

# **FastReport.Net User's manual**



# Table of contents

<b>Chapter I</b>	<b>Fundamentals</b>	<b>12</b>
The report		12
Report designer		12
Report options		14
Report pages		16
Managing pages		17
Page properties		18
Bands		21
Bands in designer		21
Configuring bands		22
Printing Bands		24
Band properties		25
Report objects		27
Common object properties		27
The "Text" object		29
Text editing		29
Displaying the expressions		30
Displaying the data columns		31
HTML tags		32
Object's properties		33
The "Rich Text" object		34
The "Picture" object		36
The "Line" object		39
The "Shape" object		39
The "Barcode" object		40
The "CheckBox" object		43
The "Table" object		44
The "Matrix" object		45
The "Chart" object		45
The "Zip Code" object		46
The "Cellular Text" object		47
Your first report in the FastReport		48
Example 1. Creating a report manually		48
Example 2. Creating a report with the wizard		50
<b>Chapter II</b>	<b>Report creation</b>	<b>56</b>
Choosing data for a report		57
Dynamic layout		58
CanGrow, CanShrink properties		58
ShiftMode property		59
GrowToBottom property		59
Anchor property		59
Dock property		60
Formatting		62

Border and fill	62
Text formatting	63
Styles	63
Data formatting	64
Conditional highlighting	66
Hiding zero values	68
Hide duplicate values	69
Highlight odd/even data rows	70
Report with one "Data" band	71
Connecting a band to data source	71
Printing the text	71
Sorting the data	71
Filtering the data	73
Data header and footer	74
Breaking data and keeping it together	75
Printing empty data rows	77
Printing "No data" text	79
Printing hierarchy	80
Master-detail report	83
Master-master report	86
Master-detail-detail report	88
Multicolumn reports	89
Page columns	89
Data band columns	90
"Booklet"-type reports	92
Adding page into a report	92
Page settings	92
Printing on odd/even pages	93
Groups and totals	95
Creating groups	97
Sorting the data	99
Nested groups	100
Managing groups	101
Printing total values	102
Repeating the header and footer	103
Group properties	104
Subreports	106
Printing modes	106
Side-by-side subreports	107
Nested subreports	107
Table-type reports	108
Configuring columns	108
Managing the size of the column	108
Configuring rows	109
Managing the size of the row	109
Configuring cells	110
Joining and splitting cells	110
Inserting objects in cells	110
Printing a table	111
Printing complex headers	113

Using totals	115
Table layout	115
Examples	116
Example 1. Printing the whole table from top to bottom	116
Example 2. Printing the table from top to bottom with a repeating row	116
Example 3. Printing the whole table from left to right	117
Example 4. Printing a table from left to right with a repeating column	118
Example 5. Printing a table with repeating rows and columns	118
Example 6. Using the data source	119
Example 7. Inserting page breaks	120
Example 8. Printing totals	121
Matrix-type reports	122
A few theory	122
Configuring the matrix	123
Configuring headers	125
Configuring cells	125
Styling the matrix	126
Row and column size management	127
Examples	128
Example 1. Simple matrix	129
Example 2. Multilevel headers	130
Example 3. Printing the name of the month	131
Example 4. Conditional highlighting	132
Example 5. Highlighting even rows	134
Example 6. Using Expressions	135
Example 7. Pictures in cells	136
Example 8. Objects in cells	137
Example 9. Filling a matrix manually	139
Interactive reports	143
Hyperlink	143
Hyperlink configuration	143
Link to the URL	144
Link to the page number	145
Link to a bookmark	145
Link to a detailed report	145
Link to a detailed page	146
Custom link	147
Report outline	148
Examples	150
Example 1. Link to a web page	150
Example 2. Building a detailed report	151
Example 3. Interactive "Matrix" object	156
Example 4. Report with table of contents, navigation and outline	160
Report inheritance	162
Creating a report	163
Changing the base report	163
Limitations	164
Reports with charts	165
Chart elements	165
Chart editor	166

Handling series	167
Setting up the appearance	168
Connecting chart to data	170
Sorting the data	171
Grouping the data	172
Collecting the data	173
Exploding the values	174
Setting up auto-series	175
Interactive charts	176
<b>Chapter III Data</b>	<b>180</b>
The "Data" window	180
Data sources	181
Creating a data source	182
Creating a SQL query	185
Query parameters	187
Passing a value to the parameter	188
Editing a connection	189
Editing a data source	189
Aliases	190
Hierarchical data sources	190
Relations	191
Creating a relation	194
Editing a relation	194
System variables	195
Functions	196
Mathematical	196
Abs	196
Acos	197
Asin	197
Atan	197
Ceiling	198
Cos	198
Exp	198
Floor	198
Log	199
Maximum	199
Minimum	199
Round	200
Sin	200
Sqrt	200
Tan	201
Truncate	201
Text	201
Asc	201
Chr	202
Insert	202
Length	202
LowerCase	202
PadLeft	203

PadRight	203
Remove	204
Replace	204
Substring	204
TitleCase	205
Trim	205
UpperCase	205
Date & Time	206
AddDays	206
AddHours	206
AddMinutes	206
AddMonths	206
AddSeconds	207
AddYears	207
DateDiff	207
DateSerial	207
Day	208
DayOfWeek	208
DayOfYear	208
DaysInMonth	208
Hour	209
Minute	209
Month	209
MonthName	209
Second	210
Year	210
Formatting	210
Format	210
FormatCurrency	214
FormatDateTime	215
FormatNumber	216
FormatPercent	216
Conversion	217
ToBoolean	217
ToByte	217
ToChar	217
ToDateTime	217
ToDecimal	218
ToDouble	218
ToInt32	218
ToRoman	218
ToSingle	219
ToString	219
ToWords	219
ToWordsEnGb	220
ToWordsRu	221
Program Flow	222
Choose	222
If	222
Switch	222

Totals	223
Creating a total	223
Conditional totals	225
Running totals	226
Page totals	226
Printing the total in the header	227
Report parameters	228
Creating a parameter	229
Using parameters in a report	229
<b>Chapter IV Expressions</b>	<b>232</b>
Expression editor	233
Reference to report objects	233
Using .Net functions	234
Reference to data elements	235
Reference to data sources	235
Reference to system variables	236
Reference to total values	237
Reference to report parameters	237
<b>Chapter V Script</b>	<b>240</b>
General information	241
Event handlers	242
Report events	242
Using .Net objects	243
Reference to report objects	244
Report and Engine objects	245
Reference to data sources	248
Reference to system variables	249
Reference to total values	249
Reference to report parameters	249
Examples	251
Example 1. Changing object's appearance	251
Example 2. Highlighting even rows of the band	252
Example 3. Data filtering	252
Example 4. Calculating total	253
Example 5. Shifting the print position	254
<b>Chapter VI Dialogue forms</b>	<b>258</b>
Controls	258
Referencing to a control from code	260
Data filtering	260
Automatic filtering - how it works	261
Filter operations	262
Adding filter into a report	263
Filtering on data range	264
Filtering on related data column	264
Filtering using cascading lists	264
Controlling the filtering from code	265
Examples	265

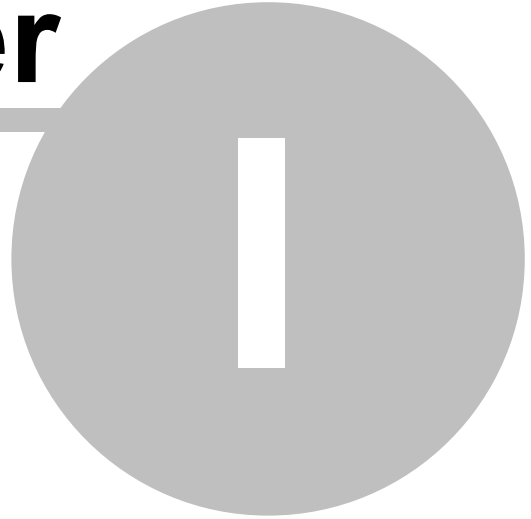


Example 1. "Hello, FastReport!"	265
Example 2. Ask for a text from the user	266
Example 3. Handling dialogue controls	267
Example 4. Handling report objects	268
Example 5. Simple filter	269
Example 6. Automatic filtering	270
Example 7. Automatic filtering by range	271
Example 8. Filtering by related data column	272
<b>Chapter VII Preview, print, export</b>	<b>276</b>
Editing the report	277
Printing the report	278
Exporting the report	280
Saving in FPX format	280
Export to Adobe Acrobat (PDF)	281
Export to Word (RTF)	282
Export to HTML	283
Export to MHT (web archive)	284
Export to Excel (XML)	285
Export to Excel 2007	286
Export to PowerPoint 2007	287
Export to OpenOffice Calc	288
Export to CSV	289
Export to TXT	290
Export to picture	291
Report design recommendations	292
Sending the report by email	293



# Chapter

---



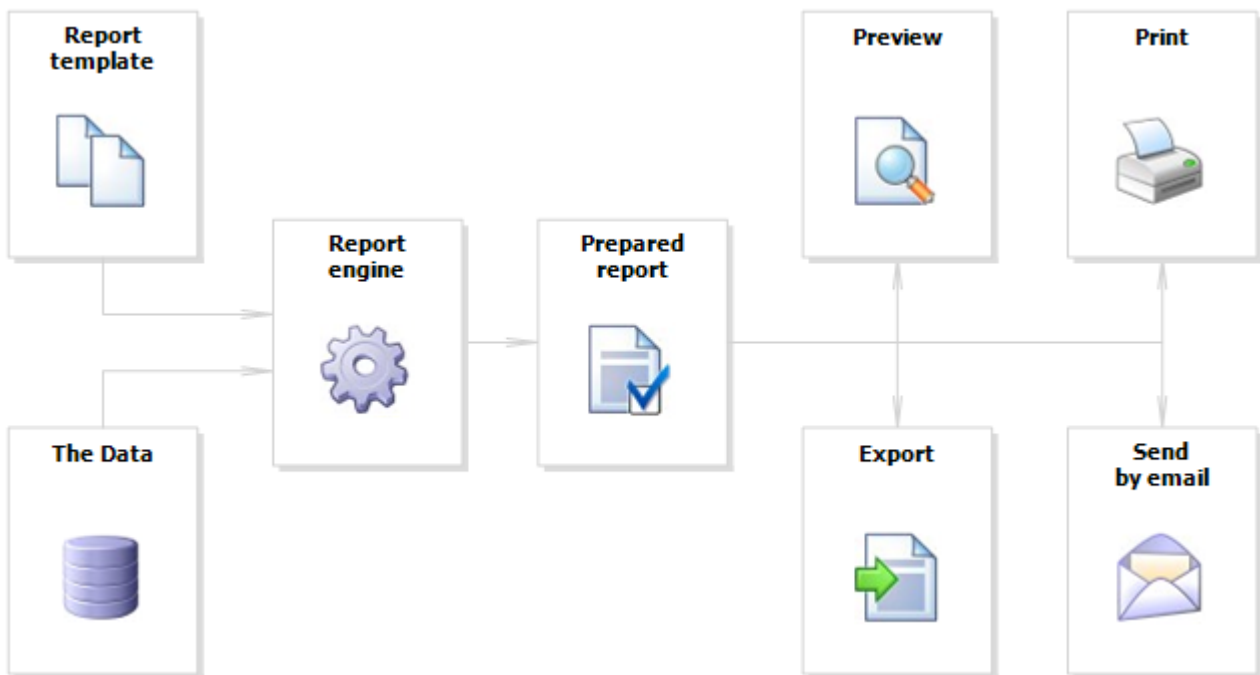
# Fundamentals

# Fundamentals

In this chapter we will learn the principles of working with a report in the FastReport. We will also take a close look at report elements such as report pages, bands, and report objects.

## The report

The report building process can be represented as follows:



Report template (later-Report) - this is, what we see in the designer. Reports are saved in files with an extension .FRX. A Report can be created with the help of designer or programmatically.

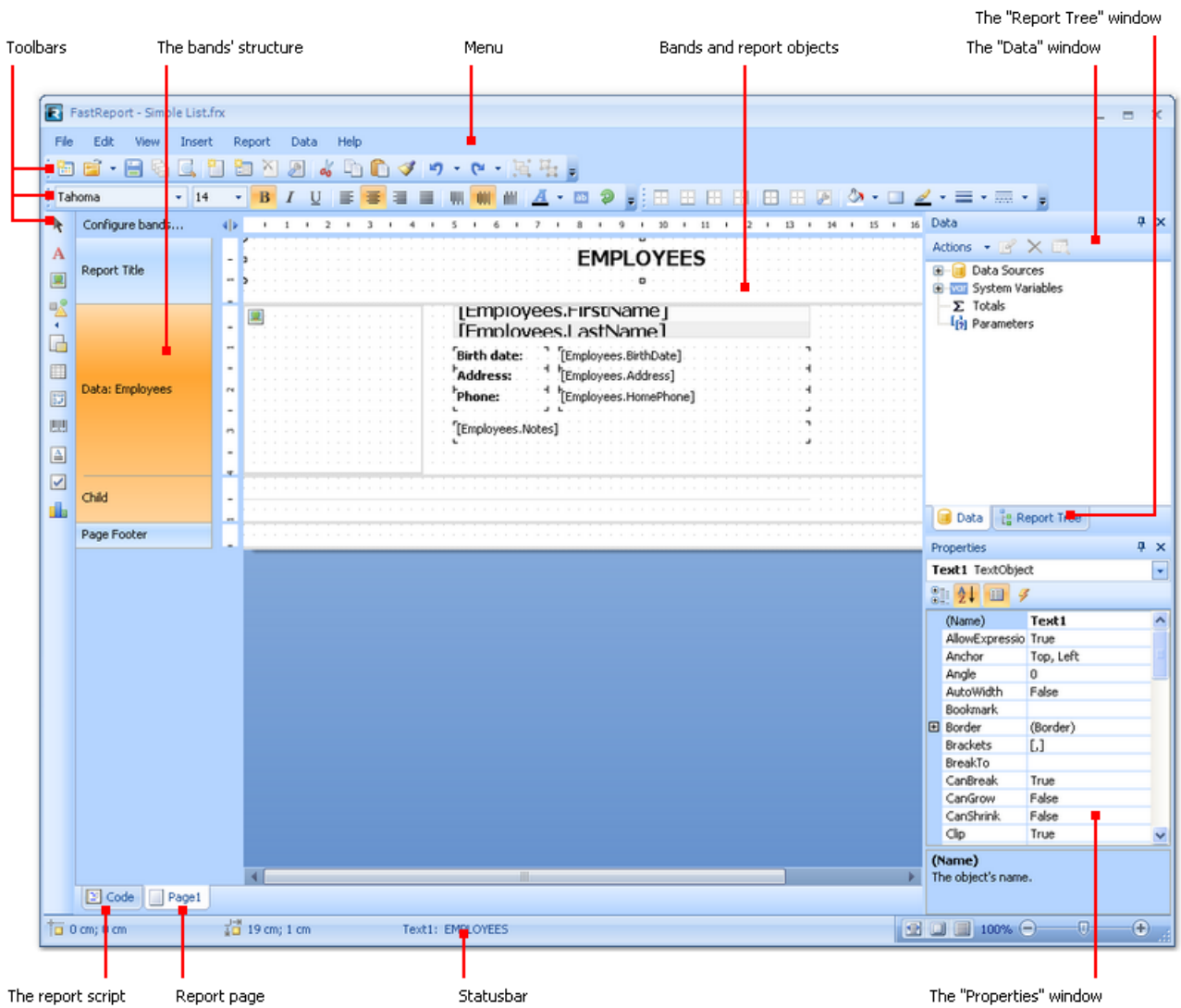
Data can be any: this is data, defined in the program, or data from DBMS, for example, MS SQL. FastReport can also work with business-logic objects (later - business-objects).

Prepared Report - this is what we see in the preview window. Prepared report can be previewed, printed, saved in one of the supported formats (.doc, .xls, .pdf and others), or can be sent by email.

## Report designer

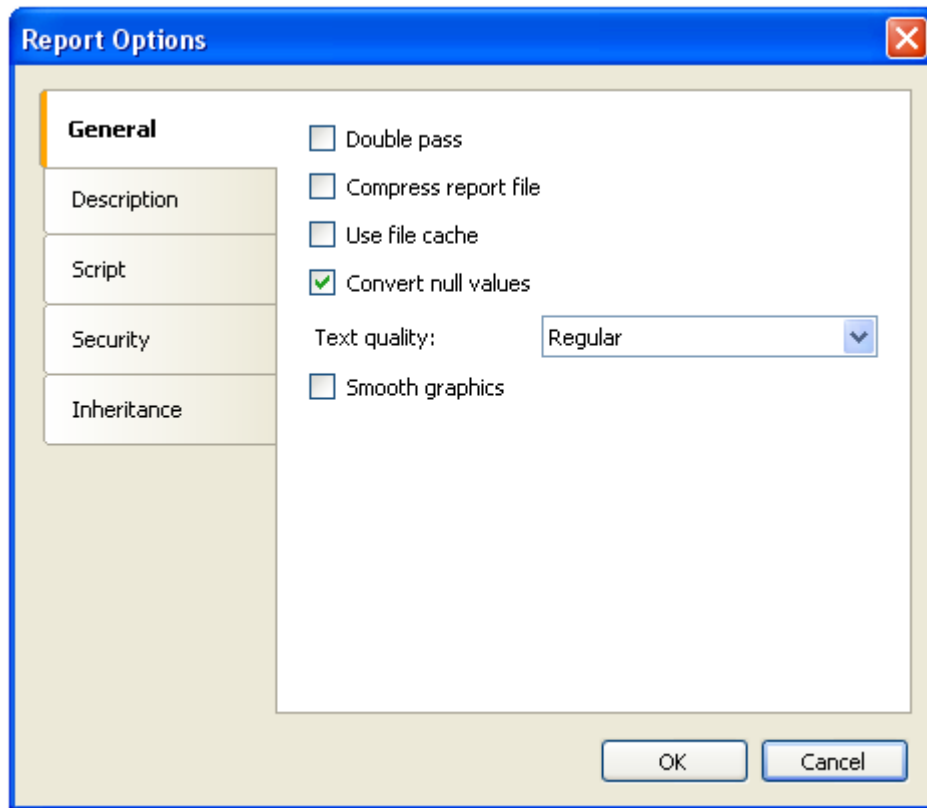
In order to create a report template, a report designer is used. A designer gives the user comfortable facilities for designing the report and allows previewing the report at the same time.

The report designer is the compound part of FastReport and does not depend on the development environment (for example, MS Visual Studio). If you are a software developer, you may include the report designer into your application. This will give your end-users the ability to either change the existing report or create a new one.



## Report options

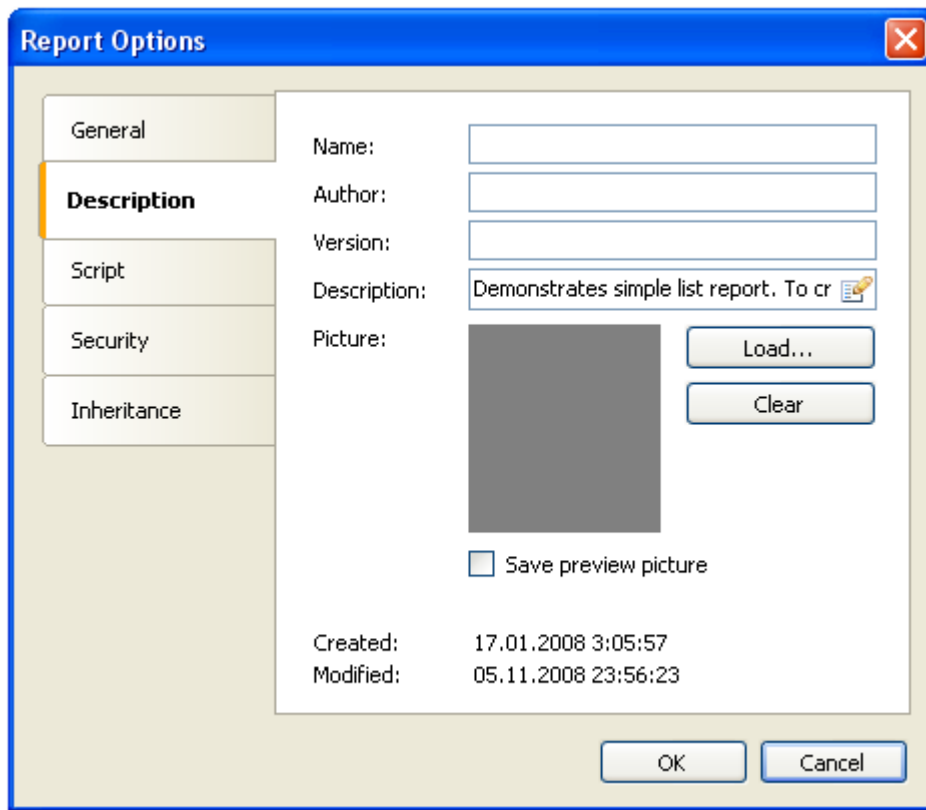
A window with report options can be called in the "Report|Options..." menu. You will see a dialogue window with several tabs:



On the "General" tab, you can control the following report parameters:

- "Double pass" parameter allows to enable two report passes. This can be necessary when you use the "total number of pages" system variable;
- "Compress report file" parameter allows saving a report in a compressed form. For compressing, zip algorithm is used, that is why you can easily extract original contents with the help of any archive;
- "Use file cache" parameter allows to save the memory when creating a report. Use this parameter if your report has got a lot of pages;
- "Convert null values" controls converting the null value data column into the default value (0, empty string, false - depending on the data type of a column);
- "Text quality" parameter allows choosing the mode of text displaying in the report. This mode does not affect printing of the report;
- "Smooth graphics" parameter allows to enable the smooth mode when drawing graphical objects (line, border, picture).

On the "Description" tab, you can give the description of the report. All these parameters are not obligatory, and they serve for informational purposes:



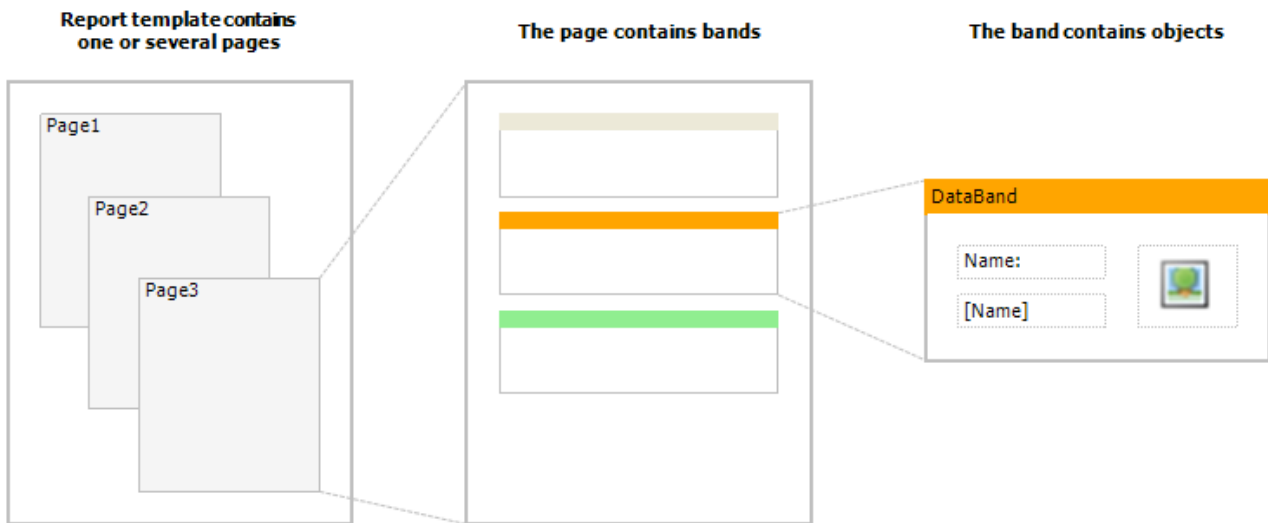
On the "Script" tab, you can choose the script language for the report. Detailed work with script can be found in the ["Script"](#) chapter.

On the "Security" tab you can give the password which will be requested when opening the report. A report which has a password, is saved in an encoded form, so do not forget your password! Restoring a report in this case will be practically impossible.

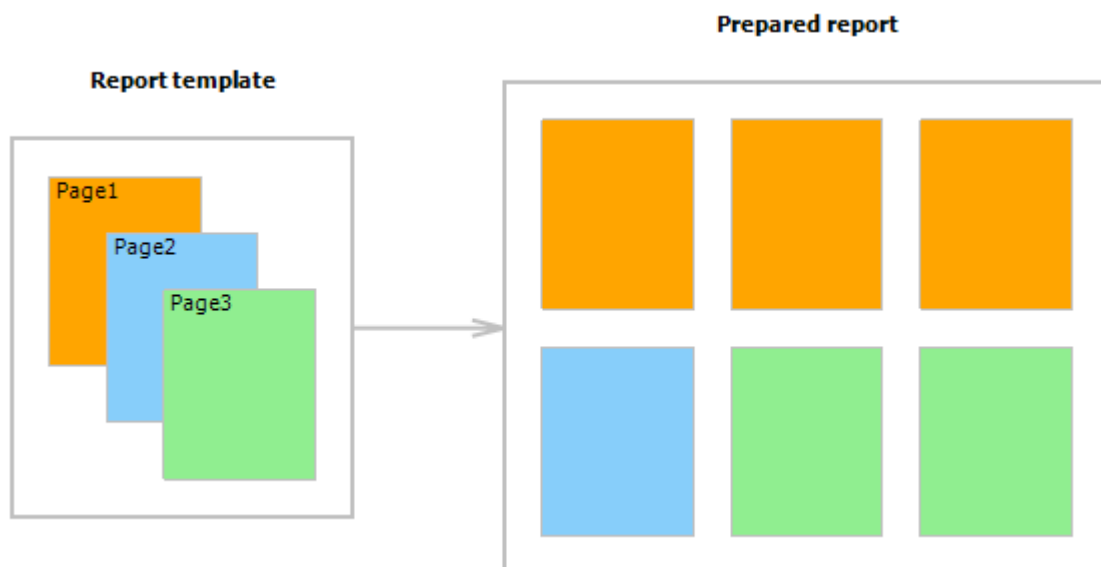
On the "Inheritance" tab, you can control report inheritance. This functionality will be looked at later.

## Report pages

Template consists of one (mostly) or several report pages. Report page, in turn, contains bands. Report objects like Text, Picture and others are placed on the band:



Report template can consist of several pages. For example, you can create a template containing title-page and a page with data. When creating such a report, the first page will be printed first, then the second page and so on. Every page of template can generate one or several pages of a prepared report – this depends on the data it contains:





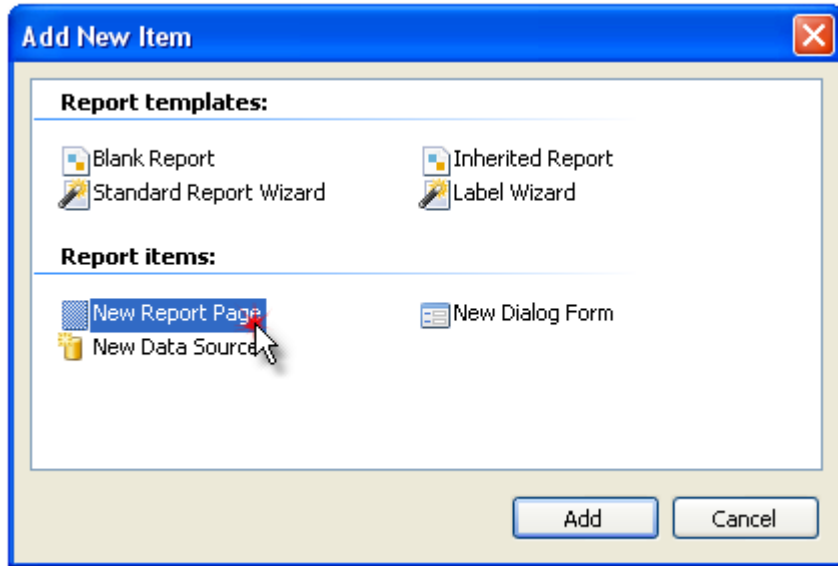
Report pages are also used when working with subreports. Contrary to other report generators, subreports in FastReport are saved in a separate template page, and not in a separate file.


Apart from report pages, a template can contain one or more dialogue forms. Dialogue forms can be used for inquiring some parameters before creating a report. Detailed work with dialogue forms will be covered in the ["Dialogue forms"](#) chapter.



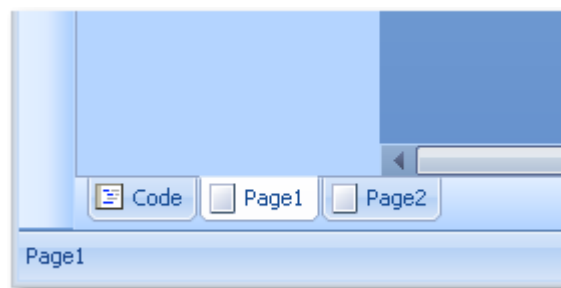
## Managing pages

When you have created a new report, it already contains one page with several bands. For adding a new page, click the  button. A page can also be added by clicking the  button and choosing "New Report Page" in the window.




In a similar way, dialogue forms can be added into the report. For this, use the  button.

Template pages are displayed in the designer as tabs:




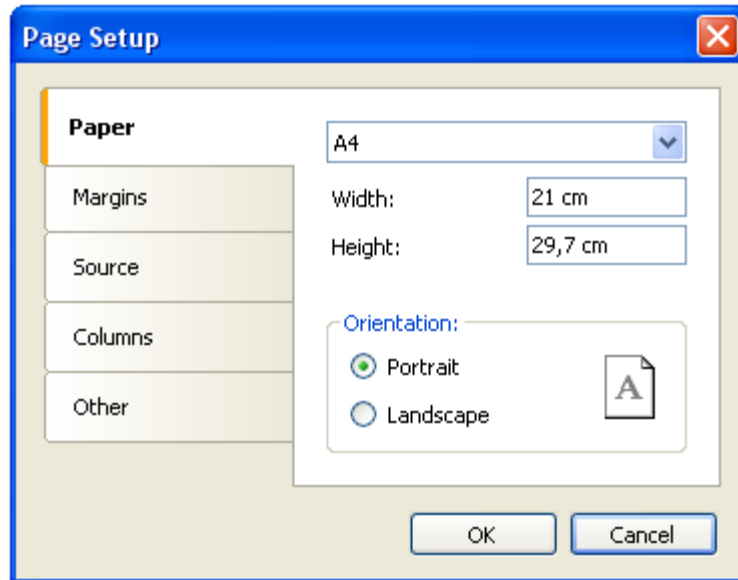
The first tab is the report code. It can neither be moved nor deleted.

In order to switch to the needed page, simply click on its tab. Changing order of the pages can be done with the help of the mouse. For this, left click on the tab and, without leaving the mouse, move the tab to the desired place.

For deleting a page, click the  button. This button is not active if the report consists of only one page.

## Page properties

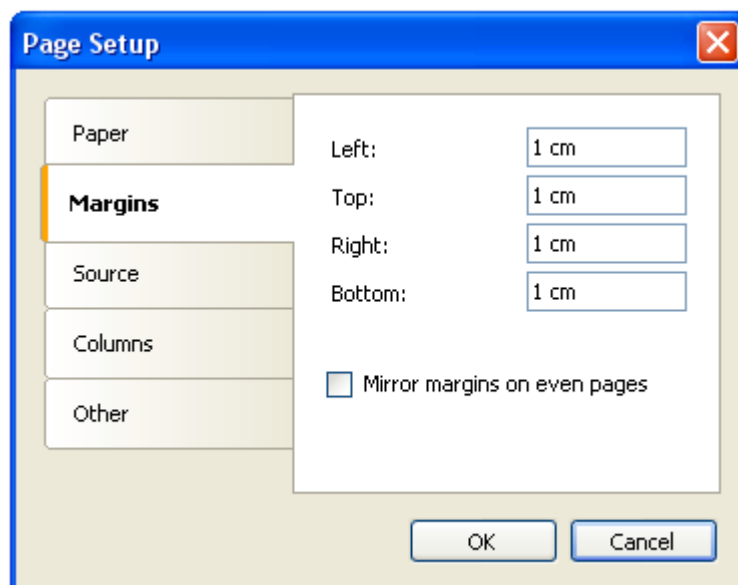
Every report page can have its own settings, such as paper size, orientation (landscape or portrait), margins, columns, paper source and others. Report template can contain several pages with different orientations and paper sizes. The window with page setup can be called by clicking the  button or by choosing the "File|Page setup..." menu item.



The "Paper" group allows to set the paper size and orientation. It is possible to choose one of the supported sizes, by using the drop-down list. It contains all paper sizes which are supported by the current printer.

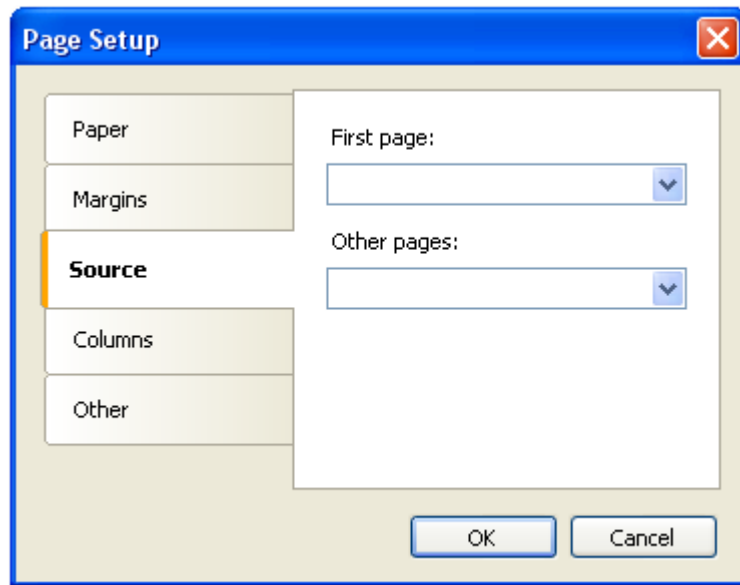
Current printer can be configured by using "File|Printer Setup..." menu.

The "Margins" group allows to setup page margins. The "Mirror margins on even pages" options can be used to print booklets:



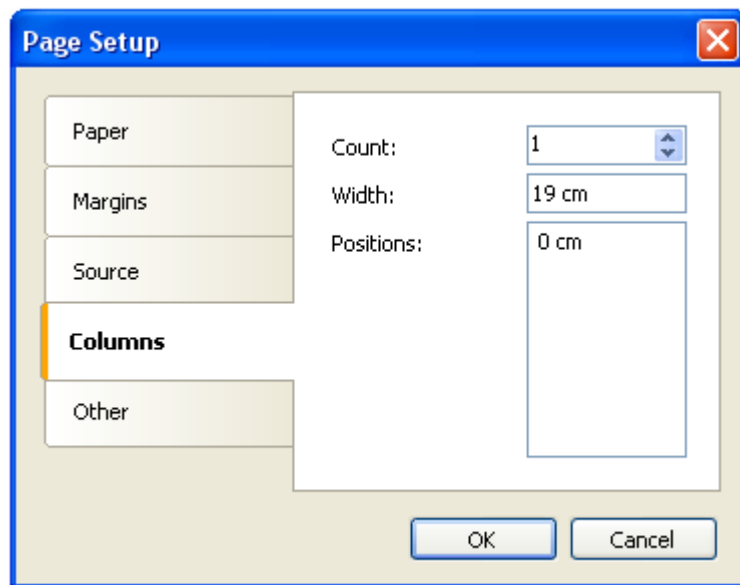
The "Source" group allows choosing the source of the paper. Note that the sources can be

given separately, that of the first page of the prepared report, and that of the rest of pages:

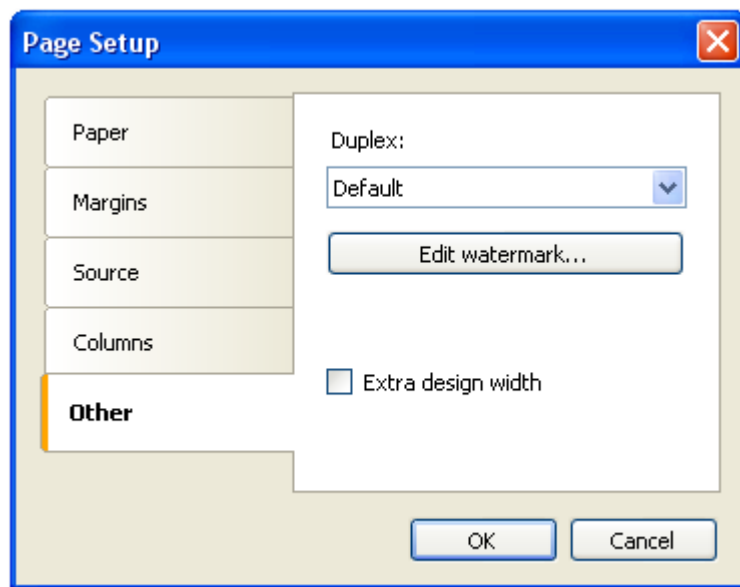


The source can be chosen in the "Print" dialog as well.

The "Columns" group allows setup the column parameters for multi-columned report. For this, the number of columns need to be indicated and (optional) correct the width of the column and the position of every column:



The "Other" group allows giving certain helpful page properties. It is possible to indicate duplex mode for duplex printing if your printer supports this mode. Here it is also possible to set the watermark, which will be printed on prepared report pages:



The "Extra design width" checkbox allows to increase the page width in the design mode. It may be useful if you work with such objects as "Table" or "Matrix".

The duplex mode can be chosen in the "Print" dialog as well.

## Bands

The band is an object which is located directly on the report page and is a container for other objects like "Text", "Picture" and others.

In all, in FastReport there are 13 types of bands. Depending on its type, the band is printed in a certain place in the report.

Band	How it's printed
Report Title	It is printed once at the very beginning of the report. You can choose the order of printing - before the "Page Header" band or after it - with the help of the "TitleBeforeHeader" page property. Changing this property can be done with the help of "Properties" window. By default, property is equal to true, that is, report title is printed before page header.
Report Summary	It is printed once at the end of the report, after the last data row, but before the "Page Footer" band.
Page Header	It is printed on top of every page of the report.
Page Footer	It is printed at the bottom of every page of the report.
Column Header	This band is used when printing a multi-columned report (when the number of columns indicated in the page setup > 1). It is printed on top of every column after the Page Header band.
Column Footer	Printed at the bottom of every column, before the Page Footer band.
Data	This band is connected to the data source and is printed as many times as there are rows in the source.
Data Header	This band is connected to the "Data" band and is printed before the first data row.
Data Footer	This band is connected to the "Data" band and is printed after the last data row.
Group Header	It is printed at the beginning of every group, when the value of the group condition changes.
Group Footer	It is printed at the end of every group.
Child	This band can be connected to any band, including another child band. It is printed immediately after its parent.
Overlay	Printed as a background on every report page.

## Bands in designer

A band in the designer appears in form of a rectangular area. A band, like many other report objects, can have a border and fill (by default they are disabled). Apart from this, a band displays a grid. To set the grid mode, go the "View|Options..." menu and choose "Report page". Grid can also be enabled or disabled in the "View" menu.

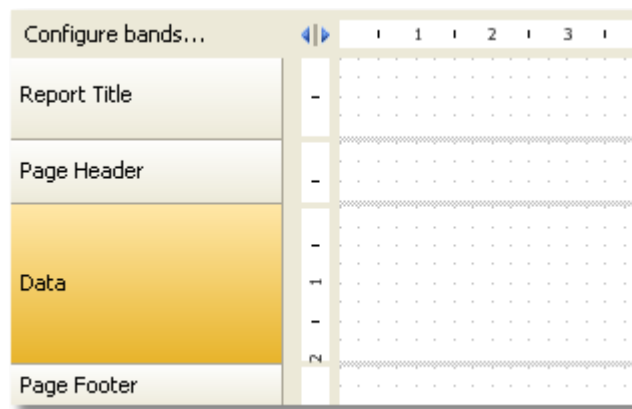
You can set the band's height in three ways:


- place the mouse pointer at the bottom of the band. The cursor shape will be changed to "horizontal splitter" and you can resize a band.
- drag the band handle on the left ruler.
- use "Properties" window to set the band's Height property.

The designer has two modes of displaying bands, between which you can switch at any time. In the first mode, every band has got a header, which contains the title of the band and useful information about it (for example, the name of the data source to which it is connected).



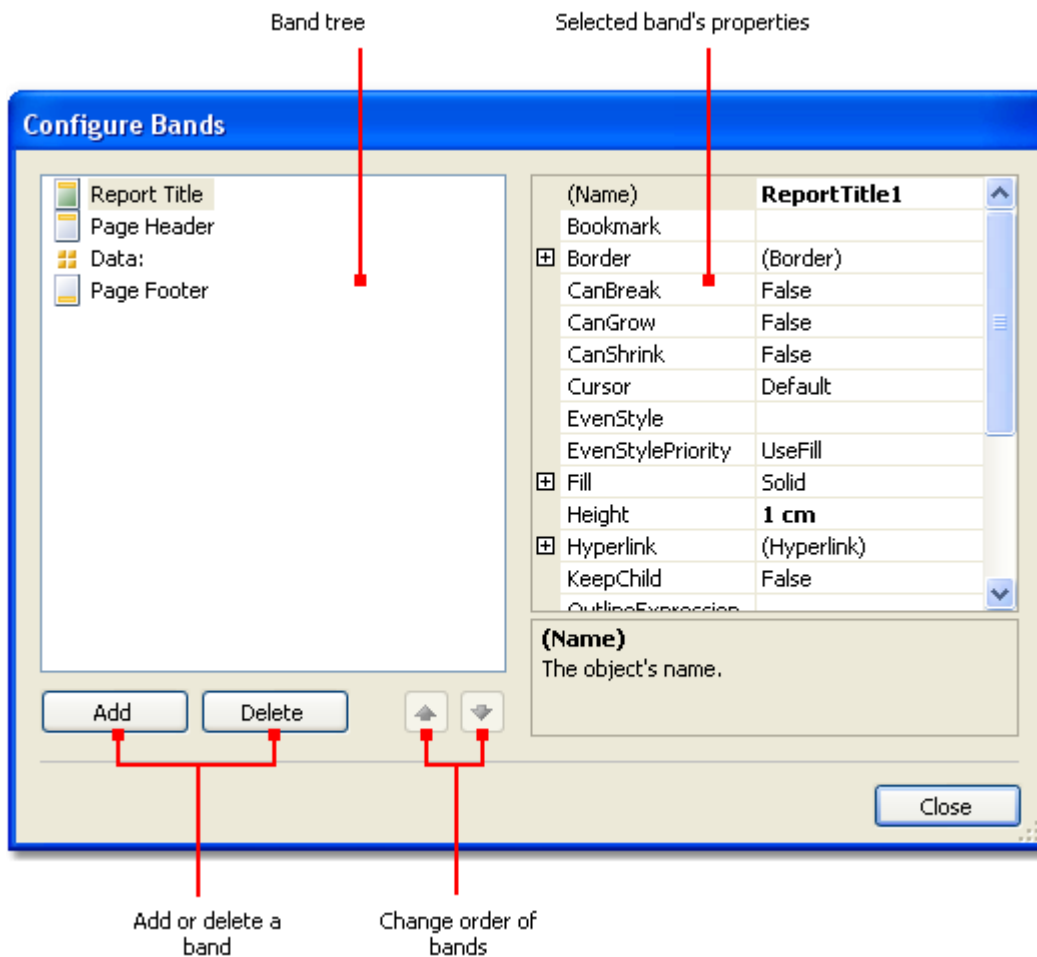
In the second mode, the band does not have a header. Instead of that, on the left side of the window, the structure of the bands is displayed. This mode helps to easily understand the structure of the report, especially if it was not created by you.



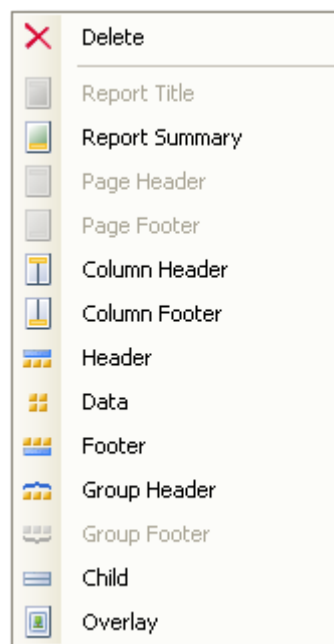
You can switch between these modes by clicking the  button.

## Configuring bands

You can configure the bands in the "Configure Bands" window. It can be called from the "Report|Configure Bands..." menu or with the help of the "Configure bands" button, placed over the bands tree:



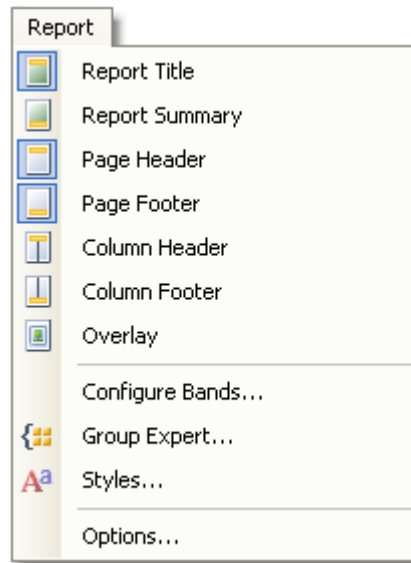
In this window, it is possible to add bands into the report, delete them or change their order. To add a band, click the "Add" button or right click on band tree. A context menu will come up containing a list of bands. A band which cannot be added is dimmed.



The "Add" operation depends on what band was chosen in the band tree. For example, adding

"Data Header" and "Data Footer" bands is possible only if the "Data" band was selected beforehand.

There is also another way of configuring some bands. This can be done from the "Report" menu:



To delete a band, select it and press "Delete" key.

When configuring bands, FastReport does not allow to do operations which leads to the creation of a wrong report template. For example, you cannot delete the "Data" band, which is connected to the group - for this, the group needs to be deleted first. Another example, when deleting the "Data" band, its header and footer are deleted automatically. Also, it is not possible to delete a band if it is the only one on the page.

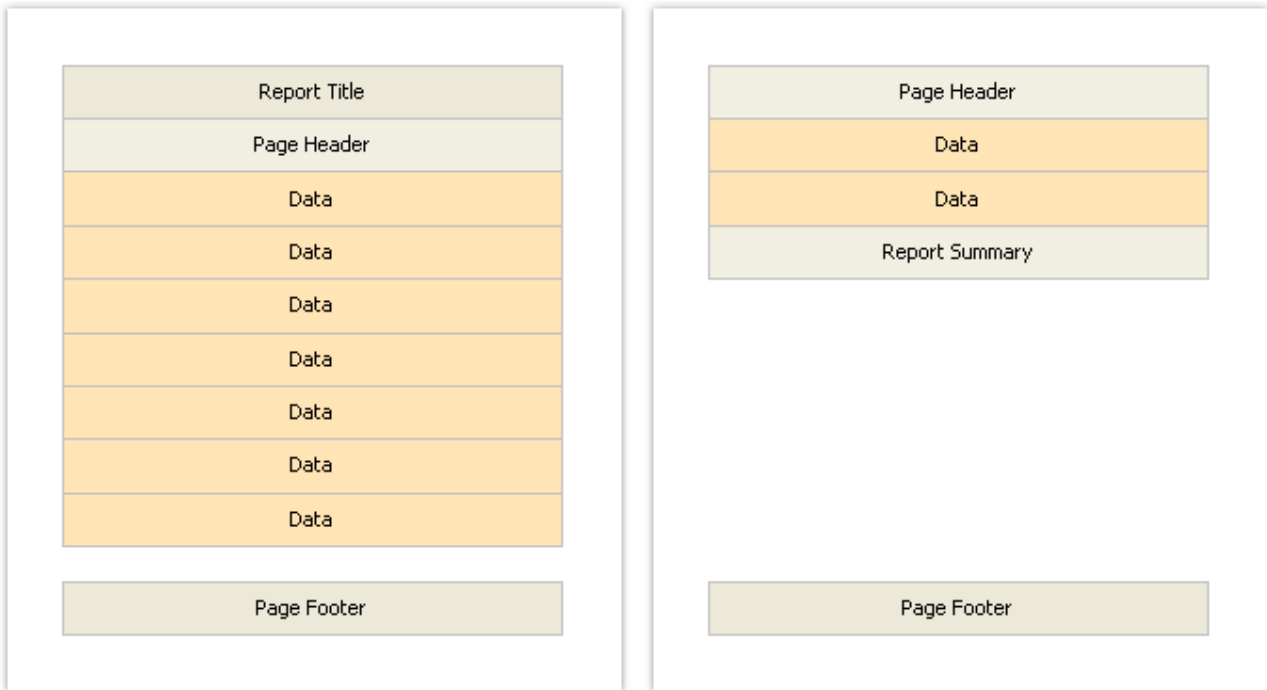
## Printing Bands

So, there are several bands placed on the page. How will FastReport compose a prepared report? Let us look at the following example:

Report Title	-	Report Title
Page Header	-	Page Header
Data: Employees	-	Data
Report Summary	-	Report Summary
Page Footer	-	Page Footer

The "Report Title" band will be printed first. The "Page Header" band will be printed immediately after it. Further, the "Data" band will be printed as many times as there are rows in the data source, to which the band is connected. After all the rows of the "Data" band have been printed, the "Report Summary" band is printed and at the bottom of the page - the "Page Footer" band. Printing of the report ends with this. A prepared report will be looking something like this:





In the process of printing, FastReport checks if there is enough space on the current page of the prepared report, so that the band can be printed. If there isn't enough space, the following occurs:

- page footer is printed;
- a new page is added;
- page header is printed;
- continues to print the band which did not fit on the previous page.

## Band properties

Every band has several useful properties, which affect the printing process. They can be configured by using the band's context menu. To do this, right-click on an empty space on the band, not occupied by other objects. Also, it is possible by clicking on the band header (if the classic display mode is used) or on band structure (otherwise). Another method – choose a band and change the corresponding properties in the "Properties" window.

Property	Description
CanGrow, CanShrink	These properties determine whether a band can grow or shrink depending on the size of the objects contained in the band. If both properties are disabled, the band will always have the size specified in the designer. Read more about this in the <a href="#">"Report Creation"</a> chapter.
CanBreak	If the property is enabled, FastReport tries to print a part of the band's contents on the available space, that is, "break" the band. Read more about this in the <a href="#">"Report Creation"</a> chapter.
StartNewPage	Printing a band with such property begins on a new page. This property is usually used when printing groups; that is, every group is printed on a new page.

PrintOnBottom	A band with this property is printed at the bottom of the page, before the "Page Footer" band .This can be useful when printing certain documents, where the total sum is supposed to be printed at the bottom of the page.
RepeatOnEveryPage	The bands - "Data Header", "Data Footer", "Group Header" and "Group Footer" - have got this property. This type of band will be printed on each new page, when data printing is being done. Read more about this in the <a href="#">"Report Creation"</a> chapter.

## Report objects

A wide variety of objects can be used in a report.

Icon	Name	Description
	"Text" (TextObject)	Displays one or several text lines.
	"Picture" (PictureObject)	Displays a picture.
	"Line" (LineObject)	Displays a line. A line can be vertical, horizontal or diagonal.
	"Shape" (ShapeObject)	Displays one of the geometrical shapes - rectangle, ellipse, triangle and others.
	"Rich Text" (RichObject)	Displays a formatted text (in RTF format).
	"Barcode" (BarcodeObject)	Displays a barcode.
	"CheckBox" (CheckBoxObject)	Displays a checkbox which can have two states - "Enabled" or "Disabled".
	"Table" (TableObject)	Displays a table containing rows, columns and cells.
	"Matrix" (MatrixObject)	Displays a matrix (also known as "Cross-tab").
	"Chart" (MSChartObject)	Displays a chart.
	"Zip Code" (ZipCodeObject)	Displays a zip code.
	"Cellular Text" (CellularTextObject)	Displays each character of a text in its individual cell.

An object can be used to display an information ("Text" object) or to improve the report appearance ("Picture", "Line", "Shape" objects). Complex objects like "Table" and "Matrix" can contain other simple objects.

## Common object properties

All report objects are inherited from one basic class (ReportComponentBase) and have got certain set of common properties. Before studying each object, we will look at these properties.

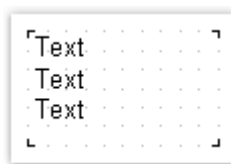
You can change the value of properties with the help of the "Properties" window. Some properties can be changed using the object's context menu or toolbars (for example, border and fill).

Property	Description
Left, Top, Width, Height	A report object, as a rule, is a rectangle. It has coordinates (properties Left, Top) and size (properties Width, Height).
Anchor	This property determines how the object will be changing its position and/or its size when the container on which it is laying grows or shrinks. By using Anchor, it can be done in such a way that, the object expands or moves synchronously with container. Read more about this property in the <a href="#">"Dynamic layout"</a> chapter.
Dock	This property determines on which side of the container the object will be docked. Read more about this property in the <a href="#">"Dynamic layout"</a> chapter.
Border, Fill	These properties control the object's border and fill. They can be changed using the "Border and Fill" toolbar.
CanGrow, CanShrink	These properties allow fitting the height of the object in such a way that it fits the whole text. Read more about this property in the <a href="#">"Dynamic layout"</a> chapter.
ShiftMode	An object, whose property is enabled, will be moving up and down, if the object above on can either grow or shrink. Read more about this property in the <a href="#">"Dynamic layout"</a> chapter.
GrowToBottom	An object, whose property is enabled, will be stretched to the bottom of a band. Read more about this property in the <a href="#">"Dynamic layout"</a> chapter.
CanBreak	Objects "Text" and "Rich Text" have this property. It determines whether the object's contents can be broken into parts.
PrintOn	This property determines on which pages the object can be printed. Read more about this property in the <a href="#">"Booklet-type report"</a> chapter.
Cursor	This property determines the type of mouse cursor when it is located over an object. The property works only in the preview window.
Visible	This property determines whether the object will be displayed in the report. Invisible object is never displayed in the preview window and is never printed on the printer as well.
Printable	This property determines whether the object will be printed on the printer. If this property is disabled, then the object will be visible in the preview window but it will not be printed.
Hyperlink	This property makes it possible to make the report object interactive. Read more about this property in the <a href="#">"Interactive reports"</a> chapter.
Bookmark	This property is used together with the "Hyperlink" property. It can contain any expression. The expression will be

	calculated when the report will be working, and its value will be used as bookmark's name.
Restrictions	This property restricts certain operations, such as moving, resizing, deleting the object.
Style	You can assign the style name to this property. When this is done, the object will become like it has been indicated in the style. If the parameters of the style changes, the appearance of the object changes as well.

## The "Text" object

The "Text" object is the main object which you will use often. It looks like this:

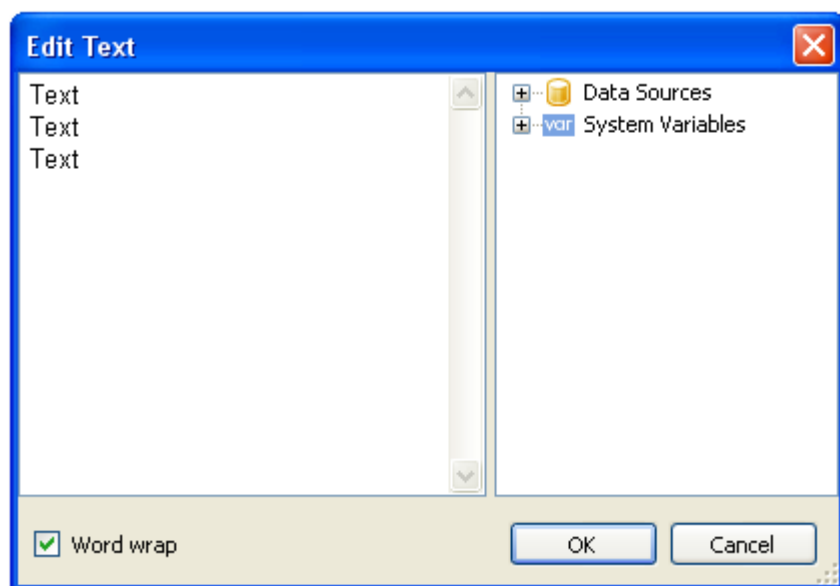


The object can display any text data, specifically:

- one or several text lines;
- data columns;
- report parameters;
- total values;
- expressions;
- any combination of the above items.

## Text editing

In order to edit an object's text, just double click on it. You will see a text editor:



There is a data tree on the right side of the editor, which elements can be added into the text. This can be done by dragging the element onto needed place by using the mouse. Another way to insert an element into the text - double click on the element, and it will be added onto the cursor's current position.

In order to save the changes and close the editor window, click the OK button or press the Ctrl+Enter keys.

Another method of editing a text - in-place editing. To do this, select the "Text" object and press Enter. To finish editing, click somewhere outside the objects bounds or press Ctrl+Enter. Press Esc key to cancel the changes.

When editing an object in-place, its size can be changed by using a mouse.

## Displaying the expressions

The "Text" object can contain a plain text mixed with expressions. For example:

*Today is [Date]*

When printing such an object, all expressions contained in the text will be calculated. So the result may look like this:

*Today is 12.09.2010*

As seen, expressions are identified by square brackets. This is set in the "Brackets" property, which by default contains the string "[,]". When needed, you can use a different set of symbols, for example "<, >", or "<!, !>". In the last case, an expression in the text will be like this:

*Today is <!Date!>*

Apart from this, it is possible to disable all expressions. To do this, set the AllowExpressions property to **false**. In this case the text will be shown "as is".

Inside the square brackets, you can use any valid expression. Read more about expressions in the ["Expressions"](#) chapter. For example, an object with the following text:

$2 * 2 = [2 * 2]$

will be printed like this:

$2 * 2 = 4$

Frequent mistake - attempt to write an expression outside the square brackets. Reminding, that it is considered an expression and gets executed only that, which is located inside square brackets. For example:

$2 * 2 = [2] * [2]$

This text will be printed this way:

$2 * 2 = 2 * 2$

There may be elements inside an expression that needs own square brackets. For example, it may be a reference to a system variable (see the ["Expressions"](#) chapter for details). Let's look

at the following example:

*The next page: [[Page] + 1]*

The text contains an expression `[Page] + 1`. `Page` is a system variable which returns the number of the current report page. It is included in own brackets. These brackets must be square brackets, regardless of the "Text" object settings.

Strict speaking, we were supposed to use two pairs of square brackets when using the "Date" system variable in the examples above:

*Today is [[Date]]*

However FastReport allows to leave out unnecessary pair of brackets if there is only one member in an expression.

## Displaying the data columns

You can print the data column in the following way:

*[Datasource name.Column name]*

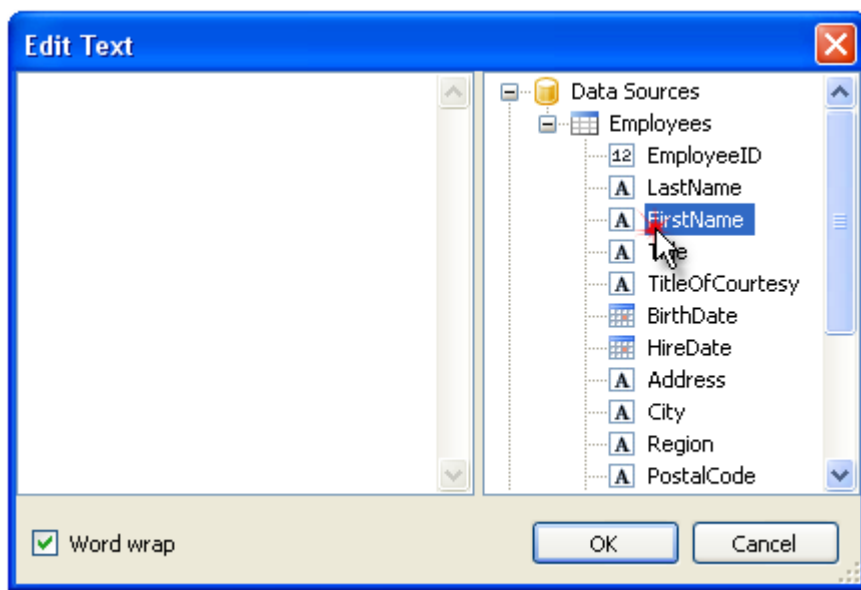
As you can see, the square brackets are used here. The data source name and data column name are separated by the period. For example:

*[Employees.FirstName]*

Read more about using the data columns in the ["Expressions"](#) chapter.

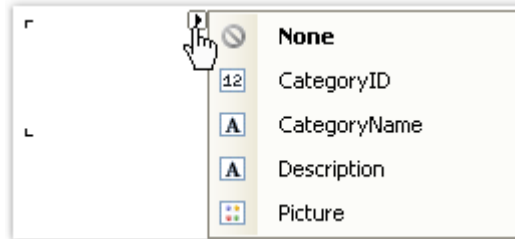
There are several ways to insert a data column into the "Text" object.

1. In the "Text" object's editor we write the name of the data column manually. This method is the most inconvenient as it is easy to make a mistake.
2. In the object's editor we choose the needed data column and drag&drop it into the text:



3. Click on the small button in the upper right corner of the object and choose the data

column from a list:



4. Drag&drop a data column from the "Data" window into the report page. In this case the "Text" object is created which contains a link to the column.

### HTML tags

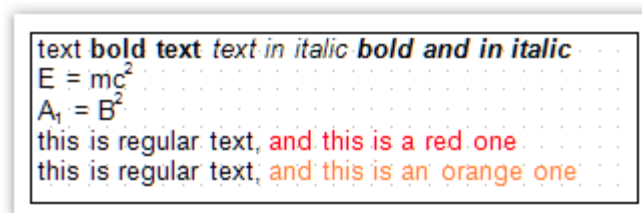
You may use some simple HTML tags in the "Text" object. By default, tags are disabled; to enable it, go "Properties" window and set the "HtmlTags" property to **true**. Here is a list of supported tags:

Tag	Description
<code>&lt;b&gt;...&lt;/b&gt;</code>	Bold text.
<code>&lt;i&gt;...&lt;/i&gt;</code>	Italic text.
<code>&lt;u&gt;...&lt;/u&gt;</code>	Underlined text.
<code>&lt;strike&gt;...&lt;/strike&gt;</code>	Strikeout text.
<code>&lt;sub&gt;...&lt;/sub&gt;</code>	Subscript.
<code>&lt;sup&gt;...&lt;/sup&gt;</code>	Superscript.
<code>&lt;font color=...&gt;...&lt;/font&gt;</code>	Font color. The color may be either the named color (such as DarkGray), or a hexadecimal code in the #RGB format, for example #FF8030.

The following examples demonstrate how these tags can be used.

```
text <b>bold text</b> <i>text in italic</i> <b><i>bold and in italic</i></b></i>
E = mc<sup>2</sup>
A<sub>1</sub> = B<sup>2</sup>
this is regular text, <font color=red>and this is a red one</font>
this is regular text, <font color=#FF8030>and this is an orange one</font>
```

This text will be displayed in the following way:





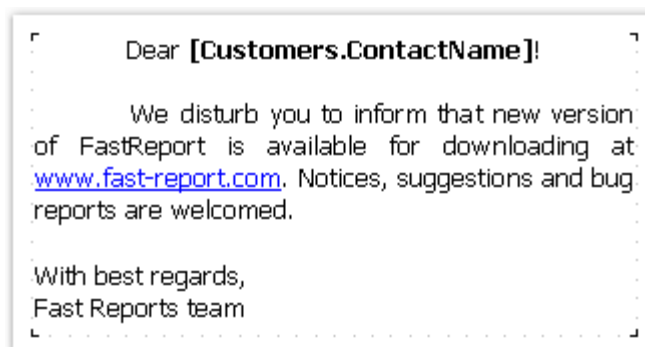
## Object's properties

Property	Description
AllowExpressions	This property allows to turn on or off the expression handling. It is on by default.
Angle	This property indicates the text rotation, in degrees.
AutoShrink	This property allows to shrink the font size or font width automatically to fit the text.
AutoShrinkMinSize	This property determines the minimum size of a font, or the minimum font width ratio, if the AutoShrink property is used.
AutoWidth	This property allows to calculate the width of object automatically.
Brackets	This property contains a pair of symbols that designate an expression.
BreakTo	With this property you can organize the text flow from one text object to another. For example, we have "A" and "B" text objects. The "A" object contains the long text that does not fit in the object's bounds. You can set the A.BreakTo to B, so the "B" object will display the part of text that does not fit in "A".
Clip	This property determines whether it is necessary to clip a text outside of object's bounds. It is on by default.
Duplicates	This property determines how the repeated values will be printed. Read more about this property in the <a href="#">"Formatting"</a> chapter.
FirstTabOffset	This property determines the offset of the first TAB symbol, in pixels.
FontWidthRatio	Use this property to make the font wider or narrower. By default the property is set to 1. To make the font wider, set the property to value > 1. To make the font narrower, set the property to value between 0 and 1.
HideValue	This property is of string type. It allows to hide values that are equal to the value of this property. Read more about this property in the <a href="#">"Formatting"</a> chapter.
HideZeros	This property allows to hide zero values. Read more about this property in the <a href="#">"Formatting"</a> chapter.
Highlight	This property allows to setup the conditional highlight. Read more about this in the <a href="#">"Formatting"</a> chapter.
HorzAlign, VertAlign	These properties determine the text alignment.
HtmlTags	Allows simple HTML tags in the object's text. Read more about this property in the <a href="#">"HTML tags"</a> chapter.
LineHeight	This property allows to explicitly set the height of a text line. By default it is set to 0, so the default line spacing is used.
NullValue	The text that will be printed instead of a null value. You also need to uncheck the "Convert null values" option in the "Report/Options..." menu.

Padding	This property allows to setup the padding, in pixels.
RightToLeft	This property indicates whether the text should be displayed in right-to-left order.
TabWidth	This property determines the width of the TAB symbol, in pixels.
Text	This property contains the text of the object.
TextFill	This property determines the text fill. Use this property editor to choose between different fill types.
Trimming	This property determines how to trim the text that does not fit inside the object's bounds. It is used only if the "WordWrap" property is set to <b>false</b> .
Underlines	This property allows to display a graphical line after each text line. This property can be used only if the text is top-aligned.
WordWrap	This property determines whether it is necessary to wrap a text by words.
Wysiwyg	This property changes the display mode of the "Text" object to match the screen and the final printout. This mode is also used if you use the justify-align or non-standard line height.

## The "Rich Text" object

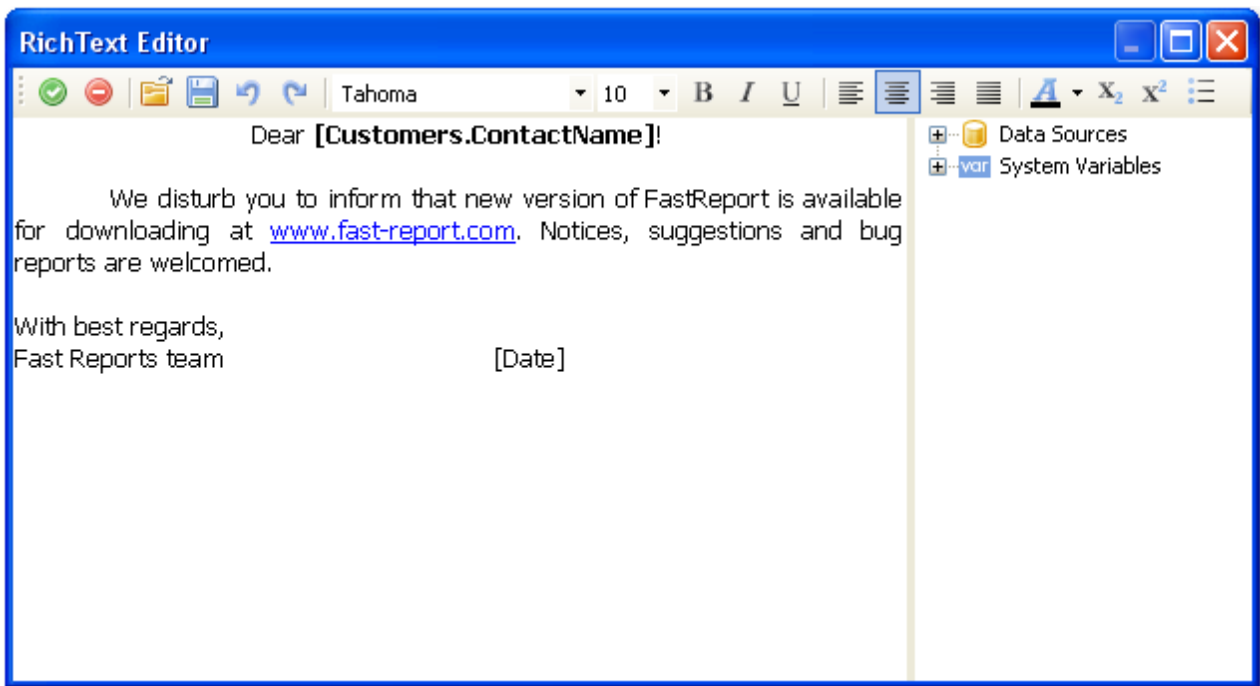
This object displays the formatted text (in the RTF format). It looks like this:




Try to use the "Text" object to display a text. When you export the report to some document formats, the "Rich Text" object will be exported as a picture.

This object supports only the solid fill type. Gradient, hatch, glass fills are not supported.

To edit the object, double click on it. You will see the editor window:



You can also use the Microsoft Word to create a text. When you have done, save the text in the .RTF format. Next, you need to open the "Rich Text" editor and load the .RTF file into it by pressing the  button.

The "Rich Text" object does not support all of the Microsoft Word formatting features.

You can display a data in this object the following ways:

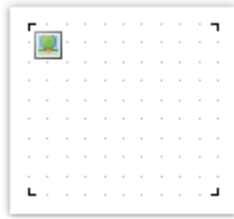
- you can insert an expression in the object's text, just as you do this in the "Text" object. Insert the necessary data column into the text;
- use the "DataColumn" property to bind the object to the column.

The object has the following properties:

Property	Description
AllowExpressions	This property allows to turn on or off the expression handling. It is on by default.
Brackets	This property contains a pair of symbols that designate an expression.
DataColumn	The data column that this object is bound to.
Text	This property contains the RTF text.
Padding	The padding, in pixels.

## The "Picture" object

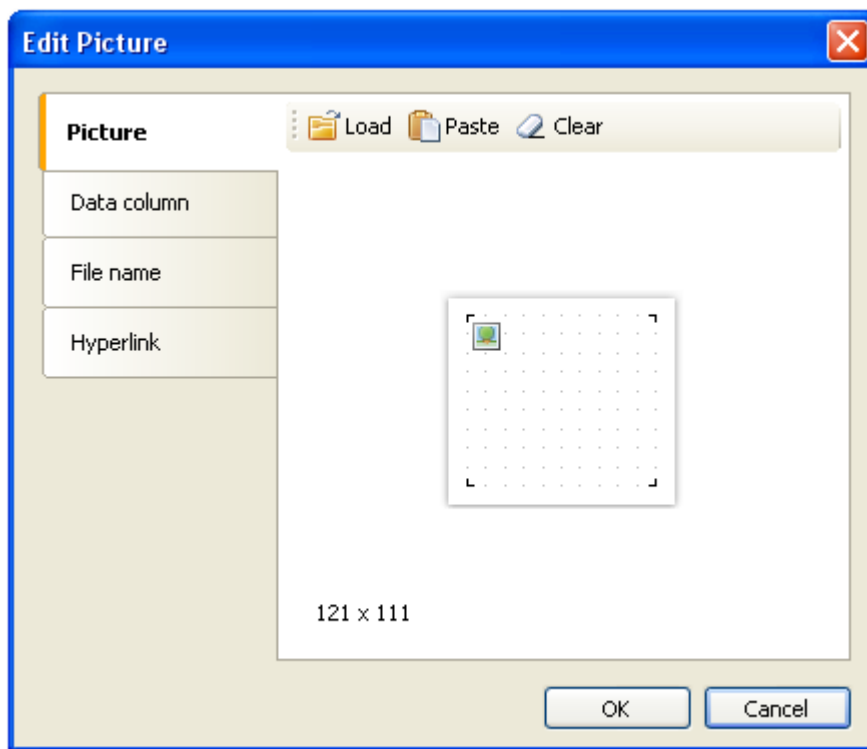
An object can display graphics in the following formats: BMP, PNG, JPG, GIF, TIFF, ICO, EMF, WMF. With the help of the "Picture" object, you can print your company logo, a photo of employee or any graphical information. The object looks like this:



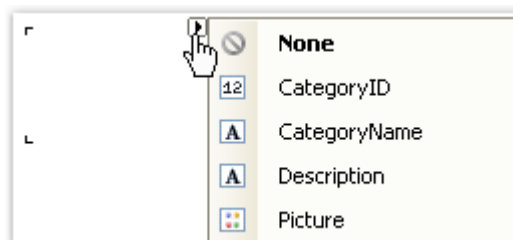
An object can show data from the following sources:

Source	Description
File with a picture	Picture is loaded from a file and is saved inside the report. Picture is stored in the "Image" property.
Data column	Picture from a data column. Name of the column is stored in the "DataColumn" property.
File name	Picture is loaded from a file with the given name. Name of file is stored in the "ImageLocation" property. Picture is never stored inside the report. You should distribute the picture file along with the report.
URL	Picture is loaded from the internet every time the report is created. Image is never stored inside the report. URL is stored in the "ImageLocation" property.

