DataGrid tool

This tool has been created for selecting data, displaying it as a grid, and perform routine operations on it. It allows paging through results, editing, deleting and inserting new data. DataGrid reads its interface from a template file with the help of Template class and utilizes DB class to access database.

Fields of DataGrid are derived from DataGridColumn so, when you need displaying complex data with an extra-ordinary interface, you just need to derive a new class and implement Init, Render, RenderEdit, and GetValue functions. You can also register toolbar commands which has their own callback functions.

DataGrid allows to add command columns, when you click on that commands, it will call the callback function which you have specified and will pass the KeyField value, in turn allowing you to perform any needed operations.

In addition to it, you register your own toolbar commands very easily. See DataGrid::RegisterToolbarCommand() for more information.

If you manipulate a complex data you can handle OnInsert, OnEdit, OnUpdate, OnDelete events and perform your own operation, then let the DataGrid to continue its procedure or not. See `OnInsertCall(), OnEditCall(), OnUpdateCall(), OnDeleteCall()` methods.

DataGrid is almost fully customizable. You can customize its functionality, its interface, its columns and commands, and create a new column types etc. For example, you can select only identifier from necessary table, create a special Column type which will using this ID display for you News, Archive materials, etc. As a result you benefit from its Paging functionality at least.

DataGrid class

This is the main class to display data and act as a grid. This class utilized multiple global functions, DB class, Template class, DataGridColumn and its derived type, DataGridToolbarCommand classes.

Its gathers data, displays, reacts against multiple post back commands. DataGrid saves its state across multiple post backs

Properties

`$Template`	An instance of Template class. This template carries the design interface of DataGrid. If you want to use a different template construct the class with a different template file name, or after construction provide a ready to use template instance
`$AllowAutoIncrement`	This property is needed for insert operation. If you have a key field that needs to be atomically incremented by the DataGrid set this property to true (default value). When this is true, and the key field if not displayed to users, DataGrid automatically increments this field and Creates the insert SQL. If you set it to false, and when this field is not displayed to users, data grid will not automatically increment the key field. If the key field is displayed to users, this property doesn't play any role
`$AllowCancelSort`	If this property is set to true, "Cancel Sort" toolbar command will be rendered to the user. See the `CancelSorting()` function for more information
`$AllowDelete`	If this is set to true, the "Delete" link will be rendered next to each row.
`$AllowDeleteSelected`	If this is set to true, the "Delete Selected" button will be rendered on the toolbar to enable users to delete selected rows all by one click
`$AllowEdit`	If this is set to true, an "Edit" link will be rendered next to each row to enable users edit that row
`$AllowInsert`	If this is set to true, a special row will be rendered to user at the bottom of table in edit more, to enable users to insert new rows. If you are working with a complex data and wont to handle insert commands, see the `OnInsertCall()` function

$AllowPage	If this is set to true, data grid will automatically page the results of select statement. And a pager will be rendered to users to enable them to switch through pages	
$AllowSelect	If this property is set to true checkboxes will be rendered to the left of each row allowing users to select rows	

$AllowSort	If this is set to true, Data Grid will enable users to automatically sort the fields. If you want to enable only particular rows to be sorted, set this property to true, and disable CanSort property of only those columns	`$d = new DataGrid();`
		`.....`
		`$d->AllowSort = true;`
		`$d->Columns[2]->CanSort = false;`

$Columns	It holds the array of columns of this Data Grid. All of these columns are instances of classes which are derived from DataGridColumn. See AddTextBoxColumn()method for more information.	
$CommandsRight	If this is set to true (default) Edit, Update, Cancel and Delete commands will be placed on the right hand side. Otherwise they will be placed on the left hand side	
$CurrentPage	If paging is enabled you can set your start page. If you set CurrentPage to 2 (zero based) data grid will display the third page. But the current page can be changed by the User when he clicks the pager buttons	
$KeyField	This is the name of key field of table. If you want to enable editing , deleting and inserting operations, you must specify the this property. Default value is “id”	
$PagerBottom	If this set to true, the pager will be placed at the bottom side of DataGrid. At the same time $PagerBottom and $PagerTop properties can be true or false.	
$PagerCount	When there are more than 20 pages DataGrid should not display all of those pager links to the user starting from 1 to 20. This might harm your design. By default the PagerCount property is set to 10. It will display paging links from 1 to 10 and, then display special “...” link to switch to next series. If you are on the 16th page it will display “...” link before 11th page, but not at the end. Set this property to any value you want the pager links to be displayed at a time	

