QtRPT

Version 1.5.5
Programmer: Aleksey Osipov
Web-site: http://www.aliks-os.tk
Email: aliks-os@ukr.net
Web-site: http://www.qtrpt.tk
Address in Facebook https://www.facebook.com/qtrpt

QtRPT is the easy-to-use print 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. Thanks to Qt library, our project can be used in programs for work in the operating systems Windows, Linux, MacOS

Features

- Supported output formats: Printer, PDF, HTML
- 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
- Full control of report building from user application
- Pure Qt4/Qt5 code
- And much more…

License

For a long time the QtRPT project is distributed under the LGPL license. This license allows you to dynamically link with your source code. In order to facilitate the user to use QtRPT and allow to produce statically linking, I decided to change the Apache 2.0 license.

License Apache 2.0 more permissive compared to LGPL, I hope that the transition will not create problems for you

Please note that QtRPT to generate bar code uses the Zint library, which is distributed under license GPL, in this case, you must use your project is also under this license or disable the use of the Zint library.

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>
         <TContainer>
         </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	
	border	integer, 0 – is false; 1 – is true	Draw or not border on the page	
	borderWidth	integer	Border width	
	borderColor	string	Border color	
	borderStyle	string	Border style	
<ReportBand>	height	integer	Band's height	
	type	• ReportTitle • ReportSummary • PageHeader • PageFooter • MasterHeader • MasterData • MasterFooter, • DataGroupHeader • DataGroupFooter	Report's type	
	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

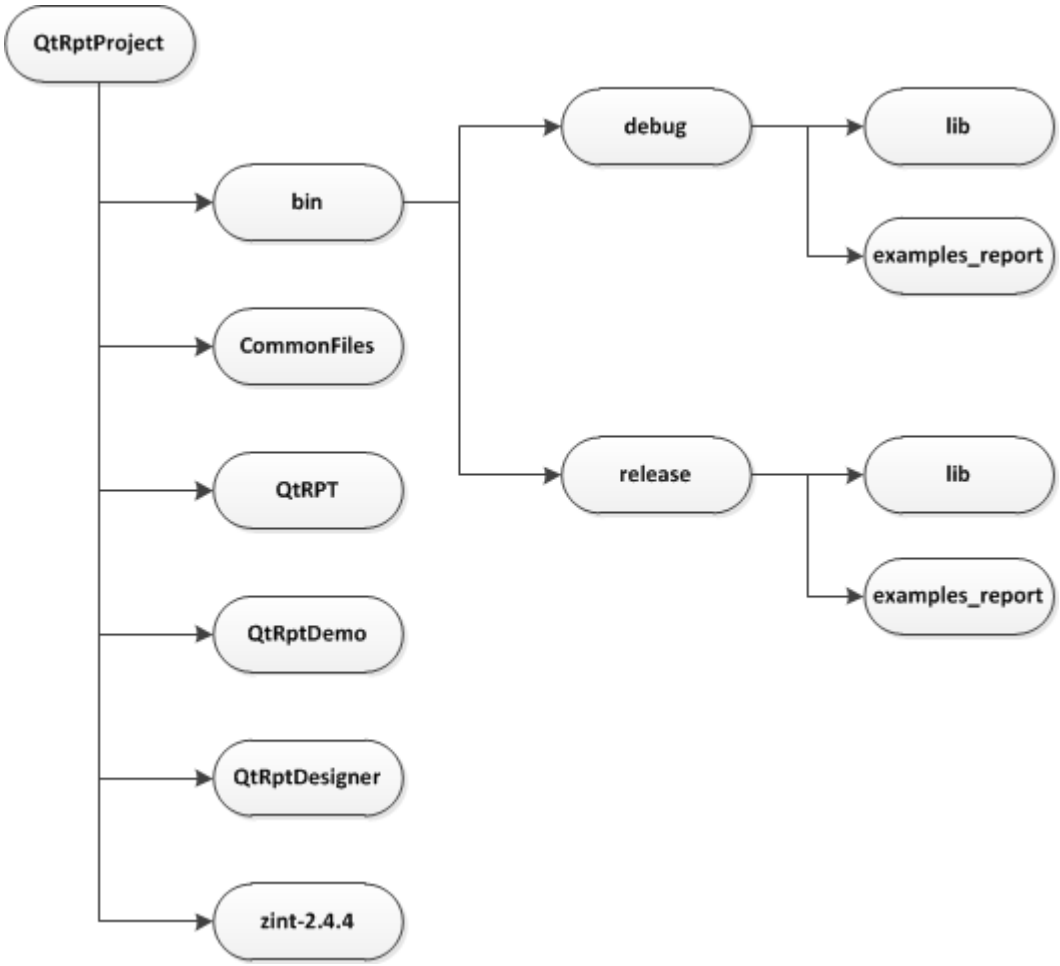
				MasterFooterBand
<TContainerField>	top	integer	Top position of the container	
	left	integer	Left position of the container	
	width	integer	Width of the container	
	height	integer	Height of the container	
	type	string • label • richText • labelImage • image • diagram • reactangle • roundedReactangle • circle • triangle • rhombus • barcode • databaseimage	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	
	aligmentH	• hLeft • hRight • hCenter • hJustify	Horizontal alignment	
	aligmentV	• 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: • 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
	barcodeHeight	integer	Height of Barcode	For type Barcode only
	textWrap	integer, 0 - is false; 1 - is true	Wrap or not text	For type Text only
	groupName	string	Group name of the field	
<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 • Line		
	name	string	Container's name	
	borderColor	string, encoded value, see Note above	Line color	
	borderStyle	string: • 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 DS	
	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	
	dbPort	integer	Number of port	
	dbConnectionName	String	Name of DB connection	
<diagram>			Holds Diagram of SQL Designer	

QtRPT project folder's structure



The QtRptProject's folder holds the following sub-directories:

- **bin**. Holds all files which are built after compilation, also this folder holds all files necessary for the correct operation of application. Respectively the **debug** and **release** versions of files are put in the corresponding folder. The folder "**lib**" holds library files after building of the library (zint). Folder "**examples_report**" holds the XML demo files.
- **CommonFiles**. Holds all common files used by all projects.
- **QtRPT**. Holds source files relating to QtRPT.
- **QtRptDemo**. Holds source files relating to Demo
- **QtRptDesigner**. Holds source files relating to QtRptDesigner
- **zint-2.4.4**. Holds source files relating to Zint library. It is a third party project

How to use it.