In order to call a picture editor, double click on the object. In the editor, you can choose the data source for the picture:



In order to bind the object to a data column, click on the small button in the upper right corner of the object and choose the data column from a list:

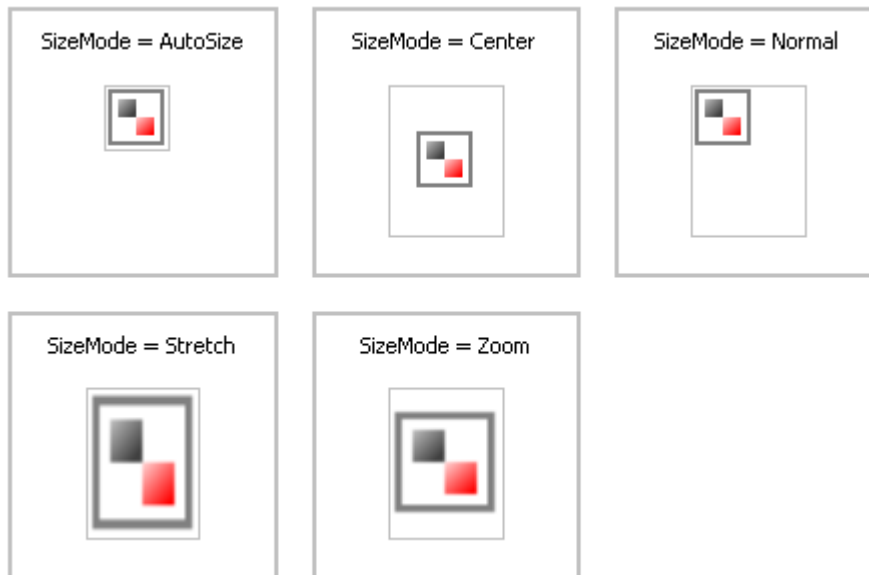


You also can drag&drop a data column from the "Data" window into the report page. In this case the "Picture" object is created which contains a link to the column. The column you drag should have the "byte[]" data type.

In the context menu of the "Picture" object you can choose the size mode:

- AutoSize. The object gets the size of the picture.
- CenterImage. The picture is centered inside the object.
- Normal. The picture is displayed in the left corner of the object.
- StretchImage. The picture is stretched to the size of the object.
- Zoom. The picture is stretched to the size of the object in accordance with the aspect ratio.

The difference between modes is shown in the following picture:

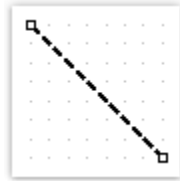


The "Picture" object has the following properties:


Property	Description
Angle	The rotation angle, in degrees. Possible values for this property are 0, 90, 180, 270.
SizeMode	The size mode.
Transparency	The degree of transparency of the pictures. The property can have values between 0 and 1. The value 0 (by default) means that the picture is opaque.
TransparentColor	The color which will be transparent when displaying the picture.
Image	The picture.
DataColumn	The data column that this object is bound to.
ImageLocation	This property can contain name of the file or URL. The picture will be loaded from this location when building the report.
Padding	The padding, in pixels.
ShowErrorMessage	Shows the "No picture" picture, in case when the picture is empty. This property makes sense to use if the picture is downloaded from the Internet.


## The "Line" object

The "Line" object can display horizontal, vertical or diagonal line. The object is as follows:



If possible, use the object's border instead of "Line" object. This will simplify the report and avoid possible problems with the export to different formats.

FastReport designer has convenient tools for drawing a line. In order to insert a line into a report, click the  button on the "Objects" toolbar and in the menu choose the "Line" object or "Diagonal Line". Place the mouse cursor at the location where the line will start from. Then left click and hold the mouse, in order to draw the line. After this, you can draw the line again.

When all the lines have been drawn, click on the  button on the "Objects" toolbar.

An ordinary line differs from a diagonal line in that you can make it only vertical or horizontal.

Do not choose the "Double" line style for this object. This style applies only to the object's border.

The "Line" object has the following properties:

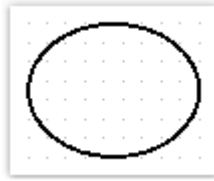
Property	Description
Diagonal	The property determines weather the line is diagonal. An ordinary line can be turned into a diagonal one by enabling this property.
StartCap, EndCap	These properties allow to setup the line caps. You can use one of the following cap styles: <ul style="list-style-type: none"><li>• ellipse;</li><li>• rectangle;</li><li>• diamond;</li><li>• arrow.</li></ul> Size of the cap can be set in the Width, Height properties of the cap. You can configure caps for each end of the line.


## The "Shape" object

The "Shape" object displays one of the following shapes:

- rectangle;
- rounded rectangle;
- ellipse;
- triangle;
- diamond.

The object is as follows:



In order to insert a shape into the report, click the  button on the "Objects" toolbar and choose the needed shape type.

The shape, like any other report object, has a fill and border. Contrary to the "Text" object, you cannot control each border line. Also, don't use the "Double" line style.

Instead of rectangular shape, use the object's borders if possible.

The "Shape" object has the following properties:

Property	Description
Shape	This property determines the type of shape.
Curve	This property is used with the "RoundRectangle" shape. It allows to set the curve.

## The "Barcode" object

The object displays barcodes in the report. It looks like this:



The object supports the following types of barcodes:

Code	Length	Allowed symbols
2 of 5 Interleaved		0-9
2 of 5 Industrial		0-9
2 of 5 Matrix		0-9
Codabar		0-9, -\$/./+.
Code128		128 ASCII chars
Code39		0-9,A-Z, -. *\$/+%
Code39 Extended		128 ASCII chars



Code93		0-9,A-Z, -. *\$/+%
Code93 Extended		128 ASCII chars
EAN8	8	0-9
EAN13	13	0-9
MSI		0-9
PostNet		0-9
UPC A	12	0-9
UPC E0	6	0-9
UPC E1	6	0-9
2-Digit Supplement	2	0-9
5-Digit Supplement	5	0-9
PDF417		any
Datamatrix		any

A detailed description of different barcode types can be found in the internet, for example, here:

<http://www.barcodeisland.com>

Barcode data in an object is of a string type. The string can contain any symbol, allowed for the chosen type of barcode. You can choose the type of barcode in the context menu of the "Barcode" object.

You can connect an object to data by using one of the following methods:

- set the barcode data in the "Text" property;
- bind the object to a data column using the "DataColumn" property;
- set the expression that returns the barcode data in the "Expression" property.

The "Barcode" object has the following properties:

Property	Description
Barcode	This property contains barcode-specific settings. Expand this property to set these settings.
Angle	This property determines the rotation of a barcode, in degrees. You can use one of the following values: 0, 90, 180, 270.
Zoom	This property allows to zoom a barcode. It is used along with the "AutoSize" property.
AutoSize	If this property is on, the object will stretch in order to display a whole barcode. If this property is off, the barcode will stretch to object's bounds.
ShowText	This property determines whether it is necessary to show the human-readable text.

DataColumn	The data column which this object is bound to.
Expression	The expression that returns the barcode data.
Text	The barcode data.
Padding	The padding, in pixels.

The following properties are specific to the barcode type. To change them, select the barcode, go "Properties" window and expand the "Barcode" property.

Property	Description																		
WideBarRatio	This property is specific to all linear barcodes. It determines the wide-to-narrow bar ratio. For most of barcode types, the value for this property should be between 2 and 3.																		
CalcChecksum	This property is specific to all linear barcodes. It determines whether is necessary to calculate the check sum automatically. If this property is off, you should provide the check digit in the barcode data.																		
AutoEncode	<p>This property is specific to the Code128 barcode. This code has three different encodings - A, B, C. You should either set the encoding explicitly in the barcode data, or set this property to true. In this case the encoding will be chosen automatically.</p> <p>Use the following control codes in the barcode data:</p> <table border="1"> <thead> <tr> <th>Code</th> <th>Meaning</th> </tr> </thead> <tbody> <tr> <td>&amp;A;</td> <td>START A / CODE A</td> </tr> <tr> <td>&amp;B;</td> <td>START B / CODE B</td> </tr> <tr> <td>&amp;C;</td> <td>START C / CODE C</td> </tr> <tr> <td>&amp;S;</td> <td>SHIFT</td> </tr> <tr> <td>&amp;1;</td> <td>FNC1</td> </tr> <tr> <td>&amp;2;</td> <td>FNC2</td> </tr> <tr> <td>&amp;3;</td> <td>FNC3</td> </tr> <tr> <td>&amp;4;</td> <td>FNC4</td> </tr> </tbody> </table> <p>If you set the "AutoEncode" property to true, all control codes will be ignored.</p> <p>Example of use the control codes:</p> <p><i>&amp;C;1234&amp;B;ABC</i></p>	Code	Meaning	&A;	START A / CODE A	&B;	START B / CODE B	&C;	START C / CODE C	&S;	SHIFT	&1;	FNC1	&2;	FNC2	&3;	FNC3	&4;	FNC4
Code	Meaning																		
&A;	START A / CODE A																		
&B;	START B / CODE B																		
&C;	START C / CODE C																		
&S;	SHIFT																		
&1;	FNC1																		
&2;	FNC2																		
&3;	FNC3																		
&4;	FNC4																		
AspectRatio	This property is specific to the PDF417 barcode. It determines the height-to-width aspect ratio and is used to calculate the barcode size automatically (in case the "Columns" and "Rows" properties are not set).																		

CodePage	This property is specific to the PDF417 and Datamatrix barcode. It determines the code page which is used to convert non-ASCII chars. For example, the default windows codepage is 1251.
Columns, Rows	These properties are specific to the PDF417 barcode. They determine the number of columns and rows in a barcode. If both properties are set to 0, the size of the barcode will be calculated automatically. In this case the "AspectRatio" property is used as well.
CompactionMode	This property is specific to the PDF417 barcode. It determines the PDF417 data compaction mode.
ErrorCorrection	This property is specific to the PDF417 barcode. It determines the error correction level.
PixelSize	This property is specific to the PDF417 barcode. It determines the size of barcode element, in screen pixels. As a rule, the element's height should be greater than the element width by 3 times or more.
Encoding	This property is specific to the Datamatrix barcode. It determines the Datamatrix data encoding.
PixelSize	This property is specific to the Datamatrix barcode. It determines the size of barcode element, in pixels.
SymbolSize	This property is specific to the Datamatrix barcode. It determines the size of barcode symbol.

## The "CheckBox" object

The object displays the checkbox in the report. It looks as follows:



The object can display two states: "Checked" and "Unchecked". Use the following ways to handle the object's state:

- set the state in the "Checked" property;
- bind the object to a data column using the "DataColumn" property;
- set the expression that returns the **true** or **false** in the "Expression" property.

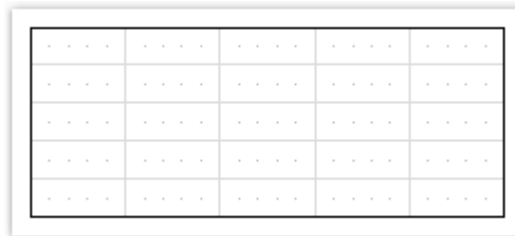
The "CheckBox" object has the following properties:

Property	Description
CheckedSymbol, UncheckedSymbol	These properties determine the symbol that is shown in the object, depending on the object's state.
CheckColor	This property determines the color of the check symbol.

CheckWidthRatio	Use this property to set the check width ratio. The width of the check symbol depends on the size of the object. You can use values in the 0.2 - 2 range.
HideIfUnchecked	This property allows to hide the object if it is unchecked.
Checked	This property controls the state of the object.
DataColumn	The data column which this object is bound to. The type of column should be either bool or int.
Expression	The expression that returns the <b>true</b> or <b>false</b> .

## The "Table" object

The "Table" object is made up of rows, columns and cells. It is a simplified analog of Microsoft Excel table. It looks like this:



You can learn more about this object in the ["Creating reports"](#) chapter.

The "Table" object has the following properties:

Property	Description
ColumnCount	Use this property to quickly set the number of columns. If columns in a table are few, they get added, and when they are more, they get deleted.
RowCount	Use this property to quickly set the number of rows. If rows in a table are few, they get added, and when they are more, they get deleted.
FixedColumns	The property determines how many columns in the table are fixed. Fixed columns form the table header. Printing of the header is controlled by the "RepeatHeaders" property.
FixedRows	The property determines how many rows in the table are fixed. Fixed rows form the table header. Printing of the header is controlled by the "RepeatHeaders" property.
RepeatHeaders	The property allows printing the table header on every new page. This property works only for tables which are formed dynamically.

## The "Matrix" object

The "Matrix" object is, like the "Table" object, made up of rows, columns and cells. At the same time, it is not known beforehand how many rows and columns will be in the matrix - this depends on the data to which it is connected.

The object looks like this:

Employee	[Year]	<b>Total</b>
[Name]	[Revenue]	
<b>Total</b>		

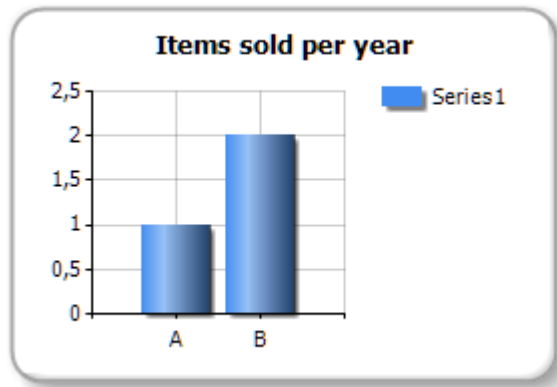
You can learn more about this object in the ["Creating reports"](#) chapter.

The "Matrix" object has the following properties:

Property	Description
RepeatHeaders	If matrix is divided on several pages, this property allows printing matrix header on each new page.
CellsSideBySide	This property determines how matrix cells will be located if the matrix has several data cell levels. Possible variants: <ul style="list-style-type: none"><li>• the cells are displayed side by side;</li><li>• the cells are displayed under each other.</li></ul>
Style	Using this property you can set a style for the whole matrix. You can choose one from predefined styles.
AutoSize	This property allows to calculate the matrix size automatically. Disable it if you want to control the object size manually.
DataSource	The property allows connecting the matrix to the data source. This property is set up automatically when you drag data column to the matrix. However, if you use expressions in cells, check that this property was set up correctly.
Filter	This property contains expression for data filtering which will be applicable to data source of the matrix (see "DataSource" property).

## The "Chart" object

The "MS Chart" object allows to display charts. There are more than 30 different series types available - bars, columns, areas, lines, bubbles, pie, circular, financial, pyramidal, ranges. The object looks like this:



You can learn more about this object in the ["Report creation"](#) chapter.

The "Chart" object has the following properties:

Property	Description
Chart	Reference to Microsoft Chart object.
AlignXValues	This property allows to align X values in different chart series (by inserting empty values). It is used if the chart contains two series or more.
AutoSeriesColumn, AutoSeriesColor, AutoSeriesSortOrder	These properties allows to set up automatically created series. Read more about this in the <a href="#">"Report creation"</a> chapter.
DataSource	The property allows connecting the chart to the data source.
Filter	This property contains expression for data filtering which will be applicable to data source of the chart (see "DataSource" property).

## The "Zip Code" object

The "Zip Code" object allows to print a zip code on envelopes. It may display numeric characters (0-9).

The object is as follows:



You can connect an object to data by using one of the following methods:

- set the zipcode data in the "Text" property;
- bind the object to a data column using the "DataColumn" property;
- set the expression that returns the zipcode data in the "Expression" property.

The "Zip Code" object has the following properties:

Property	Description
SegmentCount	The number of segments in a code. This property is set to 6 by default.
SegmentWidth, SegmentHeight	The size of a single code segment. The default size is 0.5x1cm.
Spacing	This property determines a distance between two segment's origins. The default value is 0.9cm.
ShowGrid	Determines whether it is necessary to display a grid.
ShowMarkers	Determines whether it is necessary to display the markers (bold horizontal lines above the zipcode).
DataColumn	The data column which this object is bound to.
Expression	The expression that returns the zipcode data.
Text	The text containing a zipcode.

### The "Cellular Text" object

This object can display each character of a text in its individual cell. It is often used to print some forms in financial applications.

The object is as follows:

S	i	m	p	l	e		t	e	x	t				
---	---	---	---	---	---	--	---	---	---	---	--	--	--	--

In fact, this object is directly inherited from the "Text" object. You may connect it to data in the same manner. For example, you may invoke the object's editor and type the following text:

*[Employees.FirstName]*

The "Cellular Text" object has the following properties:

Property	Description
CellWidth, CellHeight	These properties determine the size of a single cell. If both properties are 0 (by default), the cell size will be calculated automatically, depending on the font used.
HorzSpacing, VertSpacing	These properties determine the horizontal and vertical gap between adjacent cells.


## Your first report in the FastReport

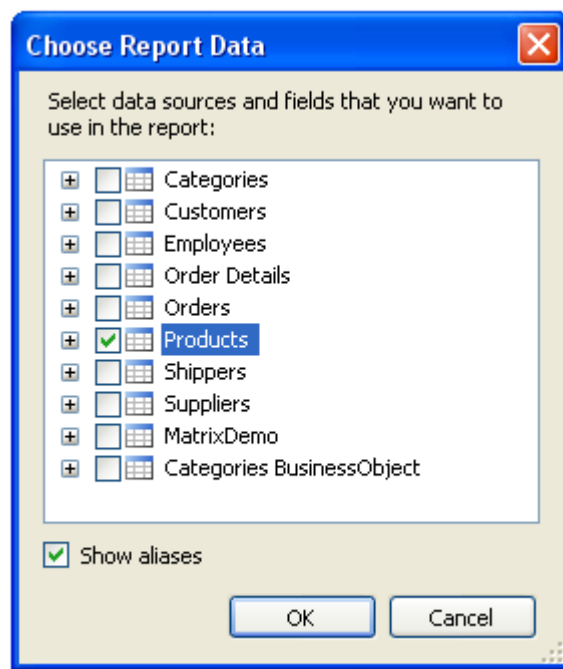
Let us create a simple report, which prints a list of products. We will use the Products table, which can be found in the demo data base as our data source.

Assuming that, you will be performing the actions written below in the demonstration program, Demo.exe, from which the report designer can be called.

### Example 1. Creating a report manually

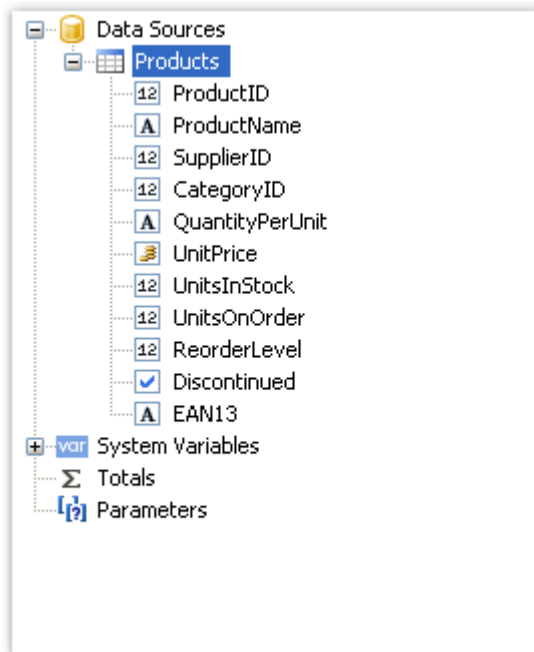
In this example, we will create a report manually. For this, we will do the following:

- press the  button on the toolbar, and in the "Add New Item" window, choose "Blank report";
- in the "Data" menu, choose the "Choose Report Data..." item and check the "Products" data source:

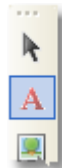


- switch to the "Data" service window (if it is not on the screen, it can be shown by choosing the "Data|Show Data Window" menu item). Expand the "Data Sources" item, then the "Products" item:

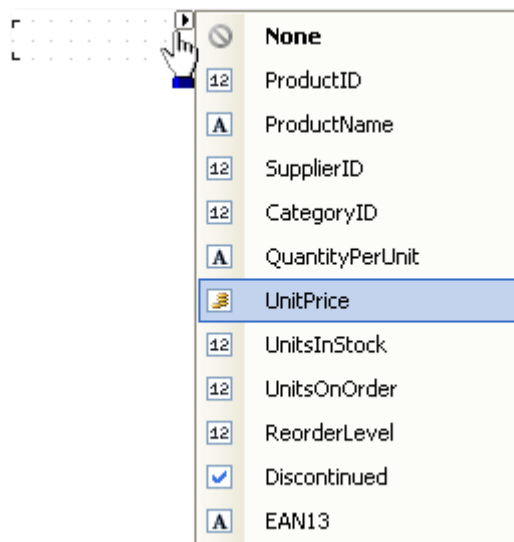




- drag the "ProductName" data column onto the "Data" band. FastReport creates a "Text" object, which is connected to this column and a header for it;
- we will create the "UnitPrice" data column by using another method. For this, press the "Text" button on the "Objects" toolbar:




- leave the mouse and drag its pointer onto the "Data" band - you will see that FastReport offers to insert an object. Choose the needed position and click the mouse to insert the object;
- place the mouse pointer on the object and click the small button in the right corner of the object. You will see a list of data columns. Choose the "UnitPrice" item from the list:



- create the "Text" object - header for the "UnitPrice" column. Place in on the "Page Header" band. Double click the object and write the text "Unit Price";
- create the "Text" object - report title. Place it on the "Report Title" band and write the text "PRODUCTS";
- set "Bold" as font style for all objects that are placed on the "Page Header" and "Report Title" bands. For this, select objects by pressing Shift, and press the **B** button on the "Text" toolbar. After this, the report will be looking like this:


Report Title	<b>PRODUCTS</b>	
Page Header	<b>ProductName</b>	<b>Unit Price</b>
Data: Products	[Products.ProductName]	[Products.UnitPrice]
Page Footer		

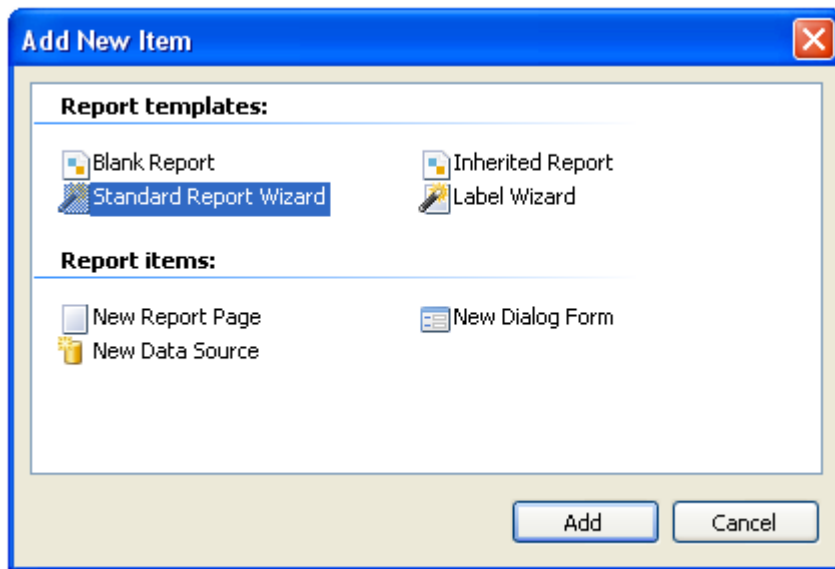
In order to run the report, click the  button on the toolbar. The report will be built and shown in the preview window:

<b>PRODUCTS</b>	
<b>ProductName</b>	<b>Unit Price</b>
Chai	18
Chang	19
Aniseed Syrup	10
Chef Anton's Cajun Seasoning	22
Chef Anton's Gumbo Mix	21,35
Grandma's Boysenberry Spread	25
Uncle Bob's Organic Dried	30
Northwoods Cranberry Sauce	40
Mishi Kobe Niku	97
Ikura	31

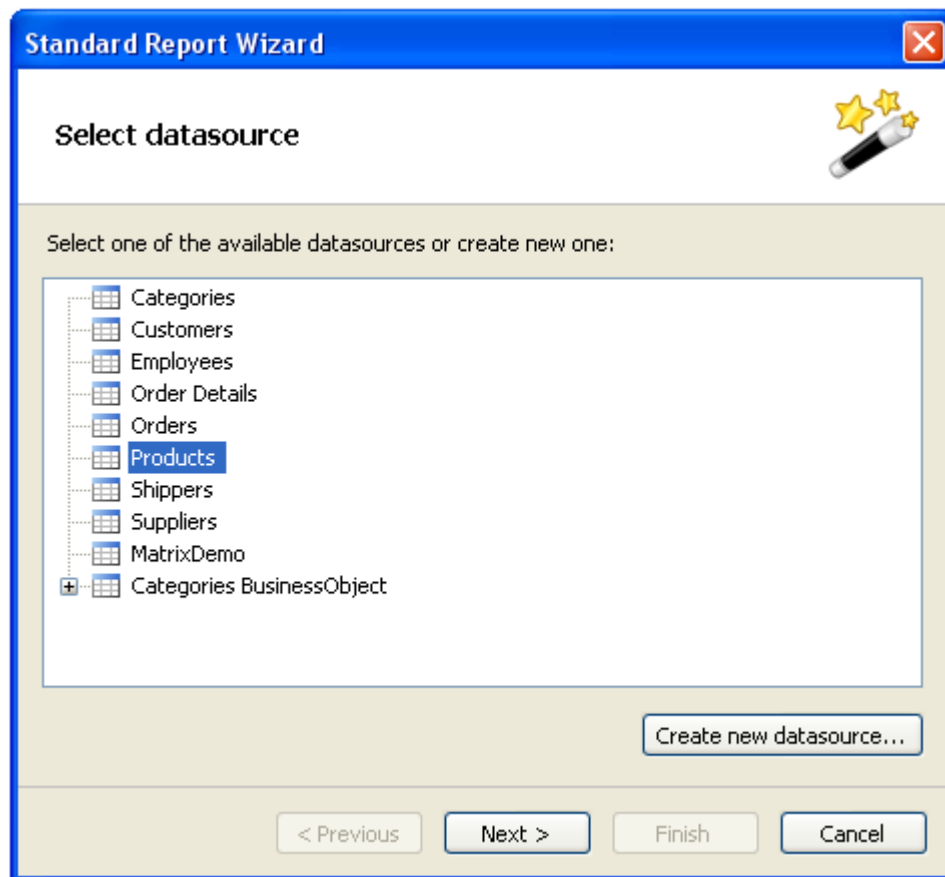
## Example 2. Creating a report with the wizard

In this example, we will create a report with the help of the "Standard Report Wizard". For this, do the following:

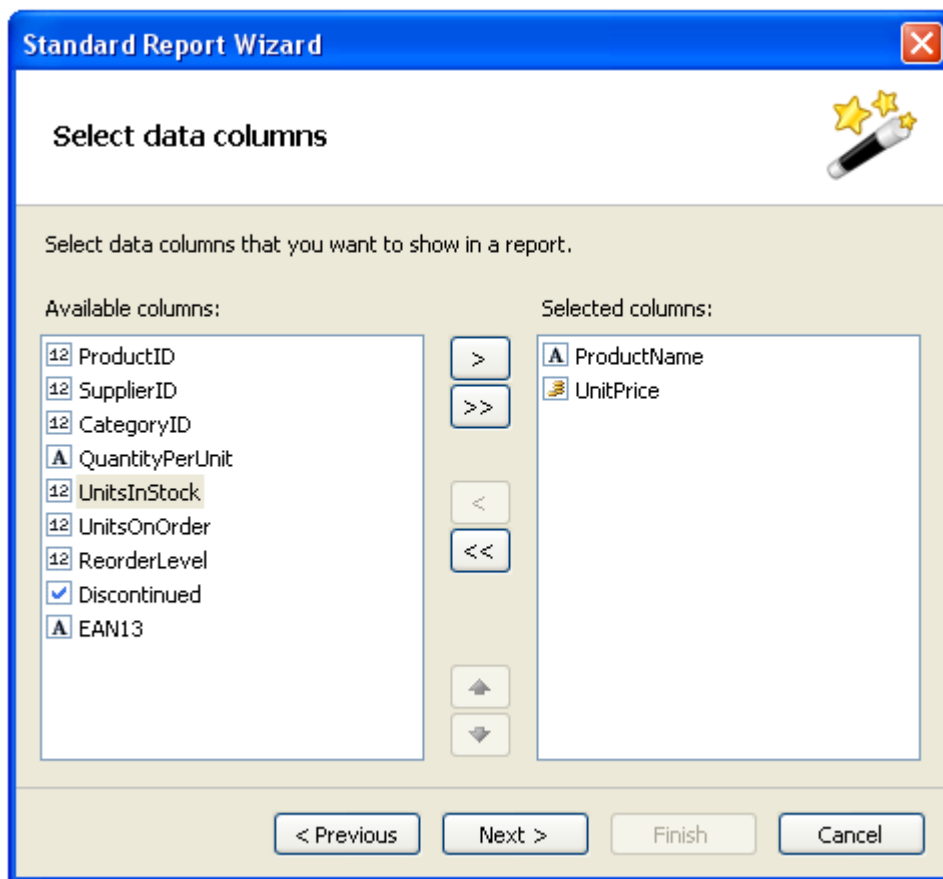
- press the  button on the toolbar and in the "Add New Item" window, choose "Standard Report Wizard":



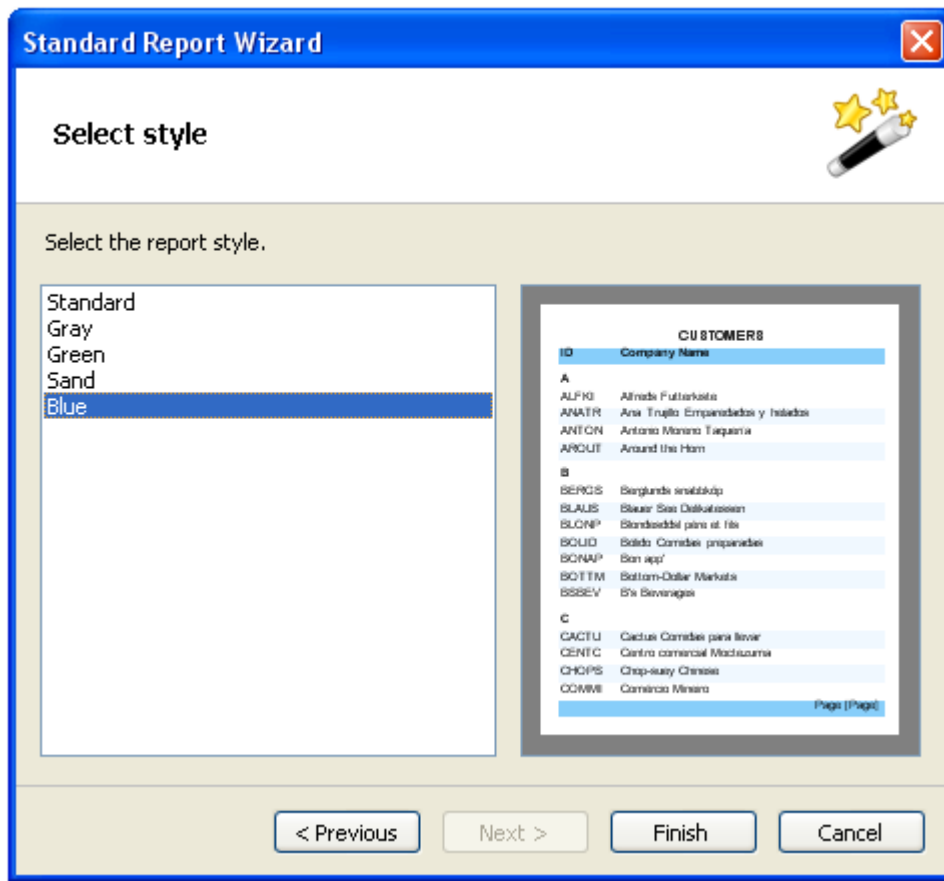
- on the first step of the wizard, choose the "Products" table and click the "Next" button:



- on the second step of the wizard, choose the "ProductName" and "UnitPrice" data columns:




- the rest of the steps can be skipped, click the "Next" button;
- on the last step of the wizard, choose "Blue" style and click the "Finish" button:



FastReport will create the following report:

Report Title	Products	
Page Header	ProductName	UnitPrice
Data: Products	[Products.ProductName]	[Products.UnitPrice]
Page Footer	Page [Page]	

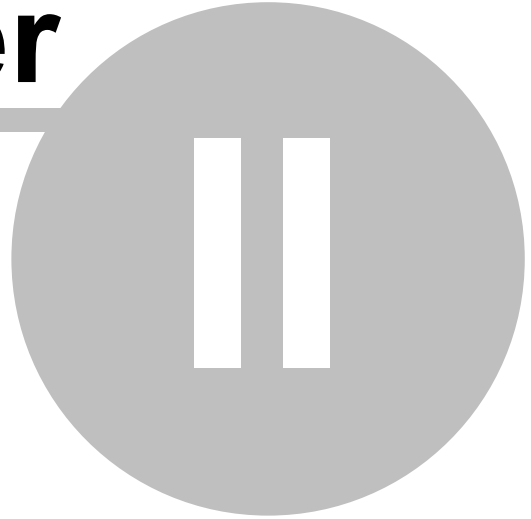
To run a report, click the  button on a toolbar. The report will be built and shown in the preview window:

Products	
ProductName	UnitPrice
Chai	18
Chang	19
Aniseed Syrup	10
Chef Anton's Cajun Seasoning	22
Chef Anton's Gumbo Mix	21,35
Grandma's Boysenberry Spread	25
Uncle Bob's Organic Dried Pears	30
Northwoods Cranberry Sauce	40
Mishi Kobe Niku	97
Ikura	31
Quang Cahrais	21



# Chapter

---



## Report creation

## Report creation

In this chapter we will look at the methods of creating common types of reports. In order to create any report, as a rule, you need to do the following:

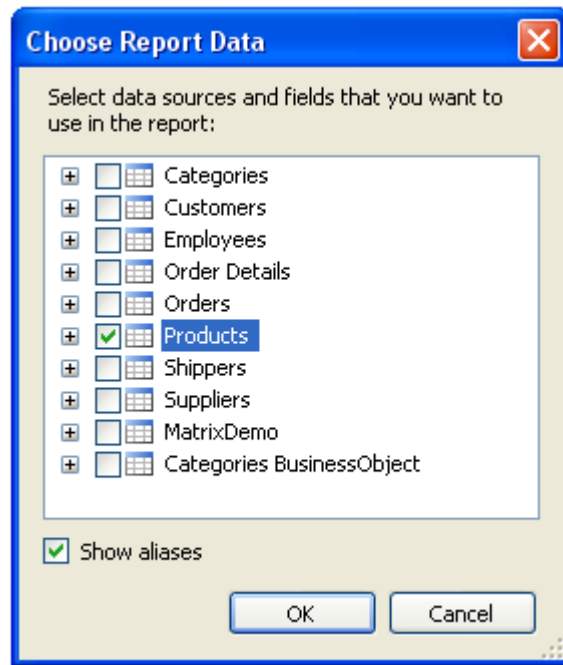
1. Choose or create data, which will be used in the report.
2. Create the report structure, by adding the needed bands into the report.
3. Connect the band to a data source.
4. Place the "Text" objects on the bands to print data.
5. Setup the appearance, formatting.



## Choosing data for a report

Before you start building a report, you need to choose the data which will be printed in the report. You can do this in two ways:

- you can choose one of the data sources, which was registered in the report by a programming method. This can be done in the "Data|Choose Report Data..." menu, by marking the needed data source:



- you create a new data source in the "Data|New Data Source..." menu.

Read more about data sources in the ["Data sources"](#) chapter.

Just after you have chosen the data source, it appears in the "Data" window. Now you can use this source in the report. Many reports use only one data source. For reports of "master-detail" type, you need to choose two data sources, related to each other (you can read more about relations in the ["Data sources"](#) chapter). Several data sources can also be needed in a report, which prints data from related sources.

## Dynamic layout

It is necessary often to print a text whose size is not known when creating a report. For example, this can be a description of goods. In this case, the following tasks will need to be solved:

- calculate the height of the object, such that it encloses the whole text;
- calculate the height of the band, such that it encloses the object with a variable amount of texts;
- move or change the height of other objects, which are contained on the band, such that, they do not disturb the general design of the report.

These tasks can be solved by using some object and band properties:

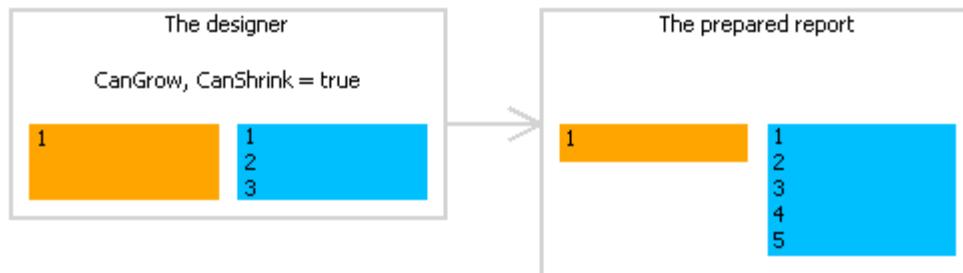
- "CanGrow" and "CanShrink" properties allow calculating the height of the object automatically;
- "ShiftMode" property allows moving objects that are located under the objects that expand;
- "GrowToBottom" property allows resizing an object to the bottom edge of the band;
- "Anchor" and "Dock" properties allow controlling the size of objects depending on the size of the band.

All these properties will be looked at below.

## CanGrow, CanShrink properties

Every band and report object has these properties. They determine whether an object can grow or shrink depending on the size of its contents. If both properties are disabled, the object always has the size specified in the designer.

These properties are very useful, if it is needed to print a text whose size is not known when designing. In order for an object to accommodate the entire text, it needs to have the "CanGrow" and "CanShrink" properties enabled:

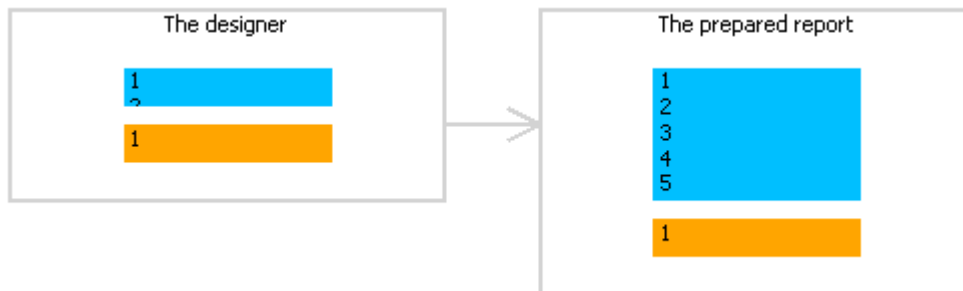


The following objects can affect the height of a band:

- "Text";
- "Rich Text";
- "Picture" (with "AutoSize" property enabled);
- "Table".

## ShiftMode property

Every report object has this property. This property is accessible only in the "Properties" window. An object, whose "ShiftMode" property is enabled, will be moving up and down, if the object above on can either grow or shrink.



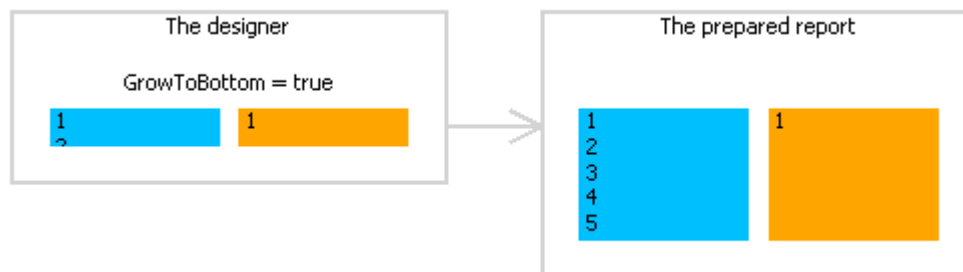
The "ShiftMode" property can have one of the following values:

- Always (by default). Meaning that the object needs to shift always.
- Never. Implies that the object does not need to shift.
- WhenOverlapped. Implies that the object needs to shift in that case, if the expanding object is located exactly over it (that is, both objects overlap horizontally).

This property is convenient to use when printing information in a table form, when several cells of the table are located on top of each other and can have a variable amount of text.

## GrowToBottom property

Every report object has this property. When printing an object with this property, it stretches up to the bottom edge of a band:



This is needed when printing information in a table form. In a table row there can be several objects which can stretch. This property makes it possible to set all objects' height to the maximum height of the band.

## Anchor property

Every report object has this property. It determines how the object will be changing its position and/or its size when the container on which it is laying will be changing its size. By using Anchor, it can be done in such a way that, the object expands or moves synchronously with its container.

The container, being referred to, in many cases will be the band. But this is not a must - this can also be the "Table" or "Matrix" objects.

The "Anchor" property can have one of the following values, and also any combination of them:

Value	Description
Left	Anchors the left edge of the object. When the container's size will be changing, the object will not be moving left/right.
Top	Anchors the top edge of the object .When the container's height will be changing, the object will not be moving up/down.
Right	Anchors the right edge of the object .When the container's width will be changing , the distance between the right edge of the object and the container will be constant. If the left edge of the container is anchored as well, then the object will be growing and shrinking synchronously with container.
Bottom	Anchors the bottom edge of the object. When the container's height will be changing, the distance between the bottom edge of the object and the container will be constant. If the top edge of the object is anchored as well, the object will be growing and shrinking synchronously with container.

By default, the value of this property is Left, Top. This means that, when the container's size will be changing, the object will not be changing. In the table below, combinations of some frequent used values are given:

Value	Description
Left, Top	Value by default. The object does not change when the size of the container changes.
Left, Bottom	The object moves up/down when the height of the container changes. The position of the object in relation to the bottom edge of the container does not change.
Left, Top, Bottom	When the height of the container is changing, the height of the object synchronously changes with it.
Left, Top, Right, Bottom	When the width and the height of the container are changing, the object grows or shrinks synchronously with it.

## Dock property

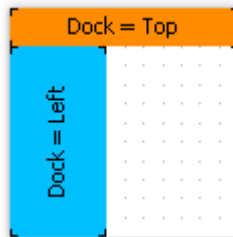
Every report object has this property. This property determines on which side of the container the object will be docked.

The "Dock" property can have one of the following values:

Value	Description
None	Value by default. The object is not docked.
Left	The object is docked to the left edge of the container. The height of the object will be equal to the height of the container*.

Top	The object is docked to the top edge of the container. The width of the object will be equal to the width of the container*.
Right	The object is docked to the right edge of the container. The height of the object will be equal to the height of the container*.
Bottom	The object is docked to the lower edge of the container. The width of the object will be equal to the width of the container*.
Fill	The object occupies all the free space of the container.

\* This is not quite so, if several objects have been docked at the same time. The figure below shows two objects, the first one has been docked to the top edge of the container and the second - to the left:



As seen, the height of the second object is equal to height of the free space, which remains after docking the first object.

The docking behavior depends on the object's creation order. You can change the order in the context menu of an object. To do this, select either the "Bring to front" or "Send to back" menu items.

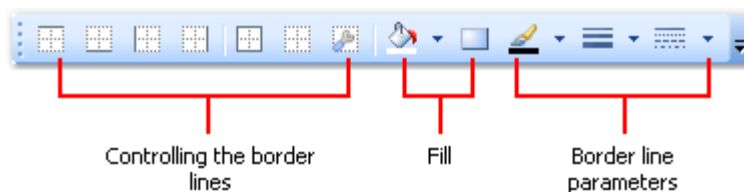
## Formatting

In this section, we will look at the following questions:


- changing an object's appearance;
- changing the format of printing values;
- automatically changing the appearance of an object when fulfilling some kind of condition;
- hide unnecessary values;
- highlighting even data rows in different colors.

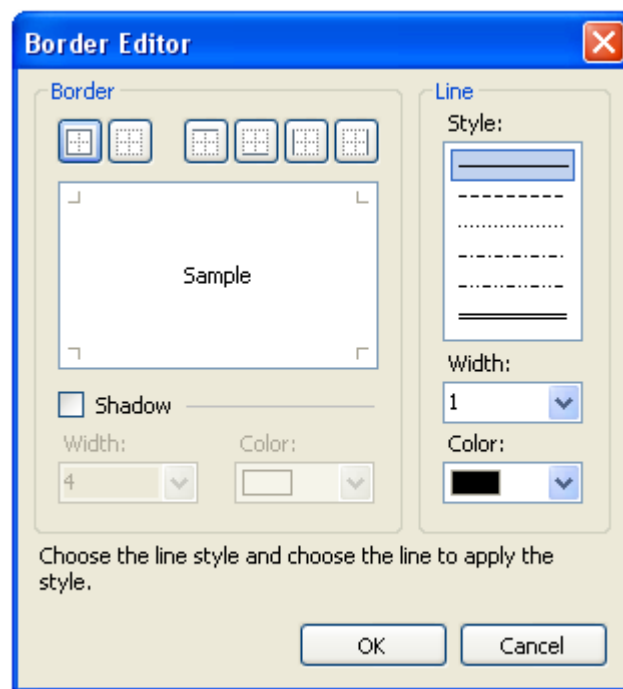
## Border and fill



Almost all report objects have the border and fill. To work with these properties, use the "Border and Fill" toolbar:

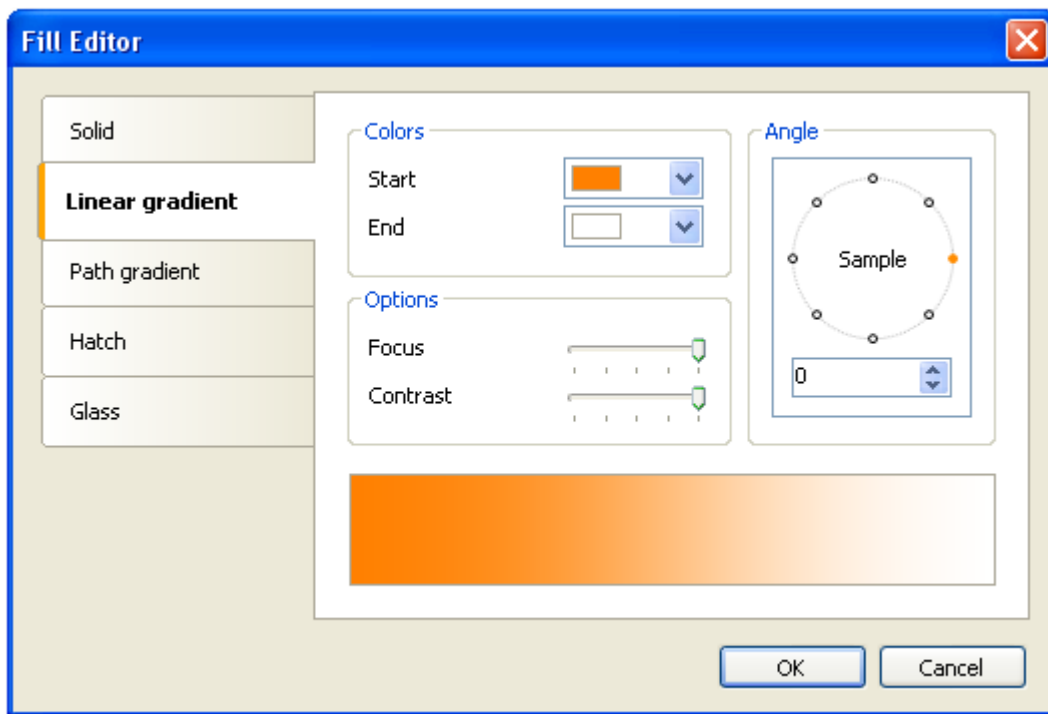


The object's border consists of four lines. Each line can have different width, color and style.

The toolbar buttons affect all lines of frame. The  button displays a dialog which allows to set up each line separately:

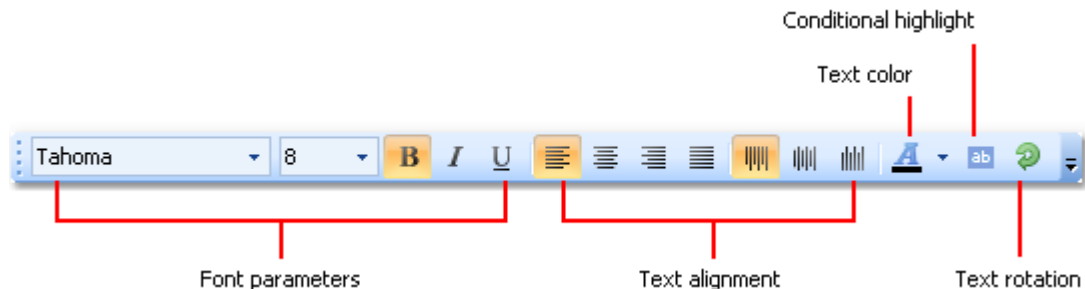


To work with fill, there are two buttons on the toolbar. The  button allows to choose a color for the solid fill type. The  button displays a dialog which allows to choose between different fill types:



## Text formatting

To change the "Text" object appearance, use the "Text" toolbar:

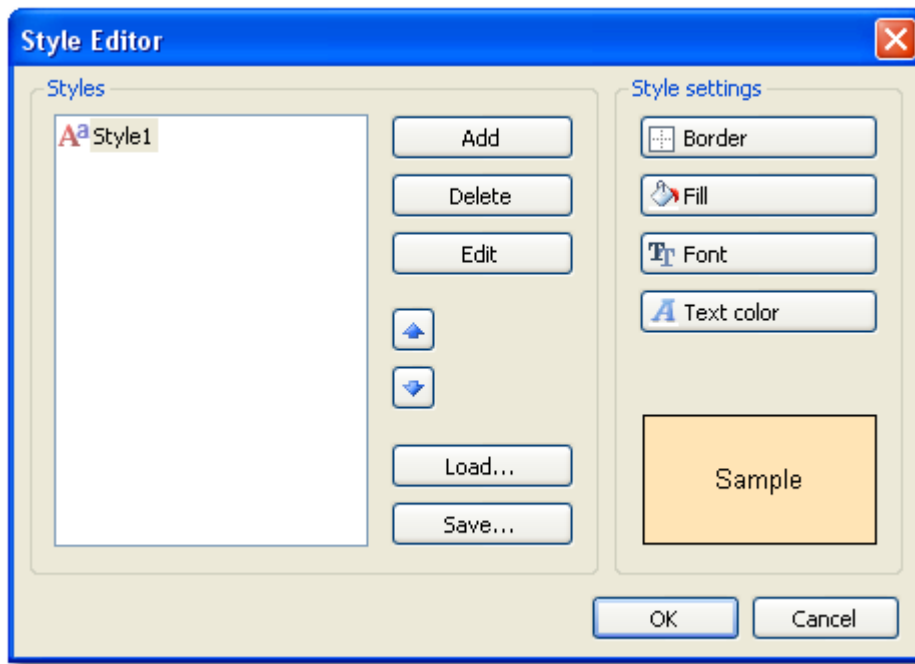


## Styles

To set up the object appearance, you may use styles. Style is a set of the following properties:

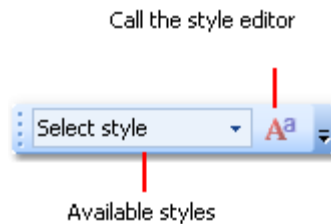
- border;
- fill;
- font;
- text color.

The list of styles is stored in a report. You can control it either from the "Report|Styles..." menu or by the **A<sup>a</sup>** button in the "Style" toolbar:



You can set an object's style in the following ways:

- set the "Style" property in the "Properties" window;
- use the "Style" toolbar:



If the toolbar is not present on the screen, enable it in the "View|Toolbars" menu.

When you set the object's style, the object's appearance will be changed according the style settings. When you change the style settings, the object with that style will change automatically.

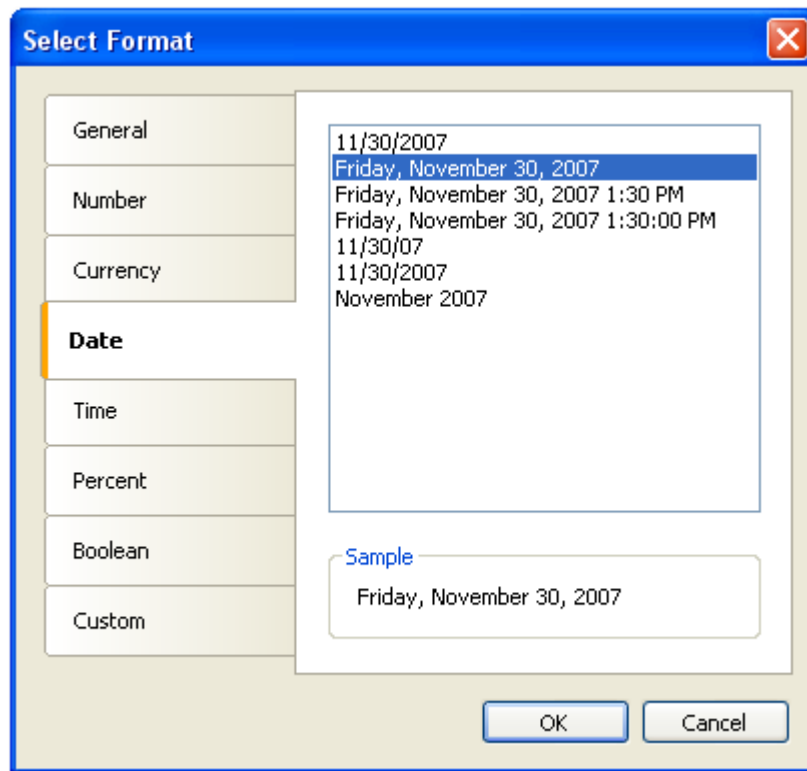
## Data formatting

To print a textual data in a report, the "Text" object is used. It applies the default formatting to all data that comes from data source. For example, the data source columns of type "DateTime" will be printed in the following way (it depends on your system's regional settings):

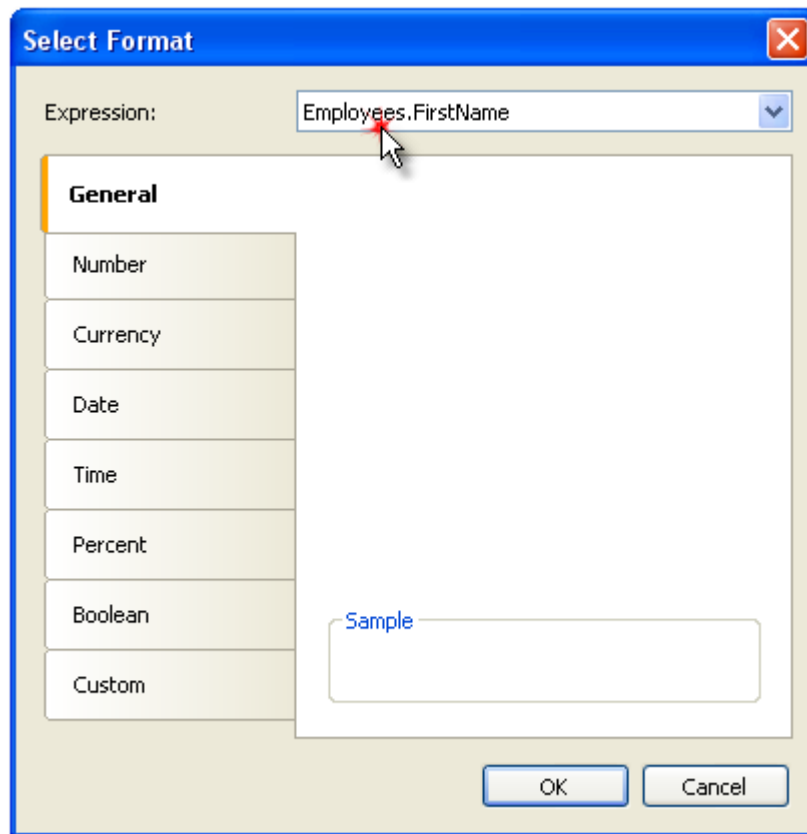
11.10.2008 18:04:52

If you need to print the date part only, you have to set up the data formatting. To do this, right-click the "Text" object to show its context menu. In the menu, choose the "Format..." item. You will see the format editor window:






You may choose one of the available formatting types or set up own formatting string. To do this, select the "Custom" formatting. If the "Text" object contains several data columns or expressions, you may choose appropriate format for each of them. To do this, select the expression in the top of the window, then choose the format:

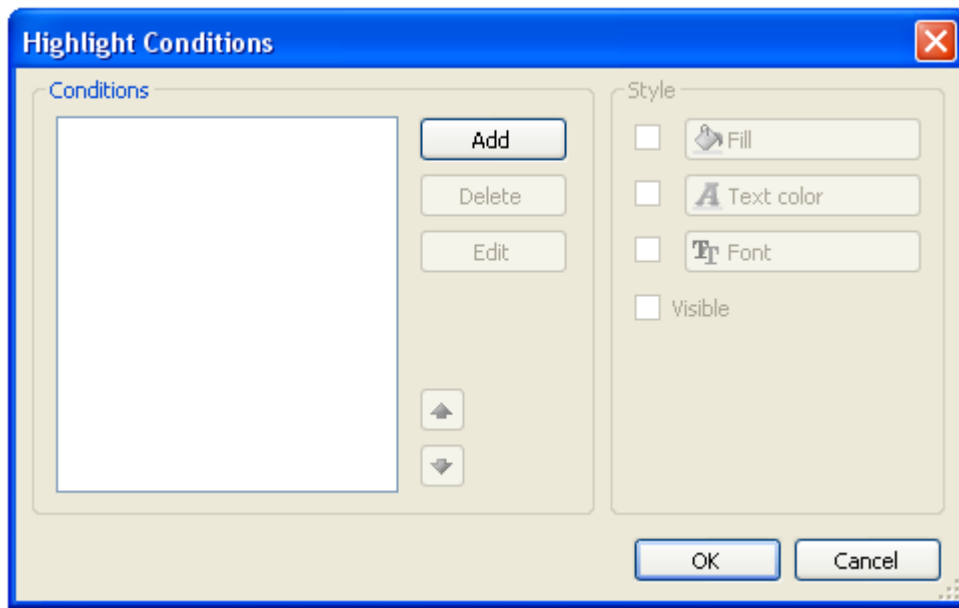


You may also format the data using the `String.Format` method. Get the help on this method in the MSDN.

```
Today is [String.Format("{0:d}", [Date])]
```

## Conditional highlighting

There is an possibility to change the "Text" object's appearance depending on the given conditions. For example, an object can be highlighted with red color if it has a negative value. This feature is called "conditional highlighting". To set up it, select the "Text" object and click the  button on the "Text" toolbar. You will see the following dialog window:

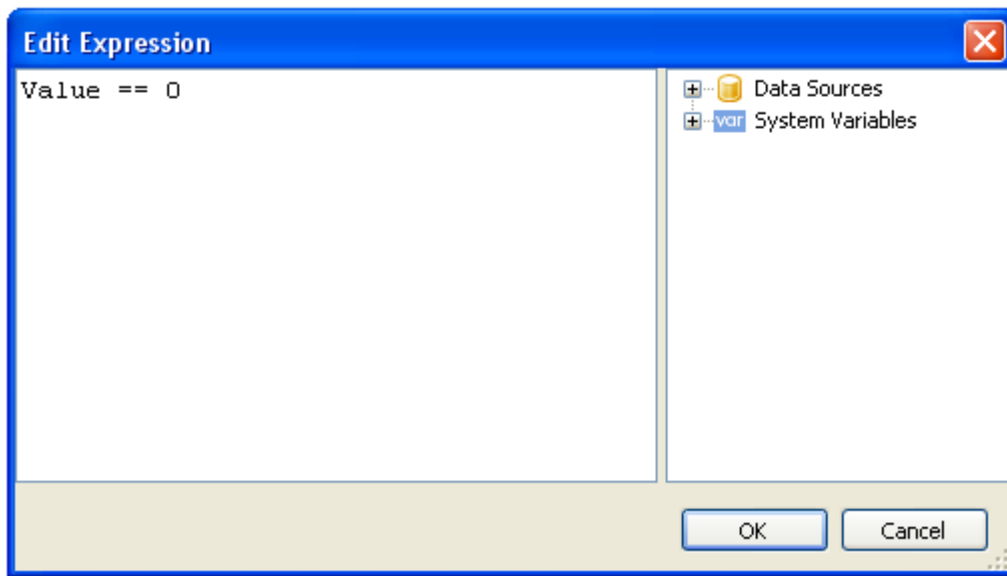


It is possible to define one or several conditions and set up the style for every condition. Style can contain one or several settings:

- fill;
- text color;
- font;
- object's visibility.

You can indicate, which settings need to be changed when the condition is met. For this, check the needed setting using the checkbox. By default, a new style contains one setting - the text color.

In order to create a new condition, click the "Add" button. You will see an expression editor:



Here, it is possible to write any expression which returns a boolean result. In many cases you will use the "Value" variable, which contains the currently printing value.

Let us look at the following example: we have a "Text" object, in which we print the amount of products in stock:

```
[Products.UnitsInStock]
```

We want to paint the object red, if the amount of products = 0. For this, we create the following condition:

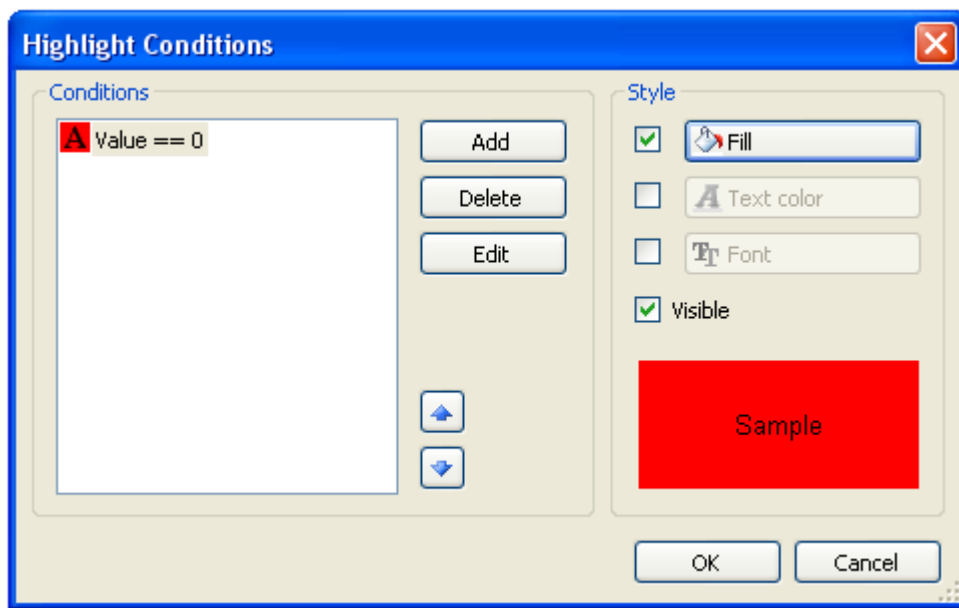
```
Value == 0
```

In the given case, we used the "Value" variable, which has got a printed value. If there are several expressions in an object, then this variable will have the value of the last expression. Instead of "Value", you can use a data column:

```
[Products.UnitsInStock] == 0
```

The expression is written in C# style. This is so, if the chosen report language is C#. For VisualBasic.Net you must use the single "=" sign. The report language can be changed in the "Report|Options..." menu.

Configure the style for the given condition in such a way that only fill can be used, and choose the red color:



When printing an object which has a zero value, it will be red. Let us make our example more complex, we will add another condition. If the units in stock is less than 10, it must be printed yellow. To do this, open the condition editor and click the "Add" button. The second condition will be like this:



*Value < 10*

In case where several conditions have been indicated, FastReport checks all the conditions, starting from the first one. If a certain condition is met, FastReport applies its style settings to the object, and the process stops. It is important to put the conditions in a correct order. The order which we have seen in this example is correct:

1. *Value == 0*
2. *Value < 10*

If we swap conditions, then the highlighting will work wrongly.

1. *Value < 10*
2. *Value == 0*

In the given case, the "*Value==0*" will not be executed, because when the value is zero, then the first condition will be met. In order to change the order of the conditions, use the  and  buttons.

## Hiding zero values

The "Text" object has the "HideZeros" property which can be used to hide zero values. Lets look at an object with the following contents:

*Total elements: [CountOfElements]*

If the value of variable CountOfElements is equal to 0, and the property HideZeros is set to true, the object will be printed as follows:

*Total elements:*

The "Text" object also has the "HideValue" property which can be used to hide the value of an expression which is equal to the given value. For example, if the property value is "0", then all the zero fields will be hidden. This property can also be used for hiding zero dates. As a rule, it's a date like "1.1.0001" or "1.1.1900". In this case the value of "HideValue" property must be like this:

*1.1.1900 0:00:00*

As you can see, apart from the date, you need to indicate time as well. This is necessary because the value of the date in .Net contains time also.

Important note: this mechanism depends on the regional settings of your system, which can be set in the control panel. This happens because FastReport compares strings using the "ToString()" method. This method converts an expression value into a string. In relation with this, be careful when building reports which can be launched on a computer with different regional settings.

Finally, the "NullValue" property of the "Text" object allows to print some text instead of a null value. It is often used to print the dash instead of a null value. Lets look at an object with the following contents:

*Total elements: [CountOfElements]*

If the value of variable CountOfElements is null, and the property NullValue is set to --, the object will be printed as follows:

*Total elements: --*

## Hide duplicate values

The "Text" object has "Duplicates" property which allows to control how duplicate values will be printed. This property can be used if the "Text" object is on the "Data" band. The values considered duplicate if they are printed in the near by data rows.

The "Duplicates" property can have one of the following values:

- Show - show the duplicates (by default).
- Hide - hide the object with duplicate value.
- Clear - clear the object's text, but show the object.
- Merge - merge several objects with the same value.

The difference between these modes is shown in the figure below:



The "Merge" mode does not work with "keep together" function (see ["Breaking data and keeping it together"](#) chapter).

### Highlight odd/even data rows

In order to improve the appearance of a report, you can highlight even data rows in different colors. This can be done by using the "EvenStyle" property of the band or its objects. The property contains a style name, which will be used to highlight even band rows.

It is preferable to use the "EvenStyle" property of the object instead of the band. This avoids possible problems when exporting the report.

In order to configure the highlighting, do the following:

1. Define the style, which will be used for highlighting the rows. This can be done in the "Report|Styles..." menu.
2. Indicate the name of the new style in the "EvenStyle" property of the band or its objects.

By default, objects use only a fill property of the style given in the "EvenStyle" property. This behavior is defined in the "EvenStylePriority" property - by default it is "UseFill". If you need to use the rest of the style parameters, set this property to "UseAll".

A ready report, which uses this technique, can look like this:

Product name	Unit price
Chai	18,00
Chang	19,00
Chartreuse verte	18,00
Côte de Blaye	263,50
Guaraná Fantástica	4,50
Ipoh Coffee	46,00

## Report with one "Data" band

This type of a report is more often required. It allows printing a list of rows from the data source. For example, this can be a customer list.

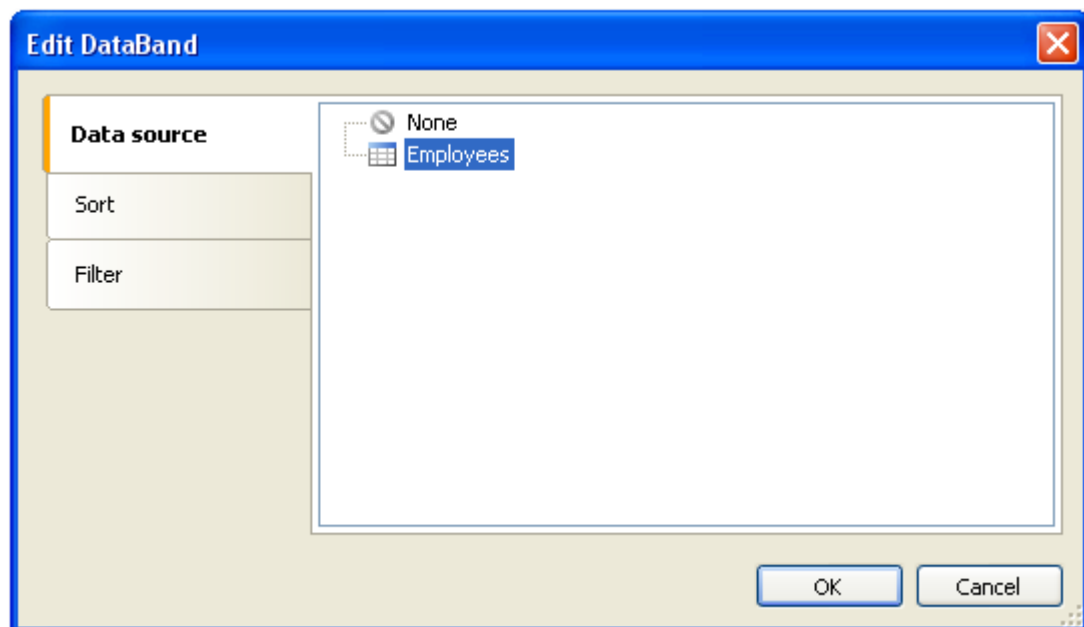
## Connecting a band to data source

To print a data from the data source, you will need the "Data" band, which is supposed to be connected to the data source. The band will be printed as many times as there are rows in the data source.

If the "Data" band is not connected to the source, it gets printed once.

When you create a new report, it already contains several empty bands, including the "Data" band. This band can also be added into the report from the "Configure Bands" window, by choosing the "Report|Configure Bands..." menu item.

In order to connect a band to data, double click it. Choose data source in the editor window and click "OK":



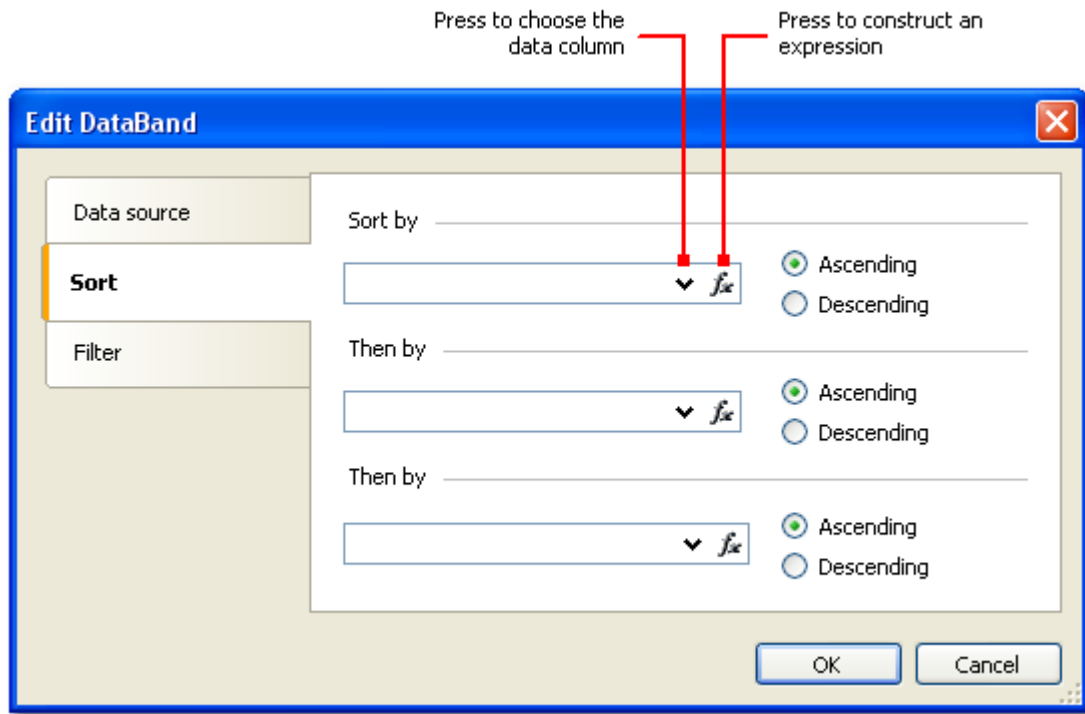
## Printing the text

After you have connected the band to a data source, you can place the "Text" object on the band, which will display the information from a data column. The fastest method to do this - drag a data column from the "Data" window and drop it on the band. Read more about the "Text" object in the ["The Text object"](#) chapter.

## Sorting the data

By default, the "Data" band prints data in natural order. Often it is needed to sort the data before printing. For example, a list of customers can be comfortably presented by sorting it in an alphabetical order.

