

# PHPReports Manual

Eustaquio “TaQ” Rangel (eustaquiorangel@yahoo.com)

January 7, 2005



This document will guide you to download, install, configure and run PHPReports. It will be a complete guide of all its features, showing how to you can use and how can you create or modify it to fit your needs. You can download recent copies of this document on <http://phpreports.sf.net>.

Thanks for my two girls, Ana Carolina and Ana Isabella, for their patience on the hours I expend on this project.

This document was made using  $\text{\LaTeX}$ .

# Contents

<b>1</b>	<b>What and how</b>	<b>5</b>
1.1	What is PHPReports? . . . . .	5
1.2	What is needed to use it? . . . . .	5
1.3	Does i need to pay you to use it? . . . . .	6
1.4	Tell me how it works . . . . .	6
<b>2</b>	<b>Installation</b>	<b>9</b>
2.1	How install and check all the stuff . . . . .	9
<b>3</b>	<b>Creating your reports</b>	<b>13</b>
3.1	Put some data there . . . . .	13
3.2	The XML report layout file . . . . .	14
3.3	The PHP code . . . . .	15
3.4	Making your report looks better . . . . .	18
3.5	Let there be colors . . . . .	20
3.6	The page element . . . . .	23
3.7	Sub groups . . . . .	25
3.8	The grand total . . . . .	29
3.9	Playing with the report . . . . .	30
3.10	Links . . . . .	36
3.11	Images . . . . .	38
3.12	Bookmarks . . . . .	39
<b>4</b>	<b>Output plugins</b>	<b>41</b>
4.1	What is an output plugin? . . . . .	41
4.2	Default output plugin . . . . .	43
4.3	Bookmarks output plugin . . . . .	43
4.4	Page output plugin . . . . .	44
4.5	TXT output plugin . . . . .	46
4.6	Creating your own output plugins . . . . .	46
4.7	The XML report data file . . . . .	48
<b>5</b>	<b>Some other things you need to know</b>	<b>49</b>
5.1	Setting your parameters from the PHP code . . . . .	49
5.2	Organizing your directory structure . . . . .	50
5.3	Breaking groups with more than one expression . . . . .	51
5.4	Save your report . . . . .	51
5.5	Restore your report . . . . .	51
5.6	Some few words about file integrity . . . . .	52
5.7	Exchanging formats . . . . .	52
5.8	Trying to make things faster . . . . .	53
5.9	Reuse your XML data . . . . .	54
5.10	Previewing your report . . . . .	55
5.11	Passing string parameters to your report . . . . .	55
5.12	Passing objects parameters to your report . . . . .	56

5.13	Inserting XHTML or PHP code into your COL . . . . .	57
5.14	Subreports . . . . .	60
<b>6</b>	<b>API</b>	<b>61</b>
6.1	Functions . . . . .	61
6.1.1	Functions you can use inside the COL element . . . . .	61
6.1.2	Functions you can use in your PHPReportMaker object . . . . .	62
6.2	XML . . . . .	66
6.2.1	REPORT . . . . .	66
6.2.2	CSS . . . . .	66
6.2.3	FORM . . . . .	66
6.2.4	DOCUMENT . . . . .	66
6.2.5	HEADER . . . . .	67
6.2.6	FOOTER . . . . .	67
6.2.7	ROW . . . . .	67
6.2.8	COL . . . . .	67
6.2.9	PAGE . . . . .	67
6.2.10	GROUPS . . . . .	68
6.2.11	GROUP . . . . .	68
6.2.12	FIELDS . . . . .	68
6.2.13	LINK . . . . .	68
6.2.14	BOOKMARK . . . . .	69
6.2.15	IMG . . . . .	69
6.2.16	XHTML . . . . .	69
<b>7</b>	<b>FAQ</b>	<b>71</b>

# Chapter 1

## What and how

### 1.1 What is PHPReports?

PHPReports is a set of PHP classes, XML instructions and XSLT scripts to transform the XML file into PHP code. The generated PHP code will be used with the PHP classes.

I started the idea when I needed to make all the stuff we use on programs like Visual Basic and Foxpro on the browser, and one of the things that I didn't found was how to make and print reports, using a SQL query, on a easy (not so easy but) way, using the browser.

Nowadays it runs fine here where I work, and I hope it could be usefull for you guys too.

**Important note:** I'm writing this info based on the 0.2.0 version. Users of previous versions will be warned about new features and some little fix stuff they need to make the 0.2.0 + version to work.

**Another VERY IMPORTANT note:** Despite the fact that PHP can run on a lot of different operational systems and webservers, this software **run better** using Linux as operational system and Apache as the webserver.

This does not say that you won't be allowed to run it on another configuration, but I can't guarantee that it will be 100% fully operational. Some dudes need to hack the code on some Windows installations.

But on a Linux/Apache system, it runs fine, and **very** fine. :-)

RUN BETTER ON LINUX



### 1.2 What is needed to use it?

You need a Apache server (<http://www.apache.org>) with PHP (<http://www.php.net>) support compiled with XML/XSLT support.

I use the Sablotron libs to make this ([http://www.gingerall.com/charlie/ga/xml/p\\_sab.xml](http://www.gingerall.com/charlie/ga/xml/p_sab.xml)) on PHP4.

On PHP5, the developers changed the Sablotron extension to the new XSL extension (<http://www.php.net/manual/en/ref.xsl.php>), and it seems to run better on Windows environment than Sablotron.

### 1.3 Does i need to pay you to use it?

Hell, no. It's under the GPL, you don't need to pay. Just follow the GPL rules and everybody will be happy.

I really was needing some way to contribute to the open source community, and I hope it was just the beginning.

If you really loved PHPReports and want to make some kind of donation or stuff, you can see my Amazon wish list at <http://www.amazon.com>. Just search for Eustaquio Rangel there. :-) Or you can donate via **PayPal**, on the Sourceforge site (<http://sourceforge.net/projects/phpreports/>). Or what the hell, send me a postcard from where you live. :-)

### 1.4 Tell me how it works

I heard one day a very wise teacher says that "the purpose of a good documentation is that you don't need the author fixed on it, on your bookshelf". I'll try to do the best I can to make it come true, so here we go.

A report, on PHPReports or any other program that deal with that, have always some divisions. I call it:

- the document layer
- the page layers
- the group layer

You just have **one document layer**, **one page layer** (you can have a lot of pages, but just **one page layer** to configure) and some group layers.

All these layers collect information about the data on your report, like the number of lines, statistics about the fields and so on.

The **document layer** stores ALL these statistics, and stores it till the report end.

The **page layer** stores it till the page end, and reset it there.

The **group layer** stores it till the end of group, let me translate here group as a set of data defined by a break expression, which could be any kind of field contained on your data set.

Each layer have its own header and footer. The group layer have one more division where it shows the data info. If you have more than one group (even when you have just a simple report, you need to add one group to deal with your info ), the most internal group will show you the data, the other ones can show it too, but its your choice. Let me try to draw the full thing here:

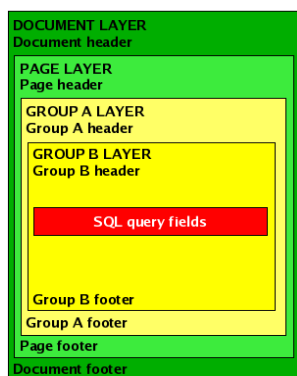


Figure 1.1: All the layers

Got it? The document contains page, page contains groups, and a group can contain another group.

One rule on that case is that the break expression of the inner group must contain the break expression of the group above. On the example, the first group break when A changes, and the second group breaks when A or B changes.

When this happens, its fired an event that notify that the group needs to print its footer and header, and all the other groups related to it are notified too, to make what needs to be done.





# Chapter 2

## Installation

### 2.1 How install and check all the stuff

Let me ask you about some things before we continue.

**a) Are you using Windows to run PHP and PHPReports?**

So it's better you run it using **PHP5**, because PHP5 have XSL support (but need to be compiled with that) and works fine on Windows. Check if the XSL support is enabled with your `phpinfo()` function. If you're using binary files, and XSL support is not enabled, you'll need to compile source code. If you're using PHP4, you can also install Sablotron (see below) on Windows, but I'm not sure it will works on a nice way. Some guys had some problems with it.

**b) Are you using Linux, with a precompiled PHP5 package?**

So you need to check if there is XSL support compiled. Check your `phpinfo()`. If not XSL enabled there, you'll need to compile PHP.

**c) Are you using Linux, with a precompiled PHP4 package?**

Oh-oh, support for XSL transformations on PHP4 is just provided by Sablotron (see below). You'll need to remove your PHP4 package and compile the sources, with Sablotron installed (as a package or compiling it's source code).

**d) Are you compiling the source code?**

So you'll have no problems.

Ok, you need to make sure your web server environment is ok. Apache is installed correctly? So, proceed.

Now its time to install Sablotron, if you use **PHP4**. I always compile the Sablotron sources, but I think you can install it with some kind of package (RPM etc). You can check

[http://www.gingerall.com/charlie/ga/xml/x\\_sabphp.xml](http://www.gingerall.com/charlie/ga/xml/x_sabphp.xml) for more information.

If you use **PHP5**, you need to compile PHP5 with XSL support (<http://www.php.net/manual/en/ref.xsl.php>).

So now its all installed, Apache, PHP and Sablotron/XSL. Now you need the PHPReports files.

Go to <http://phpreports.sourceforge.net>, click on the download link and download the latest version you see there.

Let's unpack the package. Go to your desired location and unpack the PHPReports file:

```
tar xvfz phpreports < versionhere >.tgz
```

Or use other tool you like to do that.

This will provide you this directory tree:

```
phpreports
```

```
+-- css
+-- database
+-- docs
+-- img
+-- output
+-- php
+-- xslt
```

Lets take a look on it:

The **css** directory have some samples css files used on the samples.  
The **database** directory stores the database interfaces.  
The **docs** directory is supposed to include this kind of file you're reading. ;-)  
The **img** directory have some images used on the samples.  
The **output** directory is where the output plugins are. We'll talk about output later.  
The **php** directory is where the PHP code are.  
The **xslt** directory is where the XSLT files are.

You need to set where you put the PHPReports classes. Suppose you put your classes under

```
/usr/lib/phpreports
```

so you can choose to put this path inside your **php.ini** file and restart your web server (I think this way is better) or put a **ini\_set()** call before using some PHPReports class, like this:

```
<?php
    ini_set("include_path",ini_get("include_path").":/usr/lib/phpreports/");
    include_once "PHPReportMaker.php";
    $oRpt = new PHPReportMaker();
?>
```

Now we'll make a simple test to see if your XSLT processor is working.

### If you use Sablotron

Open a console, go to the PHPReports directory and type:

```
sabcmd xslt/sabtest.xsl sabtest.xml
```

Your output should looks like:

```
Congratulations!
Sablotron is working properly.
```

Now its time to test the XSL transformation on the PHP code. Open a console and go to `<phpreports_home>/php` and type:

```
php sabtest.php
```

You must see the same result as above.

And what about it fail and gives you some kind of error? If it fails on the console, you should check your Sablotron installation. If it works on the console and fail on the browser, you should check if PHP was compiled correctly with Sablotron support. A good hint maybe the `phpinfo()` function, it should have something like this:

```
xslt
XSLT support enabled
Backend - Sablotron
```

**HOT TIP** If you got a message like unknown encoding 'ISO-8859-1', it's about the encoding I use for latin characters. Maybe you don't need it, but it works fine on the most cases.

Some people told me that they have some trouble with it running on Windows machines, but I don't have a idea on how to fix it on Windows .

If you have problems, please check [http://www.gingerall.com/charlie/ga/xml/x\\_sabphp.xml](http://www.gingerall.com/charlie/ga/xml/x_sabphp.xml) for more info and help on how to install/compile Sablotron.

### **If you use XSL**

On this case phpinfo() should return something like this:

```
xsl
XSL enabled
libxslt Version 1.0.27
libxslt compiled against libxml Version 2.5.4
EXSLT enabled
libxslt Version 1.0.27
```

Open a console on your <phpreports\_home>/php and type:

```
php xsltest.php
```

Your output should looks like:

```
Congratulations!
XSL is working properly.
```

You can run it also on the command line, with Python:

```
python xsltest.py
```

should returns the same result as above.



## Chapter 3

# Creating your reports

### 3.1 Put some data there

Let's start our work. We'll create a report with some customers, grouped by city, and list all the products they bought, grouped by product type.

Create a table like this on your database (I know the table design is horrible but it is just for our example):

```
mysql> desc sales;
+-----+-----+-----+-----+-----+-----+
| Field | Type          | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+-----+
| city  | varchar(50)   | YES  |     | NULL    |       |
| name  | varchar(50)   | YES  |     | NULL    |       |
| type  | varchar(10)   | YES  |     | NULL    |       |
| item  | varchar(50)   | YES  |     | NULL    |       |
| value | decimal(15,2) | YES  |     | NULL    |       |
+-----+-----+-----+-----+-----+-----+
```

This table was based on mysql, please create it on your database way. So, let's put some values there:

```
mysql> select * from sales;
+-----+-----+-----+-----+-----+
| city  | name          | type | item          | value |
+-----+-----+-----+-----+-----+
| Rio Preto | Eustaquio Rangel | CD   | This is the CD 1 | 10.00 |
| Rio Preto | Eustaquio Rangel | CD   | This is the CD 2 | 20.00 |
| Rio Preto | Eustaquio Rangel | CD   | This is the CD 3 | 30.00 |
| Rio Preto | Eustaquio Rangel | Book | This is the book 1 | 40.00 |
| Rio Preto | Eustaquio Rangel | Book | This is the book 2 | 50.00 |
| Rio Preto | Ana Carolina    | CD   | This is the CD 1 | 10.00 |
| Rio Preto | Ana Carolina    | CD   | This is the CD 2 | 20.00 |
| Rio Preto | Ana Carolina    | Book | This is the book 1 | 30.00 |
| Rio Preto | Ana Carolina    | Book | This is the book 2 | 40.00 |
| Rio Preto | Ana Carolina    | Book | This is the book 3 | 50.00 |
| Rio Preto | Ana Carolina    | Book | This is the book 4 | 60.00 |
| Rio Preto | Ana Carolina    | Book | This is the book 5 | 70.00 |
| Sao Paulo | Andre Kada      | CD   | This is the CD 1 | 10.00 |
| Sao Paulo | Andre Kada      | CD   | This is the CD 2 | 20.00 |
| Sao Paulo | Andre Kada      | CD   | This is the CD 3 | 30.00 |
| Sao Paulo | Andre Kada      | CD   | This is the CD 4 | 40.00 |
| Sao Paulo | Andre Kada      | CD   | This is the CD 5 | 50.00 |
| Sao Paulo | Andre Kada      | Book | This is the book 1 | 50.00 |
| Sao Paulo | Andre Kada      | Book | This is the book 2 | 60.00 |
| Sao Paulo | Andre Kada      | Book | This is the book 3 | 70.00 |
| Mirassol | Marcio Lambary  | CD   | This is the CD 1 | 10.00 |
| Mirassol | Marcio Lambary  | CD   | This is the CD 2 | 20.00 |
| Mirassol | Marcio Lambary  | CD   | This is the CD 3 | 30.00 |
| Mirassol | Marcio Lambary  | Book | This is the book 1 | 40.00 |
+-----+-----+-----+-----+-----+
```

## 3.2 The XML report layout file

Now we have something to work with. I'll create a very basic report layout file now, called *sales.xml*:

```
<?xml version="1.0" encoding="ISO-8859-1" standalone="no"?>
<!DOCTYPE REPORT SYSTEM "PHPReport.dtd">
<REPORT MARGINWIDTH="5" MARGINHEIGHT="5">
  <TITLE>Sales Report</TITLE>
  <BACKGROUND_COLOR>#FFFFFF</BACKGROUND_COLOR>
  <SQL>select CITY,NAME,TYPE,ITEM,VALUE from sales order by CITY,NAME,TYPE,ITEM</SQL>
  <INTERFACE>mysql</INTERFACE>
  <CONNECTION>localhost</CONNECTION>
  <DATABASE>phpreports</DATABASE>
  <NO_DATA_MSG>No data was found, check your query</NO_DATA_MSG>
  <PAGE BORDER="1" SIZE="10" CELLSPACING="0" CELLPADDING="5">
  </PAGE>
  <GROUPS>
    <GROUP NAME="main">
      <FIELDS>
        <ROW>
          <COL TYPE="FIELD">CITY</COL>
          <COL TYPE="FIELD">NAME</COL>
          <COL TYPE="FIELD">TYPE</COL>
          <COL TYPE="FIELD">ITEM</COL>
          <COL TYPE="FIELD">VALUE</COL>
        </ROW>
      </FIELDS>
    </GROUP>
  </GROUPS>
</REPORT>
```

The first thing you need is the XML declaration. A valid XML file must have it. The encoding parameter there is the encoding I use for Latin characters, maybe you'll need to change it if your system don't support or allow it. The DOCTYPE tag is about the PHPReport.dtd file, used to validate your XML document.

If you don't understand the XML stuff, there is some cool tutorials on the internet to teach you the basics, and believe me, learn XML worths a lot.

You can notice the REPORT element. It's the main element there and where we'll define or report layout. I'll explain all the parameters of REPORT and of other elements later. By now, we're telling the report generator that our report have a margin width of 5 pixels and a margin height of 5 pixels too.

The same way of HTML stuff, and you'll see a lot of other HTML related stuff as we are going on.

