

CuteReport-Docs



calibre 1.25.0

Introduction

CuteReport is a free report solutions based on Qt4 framework and it can be easily used with any Qt application. Generally CuteReport consists of two parts: core library and template designer. Both are totally modular and their functionality can be simply extended by writing additional modules. It's totally abstract of used data and can use as storage: file system, database, version control systems, etc.

Key Features:

- plugin system for supporting and extending of any functionality wide range of storages to keep report objects: file system, GIT (soon: SVN, database, FTP)
- some datasets for report data: SQL database, CSV files, file system information (soon: XML files)
- runtime forms are supported
- javascript as internal scripting language
- standalone WYSIWYG designer with possibility to extend functionality via new plugins
- Designer plugins: ReportProperty editor, Page editor, Script editor, Dataset editor, Preview. (soon: MultilingualReports editor)
- Designer page editor supports item selection, group selection, moving, size changing, item property changing
- Page header and page footer bands support print or skip first page, print on new page
- Detail band support aggregate functions, zebra colors, print each band on new page
- Memo item supports static data and dynamic scriptable data
- supported measure units: millimeters and inches

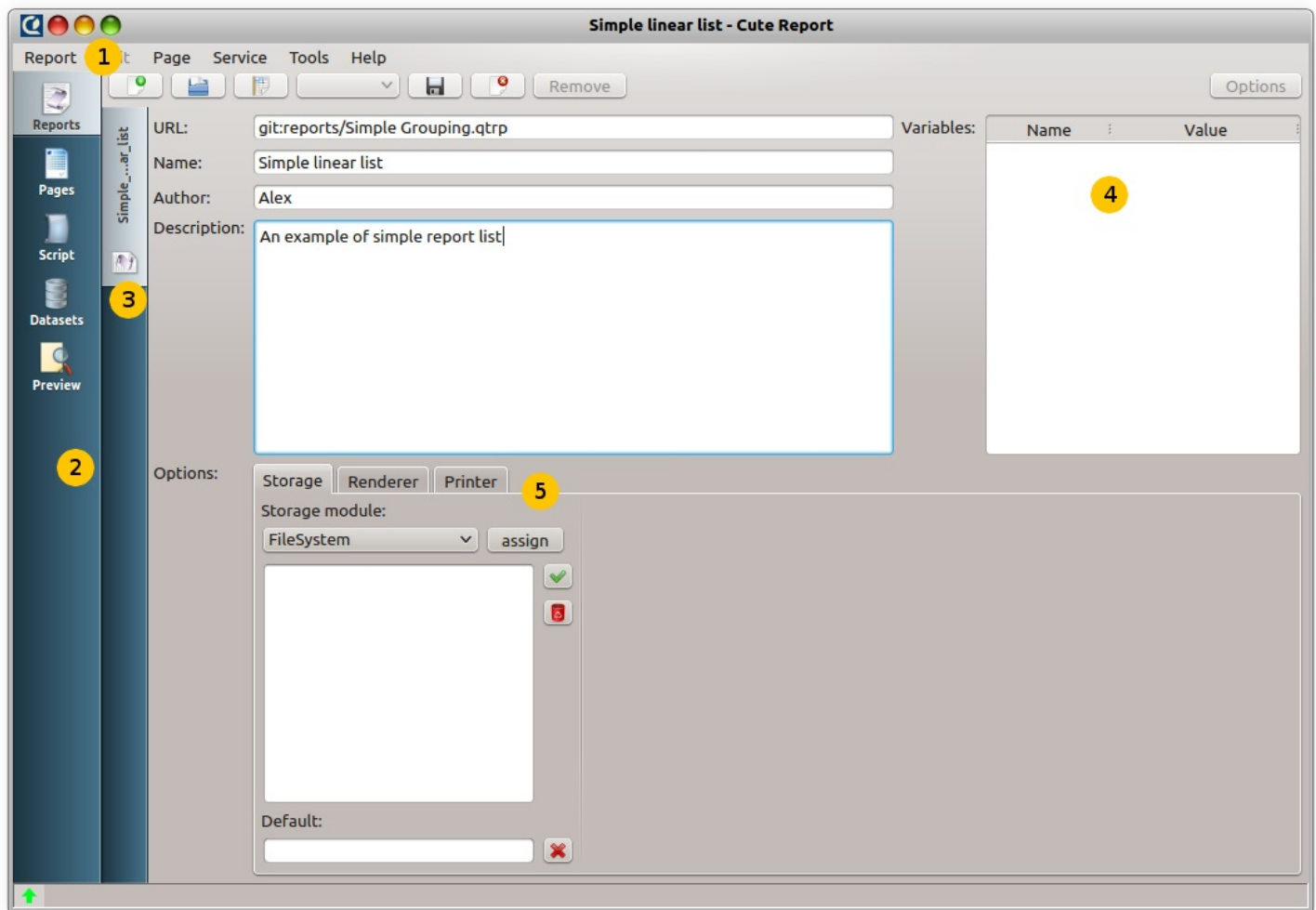
Designer

CuteReport solution comes with a standalone designer which helps to manage CuteReport's templates. CuteDesigner itself has only few functions and provides API to support modules. Modules are used to provide and extend any current or further designer's functionality. Modules can provide (gui-modules) or do not provide (nongui-modules) user interface elements. Some of the basic gui modules are: ReportEditor, PageEditor, ScriptEditor, DatasetEditor, and Preview. Each module provides its own functionality and can be dependent of the other module(s). Also any module can be replaced by another with extended functionality.

Lets take a look at the some of these modules.

ReportEditor module

ReportEditor is the first module in designer's tab bar. It is responsible for managing report objects and providing such operations as: load, save, create, delete. These operations are presented by GUI elements on ReportEditor's widget and are exported to the application's main menu. This module can support a number of open reports at the same time and switch between open reports. Also it manages embedded report's modules like: Storage, Renderer, and Printer. If there are not any joined embedded modules then the global one will be used. If you need special options for Storage, Renderer, or Printer, you should join the module you need to the report object and set the needed options. ReportEditor has a table with global report's variables and their values. These values are used in rendering the report and can be set manually in this table or by software.



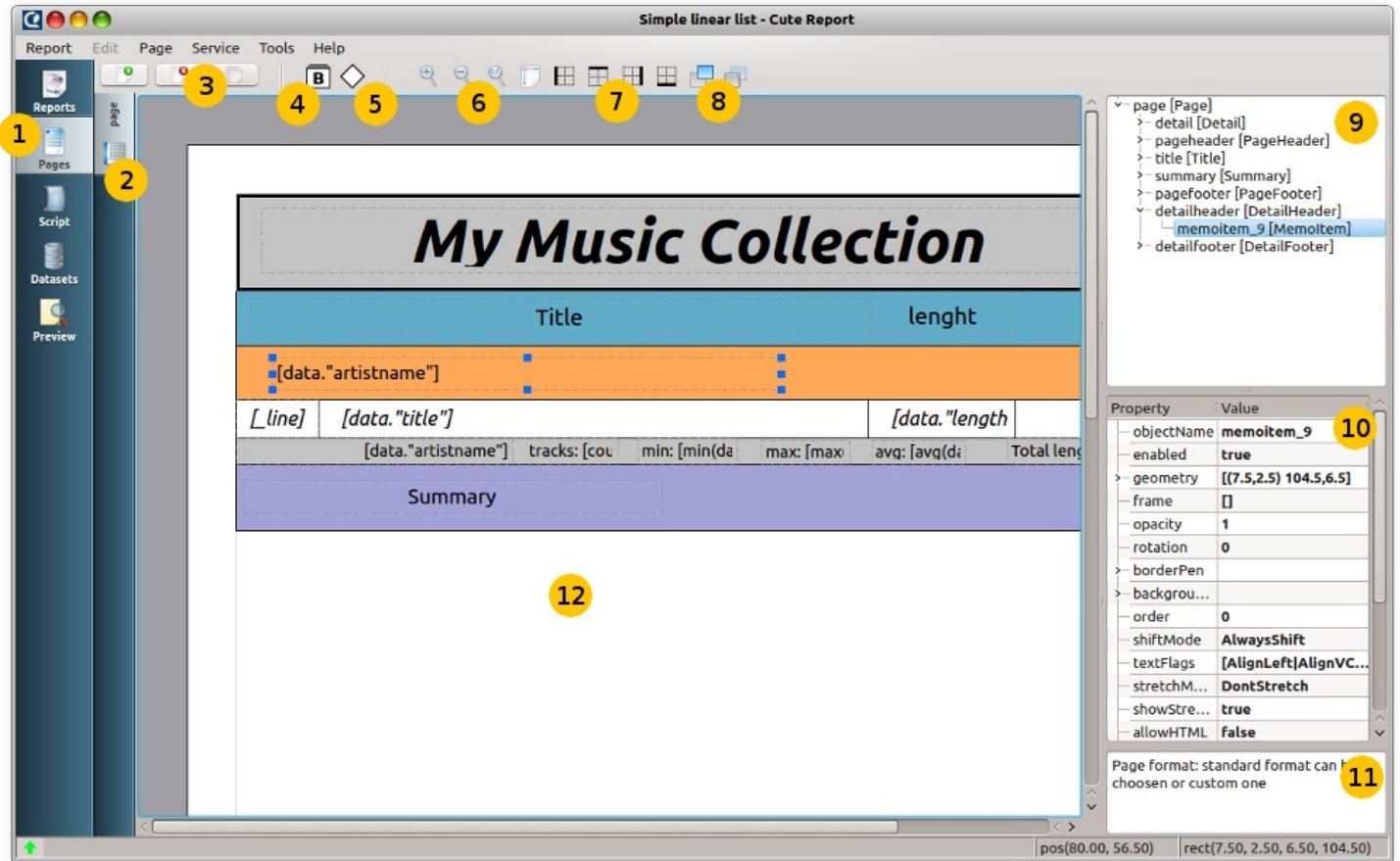
Key to ReportEditor features:

- main menu

- modules bar
- ReportEditor tab
- report variables
- embedded report modules

Page Editor module

PageEditor module is responsible for making the report page template. It provides tools for managing page bands and items. To activate PageEditor press the "Page" tab on the modules bar. Pages bar shows the tab with all created pages. You can use it for switching between pages (mouse click) or for renaming the current page (mouse double-click). There are 3 buttons on the page's toolbar: create new page, delete current page, and clone current page. Next go 2 buttons with drop-down lists, first one for bands, second one for items. After that you can see some buttons for changing zoom and next 4 buttons for enabling/disabling page magnets. If magnets are enabled, the mouse pointer will stick to the other item's borders with the coordinate that is close to the cursor coordinate. Sticking factor can be changed in the page property "magnetRate". Magnet may not always work as you want, so sometimes it's better to disable them. On the right side of the page widget you can see ObjectInspector. All items that page contains are represented there as a tree. You can switch between items using ObjectInspector or by clicking the item on the workspace(#12). PropertyEditor is the next area. There are all the editable properties that the page has. By pressing the property item you can see a short property description (#11). The Workspace (#12) shows the entire page template. You can add a new band of items using drag-n-drop from drop-down list(#4, #5) to the page on the workspace. Almost all items can be placed only on a band, and bands can be placed only on the page directly. Press the "Delete" button to delete the band or item with all their children items. There is no Undo function yet, so be careful.