Please, note, that from version 1.4.5 all parts of the project (QtRptDemo, QtRptDesigner, third parties) united in one folder. To build simultaneously all parts of the project, you must use ***QtRptProject.pro*** file. In this case all binary files (QtRptDemo, QtRptDesigner, third party library (Zint)) will be built in ./bin/(debug or release) folder depends of building mode.

The folder of destination you may indicate in the file config.pri. To change it, open config.pri which located in QtRPT folder of the project. Locate the following lines

```
CONFIG(debug, debug|release) {
    DEST_DIRECTORY = $$PWD/bin/debug
}
CONFIG(release, debug|release) {
    DEST_DIRECTORY = $$PWD/bin/release
}
```

The variable DEST_DIRECORY specifies where there will be binary files after compilation. The path of linking to the Zint library also depends from this variable. If you have folders of your project on another, please make sure that at you ways are correctly specified everywhere.

Also you may build any part of the project separately, if you need it you may do it by opening QtRptDemo.pro or QtRptDesigner.pro or Zint.pro in appropriate folder.

Please note that from version 1.4.5 the using of barcode feature you need build Zint library. I.e. before building of you own project you must have the following files in /bin/(debug/release/lib)

- ***libQtZint.a***
- ***QtZint.dll***

If you don't need barcode feature at all, you may set it in QtRPT.pri file

DEFINES += NO_BARCODE

In this case, the building will carry out without Zint library features.

Building QtRPT as a library.

If you want, you may also build QtRPT as a library. For this, un-comment

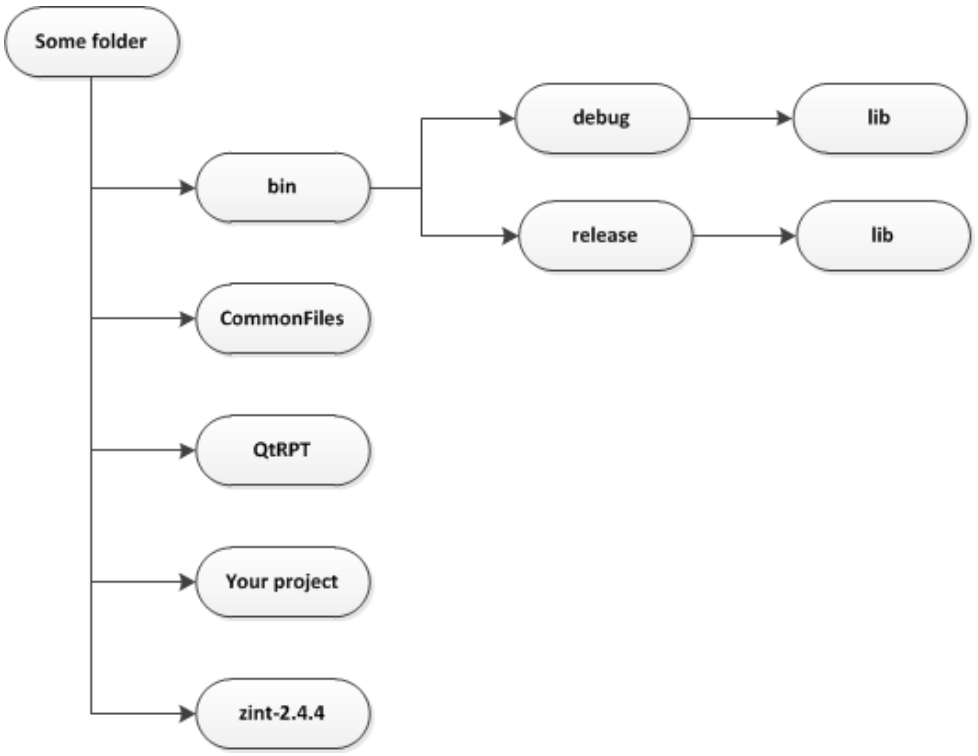
```
#DEFINES += QTRPT_LIBRARY
#contains(DEFINES,QTRPT_LIBRARY) {
#    SUBDIRS += QtRPT
#}
```

in the QtRptProject.pro file. Or include the QtRPT.pro in your project. Also in your project file you must add the following lines

```
contains(DEFINES,QTRPT_LIBRARY) {
    INCLUDEPATH += $$PWD/../QtRPT/
    LIBS += -L$${DEST_DIRECTORY}/lib -lQtRPT
}
```

Using QtRPT in your project.

For novice and easy-of-using of QtRPT please use the following folder's structure



The people how have experience in using of third-party libraries and changes of dependences between projects may use own folder's structure.

There are two ways of using QtRPT in you projects. First way is adding of all files to your project. The second way build QtRPT library and use it in your project

To use first way, you must add to your project the QtRPT.pri file. As in the example add the following line to your project.pro file.

include(../QtRPT/QtRPT.pri)

To build the QtRPT library, you must open QtRPT.pro file and build a library for using in your project.

Then, correct the folder of destination as described above (**config.pri** file).