Next, inside the REPORT element, we have our report TITLE, the BACKGROUND COLOR, the SQL query,

**IMPORTANT** You can put your SQL query here or in the PHP file, as we'll see.

**IMPORTANT** Check your field name case on your query and your report layout. If you ask for a field called city on your query and put a CITY field on your report layout, your database will be confused and will not return you a value. An error message will warn you of this problem, if it happens.

the database INTERFACE you want to use (I'm using mysql here), the CONNECTION name (localhost here), the DATABASE you want to use when opening the connection, the NO DATA MESSAGE where you specify a message to show when no data is retrieved.

The next element is the PAGE element, notice the SIZE parameter, is where I tell the page size, we'll work with a short page here to make easier to see the results.

Now, we see the GROUPS element. The GROUPS element can have one or more GROUP element inside of it, and we see a GROUP named main there. Inside of it we see a FIELDS element, with a ROW and some COLS (with the TYPE=FIELDS) that refers to our SQL columns returned by the query.

It's important to you get the basic idea here, let's try to read the PAGE and GROUPS/GROUP elements as defined there as I have a page with 10 rows, and a group of data that will list on one row the city, name, type, item and value returned by the SQL query. For now it's enough ...

### 3.3 The PHP code

Now let's see the PHP code required to make your report works, on a file called sales.php:

```
<?php
// the line below is only needed if the include_path is not set on php.ini
ini_set("include_path",ini_get("include_path").":/usr/lib/phpreports/");
include_once "PHPReportMaker.php";
$oRpt = new PHPReportMaker();
$oRpt->setUser("taq");
$oRpt->setPassword("*****");
$oRpt->setXML("sales.xml");
$oRpt->run();
?>
```

Both sales.xml and sales.php are provided with the PHPReports package. I'll suppose you unpacked the PHPReports package on /usr/lib/phpreports/ and put sales.php under http://localhost/phpreports/ for example. If you put it somewhere else (visible by the web server, of course) please change the URL as you need.

So, open your browser and put this URL on the address bar:

*http://localhost/phpreports/sales.php*

You should see something like this:

Mirassol	Marcio Lambary	Book	This is the book 1	40.00
Mirassol	Marcio Lambary	CD	This is the CD 1	10.00
Mirassol	Marcio Lambary	CD	This is the CD 2	20.00
Mirassol	Marcio Lambary	CD	This is the CD 3	30.00
Rio Preto	Ana Carolina	Book	This is the book 1	30.00
Rio Preto	Ana Carolina	Book	This is the book 2	40.00
Rio Preto	Ana Carolina	Book	This is the book 3	50.00
Rio Preto	Ana Carolina	Book	This is the book 4	60.00
Rio Preto	Ana Carolina	Book	This is the book 5	70.00
Rio Preto	Ana Carolina	CD	This is the CD 1	10.00
Rio Preto	Ana Carolina	CD	This is the CD 2	20.00
Rio Preto	Eustaquio Rangel	Book	This is the book 1	40.00
Rio Preto	Eustaquio Rangel	Book	This is the book 2	50.00
Rio Preto	Eustaquio Rangel	CD	This is the CD 1	10.00
Rio Preto	Eustaquio Rangel	CD	This is the CD 2	20.00
Rio Preto	Eustaquio Rangel	CD	This is the CD 3	30.00
Sao Paulo	Andre Kada	Book	This is the book 1	50.00
Sao Paulo	Andre Kada	Book	This is the book 2	60.00
Sao Paulo	Andre Kada	Book	This is the book 3	70.00
Sao Paulo	Andre Kada	CD	This is the CD 1	10.00
Sao Paulo	Andre Kada	CD	This is the CD 2	20.00
Sao Paulo	Andre Kada	CD	This is the CD 3	30.00
Sao Paulo	Andre Kada	CD	This is the CD 4	40.00
Sao Paulo	Andre Kada	CD	This is the CD 5	50.00

Figure 3.1: Your first report

Ugly, uh? But the main thing here is that it already connected to your database, made the query and is showing you the results.

**IMPORTANT** Make sure your web server user have write permission on the PHPReports tmp directory. If you're curious for the reason of this, it's a good chance to explain you how the things works here.

**NEW FEATURE WARNING** Now we'll see the new intermediate XML file with the report data and something about the new output plugin feature.

Change your sales.php file to this (new rows are **green**):

```
<?php
    include_once "PHPReportMaker.php";
    $oRpt = new PHPReportMaker();
    $oRpt->setUser("taq");
    $oRpt->setPassword("*****");
    $oRpt->setXML("sales.xml");
    $oOut = $oRpt->createOutputPlugin("default");
    $oOut->setClean(false);
    $oRpt->setOutputPlugin($oOut);
    $oRpt->run();
?>
```

What happened here was that we created an object of a output plugin, a default one (first bold row), and we asked it to do not clean our temporary file (second bold row) and we asked PHPReports to use this new output plugin.

Its the same thing as don't specify an output plugin, PHPReports will always create a default one if you don't specify it, but what we want now is see what lies on the tmp directory.

You will find a file like this:

```
ls tmp -rw-r---r--- nobody nobody 2,4k phrprtJIKM3h.xml
```

Is a temporary file, so it have it's weird name. ;-) Let's open it to see what it have inside:

```
<?xml version="1.0" encoding="ISO-8859-1" standalone="no"?>
<RP TITLE="Sales Report" BGCOLOR="#FFFFFF">
<PG SZ="10" AL="LEFT" PN="1" PA="5" SP="0" BO="1">
<R><C>Mirassol</C><C>Marcio Lambary</C><C>Book</C><C>This is the book 1</C><C>40.00</C></R>
<R><C>Mirassol</C><C>Marcio Lambary</C><C>CD</C><C>This is the CD 1</C><C>10.00</C></R>
<R><C>Mirassol</C><C>Marcio Lambary</C><C>CD</C><C>This is the CD 2</C><C>20.00</C></R>
<R><C>Mirassol</C><C>Marcio Lambary</C><C>CD</C><C>This is the CD 3</C><C>30.00</C></R>
<R><C>Rio Preto</C><C>Ana Carolina</C><C>Book</C><C>This is the book 1</C><C>30.00</C></R>
<R><C>Rio Preto</C><C>Ana Carolina</C><C>Book</C><C>This is the book 2</C><C>40.00</C></R>
<R><C>Rio Preto</C><C>Ana Carolina</C><C>Book</C><C>This is the book 3</C><C>50.00</C></R>
<R><C>Rio Preto</C><C>Ana Carolina</C><C>Book</C><C>This is the book 4</C><C>60.00</C></R>
<R><C>Rio Preto</C><C>Ana Carolina</C><C>Book</C><C>This is the book 5</C><C>70.00</C></R>
<R><C>Rio Preto</C><C>Ana Carolina</C><C>CD</C><C>This is the CD 1</C><C>10.00</C></R>
</PG>
<PG SZ="10" AL="LEFT" PN="2" PA="5" SP="0" BO="1">
<R><C>Rio Preto</C><C>Ana Carolina</C><C>CD</C><C>This is the CD 2</C><C>20.00</C></R>
<R><C>Rio Preto</C><C>Eustaquio Rangel</C><C>Book</C><C>This is the book 1</C><C>40.00</C></R>
<R><C>Rio Preto</C><C>Eustaquio Rangel</C><C>Book</C><C>This is the book 2</C><C>50.00</C></R>
<R><C>Rio Preto</C><C>Eustaquio Rangel</C><C>CD</C><C>This is the CD 1</C><C>10.00</C></R>
<R><C>Rio Preto</C><C>Eustaquio Rangel</C><C>CD</C><C>This is the CD 2</C><C>20.00</C></R>
<R><C>Rio Preto</C><C>Eustaquio Rangel</C><C>CD</C><C>This is the CD 3</C><C>30.00</C></R>
<R><C>Sao Paulo</C><C>Andre Kada</C><C>Book</C><C>This is the book 1</C><C>50.00</C></R>
<R><C>Sao Paulo</C><C>Andre Kada</C><C>Book</C><C>This is the book 2</C><C>60.00</C></R>
<R><C>Sao Paulo</C><C>Andre Kada</C><C>Book</C><C>This is the book 3</C><C>70.00</C></R>
<R><C>Sao Paulo</C><C>Andre Kada</C><C>CD</C><C>This is the CD 1</C><C>10.00</C></R>
</PG>
<PG SZ="10" AL="LEFT" PN="3" PA="5" SP="0" BO="1">
<R><C>Sao Paulo</C><C>Andre Kada</C><C>CD</C><C>This is the CD 2</C><C>20.00</C></R>
<R><C>Sao Paulo</C><C>Andre Kada</C><C>CD</C><C>This is the CD 3</C><C>30.00</C></R>
<R><C>Sao Paulo</C><C>Andre Kada</C><C>CD</C><C>This is the CD 4</C><C>40.00</C></R>
<R><C>Sao Paulo</C><C>Andre Kada</C><C>CD</C><C>This is the CD 5</C><C>50.00</C></R>
</PG>
</RP>
```



What we have here is the data of your report, described using XML (again!), with short tags because a verbose and clear XML data file could have a very big size. So I'm using these tags to store the data:

- RP** It's the main layer tag, it's the REPORT
- PG** It's the PAGE layer, notice we have 3 pages there
- R** It's the ROW layer
- C** It's the COL layer

**NEW FEATURE WARNING** We didn't have this intermediate XML file on previous versions, so you can notice a tiny small delay to create and render your report now.

The very good points to this XML file is that you can transform it on what file format you want to. The default one is the HTML output, but we'll see some more later. Another good point is that we can store the data file and transform it later also.

I'll tell you about output plugins and the interface that allow you to create your own plugins later.

For now, we need to understand the way it works:

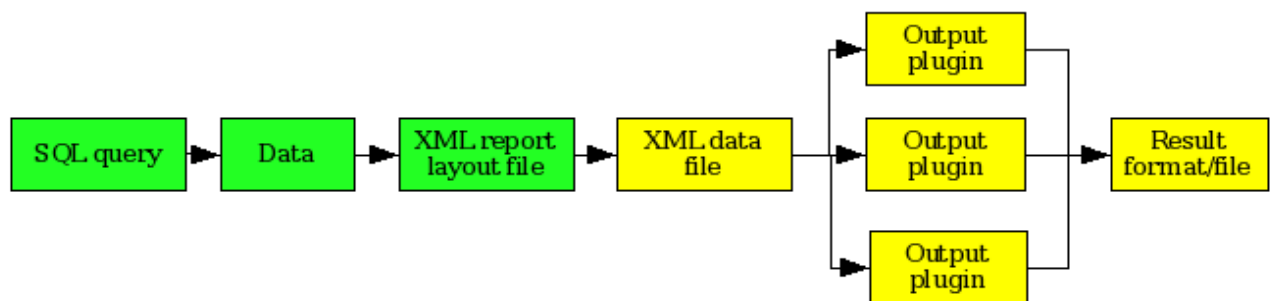


Figure 3.2: The creation/rendering path

The green rectangles is where you need to provide input to create your report, the yellow ones are the things PHPReports provides to create your report.

### 3.4 Making your report looks better

So, now you understand how the things works, let's put some more cool things on it. Suppose I want to just show a header with the name of the city (not the name of the city on every row), and the sum of all the items bought on that city on the end of that group, how can I do? The question pointed the way, we need a HEADER, but also we need to GROUP the cities to have this, and we need a FOOTER on that group to show the sum. The next change to *sales.xml* (green rows shows the new code) is:

```
<?xml version="1.0" encoding="ISO-8859-1" standalone="no"?>
<!DOCTYPE REPORT SYSTEM "PHPReport.dtd">
<REPORT MARGINWIDTH="5" MARGINHEIGHT="5">
<TITLE>Sales Report</TITLE>
  <BACKGROUND_COLOR>#FFFFFF</BACKGROUND_COLOR>
  <SQL>select CITY,NAME,TYPE,ITEM,VALUE from sales order by CITY,NAME,TYPE,ITEM</SQL>
  <INTERFACE>mysql</INTERFACE>
  <CONNECTION>localhost</CONNECTION>
  <DATABASE>phpreports</DATABASE>
  <NO_DATA_MSG>No data was found, check your query</NO_DATA_MSG>
  <PAGE BORDER="1" SIZE="10" CELLSPACING="0" CELLPADDING="5">
  </PAGE>
  <GROUPS>
    <GROUP NAME="city" EXPRESSION="CITY">
      <HEADER>
        <ROW>
          <COL ALIGN="RIGHT">city:</COL>
          <COL TYPE="EXPRESSION">${this->getValue("CITY")}</COL>
        </ROW>
      </HEADER>
      <FIELDS>
        <ROW>
          <COL TYPE="FIELD">NAME</COL>
          <COL TYPE="FIELD">TYPE</COL>
          <COL TYPE="FIELD">ITEM</COL>
          <COL TYPE="FIELD">VALUE</COL>
        </ROW>
      </FIELDS>
      <FOOTER>
        <ROW>
          <COL ALIGN="RIGHT" TYPE="EXPRESSION">"total of ".${this->getValue("CITY")}
        </ROW>
          <COL TYPE="EXPRESSION">${this->getSum("VALUE")}</COL>
        </ROW>
      </FOOTER>
    </GROUP>
  </GROUPS>
</REPORT>
```

The result is (I'll show you always the first page now):

city:	Mirassol		
Marcio Lambarly	Book	This is the book 1	40.00
Marcio Lambarly	CD	This is the CD 1	10.00
Marcio Lambarly	CD	This is the CD 2	20.00
Marcio Lambarly	CD	This is the CD 3	30.00
total of Mirassol	100		
city:	Rio Preto		
Ana Carolina	Book	This is the book 1	30.00
Ana Carolina	Book	This is the book 2	40.00
Ana Carolina	Book	This is the book 3	50.00

Figure 3.3: The basic report

As you can see, now we have the city group, with a HEADER that show us what city are the data on the next rows (so we don't need to print it on every row) and a footer that show us the amount of items bought on that city.

To make the grouping and breaking magic, we used the `EXPRESSION="CITY"` attribute on the `GROUP` element. It tells PHPReports to break the group every time the `CITY` field is different than the previous value.

If you look to the file above, you can see clearly the `HEADER` tag, with a `ROW` and two `COLs` inside of it. You can have how many `ROWS` and `COLs` you want there.

On the first `COL`, there's an attribute `ALIGN` which tells the values there to be aligned on the right side (HTML similar). Inside the `COL`, we have a plain text value.

To put plain text you just need to type it inside a `COL`.

The next `COL` have an attribute `TYPE="EXPRESSION"`, which tells PHPReports to evaluate the text inside the element as a PHP expression. It means that you can put any valid PHP code (don't forget it must be scope valid) there, some function (for example, `date("d/m/Y")`) or variable.

On the example we're printing the value of the `CITY` field, on the group scope, with the function `getValue("CITY")` (more about PHPReports functions later), returning the value of the `$this->` reference. The `$this->` reference points to the current group.

**NEW FEATURE WARNING** For the users of previous versions: I removed that horrible `$header->` reference, thanks God! But don't worry, it still works, you don't need to change your reports and now can use the `$this->` reference.

**NEW FEATURE WARNING** For the users of previous versions: now you are allowed to use `COLs` of `TYPE="EXPRESSION"` on all elements. On previous versions you couldn't use `TYPE="EXPRESSION"` on the `FIELDS` element.

The same way we have the `HEADER`, we have the `FOOTER`. The first `COL` on the `FOOTER` (always inside a `ROW` element) have an `EXPRESSION` joining a string with the value returned by `getValue("CITY")`, and the second `COL` returns the `GROUP` sum of the `VALUE` field using `getSum("VALUE")`.

## 3.5 Let there be colors

The colors of PHPReports are defined by CSS, as on HTML. If you don't know CSS, it's another good idea search for some tutorials on the net. ;-)

I'll create a file called sales.css where I'll define the colors of HEADERS, FOOTERS, FIELDS COLs and another class to define my bold fonts:

```
.HEADER {
    font-family: "arial","verdana";
    font-size: 10px;
    color: #505050;
    background: #DDDDDD;
}

.FOOTER {
    font-family: "arial","verdana";
    font-size: 10px;
    color: #505050;
    background: #CCCCCC;
}

.FIELDS {
    font-family: "arial","verdana";
    font-size: 10px;
    color: #000000;
    background: #FEFEFE;
}

.BOLD {
    font-weight: bold;
}
```

Now we must modify or sales.xml file to use the CSS classes:

```
<?xml version="1.0" encoding="ISO-8859-1" standalone="no"?>
<!DOCTYPE REPORT SYSTEM "PHPReport.dtd">
<REPORT MARGINWIDTH="5" MARGINHEIGHT="5">
    <TITLE>Sales Report</TITLE>
    <BACKGROUND_COLOR>#FFFFFF</BACKGROUND_COLOR>
    <SQL>select CITY,NAME,TYPE,ITEM,VALUE from sales order by CITY,NAME,TYPE,ITEM</SQL>
    <INTERFACE>mysql</INTERFACE>
    <CONNECTION>localhost</CONNECTION>
    <DATABASE>phpreports</DATABASE>
    <NO_DATA_MSG>No data was found, check your query</NO_DATA_MSG>
    <CSS>http://localhost/phpreports2/sales.css</CSS>
    <PAGE BORDER="1" SIZE="10" CELLSPACING="0" CELLPADDING="5">
    </PAGE>
    <GROUPS>
        <GROUP NAME="city" EXPRESSION="CITY">
            <HEADER>
                <ROW>
                    <COL ALIGN="RIGHT" CELLCLASS="HEADER">city:</COL>
                    <COL TYPE="EXPRESSION" CELLCLASS="HEADER" TEXTCLASS="BOLD" COLSPAN="3">
$this->getValue("CITY")</COL>
                </ROW>
            </HEADER>
            <FIELDS>
                <ROW>
                    <COL TYPE="FIELD" CELLCLASS="FIELDS">NAME</COL>
                    <COL TYPE="FIELD" CELLCLASS="FIELDS">TYPE</COL>
                    <COL TYPE="FIELD" CELLCLASS="FIELDS">ITEM</COL>
                    <COL TYPE="FIELD" CELLCLASS="FIELDS">VALUE</COL>
                </ROW>
            </FIELDS>
        </GROUP>
    </GROUPS>
```

```

        <FOOTER>
            <ROW>
                <COL ALIGN="RIGHT" TYPE="EXPRESSION" CELLCLASS="FOOTER" COLSPAN="3">
"total of ".$this->getValue("CITY")</COL>
                <COL TYPE="EXPRESSION" CELLCLASS="FOOTER" TEXTCLASS="BOLD">
$this->getSum("VALUE")</COL>
            </ROW>
        </FOOTER>
    </GROUP>
</GROUPS>
</REPORT>

```

Now it looks like this:

city:	Mirassol		
Marcio Lambary	Book	This is the book 1	40.00
Marcio Lambary	CD	This is the CD 1	10.00
Marcio Lambary	CD	This is the CD 2	20.00
Marcio Lambary	CD	This is the CD 3	30.00
total of Mirassol			<b>100</b>
city:	Rio Preto		
Ana Carolina	Book	This is the book 1	30.00
Ana Carolina	Book	This is the book 2	40.00
Ana Carolina	Book	This is the book 3	50.00

Figure 3.4: Basic colors

Oh yeah, that's better now! To do that, look the green code above: first, you need to tell PHPReports where your css file is using the element CSS. So, you use CELLCLASS to the class of the cell (or COL) and TEXTCLASS to the class of the font inside the cell.

