



QtRPT

Version 1.4.4

Programmer: Aleksey Osipov

Email: aliks-os@ukr.net

QtRPT is the easy-to-use report engine written in C++ QtToolkit. It allows combining several reports in one XML file. For separately taken field, you can specify some condition depending on which this field will display in different font and background color, etc. The project consists of two parts: report library QtRPT and report designer application QtRptDesigner. Report file is a file in XML format. The report designer makes easy to create report XML file.

Features

- Supported output formats: Printer, PDF
- Universal type of data source
- SQL data source
- Visual modeling of SQL query
- Report elements: Label field, Images, Diagram
- Parameters from application side
- Several reports together
- Page header/footer
- Report page/header
- Data band
- Data grouping
- Group header/footer
- Aggregate functions: AVG, SUM, COUNT
- Mathematic functions
- Highlighting of the fields by login conditions
- Show/hide fields by logic conditions
- System variables
- Images: static or from application side
- Diagrams with manual data or with aggregate functions
- Figure and line drawing
- Barcode printing
- Rich text fields
- Pure Qt4/Qt5 code
- And much more...

Structure of the report's XML file.

The common structure of the report's XML file is very simple. There are five kind of nodes are used in XML files: <reports>, <report>, <ReportBand>, <TContainerField>, <DataSource> and <graph>.

```
<reports>
  <report>
    <ReportBand>
      <TContainerField>
      </TContainerField>
    </ReportBand>
    <ReportBand>
      <TContainerField>
      </TContainerField>
      <TContainerLine>
      </TContainerLine>
      <TContainerField type="diagram">
        <graph>
        </graph>
        <graph>
        </graph>
      </TContainerField>
    </ReportBand>
    <DataSource>
      <diagram>
      </diagram>
    </DataSource>
  </report>
</reports>
```

The root node is <reports>. The root node <reports> may contain several different report's node, so user can aggregate several reports into one XML file.

The <report> may contain up to 9 child nodes <ReportBand> . ReportBand may be one of following type:

- ReportTitle
- PageHeader
- DataGroupHeader
- MasterHeader
- MasterData
- MasterFooter
- DataGroupFooter
- ReportSummary
- PageFooter

Purpose and description of the bands by priority.

- The **ReportTitle** band is a first band which printing at the report. It's printing only one and just on the first page.
- The **PageHeader** band printing at the top on the each page of the report (but after ReportTitle band).
- The **DataGroupHeader** band is indented for group of data. Printed before MasterData band before each group of data.

- The **MasterHeader** band printing before MasterData band on each page of the report (if data present on the current page). Inside or not of each of data group (depending of appropriate parameter *ShowInGroup*).
- The **MasterData** is a main band which in the circle prints data received from your application.
- The **MasterFooter** band is a band which always follows after a MasterData band. Inside or not of each of data group (depending of appropriate parameter *ShowInGroup*).
- The **DataGroupFooter** band is indented for group of data. Printed after MasterData band after each group of data.
- The **ReportSummary** band is a conclusion of the report, its printing at the bottom of the last page of the report (but before PageFooter).
- The **PageFooter** band printing at the bottom on each page of the report.

Each ReportBand node may contain <TContainerField> node. The purpose of <TContainerField> is display any kind of information which necessary to user. The node <graph> holds the information about each graph of the diagram. This node only child of the <TContainerField> type="diagram"

To control of rendering process, each node has own set of attributes.

Note: All measurements such as page width, page height, etc. are made in points. To correct correlate the real size of paper and pixels, necessary to observe a ratio: 1 cm is 40 points

Note: All colors are encoded as following rgba(100,100,100,255). Where the first digit is red, second is green, third is blue, and last one is transparency. When you need to make some thinks invisible, please use the following values rgba(255,255,255,0). This means "White" color and full transparency.

The <report> also may contain node <DataSource>. This node contain info about connection to DS

Node type	Attribute	Permitted values (type)	Purpose	Remark
<reports>	programmer	string	Name of programmer	
	email	string	Email of programmer	
	lib	QtRPT	Name of lib	
<report>	orientation	0 - portrait; 1 - landscape	Orientation of the page	
	pageWidth	integer	Page width	
	pageHeight	integer	Page height	
	marginsLeft	integer	Left margins	
	marginsRight	integer	Right margins	
	marginsTop	integer	Top margins	
	marginsBottom	integer	Bottom margins	
	pageNo	integer	Page's number	
<ReportBand>	height	integer	Band's height	
	type	<ul style="list-style-type: none"> ReportTitle 	Report's type	

		<ul style="list-style-type: none"> • ReportSummary • PageHeader • PageFooter • MasterHeader • MasterData • MasterFooter, • DataGroupHeader • DataGroupFooter 		
	name	string	Report's name	
	groupingField	string	Field on which the group of data is carried out	Just for DataGroupHeader
	startNewNumeration	integer, 0 – is false; 1 – is true	Start or not new numeration for the group	Just for DataGroupHeader
	showInGroup	Integer, 0 – is false; 1 – is true	Show band inside of Data group	Just for MasterHeaderBand and MasterFooterBand
<TContainerField>	top	integer	Top position of the container	
	left	integer	Left position of the container	
	width	integer	Width of the container	
	height	integer	Height of the container	
	type	string <ul style="list-style-type: none"> • label • richText • labelImage • image • diagram • reactangle • roundedReactangle • circle • triangle • rhombus • barcode 	Type of the container	
	value	string with value To display in the current field	Visible value of the container	
	name	string	Container's name	
	fontBold	integer, 0 - is false; 1 - is true	Font's param	
	fontItalic	integer, 0 - is false; 1 - is true	font's param	
	fontUnderline	integer, 0 - is false; 1 - is true	font's param	
	fontStrikeout	integer, 0 - is false; 1 - is true	font's param	

