QtRPT

Version 1.3.0

Programmer: Aleksey Osipov

Email: aliks-os@yandex.ru

QtRPT is a simple 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.

ATTENTION!!!

Some changes made in XML format. Users how used QtRPT version below 1.0.5. You must reopen xml files with QtRptDesigner and resave report.

It is planned to enter in the project some functions of aggregation, mathematical, etc. Due to these please don't use reserved words as a name of variables in fields.

Structure of the report's XML file.

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

```
<reports>
   <report>
      <ReportBand>
         <TContainerField>
         </TContainerField>
      </ReportBand>
      <ReportBand>
         <TContainerField>
         </TContainerField>
      </ReportBand>
   </report>
</reports>
```

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

The <report> may contains up to 6 child nodes <ReportBand> . ReportBand may be one of following type:

- ReportTitle
- PageHeader
- MasterHeader
- MasterData
- MasterFooter

- ReportSummary
- PageFooter

Below the purpose and description of the bands by priority.

- The **ReportTitle** band is a first band which printing at the report. Its 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 **MasterHeader** band printing before MasterData band on each page of the report (if data present on the current page)
- 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 (printing on the last page).
- 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.

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.

Node type	**Attribute**	**Permitted values (type)**	**Purpose**
<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	• ReportTitle • ReportSummary • PageHeader • PageFooter • MasterHeader • MasterData • MasterFooter	Report's type
	name	string	Report's name
<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 • labelImage • image	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
	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
	printing	integer, 0 - is false; 1 - is true	Printable(visible) or not field
	highlighting	string:	

How to use it.

First of all, you must add to your project two files: qtrpt.h and qtrpt.cpp.

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(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 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(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. Show preview of the report

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

And the last, 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 &paramName is a param name which currently rendering
- QVariant &paramValue - value of the current param returned to print engine

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;
    }
}
```

Reserved words

Sum, avg,

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;

How to use Conditions in the fields in the QtRptDesigner?

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

20.09.2013

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.

To Do

- QtRptDesigner. Add background image at design-time.
- QtRPT. Add some aggregate functions.