

# DataGrid Zend Framework – Manual

<http://www.zfdatagrid.com>

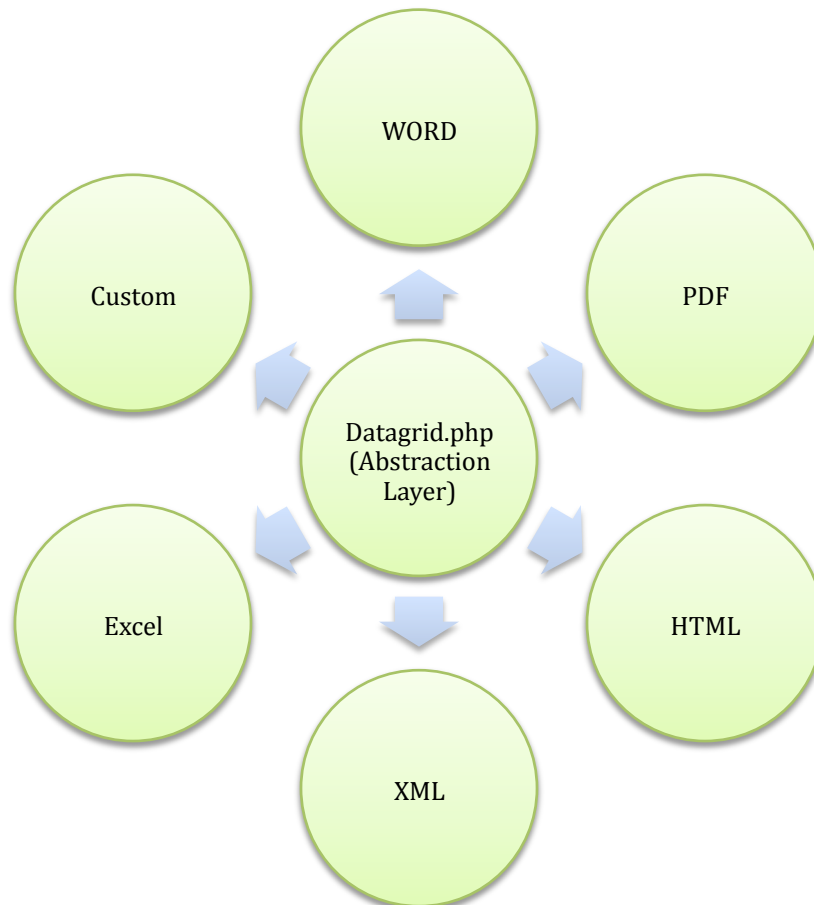
(0.45)

## Table of Contents

<b>DataGrid Zend Framework – Manual</b> .....	<b>1</b>
<b>The Datagrid</b> .....	<b>3</b>
<b>The basics</b> .....	<b>4</b>
<b>Using Zend_Db_Select instance</b> .....	<b>5</b>
<b>Specifying fields and defining options</b> .....	<b>6</b>
<b>Defining field's options</b> .....	<b>6</b>
<b>Titles</b> .....	<b>6</b>
<b>SQL Expressions</b> .....	<b>6</b>
<b>Hiding a field</b> .....	<b>6</b>
<b>Horizontal Row</b> .....	<b>7</b>
<b>Eval</b> .....	<b>7</b>
<b>Class</b> .....	<b>7</b>
<b>searchType</b> .....	<b>7</b>
<b>Format</b> .....	<b>8</b>
<b>Extra Fields</b> .....	<b>9</b>
<b>Joins</b> .....	<b>9</b>
<b>SQL Expressions</b> .....	<b>10</b>
<b>Filters</b> .....	<b>10</b>
<b>Using Arrays, JSON, XML, CSV</b> .....	<b>12</b>
<b>Array</b> .....	<b>12</b>
<b>XML</b> .....	<b>12</b>
<b>CSV</b> .....	<b>13</b>
<b>JSON</b> .....	<b>13</b>
<b>Export</b> .....	<b>13</b>
<b>CRUD</b> .....	<b>14</b>
<b>Templates</b> .....	<b>16</b>
<b>Cache</b> .....	<b>17</b>
<b>Internationalization</b> .....	<b>17</b>

## The Datagrid

The main file `Datagrid.php` will abstract the query result and return arrays corresponding to the parts that complete the Datagrid (titles, sql expressions, filters, pagination, records, ...)