	fontColor	string, encoded value, see Note above	font's color	
	fontFamily	The valid font family name	font's family	
	fontSize	integer	font's size	
	alignmentH	<ul style="list-style-type: none"> • hLeft • hRight • hCenter • hJustify 	Horizontal alignment	
	alignmentV	<ul style="list-style-type: none"> • vTop • vBottom • vCenter 	Vertical alignment	
	backgroundColor	string, encoded value, see Note above	Background color	
	borderColor	string, encoded value, see Note above	Border color	
	borderLeft	string, encoded value, see Note above		
	borderRight	string, encoded value, see Note above		
	borderTop	string, encoded value, see Note above		
	borderBottom	string, encoded value, see Note above		
	borderWidth	integer	Width of the border	
	borderStyle	string: <ul style="list-style-type: none"> • solid • dashed • dotted • dot-dash • dot-dot-dash 		
	picture	String, which must contains 84 bit base array.	Image information	For type Image only
	printing	integer, 0 - is false; 1 - is true	Printable(visible) or not field	
	highlighting	string:		
	format	string	Format of number value	
	showGrid	integer, 0 - is false; 1 - is true	Show or not grid	For type Diagram only
	showLegend	integer, 0 - is false; 1 - is true	Show or not legend	For type Diagram only

	showCaption	integer, 0 - is false; 1 - is true	Show or not diagram caption	For type Diagram only
	showGraphCaption	integer, 0 - is false; 1 - is true	Show or not Graph caption	For type Diagram only
	showPercent	integer, 0 - is false; 1 - is true	Show percent or real value	For type Diagram only
	caption	string	Diagram caption	For type Diagram only
	autoFillData	integer, 0 - is false; 1 - is true	Use manual data for graphs or use aggregate values	For type Diagram only
	autoHeight	integer, 0 - is false; 1 - is true	Stretch height of the fields and band to fit all text	Applicable only for fields which placed on MasterDataBand
	imgFormat	string	Holds the extension of image format	For type Image only
	ignoreAspectRatio	integer, 0 - is false; 1 - is true	Ignore or not aspect ratio for image	For type Image only
	barcodeType	integer	Type of the barcode	For type Barcode only
	barcodeFrameType	Integer: 0 – no border 1- bind 2- box	Type of the frame of barcode	For type Barcode only
<TContainerLine>	top	integer	Top position of the container	Always -10
	left	integer	Left position of the container	Always -10
	width	integer	Width of the container	Always -20
	height	integer	Height of the container	Always -20
	type	string <ul style="list-style-type: none"> Line 		
	name	string	Container's name	
	borderColor	string, encoded value, see Note above	Line color	
	borderStyle	string: <ul style="list-style-type: none"> solid dashed dotted dot-dash dot-dot-dash 	Line style	

	borderWidth	integer	Width of the line	
	printing	integer, 0 - is false; 1 - is true	Printable(visible) or not field	
	lineStartX	integer	X of line start	
	lineStartY	integer	Y of line start	
	lineEndX	integer	X of line end	
	lineEndY	integer	Y of line end	
	arrowStart	integer, 0 - is false; 1 - is true	Draw arrow at the start	
	arrowEnd	integer, 0 - is false; 1 - is true	Draw arrow at the end	
<graph>	caption	string	Graph caption	
	value	string	Holds the name of field or aggregate functions	For automatic calculation
	color	string, encoded value, see Note above	Color of the graph	For automatic calculation
<DataSource>		string	Node holds the sql query	
	type	string: SQL	Type of DS	
	name	string	Name of Connection	
	dbType	string	Name of the SQL driver	
	dbName	string	Name of DB	
	dbHost	string	Host address	
	dbUser	String	User name	
	dbPassword	String	Password	
	dbCoding	String	Coding of DB connection	
	charsetCoding	String	Coding of query charset	
<diagram>			Holds Diagram of SQL Designer	

How to use it.

First of all, you must add to your project the QtRPT.pri file. As example add the following line to your project.pro file.

```
include(../QtRPT/QtRpt.pri)
```

Second, we create a QtRPT object. Let's assume that **buttonPrintClicked** is a slot which is connected with the **"Print"** button. So, all code of creating QtRPT object, loading report and showing preview will be as below

```
void MainWindow::buttonPrintClicked() {
    QString fileName = "mydocument.xml";
    QtRPT *report = new QtRPT(this);
    report->loadReport(fileName);
    report->recordCount << ui->tableWidget->rowCount();
    QObject::connect(report, SIGNAL(setValue(int&, QString&, QVariant&, int)),
this, SLOT(setValue(int&, QString&, QVariant&, int)));
    report->printExec();
}
```

Let's consider in more detail.

1. Create a QtRPT object

```
QtRPT *report = new QtRPT(this);
```

2. Load a report's XML file

```
report->loadReport(fileName);
```

3. Set up the record count.

How many times the Master Data band will display data of field fitted on it depends on record count. For example, if you want to show in Master Data band all data stored in QTableWidgetItem, so, recordCount in QtRPT must be equal to rowCount of QTableWidgetItem.

Attention!!! From the version 1.1.0, the recordCount is a list of integer. You must append to the list records count of each datasource (report pages).

```
//OLD VERSION
report->recordCount = ui->tableWidget->rowCount();
```

```
//NEW VERSION
report->recordCount << ui->table1->rowCount();
report->recordCount << ui->table2->rowCount();
```

4. Connect signal of QtRPT with the appropriate slots for pass data to report.

Pass text, integer, etc. data

```
QObject::connect(report, SIGNAL(setValue(int&, QString&, QVariant&, int)), this,
SLOT(setValue(int&, QString&, QVariant&, int)));
```

Pass image data


```
QObject::connect(report, SIGNAL(setValueImage(int&, QString&, QImage&, int)), this,
SLOT(setValueImage(int&, QString&, QImage&, int)));
```

5. Before run the report, you may insert a background image

```
report->setBackgroundImage(QPixmap("./qt_background_portrait.png"));
```

6. Print or Show preview of the report

You may open preview of the report in maximize or fitted mode; Or you can print report without preview;

By default is a fitted mode and open with preview dialog and with use default printer;

```
report->printExec();
```

To open preview in maximize mode, you may write as following

```
report->printExec(true, false);
```

To direct printing without preview dialog

```
report->printExec(true, true);
```

To set the printer for use you may use the command as bellow. If printer with the name is not valid, the default printer will be used.

```
report->printExec(true, false, "name_of_the_printer");
```

To direct printing to Pdf file you may use the following function

```
report->printPdf("file_name_with_path", true);
```

The first param is a file name with a path. The second param indicates open or not after printing a pdf file in the associated application.

