

# **Руководство пользователя CuteReport**

версия 1.0

# Содержание

<b>Вступление.....</b>	<b>3</b>
Лицензирование.....	5
<b>Дизайнер.....</b>	<b>6</b>
Редактор свойств отчета.....	7
Редактор страниц.....	10
Редактор скрипта.....	13
Редактор данных.....	15
Предпросмотр.....	17
Настройки дизайнера.....	19
<b>Создание шаблона отчета.....</b>	<b>20</b>
Объекты отчета.....	21
Пример отчета «Привет, мир!».....	22
Элемент Метод.....	23
Вращение.....	23
HTML Теги.....	24
Выражения.....	24
Перетекание текста.....	25
Помощник Метод.....	26
Контейнеры.....	28
Хранилища.....	29
Хранилище «Файловая система» (Standard::Filesystem).....	29
Хранилище «GIT» (Standard::GIT).....	29
Хранилище «Resource» (Standard::Resource).....	30
Хранилище «SQL» (Standard::SQL).....	30
Наборы данных.....	31
Пример отчета «Список клиентов».....	32
Элемент Image.....	35
Отчет с иллюстрациями.....	36
Печать многострочного текста.....	38
Обтекание объектов текстом.....	41
Сложное обтекание.....	41
Печать ярлыков.....	43
Многостраничный отчет.....	45
<b>Скриптовый движок.....</b>	<b>46</b>
Объекты в скрипте.....	47
Переменные в скрипте.....	48
Локальные переменные.....	48
Глобальные переменные.....	48
Переменные рендерера.....	49
Сигналы в скрипте.....	50
<b>Использование в программах.....</b>	<b>52</b>
Настройка проекта.....	53
Встроенная библиотека.....	53
Самостоятельный фреймворк.....	53
Простой пример.....	54
Пример пользовательской программы.....	56
Наборы данных.....	57
Набор данных Модель.....	58

# Вступление

**CuteReport** – это программное решение для построения отчетов основанное на библиотеке Qt, которое может быть легко использовано с любыми Qt приложениями. В основе CuteReport состоит из двух частей: основная библиотека и Дизайнер шаблонов. Оба полностью модульные и из функциональность может быть легко расширена с помощью дополнительных модулей. CuteReport полностью абстрагирован от используемых данных и может использовать как хранилище: файловую систему, базу данных, систему контроля версий и пр. Цель проекта — предоставить мощное, но в то же время простое даже для неопытного пользователя или дизайнера средство.

## Ключевые особенности:

- Большое количество источников данных: SQL база данных, текст, файловая система, внешняя модель данных (QAbstractItemModel);
- Различные типы хранилищ для сохранения шаблонов отчетов и объектов, таких как иллюстрации, базы данных, шаблоны: файловая система, GIT, SQL базы данных, встраиваемые хранилища;
- Поддержка обычного текста и HTML;
- Разнообразие графических элементов для построения отлично выглядящих отчетов: Текст (Memo), Иллюстрация (Image), Штрихкод (Barcode), Дуга (Arc), Диаграмма (Chart), Хорда (Chord), Эллипс (Ellipse), Линия (Line), Сектор (Pie), Прямоугольник (Rectangle);
- Источники иллюстраций: статический, датасет, хранилище
- Неограниченное количество Detail контейнеров в одном отчете;
- контейнеры отчета: Заголовок (Title) и Заключение (Summary);
- контейнеры страницы: Шапка (Header) и Подвал (Footer);
- группировки объектов;
- агрегатные функции: счетчик (count), минимум (min), максимум (max), среднее (avg), сумма (sum);
- передаваемые в отчет извне параметры;
- всеобъемлющий полнофункциональный скриптовый движок для управления любыми аспектами построения отчета;
- поддерживаемые единицы измерения; миллиметры, дюймы, пиксели;
- отдельный WYSIWYG дизайнер с возможностью расширения любой функциональности через модули;
- несколько предустановленных модулей для дизайнера: Редактор свойств отчета, Редактор страниц, Редактор скриптов, Редактор данных, Предпросмотр;
- мультиплатформенность;



## Лицензирование

CuteReport распространяется в 2 версиях:

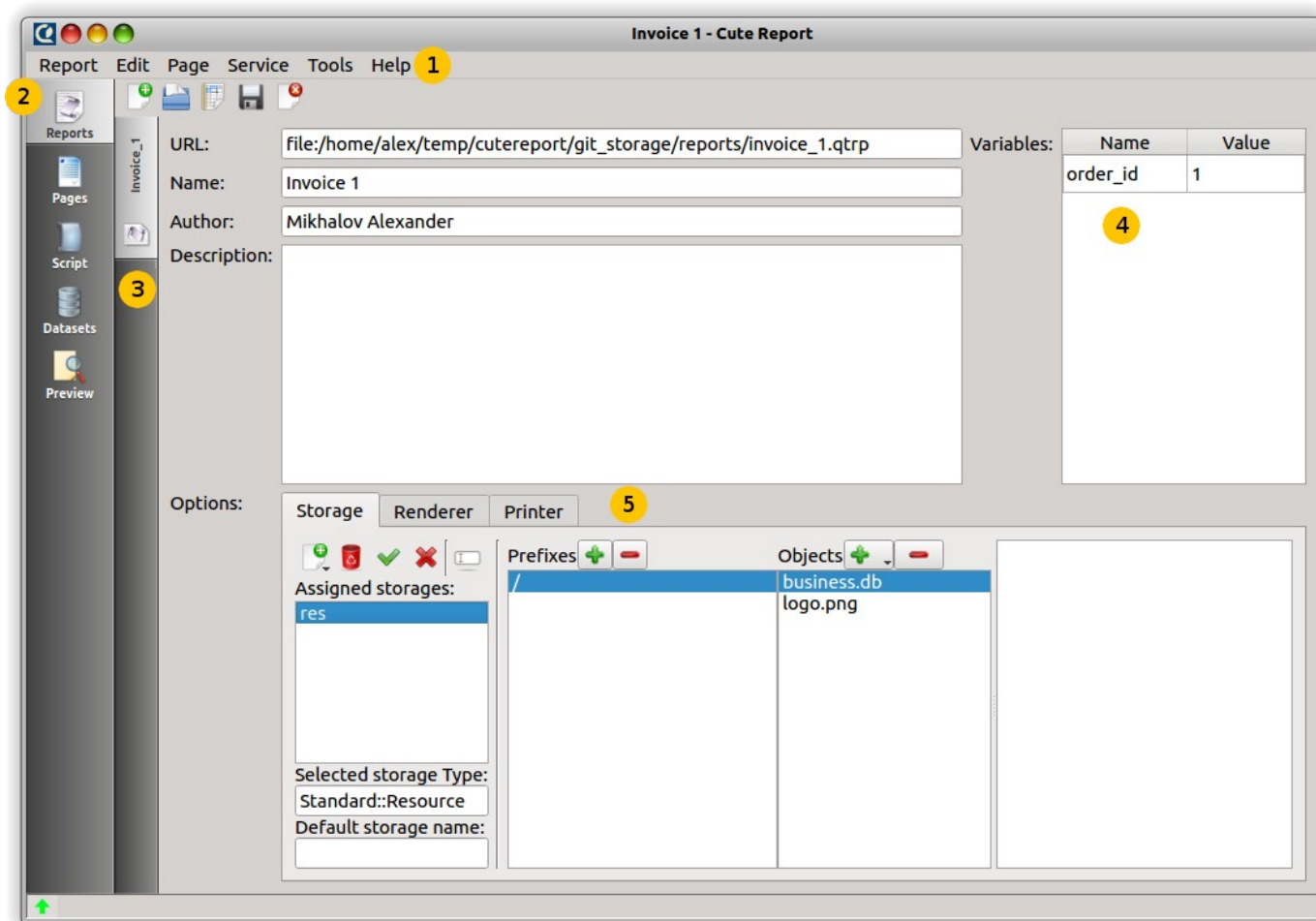
- *Общественная версия* под GNU/GPLv3 лицензией чтобы помочь разработчикам приложений с открытым исходным кодом добавить функционал для построения отчетов в их открытые проекты. Базовая библиотека поставляется под LGPLv2 и может быть динамически слинкована с проприетарными программами. Дизайнер CuteReport доступен под GPLv3 лицензией и таким образом не может быть встроен в проприетарные продукты. Ознакомьтесь с описанием GPL/LGPLv2 лицензии чтобы иметь больше информации;
- *Коммерческая версия* поставляется для обеспечения высокого уровня поддержки, приоритета в исправлении ошибок и реализации новых функций. Она также предоставляет несколько дополнительных модулей, которые недоступны в общественной версии. В этой документации такой функционал помечен текстом «только для коммерческой версии». Чтобы ознакомиться с вариантами коммерческих лицензий посетите официальный веб сайт проекта <http://cute-report.com/en/article/licenses>;

# Дизайнер

CuteReport поставляется с самостоятельным дизайнером, который помогает оперировать шаблонами отчетов. Дизайнер имеет лишь малый набор базовых функций и предоставляет программный интерфейс для поддержки модулей. Модули дизайнера используются для обеспечения и расширения любого функционала дизайнера. Модуль может предоставлять или не предоставлять графический интерфейс пользователя. Некоторые из базовых модулей: Редактор свойств отчета, Редактор страниц, Редактор скриптов, Редактор данных, Предпросмотр. Каждый модуль собственный функционал и может быть зависим от других модулей. Также каждый модуль может быть заменен другим с расширенным функционалом.

Давайте рассмотрим некоторые из таких модулей.

## Редактор свойств отчета



### Ключевые особенности:

1. главное меню
2. панель модулей
3. панель открытых отчетов
4. переменные отчета
5. встроенные объекты отчета

Редактор свойств отчета это первый модуль на панели Дизайнера. Он отвечает за управление объектами отчетов и обеспечивает следующий функционал: загрузка, сохранение, создание, удаление и пр. Эти операции представлены элементами управления в окне Редактора свойств а также в главном меню приложения. Этот модуль также поддерживает некоторое количество одновременно открытых отчетов и позволяет переключаться между ними. Также Редактор свойств управляет встроенными в отчет объектами, такими как хранилища, рендереры, принтеры. В случае отсутствия встроенных объектов какого либо типа, будут использованы глобальные объекты данного типа настроенные в CuteReport. Если вам необходимы специальные настройки для хранилища, рендерера или принтера, вам необходимо добавить объект требуемого типа в отчет и установить его настройки по потребностям. Редактор свойств также содержит таблицу глобальных переменных отчета и их значения. Эти значения используются в процессе рендеринга отчета и могут быть установлены вручную в этой

таблице или из внешней программы. Мы рассмотрим как оперировать отчетами напрямую из вашей программы позднее. Сейчас сосредоточим внимание на использовании Дизайнера.

Все открытые отчеты показаны в левой панели(#3). Используйте ее для переключения между отчетами. Главный вид модуля содержит следующие поля: URL, имя, автор, описание. URL устанавливается автоматически и отображает путь по которому хранится отчет. Поле “Имя” содержит имя отчета. Поле “Автор” содержит имя автора отчета. И поле «Описание», увы вы можете догадаться, содержит описание отчета. Вы можете использовать эти поля по вашему усмотрению, они не задействованы в генерации отчета и служат исключительно для удобства. Слева (#4), как мы упоминали ранее, находится таблица переменных отчета. Переменные автоматически появляются здесь как только они объявлены где-либо в объектах отчета. Для проверки такого поведения переключитесь на вкладку «Скрипт» и напечатайте в редакторе текст “`${test}`” без кавычек. Мы рассмотрим модуль «Скрипт» позднее. А пока мы можем переключиться назад в Редактор свойств и увидеть как переменная «test» появилась в таблице переменных. Теперь вы можете назначить любое значение для этой переменной. Вы также можете назначать значения переменных напрямую из вашего приложения. Обычно вам нужно присваивать переменным временное значение для тестирования отчета отдельно от вашей программы.

Далее следуют «Настройки» (Options). Этот фрагмент отображает все встроенные объекты типа хранилище, рендерер, принтер вместе с их параметрами. Возможно вы захотите использовать встроенные объекты в случае если ваш шаблон отчета используется на нескольких компьютерах с различными настройками доступа или для предотвращения вмешательства пользователя. Например, вы создаете шаблон отчета для специального принтера и не хотите показывать пользователю диалог печати и предоставлять возможность смены настроек печати. Используйте встроенный объект принтера со всеми предустановленными параметрами. Также вы можете пожелать внедрить логотип компании непосредственно в шаблон отчета вместо отдельного файла на диске загружаемого всякий раз при рендеринге отчета. Вы можете добавить встроенное хранилище «Ресурсы» (Resource) и сохранить в нем ваш логотип. Это поможет распространять ваш шаблон вместе с вашими ресурсами в одном файле шаблона.