You just need the TEXTCLASS if you want the font different than the CELLCLASS. CELLCLASS can define the way the font inside the COL will looks like, but sometimes, as in the bold font there, we need to specify another font inside the CELLCLASS.

I used another parameter here also, the COLSPAN parameter. The COLSPAN works like the HTML one, it spans the column for *x* columns you want. In the HEADER I make the second column with a size of 3 columns, and on the FOOTER was the first column that spans that way.

One other cool feature you can control with the CSS classes is the way even and odds rows (based on their number on the page) have its colors. Insert this code in your sales.css file:

```

.ODD {
    font-family: "arial","verdana";
    font-size: 10px;
    color: #000000;
    background: #BDDFF3;
}
.BOLD {
    font-weight: bold;
}

```

and modify sales.xml like this (I'll show you just the FIELDS element):

```

<FIELDS>
    <ROW>
        <COL TYPE="FIELD" CELLCLASSEVEN="EVEN" CELLCLASSODD="ODD">NAME</COL>

```

```

<COL TYPE="FIELD" CELLCLASSEVEN="EVEN" CELLCLASSODD="ODD">TYPE</COL>
<COL TYPE="FIELD" CELLCLASSEVEN="EVEN" CELLCLASSODD="ODD">ITEM</COL>
<COL TYPE="FIELD" CELLCLASSEVEN="EVEN" CELLCLASSODD="ODD">VALUE</COL>
</ROW>
</FIELDS>

```

This makes the even rows use the CSS class defined on CELLCLASSEVEN and the odd rows use the class defined on CELLCLASSODD. Your report looks like this now:

city:	<b>Mirassol</b>		
Marcio Lambarly	Book	This is the book 1	40.00
Marcio Lambarly	CD	This is the CD 1	10.00
Marcio Lambarly	CD	This is the CD 2	20.00
Marcio Lambarly	CD	This is the CD 3	30.00
total of Mirassol			<b>100</b>
city:	<b>Rio Preto</b>		
Ana Carolina	Book	This is the book 1	30.00
Ana Carolina	Book	This is the book 2	40.00
Ana Carolina	Book	This is the book 3	50.00

Figure 3.5: More colors

Since the 0.3.1 version, we can define CSS files for different type of media. For example, suppose you want the full coloured report above on the screen and a black-and-white version for printing. Now we can do this:

```

<CSS MEDIA="screen">http://localhost/phpreports2/sales.css</CSS>
<CSS MEDIA="print">http://localhost/phpreports2/sales_for_printing.css</CSS>

```

When using this kind of thing, your web browser will use the MEDIA="print" stylesheet when sending your report to the printer. If you don't specify a media type there, "screen" will be the default one. If you define the CSS classes there to black-and-white stuff, your report will look like the one on the grayscale figure above.

More about this here:

<http://www.w3.org/TR/REC-CSS2/media.html>

## 3.6 The page element

Let's deal with the PAGE element now. You noticed (as I told you) or PAGE is very short, just 10 rows. We'll change this right now, remove the borders, put a HEADER and a FOOTER on the page and deal with some XHTML code inside the COL element.

Here is the changes on sales.xml (**only the PAGE element**):

```
<PAGE BORDER="0" SIZE="20" CELLSPACING="0" CELLPADDING="5">
  <HEADER>
    <ROW>
      <COL CELLCLASS="HEADER" COLSPAN="4">
        <XHTML>
          <TABLE BORDER="0" CELLPADDING="2" CELLSPACING="0" WIDTH="100%">
            <TR>
              <TD CLASS="HEADER">
                <b>JOHN DOE ENTERPRISES</b>
              </TD>
              <TD ROWSPAN="2" CLASS="HEADER" style="background:#A0A0A0;
color:#FFFFFF;" ALIGN="CENTER">
                powered by<br/><b>phpreports</b>
              </TD>
            </TR>
            <TR>
              <TD CLASS="HEADER">
                <b>SALES REPORT</b>
              </TD>
            </TR>
            <TR>
              <TD COLSPAN="4"><hr/></TD>
            </TR>
          </TABLE>
        </XHTML>
      </COL>
    </ROW>
  </HEADER>
  <FOOTER>
    <ROW>
      <COL ALIGN="RIGHT" COLSPAN="3" CELLCLASS="FOOTER">page total</COL>
      <COL TYPE="EXPRESSION" CELLCLASS="FOOTER" TEXTCLASS="BOLD" ALIGN="RIGHT">
        $this->getSum("VALUE")</COL>
    </ROW>
    <ROW>
      <COL ALIGN="RIGHT" COLSPAN="4" TYPE="EXPRESSION" CELLCLASS="FOOTER">"page number " . $this->getPageNum()</COL>
    </ROW>
  </FOOTER>
</PAGE>
```

Too much green, looks like a salad! ;-) The HEADER and FOOTER are well known right now and we just changed the page size from 10 to 20, the point here is the BORDER="0", removing the borders on the PAGE and the XHTML element. Inside this element you can put valid XHTML code to render on your page.

XHTML are different from HTML because all its elements must have a start and an end tags. Notice the <BR> and <HR> tags there, they are coded like <BR/> and <HR/>.

Another thing to remember on XHTML elements is that no matter how many rows you have inside of it, how much space is required to render it, it always will count the rows on the report based on your ROW element. So, on the example, we have a table with 3 rows, but it will count just 1 row when rendering the report.

Our report looks like this now:

JOHN DOE ENTERPRISES SALES REPORT				powered by phpreports
city: <b>Mirassol</b>				
Marcio Lambary	Book	This is the book 1	40.00	
Marcio Lambary	CD	This is the CD 1	10.00	
Marcio Lambary	CD	This is the CD 2	20.00	
Marcio Lambary	CD	This is the CD 3	30.00	
total of Mirassol			<b>100</b>	
city: <b>Rio Preto</b>				
Ana Carolina	Book	This is the book 1	30.00	
Ana Carolina	Book	This is the book 2	40.00	
Ana Carolina	Book	This is the book 3	50.00	
Ana Carolina	Book	This is the book 4	60.00	
Ana Carolina	Book	This is the book 5	70.00	
Ana Carolina	CD	This is the CD 1	10.00	
Ana Carolina	CD	This is the CD 2	20.00	
Eustaquio Rangel	Book	This is the book 1	40.00	
Eustaquio Rangel	Book	This is the book 2	50.00	
Eustaquio Rangel	CD	This is the CD 1	10.00	
page total			<b>480</b>	
				page number 1

Figure 3.6: XHTML code inside



## 3.7 Sub groups

Things are going fine, but now I need to know how much every customer bought from me, and subtotals by the type of the items. So, I need a customer GROUP and an item GROUP.

**IMPORTANT** Never, never, change the correct order of GROUP, HEADER, FOOTER, and so, another GROUP, or your report will not work and **I'll kill you, yes, I'll**. To check your XML report layout file, you can use RXP (<http://www.cogsci.ed.ac.uk/richard/rxp.html>) this way:

```
rxp -V -o 0 <yourfile.xml>
```

To have an idea about how it can help, I was spending the last hour checking why my FOOTERS didn't work and forgot to check the file. Shame on me. Stupid. But now, after checking it with rxp, everything runs fine. The next version of sales.xml we'll see will be the validated one. PS: the elements inside the XHTML don't validate because it will give a lot of work to declare all of them inside the DTD file! So if you see some errors regarding XHTML stuff, don't worry about it.

Let's take a look on our brand new sales.xml:

```
<?xml version="1.0" encoding="ISO-8859-1" standalone="no"?>
<!DOCTYPE REPORT SYSTEM "PHPReport.dtd">
<REPORT MARGINWIDTH="5" MARGINHEIGHT="5">
  <TITLE>Sales Report</TITLE>
  <BACKGROUND_COLOR>#FFFFFF</BACKGROUND_COLOR>
  <CSS>http://localhost/phpreports2/sales.css</CSS>
  <SQL>select CITY,NAME,TYPE,ITEM,VALUE from sales order by CITY,NAME,TYPE,ITEM</SQL>
  <CONNECTION>localhost</CONNECTION>
  <INTERFACE>mysql</INTERFACE>
  <DATABASE>phpreports</DATABASE>
  <NO_DATA_MSG>No data was found, check your query</NO_DATA_MSG>
  <PAGE BORDER="0" SIZE="30" CELLSPACING="0" CELLPADDING="3">
    <HEADER>
      <ROW>
        <COL CELLCLASS="HEADER" COLSPAN="3">
          <XHTML>
            <TABLE BORDER="0" CELLPADDING="2" CELLSPACING="0" WIDTH="100%">
              <TR>
                <TD CLASS="HEADER">
                  <b>JOHN DOE ENTERPRISES</b>
                </TD>
                <TD ROWSPAN="2" CLASS="HEADER" style="background:#A0A0A0;color:#FFFFFF;"
ALIGN="CENTER">
                  powered by<br/><b>phpreports</b>
                </TD>
              </TR>
              <TR>
                <TD CLASS="HEADER">
                  <b>SALES REPORT</b>
                </TD>
                <TD COLSPAN="3"><HR/></TD>
              </TR>
            </TABLE>
          </XHTML>
        </COL>
      </ROW>
    </HEADER>
    <FOOTER>
      <ROW>
        <COL ALIGN="RIGHT" CELLCLASS="FOOTER" COLSPAN="2">page total</COL>
        <COL TYPE="EXPRESSION" CELLCLASS="FOOTER" TEXTCLASS="BOLD" ALIGN="RIGHT" NUMBERFORMATEX="2">
$this->getSum("VALUE")</COL>
      </ROW>
      <ROW>
        <COL ALIGN="RIGHT" COLSPAN="3" TYPE="EXPRESSION" CELLCLASS="FOOTER">"page number ".
$this->getPageNum()</COL>
      </ROW>
    </FOOTER>
  </PAGE>
</REPORT>
```

```

</FOOTER>
</PAGE>
<GROUPS>
  <GROUP NAME="city" EXPRESSION="CITY">
    <HEADER>
      <ROW>
        <COL ALIGN="RIGHT" CELLCLASS="HEADER" WIDTH="50">city:</COL>
        <COL TYPE="EXPRESSION" CELLCLASS="HEADER" TEXTCLASS="BOLD" COLSPAN="1">
$this->getValue("CITY")</COL>
        <COL CELLCLASS="HEADER"></COL>
      </ROW>
    </HEADER><FOOTER>
      <ROW>
        <COL COLSPAN="3" CELLCLASS="FOOTER"></COL>
      </ROW>
      <ROW>
        <COL ALIGN="RIGHT" TYPE="EXPRESSION" CELLCLASS="FOOTER" COLSPAN="2">
"total of ".$this->getValue("CITY")</COL>
        <COL TYPE="EXPRESSION" CELLCLASS="FOOTER" TEXTCLASS="BOLD" ALIGN="RIGHT"
NUMBERFORMATEX="2" COLSPAN="2">$this->getSum("VALUE")</COL>
      </ROW>
      <ROW>
        <COL COLSPAN="3" CELLCLASS="FOOTER"></COL>
      </ROW>
    </FOOTER>
  <GROUP NAME="customer" EXPRESSION="NAME">
    <HEADER>
      <ROW>
        <COL CELLCLASS="HEADER" ALIGN="RIGHT" WIDTH="50">customer:</COL>
        <COL CELLCLASS="HEADER" TEXTCLASS="BOLD" TYPE="EXPRESSION" COLSPAN="1">
$this->getValue("NAME")</COL>
        <COL CELLCLASS="HEADER"></COL>
      </ROW>
    </HEADER>
    <FOOTER>
      <ROW>
        <COL ALIGN="RIGHT" TYPE="EXPRESSION" CELLCLASS="FOOTER" COLSPAN="2">
"total of ".$this->getValue("NAME")</COL>
        <COL TYPE="EXPRESSION" CELLCLASS="FOOTER" TEXTCLASS="BOLD" ALIGN="RIGHT"
NUMBERFORMATEX="2">$this->getSum("VALUE")</COL>
      </ROW>
      <ROW>
        <COL COLSPAN="3" TYPE="EXPRESSION" CELLCLASS="FOOTER" TEXTCLASS="BOLD"
ALIGN="RIGHT">($this->getSum("VALUE")>200?*** Free shipping bonus!***:"Regular shipping charging")</COL>
      </ROW>
    </FOOTER>
  <GROUP NAME="item" EXPRESSION="TYPE">
    <HEADER>
      <ROW>
        <COL CELLCLASS="HEADER" ALIGN="RIGHT" WIDTH="50">type:</COL>
        <COL CELLCLASS="HEADER" TEXTCLASS="BOLD" TYPE="EXPRESSION" COLSPAN="1">
$this->getValue("TYPE")</COL>
        <COL CELLCLASS="HEADER"></COL>
      </ROW>
      <ROW>
        <COL CELLCLASS="HEADER" TEXTCLASS="BOLD" WIDTH="50">title</COL>
        <COL CELLCLASS="HEADER" TEXTCLASS="BOLD" ALIGN="RIGHT" COLSPAN="2">${</COL>
      </ROW>
    </HEADER>
    <FIELDS>
      <ROW>
        <COL TYPE="FIELD" CELLCLASSEVEN="EVEN" CELLCLASSODD="ODD" ALIGN="LEFT" COLSPAN="2">
ITEM</COL>
        <COL TYPE="FIELD" CELLCLASSEVEN="EVEN" CELLCLASSODD="ODD" ALIGN="RIGHT">VALUE</COL>
      </ROW>
    </FIELDS>
    <FOOTER>
      <ROW>
        <COL ALIGN="RIGHT" TYPE="EXPRESSION" CELLCLASS="FOOTER" COLSPAN="2">
"total of ".$this->getValue("TYPE")."(s)"</COL>
        <COL TYPE="EXPRESSION" CELLCLASS="FOOTER" TEXTCLASS="BOLD" ALIGN="RIGHT"

```

```

NUMBERFORMATEX="2">$this->getSum("VALUE")</COL>
    </ROW>
  </FOOTER>
</GROUP>    </GROUP>
</GROUP>
</GROUPS>
</REPORT>

```

Checking the **green** stuff, you see a parameter called NUMBERFORMATEX with a 2 value there. It's a way to format your values, to decimal places. The 2 means that you will put 2 decimal places there. It will also format your values with decimal and thousand separators based on the setLocale you choose on your PHP code.

Another change was a COL with an EXPRESSION which calculates if the amount is greater than 200, if so, it prints a message **\*\*\* Free shipping bonus! \*\*\*** to tell you that order will be shipped free of charge, otherwise will be a Regular shipping charging order.

This change was helpful to show how PHP code can be used inside a COL.