You can control sorting in the "Data" band editor. In order to call the editor, double click on a free space at the band:



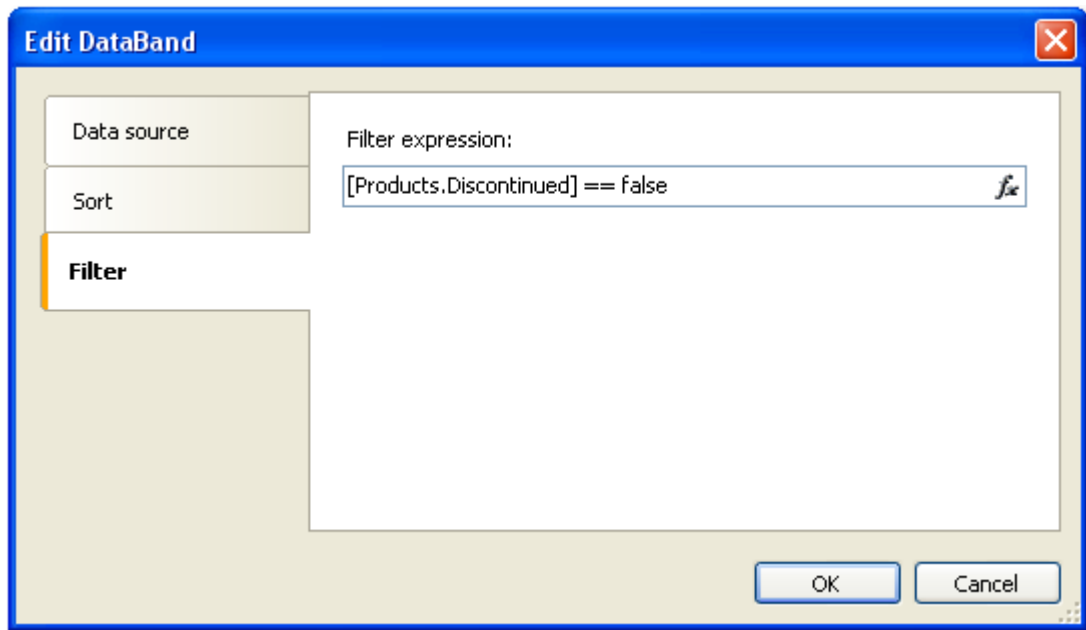
As a sort criteria, you can use either a data column or an expression. You can indicate several (not more than three) sorting conditions. This can be needed, for example, if you want to sort the list of customers by their cities, and after that, by customer's name. For each condition, you can choose the order of sorting - ascending or descending.

Another method of sorting data - use the SQL query as a data source. The query will be executed on the data server and return a sorted rows.



## Filtering the data

In order to filter a row, which is printed in the "Data" band, call its editor and switch to the "Filter" tab:



As a filter expression, you can indicate any correct expression. More details about expressions can be found in the ["Expressions"](#) chapter.

In the example above the following filter is used:

```
[Products.Discontinued] == false
```

This means that, all the data rows whose Disconnected flag is equal to false will be chosen.

You can use complex filter condition:

```
[Products.Discontinued] == false && [Products.UnitPrice] < 10
```

This means that, all the data rows whose Disconnected flag is equal to false, and whose price is less than 10 will be chosen.

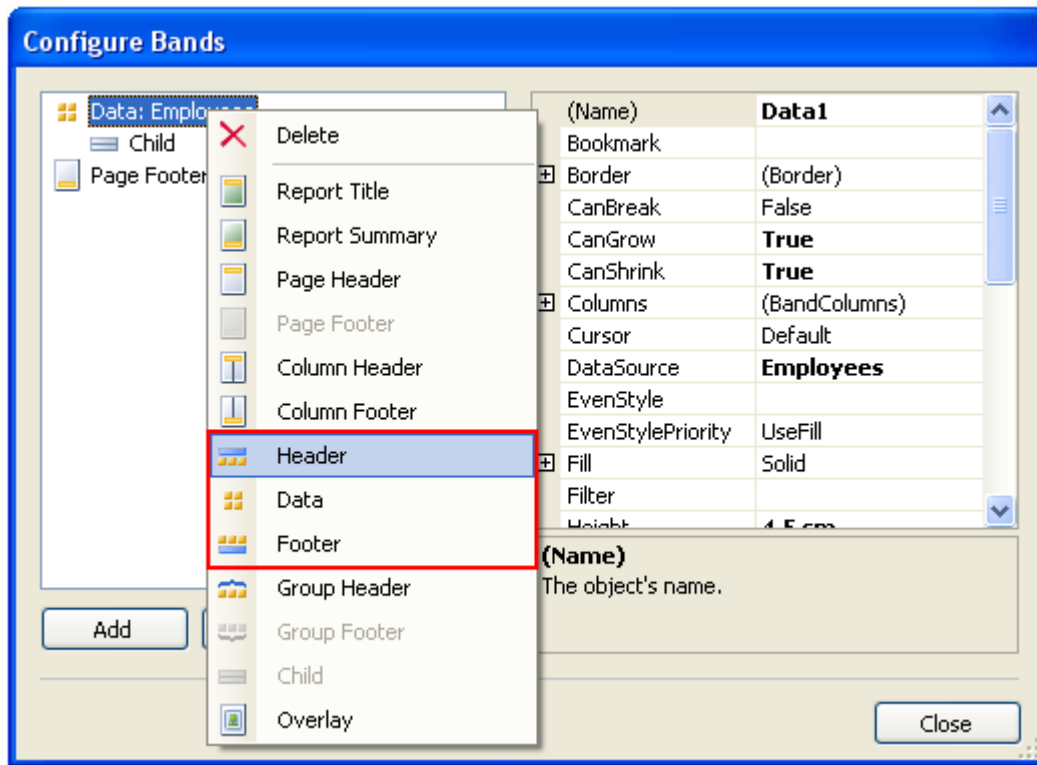
This filtration method supposes that, the data source contains all rows, part of which will be filtered. If the data source contains a large amount of rows, this can seriously slow down the report. In this case you can use SQL query as a data source, in which you can perform the needed filtration. The query will be executed on the data server and return only those rows which are needed in the report.

You also may use dialogue forms to perform data filtering. See more details in the ["Dialogue forms"](#) chapter.

## Data header and footer

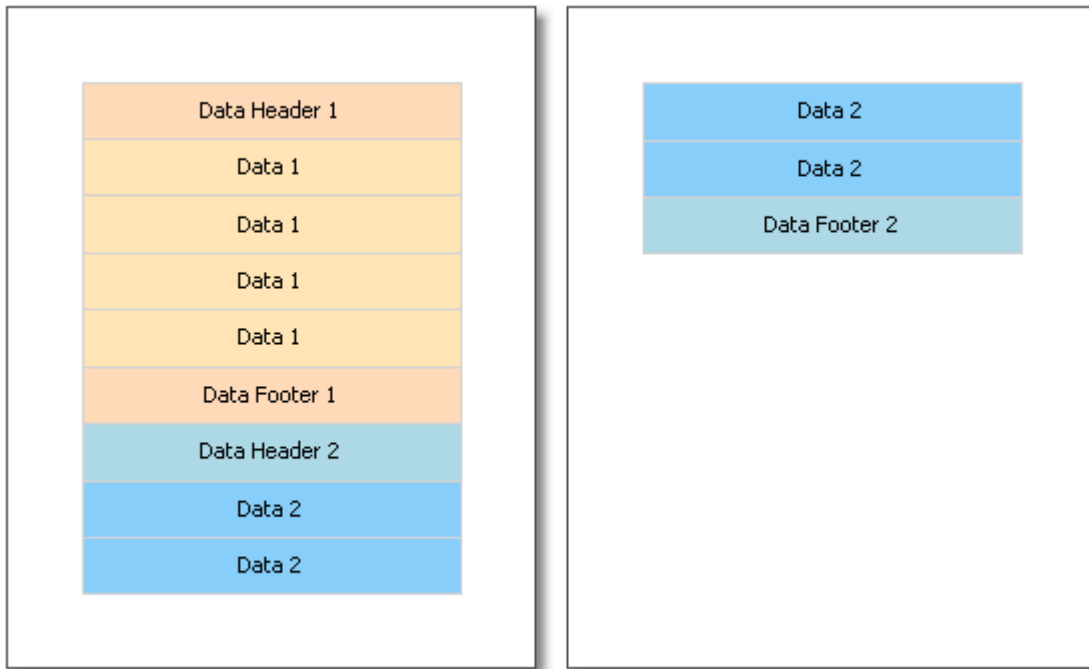
A "Data" band can contain a header and a footer. The header will be printed once before data, the footer will be printed after the output of all data.

In order to add a header and footer to a "Data" band, choose the "Report|Configure Bands..." menu item. In the window, select the "Data" band and right click the mouse. In the context menu choose the "Header" and/or "Footer" items:

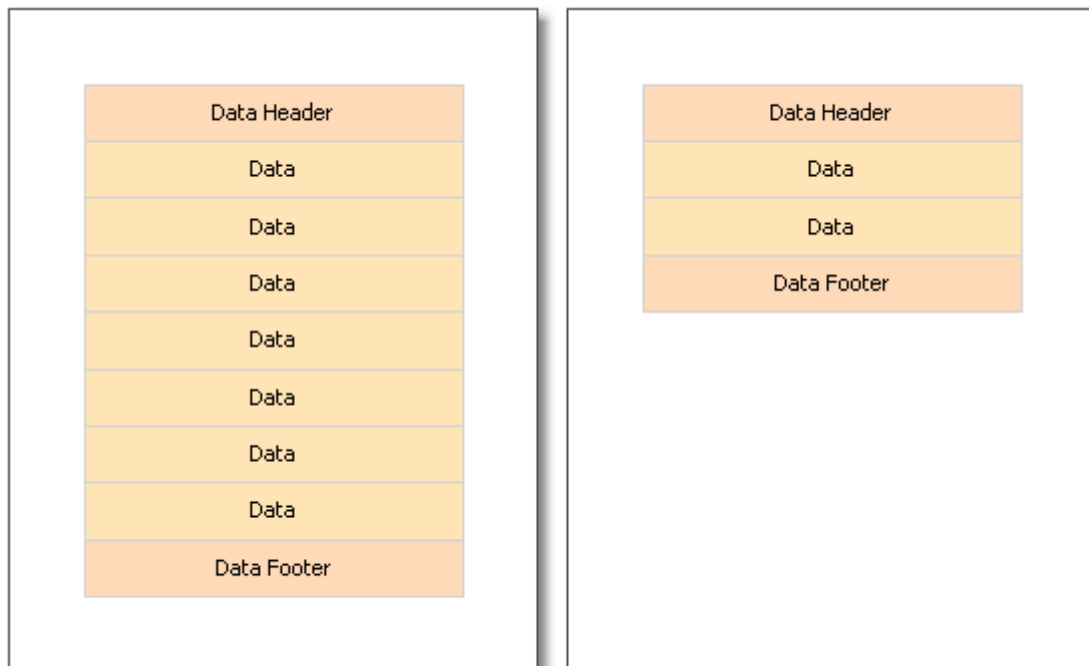


These bands can be useful in the following situations:

- when printing several lists on one page ("[master-master](#)" reports). Every data band in this case can have its own header and footer:



- when printing one list, if the list does not fit on one page of the prepared report. By using the "Repeat On Every Page" property of the data header/footer, you can print these bands on every page of the report:



### Breaking data and keeping it together

In this section, we will look at two modes of data printing - "Break" and "Keep together".

In an regular band printing mode, FastReport checks if there is enough space on the current page to print a band. If there is not, the band is printed wholly on the next page. If the "CanBreak" property of the band is enabled, FastReport will try to print the part of the band

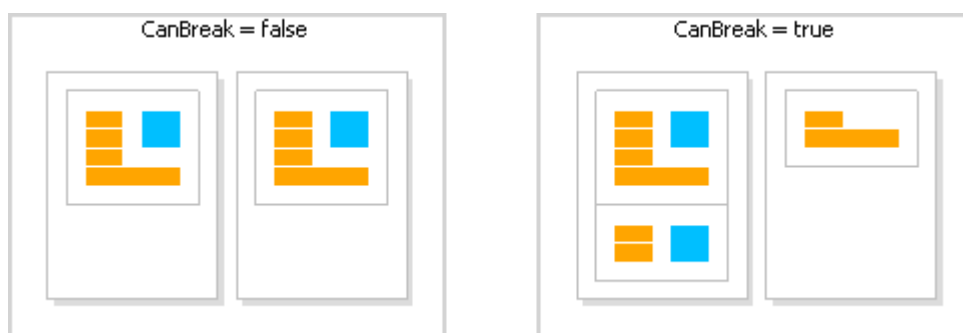
on the available space, that is, "break" it.

An attempt to break a band can be either successful or not. It depends on the type of object which has been placed on the band, and its settings. The following objects can be broken:

- "Text"
- "Rich Text"
- "Table"

These objects have the "CanBreak" property as well. If it is enabled, then the object can be broken. Non-breakable objects are always displayed wholly, there, where they have enough place.

In the figure below, how a band can be broken is shown.



Break algorithm does not always work correctly. The artifacts can occur in a situation, when there are several objects with different font size on a band.

The goal of band breaking is to save the space on the printed sheet. Data keeping's goal is contrary: display a set of bands wholly on one sheet. In this case there will be a lot of unused space on the sheets, but the data is printed in a way that it is comfortable to percept.

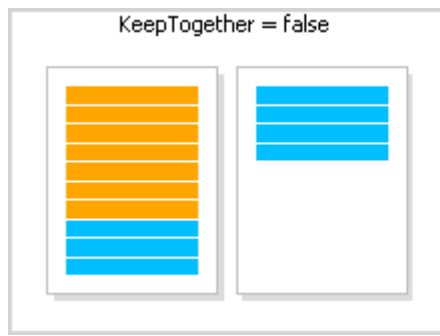
The "keep together" mechanism allows keeping a set of bands together on one page (or column, if the report has columns). If, when printing, kept data reaches the end of the page, FastReport relocates all data which has been printed already onto a new page.

You can use the "keep together" in the following cases:

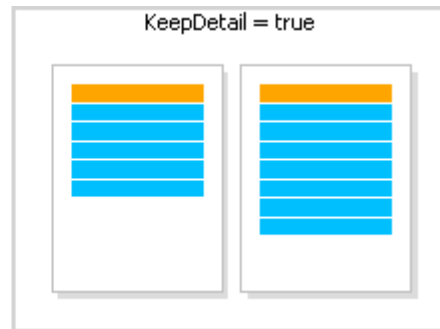
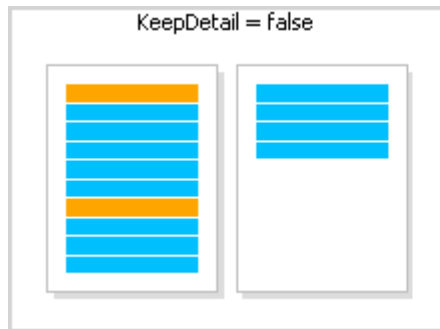
- printing all the rows of the "Data" band together;
- printing all the elements of a group (header, data, footer) together;
- printing the row of the master data source together with all detail rows (in the "master-detail" report);
- printing the report header or the data header together with at least one data row;
- printing the report footer or data footer together with at least one data row;
- printing the parent and child bands together.

Let us look at the use of "keep together" mechanism.

To keep together all data rows or group elements (header, data, footer), enable the "KeepTogether" property. This property is used in the "Data" and "Group Header" bands. The figure below shows how data is printed with and without keeping together:



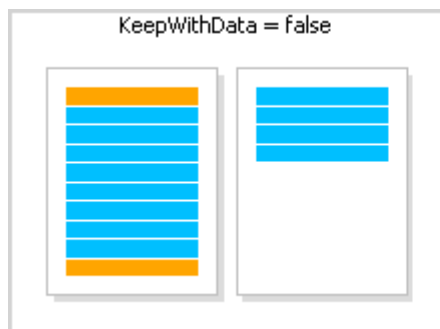
To keep master data row together with detail data rows, enable the "KeepDetail" property of the "Data" band. This property is used in a report of "master-detail" type:



To prevent "hanging" headers and footers, use the "KeepWithData" property. The following bands have got this property:

- report header;
- report footer;
- data header;
- data footer;
- group header;
- group footer.

This property allows to keep header/footer with at least one data row:

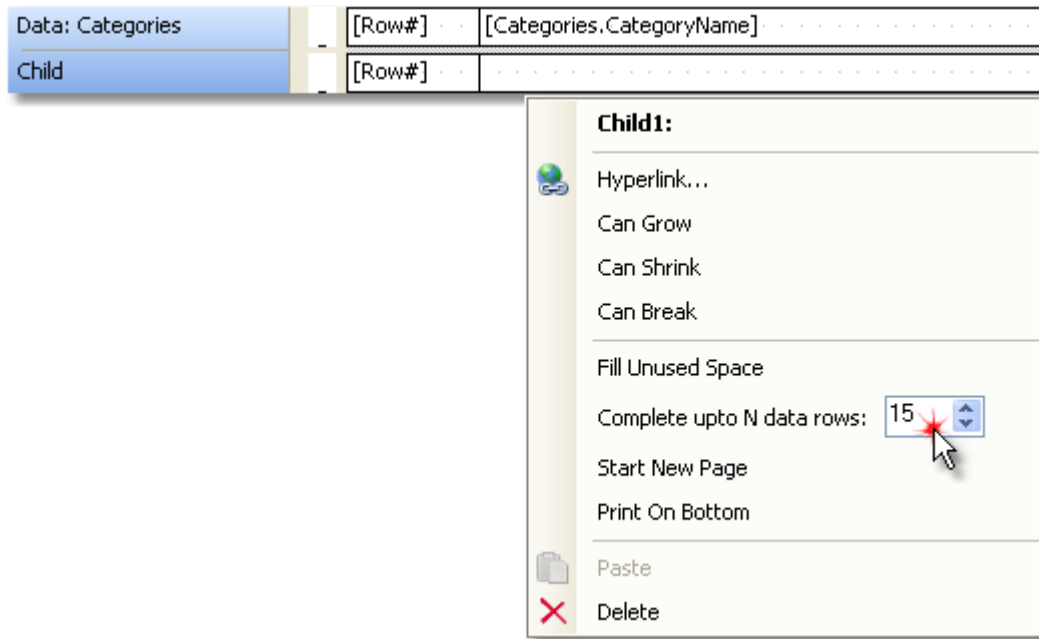


In order to keep a band and its child band together, enable the "KeepChild" property.

## Printing empty data rows

Often enough, when printing on pre-printed forms, a certain amount of data rows is supposed to be printed. If the actual data is less than needed amount of rows, then empty rows need to be printed. This can be done with the help of the "Child" band, by attaching it to the "Data" band.

The "Child" band has got a "CompleteToNRows" property. If this property is set to value greater than 0, the band will be used for additional data rows up to the indicated amount. For example, we need to print 15 rows, but there are only 8 rows in the data source. In this case, the "Child" band will be printed 7 times.



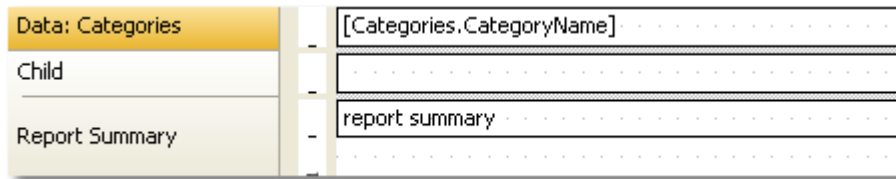
The prepared report will look like this:

1	Beverages
2	Condiments
3	Confections
4	Dairy Products
5	Grains/Cereals
6	Meat/Poultry
7	Produce
8	Seafood
9	
10	
11	
12	
13	
14	
15	

If the data source has more rows than indicated in the "CompleteToNRows" property, then an empty row will not be printed.

Another way to print an empty row is to fill the free space on a page. In this case, the "Child" band is attached to the bands of either the "Data Footer" or "Group Footer" types and fills the free space on the page. The footer band will be printed at the bottom of the page.

In order to print an empty row this way, attach the "Child" band to the footer band and enable its "FillUnusedSpace" property. You will see that the child band is now displayed above the band it is attached to. In the figure below, the "Child" band is attached to the "Report Summary" band:



When we run such a report, we will see the following:



### Printing "No data" text

When the data band is connected to an empty data source, it will not be printed. Sometimes it is required to print some text like "No data" instead of just an empty page. To do this:

- add a child band to the data band;
- set the child band's PrintIfDatabandEmpty property to true (it can be done in the "Properties" window);
- put the "Text" object on a child band and write the "No data to display" text in it.



The report will be printed in the following way:

- if the data source has some data rows, the data band will be printed, together with all its related bands (data header/data footer);
- if the data source is empty, only the child band with "No data to display" text will be printed.

## Printing hierarchy

A "Data" band allows printing a hierarchical list. For this, one band and one data source are used. Hierarchy must be defined in the data source with the help of two data columns:

1. Key column. This is the data row identifier.
2. Column which contains the key of this item's parent.

In order to print such a source in a hierarchical form, you need to set the following "Data" band properties. This can be done in the "Properties" window:

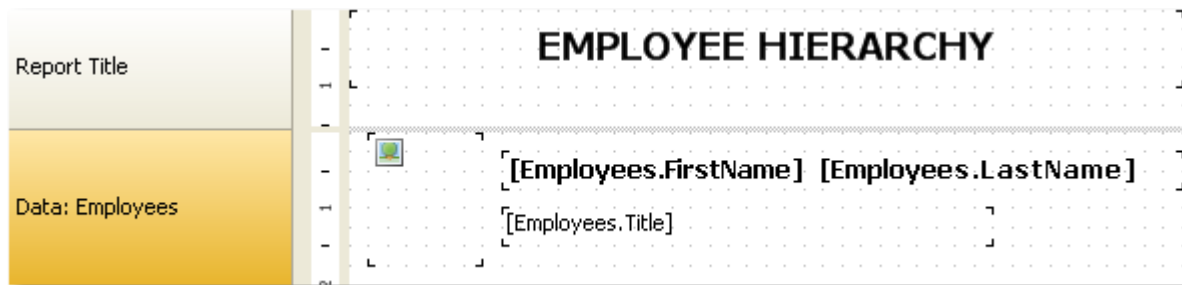
Hierarchy	
IdColumn	
Indent	1 cm
ParentIdColumn	

- indicate the key column in the "IdColumn" property;
- indicate the column containing parent value, in the "ParentIdColumn" property;
- indicate the hierarchy indent in the "Indent" property.

Let us look at an example of how to print a hierarchy of employees from the "Employees" demo table. The table has got two columns which we need:

- EmployeeID column is the key and contains the employee ID;
- ReportsTo column contains the ID of "parent" employee.

Create a report that looks like the following:

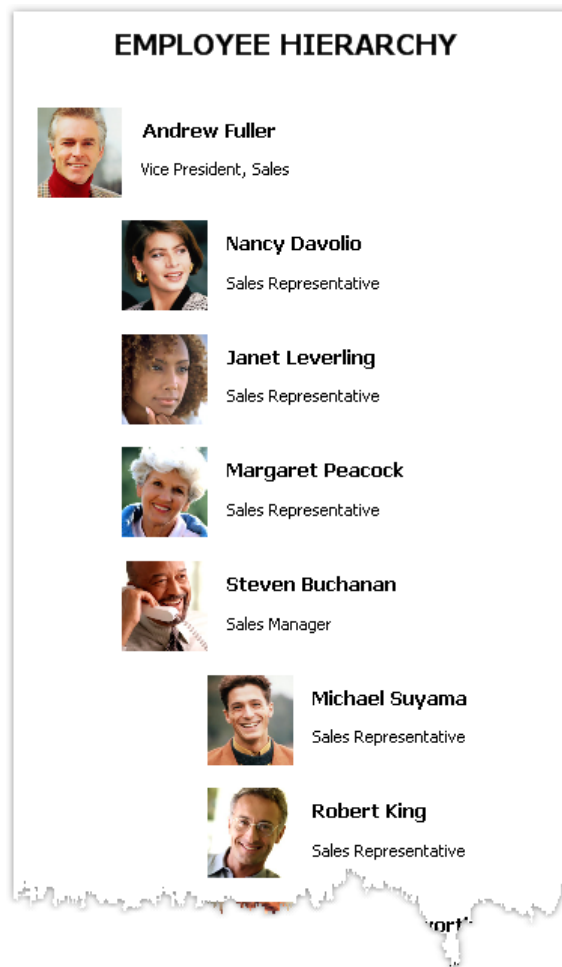


Set the "Data" band properties, which is responsible for the hierarchy, in the following way:

Hierarchy	
IdColumn	Employees.EmployeeID
Indent	1,5 cm
ParentIdColumn	Employees.ReportsTo

When we run a report, we will see the following:

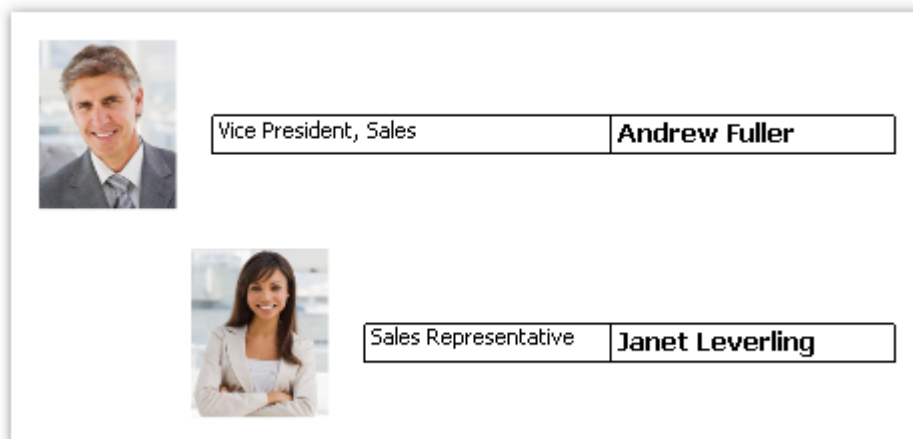




When printing hierarchy, FastReport shifts the band to the right (by a value indicated in the Indent property), and also decreases the band width by the same value. This allows you to use the Anchor property of band's objects. Here is possible values for this property that can be used in this case:

- Left, Top (by default) - the object is moved with the band;
- Right, Top - the object stays in its original position;
- Left, Right, Top - the right side of the object stays at its original position, the left side is moved with the band.

It allows you to get some useful effects:



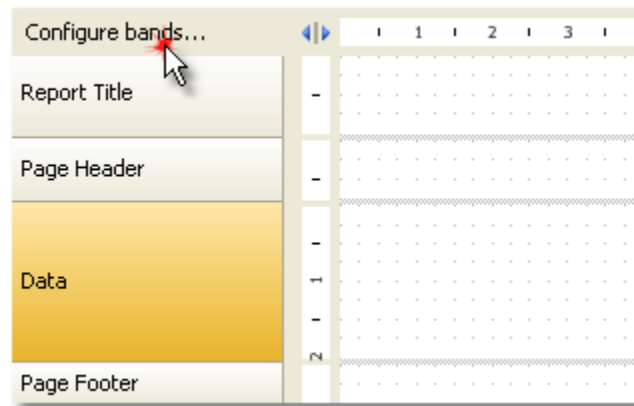
In this example, the picture object has Anchor property set to Left, Top; the object with job title is anchored to Left, Right, Top; the object with the name is anchored to Right, Top.

## Master-detail report

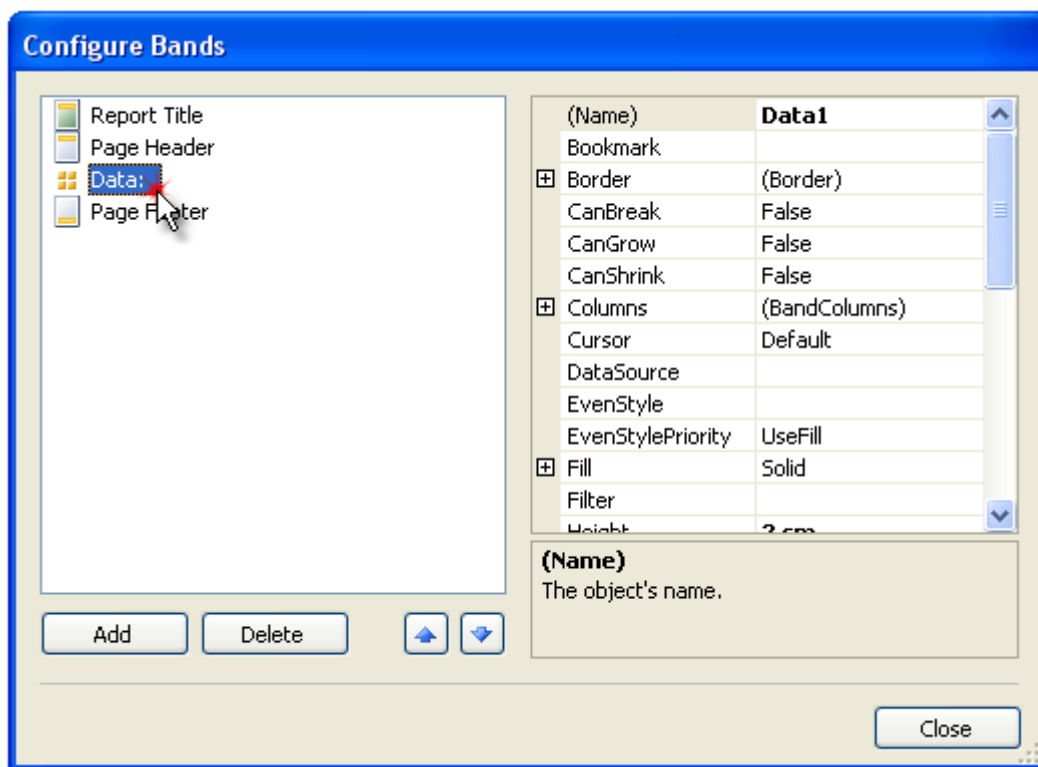
By using two "Data" bands, it is easy to create a report of the "master-detail" type. In this report, two data sources, between which there is a relation, are used. One row of the master source can correspond with several rows of the detail source. More details on relations can be found in the ["Data"](#) chapter.

It is necessary to place a band in a report in such a way that, the master band contains the detail band inside it. This can be done in the "Configure bands" window, which can be called in the "Report|Configure Bands..." menu.

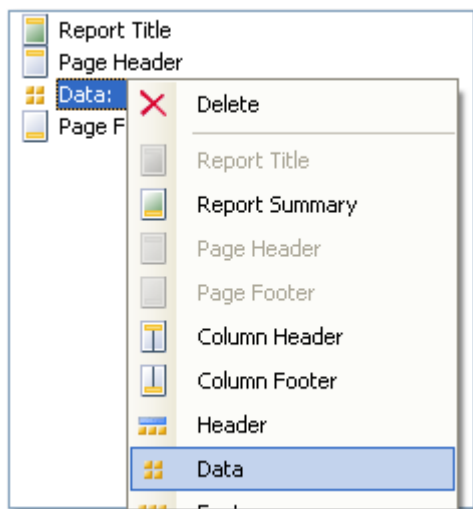
Let us look at the creation of a master-detail report from a scratch. For this, we will run the report designer and create a new empty report. It already contains one "Data" band:



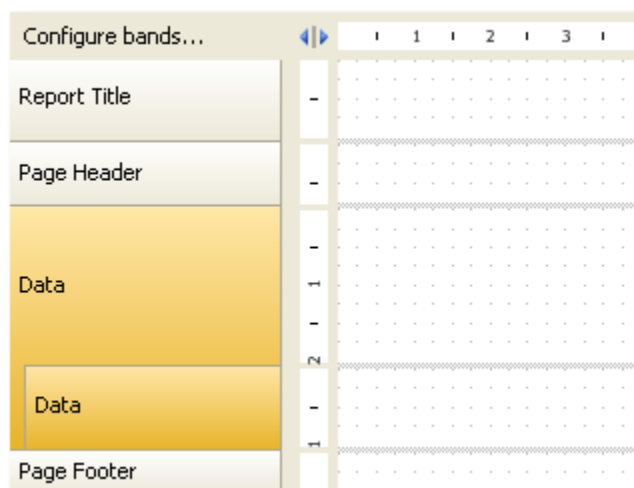
In order to add a detail data band, call the "Configure Bands" window. This can be done by pressing the "Configure bands..." button, shown in the figure, or by choosing the "Report|Configure Bands..." menu item. In the configurations window, the band structure is displayed:



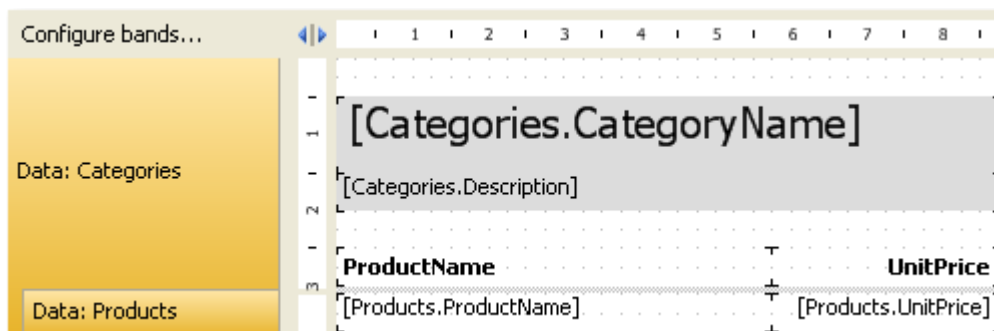
Select the "Data" band, as shown in the figure, and right click the mouse in order to show the context menu (or press the "Add" button in the lower part of the window). In the window which will open, select "Data" band:



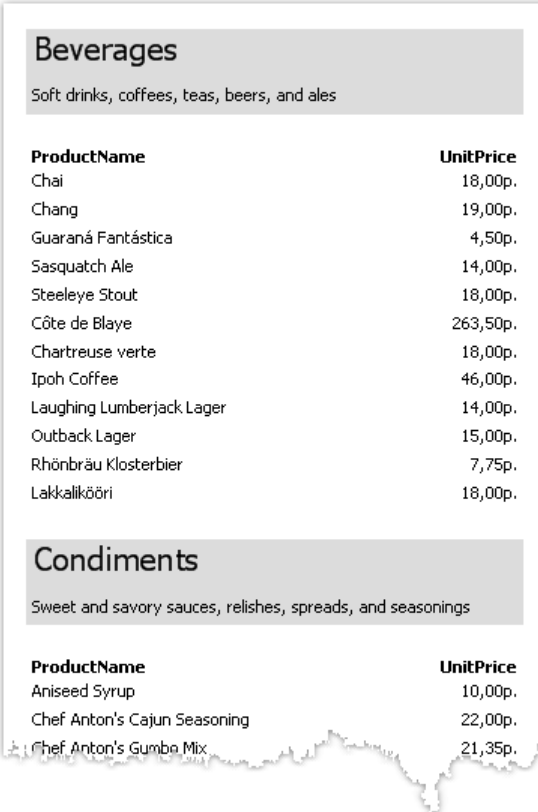
After this, a nested "Data" is added to the selected band. Close the window by pressing the "Close" button. You will see that the report template changes in the following way:



Nested data bands are clearly seen on band structures on the left part of the window. After this, you need to connect the band to the corresponding data source and place data columns on the bands. We will be using two data sources - Categories and Products - from the demo data base which comes with FastReport:



If we run the report, we will see the following:



<b>Beverages</b>	
Soft drinks, coffees, teas, beers, and ales	
ProductName	UnitPrice
Chai	18,00p.
Chang	19,00p.
Guaraná Fantástica	4,50p.
Sasquatch Ale	14,00p.
Steeleye Stout	18,00p.
Côte de Blaye	263,50p.
Chartreuse verte	18,00p.
Ipoh Coffee	46,00p.
Laughing Lumberjack Lager	14,00p.
Outback Lager	15,00p.
Rhönbräu Klosterbier	7,75p.
Lakkalikööri	18,00p.

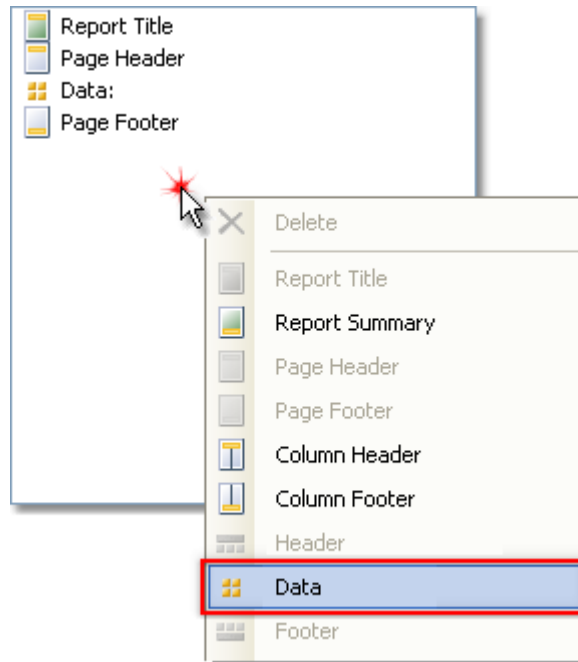
<b>Condiments</b>	
Sweet and savory sauces, relishes, spreads, and seasonings	
ProductName	UnitPrice
Aniseed Syrup	10,00p.
Chef Anton's Cajun Seasoning	22,00p.
Chef Anton's Gumbo Mix	21,35p.

In this way, you can create a master-detail report type with unlimited nested data, for example, master-detail-subdetail. Another method, which is used for the creation of master-detail report type, connected with the use of nested reports. Nested reports will be looked at in the ["Subreports"](#) section.

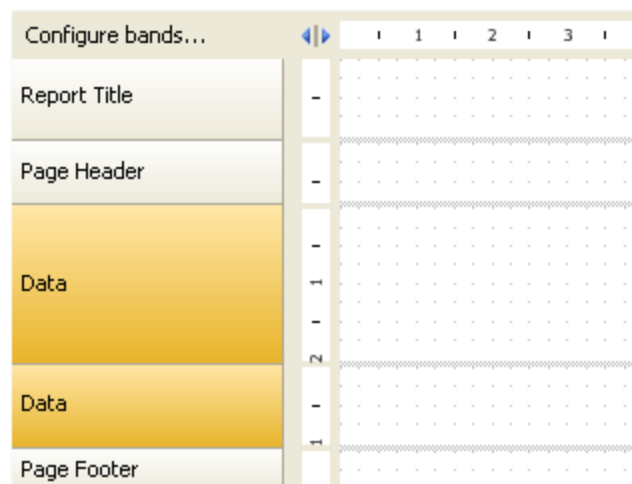
## Master-master report

On the report page, you can print several simple lists. This can be done, by placing on the page two or several "Data" bands. Contrary to master-detail report, where bands are nested into each other and prints data from related sources, in a report of this type both bands and data sources do not depend on each other.

We will show by an example, how to create a report which prints two lists on a page - Categories table and Customers table. We will create a new report and add into it needed data sources. In order to add a second "Data" band, call the "Configure Bands" window.



Right click on an empty place of the list, as shown in the figure, and select "Data" band in the context menu. This creates a new independent "Data" band. The report template will be like this:



Now, we will connect the band to the data source and place several data columns on it:

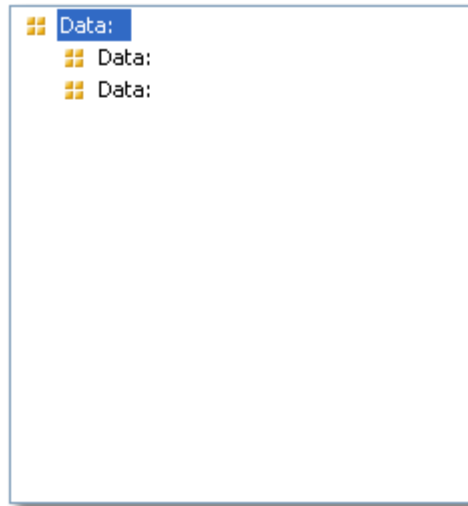
Data: Categories	[Categories.CategoryName]
Data: Customers	[Customers.CompanyName]

If we run the report, we will see the following:

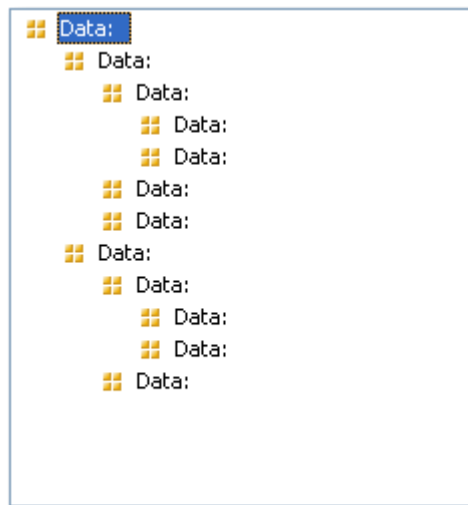
- Beverages
- Condiments
- Confections
- Dairy Products
- Grains/Cereals
- Meat/Poultry
- Produce
- Seafood
- Alfreds Futterkiste
- Ana Trujillo Emparedados y helados
- Antonio Moreno Taquería
- Around the Horn

## Master-detail-detail report

A "Data" band can contain one or several nested "Data" bands. This allows building a master-detail-detail report type. To do this, call the "Configure Bands" window, right click on the master "Data" band and add a detail "Data" band to it. Repeat this procedure in order to add the second detail band:



In this way, it is possible to add an unlimited number of detail bands to the master "Data" band. An example report structure can be like this (this is just an example; it only demonstrates the abilities of FastReport):





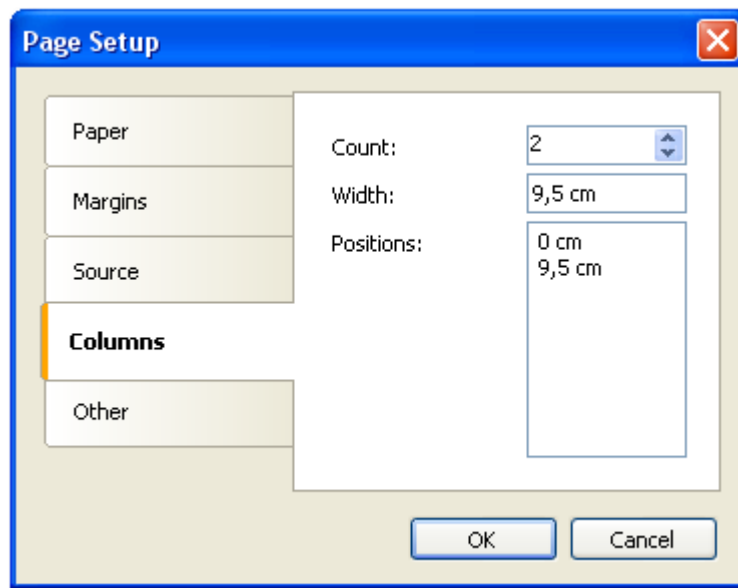
## Multicolumn reports

An ordinary report prints data as long as it has not reached the end of the page. After this, a new page is formed and printing continues on it. A report with columns prints data in several columns. When the end of the page has been reached, printing continues in a new column on the same page. In this sense, an ordinary report can be seen as a report with one column.

In FastReport there are two methods of printing columns.

## Page columns

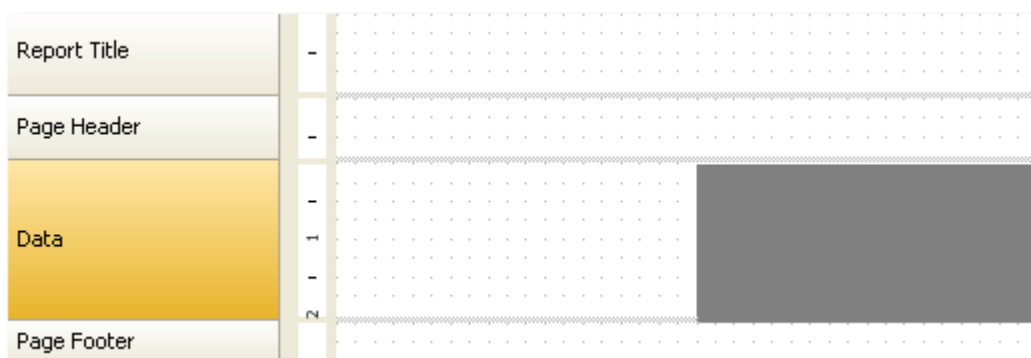
The first method is based on setting the number of columns of the report page. This is done in the "Page Setup" window on the "Columns" tab:



As seen, you can set the following column parameters:

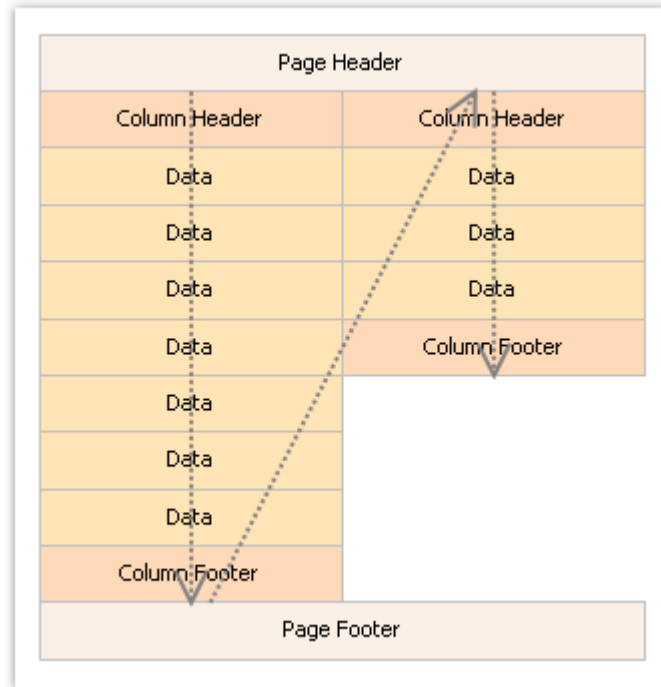
- column count;
- column width;
- the position of every column.

In order to transform an ordinary report into a report with columns, you need to set only the number of columns on the page. The rest of the parameters FastReport will calculate on its own. When you enable columns, the mode of bands in the designer changes:



The area shown in grey should never be used for placing objects on it. It is used to print next columns' objects.

For working with columns, the "Column Header" and "Column Footer" bands are used. As seen from their names, they print at the top and the bottom of every column respectively. The following figure demonstrates the bands printing order in the report with columns:



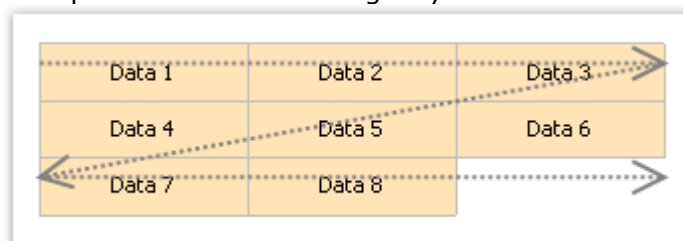
### Data band columns

Another method of printing a multicolumn report is based on using the "Data" band columns. The rest of the bands continue to be printed in one column.

Column parameters can be configured using the "Columns" property, which can be changed in the "Properties" window. You can set up the following parameters:

- number of columns;
- column width;
- column printing mode. You can select either of the two modes - "AcrossThenDown" and "DownThenAcross";
- the minimum number of rows in one column, if the chosen mode is "DownThenAcross".

Column band can be printed in either of the two modes. In the "AcrossThenDown" mode (the default one), columns are printed in the following way:



In the "DownThenAcross" mode, column printing occurs in the following way:

The diagram shows a 3x3 grid of data cells. The cells are labeled as follows:

Data 1	Data 4	Data 7
Data 2	Data 5	Data 8
Data 3	Data 6	

Arrows indicate the filling order: a vertical arrow on the left points down from Data 1 to Data 3; a vertical arrow on the right points down from Data 7 to Data 8; a diagonal arrow points from Data 1 to Data 6; and a vertical arrow on the right points down from Data 8 to an empty space below it.


In this mode, FastReport calculates the number of data rows in a column in such a way that, columns are filled equally. You can also set the minimum number of rows in a column with the "Columns.MinRowCount" property.

## "Booklet"-type reports

When printing a report in form of a booklet, you will probably face with the following demands:

- separate report page - cover, table of contents, report contents, back cover;
- different page margins for even and odd pages;
- different header and footer on even and odd pages.

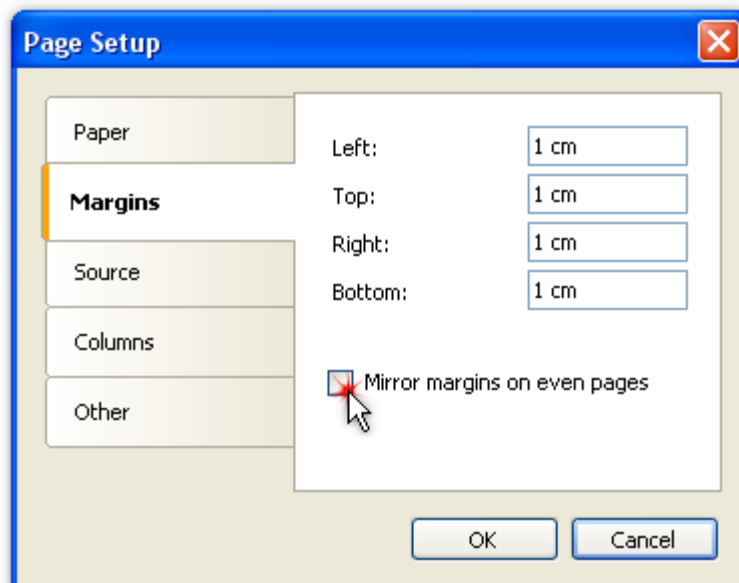
## Adding page into a report

You can add any number of pages into the report template. On each page you can place a separate report. To add a new page, click the  button on the toolbar. A page can also be added, by pressing the "Add New..." button and selecting the "Report page" item in the window.

For creating the "Table of Contents" section, you can use the technique described in the ["Interactive reports"](#) chapter.

## Page settings

In the "Page Setup" window, you can indicate that FastReport should mirror the left and the right margin for even pages:



If you need to start the page on an odd page number, set the "StartOnOddPage" property to true. When needed, FastReport prints the empty page before starting to print the indicated page.

## Printing on odd/even pages

All report objects have the "PrintOn" property. Using this property, you can print different objects on odd and even pages.

This property can be set in the "Properties" window.

This property determines on which pages the object can be printed. The property can have one of the following values or any combination of it:

- FirstPage;
- LastPage;
- OddPages;
- EvenPages;
- RepeatedBand. This value refers to a band with the "RepeatOnEveryPage" property set to true.

By default, the value of this property equals to "FirstPage, LastPage, OddPages, EvenPages, RepeatedBand". This means that the object will be printed on all pages of the report. We will give several typical examples of using this property:

Property value	Where the object will be printed
FirstPage	Only on the first page.
LastPage, OddPages, EvenPages, RepeatedBand	On all pages except the first.
FirstPage, OddPages, EvenPages, RepeatedBand	On all pages except the last.
RepeatedBand	Only on bands with the "RepeatOnEveryPage" property is set to true.
FirstPage, LastPage, OddPages, EvenPages	On all bands except the repeated one.
FirstPage, LastPage, OddPages, RepeatedBand	Only on odd pages.
FirstPage, LastPage, EvenPages, RepeatedBand	Only on even pages.

For example, to print different text on odd and even pages, put two "Text" objects on a band and setup them in the following way:

- the first object will be printed on odd pages. Set its "PrintOn" property to "FirstPage, LastPage, OddPages, RepeatedBand" (i.e. all values except "EvenPages").
- the second object will be printed on even pages. Set its "PrintOn" property to "FirstPage, LastPage, EvenPages, RepeatedBand" (i.e. all values except "OddPages").

These objects will never be printed at the same time. You could place them on top of each other.

All bands have the same property. To print different bands on odd and even pages, use the "Child" band. You can attach it to any band; this can be done in the "Configure Bands" window. Setup the main band and its child in the following way:

- the main band will be printed on odd pages. Set its "PrintOn" property to "FirstPage, LastPage, OddPages, RepeatedBand" (i.e. all values except "EvenPages").
- the child band will be printed on even pages. Set its "PrintOn" property to "FirstPage, LastPage, EvenPages, RepeatedBand" (i.e. all values except "OddPages").

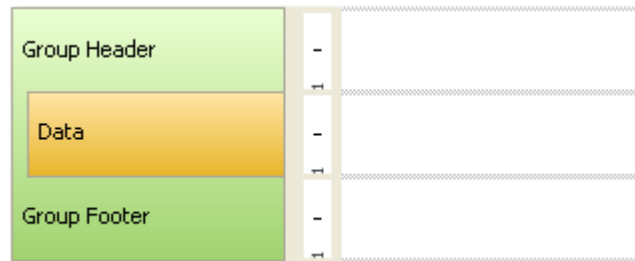
Bands can have different height, appearance and contents. Look at the following example which prints different page headers:

Page Header	-	PrintOn = FirstPage, LastPage, OddPages	[PageN]
Child	-	PrintOn = FirstPage, LastPage, EvenPages	[PageN]

## Groups and totals

Earlier we looked at the "Master-detail" report type, which printed data from two related sources. FastReport allows creating a report which looks in the same way, but uses one data source. For this, groups are used.

A group is a set of three bands: "Group header", "Data" and "Group footer". In designer, this looks as follows:



A group always contains a header and data. Group footer is optional, you can delete it.

In order to use a group, you should set the group condition for the group header, and connect the data source to the "Data" band. The condition can be any expression, but as a rule, this is one of the data source columns. Group printing is done in the following way:

1. group header is printed.
2. data row is printed.
3. checks if the grouping condition has changed.
4. if the condition has not changed, next data row is printed (p.2).
5. if the condition has changed, the group footer is printed, and starts printing a new group (p.1).

Assuming that we have a Products table with the following data:

CategoryName	ProductName
Beverages	Côte de Blaye
Beverages	Chartreuse verte
Beverages	Steeleye Stout
Beverages	Guaraná Fantástica
Beverages	Sasquatch Ale
Beverages	Rhönbräu Klosterbier
Beverages	Lakkalikööri
Beverages	Outback Lager
Beverages	Ipoh Coffee
Beverages	Laughing Lumberjack Lager
Beverages	Chang
Beverages	Chai
Condiments	Original Frankfurter grüne Soße
Condiments	Sirop d'érable
Condiments	Chef Anton's Gumbo Mix
Condiments	Northwoods Cranberry Sauce
Condiments	Grandma's Boysenberry Spread
Condiments	Chef Anton's Cajun Seasoning
Condiments	Aniseed Syrup
Condiments	Louisiana Hot Spiced Okra
Condiments	Vegie-spread
Condiments	Louisiana Fiery Hot Pepper Sauce
Condiments	Gula Malacca
Condiments	Genen Shouyu

Data can be grouped on the CategoryName column. This column will be printed in the group header. The data itself is presented by the ProductName field. The report will be as follows:

Group Header:	-	<b>[[Products.Categories.Category Name]]</b>
CategoryName	-	
Data: Products	-	[Products.ProductName]
Group Footer	-	

If we run the report, the following will be seen:



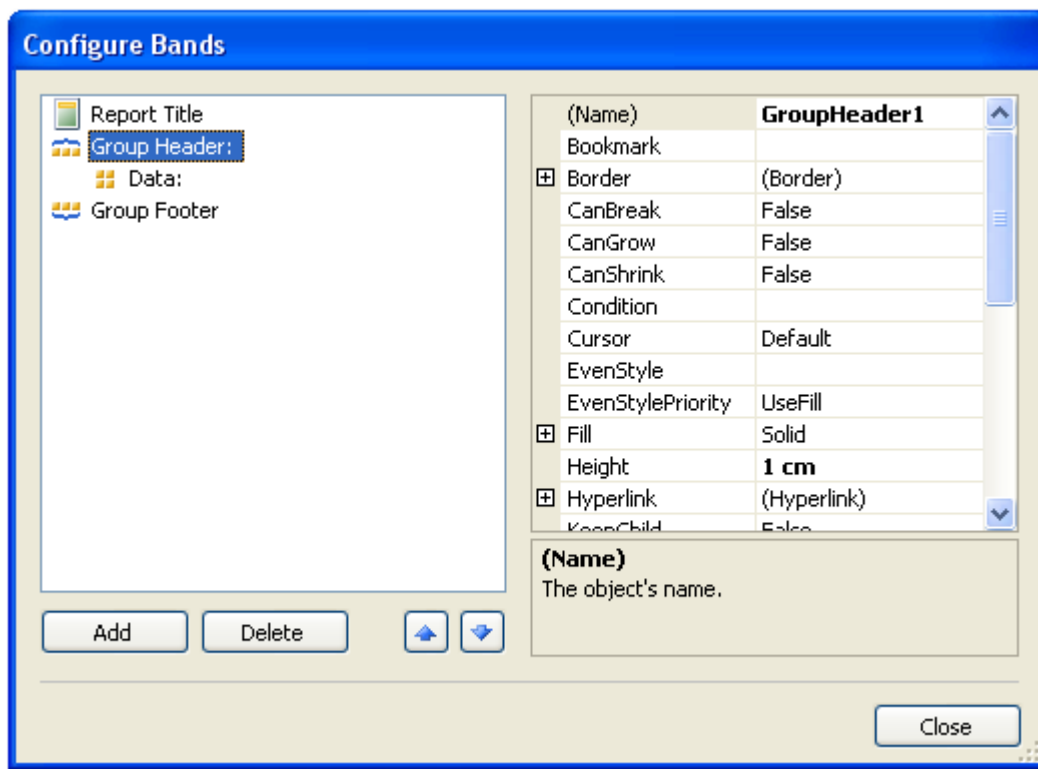
<b>Beverages</b>
Côte de Blaye
Chartreuse verte
Steeleye Stout
Guaraná Fantástica
Sasquatch Ale
Rhönbräu Klosterbier
Lakkalikööri
Outback Lager
Ipoh Coffee
Laughing Lumberjack Lager
Chang
Chai

<b>Condiments</b>
Original Frankfurter grüne Soße
Sirop d'érable
Chef Anton's Gumbo Mix
Northwoods Cranberry Sauce
Grandma's Boysenberry Spread
Chef Anton's Cajun Seasoning
Aniseed Syrup
Louisiana Hot Spiced Okra
Vegie-spread
Louisiana Fiery Hot Pepper Sauce
Gula Malacca
Genen Shouyu

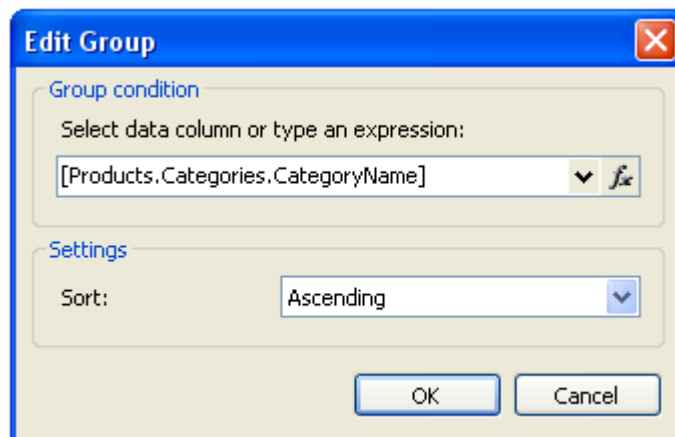
## Creating groups

Adding a group into a report can be done by using two methods.

First method: you add the "Group header" band in the "Configure Bands" window. To do this, press the "Add" button and select the "Group header" band. FastReport adds the group to the available "Data" band or will create a whole group, if such a band is not in the report:

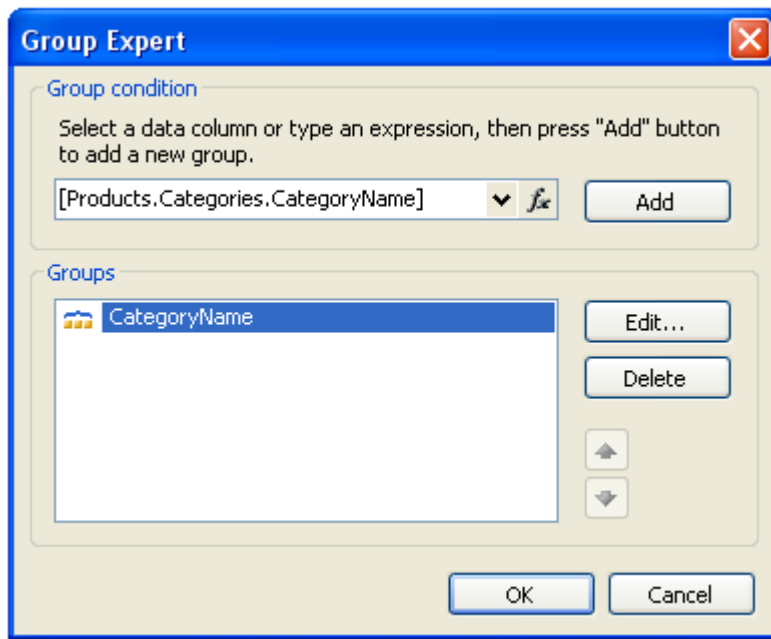


In order to configure a group, double click the "Group header" band. You will see the group header editor window:



You need to set the group condition. This can be any expression or data source column. Also choose the sorting. By default, data is sorted in ascending order.

Second method: you use the wizard, which can be called from the "Report|Group Expert..." menu. In order to create a group, enter the group condition and press the "Add" button:



The wizard will add all the elements of the group into the report. Also it creates the "Text" object on the group header, in which the group condition is printed:



### Sorting the data

For correct working of the group, it is needed to fulfill the following condition: data source must be sorted on that column which is used in the group condition. If this condition is not fulfilled, you will see a lot of groups containing 1-2 data rows:

Beverages
Sasquatch Ale
Steeleye Stout

Seafood
Inlagd Sill
Gravad lax

Beverages
Côte de Blaye
Chartreuse verte

Seafood
Boston Crab Meat
Jack's New England Clam Chowder

Fortunately, there is a possibility of sorting the data source in two ways.

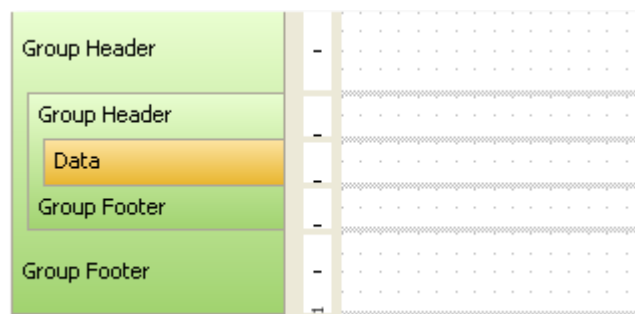
- you can set up the data sorting in the group editor. Data source will be automatically sorted on group condition;
- you can set up the sorting in the "Data" band editor.

Both methods are equivalent, however it is comfortable to use the first method. When creating groups, you set up the data grouping and sorting in one dialog.

In certain situations, the first method should not be used. Assuming that, we will set grouping on the first letter of the product's name. In such case, the product will be sorted only on the first letter, which is not acceptable. You should use the second method and indicate sorting on the full name of the product.

## Nested groups

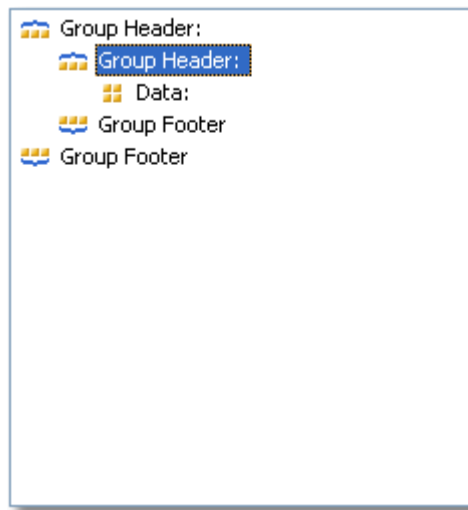
A nested group has several "Group header" bands. The last band contains the "Data" band:



Every group header has its own group condition.

Creating a nested group can be done in the same way like an ordinary. In the first case, you create a simple group and add the nested group in the "Configure Bands" window. For this,

select the existing "Group header" band, press the "Add" button and add another "Group header" band:



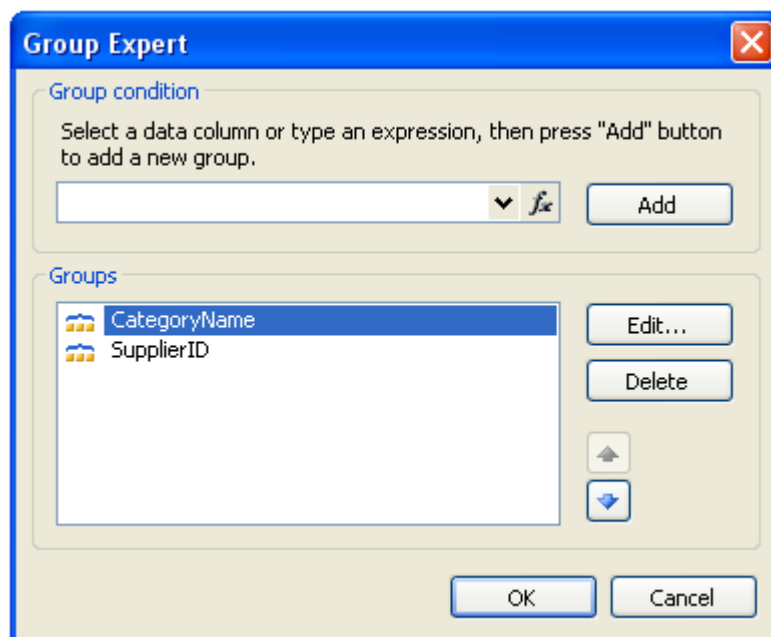
After this, call the editor of the added group and set up the group condition.



In the second case, you use the group expert which we have looked at already. Set the grouping condition and click the "Add" button. The wizard will add the new group to the existing one.

Printing of the nested group does not differ much from printing of an ordinary group. When printing data, FastReport will check all group conditions of all groups. If the condition changes, the corresponding group finishes and a new group starts printing.

## Managing groups

For managing groups, the group expert can be used. It can be called from the "Report|Group Expert..." menu:



With the help of the wizard you can either add or delete a group, and change the grouping order as well. For changing the grouping order, the buttons  and  are used. With the help of the "Edit..." button, you can change the group condition of the chosen group.

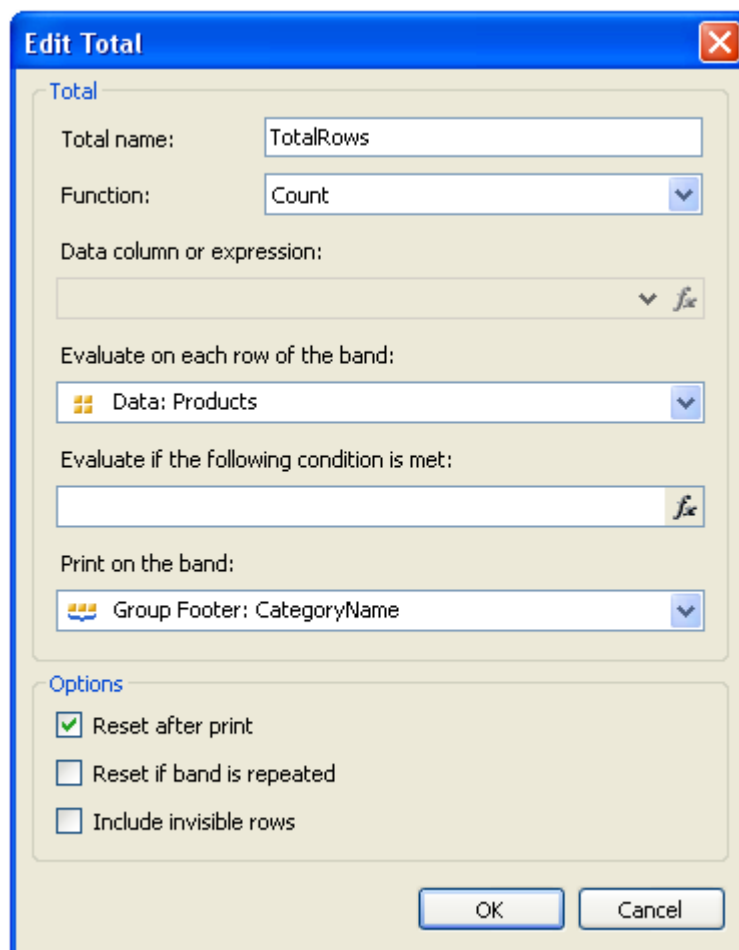
## Printing total values

Grouping is often used for printing some total values in every group. For example, this can be the number of rows in the group, or sum from one of the data columns. For printing such values, the total is used. The use of totals is described in the ["Data"](#) chapter.

In order to print the total value in the group, you need to do the following:

- create a total, by selecting the "Actions|New total..." item in the "Data" window;
- choose the group's data band in the "Evaluate on each row of the band" combobox;
- choose the group footer in the "Print on the band" combobox;
- place the "Text" object, which prints the total value, on the group footer.

For example, for printing the number of rows in every group, configure the total as follows:



The screenshot shows the "Edit Total" dialog box with the following configuration:

- Total name:** TotalRows
- Function:** Count
- Data column or expression:** (empty)
- Evaluate on each row of the band:** Data: Products
- Evaluate if the following condition is met:** (empty)
- Print on the band:** Group Footer: CategoryName
- Options:**
  - Reset after print
  - Reset if band is repeated
  - Include invisible rows

In order to display the value of the total, drag it onto the group footer:

Group Header: CategoryName	-	[Products.Categories.CategoryName]
Data: Products	-	[Products.ProductName] . . . . .
Group Footer	-	. . . . . [TotalRows]

Prepared report will be like this:

<b>Meat/Poultry</b>
Alice Mutton
Perth Pasties
Thüringer Rostbratwurst
Pâté chinois
Tourtière
Mishi Kobe Niku
6
<b>Produce</b>
Rössle Sauerkraut
Uncle Bob's Organic Dried Pears
Manjimup Dried Apples
Longlife Tofu
Tofu
5

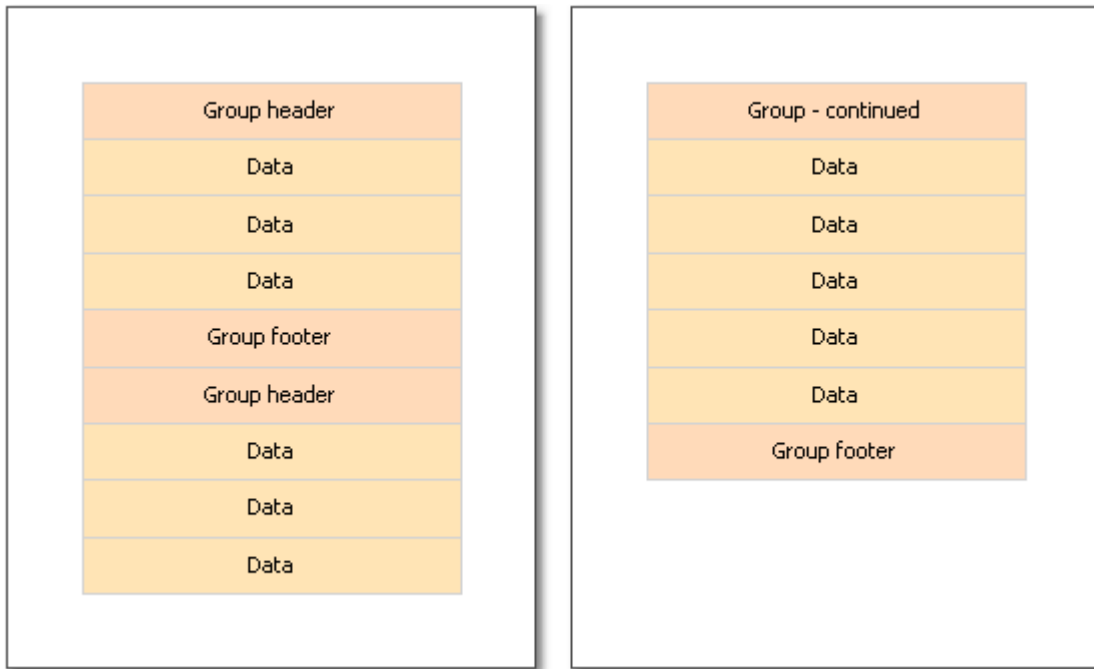
## Repeating the header and footer

Group header and footer have the "RepeatOnEveryPage" property. It can be useful if the group does not fit on one page of a prepared report. By using this property, you can print the group header/footer on each page, where the group is printed. When printing such header/footer, FastReport sets its "Repeated" flag. This can be used for printing different objects on an ordinary group header and on repeat, for example, printing the "continue..." text on a new page. For this, use the "PrintOn" property of the "Text" object (see more details in the ["Booklet" report type](#) section).

In order to print different texts, place two objects on the group header, one on top of the other:

- the first object will be printed on ordinary headers. Set its "PrintOn" property value to "FirstPage, LastPage, OddPages, EvenPages" (that is all values except the "RepeatedBand");
- the second object will be printed only on the headers which are repeated. Set its "PrintOn" property to "RepeatedBand". Add the "Text" object with "continue..." text on the header.

The report will be printed as follows:



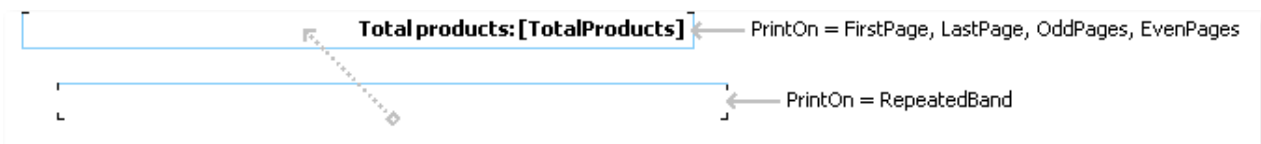
Group footer can also be repeated on every page:

<b>R</b>	
Raclette Courdavault	55,00
Ravioli Angelo	19,50
Rhönbräu Klosterbier	7,75
Röd Kaviar	15,00
Røgede sild	9,50

<b>R</b>	
Rössle Sauerkraut	45,60
<b>Total products: 6</b>	

In this report, the group footer has two objects, placed one on top of the other:



## Group properties

The "Group Header" band has some useful properties.