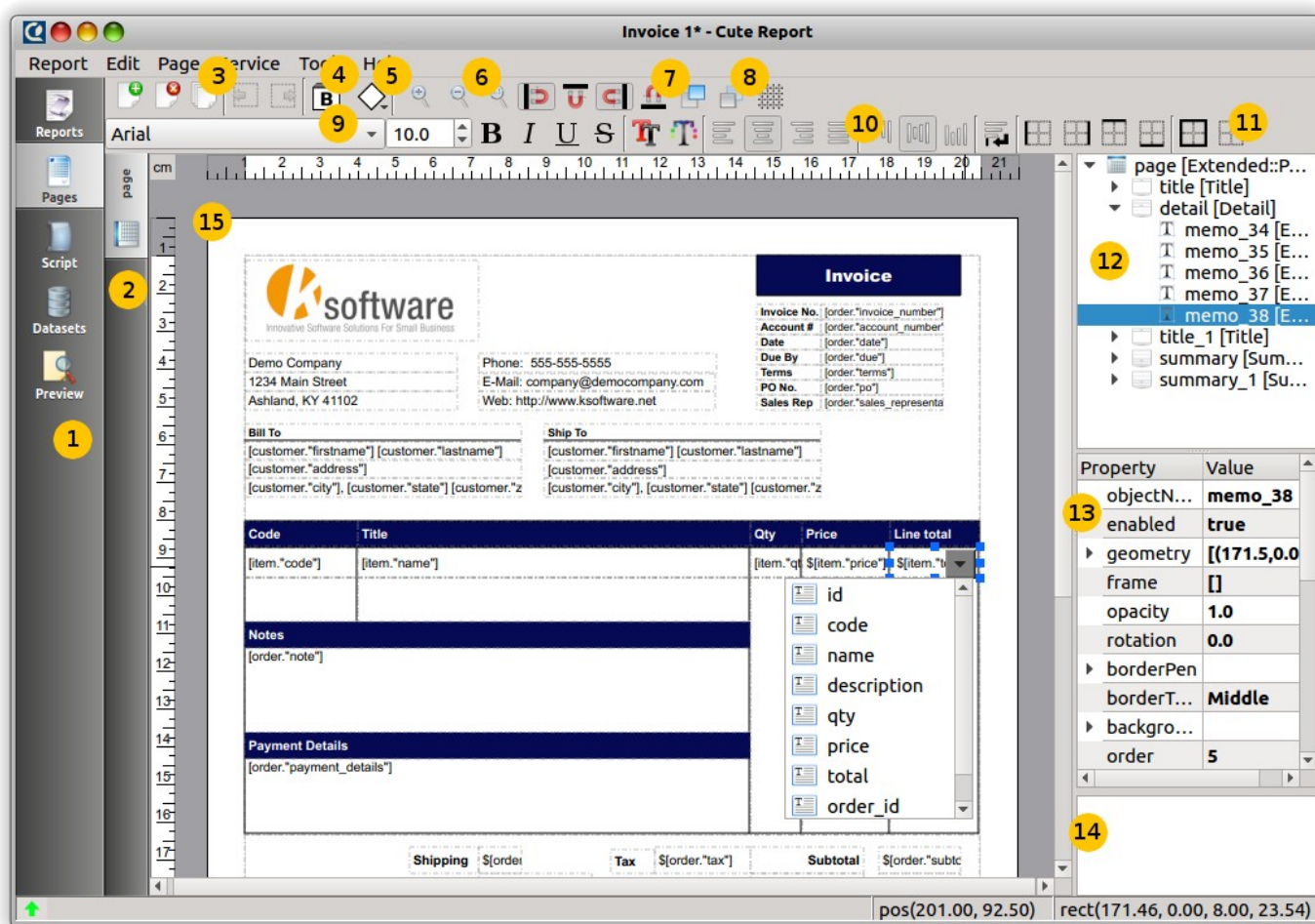
Для каждого типа встроенных объектов доступно несколько кнопок: добавить объект, удалить объект, установить объект как объект по умолчанию, убрать установки объекта по умолчанию, переименовать выбранный объект. Как можно догадаться «добавить» и «удалить» ответственны за добавление и удаление объекта в/из шаблона отчета. Объект по умолчанию для каждого типа означает что он будет использован если вы явно не указываете имя объекта в вашем отчете. Например, вы можете загружать объект в вашем отчете используя полную ссылку «`file:/home/user/images/logo.png`» или не указывая имя хранилища «`/home/user/images/logo.png`» если вы установили хранилище по умолчанию. Имейте ввиду что вы получите сообщение об ошибке если вы не обращаетесь к объекту по сокращенной ссылке и не установили объект по умолчанию.

Стоит упомянуть что тип объекта определяется именем модуля который породил объект и представляется в виде «ИмяНабора:ИмяМодуля». Стандартный набор имеет имя “Standard” (Стандартный). Коммерческая версия имеет имя набора “Extended”



(Расширенный). Вы можете также иметь сторонние наборы компонентов с их собственным именем набора. Например, стандартное SQL хранилище имеет полное имя «Standard::SQL». В то же время расширенное SQL хранилище будет иметь имя «Extended::SQL».

## Редактор страниц



### Ключевые особенности:

1. панель модулей с активированным модулем «Редактор страниц»
2. панель страниц
3. панель инструментов страницы
4. выпадающий список контейнеров
5. выпадающий список элементов
6. кнопки масштабирования
7. включение/выключение магнитов
8. поднять/опустить элемент
9. редактор шрифта
10. редактор выравнивания
11. редактор границы
12. инспектор объектов
13. редактор свойств
14. описание свойства
15. рабочее пространство

Редактор страниц ответственен за создание страниц шаблона отчетов. Он предоставляет инструменты для управления контейнерами и элементами страницы. Для активирования Редактора страниц нажмите вкладку «Pages» на панели модулей (#1).

Панель страниц (**#2**) отображает все страницы в отчете. Вы можете использовать ее для переключения между страницами (левый щелчок мыши) или переименования текущей выбранной страницы (двойной левый щелчок мыши). На панели инструментов (**#3**) находятся некоторые инструменты для выполнения основных действий над страницей: создать новую страницу, удалить страницу, клонировать текущую страницу. Далее следуют 2 кнопки с выпадающими списками. Первая — для выбора контейнера, вторая — для выбора элемента. После этого вы можете видеть несколько кнопок для управления масштабированием и следующие кнопки для включения/отключения магнитов. Если магниты включены, то курсор мыши будет залипать к другим объектам с координатами ближайшими к координатам курсора. Коэффициент расстояния залипания может быть изменен установлением нового значения в свойство страницы «**magnetRate**». Редактор страниц имеет также некоторые другие инструменты для изменения свойств элементов и контейнеров, такие как: редактор шрифта, редактор выравнивания, редактор границ.

Справа на виджете страницы вы можете увидеть Инспектор объектов. Все объекты страницы представлены в нем в виде дерева. Вы можете переключаться между объектами используя Инспектор объектов или кликая мышкой на объекте в рабочем пространстве Редактора страниц(**#12**). Редактор свойств — следующий фрагмент для рассмотрения. В нем отображены все редактируемые свойства активного объекта. Кликая на имени свойства вы можете видеть его короткое описание(**#11**). Рабочее пространство (**#12**) отображает полную страницу шаблона отчета. Вы можете добавлять новые контейнеры и элементы на страницу используя перетаскивание выбранного в выпадающем списке (**#4, #5**) контейнера или элемента на страницу в рабочем пространстве. Практически все элементы могут быть размещены только в пределах контейнера, а контейнеры могут быть расположены только непосредственно на странице. Используйте клавишу «Delete» на клавиатуре для удаления контейнера или элемента вместе со всеми объектами на них расположенными.

Также возможно выделение нескольких объектов для выполнения групповых операций над ними. Для выбора нескольких объектов кликните левой кнопкой мыши на первом объекте и затем удерживая клавишу CTRL кликайте на всех последующих объектах. Вы можете кликать непосредственно на объекте на странице или на имени объекта в Инспекторе объектов. Для выполнения последующих групповых операций используйте редакторы на панели инструментов. Изменения будут применены ко всем выбранным элементам если они допустимы.

В строке состояния приложения вы можете видеть текущие координаты курсора мыши и геометрию текущего выбранного объекта.

Некоторые элементы могут иметь помощников для графического представления свойств элемента. Иногда более удобно использовать помощников для изменения свойств объекта. Для открытия помощника сделайте двойной щелчок мышью на объекте. Например, если вы сделаете двойной щелчок на объекте Мемо (Текст), то вы сможете использовать редактор текста для введения текста или выражения. Некоторые другие редакторы также могут быть доступны в помощнике если соответствующие модули установлены. Коммерческая версия имеет несколько дополнительных модулей, отсутствующих в Общественной версии.

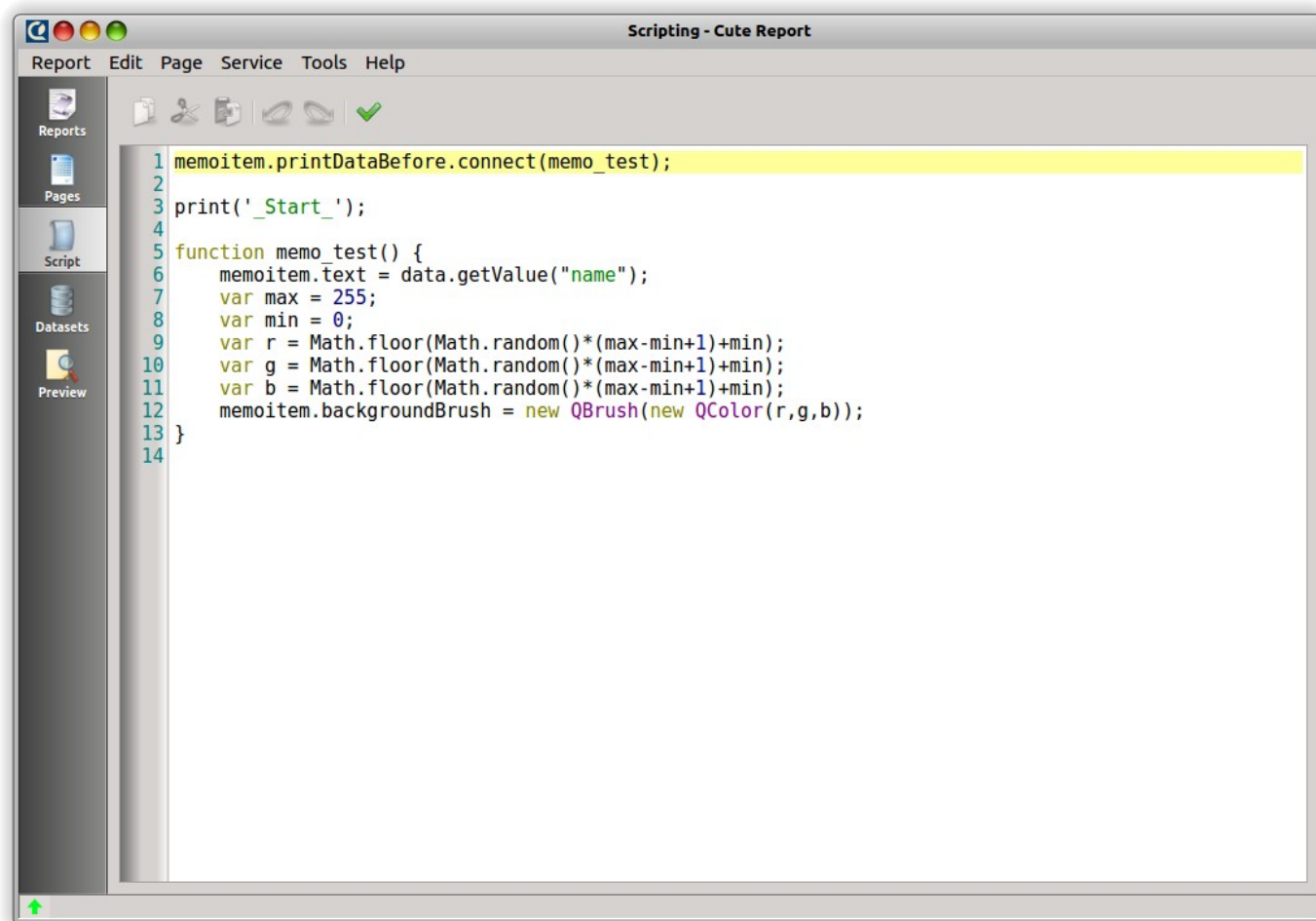
Клавиши управления

Комбинация	Действие	Описание
Ctrl+N	Report → New Report	Создать новый шаблон отчета
Ctrl+O	Report → Open Report	Открыть шаблон отчета
Ctrl+S	Report → Save Report	Сохранить текущий шаблон отчета
	Report → Save Report As...	Сохранить текущий шаблон отчета с новым именем
Ctrl+W	Report → Close Report	Закрыть текущий шаблон отчета
Del		Удалить текущий объект

Клавиши управления мыши

Операция	Описание
Левая клавиша	Выбрать объект; вставить объект; двигать или изменить размер выбранного объекта
Правая клавиша	Выбрать объект; вставить объект; двигать или изменить размер выбранного объекта с перемещением на нового родителя в позиции мыши
Левый двойной клик	Открыть помощник объекта
Колесо мыши	Листать страницу
Ctrl+левая клавиша	Добавить/удалить объект в\из группы

## Редактор скрипта



Вы можете переключиться на Редактор Скрипта нажатием вкладки «Script» в панели модулей. Редактор Скрипта — это достаточно простой модуль и содержит текстовый редактор подсветкой синтаксиса и кнопкой «Validate» (Проверка корректности). Проверка происходит только на синтаксическом уровне и не запускает ваш скрипт. Таким образом даже если ваш скрипт прошел проверку, он все еще может содержать ошибки времени выполнения. Обычно вы можете видеть ошибки нажатием зеленой клавиши в строке состояния в окне дизайнера. Если отчет содержит ошибки в скрипте, то кнопка становится красной. Редактор скрипта использует javascript как язык скриптования.