## The basics

You only need three lines of code to deploy a DataGrid. You need to pass the db instance to the constructor.

```
$db = Zend_Registry::get("db");

$grid = new Bvb_Grid_Deploy_Table($db, 'Document Title', 'temp/dir',
array('save', 'download'));

#Temp dir is necessary to build templates |

#array('save', 'download') => save to the dir and download, download only
[array('download')], save only [array('save')]

$grid->from('table');

$this->view->grid = $grid->deploy();
```

Note: If you don't want to use the DB adapter (only array) define the first argument as false.

This piece of code will output something like the table we can see here <http://petala-azul.com/grid/default/site/basic>. It will fetch all fields from the table and create a table with pagination, filters, order and exportation.

Of course you can change this behavior.

Let's imagine you have a table with 12 fields, but you only need 11, you don't want to fetch the user's password. You can tell DataGrid to hide certain fields by just doing this:

```
$grid->hide(array('password'));
```

All other fields will be fetched, except the password one. You can also do other simple things like define the default order with this code

```
$grid->order('id DESC');
```

You can also define the where clause like this

```
$grid->where("id>5");
```

If you want to omit the filters just use this code

```
$grid->noFilters (1);
```

And if you don't want to give user's the ability to order the results add the following code

```
$grid->noOrder(1);
```

There is one more thing you can do on a DataGrid. Defining the limit. But remember. If you use it, the system pagination will be disabled.

```
$grid->limit(5);
```

Or

```
$grid->limit(array(4,10));
```

## Using Zend\_Db\_Select instance

Using a Zend\_Db\_Select instance is very simple.

You just have to passe it instance to the datagrid by doing this:

```
$select = new Zend_Db_Select();  
$select->from('table');  
$select->where('id = ?',2);  
$select->order('name');  
  
$grid->query($select);
```

You can also use the `__toString()` method to return the query. You have to call this method before deploy the table.

## Multiple grids

You can have as many grids as you want on the same page.

However there are a situation that needs your attention. Every grid receives params from the URL, so, if you have two grids on the same page and you set the order for the first, the second will also be ordered.

You can change this behavior defining all other grids as secondary. This will make them ignore the url params.

To do so, you only have to write this line of code

```
$grid->setPrimary(false);
```

Every grid with this defined will ignore all url params.

## Specifying fields and defining options

There are two methods for add fields to the Datagrid. One is more simple, the other is more "object way". **The simple way:** You can specify each field you want to display by using the following method:

```
$grid->addColumn('username'); $grid->addColumn('email');
```

With the above code, only two fields will be fetched and presented to the user. Now, the good part.

## Defining field's options.

### Titles

The most obvious is setting the field title. That can be accomplished using

```
$grid->addColumn('username',array('title'=>"Username Title "));
```

With this code, the title will be " Username Title " (without the quotes).

### SQL Expressions

We may want to show the salaries average, the amount of money we make per year, etc.

```
$grid->addColumn('salary',array(sqlexp=>"avg")); $grid->groupBy('year');
```

Don't forget to define the groupBy, or you will get an error. You can find a result example here <http://www.petala-azul.com/grid/default/site/group> You can also add the HAVING clause to the query by doing this:

```
$grid->addColumn('salary',array(sqlexp=>"avg")); $grid->groupBy('year');  
$grid->having(array('field'=>'salary','operand'=>'>','value'=>'200'))
```

### Hiding a field

```
$grid->addColumn('id',array('hide'=>1));
```

But for what reason would I like to select a field and then not show him on the table? Good question that deserves a better answer. For this:

```
$grid->addColumn('username',array('title'=>"Username",'decorator'=>'<a  
href="http://example.com/user/id/{id}"> {{username}}</a>');
```

When used the index decorator, in options array, the field value will be replaced by the decorator value. Did you noticed the `{{username}}` and `{{id}}` stuff? Yes? Great. You can call the value of any field by putting its name within `{{}}`.