The "StartNewPage" property allows forming a new page before printing the group. As a result, each group will be placed on a new page.



The new page will not be added before the first group. This is done in order to avoid an empty first page.

The "ResetPageNumber" property allows resetting the page number when printing the group. Usually it is used together with the "StartNewPage" property. As a result, if both of these properties are enabled, every group will be printed on a new page, and will have its own page numeration.

## Subreports

Sometimes at a certain place of the main report, extra data needs to be shown, this can be a separate report with a very complicated structure. You can try to solve this task by using FastReport's rich collection of bands. However, in certain cases, it is preferable to use the "Subreport" object.

The "Subreport" object is an ordinary report object, which can be placed on one of the bands. When you do this, FastReport adds an extra page into the report and connects it with the subreport. On this page it is possible to create an extra report, having any structure.

When printing the report, in which there is the "Subreport" object, the following will be done:

1. The main report will be printed, while the "Subreport" object is not met;
2. The subreport bands will be printed;
3. Printing of the main report will continue.

Since subreport is formed on the sheet of the main report, it cannot contain the following bands: "Report header/footer", "Page header/footer", "Column header/footer", "Overlay".

## Printing modes

Subreport can be printed in two modes.

In the first printing mode, bands and objects of the subreport are printed on the page of the main report. Objects which can be placed on the on the bands and have limits:

- "Subreport" object must be located in lower border of the band;
- Never place other objects under the "Subreport" object. When the report will be working, such objects will be overlapping with the objects of the subreport:



To place other objects under the subreport, use the "Child" band. Place the objects in the following way:



The second printing mode differs in that subreport's objects are printed on the band, which contains the "Subreport" object. You can enable this mode from the context menu of the "Subreport" object. To do this, select the "Print on Parent" item. This mode does not put a limit on placing of the objects. Apart from that, in this mode, a parent band can either grow or shrink depending on how much data has been printed in the subreport.

The only problem with the second mode is that there may be a lot of data in the subreport. When printing it, the parent band will have a big height. In order to print such a band correctly, it is required to break its contents ("CanBreak" property). The break algorithm does not provide 100% quality and in some cases it can lead to the displacement of objects.

## Side-by-side subreports

By placing two "Subreport" objects side-by-side on the same band, you can print two independent data lists. When printing such a report, FastReport acts in the following way:

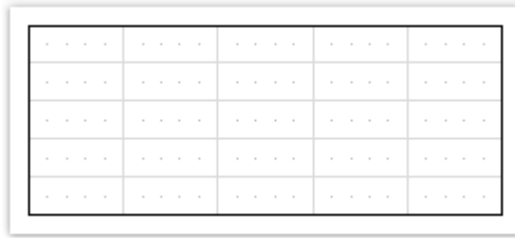
- prints the main report until, the "Subreport" object is not met;
- prints the first subreport;
- turn to the page where the printing of subreport started from, and prints the following subreport;
- after all the subreports have been printed, the main report continues to be printed from where the longest subreport has stopped.

## Nested subreports

On the page of a subreport, you can place another "Subreport" object, so the subreport becomes nested. The number of nesting levels formally is not restricted; however do not get carried away by that. Multiple nesting is very difficult to understand. If you have the possibility, use data bands for printing nested data. A "Data" band can have one or several nested "Data" bands. If you need to print a report of the master-detail type or master-detail-subdetail, there is no need to use the subreport.

## Table-type reports

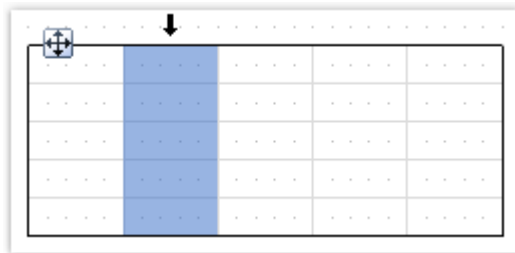
The "Table" object is made up of rows, columns and cells. It is a simplified analog of Microsoft Excel table. It looks in the following way:



## Configuring columns

You can add or delete columns with the help of the context menu. For that:

- select the table or any of its elements and place the cursor on the needed column. The cursor's form changes to a small black arrow.



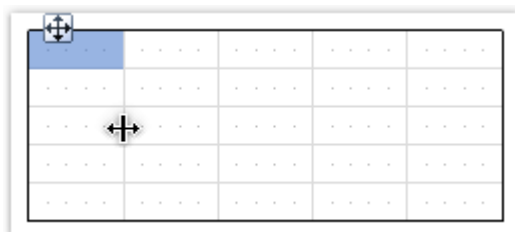
- left click, in order to select the column;
- right click in order to show the column's context menu;
- if you need to select several adjacent columns, left click and, without leaving it, move the mouse to the right or to the left, in order to select adjacent columns.

Column's context menu can also be called in the "Report Tree" window. Open the window, select the needed column and right click the mouse.

## Managing the size of the column

You can set up the width of the column by using one of the following methods:

- select the table or any of its elements and place the cursor on the border between two columns. The form of a cursor changes into a horizontal splitter:



- select a column and indicate the needed width in the "Width" properties. This property is accessible in the "Properties" window.

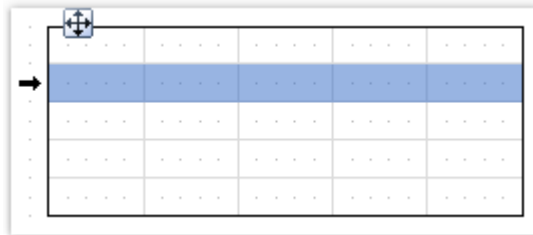
You can also enable the "AutoSize" column property. When running the report, the width of the column will be calculated automatically. In order to limit the width of the column, you can indicate the "MinWidth" and "MaxWidth" properties.

Width of the column should never be larger than the page's width.

## Configuring rows

Rows are configured in the same way. In order to select a row, do the following:

- select the table or any of its elements and place the cursor to the left of the needed row. The cursor's form changes to a small black arrow:



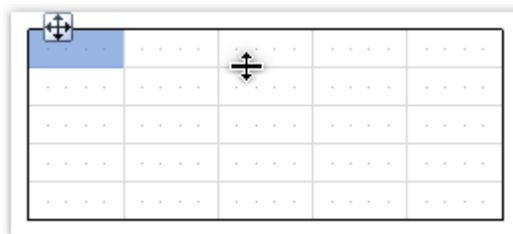
- left click the mouse, in order to select the row;
- right click the mouse in order to show the row's context menu.

If you need to select several adjacent rows, then left click the mouse and, without leaving it, move the mouse to right or to the left, so as to select the adjacent rows.

## Managing the size of the row

You can set up the height of the row by using one of the following methods:

- select the table or any of its elements and place the cursor on the border between two rows. The cursor's form changes into a vertical splitter:



Left click and move the mouse, in order to change the size of the row.

- select the row and indicate the needed height in the "Height" property. This property is accessible in the "Properties" windows.

You can also enable the row's "AutoSize" property. When the report is run, the height of the row will be calculated automatically. In order to limit the height of the row, you can use the minimum height ("MinHeight") and Maximum height ("MaxHeight") properties:

Height of the row should never be larger than the page's height.

## Configuring cells

Cells are text objects. In essence, the cells class is inherited from the "Text" object. Everything which has been said above about the "Text" object applies to the table's cells as well.

Editing the cells' text can be done just like the "Text" object. Besides, you can drag and drop an element from the "Data" window into the cells.

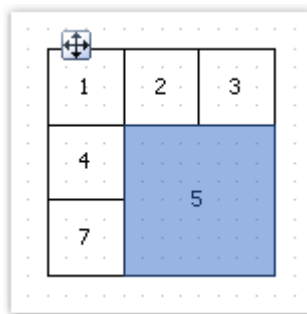
Border and fill of a cell can be configured with the help of the "Border and Fill" toolbar.

In order to call a cell's context menu, just right click on the cell.

## Joining and splitting cells

You can join adjacent cells of the table. As a result, there will be one big cell. In order to do this:

- select the first cell with the help of a mouse;
- left click and, without leaving it, move the mouse in order to select the group of cells;
- on the selected region, right click the mouse in order to show the context menu of the cells;
- in the context menu, choose the "Join cells" item.



In order to split a cell, call its context menu and choose the "Split cell" item.

## Inserting objects in cells

In the cells, you can insert other objects, for example, the picture. The following objects can never be added into the cells:

- "Table";
- "Matrix";
- "Subreport".

In order to add an object into the cell, simply drag it inside the cell. You can freely move an object between cells, and also take it back beyond the table's boundary.

<b>Name</b>	[Employees.FirstName] [Employees.LastName]	
<b>Title</b>	[Employees.Title]	
<b>Phone</b>	[Employees.HomePhone]	
<b>Photo</b>		Total: [Count (Cell2)]

A cell serves as a container to objects placed into it. This means that, you can use the "Dock" and "Anchor" properties of an object inside the cell. This allows changing the size of the object when the size of the cell is changing.

## Printing a table

A table can be printed in two modes:

In the first mode, the table is printed inside the band which it belongs to, and looks just the same as in the designer. In this mode, the table does not split across pages if its width is bigger than the width of the report page. This is the mode of printing by default.

The second mode is dynamic. In this mode, the table is built with the help of script. During this, the resulting table can be different from the initial table, just like the prepared report of FastReport differs from the report template. In dynamic mode, the table can split across pages if it does not fit on the report page.

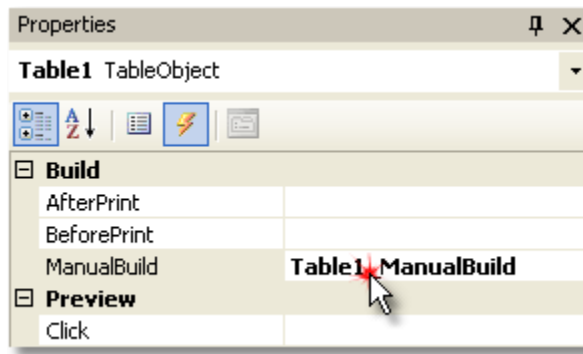
In the dynamic mode, the table does not get printed on the band on which it was placed. Instead of this, the table itself generates a set of bands, which contain parts of the resulting table. This mode of work imposes the following limits:

- never put other objects under the table or near it. Instead of this, use the "Child" band;
- never put two "Table" objects on one band.

Let us look at the dynamic mode in details.

This mode is connected with programming and needs higher qualifications from the report developer.

Formation of the table is done with the help of script. In order to create a script, select the "Table" object, in the "Properties" window click the "Events" button and double click on ManualBuild event:



When this is done, an empty event handler is added into the report code:

```
private void Table1_ManualBuild(object sender, EventArgs e)
{
}

```

In this mode, the source table is used as a template. In the event code, you can print rows and columns from the source table as many times as it is needed. During this, the resulting table will be formed, which can contain an unlimited number of rows and columns. Such a table can split across pages if it does not fit on the report page.

For printing a table, the following methods of the "Table" object are used:

Method	Parameters	Description
PrintRow	int index	Prints the row with the specified index. Row numbering starts from 0.
PrintColumn	int index	Prints the column with the specified index. Column numbering starts from 0.
PrintRows	int[] indices	Prints several rows of the table.
PrintRows	-	Prints all rows of the table.
PrintColumns	int[] indices	Prints several columns of the table.
PrintColumns	-	Prints all columns of the table.
PageBreak	-	Inserts a page break before printing the next column or row.

Printing a table can be done by using one of the following methods:

The first method - printing from top to bottom, then from left to right. This method fits better a table with a variable amount of rows. You must call the methods in the following order:

- PrintRow(row index);
- one or more calls of the PrintColumn(column index) or PrintColumns(columns indices) methods for printing the indicated columns;
- or one call of the PrintColumns() method for printing all columns;
- repeat this sequence in order to print all the needed rows of the table.

Every row of the table must contain the same number of columns. Keep it in mind, when using the PrintColumn(int index) and PrintColumns(int [] indices) methods.



The second method - printing from left to right, then from top to bottom. This method is better for printing a table with a variable number of columns. You must call the methods in the following sequence:

- `PrintColumn(column index);`
- one or several calls of the `PrintRow(row number)` or `PrintRows(rows indices)` for printing the indicated rows;
- or one call of the `PrintRows()` method for printing all rows;
- repeat this sequence in order to print all the needed columns of the table.

Every column of the table must contain the same number of rows. Keep it in mind, when using the `PrintRow(int index)` and `PrintRows(int [] indices)` methods.

Violation of the order of calling the printing methods leads to errors when executing the report. One of the errors - attempting to print the table with the help of the following code:

```
Table1.PrintRows();  
Table1.PrintColumns();
```

This sequence of methods is not correct. You must start printing the table with either the `PrintRow` or `PrintColumn` method.

## Printing complex headers

Here we are talking about headers which contain spanned cells. When printing rows or columns of the table, which has got spanned cells, the cells automatically increase in size. We will show this in the next example:

Header	
1	2

We will create a `ManualBuild` event handler, which will be printing the first column 3 times and the second column 1 time:

```
private void Table1_ManualBuild(object sender, EventArgs e)  
{  
    // printing row 1 and columns 0, 0, 0, 1  
    Table1.PrintRow(0);  
    Table1.PrintColumn(0);  
    Table1.PrintColumn(0);  
    Table1.PrintColumn(0);  
    Table1.PrintColumn(1);  
  
    // printing row 1 and columns 0, 0, 0, 1  
    Table1.PrintRow(1);  
    Table1.PrintColumn(0);  
    Table1.PrintColumn(0);  
    Table1.PrintColumn(0);  
    Table1.PrintColumn(1);  
}
```

Pay attention that we have printed the same number of columns in every row. If this rule is violated, then we will get an unpredictable result.

As a result of executing this code, we will get the following:

Header			
1	1	1	2

As seen, the header cell is spanned automatically. We will make the code a little more complex, so that two groups of columns can be printed:

```
private void Table1_ManualBuild(object sender, EventArgs e)
{
    // print 0 row and two groups of 0, 0, 0, 1 columns
    Table1.PrintRow(0);
    // group 1
    Table1.PrintColumn(0);
    Table1.PrintColumn(0);
    Table1.PrintColumn(0);
    Table1.PrintColumn(1);
    // group 2
    Table1.PrintColumn(0);
    Table1.PrintColumn(0);
    Table1.PrintColumn(0);
    Table1.PrintColumn(1);

    // print 1 row and two groups of 0, 0, 0, 1 columns
    Table1.PrintRow(1);
    // group 1
    Table1.PrintColumn(0);
    Table1.PrintColumn(0);
    Table1.PrintColumn(0);
    Table1.PrintColumn(1);
    // group 2
    Table1.PrintColumn(0);
    Table1.PrintColumn(0);
    Table1.PrintColumn(0);
    Table1.PrintColumn(1);
}
```

When we run the report, we will see the following result:

Header				Header			
1	1	1	2	1	1	1	2

When printing the second column with the following code:

```
Table1.PrintColumn(1);
```

the header is finished and further printing of the first column starts a new header:

```
// group 2
Table1.PrintColumn(0);
```

## Using totals

In the dynamic mode of the "Table" object, the following total functions are supported:

Function	Parameters	Description
Sum	TableCell cell	Returns the sum of the values contained in cell.
Min	TableCell cell	Returns the minimum from the values contained in a cell.
Max	TableCell cell	Returns the maximum from the values contained in a cell.
Avg	TableCell cell	Returns the average of the values contained in a cell.
Count	TableCell cell	Returns the number of rows, contained the specified cell.

In an ordinary printing mode (not dynamic) these functions will not work.

In order to use the total function, place it in the cell of the table. For example, the following function calculates the sum of the values contained in a cell named "Cell1":

```
[Sum(Cell1)]
```

During this, all the cells located higher and on the left of the current cell (in which we are calculating the sum) are analyzed.

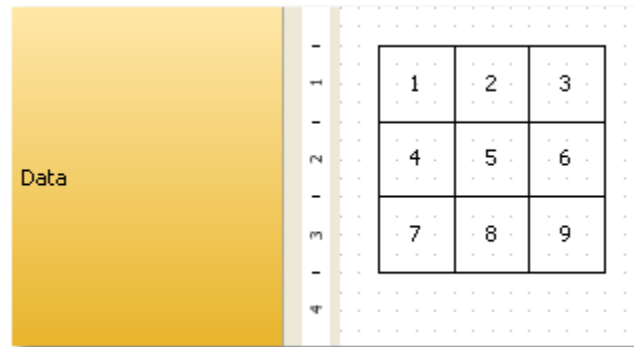
## Table layout

The table which is built dynamically, can be automatically splitted across pages. This behavior is controlled by the "Layout" property of the table. You can choose one of the following values:

Value	Description
AcrossThenDown	Table is rendered across then down.
DownThenAcross	Table is rendered down then across.
Wrapped	Wide table is wrapped and rendered on the same page.

## Examples

Let us look at printing tables for example. As a template, we will use the following report:



1	2	3
4	5	6
7	8	9

To start, select the table and create an event handler for "ManualBuild" event.

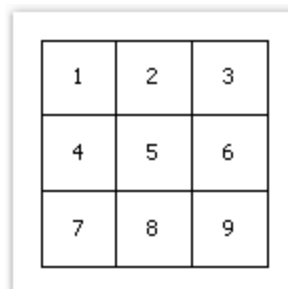
### Example 1. Printing the whole table from top to bottom

```
private void Table1_ManualBuild(object sender, EventArgs e)
{
    // printing row 0 and all of its columns
    Table1.PrintRow(0);
    Table1.PrintColumns();

    // printing row 1 and all of its columns
    Table1.PrintRow(1);
    Table1.PrintColumns();

    // Printing row 2 and all of its columns
    Table1.PrintRow(2);
    Table1.PrintColumns();
}
```

As a result, the following table will be printed, which does not differ from the template:



1	2	3
4	5	6
7	8	9

### Example 2. Printing the table from top to bottom with a repeating row

```
private void Table1_ManualBuild(object sender, EventArgs e)
{
    // Printing row 0 and all of its columns
    Table1.PrintRow(0);
    Table1.PrintColumns();

    // Printing 3 copies of row 1 and all of its columns
    for (int i = 0; i < 3; i++)
    {
```

```

    Table1.PrintRow(1);
    Table1.PrintColumns();
}

// printing row 2 and all of its columns
Table1.PrintRow(2);
Table1.PrintColumns();
}

```

In this example, the middle row gets printed 3 times. As a result, we get the following:

1	2	3
4	5	6
4	5	6
4	5	6
7	8	9

### Example 3. Printing the whole table from left to right

```

private void Table1_ManualBuild(object sender, EventArgs e)
{
    // printing column 0 and all of its rows
    Table1.PrintColumn(0);
    Table1.PrintRows();

    // printing column 1 and all of its rows
    Table1.PrintColumn(1);
    Table1.PrintRows();

    // printing column 2 and all of its rows
    Table1.PrintColumn(2);
    Table1.PrintRows();
}

```

As a result, the following table will be printed, which does not differ from the template:

1	2	3
4	5	6
7	8	9

#### Example 4. Printing a table from left to right with a repeating column

```
private void Table1_ManualBuild(object sender, EventArgs e)
{
    // printing column 0 and all of its rows
    Table1.PrintColumn(0);
    Table1.PrintRows();

    // printing 3 copies of column 1 and all of its rows
    for (int i = 0; i < 3; i++)
    {
        Table1.PrintColumn(1);
        Table1.PrintRows();
    }

    // printing column 2 and all of its rows
    Table1.PrintColumn(2);
    Table1.PrintRows();
}
```

In this example, the middle column of the table gets printed 3 times. As a result, we get the following:

1	2	2	2	3
4	5	5	5	6
7	8	8	8	9

#### Example 5. Printing a table with repeating rows and columns

```
private void Table1_ManualBuild(object sender, EventArgs e)
{
    // ----- printing row 0
    Table1.PrintRow(0);
    // printing column 0
    Table1.PrintColumn(0);
    // printing 3 copies of column 1
    for (int i = 0; i < 3; i++)
        Table1.PrintColumn(1);
    // printing column 2
    Table1.PrintColumn(2);

    // ----- printing 3 copies of row 1
    for (int j = 0; j < 3; j++)
    {
        Table1.PrintRow(1);
        // printing column 0
        Table1.PrintColumn(0);
        // printing 3 copies of column 1
        for (int i = 0; i < 3; i++)
            Table1.PrintColumn(1);
        // printing column 2
        Table1.PrintColumn(2);
    }

    // ----- printing row 2
}
```

```

Table1.PrintRow(2);
// printing column 0
Table1.PrintColumn(0);
// printing 3 copies of column 1
for (int i = 0; i < 3; i++)
    Table1.PrintColumn(1);
// printing column 2
Table1.PrintColumn(2);
}

```

Pay attention that we printed the same number of columns in every row. If this rule is violated, then we will get an unpredictable result.

In this example, the middle row and middle column of the table were printed 3 times. And as a result, we get the following:

1	2	2	2	3
4	5	5	5	6
4	5	5	5	6
4	5	5	5	6
7	8	8	8	9

### Example 6. Using the data source

In all the examples considered, we printed a table, which contains an ordinary text. In this example we will show how to form a table using data source. For this, we will create a table having the following form:

Product Name	Unit Price	Units In Stock
[Products.ProductName]	[Products.UnitPrice]	[Products.UnitsInStock]

We will create the ManualBuild event handler, which will be doing the following:

- get the data source, defined in the report;
- initialize it (fill it with data);
- print the table's rows as many times as there are rows in the data source.

Here is the code of the handler:

```

private void Table1_ManualBuild(object sender, EventArgs e)
{
    // get the data source by its name
    DataSourceBase rowData = Report.GetDataSource("Products");
    // initialize it
    rowData.Init();
}

```

```

// printing the table header
Table1.PrintRow(0);
Table1.PrintColumns();

// loop through the data source rows
while (rowData.HasMoreRows)
{
    // printing the table row
    Table1.PrintRow(1);
    Table1.PrintColumns();

    // select the next data row
    rowData.Next();
}

// printing the table footer
Table1.PrintRow(2);
Table1.PrintColumns();
}

```

If we run the report, we will get the following:

Product Name	Unit Price	Units In Stock
Chai	18,00	39
Chang	19,00	17
Aniseed Syrup	10,00	13
Chef Anton's Cajun Seasoning	22,00	53
Chef Anton's Gumbo Mix	21,35	0
Grandma's Boysenberry Spread	25,00	120
Uncle Bob's Organic Dried Pears	30,00	15
Northwoods Cranberry Sauce	40,00	6

## Example 7. Inserting page breaks

Using the "PageBreak" method of the "Table" object, you can insert a page break when printing a table. Call it before you print a row/column.

We will use the Example 1 to demonstrate how the "PageBreak" method works. Let us print the third row on a new page.

```

private void Table1_ManualBuild(object sender, EventArgs e)
{
    // print the row 0 with all its columns
    Table1.PrintRow(0);
    Table1.PrintColumns();

    // print the row 1 with all its columns
    Table1.PrintRow(1);
    Table1.PrintColumns();

    // insert page break before the row 2
    Table1.PageBreak();

    // print the row 2 with all its columns
    Table1.PrintRow(2);
    Table1.PrintColumns();
}

```



}

As a result, we get the following:

1	2	3
4	5	6

7	8	9
---	---	---

### Example 8. Printing totals

We will look at the use of the total function in Example 6. We will modify it in the following way:

Product Name	Unit Price	Units In Stock
[Products.ProductName]	[Products.UnitPrice]	[Products.UnitsInStock]
	<b>Total:</b>	<b>[Sum(Cell8)]</b>

If we run the report, we will get the following:

Outback Lager	15,00	15
Fløtemysost	21,50	26
Mozzarella di Giovanni	34,80	14
Röd Kaviar	15,00	101
Longlife Tofu	10,00	4
Rhönbräu Klosterbier	7,75	125
Lakkalikööri	18,00	57
Original Frankfurter grüne Soße	13,00	32
	<b>Total:</b>	<b>3119</b>

## Matrix-type reports

The "Matrix" object is a variety table and like the "Table" object, is made up of rows, columns and cells. At the same time, it is not known before hand how many rows and columns will be in the matrix - this depends on the data to which it is connected.

The object looks like this:

Employee	[Year]	Total
[Name]	[Revenue]	
Total		

When printing, the matrix fills up the values and grows up and down. The result can be as follows:

Employee	1999	2000	2001	2002	Total
Andrew Fuller	\$3,900.00	\$2,100.00		\$1,800.00	\$7,800.00
Janet Leverling	\$6,100.00	\$3,200.00			\$9,300.00
Nancy Davolio	\$3,300.00	\$2,700.00	\$3,100.00	\$1,700.00	\$10,800.00
Steven Buchanan		\$3,999.00	\$8,100.00		\$12,099.00
<b>Total</b>	\$13,300.00	\$11,999.00	\$11,200.00	\$3,500.00	<b>\$39,999.00</b>

## A few theory

Let us look at the elements of a matrix:

	1	2	3	4
a	a1	a2	a3	a4
b	b1	b2	b3	b4

In the figure, we see a matrix with 2 rows and 4 columns. Here a, b - row header, 1, 2, 3, 4 - column header, a1...a4, b1...b4 - cells. In order to build such a report, only one data source will be needed, which has got 3 columns and contains the following data:

```
a 1 a1
a 2 a2
a 3 a3
a 4 a4
b 1 b1
b 2 b2
b 3 b3
b 4 b4
```

As seen, the first column represents the matrix row, the second - matrix column, and the third - contents of the cells at the intersection of rows and columns with the indicated number. When creating a report, FastReport creates a matrix in the memory and fills it with data. During this, the matrix dynamically increases, if the row or column with the given number doesn't exist yet.

A header can have more than one level. Let us look at the following example:

	10		20	
	1	2	1	2
a	a10.1	a10.2	a20.1	a20.2
b	b10.1	b10.2	b20.1	b20.2

In this example, a column is compound, that is, it has got two values. This report requires the following data:

```

a      10      1      a10.1
a      10      2      a10.2
a      20      1      a20.1
a      20      2      a20.2
b      10      1      b10.1
b      10      2      b10.2
b      20      1      b20.1
b      20      2      b20.2
    
```

Here, the first column represents the row, the second and the third represent the matrix column. The last data column contains the value of the cell.

The next matrix element - subtotal and grand total, the next figure demonstrates it:

	10			20			Total
	1	2	Total	1	2	Total	
a	a10.1	a10.2	a10.1+a10.2	a20.1	a20.2	a20.1+a20.2	sum(a)
b	b10.1	b10.2	b10.1+b10.2	b20.1	b20.2	b20.1+b20.2	sum(b)
Total	a10.1+b10.1	a10.2+b10.2	a10.1+b10.1+a10.2+b10.2	a20.1+b20.1	a20.2+b20.2	a20.1+b20.1+a20.2+b20.2	sum(a)+sum(b)

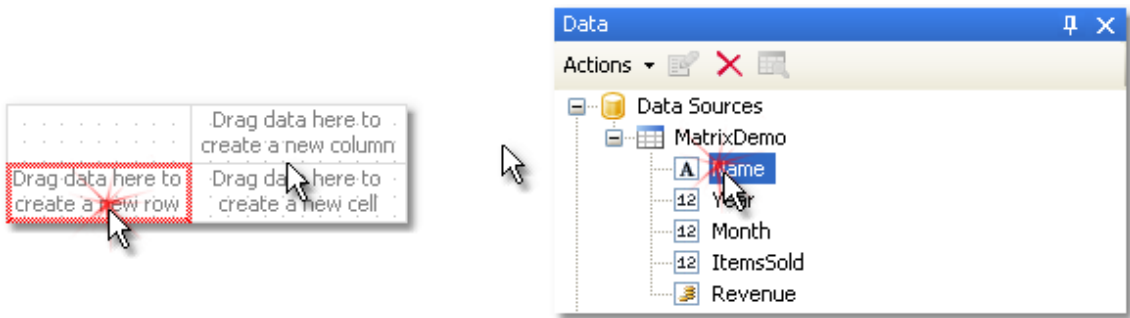
This report is built on the same data used in the previous example. Rows that shown grey in the figure, are calculated automatically.

### Configuring the matrix

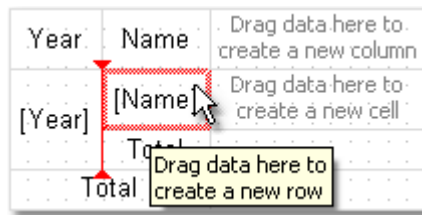
After you have placed a new "Matrix" object on a sheet, it will be as follows:



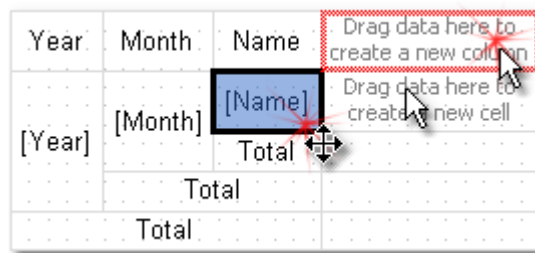
Matrix can be configured with the help of the mouse. To do this, drag and drop data source columns from the "Data" window onto the matrix, to create rows, columns and cells. The matrix highlights a red frame to a place where the new data will be placed:



If the matrix contains some elements already, then when placing a new element, an indicator will be shown. In the given case, the new data will be placed between the "Year" and "Name" elements:



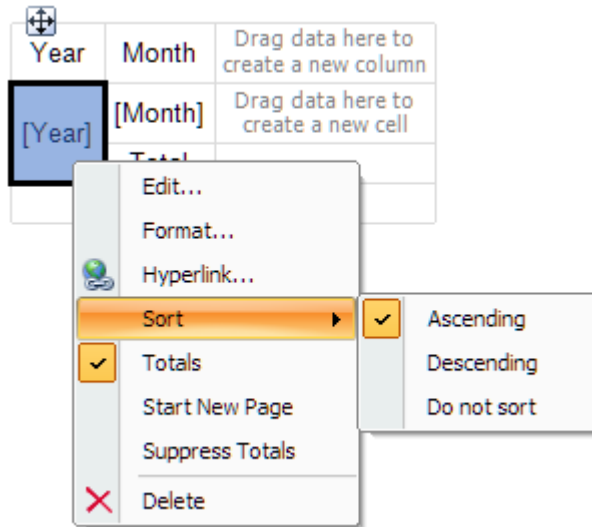
You can also change the order of the existing elements. To do this, click on the element's border (shown with black) and drag it to the needed place:



To delete an element, select it with the mouse, and press the "Delete" key.

## Configuring headers

To configure the header element, select it and right-click to display its context menu:



By default, data in the matrix header is sorted in ascending order. You can change the order of sorting, by selecting the "Sort" item.

Ordinarily, every item in the matrix header has a total (this is a cell with a "Total" text). You can delete the total, selecting it and pressing the "Delete" key. In order to enable total again, select an element to which it belongs, and choose the "Total" item in its context menu.

The "Start New Page" menu item tells the matrix to insert page breaks after printing each header value. For example, if you enable page breaks for the "Year" item (as shown in the picture above), every year value will be printed on its own page.

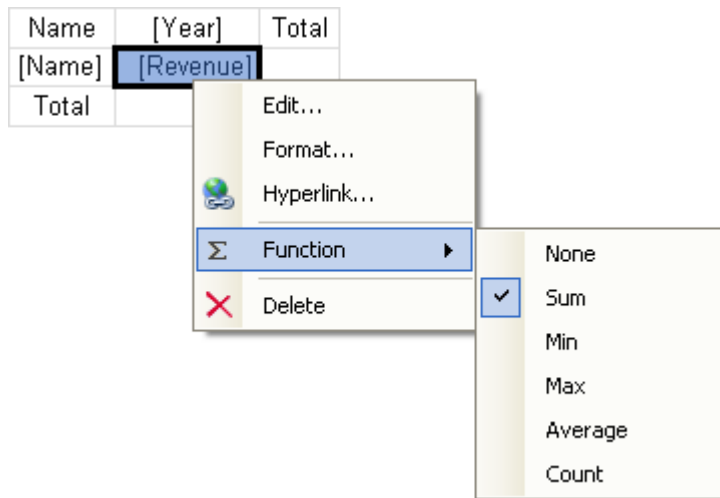
The "Suppress Totals" item allows to suppress totals in case when the group (on which the total value is calculated) contains only one value.

## Configuring cells

For a matrix cell, you can choose a function which will be used when calculating the total. A list of functions which can be used is given below:

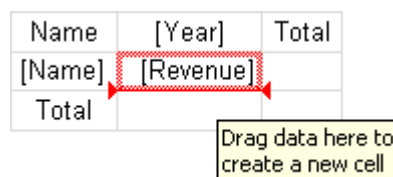
Function	Description
None	Cell value is not processed.
Sum	Returns the sum of the values in the matrix cell.
Min	Returns the minimum value.
Max	Returns the maximum value.
Average	Returns the average value.
Count	Returns number of nonempty values.

By default, the Sum function is used. You can change this by selecting a cell and choosing the "Function" item in its context menu:



Choose the "None" function, if you don't intend to print the total in the given cell.

In a matrix, there can be one or several data cells. In case the matrix has several cells, they can be arranged side-by-side or on top of each other. The "CellsSideBySide" property of the matrix controls how the cells are arranged. It can be changed from the context menu of the "Matrix" object. You can also choose the arrangement order when adding the second cell in the matrix. When doing this, look at the red indicator which shows where the second value will be placed:



After you have added the second value, the rest of the values will be added in the chosen order.

## Styling the matrix

In order to change the appearance of a matrix cell, click on the needed cell. With the help of the toolbar you can set up the font, border and fill. In order to change the appearance of several cells at once, select the group of cells. To do this, select the top left cell, and, without leaving the mouse, move the mouse so as to select the group:



You can use styles in order to change the appearance of the whole matrix. To do this, invoke the context menu of the "Matrix" object and choose the style:

## Row and column size management

Since the "Matrix" object is a kind of "Table" object, it allows setting row and column sizes in the same way.

By default, a matrix has got the "AutoSize" mode enabled. In this mode, the matrix calculates the column/row sizes automatically. You can as well manage the size of the object manually. To do this, disable the "AutoSize" property of the matrix. Rows and columns have the same property, you can use it if autosize of the matrix is disabled.

In order to limit the minimum and maximum width of a column, select a column and set up its "MinWidth" and "MaxWidth" properties.

In order to limit the minimum and maximum height of a row, select a row and set up its "MinHeight" and "MaxHeight" properties.

## Examples

Let us look at examples of using the "Matrix" object. For a start create a new report and put the "Matrix" object on the "Report Title" band. You can also use the "Data" band - in that case there is no need to connect the band to a data source. In the given case, it does not matter on which of the two bands you will put the matrix, since both bands will be printed once when the report is started. The report looking as follows:



Do not put "Matrix" object on bands which will be printed on every new page - "Page header", "Page footer", etc. The matrix in this case will be created every time when the band will be printed, which will lead to stack overflow.

Most examples will be using the "MatrixDemo" table, which is bundled with the FastReport package. This table contains the following data:

Name	Year	Month	ItemsSold	Revenue
Nancy Davolio	1999	2	1	1000
Nancy Davolio	1999	11	1	1100
Nancy Davolio	1999	12	1	1200
Nancy Davolio	2000	1	1	1300
Nancy Davolio	2000	2	2	1400
Nancy Davolio	2001	2	2	1500
Nancy Davolio	2001	3	2	1600
Nancy Davolio	2002	1	2	1700
Andrew Fuller	2002	1	2	1800
Andrew Fuller	1999	10	2	1900
Andrew Fuller	1999	11	2	2000
Andrew Fuller	2000	2	2	2100
Janet Leverling	1999	10	3	3000
Janet Leverling	1999	11	3	3100
Janet Leverling	2000	3	3	3200
Steven Buchanan	2001	1	3	4000
Steven Buchanan	2001	2	4	4100
Steven Buchanan	2000	1	4	3999



## Example 1. Simple matrix

The matrix will contain one value in a row and a column as well as one data cell. In order to build a matrix you need to add "MatrixDemo" data columns in the following way:

- add "Year" data column to the row header;
- add "Name" data column to the column header;
- add "Revenue" data column to the matrix cell.

After that the matrix will look as follows:

Year	[Name]	Total
[Year]	[Revenue]	
Total		

Let us improve the matrix appearance:

- choose "Orange" style for the matrix;
- choose "Tahoma, 8" font for all matrix cells;
- select the word "Total" with bold type;
- choose "Glass" type filling for the cells in upper row;
- disable the autosize of the matrix and increase the size of rows and columns.

After that the matrix will have the following view:

Year	[Name]	<b>Total</b>
[Year]	[Revenue]	
<b>Total</b>		

Run the report and you will see the following result:

Year	Andrew Fuller	Janet Leverling	Nancy Davolio	Steven Buchanan	<b>Total</b>
1999	3900	6100	3300		13300
2000	2100	3200	2700	3999	11999
2001			3100	8100	11200
2002	1800		1700		3500
<b>Total</b>	7800	9300	10800	12099	39999

There is lack of the following things in the matrix:

- there is no title for the "Name" data column;
- sums are not printed in currency format;

You can add a title for "Name" data column in the following way:

- the text "Year/Employee" can be put into the left upper corner of the matrix;
- diagonal line and the second "Text" object can be placed there, as shown below:

Employee		
Year	[Name]	Total
[Year]	[Revenue]	
Total		

- enable the matrix title. To do this, choose the "Show Title" item in the context menu of the "Matrix" object. Any text can be included in the title:

Employee		
Year	[Name]	Total
[Year]	[Revenue]	
Total		

In order to set the data formatting, select the whole cell area, as shown in the figure below, and set format by selecting the "Format..." item in the context menu:

Employee		
Year	[Name]	Total
[Year]	[Revenue]	
Total		

After that, a prepared report will be as follows:

Employee					
Year	Andrew Fuller	Janet Leverling	Nancy Davolio	Steven Buchanan	Total
1999	\$3,900.00	\$6,100.00	\$3,300.00		\$13,300.00
2000	\$2,100.00	\$3,200.00	\$2,700.00	\$3,999.00	\$11,999.00
2001			\$3,100.00	\$8,100.00	\$11,200.00
2002	\$1,800.00		\$1,700.00		\$3,500.00
Total	\$7,800.00	\$9,300.00	\$10,800.00	\$12,099.00	\$39,999.00

### Example 2. Multilevel headers

The matrix will have one value in a row, two values in a column and one data cell. We will get the previous example as a base and add a new item into it:

- we will add the "Month" data column to the row header to the right of "Year" item.

After adding a new item, improve the appearance of the matrix. It is also necessary to set up formatting for cells. After that, the matrix will be as follows:


Employee			
Year	Month	[Name]	Total
[Year]	[Month]	[Revenue]	
	Total		
Total			

Run the report and you will have the following result:

		Employee				
Year	Month	Andrew Fuller	Janet Leverling	Nancy Davolio	Steven Buchanan	Total
1999	2			\$1,000.00		\$1,000.00
	10	\$1,900.00	\$3,000.00			\$4,900.00
	11	\$2,000.00	\$3,100.00	\$1,100.00		\$6,200.00
	12			\$1,200.00		\$1,200.00
	Total	\$3,900.00	\$6,100.00	\$3,300.00	\$0.00	\$13,300.00
2000	1			\$1,300.00	\$3,999.00	\$5,299.00
	2	\$2,100.00		\$1,400.00		\$3,500.00
	3		\$3,200.00			\$3,200.00
	Total	\$2,100.00	\$3,200.00	\$2,700.00	\$3,999.00	\$11,999.00
2001	1				\$4,000.00	\$4,000.00
	2			\$1,500.00	\$4,100.00	\$5,600.00
	3			\$1,600.00		\$1,600.00
	Total	\$0.00	\$0.00	\$3,100.00	\$8,100.00	\$11,200.00
2002	1	\$1,800.00		\$1,700.00		\$3,500.00
	Total	\$1,800.00	\$0.00	\$1,700.00	\$0.00	\$3,500.00
<b>Total</b>		\$7,800.00	\$9,300.00	\$10,800.00	\$12,099.00	\$39,999.00

### Example 3. Printing the name of the month

In the previous example, there were numbers of months printed in a matrix. This occurred, because the "Month" data column contains the number of the month, not its name. In order to print the name of the month, do the following:

- choose a cell where the number of the month is printed. In our case it is a cell with a name "Cell8";
- in the "Property" window press the  button and double click "BeforePrint" event;
- FastReport will add an empty event handler in the report script. Write the following code:


```
private void Cell8_BeforePrint(object sender, EventArgs e)
{
    string[] month Names = new string[] {
        "January", "February", "March", "April",
        "May", "June", "July", "August",
        "September", "October", "November", "December" };
    // Cell8 is a cell that prints the month number.
    // Cell8.Value is a value printed in the cell (i.e. the month number).
    // This value is of System.Object type, so we need to cast it to int
    Cell8.Text = monthNames[(int)Cell8.Value - 1];
}
```

When you run the report, you will have the following result:

Year	Month	Andrew Fuller	Janet Leverling
1999	February		
	October	\$1,900.00	\$3,000.00
	November	\$2,000.00	
	December		
	<b>Total</b>	\$3,900.00	
2000	January		
	February		
	March		

#### Example 4. Conditional highlighting

You can set conditional highlighting for matrix cells, just like for "Text" object. More details about this can be found in the ["Conditional highlighting"](#) section.

Let us look at Example 2, and see how to highlight an amount more than 3000 in red. For this, select the cell with "Revenue" text and press the  button on "Text" toolbar. In the conditions editor, add the following condition:

*Value > 3000*

Choose red text color for the condition. A prepared report will be as follows:

Year	Month	Employee				Total
		Andrew Fuller	Janet Leverling	Nancy Davolio	Steven Buchanan	
1999	2			\$1,000.00		\$1,000.00
	10	\$1,900.00	\$3,000.00			\$4,900.00
	11	\$2,000.00	\$3,100.00	\$1,100.00		\$6,200.00
	12			\$1,200.00		\$1,200.00
	<b>Total</b>	\$3,900.00	\$6,100.00	\$3,300.00	\$0.00	\$13,300.00
2000	1			\$1,300.00	\$3,999.00	\$5,299.00
	2	\$2,100.00		\$1,400.00		\$3,500.00
	3		\$3,200.00			\$3,200.00
	<b>Total</b>	\$2,100.00	\$3,200.00	\$2,700.00	\$3,999.00	\$11,999.00
2001	1				\$4,000.00	\$4,000.00
	2			\$1,500.00	\$4,100.00	\$5,600.00
	3			\$1,600.00		\$1,600.00
	<b>Total</b>	\$0.00	\$0.00	\$3,100.00	\$8,100.00	\$11,200.00
2002	1	\$1,800.00		\$1,700.00		\$3,500.00
	<b>Total</b>	\$1,800.00	\$0.00	\$1,700.00	\$0.00	\$3,500.00
<b>Total</b>		\$7,800.00	\$9,300.00	\$10,800.00	\$12,099.00	\$39,999.00

As seen, total values are not highlighted. This occurred, because we chose highlight condition for only one cell. To highlight the rest of the values, it is needed to set up the highlight for all matrix cells.

In this example we used conditional highlight which depends on cell value itself. Besides, you

can highlight a cell depending on values from matrix headers. We will show by the following example, how to highlight cells, which are belongs to 2000 year, in red. For this, select matrix cells as shown in the figure below:

		Employee	
Year	Month	[Name]	Total
[Year]	[Month]	[Revenue]	
	Total		
Total			

Set the following highlight condition:

```
(int)Matrix1.RowValues[0] == 2000
```

In this case "Matrix1" is a name of our matrix. The "RowValues" property of the matrix has got an "object[]" type and contains an array of values from the row header of the current printed row. Number of values in array is equal to number of levels in a header. There are two values in our example, the first one is "Year" and the second one is "Month".

Do not highlight the last row. "RowValues" property has an undetermined value for it and will cause an error when building the report.

When we run the report, we will have the following result:

		Employee				
Year	Month	Andrew Fuller	Janet Leverling	Nancy Davolio	Steven Buchanan	Total
1999	2			\$1,000.00		\$1,000.00
	10	\$1,900.00	\$3,000.00			\$4,900.00
	11	\$2,000.00	\$3,100.00	\$1,100.00		\$6,200.00
	12			\$1,200.00		\$1,200.00
	<b>Total</b>	\$3,900.00	\$6,100.00	\$3,300.00	\$0.00	\$13,300.00
2000	1			\$1,300.00	\$3,999.00	\$5,299.00
	2	\$2,100.00		\$1,400.00		\$3,500.00
	3		\$3,200.00			\$3,200.00
	<b>Total</b>	\$2,100.00	\$3,200.00	\$2,700.00	\$3,999.00	\$11,999.00
2001	1				\$4,000.00	\$4,000.00
	2			\$1,500.00	\$4,100.00	\$5,600.00
	3			\$1,600.00		\$1,600.00
	<b>Total</b>	\$0.00	\$0.00	\$3,100.00	\$8,100.00	\$11,200.00
2002	1	\$1,800.00		\$1,700.00		\$3,500.00
	<b>Total</b>	\$1,800.00	\$0.00	\$1,700.00	\$0.00	\$3,500.00
<b>Total</b>		\$7,800.00	\$9,300.00	\$10,800.00	\$12,099.00	\$39,999.00

You can also use matrix's "ColumnValues" property for column values reference.

### Example 5. Highlighting even rows

To improve the appearance of a matrix, you can highlight even rows or columns with other color. We will use the Example 2 to show how to do it.

Select the whole area of matrix data as it is shown in the figure:

		Employee	
Year	Month	[Name]	Total
[Year]	[Month]	[Revenue]	
	<b>Total</b>		
<b>Total</b>			

Call the conditional highlighting editor. Add the following condition:

```
Matrix1.RowIndex % 2 != 0
```

and choose background color a little darker than the previous one. In this example "Matrix1" is a name of our matrix. The "RowIndex" property of the matrix returns the number of the current printed line.

For column highlighting, use the matrix "ColumnIndex" property in the same way.

When we run the report, we will see the following:

		Employee				
Year	Month	Andrew Fuller	Janet Leverling	Nancy Davolio	Steven Buchanan	Total
1999	2			\$1,000.00		\$1,000.00
	10	\$1,900.00	\$3,000.00			\$4,900.00
	11	\$2,000.00	\$3,100.00	\$1,100.00		\$6,200.00
	12			\$1,200.00		\$1,200.00
	<b>Total</b>	\$3,900.00	\$6,100.00	\$3,300.00	\$0.00	\$13,300.00
2000	1			\$1,300.00	\$3,999.00	\$5,299.00
	2	\$2,100.00		\$1,400.00		\$3,500.00
	3		\$3,200.00			\$3,200.00
	<b>Total</b>	\$2,100.00	\$3,200.00	\$2,700.00	\$3,999.00	\$11,999.00
2001	1				\$4,000.00	\$4,000.00
	2			\$1,500.00	\$4,100.00	\$5,600.00
	3			\$1,600.00		\$1,600.00
	<b>Total</b>	\$0.00	\$0.00	\$3,100.00	\$8,100.00	\$11,200.00
2002	1	\$1,800.00		\$1,700.00		\$3,500.00
	<b>Total</b>	\$1,800.00	\$0.00	\$1,700.00	\$0.00	\$3,500.00
<b>Total</b>		\$7,800.00	\$9,300.00	\$10,800.00	\$12,099.00	\$39,999.00

## Example 6. Using Expressions

In previous examples we created matrix by dragging columns from the "Data" window. You can also use expressions for this purpose. In order to insert an expression into a matrix do the following:

- add any element from "Data" window into matrix. It can be any element, for example, a system variable "Date" - we just use it to create a matrix element;
- double click the element and select the needed expression in the expressions editor window.

If your matrix has expressions instead of data field, you have to check that the matrix "DataSource" property was set up correctly. When working with data columns, this property is filled automatically when you drag a column onto a matrix.

Let us consider an example on how to use expressions. For this, we will use the "Order Details" table as a data source, which contains a list of sold products, grouped by employees. There are several relations in this table, which gives an access to the name of an employee, product name and its category.

Our matrix will show each employee's sales, categorized by products. In order to build the matrix, do the following:

- add the "Order Details.Products.Categories.CategoryName" data column to the column header;
- add any item to the row header in order to create a matrix element. Then set the following expression for the header element:

```
[Order Details.Orders.Employees.FirstName] + " " + [Order Details.Orders.Employees.LastName]
```

- add any item to data cell in order to create a matrix element. Then set the following expression for the cell:

```
[Order Details.UnitPrice] * [Order Details.Quantity] * (decimal)(1 - [Order Details.Discount])
```

Why did we indicate such a long data column as a name of an employee though we could get a name from the "Employees.FirstName"? We did that because the matrix is connected to "Order Details" data source. Using relations between this data source and other tables, it is easy to refer to its columns (more details about relations can be read in the "Data" chapter). If we refer directly to the "Employees.FirstName" data column, we will get a name of the first employee in a table.

Set up the matrix appearance. After that it will look as the follows:

Employee	[CategoryName]	Total
[Order Details.Orders.Employees.FirstName] + " " + [Order Details.Orders.Employees.LastName]	[[Order Details.UnitPrice] * [Order Details.Quantity] * (decimal)(1 - [Order Details.Discount])]	
Total		

When we run the report, we will see quite a big matrix which occupies 2 sheets:

Employee	Beverages	Condiments	Confections	Dairy Products	Grains/Cereals
Andrew Fuller	40 248,25	14 850,67	21 455,69	23 812,55	11 111,11
Anne Dodsworth	19 642,56	10 125,55	8 053,16	21 101,13	1 234,56
Janet Leverling	44 757,41	13 381,64	33 622,40	32 320,84	21 111,11
Laura Callahan	17 897,85	14 637,66	21 699,91	21 269,47	1 111,11
Margaret Peacock	50 308,21	23 314,87	27 768,73	35 989,89	1 111,11
Michael Suyama	9 450,20	4 648,47	6 859,63	1 111,11	1 111,11
Nancy Davolio	46 599,36	13 561,56	28 568,21	1 111,11	1 111,11
Robert King	27 963,83	8 851,38	1 111,11	1 111,11	1 111,11
Steven Buchanan	11 000,53	1 111,11	1 111,11	1 111,11	1 111,11

### Example 7. Pictures in cells

Matrix cells are inherited from the "Text" object and can display text data. If it is not enough, you can put any object into the cell. Let us see how to display a picture in a matrix.

For this we will take Example 6 as a basis. Let's add a photo of an employee ("Employees.Photo" data column) and the category picture ("Categories.Picture" data column). Do the following:

- select the cell containing the employee's name and increase its size;
- add the "Picture" object to this cell;
- in order to show an employee's photo, bind the "Picture" object to the following data column (this can be done in the object editor):

*Order.Details.Orders.Employees.Photo*

- select the cell containing the category name and increase its size;
- add the "Picture" object to this cell;
- in order to show a category picture, bind the "Picture" object to the following data column (this can be done in the object editor):




*Order.Details.Products.Categories.Picture*

After that the matrix will look as follows:



Employee	[CategoryName]	Total
[Order]	[[Order Details.UnitPrice] * [Order Details.Quantity] * (decimal)(1 - [Order Details.Discount])]	
Total		

When we run the report, we will see the following:

Employee	Beverages	Condiments	Confections
Andrew Fuller 	40 248,25	14 850,67	
Anne Dodsworth 	19 642,56	10 125,67	
Janet Leverling 	44 700,00		

### Example 8. Objects in cells

Using objects inserted into matrix cells you can have various visual effects. We will show in the following example, how to draw a simple scale indicating employee's sales level.

The matrix will use the "MatrixDemo" data source. To build a matrix, add data columns in the following way:

- add "Year" data column to the row header;
- add "Name" data column to the column header;
- add "Revenue" data column to the matrix cell.

Set the appearance of the matrix in the following way:

Employee	[Year]	Total
[Name]	[Revenue]	
Total		

Now let us add three "Shape" objects to the cell with "Revenue" value. These objects will serve as indicators in the following way:

- if value in a cell is less than 100, only one object of red color will be shown;
- if value in a cell is less than 3000, two objects of yellow color will be shown;
- if value in a cell is more or equal to 3000, three objects of green color will be shown.

Now the matrix looks like this:

Employee	[Year]	Total
[Name]	■■■ [Revenue]	
Total		

To control objects, we will use an event handler for a matrix cell. For this, select the "Revenue" cell, and create the "BeforePrint" event handler using the "Properties" window. Write the following code in the handler:

```
private void Cell4_BeforePrint(object sender, EventArgs e)
{
    // In our example, a cell has the Cell4 name.

    // Get cell value which is in the Cell4.Value property.
    // Some cells in our matrix will be empty. We'll take it into account (null check).
    // The value should be cast to decimal type, because data source column
    // [MatrixDemo.Revenue] is of System.Decimal type.
    decimal value = Cell4.Value == null ? 0 : (decimal)Cell4.Value;

    // Switch shape objects on or off depending on the value:
    // value < 100 - one object is visible;
    // value < 3000 - two objects are visible;
    // value >= 3000 - all objects are visible
    Shape1.Visible = true;
    Shape2.Visible = value >= 100;
    Shape3.Visible = value >= 3000;

    // Choose the color of objects:
    // value < 100 - red color;
    // value < 3000 - yellow color;
    // value >= 3000 - green color
    Color color = Color.Red;
    if (value >= 100)
        color = Color.Yellow;
    if (value >= 3000)
        color = Color.GreenYellow;

    // Set the objects' color
    Shape1.Fill = new SolidFill(color);
    Shape2.Fill = new SolidFill(color);
}
```

```
Shape3.Fill = new SolidFill(color);
}
```

When we run the report, we will see the following:

Employee	1999	2000	2001	2002	Total
Andrew Fuller	3 900,00	2 100,00		1 800,00	7 800,00
Janet Leverling	6 100,00	3 200,00			9 300,00
Nancy Davolio	3 300,00	2 700,00	3 100,00	1 700,00	10 800,00
Steven Buchanan		3 999,00	8 100,00		12 099,00
<b>Total</b>	13 300,00	11 999,00	11 200,00	3 500,00	39 999,00

### Example 9. Filling a matrix manually

In all the examples we have looked at, the matrix was filled with data automatically because it was connected to data source. Data source for the matrix is indicated in the "DataSource" property. Though we did not set the value of this property manually, it occurred implicitly while adding data columns to the matrix.

Using script it is possible to fill in the matrix manually. For this, it is needed to create the "ManualBuild" event handler of the matrix. Call "AddValue" method in the handler code to add a value. Let us show how to create a matrix which will print a 10x10 table of the following kind:

	1	2	3	...
1	1			
2		2		
3			3	
...				...


Do the following:

- add an empty matrix into the report;
- put any element from the "Data" window into the row, column and cell of the matrix. Then call expression editor by double clicking the matrix element and clear an expression;
- clear the "DataSource" property of the matrix.

These steps are required to create a "dummy" matrix which has one row, column and cell. As a result the matrix will be as follows:

	☐	<b>Total</b>
☐	☐	
<b>Total</b>		

Now create a "ManualBuild" event handler. For that, select the matrix, go "Properties" window

and press the  button. Double click the "ManualBuild" event and FastReport will create an empty event handler. Write the following code in it:

```
private void Matrix1_ManualBuild(object sender, EventArgs e)
{
    // Our matrix has one level in row, column and cell.
    // Create 3 arrays of object[] type, each with one element
    // (per number of levels).
    object[] columnValues = new object[1];
    object[] rowValues = new object[1];
    object[] cellValues = new object[1];

    for (int i = 1; i <= 10; i++)
    {
        // Filling arrays
        columnValues[0] = i;
        rowValues[0] = i;
        cellValues[0] = i;

        // Adding data into the matrix
        Matrix1.AddValue(columnValues, rowValues, cellValues);
    }
}
```

In a handler, you should use the "AddValue" method of the "Matrix" object in order to fill it with data. This method has three parameters each of which is an array of System.Object type. The first parameter is a column value, the second one is the row value, and the third one is the cell value. Note that the number of values in every array should comply with the object's settings! In our case an object has one level in column, row and cell, correspondingly we supply one value for columns, one for rows and one for cells.

When we run the report, we will see the following:

	1	2	3	4	5	6	7	8	9	10	Total
1	1										1
2		2									2
3			3								3
4				4							4
5					5						5
6						6					6
7							7				7
8								8			8
9									9		9
10										10	10
Total	1	2	3	4	5	6	7	8	9	10	55

Let us demonstrate how to add a value "21" to the matrix, at the intersection of column 7 and row 3. For that, change a code in the following way:

```
private void Matrix1_ManualBuild(object sender, EventArgs e)
{
    object[] columnValues = new object[1];
    object[] rowValues = new object[1];
    object[] cellValues = new object[1];

    for (int i = 1; i <= 10; i++)
    {
        columnValues[0] = i;
        rowValues[0] = i;
        cellValues[0] = i;

        Matrix1.AddValue(columnValues, rowValues, cellValues);
    }

    columnValues[0] = 7;
    rowValues[0] = 3;
    cellValues[0] = 21;
    Matrix1.AddValue(columnValues, rowValues, cellValues);
}
```

As a result, we have the following:

	1	2	3	4	5	6	7	8	9	10	Total
1	1										1
2		2									2
3			3				21				24
4				4							4
5					5						5
6						6					6
7							7				7
8								8			8
9									9		9
10										10	10
Total	1	2	3	4	5	6	28	8	9	10	76

As seen, the matrix automatically calculates the totals.

You can use the "ManualBuild" event handler for the matrix which is connected to data. In this case, event handler is called first, then the matrix is filled with data from the data source.

## Interactive reports

A FastReport's prepared report can be made interactive. This means that, it will react to the user's actions in the preview window. You can use the following interaction:

- when clicking on the report object, some kind of operation is performed. For example, you can run detailed report and show it in a separate window;
- preview window can show the report outline, which can be used for navigating on the report.

## Hyperlink

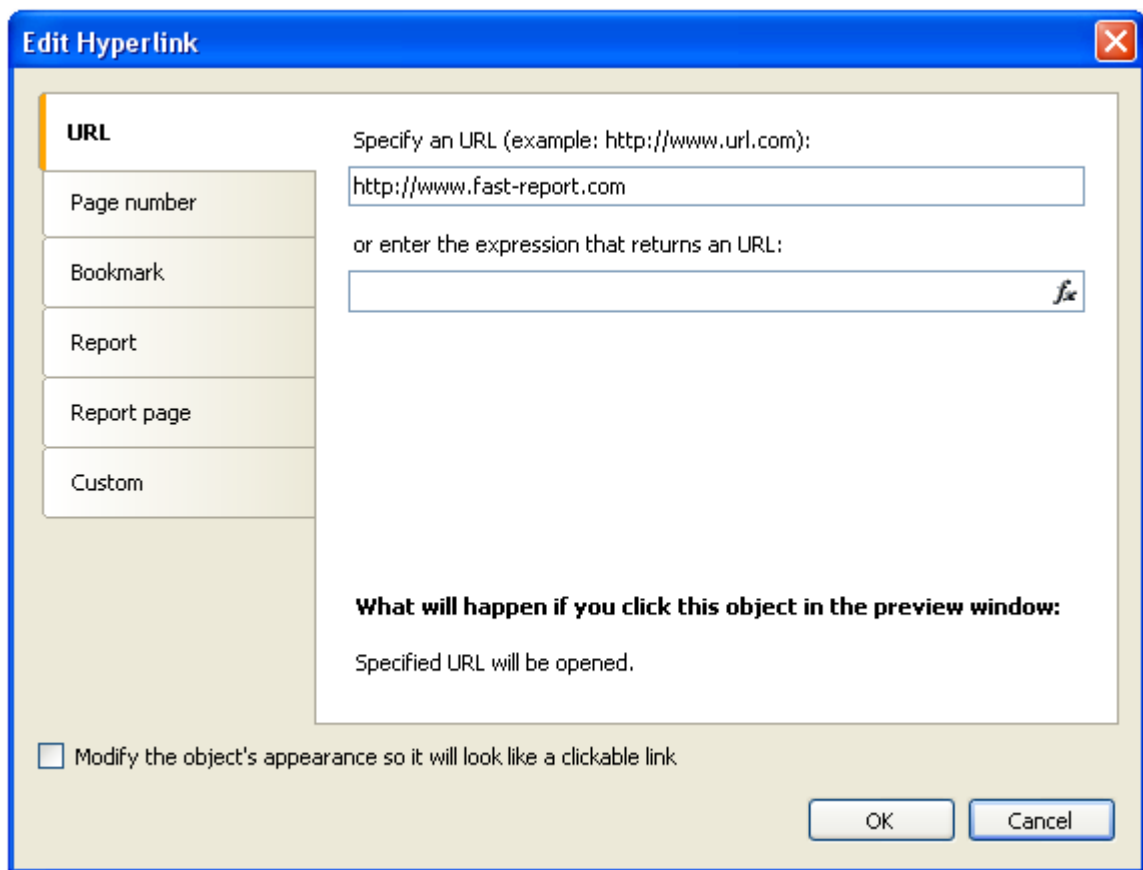
Almost all report objects have the "Hyperlink" property. Using this property, you can define an object's reaction to the mouse click in the preview window.

When clicking such an object, one of the following can occur:

- navigate to the URL address;
- send e-mail;
- execute any kind of system command;
- navigate to the report page with the indicated number;
- navigate to the bookmark, defined in another report object;
- run detailed report in a separate preview window;
- custom action, defined in a script.

## Hyperlink configuration

To configure a hyperlink, select the object which you want to make interactive, and right click on it. In the context menu, select the "Hyperlink..." item. The hyperlink editor window will open:



Choose the type of hyperlink by selecting the tab in the left side of the window. After you have done, you may click the "Modify the object's appearance..." checkbox at the bottom of the window. The appearance of the object will change in the following way:

- blue color will be set for the text and it will underlined;
- a hand cursor form will be set.

In some cases hyperlink needs to be shown in the preview window, but there is no need to print it. This is easy to do, if you disable the "Printable" object property. This can be done in the "Properties" window.

## Link to the URL

Using this type of link, you can:

- navigate to the given internet address;
- execute some kind of system commands, for example, "mailto:" for sending an email.

When clicking on the link of this type, the System.Diagnostics.Process.Start method is executed with the link's value as a parameter.

You can indicate the value of the link by using two methods:

- indicate the value directly, for example, "http://www.fast-report.com";
- indicate an expression, which returns the value of the link. This expression will be calculated when you run the report.



## Link to the page number

By using this link type, you can organize navigation on the pages of a prepared report. Most often, navigation to the first page is used. For this, indicate the page number (1 in the given case) as a link value.

You can indicate the page number by using two methods:

- indicate the number directly, for example,1;
- indicate an expression, which returns the page number. This expression will be calculated when you run the report.

## Link to a bookmark

By using this type of link, you can navigate to a bookmark, defined in another report object.

For those who know the HTML language, it is enough to say that a bookmark acts like an anchor. A bookmark has got a name and a definite position in a prepared report (page number and position on the page). When moving to a bookmark by its name, you navigate onto the indicated position.

In order to use this link type, you first need to define the bookmark. In order to do this, select the object, where you want to move when you click on the link. Almost all report objects have the "Bookmark" property. Changing this property can be done with the help of the "Properties" window.

The "Bookmark" contains an expression, which you can use in the following way:

- indicate the bookmark name as a string:

*"MyBookmark"*

- indicate an expression, which returns the name of the bookmark. For example, you can use a data column as an expression. The value of the expression will be calculated when the report is run.

After the bookmark has been defined, you can indicate its name in the hyperlink configurations window. This can be done by using two methods:

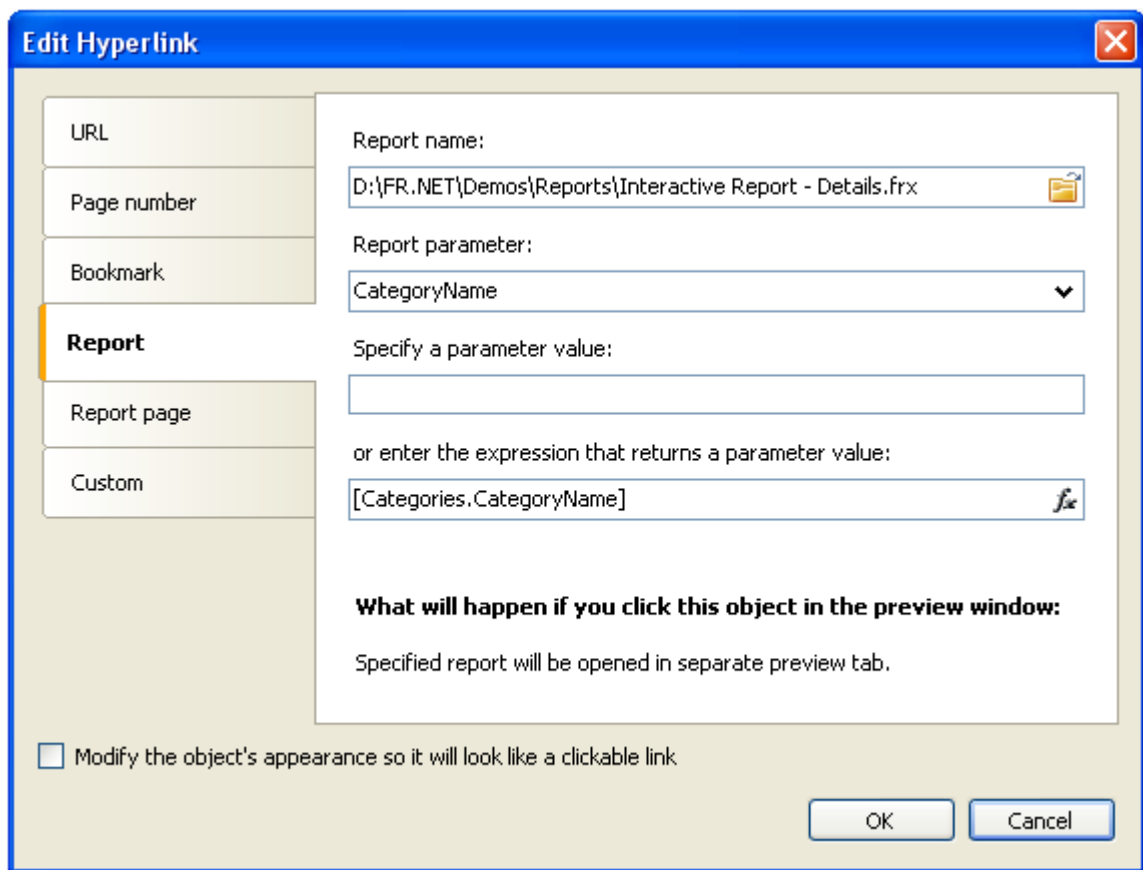
- indicate the name of the bookmark directly;
- indicate an expression which returns the name of the bookmark. For example, this can be a data column. This expression will be calculated when the report is run.

## Link to a detailed report

Using this link type, you can execute another report and show it in a separate preview window.

You must set the following parameters for this type of hyperlink:

- detailed report's name;
- name of the report's parameter, which will take the hyperlink's value;
- hyperlink value.



When the link is clicked, the following will take place:

- the indicated report will be loaded;
- the report's parameter will be set to the hyperlink's value;
- the report will be built and run in a separate preview window.

Report parameter's value can be indicated by using two methods:

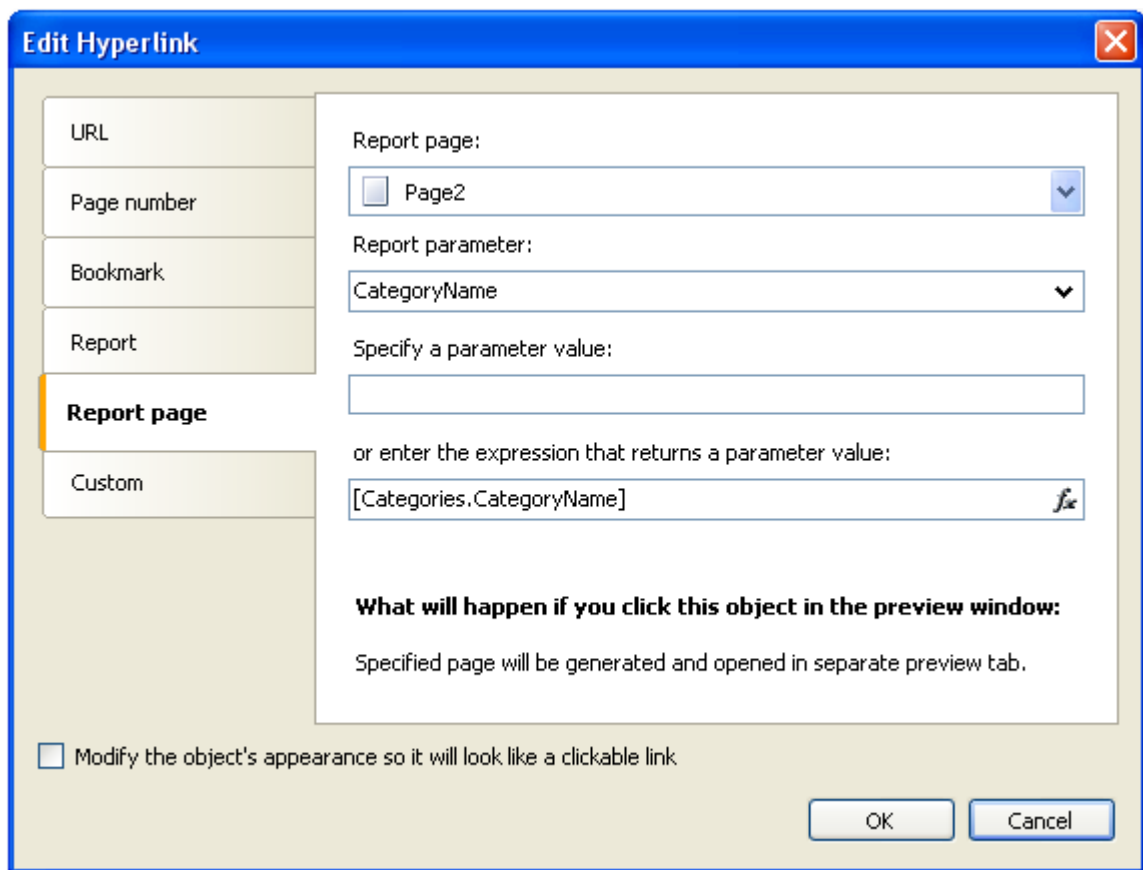
- indicate the value directly;
- indicate an expression, which returns the value. This expression will be calculated when the report is run.

### Link to a detailed page

This link type works in the same way, except that, another page in the current report is used as a detailed report. For this, your report must contain at least two pages: one with the main report, another with detailed one.

You must set the following parameters for this type of hyperlink:

- page name in that report;
- name of the report's parameter, which will take the hyperlink's value;
- hyperlink value.



When the link is clicked, the following will take place:

- the report's parameter will be set to the hyperlink's value;
- the indicated report page will be built and shown in a separate preview window.


Report parameter's value can be indicated by using two methods:

- indicate the value directly;
- indicate an expression, which returns the value. This expression will be calculated when the report is run.

When you choose a report page, its "Visible" property resets to **false**. This means that, when the main report will be built, this page will be skipped.

## Custom link

Using this type of link you can define own reaction to the clicking of the mouse. For this, use the "Click" event handler of the object. To do this:

- select the object and open the "Properties" window;
- Click the  button to show the object's events;
- double click on the "Click" event. FastReport switches to the "Code" window and creates an empty event handler.

In the handler's code, do everything what you need. You, most likely, will need a link to the object, which the handler calls, and a hyperlink's value. Use the handler's parameter sender:

```

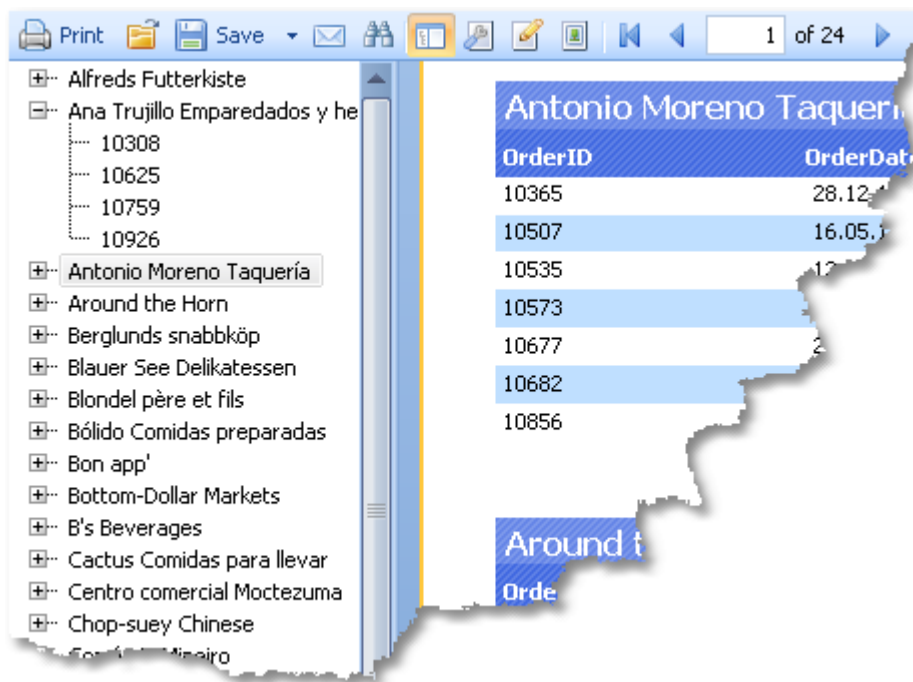
private void Text2_Click(object sender, EventArgs e)
{
    // sender - this is the object which was clicked.
    // In order to receive the value of the hyperlink, you need
    // to cast the sender to ReportComponentBase type.
    object hyperlinkValue = (sender as ReportComponentBase).Hyperlink.Value;

    MessageBox.Show("Hyperlink value = " + hyperlinkValue.ToString());
}


```

## Report outline

Report outline (also known as the "document map") is a TreeView control displayed in the preview window:



This control shows a tree structure, which was formed during the report building. If you click the tree element, you will navigate to the corresponding report element.