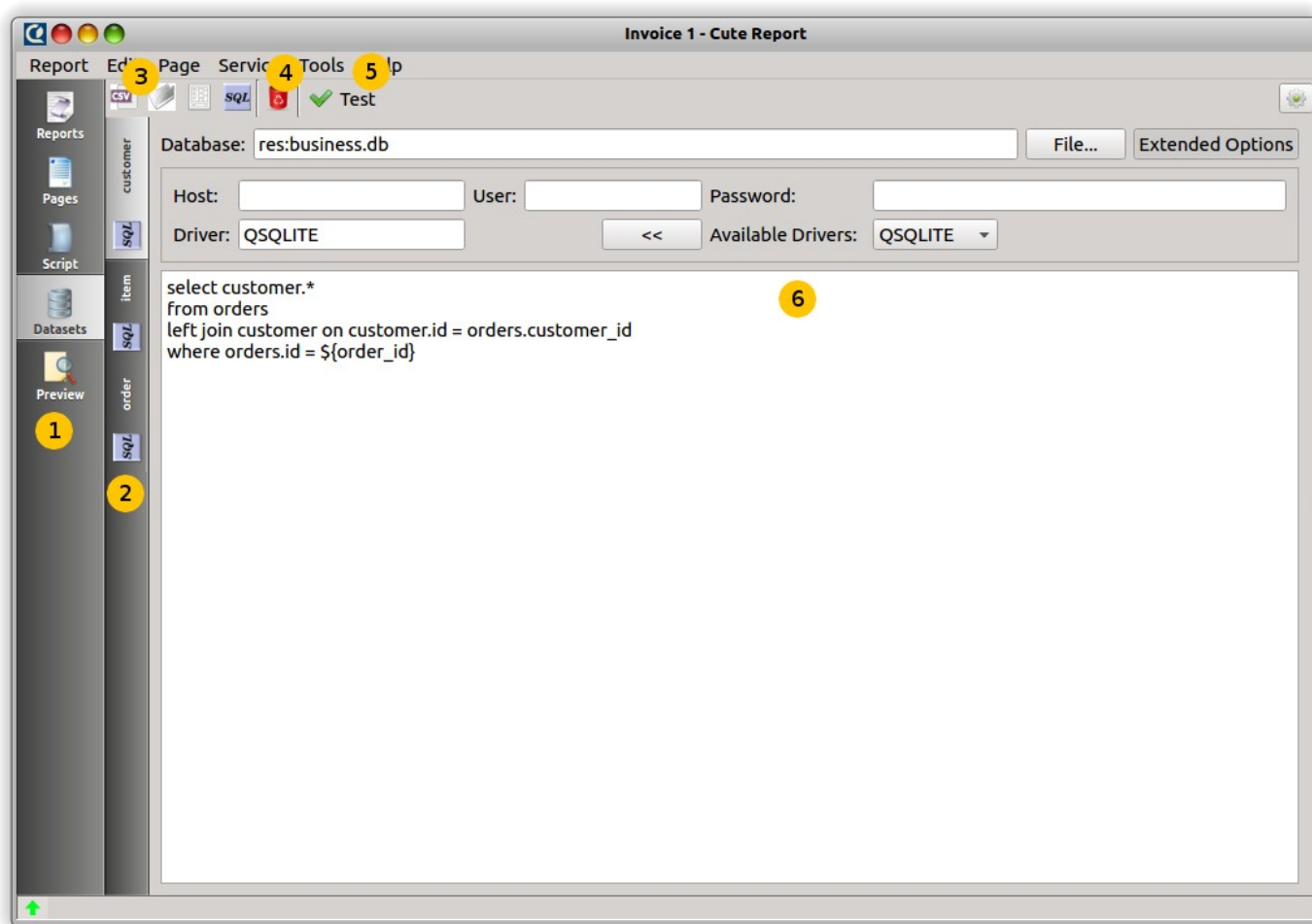
Список элементов управления и горячих клавиш:

Клавиша	Описание
Стрелки курсора	изменяет позицию курсора
PageUp, PageDown	Перейти к предыдущей/следующей странице
Ctrl+PageUp	Перейти в начало текста
Ctrl+PageDown	Перейти в конец текста
Home	Перейти в начало строки
End	Перейти в конец строки
Enter	Перейти на следующую строку

Клавиша	Описание
Delete	Удалить символ в позиции курсора; удалить выделенный текст
Backspace	Удалить символ слева от курсора
Ctrl+A	Выделить весь текст

Так как CuteReport использует стандартный синтаксис JavaScript, обратитесь к документации по JavaScript если вы испытываете трудности с этим языком.

# Редактор данных



## Ключевые особенности:

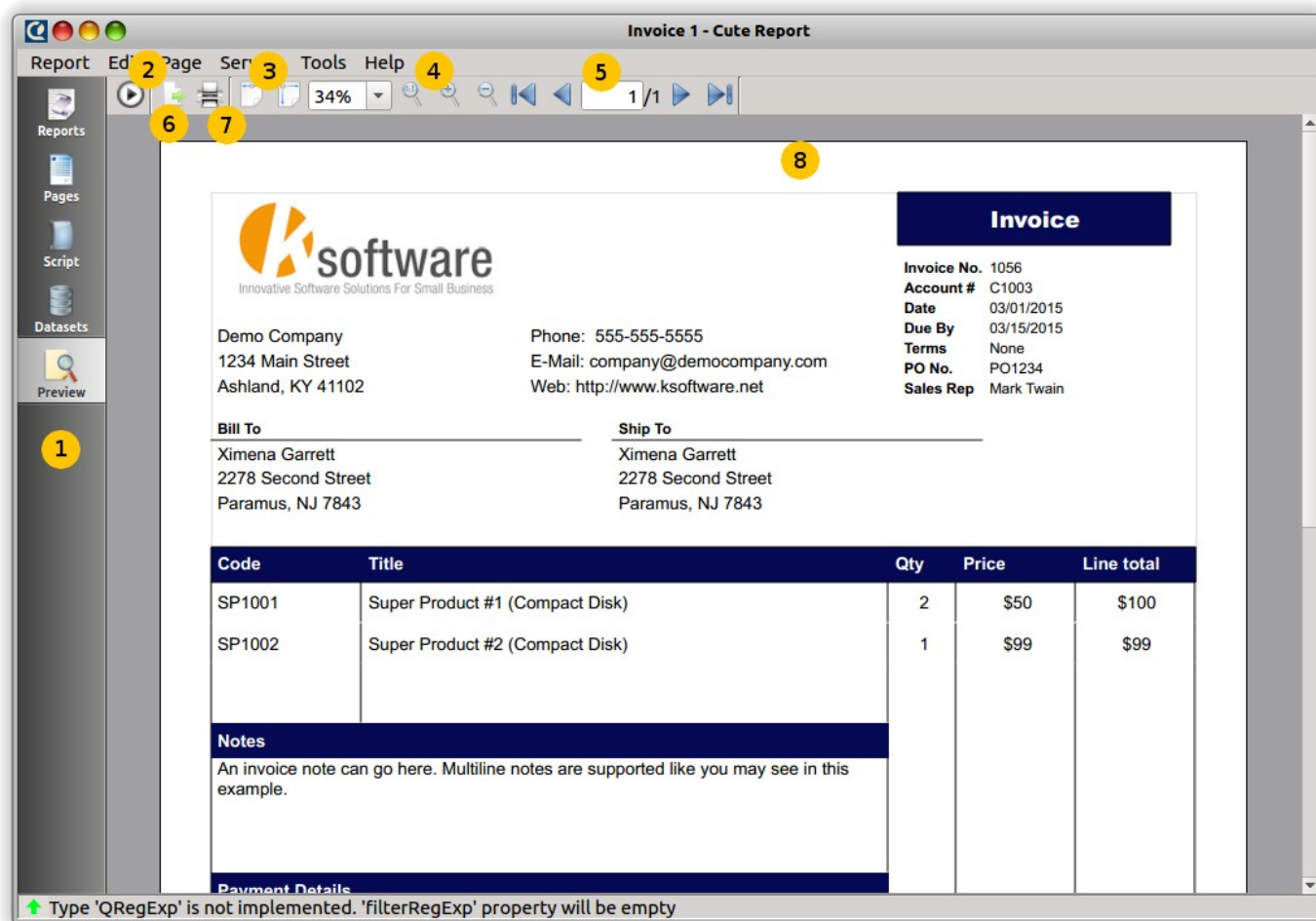
1. панель модулей с активированным Редактором данных
2. панель наборов данных
3. кнопка создания нового набора данных
4. удалить текущий набор данных
5. тест набора данных
6. помощник набора данных

Вы можете переключиться на Редактор данных нажатием вкладки «Dataset» на панели модулей (#1). Все созданные наборы данных текущего отчета показаны в панели наборов данных (#2). Используя эту панель вы можете переключаться между наборами данных (клик мышкой) или переименовывать текущий набор данных (двойной клик). Для создания нового набора, нажмите кнопку соответствующую необходимому типу данных #3. Основной комплект поставки CuteReport содержит 4 типа наборов данных: CSV, SQL, Файловая система и Модель. Модули предоставляющие данные типы данных имеют имена «Standard::CSV», «Standard::SQL», «Standard::Filesystem» и «Standard::Model» соответственно. Ознакомьтесь с описанием каждого типа данных ниже. Для удаления текущего набора нажмите клавишу с изображением мусорного бака #4. Когда вы установили все необходимые настройки набора данных, вы можете нажать кнопку «Test it» (#5) и убедиться все ли правильно. Все наборы данных имеют общий программный

интерфейс и предоставляют данные в виде таблицы. Каждый набор имеет собственных виджет для настроек (#6).



## Предпросмотр



### Ключевые особенности:

1. панель модулей с активированным предпросмотром
2. кнопка для старта/стопа процесса рендеринга
3. элементы подгонки масштабирования
4. элементы управления масштабированием
5. элементы навигации по страницам
6. клавиша экспорта
7. клавиша печати
8. сгенерированная страница

Задача модуля Предпросмотра — отображение сгенерированных страниц отчета. В окне модуля есть некоторые полезные элементы управления. Во-первых, это клавиша для запуска процесса генерации отчета по текущему шаблону (#2). Всякий раз когда вам необходимо построить отчет или обновить его результат вы можете воспользоваться этой клавишей. Как альтернативный вариант вы можете использовать главное меню -> Service -> Render или просто нажать F5 на клавиатуре. Если движку необходимо длительное время для построения отчета, вам будет показан диалог прогресса. Для прерывания процесса построения отчета нажмите ту же клавишу снова. Когда отчет построен, вы можете изменять масштаб используя кнопки подгона и масштабирования (#3, #4) или вы можете установить непосредственно необходимое процентное значение масштаба. Для

навигации по страницам используйте следующие клавиши: Первая страница, Предыдущая страница, Следующая страница, Последняя страница. Или установите номер страницы явно. В заключение, вы можете распечатать построенный отчет (**#7**) или экспортировать его в файл(**#6**).

**Настройки дизайнера**

# Создание шаблона отчета

В этой главе мы рассмотрим некоторые основные аспекты создания шаблона отчета. Мы взглянем ближе на некоторые важные элементы и их свойства и создадим несколько примеров отчетов. Убедитесь что вы установили CuteReport на ваш компьютер и попробуйте повторить эти примеры самостоятельно используя предустановленные наборы данных. Поскольку CuteReport активно развивается, некоторые части этого документа могут отличаться от вашей инсталляции CuteReport.

## Объекты отчета

Редактор страниц в Дизайнере CuteReport служит для представления отчета как набора схематических страниц. Все визуальные объекты отчета размещаются где-то на странице отчета и используются для отображения текстовой или графической информации. В общественную версию CuteReport включены все основные графические объекты. Некоторые расширенные объекты включены только в Профессиональную версию.

Давайте рассмотрим набор графических объектов.

### Контейнеры:




- **PageHeader:** контейнер, который располагается вверху страницы
- **PageFooter:** контейнер, который располагается внизу страницы
- **Detail:** контейнер соединяемый с набором данных и обрабатываемый при каждой итерации набора данных
- **DetailHeader:** контейнер, расположенный вверху группы detail контейнеров
- **DetailFooter:** контейнер, расположенный внизу группы detail контейнеров
- **Title:** контейнер, располагаемый перед всеми контейнерами одной группы
- **Summary:** контейнер, располагаемый после всех контейнеров одной группы
- **Overlay:** контейнер со свободным размещением, может быть помещен куда угодно на странице без выравнивания

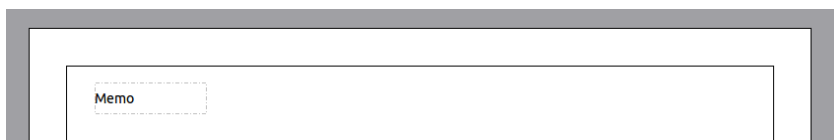
### Элементы:

- **Arc:** элемент для рисования дуги
- **Barcode:** элемент для рисования штрихкода
- **Chart:** элемент для рисования любых типов диаграмм
- **Chord:** элемент для рисования хорды
- **Ellipse:** элемент для рисования эллипса и круга
- **Image:** элемент для рисования динамических или статических иллюстраций в форматах PNG, JPG, BMP и других
- **Line:** элемент для рисования горизонтальных, вертикальных или диагональных линий
- **Memo:** элемент для отображения любой текстовой информации в формате обычного текста или HTML
- **Pie:** элемент для рисования сектора
- **Rectangle:** элемент для рисования прямоугольника

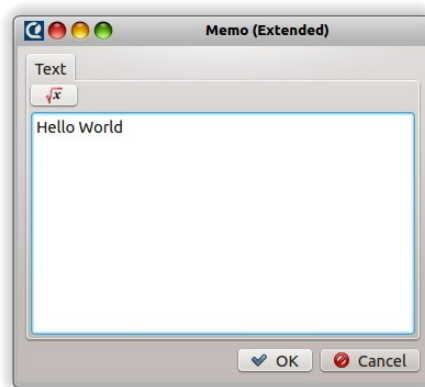
Основные объекты наиболее часто используемые это - «Detail» контейнер и «Мемо» элемент. Вы научитесь использовать их возможности подробнее в этой главе.