$PagerSizes	This is an array of page sizes. A drop down list will be displayed to enable users to change page size. By default it displays 5, 10, 20, ..., 100.
	If you want to change this array create a new one and assign to this property as in the sample on the right
		`$d = new DataGrid();`
		`.....`
		`$d->PagerSizes = array ("1","2","3","4","5");`

$PagerTop	If this set to true, the pager will be placed at the top side of DataGrid. At the same time $PagerBottom and $PagerTop properties can be true or false.	
$PageSize	Specifies the number of rows to be displayed at a time. You can enable to	

	automatically change the page size by setting `$ShowPageSizer` to true (default value) or to false not to let them to change the page size
`$SelectSQL`	Specify your selection sql if you want to manipulate a complex data. See `PrepareSQL()` method for more information.
`$ShowCaption`	Set to true (default value) to display column captions. Or set to false to hide column captions
`$ShowPageSizer`	When set to true, displays the drop down list on the pager, to enable users to change the page size
`$TableName`	When editing a single table, you must set this property to the table name you are editing. This will enable Edit, Delete, Update, Insert commands to work properly.
`$WhereClause`	If this property is not NULL or empty this will be added to the end of Selection SQL. So, you can create a filter for your data grid and place this filter on the toolbar

Methods

`DataGrid ($template_file = NULL)`

Costructor. Constructs a new DataGrid object.
`$template_file` is the name of file from which the interface will be loaded. If it is null, DataGrid will load default datagrid.tpl file.

`OnInsertCall($callBack)`

Registers an Insert command handler. When users clicks the insert button this `$callBack` function will be loaded and the data will be passed into this function. `$AllowInsert` must be set to true to enable this feature
Sample:

```
$d = new DataGrid();
.....
$d->AllowInsert = true;

$d->OnInsertCall ("on_insert");
......
function on_insert($values, $cancel)
{
               echo "Inserting:<br>";
               print_r($values);
               $cancel = true;  // do not allow data grid to insert
}
```

Note: If in this function `$cancel` is set true, the DataGrid will not perform its own insert operation, if not, data grid will itself perform the insert operation

---

`OnUpdateCall($callBack)`

Registers an Update command handler. When users clicks the update button of an editable row, this `$callBack` function will be called and the data will be passed into this function. `$AllowEdit` must be set to true to enable this feature

```
```

```
$d = new DataGrid();
.....
$d->AllowEdit = true;

$d->OnUpdateCall ("on_update");

......

function on_update($id, $new_values, $cancel)
{
               echo "On update called<br>ID=$id";
               echo "<br>New Values: ";
               print_r($new_values);
               $cancel = false;  // let the data grid to update rows by itself
}
```

Note: If in this function `$cancel` is set true, the DataGrid will not perform its own update operation, if not, data grid will itself perform the update operation

---

OnDeleteCall($callBack)

Registers a Delete command handler. When users clicks the delete button of an editable row, this $callBack function will be called and the data will be passed into this function. `$AllowDelete` must be set to true to enable this feature.

```
$d = new DataGrid();
.....
$d->AllowDelete = true;

$d->OnDeleteCall ("on_delete");

......

function on_delete($id, $cancel)
{
          echo "Deleting $id<br>";
          $cancel = true; // let the data grid to update rows by itself
     }
```

Note: If in this function `$cancel` is set true, the DataGrid will not perform its own delete operation, if not, data grid will itself perform the delete operation

---

OnEditCall($callBack)