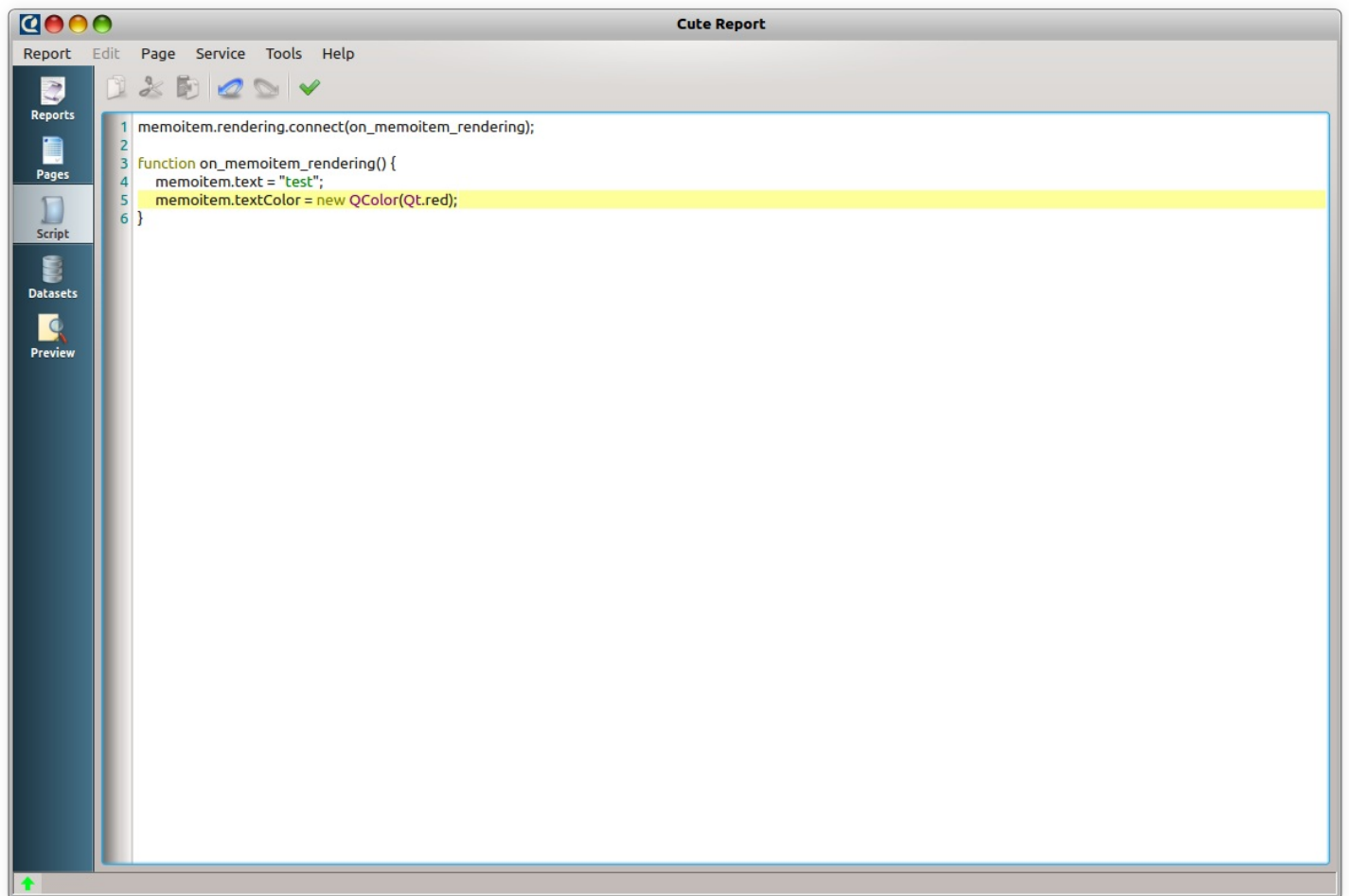


Key to PageEditor features:

- modules bar with PageEditor activated
- pages bar
- buttons: "new page", "remove page"
- drop-down list of bands
- drop-down list of items
- zooming buttons
- magnets enabling/disabling
- rise/lower item
- object inspector
- property editor
- property description
- workspace

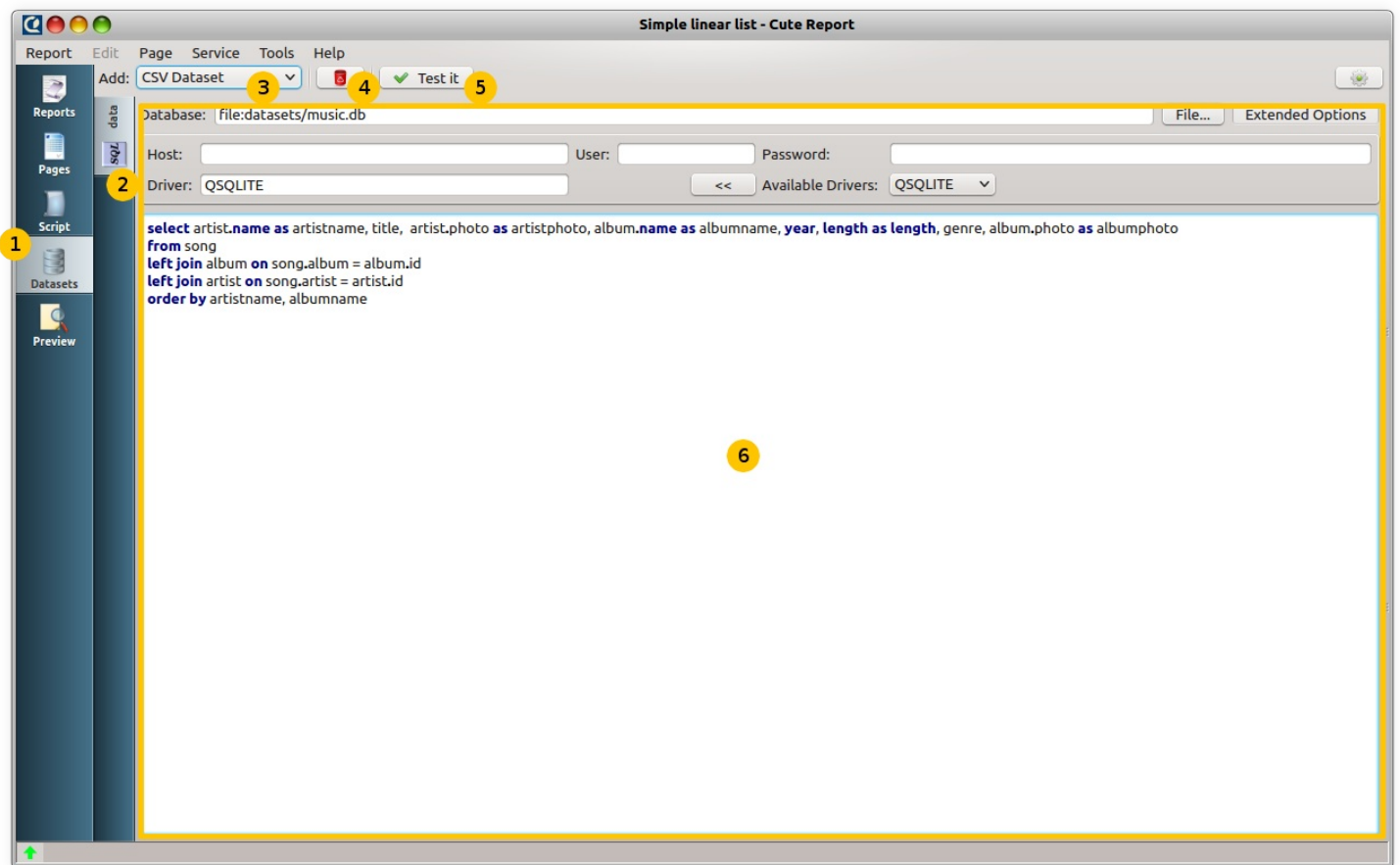
ScriptEditor module

You can switch to ScriptEditor by pressing the "Script" button on the module bar. Script editor is a pretty simple module and contains an editor with syntax highlighting and "Validate" button. Validation checks only the syntax correctness of your script and doesn't actually run the script. So even if your script passed validation, it still may contain runtime errors. Usually you can see all errors by pressing the green button on the left bottom of the Designer window. If there are errors in the script, the button becomes red. ScriptEditor uses javascript as scripting language.



DatasetEditor module

You can switch to ScriptEditor by pressing the "Dataset" button on the module bar (#1). All created datasets of the current report are shown on the datasets bar (#2). Using this bar you can switch between datasets (mouse click) or rename the current dataset (mouse double-click). For creating a new dataset, press combobox #3 and choose the dataset type you want. Basic distribution provides 3 datasets: CSV dataset, SQL dataset and FileSystem Dataset. Each dataset is described below. To delete the current dataset, press button #4. There is no Undo function implemented yet, so be careful while deleting. When you have set all the options for the created dataset, you can press the "Test it" button (#5). All datasets have a common interface and provide data as a table. Each dataset has its own configuration widget (#6).



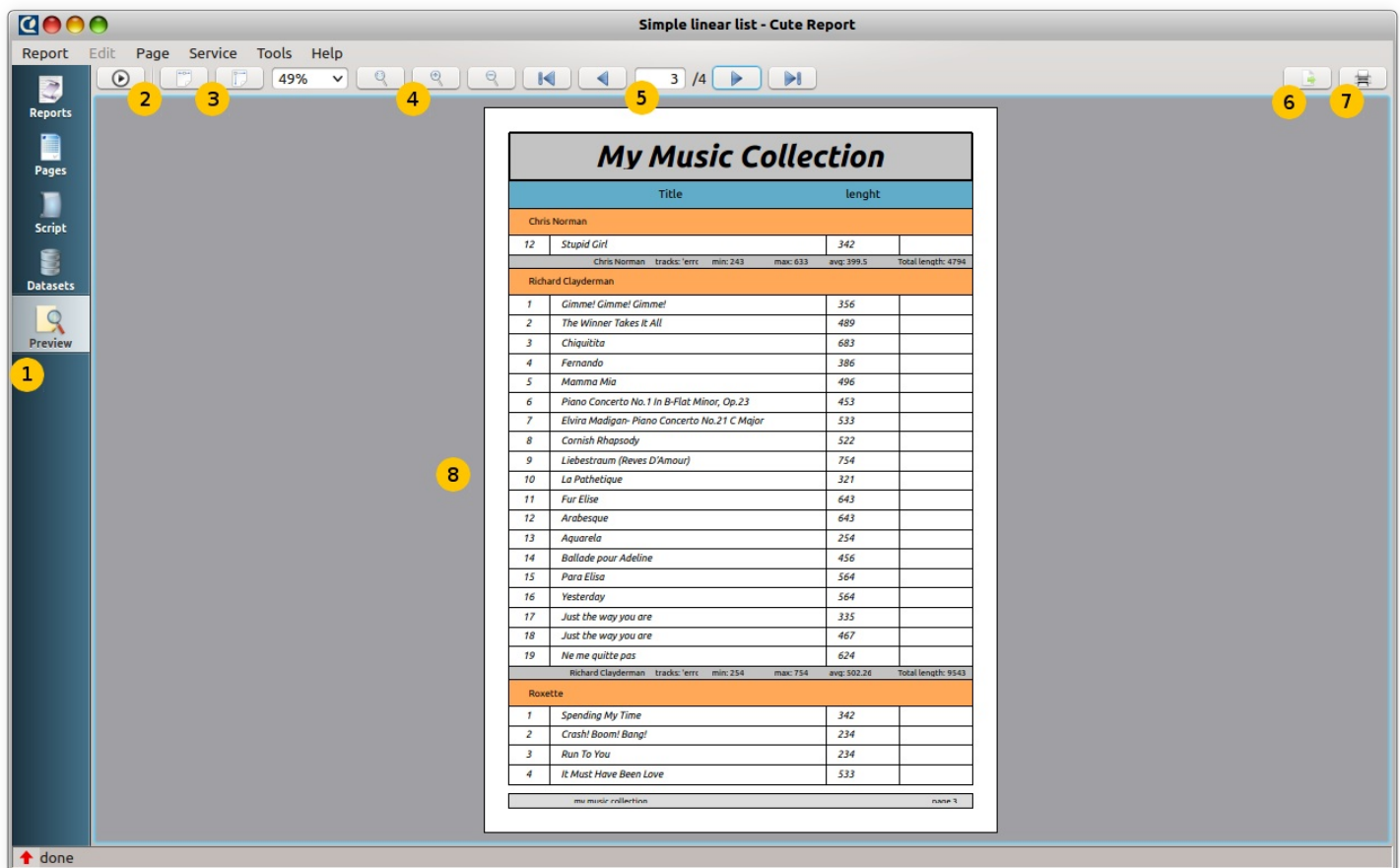
Key to DatasetEditor features:

- modules bar with DatasetEditor activated
- datasets bar
- new dataset combobox
- delete current dataset

- test dataset
- dataset helper

Preview module

Preview module task is to display rendered reports. There are some button groups to help you. First is the button for rendering current report template (#2). Every time you need to render or rerender your report this button will be helpful. Or you can use: Main Menu -> Service -> Render or just press F5. If your report need some time for rendering, process dialog will appears. To stop current rendering press this button again (or press F5). After report is rendered you can change zoom by using buttons: fit to page, fit width, zoom in, zoom out (#3, #4) or you can set any zoom you need in percent widget. To switch current page use buttons: First page, Previous Page, Next Page, Last Page or set page number direct in page number widget. Finally you can print rendered report (#7) or export it to the file(#6).