If the report has got an outline, it will be shown automatically. You can show or hide the outline by clicking the  button on the toolbar. The report does not create an outline automatically - you should take care about this.

The report page and all its bands have the "OutlineExpression" property. To fill the outline, indicate an expression which returns the element's text in this property. This expression will be calculated when printing a band, and its value will be added to the outline. If your report is of master-detail or group type, the structure of the outline will be similar to the report's structure.

The "OutlineExpression" property can be set in the "Properties" window.

Here are the recommendations on how to configure the outline for different types of reports:

- if you want to show the sheets of a prepared report in the outline, set the

"OutlineExpression" property of the report page. The expression will return the number of the page:

*[PageN]*

- in the "Simple list" report type with one "Data" band, set the "OutlineExpression" property of that band. As an expression, use any data column which is printed in the band;
- in the master-detail report type with two "Data" bands, set the "OutlineExpression" property of the corresponding bands. For example, in the "Category/Product" report type, the "OutlineExpression" for the first band will contain the name of the category, for the second - product's name;
- in the group report, configure the "OutlineExpression" property of the group header and a "Data" band. As an expression for the group header, use the grouping condition. For the "Data" band, use any data column which is printed in the band.

## Examples

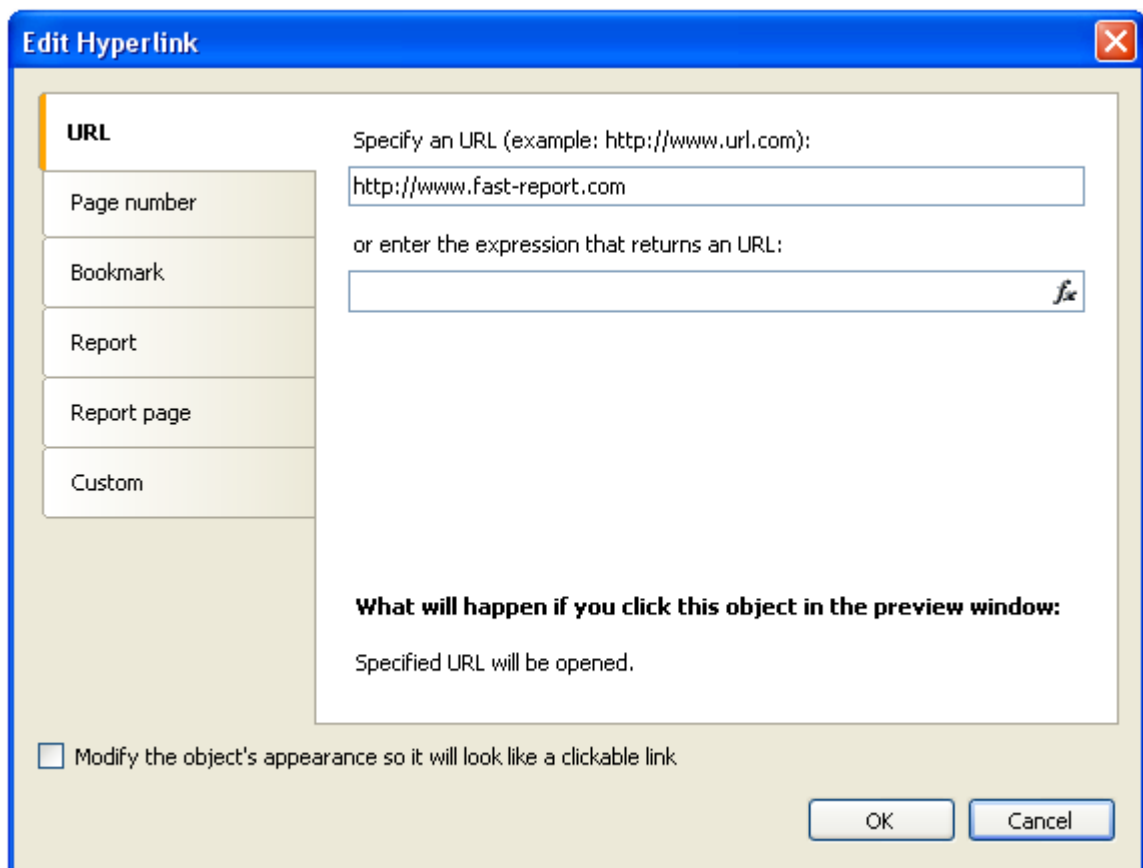
### Example 1. Link to a web page

In this example, we will create a simple report with one "Text" object. When clicking on the object in the preview window we will move to the FastReport web-page.

Create a new report, and add the "Text" object. Write the following text into it:

*Go FastReport home page*

Right click on the object and select the "Hyperlink..." item in the context menu. Configure the link in the following way:



After this, enable the "Modify the object's appearance..." checkbox, in order to apply some link attributes (blue text color, underlining and a hand-like cursor) to the object.

Run the report and click on the object. The web-browser window opens, and you will move to the FastReport home page.

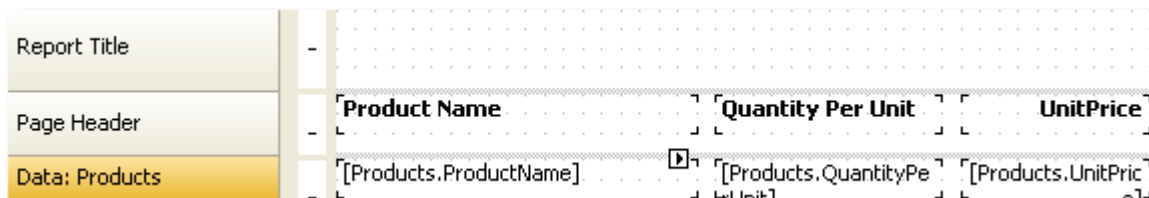
## Example 2. Building a detailed report

In this example we will build a report that displays the category list. When clicking on the category name, a detailed report that contains the list of products' in the given category will be shown.

You need to do the following:

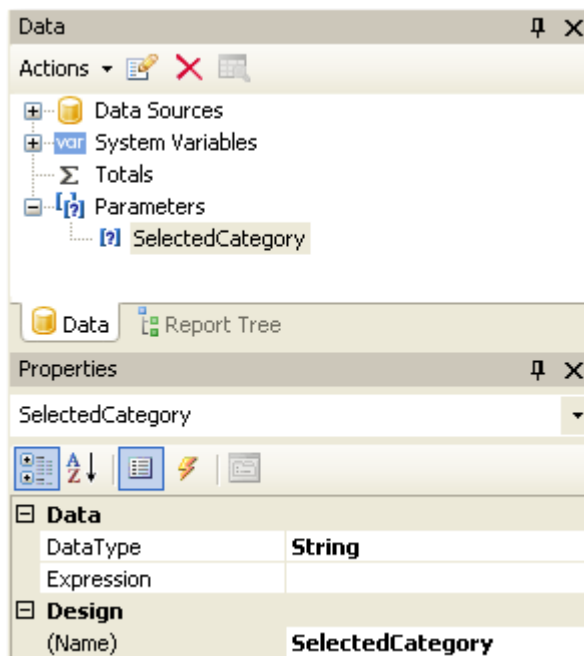
- first create a detailed report;
- define the report parameter which identifies a category;
- set up data filtration on this parameter;
- create the main report;
- in the main report, configure the hyperlink such that, the detailed report can be run with the parameter set to the chosen category.

Firstly, we will create a detailed report which prints a list of products. For this, create a new report and choose the "Products" table as a data source. Place the objects in the following way:

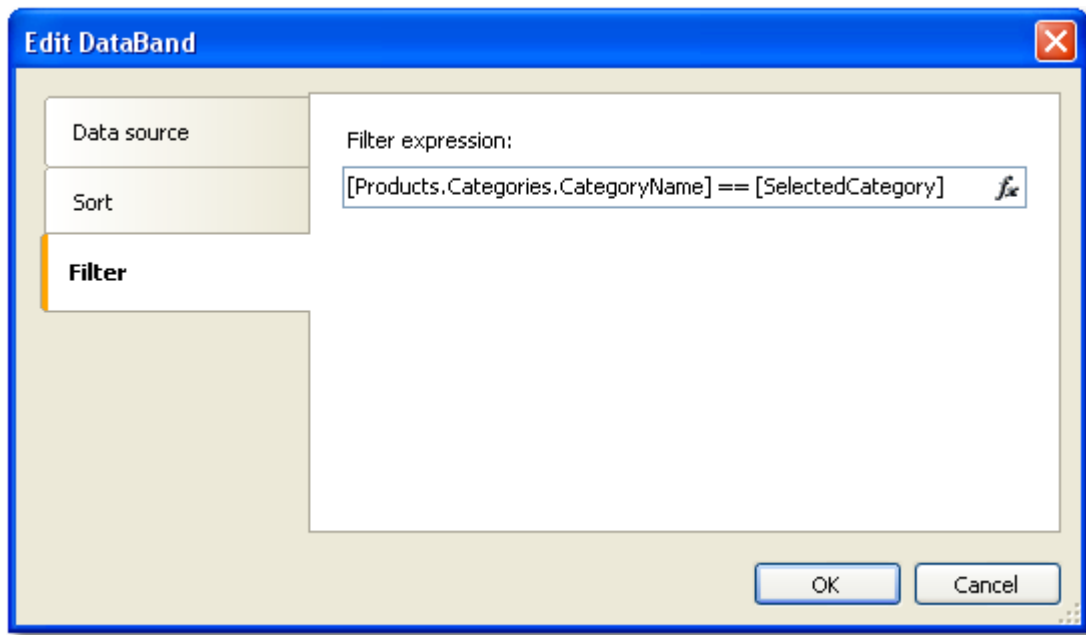


Report Title	-						
Page Header	-						
Data: Products	-						
	<table border="1"><thead><tr><th>Product Name</th><th>Quantity Per Unit</th><th>UnitPrice</th></tr></thead><tbody><tr><td>[Products.ProductName]</td><td>[Products.QuantityPe</td><td>[Products.UnitPric</td></tr></tbody></table>	Product Name	Quantity Per Unit	UnitPrice	[Products.ProductName]	[Products.QuantityPe	[Products.UnitPric
Product Name	Quantity Per Unit	UnitPrice					
[Products.ProductName]	[Products.QuantityPe	[Products.UnitPric					

Create the parameter which will be used to pass a selected category from the main report to the detailed one. For category identification, we will use the "CategoryID" column which is contained in both "Categories" and "Products" tables. Configure the parameter in the following way:



Now we need to set data filtering to filter all products that belong to the specified category. To do this, double click the "Data" band. Switch to the "Filter" tab and indicate the following condition:



Now create the main report. Create a new report and choose the "Categories" table as a data source. Place the objects in the following way:



Right click on the "Text" object and select the "Hyperlink..." menu item. Set up the link in the following way:



**Edit Hyperlink**

URL

Page number

Bookmark

**Report**

Report page

Custom

Report name:  
detailReport.frx

Report parameter:  
SelectedCategory

Specify a parameter value:  
[Categories.CategoryName]

or enter the expression that returns a parameter value:  
[Categories.CategoryName]

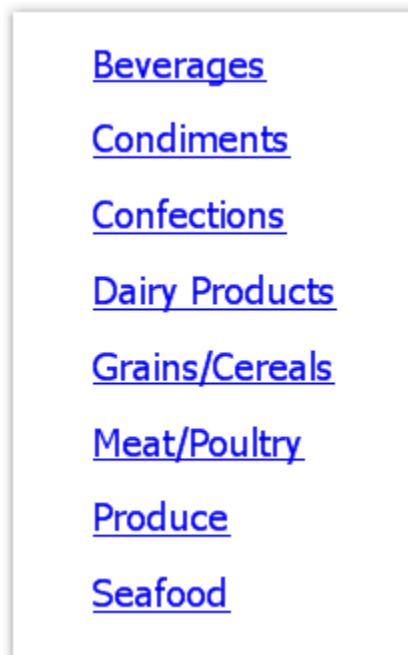
**What will happen if you click this object in the preview window:**  
Specified report will be opened in separate preview tab.

Modify the object's appearance so it will look like a clickable link

OK Cancel

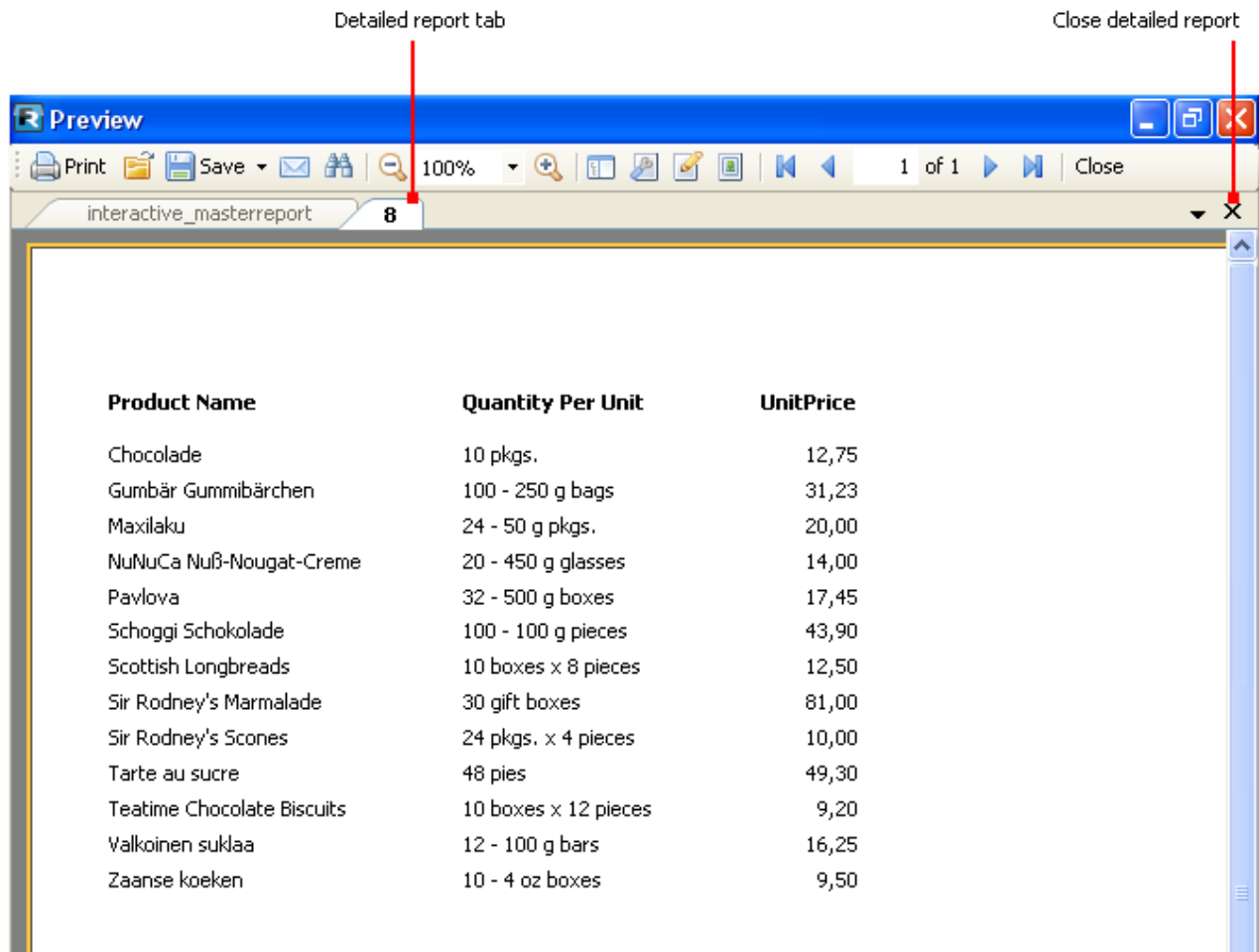
As a report name, choose the name of the detailed report file. Report parameter can be chosen from the drop-down list, by pressing the button on the right side of the list. As a parameter value, indicate the "[Categories.CategoryID]" expression.

Run the report, and you will see the categories list:



If you click on one of the categories, a detailed report will be built. It will be shown on a

separate tab of the preview window:



As seen on the picture, the title of the tab is set to the hyperlink's value. In our case, this is the numeric value contained in the "CategoryID" data column. This appears not informative and not beautiful. Let's change our report to use the category name instead of its number. For this, do the following:

In the detail report:

- change the parameter's "DataType" property to the "String";
- add the "Categories" data source into the report. It will be used for referring to the "CategoryName" column when filtering data;
- change the filtering expression of the "Data" band:

```
[Products.Categories.CategoryName] == [SelectedCategory]
```

In the main report:

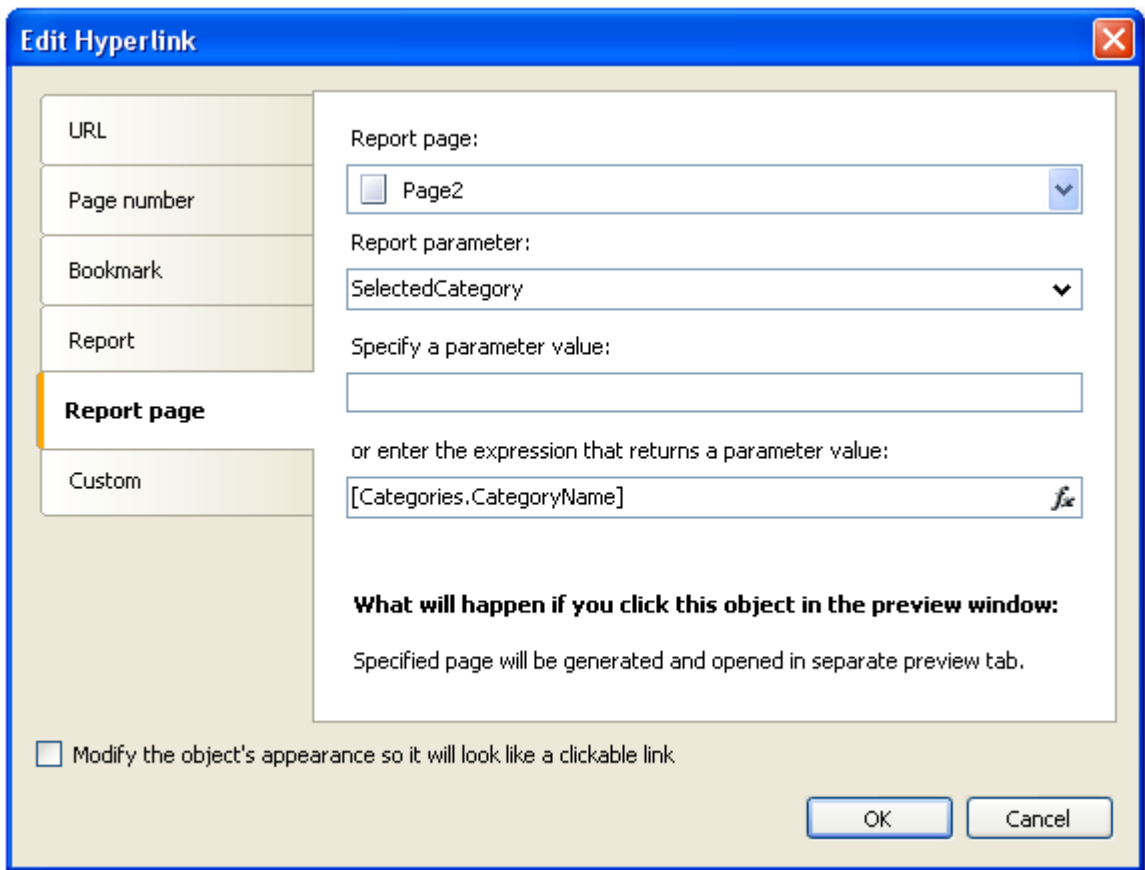
- change the hyperlink settings. Now we will pass the "[Categories.CategoryName]" value into the report parameter.

If we run the report now, we will see that the title of the tab is set to category name. We can improve the detailed report a little. Add the "Text" object, which will print the name of the chosen category in the report title:

The screenshot shows a 'Preview' window with a toolbar at the top containing icons for Print, Save, and navigation. The report title is 'Confections'. Below the title is a table with the following data:

Product Name	Quantity Per Unit	UnitPrice
Chocolade	10 pkgs.	12,75
Gumbär Gummibärchen	100 - 250 g bags	31,23
Maxilaku	24 - 50 g pkgs.	20,00
NuNuCa Nuß-Nougat-Creme	20 - 450 g glasses	14,00
Pavlova	32 - 500 g boxes	17,4
Schoggi Schokolade	100 - 100 g pieces	43,0
Scottish Longbreads	10 boxes x 8 pieces	
Sir Rodney's Marmalade	30 gift boxes	
Sir Rodney's Scones	24 pkgs. x 4 pieces	
Tarte au sucre	48 pies	
Teatime Chocolate Biscuits	10 boxes x 12 pieces	
Valkoinen suklaa	12 - 100 g bars	
Zaanse koeken	10 - 4 oz boxes	

While we are working with this example, we have created two reports and swap between them several times. This is not very comfortable. In order to make the task easier, two reports can be placed into one: the main report will be on the first page, the detail one on the second page. In this case the hyperlink needs to be set in the following way:



In the given case, we need to choose Page2 as the detail report page.

### Example 3. Interactive "Matrix" object

In this example we will see how to build a detailed report if we click on the cell of the "Matrix" object. As an example, we will use a matrix which displays the sales of employees grouped by year.

As a data source for the matrix, the "MatrixDemo" table is used. It presents the sales of the employees, grouped by year and month:

Name	Year	Month	ItemsSold	Revenue
Nancy Davolio	1999	2	1	1000
Nancy Davolio	1999	11	1	1100
Nancy Davolio	1999	12	1	1200
Nancy Davolio	2000	1	1	1300
Nancy Davolio	2000	2	2	1400
Nancy Davolio	2001	2	2	1500
Nancy Davolio	2001	3	2	1600
Nancy Davolio	2002	1	2	1700

Andrew Fuller	2002	1	2	1800
Andrew Fuller	1999	10	2	1900
Andrew Fuller	1999	11	2	2000
Andrew Fuller	2000	2	2	2100
Janet Leverling	1999	10	3	3000
Janet Leverling	1999	11	3	3100
Janet Leverling	2000	3	3	3200
Steven Buchanan	2001	1	3	4000
Steven Buchanan	2001	2	4	4100
Steven Buchanan	2000	1	4	3999

Configure the matrix in the following way:

- put the "MatrixDemo.Year" data column in the column header;
- put the "MatrixDemo.Name" data column in the row header;
- put the "MatrixDemo.Revenue" data column in the cell.

A prepared matrix will be as follows:

	Year				
Employee	1999	2000	2001	2002	Total
Andrew Fuller	3 900,00	2 100,00		1 800,00	7 800,00
Janet Leverling	6 100,00	3 200,00			9 300,00
Nancy Davolio	3 300,00	2 700,00	3 100,00	1 700,00	10 800,00
Steven Buchanan		3 999,00	8 100,00		12 099,00
<b>Total</b>	13 300,00	11 999,00	11 200,00	3 500,00	39 999,00

As seen, the value of a cell is the sum of employee's sales for the whole year. Let us create a detailed report which will be displayed when we click the cell. In our case the detailed report can contain the sales of a selected employee for every month of a selected year.

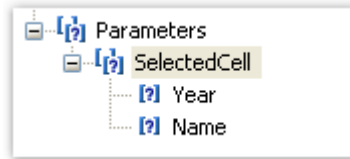
How to connect a cell with data, on which basis it was printed? Each cell of the matrix has got its own address. This is a combination of the values from the column and row headers. In our example, the address of the cell is a combination of the year and name of the employee. Exactly this data can be passed to the detailed report. How can this be done? Very simple: set the hyperlink, showing only the report name and name of the parameter. Parameter values do not need to be indicated: for a matrix cell, FastReport itself forms the value and passes it into the parameter.

Assuming that, we have clicked on the top left cell, containing the number 3900. This is the sum of the sales of the employee named "Andrew Fuller" for the year 1999. What form is used to pass this value into the parameter? FastReport combines column and row values, by using a separator:

*1999;Andrew Fuller*

Does it mean that we must extract the value of the year and the name of the employee from

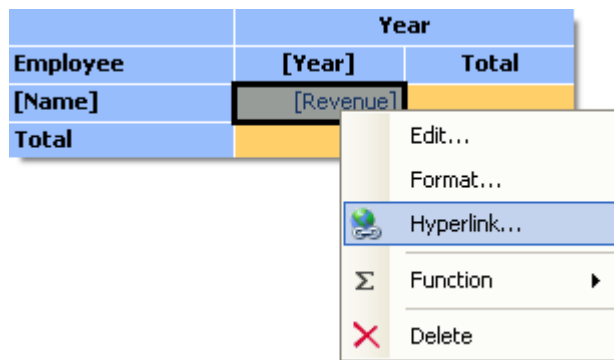
this string, convert the year into the int, and use these values for data filtering? No, it's much simpler. All that we need to do is to create a parameter that has **nested parameters**. You can learn about this in the "[Data](#)" chapter. In the given case, parent parameter can be like this:



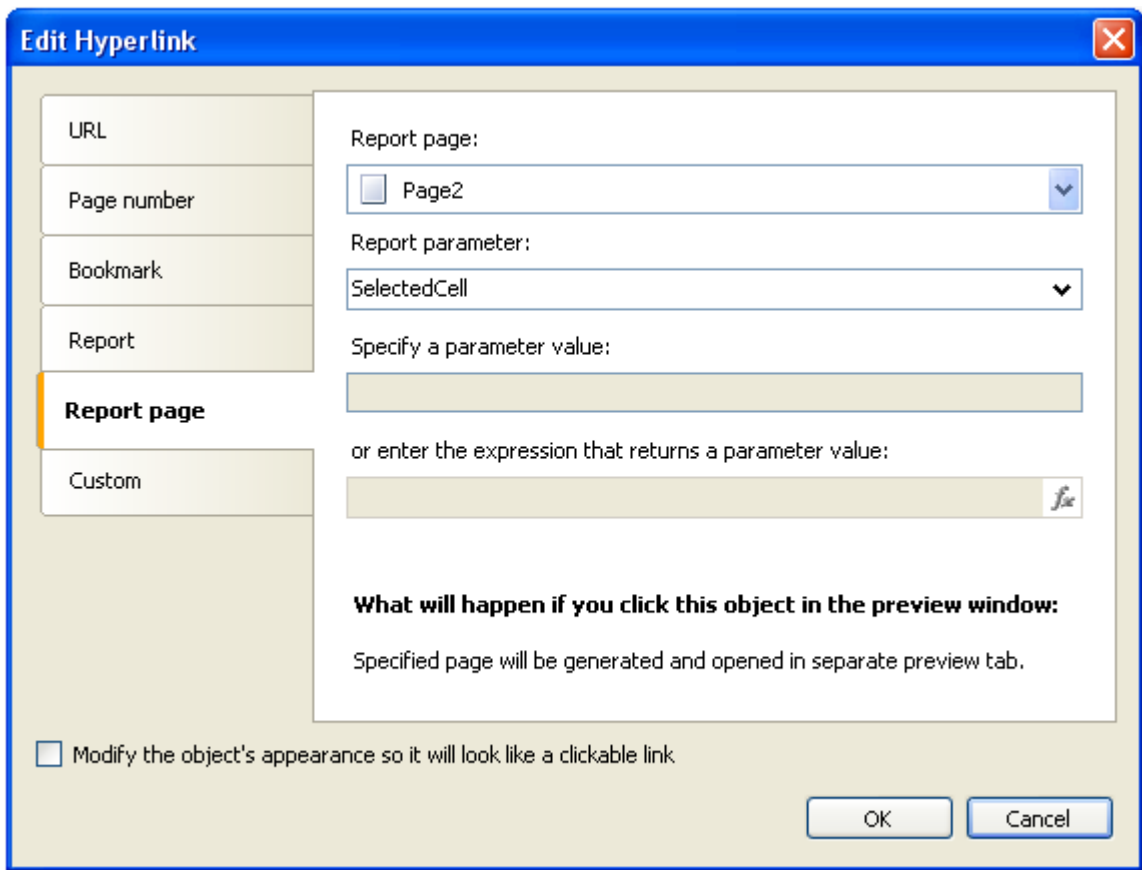
When creating the parameter, consider the following moments:

- you don't need to set up the parent parameter. Just give it the name;
- a parent parameter must have as many nested parameters, as there are values passed from the matrix. In the given case, there are two values;
- order of the nested parameters must correspond with the order of the values passed from the matrix. In our case, the year will be passed in the first parameter, and the employee name will be passed in the second parameter;
- nested parameters can be named as you wish, but it is better to give them names which correspond with the names of the matrix elements;
- it is very important to set the data type for every nested parameter correctly. Data type must correspond with the value, which is passed into the parameter. In our case, the first parameter (Year) must be an integer type, and the second (employee name) - String.

After we have clarified all the needed things, we will create the report. Select the cell of the matrix and call the hyperlink editor:



In the hyperlink configuration, indicate the parent parameter as a report parameter (in our example - "SelectedCell"):



FastReport passes the values into the SelectedCell.Year and SelectedCell.Name nested parameters. These values will be converted into data types, indicated in the parameter configuration - this is why it is important to configure parameter data types correctly.

Detailed report is placed on a separate page of the main report and uses the same data source:

Report Title	-	[SelectedCell.Name] sales in [SelectedCell.Year] year
Page Header	-	Month Revenue
Data: MatrixDemo	-	[MatrixDemo.Mon] [MatrixDemo.Rev]

In order to show the sales of a chosen employee for a chosen year, set up the filtering. For this, open the "Data" band editor and indicate the following filtering condition:

`[MatrixDemo.Year] == [SelectedCell.Year] && [MatrixDemo.Name] == [SelectedCell.Name]`

The report is ready. Run it for execution and click on the top left cell. A detailed report will be opened, having the following data:

## Andrew Fuller sales in 1999 year

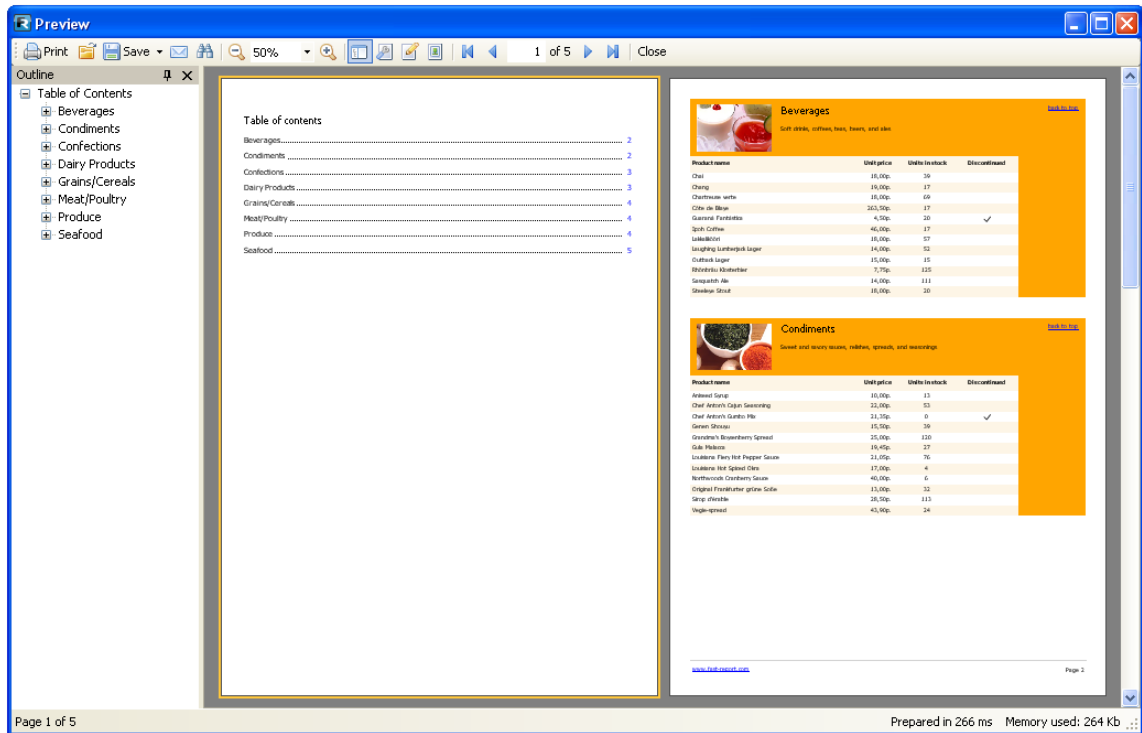
Month	Revenue
10	1 900,00
11	2 000,00

As seen, the sum of the values (1900+2000) corresponds with the cell of the matrix, on which we clicked.

### Example 4. Report with table of contents, navigation and outline

In this example, we will look at creating a report, which has the following features:

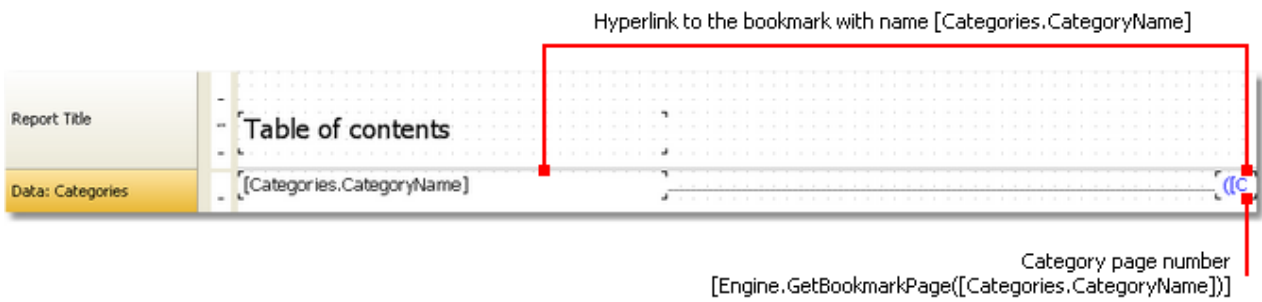
- on the first page it will print the "Table of Contents" (TOC) which is interactive, i.e. you can click its elements to navigate to the corresponding page;
- in the preview window, it will display the outline, which is interactive as well.



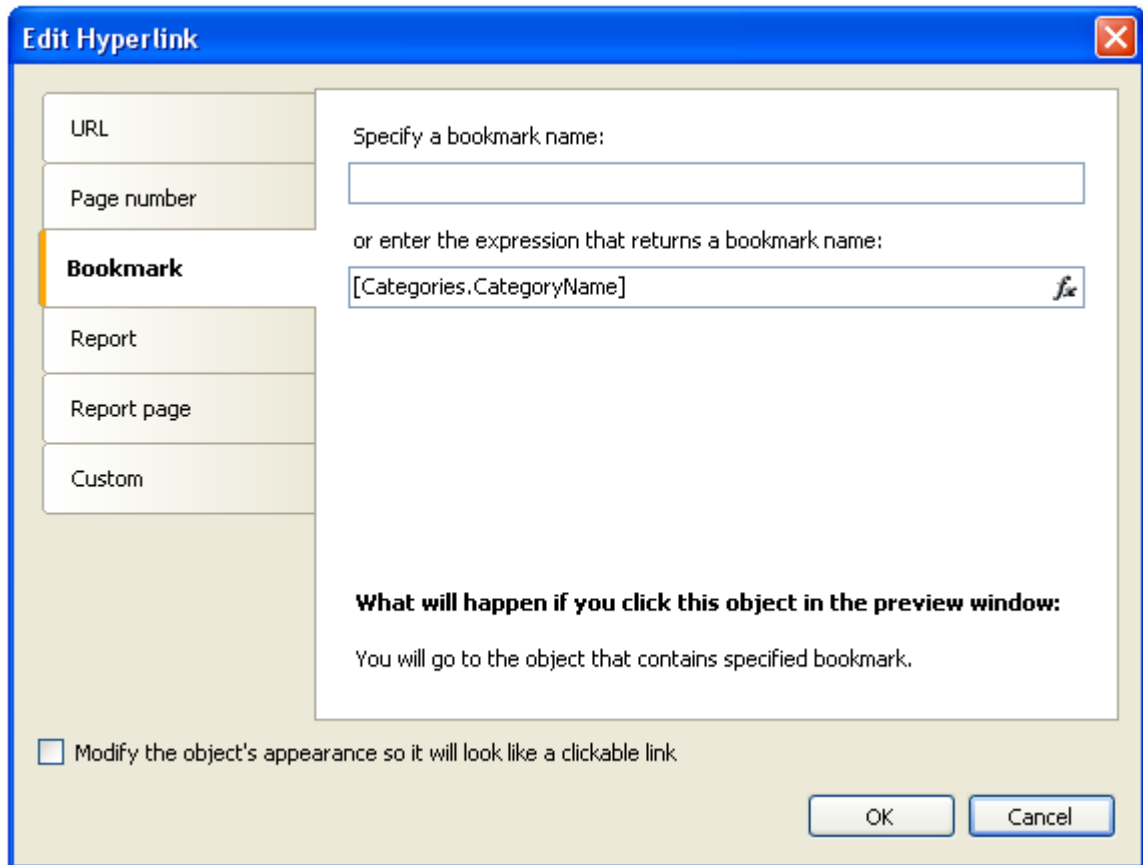
The report will use the "Categories" and "Products" tables. In the TOC, we will print the categories list. The rest of report will print the categorized list of products. Report template will be made up of two pages: the first page will be used to print the TOC; the second one is the main part of the report.

We will discuss the TOC firstly. Create a new report and add "Categories" and "Products" data sources into it. Connect the "Data" band to the "Categories" table and place the objects in the following way:





In order to make the TOC objects interactive, configure its "Hyperlink" property:



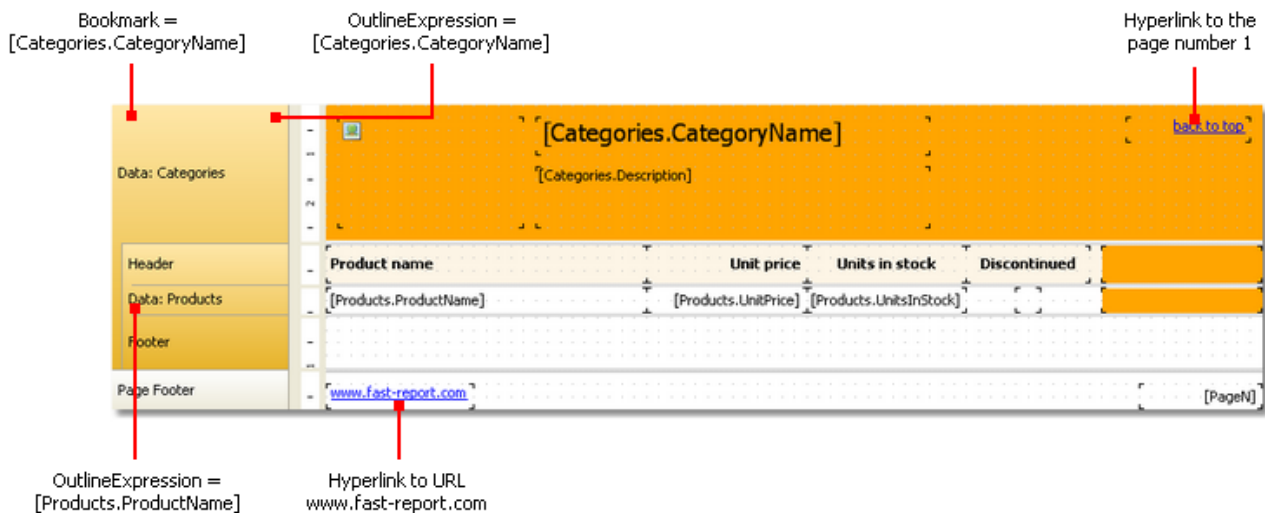
Indicate the category name as a bookmark. We will define the object's bookmark later.

In order to print the page number in the TOC, you need to do the following:

- enable the "double pass" setting of the report. This can be done in the "Report|Options..." menu. This needs to be done, because the TOC page is printed before other pages. At this moment FastReport does not know where the categories will be printed;
- use the "Engine.GetBookmarkPage" function, which returns page number for the specified bookmark. In our case, we use the "[Categories.CategoryName]" expression as a bookmark name, so the function call will be as follows:

```
[Engine.GetBookmarkPage([Categories.CategoryName])]
```

On the second page of the report, we will create a master-detail report as shown in the figure below:



Set up the bookmark we navigate to when clicking on an element in the TOC. For this, select the first "Data" band and indicate the following expression in its "Bookmark" property:

`[Categories.CategoryName]`

To set up the report outline, do the following:

- select the first report page. This can be done by switching to the page;
- in the "Properties" window, set the following value to the "OutlineExpression" property:

*"Contents"*

- switch to the second report page;
- select the first "Data" band and set its "OutlineExpression" property:

`[Categories.CategoryName]`

- select the second "Data" and set its "OutlineExpression" property:

`[Products.ProductName]`

## Report inheritance

Often we have many reports with the same data in it - for example, same header/footer with company logo and some data - email, address etc. Now imagine the situation, that you need to change some company data - for example, email. You have to do this in each report! To avoid this, you can use report inheritance. What is it?

For example, you have some common elements in each report (logo, company name, email etc). These elements are typically placed on the report title and/or page header. You can create a base report that contains only common elements. All other reports will use base report and thus will contain such common elements plus own elements defined in a report.

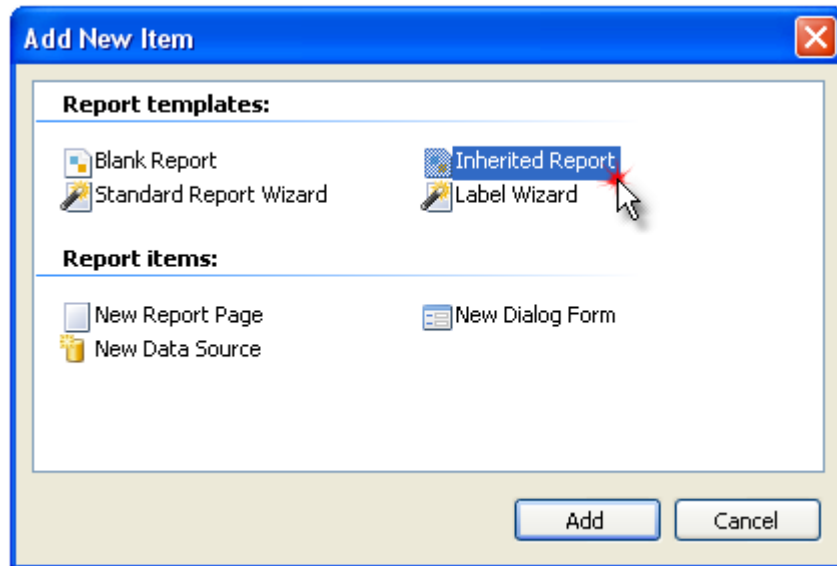
In case you need to change something (logo, email) you just open the base report and make necessary changes. All other reports that inherit from a base, will be changed automatically. In fact, when you open a report that is inherited one, the base report is opened first, then the inherited one.

## Creating a report

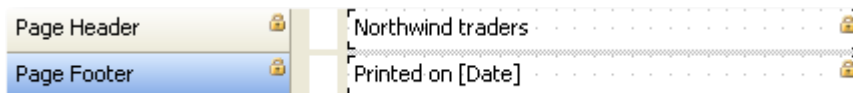
To use the inheritance, you need to do the following:

- create a base report and save it to a file;
- create a new report that inherits from base.

To create an inherited report, choose the "File|New..." menu item, then choose the "Inherited report" item in the window:



You will be asked to select a base report file. That file must be created at this moment. After that, the base report will be loaded into the designer. You can change it as you want. You see that objects from the base report are marked with the "lock" sign:



That means, you cannot delete such objects, rename it or move to another band.

You can add new objects or bands, change the object's appearance, size and location. When you have done, save the report.

## Changing the base report

Let us look what happens if we change the base report. We can:

- delete the object from the base report. This object will be deleted from the inherited report as well;
- add the object into the base report. This object will appear automatically in the inherited report;
- change the size, location, text, appearance of the object. All changes will be reflected in the inherited report, in case where this object was not changed in the inherited report.

The last point requires some explanations. Let us look at two examples of using inheritance. In the first example, we will do the following:

- create a base report which contains the Text1 object;
- create an inherited report and just save it, without changing anything;
- open the base report and move the Text1 object;
- open the inherited report and we will see that the Text1 object **is moved as well**.

In the second example, we will do the following:

- create a base report which contains the Text1 object;
- create an inherited report;
- in the inherited report, move the Text1 object to a new position and save the report;
- open the base report and move the Text1 object to different position;
- open the inherited report and we will see that the Text1 object **is not moved**.

It happens because we have changed the object in the inherited report. This change was saved in the inherited report file. Now, if we change the original object in the base report, it will be ignored in the inherited report. In this case, the new object's location will be ignored. All other changes (such as text color, for example) will still be reflected in the inherited report.

This behavior will become clear if we look at the contents of inherited report file. For example, this is how the original object is saved in the inherited report, in case this object was not changed:

```
<inherited Name="Text1"/>
```

If we change the object's location in the inherited report, it will be saved like this:

```
<inherited Name="Text1" Left="255.15" Top="28.35"/>
```

When opening the inherited report, FastReport will load all object's properties, defined in the base report, plus properties, saved in the inherited report.

## Limitations

The report inheritance was designed to meet the following goal: save common report elements such as headers and footers in separate files, and reuse them in inherited reports. Do not try to use the inheritance to perform more complex tasks. In particular, avoid to do the following:

- do not inherit the report from the inherited one (i.e. do not inherit twice);
- do not use complex objects such as Table and Matrix, in the base report;
- do not use script in the base report;
- do not use parameters in the base report.

## Reports with charts

FastReport uses Microsoft Chart library to display charts. This library will be included in .Net Framework 4.0. Now it is available as a separate download here:

<http://www.microsoft.com/downloads/details.aspx?FamilyID=130f7986-bf49-4fe5-9ca8-910ae6ea442c>

This library requires .Net Framework 3.5 SP1. It is freeware; you can learn about its features here:

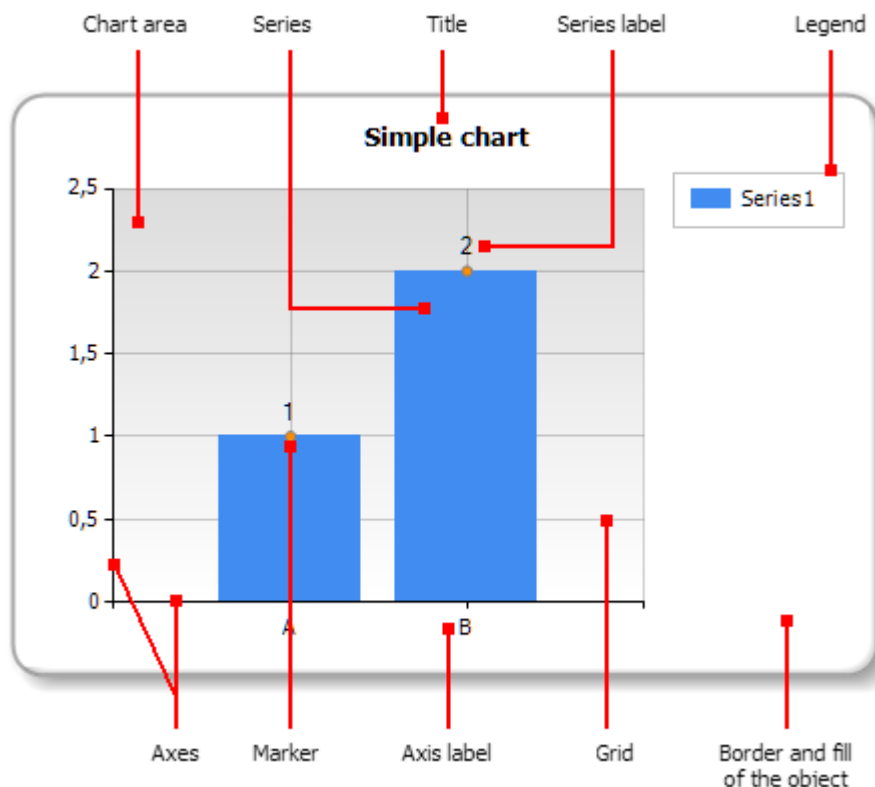
<http://code.msdn.microsoft.com/mschart>

Let us highlight some features of Microsoft Chart:

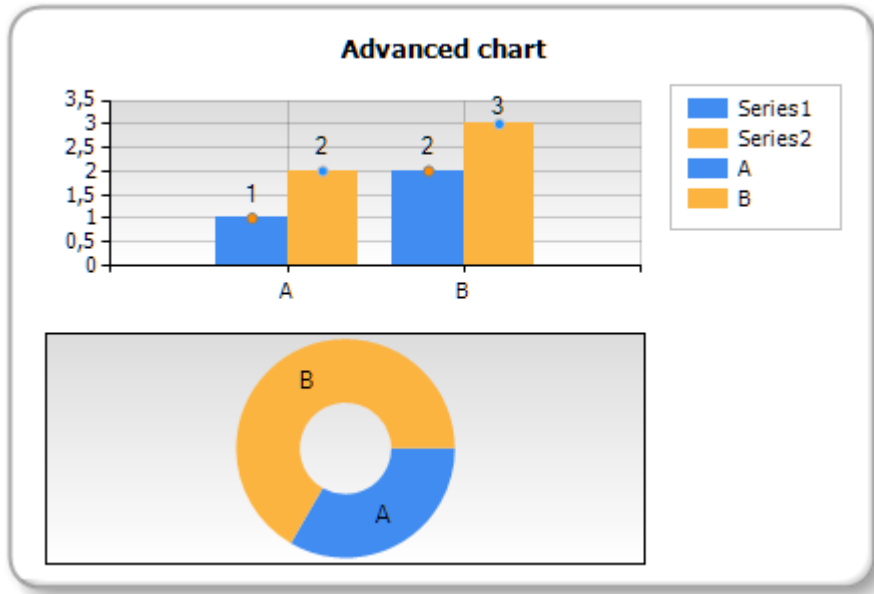
- more than 30 types of series (bars, columns, areas, lines, bubbles, pie, circular, financial, pyramidal, ranges);
- 3D support;
- supports several series of different types in one chart;
- full control over appearance and behavior of each chart element.

## Chart elements

Microsoft Chart consists of the following elements:



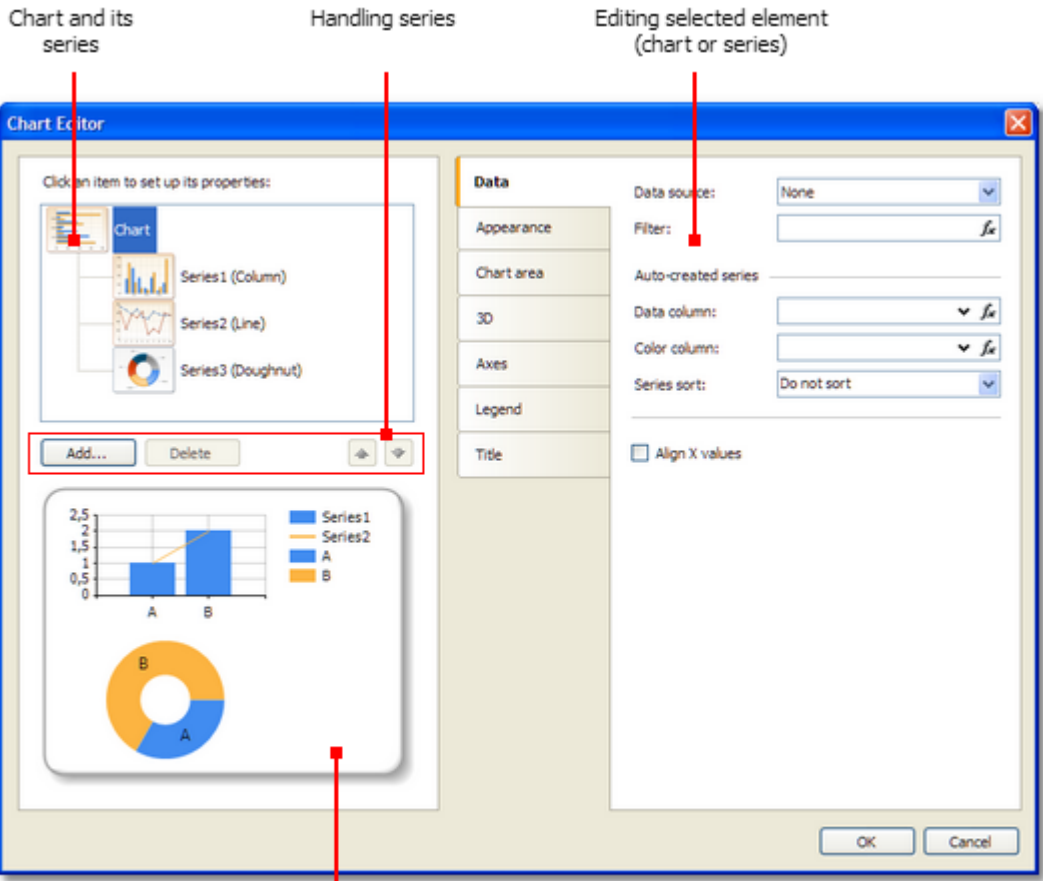
One chart may have one or several chart areas. One chart area may contain one or several series. Below you can see the chart which contains two chart areas (the first area contains two series, the second area contains one series):



Some series (for example, pie series) require exclusive chart area.

### Chart editor

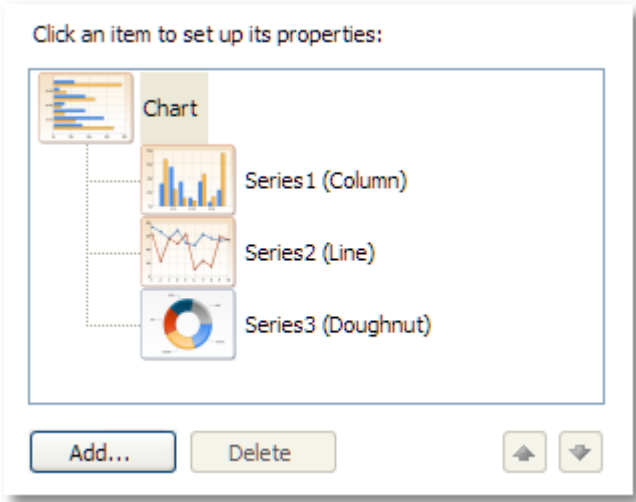
The "Chart" object contains numerous settings which can be handled in the chart editor. To invoke the editor, double click the "Chart" object:



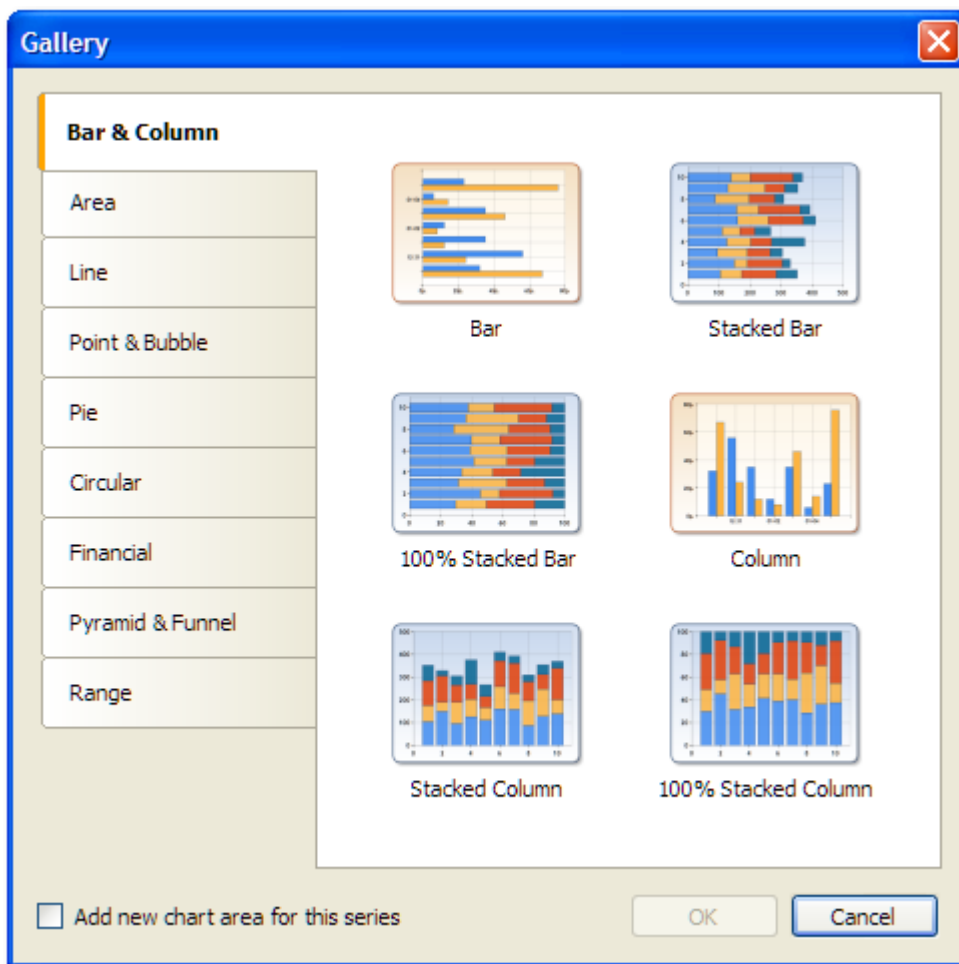
Example. Its elements are clickable

### Handling series

The "Chart" object can contain one or several series. Series list is displayed in the editor:



To add a new series, press the "Add..." button. You will see the "Gallery" dialog:



Select the needed category, then - needed series type. If you want to place the series in its own chart area, check the "Add new chart area for this series" checkbox. For some series types (such as pie, circular, financial, pyramidal) the new chart area is added automatically regardless of this checkbox state.

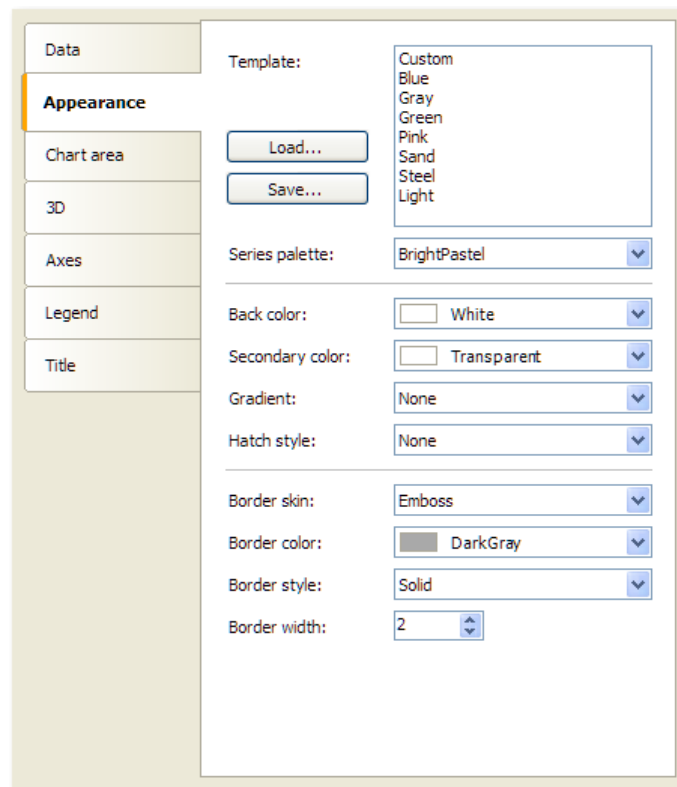
To delete the series, press the "Delete" button. To change series order, use "Up" and "Down" buttons.

### Setting up the appearance

Using the chart editor, you can set up appearance of each chart element. All properties (more than 100) are splitted in several categories. Some of them are specific to "Chart" object, while others are part of series.

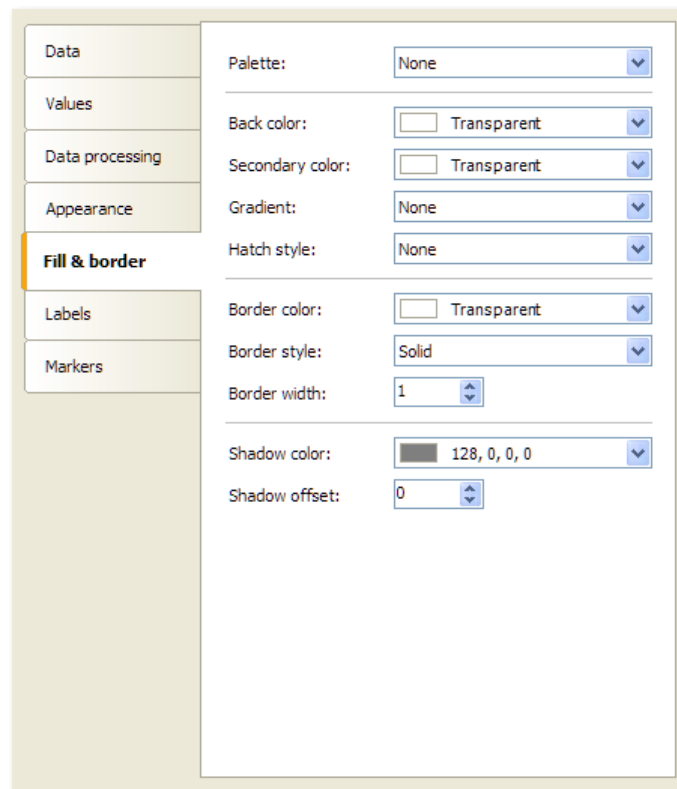
If you choose the "Chart" object from the series list, you will see the following property pages:





- "Appearance" - border and fill of the chart;
- "Chart area" - border, fill, shadow;
- "3D" - 3D settings;
- "Axes" - setup appearance of axis, its title, labels, grid, markers, custom labels and strips;
- "Legend" - style of legend, docking, border, fill, shadow and font;
- "Title" - style of title, docking, border, fill, shadow, font.

If you choose the series object from the series list, you will see the following property pages:



- "Appearance" - some settings specific to the selected series type;
- "Fill & border" - fill and border of the series values;
- "Labels" - series labels. You can choose label type, font, color and fill;
- "Markers" - series markers. You can choose marker type, its color and border.

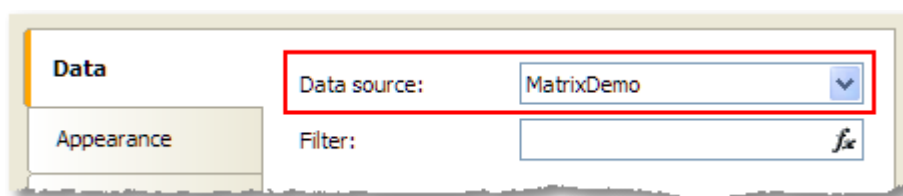
## Connecting chart to data

You can fill the chart with data in several ways:

1. Use data source. To do this, you need to indicate the data source for the "Chart" object and connect each series to data columns.
2. Use fixed values for each series.
3. Fill the object with data using the script.

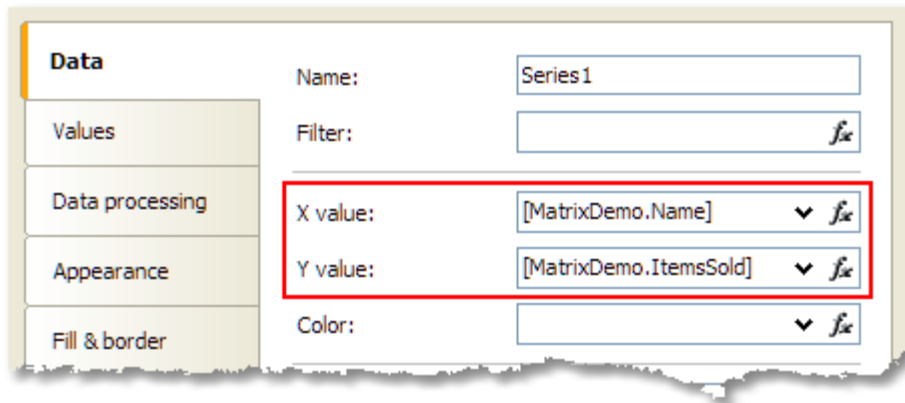
To connect the chart to a data source, follow these steps:

- select the "Chart" object in the series list;
- switch to the "Data" tab;
- choose the data source:



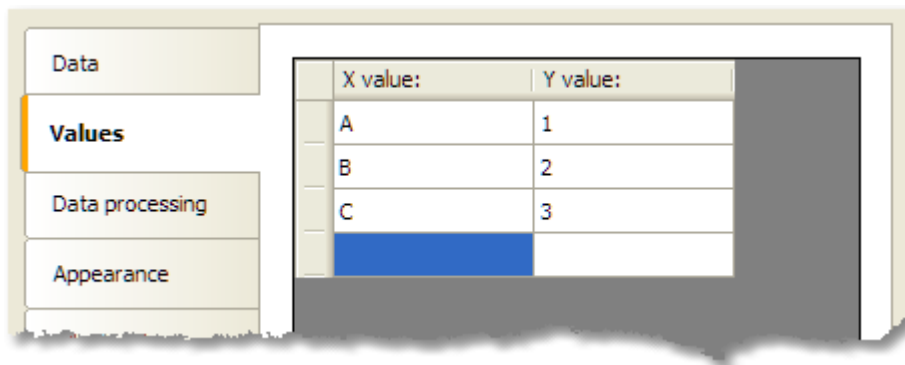
- if necessary, set the data filter expression. This filter will be applied to all chart series;
- select the series in the series list;

- switch to the "Data" tab;
- choose data columns for each series value. Depending on series type, it may have two or more values. Most series types have two values - X value and Y value:



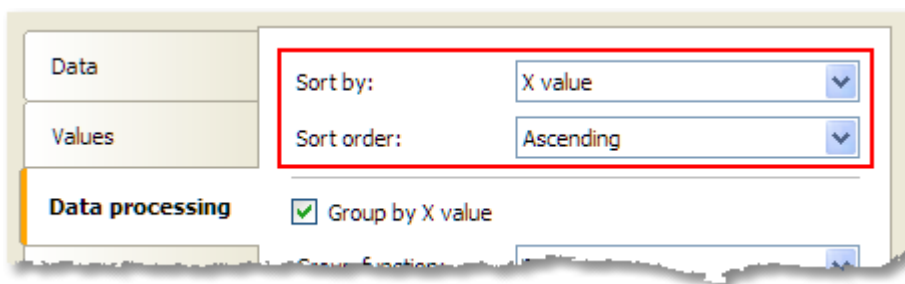
- if necessary, set the data filter expression. This filter will be applied to current series only;
- in the "Color" control, you may indicate a data column which returns a color value.

You may also provide list of values for the series. In this case, the data connection is not needed. To do this, select a series in the series list and switch to the "Values" tab. Fill the table with values:



## Sorting the data

By default, the chart object displays data in natural order. You can change the sort order; to do this, select the series from the series list and switch to the "Data processing" tab:



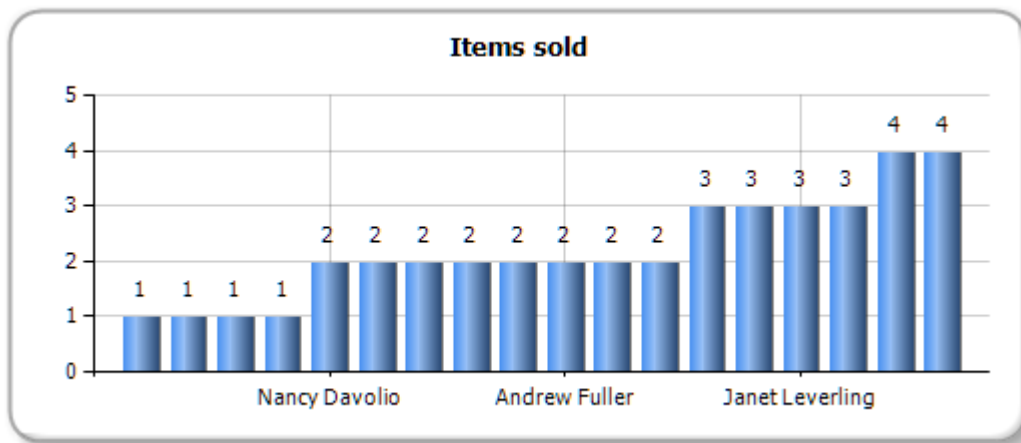
You can choose one of the sort modes - do not sort, sort by X value, and sort by Y value.

## Grouping the data

Sometimes we face a problem when series is filled with data with several identical X values. For example, the MatrixDemo table, which is used to demonstrate charts, has the following data:

Name	Year	Month	ItemsSold	Revenue
Andrew Fuller	2002	1	2	1800
Andrew Fuller	1999	10	2	1900
Andrew Fuller	1999	11	2	2000
Andrew Fuller	2000	2	2	2100
Janet Leverling	1999	10	3	3000
Janet Leverling	1999	11	3	3100
Janet Leverling	2000	3	3	3200
...				

If we try to build a chart based on this data (for example, employee's sales - set X value to Name column, Y value to ItemsSold column), we will get the following wrong result:



In this situation, we need to group the same employees into one value. To do this, select the series in the series list, and switch to the "Data processing" tab. Select the group type - "X value" and choose "Sum" as group function:

**Data processing**

Appearance

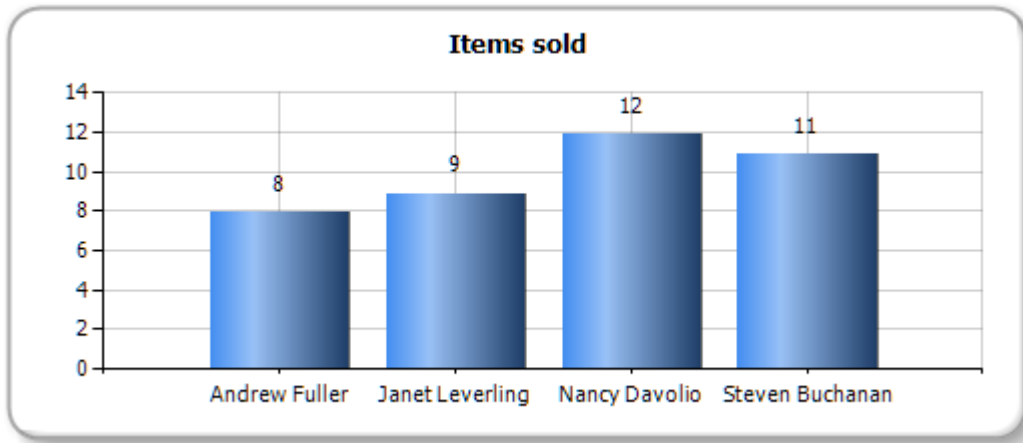
Fill & border

Group by: X value ▼

Group interval: 1,00 ▲▼

Group function: Sum ▼

As a result, all identical employees will be grouped into one value, their sales will be summarized. You will see the following result:



### Collecting the data

This instrument for data processing allows to collect several series value into one value. You can choose one of the following algorithms:

Algorithm	Description
TopN	Only top N values are displayed. All other values are collected and displayed as "others" value (you can choose the label for this value).
BottomN	Bottom N values are displayed. If the text for the collected value is not set, this value is not displayed.
Less than value	Series values less than specified value, are collected and displayed as "others" value.
Less than percent	Series values less than specified percent, are collected and displayed as "others" value.
Greater than value	Series values greater than specified value, are collected and displayed as "others" value.
Greater than percent	Series values greater than specified percent, are collected and displayed as "others" value.