7. Get data into report

We need defined a slot, which will a pass our data to the report during execution. We pass the data via appropriate reference variable.

- int &recNo is a current record in the Master Data band
- QString ¶mName is a param name which currently rendering
- QVariant ¶mValue - value of the current param returned to print engine

For example, if current paramName is "Goods" (defined in the report), we look at the QTableWidgetItem the row with the number recNo and return paramValue.

```

if (paramName == "Goods") {
    if (ui->tableWidget->item(recNo,0) == 0) return;
    paramValue = ui->tableWidget->item(recNo,0)->text();
}

```

Slot **setValue** from the example project.

```

void MainWindow::setValue(int &recNo, QString &paramName, QVariant &paramValue, int
reportPage) {
    if (paramName == "customer")
        paramValue = ui->edtCustomer->text();
    if (paramName == "date")
        paramValue = ui->ntp->date().toString();
    if (paramName == "NN")
        paramValue = recNo+1;
    if (paramName == "Goods") {
        if (ui->tableWidget->item(recNo,0) == 0) return;
        paramValue = ui->tableWidget->item(recNo,0)->text();
    }
    if (paramName == "Quantity") {
        if (ui->tableWidget->item(recNo,1) == 0) return;
        paramValue = ui->tableWidget->item(recNo,1)->text();
    }
    if (paramName == "Price") {
        if (ui->tableWidget->item(recNo,2) == 0) return;
        paramValue = ui->tableWidget->item(recNo,2)->text();
    }
    if (paramName == "Sum") {
        if (ui->tableWidget->item(recNo,3) == 0) return;
        paramValue = ui->tableWidget->item(recNo,3)->text();
    }
}

```

Slot **setValueImage** from the example project.

```

void MainWindow::setValueImage(int &recNo, QString &paramName, QImage &paramValue,
int reportPage) {
    if (paramName == "image") {
        QImage *image = new
QImage(QCoreApplication::applicationDirPath()+"/qt.png");
        paramValue = *image;
    }
}

```

Diagrams

There are two modes getting data for diagram: Manual and on the basis of aggregate functions. Manual mode it is when user in the application form each element and its property of diagram. The properties are *color*, *caption*, *value*. In the mode on the basis of aggregate function, the all properties are sets directly in the report.

8. Manual setting data for Diagram

To use the diagram in manual mode, we need defined a slow, which will pass diagram's data to the report during execution. We pass the data via appropriate reference variable.

Chart &chart is a object which holds a diagram.

Let look at the example 7. First of all, we check that chart's name is a *diagram1*. Next, we create a a struct **GraphParam**, which holds data for each Graph of the diagram. The GraphParam has the following fields:

type	name	description
QColor	color	Color of the graph
float	valuePercent	value as a percentage. Just for inner using
float	valueReal	Value pass to graph by application
QString	caption	Caption of the graph (legend)

Each GraphParam pass to the chart by function *chart.setData(param);*

In the XML file of report, if the Diagram's Graph caption parameter set to "Real value" it is enough. In this case the highest of the graph will be graph with the biggest value.

If the Diagram's Graph caption parameter set to "Percent value", the biggest value will correspondents to 100%. To have possibilities display value more than 100%, you must defined the value for 100%. You must to do it in the last pass data to the diagram. For example:

```
chart.setData(param,150);
```

The 100% of the will correspondents with a 150 of value.

```
void ExampleDlg7::setValueDiagram(Chart &chart) {
    if (chart.objectName() == "diagram1") {
        GraphParam param;

        param.color = colorFromString("rgba(255,255,0,255)");
        param.valueReal = 150;
        param.caption = "Graph 1";
        chart.setData(param);

        param.color = colorFromString("rgba(0,0,255,255)");
        param.valueReal = 70;
        param.caption = "Graph 2";
        chart.setData(param);

        param.color = colorFromString("rgba(255,0,0,255)");
        param.valueReal = 220;
        param.caption = "Graph 3";
        chart.setData(param);

        param.color = colorFromString("rgba(0,128,128,255)");
        param.valueReal = 30;
        param.caption = "Graph 4";
        chart.setData(param,150);
    }
}
```