## Пример отчета «Привет, мир!»

Этот простой пример отчета содержит всего одну часть информации — текст «Привет, мир!». Откройте Дизайнер отчетов, создайте новый шаблон отчета используя Report → Create Report, перейдите в Редактор страниц используя левую панель модулей и создайте новую страницу. Поскольку любой элемент может быть размещен только на несущем контейнере, мы должны разместить прежде контейнер. Кликните на кнопке "Bands" (контейнеры)  и выберите любой простой контейнер, например Page Header (Заголовок страницы). Кликните где-нибудь на странице чтобы разместить этот контейнер. Потом кликните на кнопке с заголовком "Items" (элементы) или пиктограмме  и выберите "Мемо" . Также в профессиональной версии доступен расширенный «Мемо» элемент с дополнительным функционалом. Используйте любой из этих элементов если вам доступны оба. После выбора элемента «Мемо», кликните где-нибудь в пределах Page Header (Заголовок страницы) для того чтобы разместить элемент на нем. Элемент будет размещен в позиции курсора.



В зависимости от настроек Дизайнера сразу же будет открыт помощник элемента или вы можете вызвать его двойным кликом мыши на элементе. Напечатайте в редакторе помощника текст «Привет, мир!» и потом нажмите кнопку «Ok».

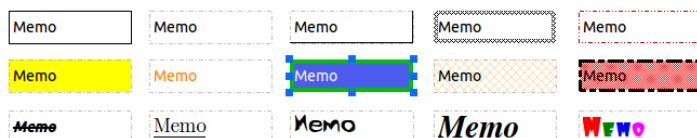


Теперь шаблон отчета готов. Для построения настоящего отчета из шаблона выберите в главном меню «Service → Run» или нажмите «F5» на клавиатуре. Дизайнер переключится на Предпросмотр и вы увидите сгенерированную страницу отчета с текстом «Привет, мир!». Построенный отчет может быть распечатан или экспортирован в один из поддерживаемых форматов.

## Элемент Мемо

Элемент Мемо имеет множество отличных особенностей для печати текста. Он может печатать текст в рамке и может иметь заливку произвольного цвета. Текст может быть отображен с использованием различных шрифтов, размера и стиля. Все свойства могут быть установлены используя Редактор Свойств или визуально в помощь инструментов в панели инструментов или с помощью помощника Мемо.

Ниже приведены несколько примеров:

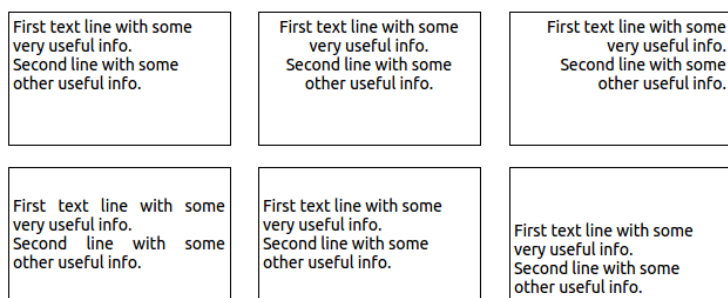


Мы создадим простой пример с использованием в Мемо двух строк текста:

*First text line with some very useful info.*

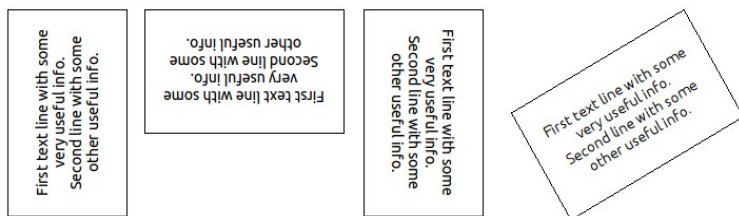
*Second line with some other useful info.*

Включите отображение границ в Мемо из Редактора Свойств или инструмента на панели инструментов и растяните элемент до размера 90x30 мм используя мышь или тот же Редактор Свойств. Как вы теперь можете увидеть Мемо может отображать текст не только одной строкой, но также текст содержащий несколько строк. Попробуйте уменьшить размер элемента до 50 мм. Очевидно, строки не могут быть вложены в границы элемента и будут перенесены. Это поведение настраивается значением **TextFlags::TextWordWrap** в свойстве **TextFlags**. Если оно отключено, то любые длинные строки будут отсекаются. Давайте немного поэкспериментируем с другими значениями этого свойства и посмотрим что получается.



## Вращение

Давайте взглянем на другую особенность: вращение. Любой объект, включая Мемо, может быть повернут на любой угол в пределах 0..360 градусов. Установите необходимый угол поворота в Редакторе Свойств изменяя свойство **"rotation"**. Границы Мемо будут выровнены соответственно указанному углу. Таким образом вам не стоит беспокоиться о границах.



## HTML Теги

Объект Мемо позволяет использовать большинство HTML тегов. Тег может быть размещен в тексте Мемо элемента. Теги отключены по умолчанию. Для того чтобы включить распознавание HTML тегов, установите свойство **"allowHTML"** в значение "true". Ниже приведены несколько примеров.

```
<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>
```

<i>Memo example</i>
E = mc <sup>2</sup>
A <sub>1</sub> = B <sup>2</sup>
this is a usual text, and this is a red one
this is a usual text, and this is an orange one

## Выражения

Выражения — это одна из наиболее важных особенностей элемента Мемо. Она позволяет показывать не только статичные текст, но и вычислять выражения в процессе генерации отчетов. Выражения могут чередоваться со статичным текстом. Чтобы понять как это работает, введите текст ниже в редакторе помощника Мемо элемента:

Now is [QDateTime.currentDateTime()]

(или упрощенный вариант с учетом локали: "Now is [DATE]")

и сгенерируйте отчет нажатием «F5» на клавиатуре. Вы увидите результат похожий на:

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

(или для переменной DATE: 18/07/2014 или 07/18/2014 в зависимости от вашей локали)

Почему так? Внутренний генератор CuteReport определяет каждое выражение в тексте, высчитывает его результат и подменяет исходное выражение его результатом. Текст Мемо элемента может содержать неограниченное количество выражений. Выражения могут использовать сложные математические операции, константы, переменные, объекты и их свойства. Но здесь кроется несколько потенциальных проблем, в случае если обычный текст содержит квадратные скобки, которые не



предполагались как выражение. Например, мы желаем напечатать:

```
аггау[0] = 'Банан'
```

Фрагмент [0] будет распознан как выражение, подсчитан с результатом 0 и будет помещен в текст. В результате мы получим:

```
аггау0 = 'Банан'
```

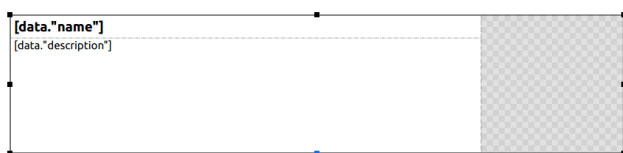
что определенно не то что мы хотели получить. Существует два решения этой проблемы:

- установить свойство **«allowExpressions»** в «false» чтобы запретить любые выражения в данном Мемо элементе
- изменить квадратные скобки на что-либо другое как символы определяющие границы выражений с помощью свойства **«expressionDelimiter»**

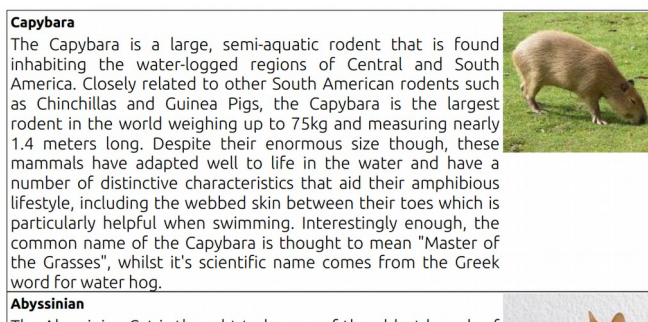
В первом случае весь текст будет считаться простым текстом без выражений. Во втором случае выражения будут распознаваться с помощью других символов, определяющих границы выражений. Вы могли бы использовать "<" как начало и ">" как конец. Но не используйте HTML текст в таком случае. Если вам нужно использовать HTML, вы могли бы использовать "<%" как начало и "%>" как конец выражения. Детектор выражений работает перед непосредственно рендерингом текста, таким образом, любые символы указанные в качестве разделите, будут вырезаны из текста. Не используйте один и тот же набор символов в качестве начала и конца выражения.

## Перетекание текста

Функция перетекания текста доступна в Профессиональной версии CuteReport. Она позволяет огигать текстом другие элементы. Давай сделаем простой пример для демонстрации данного функционала. Ниже приведен пример шаблона для построения списка, состоящего из имени животных и их описания:




Когда мы сгенерируем отчет, мы получим:




Здесь можно заметить пустое пространство под картинкой. Было бы неплохо использовать это пространство под текст. Чтобы добиться этого, мы добавим новый элемент Мемо под существующим элементом, содержащим описание, отключим растягивание для существующего Мемо. Вместо этого мы установим свойство «flowTo» в

нашем только что созданном Мемо (memo\_2) в «memo\_1». Таким образом это должно выглядеть так:

<code>[data."name"]</code>	
<code>[data."description"]</code>	

Наш отрендеренный результат:

**Capibara**  
The Capibara 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 Capibara 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 Capibara is thought to mean "Master of the Grasses", whilst it's scientific name comes from the Greek word for water hog.

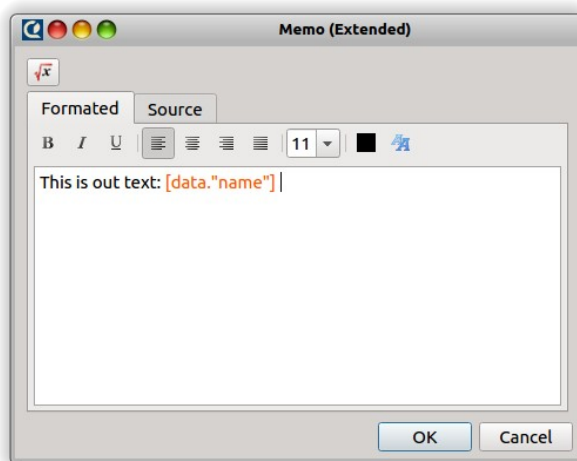
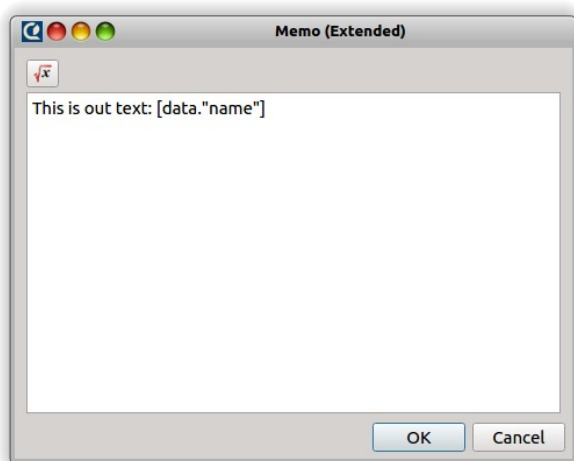


Это намного красивее. Не так ли?


Полее подробному рассмотрению функции обтекания текстом посвящена одна из следующих глав.

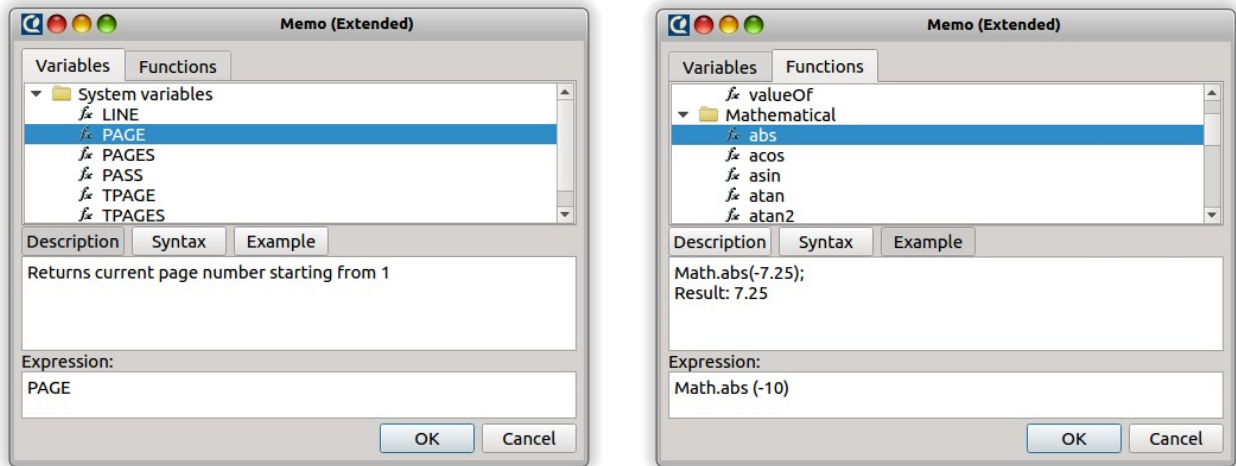
## Помощник Мемо

Использование Редактора Свойств для изменения свойств Мемо не является единственным путем. Вы можете воспользоваться Помощником. Он является полезным при вводе длинного текста или построении выражений или для форматирования HTML текста. Для вызова Помощника дважды кликните на элементе Мемо. В зависимости от свойства «allowHTML» открытое диалоговое окно будет иметь или не иметь панель инструментов HTML текста. Ниже 2 вида для обоих случаев:



Используйте элементы управления HTML как в любом другом редакторе. Также вы можете прямо использовать HTML теги для получения желаемого HTML текста во вкладке «Source». Все изменения будут моментально отражены во вкладке «Formatted». Вы можете совмещать HTML код с выражениями безо всякого ограничения, кроме ограничения использовать корректный разделитель выражений, который не