Key to Preview features:

- modules bar with Preview activated
- Render report button
- fit page to view button group
- zooming button group

- page navigation
- export to file
- print

Creating Report

In this chapter we will review some general aspects of report designing. We will take close on some items and their properties and will make some report examples. Make sure you have CuteReport installed and try to make these examples by yourself using preinstalled test databases. Since CuteReport is in active developing, some parts of this documentation can differ from your CuteReport installation.

Report objects


CuteReport Designer's PageEditor module designed to represent report as a set of schematic pages. All objects are placed somewhere on a report page and they are used to display any text and graphics information. Basic CuteReport objects are included to the Community CuteReport edition package. Some extended objects are included to the Commercial package. Objects are:



- **Bands:**
 - **PageHeader:** band located on the top of page
 - **PageFooter:** band located on the bottom of page
 - **Detail:** band that connected to dataset and processed with each dataset iteration
 - **DetailHeader:** band that located on top of details group
 - **DetailFooter:** band that located on bottom details group
 - **Title:** band that located before detail band(s)
 - **Summary:** band that located after detail band(s)
 - **Overlay:** band with the free accommodation, can be placed anywhere on page without layouts
- **Items:**
 - **Arc:** item that draws arc
 - **Barcode:** item represents barcode
 - **Chart:** item that draws any kind of charts
 - **Chord:** items that draws chord
 - **Ellipse:** item that draws ellipse
 - **Image:** item that draws dynamic or static image in PNG, JPG, BMP and other formats
 - **Line:** item that draws horizontal, vertical or diagonal line
 - **Memo:** item that represents any text information, plaintext and HTML formats are supported
 - **Pie:** item that draws pie
 - **Rectangle:** item that draws rectangle

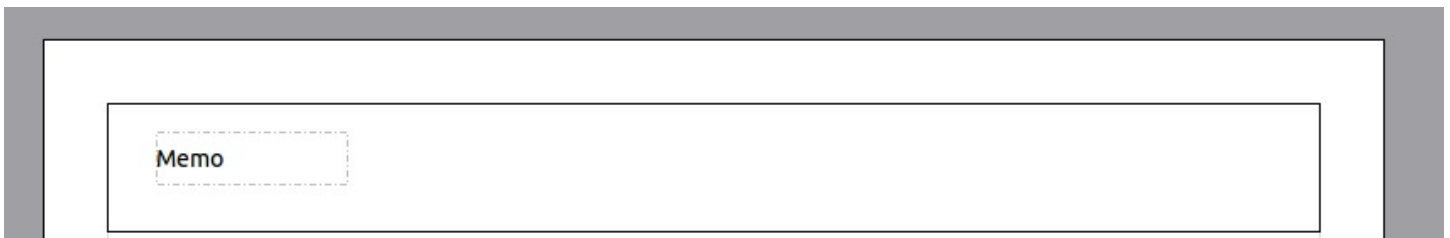
The basic objects most commonly used are the “Band” and “Memo” objects. You will learn about their capabilities in detail later in this chapter.

"Hello World" report example

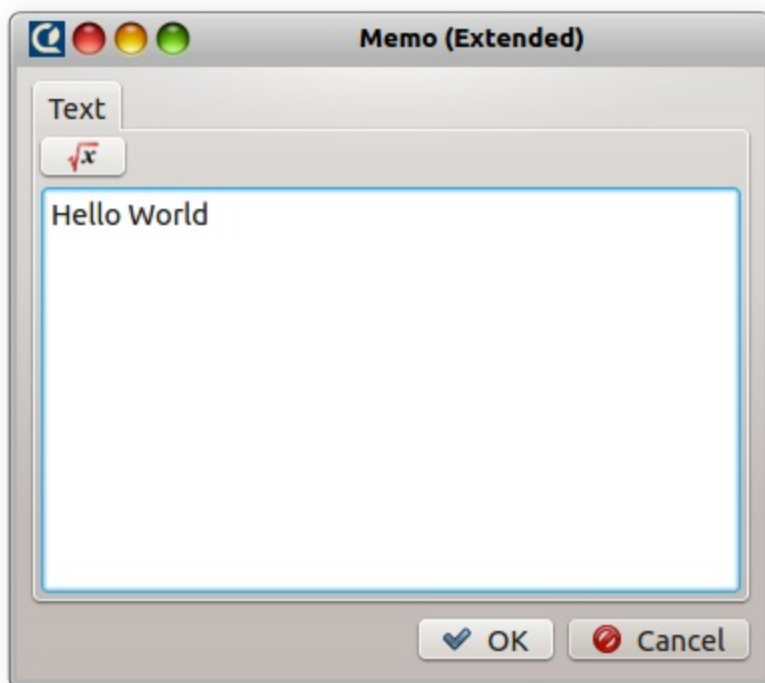
This simple report example contains just one piece of information: "Hello World!" text. Open CuteReport Designer, go to "Page editor" using right tab panel. Since any item can be placed only on carrier band, we must place any band first. Click on the button

"Bands"  and select any simple band, for example PageHeader. Click somewhere on page to place this band (page have to be created before). Then click on button with the

title "Items" or icon  and select "Memo" . There is also Memo (Extended) exists in the commercial version, that extends functionality of the Memo. Use any of them if you have both. After selecting Memo item, click somewhere inside PageHeader to place selected Memo. The object will be placed in mouse position.



Depending of your local settings Memo Helper dialog will appear immediately or you can make double-click on the Memo to show this dialog. Type "Hello World!" and then click "Ok" button.



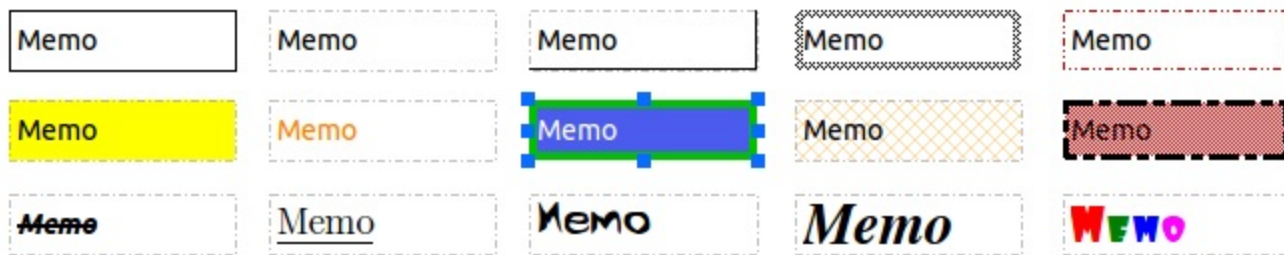
Now Report template is completed. To generate actual report select in main menu "Service->Run" or press "F5" on your keyboard. Designer will be switched to the "Preview" tab and rendered report page with "Hello World!" will appear. Rendered report can be printed or exported to the one of the supported export formats.

Memo object

The Memo object has many great features to draw text on a page. It can draw text in a frame and can be filled with some color. The text can be displayed using any font with any size and style. All the properties can be set using Property Editor ![TODO: or visually with the help of the toolbar]:

[ToolBarImage TODO]

Here you can see some samples:



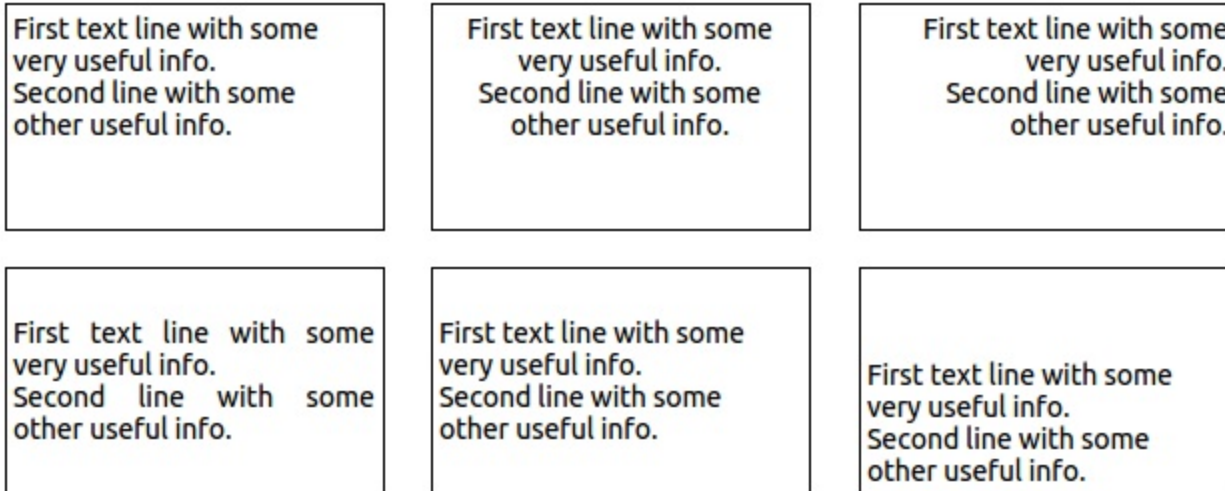
Let's look close at the Memo item features.

Text Aligning

We will make a simple example of Memo with two lines of text:

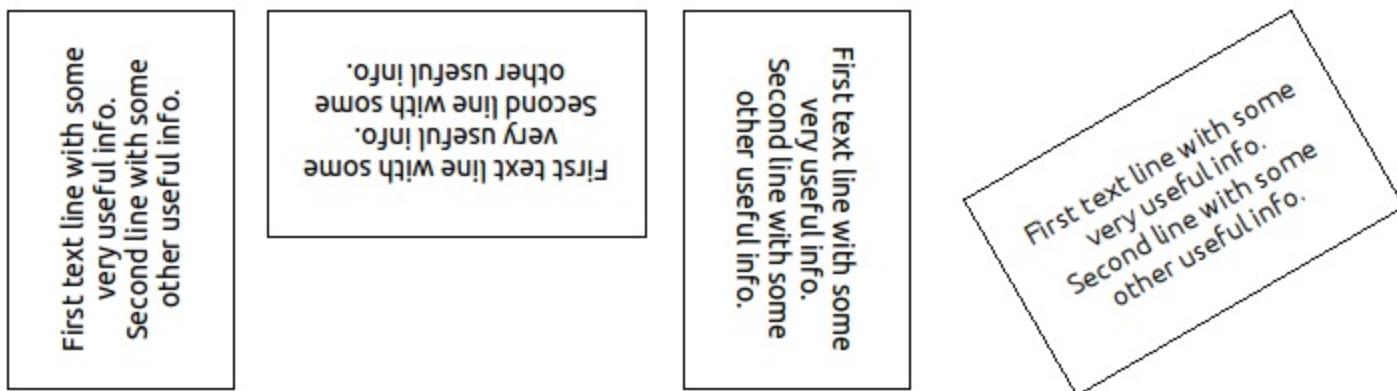
First text line with some very useful info.
Second line with some other useful info.

Enable Memo frame from Property Editor and resize item up to 90x30 mm using mouse or Property Editor. As you can see now Memo can display not only a single line but several lines of text as well. Try to reduce Memo width to 50 mm. Obviously, lines can not fit to the object's border and will be wrapped. This is controlled by **TextFlags::TextWordWrap** object property. If it is disabled any long line will be cut short. Let's play with other **TextFlags** and see what we can have.



Rotating

Lets take a look at the other feature: rotation. Any object including Memo can be rotated to any angle in degree range 0..360. Set required angle in the PropertyEditor by changing property "**rotation**". Memo borders will be aligned accordingly, so you don't need to care about the borders.