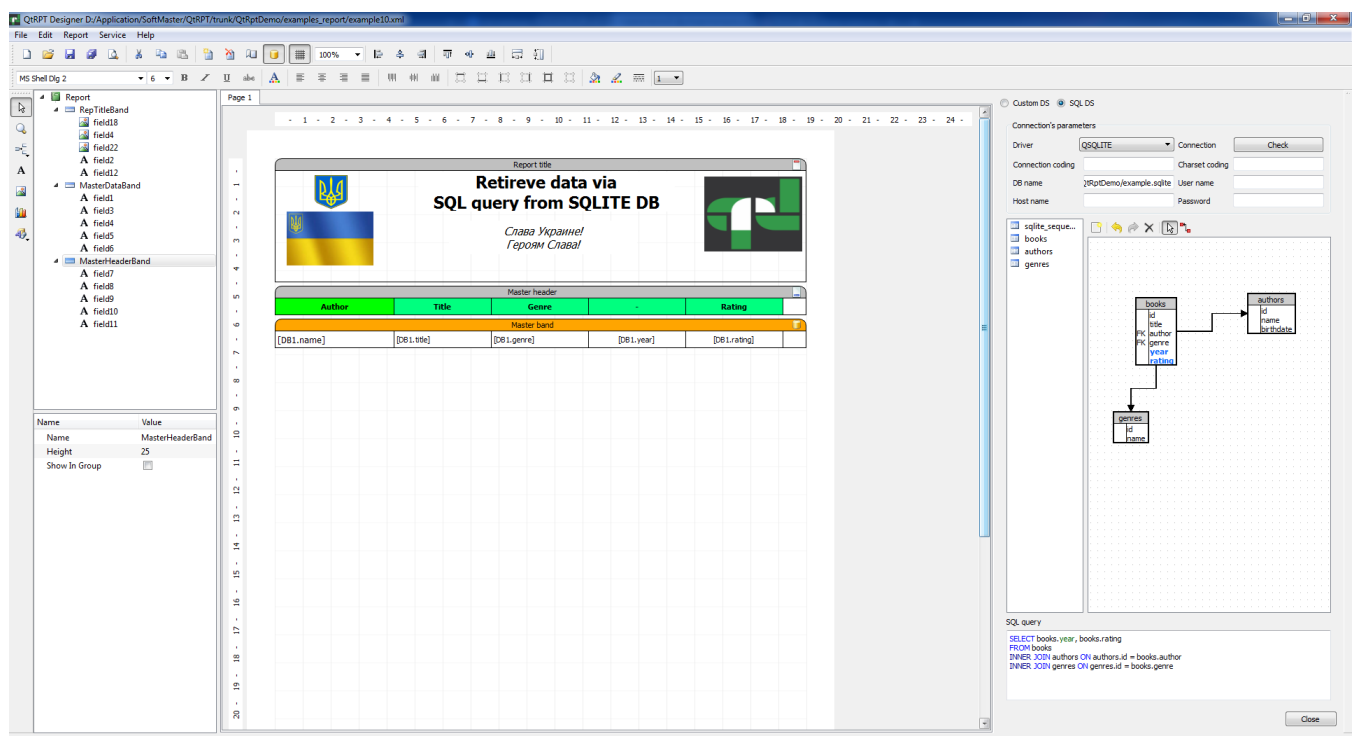
Reserved words

Sum, Avg, Count, Date, Time



QtRptDesigner

You may make xml file of the report in any text editor program. Or use any xml editor. Or use QtRPT Designer. How to use it, please check out our video file



How to use the field in the QtRptDesigner?

If you want enter some static text, you directly enter it. Into the field, you may enter the variable and some functions. The variable will read the data from your program. The variable has to be limited from both sides by '[' and ']'.

The QtRPT has some embedded functions such as:

<Date> - for printing current date

<Time> - for printing current time

<Page> - for printing number of current page

<TotalPages> - for printing Total Pages of report

<LineNo> - for printing Current record number of the Master Data band

Example: Let's assume you bring in a field the next line

Hello [FirstName], today is <Date> and now is <Time>. This prints on the <Page> page.

During the printing, the variable [FirstName] will be retrieved from the program, let's assume (Aleksey Osipov), so we will have the following:

Hello Aleksey Osipov, today is 18.06.2013 and now is 23:16:52. This print on the 1 page;

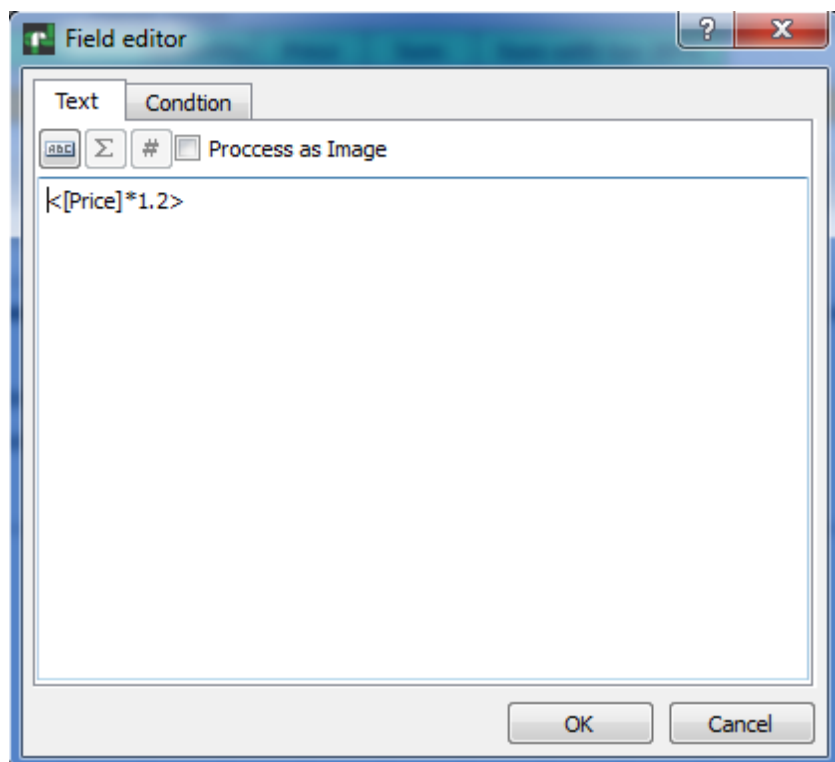
Processing of mathematical expressions

You may use in the fields mathematical expressions and some aggregate functions which can use to calculate sum, average and count such as

- Sum – sum the value of the field
- Avg – average value of the field
- Count – count the value of the field

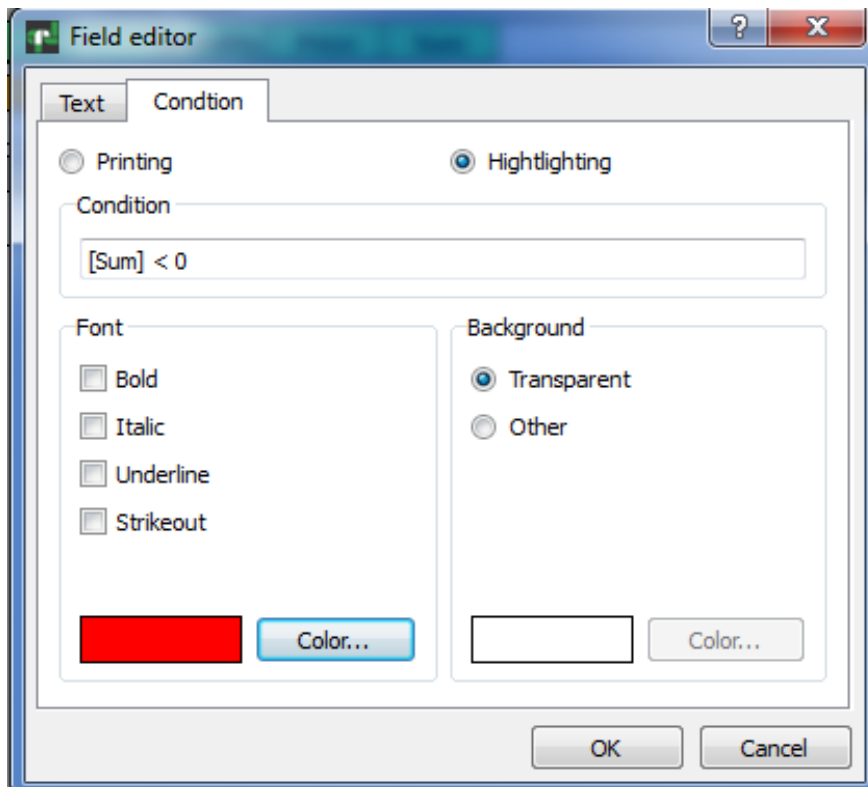
Example:

