

addSeparator();

EJEMPLO DE USO:

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

IMPORTANTE: Esta función solo es válida si la propiedad **toolbar** es verdadera

Adiciona opciones a la barra de botones:

FUNCIÓN:

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

PARÁMETROS:

img	:	/path/to/image.png
		Ruta y nombre del icono a mostrar en la barra de botones
action	:	javascriptFunction()
		Nombre de la función javascript que desea ejecutar
message	:	Message In Toolbar
		Mensaje que acompañará al icono

EJEMPLO DE USO:

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

IMPORTANTE: Esta función solo es válida si la propiedad **toolbar** es verdadera

Adiciona un menú desplegable a la barra de botones:

FUNCIÓN:

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

PARÁMETROS:

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()
		Nombre de la función javascript que desea ejecutar
message	:	Message In Toolbar
		Mensaje que acompañará al icono

EJEMPLO DE USO:

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

IMPORTANTE: Esta función solo es válida si la propiedad **toolbar** es verdadera

Esta función está disponible para la compatibilidad con versiones anteriores, no se recomienda para ser utilizada, es ideal que se inserte el JS y CSS manualmente o usando la función set_DG_Header:

FUNCIÓN:

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

PARÁMETROS:

phpScriptFile	:	sample.php Nombre del script que se esta ejecutando
jsFile	:	js/dgscripts.js Ruta y nombre del archivo de scripts del datagrid
cssFile	:	css/dgstyle.css Ruta y nombre del archivo de estilos del datagrid
jsCalFile	:	js/dgcalendar.js Ruta y nombre del archivo que contiene el calendario Datepicker
cssCalFile	:	css/dgcalendar.css Ruta y nombre del archivo de estilos del calendario
jsmmenu	:	mmscripts.js Ruta y nombre del archivo que controla el menú contextual en el grid

EJEMPLO DE USO:

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

IMPORTANTE: Para mayor flexibilidad y facilidad del programador se recomienda NO usar esta función, en su lugar inserte los archivos CSS y JS como normalmente lo haria con otros archivos de este tipo.

Finalmente, realice el llamado a la función que renderizará el grid basándose en toda la configuración anteriormente suministrada:

FUNCIÓN:

grid();

EJEMPLO DE USO:

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

Si necesita pasar parámetros a una consulta SQL se recomienda utilizar esta función la cual analizará el contenido de la variable, y la modificará de ser necesario para evitar inyecciones SQL.:

FUNCIÓN:

magic_quote($variable);

EJEMPLO DE USO:

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

IMPORTANTE: Tenga en cuenta que esta función no hace parte de la classe del datagrid, por lo cual, no se debe llamar como un objeto sino como función independiente!

Se recomienda a la hora de incluir los archivos JS y CSS hacerlo usando esta función, ver ejemplos para mayor claridad:

FUNCIÓN:

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

EJEMPLO DE USO:

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

IMPORTANTE: Tenga en cuenta que esta función no hace parte de la classe del datagrid, por lo cual, no se debe llamar como un objeto sino como función independiente!