Registers an Edit command handler. When users clicks the edit button of an editable row, this `$callBack` function will be called and the data will be passed into this function. `$AllowEdit` must be set to true to enable this feature.

```
$d = new DataGrid();
.....
$d->AllowEdit = true;

$d->OnEditCall ("on_edit");

......

function on_edit($id, $cancel)
{
          echo "editing $id<br>";
          $cancel = false; // let the data grid to get into edit mode
     }
```

Note: If in this function `$cancel` is set true, the DataGrid will not perform its own edit operation, if not, data grid will itself perform the edit operation

---

DeleteSelected($values)

This is a protected event handler. This function is automatically called when the “Delete Selected” button is clicked on the toolbar. $AllowDeleteSelected property must be set to true enable this feature

---

CancelSorting()

Call this function to cancel sorting. When $AllowSort is set to true, DataGrid automatically sorts the columns when user clicks on them, for the first click data is sorted ascending, for the second click data is sorted descending. The sorted column is highlighted. This function is a protected event handler of “Cancel Sorting” toolbar button. $AllowCancelSort property must be set to true to enable this feature.

RegisterToolbarCommand($commandName, $template, $callBack)

This function allows developers to register new toolbar commands. This commands are displayed on the toolbar of datagrid and when they are fired by the user your callback function is called and the selected rows are passed into your function. “Delete Selected” and “Cancel Sorting” commands are the internal toolbar commands of DataGrid

$commandName is the name of this command. This is to identify your command.
$template is an instance of Template class. This parameter must contain your buttons HTML to be rendered. It should also contain “<:link:>” tag. This tag will be automatically replaced with post back event raiser.
$callBack is the name of call back function to be called when this button is clicked

```
$tpl = new Template();
$tpl->SetContent("<a href=\"<:link:>\" class=\"DGTBButton\">Get Selected</a> ");

$grid->RegisterToolbarCommand("getSelected", $tpl, "get_selected");

function get_selected($values)
{
        echo "Selected Values are<br>";
        print_r ($values);
}
```

Note, if no Toolbar command is registered the toolbar is not rendered. To turn of toolbar if you don’t register any toolbar commands, set $AllowCancelSort and $AllowDeleteSelected properties to false

---

AddTextBoxColumn ($field_name, $caption, $canSort = true)

$field_name is the name of field in the database.
$caption is the user friendly name of this column to be displayed to users.
$canSort says if this column can cause Sort command

This function adds a new field to the data grid. This function automatically creates a new instance of DataGridTextBoxColumn and initializes it with the provided parameters, then adds it to the array of columns. Columns property contains all the Columns of DataGrid. These columns are derived from DataGridColumn class. You can also create your own classes to display a different kind of data a different way. See DataGridColumn, DataGridTextBoxColumn, DataGridCommandColumn, DataGridComboColumn classes for more information.
Note, you can add columns to this data grids Columns collection manually, after creating a new Column instance. See $Columns property for more information

Sample

```
$grid = new DataGrid();
$grid->TableName = "MyTable";
$grid->AddTextBoxColumn("original", "Original");
```

Combo box column sample

```
$column = new DataGridComboColumn("lang_id", "Language");
$column->AddOption ("az", "Azerbaijan");
$column->AddOption ("en", "English");
$column->AddOption ("ru", "Russian");
$grid->Columns[] = $column;

// OR
$grid->AddComboColumn("lang_id", "Language", array("az"=>"Azerbaijan", "en"=>"English",
"ru"=>"Russian"));
```

Command columns sample

```
$tpl = new Template();
$tpl->SetContent("<a href=\"<:link:>\" class=\"myclass\">Select</a> ");
$col = new DataGridCommandColumn("original", "Display As", $tpl, "cmd_name",
                                 "my_column_clicked");
$grid->Columns[] = $col;
function my_column_clicked ($value)
{
        echo "You clicked $value";
}
```

AddComboColumn ($field_name, $caption, $options)

`$field_name` is the name of field in the database.
`$caption` is the user friendly name of this column to be displayed to users.
`$options` an array of options to be added for the drop down box

This function adds a new column to the data grid. This function automatically creates a new instance of DataGridComboColumn and initializes it with the provided parameters, then adds it to the array of columns.