- Multiply the variable [Price] on 1.2 - <[Price]*1.2>
- Sum of the two aggregate functions - <sum([Quantity]) + sum([Price])>
- Multiply float on aggregate function - <sum([Sum]) * 1.2>
- Average price - <Avg([Price])>



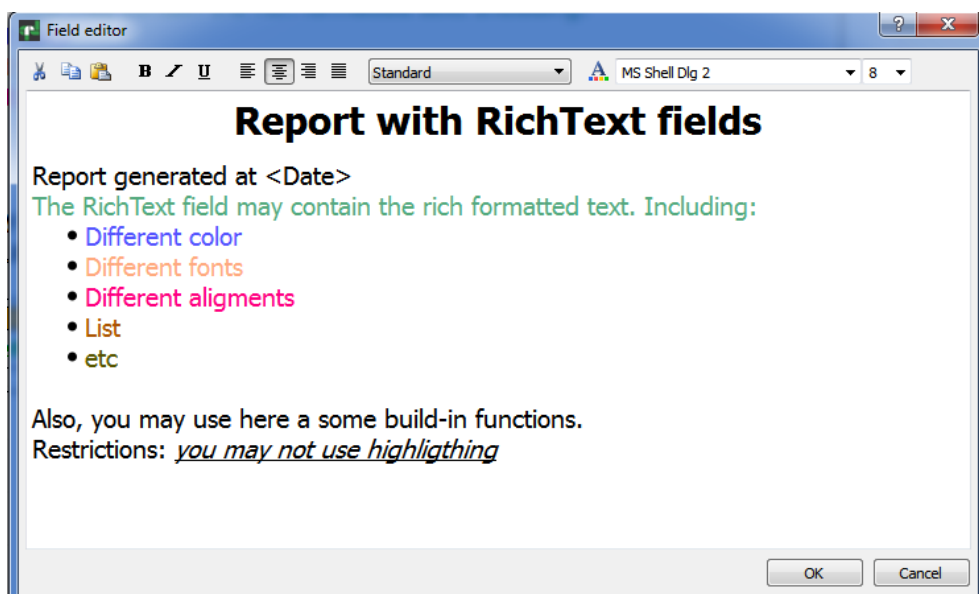
How to use Conditions in the fields in the QtRptDesigner?

Description in progress

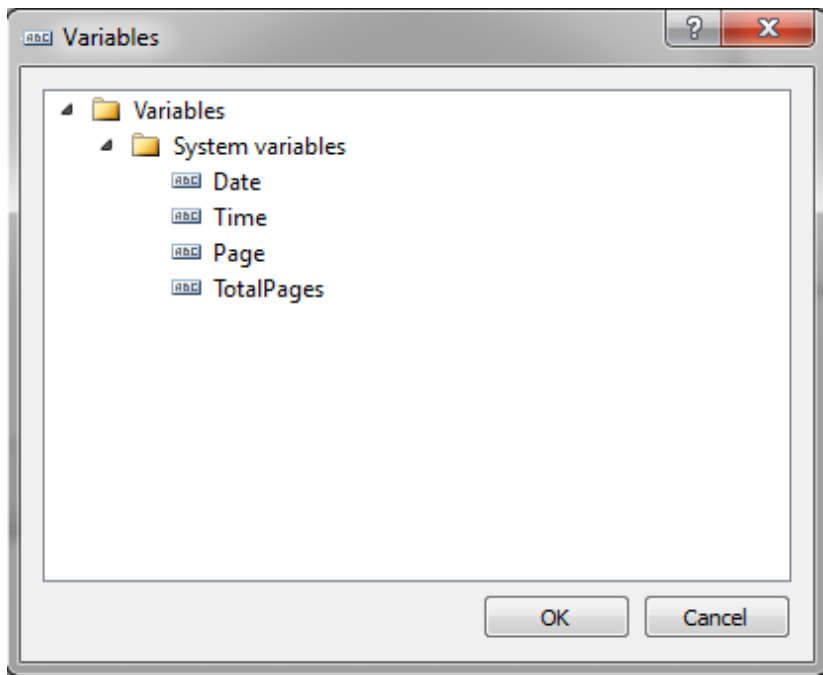


Rich Text field editor

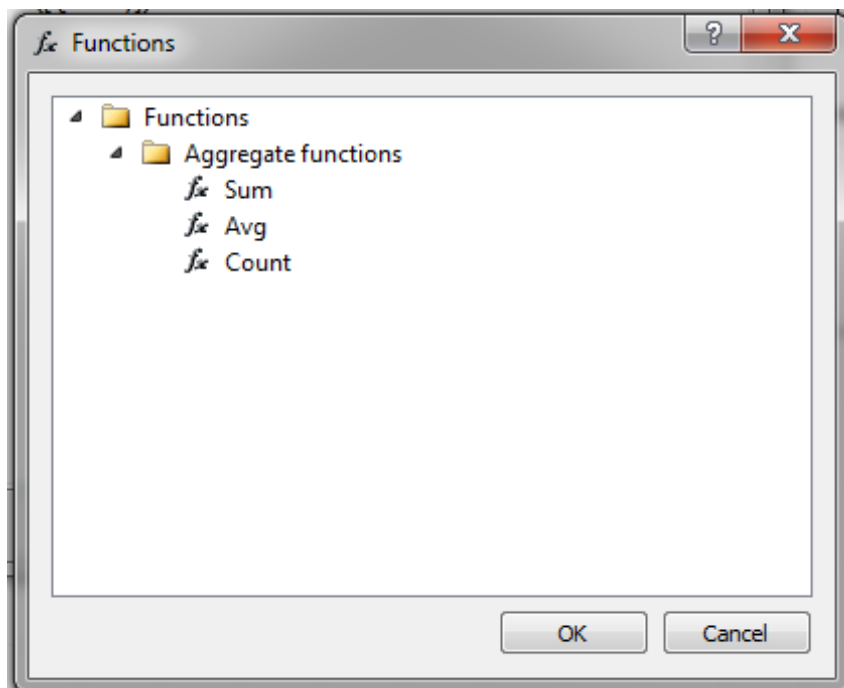
To edit the field "RichText", double click on the chosen container. The dialog box in which you can bring the text will open, change its property. You may change font's property such as Bold, Italic, Underline, change the color of the text. This field also supports system variables and work with external data. However this field doesn't support work with conditions, such as change of color registration depending on value of data.