For example, to display top 5 values, set up the series in the following way:

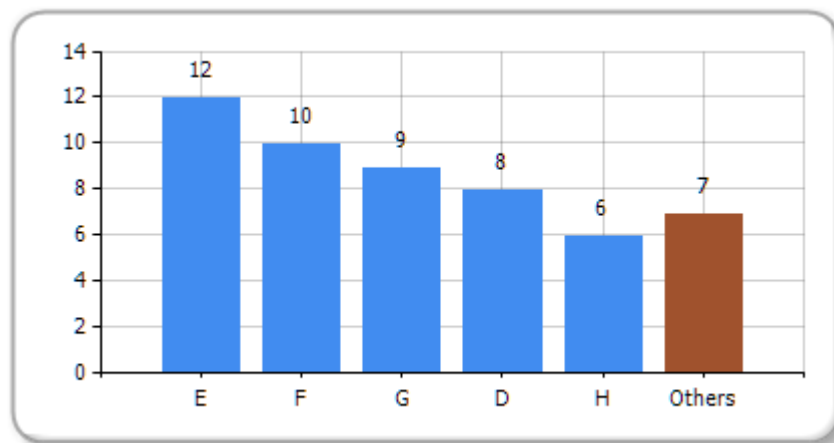
Collect data:

Value:

Collected text:

Collected color:

You will get the following result:



## Exploding the values

For pie-type series, you can explode some values. To do this, select the series in the series list and switch to the "Data processing" tab:

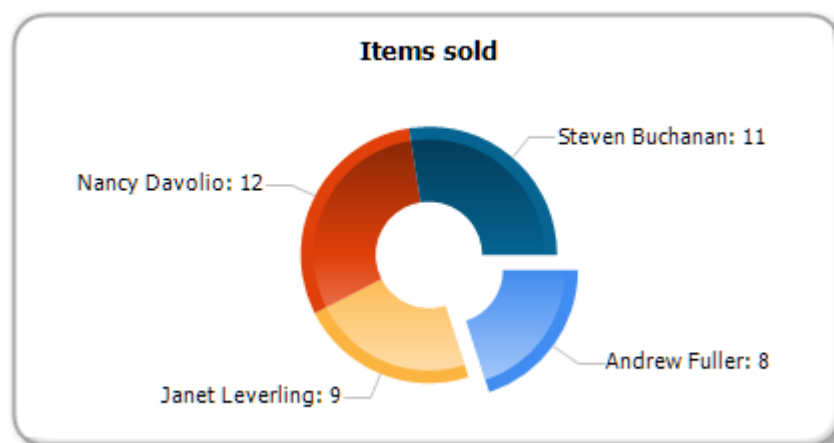
Explode:

Value:

You can use one of the following explode types: biggest value, lowest value and specific value. If you choose the latter mode, you have to specify a value which you want to explode. It may be any expression (see the ["Expressions"](#) chapter for details). For example, if you need to explode Andrew Fuller's value, use the following expression:

*"Andrew Fuller"*

You will get the following result:



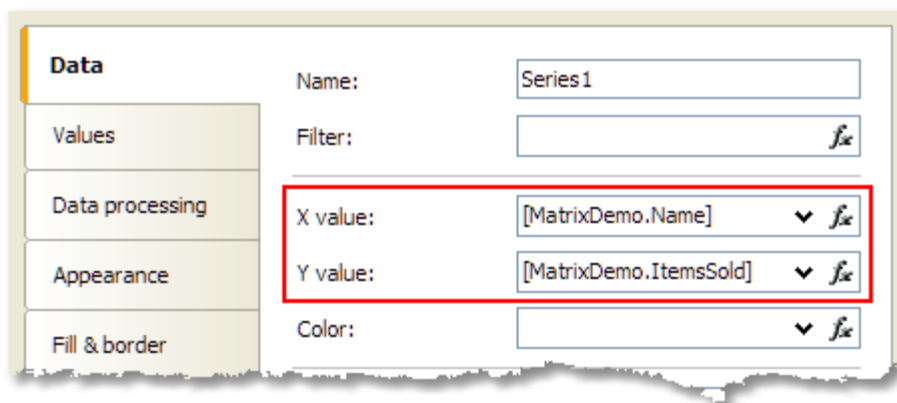
## Setting up auto-series

You can set up the chart so it will create new series automatically, depending on data in a data source. To set up auto-series, do the following:

- create one series and set up its properties. This series will be used as a template for all new series;
- select "Chart" object and set up the auto-series data column. The value of this column will be a name of new series. If there is no series with such name yet, the new series will be added.

Let us demonstrate how to create auto-series. We will use the MatrixDemo data table to get a chart of employee's sales per year. One series will represent one year. To do this:

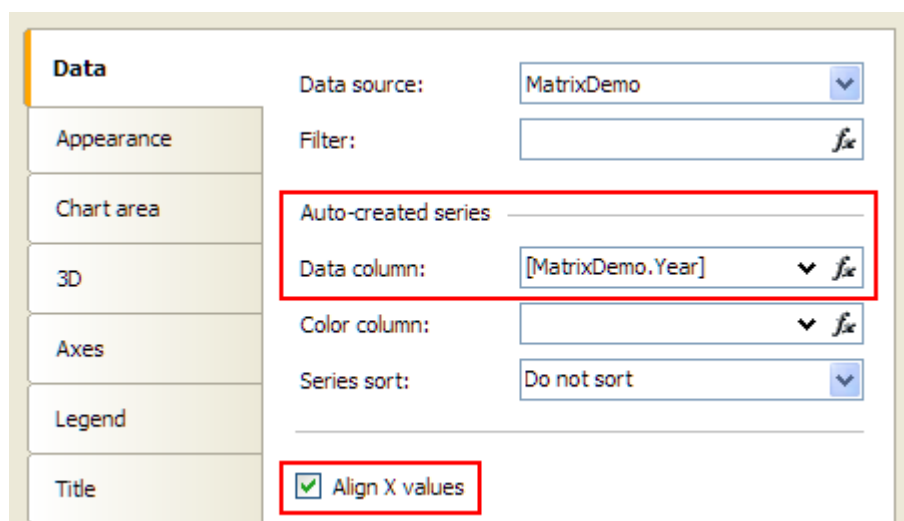
- connect the chart to the MatrixDemo data source;
- create one series and set up its data:



The screenshot shows the 'Data' tab of a configuration window. On the left, there is a vertical menu with options: 'Values', 'Data processing', 'Appearance', and 'Fill & border'. The 'Data' tab is selected. The main area contains the following fields:

- Name: Series1
- Filter: (empty field with a function icon)
- X value: [MatrixDemo.Name] (highlighted with a red box)
- Y value: [MatrixDemo.ItemsSold] (highlighted with a red box)
- Color: (empty field with a function icon)

- on the "Data processing" tab, check the "Group by X value" checkbox. It is necessary because our data source has several employees with the same name;
- select the chart in the series list and set up its auto-series column on the "Data" tab:

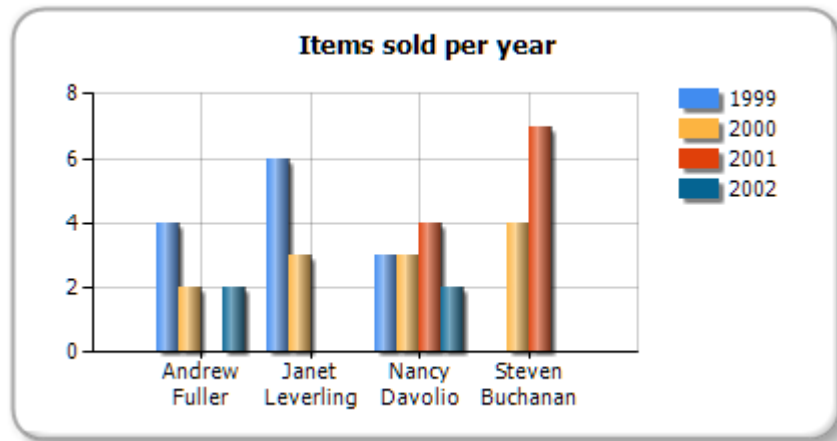


The screenshot shows the 'Data' tab of a configuration window. On the left, there is a vertical menu with options: 'Appearance', 'Chart area', '3D', 'Axes', 'Legend', and 'Title'. The 'Data' tab is selected. The main area contains the following fields:

- Data source: MatrixDemo
- Filter: (empty field with a function icon)
- Auto-created series: (empty field)
- Data column: [MatrixDemo.Year] (highlighted with a red box)
- Color column: (empty field with a function icon)
- Series sort: Do not sort
- Align X values:  (highlighted with a red box)

- our series may have different number of values (because some employees do not have sales in this particular year). To align series values, check the "Align X values" checkbox.

We will get the following result:



## Interactive charts

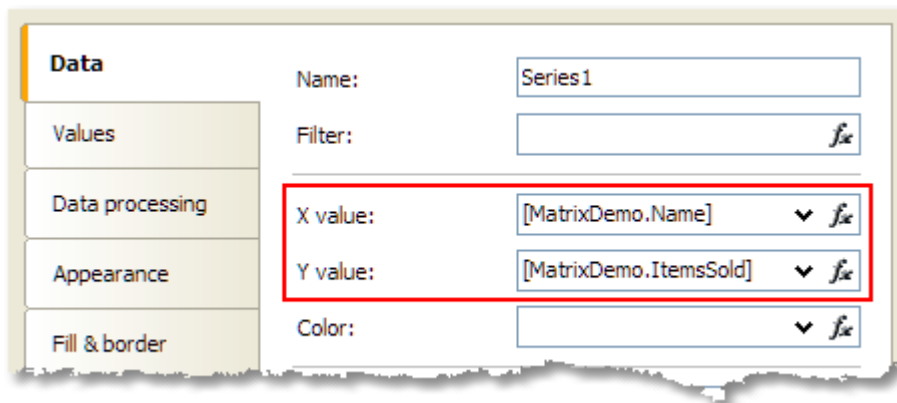
The chart, as well as other report objects, can be interactive. You can set up chart in the way, that when you click its value, another (detailed) report will be executed and shown. To do this, you need to set up the "Hyperlink" property as described in the ["Interactive reports"](#) chapter. The chart will pass the value to a hyperlink by itself, when you click on its element.

Let us observe the "Charts/Interactive Chart" report from the FastReport demo program.

Create a report with two pages. The first page will contain a chart, the second one will contain a detailed report that will be displayed when you click a chart value.

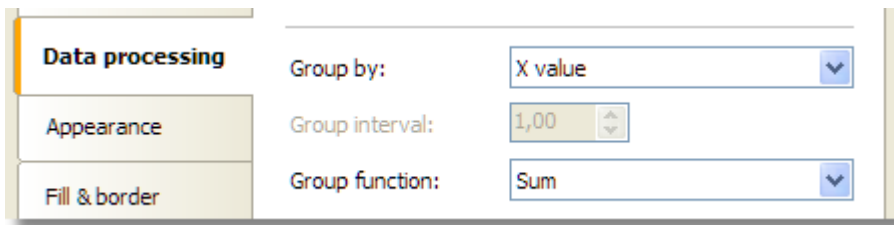
Place the "Chart" object on the first report page and set up its properties in the chart editor:

- select the "Chart" element from the series list and choose the MatrixDemo data source;
- select the series from the series list and set up X and Y values: X = [MatrixDemo.Name], Y = [MatrixDemo.ItemsSold]:



- switch to the "Data processing" tab and select the group type - "X value":





On the second report page, create the list-type report:

- in the "Data" window, create a new report parameter called "SelectedEmployee";
- create the following report layout:

ReportTitle	[SelectedEmployee] orders			
Page Header	Name	Year	Month	ItemsSold
Data: MatrixDemo	[MatrixDemo.Name]	[MatrixDemo.Year]	[MatrixDemo.Month]	[MatrixDemo.ItemsSold]
Report Summary	Total:			[TotalItems]

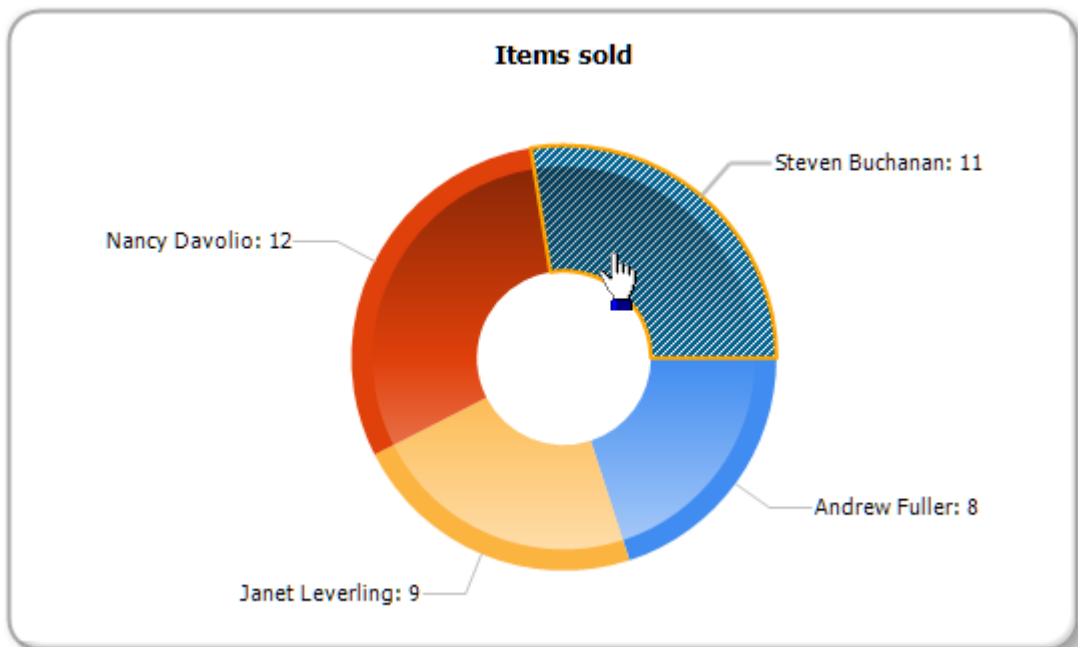
- open the data band editor and indicate the following filter condition:

*[MatrixDemo.Name] == [SelectedEmployee]*

Now set up the hyperlink of the "Chart" object:

- in the context menu of the "Chart" object, select "Hyperlink...";
- choose the hyperlink type - "Report page";
- choose the second report page and indicate the parameter name - SelectedEmployee.

The report is finished. Run it and move the mouse to any chart value. This value will be visually selected, and the mouse cursor will change its shape:



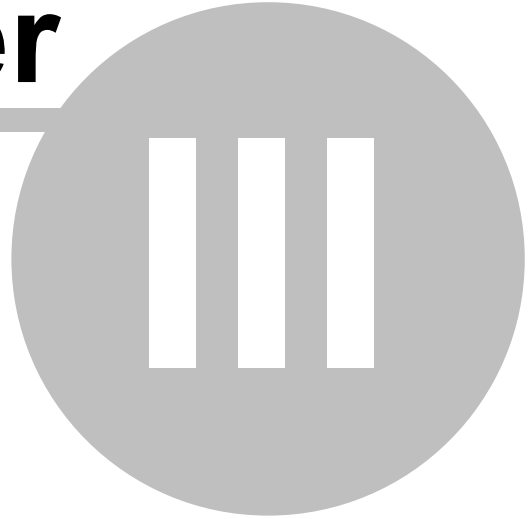
If you click the value, you will see the following detailed report:

## Steven Buchanan orders

Name	Year	Month	ItemsSold
Steven Buchanan	2001	1	3
Steven Buchanan	2001	2	4
Steven Buchanan	2000	1	4
<b>Total:</b>			11

# Chapter

---



**Data**

# Data

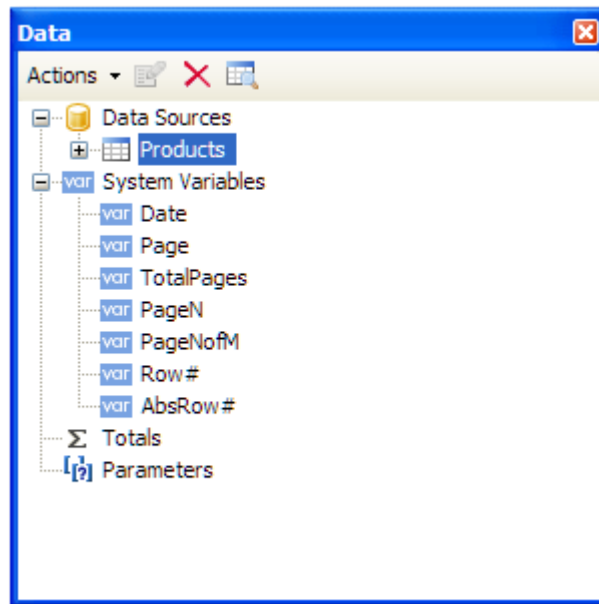
Any report prints some data. In FastReport, you can operate with the following data:

- data sources;
- system variables;
- total values;
- report parameters;
- expressions, containing any of the above mentioned data.

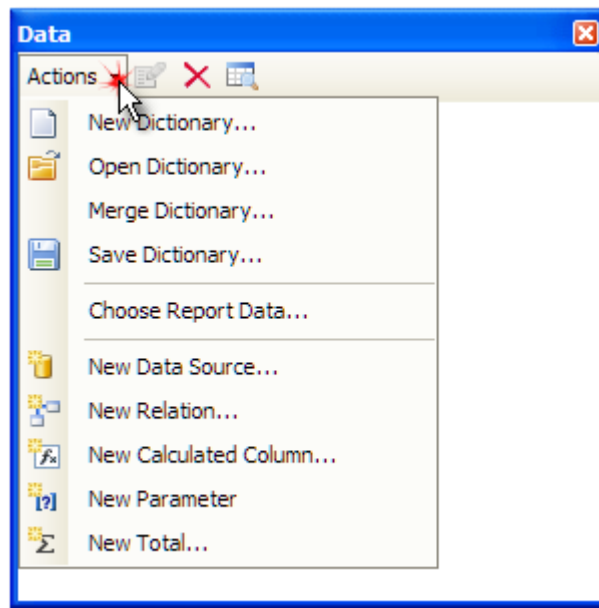
In this chapter, we will look at how to work with these data.

## The "Data" window

All data is accessible from the "Data" window. This window can be shown by choosing the "Data|Show Data Window" menu.



The "Data" window allows to operate with all data elements and also to drag them into the report page. All operations can be done with the help of the toolbar and "Action" menu:



A part of these operations is duplicated in a context menu of the "Data" window. For example, if you select a data source, you can use its context menu to create a calculated column, delete a data source, or view its data.

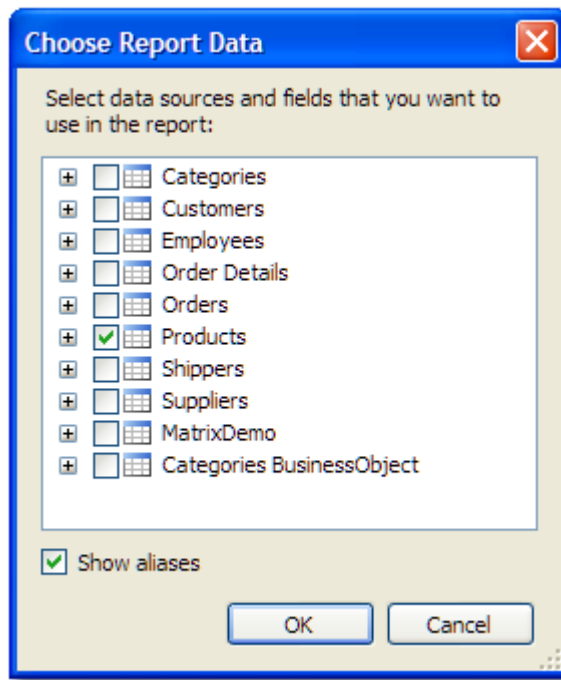
## Data sources

Ordinarily, the data source represents a DB table or SQL query. There can be several data sources in a report. For most of reports, only one data source is needed. A report like Master-Detail needs two data sources which are connected to each other using a relation (we will learn about it later in this chapter).

Data source has one or several data columns. Each column has a definite data type. To look at column type, select it and open the "Properties" window. Column type is indicated in the "DataType" property. The icon near the column name also helps to determine its type.

There are two ways to define a data source for the report.

The first method - data source is defined in the application and registered in a report. It's up to the programmer who created this application (see details in the "Programmer's manual"). A user should only choose the needed data source in order to use it in a report. It can be done in the "Data|Choose Report Data..." menu.



All data registered in a report is listed in this window. Just tick off those data which are needed in your report. It can be done at any moment while working with a report.

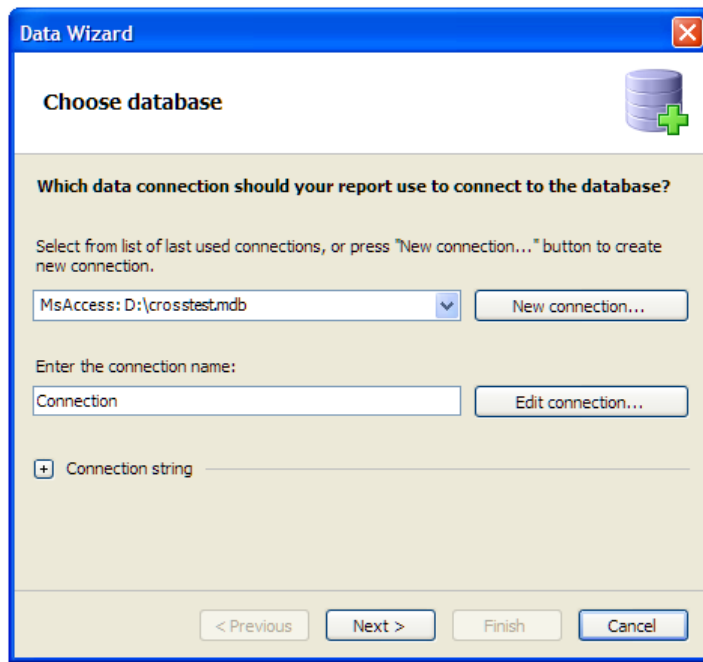
The second method - you create a new data source yourself. It can be a DB table or a SQL query. In such a case, data source definition is saved in a report file.

FastReport allows connecting to many popular DBMS (data base management systems) such as MS SQL, Oracle, Interbase, Access. Besides this, you can use data files which are saved in xml/xsd format.

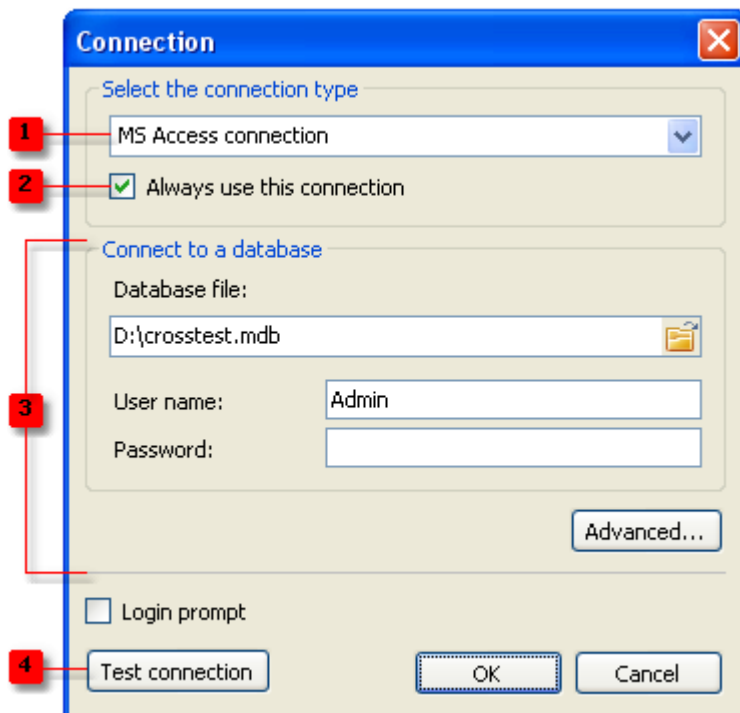
DB table content is not saved in a report file. Instead, the connection string and the data source schema are stored. A connection string can contain such data as login and password, that is why it is kept ciphered in a report file. When needed, you may increase the safety by using own key for data ciphering. In this case a report file can be opened correctly only in your program.

### Creating a data source

To create a new data source, choose the "Data|Add Data Source..." menu item or press the "Actions" button in "Data" window and choose the "New Data Source..." item. You will see the "Data Wizard" window:



First of all, you are offered to create the connection. For that, press the "New connection..." button. You will see a window with connection settings:

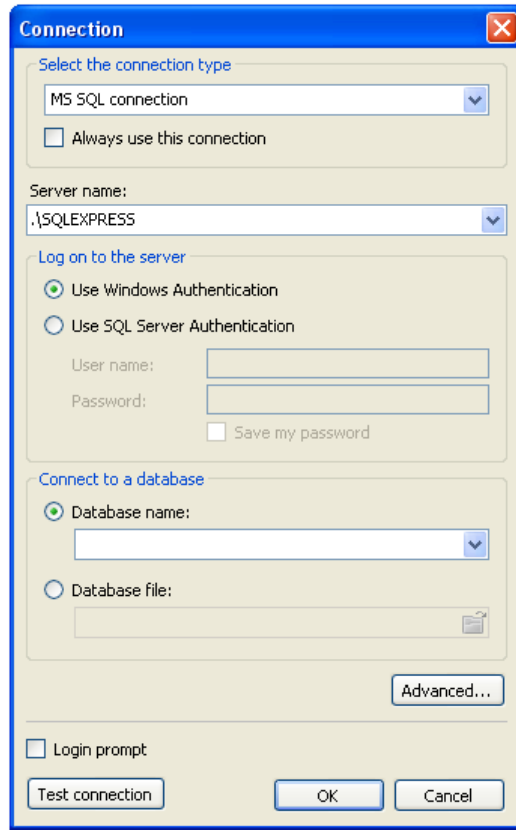


The following elements are shown in the figure:

1. Connection type;
2. If enabled, the chosen connection type will be used by default.
3. Connection settings;
4. Test connection button

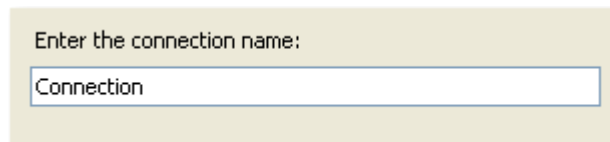
Connection with MS Access data base is shown in the picture. If another type of connection is chosen, then connection settings area (3) will be changed. For example, connection to MS SQL

data base has the following settings:



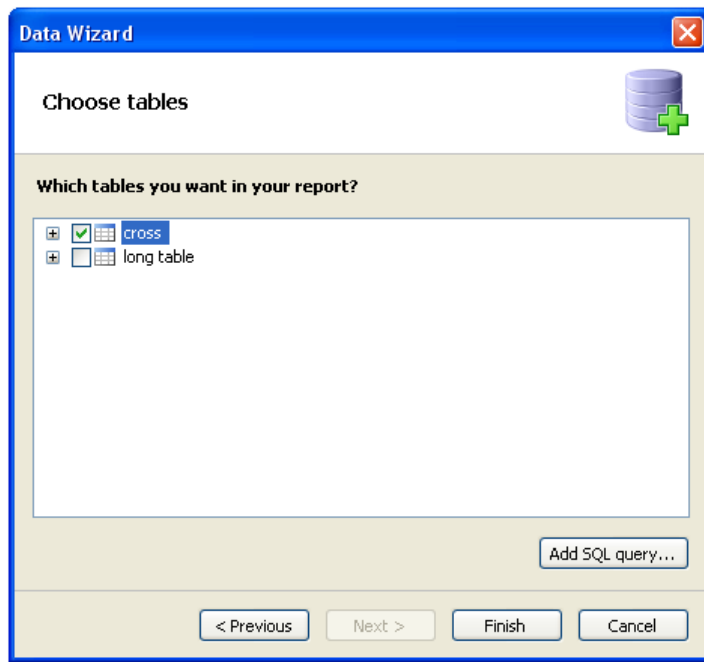
Choose the needed connection type and set up its parameters. After pressing the OK button, the window will be closed and you will return to the data wizard window.

Next, you need to set a connection name. This name will appear in the "Data" window.

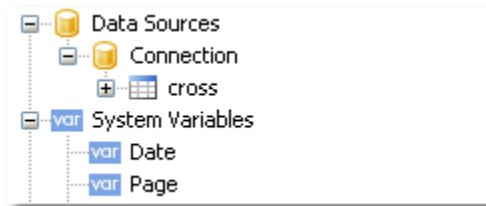


Press the "Next" button to continue. Here you will be offered to choose tables which are accessible in the data base:



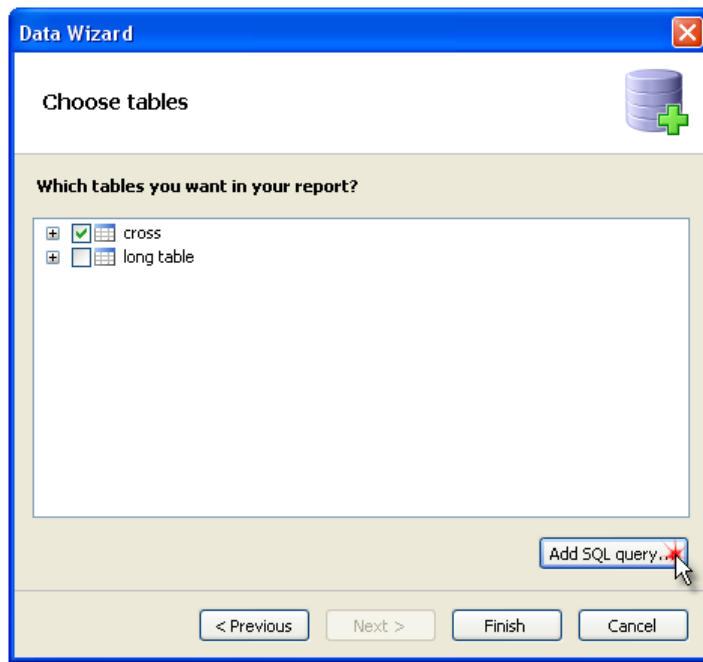


Tick off the needed tables and close the wizard by pressing the "Finish" button. Now you can see in the "Data" window a connection created by you which contains the chosen data sources:



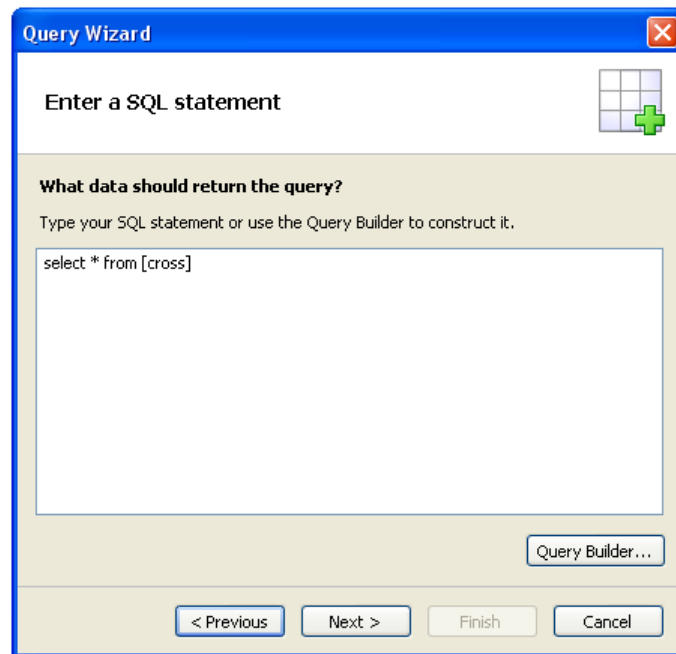
## Creating a SQL query

Data wizard allows quickly choosing tables, contained in a data base. Creation of SQL query requires additional work. For this, press the "Add SQL query..." button in the second step of the wizard.



You will see the query wizard window. Query wizard has four pages. Use the "Next" and "Back" buttons to switch between the pages.

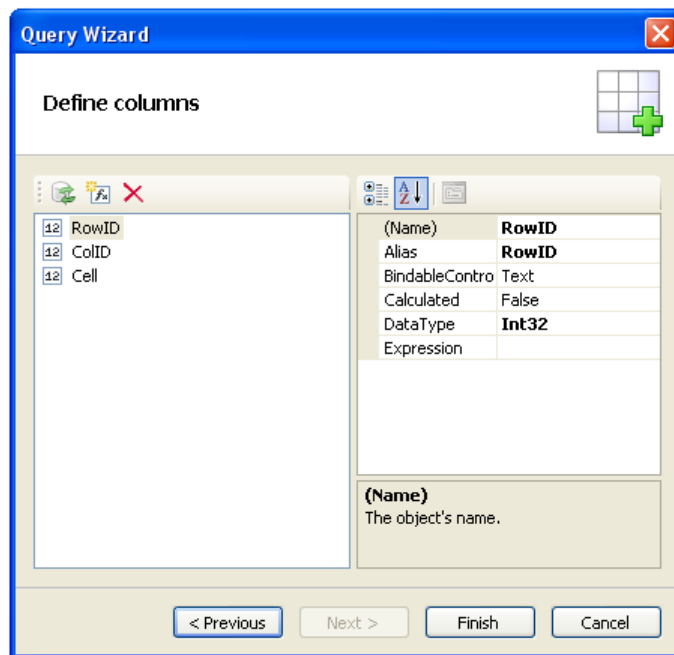
In the first step, you need to set the name of a query. This name will appear in the "Data" window. Enter any unique name and press the "Next" button.



In the second step, you have to enter a query in SQL language. Use the language dialect which is supported by your MSDB. You can use the query builder for visual query creation. To do this, press the "Query Builder" button. The query builder will be described in details later.

After you have entered the query text, press the "Next" button. In the third step, you can define the query parameters. It is required if your query has parameters. We will consider parameters later in this chapter.

On the last step of the wizard, you can set up the columns which were returned by the query:



If you made a mistake in the query text or in the parameter definition, you will see error message when turning to the last page of the wizard.

As a rule, it is enough to be assured that the query has returned all the needed columns. On this step, you can do the following:

- delete unnecessary columns using the "Delete" button;
- reset the columns by pressing the "Refresh" button;
- add a calculated column by pressing the "Add calculated column" button. For a new column, it is necessary to set the "Name", "DataType" and "Expression" properties.

After closing the wizard by pressing the "Finish" button, you will return to the "Data wizard" window.

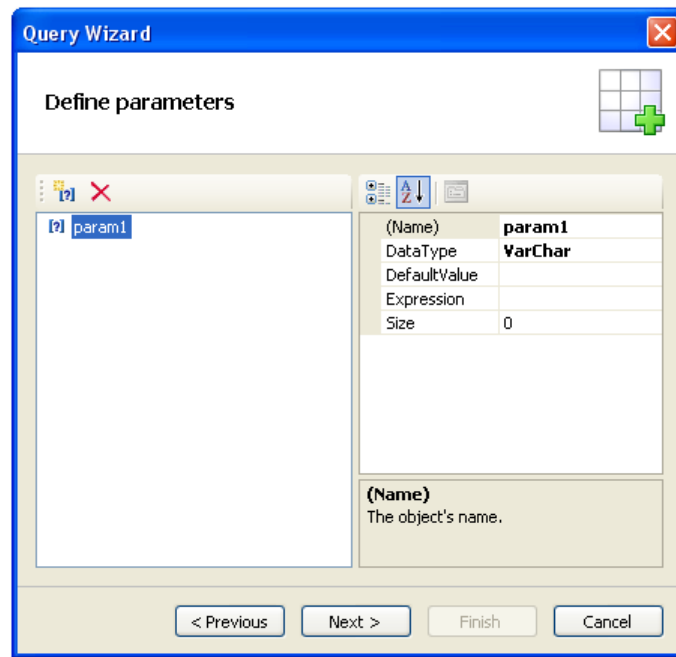
## Query parameters

There can be parameters in a query text. Let us see the following query:

```
select * from DVDs
where Title = @param1
```

This is the query to MS SQL demonstration database. The parameter with "param1" name is defined in a query. Here it should be noted: method of describing parameters in a query differs for different DBMS. For MS SQL a parameter is marked by a "@" symbol, MS Access parameters do not have names and are marked by the "?" symbol.

If your SQL query contains parameters, you have to declare them. It can be done in the third step of the "Query Wizard" which we have looked at above. To create a parameter, press the "Add parameter" button. A new parameter will be created:



The following parameter's properties should be set in the properties window:

Property	Description
Name	Parameter name. Here you need to indicate the same name which you use in the query text. Some DBMS (for example, MS Access) do not support named parameters. In this case do not change this property.
DataType	Parameter data type.
DefaultValue	Value which will be used if the "Expression" property is not specified, or if it is impossible to calculate it (for example, when operating with the query in the report design mode).
Expression	Expression which returns parameter's value. This expression will be processed when you run the report. You can indicate any expression in this property (see details in the "Expressions" chapter).
Size	Parameter data size. This property should be indicated if the parameter is of "string" data type.

If you set the parameter properties incorrectly, you will get an error when turning to the last page of the wizard.

### Passing a value to the parameter

Parameters are often used to ask a value from the user. Let us look at two ways to pass a value to the query parameter.

In the first way, you pass a value programmatically. Since there is no easy way to pass a value directly to the query parameter, you need to use the report parameter, which can be easily set via code. You should do the following:

- Create the report parameter (we will discuss the report parameters later in this chapter). Set the same `DataType` for the report parameter, as it is used in the query parameter.
- In the "Expression" property of the query parameter, refer to a report parameter, for example:

```
[MyReportParameter]
```

- Pass a value to the report parameter:

```
report1.SetParameterValue("MyReportParameter", 10);
```

In the second way, you use the dialogue forms to do this (dialogue forms will be discussed later). For example, if you need to ask a string-type value, do the following:

- add a dialog into your report;
- put the "TextBoxControl" on it. This control will be used to enter the string value;
- set up the parameter as follows:

```
Name=param1
DataType=VarChar
DefaultValue= (empty string)
Expression=TextBox1.Text
Size=255
```

Where `TextBox1` is a control which contains a value entered by the user.

## Editing a connection

Data connection which was created with the help of the "Data Wizard", can be edited. In order to do this, choose the data connection in the "Data" window and press the "Edit" button on the toolbar. You will see a data wizard window which we have looked at earlier. In this window, you can change the connection settings, by pressing the "Edit connection..." button. The type of connection cannot be changed.

On the second page of the wizard, you can select the tables which you want in your report. When you have done, press the "Finish" button.

## Editing a data source

Data source which was created with the help of the "Data Wizard", can be edited. In order to do this, choose the data source in the "Data" window and press the "Edit" button on the toolbar. You will see a "Query Wizard" window which we have looked at earlier. In this window, you can change the SQL query text, set up the query parameters and data columns.

In order to delete the data source, select it and press the "Delete" button on the toolbar. During this physical deletion of the source does not occur, it just changes to inaccessible. You can enter into the "Data|Choose Report Data..." menu and enable such a data source. However, this should never trouble you, because deleted data sources are never saved in the report file, and accordingly, do not get restored when the report is being read the next time.

## Aliases

Every data element (data sources and columns) has got its own name. By default, this is the name defined in the database. In some cases, it can be difficult to understand what is hidden behind such name, for example, ProdID.

Data elements have got a second name - alias. By using an alias, you can rename an element. For example, if we have got a data source CATEGORY\_TABLE with a column called "PROD\_ID", you can give the following alias:

```
CATEGORY_TABLE --> Categories  
PROD_ID --> Product ID
```

You can refer to such a data column in the following way:

```
[Categories.Product ID]
```

When referring to the data element, you must use the alias, if it has been defined. Never refer to an element by using its original name in this case.

In order to rename a data element in this case, choose it in the "Data" window, and press F2. Also, you can select a "Rename" item in the object's context menu. After this, enter the required name and press Enter.

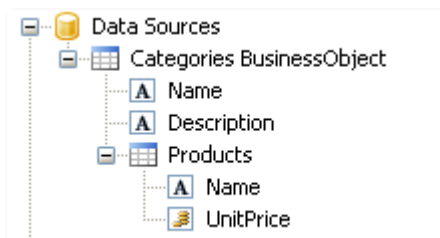
You can also rename an element by using the "Properties" window. Select an element in the "Data" window, switch to the "Properties" window and change the value of the "Alias" property.

In order to delete an alias (reset to the original name), select an element and choose the "Delete alias" item in its context menu.

## Hierarchical data sources

The data sources we have looked at, are relational, that is, they come from a relational DBMS (often called **RDBMS**). FastReport also supports other kinds of data - the hierarchical data sources. Such data sources come from so-called **business objects**, which are often used in applications to represent a relational data sources as a .Net classes.

The only way to add a hierarchical data source in your report is to register it programmatically. It will be discussed in the "Programmer's manual". Now we will look at some differences between ordinal and hierarchical data sources. In the figure below you can see two data sources, "Categories BusinessObject" and "Products". As you can see, the "Products" data source is contained within its parent, "Categories BusinessObject":



This means that, these two data sources are related to each other and can be used in the "master-detail" report type. You can also use each of these data sources separately in a "simple list" report type.

## Relations

Between two data sources, a relation can be set. The relation is used to define the "master-detail" relationship. For example, one record in the "Categories" table can have multiple entries in the "Products" table:

Categories	
CategoryID	CategoryName
1	Beverages

Products		
ProductID	CategoryID	Product name
1	1	Chai
2	1	Chang
39	1	Chartreuse verte
38	1	Côte de Blaye
24	1	Opavská Fantástica

In order to create a relation, you need to indicate the following:

- Parent table;
- Child table;
- Set of key columns in the parent table;
- Set of key columns in the child table.

As an example, we will look at "Categories" and "Products" tables from the demo database. They have the following structure:

Categories	
CategoryID	Primary Key
CategoryName	
Description	
Picture	

Products	
ProductID	Primary Key
ProductName	
SupplierID	
CategoryID	
QuantityPerUnit	
UnitPrice	
UnitsInStock	
UnitsOnOrder	
ReorderLevel	
Discontinued	
EAN13	

Both tables have got the CategoryID field, on which the relationship can be set. So, one category may contain several products.

How can related data sources be used in FastReport? There are two methods of doing this.

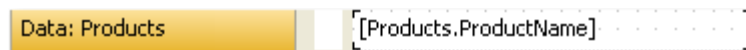
First method makes it possible to build reports of "master-detail" type. To do this, two "Data" bands are used. The master band is connected to the master data source, the detail band - to detail data source. Our example will be like this:



Such a report, if you run it, will print a list of products in every category:



The second method allows referring to the master from the detail data source. We will show this by an example. Let us say, we want to print a list of all the products. For this, we need one "Data" band, which is connected to the "Products" table:



Such a report will print all the products from all the categories. Let us say, beside each product, we want to print a category name to which it belongs. Without using relation, this would have been harder. All we know about the product's category is its id (represented by "CategoryID" column in the "Products" table). Category name, which we would like to print, is stored in the "CategoryName" column of the "Categories" table. With the help of relation, we can refer to the name of a category in the following way:

*[Products.Categories.CategoryName]*

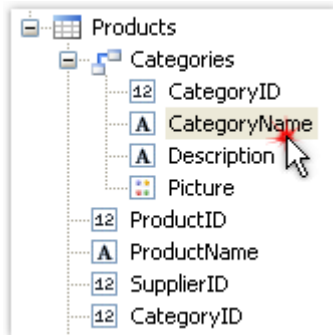


For the current row of the "Products" table, FastReport will find the corresponding parent row in the "Categories" table, and return a value of the "CategoryName" column.

In a general case, way of referring to a parent table field allows an unlimited number of table ancestors:

*[Child\_table.Its\_parent.Parent\_of\_a\_parent.And\_so\_on.Column\_name]*

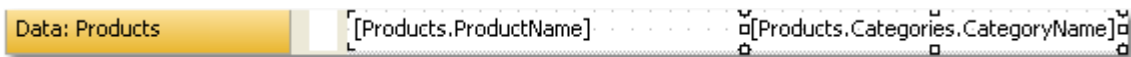
To add such a data column into a report, open the "Products" table in the "Data" window. You will see that among its columns, there is a link to the "Categories" table:



If we drag the column shown above into the report, then we will get a "Text" object with a text:

*[Products.Categories.CategoryName]*

Our report will be as follows:

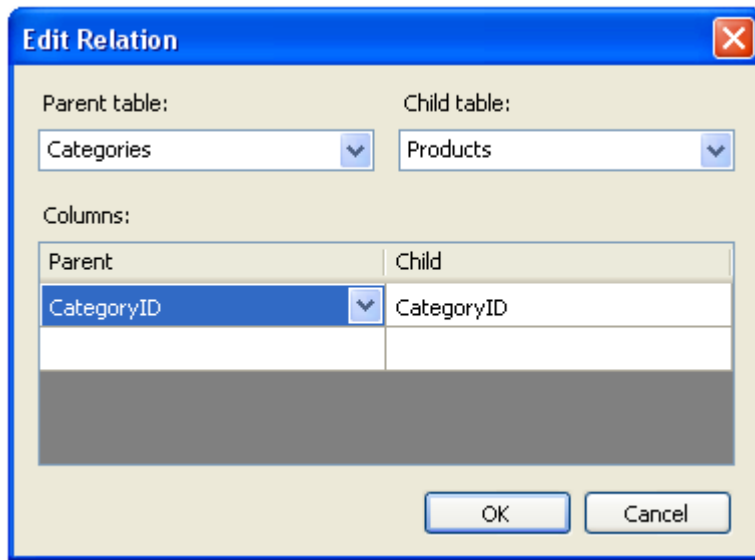


If we run it, we will see the following:

Alice Mutton	Meat/Poultry
Aniseed Syrup	Condiments
Boston Crab Meat	Seafood
Camembert Pierrot	Dairy Products
Carnarvon Tigers	Seafood
Chai	Beverages
Chang	Beverages

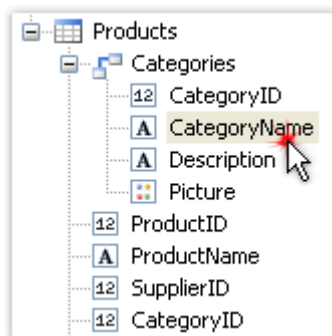
## Creating a relation

In order to create a relation, click the "Actions" button in the "Data" window, and select the "New relation..." item. You will see the relation editor:



In the first place, you need to choose the parent and the child tables. After this, in the lower part of the window, you need to choose related data columns. The tables can be related with the help of one or several data columns. After the columns have been set, close the relation editor by pressing the Ok button.

The relation that you've created can be seen in the "Data" window, if you choose the child data source and open a list of its columns. Among the columns, you will see the relationship with the parent source:



Parent source's data column can be inserted onto the report by using the drag&drop method. So, if we choose the columns shown in the figure, and drag it onto the report page, we will get a "Text" object with the following contents:

```
[Products.Categories.CategoryName]
```

## Editing a relation

In order to edit a relation, open the list of columns of the child data source, find the needed relation and click the "Edit..." button located on the toolbar. This will invoke the relation editor we have looked at earlier.

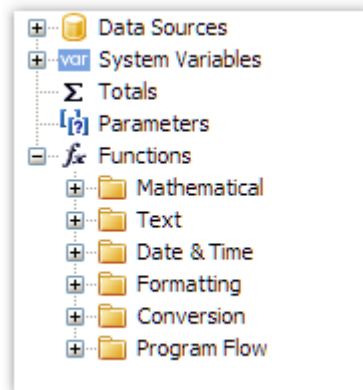
## System variables

There is a list of system variables that can be used in a report:

Variable	Description
Date	Date and time of the report's start.
Page	Current page number.
TotalPages	Total number of pages in the report. To use this variable, you need to enable the report's double pass. You can do this in "Report Properties..." menu.
PageN	Page number in the form: "Page N".
PageNofM	Page number in the form: "Page N of M".
Row#	Data row number inside the group. This value is reset at the start of a new group.
AbsRow#	Absolute number of data row. This value is never reset at the start of a new group.
Page#	Current page number. If you join several prepared reports into one package, this variable will return current page number in a package.  This variable is actually a macro. Its value is substituted when the component is viewed in the preview window. That means you cannot use it in an expression.
TotalPages#	Total number of pages in the report. If you join several prepared reports into one package, this variable will return the number of pages in a package. You don't need to use double pass to get the correct value.  This variable is actually a macro. Its value is substituted when the component is viewed in the preview window. That means you cannot use it in an expression.
HierarchyLevel	Current level of hierarchy in a hierarchical report (see " <a href="#">Printing hierarchy</a> "). The top level is equal to 1.
HierarchyRow#	Full row number like "1.2.1" in a hierarchical report.

## Functions

FastReport.Net contains a lot of built-in functions (over 60). All functions are splitted to several categories and are accessible through the "Data" window:



You may use a function in any expression, in the script (see the "[Script](#)" chapter), or print its value in the "Text" object. For example, the following text in the "Text" object:

```
[Sqrt(4)]
```

will be printed as "2" (square root of 4).

The following expression will return 4:

```
Sqrt(4) + 2
```

Let us look at the ways to insert a function in a report:

- you may drag&drop a function from the "Data" window to a report page. The "Text" object will be created, which contains a function call. You have to edit the text to add parameters to the function call;
- you can drag&drop a function to the script code;
- in the expression editor, you can see a copy of the "Data" window which acts the same way - you may drag items from it and drop them in the expression text.

Below we will describe each function in details.

## Mathematical

### Abs

Function	Parameters	Return value
Abs	sbyte value	sbyte
Abs	short value	short
Abs	int value	int
Abs	long value	long

Abs	float value	float
Abs	double value	double
Abs	decimal value	decimal

Returns the absolute value.

**Example:**

`Abs(-2.2) = 2.2`

**Acos**

Function	Parameters	Return value
Acos	double d	double

Returns the angle (in radians) whose cosine is d. d must be in range between -1 and 1.

Multiply the return value by `180 / Math.PI` to convert from radians to degrees.

**Example:**

`Acos(0) * 180 / Math.PI = 90`

**Asin**

Function	Parameters	Return value
Asin	double d	double

Returns the angle (in radians) whose sine is d. d must be in range between -1 and 1.

Multiply the return value by `180 / Math.PI` to convert from radians to degrees.

**Example:**

`Asin(0) = 0`

**Atan**

Function	Parameters	Return value
Atan	double d	double

Returns the angle (in radians) whose tangent is d.

Multiply the return value by `180 / Math.PI` to convert from radians to degrees.

**Example:**

`Atan(1) * 180 / Math.PI = 45`

## Ceiling

Function	Parameters	Return value
Ceiling	double d	double
Ceiling	decimal d	decimal

Returns the smallest integer greater than or equal to d.

### Example:

```
Ceiling(1.7) = 2
```

## Cos

Function	Parameters	Return value
Cos	double d	double

Returns the cosine of the specified angle (d). The angle must be in radians.

Multiply by `Math.PI / 180` to convert degrees to radians.

### Example:

```
Cos(90 * Math.PI / 180) = 0
```

## Exp

Function	Parameters	Return value
Exp	double d	double

Returns e (2.71828), raised to the specified power d.

### Example:

```
Exp(1) = 2.71828
```

## Floor

Function	Parameters	Return value
Floor	double d	double
Floor	decimal d	decimal

Returns the largest integer less than or equal to d.

### Example:

```
Floor(1.7) = 1
```

## Log

Function	Parameters	Return value
Log	double d	double

Returns the logarithm of a specified number d.

### Example:

$\text{Log}(2.71828) = 1$

## Maximum

Function	Parameters	Return value
Maximum	int val1, int val2	int
Maximum	long val1, long val2	long
Maximum	float val1, float val2	float
Maximum	double val1, double val2	double
Maximum	decimal val1, decimal val2	decimal

Returns the larger of val1, val2.

### Example:

$\text{Maximum}(1,2) = 2$

## Minimum

Function	Parameters	Return value
Minimum	int val1, int val2	int
Minimum	long val1, long val2	long
Minimum	float val1, float val2	float
Minimum	double val1, double val2	double
Minimum	decimal val1, decimal val2	decimal

Returns the smaller of val1, val2.

**Example:**

```
Minimum(1,2) = 1
```

**Round**

Function	Parameters	Return value
Round	double d	double
Round	decimal d	decimal

Rounds d to the nearest integer.

**Example:**

```
Round(1.47) = 1
```

Function	Parameters	Return value
Round	double d, int digits	double
Round	decimal d, int digits	decimal

Rounds d to a precision specified in the "digits" parameter.

**Example:**

```
Round(1.478, 2) = 1.48
```

**Sin**

Function	Parameters	Return value
Sin	double d	double

Returns the sine of the specified angle (d). The angle must be in radians.

Multiply by `Math.PI / 180` to convert degrees to radians.

**Example:**

```
Sin(90 * Math.PI / 180) = 1
```

**Sqrt**

Function	Parameters	Return value
Sqrt	double d	double

Returns the square root of d.



**Example:**

```
Sqrt(4) = 2
```

**Tan**

Function	Parameters	Return value
Tan	double d	double

Returns the tangent of the specified angle (d). The angle must be in radians.

Multiply by Math.PI / 180 to convert degrees to radians.

**Example:**

```
Tan(45 * Math.PI / 180) = 1
```

**Truncate**

Function	Parameters	Return value
Truncate	double d	double
Truncate	decimal d	decimal

Calculates the integral part of d.

**Example:**

```
Truncate(1.7) = 1
```

**Text**

Note:

- these functions do not modify the string value passed in. Instead, they return a new modified string;
- the first character in a string has 0 index. Keep it in mind when working with functions that take the character index, such as Insert.

**Asc**

Function	Parameters	Return value
Asc	char c	int

Returns an Integer value representing the character code corresponding to a character.

**Example:**

`Asc('A') = 65`

## Chr

Function	Parameters	Return value
Chr	int i	char

Returns the character associated with the specified character code.

### Example:

`Chr(65) = 'A'`

## Insert

Function	Parameters	Return value
Insert	string s, int startIndex, string value	string

Inserts a "value" substring into the "s" string at a specified index position "startIndex" and returns a new string.

### Example:

`Insert("ABC", 1, "12") = "A12BC"`

## Length

Function	Parameters	Return value
Length	string s	int

Returns the length of "s".

### Example:

`Length("ABC") = 3`

## LowerCase

Function	Parameters	Return value
LowerCase	string s	string

Converts all characters of "s" to lower case and returns a result.

### Example:

`LowerCase("ABC") = "abc"`

## PadLeft

Function	Parameters	Return value
PadLeft	string s, int totalWidth	string

Right-aligns the characters in the "s" string, padding with spaces on the left for a total width specified in the "totalWidth" parameter.

### Example:

```
PadLeft("ABC", 5) = "  ABC"
```

Function	Parameters	Return value
PadLeft	string s, int totalWidth, char paddingChar	string

Right-aligns the characters in the "s" string, padding with "paddingChar" characters on the left for a total width specified in the "totalWidth" parameter.

### Example:

```
PadLeft("ABC", 5, '0') = "00ABC"
```

## PadRight

Function	Parameters	Return value
PadRight	string s, int totalWidth	string

Left-aligns the characters in the "s" string, padding with spaces on the right for a total width specified in the "totalWidth" parameter.

### Example:

```
PadRight("ABC", 5) = "ABC  "
```

Function	Parameters	Return value
PadRight	string s, int totalWidth, char paddingChar	string

Left-aligns the characters in the "s" string, padding with "paddingChar" characters on the right for a total width specified in the "totalWidth" parameter.

### Example:

```
PadRight("ABC", 5, '0') = "ABC00"
```

## Remove

Function	Parameters	Return value
Remove	string s, int startIndex	string

Deletes all the characters from the "s" string beginning at "startIndex" position and continuing through the last position.

### Example:

```
Remove("ABCD", 3) = "ABC"
```

Function	Parameters	Return value
Remove	string s, int startIndex, int count	string

Deletes a number of characters specified in the "count" parameter from the "s" string, beginning at a "startIndex" position.

### Example:

```
Remove("A00BC", 1, 2) = "ABC"
```

## Replace

Function	Parameters	Return value
Replace	string s, string oldValue, string newValue	string

Returns a string "s" in which a specified substring "oldValue" has been replaced with another substring "newValue".

### Example:

```
Replace("A00", "00", "BC") = "ABC"
```

## Substring

Function	Parameters	Return value
Substring	string s, int startIndex	string

Retrieves a substring from the "s" string. The substring starts at a character position specified in the "startIndex" parameter.

### Example:

```
Substring("ABCDEF", 4) = "EF"
```

Function	Parameters	Return value
Substring	string s, int startIndex, int length	string

Retrieves a substring from the "s" string. The substring starts at a character position specified in the "startIndex" parameter and has a length specified in the "length" parameter.

**Example:**

```
Substring("ABCDEF", 1, 3) = "BCD"
```

### TitleCase

Function	Parameters	Return value
TitleCase	string s	string

Converts the specified string to titlecase.

**Example:**

```
TitleCase("john smith") = "John Smith"
```

### Trim

Function	Parameters	Return value
Trim	string s	string

Removes all occurrences of white space characters from the beginning and end of the "s" string.

**Example:**

```
Trim(" ABC ") = "ABC"
```

### UpperCase

Function	Parameters	Return value
UpperCase	string s	string

Converts all characters of "s" to upper case and returns a result.

**Example:**

```
UpperCase("abc") = "ABC"
```

## Date & Time

### AddDays

Function	Parameters	Return value
AddDays	DateTime date, double value	DateTime

Adds the specified number of days ("value") to the "date" date and returns a new date.

#### Example:

```
AddDays(#7/29/2009#, 1) = #7/30/2009#
```

### AddHours

Function	Parameters	Return value
AddHours	DateTime date, double value	DateTime

Adds the specified number of hours ("value") to the "date" date and returns a new date.

#### Example:

```
AddHours(#7/29/2009 1:30#, 1) = #7/29/2009 2:30#
```

### AddMinutes

Function	Parameters	Return value
AddMinutes	DateTime date, double value	DateTime

Adds the specified number of minutes ("value") to the "date" date and returns a new date.

#### Example:

```
AddMinutes(#7/29/2009 1:30#, 1) = #7/29/2009 1:31#
```

### AddMonths

Function	Parameters	Return value
AddMonths	DateTime date, int value	DateTime

Adds the specified number of months ("value") to the "date" date and returns a new date.

#### Example:

AddMonths(#7/29/2009#, 1) = #8/29/2009#

## AddSeconds

Function	Parameters	Return value
AddSeconds	DateTime date, double value	DateTime

Adds the specified number of seconds ("value") to the "date" date and returns a new date.

### Example:

AddSeconds(#7/29/2009 1:30:01#, 1) = #7/29/2009 1:30:02#

## AddYears

Function	Parameters	Return value
AddYears	DateTime date, int value	DateTime

Adds the specified number of years ("value") to the "date" date and returns a new date.

### Example:

AddYears(#7/29/2009#, 1) = #7/29/2010#

## DateDiff

Function	Parameters	Return value
DateDiff	DateTime date1, DateTime date2	TimeSpan

Returns the interval (number of days, hours, minutes, seconds) between two dates.

### Example:

DateDiff(#1/2/2009#, #1/1/2009#) = 1.00:00:00

## DateSerial

Function	Parameters	Return value
DateSerial	int year, int month, int day	DateTime

Creates a new DateTime value from the specified year, month and day.

### Example:

```
DateSerial(2009, 7, 29) = #7/29/2009#
```

## Day

Function	Parameters	Return value
Day	DateTime date	int

Gets the day of the month (1-31) represented by the specified date.

### Example:

```
Day(#7/29/2009#) = 29
```

## DayOfWeek

Function	Parameters	Return value
DayOfWeek	DateTime date	string

Gets the localized name of the day of the week represented by the specified date.

### Example:

```
DayOfWeek(#7/29/2009#) = "wednesday"
```

## DayOfYear

Function	Parameters	Return value
DayOfYear	DateTime date	int

Gets the day of the year (1-365) represented by the specified date.

### Example:

```
DayOfYear(#7/29/2009#) = 210
```

## DaysInMonth

Function	Parameters	Return value
DaysInMonth	int year, int month	int

Returns the number of days in the specified month and year.

### Example:

```
DaysInMonth(2009, 7) = 31
```



## Hour

Function	Parameters	Return value
Hour	DateTime date	int

Gets the hour component (0-23) represented by the specified date.

### Example:

```
Hour(#7/29/2009 1:30#) = 1
```

## Minute

Function	Parameters	Return value
Minute	DateTime date	int

Gets the minute component (0-59) represented by the specified date.

### Example:

```
Minute(#7/29/2009 1:30#) = 30
```

## Month

Function	Parameters	Return value
Month	DateTime date	int

Gets the month component (1-12) represented by the specified date.

### Example:

```
Month(#7/29/2009#) = 7
```

## MonthName

Function	Parameters	Return value
MonthName	int month	string

Gets the localized name of the specified month (1-12).

### Example:

```
MonthName(1) = "January"
```

## Second

Function	Parameters	Return value
Second	DateTime date	int

Gets the seconds component (0-59) represented by the specified date.

### Example:

```
Second(#7/29/2009 1:30:05#) = 5
```

## Year

Function	Parameters	Return value
Year	DateTime date	int

Gets the year component represented by the specified date.

### Example:

```
Year(#7/29/2009#) = 2009
```

## Formatting

### Format

Function	Parameters	Return value
Format	string format, params object[] args	string

Replaces the format item in a specified "format" string with the value of a corresponding Object instance in a specified "args" array.

For example, the following function call:

```
Format("Name = {0}, hours = {1:hh}", myName, DateTime.Now)
```

contains the following format items: "{0}" and "{1:hh}". They will be replaced with values of myName and DateTime.Now. The result may look as follows:

```
Name = Alex, hours = 12
```

Each format item takes the following form:

```
{index[,alignment][:formatString]}
```

- index - a zero-based integer that indicates which element in a list of objects to format;
- alignment - an optional integer indicating the minimum width of the region to contain the formatted value. If the length of the formatted value is less than alignment, then the region

- is padded with spaces. If alignment is negative, the formatted value is left justified in the region; if alignment is positive, the formatted value is right justified;
- `formatString` -an optional string of format specifiers.

The following table describes the standard numeric format strings.

Format Specifier	Name	Description
C or c	Currency	The number is converted to a string that represents a currency amount.  <code>Format("{0:C}", 10) = "\$10.00"</code>
D or d	Decimal	This format is supported for integral types only. The number is converted to a string of decimal digits (0-9).  <code>Format("{0:D}", 10) = "10"</code>
E or e	Scientific	The number is converted to a string of the form "-d.ddd...E+ddd" or "-d.ddd...e+ddd", where each 'd' indicates a digit (0-9).  <code>Format("{0:E}", 10) = "1,000000E+001"</code>
F or f	Fixed-point	The number is converted to a string of the form "-ddd.ddd..." where each 'd' indicates a digit (0-9).  <code>Format("{0:F}", 10) = "10.00"</code>
G or g	General	The number is converted to the most compact notation.  <code>Format("{0:G}", 10) = "10"</code>
N or n	Number	The number is converted to a string of the form "-d,ddd,ddd.ddd...", where each 'd' indicates a digit (0-9).  <code>Format("{0:N}", 1234.56) = "1,234.56"</code>
P or p	Percent	The number is converted to a string that represents a percent. The converted number is multiplied by 100 in order to be presented as a percentage.  <code>Format("{0:P}", 0.15) = "15.00%"</code>
X or x	Hexadecimal	The number is converted to a string of hexadecimal digits. The case of the format specifier indicates whether to use uppercase or lowercase characters for the hexadecimal digits greater than 9. For example, use 'X' to produce "ABCDEF", and 'x' to produce "abcdef".  <code>Format("{0:X}", 26) = "1A"</code>

If you format the floating-point values, you may indicate a number of decimal places after the format string:

`Format("{0:C1}", 12.23) = "$12.2"`

If the standard numeric format specifiers do not provide the type of formatting you require,

you can use custom format strings:

Format character	Description
0	Zero placeholder. If the value being formatted has a digit in the position where the '0' appears in the format string, then that digit is copied to the result string. The position of the leftmost '0' before the decimal point and the rightmost '0' after the decimal point determines the range of digits that are always present in the result string.
#	Digit placeholder. If the value being formatted has a digit in the position where the '#' appears in the format string, then that digit is copied to the result string. Otherwise, nothing is stored in that position in the result string.
.	Decimal point. The first '.' character in the format string determines the location of the decimal separator in the formatted value.
,	Thousand separator. If the format string contains a ',' character, then the output will have thousand separators inserted between each group of three digits to the left of the decimal separator.
%	Percentage placeholder. The presence of a '%' character in a format string causes a number to be multiplied by 100 before it is formatted.
;	Section separator. The ';' character is used to separate sections for positive, negative, and zero numbers in the format string.

Examples of use:

```
Format("{0:$#,##0.00}", 1024.25) = "$1,024.25"
Format("{0:00%}", 0.25) = "25%"
Format("{0:$#,##0.00;($#,##0.00);Zero}", 1024.25) = "$1,024.25"
Format("{0:$#,##0.00;($#,##0.00);Zero}", -1024.25) = "($1,024.25)"
Format("{0:$#,##0.00;($#,##0.00);Zero}", 0) = "Zero"
```

The following table describes the standard format specifiers for formatting the DateTime values:

Format Specifier	Name	Example
d	Short date pattern	"8/9/2009"
D	Long date pattern	"Sunday, August 09, 2009"
f	Full date/time pattern (short time)	"Sunday, August 09, 2009 2:44 PM"
F	Full date/time pattern (long time)	"Sunday, August 09, 2009 2:44:01 PM"
g	General date/time pattern (short time)	"8/9/2009 2:44 PM"
G	General date/time pattern (long time)	"8/9/2009 2:44:01 PM"

t	Short time pattern	"2:44 PM"
T	Long time pattern	"2:44:01 PM"

The following table describes the custom date/time format specifiers and the results they produce.

<b>Format Specifier</b>	<b>Description</b>
d	Displays the current day of the month, measured as a number between 1 and 31, inclusive. If the day is a single digit only (1-9), then it is displayed as a single digit.
dd	Displays the current day of the month, measured as a number between 1 and 31, inclusive. If the day is a single digit only (1-9), it is formatted with a preceding 0 (01-09).
ddd	Displays the abbreviated name of the day.
dddd	Displays the full name of the day.
f or F	Displays the most significant digit of the seconds fraction.
h	Displays the hour in the range 1-12. If the hour is a single digit (1-9), it is displayed as a single digit.
hh	Displays the hour in the range 1-12. If the hour is a single digit (1-9), it is formatted with a preceding 0 (01-09).
H	Displays the hour in the range 0-23. If the hour is a single digit (1-9), it is displayed as a single digit.
HH	Displays the hour in the range 0-23. If the hour is a single digit (1-9), it is formatted with a preceding 0 (01-09).
m	Displays the minute in the range 0-59. If the minute is a single digit (0-9), it is displayed as a single digit.
mm	Displays the minute in the range 0-59. If the minute is a single digit (0-9), it is formatted with a preceding 0 (01-09).
M	Displays the month, measured as a number between 1 and 12, inclusive. If the month is a single digit (1-9), it is displayed as a single digit.
MM	Displays the month, measured as a number between 1 and 12, inclusive. If the month is a single digit (1-9), it is formatted with a preceding 0 (01-09).
MMM	Displays the abbreviated name of the month.
MMMM	Displays the full name of the month.
s	Displays the seconds in the range 0-59. If the second is a single digit (0-9), it is displayed as a single digit only.
ss	Displays the seconds in the range 0-59. If the second is a single digit (0-9), it is formatted with a preceding 0 (01-09).
t	Displays the first character of the A.M./P.M. designator.

tt	Displays the A.M./P.M. designator.
y	Displays the year as a maximum two-digit number. The first two digits of the year are omitted. If the year is a single digit (1-9), it is displayed as a single digit.
yy	Displays the year as a maximum two-digit number. The first two digits of the year are omitted. If the year is a single digit (1-9), it is formatted with a preceding 0 (01-09).
yyyy	Displays the year, including the century. If the year is less than four digits in length, then preceding zeros are appended as necessary to make the displayed year four digits long.
z	Displays the time zone offset for the system's current time zone in whole hours only. The offset is always displayed with a leading sign (zero is displayed as "+0"), indicating hours ahead of Greenwich mean time (+) or hours behind Greenwich mean time (-). The range of values is -12 to +13. If the offset is a single digit (0-9), it is displayed as a single digit with the appropriate leading sign.
zz	Displays the time zone offset for the system's current time zone in whole hours only. The offset is always displayed with a leading or trailing sign (zero is displayed as "+00"), indicating hours ahead of Greenwich mean time (+) or hours behind Greenwich mean time (-). The range of values is -12 to +13. If the offset is a single digit (0-9), it is formatted with a preceding 0 (01-09) with the appropriate leading sign.
zzz	Displays the time zone offset for the system's current time zone in hours and minutes. The offset is always displayed with a leading or trailing sign (zero is displayed as "+00:00"), indicating hours ahead of Greenwich mean time (+) or hours behind Greenwich mean time (-). The range of values is -12:00 to +13:00. If the offset is a single digit (0-9), it is formatted with a preceding 0 (01-09) with the appropriate leading sign.
:	Time separator.
/	Date separator.

#### Examples of use:

```
Format("{0:d MMM yyyy}", DateTime.Now) = "9 Aug 2009"
Format("{0:MM/dd/yyyy}", DateTime.Now) = "08/09/2009"
Format("{0:MMMM, d}", DateTime.Now) = "August, 9"
Format("{0:HH:mm}", DateTime.Now) = "16:07"
Format("{0:MM/dd/yyyy hh:mm tt}", DateTime.Now) = "08/09/2009 04:07 PM"
```

## FormatCurrency

Function	Parameters	Return value
FormatCurrency	object value	string

Formats the specified value as a currency, using the Windows regional settings.

#### Example:

```
FormatCurrency(1.25) = "$1.25"
```

Function	Parameters	Return value
FormatCurrency	object value, int decimalDigits	string

Formats the specified value as a currency. The "decimalDigits" parameter indicates how many places are displayed to the right of the decimal.

**Example:**

```
FormatCurrency(1.25, 1) = "$1.3"
```

### FormatDateTime

Function	Parameters	Return value
FormatDateTime	DateTime value	string

Formats the specified value as a date/time, using the Windows regional settings. This function does not include neutral values in the resulting string.

**Example:**

```
FormatDateTime(#1/1/2009#) = "01/01/2009"  
FormatDateTime(#1/1/2009 1:30#) = "01/01/2009 1:30:00 AM"  
FormatDateTime(#1:30#) = "1:30:00 AM"
```

Function	Parameters	Return value
FormatDateTime	DateTime value, string format	string

Formats the specified value as a date/time, using the named format specified in the "format" parameter. The valid values for this parameter are:

- "Long Date"
- "Short Date"
- "Long Time"
- "Short Time"

**Example:**

```
FormatDateTime(#1/1/2009 1:30#, "Long Date") = "Thursday, January 01, 2009"  
FormatDateTime(#1/1/2009#, "Short Date") = "01/01/2009"  
FormatDateTime(#1:30#, "Short Time") = "01:30 AM"  
FormatDateTime(#1:30#, "Long Time") = "1:30:00 AM"
```

## FormatNumber

Function	Parameters	Return value
FormatNumber	object value	string

Formats the specified value as a number, using the Windows regional settings.

### Example:

```
FormatNumber(1234.56) = "1,234.56"
```

Function	Parameters	Return value
FormatNumber	object value, int decimalDigits	string

Formats the specified value as a number. The "decimalDigits" parameter indicates how many places are displayed to the right of the decimal.

### Example:

```
FormatNumber(1234.56, 1) = "1,234.6"
```

## FormatPercent

Function	Parameters	Return value
FormatPercent	object value	string

Formats the specified value as a percent, using the Windows regional settings.

### Example:

```
FormatPercent(0.15) = "15.00%"
```

Function	Parameters	Return value
FormatPercent	object value, int decimalDigits	string

Formats the specified value as a percent. The "decimalDigits" parameter indicates how many places are displayed to the right of the decimal.

### Example:

```
FormatPercent(0.15, 0) = "15%"
```



## Conversion

### ToBoolean

Function	Parameters	Return value
ToBoolean	object value	bool

Converts the specified value to boolean.

#### Example:

```
ToBoolean(1) = true  
ToBoolean(0) = false
```

### ToByte

Function	Parameters	Return value
ToByte	object value	byte

Converts the specified value to byte.

#### Example:

```
ToByte("55") = 55
```

### ToChar

Function	Parameters	Return value
ToChar	object value	char

Converts the specified value to char.

#### Example:

```
ToChar(65) = 'A'
```

### ToDateTime

Function	Parameters	Return value
ToDateTime	object value	DateTime

Converts the specified value to date/time.

#### Example:

```
ToDateTime("1/1/2009") = #1/1/2009#
```

## ToDecimal

Function	Parameters	Return value
ToDecimal	object value	decimal

Converts the specified value to decimal.

### Example:

```
ToDecimal(1) = 1m  
ToDecimal("1") = 1m
```

## ToDouble

Function	Parameters	Return value
ToDouble	object value	double

Converts the specified value to double.

### Example:

```
ToDouble(1) = 1  
ToDouble("1") = 1
```

## ToInt32

Function	Parameters	Return value
ToInt32	object value	int

Converts the specified value to int.

### Example:

```
ToInt32(1f) = 1  
ToInt32("1") = 1
```

## ToRoman

Function	Parameters	Return value
ToRoman	object value	string

Converts the specified numeric value to its roman representation. The value must be in range 1-3998.

### Example:

```
ToRoman(9) = "IX"
```

## ToSingle

Function	Parameters	Return value
ToSingle	object value	float

Converts the specified value to float.

### Example:

```
ToSingle(1m) = 1f  
ToSingle("1") = 1f
```

## ToString

Function	Parameters	Return value
ToString	object value	string

Converts the specified value to string.

### Example:

```
ToString(false) = "False"  
ToString(DateTime.Now) = "08/09/2009 4:45:00 PM"
```

## ToWords

Function	Parameters	Return value
ToWords	object value	string

Converts the specified currency value to words.

### Example:

```
ToWords(1024.25) = "One thousand and twenty-four dollars and 25 cents"
```

Function	Parameters	Return value
ToWords	object value, string currencyName	string

Converts the specified currency value to words. The "currencyName" parameter indicates the currency. Valid values for this parameter are:

```
"USD"  
"EUR"  
"GBP"
```

### Example:

```
ToWords(1024.25, "EUR") = "One thousand and twenty-four euros and 25 cents"
```

Function	Parameters	Return value
ToWords	object value, string one, string many	string

Converts the specified integer value to words. The "one" parameter contains the name in singular form; the "many" parameter contains the name in plural form.