But the important stuff here is the GROUPS. Check the way they are. Don't forget this order:

**GROUP,HEADER,FOOTER,GROUP,HEADER,FOOTER ...**

We have the first group called city with the break expression on the CITY field. Inside of it we have the customer group breaking on the NAME field. And inside of it we have the item group breaking on the TYPE field. After playing with some COLSPAN, TEXTCLASS and CELLCLASS, we have this kind of output (I'll put all the pages now ):

JOHN DOE ENTERPRISES		powered by
SALES REPORT		phpreports
city:	<b>Mirassol</b>	
customer:	<b>Marcio Lambarly</b>	
type:	<b>Book</b>	
<b>title</b>		<b>\$</b>
This is the book 1	40.00	
total of Book(s)	<b>40.00</b>	
type:	<b>CD</b>	
<b>title</b>		<b>\$</b>
This is the CD 1	10.00	
This is the CD 2	20.00	
This is the CD 3	30.00	
total of CD(s)	<b>60.00</b>	
total of Marcio Lambarly	<b>100.00</b>	
<b>Regular shipping charging</b>		
total of Mirassol	<b>100.00</b>	
city:	<b>Rio Preto</b>	
customer:	<b>Ana Carolina</b>	
type:	<b>Book</b>	
<b>title</b>		<b>\$</b>
This is the book 1	30.00	
This is the book 2	40.00	
This is the book 3	50.00	
This is the book 4	60.00	
This is the book 5	70.00	
total of Book(s)	<b>250.00</b>	
page total	<b>350.00</b>	
page number	1	

JOHN DOE ENTERPRISES		powered by
SALES REPORT		phpreports
This is the CD 1	10.00	
This is the CD 2	20.00	
total of CD(s)	<b>30.00</b>	
total of Ana Carolina	<b>280.00</b>	
<b>**** Free shipping bonus!****</b>		
customer:	<b>Eustaquio Rangel</b>	
type:	<b>Book</b>	
<b>title</b>		<b>\$</b>
This is the book 1	40.00	
This is the book 2	50.00	
total of Book(s)	<b>90.00</b>	
type:	<b>CD</b>	
<b>title</b>		<b>\$</b>
This is the CD 1	10.00	
This is the CD 2	20.00	
This is the CD 3	30.00	
total of CD(s)	<b>60.00</b>	
total of Eustaquio Rangel	<b>150.00</b>	
<b>Regular shipping charging</b>		
total of Rio Preto	<b>430.00</b>	
city:	<b>Sao Paulo</b>	
customer:	<b>Andre Kada</b>	
type:	<b>Book</b>	
<b>title</b>		<b>\$</b>
This is the book 1	50.00	
page total	<b>230.00</b>	
page number	2	

JOHN DOE ENTERPRISES		powered by
SALES REPORT		phpreports
This is the book 2	60.00	
This is the book 3	70.00	
total of Book(s)	<b>180.00</b>	
type:	<b>CD</b>	
<b>title</b>		<b>\$</b>
This is the CD 1	10.00	
This is the CD 2	20.00	
This is the CD 3	30.00	
This is the CD 4	40.00	
This is the CD 5	50.00	
total of CD(s)	<b>150.00</b>	
total of Andre Kada	<b>330.00</b>	
<b>**** Free shipping bonus!****</b>		
total of Sao Paulo	<b>330.00</b>	
page total	<b>280.00</b>	

Cool!!! Grouping by CITY, then by CUSTOMER, then by TYPE, with sum for all the groups! You can use getSum() to do that and some other grouping functions also, please check the function reference below.

## 3.8 The grand total

Remember how I explain the layers on the beginning of this document? The document, page and group layers? Ok, we already worked with page and group layers, but we have also the document one.

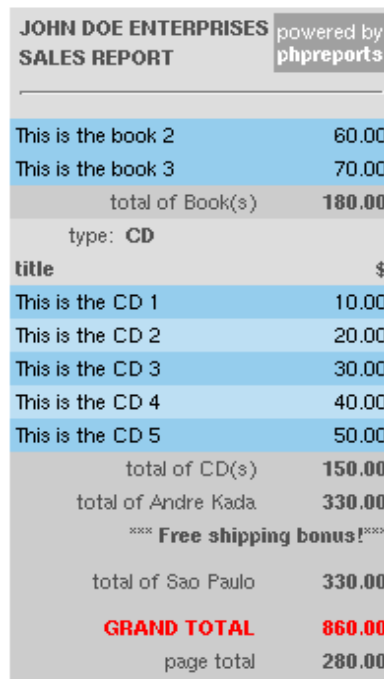
If you want something on the beginning or on the end of your report, that's the element you must use. First let's put another CSS class on our sales.css file to identify the grand total:

```
.BOLDRED {
  color: #FF0000;
  font-weight: bold;
}
```

And now put this before your PAGE element on sales.xml:

```
<DOCUMENT>
  <FOOTER>
    <ROW>
      <COL COLSPAN="2" CELLCLASS="FOOTER" TEXTCLASS="BOLDRED" ALIGN="RIGHT">GRAND TOTAL</COL>
      <COL CELLCLASS="FOOTER" TEXTCLASS="BOLDRED" TYPE="EXPRESSION" ALIGN="RIGHT" NUMBERFORMAT="2">
        $this->getSum("VALUE")</COL>
    </ROW>
  </FOOTER>
</DOCUMENT>
```

It will make the last page looks like this:



The screenshot shows a sales report for 'JOHN DOE ENTERPRISES' powered by 'phpreports'. The report lists items and their prices, with a grand total highlighted in red. The items are grouped by type: Books and CDs. The grand total is 860.00, and the page total is 280.00.

JOHN DOE ENTERPRISES		powered by
SALES REPORT		phpreports
This is the book 2	60.00	
This is the book 3	70.00	
total of Book(s)	180.00	
type: CD		
title	\$	
This is the CD 1	10.00	
This is the CD 2	20.00	
This is the CD 3	30.00	
This is the CD 4	40.00	
This is the CD 5	50.00	
total of CD(s)	150.00	
total of Andre Kada	330.00	
**** Free shipping bonus!****		
total of Sao Paulo	330.00	
<b>GRAND TOTAL</b>	<b>860.00</b>	
page total	280.00	

Figure 3.7: The grand total

Now you have a complete report!

### 3.9 Playing with the report

Now let's play a little with our report. First thing we need is remove the type group, and put an type column there. Make your sales.xml file this way:

```
<?xml version="1.0" encoding="ISO-8859-1" standalone="no"?>
<!DOCTYPE REPORT SYSTEM "PHPReport.dtd">
<REPORT MARGINWIDTH="5" MARGINHEIGHT="5">
  <TITLE>Sales Report</TITLE>
  <BACKGROUND_COLOR>#FFFFFF</BACKGROUND_COLOR>
  <CSS>http://localhost/phpreports2/sales.css</CSS>
  <SQL>select CITY,NAME,TYPE,ITEM,VALUE from sales order by CITY,NAME,TYPE,ITEM</SQL>
  <CONNECTION>localhost</CONNECTION>
  <INTERFACE>mysql</INTERFACE>
  <DATABASE>phpreports</DATABASE>
  <NO_DATA_MSG>No data was found, check your query</NO_DATA_MSG>
  <DOCUMENT>
    <FOOTER>
      <ROW>
        <COL COLSPAN="2" CELLCLASS="FOOTER" TEXTCLASS="BOLDRED" ALIGN="RIGHT">GRAND TOTAL</COL>
        <COL CELLCLASS="FOOTER" TEXTCLASS="BOLDRED" TYPE="EXPRESSION" ALIGN="RIGHT"
NUMBERFORMATEX="2">$this->getSum("VALUE")</COL>
      </ROW>
    </FOOTER>
  </DOCUMENT>
  <PAGE BORDER="0" SIZE="30" CELLSPACING="0" CELLPADDING="3">
    <HEADER>
      <ROW>
        <COL CELLCLASS="HEADER" COLSPAN="3">
          <XHTML>
            <TABLE BORDER="0" CELLPADDING="2" CELLSPACING="0" WIDTH="100%">
              <TR>
                <TD CLASS="HEADER">
                  <b>JOHN DOE ENTERPRISES</b>
                </TD>
                <TD ROWSPAN="2" CLASS="HEADER" style="background:#A0A0A0;color:#FFFFFF;"
ALIGN="CENTER">
                  powered by<br/><b>phpreports</b>
                </TD>
              </TR>
              <TR>
                <TD CLASS="HEADER">
                  <b>SALES REPORT</b>
                </TD>
                <TD COLSPAN="3"><HR/></TD>
              </TR>
            </TABLE>
          </XHTML>
        </COL>
      </ROW>
    </HEADER>
  </FOOTER>
  <ROW>
    <COL ALIGN="RIGHT" CELLCLASS="FOOTER" COLSPAN="2">page total</COL>
    <COL TYPE="EXPRESSION" CELLCLASS="FOOTER" TEXTCLASS="BOLD" ALIGN="RIGHT"
NUMBERFORMATEX="2">$this->getSum("VALUE")</COL>
  </ROW>
  <ROW>
    <COL ALIGN="RIGHT" COLSPAN="3" TYPE="EXPRESSION" CELLCLASS="FOOTER">"page number ".
$this->getPageNum()</COL>
  </ROW>
</FOOTER>
</PAGE>
<GROUPS>
  <GROUP NAME="city" EXPRESSION="CITY">
    <HEADER>
      <ROW>
        <COL ALIGN="RIGHT" CELLCLASS="HEADER" WIDTH="50">city:</COL>
        <COL TYPE="EXPRESSION" CELLCLASS="HEADER" TEXTCLASS="BOLD" COLSPAN="2">
$this->getValue("CITY")</COL>
      </ROW>
    </HEADER>
  </GROUP>
</GROUPS>
</REPORT>
```

```

</HEADER><FOOTER>
  <ROW>
    <COL COLSPAN="3" CELLCLASS="FOOTER"></COL>
  </ROW>
  <ROW>
    <COL ALIGN="RIGHT" TYPE="EXPRESSION" CELLCLASS="FOOTER" COLSPAN="2">"total of ".
$this->getValue("CITY")</COL>
    <COL TYPE="EXPRESSION" CELLCLASS="FOOTER" TEXTCLASS="BOLD" ALIGN="RIGHT"
NUMBERFORMATEX="2">$this->getSum("VALUE")</COL>
  </ROW>
  <ROW>
    <COL COLSPAN="3" CELLCLASS="FOOTER"></COL>
  </ROW>
</FOOTER>
<GROUP NAME="customer" EXPRESSION="NAME">
  <HEADER>
    <ROW>
      <COL CELLCLASS="HEADER" ALIGN="RIGHT" WIDTH="50">customer:</COL>
      <COL CELLCLASS="HEADER" TEXTCLASS="BOLD" TYPE="EXPRESSION" COLSPAN="2">
$this->getValue("NAME")</COL>
    </ROW>
    <ROW>
      <COL CELLCLASS="HEADER" TEXTCLASS="BOLD" ALIGN="LEFT">type</COL>
      <COL CELLCLASS="HEADER" TEXTCLASS="BOLD" ALIGN="LEFT">title</COL>
      <COL CELLCLASS="HEADER" TEXTCLASS="BOLD" ALIGN="RIGHT">$/</COL>
    </ROW>
  </HEADER>
  <FIELDS>
    <ROW>
      <COL TYPE="FIELD" CELLCLASSEVEN="EVEN" CELLCLASSODD="ODD" ALIGN="LEFT">TYPE</COL>
      <COL TYPE="FIELD" CELLCLASSEVEN="EVEN" CELLCLASSODD="ODD" ALIGN="LEFT">ITEM</COL>
      <COL TYPE="FIELD" CELLCLASSEVEN="EVEN" CELLCLASSODD="ODD" ALIGN="RIGHT">VALUE</COL>
    </ROW>
  </FIELDS>
  <FOOTER>
    <ROW>
      <COL ALIGN="RIGHT" TYPE="EXPRESSION" CELLCLASS="FOOTER" COLSPAN="2">
"total of ". $this->getValue("NAME")</COL>
      <COL TYPE="EXPRESSION" CELLCLASS="FOOTER" TEXTCLASS="BOLD" ALIGN="RIGHT"
NUMBERFORMATEX="2">$this->getSum("VALUE")</COL>
    </ROW>
    <ROW>
      <COL COLSPAN="3" TYPE="EXPRESSION" CELLCLASS="FOOTER" TEXTCLASS="BOLD"
ALIGN="RIGHT">($this->getSum("VALUE")>200?"*** Free shipping bonus!***":"Regular shipping charging")</COL>
    </ROW>
  </FOOTER>
</GROUP> </GROUP>
</GROUPS>
</REPORT>

```

Now the report looks like this:

JOHN DOE ENTERPRISES		powered by
SALES REPORT		phpreports
city: <b>Mirassol</b>		
customer: <b>Marcio Lambary</b>		
type	title	\$
Book	This is the book 1	40.00
CD	This is the CD 1	10.00
CD	This is the CD 2	20.00
CD	This is the CD 3	30.00
total of Marcio Lambary		100.00
Regular shipping charging		
total of Mirassol		100.00
city: <b>Rio Preto</b>		
customer: <b>Ana Carolina</b>		
type	title	\$
Book	This is the book 1	30.00
Book	This is the book 2	40.00
Book	This is the book 3	50.00
Book	This is the book 4	60.00
Book	This is the book 5	70.00
CD	This is the CD 1	10.00
CD	This is the CD 2	20.00
total of Ana Carolina		280.00
**** Free shipping bonus ****		
customer: <b>Eustaquio Rangel</b>		
type	title	\$
Book	This is the book 1	40.00
page total		420.00
page number 1		

JOHN DOE ENTERPRISES		powered by
SALES REPORT		phpreports
Book	This is the book 2	50.00
CD	This is the CD 1	10.00
CD	This is the CD 2	20.00
CD	This is the CD 3	30.00
total of Eustaquio Rangel		150.00
Regular shipping charging		
total of Rio Preto		430.00
city: <b>Sao Paulo</b>		
customer: <b>Andre Kada</b>		
type	title	\$
Book	This is the book 1	50.00
Book	This is the book 2	60.00
Book	This is the book 3	70.00
CD	This is the CD 1	10.00
CD	This is the CD 2	20.00
CD	This is the CD 3	30.00
CD	This is the CD 4	40.00
CD	This is the CD 5	50.00
total of Andre Kada		330.00
**** Free shipping bonus ****		
total of Sao Paulo		330.00
<b>GRAND TOTAL</b>		<b>860.00</b>
page total		440.00
page number 2		

See that Book, Book, Book, CD, CD, CD? There is an option on the COL element that refuses to print the COL value if it's the same that the last one printed. It's the SUPPRESS option, and it's a boolean one. Change the COL (inside the FIELDS) who contains the field TYPE to use SUPPRESS=TRUE.

```
<FIELDS>
  <ROW>
    <COL TYPE="FIELD" CELLCLASSEVEN="EVEN" CELLCLASSODD="ODD" ALIGN="LEFT" SUPPRESS="TRUE">TYPE</COL>
    <COL TYPE="FIELD" CELLCLASSEVEN="EVEN" CELLCLASSODD="ODD" ALIGN="LEFT">ITEM</COL>
    <COL TYPE="FIELD" CELLCLASSEVEN="EVEN" CELLCLASSODD="ODD" ALIGN="RIGHT">VALUE</COL>
  </ROW>
</FIELDS>
```

This is what happens:



JOHN DOE ENTERPRISES		powered by
SALES REPORT		phpreports
city: <b>Mirassol</b>		
customer: <b>Marcio Lambary</b>		
type	title	\$
Book	This is the book 1	40.00
CD	This is the CD 1	10.00
	This is the CD 2	20.00
	This is the CD 3	30.00
total of Marcio Lambary		100.00
Regular shipping charging		
total of Mirassol		100.00
city: <b>Rio Preto</b>		
customer: <b>Ana Carolina</b>		
type	title	\$
Book	This is the book 1	30.00
	This is the book 2	40.00
	This is the book 3	50.00
	This is the book 4	60.00
	This is the book 5	70.00
CD	This is the CD 1	10.00
	This is the CD 2	20.00
total of Ana Carolina		280.00
**** Free shipping bonus !****		
customer: <b>Eustaquio Rangel</b>		
type	title	\$
Book	This is the book 1	40.00
page total		420.00
page number 1		

JOHN DOE ENTERPRISES		powered by
SALES REPORT		phpreports
	This is the book 2	50.00
CD	This is the CD 1	10.00
	This is the CD 2	20.00
	This is the CD 3	30.00
total of Eustaquio Rangel		150.00
Regular shipping charging		
total of Rio Preto		430.00
city: <b>Sao Paulo</b>		
customer: <b>Andre Kada</b>		
type	title	\$
Book	This is the book 1	50.00
	This is the book 2	60.00
	This is the book 3	70.00
CD	This is the CD 1	10.00
	This is the CD 2	20.00
	This is the CD 3	30.00
	This is the CD 4	40.00
	This is the CD 5	50.00
total of Andre Kada		330.00
**** Free shipping bonus !****		
total of Sao Paulo		330.00
<b>GRAND TOTAL</b>		<b>860.00</b>
page total		440.00
page number 2		

**IMPORTANT TIP** Sometimes the width of the pages can be variable. It's because the field width of one page are larger than the others on the other pages. To fix it, it's a good idea put some fixed size on your COL elements using the WIDTH parameter, so the fields will be of the same width for all your report.

Now, looking at the second page, you can see that the first row was printed This is the Book 2, but if I didn't named the item this way you'll need to look on the last row of the first page to check the type of the first item of the second page, and it is a book. I mean, it don't mention it's type because it is using the SUPPRESS parameter.