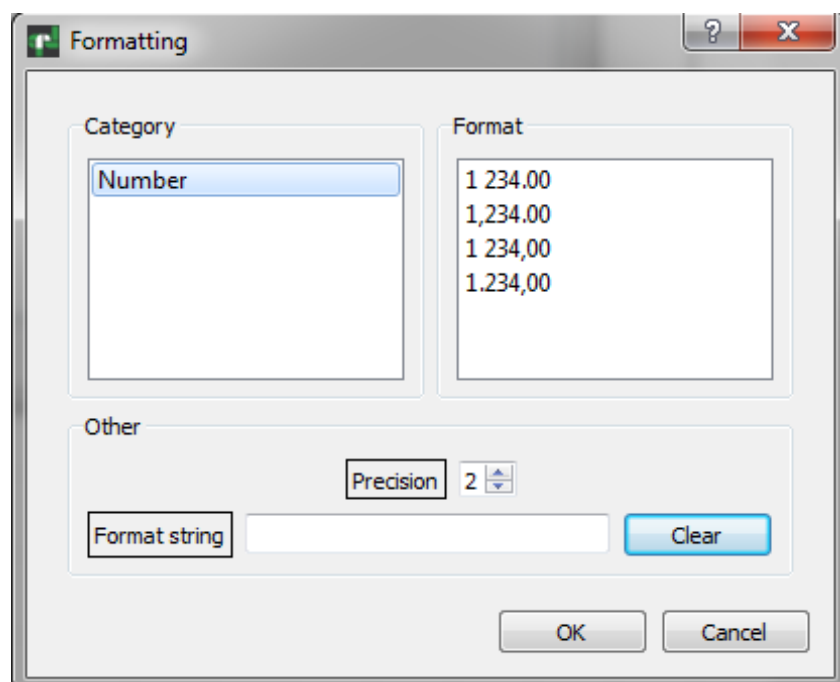
Adding variables



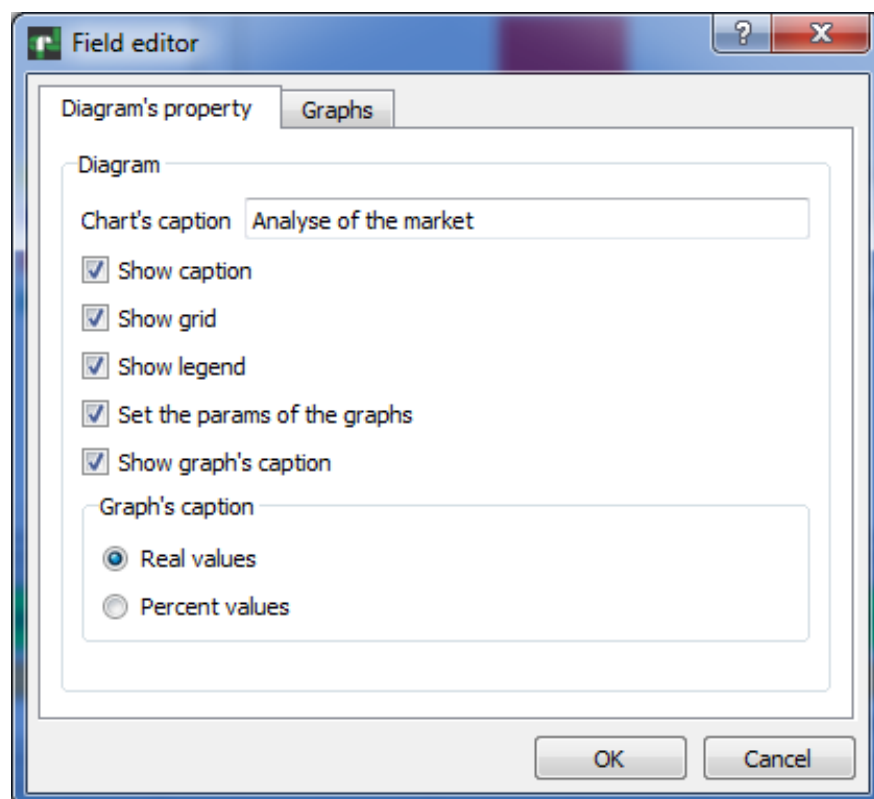
Adding functions



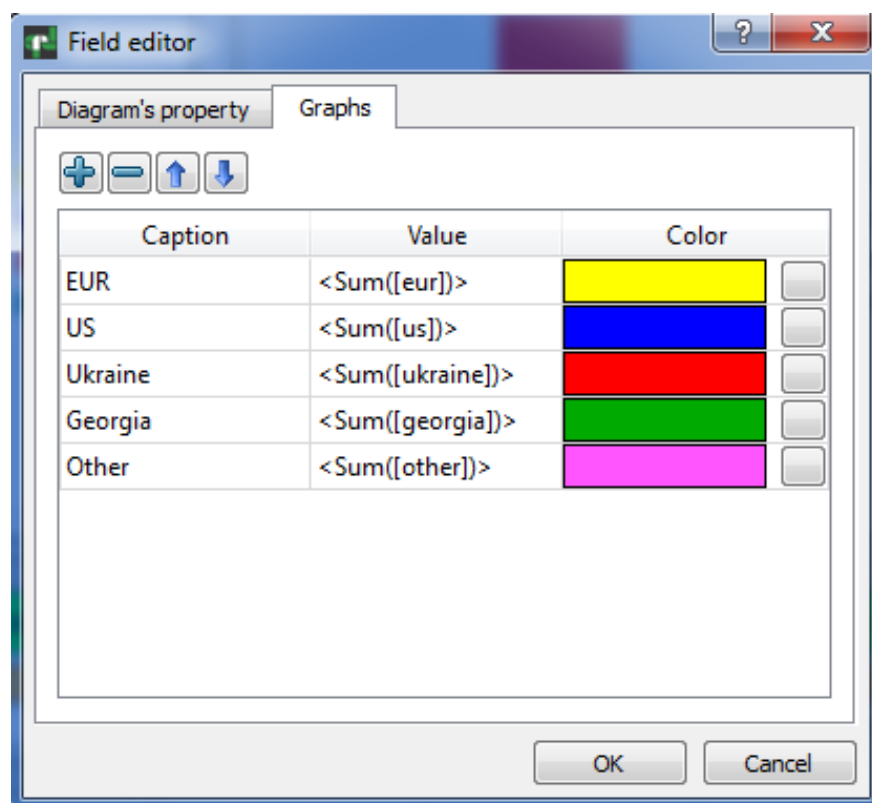
Adding formatting



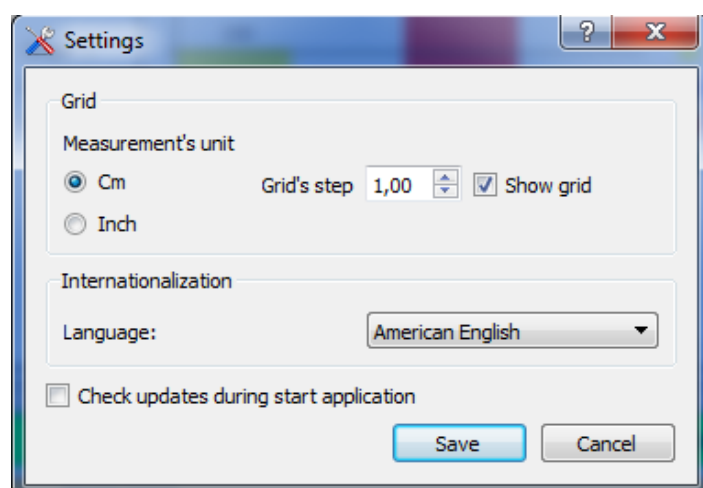
Diagram's parameters




Graph's parameters





Settings



Data group property



Data Group property



Data filed grouping

[Market]

☒ Start line numeration for each group

☐ Start new page for each group

OK

Cancel

Barcode property



Field editor





4820003351659

Barcode type

EANX

Frame type

NO_BORDER

Value:

482000335165

OK

Cancel

SQL Query Designer

The screenshot displays the SQL Query Designer application window. At the top, there are two radio buttons: "Custom DS" and "SQL DS", with "SQL DS" selected. Below this is a section titled "Connection's parameters" containing fields for "Driver" (set to "QSQLITE"), "Connection", "Check", "Connection coding", "Charset coding", "DB name" (set to "tRptDemo/example.sqlite"), "User name", "Host name", and "Password".

On the left side, there is a tree view showing a list of tables: "sqlite_seque...", "books", "authors", and "genres". The "books" table is currently selected.

The main area is a diagram showing the relationships between the tables. The "books" table is at the top left, with fields "id", "title", "author", "genre", "year", and "rating". The "authors" table is at the top right, with fields "id", "name", and "birthdate". The "genres" table is at the bottom left, with fields "id" and "name". A line connects the "author" field of the "books" table to the "id" field of the "authors" table. Another line connects the "genre" field of the "books" table to the "id" field of the "genres" table.

At the bottom, there is a text area labeled "SQL query" containing the following SQL statement:

```
SELECT books.year, books.rating
FROM books
INNER JOIN authors ON authors.id = books.author
INNER JOIN genres ON genres.id = books.genre
```

A "Close" button is located at the bottom right of the window.