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

```
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(const int, const QString, QVariant&,
const int)), this, SLOT(setValue(const int, const QString, QVariant&, const 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 QTableWidget, so, recordCount in QtRPT must be equal to rowCount of QTableWidget.

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 VERISION
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(const int, const QString, QVariant&, const int)), this, SLOT(setValue(const int, const QString, QVariant&, const int)));
```

Pass image data

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

- Const int recNo is a current record in the Master Data band
- Const QString paramName is a param name which currently rendering
- QVariant &paramValue - value of the current param returned to print engine
- Const int reportPage is a current reportPage (page of the report – not printing page)

For example, if current paramName is "Goods" (defined in the report), we look at the QTableWidget 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->dtp->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;
    }
}
```

8. Get full control of report building

Since version 1.4.5 there are possible take a full control of report building from user application. You may change any parameter of any object of the report. Moreover, you may build your report without XML file. Take a look at exampledlg14.cpp

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

//Make a page of report
RptPageObject *page = new RptPageObject();
page->pageNo=0;
report->pageList.append(page); //Append page to the report

//Make a ReportTitleBand
RptBandObject *band1 = new RptBandObject();
band1->name = "ReportTitle";
band1->height = 80;
band1->type = ReportTitle;
page->addBand(band1); //Append band to the page

//Make a field
RptFieldObject *field1 = new RptFieldObject();
field1->name = "field1";
field1->fieldType = Text;
field1->rect.setTop(0);
field1->rect.setLeft(0);
```

```
field1->rect.setHeight(60);
field1->rect.setWidth(700);
field1->font.setBold(true);
field1->font.setPointSize(24);
field1->aligment = Qt::AlignCenter;
field1->textWrap = 1;
field1->value = "Creation of the report from user application without XML file";
field1->borderLeft = Qt::white;
field1->borderRight = Qt::white;
field1->borderBottom = Qt::white;
field1->borderTop = Qt::white;
```

band1->addField(field1); //Append field to the ReportTitleBand

```
report->recordCount << 4;
QObject::connect(report, SIGNAL(setField(RptFieldObject &)), this,
SLOT(setField(RptFieldObject &)));
```

report->printExec();

Now we have a new SIGNAL setField(RptFieldObject &). This signal emits before signal ***setValue***, which were described above. The signal ***setField*** pass to user application a RptFieldObject which represents a field on the band. Changing the properties of this object, you can change any parameter of the field. To make so that the field looked as it is necessary for you.

From **RptFieldObject**, you may have access to the parent **RptBandObject** and from it to parent **RptPageObject**. For more information please take a look to the API of the class.

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.

9. 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 fileds:

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.