To fix this and reprint the suppressed values on every page break, put a

`RESET_SUPPRESS_ON_PAGEBREAK=TRUE`

on your GROUP customer:

JOHN DOE ENTERPRISES SALES REPORT		powered by
		phpreports
city: Mirassol		
customer: Marcio Lambary		
type	title	\$
Book	This is the book 1	40.00
CD	This is the CD 1	10.00
	This is the CD 2	20.00
	This is the CD 3	30.00
total of Marcio Lambary		100.00
Regular shipping charging		
total of Mirassol		100.00
city: Rio Preto		
customer: Ana Carolina		
type	title	\$
Book	This is the book 1	30.00
	This is the book 2	40.00
	This is the book 3	50.00
	This is the book 4	60.00
	This is the book 5	70.00
CD	This is the CD 1	10.00
	This is the CD 2	20.00
total of Ana Carolina		280.00
*** Free shipping bonus!***		
customer: Eustaquio Rangel		
type	title	\$
Book	This is the book 1	40.00
page total		420.00
page number		1

JOHN DOE ENTERPRISES SALES REPORT		powered by
		phpreports
Book This is the book 2 50.00		
CD This is the CD 1 10.00		
This is the CD 2 20.00		
This is the CD 3 30.00		
total of Eustaquio Rangel		150.00
Regular shipping charging		
total of Rio Preto		430.00
city: Sao Paulo		
customer: Andre Kada		
type	title	\$
Book	This is the book 1	50.00
	This is the book 2	60.00
	This is the book 3	70.00
CD	This is the CD 1	10.00
	This is the CD 2	20.00
	This is the CD 3	30.00
	This is the CD 4	40.00
	This is the CD 5	50.00
total of Andre Kada		330.00
*** Free shipping bonus!***		
total of Sao Paulo		330.00
<b>GRAND TOTAL</b>		<b>860.00</b>
page total		440.00
page number		2

Now the values are still suppressed, but you can see that the first one on the page is not. To make more clear on the second page of what we're dealing with, maybe we can reprint the GROUP HEADERS on every page break. To make this, use:

`REPRINT_HEADER_ON_PAGEBREAK="TRUE"`

on the GROUPs elements you want. I put it on both GROUPs (city and customer) and got this:

JOHN DOE ENTERPRISES SALES REPORT		powered by
		phpreports
city: Mirassol		
customer: Marcio Lambary		
type	title	\$
Book	This is the book 1	40.00
CD	This is the CD 1	10.00
	This is the CD 2	20.00
	This is the CD 3	30.00
total of Marcio Lambary		100.00
Regular shipping charging		
total of Mirassol		100.00
city: Rio Preto		
customer: Ana Carolina		
type	title	\$
Book	This is the book 1	30.00
	This is the book 2	40.00
	This is the book 3	50.00
	This is the book 4	60.00
	This is the book 5	70.00
CD	This is the CD 1	10.00
	This is the CD 2	20.00
total of Ana Carolina		280.00
*** Free shipping bonus!***		
customer: Eustaquio Rangel		
type	title	\$
Book	This is the book 1	40.00
page total		420.00
page number		1

JOHN DOE ENTERPRISES SALES REPORT		powered by
		phpreports
city: Rio Preto		
customer: Eustaquio Rangel		
type	title	\$
Book	This is the book 2	50.00
CD	This is the CD 1	10.00
	This is the CD 2	20.00
	This is the CD 3	30.00
total of Eustaquio Rangel		150.00
Regular shipping charging		
total of Rio Preto		430.00
city: Sao Paulo		
customer: Andre Kada		
type	title	\$
Book	This is the book 1	50.00
	This is the book 2	60.00
	This is the book 3	70.00
CD	This is the CD 1	10.00
	This is the CD 2	20.00
	This is the CD 3	30.00
	This is the CD 4	40.00
	This is the CD 5	50.00
total of Andre Kada		330.00
*** Free shipping bonus!***		
total of Sao Paulo		330.00
<b>GRAND TOTAL</b>		<b>860.00</b>
page total		440.00
page number		2

JOHN DOE ENTERPRISES SALES REPORT		powered by
		phpreports
<b>GRAND TOTAL</b>		<b>860.00</b>
page total		0.00
page number		3

If you want a page break on the first GROUP there, put the attribute:

`PAGEBREAK=TRUE`

on that GROUP. Let me explain how will be a little complicated put this parameter on the inner GROUPs.

Suppose or sample, if we put the page break on the customer GROUP, every time it breaks a new PAGE will be opened. When this happens, to close the current PAGE, it will print the customer FOOTER, and so what? Do we must print the parent GROUP footer or not?

On our example the answer is yes, but if we have more than a customer there? If we print the city FOOTER, will be invalid, because all the customers of that city are not printed yet.

And will be weird just a FOOTER printed on some other page. Example:

JOHN DOE ENTERPRISES		powered by
SALES REPORT		phpreports
city: <b>Mirassol</b>		
customer: <b>Marcio Lambary</b>		
type	title	\$
Book	This is the book 1	40.00
CD	This is the CD 1	10.00
	This is the CD 2	20.00
	This is the CD 3	30.00
total of Marcio Lambary		<b>100.00</b>
Regular shipping charging		
total of Mirassol		<b>100.00</b>
page total		<b>100.00</b>
page number 1		

JOHN DOE ENTERPRISES		powered by
SALES REPORT		phpreports
city: <b>Rio Preto</b>		
customer: <b>Ana Carolina</b>		
type	title	\$
Book	This is the book 1	30.00
	This is the book 2	40.00
	This is the book 3	50.00
	This is the book 4	60.00
	This is the book 5	70.00
CD	This is the CD 1	10.00
	This is the CD 2	20.00
total of Ana Carolina		<b>280.00</b>
*** Free shipping bonus!***		
customer: <b>Eustaquio Rangel</b>		
type	title	\$
Book	This is the book 1	40.00
	This is the book 2	50.00
CD	This is the CD 1	10.00
	This is the CD 2	20.00
	This is the CD 3	30.00
total of Eustaquio Rangel		<b>150.00</b>
Regular shipping charging		
total of Rio Preto		<b>430.00</b>
page total		<b>430.00</b>
page number 2		

JOHN DOE ENTERPRISES		powered by
SALES REPORT		phpreports
city: <b>Sao Paulo</b>		
customer: <b>Andre Kada</b>		
type	title	\$
Book	This is the book 1	50.00
	This is the book 2	60.00
	This is the book 3	70.00
CD	This is the CD 1	10.00
	This is the CD 2	20.00
	This is the CD 3	30.00
	This is the CD 4	40.00
	This is the CD 5	50.00
total of Andre Kada		<b>330.00</b>
*** Free shipping bonus!***		
total of Sao Paulo		<b>330.00</b>
<b>GRAND TOTAL</b>		<b>860.00</b>
page total		<b>330.00</b>
page number 3		

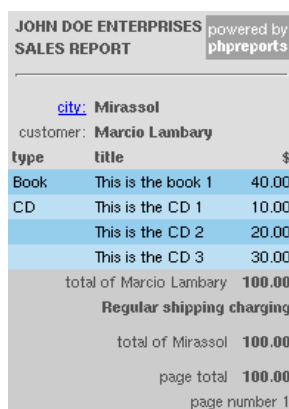
## 3.10 Links

Suppose now you want to put a link (to another URL, for example, it HTML based reports) in some COL you have on your report.

We can use the LINK element, and the LINK can have three types: **STATIC**, **DYNAMIC** and **EXPRESSION**.

Let's use the first of them, the **STATIC** type. Static links are immutable links you can put on your column, so it will never change. Suppose you know a website located at *http://www.whatisacity.com* (didn't checked it it exists, please, I'm using just as an example) and there they explain to you the concept of what is a city (pretty useless thing to make a website and register a domain, but forget about this) and you want that everytime the word city there on your CITY group header appears it have a link to that weird website. So you can change that COL to this:

```
<COL ALIGN="RIGHT" CELLCLASS="HEADER" WIDTH="50">
<LINK TYPE="STATIC" TARGET=".blank" TITLE="explain of what is a city">http://www.whatisacity.com</LINK>city:
</COL> Here is a screenshot of what happens:
```



The screenshot shows a report titled "JOHN DOE ENTERPRISES SALES REPORT" with a "powered by phpreports" logo. The report content includes a header for "city: Mirassol" and "customer: Marcio Lambary". Below this is a table with columns "type", "title", and "\$". The table lists items: "Book" (This is the book 1, 40.00), "CD" (This is the CD 1, 10.00), "CD" (This is the CD 2, 20.00), and "CD" (This is the CD 3, 30.00). Summary rows include "total of Marcio Lambary" (100.00), "Regular shipping charging", "total of Mirassol" (100.00), "page total" (100.00), and "page number 1". A blue link is visible under the word "city" in the header.

Figure 3.8: Static link

As you can see, there's a link on the city word there, and if you click on it you'll go to a new window, specified by the TARGET parameter, as on HTML stuff. It also have a TITLE that is like a tooltip that appears when you put the mouse over the link.

The second type, **DYNAMIC**, gave us some help on previous versions (older than 0.2). You can put there inside the LINK element the name of a report column to use as a link. It was useful when you build a URL inside the SQL query and show there, but it's better use the third type, the **EXPRESSION** one.

On the EXPRESSION type, you can put any expression you want, as on the COL elements. Suppose you want to check a URL called *getcustomerinfo.php* on your localhost using the name of your customer. Change the COL element there on the CUSTOMER GROUP HEADER to:

```
<COL CELLCLASS="HEADER" TEXTCLASS="BOLD" TYPE="EXPRESSION" COLSPAN="2">
<LINK TYPE="EXPRESSION" TARGET=".blank" TITLE="click here to check info about the customer">
"http://localhost/getcustomerinfo.php?name=".&this->getValue("NAME")</LINK>&this->getValue("NAME")
</COL>
```

Now you have:

JOHN DOE ENTERPRISES		powered by
SALES REPORT		phpreports
city:	Mirassol	
customer:	<a href="#">Marcio Lambary</a>	
type	title	\$
Book	This is the book 1	40.00
CD	This is the CD 1	10.00
	This is the CD 2	20.00
	This is the CD 3	30.00
total of Marcio Lambary		100.00
<b>Regular shipping charging</b>		
total of Mirassol		100.00
page total		100.00
page number		1

Figure 3.9: Dynamic link

with a link there on the customer name, pointing to *http://localhost/getcustomerinfo.php?name=<the customer's name here>*. On this case, the link is *http://localhost/getcustomerinfo.php?name=Marcio%20Lambary*.

## 3.11 Images

Images can be inserted in your report on two ways: you can specify any kind of parameters you can use in XHTML inside a XHTML element, or with the element IMG.

Inserting inside the XHTML element is good when you will just render your report as HTML code in your browser, but let's suppose we have an output plugin that really needs to know that what we have there is a graphic file that needs to be inserted in some way. On this case is better use the IMG element. I changed the PAGE HEADER to this:

```
<HEADER>
  <ROW>
    <COL CELLCLASS="HEADER" COLSPAN="2">
      <XHTML>
        <TABLE BORDER="0" CELLPADDING="2" CELLSPACING="0" WIDTH="100%">
          <TR>
            <TD CLASS="HEADER">
              <b>JOHN DOE ENTERPRISES</b>
            </TD>
            <TD ROWSPAN="2" CLASS="HEADER" style="background:#A0A0A0;color:#FFFFFF;" ALIGN="CENTER"> powered by<br/><img alt="penguin logo" data-bbox="548 508 582 532"/>
            </TD>
          </TR>
          <TR>
            <TD CLASS="HEADER"> <b>SALES REPORT</b>
            </TD>
          </TR>
          <TR>
            <TD COLSPAN="3"><hr/></TD>
          </TR>
        </TABLE>
      </XHTML>
    </COL>
    <COL CELLCLASS="HEADER"><IMG>http://localhost/img/penguin.png</IMG></COL>
  </ROW>
</HEADER>
```

And got this:


JOHN DOE ENTERPRISES		powered by	
SALES REPORT		phpreports	
city:	Mirassol		
customer:	Marcio Lambary		
type	title		\$
Book	This is the book 1	40.00	
CD	This is the CD 1	10.00	
CD	This is the CD 2	20.00	
CD	This is the CD 3	30.00	
total of Marcio Lambary		100.00	
Regular shipping charging			
total of Mirassol		100.00	
city:	Rio Preto		
customer:	Ana Carolina		
type	title		\$
Book	This is the book 1	30.00	
Book	This is the book 2	40.00	
Book	This is the book 3	50.00	
Book	This is the book 4	60.00	
Book	This is the book 5	70.00	
CD	This is the CD 1	10.00	
CD	This is the CD 2	20.00	
total of Ana Carolina		280.00	
*** Free shipping bonus!***			
customer:	Eustaquio Rangel		
type	title		\$
Book	This is the book 1	40.00	
page total		420.00	
page number		1	

Figure 3.10: IMG element

What a cute penguin. :-)

## 3.12 Bookmarks

Bookmarks are used as guides to key points on your report. For example, if you want to access quickly the cities on the report above, you could put some bookmarks on the COL element where the city belongs and describe it with the city name.

Bookmarks, as the LINK element, have 3 types: **STATIC**, **DYNAMIC** and **EXPRESSION**, and works exactly the same way as in LINK. For our example, let's use the EXPRESSION one. Change the COL element where the city name is to:

```
<COL TYPE="EXPRESSION" CELLCLASS="HEADER" TEXTCLASS="BOLD" COLSPAN="2">
<BOOKMARK TYPE="EXPRESSION">${this->getValue("CITY")}</BOOKMARK>${this->getValue("CITY")}
</COL>
```

Now you ask me: but where this bookmarks will fit in the report? If I put it all on the report, maybe you have a lot of bookmarks and will be horrible to print all that stuff. So I make them fit on another frame in the browser window.

**IMPORTANT TIP** To make this work, I mean, have the bookmarks and report rendered, I need to make a copy of the XML report result file and render it with different XSLT scripts, so you can notice two files on the tmp dir while it processes that.

If you try to run the report the same way we were running, with the BOOKMARKS there, you'll see the same report we were seeing till now. To use stuff like BOOKMARKS, we need to tell PHPReports we need (remember this)

*another way to render our XML result report file*

And this is done using **output plugins**.





## Chapter 4

# Output plugins

### 4.1 What is an output plugin?

Output plugins are ways to render/convert our report file into another stuff than HTML (or another ways to render HTML). On the BOOKMARKS case is specify a way to render the file using frames. Remember the graphic on the section *The PHP Code*:

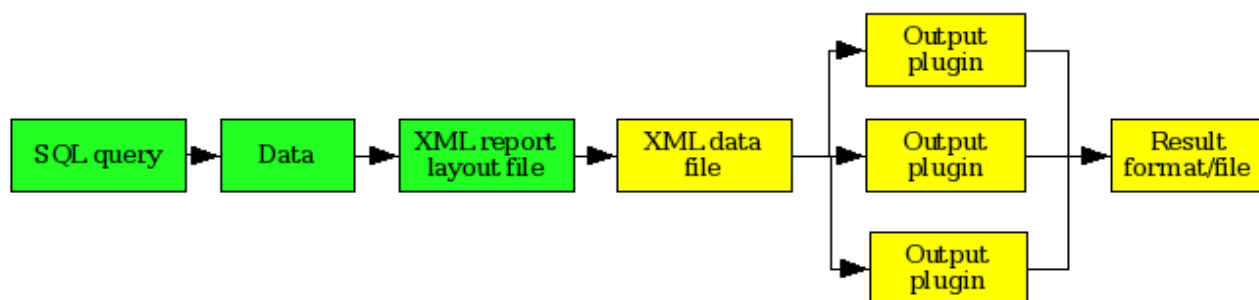


Figure 4.1: The creation/rendering path

We're dealing with the first yellow box here: our report is already processed with all the data (now its only text) inside of it, and we're going to send it to a output plugin and make it looks the way we want. If you didn't noticed yet, even the simplest report works this way, with the default output plugin that renders it on a simple HTML page.

So, to tell PHPReports that now we need to see our report with a frameset showing us our bookmarks, we change our sales.php to this:

```
<?php
    include_once "PHPReportMaker.php";
    $oRpt = new PHPReportMaker();
    $oRpt->setUser("taq");
    $oRpt->setPassword("*****");
    $oRpt->setXML("sales.xml");
    $oOut = $oRpt->createOutputPlugin("bookmarks");
    $oRpt->setOutputPlugin($oOut);
    $oRpt->run();
?>
```

Look the green lines: on the first one I create an output plugin called bookmarks and on the second one I told the report and it must use it. The result is something like this:

[Mirassol](#)  
[Rio Preto](#)  
[Sao Paulo](#)

type	title	\$
Book	This is the book 1	40.00
CD	This is the CD 1	10.00
	This is the CD 2	20.00
	This is the CD 3	30.00
total of Marcio Lambarly		100.00
<b>Regular shipping charging</b>		
total of Mirassol		100.00
page total		100.00
page number		1

Figure 4.2: Bookmarks

Uhn. That screenshot is a little ugly. Before you ask “hey, is there a way to make the BOOKMARKS a little more beautiful than that?” let’s spend just a little more time with the bookmark output plugin before going deeper again with the rest of the explanation about it.

Just change sales.xml to:

```
<COL TYPE="EXPRESSION" CELLCLASS="HEADER" TEXTCLASS="BOLD" COLSPAN="2">
<BOOKMARK TYPE="EXPRESSION" CELLCLASS="CITY" TEXTCLASS="BOOKMARK_TEXT">$this->getValue("CITY")</BOOKMARK>
$this->getValue("CITY")
</COL>
```

and sales.php to:

```
<?php
include_once "PHPReportMaker.php";
$oRpt = new PHPReportMaker();
$oRpt->setUser("taq");
$oRpt->setPassword("*****");
$oRpt->setXML("sales.xml");
$oOut = $oRpt->createOutputPlugin("bookmarks");
$oOut->setCSS("http://localhost/phpreports/css/bookmarks.css");
$oRpt->setOutputPlugin($oOut);
$oRpt->run();
?>
```

Put the correct file path there. It’s provided with the PHPReports package. That classes, CITY and BOOKMARK\_TEXT, are defined inside the bookmarks.css file. Now our output looks like this:

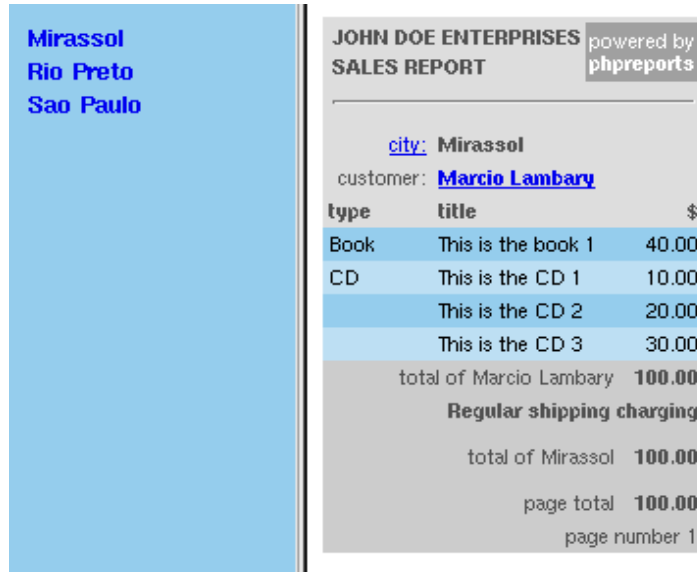


Figure 4.3: Bookmarks with CSS

Very better. You can control the way your BOOKMARKS will looks like on your css file. Now, back to the output plugins. We have four plugins that comes with the PHPReports package (to be honest there is one more, toolbar, but I kind of play with it, not a serious plugin :-)):

1. **The default plugin** it's the default plugin that renders you report on a single HTML page.
2. **The bookmarks plugin** we were talking about it now, render your report on right side of the frame and the bookmarks on the left side.
3. **The page plugin** renders your report page by page, with an index below it, just as Google searches.
4. **The text plugin** renders your report like a TXT file.