SQL Query Designer allows connects to DB. After connection you may drag and drop table from table's tree to the diagram. You may establish a relationship between some tables. Setup a properties of the relationship. During visual modeling the SQL query is automatically generated. At any time you may correct the SQL query.

Thanks to

- Lukas Lalinsky for DBmodel
- Muhammad Bashir Al-Noimi <mbnoimi@gmail.com> for Arabic translation
- Luis Brochado <luisbrochado@softpack.pt> for Portuguese translation
- Li Wei <<ysu533@163.com>> - for Chinese translation



Donation

To develop and support the project your donations are welcome. To make donation please contact with me by email: aliks-os@ukr.net



Russian invaders

Because of military aggression of Russia in Ukraine, the QtRPT project doesn't give any consulting help to the users living in Russia.

And also we doesn't accept any councils and recommendations from ***Russian users***.

Any cooperation with the Russian users will be resumed only after a full conclusion of the troops from Ukraine and Crimea liberation.

QtRPT team

History

24.06.2012 QtRPT

- Published version 1.0.0

24.02.2013 QtRPT Version 1.0.1

- Added image support

14.04.2013 QtRPT Version 1.0.2

- QtRPT converted to Qt5

05.05.2013 QtRPT Version 1.0.3

- Add Border width and Border style support

18.06.2013 QtRptDesigner Version 1.0.2.1

- Multilanguage support. All who want translate to native language – are welcome

18.06.2013 QtRPT Version 1.0.4

- Added system variables to fields: Page number, Date, Time

04.07.2013 QtRPT Version 1.0.4.1

- Added system variable to fields: TotalPages

30.07.2013 Version 1.0.5

- Added additional band: MasterFooter.
- Added possibilities at run-time add images

20.08.2013

QtRptDesigner Version 1.1.0

- Added Multipage support.
- Some bug fixed
- Changed the priority of the bands
- Added additional band: MasterHeader.

QtRPT Version 1.1.0

- Added Multi page/datasource support
- Added three examples:
 - Invoice example (with background image)
 - Two pages(datasources)
 - Embedded report
- Some bug fixed
- Background image support during run-time

- Changed the priority of the bands
- Added possibilities at run-time add
- Added additional band: MasterHeader.