HTML tags

Memo object allow most of HLML tags. Tags can be placed within Memo text. Tags are disabled by default. For HTML tags detection set property "**allowHTML**" to "true". There are some examples below.

```
<i>Memo</i> <b>example</b><br>
```

```
E = mc<sup>2</sup><br>
```

```
A<sub>1</sub> = B<sup>2</sup><br>
```

```
this is a usual text, <font color=red>and this is a red one  
</font><br>
```

```
this is a usual text, <font color="#FF8030">and this is an orange  
one</font>
```

Memo **example**

$E = mc^2$

$A_1 = B^2$

this is a usual text, and this is a red one

this is a usual text, and this is an orange one

Expressions

Expressions is one of the most important feature of Memo object. It allows to display not only a static text, expressions results as well. Expressions can be mixed with static text. Enter text above to the Memo object:

Now is [QDateTime.currentDateTime()]

and render report by pressing "F5" on your keyboard. You'll see result of the rendering, something like that:

Now is Fri Jul 18 2014 00:44:22 GMT-0700 (PDT)

Why is that so? CuteReport's renderer recognizes every expression instance, calculates it and replaces expression with the result. Memo text can contain a number of expressions. Expressions can use complex arithmetic, constants, variables, objects and theirs properties. But there are some potential issues exists if our normal text contains square brackets that don't mean to be an expression. For example, we want to draw:

`array[0] = 'Banana'`

Entry [0] will be recognized as expression, calculated with result 1 and placed to our text. As result we will see:

```
array0 = 'Banana'
```

which is definitely not what we suppose to see. There are 2 solutions:

- set "**allowExpressions**" property to "false"
- change default square brackets to any symbols or symbols set you want in the "**expressionDelimiter**" property

In the first case entire text will be recognized as regular text without expressions. In the second case expressions will be recognized using another "begin sign" and "end sign". You can use "<" as begin and ">" as end, but only if you don't use HTML text. If you do use HTML you might use "<%" as begin and "%>" as end. Expression detector works before text rendering so any your expression delimiters will be cut off from your text. Do not use the same symbol set as the "begin" and the "end" sign.

Other Memo properties: [TODO]

- **stretchMode**: object stretchability to fit text
- **showStretchability**: do stretching not only on rendered page, but on Designer' PageEditor too
- **expressionDelimiter**: two string separated by comma which means begin and end of scripting block.
- **stretchFont**: set automatic font size to fit Memo text within Memo width

Bands

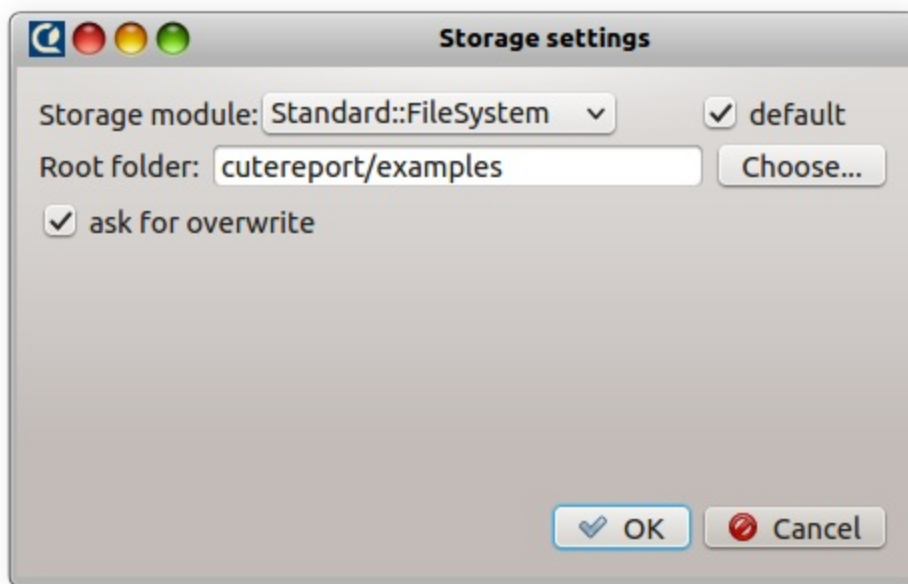
Bands are designed for positioning items on a page. Each band has its own position and functionality. CuteReport has some special bands designer to represent data from a dataset. Dataset contains structured data organized into rows(lines) which have one or more columns(fields). To print data from datasets CuteReport uses these special bands named "Detail...". To make it work, add one or more such bands to the page, connect them to the dataset and place Memo items on them. Once band is connected to the dataset, Memo will have button on the right side with the dropdown list of dataset's fields names (commercial version only). While rendering process these bands will be printed on the rendered page. "Detail" band will be printed once per dataset row, DetailHeader and DetailFooter will be printed accordingly to theirs "condition" property. If there is no free space to print new band, new rendered page will be created before continue. New page will print all page headers and footers before continue to print Detail bands.

There are some bands included into standard package and theirs short description:

- **PageHeader:** displays all nested items on the top of the page
- **PageFooter:** displays all nested items on the bottom of the page
- **Title:** displays all nested items on the report begin
- **Summary:** displays all nested items on the report end
- **Detail:** must be joined to a dataset and displays all nested items on every dataset iteration
- **DetailHeader:** must be joined to a dataset and can be shown on every dataset iteration or when "condition" property calculated as "true". It can be used for grouping for show group header
- **DetailFooter:** must be joined to a dataset and can be shown on every dataset iteration or when "condition" property calculated as "true". It can be used for grouping for show group summary
- **Overlay:** can be located anywhere on a page without respecting any layouts. Can be used as carrier band for foreground, background, watermarks.

Storages

Right now we will try to figure out what is storages and how to work with them. Storage it is... well, storage, the place where all report's object are stored. It can be filesystem, GIT, resource, HTTP server, FTP server. Certainly, you are familiar with some of them. CuteReport can use a lot of objects for report rendering, such as images and database files. Report files (templates) also stored somewhere. Usually you use your local filesystem to store all these objects as a file. But CuteReport is not limited by only filesystem. When you start CuteReport first time you will see "Storage settings" dialog:



If you press on combobox on the top of the screen, you can see a list of all storages available. You need to select your primary storage and check "default". "Default" means storage for all url where schema is not indicated. For example, if you set default storage name as "Filesystem" then any objects with short url like "/objects/image.png" will be transformed to full url "file:/object/image.png". If you set "GIT" as default storage, then object "images/logo.png" will be transformed to "git:images/logo.png". Absolute or relative path is acceptable. You can change these settings anytime in the main menu "Tools->Options->Storage". Storage can be global or report's local. global storages are used by CuteReport core. If report object has its own assigned local storage than it will be used before global one.

FileSystem storage

FileSystem is the most commonly used storage. It has only few options: "root folder" and "ask for rewrite". Root folder is the upper directory accessible to user. "Ask for rewrite" option is used to detect is it needed to show dialog for overwrite file if it's already exists while saving.

GIT storage

It can be used for keeping all reports and theirs objects in local or remote GIT. Options:

- remote url: git repository url
- login, password: credentials to access git repository
- local path: local directory where git repo will be cloned in
- git binary: git console binary. CuteReport uses external git binary to operate git repository, so it has to be defined
- "sync now" button: button for cloning or pulling data from remote repository

Resource storage

All objects stored in this storage will be included to report's file while saving.

All storages has it's own URL schema like: "git", "file", "res", "sql" so any file on these storages can be accessed using "git:objects/test.png", "res:/prefix/file1.jpg", "file:/home/user/file.bmp".

SQLStorage

This storage is designed to make you easy to save and load report templates and report objects from SQL databases without writing any code in your application. Just provide info about database, table and field where reports or object are stored.

Datasets

As it was mentioned before Dataset is the objects that contains structured data organized into rows(lines) which have one or more columns(fields). Dataset can fetch data from any source and provide common interface to this data. There are some datasets provided in the basic CuteReport edition: SQLDataset, CSVDataset, FilesystemDataset, ModelDataset. All of them fetch data from different sources.

Datasets:

- **SQLDataset:** It provides interface to fetch data from SQL databases, It can work with any databases supported by Qt, ie that has Qt database driver. There are some setting to connect to a remote database (like mysql, postgresql) or to use database file (like sqlite)
- **CSVDataset:** It provides interface to data stored in a file and separated by comma or other defined symbol. It can load data from an external file every time when populated or load and cache data internally
- **FilesystemDataset:** it provides interface to fetch information from filesystem, and show files, directories and more less detailed info about them. There are some options: filtering, recursion level, max number of exposed records. It is useful for making files or photo catalogs, etc.
- **ModelDataset:** it is used to export prepared data from custom application to report object. If you have widget like QTableView or QTableWidget or your custom filtered/sorted model you can easily print data stored in there.

Try to play with each dataset to understand how they work. You can see the populated dataset rows anytime by pressing "Test it" button. Data from dataset can be used in scriptable form in items. To get data from dataset where it's allowed you can use [datasetname."fieldname"] or [datasetname.value("fieldname")] form. First form is just shortening of the second one. Every short form replaced by full form internally before script execution.

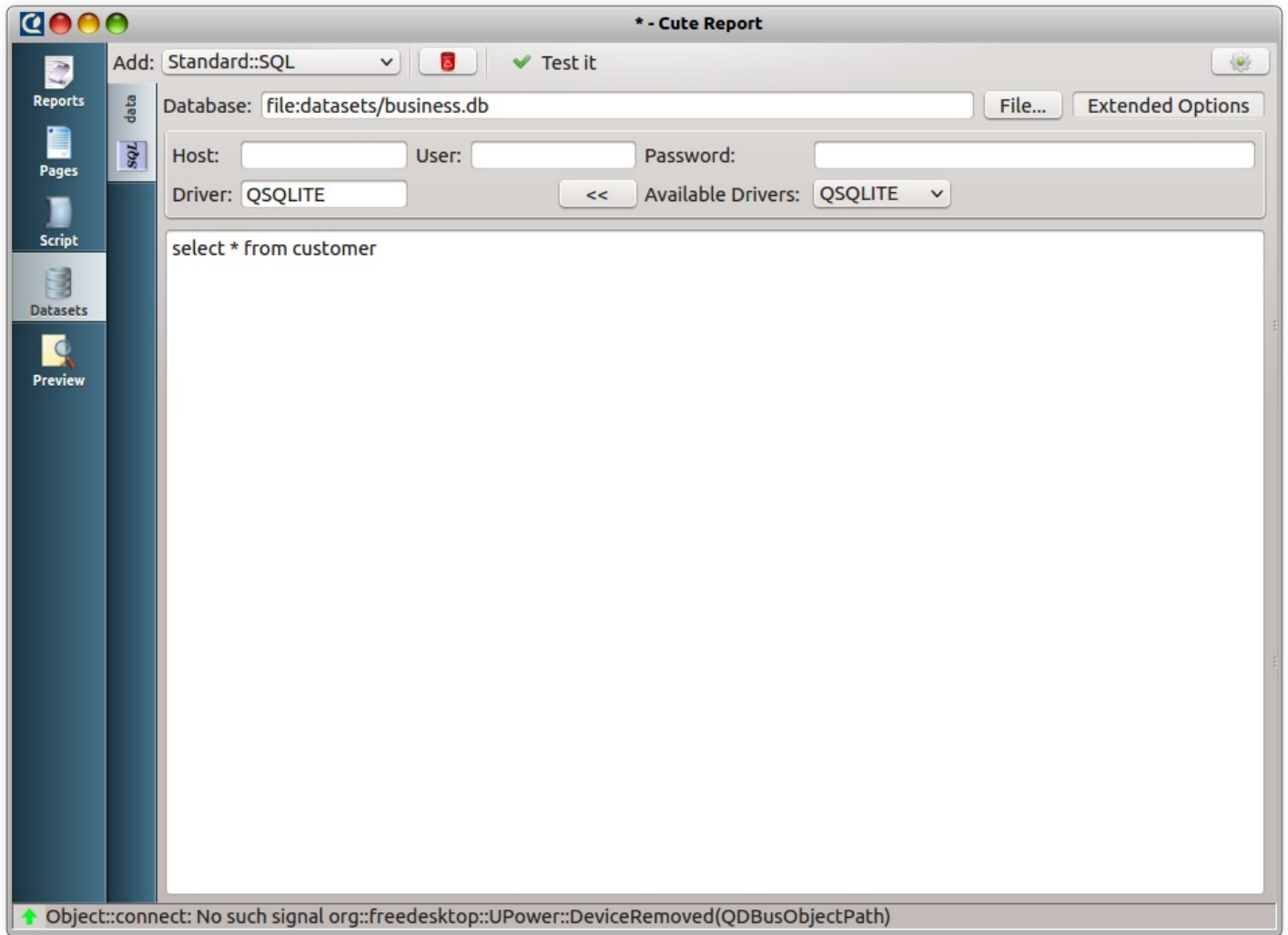
Further we will see how to connect dataset to a band and to use data more detailed.