## 4.2 Default output plugin

There's nothing to say about it: if you don't specify an output plugin, it will be used. If you want to force it, you can use:

```
<?php
    include_once "PHPReportMaker.php";
    $oRpt = new PHPReportMaker();
    $oRpt->setUser("taq");
    $oRpt->setPassword("*****");
    $oRpt->setXML("sales.xml");
    $oOut = $oRpt->createOutputPlugin("default");
    $oRpt->setOutputPlugin($oOut);
    $oRpt->run();
?>
```

## 4.3 Bookmarks output plugin

### IMPORTANT NOTE

This plugin requires a /tmp directory with write permission under your DOCUMENT\_ROOT

We saw a good explanation about this one before.

All that you need to know is that it have the `setCSS(<string>)` and `getCSS()` methods, used to point to a CSS file where you can define the classes you use with your BOOKMARKs elements.

## 4.4 Page output plugin

### IMPORTANT NOTE

This plugin requires a `/tmp` directory with write permission under your `DOCUMENT_ROOT`

Now a new one. Using this plugin you can have your report viewed page-by-page.

We have some methods here:

- setIncr(number)** The number of pages where the index will be calculated.  
It's the number of pages you'll see on the index. The default is 10.
- setNext(string)** The label of the next page, from the current one.
- setPrev(string)** The label of the previous page, from the current one.
- setFirst(string)** The label of the first page.
- setLast(string)** The labels of the last page.

You can control how the page index will be displayed using CSS with these classes:

- PHPREPORTS\_PAGE\_CELL** Every number/link is inside a table cell.  
This element controls how it will look like.
- PHPREPORTS\_PAGE\_LINK** How the links will be rendered.
- PHPREPORTS\_PAGE\_LINK\_BOLD** How the bold link will be rendered.  
The bold link refers to the number of the current page.

Changing `sales.php` to:

```
<?php
    include_once "PHPReportMaker.php";
    $oRpt = new PHPReportMaker();
    $oRpt->setUser("taq");
    $oRpt->setPassword("*****");
    $oRpt->setXML("sales.xml");
    $oOut = $oRpt->createOutputPlugin("page");
    $oOut->setIncr(2);
    $oOut->setFirst("first");
    $oOut->setLast("last");
    $oOut->setNext("next");
    $oOut->setPrev("prev");
    $oRpt->setOutputPlugin($oOut);
    $oRpt->run();
?>
```

and inserting this on `sales.css`:

```
.PHPREPORTS_PAGE_CELL {
    font-family:"verdana","arial";
    font-size:12px;
    color:#000000;
}

.PHPREPORTS_PAGE_LINK {
    font-family:"verdana","arial";
    font-size:12px;
    color:#000000;
}

.PHPREPORTS_PAGE_LINK_BOLD {
```

```
font-size:16px;  
font-weight:bold;  
}
```

gives me this:

JOHN DOE ENTERPRISES		powered by
SALES REPORT		phpreports
city: <a href="#">Mirassol</a>		
customer: <a href="#">Marcio Lambarly</a>		
type	title	\$
Book	This is the book 1	40.00
CD	This is the CD 1	10.00
	This is the CD 2	20.00
	This is the CD 3	30.00
total of Marcio Lambarly		<b>100.00</b>
<b>Regular shipping charging</b>		
total of Mirassol		<b>100.00</b>
page total		<b>100.00</b>
page number 1		

[1](#) [2](#) [last](#)

Figure 4.4: Page output plugin

I hope you got the point about it. :-)

**IMPORTANT TIP** Since we need to re-render the report on every page, the temporary file is not erased. You need to erase it manually.

## 4.5 TXT output plugin

The TXT output plugin just transforms your report in plain text. You can see it on the browser or save it to a file.

Just change this on sales.php:

```
$oOut = $oRpt->createOutputPlugin("txt");
```

and you got this:

```
JOHN DOE ENTERPRISES
powered by
PHPReports
SALES REPORT

city: Mirassol
customer: Marcio Lambary
type title $
Book This is the book 1 40.00
CD This is the CD 1 10.00
  This is the CD 2 20.00
  This is the CD 3 30.00
total of Marcio Lambary 100.00
Regular shipping charging

total of Mirassol 100.00

page total 100.00
page number 1

-----
```

Ugly, but is what is supposed to do.

**IMPORTANT TIP** Since the browser change to the text file when converted, the temporary file is not erased. You need to erase it manually.

## 4.6 Creating your own output plugins

A cool thing is that if you want to render the report on a format that is not provided with the default package, you can, if you know some PHP and XML/XSLT programming. Let's first check the directory structure related to this kind of thing:

```
+--- output
|
+----- bookmarks
+----- common
+----- default
+----- page
+----- toolbar
+----- txt
```

We will pick a simple one, the TXT output plugin. Check inside the txt dir and you'll find:

```
PHPReportOutput.php txt.xsl
```

Check the PHPReportOutput.php file, take a look on the PHPReportOutputObject class, it has the following methods and properties:

- **set/getInput(file)** The XML used to render the report on this class format.

- **set/getOutput(file)** Where to write the report
- **setClean/isCleaning** Delete or not the input file
- **setJump/isJumping** Jump or not to the generated file (for example, the TXT files)
- **loadFrom(file)** Location of the saved report to load (see section 23)
- **run()** Run the output plugin

These basic operations allows you to create and run your output plugins, and to make a standard, always extend your plugins from this class, and use PHPReportOutput.php as your plugin main class.

Now that we know all this stuff we can check the file above again. It has just one method, the run method, where it gets the file and the url paths, based on the file path refers to the txt XLS file, based on the file path get the XML report file name, check if there is not an output file name, if not, based on the file path creates a new one, or use the one we told it to use, creates the XSLT processor, runs the XSLT transformations on the XML report file, free its resources, send the the generated txt file to the browser and if we told it to clean the XML report file, delete it.

This part is pretty simple, all you have to do is not forgot these rules:

- put it under the output directory
- extends your plugin from PHPReportOutputObject and
- name your plugin main file as PHPReportOutput.php

Everytime you use a plugin, it will search on a subdirectory under the output directory. Want to give a look on the whole process? Ok, here it goes, open the sales.php file:

```
<?phpinclude_once "PHPReportMaker.php";
    $oRpt = new PHPReportMaker();
    $oRpt->setUser("taq");
    $oRpt->setPassword("*****");
    $oRpt->setXML("sales.xml");
    $oOut = $oRpt->createOutputPlugin("default");
    $oRpt->setOutputPlugin($oOut);
    $oRpt->run();
?>
```

We inform the user, the password and the XML file, create the output plugin, put it on the report maker object and run the report make. So the report maker creates all the PHP code, run it asking the database about the data, create the XML report file with all the data described inside, and checks about the output plugin (if there is not one, it will create a default one), inform the output about the XML report data file name and run it.

To create the output plugin the report make calls it method createOutputPlugin, where it includes the desired PHPReportOutput.php file located on the PHPREPORTFILE/output/{plugin name}/ directory. So, this is the reason of a good plugin name and why all the plugin main classes needs the PHPReportOutput.php name. ;-)

Now we'll see the txt.xml file, used to transform out XML report data file into a plain text file:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform" version="1.0">
<xsl:output method="text" encoding="ISO-8859-1" indent="no"/>
<xsl:strip-space elements="*"/>
<xsl:template match="/RP">
    <xsl:apply-templates/>
</xsl:template>
<xsl:template match="PG">
    <xsl:apply-templates/>
    <xsl:text>&#10;</xsl:text>
    <xsl:text>-----&#10;</xsl:text>
    <xsl:text>&#10;</xsl:text>
</xsl:template>
<xsl:template match="R">
    <xsl:apply-templates/>
    <xsl:text>&#10;</xsl:text>
```

```

</xsl:template>
<xsl:template match="C">
  <xsl:apply-templates/>
  <xsl:text> </xsl:text>
</xsl:template>
<xsl:template match="LI">
  <xsl:value-of select="text()"/>
</xsl:template>
<xsl:template match="XHTML">
  <xsl:call-template name="HTML_ELEM"/>
</xsl:template>
<xsl:template match="BK">
</xsl:template>
<xsl:template match="node()[ancestor::XHTML]">
  <xsl:call-template name="HTML_ELEM"/>
</xsl:template>
<xsl:template match="text()[ancestor::XHTML]">
  <xsl:if test="string-length(.)>0 and not(node())">
    <xsl:value-of select="normalize-space()"/>
    <xsl:text>&#10;</xsl:text>
  </xsl:if>
</xsl:template>
<xsl:template name="HTML_ELEM">
  <xsl:apply-templates/>
</xsl:template>
</xsl:stylesheet>

```

If you know about XML/XSLT transformations it is pretty simple, if you don't, it is simple also, but check <http://www.w3schools.com> about all that stuff. To make it clear let's say that we have some templates there (the green words) and everytime XSLT find some of them, it applies the code there. For example, the first one is RP, who stands for REPORT, the second one is PG who stands for PAGE, look that every time it find a page it applies all the other templates found inside there and insert a line break, a row with some '-' chars and another page break. And so on.

But hey, where the hell did I name it as RP, PG, etc? I know you can be thinking about this as some stupid thing, but let me say to you that XML is a good way to describe your data, but it can be very verbose, so, in the name of the file sizes I choose to use this kind of thing. But don't worry, is not so bad. On the next section we will see more about the shorter tags and what they means.

## 4.7 The XML report data file

Inside this file (hey, is your report data there, give a look!) you can find some elements, and I'll try to tell you about them here.

- **RP** is the REPORT element, where all the other report elements are. It has the TITLE, CSS, BGCOLOR and BACKGROUND attributes (hey, those name are not shorter names, but, yes, they just appears maybe just once on the whole report)
- **PG** is the PAGE element, with **AL** (ALIGN), **BO** (BORDER), **WI** (WIDTH), **HE** (HEIGHT), **PA** (CELLPADDING) and **SP** (CELLSPACING) attributes
- **R** is the ROW element
- **C** is the COLUMN element, with **WI** (WIDTH), **HE** (HEIGHT), **AL** (ALIGN), **VA** (VALIGN), **CS** (COLSPAN), **RS** (ROWSPAN), **CC** (CELLCLASS), **TC** (TEXTCLASS), **LI** (LINK) and **BK** (BOOKMARK) attributes.
- **XHTML** all the XHTML code (take care here: you need to deal with XHTML code, and not XML here, but anyway, it's almost the same)

Just to give another example: if you open the `<phpreports_home>/output/common/PHPReport*.xsl` files, you'll see that every time we find a PG, we put a TABLE tag on the output, if we find a R we put a TR tag, if we find a C, we put a TD tag, and so on ...



# Chapter 5

## Some other things you need to know

### 5.1 Setting your parameters from the PHP code

I told you how to set your user name, password, database interface, database connection, database to be used and so on, on the XML layout file. Now let me explain to you why this could not be a good idea and how to fix that.

As the XML needs to be readable by the web server user, to read and process them, all the info available inside the files can be reached using a direct URL to the file, without using it to process your report. Suppose you have *sales.xml* under */var/www/hdtocs/phpreports*. Using

```
http://localhost/sales.xml
```

Voil! The browser will show you all the info inside the file. And, wow, if you put some user name and password there, your database will not be on a safe place now. So, it's a **very bad idea** put user names and passwords there (I already told you about that but I will repeat this how many times it will be necessary to put it in your mind).

Even the database name, interface and info about your connection is not really a good idea put there. People can find your local paths and knows what kind of database you have, and a name to try to connect there.

To avoid some headaches, use the methods like `setUser()`, `setPassword()`, `setDatabase()`, `setConnection()`, `setDatabaseInterface()` (check the API below). Try to use also `setSQL()` to avoid people knows about your queries, because they can know about your table names and so on.

And `setSQL()` is the answer when you need to pass parameters to your query. Suppose you have a query like

```
select * from customers where code=?
```

where `?` is a variable parameter. You can modify the query *before* you send it to processing, on your PHP code, so suppose you're running the query based on a `$_POST["customer_num"]` variable, you can send it like this:

```
$oRpt->setSQL("select * from customers where code=".$_POST["customer_num"]);
```

## 5.2 Organizing your directory structure

It's a very good idea to keep your PHPReports scripts not accessible to an URL typed on your browser address bar. I didn't find any exploit to PHPReports, but you know, if there's something that can improve the safety of the whole thing, let's do it.

So, unpack your PHPReports for example under */usr/lib*, */usr/include*, */usr/share*, it's your choice. If you're using Windows, unpack it under some directory out of the webserver scope (I can't give any tips about names on Windows :-). You need to make sure the directory is readable by the webserver user ("nobody" on the most Apache installations).

This is very good to you deal with your files and make easier an upgrade to another PHPReports version.

And put your XML data layout files out of webserver reach also! So people will not see your data layout on the web (no problem with that, if you don't put sensitive data like user and passwords there).

Suppose you put your PHPReports classes under */usr/lib/phpreports*, your XML data layout files under */var/xml* and your CSS files under *http://www.yoursite.com/css/*:

```
<?php
    // the line below is only needed if the include_path is not set on php.ini
    ini_set("include_path",ini_get("include_path").":/usr/lib/phpreports/");
    include_once "PHPReportMaker.php";
    $oRpt = new PHPReportMaker();
    $oRpt->setUser("taq");
    $oRpt->setPassword("*****");
    $oRpt->setXML("/var/xml/sales.xml"); // changed to the XML dir
    $oRpt->run();
?>
```

On your *sales.xml* file:

```
...
<!-- changed to the CSS dir //-->
<CSS MEDIA="screen">http://www.yoursite.com/css/sales.css</CSS>
...
```

So, on the PHP code I told the script to look for *phpreports* directory (with *ini\_set()* or defined in *php.ini*) and include *PHPReportMaker.php*. This should be the only class you need to include.

So I asked to use the *sales.xml* file on the */var/xml* directory, which have a reference inside to *sales.css* on the *http://www.yoursite.com/css* directory. This can keep simple your structure and the way you manage your files.

Keeping this that way will not allow people know that you have PHPReports installed, will not allow them to reach the PHPReports classes from the browser address bar, will not show them your XML data layout files.

Just don't forget to erase temporary files created with some output plugins like bookmarks and page. The default ones are stored on your OS temp dir, not accessible by the webserver on the most part of the cases, so it will not be a problem.

## 5.3 Breaking groups with more than one expression

I told you how to break your groups using `EXPRESSION="your_field_here"`; but what if you want a combination of values to act like a group break expression?

Suppose you want the group breaking everytime a change happens on `CITY` and `ZIP_CODE`. You can use this way:

```
<GROUP NAME="CITY_ZIP" EXPRESSION="CITY, ZIP_CODE">
```

You need to use `,` between your field names there.

To make this works PHP concatenates the two results and use as a unique string to check if there was a change on the break string.

## 5.4 Save your report

To save your report to view it later, you need to use the `saveTo(<path>)` function, before running your report.

Remember that the XML result file can be deleted when you run your report, so we need to save it before that.

**IMPORTANT TIP** To save you report you need to use PHP with zlib compiled (`-with-zlib=<dir>`), to use gzip compression.

**IMPORTANT TIP** All the paths are relative to the PHPReports path, for example, `tmp/test.tgz` stands for `PHPREPORTFILE/tmp/test.tgz`.

It's very simple, you can use this way:

```
<?php
include_once "PHPReportMaker.php";
$oRpt = new PHPReportMaker();
$oRpt->setUser("taq");
$oRpt->setPassword("*****");
$oRpt->setXML("sales.xml");
$oRpt->saveTo("tmp/test.tgz");
$oRpt->run();
?>
```

This will save your report on a file called `test.tgz`, on the `tmp` directory.

On our example the XML file is about 8819 bytes,, using the zlib compression it's just 1197 bytes on the `tgz` file. It's a nice way to store that (86.6% compression).

## 5.5 Restore your report

Restoring your report is easy: all you need to do is create the desired output plugin where you want it will be processed and run it using the `loadFrom(<path>)` function.

**IMPORTANT TIP** All the paths are relative to the PHPReports path, for example, `tmp/test.tgz` stands for `PHPREPORTFILE/tmp/test.tgz`.

For example:

```
<?php
include_once "PHPReportMaker.php";
$oRpt = new PHPReportMaker();
$oOut = $oRpt->createOutputPlugin("default");
$oOut->loadFrom("tmp/test.tgz");
?>
```

If you want a different plugin to render your saved report, just change `default` to the plugin you want.

## 5.6 Some few words about file integrity

When you save your report, it will be saved with the **md5** checksum of the file. When restoring your report, this checksum will be validated again, and if there is some difference between the data it has when was saved and the data you're trying to restore, PHPReports will return with an error and will not show your report.

## 5.7 Exchanging formats

Suppose you have your HTML rendered output but now want to be able to convert it to another supported output format, and wants to make it with this in mind:

1. You'll process your report just once
2. The user will be allowed to click on some links to easily change the output formats

Let's work with some frames here. Create a frameset page file called outindex.html using this:

```
<FRAMESET COLS="100,*">
  <FRAME SRC="outmenu.php" NAME="MENU">
  <FRAME SRC="" NAME="MAIN">
</FRAMESET>
```

Now create the outmenu.php file like this:

```
<?php
include_once "PHPReportMaker.php";
$sXML = tempnam(getPHPReportsTmpPath(),"phprpt");
$sBase= basename($sXML);
$oRpt = new PHPReportMaker();
$oRpt->setUser("taq");
$oRpt->setPassword("*****");
$oRpt->setXML("sales.xml");
$oRpt->setXMLOutputFile($sXML);
$oOut = $oRpt->createOutputPlugin("default");
$oOut->setClean(false);
$oOut->setOutput("/dev/null");
$oRpt->setOutputPlugin($oOut);
$oRpt->run();
?>
<html>
  <head>
    <title>PHPReports exchanging formats</title>
    <link rel="stylesheet" type="text/css" href="css/phpreports.css">
  </head>
  <body>
    <p class="REGULAR" style="margin:15px;">
      <?php
        print "converting <b>$sBase</b> to<br><br>";
        print "<a href='PHPReportConvert.php?output=default&file=$sBase' target='MAIN'>
default html</a><br>";
        print "<a href='PHPReportConvert.php?output=page&file=$sBase' target='MAIN'>
page to page</a><br>";
        print "<a href='PHPReportConvert.php?output=txt&file=$sBase' target='MAIN'>
text file</a><br>";
      ?>
    </p>
  </body>
</html>
```

Need your attention here: see the green lines? Is where I create the temporary XML filename where the report data will be put and where I tell to the report and to the output plugin file to don't erase my XML file. So I'll be allowed to use it on another conversion.

Using those files we'll see two frames, one with the menu with three file formats and another one rendering the format you click on the menu. So, is a good idea put something to erase your temporary XML file after the user makes all the conversion he wants.

How to do that is your choice, put another link there to erase the file, schedule some cron jobs blah blah blah. :-) The result will be like this:

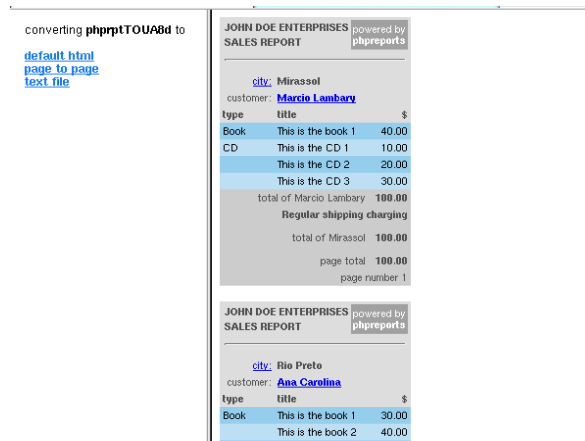


Figure 5.1: Default output

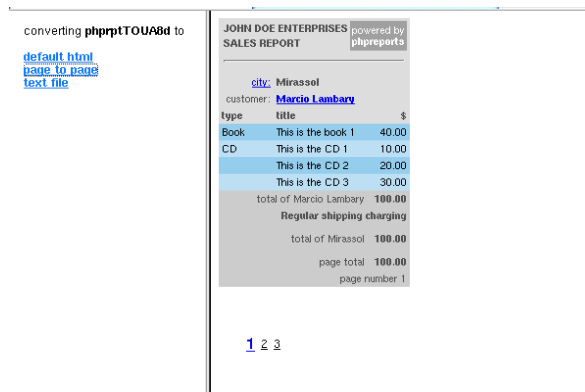


Figure 5.2: Page output

## 5.8 Trying to make things faster

If you have huge reports and some RAM available, you can try this.

I need to tell you how things works here. When you send a ROW with some data to the report it's stored first on a memory variable (fewer disk access means more speed), till a specified limit where you say "hey, no enough memory to handle more", and at this point is created the intermediate XML file and all the memory data is written and flushed there. From this point, all the ROW data that the report receives is automatically written to the file.

On the REPORT element you have the **MAX\_ROW\_BUFFER** attribute, where you can control how many ROWS will be stored on memory before the file write begins.

If you have just a few megabytes, decrease the size (the default value is 2500 rows), if you have a lot of

```

converting phprptTOU88d to
default.html
page to page
text file

JOHN DOE ENTERPRISES
powered by
phpreports
SALES REPORT

city: Mirassol
customer: Marcio Lambary
type title $
Book This is the book 1 40.00
CD This is the CD 1 10.00
This is the CD 2 20.00
This is the CD 3 30.00
total of Marcio Lambary 100.00
Regular shipping charging
total of Mirassol 100.00

page total 100.00
page number 1
-----

JOHN DOE ENTERPRISES
powered by
phpreports
SALES REPORT

city: Rio Preto
customer: Ara Carolina
type title $
Book This is the book 1 30.00
This is the book 2 40.00

```

Figure 5.3: TXT output

RAM increase it.

**Important note:** This kind of thing will only speed up things while creating your report, because on the end there is no escape: all the data on memory is written to the disk. It's up to you make your benchmarks and see what's the best behaviour.

### 5.9 Reuse your XML data

A guy asked me this some days ago and I was checking how to do that. He asked me:

*“I have to make a large number of reports that are identical in structure, number of columns and groups are the same, but the name of the columns and groups change from one form to the next. Is it possible to pass php variables to the xml file so that the value of the variable is used by phpreports instead of the hardcoded field name?”*

I'll answer here as I answered him, “XML should have a way to do that”. So I start to search and found that we can use XML **entities** to make it works that way. On my example I'll deal with HEADERS and FOOTERS. Suppose you always use the same HEADER and the same FOOTER on your reports. So, I have a **header.xml** with this:

```

<HEADER>
  <ROW>
    <COL CELLCLASS="HEADER" COLSPAN="3">
      <XHTML>
        <TABLE BORDER="0" CELLPADDING="2" CELLSPACING="0" WIDTH="100%">
          <TR>
            <TD CLASS="HEADER">
              <b>JOHN DOE ENTERPRISES</b>
            </TD>
            <TD ROWSPAN="2" CLASS="HEADER" style="background:#A0A0A0;color:#FFFFFF;" ALIGN="CENTER">
              powered by<br/><b>phpreports</b>
            </TD>
          </TR>
          <TR>
            <TD CLASS="HEADER">
              <b>SALES REPORT</b>
            </TD>
          </TR>
          <TR>
            <TD COLSPAN="3"><HR/></TD>
          </TR>
        </TABLE>
      </XHTML>
    </COL>
  </ROW>
</HEADER>

```

and a **footer.xml** with this:

```

<FOOTER>
  <ROW>
    <COL ALIGN="RIGHT" CELLCLASS="FOOTER" COLSPAN="2">page total</COL>
    <COL TYPE="EXPRESSION" CELLCLASS="FOOTER" TEXTCLASS="BOLD" ALIGN="RIGHT" NUMBERFORMAT="2">
      $this->getSum("VALUE")</COL>
  </ROW>
  <ROW>
    <COL ALIGN="RIGHT" COLSPAN="3" TYPE="EXPRESSION" CELLCLASS="FOOTER">page number ".$this->getPageNum()</COL>
  </ROW>
</FOOTER>

```

Now inside my XML report layout file I need to change some stuff to tell it to use these files. I know that there is **Xinclude** and we could use it, but the **entities** right now works better, in my opinion. If you have some example you made using XInclude, please send me. But back to the XML file. Suppose it's our old friend **sales.xml**, I must tell it where to find and include the files above. Check this out:

```

<?xml version="1.0" encoding="ISO-8859-1" standalone="no"?>
<!DOCTYPE REPORT SYSTEM "PHPReport.dtd" [
  <!ENTITY header SYSTEM "header.xml">
  <!ENTITY footer SYSTEM "footer.xml">
]>

```

I told there are some files called **header.xml** and **footer.xml** and we'll call them as **header** and **footer** entities now. To use them inside the XML file is very easy:

```

<PAGE BORDER="0" SIZE="30" CELLSPACING="0" CELLPADDING="3">
  &header;
  &footer;
</PAGE>

```

Now all the elements from **header.xml** and **footer.xml** are inside the PAGE element and you can reuse it on the way you want. :-)

## 5.10 Previewing your report

If you want to take a look on how your report will look like (this only works for HTML output) you can use:

```

<?php
  include_once "PHPReportMaker.php";
  $oRpt = new PHPReportMaker();
  $oRpt->preview("sales.xml");
?>

```

For **sales.xml** it will look like:

## 5.11 Passing string parameters to your report

If you want to pass parameters from your PHP code to your reports, you'll need to use the **setParameter()** function.

Suppose you have **init\_date** and **end\_date**, and need to put it on your HEADER element. You will need something like this:

**PHP code**

```

[link] city: [bookmark] $this->getValue("CITY")
customer: [link] $this->getValue("NAME")

```

type	title	\$
TYPE	ITEM	VALUE
	"total of ".\$this->getValue("NAME")	\$this->getSum("VALUE")
	(\$this->getSum("VALUE")>200?"**** Free shipping bonus!****";"Regular shipping charging")	
	"total of ".\$this->getValue("CITY")	\$this->getSum("VALUE")
	page total	\$this->getSum("VALUE")
	page number ".\$this->getPageNum()	
<b>GRAND TOTAL</b>	<b>\$this-&gt;getSum("VALUE")</b>	

Figure 5.4: HTML report preview

```

$aParams = Array();
$aParams["init_date"];
$aParams["end_date"];

// code to create the PHPReportMaker object
$oRpt->setParameters($aParams);
// code to run PHPReportMaker object

```

#### XML code

```

<COL TYPE="EXPRESSION">$this->getParameter("init_date")</COL>
<COL TYPE="EXPRESSION">$this->getParameter("end_date")</COL>

```

Since the 0.3.1 version, you can send 10 parameters to your report.

## 5.12 Passing objects parameters to your report

To pass parameters that are PHP objects (classes), you need to use these functions:

**setEnvObj(id,object)** Stores the *object* with the *id* for reference later.  
**getEnvObj(id)** Get the object reference by *id*.

You can use like this:

#### PHP code

```

// define your object here
// let's suppose your MyClass class have a method
// called printHello(), that prints 'Hello, World!', ok?
$MyClass = new MyClass();

// code to create the PHPReportMaker object
$oRpt->setEnvObj("myclass",$MyClass);
// code to run PHPReportMaker object

```

#### XML code

```

<COL TYPE="EXPRESSION">$this->getEnvObj("myclass")->printHello()</COL>

```

Note that the code above will only works on PHP5. PHP4 can not reference a method on an object returned by a function. Check more about "Dereferencing objects returned from functions" on:



<http://www.php.net/zend-engine-2.php>

The above functions will put your classes/objects inside the PHPReportMaker, and it will make them available to the COL elements there.

## 5.13 Inserting XHTML or PHP code into your COL

You can insert XHTML or PHP code into your XML report layout file, using an external PHP function to do that. The function needs to be available on your report scope, for the file where you create your PHPReportMaker object.

Let's take a look on this simple XML file:

```
<?xml version="1.0" encoding="ISO-8859-1" standalone="no"?>
<REPORT MARGINWIDTH="5" MARGINHEIGHT="5">
<TITLE>Sales Report</TITLE>
<BACKGROUND_COLOR>#FFFFFF</BACKGROUND_COLOR>
  <CSS>sales.css</CSS>
  <SQL>select distinct NAME from sales order by NAME</SQL>
  <CONNECTION>localhost</CONNECTION>
  <INTERFACE>oracle</INTERFACE>
  <NO_DATA_MSG>No data was found, check your query</NO_DATA_MSG>
  <PAGE BORDER="0" SIZE="30" CELLSPACING="0" CELLPADDING="3">
    <HEADER>
      <ROW>
        <COL CELLCLASS="HEADER" COLSPAN="3">
          <XHTML>
            <TABLE BORDER="0" CELLPADDING="2" CELLSPACING="0" WIDTH="100%">
              <TR>
                <TD CLASS="HEADER">
                  <b>JOHN DOE ENTERPRISES</b>
                </TD>
                <TD ROWSPAN="2" CLASS="HEADER" style="background:#A0A0A0;color:#FFFFFF;" ALIGN="CENTER">
                  powered by<br/><b>phpreports</b>
                </TD>
              </TR>
              <TR>
                <TD CLASS="HEADER">
                  <b>SALES REPORT</b>
                </TD>
              </TR>
              <TR>
                <TD COLSPAN="3"><hr/></TD>
              </TR>
            </TABLE>
          </XHTML>
        </COL>
      </ROW>
    </HEADER>
    <FOOTER>
      <ROW>
        <COL ALIGN="RIGHT" COLSPAN="3" TYPE="EXPRESSION" CELLCLASS="FOOTER">"page number ".$this->getPageNum()</COL>
      </ROW>
      <ROW>
        <COL ALIGN="RIGHT" COLSPAN="3" TYPE="EXPRESSION" CELLCLASS="FOOTER">insert_date()</COL>
      </ROW>
    </FOOTER>
  </PAGE>
  <GROUPS>
    <GROUP NAME="customer">
      <FIELDS>
        <ROW>
          <COL CELLCLASS="ODD" ALIGN="LEFT" TYPE="EXPRESSION">$this->getValue("NAME")</COL>
          <COL CELLCLASS="ODD" ALIGN="LEFT" TYPE="RAW_EXPRESSION">show_products($this->getValue("NAME"))</COL>
        </ROW>
      </FIELDS>
    </GROUP>
  </GROUPS>
</REPORT>
```

Just let me say that I'm using the Oracle interface to test it, and let's see the PHP file that calls the XML:

```
<?php
require_once("PHPReportMaker.php");

function show_products($sCustomer) {
    // deal with your database here
    $oCon = @ociLogon("taq","*****","localhost") or die("could not connect");
    $oStmt= @ociParse($oCon,"select TYPE,ITEM from sales where NAME='$sCustomer' order by TYPE,ITEM")
    @ociExecute($oStmt) or die("could not execute");
    $sReturn = "<XHTML><SELECT>";
    while(@ociFetch($oStmt))
        $sReturn .= "<OPTION>".ociResult($oStmt,"TYPE")."-".ociResult($oStmt,"ITEM")."</OPTION>";
    $sReturn .= "</SELECT></XHTML>";
    @ociFreeStatement($oStmt);
    return $sReturn;
}

function insert_date(){
    return "today is ".date('d/m/Y');
}

$oRpt = new PHPReportMaker();
$oRpt->setUser("taq");
$oRpt->setPassword("*****");
$oRpt->setXML("sales.xml");
$oOut = $oRpt->createOutputPlugin("default");
$oRpt->setOutputPlugin($oOut);
$oRpt->run();
?>
```

As you can see, inside the FIELDS element there is a COL with the TYPE="EXPRESSION", calling the **show\_products** function. The show\_products function get as parameter the name of the customer, from the SQL query, and creates a SELECT HTML element **inside** (mandatory!) a XHTML element, and fills the SELECT with all the products bought, executing another query on the database.

Of course this is just a sample, there is better ways to do that, use a open connection to query bla bla but let do this thing this way just for the sample, right?

When running this sample you can see this on your browser:

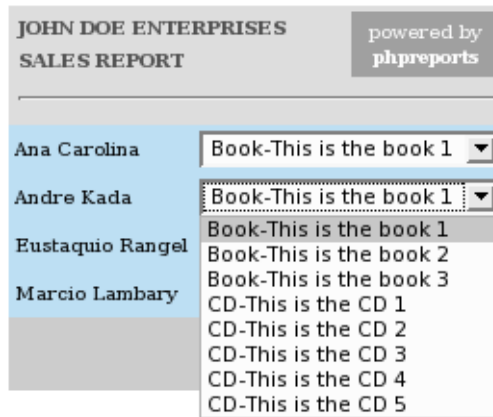


Figure 5.5: XHTML code inserted

Cool uhn? But do not forget, you REALLY need to put this kind of thing inside a XHTML element. But what is that RAW\_EXPRESSION TYPE there? I will explain. When you use EXPRESSION, if the value of the expression have some special chars like `;`, `;` and `&`, this will break your XML file with the report data inside. So before storing the value, the `htmlspecialchars()` is called to convert all this kind of chars in HTML entities. But if insert XHTML code this way, it really needs to keep all those chars as they are. So using RAW\_EXPRESSION is avoiding the use of the `htmlspecialchars` function.

Now, suppose you have a complex PHP code that needs to be called as you process your report. Inserting the PHP code there will be a little weird because the default output, don't forget, is HTML, and if you put PHP code there your web server will not know what to do with that.

The best way is the same approach as above, make a some function with your PHP code, or calling it, available on the scope of the file and call it. Just take a look on the FOOTER element, it has a COL element with the TYPE="EXPRESSION" (not need to take care of HTML special chars at least here on this sample) calling the `insert_date()` function. You can pass parameters and make the most complex things there, just remember that what it returns needs to fit on the well-formed way a XML file is done.

The simple result of this function will be

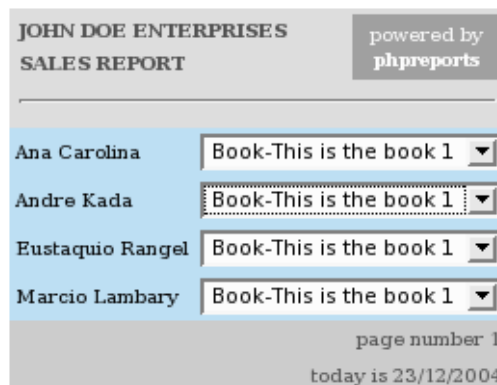


Figure 5.6: PHP function called

## 5.14 Subreports

Subreports are already there. Check the section where I told you about LINKS.

I created there two links, one for the city name and one for the customer name. On the customer name, I used as example a file named *getcustomerinfo.php* and as a parameter I send the customer name, returned using the current value from the SQL query, and it opens on another window. Here is your subreport!

You can fire new reports from there, on this example, you can open another PHP file with another PHPReport with all the info you want from that customer.

Please pay attention that if you want to reflect *exactly* data that can change *while you're creating your report*, maybe it's a good idea on the first report create a table with the values you have on that specific moment. Let's suppose you have a summarize report where you have the customer name, and the ammount the customer bought from you till the *present moment*. Let's say I bought from you some products, and it's value sum is \$100. You can create a link to another report there that "opens" all the products description that compose that \$100. But what happens if, after you create the first report that show you the \$100 ammount, I buy some more stuff, let me say, more \$50? Your first report will show you \$100, then the subreport will show you \$150. That's no good.

The first thing that will happen is you write to me and say "this thing does not work" ehehe. :-)

So, to make this kind of report (where the subreports values can change after you create the "root report"), create a temporary query with the subreports data, then create the first report based on the temporary table, and make the subreports query this table also. So, the report and the subreports will reflect the same values.

Suppose I bought from you:

- Slayer - DVD War at the Warfield - \$15
- Slayer - DVD Still Reigning - \$15

So I have a \$30 total, and if I made a master report for this data, using for example