## Horizontal Row

One cool option that you could also use is the horizontal row. When activated the horizontal row will produce something like this <http://www.petala-azul.com/grid/default/site/hrow>.

```
$grid->addColumn('username', array('hRow'=>1));
```

*Please note the case sensitive index.*

## Eval

As most of you already figured out, this code will be passed through the php function eval. Note that even here you can use the field's values using `{{id}}`, or any other field

```
$grid->addColumn('username', array('eval'=>"ucfirst('{{username}}') "));
```

## Class

As the name implies, this will pass to the template the CSS class name to be applied.

```
$grid->addColumn('username', array('class'=>"red"));
```

## searchType

This is used to help filters knowing which expression they will apply when searching. By default the LIKE option is used, but you may want to use the "=" or any other.

```
$grid->addColumn('username', array('searchType'=>"!="));
```

### Options available:

DataGrid	SQL
like	LIKE '%value%'
equal	=
=	=
rlike	LIKE 'value%'
llike	LIKE '%value'
>=	>=
>	>

!=	!=
<>	<>
<=	<=
<	<

## Format

This index will call a class that has been previously added to the DataGrid

```
$grid->addColumn('amount',array('format'=>"number"));
```

Optionally you can set the format index as an array to pass optional arguments to the class constructor

```
$grid->addColumn('amount', array(format=> array('number',array('arg'=>1, 'arg'=>2))));
```

The DataGrid will try to find a class with the name Bvb\_Grid\_Format\_Number and with a method called 'format' You can add your own formats by setting them after instantiate the grid

```
$grid = new Bvb_Grid_Deploy_Table();
$grid->addFormatterDir('Bvb/Grid/Formatter','Bvb_Grid_Formatter');
```

You can add as many as dir as you want. The DataGrid will try to find the class on reverse order. So, if you add this

```
$grid = new Bvb_Grid_Deploy_Table();
$grid->addFormatterDir('Bvb/Grid/Formatter','Bvb_Grid_Formatter');
$grid->addFormatterDir('My/Grid/Formatter','My_Grid_Formatter');
```

The system will look first for a class named My\_Grid\_Formatter\_Number and then for other called Bvb\_Grid\_Formatter\_Number If no match found, the field value will be returned. The object way

```
$grid = $this->grid ( 'table' );# The constructor argument is the field name
$grid->from ( 'Country' ) ->order ( 'Name' ) ->setPagination ( 20 );
$cap = new Bvb_Grid_Column ( 'Name' );
$cap->title ( 'Country (Capital)' )
    ->decorator ( '{{ Name}}' );
$continent = new Bvb_Grid_Column ( 'Continent' );
$continent->title ( 'Continent' );
```

```

$population = new Bvb_Grid_Column ( 'Population' );
$population->title ( 'Population' )
    ->class ( 'width_80' );
$lifeExpectation = new Bvb_Grid_Column ( 'LifeExpectancy' );
$lifeExpectation->title ( 'Life E.' )
    ->class ( 'width_50' );
$grid->addColumn ( $cap, $continent, $population, $lifeExpectation);

```

## Extra Fields

You can add extra fields to the table on the left or on the right. Every extra field is an array. It looks something like this.

```

$right = new Bvb_Grid_ExtraColumns();
$right ->position('right')
    ->name('Right')
        ->decorator("<input class='input_p'type='text' value=
        \"{{LifeExpectancy}}\" size=\"3\"
        name='number[]'> " );
$left = new Bvb_Grid_ExtraColumns();
$left ->position('left')
    ->name('Left')
        ->decorator("<input type='checkbox' name='number[]'>");
$grid->addExtraColumns($right,$left);