"Customer List" example

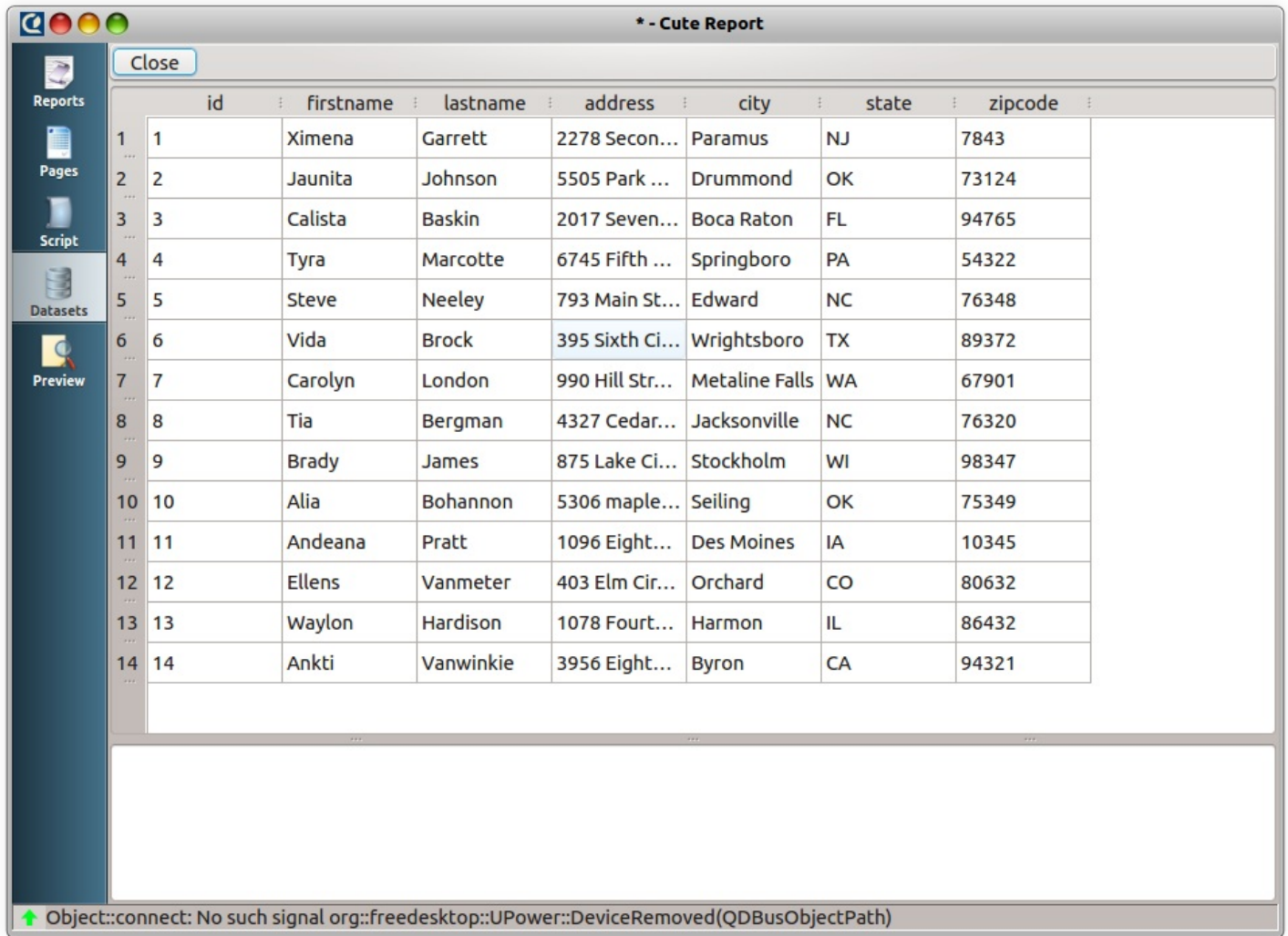
Now we will create our second report and will learn how to use datasets. For that we will use test sqlite database named "business.db". By first create new empty report by pressing "main menu -> Report -> New Report". Then go to the "Datasets" tab, click on dataset names combobox and select SQL. Now you have one SQL dataset in your report. Lets set correct parameters for the dataset. Click "File...". When "Open database file" dialog appears, make sure you have correct storage selected in the "Storage" combobox, locate database file named "business.db" and choose it. Now you should have field "Database:" filled with "file:dataset.db". Since we have no host, user and password for this database we will skip these fields. Choose "SQLITE" driver in the "Available Drivers:" list and press "<<" to copy driver name to the field "Driver". Now we can connect to our test database. Lets write a simple SQL query to test it:


```
select * from customer
```

As a result you will have something like on the picture below:

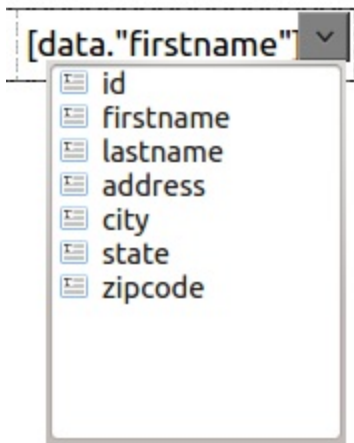


Click "Test it" button and table with fetched from database data will appear. If there is error exists you will see it in the bottom frame. Well done! Now you have dataset completed.



Go to the tab "Page" and create new page if not exists. There are some types of page possible. For now we will use "Standard Page" or "Extended page"(for commercial version). Click on combobox and select appropriate page type and then click on the button "Add new page" . Add "Title" band to the page, and set correct size by dragging mouse on blue handle or by setting correct size in the PropertyEditor. Now place "Memo" item to the center of the band and set correct geomery as well. Double click on the Memo and type "Customer List" and press "Ok". To make this text centered, click on "TextFlags" property in the PropertyEditor and enable "AlignHCenter" flag. Next step is adding "Detail" band to the page and joining it to our customers dataset. To do that press on the Dataset and to make it active and type "data" to the band's property "dataset". "Data" is the name of our dataset. You can change it anytime by double-clicking on the dataset tab in the DatasetsEditor. Now it's time to add some "Memo" items to the band: number, first name, second name, address, city, zip code. For make it easy to stick each other, you can enable magnets by pressing buttons on the top of PageEditor. If you want to change name of the objects and make it more understandable in the Object Inspector, go to PropertyEditor and change "objectName" to something like: memoFirstName, memoSecondName, memoAddress and go on. We

don't use these names in this example, but it can be useful to you later. Next to add instructions to our Memos what to display. Go to the first one, double click on it and type "[_line]". As we discussed before "[string]" means expression and _line is internal variable represents current dataset row. Press "F5" and you will see how does it work. Return back to the PageEditor by clicking on "Pages" at the leftsize and fill memo with the customer's first name. Add text "[data."firstname"]" to this Memo. User of the CuteReport commercial edition can simply click on the button that appears on the right side of the Memo as you can see here:



Do the same for all other items and set correct dataset field to draw. Press "F5" and you should see something like this:

Customer list					
1	Ximena	Garrett	2278 Second Street	Paramus	7843
2	Jaunita	Johnson	5505 Park Boulevard	Drummond	73124
3	Calista	Baskin	2017 Seventh Street	Boca Raton	94765
4	Tyra	Marcotte	6745 Fifth Circle	Springboro	54322
5	Steve	Neeley	793 Main Street	Edward	76348
6	Vida	Brock	395 Sixth Circle	Wrightsboro	89372
7	Carolyn	London	990 Hill Street	Metairie Falls	67901
8	Tia	Bergman	4327 Cedar Avenue	Jacksonville	76320
9	Brady	James	875 Lake Circle	Stockholm	98347
10	Alia	Bohannon	5306 maple Avenue	Seiling	75349
11	Andeana	Pratt	1096 Eighth Boulevard	Des Moines	10345
12	Ellens	Vanmeter	403 Elm Circle	Orchard	80632
13	Waylon	Hardison	1078 Fourth Boulevard	Harmon	86432
14	Ankti	Vanwinkie	3956 Eighth Circle	Byron	94321

If you don't have such result, you can load report CustomerList.qtrp from CuteReport package and figure out what you did wrong.

Image object

Image object is designed to represent any images in supported graphical formats. Currently supported formats: BMP (Windows Bitmap), GIF (Graphic Interchange Format), JPG (Joint Photographic Experts Group), JPEG (Joint Photographic Experts Group), PNG (Portable Network Graphics), PBM (Portable Bitmap), PGM (Portable Graymap), PPM (Portable Pixmap), XBM X11 (Bitmap), XPM (X11 Pixmap). Lets look closer to this object. Create new report, add new page, add Image item. There are some data sources for the Image available: Static, Storage, Dataset. Type of source defined in **sourceType** property. Lets review each option:

- **Static** allows you to load Image from the file and keep loaded data inside the object. File must be loaded manually by pressing "image" property in the PropertyEditor.
- **Storage** allows you to load file in runtime. Image property "source" must be defined and it should be file url. If url schema is not defined, url schema from default storage will be added. Property "source" can contain normal text url and/or expression. For example , if source = "[data."filename"]" it will take file name from a database and will load this file from a storage.
- **Dataset** allows you to load file from dataset blob. Image property "source" must be defined the same way as for "Storage".

Some of the other properties. if **keepAspectRatio** is set to "true", image when scaled will always keep original aspect ratio. If **center** is set to true, image when scaled with keeping aspect ration will always be centered.

Report with Images

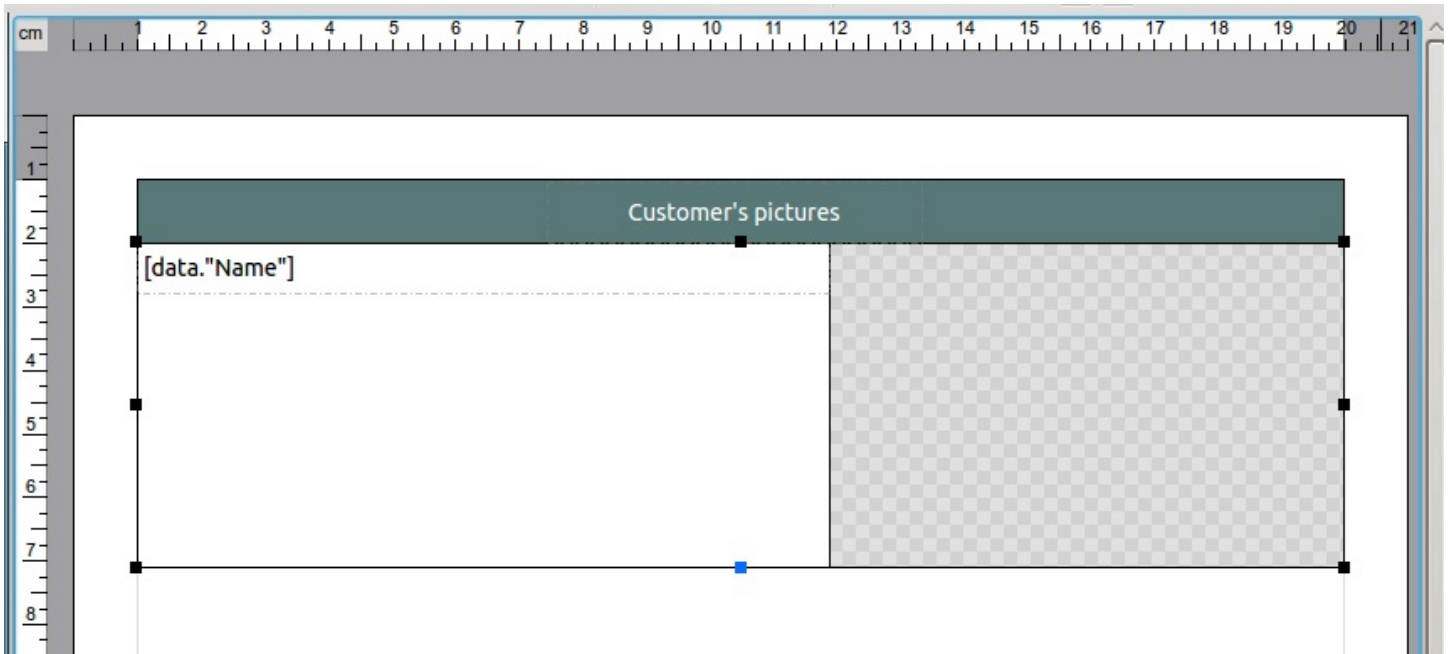
In this chapter we will learn how to use Image object and FileSystem storage. Lets create new report, add new page, add Title band and Detail band. Nexts is creating FileSystem dataset. Go to "Datasets" tab, choose Standard::FileSystem and press it. Choose any directory that contains some pictures by pressing "Select dir..." button. Set "maxNumber" to 6. This is maximum records that dataset will contain. Disable flags: Directories, All Directories. Add filter: ".jpg; .png" or any other graphic format you have. Set "Path appearance" to "AbsolutePath", so we will be able to load picture using it's absolute path. Now press "Test it". As result you should see a list of 6 files or less with file format you set in your filter.