AddCheckBoxColumn ($field_name, $caption, $true_text = "True", $false_text = "False")

`$field_name` is the name of field in the database.
`$caption` is the user friendly name of this column to be displayed to users.
`$true_text` text to display if the checkbox is checked. (the field contains true or 1)
`$false_text` text to display if the checkbox is not checked. (the field contains NULL, false or 0)

This function automatically creates an instance of DataGridCheckColumn, initializes it and adds it to the list of Columns of DataGrid. See DataGridCheckColumn for more information

---

GetSelectedValues()

You can call this function to get selected values. Values are an array containing key field data of the selected rows. You can register a toolbar command. Then these values are automatically sent to you. If you wonna handle selected values out of data grid you can use this function. Note, `$AllowSelect` property must be set to true to allow users to select rows

---

PrepareSQL($includeLimit = true)

This function is automatically called by Render and other private functions to generate select statement.

$includeLimit tells if the SQL should include paging staments such as limit. The limit statement must be turned off to get the real count of rows to display paging. You can call this function if DataGrid gives any errors and doesn't work. Use this function to see what sql does DataGrid work. To display a very complex data which has been handled by joining multiple tables, you must initialize the $SelectSQL property. This property must nit include ORDER BY or LIMIT statements. They are included by this function. If you edit a single table don't need to specify that URL, because when $SelectSQL property is set, several automatic functions like "update", "delete", are disabled. Below are the examples on how to use datagrid for a single table and a complex result set.

Sample to edit a single table

```
$grid= new DataGrid();
$grid->TableName = "your_table_name";

// add columns
$grid->AddTextBoxColumn("field_name", "Caption for this field");
.....
// specify here if needed
$grid->AllowSort = false;
$grid->AllowPage = true;
$grid->AllowEdit = false;
$grid->AllowDelete = false;
$grid->AllowInsert = false;

// you can set Width for your columns
$grid->Columns[0]->Width = "20%";
$grid->Columns[1]->Width = "80%";

$grid->Render();
```

Sample to edit a complex resultset

```
$grid= new DataGrid();
$grid->TableName = "your_table_name";
$grid->SelectSQL = "SELECT sun_galaxy ... INNER JOIN ... Jupiter :) ... blah blah";

// add columns
$grid->AddTextBoxColumn("field_name", "Caption for this field");
.....
// specify here features if needed, such as allowSort ...
$grid->AllowSort = false;
...
$grid->OnInsertCall(...); // listen for insert, data grid
                          // can not automatically insert
$grid->OnUpdateCall(...)  // also for update and other needed commands
$grid->Render();
```

Render($buffer = false)

This function renders the data grid to users.
If `$buffer` parameter is set to true, it doesn't render the data grid to the user, instead it returns the entire output as a return parameter. Data Grid saves its state across multiple post backs. Even if there are multiple data grids on the same page, even the first grid is sorted by the second column descending, the current page is 5 , and the 7th row is being edited, or 3rd and 4th rows are selected, you can use other data grids or other controls that fire post back events. The first grid won't loose its state. Render function automatically initializes the data grid, processes post back data, fires user functions, prepares select statement, renders the data grid.

Important: DataGrid doesn't render any <form> tags. You must manually write the form tag to the page and manually close that tag. Having a single form tag will multiple data grids to store their state across post backs. If you don't render any form tags, DataGrid won't work

Sample

```
 echo "<form method=\"POST\">"; // start the form

$grid= new DataGrid();
// initialize the grid
$output = $grid->Render(true);
// do several operations
...
// and send data grid to the client
echo $output;
```

PostBackLink ($argument)

This function creates a client side post back command which is specific to this datagrid. When you create new columns types which must derive from the DataGridColumn class, utilize this function to create post back events

```
$link = $grid->PostBackLink("goto=0");
echo "<a href=\"$link\"> Click here go to the first page</a>";
```