```

*The extra fields can't be filtered or ordered.*

## Joins

Working with joins isn't harder then working with simple tables. But there are some aspects that need your attention.

```

$grid->from( "nr_users u INNER JOIN nr_user_visits v ON u.id =
v.user_id");

```

And we MUST declare the tables in use

```

$grid->table(array('v'=>'nr_user_visits','u'=>'nr_users'));

```

And, obviously, when naming fields we need to prefix them with the table name.

```
$grid->addColumn('u.username',array('title'=>'Username'));  
$grid->addColumn('u.email',array('title'=>'Email'));  
$grid->addColumn('v.month',array('title'=>'Month'));  
$grid->addColumn('v.year',array('title'=>'Year'));
```

**INFO:** If two tables have a field with the same name, you must rename the output of one of them

```
$grid->addColumn('u.username',array('title'=>'Username'));  
$grid->addColumn('u.email',array('title'=>'Email'));  
$grid->addColumn('v.username AS user',array('title'=>'User'));  
$grid->addColumn('v.year',array('title'=>'Year'));
```

Example: When setting the order

```
$grid->order ("u.id");
```

When hiding

```
$grid->hide(array('u.id','v.user_id','u.password','u.id'));
```

etc, etc,

## SQL Expressions

The expression result will be presented before the pagination as a new table row (tr) and below to matching field. Using SQL expressions is as easy as this

```
$grid->sqlexp (array('id'=>'COUNT','total'=>'SUM','sales'=>'AVG'));
```

The output: Count all records from id field Sum all records from total field Calculate Average from sales field Can be used all expressions like this SELECT EXP(FIELD) FROM .... Some examples:  
SUM MAX AVG

Note: You can also have the 'sqlexp' when using the array adapter. At this moment only six functions are supported (min, max, avg, sum, product, count)

## Filters

Well, there isn't much to talk about filters, they are lonely people... By default filters are enabled and to disable them we must use the following code (as stated before)

```
$grid->noFilters(1);
```

You can also create a dropdown menu with the values that user is allowed to filter

```
$filters = new Bvb_Grid_Filters();  
$filters->addFilter('email')  
    ->addFilter('username',array('values'=>$values));  
$grid->addFilters($filters);
```

\$values is the result of

```
$values = $db->fetchAll("SELECT username AS name, email AS value FROM  
users");
```

Important: The fields output names must be name and value. We can also apply a style to the filter input form

```
$grid->filters=array('email'=>array('style'=>'width:75px;'),  
'username'=>array('style'=>"width:175px; "));
```

A great option you can get is auto select the distinct values for every field on your table.

```
$grid->filters=array( u.username=> array('distinct'=>array(  
'field'=>'u.id' , 'name'=>'u.username')));
```

The previous code will fetch all distinct u.id values from the query and present them as a select menu. The value will be the 'field' index and the caption will be the name index.

## Filter Tips

You can use special 'commands' to improve filtering in the frontend. You can do search on a field using this \*text\*. Or using >2.

The available options are:

\*text | text\* | \*text\* | = | > | >= | < | <= | <>

## Ajax

Using ajax is incredible simple

All you have to do is this:

```
$grid->ajax(true);  
$grid->ajaxId('grid'); // The id to place the ajax response
```

# Using Arrays, JSON, XML, CSV

## Array

To create a datagrid using a array the array must have this format

```
$array = array( array('name'=>'Jonh','age'=>'20','country'=>'England'),  
array('name'=>'Santos','age'=>'25','country'=>'England'),  
array('name'=>'Martin','age'=>'40','country'=>'Spain') )
```

And then the only thing you need to do is this

```
$grid->setDataFromArray($array);
```

The system will recognize the index's as titles, but you can change the name later. **How?** It's very simple.

```
$grid->updateColumn('age',array('title'=>'My Age'));
```