Now go to the Page Editor. Add Memo object to the Title, type there "Customer's pictures" and make text centered. For better appearance change "backgroundBrush" for the Title. set "backgroundBrush::style" to "SolidPattern" and "backgroundBrush::color" to #688482. Now change text color. Click on the Memo and change property "textColor" to white. Well done!

Now we are starting to make image display. Set height of the Detail to 30mm. Add Image object to the right side of the Detail and change size to fit to the detail. set "sourceType" = "Dataset" and "source" = "file://[data."name"]". As you remember everything inside square brackets will be identified as expression and will be replaced by expression result. So our expression finally will looks like "file://picture_path/picture.jpg". Well there is another very important thing: **if report loads any data in runtime, it MUST have storage assigned!**. This is done for security. In some cases you may not allow user to use any storage CuteReport has. So go to the "Reports" tab, and add "Standard::FileSystem" storage in the "Storage" tab. We have images path as absolute path, so clean up "Root folder" for this storage.

And the last step: Memo item that contains file's path. Add new Memo to the right side of the Detail band and type there: "[data."Name"]". Users of the Commercial edition can simply press on the button, that appears when you hover mouse over the Memo.

Finally you report template should look like this:



Now it's time to press "F5" and enjoy result:

Customer's pictures

/home/alex/temp/cutereport/git_storage/objects/116521662132



/home/alex/temp/cutereport/git_storage/objects/1218095387_c



/home/alex/temp/cutereport/git_storage/objects/57.GIF



/home/alex/temp/cutereport/git_storage/objects/67.GIF



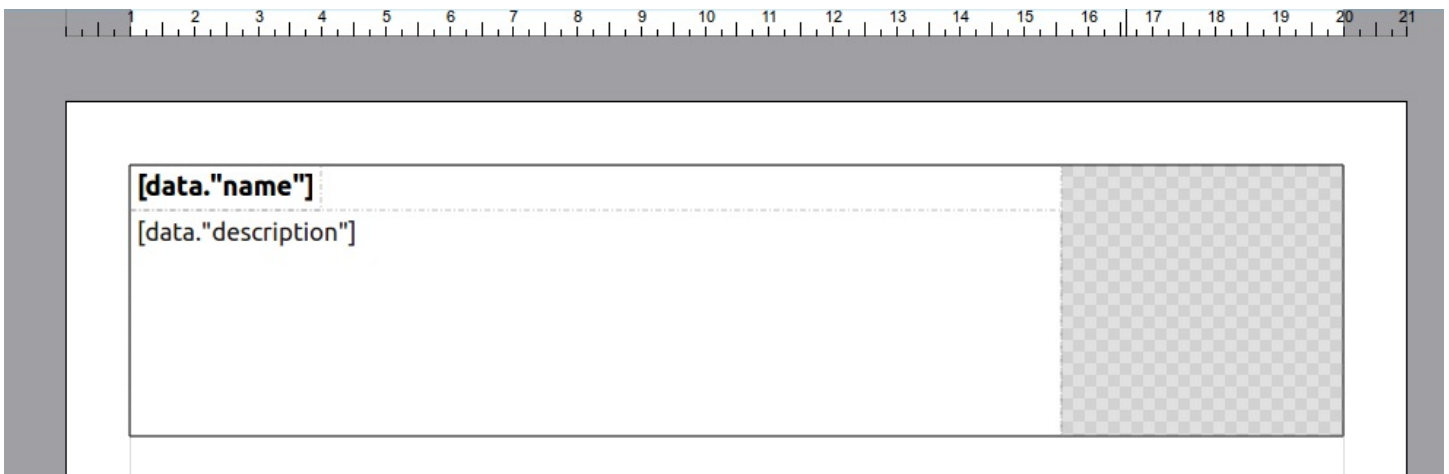
/home/alex/temp/cutereport/git_storage/objects/914813250.jpg



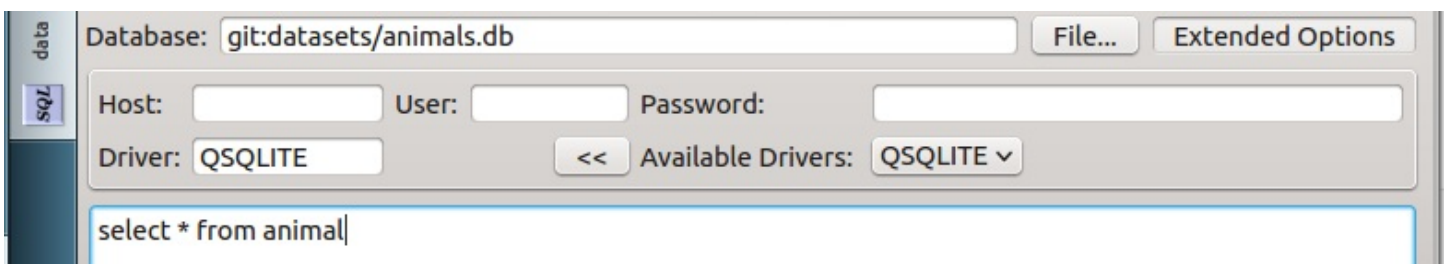
Multi-lined text display

Let's proceed further and learn how to manage multi-line text. In the previous chapter we have learned how to make new report, create dataset and connect dataset to a band. So let's do it:

- create new empty report
- add new SQL dataset and use test sqlite dataset "animals.db" which you can find along the CuteReport distribution or take it there:
https://github.com/AlFoX/CuteReport_examples/tree/master/datasets.
- add Detail band
- put 2 Memos on the band: first is the animal name, second one is the animal description
- put Image item to the right side of the band









Now let's look closer how to make Dataset. After you have added SQLdataset, point it to your "animals.db" file, set Driver - "SQLITE" and add sql query - "select * from animal". Press "Test it" and check if all fine. Below you can see how it should look.



When this part is completed, go back to the PageEditor and set correct fields to display for the Memo's. First topmost Memo is the animal's name. So double-click it and type: [data."name"]. Users of the commercial version, can simply click on the appeared

button on the rightside of the Memo and choose field from the dropdown list. If it doesn't appear check if your band is connected to correct dataset, ie dataset under Memo has filled field "dataset". Set Bold text for the animal name, by clicking on the "font" property in the PropertyEditor. Now go to the second Memo. It's description. So type or select there: [data."description"]. Simple, huh? Lets look the result and render our report (press "F5"):

Capybara The Capybara is a large, semi-aquatic rodent that is found inhabiting the water-logged regions of Central and South America. Closely related to other South American rodents such as Chinchillas and Guinea Pigs, the Capybara is the largest rodent in the world weighing up to 75kg and measuring nearly 1.4 meters long. Despite their enormous size though, these mammals have adapted well to life in the water and have a number of distinctive characteristics that aid their amphibious lifestyle, including the webbed skin between their toes which is	
Abyssinian The Abyssinian Cat is thought to be one of the oldest breeds of domestic Cat in the world, as the first domestication of the Abyssinian Cat occurred in Ancient Egyptian times. It is thought that Abyssinian Cats were bought and sold on the banks of the River Nile by traders, where the African Wild Cats (the ancestors of all domestic Cats) lived in their native habitats. Abyssinian Cats are most easily identified by their "ticked" fur which gives their coat a mottled appearance.	
Adelie Penguin The Adelie Penguin is the smallest and most widely distributed species of Penguin in the Southern Ocean and is one of only two species of Penguin found on the Antarctic mainland (the other being the much larger Emperor Penguin). The Adelie Penguin was named in 1840 by French explorer Jules Dumont d'Urville who named the Penguin for his wife, Adelie. Adelie Penguins have adapted well to life in the Antarctic as these migratory Birds winter in the northern pack-ice before returning south to the Antarctic coast for the warmer summer months.	
Baboon The Baboon is a medium to large sized species of Old World Monkey that is found in a variety of different habitats throughout Africa and in parts of Arabia. There are five different species of Baboon which are the Olive Baboon, the Guinea Baboon, the Chacma Baboon, the Yellow Baboon and the Hamadryas Baboon which differs most from the others wide it's bright red face and cliff-dwelling lifestyle (the other four species are collectively known as Savanna Baboons). However, there is some debate over the classification of the different species	
Camel The Camel (also known as the Dromedary Camel, the Arabian Camel and the One-Humped Camel) is a large hooved animal that is most commonly found in the hot deserts of Northern Africa and the Middle East. Thought to have been first domesticated by native people more than 5,000 years ago, these hardy animals have proved vital to the survival of humans in these areas as they are not just used for transporting both people and goods, but also provide a good source of milk, meat and wool. The Camel is one the most unique mammals on the planet.	
Dog Dogs are thought to have been first domesticated in East Asia thousands of years ago. People primarily used dogs for guarding the hunters and areas of land. Today's domestic dog is actually a subspecies of the gray wolf, a type of dog that is feared by most humans. Many people today, in all countries around the world, keep dogs as household pets and many even regard their dog as a family member.	

Doesn't look good, right? Some of the descriptions were cut-off. Sure we can simply

change description Memo height to fit largest text, but there are some disadvantages:

- paper wasting, since we will not use entire Memo's room for small text
- you newer know how long text will be or it can be changed in future
- it just doesn't look pretty :)

Go ahead and fix it: set Memo property "stretchMode" to "ActualHeight", set Detail propery "stretchable" to "true" and generate report.

Capybara

The Capybara is a large, semi-aquatic rodent that is found inhabiting the water-logged regions of Central and South America. Closely related to other South American rodents such as Chinchillas and Guinea Pigs, the Capybara is the largest rodent in the world weighing up to 75kg and measuring nearly 1.4 meters long. Despite their enormous size though, these mammals have adapted well to life in the water and have a number of distinctive characteristics that aid their amphibious lifestyle, including the webbed skin between their toes which is particularly helpful when swimming. Interestingly enough, the common name of the Capybara is thought to mean "Master of the Grasses", whilst it's scientific name comes from the Greek word for water hog.



Abyssinian

The Abyssinian Cat is thought to be one of the oldest breeds of domestic Cat in the world, as the first domestication of the Abyssinian Cat occurred in Ancient Egyptian times. It is thought that Abyssinian Cats were bought and sold on the banks of the River Nile by traders, where the African Wild Cats (the ancestors of all domestic Cats) lived in their native habitats. Abyssinian Cats are most easily identified by their "ticked" fur which gives their coat a mottled appearance.



Adelie Penguin

The Adelie Penguin is the smallest and most widely distributed species of Penguin in the Southern Ocean and is one of only two species of Penguin found on the Antarctic mainland (the other being the much larger Emperor Penguin). The Adelie Penguin was named in 1840 by French explorer Jules Dumont d'Urville who named the Penguin for his wife, Adelie. Adelie Penguins have adapted well to life in the Antarctic as these migratory Birds winter in the northern pack-ice before returning south to the Antarctic coast for the warmer summer months.



Baboon

The Baboon is a medium to large sized species of Old World Monkey that is found in a variety of different habitats throughout Africa and in parts of Arabia. There are five different species of Baboon which are the Olive Baboon, the Guinea Baboon, the Chacma Baboon, the Yellow Baboon and the Hamadryas Baboon which differs most from the others wide it's bright red face and cliff-dwelling lifestyle (the other four species are collectively known as Savanna Baboons). However, there is some debate over the classification of the different species due to the fact that some have been known to interbreed, indicating that they could be sub-species instead. Baboons are incredibly sociable and intelligent animals that are known to form close bonds with other members of the troop that often last for life. They are also incredibly adaptable animals but their population numbers are declining throughout their natural range primarily due to hunting and habitat loss.



Camel

The Camel (also known as the Dromedary Camel, the Arabian Camel and the One-Humped Camel) is a large hoofed animal that is most commonly found in the hot deserts of Northern Africa and the Middle East. Thought to have been first domesticated by native people more than 5,000 years ago, these hardy animals have proved vital to the survival of humans in these areas as they are not just used for transporting both people and goods, but also provide a good source of milk, meat and wool. The Camel is one the most unique mammals on the planet and has adapted perfectly to life in the desert where food and water can often be scarce, and the temperature changes rapidly from the scorching-hot days to the cooler nights. However, although they would have once been found freely roaming the Arabian deserts, they are today extinct from the wild but the domestic population is widespread and numerous.