**Example:**

```
ToWords(124, "page", "pages") = "One hundred and twenty-four pages"
ToWords(1, "page", "pages") = "One page"
```

**ToWordsEnGb**

Function	Parameters	Return value
ToWordsEnGb	object value	string

Converts the specified currency value to words in Great Britain english. There are the following differences between this function and ToWords:

- the GBP currency is used by default;
- different words used when converting milliards and trilliards.

**Example:**

```
ToWordsEnGb(121) = "One hundred and twenty-one pounds and 00 pence"
```

Function	Parameters	Return value
ToWordsEnGb	object value, string currencyName	string

Converts the specified currency value to words in Great Britain english. The "currencyName" parameter indicates the currency. Valid values for this parameter are:

```
"USD"
"EUR"
"GBP"
```

**Example:**

```
ToWordsEnGb(1024.25, "EUR") = "One thousand and twenty-four euros and 25 cents"
```

Function	Parameters	Return value
ToWordsEnGb	object value, string one, string many	string

Converts the specified integer value to words in Great Britain english. The "one" parameter

contains the name in singular form; the "many" parameter contains the name in plural form.

**Example:**

```
ToWordsEnGb(124, "page", "pages") = "One hundred and twenty-four pages"  
ToWordsEnGb(1, "page", "pages") = "One page"
```

**ToWordsRu**

Function	Parameters	Return value
ToWordsRu	object value	string

Converts the specified currency value to words in russian.

**Example:**

```
ToWordsRu(1024.25) = "Одна тысяча двадцать четыре рубля 25 копеек"
```

Function	Parameters	Return value
ToWordsRu	object value, string currencyName	string

Converts the specified currency value to words in russian. The "currencyName" parameter indicates the currency. Valid values for this parameter are:

- "RUR"
- "UAH"
- "USD"
- "EUR"

**Example:**

```
ToWordsRu(1024.25, "EUR") = "Одна тысяча двадцать четыре евро 25 евроцентов"
```

Function	Parameters	Return value
ToWordsRu	object value, bool male, string one, string two, string many	string

Converts the specified integer value to words in russian. The "male" parameter indicates the gender of the name. The "one", "two" and "five" parameters contain a form of the name used with "1", "2" and "5" numbers.

**Example:**

```
// the "страница" word is of female gender, male = false  
ToWordsRu(122, false, "страница", "страницы", "страниц") =  
"Сто двадцать две страницы"  
  
// the "лист" word is of male gender, male = true  
ToWordsRu(122, true, "лист", "листа", "листов") =
```

"Сто двадцать два листа"

## Program Flow

### Choose

Function	Parameters	Return value
Choose	double index, params object[] choice	object

Returns an element of the "choice" array with the index specified in the "index" parameter. The first array element has 1 index.

#### Example:

```
Choose(2, "one", "two", "three") = "two"
```

### IIf

Function	Parameters	Return value
IIf	bool expression, object truePart, object falsePart	object

Returns the "truePart" value, if the "expression" is true. Otherwise, returns the "falsePart" value.

#### Example:

```
IIf(2 > 5, "true", "false") = "false"
```

### Switch

Function	Parameters	Return value
Switch	params object[] expressions	object

The argument supplied to *expressions* consists of paired expressions and values. The Switch function evaluates the odd-numbered expressions from lowest to highest index, and returns the even-numbered value associated with the first expression that evaluates to True.

#### Example:

```
// returns one of the following values - "a greater than 0",  
// "a less than 0", "a equals to 0", depending on "a" value  
Switch(  
    a > 0, "a greater than 0",  
    a < 0, "a less than 0",  
    a == 0, "a equals to 0")
```

## Totals

In many reports, we may need to show some total information: sum of the group, number of rows in the list, and many others. FastReport uses totals to perform this task. For the total, you need to indicate the following parameters:

- The total function type;
- The expression, which is supposed to be calculated. For the "Count" function, you do not need to indicate the expression;
- The condition. The function will be calculated if the condition is met. It's not obligatory to set up the condition.
- The data band, for which the function will be processed;
- The band, in which the total value will be printed.

The list of total functions is given below:

Function	Description
Sum	Calculates the sum of the expression.
Min	Calculates the minimum value of the expression.
Max	Calculates the maximum value of the expression.
Average	Calculates the average value of the expression.
Count	Returns the number of rows.

## Creating a total

We will look at using the total function as an example. Let us create the "master-detail" report that uses two tables - "Categories" and "Products":



The prepared report will be as follows:

Beverages		Condiments	
Product Name	Units In Stock	Product Name	Units In Stock
Chai	39	Aniseed Syrup	13
Chang	17	Chef Anton's Cajun Seasoning	53
Chartreuse verte	69	Chef Anton's Gumbo Mix	0
Côte de Blaye	17	Genen Shouyu	39
Guaraná Fantástica	20	Grandma's Boysenberry Spread	120
Ipoh Coffee	17	Gula Malacca	27
Lakkalikööri	57	Louisiana Fiery Hot Pepper Sauce	76
Laughing Lumberjack Lager	52	Louisiana Hot Spiced Okra	4
Outback Lager	15	Northwoods Cranberry Sauce	6
Rhönbräu Klosterbier	125	Original Frankfurter grüne Soße	32
Sasquatch Ale	111	Sirop d'érable	113
Steeleye Stout	20	Veggie-spread	24

Let us add total in this report which will be printing the total quantity of units in stock for each category - sum of "UnitsInStock" data column. Total will be printed in the "Data Footer" band.

To print total value, you need to create it first. For this, press "Action" button in the "Data" window, and choose the "New total" item. Another method - right click the "Totals" element in data tree and choose "New total" menu item. You will see the total editor window.

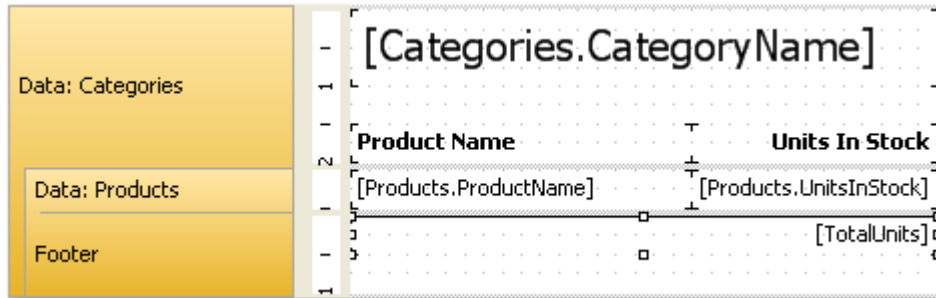
First of all, you will be asked to indicate total's name. You will be referring to the total by its name, so name the total in such a way that it will be easy to understand as what it calculates. Let us call our total as "TotalUnits".



Then we choose the "Sum" function for the total.

Now we need to indicate data range for which total will be calculated. For that in "Evaluate on each row of the band:" field, we choose the "Data" band in which a list of products is printed. In the "Print on the band:" field we choose a band in which total will be printed - that is, the "Data Footer" band.

Close editor by pressing OK button. You will see the new total appears in the "Data" window. Now you can drag it into the report:



When we run the report, we will see the following:

Beverages		Condiments	
Product Name	Units In Stock	Product Name	Units In Stock
Chai	39	Aniseed Syrup	13
Chang	17	Chef Anton's Cajun Seasoning	53
Chartreuse verte	69	Chef Anton's Gumbo Mix	0
Côte de Blaye	17	Genen Shouyu	39
Guaraná Fantástica	20	Grandma's Boysenberry Spread	120
Ipoh Coffee	17	Gula Malacca	27
Lakkalikööri	57	Louisiana Fiery Hot Pepper Sauce	76
Laughing Lumberjack Lager	52	Louisiana Hot Spiced Okra	4
Outback Lager	15	Northwoods Cranberry Sauce	6
Rhönbräu Klosterbier	125	Original Frankfurter grüne Soße	32
Sasquatch Ale	111	Sirop d'érable	113
Steeleye Stout	20	Vegie-spread	24
	559		507

## Conditional totals

In the previous example, total was calculated for all data rows. We can limit this range by indicating the condition in the total editor. Total will be calculated for only those rows, whose condition returns true.

For example, we can set the following conditions:

**Edit Total**

Total

Total name: TotalUnits

Function: Sum

Data column or expression: [Products.UnitsInStock]

Evaluate on each row of the band: Data: Products

Evaluate if the following condition is met: ! [Products.Discontinued]

Print on the band: Footer: Products

Options

- Reset after print
- Reset if band is repeated
- Include invisible rows

OK Cancel

This will mean that, total should be calculated for those products, whose "Discontinued" flag is not set.

## Running totals

In our example totals were reset after printing the "Data Footer" band. This occurred because we indicated in total editor that it is necessary to reset the total after printing it. As a result, each category printed its own total values.

If we uncheck the "Reset after print" checkbox, the total will not be reset after printing. This is what is called running total.

If you need to print two types of totals at the same time - ordinary totals and running totals - create one more total with similar settings and uncheck the "Reset after print" flag.

## Page totals

In order create a total which will be printed on the page footer, you will have the indicate page footer in the "Print on the band:" field.

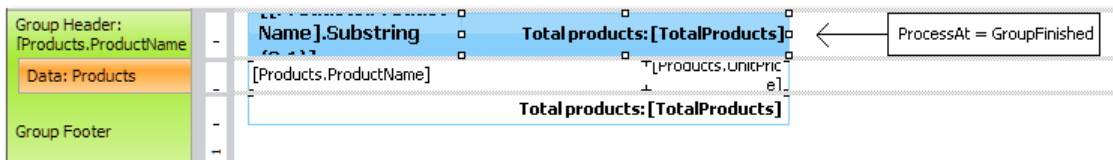
## Printing the total in the header

Usually you will print total values on the footer bands (such as data footer, group footer, etc). It's a natural printing order because when you print the total, its value is properly calculated and is ready to use. However, in some cases, you would need the total to be printed on the header (for example, on the group header). If you try to do this, you will see a zero value. At this moment, when you print a total, it is not calculated yet.

To solve this problem, FastReport has a feature called "delayed print". The "Text" object has a property called "ProcessAt" which can have one of the following values:

Value	Description
Default	The default printing mode. This is the default value.
ReportFinished	The object's value will be calculated at the end of the report.
ReportPageFinished	The object's value will be calculated when all bands in the page will be finished.
PageFinished	The object's value will be calculated at the end of the page.
ColumnFinished	The object's value will be calculated at the end of the column.
DataFinished	The object's value will be calculated at the end of the data band (when its footer is printed).
GroupFinished	The object's value will be calculated at the end of the group (when its footer is printed).

Let's look at how it works. Put the "Text" object which prints the total, on the group header. Set the "ProcessAt" property of the "Text" object to "GroupFinished":



When you run the report, FastReport will do the following:

- it will print the group header. The total value will be printed as 0 (wrong), but FastReport will remember this object to process it later;
- it will print all data rows;
- it will print the group footer. At this moment, FastReport will take the object that was printed in the group header, and process it again to print the correct total value.

The prepared report will be as follows:

A		Total products: 2
Alice Mutton		\$39.00
Aniseed Syrup		\$10.00
		Total products: 2

B		Total products: 1
Boston Crab Meat		\$18.40
		Total products: 1

C		Total products: 9
Camembert Pierrot		\$34.00
Carnarvon Tigers		\$62.50
Chai		\$18.00
		\$19.00

Using other values of the "ProcessAt" property, you may print the report total in the report title (set ProcessAt = ReportFinished), or print the page total in the page header (set ProcessAt = PageFinished).

The delayed print feature will not work if you turn on the report file cache ("Report/Options..." menu, "Use file cache" checkbox).

## Report parameters

You can define parameters in a report. Parameter is a variable, the value of which can be defined both in a report itself and outside of it (a program, calling a report can transfer parameters values into it. See details in "Programmer's manual"). A parameter can be used in expressions and be displayed in report objects like the "Text" object.

Most common methods of using parameters:

- data filtering by condition set in a parameter;
- printing parameter value in a report.

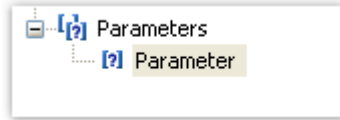
A parameter has the following properties:

Property	Description
Name	Parameter's name can have any symbols except dot ".".
DataType	Parameter data type.
Expression	Expression which returns parameter's value. More details about expressions can be found in the "Expression" chapter. This expression will be processed when calling a parameter.
Value	Parameter value. This property is not available in the designer and can be filled programmatically.

You have to set up "Name" and "DataType" properties. The "Expression" property can be left empty. In this case parameter's value should be passed programmatically.

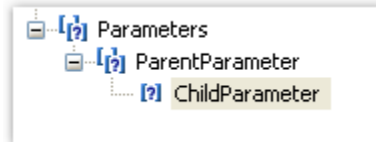
## Creating a parameter

To create a parameter, select the "Parameters" element in the "Data" window, right-click it and choose the "New parameter" item in a context menu:



Press F2 and give a parameter name, then go to the "Properties" window and set the parameter's "DataType" property.

Parameters can be nested. To create a nested parameter, select a parent parameter, right-click it and choose the "New parameter" item in a context menu:



You can refer to both, parent parameter and nested one. Nesting level is not limited.

## Using parameters in a report

You can refer to a parameter from an expression using square brackets:

```
[Parameter name]
```

To a nested parameter you need to refer using this method:

```
[Parent parameter.Child parameter]
```

Since a parameter has got a definite type (it is given in the `DataType` property), then with parameters, you can perform those actions which are allowed for data type. So, string type parameters can be used in an expression the following way:

```
[StringParameter].Substring(0, 2)
```

Let us see one example of using parameters. Assuming we have a report which prints "Employees" table. We want to modify the report to print information about an employee with an indicated number. To do this, we need to filter the data on the "EmployeeID" data column. Create a parameter with "EmployeeID" name. Indicate parameter's type - Int32, as exactly this type has the "EmployeeID" data column. To filter an employee with an indicated ID we need to enter "Data" band editor and indicate the following expression in "Filter" tab:

```
[Employees.EmployeeID] == [EmployeeID]
```

To pass parameter value from your program to the report, use the following code:

```
report1.SetParameterValue("EmployeeID", 2);
```



# Chapter

---



# IV

# Expressions

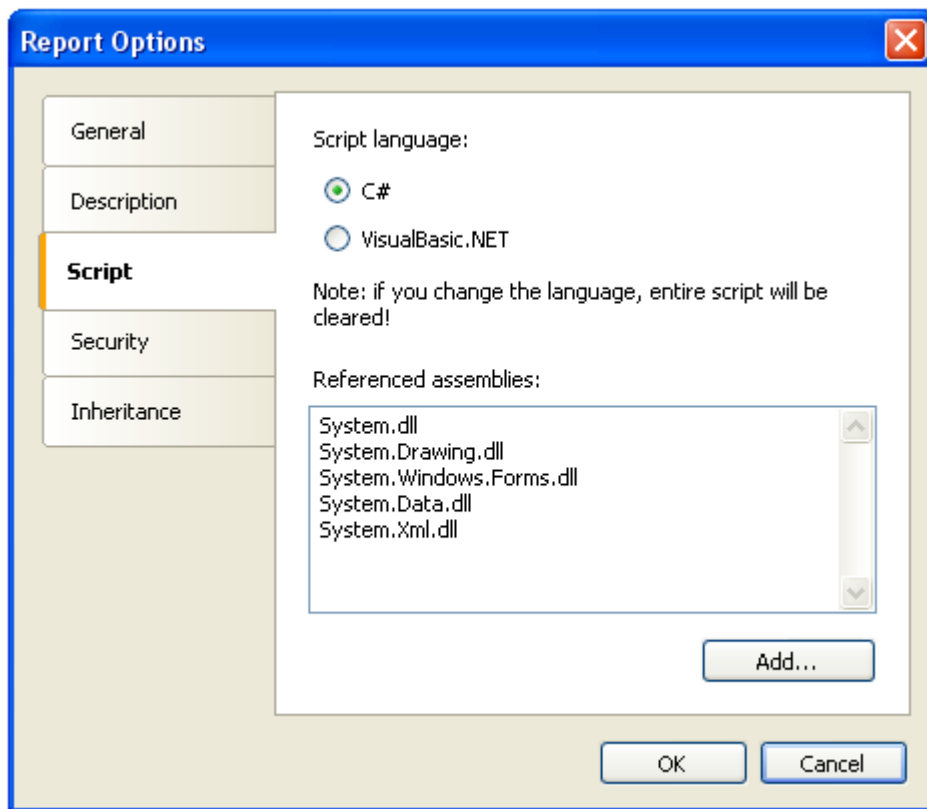
# Expressions

In many places in FastReport, expressions are used. For example, the "Text" object can contain expressions in square brackets.

An expression is a code in C# or VB.Net language, which returns any value. For example:

2 + 2

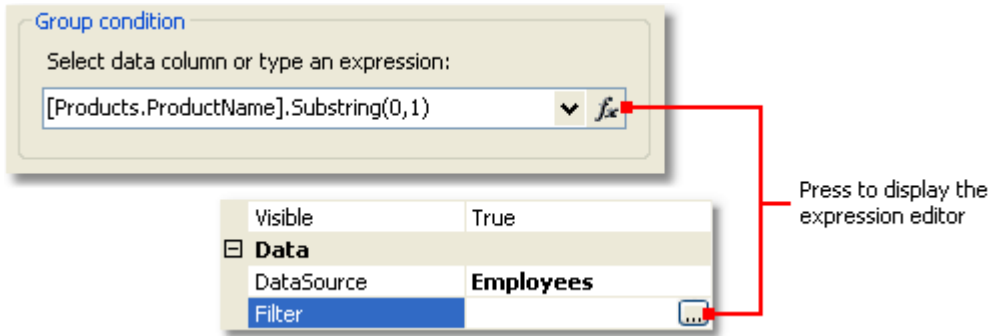
An expression should be written in a language chosen as a script in a report. By default, it is C#. You can change the language in the "Report|Options..." menu by choosing the "Script" element in a window.



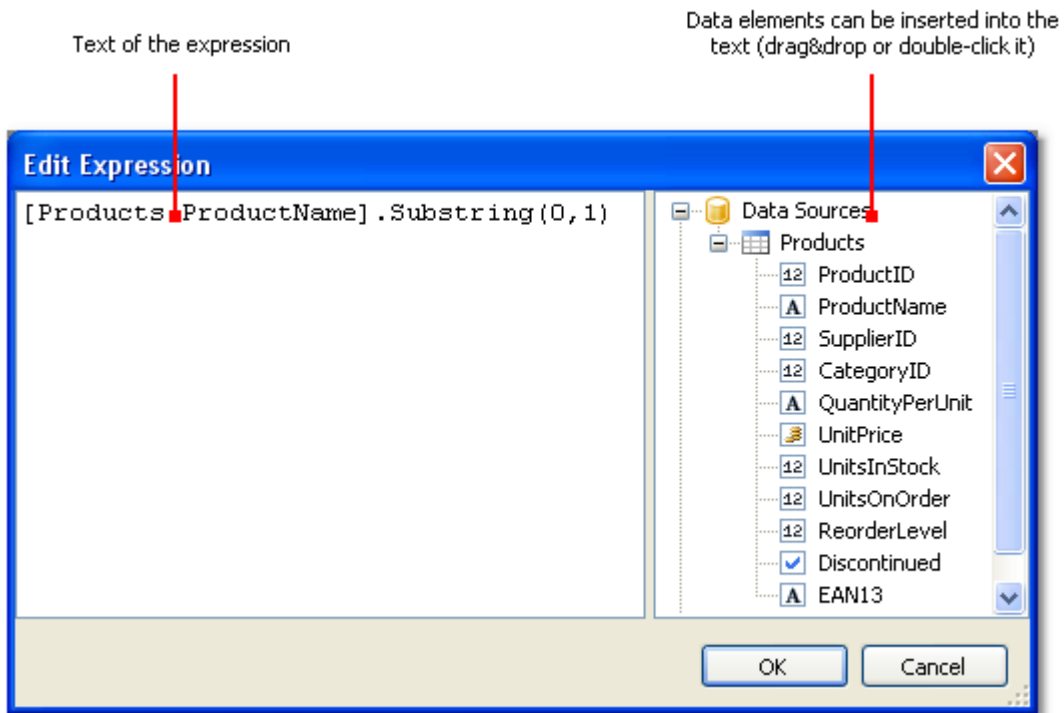


## Expression editor

To write an expression quickly, use the expression editor. It can be invoked in such places of FastReport UI, where you can type an expression:



The expression editor presents a window where you can write an expression and insert some data elements into it:



## Reference to report objects

For referring to report objects, use the object's name. The following example will return the height of the Text1 object:

```
Text1.Height
```

For referring to report properties, use Report variable. The following example returns file name from which a report was loaded.

```
Report.FileName
```

Besides, you can refer to nested object properties. The following example will return a report name:

```
Report.ReportInfo.Name
```

## Using .Net functions

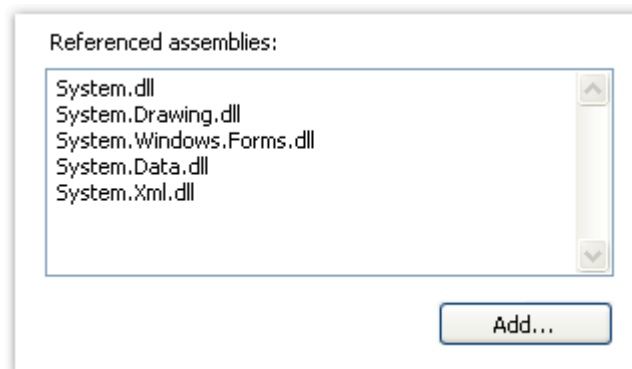
You can use any .Net objects in expressions. The following example demonstrates Max function use:

```
Math.Max(5, 10)
```

By default a report uses the following .Net assemblies:

```
System.dll  
System.Drawing.dll  
System.Windows.Forms.dll  
System.Data.dll  
System.Xml.dll
```

You have access to all .Net objects declared in these assemblies. If you need to have access to another assembly, add its name in report assemblies list. You can do it in the "Report|Options..." menu, by choosing the "Script" element in a window:



For example, if you want to use a function in your report which was declared in your application, add application assembly (.exe or .dll) in a report assemblies list. After that you can call the function by using namespace of your application. For example, if the following function is defined in application:

```
namespace Demo  
{  
    public static class MyFunctions  
    {  
        public static string Func1()  
        {  
            return "Hello!";  
        }  
    }  
}
```

You can use it in your report in the following way:

```
Demo.MyFunctions.Func1()
```

If you add the "using Demo" line at the top of the report's script, it will allow you to shorten the syntax:

```
MyFunctions.Func1()
```

To refer to the function or variable that was defined in a script, just use its name:

```
myPrivateVariableThatIHaveDeclaredInScript
```

```
MyScriptFunction()
```

You can use in an expression only those functions which return a value.

## Reference to data elements

Apart from standard language elements, you can use the following report elements in expressions:

- data source columns;
- system variables;
- total values;
- report parameters.

All these elements are contained in the "Data" window. See more details in the ["Data"](#) chapter. Any of these elements can be used in an expression, by including it in square brackets. For example:

```
[Page] + 1
```

This expression returns next printed page number. A system variable "Page", which returns current report page number, is used in the expression. It is enclosed in square brackets.

## Reference to data sources

For reference to data source columns, the following format is used:

```
[DataSource.Column]
```

Source name is separated from column name by a period, for example:

```
[Employees.FirstName]
```

Source name can be compound in case, if we refer to a data source by using a relation. See more details in the "Data" section. For example, this is how you can refer to a related data source column:

```
[Products.Categories.CategoryName]
```

Let us look at the following example of using columns in an expression:

```
[Employees.FirstName] + " " + [Employees.LastName]
```

Here it should be noted that: every column has a definite data type, which is set in the

"DataType" property (you can see it in the "Properties" window if to choose data column in the "Data" window beforehand). How a column can be used in an expression depends on its type. For instance, in above mentioned example, both columns - first name and last name - have got a string type and that is why they can be used in such a way. In the following example, we will try to use "Employees.Age" column of numeric type, which will lead to an error:

```
[Employees.FirstName] + " " + [Employees.Age]
```

The error occurs because, you never mix strings with numbers. For this, you need to convert the number into a string:

```
[Employees.FirstName] + " " + [Employees.Age].ToString()
```

In this case, we refer to the "Employees.Age" column as if it is an integer variable. And it is so. We know that all expressions are compiled. All non-standard things (like referring to data columns) from a compiler's point of view are converted into another type, which is understandable to a compiler. So, the last expression will be turned into the following form:

```
(string) (Report.GetColumnValue("Employees.FirstName")) + " " +  
(int) (Report.GetColumnValue("Employees.Age")).ToString()
```

As seen, FastReport changes reference to data columns in the following way:

```
[Employees.FirstName] --> (string) (Report.GetColumnValue("Employees.FirstName"))  
[Employees.Age] --> (int) (Report.GetColumnValue("Employees.Age"))
```

That is, we can use data column in an expressions as if it is a variable having a definite type. For example, the following expression will return the first symbol of an employee's name:

```
[Employees.FirstName].Substring(0, 1)
```

## Reference to system variables

You can use the following system variables in expressions (they are accessible in the "Data" window):

Variable	.Net data type	Description
Date	DateTime	Date and time of the report's start.
Page	int	Current page number.
TotalPages	int	Total number of pages in the report. To use this variable, you need to enable the report's double pass. You can do this in "Report Properties..." menu.
PageN	string	Page number in the form: "Page N".
PageNofM	string	Page number in the form: "Page N of M".
Row#	int	Data row number inside the group. This value is reset at the start of a new group.
AbsRow#	int	Absolute number of data row. This value is never reset at the start of a new group.

Every variable has a definite data type. And on this, depends how it will be used in the expression. Here is an example of an expression where date is being used:

```
[Date].Year
```

This expression returns the current year. As "Date" variable has DateTime type, we can refer to its "Year" property. We can get the current month similarly ([Date].Month).

FastReport converts reference to system variable into the following form (for example, the "Date" variable):

```
((DateTime)Report.GetVariableValue("Date"))
```

## Reference to total values

In order to refer to a total value, use its name:

```
[TotalSales]
```

FastReport converts reference to totals into the following form:

```
Report.GetTotalValue("TotalSales")
```

As you can see, the data type is not used here. It is so, because the total value is of FastReport.Variant type. It can be used directly in any expressions because it is automatically converted to any type. For example:

```
[TotalSales] * 0.2f
```

## Reference to report parameters

In order to refer to report parameters, you should use its name:

```
[Parameter1]
```

Parameters can be nested. In this case, you should use both parent and child parameter names in the following form:

```
[ParentParameter.ChildParameter]
```

Parameters have a definite data type. It is set in the "DataType" property of the parameter. The way it can be used in an expression depends on parameter's data type.

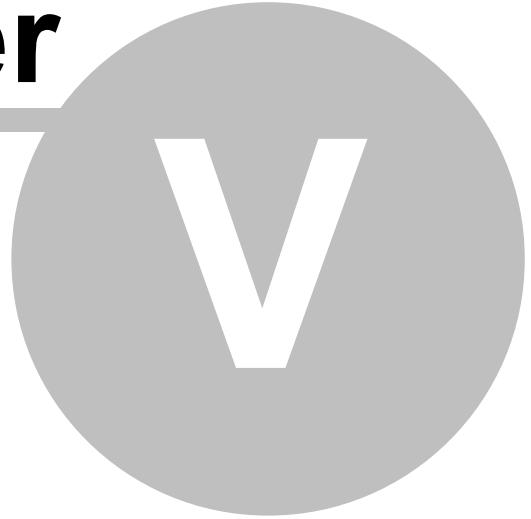
FastReport converts reference to a report parameter into the following way:

```
((string)Report.GetParameterValue("Parameter1"))
```



# Chapter

---



# Script

# Script

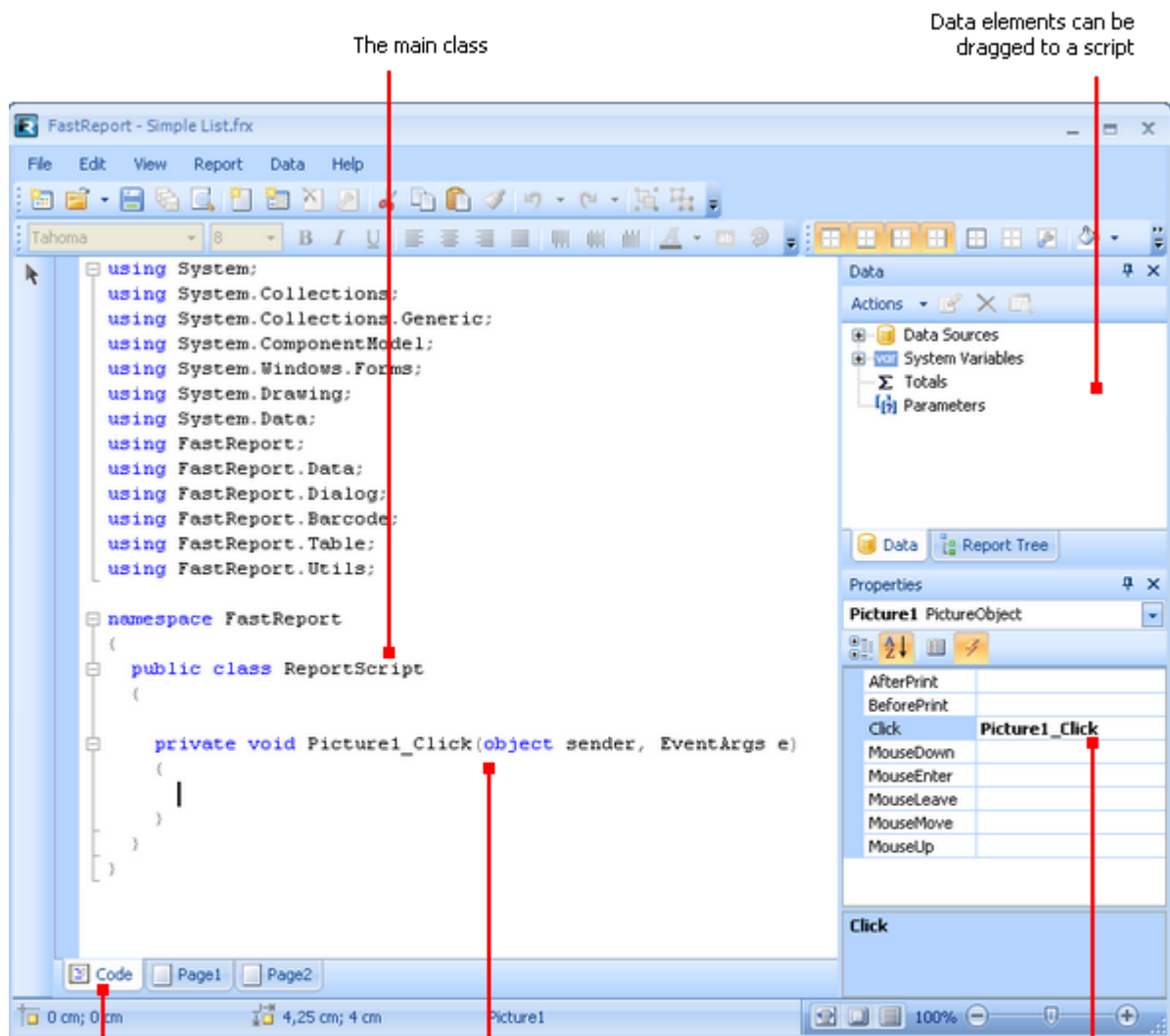
Script is a higher-level programming language, which is part of the report. Script can be written in one of the following .Net languages:

- C#
- VisualBasic.Net

A script is applicable in many places. Using the script, you can do the following:

- perform data handling, which cannot be done via regular means of the FastReport engine;
- control the printing of report pages and bands on the page;
- control the interaction between elements on dialogue forms;
- control the formation of dynamic "Table" objects;
- and many more.

In order to see the report's script, switch to the "Code" tab in the designer:



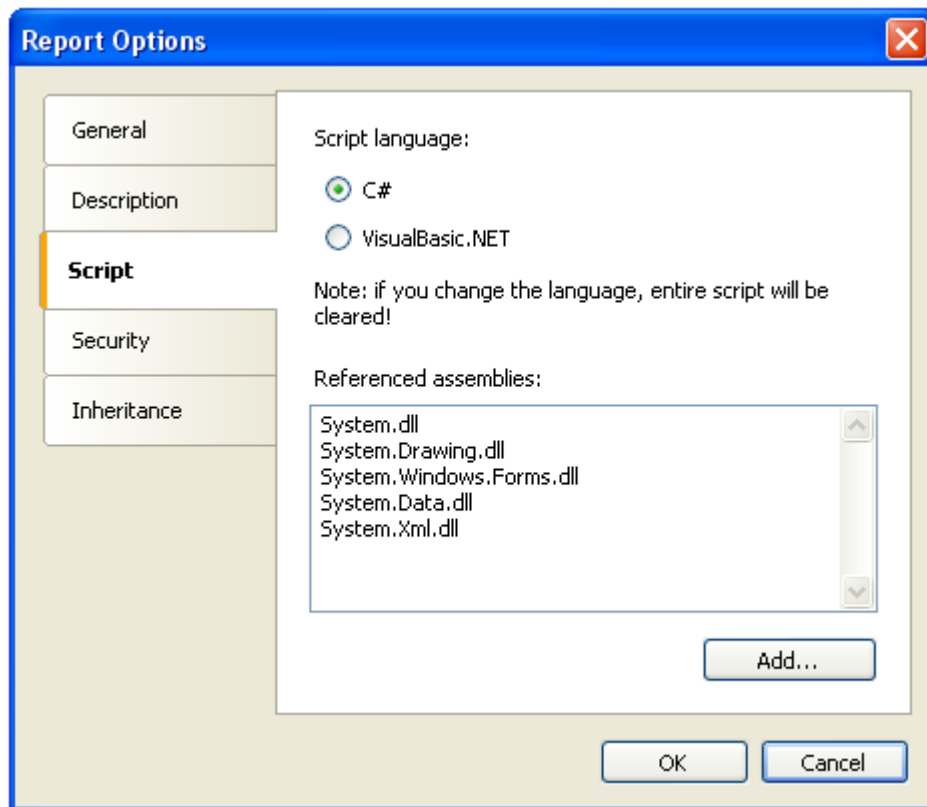
The "Code" tab

Event handler that was created in the "Properties" window

In the "Properties" window, you can create event handlers, by double-clicking the event



Script language can be set in the "Report|Options..." menu. This is supposed to be done just after you have created a new report, because when changing the language, the existing script gets deleted.



## General information

Contrary to other report generators, script in FastReport contains only what you have written. In the script, you can:

- add your variables, methods and properties to the main script class;
- create a report object's events handler;
- add new classes to the script, if needed. A class can be added either before the ReportScript main class or after it.


You cannot:

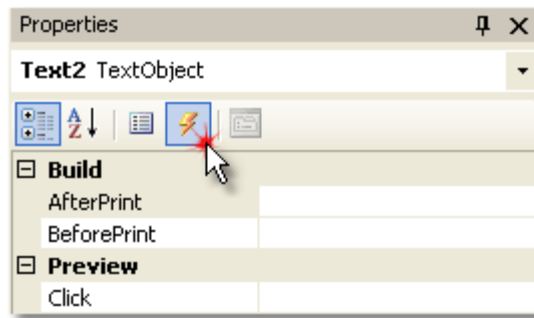
- delete, rename or change the visibility area of the ReportScript main class;
- rename a namespace in which the main class is located.

When the report is running, the following occurs:

- FastReport adds into the script a list of variables, whose names corresponds with the names of the report objects. This is done before compiling the script and allows you to refer to the report objects by their names;
- an expression handler is added to the script, which handles all expressions found in the report;
- a script is compiled, if it is not empty;
- the script class is initialized;
- the report is run.

## Event handlers

A script is mainly used for creating objects' event handlers. For creating event handler select the needed object. In the "Properties" window, press the  button, in order to switch on the list of events:



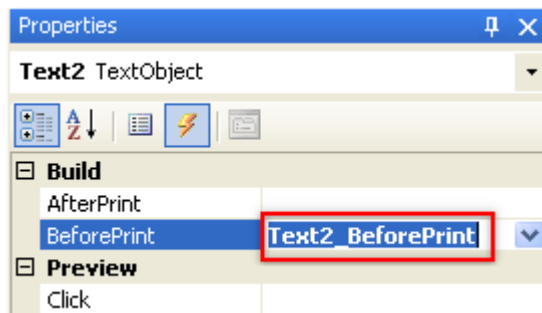
Select the event you want and double click it. FastReport adds an empty event handler into the report code:

```
private void Text2_BeforePrint(object sender, EventArgs e)
{
}
}
```

The "Report" object has got events as well. This object can be chosen by the following method:

- select "Report" in the "Report Tree" window;
- select "Report" in the drop-down list in the "Properties" window.

In order to delete the event handler, choose an event in the "Properties" window, select the text and press the Delete key:



## Report events

In order to control the report with maximum flexibility, every report object has got several events. For example, in a handler, connected to the "Data" band, you can filter records, that is, hide or show the band depending on certain conditions.

Let us consider the events which are fired during the report generation process. As an example, we will take a simple report, containing one page, one "Data" band and two "Text" objects on the band:

Data: Products

- [Products.ProductName] [Products.UnitPrice]

In the beginning of the report, the "Report" object fires the StartReport event. Before formation of the report page, the StartPage event is fired. This event is fired once for every template page (do not confuse with prepared report page!). In our case, regardless of how many pages were in the prepared report - event is fired once, since the template report has got one page.

Further, printing of the "Data" band row starts. This happens in the following way:

1. the BeforePrint band event is fired;
2. the BeforePrint event of all objects lying on the band is fired;
3. all objects are filled with data;
4. the AfterData event of all objects lying on the band is fired;
5. the BeforeLayout band event is fired;
6. objects are placed on the band, the height of the band is calculated and band is stretched (if it can);
7. the AfterLayout band event is fired;
8. if the band cannot fit on a free space on the page, a new page is formed;
9. the band and all its objects are displayed on a prepared report page;
10. the AfterPrint band event is fired;
11. the AfterPrint event of all the band objects is fired.

Printing of the band row occurs as long as there is data in the source. After this, the formation of the report in our case ends. The FinishPage event of a page is fired and finally - the FinishReport event of the "Report" object.

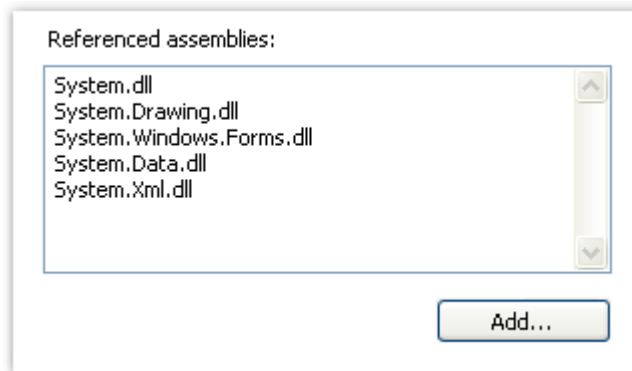
So, by using events of different objects, you can control every step of report formation. The key to correct use of events - full understanding of the band printing process, expound in the eleven steps above. So, a lot of operations can be done, by using only the BeforePrint band - any change, done to the object, will also be displayed. But in this event, it is not possible to analyze, on which page will the band be printed, if it stretches, because the height of the band will be calculated on step 6. This can be done with the help of the AfterLayout event in step 7 or AfterPrint in step 10, but in the latter case, the band is already printed and operations with objects do not give out anything. In one word, you must clearly state, at what moment each event is fired and use, those, which correspond with the given task.

## Using .Net objects

In a script, you can use any .Net objects, which are defined in the following assemblies:

```
System.dll  
System.Drawing.dll  
System.Windows.Forms.dll  
System.Data.dll  
System.Xml.dll
```

Apart from that, you can use any object, defined in the FastReport assembly. If you need access to another assembly, add it to the list of assemblies. This can be done in the "Report|Options..." menu, by choosing the "Script" tab:



For example, if you want to use a function in your report which was declared in your application, add application assembly (.exe or .dll) in a report assemblies list. After that you can call the function by using namespace of your application. For example, if the following function is defined in application:

```
namespace Demo
{
    public static class MyFunctions
    {
        public static string Func1 ()
        {
            return "Hello!";
        }
    }
}
```

Calling it in the script can be done in the following way:

```
string hello = Demo.MyFunctions.Func1();
```

If you add the "using Demo" line at the top of the report's script, it will allow you to shorten the syntax:

```
string hello = MyFunctions.Func1();
```

## Reference to report objects

For referring to report objects (for example, "Text" object) use the name of the object. The following example returns the height of Text1 object:

```
float height = Text1.Height;
```

Note that report's native unit of measurement is screen pixels. Keep it in mind when using such object's properties like Left, Top, Width, and Height. To convert pixels into centimeters and back, use the constants, defined in the "Units" class:

```
float heightInPixels = Text1.Height;
float heightInCM = heightInPixels / Units.Centimeters;
Text1.Height = Units.Centimeters * 5; // 5cm
```

## Report and Engine objects

Apart from objects, which are contained in the report, there are two variables defined in the script: Report and Engine.

The Report variable refers to the current report. In the list below, a list of the report object's methods is given:

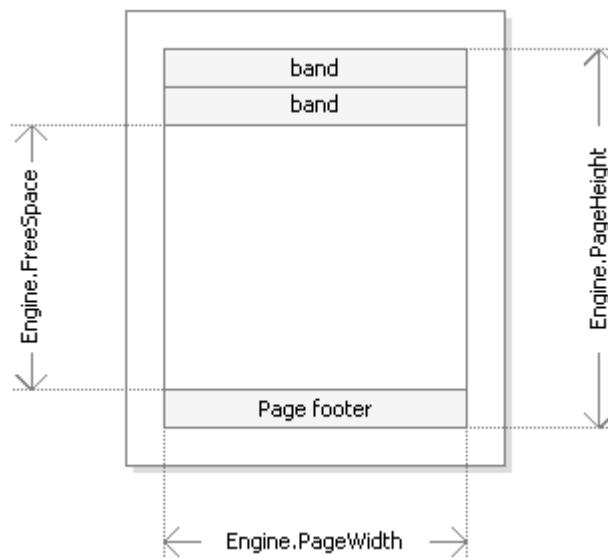
Method	Description
<code>object Calc (string expression)</code>	Calculates an expression and returns the value. When calling this method the first time, an expression gets compiled, which needs some time.
<code>object GetColumnValue (string complexName)</code>	Returns the value of the data column. The name must be presented in the "DataSource.Column" form. If the column has got the <b>null</b> value, it is converted into a value by default (0, empty string, false).
<code>object GetColumnValueNullabl (string complexName)</code>	Returns the value of the data column. Contrary to the previous method, it does not get converted into a default value and may be <b>null</b> .
<code>Parameter GetParameter (string complexName)</code>	Returns the reports parameter with the indicated name. Name can be compounded when referring to the nested parameter: "MainParam.NestedParam".
<code>object GetParameterValue (string complexName)</code>	Returns the value of the report parameter with the indicated name.
<code>void SetParameterValue (string complexName, object value)</code>	Sets the value of the report parameter with the indicated name.
<code>object GetVariableValue (string complexName)</code>	Returns the value of the system variable, for example, "Date".
<code>object GetTotalValue (string name)</code>	Returns the value of the total, defined in the "Data" window, by its name.
<code>DataSourceBase GetDataSource (string alias)</code>	Returns the data source, defined in the report, by its name.

The Engine object is an engine that controls the report creation. By using the methods and properties of the engine, you can manage the process of placing bands onto the page. You can use the following properties of the Engine object:

Property	Description
<code>float CurX</code>	Current coordinates on the X-axis. This property can be assigned a value, so as to shift the printed object.
<code>float CurY</code>	Current printing position on the Y-axis. To this property, a value can be assigned so as to shift the printed object.
<code>int CurColumn</code>	Number of the current column in a multicolumn report. The first column has the number 0.

<code>int</code> CurPage	Number of the page being printed. This value can be received from the "Page" system variable.
<code>float</code> PageWidth	Width of the page minus the size of the left and right margins.
<code>float</code> PageHeight	Height of the page minus the size of the top and bottom margins.
<code>float</code> PageFooterHeight	Height of the page footer (and all its child bands).
<code>float</code> ColumnFooterHeight	Height of the column footer (and all of its child bands).
<code>float</code> FreeSpace	Size of the free space on the page.
<code>bool</code> FirstPass	Returns <b>true</b> , if the first (or only) report pass is being executed. Number of passes can be obtained from the Report.DoublePass property.
<code>bool</code> FinalPass	Returns <b>true</b> , if the last (or only) report pass is being executed.

On the figure below, you can see the meaning of some properties listed above.



Engine.PageWidth and Engine.PageHeight properties determine the size of the printing area, which is almost always less than the actual size of the page. Size of the printed area is determined by the page margins, which is given by the LeftMargin, TopMargin, RightMargin and BottomMargin page properties.

Engine.FreeSpace property determines the height of the free space on the page. If there is the "Page footer" band on the page, its height is considered when calculating the FreeSpace. Note that, after printing a band, free space is reduced.

How does the formation of a prepared report page take place? FastReport engine displays bands on the page until there is enough space for band output. When there is no free space, the "Report footer" band is printed and a new empty page is formed. Displaying a band starts from the current position, which is determined by the X and Y coordinates. This position is returned by the Engine.CurX and Engine.CurY properties. After printing a band, CurY automatically increases by the height of the printed band. After forming a new page, the

position of the CurY is set to 0. The position of the CurX changes when printing a multicolumn report.

Engine.CurX and Engine.CurY properties are accessible not only for reading, but also for writing. This means that you can shift a band manually by using one of the suitable events. Examples of using these properties can be seen in the "Examples" section.

When working with properties, which return the size or position, remember that, these properties are measured in the screen pixels.

In the Engine object, the following methods are defined:

Method	Description
<code>void AddOutline(string text)</code>	Adds an element into the report outline (read the chapter " <a href="#">Interactive reports</a> ") and sets the current position to the added element.
<code>void OutlineRoot()</code>	Sets the current position on the root of the outline.
<code>void OutlineUp()</code>	Shifts the current position to a higher-level outline element.
<code>void AddBookmark(string name)</code>	Adds a bookmark (read the chapter " <a href="#">Interactive reports</a> ").
<code>int GetBookmarkPage(string name)</code>	Returns the page number on which the bookmark with the indicated name is placed.
<code>void StartNewPage()</code>	Starts a new page. If the report is multicolumn, a new column is started.

By using the AddOutline, OutlineRoot, OutlineUp methods, you can form the report outline manually. Usually, this is done automatically with the help of the OutlineExpression property, which every band and report page have got.

The AddOutline method adds a child element to the current outline element, and makes it current. The current report page and the current position on the page are associated with the new element. If you call the AddOutline method several times, then you will have the following structure:

```
Item1
  Item2
    Item3
```

For controlling the current element, there are OutlineUp and OutlineRoot methods. The first method moves the pointer to the element, located on a higher level. So, the script

```
Engine.AddOutline("Item1");
Engine.AddOutline("Item2");
Engine.AddOutline("Item3");
Engine.OutlineUp();
Engine.AddOutline("Item4");
```

will create the following outline:

```
Item1
  Item2
```

```
Item3
Item4
```

The OutlineRoot method moves the current element to the root of the outline. For example, the following script:

```
Engine.AddOutline("Item1");
Engine.AddOutline("Item2");
Engine.AddOutline("Item3");
Engine.OutlineRoot();
Engine.AddOutline("Item4");
```

will create the following outline:

```
Item1
  Item2
    Item3
Item4
```

For working with bookmarks, the AddBookmark and GetBookmarkPage methods of the Engine object are used. Usually bookmarks are added automatically when using the Bookmark property, which all objects of the report have got.

By using the Add Bookmark method, you can add a bookmark programmatically. This method creates a bookmark on the current page at the current printing position.

The GetBookmarkPage method returns the page number on which the bookmark is placed. This method is often used when creating the table of contents, for displaying page numbers. In this case, the report must have a double pass.

## Reference to data sources

Contrary to the FastReport expressions (covered in the ["Expressions"](#) section), never use square brackets in script for referring to the data sources. Instead of this, use the GetColumnValue method of the Report object, which returns the value of the column:

```
string productName = (string)Report.GetColumnValue("Products.Name");
```

As seen, you need to indicate the name of the source and its column. The name of the source can be compound in case, if we are referring to the data source by using a relation. Details about relations can be found in the ["Data"](#) chapter. For example, you can refer to a column of the related data source in this way:

```
string categoryName = (string)Report.GetColumnValue("Products.Categories.CategoryName");
```

For making the work easier, use the "Data" window. From it you can drag data elements into the script, during this FastReport automatically creates a code for referring to the element.

For referring to the data source itself, use the GetDataSource method of the Report object:

```
DataSourceBase ds = Report.GetDataSource("Products");
```

Help on properties and methods of the DataSourceBase class can be received from the FastReport.Net Class Reference help system. As a rule, this object is used in the script in the following way:

```
// get a reference to the data source
```



```

DataSourceBase ds = Report.GetDataSource("Products");

// initialize it
ds.Init();

// enum all rows
while (ds.HasMoreRows)
{
    // get the data column value from the current row
    string productName = (string)Report.GetColumnValue("Products.Name");
    // do something with it...
    // ...

    // go next data row
    ds.Next();
}

```

## Reference to system variables

For reference to system variables, use the `GetVariableValue` method of the Report object:

```
DateTime date = (DateTime)Report.GetVariableValue("Date");
```

A list of system variables can be seen in the "Data" window. From it, you can drag a variable into a script, during this FastReport automatically creates a code for referring to the variable.

## Reference to total values

For reference to the total value, use the `GetTotalValue` method of the Report object:

```
float sales = Report.GetTotalValue("TotalSales");
```

A list of totals can be seen in the "Data" window. From it, you can drag a total into the script, during this FastReport automatically creates a code for referring to the total.

Total value has got the `FastReport.Variant` type. It can be used directly in any expression, because the `FastReport.Variant` type is automatically converted to any type. For example:

```
float tax = Report.GetTotalValue("TotalSales") * 0.2f;
```

Reference to the total value can be done at that time when, it is being processed. Usually the total is "ready to use" at the moment of printing the band, on which it is located in the report.

## Reference to report parameters

For referring to report parameters, use the `GetParameterValue` method of the Report object:

```
int myParam = (int)Report.GetParameterValue("MyParameter");
```

Parameters can be nested. In this case, indicate the name of the parent parameter and after the period, the name of the child parameter:

```
Report.GetParameterValue("ParentParameter.ChildParameter")
```

Parameters have got a definite data type. It is given in the `DataType` property of the

parameter. You must take this into account when referring to parameters. You can see a list of parameters in the "Data" window. From it, you can drag parameters into the script, during this FastReport automatically creates a code for referring to the parameters.

For changing the value of the parameter, use the `SetParameterValue` method of the report object:

```
Report.SetParameterValue("MyParameter", 10);
```

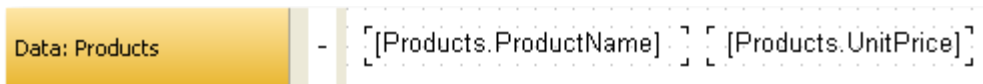
## Examples

### Example 1. Changing object's appearance

In this example we will show how to change the color of the text depending on the value printed in the object. We will be using:

- the BeforePrint event;
- reference to the data column from script.

Create a simple report having the following appearance:



Select the object, which prints the "UnitPrice" column, and create a BeforePrint event handler:

```
private void Text2_BeforePrint(object sender, EventArgs e)
{
    if ((Decimal)Report.GetColumnValue("Products.UnitPrice") > 20)
        Text2.TextColor = Color.Red;
}
```

In order to insert the "Products.UnitPrice" data column into the script, drag it from the "Data" window. During this, the following string will be added in the script:

```
((Decimal)Report.GetColumnValue("Products.UnitPrice"))
```

If we run the report, we will see that all the products, having the price > 20, are highlighted in red:

Chai	18,00
Chang	19,00
Aniseed Syrup	10,00
Chef Anton's Cajun Seasoning	22,00
Chef Anton's Gumbo Mix	21,35
Grandma's Boysenberry Spread	25,00

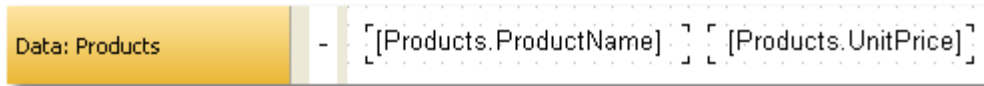
The same effect can be achieved with the help of the conditional highlighting (you can read more about this in the ["Conditional highlighting"](#) section of the "Report creation" chapter).

## Example 2. Highlighting even rows of the band

In this example we will show how to change the fill color of the "Data" band's even rows. We will be using:

- BeforePrint band event;
- reference to the "Row#" system variable from the script.

Create a simple report having the following appearance:



Create a BeforePrint event handler for the band:

```
private void Data1_BeforePrint(object sender, EventArgs e)
{
    if (((Int32)Report.GetVariableValue("Row#")) % 2 == 0)
        Data1.FillColor = Color.Gainsboro;
}
```

The "Row#" system variable returns the number of the row of the printed band. In order to insert into the script a reference to the variable, drag it from the "Data" window. During this, in the script a string will be inserted:

```
((Int32)Report.GetVariableValue("Row#"))
```

If we run the report, we will see that even rows will be highlighted in light-gray color:

Chai	18,00
Chang	19,00
Aniseed Syrup	10,00
Chef Anton's Cajun Seasoning	22,00
Chef Anton's Gumbo Mix	21,35

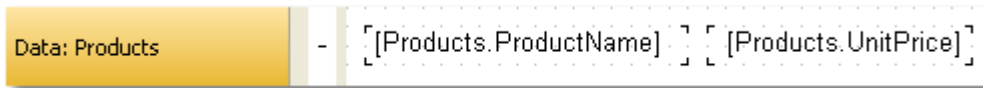
The same effect can be achieved with the help of the "EvenStyle" property of the "Data" band. You can read more about this in the ["Highlight odd/even data rows"](#) section of the "Report creation" chapter.

## Example 3. Data filtering

In this example, we will show how to hide the "Data" band row depending on the given conditions. We will be using:

- BeforePrint band event;
- reference to the data source from a script.

Create a simple report having the following appearance:



Create a BeforePrint event handler for the band:

```
private void Data1_BeforePrint(object sender, EventArgs e)
{
    if (((Decimal)Report.GetColumnValue("Products.UnitPrice")) > 20)
        Data1.Visible = false;
}
```

In the given case, the band rows which have the unit price > 20 will be hidden:

Chai	18,00
Chang	19,00
Aniseed Syrup	10,00
Konbu	6,00
Genen Shouyu	15,50
Pavlova	17,45

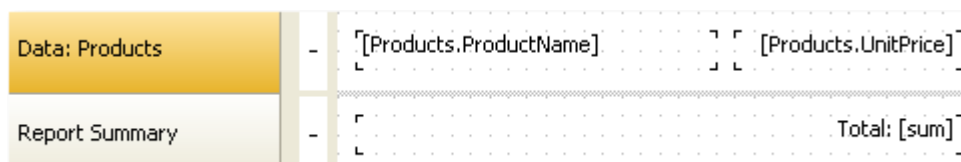
The same effect can be achieved by using the data filter which can be set in the "Data" band editor.

#### Example 4. Calculating total

In this example, we will show how to calculate the sum by using programming methods. We will use the following:

- BeforePrint band event;
- reference to the data column from a script;
- local variable, whose value will be printed in the report.

Create a report of the following form:



In the script, declare the "sum" variable and create a BeforePrint event handler belonging to the band:

```

public class ReportScript
{
    private decimal sum;

    private void Data1_BeforePrint(object sender, EventArgs e)
    {
        sum += (Decimal)Report.GetColumnValue("Products.UnitPrice");
    }
}

```

The "Products.UnitPrice" data column can be placed into the script, dragging it from the "Data" window.

If you run the report, you will see the following:

Röd Kaviar	15,00
Longlife Tofu	10,00
Rhönbräu Klosterbier	7,75
Lakkalikööri	18,00
Original Frankfurter grüne Soße	13,00
Total: 2222,71	

The same effect can be achieved by using totals.

### Example 5. Shifting the print position

In this example, we will show how to shift the position of a band manually, by using the Engine object. We will be using:

- BeforePrint band event;
- Engine object.

Create a simple report of the following appearance:

Data: Products	-	[Products.ProductName]	[Products.UnitPrice]
----------------	---	------------------------	----------------------

Create a BeforePrint event handler for band:

```

private void Data1_BeforePrint(object sender, EventArgs e)
{
    Engine.CurX = ((Int32)Report.GetVariableValue("Row#")) * 10;
}

```

If you run the report, you will see the following:

Chai	18,00
Chang	19,00
Aniseed Syrup	10,00
Chef Anton's Cajun Seasoning	22,00
Chef Anton's Gumbo Mix	21,35





# Chapter

---




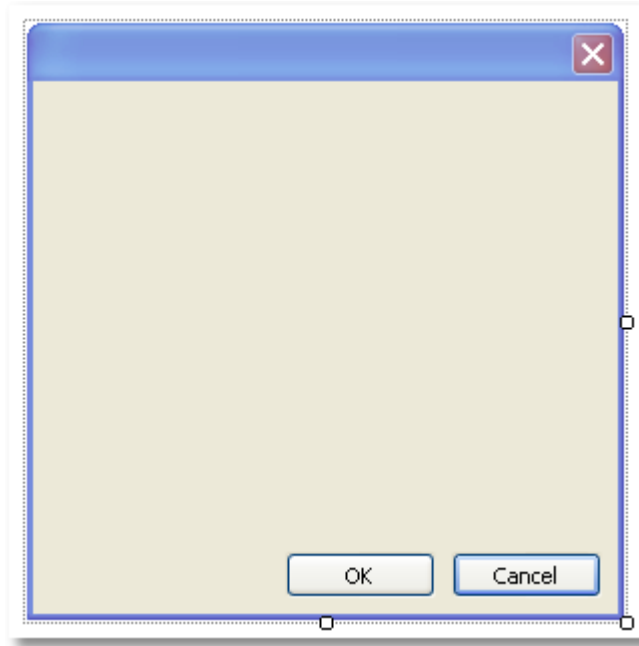
VI

## Dialogue forms

## Dialogue forms

Apart from an ordinary page, a report can contain one or several dialogue forms. Such dialogue form will be shown when the report is started. In the dialogue, you can enter any type of information, needed for creating a report. Also, a dialogue may be used to filter data, which is shown in the report.

For adding a dialogue into a report, press the  button on the designer's toolbar. A new dialogue looks as follows:























A report, having one or several dialogues, works like this:

- when started, the report shows the first dialogue;
- if the dialogue is closed with the help of the "OK" button, then the next dialogue is shown; otherwise, the report stops working;
- after all the dialogues have been shown, the report is built.

## Controls

On a dialogue from, the following controls can be used:

Icon	Name	Description
	ButtonControl	Represents a Windows button control.
	CheckBoxControl	Represents a Windows CheckBox.
	CheckedListBoxControl	Displays a ListBox in which a check box is displayed to the left of each item.

	ComboBoxControl	Represents a Windows combo box control.
	DataGridViewControl	Displays data in a customizable grid.
	DataSelectorControl	Displays two lists and allows relocating an item from one list to the other.
	DateTimePickerControl	Represents a Windows control that allows the user to select a date and a time and to display the date and time with a specified format.
	GroupBoxControl	Represents a Windows control that displays a frame around a group of controls with an optional caption.
	LabelControl	Represents a standard Windows label.
	ListBoxControl	Represents a Windows control to display a list of items.
	ListViewControl	Represents a Windows list view control, which displays a collection of items that can be displayed using one of four different views.
	MaskedTextBoxControl	Uses a mask to distinguish between proper and improper user input.
	MonthCalendarControl	Represents a Windows control that enables the user to select a date using a visual monthly calendar display.
	NumericUpDownControl	Represents a Windows spin box (also known as an up-down control) that displays numeric values.
	PanelControl	Used to group collections of controls.
	PictureBoxControl	Represents a Windows picture box control for displaying an image.
	RadioButtonControl	Enables the user to select a single option from a group of choices when paired with other RadioButton controls.
	RichTextBoxControl	Represents a Windows rich text box control.
	TextBoxControl	Represents a Windows text box control.
	TreeViewControl	Displays a hierarchical collection of labeled items.

All controls, except the DataSelectorControl, are a full analog of the standard Windows.Forms controls. The name of the element has got a Control suffix, in order to avoid duplicate names. So, the FastReport's ButtonControl corresponds with the standard Button control.

## Referencing to a control from code

Referencing to a control can be done by using its name:

```
TextBoxControl1.Text = "my text";
```

In fact, the FastReport's control is just a wrapper for the standard .Net control. It wraps many, but not all, properties of the standard control. If you need some property that is not implemented by the FastReport control, you may access a wrapped standard control in the following way:

- using the "Control" property, which is of System.Windows.Forms.Control type:

```
(TextBox1.Control as TextBox).ShortcutsEnabled = false;
```

- using the property which has the same name as the control itself, but without the "Control" suffix. For example, the TextBoxControl has got the "TextBox" property, which is of System.Windows.Forms.TextBox type and returns the wrapped TextBox control:

```
TextBox1.TextBox.ShortcutsEnabled = false;
```

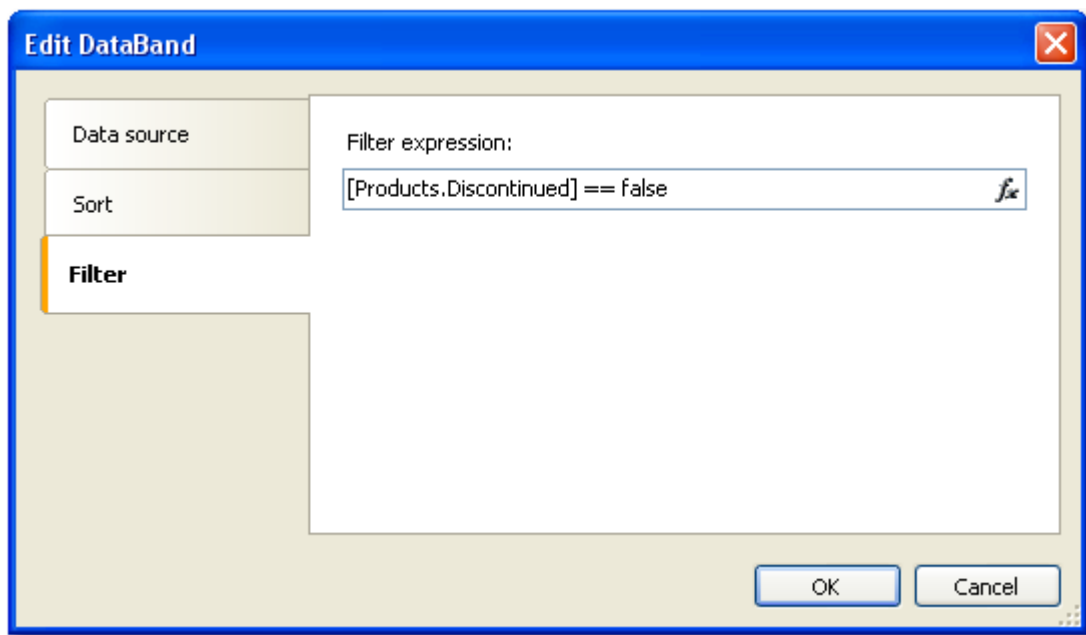
Help on properties and methods of the controls can be accessed from the MSDN.

## Data filtering

Dialogs can be used to filter the data which is printed in a report. For example, you have a report, which prints a list of all the employees. By using a dialog, you can choose one or several from it, and then when building the report, the data is filtered so that only the chosen employees can be shown.

For using the data filtering, it is necessary that the initial report contains all the data. The name "filtering" itself, assumes that, unnecessary data will not be printed when building the report.

The simplest method for organizing data filtering is to use the Filter property on the "Data" band. In the band editor, you can indicate the filter expression, for example:



By using the dialogue, you can ask a value from a user, and use it in the filtering expression. Look at the "Simple filter" example in the ["Examples"](#) section.

This method can be used, if a simple value is needed. If the task is to display a list of values and inquire one or several from it, implementing this becomes difficult. You may think that, it is a simple task - showing a list of employees in the ListBoxControl control element and choosing one or several values. For implementing this, you need to use the script, which does the following:

- get the data source by its name;
- initialize data;
- fill the ListBoxControl with the data from data source;
- after choosing the employee, build a filter expression that will be used in the "Data" band.


FastReport can do this automatically. For this, automatic filtering is used, which we will observe now.











### Automatic filtering - how it works

Control is connected to the data column using the "DataColumn" property. If the control can display a list of values (for example, ListBoxControl), it fills with values from the indicated data column. This happens automatically when the dialogue is shown. Further, the user works with the dialogue, selects one or several items in the control and closes the dialogue. At this time, the data source which was indicated in the "DataColumn" property is filtered automatically.

The advantage of this method is that, you can use it in any report without writing any code.

Automatic filtering is supported by the following controls:

Icon	Name
	CheckBoxControl

	CheckedListBoxControl
	ComboBoxControl
	DataSelectorControl
	DateTimePickerControl
	ListBoxControl
	MaskedTextBoxControl
	MonthCalendarControl
	NumericUpDownControl
	RadioButtonControl
	TextBoxControl

## Filter operations

By default FastReport filters the data rows, which contain the value, equal to the value of the control. This behavior is defined in the "FilterOperation" property of the control. You can use the following operations:

Operation	Equivalent	Effect
Equal	=	Filter the value if it is equal to the control's value.
NotEqual	<>	Filter the value if it is not equal to the control's value.
LessThan	<	Filter the value if it is less than the control's value.
LessThanOrEqualTo	<=	Filter the value if it is less than or equal to the control's value.
GreaterThan	>	Filter the value if it is greater than the control's value.
GreaterThanOrEqualTo	>=	Filter the value if it is greater than or equal to the control's value.

For example, if the "FilterOperation" property of the control is set to "LessThanOrEqualTo" and you enter the value 5 in the control, then all the data rows will be chosen, for which the corresponding column value is less than or equal to 5.

For the data of "string" type, you can use extra operations:

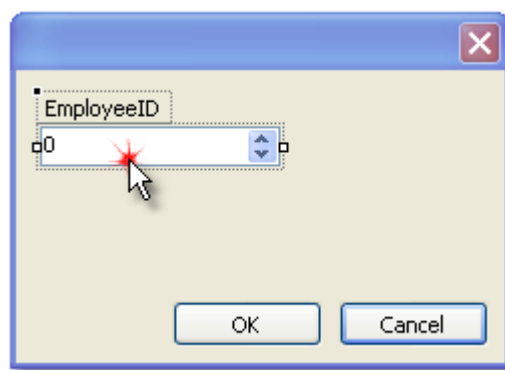
Operation	Effect
-----------	--------

Contains	Filter the value if it contains the control's value.
NotContains	Filter the value if it does not contain the control's value.
StartsWith	Filter the value if it starts with the control's value.
NotStartsWith	Filter the value if it does not start with the control's value.
EndsWith	Filter the value if it ends with the control's value.
NotEndsWith	Filter the value if it does not end with the control's value.

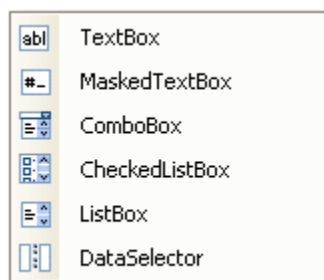
For example, if the "FilterOperation" property of the control is set to "StartsWith" and you enter the "A", then, all data rows whose corresponding data column's value starts with "A", will be chosen.

## Adding filter into a report

FastReport's dialogue designer has got very comfortable facilities for adding controls, which support the data filtering. For this, drag&drop the data column from the "Data" window onto the dialogue from. During this, FastReport creates the header (LabelControl control) and an actual control which will be used for data filtering:



The control type depends on the type of the data column. If the column is of string type, then after inserting it, you will be offered to choose the control type:



If you have inserted two similar controls, connected to the same data column, FastReport automatically configures the data range with the help of the "FilterOperation" property. The first control will have FilterOperation = GreaterThanOrEqual, the second - LessThanOrEqual. This will be done in case, if you insert a column which is not of string type.

As such, for adding data filtering into any report, you need to do the following:

- add a new dialogue into the report;
- drop onto the dialogue a data column, on which you want to filter the report.

## Filtering on data range

This method of filtering is comfortable for using when working with values, having a quantitative characteristic, for example, the cost. You can filter goods, having the cost less than or more than the given. In order to indicate, how to interpret the value entered in the element, use the "FilterOperation" property, looked at above.

Using two controls, which are connected to the same data column and have got different settings of the "FilterOperation" property, you can indicate the beginning and the ending of the data range. For the first control, you need to indicate the FilterOperation = GreaterThanOrEqual, for the second - LessThanOrEqual.

## Filtering on related data column

As we know, between two data sources, a relation can be set. See more details in the ["Data"](#) chapter. With the help of relation, it is possible to filter data in the source, by using a data column from a different source.

Assuming, you have placed on the dialogue a ListBoxControl and indicated the following data column in the "DataColumn" property:

```
Products.Categories.CategoryName
```

How will filtering work?

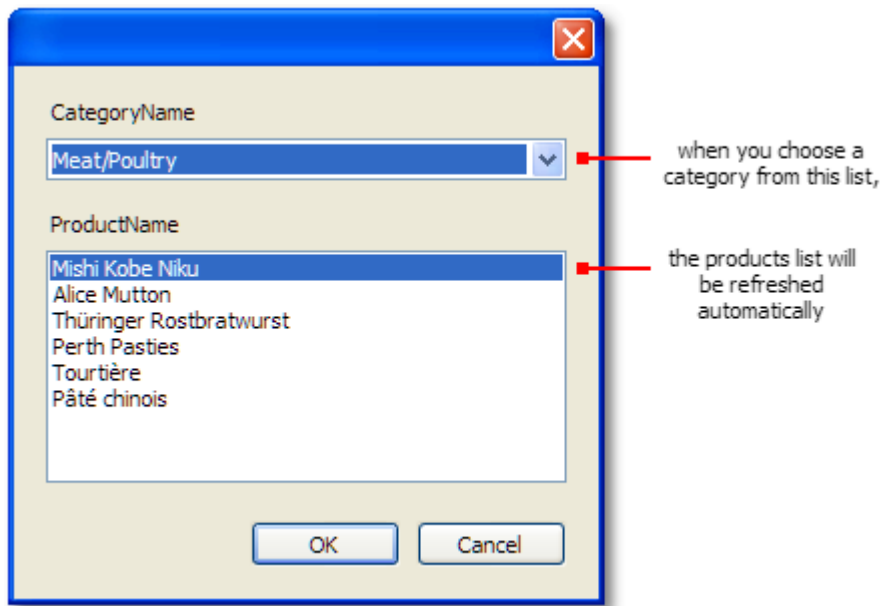
- when filling the control with values, the "CategoryName" column from the "Categories" data source will be used;
- the filter will be applied to the "Products" table. Those data rows will be filtered for which the following condition is correct:

```
the [Products.Categories.CategoryName] contains one of the values, selected by the user
```

## Filtering using cascading lists

A cascading list is a list with choices that change based on the value a user selects in another list. For example, you have two lists on a form - one with categories, another one with products. When you choose a category in the first list, you will see in the second list the products in a chosen category.





To create a cascading list, you need to use two data sources with master-detail relation between them (read more about data sources and relations in the ["Data"](#) chapter). Attach the master list to a column in the master data source; attach the detail list to a column in the detail data source. Also set the master list's "DetailControl" property to the detail list.

### Controlling the filtering from code

Even if automatic filtering is enough for many cases, you have the possibility of managing it manually. For this, the following methods and properties are used.