The method addColumn will not erase the previous params, instead it will append them using the array\_merge function. Because of this, 3 new methods have been created

```
$grid->resetColumn('column_name'); #Will erase all params previously  
defined  
  
$grid->removeColumn('column_name'); #The column will be removed from the  
set  
  
$grid->removeColumns(array('age','name')); #Remove an array of columns  
from the set
```

These 3 new methods are very useful when using foreign content (XML, CSV, JSON)

## XML

To create a grid using XML use this function

```
$grid->setDataFromXml($url, $loop, $columns = null);  
  
//The first argument is the xml location, the second is the loop location  
and the third is the columns location
```

Lets suppose you want to fetch a RSS file from blogspot. It will stay something like this

```
$grid->setDataFromXml(  
'http://my_url.blogspot.com/feeds/posts/default?alt=rss','channel,item');
```

```
//The first argument is the xml location, the second is the loop location
and the third is the columns location (optional)

//The second argument is the path to the loop. The data we want to fetch
is inside the channel and item array. (check the XML structure for
confirmation)
```

## CSV

More easy its impossible.

```
$grid->setDataFromCsv($file, $field = null, $separator = ',');

//The first param is the file location //Set field to 'true' if first line
if the file isn't the columns name //Third param the columns separator
```

Most of the times you will be using something like this

```
$grid->setDataFromCsv('media/files/grid.csv');
```

## JSON

It's also very easy to work with.

```
$grid->setDataFromJson($json, $file = false, $loop = null, $columns =
null);

//1-the data //2- set 'true' if the data is on a file or URL //3- The root
to the loop (similar when using XML files) //4-Columns name (optional)
```

Example

```
$grid->setDataFromJson ('http://services.sapo.pt/JobOffers/JSON' ,true,
'rss,channel,item');
```

## Export

To allow results exportation the only thing you need to do is this:

```
$grid->export = array('pdf','word','wordx','excel','print'); # or just
pdf and word...
```

Or choose the formats you want.

# CRUD

It's easier than ever to add record to a table But before a little more how it's done The way system treats records inserting If there are not defined fields the system will automatically hide the auto-increment field, and show all the others If the field type is enum, the system will show a DropDown menu with the options set in the field and if the field is 'set' a multi select will show up The system will not validate or filter any data if not specified (except the one made by Zend\_Db) CRUD Operations

Option	Type	Description
add	0 1	Gives user permission to add records to the table
delete	0 1	Gives user permission to delete records from the table
edit	0 1	Gives user permission to edit records from the table
button	0 1	This will place a small text before the form with a link to add a new record
double_tables	0 1	If you want to have the form and the grid on the same page, or if when you are adding or editing results the datagrid is omitted
onAddForce	array	Insert values that are not part of the form. The most common example is when you want to add the user id, the update date, etc. Ex: <code>\$form-&gt; onAddForce ( array ('date_added' =&gt; date('Y-m-d H:i:s'), 'user_id' =&gt; '1' ) );</code>
onEditForce	array	The same that onAddForce
Where onDeleteAdd	string	This is used for security. Imagine that you only want to allow a user to remove his own records. You will have to do something like this: Ex: <code>\$form-&gt; onDeleteAddWhere(" user_id='1' ");</code> Instead of <code>DELETE FROM table WHERE id='34'</code> you will get <code>DELETE FROM table WHERE id='34' AND user_id='1'</code>
onDeleteCasca de	array	This option is used for delete records from another table that matches a value that is being deleted from the table. Suppose you are removing a user from the users table. Probably will also want to remove the user articles, images, etc, etc.  EX: <code>\$form-&gt;</code>

	<pre>&gt;onDeleteCascade(array('table'=&gt;articles,'parentField'=&gt;'id','childField'=&gt;user_id,'operand'=&gt;'='));</pre> <p style="text-align: center;">\$form-</p> <pre>&gt;onDeleteCascade(array('table'=&gt;images,'parentField'=&gt;'id','childField'=&gt;user_id,'operand'=&gt;'='));</pre> <p>By default the parentField is the <b>primary_key</b> and the operand is <b>=</b>, so you can omit that indexes</p>
--	--