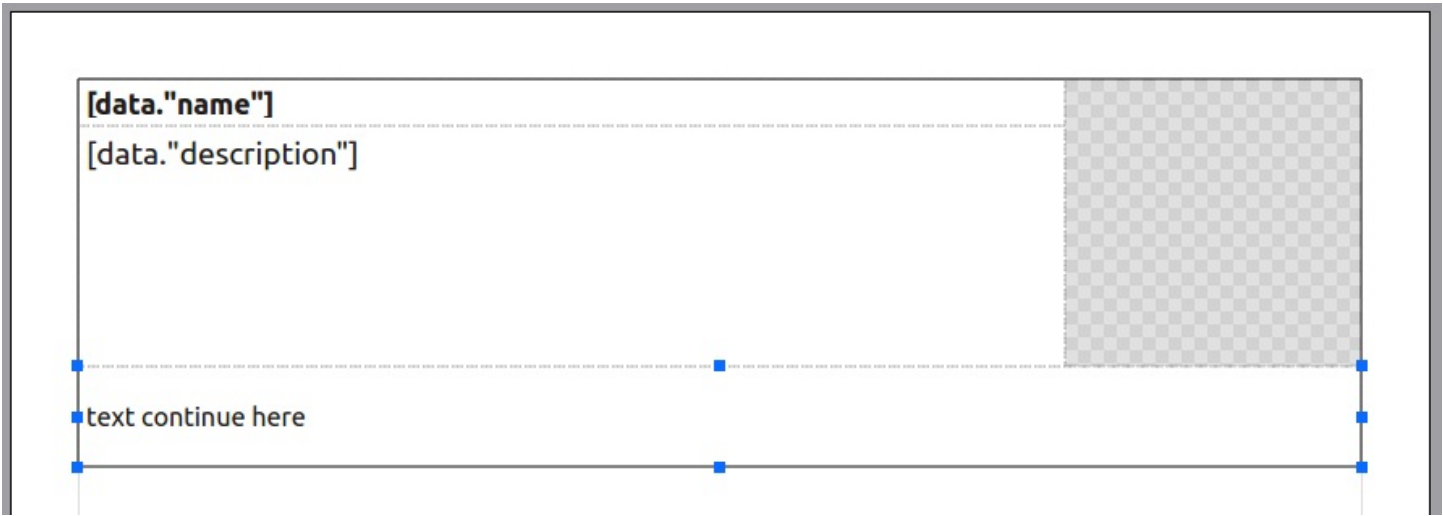
As you can see "stretchMode" do the work. There are all options:

- DontStretch - do not stretch the object
- ActualHeight - stretch the height of the object to fit all assigned text
- MAxHeight - stretch the height of the object to reach bottom of the band

Text wrap of objects

(commercial version only)

Sometimes you might want to develop report design that requires text wrapping around other objects. it can be Image or table. This is simple challenge with CuteReport. Simply add one new Memos where text should flow to. For the second Memo set property "flowTo" with the name of the first Memo where is text begin. For example if first Memo has name "memo_1", set second Memo property "flowTo" to "memo_1". Property "StretchMode" for the first Memo should be set to "DontStretch" and for the second one to "ActualHeight". That's it.



Now lets render this template and see how it looks:

Capybara

The Capybara is a large, semi-aquatic rodent that is found inhabiting the water-logged regions of Central and South America. Closely related to other South American rodents such as Chinchillas and Guinea Pigs, the Capybara is the largest rodent in the world weighing up to 75kg and measuring nearly 1.4 meters long. Despite their enormous size though, these mammals have adapted well to life in the water and have a number of distinctive characteristics that aid their amphibious lifestyle, including the webbed skin between their toes which is particularly helpful when swimming. Interestingly enough, the common name of the Capybara is thought to mean "Master of the Grasses", whilst it's scientific name comes from the Greek word for water hog.



Abyssinian

The Abyssinian Cat is thought to be one of the oldest breeds of domestic Cat in the world, as the first domestication of the Abyssinian Cat occurred in Ancient Egyptian times. It is thought that Abyssinian Cats were bought and sold on the banks of the River Nile by traders, where the African Wild Cats (the ancestors of all domestic Cats) lived in their native habitats. Abyssinian Cats are most easily identified by their "ticked" fur which gives their coat a mottled appearance.



Adelie Penguin

The Adelie Penguin is the smallest and most widely distributed species of Penguin in the Southern Ocean and is one of only two species of Penguin found on the Antarctic mainland (the other being the much larger Emperor Penguin). The Adelie Penguin was named in 1840 by French explorer Jules Dumont d'Urville who named the Penguin for his wife, Adelie. Adelie Penguins have adapted well to life in the Antarctic as these migratory Birds winter in the northern pack-ice before returning south to the Antarctic coast for the warmer summer months.



Baboon

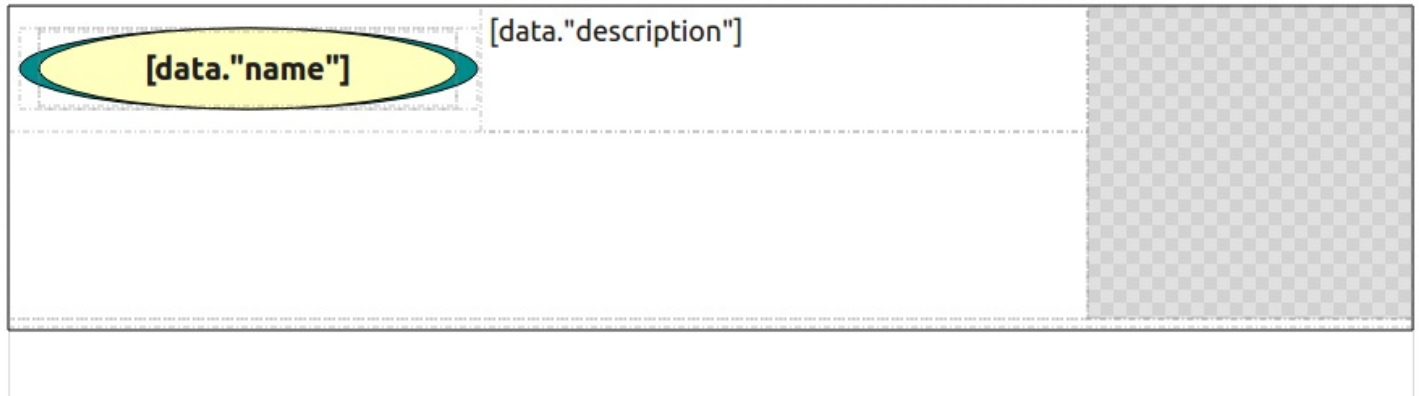
The Baboon is a medium to large sized species of Old World Monkey that is found in a variety of different habitats throughout Africa and in parts of Arabia. There are five different species of Baboon which are the Olive Baboon, the Guinea Baboon, the Chacma Baboon, the Yellow Baboon and the Hamadryas Baboon which differs most from the others wide it's bright red face and cliff-dwelling lifestyle (the other four species are collectively known as Savanna Baboons). However, there is some debate over the classification of the different species due to the fact that some have been known to interbreed, indicating that they could be sub-species instead. Baboons are incredibly sociable and intelligent animals that are known to form close bonds with other members of the troop that often last for life. They are also incredibly adaptable animals but their population numbers are declining throughout their natural range primarily due to hunting and habitat loss.



For this example we have set **textFlag** property to **AlignJustify**. It is not necessary to set text flags to all Memos, so set them only for the first one. Every subsequent Memo inherits this setting.

Complex wrapping

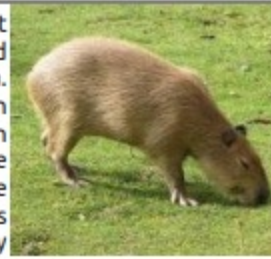
Making complex wrapping is not really much complicated. Take a look at the next example:



As you might notice there are 3 Memo objects that used for fit text: first in the middle top contains text "[data.\"description\"]"; second one is laying under Title memo and cover right and central part; third one is located on the bottom under all other objects. Its height set to minimal, since for some short texts it will not be used. So we do not need space wastage. Every next Memo joined to previous one by setting property "**flowTo**" and property **stretchMode** of the last item is set to "ActualHeight". First two items do no need to stretch, so theirs property is set to "DontStretch". "F5" to render and voilà:

Capybara

The Capybara is a large, semi-aquatic rodent that is found inhabiting the water-logged regions of Central and South America. Closely related to other South American rodents such as Chinchillas and Guinea Pigs, the Capybara is the largest rodent in the world weighing up to 75kg and measuring nearly 1.4 meters long. Despite their enormous size though, these mammals have adapted well to life in the water and have a number of distinctive characteristics that aid their amphibious lifestyle, including the webbed skin between their toes which is particularly helpful when swimming. Interestingly enough, the common name of the Capybara is thought to mean "Master of the Grasses", whilst it's scientific name comes from the Greek word for water hog.



Abyssinian

The Abyssinian Cat is thought to be one of the oldest breeds of domestic Cat in the world, as the first domestication of the Abyssinian Cat occurred in Ancient Egyptian times. It is thought that Abyssinian Cats were bought and sold on the banks of the River Nile by traders, where the African Wild Cats (the ancestors of all domestic Cats) lived in their native habitats. Abyssinian Cats are most easily identified by their "ticked" fur which gives their coat a mottled appearance.



Adelie Penguin

The Adelie Penguin is the smallest and most widely distributed species of Penguin in the Southern Ocean and is one of only two species of Penguin found on the Antarctic mainland (the other being the much larger Emperor Penguin). The Adelie Penguin was named in 1840 by French explorer Jules Dumont d'Urville who named the Penguin for his wife, Adelie. Adelie Penguins have adapted well to life in the Antarctic as these migratory Birds winter in the northern pack-ice before returning south to the Antarctic coast for the warmer summer months.



Baboon

The Baboon is a medium to large sized species of Old World Monkey that is found in a variety of different habitats throughout Africa and in parts of Arabia. There are five different species of Baboon which are the Olive Baboon, the Guinea Baboon, the Chacma Baboon, the Yellow Baboon and the Hamadryas Baboon which differs most from the others with its bright red face and cliff-dwelling lifestyle (the other four species are collectively known as Savanna Baboons). However, there is some debate over the classification of the different species due to the fact that some have been known to interbreed, indicating that they could be sub-species instead. Baboons are incredibly sociable and intelligent animals that are known to form close bonds with other members of the troop that often last for life. They are also incredibly adaptable animals but their population numbers are declining throughout their natural range primarily due to hunting and habitat loss.



Camel

The Camel (also known as the Dromedary Camel, the Arabian Camel and the One-Humped Camel) is a large hoofed animal that is most commonly found in the hot deserts of Northern Africa and the Middle East. Thought to have been first domesticated by native people more than 5,000 years ago, these hardy animals have proved vital to the survival of humans in these areas as they are not just used for transporting both people and goods, but also provide a good source of milk, meat and wool. The Camel is one the most unique mammals on the planet and has adapted perfectly to life in the desert where food and water can often be scarce, and the temperature changes rapidly from the scorching-hot days to the cooler nights. However, although they would have once been found freely roaming the Arabian deserts, they are today extinct from the wild but the domestic population is widespread and numerous.



You do not have to care about object insertion order, you can add Memo objects to

report page in arbitrary order. Once you set correct "flowTo" name all is done. CuteReport is smart enough to understand what do you want and it hides all routine work from your sight. Enjoy!