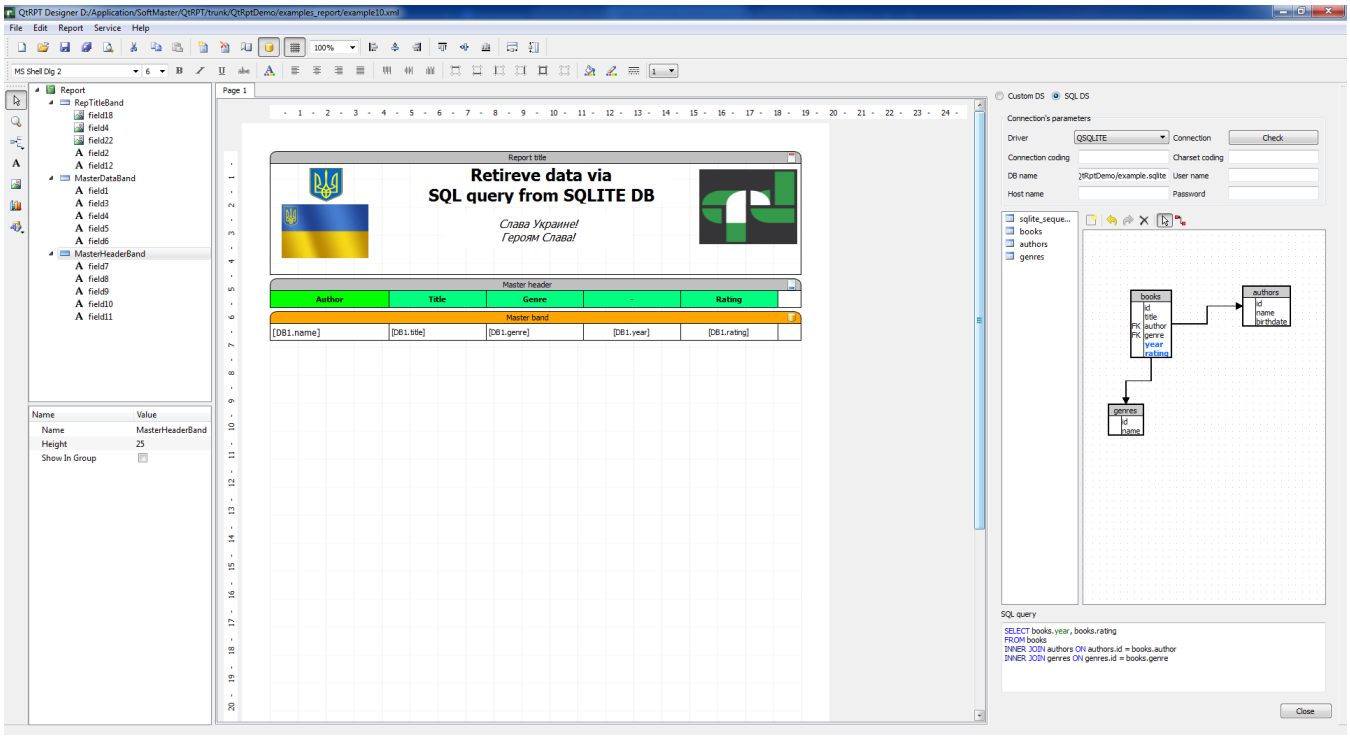
```
void ExampleDlg7::setValueDiagram(Chart &chart) {
    if (chart.objectName() == "diagram1") {
        GraphParam param;

        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



Bands in QtRPT

Bands are used for placing the objects they contain at particular locations on the output page. For example, when placing an object in the "PageHeader" band we tell the report engine that the given object must be displayed at the top of each page in the finished report. Similarly, objects in the "PageFooter" band are displayed at the bottom of each page.

After starting of the QtRptDesigner , you will see a empty report. Now you add a new band. Click the "Insert band" button on the object toolbar and select "appropriate band from drop-down list. We see a new band is added to the page. QtRptDesigner automatically positions bands on the page - according the bands priority.

Fields' navigation by keyboard

You may move and change size of the fields by mouse. But sometimes it is necessary to do it more accurately using the keyboard. Select field, hold down **Ctrl** key and press Up/Down/Left/Right key. In this case the field will moved to appropriate side. To change the size of the field, hold down **Shift** key and press Up/Down/Left/Right key.

To select multiple fields, select one field, then hold down **Shift** key, and then click on another field. You may simultaneously move and resize fields. You can also group the selected fields, which will serve as one of the field during the selection.

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;

Buil-in functions

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

Aggregate functions

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

- Max – max value of the field
- Min – min value of the field

Text functions

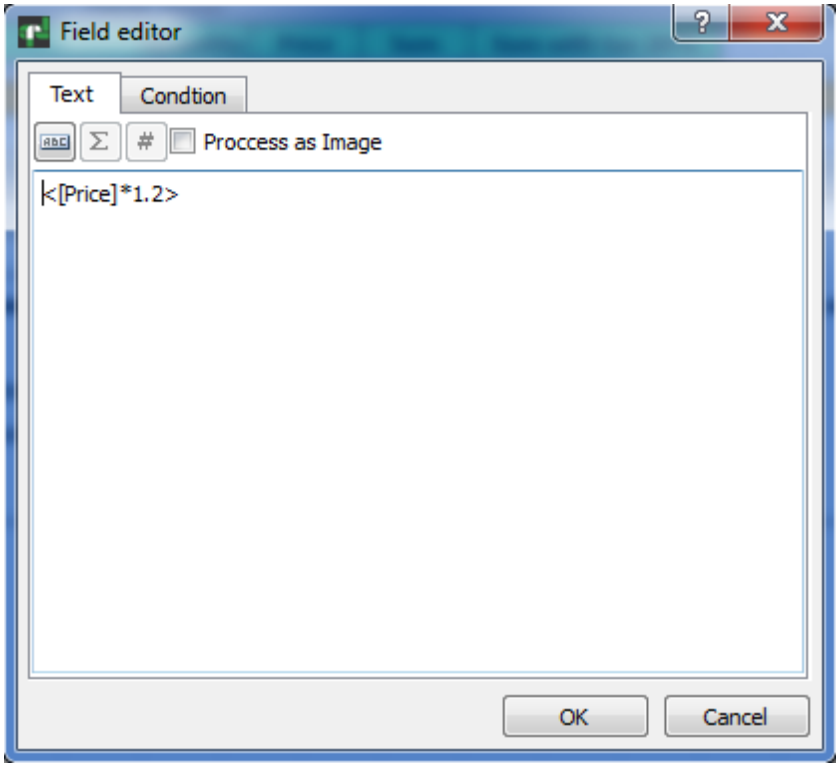
- NumberToWords (ENG) – transfer the number to words (English)

Math functions

- Frac - return the fractal part of number (double)