Example:

```
$form = new Bvb_Grid_Form ( );
$form->add ( 1 )
    ->button ( 1 )
    ->delete ( 1 )
    -> onAddForce ( array ( 'date_added' => date ( 'Y-m-d H:i:s' ) ) )
->onEditForce ( array ( 'date_added' => date ( 'Y-m-d H:i:s' ) ) );
```

Add columns to the form Option Type Description title string The title for the field description string The field description validators array An array with validators for a given field

```
$fields->validators ( array ( 'EmailAddress' ) );
```

filters array An array with filters for a given field

```
$field->filters(array('StringToLower','Alpha','StringTrim'));
```

values array An array containing values that will appear on a select input IMPORTANT: The datagrid will auto recognize the enum and set fields types, so you don't need to set any values, filters, or validators for them, as the system will check itself. However, if you define the values they will override the ones set on the db field Form

Example:

```
$fAdd = new Bvb_Grid_Form_Column ( 'firstname' );
$fAdd->title ( 'First name' )
    ->validators ( array ( 'EmailAddress' ) )
    ->description ( 'Insert you email address' );
$lastName = new Bvb_Grid_Form_Column ( 'lastname' );
$lastName->title ( 'Last name' );
```

```

$country = new Bvb_Grid_Form_Column ( 'country' );
$country->title ( 'Country' )
    ->description ( 'Choose your Country' )
    ->values ( $country );

$country = $db->fetchCol ( "SELECT DISTINCT(Name) FROM Country ORDER BY
Name ASC " );

$lang = new Bvb_Grid_Form_Column ( 'language' );
$lang->title ( 'Language' )
    ->description ( 'Your language' )
    ->values ( $language );

$language = $db->fetchCol ( "SELECT DISTINCT(Language) FROM
CountryLanguage ORDER BY Language ASC" );

$form->addColumns ( $fAdd, $lastName, $country, $lang );
$grid->addForm ( $form );

```

**Note:** The default validators and filters are the ones that ship with Zend Framework. If you want to add your custom, you have to define them after instantiate the datagrid

```

$grid->addElementDir ( 'My/Validate', 'My_Validate', 'validator' );
$grid->addElementDir ( 'My/Filter', 'My_Filter', 'filter' );

```

## Templates

At this point only word, pdf and table templates are customizable. The grid will have some default templates, but you may, and probably will, want to change them to fit your needs. So, to add a new template we have to do this

```

$grid = new Bvb_Grid_Deploy_Table($db,'Document Title');
$grid->addTemplateDir('My/Template/Table','My_Template_Table','table')
$grid->addTemplateDir('My/Template/Print','My_Template_Print','print')
$grid->addTemplateDir('My/Template/Pdf','My_Template_Pdf');

```

And then assign the template

```
$grid->setTemplate('personal','table');  
$grid->setTemplate('company','pdf');
```

The Datagrid will look for a file called `personal` at `My/Template/Table/Personal.php` and will expect a class with the name

```
My_Template_Table_Personal
```

**CRITICAL:** All templates **MUST** extend the ones that ship with the grid. So in this case will be:

```
My_Library_Template_Tables extends Bvb_Grid_Template_Table
```

**Important:** You need to inform the complete url for the images that appear on the table (for exportation). You can do that with the following code

```
$grid->imagesUrl = 'http://www.example.com/public/images/grid/';
```

There is no better explanation then look at the templates. They are pretty simple.

## Cache

The cache use can be very useful, especially on large environments. To use cache you only need the add the following line of code

```
$grid->cache = array('use'=>1, 'instance'=>Zend_Registry::get('cache'),  
'tag'=>'grid');
```

When performing CRUD operations, the cache will be cleaned.

## Internationalization

Fields titles, fields descriptions and a bunch of other strings are translatable. All you have to do for make it happen is registry in the `Zend_Registry` a translator instance with the name `Zend_Translate`