```
select customer_name, sum(value) from orders order by customer_name
```

it will show you the 30 bucks.

But if I buy "Primus - DVD Hallucino-Genetics Live 2004 - \$20" five minutes after you create your master report, the subreports that detail for example

```
select customer_name, product, value from orders
```

will show you that latest bought DVD also and tell you that my ammount is \$50.

To avoid that, create a temporary table like

```
create table orders_putatimestamphere as select customer_name, product, value from orders
```

and make your master report query it

```
select customer_name, sum(value) from orders_putatimestamphere order by customer_name
```

and also your subreports

```
select customer_name, product, value from orders_putatimestamphere
```

After you ensure that don't need the table for the master and subreports, you can safely drop it.

# Chapter 6

## API

### 6.1 Functions

#### 6.1.1 Functions you can use inside the COL element

The functions on this section you can use inside the COL elements in your XML report layout file. You can reach them from your PHP code also, but they just will be useful when the report is running.

##### **getValue(<field name>)**

Returns the current value of the FIELD on the current GROUP. It's the COLUMN value of the current ROW being processed.

##### **getSum(<field name>)**

Returns the sum of the FIELD on the current GROUP.

##### **getMax(<field name>)**

Returns the maximum value of the FIELD on the current GROUP.

##### **getMin(<field name>)**

Returns the minimum value of the FIELD on the current GROUP.

##### **getAvg(<field name>)**

Returns the average of the FIELD on the current GROUP. It's the max/the row count.

##### **getRowCount()**

Returns the current row count of the GROUP.

##### **getRowNum()**

Returns the current row number.

##### **getPageNum()**

Returns the current page number.

##### **getParameter(<name>)**

Returns the value of the parameter with the specified name.

## 6.1.2 Functions you can use in your PHPReportMaker object

The functions on this section can be used on your PHPReportMaker object. For example,

```
$oRpt = new PHPReportMaker();  
$oRpt->function_goes_here();
```

### **run()**

This function must be called after all the others you may call here. It will run the PHPReports engine and make things work.

### **setXML(<file>)**

Sets the file path where your XML report layout file is. You need to tell the PHPReportMaker object what file to use.

### **getXML()**

Returns the current file name informed on setXML(). See above.

### **setXSLT(<file>)**

Sets the file path of the XSL file used to transform the XML report layout file to the PHP code. You don't need to deal with this unless you want to test/debug PHPReports with a self-made custom XSL file.

### **getXSLT()**

Returns the current file name informed on setXSLT(). See above.

### **setUser(<user>)**

Sets the user name used to open the database connection. You can store it also on the XML file on the USER element, but if your XML file can be read people will know usernames from your network. This is not a good idea, so use the setUser() function. This function overwrite the contents of a USER element, if there is one.

### **getUser()**

Returns the current user name informed on setUser(). See above.

### **setPassword(<password>)**

Sets the password used to open the database connection. You can store it also on the XML file on the PASSWORD element, but if your XML file can be read people will know a database password. This is a **VERY BAD IDEA**, so use the setPassword() function. This function overwrite the contents of a PASSWORD element on the XML file, if exists.

### **getPassword()**

Returns the current password informed on setUser(). See above.

### **setConnection(<connection>)**

Set the database connection string. For Oracle, for example, is the TNS entry name. This function overwrite the contents of a CONNECTION element on the XML file, if exists.

### **getConnection()**

Returns the current connection string informed on `setConnection()`. See above.

### **setDatabaseInterface(<interface>)**

Sets the name of the database you're connecting. You **MUST** inform this, because if you did not, PHPReports will not know on what type of database it will connect and will fail to load it's interface. You can check the available interfaces on the **database** directory under the PHPReports directory. Some interfaces are adodb, informix, interbase (firebird), mssql (Microsoft SQL Server), mysql, odbc, oracle, peardb, postgresql. This function overwrite the contents of a `INTERFACE` element on the XML file, if exists.

### **getDatabaseInterface()**

Returns the current database interface string informed on `setDatabaseInterface()`. See above.

### **setDatabase(<database>)**

Sets the name of the database that will be selected after the connection is open. This function overwrite the contents of a `DATABASE` element on the XML file, if exists.

### **getDatabase()**

Returns the current database informed on `setDatabase()`. See above.

### **setSQL(<sql>)**

Sets the SQL query string. Using this function allows you to create the query the way you want before sending it for processing. If you store your query in the `SQL` element on the XML file, it **will always be static**. But if you use this function, you can manipulate it the way you want. This function overwrite the contents of a `SQL` element on the XML file, if exists.

### **getSQL()**

Returns the current SQL query string informed on `setSQL()`. See above.

### **setParameters(<array>)**

Sets the array parameters. You can reference to it from your XML file using the `getParameter(<name>)` function. It's a good idea using it by name reference and not numeric reference, I mean, "myparameter" and not 0 is more clear.

### **getParameters()**

Returns parameters array informed on `setSQL()`. See above.

### **setCodeOutput(<file>)**

Sets a file (need write permission) where the PHP generated code will be written. If null, the code will remains on the memory. This function is useful for debugging purposes, when you need to know the final PHP code that will be used to create the XML file with your data.

### **getCodeOutput()**

Returns the file name informed on `setCodeOutput()`. See above.

### **setOutput(<file>)**

Sets a file (need write permission) where the final result of the report processing will be stored. If you're using for example the default plugin, will be a HTML file with all your report HTML code. Useful if you don't want to show the report on the screen and provide a link to it.

### **getOutput()**

Returns the file name informed on setOutput(). See above.

### **setXMLOutputFile(<file>)**

Sets a file (need write permission) where your data will be stored. It's a XML file that will be used by the output plugin (see below) to render your report. Useful when you need to see how your data is being stored.

### **getXMLOutputFile()**

Returns the file name informed on setXMLOutputFile(). See above.

### **setNoDataMsg(<message>)**

Sets the message that will be displayed when no data is found after running your SQL query. Defaults to "NO DATA FOUND" message.

### **getNoDataMsg()**

Returns the message informed on setNoDataMsg(). See above.

### **createOutputPlugin(<plugin>)**

Create an output plugin object. The output plugins are under the **output** directory (just **common** is not an output plugin there).

### **setOutputPlugin(<plugin>)**

Sets the output plugin that will be used to render your XML data file. Created with createOutputPlugin() above. If no output plugin is set, PHPReports will use the **default** one.

### **getNoDataMsg()**

Returns the name of the output plugin informed on setOutputPlugin(). See above.

### **setEnvObj(<id>,<object>)**

Put an object/class inside the environment array to be returned on the COL element using getEnvObj().

### **getEnvObj(<id>)**

Get an object/class from the the environment array.

### **saveTo(<file>)**

Sets a file (needs write permission) where your report will be saved, for later retrieval with the output plugin loadFrom() function.

### **save()**

Saves your report on the file informed on the saveTo() function (see above). You'll need PHP compiled with **zlib** and **md5** support.



**preview(<XML>)**

Preview your XML layout report, with no data yet.

## 6.2 XML

### 6.2.1 REPORT

Refers to the main report object, the one who contain the others.

#### Elements

<b>TITLE</b>	The report title.
<b>PATH</b>	The path where the PHPReports classes are.
<b>BACKGROUND_COLOR</b>	On the hex format.
<b>BACKGROUND_IMAGE</b>	A valid URL to it.
<b>CSS</b>	A valid URL to the CSS file.
<b>SQL</b>	Your SQL query to ask for it on the database.
<b>USER</b>	The user id to open the database connection.
<b>PASSWORD</b>	The password to open the database connection.
<b>CONNECTION</b>	The database connection name to use (ex: the Oracle TNS entry name).
<b>INTERFACE</b>	The database interface to use.
<b>NO_DATA_MSG</b>	Customized message when no data is found.
<b>TEMP</b>	Not used now - only here to valid the old DTDs files.
<b>DEBUG</b>	Used to show debugging messages through the report.
<b>FORM</b>	Used to create a HTML FORM with your report.
<b>DOCUMENT</b>	The DOCUMENT element.
<b>PAGE</b>	The PAGE element.
<b>GROUPS</b>	The GROUPS element.

#### Attributes

<b>MARGINWIDTH</b>	The margin width, in pixels.
<b>MARGINHEIGHT</b>	The margin height, in pixels.
<b>MAX_ROW_BUFFER</b>	Maximum number of rows to store on the memory before write it to a file. The default value is 2500 rows.

### 6.2.2 CSS

The CSS files you can use in your report.

#### Attributes

**MEDIA** The type of media to use this CSS file. You can specify, for example, "screen" or "print".

### 6.2.3 FORM

Refers to an element used to create HTML FORMs with your report.

#### Elements

<b>FORM_NAME</b>	The form name.
<b>FORM_METHOD</b>	The form method.
<b>FORM_ACTION</b>	The form action.

### 6.2.4 DOCUMENT

The element where the values of all the report are stored, I mean, it's the global values handler.

#### Elements

<b>HEADER</b>	The header element.
<b>FOOTER</b>	The footer element.

## 6.2.5 HEADER

The element with the ROWs used on the top of PAGES/GROUPs.

### Elements

**ROW** One or more ROW elements.

## 6.2.6 FOOTER

The element with the ROWs used on the bottom of PAGES/GROUPs.

### Elements

**ROW** One or more ROW elements.

## 6.2.7 ROW

The element with the COLs.

### Elements

**COL** The COL element.

## 6.2.8 COL

The element with the FIELDS and expression values.

### Elements

**LINK** The LINK element.  
**BOOKMARK** The BOOKMARK element.  
**XHTML** The XHTML element.

### Attributes

<b>TYPE</b>	Can be REGULAR, EXPRESSION, RAW_EXPRESSION or FIELD.
<b>NUMBERFORMAT</b>	It uses the sprintf (you know,C,PHP...) to format numeric values.
<b>NUMBERFORMATEX</b>	Format values with your current thousand and decimal separators, specify the decimal places.
<b>CELLCLASS</b>	CSS class to format COL (cell).
<b>CELLCLASSEVEN</b>	CSS class to format even COLs (cell).
<b>CELLCLASSODD</b>	CSS class to format odd COLs (cell).
<b>CELLCLASSEXPRESSION</b>	Expression to calculate the CSS class to format COL (cell).
<b>TEXTCLASS</b>	CSS class to format the text inside the COL (cell).
<b>ROWSPAN</b>	The number of ROWs to span.
<b>COLSPAN</b>	The number of COLs to span.
<b>WIDTH</b>	The COL width, in pixels.
<b>HEIGHT</b>	The COL height, in pixels.
<b>ALIGN</b>	The COL horizontal alignment (use HTML values).
<b>VALIGN</b>	The COL vertical alignment (use HTML values).
<b>SUPPRESS</b>	If the COL value will be printed or not if it's equal the previous one (boolean).

## 6.2.9 PAGE

The PAGE element, where format the page features and show the values.

## Elements

- HEADER** The HEADER element.  
**FOOTER** The FOOTER element.

## Attributes

- SIZE** The PAGE size, in rows.  
**WIDTH** The PAGE width, in pixels.  
**HEIGHT** The PAGE height, in pixels.  
**CELLPADDING** The padding between the COLs (cells).  
**CELLSPACING** The spacing between the COLs (cells).  
**BORDER** The border size between the COLs, default to 0.  
**ALIGN** PAGE alignment - think about the ALIGN on a HTML table.

### 6.2.10 GROUPS

The element that contain GROUPs elements.

## Elements

- GROUP** The GROUP element.

### 6.2.11 GROUP

This is where you group your data, make breaks etc.

## Elements

- HEADER** The HEADER element.  
**FIELDS** The FIELDS element.  
**FOOTER** The FOOTER element.  
**GROUP** Another GROUP inside.

## Attributes

- NAME** The NAME of the group.  
**EXPRESSION** The expression where the GROUP breaks.  
**PAGEBREAK** If you need a page break after the group breaks, put TRUE here.  
**REPRINT\_HEADER\_ON\_PAGEBREAK** Put TRUE if you need to reprint the header on the page break.  
**RESET\_SUPPRESS\_ON\_PAGEBREAK** If TRUE, reset all values suppressed when the page breaks.

### 6.2.12 FIELDS

This is just a container to ROWs that contain COLs that are of the TYPE="FIELDS".

## Elements

- ROW** Some ROW elements.

### 6.2.13 LINK

An element to make a link to another URL.

## Attributes

- TYPE** STATIC,DYNAMIC or EXPRESSION.  
**TARGET** The target frame where the URL will open (same as HTML target).  
**TITLE** The text that will appears on the tooltip (on mouse over).

### 6.2.14 BOOKMARK

An element to make a bookmark on the current COL.

#### Attributes

**TYPE**           STATIC,DYNAMIC or EXPRESSION.  
**CELLCLASS**    CSS class to format COL (cell).  
**TEXTCLASS**    CSS class to format the text inside the COL (cell).

### 6.2.15 IMG

An element to insert an image file inside a COL. Put the image URL between <IMG> and </IMG>.

#### Attributes

**WIDTH**        Image width (pixels).  
**HEIGHT**       Image height (pixels).  
**BORDER**       Image border (pixels).  
**ALT**           Alternate text.

### 6.2.16 XHTML

All kind of XHTML elements.



# Chapter 7

## FAQ

### 1. What is PHPReports?

It's a program designed to make easier you create reports using XML template files running on a web server environment.

### 2. What do I need to make it works?

You need an Apache web server (<http://www.apache.org>), PHP running on Apache (<http://www.php.net>) and the Sablotron lib running on PHP4 ([http://www.gingerall.com/charlie/ga/xml/p\\_sab.xml](http://www.gingerall.com/charlie/ga/xml/p_sab.xml)) or the XSL extension compiled with PHP5 (<http://www.php.net/manual/en/ref.xsl.php>).

### 3. Ok, but how can I install that? I already have Apache and PHP, but tell me about Sablotron.

Please check this link ([http://www.gingerall.com/charlie/ga/xml/x\\_sabphp.xml](http://www.gingerall.com/charlie/ga/xml/x_sabphp.xml)) for more info about how you can do that.

There's also information about how to do that running on Windows.

### 4. My thirdy party web server doesn't have Sablotron installed and I think it won't install it. What can I do?

Tell him about PHPReports and about XML and XSLT. Maybe he likes the idea and install Sablotron. Or if he/she uses PHP5, just ask to compile it with XSL support.

If he doesn't, so sorry, you can't use PHPReports there. :-)

### 5. I'm happy you mention Windows. Could you help me with more info on how to install it there?

No, sorry, I don't use Windows to run that. I really can't help you with that, but there is some guys on the [phpreports-users@lists.sourceforge.net](mailto:phpreports-users@lists.sourceforge.net) that can. :-)

### 6. Can you help me to install Apache, PHP and Sablotron/XSL?

No, sorry. Every tool have some good tutorials on how to do it. Please check their websites.

### 7. I got a message unknown encoding 'ISO-8859-1', and my report don't works.

The ISO-8859-1 is the encoding I use for Latin characters. Maybe you don't need it or your system don't allow it (Windows systems, most cases). It's safe to remove the encoding parameter if you don't need chars like á,é,í,ó,ú, etc.

### 8. All my HTML escaped code is not working. I put something like &lt;HR/&gt; there but I can't see it on the browser.

Please put it inside a XHTML element now. To make all the elements go on a nice way to the intermediate XML file, it will need it, because if you use escaped code it will be interpreted on the first XLST transformation and will not match any of the pre-defined elements on the final code.

### 9. Is there a way to change the color of a COL (cell) based on its value?

Since 0.3.4 there is a way to do that. Use the CELLCLASSEXPRESSON COL parameter to do that. For example, to render COLs with negative values in red and the other ones in black:

```
<COL CELLCLASSEXPRESSON="($this->getValue("VAL")&lt;0?'RED':'BLACK')" TYPE="FIELD">VAL</COL>
```

10. **I'm just seeing a blank page or a timeout message**

If you are creating a huge report or your database is taking much time to execute your query, you can face a PHP timeout. PHP scripts are allowed to run for 30 seconds. If this happens, you need to extend your timeout using the `set_time_limit()` function. More about it here:

<http://www.php.net/manual/en/function.set-time-limit.php>

You need also check the disk space the webserver user is allowed to use. Remember that we have a data file with all your data inside there, and if you have a 10000 rows report, you'll need some space there. Of course when we save it, it is compressed and becomes very shorter.