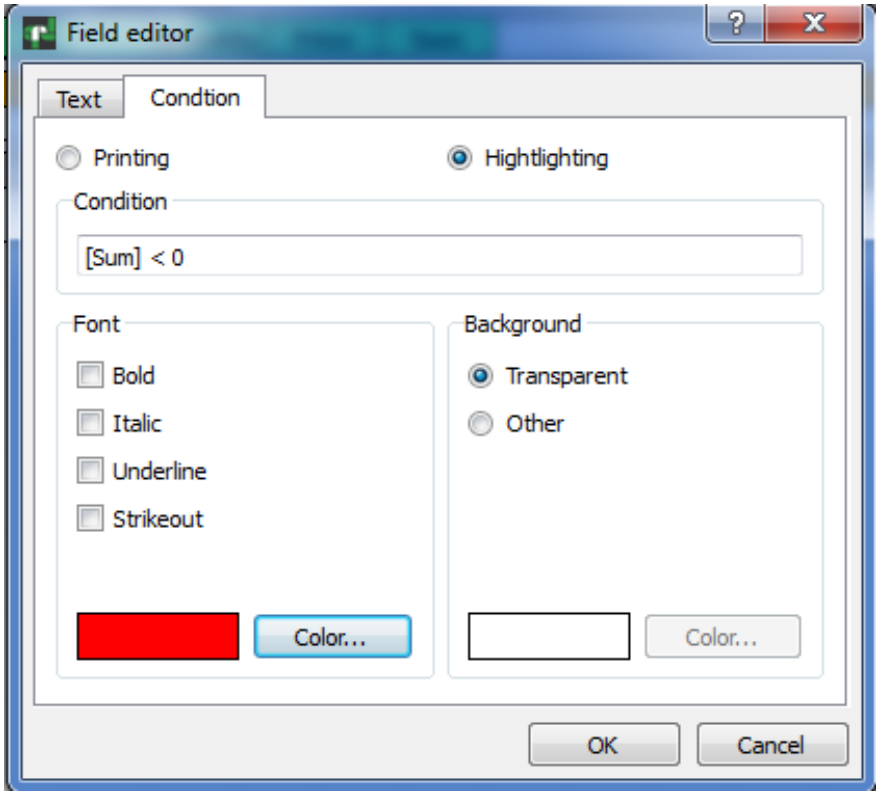
Example:

- 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])>
- Return Frac part of number, 45 - <Frac(3.45)>



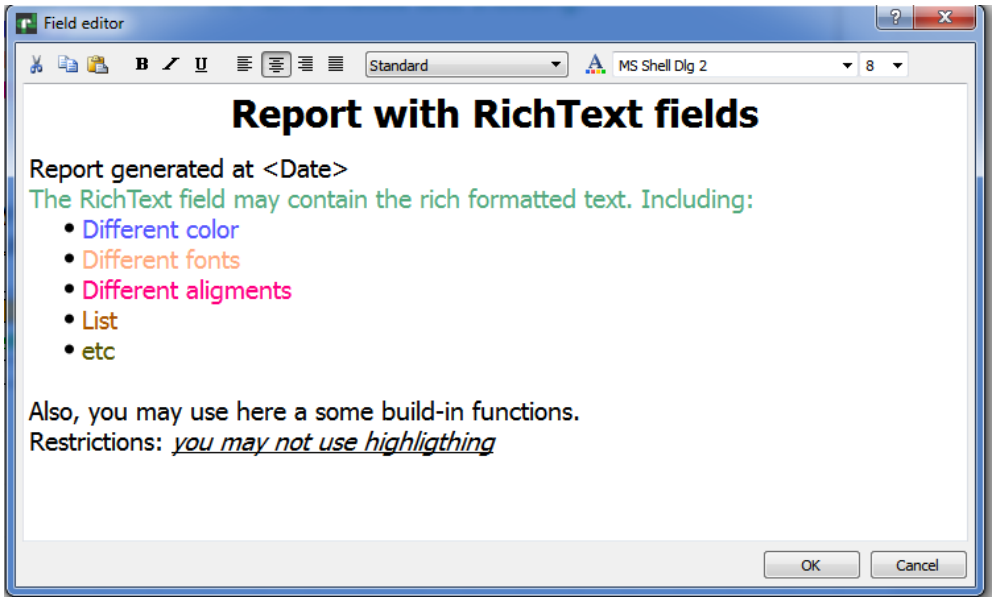
How to use Conditions in the fields in the QtRptDesigner?

Description in progress

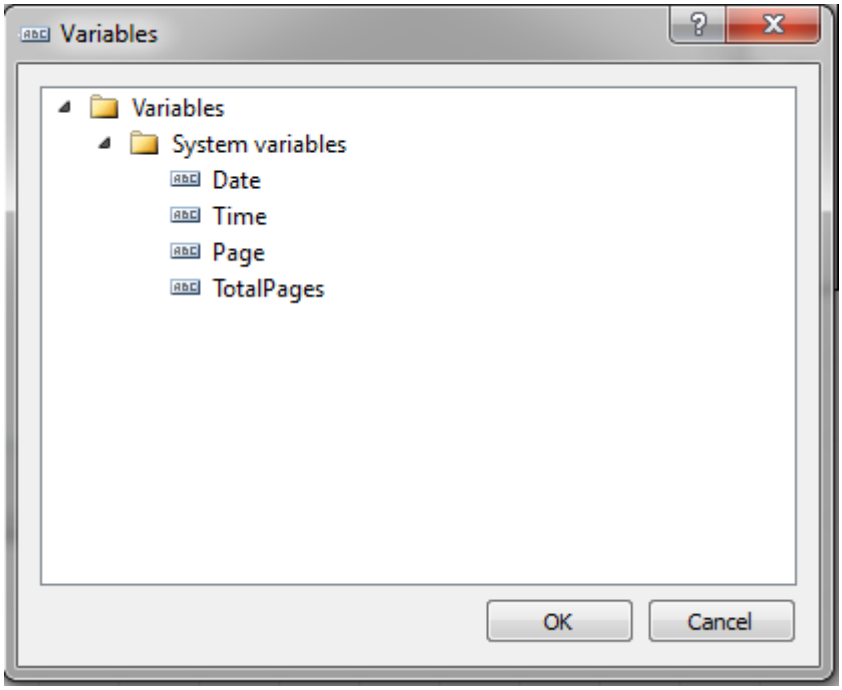


Rich Text field editor

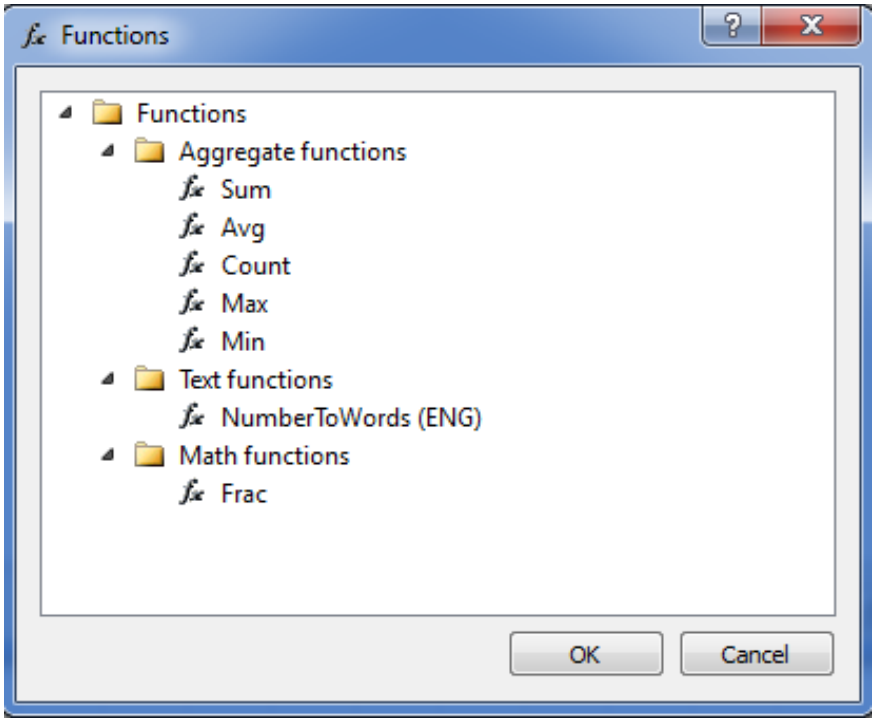
To edit the field "RichText", double click on the chosen container. The dialog box in which you can bring the text will open, change its property. You may change font's property such as Bold, Italic, Underline, change the color of the text. This field also supports system variables and work with external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



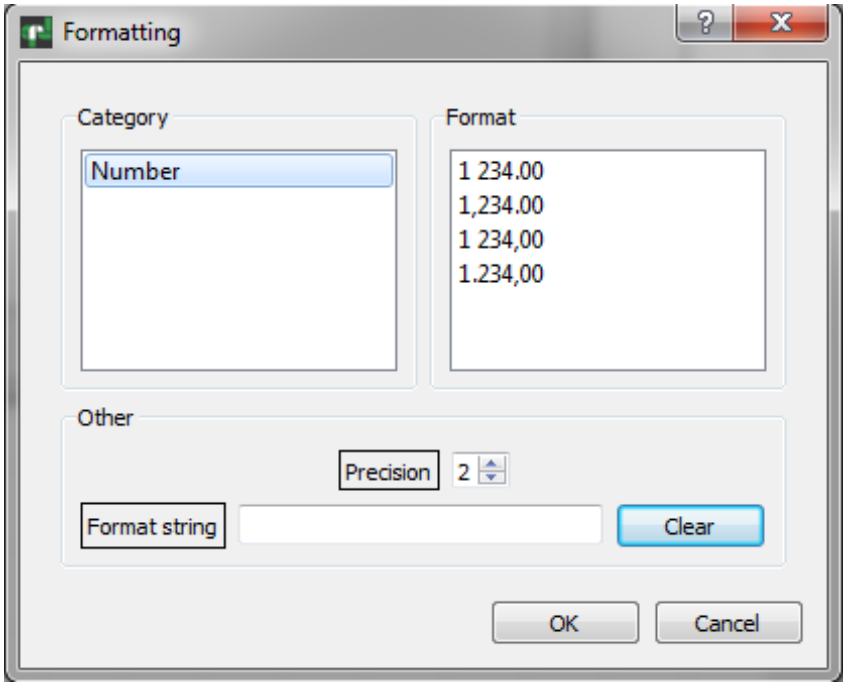
Adding variables



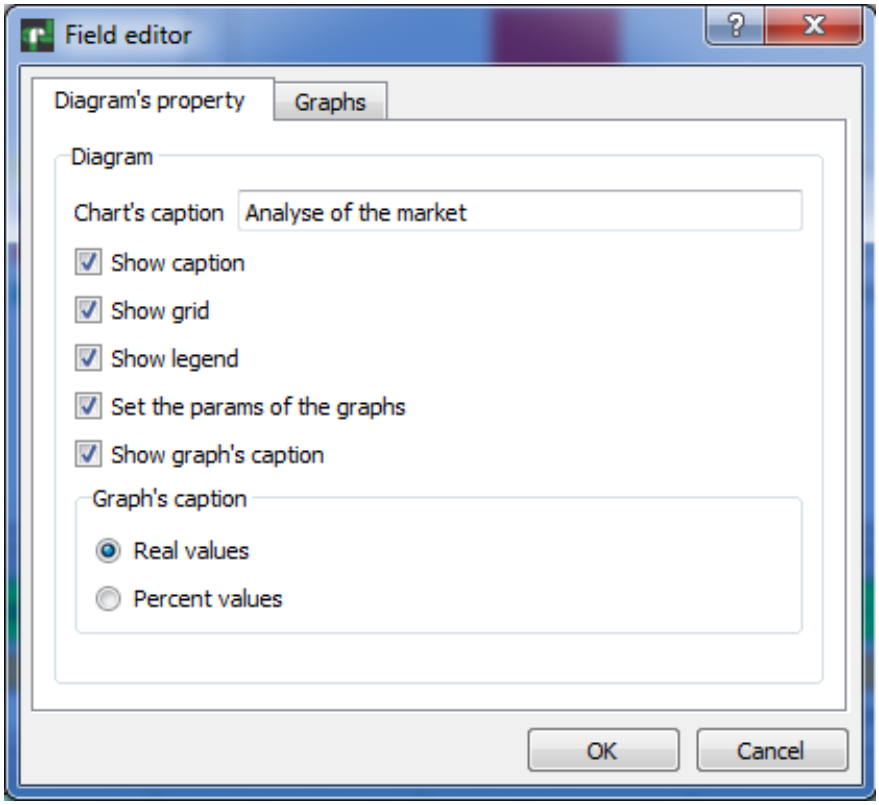
Adding functions



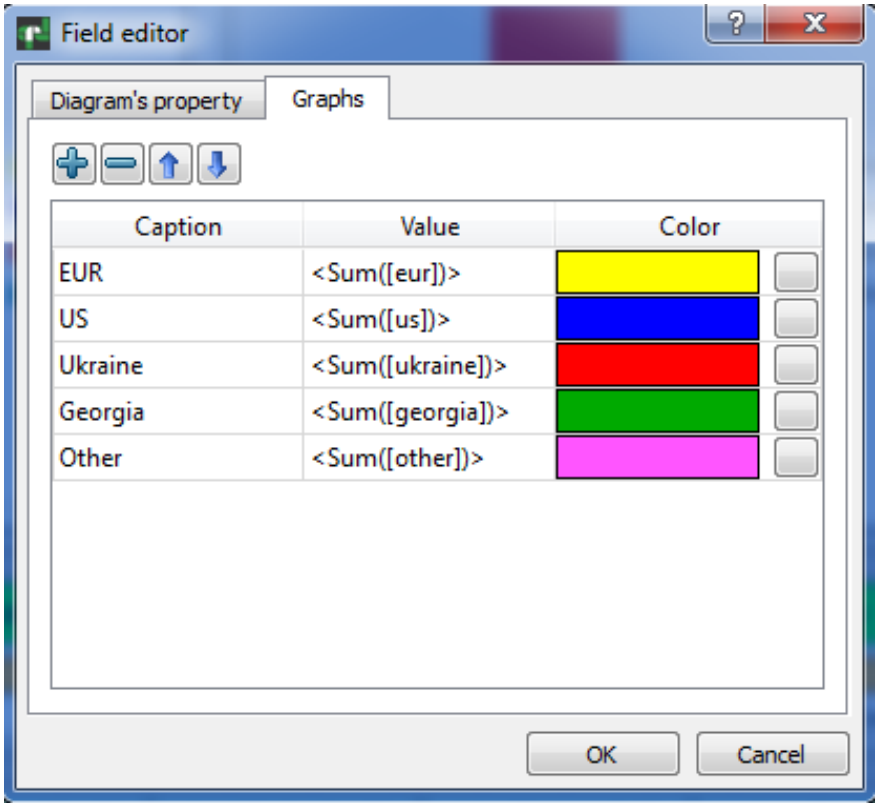
Adding formatting



Diagram’s parameters

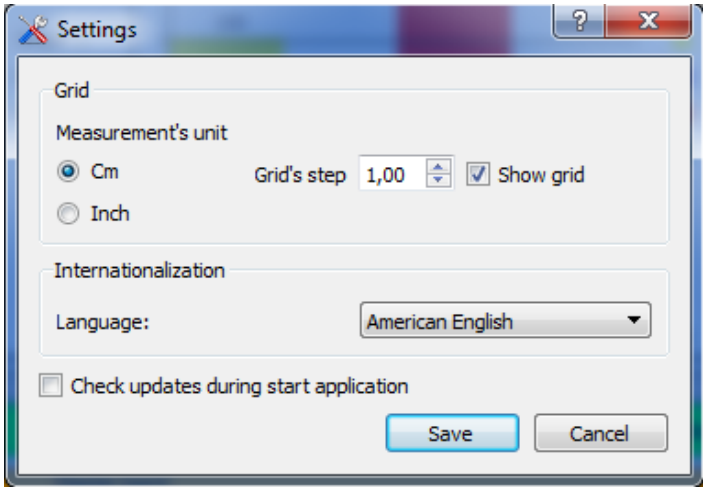


Graph’s parameters



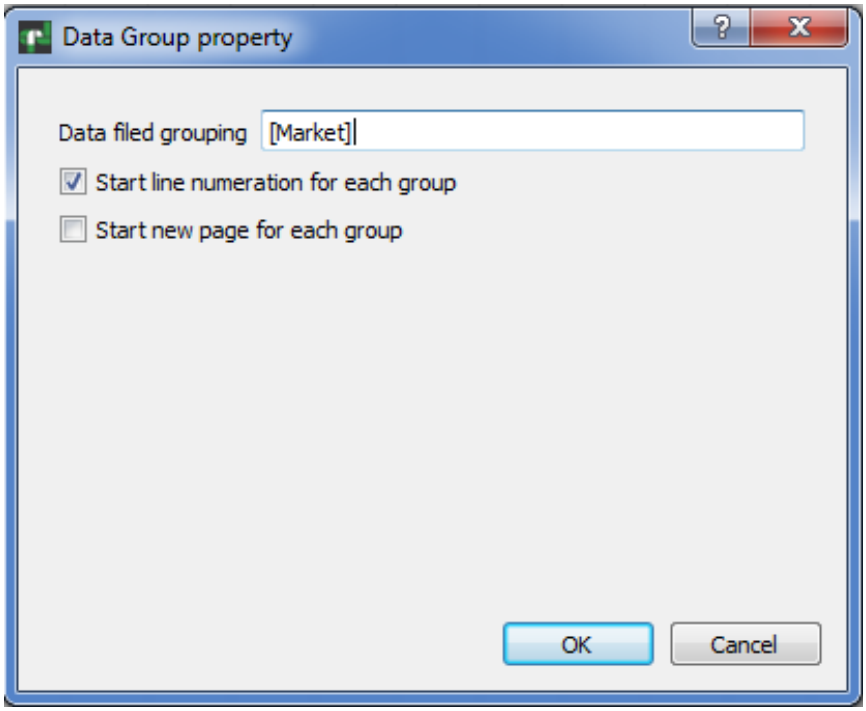
Settings

Set the designer options.



Here you can set the preferred units (centimeters, inches) and grid spacing and visibility. You can set the language of the QtRptdesigner. If you want that during the lanch QtRptDesigner checks updates from Internet, please check **“Check updates during start application”**.

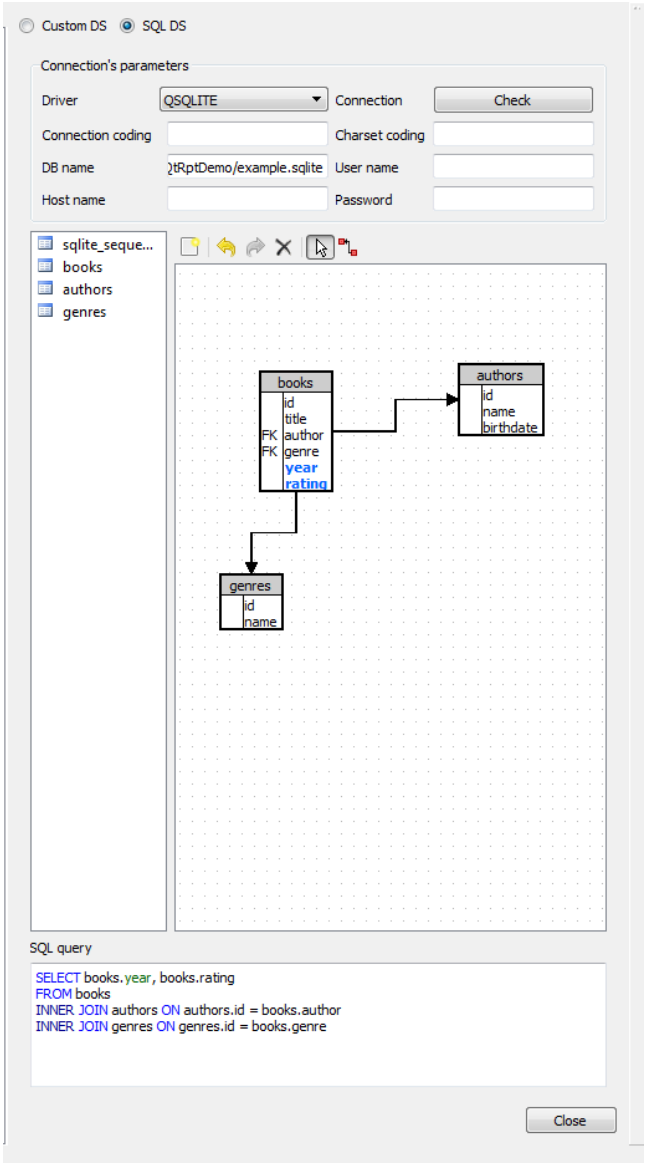
Data group property



Barcode property

Please note: that since version 1.4.5, to use barcode feature, you need to have built QtZint library. The source files are in folder Zint-2.4.4. After building, place (QtZint.dll, QtZint.so file) library into the folder where your application able to find it.

SQL Query Designer



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.

Donation

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

Thanks for donation

- Sailendram

Thanks for help in project developing

- Lukas Lalinsky for DBmodel
- Norbert Schlia for help in developing