The "AutoFill" property controls the filling of controls with data. It is used by controls, which can show a list of values, for example, the `ListBoxControl`. Before showing a dialogue, FastReport fills such controls with data. By default, the property is set to true. If it is disabled, the control will not be filled, and you must do it yourself, by calling the "FillData" method:


```
ListBox1.FillData();
```

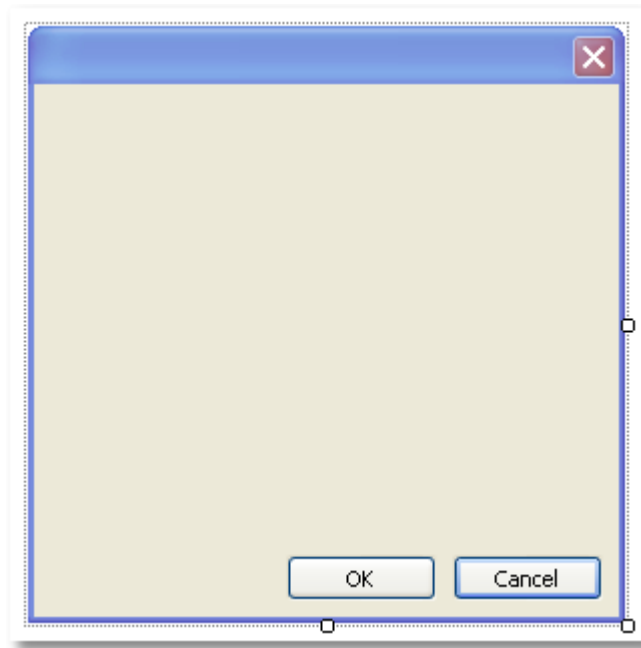
The "AutoFilter" property controls the data filtering. It is used by all controls. After the dialogue has been closed by the "OK" button, FastReport applies data filter automatically. By default, the property is set to true. If it is disabled, the filtering will not happen, and you must do it yourself, by calling the "FilterData" method:

```
ListBox1.FilterData();
```

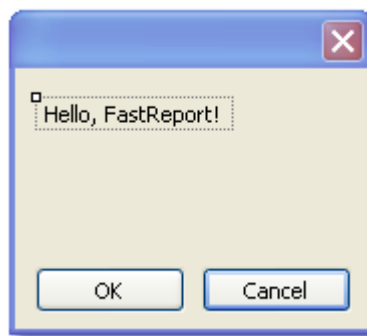
### Examples

#### Example 1. "Hello, FastReport!"

In this example, all we will do - create a dialogue, which will be showing greetings. Create a new report and add a dialogue to into. For this, press the  button on the toolbar:



On the dialog, place the LabelControl and set its "Text" property in the "Properties" window:

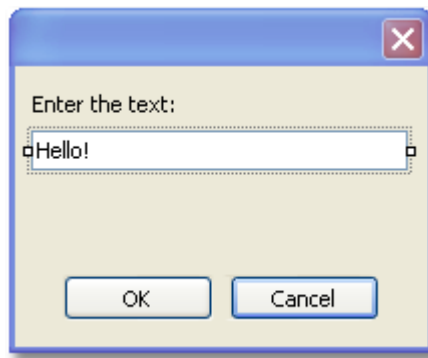


If you run the report, you will see the dialogue. Close it by the "OK" button, and the report will be built. If the dialogue is closed by the "Cancel" button or by the "X" button, the report will stop working, and you return to the designer.

### **Example 2. Ask for a text from the user**

In this example, we will create a dialogue, which will be asking for an arbitrary text from the user and later print the entered value in the report.

Create a new report and add a dialogue into it. On the dialogue, place LabelControl and TextBoxControl controls:



In the given case, the value we have entered is contained in the "Text" property of the TextBoxControl. In order to print this value in the report, add a new "Text" object on the "Report Title" band and write the following in it:

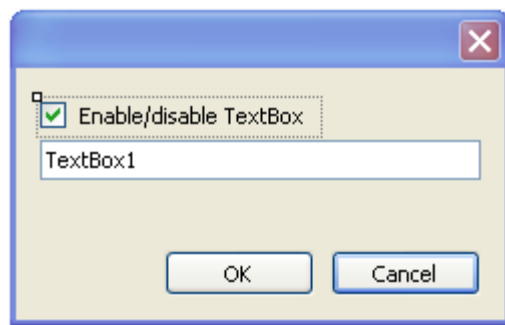
*You have entered: [TextBox1.Text]*


where TextBox1 is a name of the TextBoxControl control.

### Example 3. Handling dialog controls

By using the script and events of the controls, you can handle controls just like it is done in Visual Studio. We will show by example, how the CheckBoxControl can handle the TextBoxControl accessibility.

Create a new report and add a dialogue into it. On the dialogue, place the CheckBoxControl and TextBoxControl controls, as shown in the figure below:



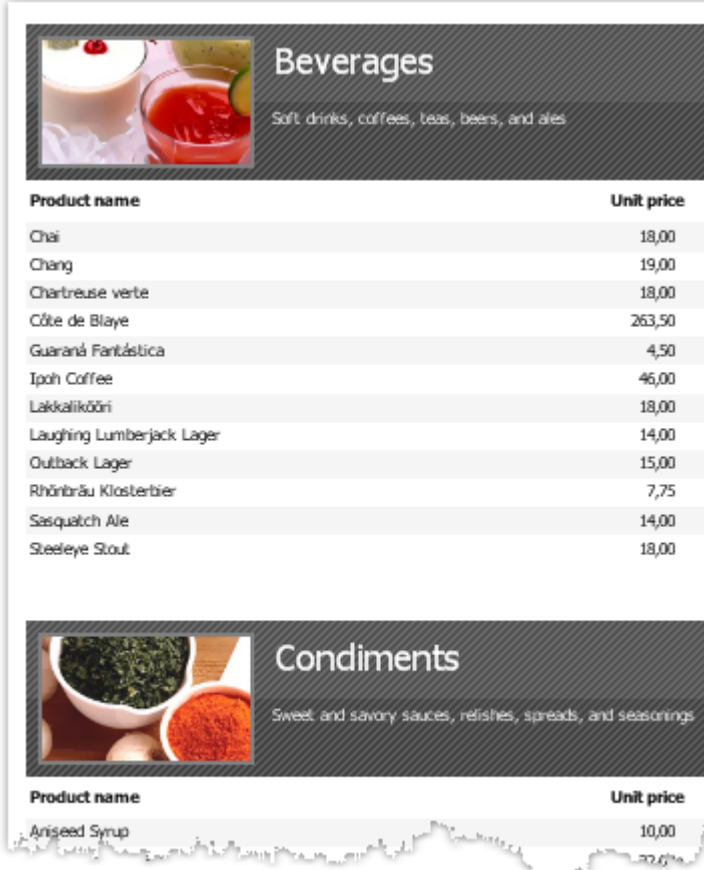
Now select the CheckBoxControl, open the "Properties" window and click the  button. Double click the "CheckedChanged" event, which is fired when changing the status of the checkbox. FastReport will create an empty handler for that event. Write the following code in it:

```
private void CheckBox1_CheckedChanged(object sender, EventArgs e)
{
    TextBox1.Enabled = CheckBox1.Checked;
}
```

If we run the report, we can enable or disable the TextBoxControl by the checkbox.

## Example 4. Handling report objects

Let us look at an example of a report, which prints a list of categories and products in each category:

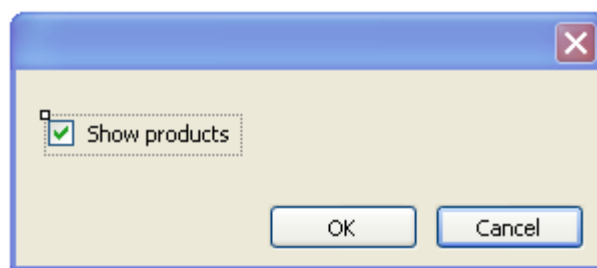


Product name	Unit price
Chai	18,00
Chang	19,00
Chartreuse verte	18,00
Côte de Blaye	263,50
Guaraná Fantástica	4,50
Ippoh Coffee	46,00
Lakkalikööri	18,00
Laughing Lumberjack Lager	14,00
Outback Lager	15,00
Rhönrau Klosterbier	7,75
Sasquatch Ale	14,00
Steeleye Stout	18,00

Product name	Unit price
Aniseed Syrup	10,00

We will show how to stop printing products and print just categories, with the help of the dialogue. For this, add a dialogue into the report:



Double click on the "OK" button. FastReport creates an empty event handler for the "Click" event. Write the following code in it:

```
private void btnOk_Click(object sender, EventArgs e)
{
    Data2.Visible = CheckBox1.Checked;
}
```

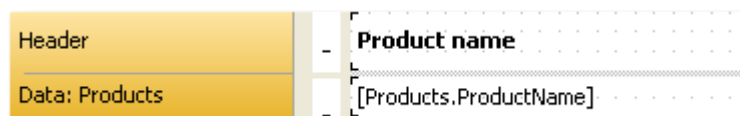
We will control the visibility of the band, which prints the product's list. In our example, this is a band with a name "Data2". If the report is run, and the checkbox is unchecked, we will have

the following result:

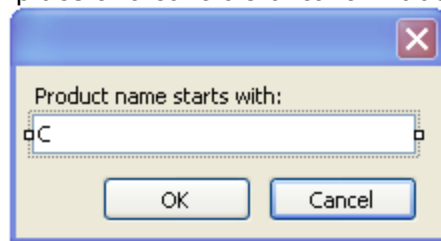


### Example 5. Simple filter

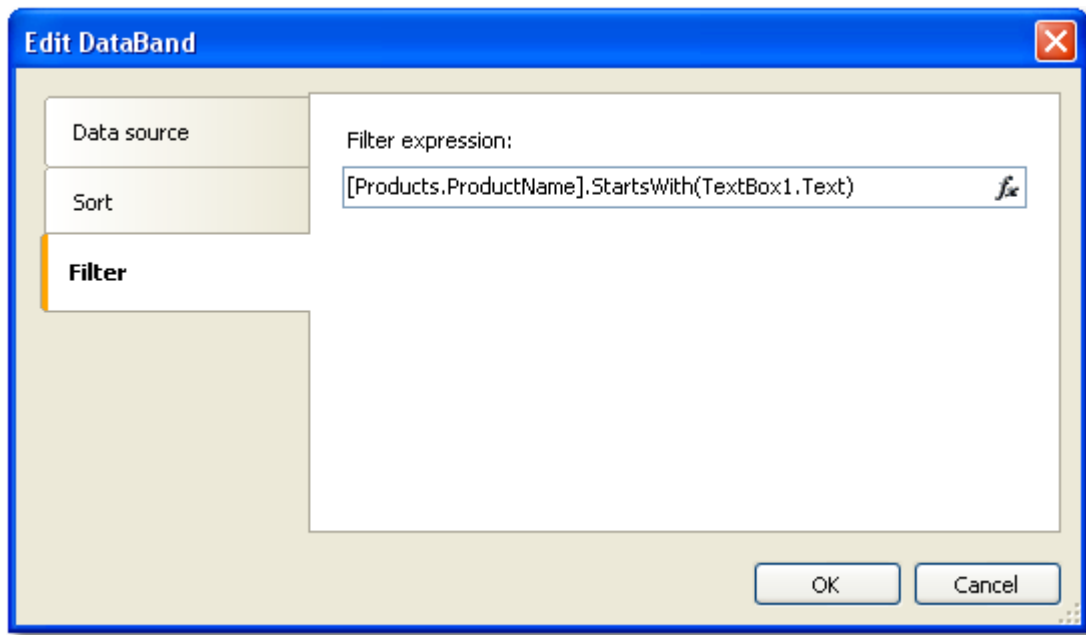
Let us look at the following report, which prints a products list:



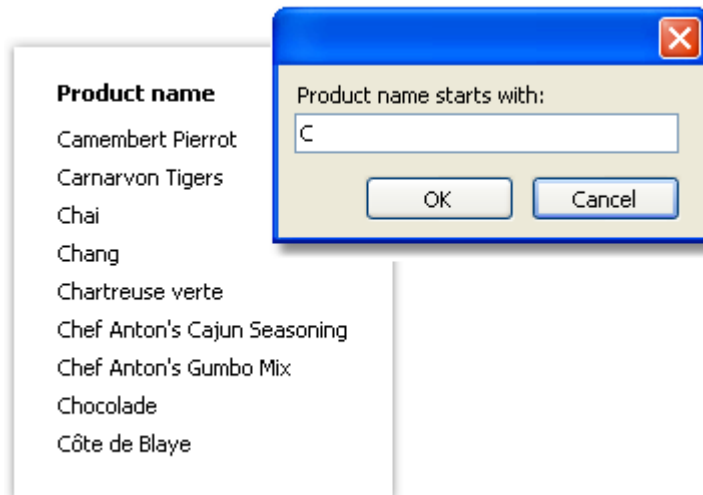
We will show in this example how to filter a products list according to the first letter of the product's name. During this, we will not use automatic data filtering facilities. For this, add a new dialog into the report and place two controls onto it - LabelControl and TextBoxControl:



Now open the "Data" band editor and indicate the following filter expression:



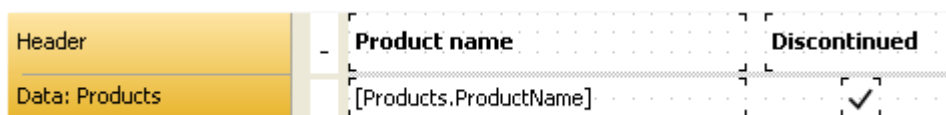
Run the report and make sure that everything works:




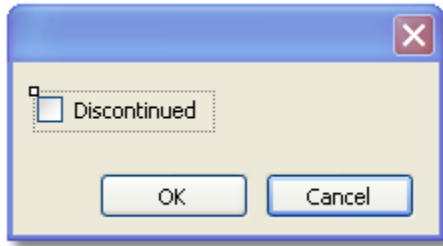
### Example 6. Automatic filtering

In this example we will show how to add a filter into the report which prints a product's list from the "Products" table. We will filter the data according to the "Products.Discontinued" column.

The report looks as follows:



Add a new dialogue into the report by pressing the  button on the toolbar, and drag the "Products.Discontinued" column from the "Data" window onto the dialogue form:



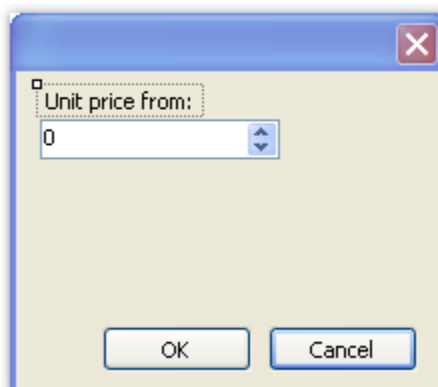
This is all we need to do - we did it just by two clicks. FastReport automatically connects the control to the data column.

Run the report and enable the Discontinued flag. After that, press the "OK" button, and you will see the report which contains only products with the Discontinued flag:

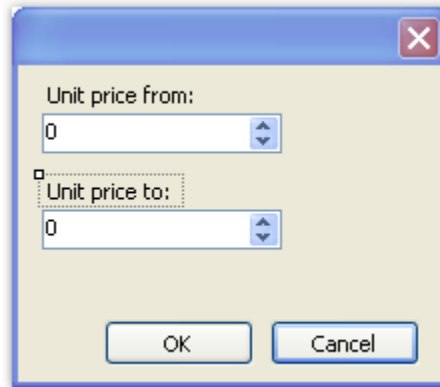
Product name	Discontinued
Alice Mutton	✓
Chef Anton's Gumbo Mix	✓
Guaraná Fantástica	✓
Mishi Kobe Niku	✓
Perth Pasties	✓
Rössle Sauerkraut	✓
Singaporean Hokkien Fried Mee	✓
Thüringer Rostbratwurst	✓

### Example 7. Automatic filtering by range

We will show with a report from the previous example, how to print products having the cost in the indicated range. For that we will add a dialogue into the report and drag "Products.UnitPrice" data column into it. After that we will correct the label text:



Now, in the same way, add one more "Products.UnitPrice" column and correct its header:



That is all we need to do, the rest of the work FastReport has done: connected the controls to the data column and set up their FilterOperation properties. The first control has got FilterOperation = GreaterThanOrEqual, the second - LessThanOrEqual.

Run the report and indicate the values, for example from 20 up to 30. When pressing the "OK" button, a report will be built. It contains products having values in the indicated range.

Product name	UnitPrice
Chef Anton's Cajun Seasoning	22,00
Chef Anton's Gumbo Mix	21,35
Fløtemysost	21,50
Grandma's Boysenberry Spread	25,00
Gravad lax	26,00
Gustaf's Knäckebröd	21,00
Louisiana Fiery Hot Pepper Sauce	21,05
Maxilaku	20,00
Nord-Ost Matjeshering	25,89
Pâté chinois	24,00
Queso Cabrales	21,00
Sirop d'érable	28,50
Tofu	23,25
Uncle Bob's Organic Dried Pears	30,00

### Example 8. Filtering by related data column

In this example, we will use a column from a related data source to perform data filtering.

We will look at a "Simple list" report type, which prints products list. Category name is printed beside each product. This is done with the help of the relation:

```
[Products.Categories.CategoryName]
```

The report will be as follows:

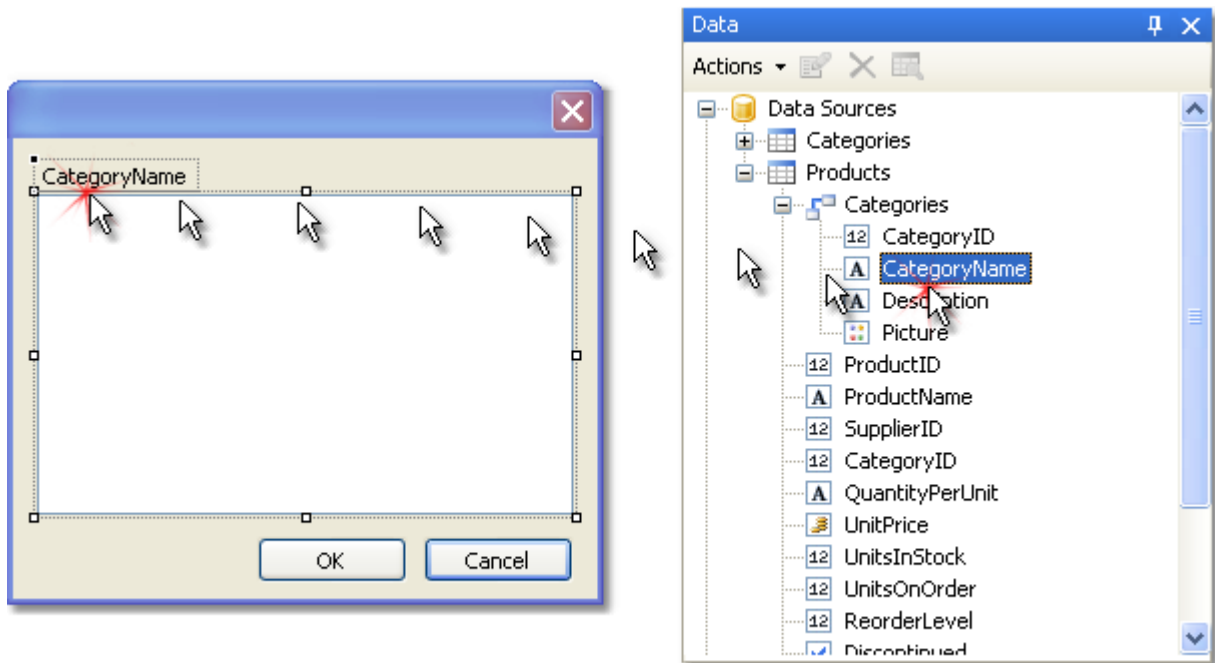


Report Title	<b>PRODUCT CATALOG</b>		
Header	<b>Product name</b>	<b>Category name</b>	<b>Unit price</b>
Data: Products	[Products.ProductName]	[Products.Categories.CategoryName]	[Products.UnitPrice]

When we run the report, we will see the following:

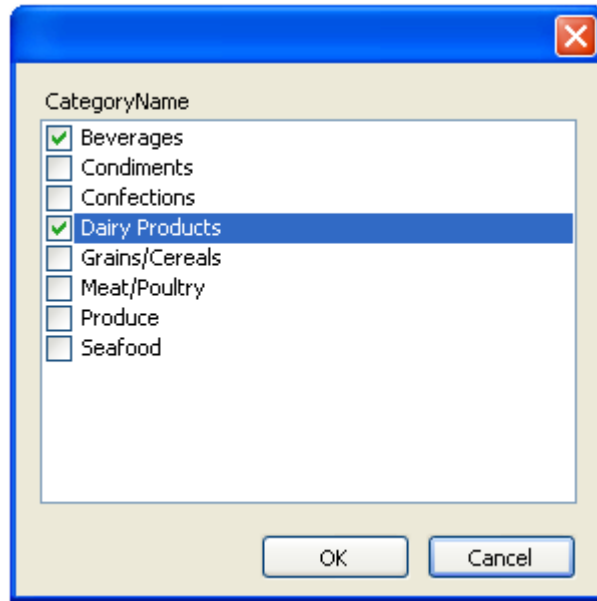
<b>PRODUCT CATALOG</b>		
<b>Product name</b>	<b>Category name</b>	<b>Unit price</b>
Alice Mutton	Meat/Poultry	39,00
Aniseed Syrup	Condiments	10,00
Boston Crab Meat	Seafood	18,40
Camembert Pierrot	Dairy Products	34,00
Carnarvon Tigers	Seafood	62,50
Chai	Beverages	18,00
Chang	Beverages	19,00
Chartreuse verte	Beverages	18,00
Chef Anton's Cajun Seasoning	Condiments	22,00
Chef Anton's Gumbo Mix	Condiments	21,35
Chocolade	Confections	12,75

Let us add filtering by category name. For this, add a new dialogue and drag the "Products.Categories.CategoryName" column onto it:



When creating control, you will be required to select its type. Choose `CheckedListBoxControl`.

If we run the report, we will see the following dialogue:



Choose several categories and click the "OK" button. After this, the data will be filtered and you will see the following report:

PRODUCT CATALOG		
Product name	Category name	Unit price
Camembert Pierrot	Dairy Products	34,00
Chai	Beverages	18,00
Chang	Beverages	19,00
Chartreuse verte	Beverages	18,00
Côte de Blaye	Beverages	263,50
Fløtemysost	Dairy Products	21,50
Geitost	Dairy Products	2,50
Gorgonzola Telino	Dairy Products	12,50
Guaraná Fantástica	Beverages	4,50
Gudbrandsdalsost	Dairy Products	36,00
Ipoh Coffee	Beverages	46,00
Le Kaffee	Beverages	18,00

As seen, only products have remained, which are in the chosen category.

# Chapter

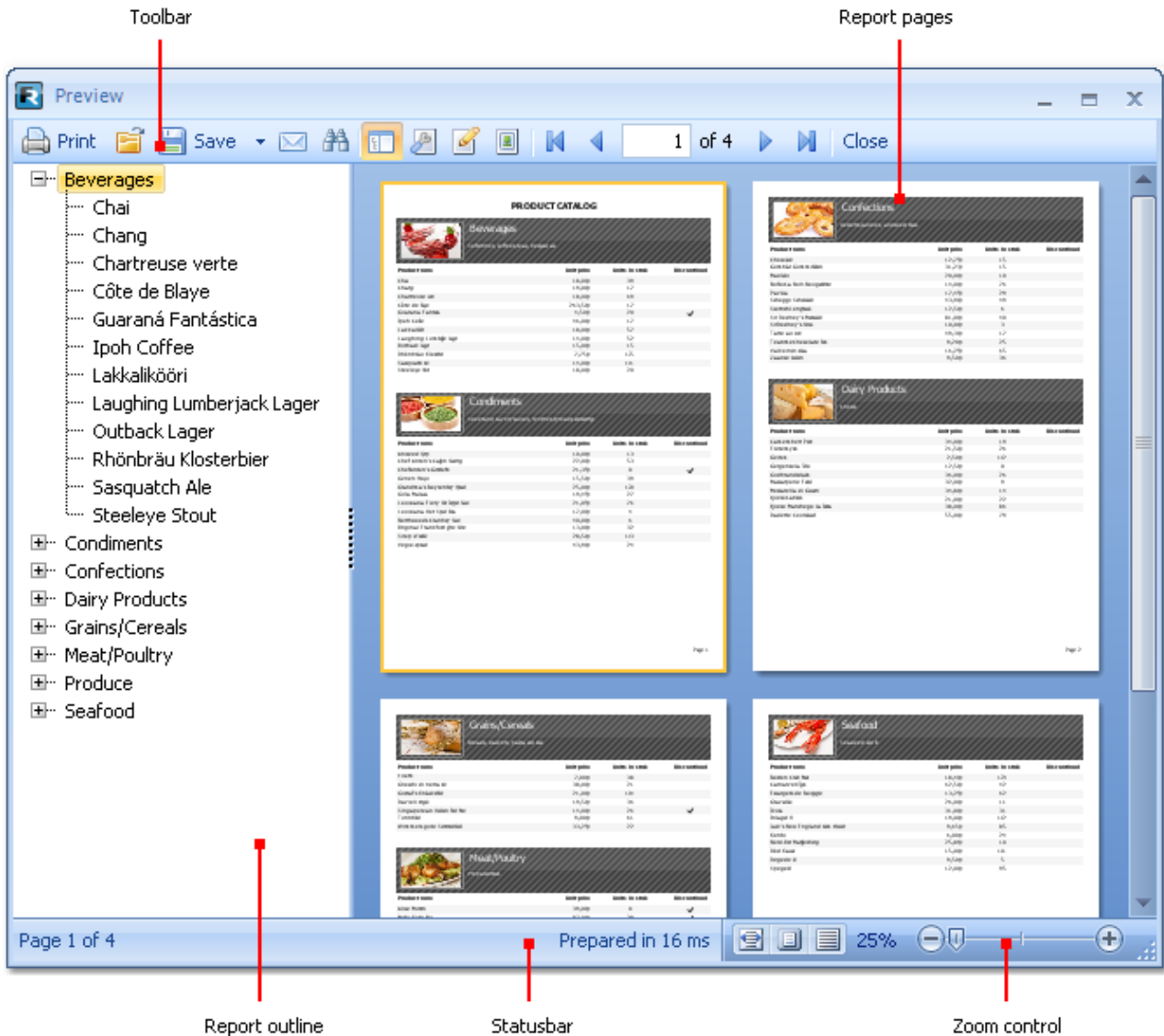
---



**Preview, print, export**




# Preview, print, export











A built report can be shown on the screen, printed on the printer or exported into one of the supported formats. All these can be done in the preview window:



On the toolbar, you can find the following buttons:




Button	Description
	Print the report.
	Open the prepared report file in FPX format.
	Save the report in one of the supported formats.

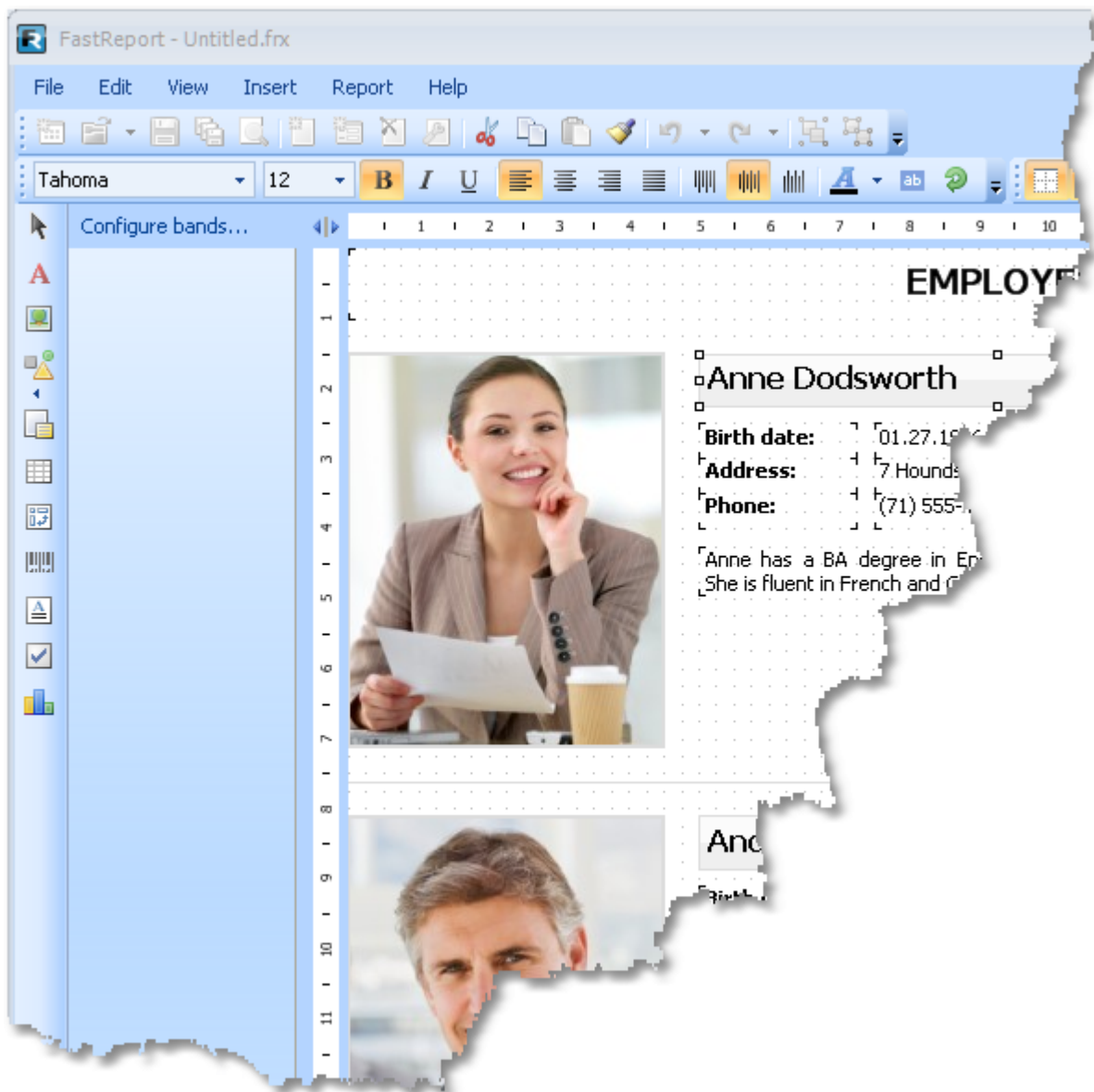
	Send the report by email.
	Text search in the report.
	Shows or hides report outline.
	Page settings.
	Edit current report page.
	Watermark settings.
	Navigate to the first page.
	Navigate to the previous page.
1	Navigate to the indicated page. Enter the page number and press Enter.
	Navigate to the next page.
	Navigate to the last page.

You can use the following keyboard control:

Key	Description
Ctrl+P	Print the report.
Ctrl+F	Text search.
Arrows	Scroll the preview.
PageUp, PageDown	Page up/down.
Home	Navigate to the first page.
End	Navigate to the last page.
Esc	Close the preview window.


## Editing the report

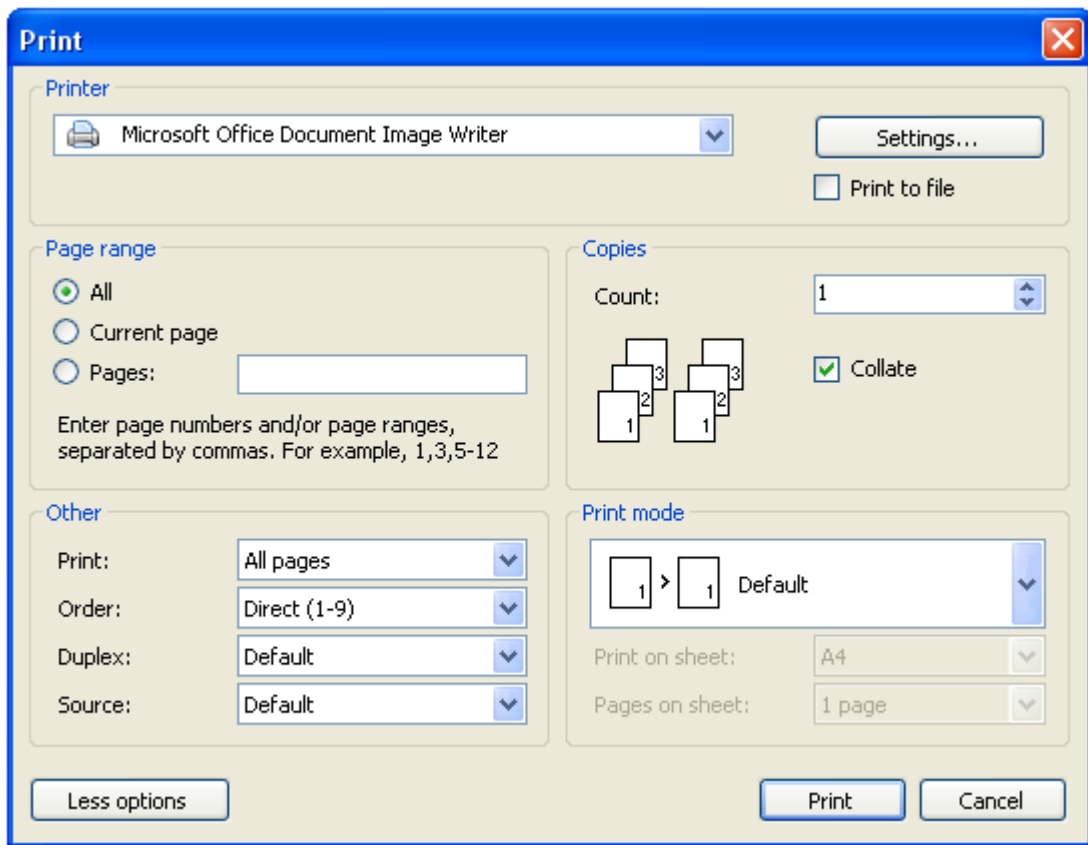
For editing a page of a prepared report, click the  button in the preview window. During this, the current page will loaded in the report designer, where you can do whatever you want with it:



After editing, close the designer. When doing this, you will be asked to save the changes in the report page.

## Printing the report

In order for print a report, press the  button (or press a combination of Ctrl+P). You will see the print dialogue:



We will look at the settings accessible in this dialogue. The button "More/Less" allows showing the whole dialogue or just the basic settings. By default, a dialogue is shown in a simple form.

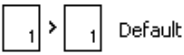

"Printer" group: here, you can choose the printer, change its settings ("Settings..." button) and choose print to the file.

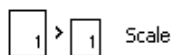
"Page" group: here you can choose, which pages to print (all, current or the given page number).

"Copy" group: here you can set the number of copies and choose the order of the pages in the copies ("Collate"):

"Others" group: here you can choose, which pages to print (all, even, odd), choose the order of printing (direct, reverse), set up the duplex printing (if your printer supports it) and choose the paper source.

"Print mode" group allows choosing one of the printing modes:

Mode	Description
	The printer prints on a paper, indicated in the report. One report page corresponds with one printed sheet.
	Use this mode, if you need to print A3 report on a A4 format paper. One report page will produce two printed sheets. When using this mode, you have to choose the paper format from the "Print on sheet" list.

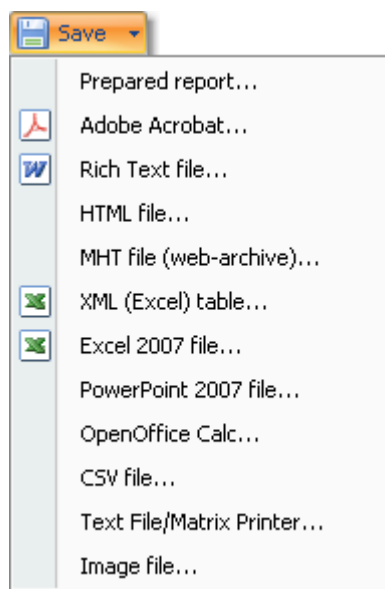


Use this mode, if you need to print A4 report having on a A3 format paper. On one printing sheet, you can print 1, 2, 4 or 8 report pages. When using this mode, you need to choose the format of the paper on which you want to print, from the "Printing on sheet" list, and also indicate the number of pages in the "Pages on sheet" list.

After pressing the "Print" button, printing of the report will start. If the "Print to file" flag is chosen, then the name of the file will be requested and the report will be saved in that file (file with a PRN extension).

## Exporting the report

FastReport allows exporting the built report into different formats. At the moment, export to 11 formats is supported: PDF, RTF, HTML, MHT, Excel (XML), Excel 2007, PowerPoint 2007, OpenOffice Calc, CSV, TXT, image file. For choosing export, press the "Save" button in the preview window and choose the export:



## Saving in FPX format

FPX format is FastReport's "native" format. The advantages of this format are as follows:

- saving the report without losing the quality. Opening an already saved file, you can do all operations with it, like print, export, editing;
- compact format based on XML, compressed with the help of ZIP;
- when needed, the report file can be unpacked by any archiver that supports the ZIP-format and corrected manually in any text editor.

The only thing lacking with the format is that, to view it, you need to have FastReport.Net.

In order to save in the FPX format, press the "Save" button in the preview window and choose the "Prepared report" file type. For opening already saved files, press the "Open" button.

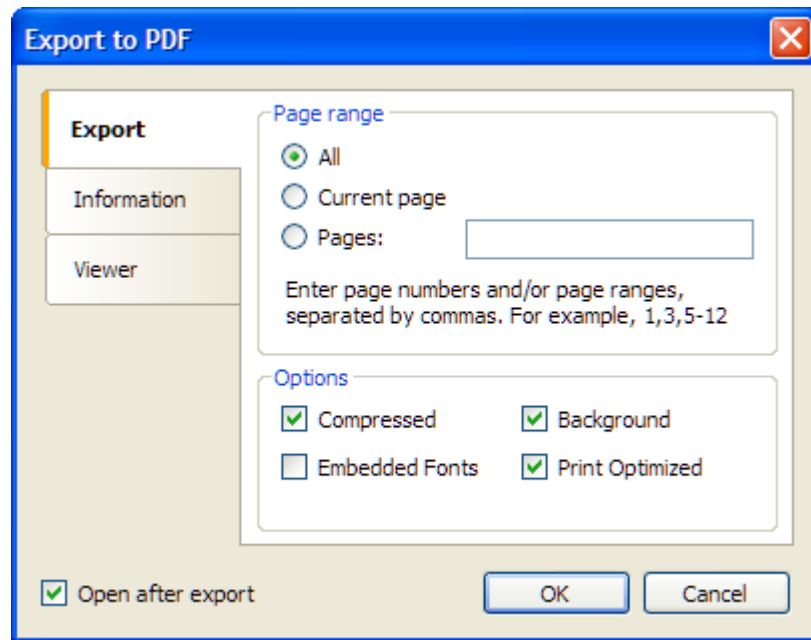


## Export to Adobe Acrobat (PDF)

PDF (Portable Document Format) is a platform-independent format of electronic documents created by Adobe Systems. The free Acrobat Reader package is used for viewing. This format is rather flexible – it allows the inclusion of necessary fonts, vector and bitmap images; it allows transferring and storage of documents intended for viewing and further printing.

Export method is a layered one.

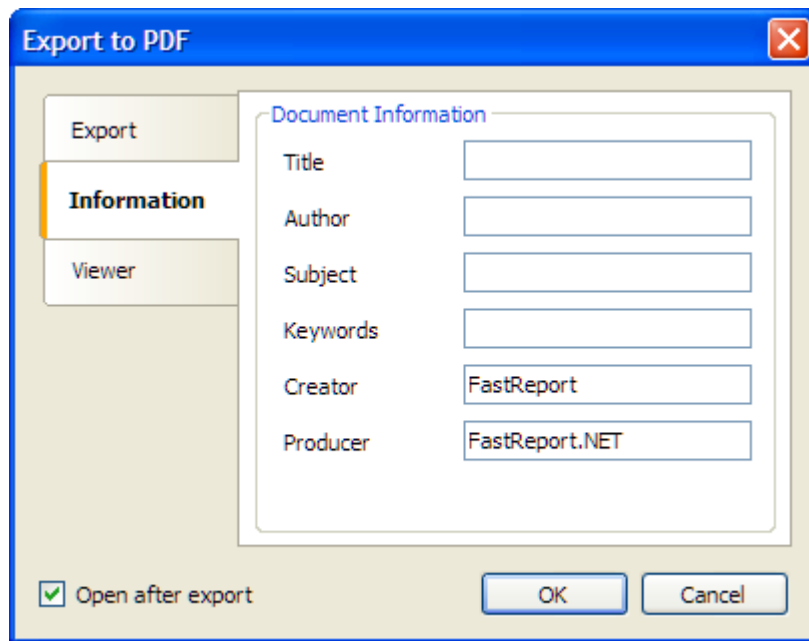
When exporting to Excel, there will be a dialogue window for setting parameters of the output file:



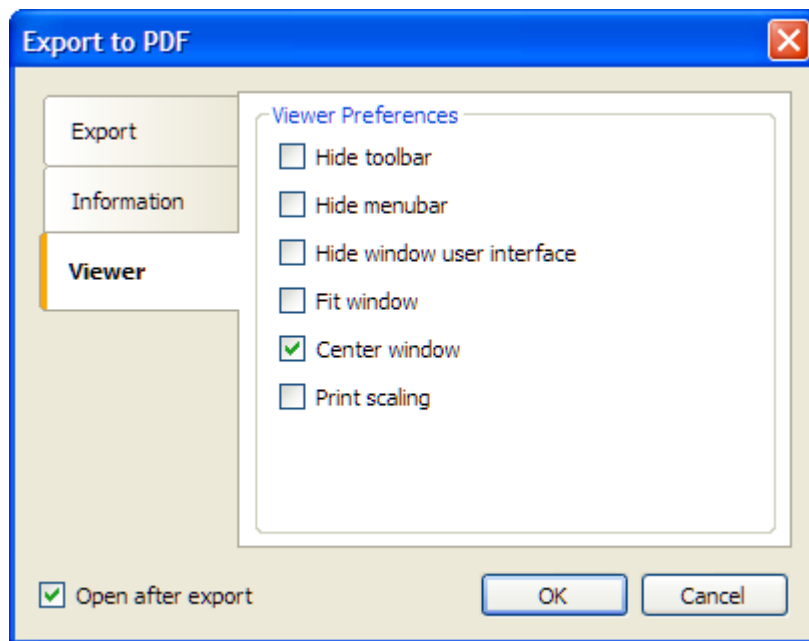
Export parameters:

- "Compressed" - output file is compressed. It reduces file size but increases export time;
- "Embedded fonts" - all fonts used in a report will be included into PDF file. This will significantly increase the file size;
- "Background" - the page watermark will be exported as an image. This will significantly increase the file size;
- "Print optimized" - output of all graphics objects (such as pictures, charts) in high resolution for further printing.

On the "Information" tab, you may fill in the document information fields:



On the "Viewer" tab, you may set up some options related to Adobe Acrobat document viewer:

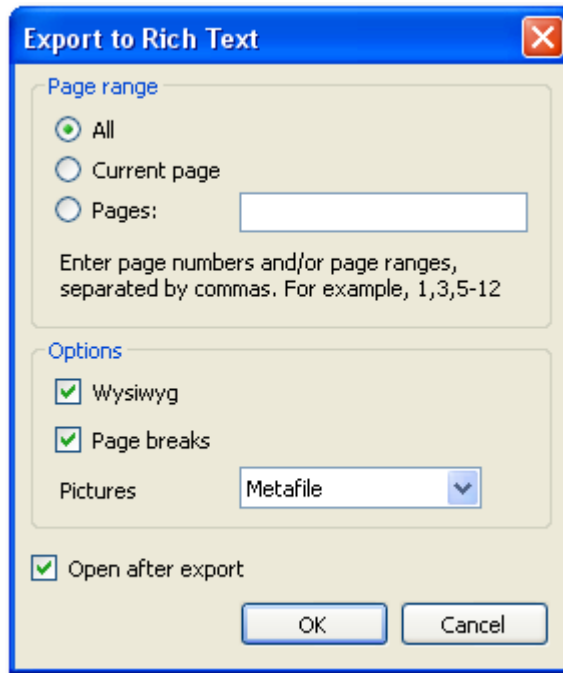


## Export to Word (RTF)

RTF (Rich Text Format) was developed by Microsoft as a standard format for exchanging text information. At the moment RTF-documents are compatible with many new text editors and operation systems.

Export method: tabular.

When exporting to RTF, there will be a dialogue window for setting parameters of the output file:



Export parameters:

- "Wysiwyg" - the result will be as close to the report as possible. If this option is disabled, FastReport will reduce the number of rows and columns in the resulting;
- "Page breaks" - enables page breaks in the RTF file;
- "Pictures" - select the format of pictures in the RTF file. Note that "Metafile" format is best for displaying of such report objects as MSChartObject and ShapeObject.

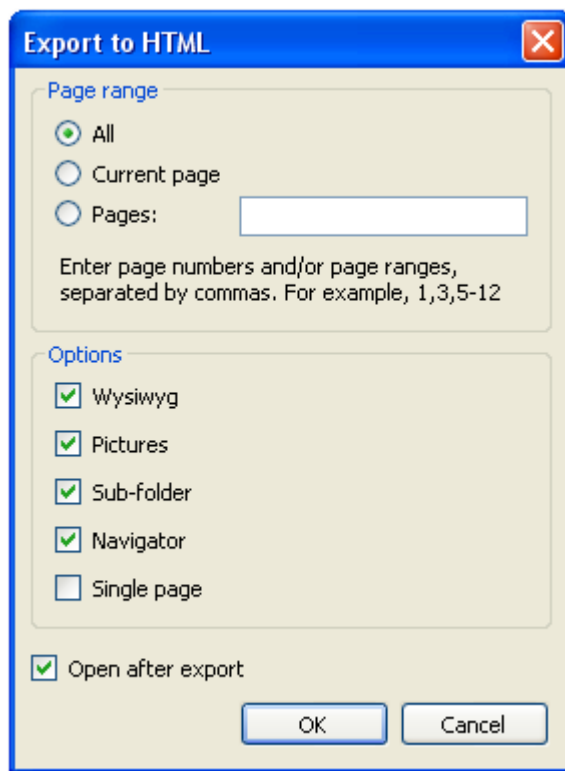
Appearance and size of the resulting file depends on the report template (see the ["Recommendations on report development"](#) section).

## Export to HTML

HTML (HyperText Markup Language) is the predominant markup language for Web pages. It provides a means of describing the structure of text-based information in a document - by denoting certain text as links, headings, paragraphs, lists, and so on - and to supplement that text with interactive forms, embedded images, and other objects.

Export method: tabular.

When exporting into HTML, a dialogue window will be offered for setting the parameters:



Export parameters:

- "Wysiwyg" - the export result will be as close to the report as possible;
- "Pictures" - enables to export pictures;
- "Sub-folder" - all extra files are saved in a separate folder called ".files";
- "Navigator" - creates a special navigator for navigating on pages;
- "Single page" - all pages will be saved in one file.

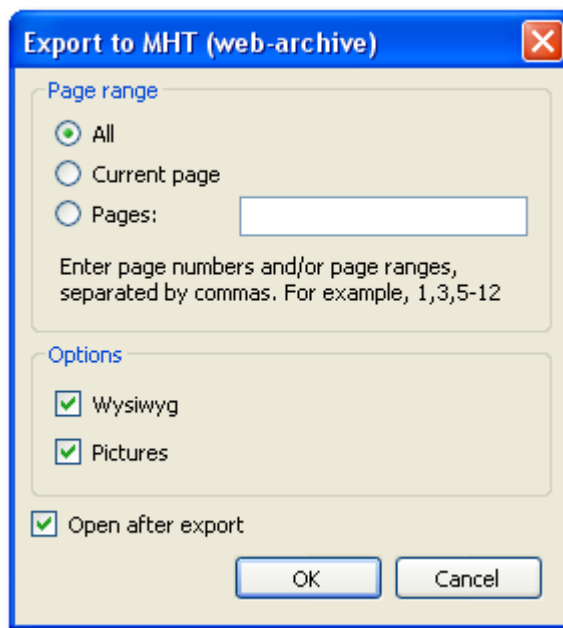
Appearance and size of the resulting file depends on the report template (see the ["Recommendations on report development"](#) section).

### Export to MHT (web archive)

MHT, short for MIME HTML, is a web page archive format used to bind resources which are typically represented by external links (such as images, Flash animations, Java applets, audio files) together with HTML code into a single file. The content of an MHT file is encoded as if it were an HTML e-mail message, using the MIME type multipart/related.

Export method: tabular.

When exporting into MHT, a dialogue window will be offered for setting the parameters:



Export parameters:

- "Wysiwyg" - the export result will be as close to the report as possible;
- "Pictures" - enables to export pictures.

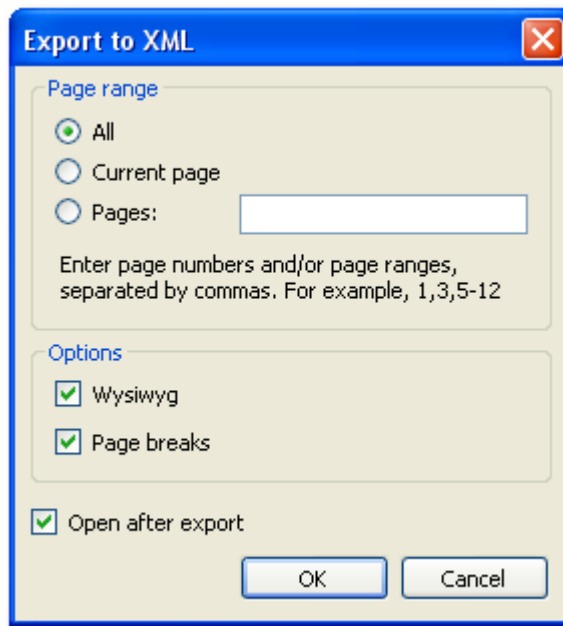
Appearance and size of the resulting file depends on the report template (see the ["Recommendations on report development"](#) section).

## Export to Excel (XML)

Excel is an application for working with electronic worksheets. It is included into Microsoft Office.

Export method: tabular.

When exporting to Excel, there will be a dialogue window for setting parameters of the output file:



Export parameters:

- "Wysiwyg" - the result will be as close to the report as possible. If this option is disabled, FastReport will reduce the number of rows and columns in the resulting file;
- "Page breaks" - enables page breaks in the resulting file.

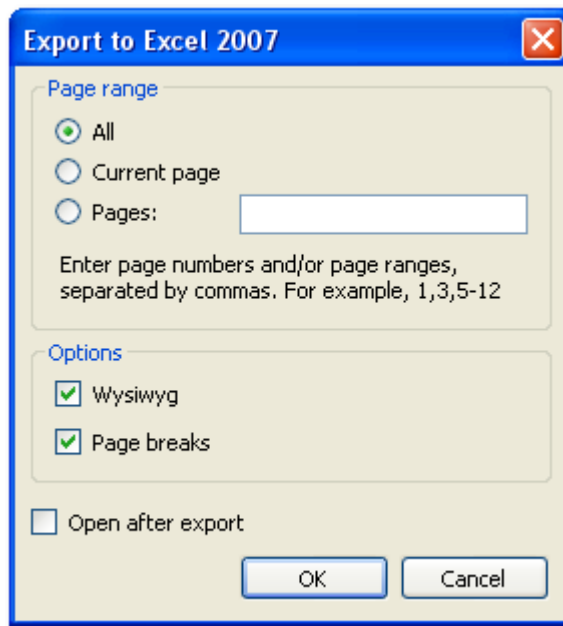
Appearance and size of the resulting file depends on the report template (see the ["Recommendations on report development"](#) section).

## Export to Excel 2007

Excel 2007 is an application for working with electronic worksheets. It is included into Microsoft Office 2007.

Export method: tabular.

When exporting to Excel, there will be a dialogue window for setting parameters of the output file:



Export parameters:

- "Wysiwyg" - the result will be as close to the report as possible. If this option is disabled, FastReport will reduce the number of rows and columns in the resulting file;
- "Page breaks" - enables page breaks in the resulting file.

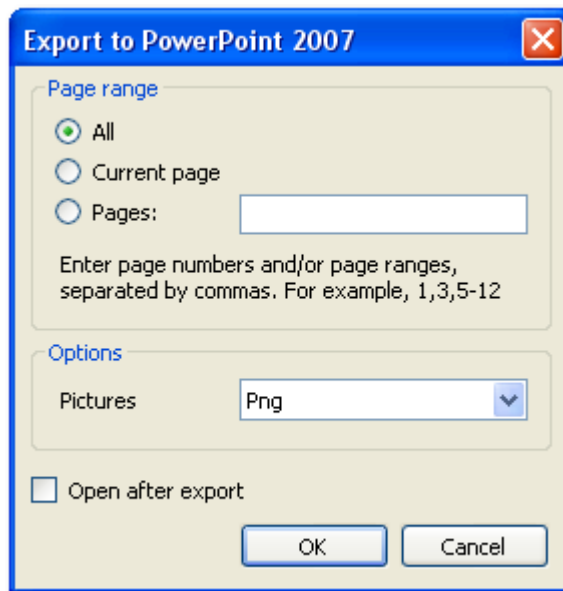
Appearance and size of the resulting file depends on the report template (see the ["Recommendations on report development"](#) section).

## Export to PowerPoint 2007

PowerPoint 2007 is an application for working with electronic presentations. It is included into Microsoft Office 2007.

Export method is a layered one.

When exporting to PowerPoint, there will be a dialogue window for setting parameters of the output file:



Export parameters:

- "Pictures" - select the format of pictures in the resulting file.

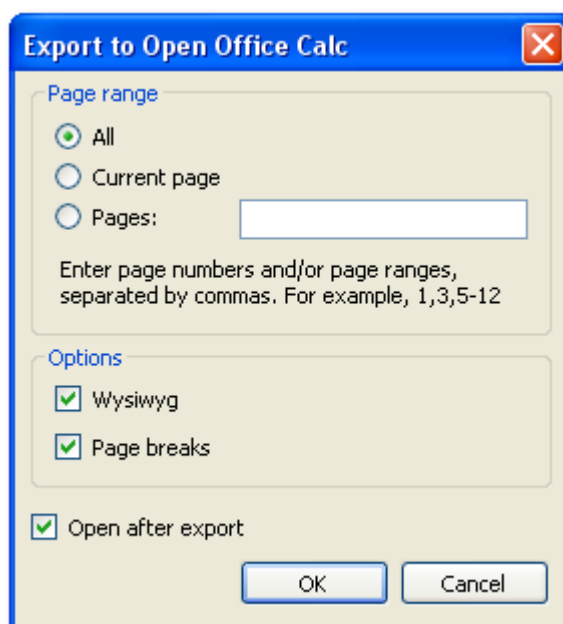
## Export to OpenOffice Calc

OpenDocument Format (ODF, OASIS Open Document Format for Office Application) was designed by OASIS and based on XML format used in OpenOffice.

FastReport supports export to table (.ods file). These files can be opened in OpenOffice.

Export method: tabular.

When exporting to OpenOffice Calc, there will be a dialogue window for setting parameters of the output file:





Export parameters:

- "Wysiwyg" - the result will be as close to the report as possible. If this option is disabled, FastReport will reduce the number of rows and columns in the resulting file;
- "Page breaks" - enables page breaks in the resulting file.

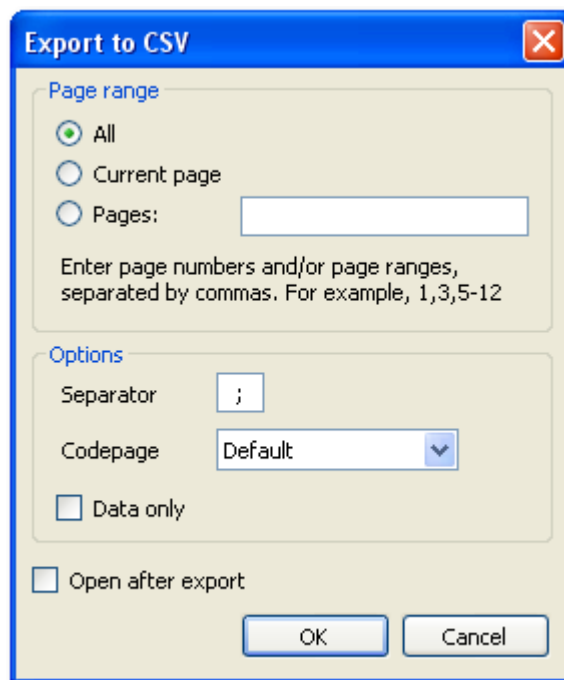
Appearance and size of the resulting file depends on the report template (see the ["Recommendations on report development"](#) section).

## Export to CSV

The CSV file is used for the digital storage of data structured in a table of lists form. Each line in the CSV file corresponds to a row in the table. Within a line, fields are separated by commas, each field belonging to one table column. CSV files are often used for moving tabular data between two different computer programs, for example between a database program and a spreadsheet program.

Export method: tabular.

When exporting to CSV, there will be a dialogue window for setting parameters of the output file:



Export parameters:

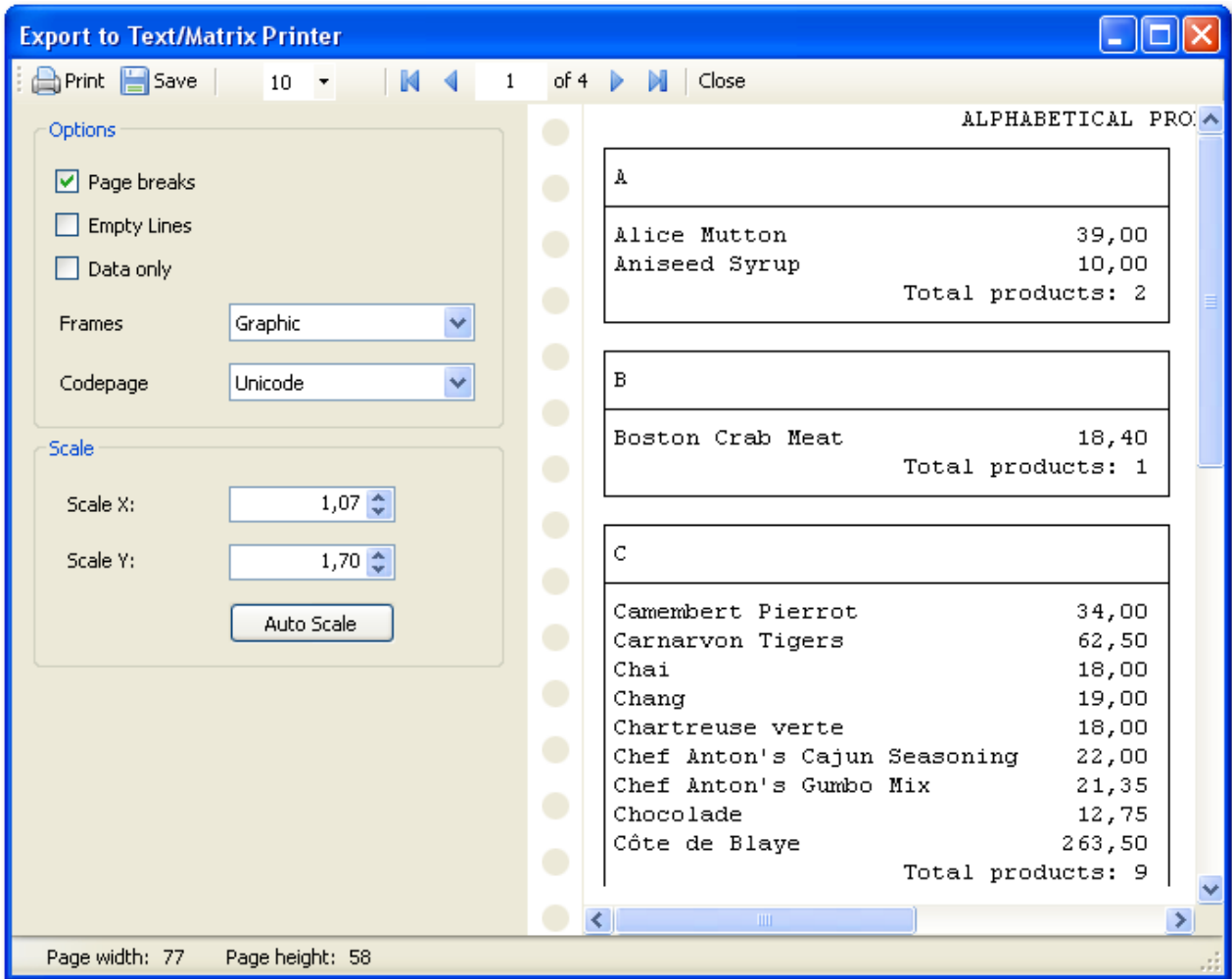
- "Separator" - the field separator character;
- "Codepage" - codepage used to encode the text in resulting file. The "Default" codepage refers to Windows default codepage. Note that Excel does not support unicode codepages;
- "Data only" - enable this checkbox to export objects laying on Data band only.

## Export to TXT

TXT is a regular text file that can be opened in any text editor, or printed to a dot-matrix printer.

Export method: tabular.

When exporting to TXT, there will be a dialogue window for setting parameters of the output file:



Export parameters:

- "Page breaks" - enables page breaks in the resulting file;
- "Empty Lines" - enables empty lines in the resulting file;
- "Data only" - enable this checkbox to export objects laying on Data band only;
- "Frames" - type of object's borders. Select "None" if you don't want to export borders;
- "Codepage" - codepage used to encode the text in resulting file;
- "Scale X" - horizontal scale;
- "Scale Y"- vertical scale;
- "Auto Scale" - calculates scale X and scale Y automatically to avoid data loss.

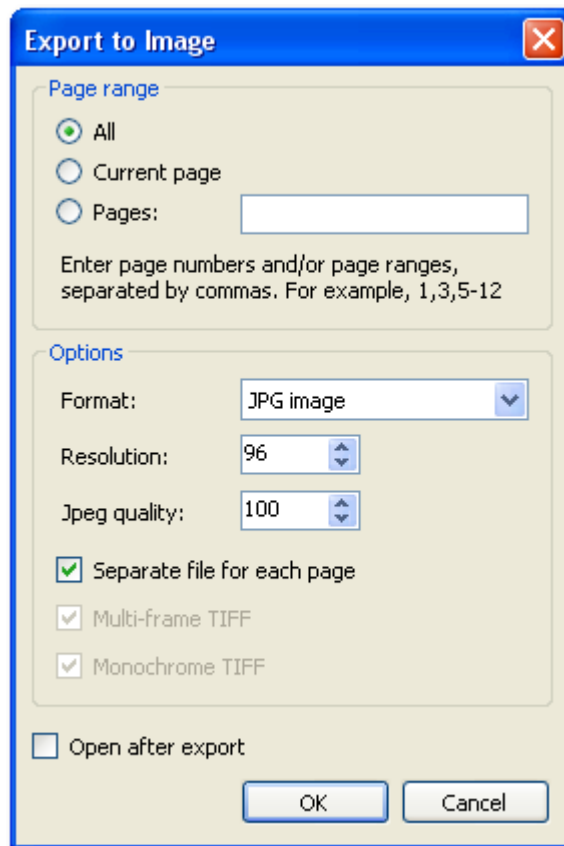
## Export to picture

FastReport allows exporting information into the following graphical formats:

- BMP
- PNG
- JPG
- GIF
- TIFF
- Windows Metafile (EMF,WMF)

Exporting method: drawing.

When exporting into picture files, a dialogue window will be offered for setting the parameters:



Export parameters:

- "Resolution" - resolution of the graphical image. Use 96dpi for displaying, 300dpi for printing. When exporting into the TIFF format, you will be able to set separate values for horizontal and vertical resolution;
- "Jpeg quality" - JPG file compression level. This option is used only when exporting into the Jpeg format;
- "Separate file for each page" - if the option is enabled, then each report page will be exported into a separate file, the name of the file will be formed on the basis of the chosen page with the given number;
- "Multi-frame TIFF" - this option produces the multi-frame TIFF file. It is used only when exporting into the TIFF format;
- "Monochrome TIFF" - this option produces the monochrome TIFF file. It is used only when

exporting into the TIFF format.

When exporting several pages into one file (when the "Separate file for each page" option is disabled), the export will use a lot of CPU/Memory resources.

## Report design recommendations

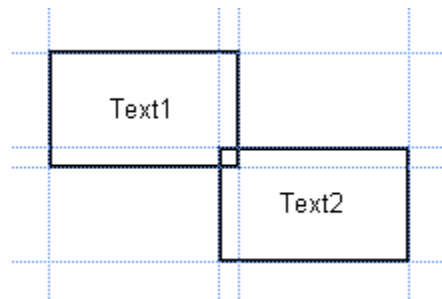
In this chapter, special design requirements of reports intended for export to other data formats will be discussed.

FastReport allows a great number of ways to manipulate objects during report creation. This gives the advantage of fast development of any reports and their further printing. Printed document will look just as on display. And this is the primary intent of FastReport report generator usage.

The downside of such development freedom is the complexity of exporting the FastReport document to different data formats, which have their own limits and requirements for information presentation, and are sometimes rather complex. Many formats, such as HTML, XLS or RTF, use table data presentation. These formats do not allow cell crossing or arranging in layers when table marking.

Export filters, as a rule, take into account these requirements. This is carried out by special algorithm which takes object crossings into account and places them optimally. At object crossing, there are new columns and rows in the resulting output table appear. That is necessary for getting maximum resemblance between the result and original report. A large number of crossed objects in report design, leads to an increased number of columns and rows in the resulting table, that affects the file size and its complexity.

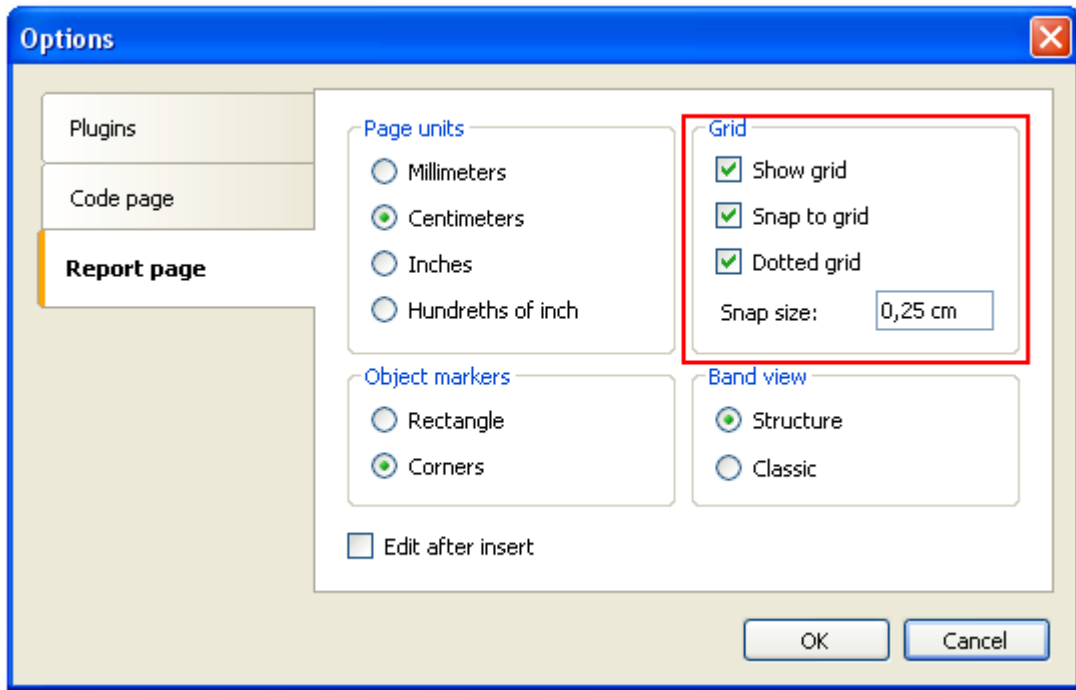
The quality of the export depends greatly on competent design of initial report. Let us look at the following example:



There is a slight crossing of two objects placed one under another on the same band. The number of records on report forming was 150. On export to RTF format 450 rows will be created (150 rows for each object and 150 ones for crossing). If we remove crossing, there will be only 300 rows in the resulting table. In large reports and on huge number of objects the difference will be really tremendous. That, of course, will affect output file size.

When creating tables in report, pay attention to neighboring cell's borders. It is important that cells do not cross and arrange in layers. Export filter algorithm will cut off cells but export result may be far from desirable (you will see not exactly what you wanted to). Arrange objects in such a way that they are placed in line vertically as well as horizontally. Guidelines can help to perform this.

The grid alignment can also be helpful in case of cells overlapping. Enable grid alignment in designer options. In order to simplify alignment you can extend grid pitch. Setting of grid pitch and alignment can be found in the "View|Options..." menu:



For text framing it is better to use text object's border instead of single graphic objects such as lines, rectangles, etc. Try not to use background objects under transparent text objects.

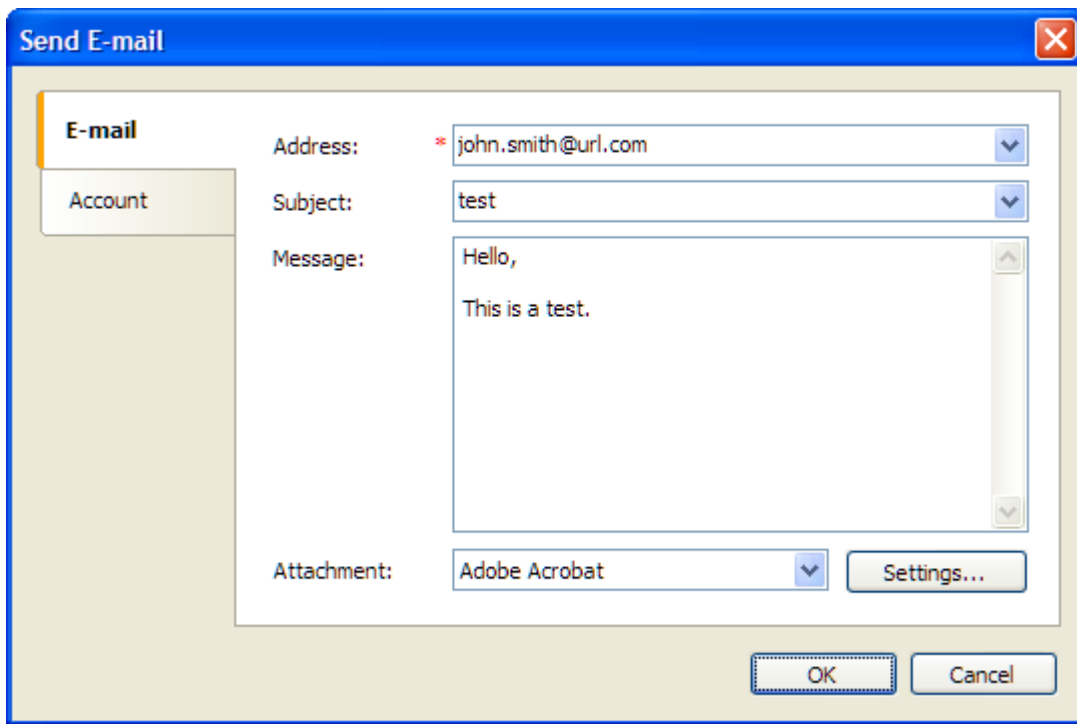
Applying these simple rules will help you to create a report which will look perfect after being exported to any table-based format.

## Sending the report by email

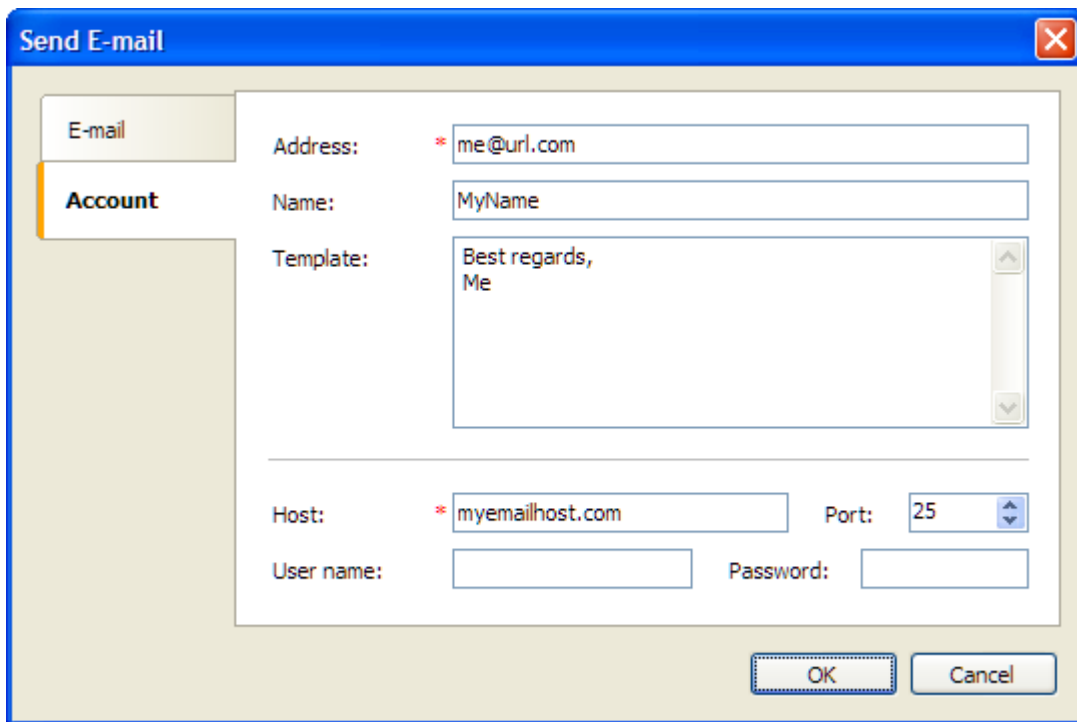
FastReport allows you to send a prepared report by email. It may work in two modes:

Mode	Description
SMTP	This is default mode. To send an email, you don't need any external programs.
MAPI	You may turn on this mode programmatically. To do this, set <code>Config.EmailSettings.UseMAPI = true</code> , or, if you use <code>EnvironmentSettings</code> component, set its <code>EnvironmentSettings.EmailSettings.UseMAPI</code> property to true.  To send an email, FastReport uses the default email client such as Outlook Express. This client must support the MAPI protocol.

To send an email, you need to specify a recipient's email address. Also you need to specify the subject and email body, but this is not required. At the bottom of the dialog, select the format of your report - it will be attached to the message:



If you use the SMTP mode, you need to set up an account. It is necessary to do only once. Once you have done it, FastReport will save the parameters in the configuration file. The parameters can be found on the "Account" tab. All obligatory fields are marked by red asterisk:



If your host server requires an authentication, you need to fill in "User name" and "Password" fields as well.