Custom application example

Now let's do some coding. To add CuteReport library to your application you can do something like this:

```
#include "reportcore.h"

/* create reportcore instance */
CuteReport::ReportCore * reportCore = new CuteReport::ReportCore(0 ,0,
false);

/* create report preview widget */
CuteReport::ReportPreview * preview = new
CuteReport::ReportPreview(parentWidget);

/* assign report core to our preview */
preview->setReportCore(reportCore);

/* loading report template from file and creating of report object */
CuteReport::ReportInterface * reportObject = reportCore-
>loadReport("git:report.qtrp");

/* connect created report object to the preview */
preview->connectReport(reportObject);

/* show preview widget */
preview->show();

/* start report rendering */
preview->run();
```

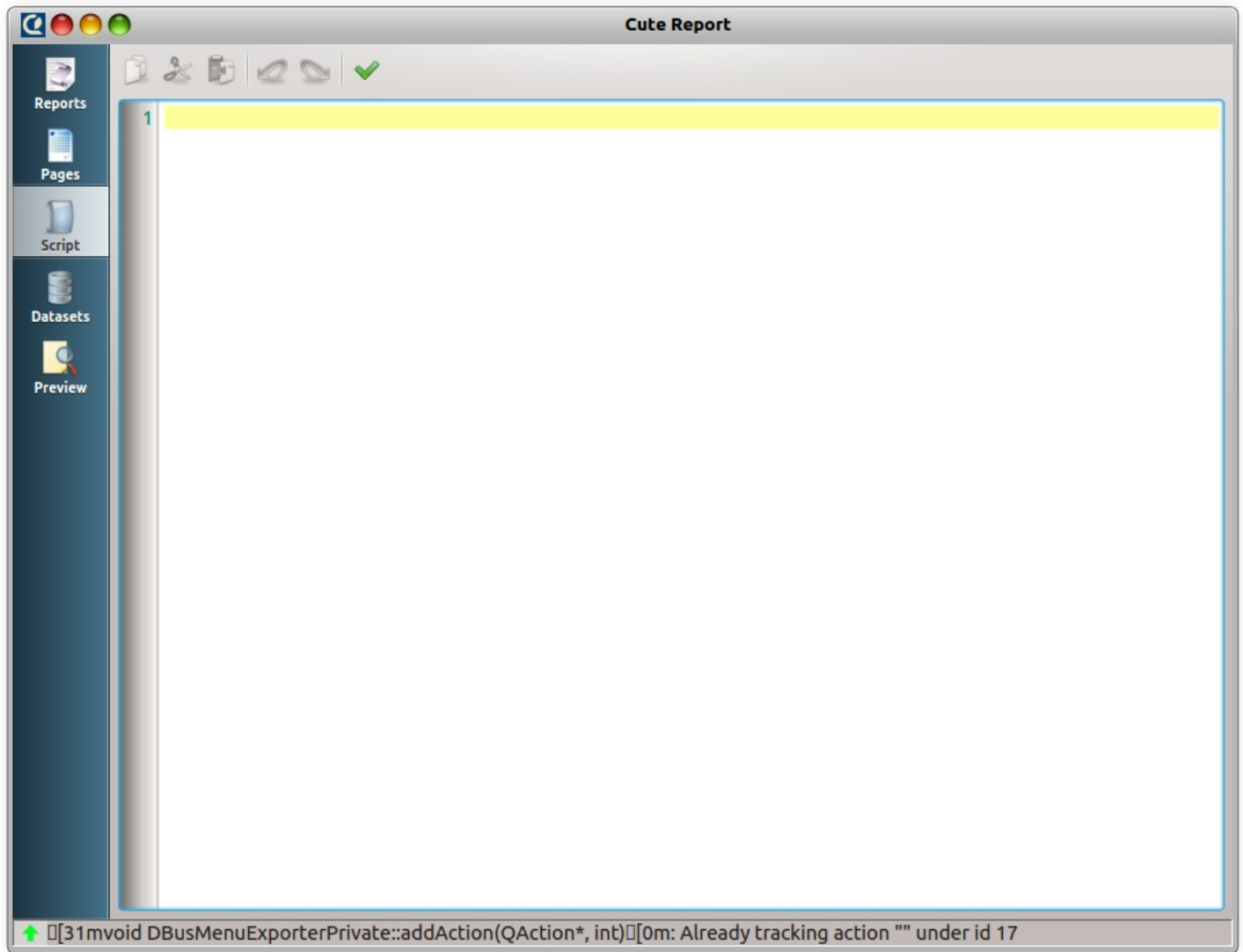
Usually you need only one reportCore instance in your application. Any number of Preview widget can be assigned to the core.

Script

In this chapter we will learn how to work with CuteReport script. Scripting feature brings an extremely high level of flexibility. Using script user can controll almost every rendering step and design really complex reports. There are main script in a report that control everything from report starting till it's rendered. Some items like Memo or Barcode are supporting script in their text properties. Usualy scripting expression must be framed by [], so the scripting engine will know this is script and not regular text. But to some fields where only script is allowed expression can be written without []. Some items can reimplements [] to use something else and provide additional field to define script expression borders, like "expDelimiter" in a Memo object.

Script Editor

CuteDesigner has a module named ScriptEditor to work with script. You can find ScriptEditor by pressing "Script" tab on the right panel of the Cutereport Designer



List of the shortcut keys which can be used in the ScriptEditor:

Key	Description
Cursor arrows	move cursor position
PageUp, PageDown	go to previous/next page
Ctrl+PageUp	go to beginning of the text
Ctrl+PageDown	go to end of the text
Home	go to beginning of the line

End	go to end of the line
Enter	go to next line
Delete	delete symbol at cursor position; delete selected text
Backspace	delete symbol to the left of the cursor
Ctrl+A	select whole text

[TODO]

CuteReport uses standard JavaScript syntax, so please read JavaScript documentation if you are not familiar with this language.

Script Objects

All report objects is accessible from a script by theirs name. For example if you have Memo object named memo_1 and want to set its color, you can do it by such way:

```
memo_1.backgroundBrush = new QBrush(new QColor("#665544"));  
memo_1.color = new QColor(Qt.red);
```

All object prpperties thar you can see in a PropertyEditor, can be managed through the script. Some objects support several types of property like "stretchMode" in Memo object. There are 2 types: enum and string. Use whatever you want like there:

```
memo_1.stretchMode = "DownStretch";  
memo_2.stretchMode = Memo.ActualHeight;
```

Script variables

There is standard rule - any variables such as global report's variables, engine's variables or any local script variable are script objects. The difference is only naming. Renderer uses prefix "_" for engine's variables and "__" for global variables. So they all can be accessible using this logic. But some of them can also have other more specific name, like global variables do.

Local script variables

Variables can be declared and used locally within a script. Once declared a script variable can have value assigned to it. Here you can see an example:

```
var myVar = "Hello, World!";
```

After you have variable created you can use it in any report objects, for example in Memo by writing "[myVar]" to the **"text"** property. For more detailed information about script variables read documentation about JavaScript language.

Global report's variables

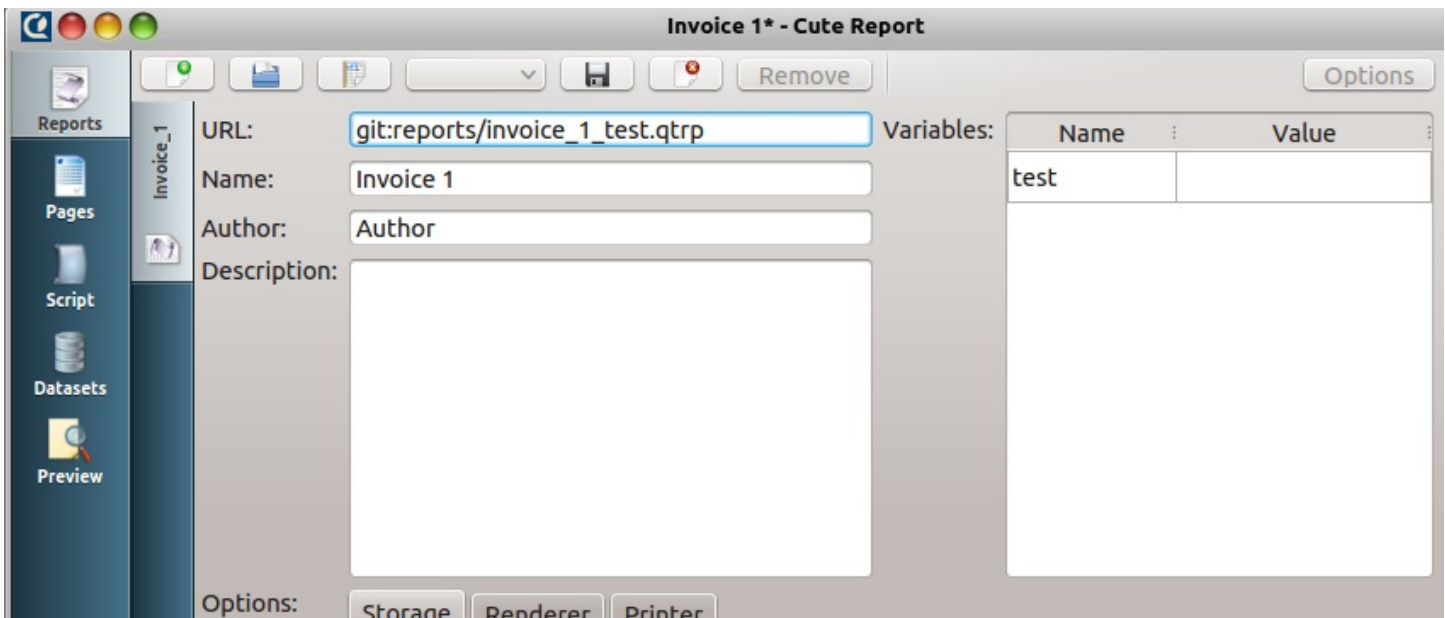
Any global variable that is defined in a report could be accessed from a script. The variable name should be written in such way: `${my_variable}` without spaces between rounding signs and within variable name. There are 2 recommended way to go if you want to give your variable some complex name like "my super duper variable". First way is to replace spaces with "_" sign. And second one is to remove spaces and use capital letter on the. Internally script engine use special named object to represent global variables. So using "_" in begin of your variable name is highly not recommended. On the other hand you can have local script variable and global report variable with the same name:

```
var myVar = 5;  
print(`${myVar}`);  
print(myVar);
```

Withing script variables "myVar" and "\${myVar}" are totally different objects. You can safely get/put values to/from global variable like you do with regular JavaScript variables:

```
print(${myNumber});  
${myNumber} = 10;
```

Once you mentioned your global variable in the script it will be automatically added to the report variable list if it's not still exists there. Let's try it. Open new empty report, go to the "Script" tab and enter `{ $test }`. Now switch to the "Reports" tab. You will see your variable in the list of variables and could test your script by setting any test value for the variable.



Renderer's variables

Renderer module has its own variables and the full list of variables depends on the renderer itself. There are lists of Standard renderer's variables:

- `_line` - current dataset line starting from 1
- `_page` - current page number starting from 1
- `_pages` - total pages (require report double pass)
- `_passes` - report pass number
- `_template_page` - current page of template: means number of page in designer
- `_template_pages` - total pages of template: means number of pages in designer (starting from 1)

Script Signals

When you write a script it means you write main function, which is processed on report generator start. In this function user can made some variable initialisation or some other preparation. You still might want more control over report processing. To make it possible almost everyone report object has signals and you can assign your custom slot to be processed on these signals. For example you can assign your custom filter to Detail band and hide some bands while pass another. Lets review some signals and latter will see how we can use them.

Common item Signals:

Signal Name	Description
printInit	emited when all items are preparing to be printed
printReset	emited when all items are cleaned up after printing
printBefore	emited before item printing. All property changes affects original template item
printDataBefore	emited when initial data for printed item is prepared. All property changes affects only current printed item and will be reset
printDataAfter	emited after all item's data is processed, but before actual printing
printAfter	emited after item is printed on a page

Also every item can have its own signals. You can see full signal list common with signal description in a PropertyEditor.

Renderer Signals:

Signal Name	Description
reportStart()	emited after report started
bandBefore(CuteReport::BandInterface * band)	emited before band rendering
bandAfter(CuteReport::BandInterface * band)	emited after band is rendered
bandGeometryAfter(CuteReport::BandInterface * band)	emited when band's geometry is managed
itemBefore(CuteReport::BaseItemInterface * item)	emited before item rendering
itemAfter(CuteReport::BaseItemInterface * item)	emited after item is rendered
itemGeometryAfter(CuteReport::BaseItemInterface * item)	emited after item's geometry managed

<code>datasetBefore(CuteReport::DatasetInterface * dataset)</code>	emited before dataset processing
<code>datasetAfter(CuteReport::DatasetInterface * dataset)</code>	emited after dataset processed
<code>datasetIteration(CuteReport::DatasetInterface * dataset)</code>	emited on every dataset iteration
<code>pageBefore(CuteReport::PageInterface * page)</code>	emited before template page processing
<code>pageAfter(CuteReport::PageInterface * page)</code>	emited after template page processing
<code>formBefore(CuteReport::FormInterface * dataset)</code>	emited before form is shown
<code>formAfter(CuteReport::FormInterface * dataset)</code>	emited after form is closed
<code>reportDone()</code>	emited after report rendering is done

[TODO]