- 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
- Laurent Guilbert <lauguidev@gmail.com> - for French translation
- David Heremans <dhran@scarlet.be> - for Dutch translation
- Mirko Marx <kqxtvqpr@gmail.com> - for German translation
- Manuel Soriano <manu@manu.ms> - for Spahish translation

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 don'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". This 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: **reactangle, roundedReactangle, 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 it the popup menu for containers: “Move to font” and “Move to back”
- Undo/Redo for container’s adding/deleting
- Undo/Redo for container’s property changing
- Rich text editor

16.12.2014

QtRPT version 1.4.5

- There is a possible pass text with HTML tags from user’s application to QtRPT report. (RichTextField)
- Changed example “Report with RichText Filed”
- Project’s Folders reorganization, please see file “Folder’s structure”
- Please note: that since version 1.4.5, to use barcode feature, you need to have built QtZint library. The source files are in folder Zint-2.4.4.
- Bug fixing “Data group header and footer overlaps with page footer”. Thanks to Puterk
- Added new property to **TContainerField (type TextField) –** “WrapText”
- Now possible change properties of any object of the report from user application during report building
- User now have full control of report building from own application.
- Added example 13 “**Report's control from user application**”
- Added example 14 “**Creation of the report from the user application without XML file**”
- **Very important**. Changed method of connection (using) QtRPT in your projects. Please take a look in chapters “**Hot to use it**” and “**Using QtRPT in your project.”**

QtRptDesigner version 1.4.5

- RTL direction of text in “TextField” and “RichTextField”
- Please note: that since version 1.4.5, to use barcode feature, you need to have built QtZint library. The source files are in folder Zint-2.4.4.