пересекается с HTML тегами. Если ваш пакет CuteReport содержит дополнительные модули обеспечивающие дополнительный функционал, вы увидите дополнительные кнопки сверху окна Помощника. В нашем примере мы имеем установленный Редактор Выражений (Expression Editor). Если вы нажмете кнопку , то появится форма Редактора Выражений. Она содержит несколько вкладок и предоставляет список всех доступных переменных, функций, методов которые могут быть использованы для построения выражений:



#### Другие полезные свойства элемента Мемо:

- **stretchMode**: растяжение элемента чтобы вместить весь текст
- **showStretchability**: показывать растягивание в дизайнера
- **expressionDelimiter**: две строки разделенные запятой определяющие начало и конец выражения.
- **stretchFont**: устанавливать автоматический размер шрифта чтобы вместить текст

# Контейнеры

Контейнеры служат для динамического или статического позиционирования графических элементов на странице. Каждый контейнер имеет свои собственные позицию и набор функций. CuteReport содержит несколько специальных контейнеров предназначенных для представления данных из наборов данных. Набор данных (Dataset) содержит структурированные данные организованные в строки имеющие одно или более столбцов (полей). Для печати данных из наборов данных в CuteReport используются специальные контейнеры и именами "Detail...". Для того чтобы они заработали добавьте один или более таких контейнеров на страницу, соедините их с набором данных и разместите на них Мемо элемент. Как только контейнер подключен к набору данных, Мемо будет иметь появляющуюся при наведении на него кнопку справа с выпадающим списком всех полей набора данных (только профессиональная версия). В процессе рендеринга отчета эти контейнеры будут распечатаны на странице. «Detail» контейнер обычно будет распечатан один раз на каждую строкц набора данных, «DetailHeader» и «DetailFooter» будут напечатаны в соответствии с их свойством «condition» (условие). Если на странице не остается более свободного пространства для печати контейнера, то для продолжения будет автоматически создана новая страница. На вновь созданной странице будут предварительно напечатаны заголовок и подвал страницы. Этот процесс размещения контейнеров и элементов и называется рендеринг. Ниже приведен список контейнеров содержащихся в стандартной поставке и их краткое описание:

- **PageHeader (Заголовок страницы):** содержит все элементы печатаемые вверху страницы
- **PageFooter (Подвал страницы):** содержит все элементы печатаемые внизу страницы
- **Title (Заголовок):** содержит все элементы печатаемые в начале отчета
- **Summary:** содержит все элементы печатаемые в конце отчета
- **Detail:** должен быть соединен с набором данных и содержит все элементы печатаемые для каждой строки набора данных
- **DetailHeader:** должен быть соединен с набором данных и содержит все элементы печатаемые для каждой строки набора данных если результат подсчета выражения в свойстве «condition» является «true». Контейнер может быть использован для печати шапки группы контейнеров.
- **DetailFooter:** должен быть соединен с набором данных и содержит все элементы печатаемые для каждой строки набора данных если результат подсчета выражения в свойстве «condition» является «true». Контейнер может быть использован для печати подвала группы контейнеров.
- **Overlay:** может быть размещен в произвольном месте страницы. Может быть использован как контейнер для переднего слоя, фона, водяного знака.

## Хранилища

Теперь мы рассмотрим следующий класс объектов в CuteReport. Хранилище — это структура, которая сохраняет все данные используемые в отчете, такие как иллюстрации, шаблоны, базы данных и прочее. CuteReport предоставляет несколько стандартных типов хранилищ: FileSystem (файловая система), GIT, resource (ресурсы), Database (база данных). Определенно, вы знакомы с некоторыми или всеми из них. Позднее мы рассмотрим каждое хранилище детально, но пока обратим внимание на общие аспекты хранилищ. В CuteReport вы можете создавать любое количество объектов каждого типа. Это полезно если вы имеете несколько удаленных систем, которые хранят шаблоны отчетов с различными настройками. Как один GIT сервер с доступом только для чтения и один GIT сервер с полным доступом. В этом случае вы можете создать 2 хранилища типа «GIT» и установить соответствующие настройки для каждого из них. Все хранилища в CuteReport имеют собственное уникальное имя, основанное на схеме хранилища. Таким образом 2 объекта с одинаковым типом «GIT» будут иметь имена «git\_1» и «git\_2». Любое имя может быть изменено на любое желаемое, но оно все же должно быть уникальным в пределах отчета. Ядро CuteReport не позволит вам назначить уже существующее имя. Чтобы получить доступ к объектам в хранилище, вы будете использовать путь в стиле ссылки, подобно: «file\_1:/file\_path/image\_file.jpg», где «file\_1» - это имя хранилища и «/file\_path/image\_file.jpg» - путь к файлу в этом хранилище. Также возможно назначить хранилище по умолчанию для всех отчетов в диалоге настроек Дизайнера используя "Tools → Options → Storage", так что любое имя файла без указанного имени хранилища будет передано в хранилище по умолчанию.

Существует 2 вида хранилищ с различным приоритетом: глобальные хранилища и собственные хранилища отчета. Когда ядро CuteReport встречает ссылку на файл, во-первых оно проверяет собственные хранилища отчета, потом глобальные хранилища. Поэтому если вы хотите специальное хранилище со специфическими настройками для вашего отчета, добавьте хранилище нужного типа прямо в шаблон вашего отчета во вкладке «Reports». Ваш отчет будет всегда использовать это хранилище независимо от настроек и наличия объектов глобальных хранилищ.

### Хранилище «Файловая система» (Standard::Filesystem)

«Файловая система» (FileSystem) — наиболее часто используемое хранилище. Оно имеет только несколько опций: «корневой каталог» (root folder) и «спрашивать о перезаписи» (ask for rewrite). «Корневой каталог» — это самый верхний каталог доступный для пользователя. «Спрашивать о перезаписи» - опция используемая для определения надо ли показывать диалог при перезаписи файла или нет.

### Хранилище «GIT» (Standard::GIT)

Этот тип хранилищ может быть использован для хранения шаблонов отчетов и их объектов на локальной или удаленной системе контроля версий GIT. Присутствуют следующие настройки:

- remote url: ссылка на репозиторий
- login, password: логин и пароль
- local path: локальный путь куда git репозиторий будет клонирован

- git binary: путь к бинарному файлу запуска git. CuteReport использует внешний бинарный файл для управления репозиторием git, так что для правильной работы хранилища он должен быть указан
- кнопка «sync now»: кнопка для клонирования и отправки данных из/в удаленный репозиторий

## **Хранилище «Resource» (Standard::Resource)**

Все объекты в этом хранилище будут внедрены непосредственно в файл шаблона отчета.

## **Хранилище «SQL» (Standard::SQL)**

Это хранилище сделано чтобы помочь вам просто сохранять и загружать шаблоны отчетов в/их SQL базы данных без написания строчки кода в вашем приложении. Просто укажите информацию о вашей базе данных и корректную таблицу и поле таблицы где хранятся отчеты или объекты отчетов.

## Наборы данных

Как было упомянуто ранее набор данных — это объект, который содержит структурированные данные организованные в строки с одним или более столбцом(полем). Набор данных может получать значения полей с различных источников и он обеспечивает общий программный интерфейс для данных. CuteReport предоставляет несколько стандартных наборов данных: SQL, CSV, FileSystem (файловая система), Model (модель). Каждый из них получает данные из разных источников.

### Типы наборов данных:

- **SQL Dataset (Standard::SQL):** обеспечивает интерфейс для получения значений из SQL базы данных. Может работать с различными базами данных поддерживаемых Qt, т.е для которых в Qt есть драйвер. Имеет несколько настроек для подключения к удаленной базе данных (таких как mysql, postgresql) или для использования встраиваемой базы данных (как sqlite).
- **CSV Dataset (Standard::CSV):** обеспечивает интерфейс для данных размещенных в текстовом файле и разделенных запятой или другим символом. Он может загружать данные из внешнего файла каждый раз при наполнении или загрузить из и держать во внутреннем кеше.
- **Filesystem Dataset (Standard::Filesystem):** обеспечивает интерфейс для получения информации из файловой системы и отображать информацию о дисках, файлах, каталогах в виде данных структурированных в строки. Имеет несколько настроек: фильтрацию, уровень рекурсии, максимальное количество выводимых строк. Этот набор данных может быть использован для получения списка товаров, фото каталога и пр.
- **ModelDataset (Standard::Model):** Используется для экспорта подготовленных данных из вашей программы в объект отчета. Если у вас в программе имеется такой виджет как QTableView или QTableWidget или модель отнаследованная от QabstractItemModel вы можете легко распечатать данные из нее используя это набор данных.

Попробуйте поэкспериментировать с каждым набором данных чтобы понять как он работает. Вы можете получать данные из набора и просматривать их в виде таблицы через нажатие кнопки «Test it». Данные из всех наборов данных доступны из скриптового движка и могут быть использованы в любом из объектов отчета. Чтобы получить данные из набора данных там где это доступно вы можете использовать выражение типа **[имяНабораДанных."имяПоля"]** или **[имяНабораДанных .getValue("имяПоля")]** . Первая форма является сокращением второй. Каждая сокращенная форма выражения заменяется полной формой непосредственно перед запуском процесс построения отчета автоматически..

Для более детальной информации о каждом наборе данных обратитесь к главе «Наборы данных».

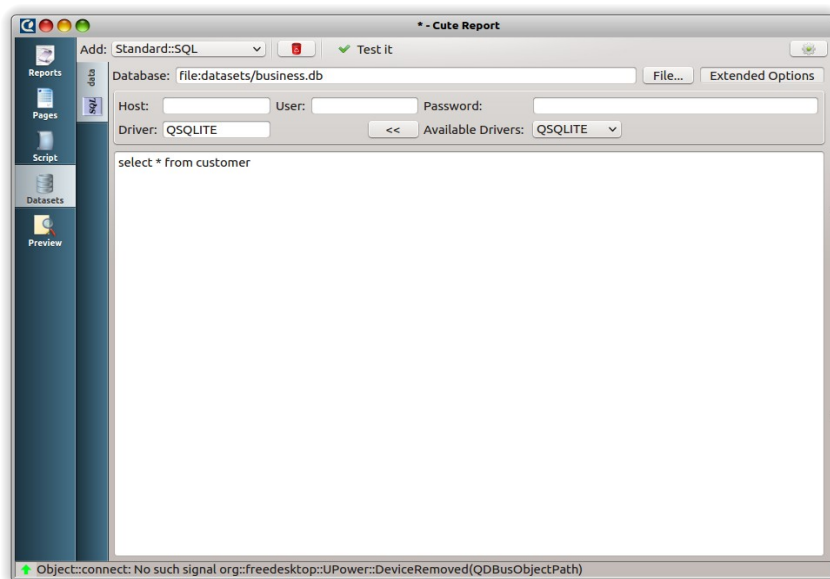
Позднее мы рассмотрим как подключать наборы данных к контейнерам и как использовать эти данные.

## Пример отчета «Список клиентов»

Теперь мы создадим наш второй отчет и рассмотрим как использовать наборы данных. Для этого мы будем использовать тестовую sqlite базу данных с именем «business.db». Во первых, создадим новый пустой отчет нажатием «main menu -> Report -> New Report». Потом перейдем на вкладку «Datasets» и кликнем на кнопке SQL типа для создания набора данных. Теперь у нас есть SQL набор данных в нашем отчете. Давайте установим для него корректные параметры. Кликнем «File...». Когда появится диалог открытия файла базы данных убедимся что выбрано правильное хранилище в «Storage» комбобоксе, найдем файл с именем «business.db» и выберем его. Теперь у вас должно быть заполнено поле «Database:» и оно должно содержать путь к файлу, что-то вроде «file:dataset.db». Поскольку у нас нет имени хоста, имени пользователя и пароля, оставляем эти поля пустыми. Выберите «SQLITE» драйвер в списке доступных драйверов «Available Drivers:» и нажмите «<<» чтобы скопировать имя драйвера в поле «Driver». Теперь мы можем подключиться к нашей тестовой базе. Давайте напишем простой запрос чтобы проверить ее:

```
select * from customer
```

Как результат мы имеем что-то типа изображенного на картинке:



Нажмем "Test it" и откроется таблица с полученными данными. Если что-то пошло не так и вы не видите результата, проверьте наличие ошибок в виджете ошибок в статусной строке. Отлично! Теперь вы закончили создание набора данных.