31.08.2013

QtRptDesigner version 1.2.0

- Added possibilities select several fields. Press Shift/Ctrl and click on the next field
- You can change of the size and movement of several fields
- You can alignment some selected fields on edge

18.09.2013

QtRPT version 1.3.0

- Added new property to TContainerField – “printing”. Which allow control visibility/printing of the field.
- Added new property to TContainerField – “highlighting”. Which holds information for controlling Font’s property such as Bold, Italic, Underline, Color and Color of the background depends of the some conditions.
- Added function to fields: LineNo

QtRptDesigner version 1.3.0

- At the dialog of the field’s property added page for editing of the Condition of the Visibility/Highlighting of the field.
- History last opened files.

05.02.2014

QtRPT version 1.3.1

- Mathematic functions
- Aggregate functions Sum, Avg, Count

QtRptDesigner version 1.3.1

- Bug fixed
- Copy/paste several fields
- QtRptDesigner moved to Qt5

15.03.2014

QtRPT version 1.3.2

- Possible select how to open preview window: fitted or maximize mode
- Bug fixed

QtRptDesigner version 1.3.2

- Bug fixed
- Possible setup a step of the grid

13.04.2014

QtRPT version 1.3.3

- Bug fixed

QtRptDesigner version 1.3.3

- Bug fixed
- Added new page size – Letter
- Added Inch measurement
- Added Ukraine language

11.05.2014

Our project now have own logo

QtRPT version 1.3.4

- Bug fixed (print selected pages)
- Bug fixed (set custom paper size)
- Direct printing without preview dialog

QtRptDesigner version 1.3.4

- Added Zooming

14.06.2014

QtRPT version 1.3.5

- Bug fixed (two and more reports with different page orientation)
- Print to pdf file
- Added additional band: **DataGroupHeaderBand**.
- Added additional band: **DataGroupFooterBand**.
- Added new property to **ReportBand** – “groupingField”. Field on which the group of data is carried out. Just for DataGroupHeader
- Added new property to **ReportBand** – “startNewNumeration”. Start or not new numeration for the group. Just for DataGroupHeader
- Added new property to **ReportBand** – “showInGroup”. Show band inside of each Data group. Just for MasterHeaderBand and MasterFooterBand
- Added two more examples

QtRptDesigner version 1.3.5

- Added font property: Strikeout
- Bug fixed (saving band’s height)

10.07.2014

QtRPT version 1.4.0

- Added new property to **ReportBand** – “startNewPage”. Start or not new page for each Data group. Just for DataGroupHeader
- Added new type of the TContainerField – diagram. The container type has the following properties: **showGrid, showLegend, showCaption, showGraphCaption, showPercent, caption, autoFillData.**
- Added one example with a diagram (manual and auto control)
- Added new node to the XML structure - <graph>. It is a child of the TContainerField
- Added additional files to project.
- Added new property to TContainerField – “format”. Which holds the formatting string of the numeric values.

QtRptDesigner version 1.4.0

- Added function for automatic/manual checking and downloading updates. In program setting possible switch off automatic checking of updates. Possible use manual checking/downloading of updates. This function works only for customers.
- Added possibilities add a diagram to the report.
- Added Dialog of adding functions
- Added Dialog of adding formatting
- Added Arabic language

06.08.2014

QtRPT version 1.4.1

- Bug fixed
- Added new property to TContainerField – “autoHeight”
- Reorganized the folders structure. Now we have additional folder QtRptDemo for demo project
- Changed how to include the QtRPT to your project. See the **Chapter How to use it**
- Added new five types of the TContainerField: **rectangle, roundedRectangle, circle, triangle, rhombus**
- Added three more examples
- Added new node to the XML structure - <DataSource>. It is a child of the <report>

QtRptDesigner version 1.4.1

- Some bug fixed
- Selection field bug fixed
- Select catalog during update bug fixed
- Font's size after changing zoom bug fixed
- Field's position and size after changing zoom bug fixed
- The BackgroundColor, FontColor, BorderColor now possible edit from Tree of params
- Added preview of the report
- Added possibilities of drawing some figures.
- Dialog for editing data source. Possible editing of the SQL query

23.09.2014

QtRPT version 1.4.2

- Bug fixed with using Field's name containing words Data, Time...
- Bug fixed to prevent change LineNo from user application
- Bug fixed with Landscape orientation
- Added new node to the XML structure - <diagram>. This node holds information about visual objects of the SQL query diagram
- Added new node to the XML structure - <TContainerLine>. This node holds information about Line
- Line drawing is available now
- Use predefined printer for use.

QtRptDesigner version 1.4.2

- Change parameters of multi-selected fields
- Bug fixed. Clear last empty lines in the field
- SQL editor now have highlighter
- Restart application after language changed
- Visual modeling of SQL query
- Added Portuguese language
- Added new container for Line drawing <TContainerLine>
- Line drawing is available now

20.10.2014

QtRPT version 1.4.3

- Added new property to TContainerField – “imgFormat”
- Now possible use any of supported image's formats
- Added new property to TContainerField – “ignoreAspectRatio”
- Now possible keep or ignore aspect ratio of image
- Added one example “Barcode generator”
- Bug fixing with grouping

QtRptDesigner version 1.4.3

- Change horizontal rule direction for RTL languages
- Undo/Redo for container's moving/resizing
- Added new container type: Barcode

11.11.2014

QtRPT version 1.4.4

- The container “TextImage” now draw keeping aspect ratio and take into account Vertical and Horizontal Alignment. Thanks to Mauro Anjo
- Added new type of TContainerField – “**richText**”
- Added one example “Report with RichText Filed”

QtRptDesigner version 1.4.4

- SQLDesigner bug fixing
- Added translation to Chinese
- Bug fixing with border color
- Bug fixing with border width
- The BorderWidth now possible edit from Tree of params
- New items in the popup menu for containers: “Move to front” and “Move to back”
- Undo/Redo for container’s adding/deleting
- Undo/Redo for container’s property changing
- Rich text editor

To Do

- QtRptDesigner. Add background image at design-time.
- QtRptDesigner. Offer change font params in the field’s editor, still in plan.
- Field’s auto width