- Bug fixing of the multiple Cut/Paste
- Bug fixing of the crash when deleting just one page of the report
- After adding field to the band, editor’s property does not open. After adding you must double click to open.
- Speed up report loading

05.02.2015

QtRPT version 1.5.0

- Now we have own web-site at http://qtrpt.sourceforge.net/ or https://sourceforge.net/projects/qtrpt/
- Memory leak fixed
- Adoption to use in MacOS and Linux
- Adoption to use with Qt4x
- Now you may build QtRPT as a library
- Added system variable “LineCount”

QtRptDesigner version 1.5.0

- Added French language
- Prepared files for Spanish language
- Some bug fixed
- Adoption to use in MacOS and Linux
- Adoption to use with Qt4x
- ImageField. Now possible save image to the file.

08.04.2015

QtRPT version 1.5.1

- QtRPT now inherits QObject
- Added new example “Print to PDF”
- Added new example “QtRptCGI”, shows how to make CGI to generate reports under WebServer Apache2
- Added function to change SQL query “setSqlQuery”
- QtRPT now have possibilities generate HTML report
- Added button “Export to PDF” to PrintPreviewDialog
- Added button “Export toHTML” to PrintPreviewDialog
- Changed signals to:

1. `void setValue(const int recNo, const QString paramName, QVariant &paramValue, const int reportPage);`
2. `void setValueImage(const int recNo, const QString paramName, QImage &paramValue, const int reportPage);`

- Change license to Apache 2.0

QtRptDesigner version 1.5.1

- The page size Letter corrected to H-1118px W-863px
- Added page size A3, A5
- Possibility of fields grouping
- Speed up during containers moving
- Font's size bug fix after copy/paste
- Added Dutch language

12.05.2015

QtRPT version 1.5.2

- Drawing bug fixing
- Background highlighting bug fixing
- Bug fixing of using QtRPT as library
- Added following built-in functions:
  - NumberToWords (ENG)
  - Frac
  - Max
  - Min

QtRptDesigner version 1.5.2

- Located problem of DB drivers
- Zooming bug fixed
- Remove highlighting bug fixing