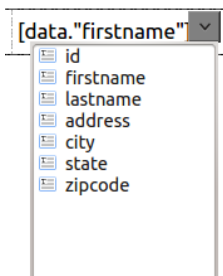


The screenshot shows the 'Cute Report' application window. On the left is a sidebar with icons for Reports, Pages, Script, Datasets, and Preview. The main area displays a table with 14 rows of client data. The status bar at the bottom shows a connection error: 'Object::connect: No such signal org::freedesktop::UPower::DeviceRemoved(QDBusObjectPath)'.

	id	firstname	lastname	address	city	state	zipcode
1	1	Ximena	Garrett	2278 Secon...	Paramus	NJ	7843
2	2	Jaunita	Johnson	5505 Park ...	Drummond	OK	73124
3	3	Calista	Baskin	2017 Seven...	Boca Raton	FL	94765
4	4	Tyra	Marcotte	6745 Fifth ...	Springboro	PA	54322
5	5	Steve	Neeley	793 Main St...	Edward	NC	76348
6	6	Vida	Brock	395 Sixth Ci...	Wrightsboro	TX	89372
7	7	Carolyn	London	990 Hill Str...	Metairie Falls	WA	67901
8	8	Tia	Bergman	4327 Cedar...	Jacksonville	NC	76320
9	9	Brady	James	875 Lake Ci...	Stockholm	WI	98347
10	10	Alia	Bohannon	5306 maple...	Seiling	OK	75349
11	11	Andeana	Pratt	1096 Eight...	Des Moines	IA	10345
12	12	Ellens	Vanmeter	403 Elm Cir...	Orchard	CO	80632
13	13	Waylon	Hardison	1078 Fourt...	Harmon	IL	86432
14	14	Ankti	Vanwinkie	3956 Eight...	Byron	CA	94321

Перейдите во вкладку "Page" и создайте новую страницу если она еще не создана. Возможны несколько типов страниц. Пока что выберите "Standard::Page" или "Extended::Page" (для профессиональной версии). Кликните на комбобоксе и выберите нужный тип страницы и после кликните на кнопке "Add new page" . Добавьте на страницу контейнер "Title", установите корректный размер протягиванием синих направляющих или установкой корректного размера в Редакторе свойств. Теперь разместите элемент "Мемо" в центре контейнера и установите его размеры тоже. Теперь сделайте двойной клик на элементе Мемо и напечатайте «Список клиентов» и после этого нажмите "Ok". Чтобы отцентрировать текст, кликните на свойстве "TextFlags" элемента Мемо в Редакторе свойств и включите "AlignHCenter". Следующий шаг это добавление контейнера "Detail" и подключение его к набору данных клиентов. Чтобы сделать это активируйте контейнер щелчком на нем и напечатайте "data" в его свойстве "dataset". Введенный текст "data" — это имя нашего набора данных. Вы можете изменить имя любого набора данных дважды кликнув на вкладке с именем набора данных в Редакторе Данных. Теперь время добавить несколько элементов на контейнер: номер, Имя, Фамилия, Адрес, Город, Индекс. Для того чтобы проще выравнивать элементы вы можете включить магниты нажатием на кнопки магнитов вверху Редактора Страниц. Если вы желаете изменить имя элемента и назначить ему более понятное и распознаваемое в Инспекторе Объектов имя, идите к Редактору свойств и измените свойство "objectName" на что-то вроде: memoFirstName, memoSecondName, memoAddress и так далее. Мы не будем использовать эти имена в этом примере, но это может пригодиться вам позднее. Далее добавим инструкции к элементам Мемо что им необходимо отображать. Перейдем к первому, сделаем двойной щелчок и напечатаем "[LINE]" в редакторе Помощника. Как было объяснено ранее, "[]" определяют границы выражений и текст "LINE" - это внутренняя переменная которая хранит значение текущей строки набора данных. Нажмите "F5" и вы увидите как это работает.

Теперь вернемся назад в Редактор Страниц путем нажатия на вкладку "Pages" в панели модулей и заполним второй элемент Мемо данными об имени клиента. Добавьте текст "[data."firstname"]" в этот элемент. Пользователи Профессиональной версии CuteReport могут просто кликнуть на серой кнопке появляющейся в правой стороне элемента как показано здесь:



Этот выпадающий список содержит все поля набора данных к которому относится элемент Мемо. Это будет работать только если Мемо расположен на контейнере который имеет заполненное свойство "dataset" и если набор данных с таким именем существует и правильно настроен.

Прodelайте то же самое со всеми элементами и установите корректное поле для отображения. Нажмите "F5" и вы увидите что-то вроде этого:

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	Selling	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

Если вы не получили такого результата, вы можете загрузить готовый файл CustomerList.qtrp из поставки CuteReport и найти что вы сделали не так.

## Элемент Image

Элемент Image предназначен для представления любых изображений в поддерживаемых графических форматах. В настоящее время поддерживаются следующие форматы: 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).

Давайте рассмотрим этот элемент ближе. Создайте новый отчет, добавьте новую страницу, добавьте элемент Image. Для элемента доступно несколько источников данных: Static, Storage, Dataset. Тип источника определен в свойстве **sourceType**.

Рассмотрим каждый источник:

- **Static:** позволяет загрузить иллюстрацию из файла и сохранять загруженные данные внутри объекта. Файл должен быть загружен вручную нажатием на свойство "image" в Редакторе Свойств.
- **Storage:** позволяет загружать файл в процессе генерации отчета. Свойство элемента "source" должно содержать путь к необходимому. Если путь не содержит имени хранилища, то будет использовано хранилище по умолчанию. Свойство "source" может содержать выражения. Например, если источник определен как "file:[data."filename"]" то движок отчета попытается найти имя файла в наборе данных с именем "data" и потом попытается загрузить файл с таким именем из хранилища с именем "file".
- **Dataset:** позволяет загружать файл из бинарного поля набора данных. Свойство "source" должно быть определено подобно рассмотренному выше «Storage».

Перечень других важных свойств:

- **keepAspectRatio:** Если установлено в «true», то Image будет всегда сохранять оригинальные пропорции при изменении размеров.
- **Center:** Если установлено в «true», то иллюстрация всегда будет отцентрирована в границах объекта.

## Отчет с иллюстрациями

В этой главе мы увидим как использовать элемент Image и набор данных «Файловая система (File System)».

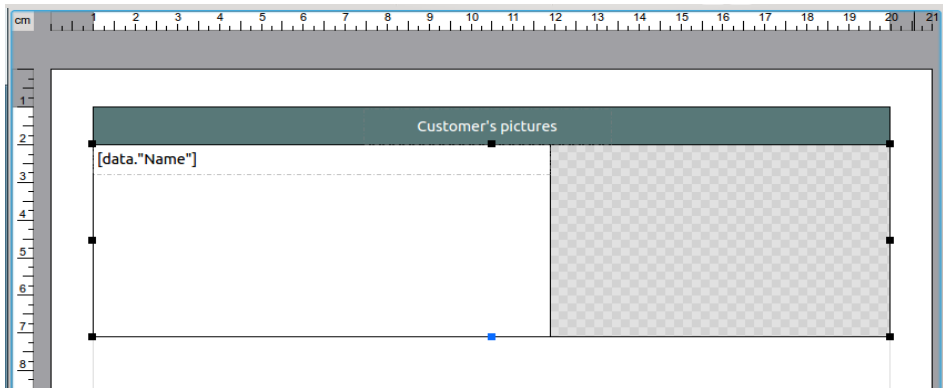
Давайте создадим новый отчет, добавим новую страницу, добавим контейнеры «Title» и «Detail». Далее мы создадим объект набора данных File System. Перейдите в Редактор Данных через вкладку "Datasets", кликните на «Standard::FileSystem». Выберите любой каталог в котором есть иллюстрации нажатием на кнопку "Select dir...". Установите «maxNumber» в 6. Это максимальное количество записей, которые набор данных получит. Отключите флаги: Directories, All Directories. Добавьте фильтр: «.jpg; .png» или любые другие форматы которые вы хотите. Установите «Path appearance» в «AbsolutePath», таким образом мы сможем загрузить иллюстрации используя их абсолютный путь. Теперь нажмите "Test it". В результате вы сможете увидеть 6 строк или менее с файлами выбранного вами в фильтре формата.

Перейдите в Редактор Страниц. Добавьте элемент Мемо на Title контейнер, напечатайте здесь "Фотографии клиента" и выровняйте текст по центру. Для лучшего вида установите «backgroundBrush» для контейнера Title. Установите «backgroundBrush::style» в «SolidPattern» и «backgroundBrush::color» в «#688482». Теперь измените цвет текста. Кликните на Мемо и измените значение свойства «textColor» на белый. Готово!

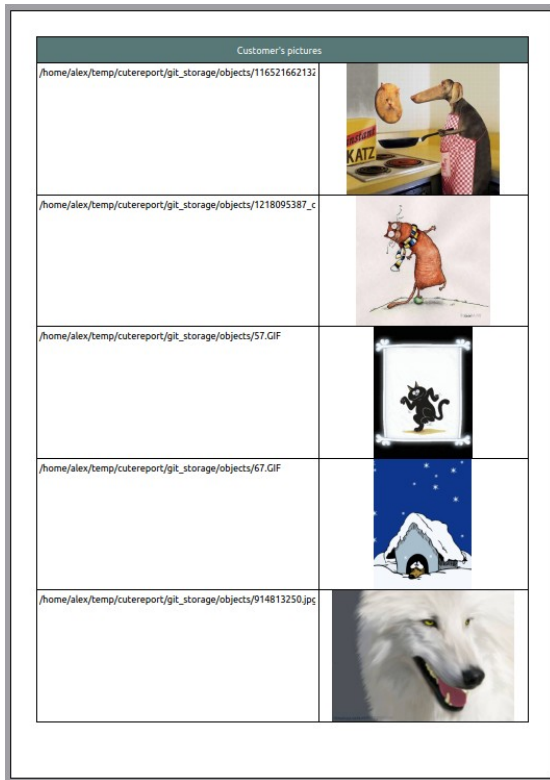
Теперь приступим к созданию картинки. Установите высоту контейнера Detail в 30 мм. Добавьте элемент Image на правую часть контейнера и измените его размер так чтобы он соответствовал контейнеру. Установите «sourceType» в «Dataset» и «source» в «file://[data."name"]». Как вы помните все в середине квадратных скобок будет определено как выражение и будет заменено его фактическим значением. Таким образом наш источник будет выглядеть примерно как «file://picture\_path/picture.jpg». Да, здесь другая важная деталь: **если отчет загружает какие-нибудь данные в процессе рендеринга, он обязан иметь встроенный объект хранилища!** Это сделано для безопасности. В некоторых случаях вы захотите не позволять пользователю использовать какое-нибудь глобальное хранилище CuteReport. Поэтому перейдите во вкладку «Reports» и добавьте хранилище «Standard::FileSystem». Мы имеем абсолютный путь к иллюстрациям, поэтому очистите «Root folder» для этого хранилища.

И последний шаг — это создание элемента Мемо который содержит путь файла. Добавьте новый Мемо в левую часть контейнера Detail и напечатайте там: «[data."Name"]». Пользователи Профессиональной версии могут просто нажать на кнопку появляющуюся при наведении мыши на элемент.

В конечном итоге наш шаблон отчета должен выглядеть так:



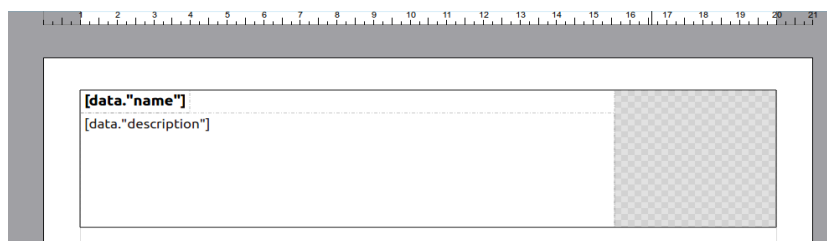
Теперь время нажать на «F5» и насладиться результатом:



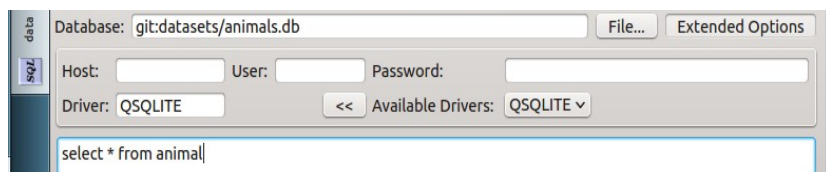
## Печать многострочного теста

Давайте проследуем дальше и рассмотрим как управлять многострочным текстом. В предыдущей главе мы научились создавать новые отчеты, создавать наборы данных и подключать наборы данных к контейнерам. Давайте сделаем это:







- создать новый пустой шаблон отчета
- добавить новый набор данных SQL и использовать тестовую базу данных «animals.db», которую вы можете найти в комплекте с CuteReport пакетом или взять тут [https://github.com/AlFoX/CuteReport\\_examples/tree/master/datasets](https://github.com/AlFoX/CuteReport_examples/tree/master/datasets).
- Добавить контейнер Detail
- положить 2 элемента Мемо на контейнер: первый — название животного, второй — его описание
- положить элемент Image на правую сторону контейнера



Теперь взглянем ближе на создание набора данных. Когда вы добавили набор данных SQL, укажите ему файл «animals.db», установите драйвер (Driver) - «SQLITE» и напечатайте текст запроса к базе данных "select \* from animal". Нажмите "Test it" проверьте все ли сделано правильно. Ниже вы можете видеть как оно должно выглядеть в итоге.








Когда эта часть готова, возвращайтесь в Редактор Страниц и установите корректные поля для показа в элементах Мемо. Первый верхний Мемо — это название животного. Поэтому сделайте двойной клик на нем и напечатайте: [data."name"]. Пользователи Профессиональной версии могут просто нажать на кнопку появляющейся при наведении мыши на элемент и выбрать поле из выпадающего списка. Если кнопка не появляется, проверьте подключен ли ваш контейнер к корректному набору данных, т.е. заполнено ли свойство "dataset" контейнера под этим Мемо. Установите жирный шрифт для названия животного нажатием на свойство «font» в Редакторе Свойств или используя редактор шрифта в панели инструментов. Теперь переходите ко второму Мемо. Это описание животного. Поэтому впечатайте: [data."description"]. Просто, да? Давайте посмотрим на результат и сгенерируем отчет(нажмите «F5»):

<b>Capybara</b> 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	
<b>Abyssinian</b> 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.	
<b>Adelie Penguin</b> 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.	
<b>Baboon</b> 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	
<b>Camel</b> 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.	
<b>Dog</b> 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.	

Выглядит не очень, правда? Некоторые описания были обрезаны. Конечно мы можем просто изменить высоту для Мемо с описанием животного чтобы оно вместило самый большой текст, но в таком решении есть несколько недостатков:

- чрезмерной расход бумаги, поскольку мы не будем полностью использовать все пространство элемента Мемо если он содержит мало текста
- вы никогда не знаете какой длины будет текст или он может измениться в будущем
- это просто не выглядит красиво :)

Исправим же это: установим свойство «stretchMode» объекта Мемо в «ActualHeight», установим свойство "stretchable" контейнера Detail в «true» и перестроим отчет.