01.08.2015

QtRPT version 1.5.3

- Added public function "**setResolution**" to change resolution of printing
- Now possible QPrinter and QPainter from user application. Added public functions: "**setPrinter"** and "**setPainter"**
- Added signal newPage(int page) which emits when creating new page during printing
- Now possible draw border of the page
- Added new property to TContainerField – "barcodeHeight"

QtRptDesigner version 1.5.3

- Added German language
- Added Spanish language
- At the dialog "Page settings" possible sets the properties of border

14.09.2015

QtRPT version 1.5.4

- Added possibilities pass SQL params and SQL query from user application
- Added new field's type DatabaseImage
- Added possibility retrieve images from DB
- Added following built-in functions:
    - NumberToWords (GER)
    - NumberToWords (UKR)
    - Frac
    - Floor
    - Ceil
    - Round
- Retrieve SQL data in user functions bug fix
- AutoHeight for HTML bug fix

QtRptDesigner version 1.5.4

- Added Example 16. How to retrieve Images from DB
- Added Example 17. How to use built-in functions

15.12.2015

QtRPT version 1.5.5

- RichText field bug fixing
- Bug fixing of building Zit library with MSVC. http://stackoverflow.com/questions/24736304/unable-to-use-inline-in-declaration-get-error-c2054
- Changed folder's structure of the project. Created folder "3rdparty". Zint library moved to it
- Bug fixing. Error in Grouping without GroupingFooterBand
- Added following built-in functions: NumberToWords (ESP) for Spanish
- Added following built-in functions: NumberToWords (FR) for French, French(BE), French(CH)
- Frac function corrected
- Examples path fix for MacOS

QtRptDesigner version 1.5.5

- Added possibility open report file in OS via right click and select "open with"
- Bug fixing. Error during creating new report. Page's border was true, page's border width was sets to 99.
- Bug fixing. Set initial height of the Barcode

To Do

- QtRptDesigner. Add background image at design-time.

- Field’s auto width