<b>Capybara</b> 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	
<b>Abyssinian</b> 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.	
<b>Adelie Penguin</b> 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.	
<b>Baboon</b> 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.	
<b>Camel</b> 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.	

Как вы можете видеть «stretchMode» исправило положение. Ниже приводятся другие значения этого свойства:

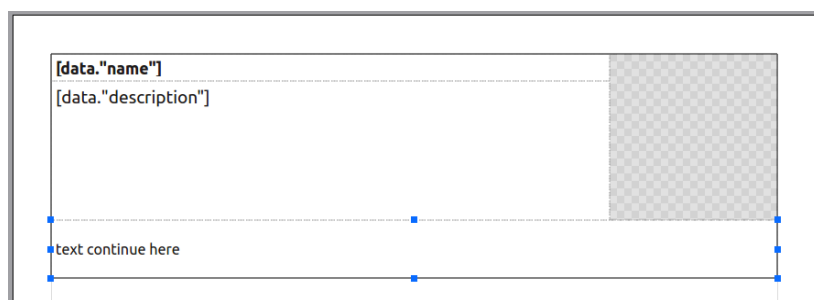
- DontStretch — не растягивать объект
- ActualHeight — растягивать высоту объекта чтобы вместить весь его текст
- MaxHeight — растягивать высоту объекта чтобы достигать нижней границы контейнера



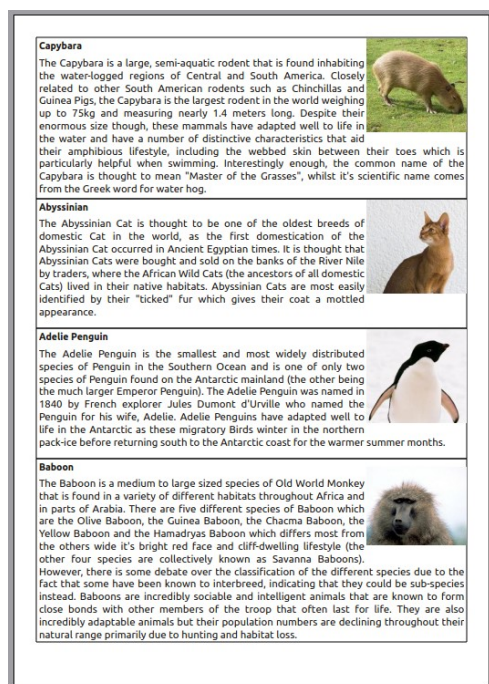
# Обтекание объектов текстом

(только профессиональная версия)

Иногда вам потребуется создать отчет требующий обтекания текста вокруг объектов. Это может быть иллюстрация или таблица. Ранее в главе посвященной Мемо элементу мы уже затрагивали эту особенность. На самом деле эта задача не сложна в CuteReport. Просто добавим один дополнительный Мемо куда будет перетекать текст. Для второго Мемо установим свойство «flowTo» с именем первого Мемо где текст начинается. Например, если первый Мемо имеет имя «memo\_1», установите свойство «flowTo» второго Мемо в «memo\_1». Свойство «StretchMode» первого Мемо должно быть установлено в "DontStretch" и это же свойство второго Мемо в "ActualHeight". Это всё.



Теперь построим отчет и посмотрим как он выглядит:



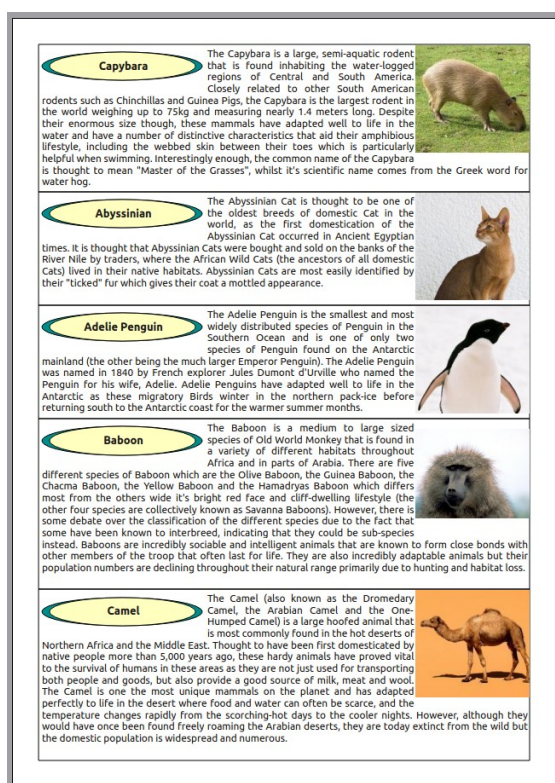
Для этого примера мы установили флаг «**AlignJusify**» в свойстве «**textFlag**». Не является необходимым выставлять этот флаг для всех связанных Мемо элементов, так как все последующие связанные элементы наследуют форматирование предыдущих.

## Сложное обтекание

Создание сложного обтекание не намного белее сложная процедура. Сделаем следующий пример:

<b>[data."name"]</b>	<b>[data."description"]</b>	

Как вы можете заметить здесь 3 Мемо объектов, которые используются для того чтобы вместить текст: первый в центре сверху содержит текст «[data."description"]». Второй находится под первым и занимает левую и центральную часть. Третий расположен снизу под всеми другими элементами. Его высота установлена минимальной, поскольку он не будет использоваться в случае короткого текста. Нам не нужен расход пространства. Каждой последующий элемент Мемо соединен с предыдущим установкой свойства «**flowTo**». Свойство «**stretchMode**» последнего элемента установлено в «**ActualHeight**». Первые два элемента не нуждаются в растягивании, поэтому их свойство установлено в "DontStretch". Нажмите "**F5**" для того чтобы построить отчет и вуаля:



Вам не нужно беспокоиться о порядке добавления элементов, вы можете добавлять Мемо объекты в произвольном порядке. Как только вы установите корректное значение свойства «**flowTo**» все будет сделано автоматически. CuteReport достаточно умен чтобы понимать что вам нужно и прячет от вашего взгляда всю рутинную работу. Наслаждайтесь!

# Печать ярлыков

(только коммерческая версия)

В этой главе мы увидим как используя CuteReport создать отчет с колонками. Это может быть полезным для печати ярлыков и чего-то похожего. Давайте сделаем простой отчет, соержащий ярлыки заказчиков для распечатывания и наклеивания на папки наших заказчиков. Ниже приведен пример:

Customer list	
[LI] [data."firstname"]	05/2006
[data."lastname"]	
[data."address"]	
[data."city"]	
[data."zipcod"]	

После построения отчета мы получим следующее:

Customer list	
1 Ximena Garrett 2278 Second Street Paramus 7843	
2 Jaunita Johnson 5505 Park Boulevard Drummond 73124	
3 Calista Baskin	

Как вы можете видеть пропадает очень много места справа, а значит пропадает много бумаги. Для оптимизации использования пространства мы сделаем несколько колонок чтобы заполнить весь лист ярлыками. Это может быть сделано используя свойство страницы **«columns»**. Установите его в «3» в Редакторе Свойств на правой стороне экрана. Потом нажмите **"F5"** для построения отчета.

Customer list		
1 Ximena Garrett 2278 Second Street Paramus 7843	7 Carolyn London 990 Hill Street Metaline Falls 67901	13 Waylon Hardison 1078 Fourth South Harmon 86432
2 Jaunita Johnson 5505 Park Boulevard Drummond 73124	8 Tia Bergman 4327 Cedar Avenue Jacksonville 76320	14 Ankti Vanwinkie 3956 Eighth Circle Byron 94321
3 Calista Baskin 2017 Seventh Street Boca Raton 94765	9 Brady James 875 Lake Circle Stockholm 98347	

Существует два типа заполнения колонок: вертикальное **"Vertical"** и горизонтальное **"Horizontal"**, котрые устанавливаются через свойство страницы **"fillDirection"**. На иллюстрации выше мы можем увидеть вертикальный тип заполнения, подразумевающий что ярлыки будут печататься один под другим пока в колонке есть свободное место. Когда в колонке нету свободного места, движок отчета создаст новую



колонку и начнет заполнять ее сверху. Горизонтальный тип подразумевает что каждый следующий ярлык будет напечатан справа от предыдущего пока есть свободное место справа. Если места справа нет, движок отчета будет печатать следующий ярлык на следующей строке как вы можете увидеть ниже:

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

Вы можете выбирать тип в зависимости от ваших нужд. Не все контейнеры принимают во внимание настройку колонок. Некоторые игнорируют их, такие как PageHeader, PageFooter. Некоторые другие имеют специальные свойства для управления таким поведением, как DetailHeader и DetailFooter. Используя эту настройку вы можете строить шаблоны отчетов с колонками со сложным дизайном. Один из примеров ниже, это отчет содержащий сгруппированные колонки:

My Music Collection				
Title		length		
Chris Norman				
3	Sarah (You Take My Breath Away)	342	4 I Want To Be Needed (Duet With)	633
5	Never Make My Angel Cry	345	6 Hunters Of The Night	345
7	As Good As It Gets	532	8 Every Little Thing	432
9	Red Hot Screaming Love	342	10 Still In Love With You	243
11	Jealous Heart	342	12 Stupid Girl	342
Chris Norman tracks: 12 min: 243 max: 633 avg: 399.00 Total length: 4794				
Richard Clayderman				
1	Gimme! Gimme! Gimme!	356	2 The Winner Takes It All	489
3	Chiquitita	683	4 Fernando	386
5	Mamma Mia	496	6 Piano Concerto No.1 In B-Flat	453
7	Elvira Madigan- Piano Concert	533	8 Cornish Rhapsody	522
9	Liebestraum (Reves D'Amour)	754	10 La Pathetique	321
11	Fur Elise	643	12 Arabesque	643
13	Aquarela	254	14 Ballade pour Adeline	456
15	Para Elisa	564	16 Yesterday	564
17	Just the way you are	335	18 Just the way you are	467
19	Ne me quitte pas	624		
Richard Clayderman tracks: 19 min: 254 max: 754 avg: 502.68 Total length: 9543				
Roxette				
1	Spending My Time	342	2 Crash! Boom! Bang!	234
3	Run To You	234	4 It Must Have Been Love	533
5	I'm Sorry	264	6 Almighty 7 Mix	245
7	Almighty 12 Definitive Mix	269	8 Almighty Alternate 12	439
9	X-Treme Extended Mix	298	10 Opportunity Nox	659
11	The Look	342	12 Dressed For Success	432
13	Dangerous	532	14 Joyride	432
my music collection		page 2 of 6		

# Многостраничный отчет

В CuteReport возможно создать многостраничный шаблон отчета. Эта особенность полезна если отчет должен содержать различные страницы с разным размером, ориентацией и т.д. В этом случае движок отчета будет сначала полностью обрабатывать первую страницу, потом вторую и т. д. Максимальное количество страниц шаблона не ограничено. Давайте рассмотрим простой пример с 2-мя страницами, где первая страница титульная и вторая — это непосредственно страница отчета с таблицей. Мы будем использовать наш предыдущий пример "Список клиентов". Чтобы добавить новую страницу, кликните на кнопке  на панели инструментов Редактора Страниц. Если вам доступны страницы нескольких типов, вы увидите выпадающий список. Выберите стиль страницы кликом на нем. Новая страницы будет добавлена к отчету. Передвиньте ее на первое место кликом на . Поскольку элемент не может быть расположен прямо на странице, мы добавим контейнер **«Overlay»** в середину страницы. Теперь добавим элемент Мемо на контейнер и введем в нем текст «Список клиентов». Построим отчет и теперь он должен выглядеть примерно так:

## Customer report

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	Nesley	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	Alla	Bohannon	5306 maple Avenue	Selling 75349
11	Andeana	Pratt	1096 Eighth Boulevard	Des Moines 10345
12	Ellens	Vanmeter	403 Elm Circle	Orchard 80632
13	Waylon	Hardison	1078 Fourth South Boulevard	Harmon 86432
14	Ankti	Vanwinkle	3956 Eighth Circle	Byron 94321

# Скриптовый движок

В этой главе мы рассмотрим как работать со скриптовым движком CuteReport. Возможность скриптования привносит чрезвычайно высокий уровень гибкости. Используя скрипт пользователь может контролировать практически каждый шаг рендеринга и строить действительно сложные отчеты. В отчете присутствует главный скрипт, который контролирует все от старта рендеринга до его окончания. Некоторые элементы, такие как Мемо или Barcode поддерживают скриптование в их текстовых свойствах. Обычно скриптовые выражения должны быть обрамлены квадратными скобками [], так чтобы скриптовый движок знал что это скрипт а не обычный текст. Но для некоторых свойств допустимы только скриптовые выражения и они могут быть написаны без []. Некоторые элементы могут переопределять «[]» и использовать что-то другое или предоставлять дополнительное свойство для определения границ скриптовых выражений, как, например, это делает свойство «expDelimiter» в элементе Мемо.

## Объекты в скрипте

Все объекты отчета доступны из скрипта по их имени. Например, если у вас есть объект Мемо с именем мемо\_1 и вы хотите установить его цвет, вы можете сделать это следующим способом:

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

Все свойства объекта которые вы можете видеть в Редакторе Свойств могут управляться через скрипт. Некоторые объекты имеют несколько типов одного и того же свойства, как свойство "stretchMode" в элементе Мемо. Имеется 2 типа: enum and string. Пользуйтесь тем что вам удобнее:

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

## Переменные в скрипте

Каждый объект или значение сохраняется в скрипте как переменная. Вы можете создавать свои собственные переменные при старте отчета и изменять их значение во время построения отчета. Скриптовый движок может иметь некоторые внутренние переменные доступные пользователю.

### Локальные переменные

Переменные могут объявляться и использоваться локально в скрипте. После того как переменная объявлена, ей может быть присвоено значение. Вот один пример:

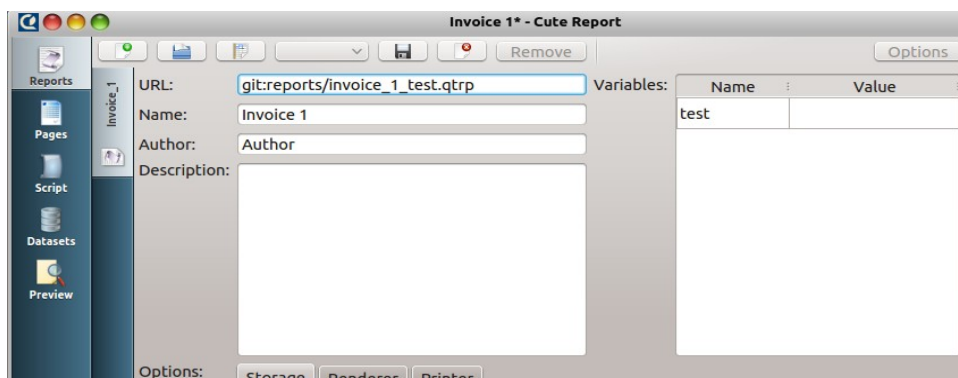
```
var myVar = "Привет, мир!";
```

Когда вы объявили переменную вы можете использовать ее в любых объектах отчета, например в Мето напечатав «[myVar]» в свойстве **«text»**. Для более детальной информации о переменных JavaScript обратитесь к документации по языку JavaScript.

### Глобальные переменные

Любые глобальные переменные которые определены в отчете могут быть доступны из скрипта. Имя переменной должно быть объявлено как: `${моя_переменная}` без пробелов между фигурными скобками и в имени переменной. Рекомендуется 2 варианта именования переменных из нескольких слов как «my super duper variable». Первый путь — это заменять пробелы символами подчеркивания «\_». Второй путь — удалить пробелы и использовать заглавную букву для каждого следующего слова в переменной. Внутренне скриптовый движок использует специальные имена для хранения глобальных объектов. Поэтому использование "\_" в начале переменной крайне не рекомендуется.

Как только переменная объявлена, она будет автоматически добавлена в список глобальных переменных отчета. Давайте проверим. Откройте новый пустой отчет, перейдите во вкладку «Script» и введите `${test}`. Теперь переключитесь на вкладку «Reports». Вы увидите как переменная появилась в списке переменных и вы можете тестировать ваш отчет присвоением переменной какого-нибудь значения.





## Переменные рендерера

Движок рендерера имеет свои собственные переменные и полный список переменных зависит только от рендерера.

### Вот некоторые из переменных рендерера:

- **LINE** — номер текущей строки набора данных начиная с 1
- **PAGE** — номер текущей страницы начиная с 1
- **PAGES** — всего страниц (требует двойного прохода для отчета)
- **PASSES** — номер прохода отчета
- **TPAGE** - текущий номер страницы шаблона: номер страницы в Дизайнере начиная от 1
- **TPAGES** - всего страниц в шаблоне : количество страниц в Дизайнере
- **DATE** — дата когда генерация отчета была запущена (только профессиональная версия). Вместо этого может быть использовано `QDate.currentDate()`.
- **TIME** — время когда генерация отчета была запущена (только профессиональная версия). Вместо этого может быть использовано `QTime.currentTime()`.

## Сигналы в скрипте

Когда вы пишете скрипт это подразумевает что вы пишете его главную функцию которая будет обработана сразу после старта генератора. В этой функции пользователь может создавать некоторые переменные, инициализировать их или делать любые другие подготовительные операции. Вы все же можете желать большего контроля над процессом генерации. Для того чтобы это было возможным практически каждый объект отчета имеет сигналы и вы можете присоединять к ним ваш собственный обработчик сигналов — слот. Например, вы можете присоединить ваш собственный фильтр к контейнеру Detail и скрывать некоторые банды, пропуская другие. Давайте рассмотрим некоторые сигналы и позднее мы рассмотрим как их использовать.

### Общие сигналы элементов и контейнеров

Сигнал	Описание
printInit	Испускается когда объект готов к печати
printReset	Испускается когда объект очищается после печати
printBefore	Испускается перед печатью объекта. Все изменения свойств относятся к оригинальному объекту шаблона
printDataBefore	Испускается когда начальные данные объекта подготовлены. Все изменения свойств относятся к печатаемой копии объекта шаблона не затрагивая оригинальный объект шаблона
printDataAfter	Испускается после того как данные объекта обработаны перед непосредственно печатью
printAfter	Испускается после того как объект напечатан

Каждый объект может иметь свои собственные сигналы кроме описанных.

### Сигналы рендера:

Сигнал	Описание
reportStart	Испускается когда генератор запустился
bandBefore(CuteReport::BandInterface *)	Испускается перед рендерингом контейнера
bandAfter(CuteReport::BandInterface *)	Испускается после рендерингом контейнера
bandGeometryAfter(CuteReport::BandInterface *)	Испускается после изменения геометрии контейнера
itemBefore(CuteReport::BaseItemInterface *)	Испускается перед рендерингом контейнера или элемента
itemAfter(CuteReport::BaseItemInterface *)	Испускается после рендеринга контейнера или элемента
itemGeometryAfter(CuteReport::BaseItemInterface *)	Испускается после изменения геометрии контейнера или элемента
datasetBefore(CuteReport::DatasetInterface *)	Испускается перед обработкой набора данных

Сигнал	Описание
datasetAfter(CuteReport::DatasetInterface *)	Испускается после обработки набора данных
datasetIteration(CuteReport::DatasetInterface *)	Испускается перед каждой итерацией набора данных
pageBefore(CuteReport::PageInterface *)	Испускается перед началом обработки страницы шаблона
pageAfter(CuteReport::PageInterface *)	Испускается после обработки страницы шаблона
formBefore(CuteReport::FormInterface *)	Испускается перед показом формы
formAfter(CuteReport::FormInterface *)	Испускается после закрытия формы
reportDone	Испускается когда генератор закончил работу

# Использование в программах

В этой главе вы увидите как использовать CuteReport в ваших программах.

# Настройка проекта

Существует 2 способа использования CuteReport в вашей программе: как самостоятельный фреймворк или как встроенную библиотеку. Рассмотрим эти два способа.

## Встроенная библиотека

Нужно проделать следующие шаги для использования CuteReport как встроенной библиотеки:

- добавить все необходимые данные в ваш файл проекта(.pro);
- добавить заголовки CuteReport в ваш cpp файл;
- создать и инициализировать ядро CuteReport;

Добавьте следующие строки в ваш .pro файл:

```
INCLUDEPATH += path_to_CuteReport_headers
DEPENDPATH += $$INCLUDEPATH

LIBS += -Lpath_to_cutereport_shared_files -lCuteReport -lCuteReportWidgets
```

Добавьте следующие заголовки в ваш код:

```
#include "reportcore.h"
#include "reportinterface.h"
```

## Самостоятельный фреймворк

Для использования CuteReport как самостоятельный фреймворк вы можете просто установить CuteReport используя установщик. В этом есть несколько преимуществ:

- если у вас установлено несколько программ использующих CuteReport, вам не нужно сделать за обновлением CuteReport для каждой программы. Обновите CuteReport и все программы его использующие получат новую версию.
- Вы можете использовать официальные репозитории CuteReport и поддерживать его в актуальном состоянии автоматически в дистрибутивах Линукс.

Для подключения CuteReport в ваш проект добавьте в ваш pro файл нижеследующее:

```
!include( path_to_cutereport_include_directory/CuteReport.pri ) {
    error( Cannot find the CuteReport.pri file! )
}
```

Путь к CuteReport.pri зависит от вашего дистрибутива.

Ниже указаны стандартные пути для большинства инсталляций:

Linux: /usr/include/cutereport/CuteReport.pri

Windows: c:/Program Files/CuteReport/dev/include/CuteReport.pri

Второй шаг — это добавление заголовочного файла в ваш C++ файл:

```
#include <CuteReport>
```

## Простой пример

Далее создайте объект `CuteReport::ReportCore` и инициализируйте его:

```
CuteReport::ReportCore * reportCore = new CuteReport::ReportCore();
```

Ниже перечислены параметры которые вы можете передать в конструктор:

- *parent*: устанавливает родительский объект для объекта `CuteReport`. Если он установлен, то вам не стоит беспокоиться о корректном удалении объекта `CuteReport`;
- *settings*: указатель на объект `QSettings`. Вы можете использовать этот объект из вашей программы чтобы дать возможность `CuteReport` сохранять свои настройки и состояния в этом объекте настроек используя группу `[CuteReport]`. Настройки в этом файле могут содержать некоторую информацию для инициализации `CuteReport`. Вместо написания большого объема кода для добавления объектов таких как хранилища, рендереры, принтеры в ядро отчета, вы можете просто использовать специальные параметры в файле конфигурации. Мы рассмотрим этот аспект позднее. Если указатель на `QSettings` не определен, то `CuteReport` создаст свой `ini` файл.
- *interactive*: используется для указания являются ли объекты отчетов статическими или могут изменяться. Если вы разрабатываете некую программу, которая не меняет файлы шаблонов и объекты отчетов, вы можете высвободить некоторые ресурсы используя «false». Это значение по умолчанию;
- *initLogSystem*: определяет нужно ли включать ведение лога для `CuteReport`. Параметр выставлен в «True» по умолчанию. Следует упомянуть что направление лога и уровень могут быть установлены раздельно.

Дополнительно вы можете соединить некоторые сигналы для определения факта рендеринга отчета или его экспорта:

```
connect(reportCore, SIGNAL(exportDone(QString,bool)), this, SLOT(slotExportDone(QString,bool)));  
connect(reportCore, SIGNAL(printingDone(QString,bool)), this, SLOT(slotPrintDone(QString,bool)));
```

Теперь когда у нас есть объект ядра отчета, мы можем загрузить шаблон отчета:

```
CuteReport::ReportInterface * report = reportCore->loadReport("file:/path/myreport.qtrp");
```

В большинстве случаев вы можете пожелать видеть предпросмотр отчета, поэтому создадим виджет предпросмотра:

```
CuteReport::ReportPreview * preview = new CuteReport::ReportPreview(reportCore);  
preview->connectReport(report);
```

Передавая указатель на ядро отчета в виджет предпросмотра вы указываете какое ядро представляет предпросмотр.

Теперь мы можем запустить построение отчета. Для этого есть несколько способов. Первый — это нажать кнопку "Run" в виджете предпросмотра. Второй путь — это вызвать метод *render(report)* в CuteReport ядре.

```
reportCore->render(report);
```

Третий путь — это вызвать метод *run()* виджета предпросмотра.

```
preview->run();
```

Выбирайте любой путь.

## Пример пользовательской программы

Теперь давайте немного покодим. Чтобы добавить библиотеку CuteReport в вашу программу вы можете написать что-то вроде:

```
#include "reportcore.h"

/* создать объект ядра CuteReport */
CuteReport::ReportCore * reportCore = new CuteReport::ReportCore(0,0, false);

/* создать виджет предпросмотра */
CuteReport::ReportPreview * preview = new CuteReport::ReportPreview(parentWidget);

/* соединить ядро отчета с вашим предпросмотром */
preview->setReportCore(reportCore);

/* загрузить шаблон отчета из файла и создание объекта отчета */
CuteReport::ReportInterface * reportObject = reportCore->loadReport("git:report.qtrp");

/* соединить созданный объект отчета с предпросмотром */
preview->connectReport(reportObject);

/* показать виджет предпросмотра */
preview->show();

/* запустить построение отчета */
preview->run();
```

Обычно вам нужен всего один объект ReportCore в вашей программе. Любое количество виджетов предпросмотра может быть подключено к ядру.



**Наборы данных**

## Набор данных Модель

Набор данных Модель предназначен для вывода данных модели (наследник QAbstractItemModel) из приложения в отчет.

Для вывода данных модели необходимо проделать несколько шагов:

- В вашем отчете создать ModelDataset
- В поле "Имя модели" указать любое имя для своей модели. Если в отчете несколько таких наборов данных, то имена должны отличаться.
- Для тестирования отчета можно заполнить тестовую модель данными. В ней можно создать любое количество колонок и строк.

Отчет готов, теперь для его просмотра, необходимо в вашем приложении передать ссылку(в виде quint64) на существующую модель в параметр отчета.

Для примера:

```
CuteReport::ReportCore * cuteReport = new CuteReport::ReportCore();

// Загружаем отчет
QString err;
CuteReport::ReportInterface * reportObject = cuteReport->loadReport("file:test.qtrp", &err);

// Если не удалось загрузить, то выходим с ошибкой
if (!reportObject) {
    QMessageBox::critical(this, "loadReport", err);
    return;
}

// Создание тестовой модели
QStringList list;
list << "11111" << "2222" << "333" << "44" << "5";

model = new QStringListModel();
model->setStringList(list);

// ВАЖНО!!! Ссылка на модель в отчет передается в переменную отчета как quint64
// Имя переменной указать то, какое имя указали в ModelDataset отчета
reportObject->setVariableValue("model1", quint64(model));

// Создаем окно просмотра отчета
CuteReport::ReportPreview * preview = new CuteReport::ReportPreview(cuteReport);
if (reportObject) {
    // Указываем ядро и указываем загруженный отчет
    preview->setReportCore(cuteReport);
    preview->connectReport(reportObject);
    // Обработываем отчет
    preview->run();

    // Показываем окно просмотра
    preview->show();
}
```

При генерации отчета модель с данными клонируется, т.к. `QAbstractItemModel` не потокобезопасная.